Goethe-Universität Frankfurt am Main
Institut für Informatik
Theoretische Informatik

Diskrete Modellierung

Eine Einführung in grundlegende Begriffe und Methoden der Theoretischen Informatik

Skript zur Vorlesung
(Fassung vom 5. Februar 2020)[1]

Prof. Dr. Georg Schnitger

Große Teile dieses Skripts stimmen eins-zu-eins überein mit dem gleichnamigen Skript von Prof. Dr. Nicole Schweikardt zu der im Wintersemester 12/13 gehaltenen Vorlesung „Diskrete Modellierung“ an der Goethe-Universität Frankfurt.

[1] Vielen Dank an Mario Holldack und Hannes Seiwert für viele inhaltliche Anregungen und an Jens Donart für die Erstellung von Grafiken

Inhaltsverzeichnis

1. Einführung ins Thema „Diskrete Modellierung“

1.1. Diskrete Modellierung: Wir holen die „Problemstellung“ in den Rechner

In der Informatik wird das Modellieren mittels diskreter Strukturen als typische Arbeitsmethode in vielen Bereichen angewandt. Es dient der präzisen Beschreibung von Problemen durch spezielle Modelle und ist damit Voraussetzung für die systematische Lösung eines Problems. In den verschiedenen Gebieten der Informatik werden unterschiedliche, jeweils an die Art der Probleme und Aufgaben angepasste, diskrete Modellierungsmethoden verwendet. Ziel ist jeweils, nur die zur Lösung des Problems *relevanten* Aspekte präzise zu beschreiben.

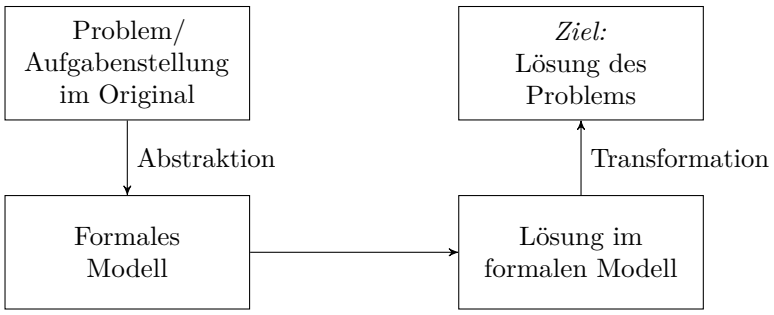


Abbildung 1.1.: Generelle Vorgehensweise in der Informatik

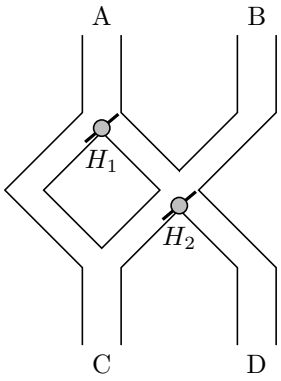
In der Veranstaltung „Diskrete Modellierung“ werden wir zuerst lernen, wie man denn korrekt argumentiert und beschreibt: Wir führen die Aussagenlogik (Kapitel 3) ein, und wenden in Kapitel 4 Beweismethoden wie direkte Beweise, Beweise durch Kontraposition oder durch Widerspruch an. Die Beweismethode der vollständigen Induktion wird sich als sehr wichtig herausstellen: Diese Methode zu verstehen ist gleichbedeutend mit dem Verständnis der rekursiven Programmierung. Mit der Sprache der Mengenlehre, die Thema des nächsten Kapitels ist, können wir Ungenauigkeiten der Umgangssprache umgehen.

Die Aussagenlogik dient aber nicht nur einer sorgfältigen Argumentation, sondern wird wie auch die Prädikatenlogik (Kapitel 9) als „Modellierungswerkzeug“ eingesetzt. Mit Graphen und Bäume (Kapitel 5), Markov-Ketten (Kapitel 6), bzw. mit formalen Sprachen (Transitionsystemen / endlichen Automaten in Kapitel 7 sowie kontextfreien Grammatiken in Kapitel 8) werden wir weitere mächtige Werkzeuge für die Modellierung diskreter Systeme kennenlernen.

Hier ist ein erstes Beispiel einer erfolgreichen Modellierung.

Beispiel 1.1 (Problem „Murmeln“).
Die nachfolgende Abbildung zeigt ein Spiel, in dem Murmeln bei A oder B in die Spielbahn fallen

gelassen werden.



Je nach Stellung der Hebel H_1 und H_2 rollen die Murmeln in der Spielbahn nach links oder rechts. Sobald eine Murmel auf einen dieser Hebel trifft, wird der Hebel nach dem Passieren der Murmel umgestellt, so dass die nächste Murmel in die andere Richtung rollt. Zu Beginn ist jeder der beiden Hebel so eingestellt, dass die nächste Murmel, die auf den Hebel trifft, nach links rollt. Wenn beispielsweise nacheinander drei Murmeln fallen gelassen werden, wobei die erste und dritte Murmel bei A und die zweite Murmel bei B fallen gelassen wird, dann kommen die ersten beiden Murmeln an der Öffnung C und die letzte Murmel an der Öffnung D heraus.

Frage: Aus welcher Öffnung fällt die letzte Murmel, wenn sieben Murmeln fallen gelassen werden, wobei die erste, zweite, vierte und letzte Murmel bei A und alle anderen Murmeln bei B fallen gelassen werden?

Lösungsansätze:

1. Knobeln, um eine Lösung per „Geistesblitz“ zu erhalten
2. Systematisches Vorgehen unter Verwendung von Informatik-Kalkülen

Hier wird der 2. Ansatz verfolgt.

Erste Analyse des Problems:

- *relevante Objekte:*
  Spielbahn, Eingänge A und B, Ausgänge C und D, Hebel H_1 und H_2, Murmeln
- *Tätigkeit:*
  Einwerfen von Murmeln an Eingängen A und/oder B
- *Start:*
  Hebel H_1 und H_2 zeigen nach links
- *Ziel:*
  Herausfinden, aus welchem Ausgang die letzte Murmel rollt, wenn nacheinander Murmeln an folgenden Eingängen eingeworfen werden: A, A, B, A, B, B, A
- *Eigenschaften/Beziehungen:*
  - Hebelpositionen: H_1 zeigt entweder nach links oder nach rechts, H_2 zeigt entweder nach links oder nach rechts.

– Für jeden der beiden Hebel H_1 bzw. H_2 gilt: Wenn er nach links (bzw. rechts) zeigt so rollt die nächste an ihm vorbeirollende Murmel nach links (bzw. nach rechts) weiter.

– Jeder der beiden Hebel H_1 bzw. H_2 ändert bei jedem Kontakt mit einer Murmel seine Richtung.

– Eine bei A eingeworfene Murmel rollt zu Hebel H_1.
Eine bei B eingeworfene Murmel rollt direkt zu Hebel H_2, ohne Hebel H_1 zu passieren.

– Zeigt H_1 nach links, so rollt eine bei A eingeworfene Murmel direkt zu Ausgang C.
Zeigt H_1 nach rechts, so rollt eine bei A eingeworfene Murmel zu Hebel H_2.

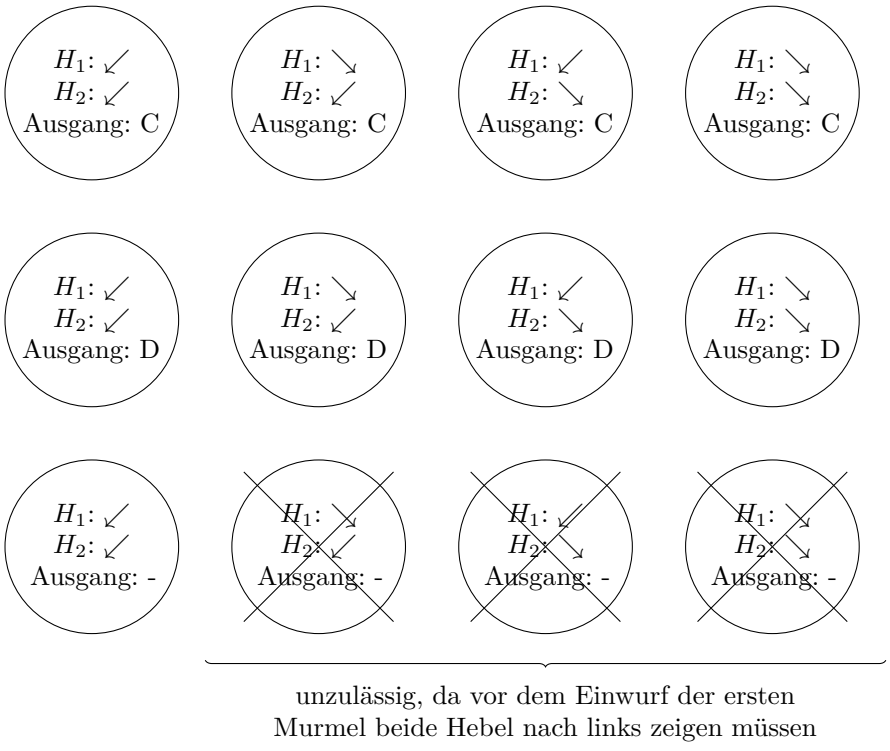
– Zeigt H_2 nach links, so rollt eine diesen Hebel passierende Murmel zu Ausgang C.
Zeigt H_2 nach rechts, so rollt eine diesen Hebel passierende Murmel zu Ausgang D.

Abstraktionen:

1. Nutze Abkürzungen:

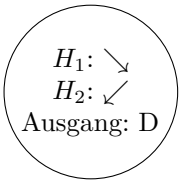
$H_1 : \swarrow$	$\widehat{=}$	Hebel H_1 zeigt nach links
$H_1 : \searrow$	$\widehat{=}$	Hebel H_1 zeigt nach rechts
$H_2 : \swarrow$	$\widehat{=}$	Hebel H_2 zeigt nach links
$H_2 : \searrow$	$\widehat{=}$	Hebel H_2 zeigt nach rechts
Ausgang: C	$\widehat{=}$	die zuvor fallen gelassene Murmel ist an Ausgang C herausgerollt
Ausgang: D	$\widehat{=}$	die zuvor fallen gelassene Murmel ist an Ausgang D herausgerollt
Ausgang: –	$\widehat{=}$	es wurde noch keine Murmel eingeworfen

2. Betrachte die möglichen „Zustände“, die auftreten dürfen:



unzulässig, da vor dem Einwurf der ersten Murmel beide Hebel nach links zeigen müssen

3. Formale Modellierung der „Zustände“: Repräsentiere den „Zustand“



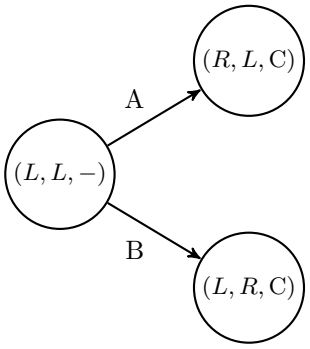
   durch das Tupel (R, L, D).

   Allgemein wird ein Zustand durch ein Tupel (x, y, z) repräsentiert mit $x \in \{L, R\}$, $y \in \{L, R\}$ und $z \in \{\mathrm{C}, \mathrm{D}, -\}$, für das folgende Bedingung erfüllt ist: falls $z = -$, so ist $x = y = L$.

Übergänge von einem Zustand in einen anderen Zustand:
Vom Zustand $(L, L, -)$ aus kann man durch Einwerfen einer einzelnen Murmel in folgende Zustände gelangen:

- (R, L, C), indem die Murmel bei A eingeworfen wird,
- (L, R, C), indem die Murmel bei B eingeworfen wird.

Graphische Darstellung:



Insgesamt ergibt sich das in der Abbildung auf der nächsten Seite dargestellte Zustandsdiagramm.

Lösung des Problems „Murmeln“:
An diesem Bild lässt sich unser ursprüngliches Problem „Murmeln“ (Frage: Aus welchem Ausgang rollt die letzte Murmel, wenn nacheinander Murmeln an den Eingängen A, A, B, A, B, B, A eingeworfen werden?) leicht lösen, indem man einfach einen Weg vom „Startzustand“ sucht, bei dem die Pfeile nacheinander mit A, A, B, A, B, B, A beschriftet sind. Im Zustandsdiagramm gibt es genau einen solchen Weg; er endet mit dem Zustand (L, R, C). Die Antwort auf die ursprünglich gestellte Frage lautet also: Wenn nacheinander Murmeln an den Eingängen A, A, B, A, B, B, A eingeworfen werden, so rollt die letzte Murmel durch Ausgang C.

Man beachte, dass man anhand des Zustandsdiagramms auch die folgende Frage beantworten kann:

> Ist es möglich, vom Startzustand aus durch geschicktes Einwerfen von Murmeln zu erreichen, dass die letzte Murmel aus Ausgang D herausrollt und danach beide Hebel nach rechts zeigen?

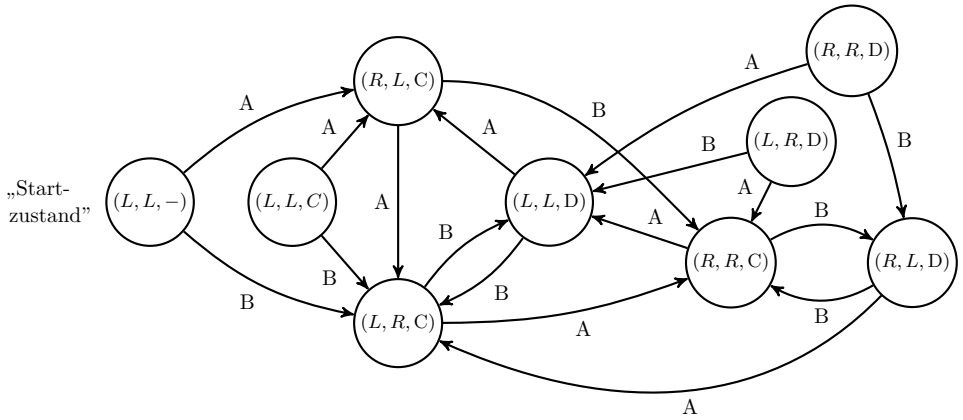


Abbildung 1.2.: Übergänge zwischen den Zuständen beim Problem „Murmeln“

Um diese Frage zu beantworten muss man einfach nachprüfen, ob es im Zustandsdiagramm einen Weg vom Startzustand zum Zustand (R, R, D) gibt. Man sieht leicht, dass es keinen solchen Weg gibt. Folglich lautet die korrekte Antwort auf obige Frage „nein“.

□ Ende Beispiel 1.1

Anmerkung:
Wir haben hier den Kalkül der **Transitionssysteme** (auch bekannt als **endliche Automaten** bzw. **Zustandsübergangsdiagramme** oder **Statecharts**) benutzt. Dieser Kalkül eignet sich besonders gut, wenn Abläufe in Systemen mit Übergängen zwischen verschiedenen Zuständen beschrieben werden sollen. Mehr dazu findet sich in Kapitel 7.

Wir betrachten ein weiteres Beispiel, das auf ähnliche Art gelöst werden kann.

Beispiel 1.2 (Problem „Flussüberquerung“).
Ein Mann steht mit einem Wolf, einer Ziege und einem Kohlkopf am linken Ufer eines Flusses, den er mit allen drei überqueren will. Er hat ein Boot, das gerade groß genug ist, ihn und ein weiteres Objekt zu transportieren, so dass er immer nur eines der drei mit sich hinübernehmen kann. Falls der Mann allerdings den Wolf mit der Ziege oder die Ziege mit dem Kohlkopf unbewacht an einem Ufer zurück lässt, wird die Ziege bzw. der Kohlkopf gefressen. *Frage:* Ist es möglich, den Fluss zu überqueren, ohne dass die Ziege oder der Kohlkopf gefressen wird?

Wie kommen wir zwangsläufig zu einer Lösung?

1.2. Zusammenfassung und Ausblick

Der Begriff „Modell“ wird in verschiedenen Zusammenhängen mit unterschiedlichen Bedeutungen verwendet. Ein Modell kann ein Abbild eines vorhandenen oder noch zu schaffenden Originals sein (z.B. ein Modellflugzeug oder ein Gebäude in kleinem Maßstab), es kann aber auch zur Repräsentation einiger Aspekte der realen Welt dienen (z.B. verschiedene Atommodelle).
Modelle werden u.a. benutzt, um

- ein konkretes Problem zu lösen (z.B. Beispiel 1.1 „Murmelproblem“),
- bestimmte Aspekte eines komplexen Gebildes zu untersuchen, zu verstehen oder zu vermitteln (z.B. Geschäftsabläufe in einer Firma),
- die Kommunikation zwischen einem Auftraggeber und einem Hersteller des Originals zu vereinfachen (z.B. beim Bau eines Hauses oder bei der Software-Entwicklung),
- Anforderungen für die Herstellung des Originals zu fixieren (z.B. Spezifikation von und Anforderungen an Software),
- Operationen durchzuführen, die man am Original nicht durchführen kann (z.B. Computer-Simulation dessen, was bei einem Flugzeugabsturz über einem Kernkraftwerk passieren könnte),
- ein Modell zu validieren (engl. *Model Checking*), d.h. um nachzuweisen, dass die relevanten Eigenschaften des Originals korrekt und vollständig im Modell erfasst sind (z.B. zur Prüfung, ob ein Finanzplan alle Kosten erfasst, sie korrekt aufsummiert und die vorgegebene Kostengrenze eingehalten wird).

Modelle sind **absichtlich** nicht originalgetreu. Sie heben bestimmte Eigenschaften hervor und lassen andere weg. Der konkrete Verwendungszweck des Modells bestimmt, welche Eigenschaften modelliert werden und welcher Kalkül dafür besonders geeignet ist.
Ein Modell beschreibt stets nur einige bestimmte Aspekte des Originals, etwa

- die Struktur oder die Zusammensetzung des Originals (z.B. das Organigramm einer Firma).

  Dafür geeignete Kalküle sind z.B. Wertebereiche, Entity-Relationship-Modell, Bäume, Graphen.
- Eigenschaften von Teilen des Originals (z.B. Farbe und Wert einer Spielkarte).

  Dafür geeignete Kalküle sind z.B. Wertebereiche, Logik, Entity-Relationship-Modell.
- Beziehungen zwischen Teilen des Originals (z.B. „Wolf frisst Ziege, Ziege frisst Kohlkopf“ in Beispiel 1.2).

  Dafür geeignete Kalküle sind z.B. Graphen, Logik, Entity-Relationship-Modell.
- Verhalten des Originals unter Operationen (z.B. aufeinanderfolgende Zustände bei wiederholtem Einwurf von Murmeln in Beispiel 1.1).

  Dafür geeignete Kalküle sind z.B. Zustandsübergangsdiagramme, Petri-Netze, Graphen.

Beispiel 1.3.

(a) Unterschiedliche Modelle, die beim Bau eines Hauses verwendet werden:
- Gebäudemodell: zur Vermittlung eines optischen Eindrucks
- Grundriss: zur Einteilung der Räume und des Grundstückes
- Kostenplan: zur Finanzierung

(b) Frankfurter S- und U-Bahn Netzplan: siehe Abbildung 1.3

Ziel: Beschreibung, welche Haltestellen von welchen Linien angefahren werden und welche Umsteigemöglichkeiten es gibt

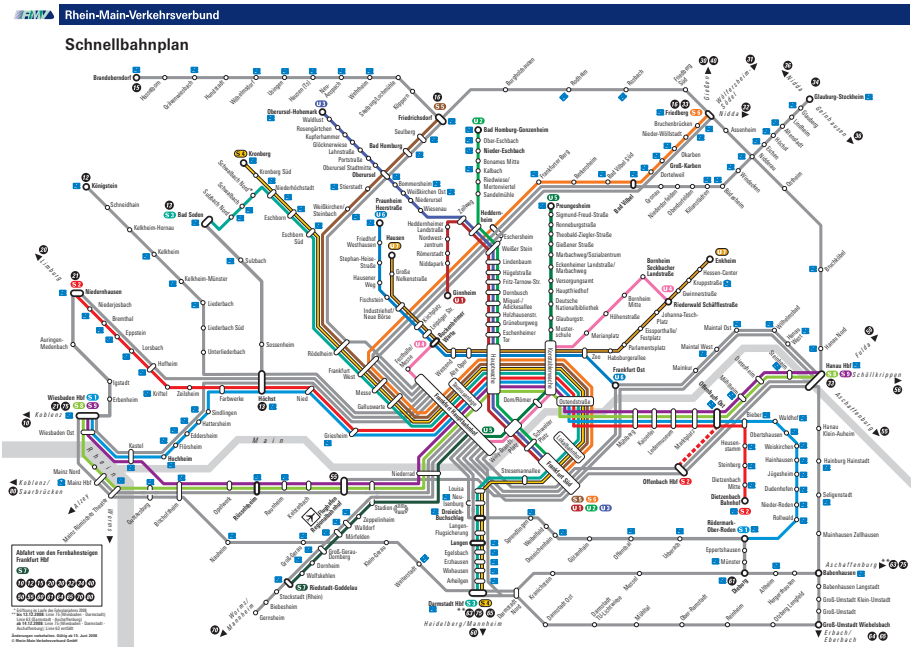


Abbildung 1.3.: Schnellbahnplan des Rhein-Main-Verkehrsverbundes (RMV)

Vernachlässigt: genauere topografische Informationen (Entfernung, genaue Lage, Straßenverläufe etc.), Abfahrtszeiten

(c) Fahrplan der U4 an der Haltestelle „Bockenheimer Warte“: siehe Abbildung 1.4

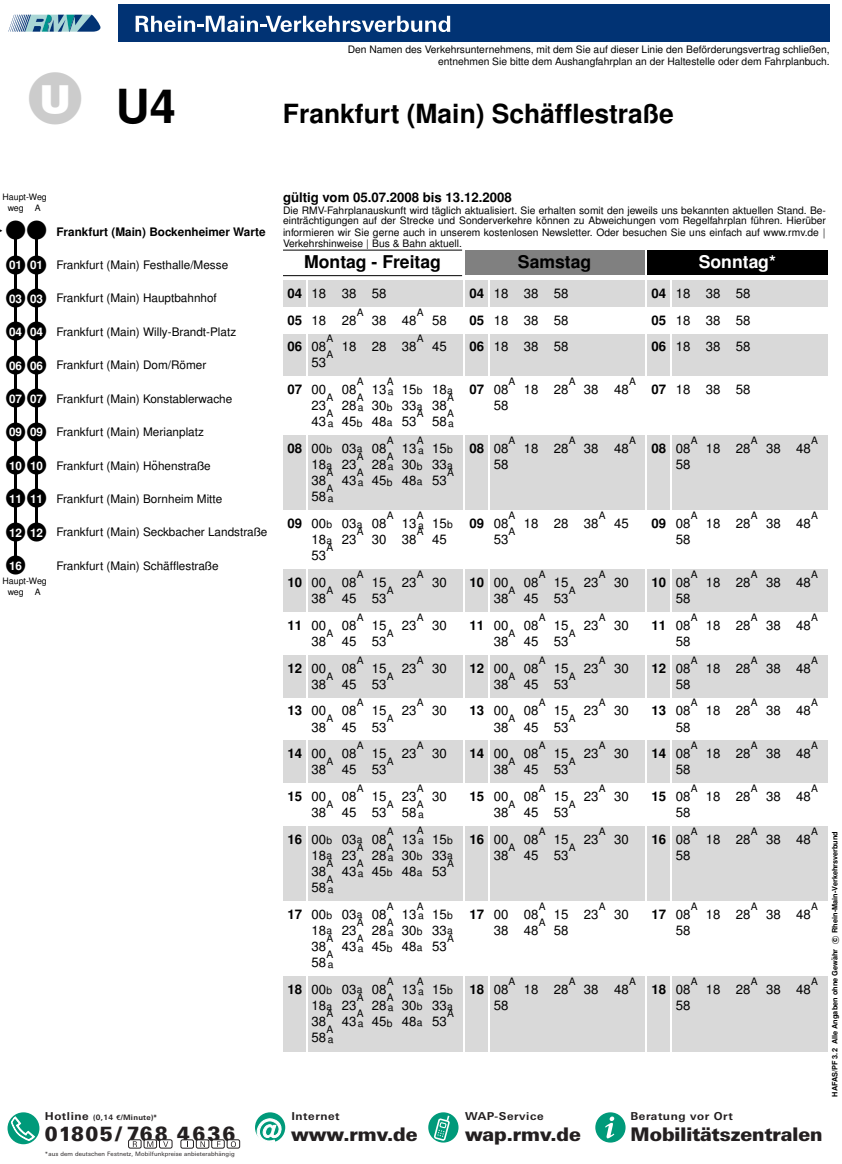
Ziel: Angabe der Abfahrtszeiten der U4 an der Haltestelle „Bockenheimer Warte“ sowie Informationen darüber, wie viele Minuten die U4 von dort bis zu anderen Haltestellen auf ihrer Strecke braucht.

□Ende Beispiel 1.3

1.3. Literaturhinweise zu Kapitel 1

Als vertiefende Lektüre sei Kapitel 1 in [14] empfohlen.

Quellennachweis: Teile dieses Kapitels orientieren sich an Kapitel 1 in [14]; insbesondere Beispiel 1.3 sowie das in Beispiel 1.2 betrachtete „Flussüberquerungsproblem“ sind im Wesentlichen aus [14] übernommen. Das „Flussüberquerungsproblem“ findet sich bereits in [12]; das „Murmelproblem“ aus Beispiel 1.1 ist eine vereinfachte Variante von Aufgabe 2.3 in [12].



Rhein-Main-Verkehrsverbund

Den Namen des Verkehrsunternehmens, mit dem Sie auf dieser Linie den Beförderungsvertrag schließen, entnehmen Sie bitte dem Aushangfahrplan an der Haltestelle oder dem Fahrplanbuch.

U4

Frankfurt (Main) Schäfflestraße

Haupt-Weg weg A

- Frankfurt (Main) Bockenheimer Warte
- Frankfurt (Main) Festhalle/Messe
- Frankfurt (Main) Hauptbahnhof
- Frankfurt (Main) Willy-Brandt-Platz
- Frankfurt (Main) Dom/Römer
- Frankfurt (Main) Konstablerwache
- Frankfurt (Main) Merianplatz
- Frankfurt (Main) Höhenstraße
- Frankfurt (Main) Bornheim Mitte
- Frankfurt (Main) Seckbacher Landstraße
- Frankfurt (Main) Schäfflestraße

Haupt-Weg weg A

gültig vom 05.07.2008 bis 13.12.2008

Die RMV-Fahrplanauskunft wird täglich aktualisiert. Sie erhalten somit den jeweils uns bekannten aktuellen Stand. Beeinträchtigungen auf der Strecke und Sonderverkehre können zu Abweichungen vom Regelfahrplan führen. Hierüber informieren wir Sie gerne auch in unserem kostenlosen Newsletter. Oder besuchen Sie uns einfach auf www.rmv.de | Verkehrshinweise | Bus & Bahn aktuell.

Montag - Freitag						Samstag						Sonntag*					
04	18	38	58			04	18	38	58			04	18	38	58		
05	18	28A	38	48A	58	05	18	38	58			05	18	38	58		
06	08A 53A	18	28	38A	45	06	18	38	58			06	18	38	58		
07	00A 23A 43Aa	08A 28Aa 45b	13Aa 30b 48a	15b 33Aa 53A	18a 38A 58Aa	07	08A 58	18	28A	38	48A	07	18	38	58		
08	00b 18a 38A 58Aa	03a 23A 43Aa	08A 28Aa 45b	13Aa 30b 48a	15b 33Aa 53A	08	08A 58	18	28A	38	48A	08	08A 58	18	28A	38	48A
09	00b 18a 53A	03a 23A	08A 30	13Aa 38A	15b 45	09	08A 53A	18	28	38A	45	09	08A 58	18	28A	38	48A
10	00A 38A	08A 45	15A 53A	23A	30	10	00A 38A	08A 45	15A 53A	23A	30	10	08A 58	18	28A	38	48A
11	00A 38A	08A 45	15A 53A	23A	30	11	00A 38A	08A 45	15A 53A	23A	30	11	08A 58	18	28A	38	48A
12	00A 38A	08A 45	15A 53A	23A	30	12	00A 38A	08A 45	15A 53A	23A	30	12	08A 58	18	28A	38	48A
13	00A 38A	08A 45	15A 53A	23A	30	13	00A 38A	08A 45	15A 53A	23A	30	13	08A 58	18	28A	38	48A
14	00A 38A	08A 45	15A 53A	23A	30	14	00A 38A	08A 45	15A 53A	23A	30	14	08A 58	18	28A	38	48A
15	00A 38A	08A 45	15A 53A	23A 58Aa	30	15	00A 38A	08A 45	15A 53A	23A	30	15	08A 58	18	28A	38	48A
16	00b 18a 38A 58Aa	03a 23A 43Aa	08A 28Aa 45b	13Aa 30b 48a	15b 33Aa 53A	16	00A 38A	08A 45	15A 53A	23A	30	16	08A 58	18	28A	38	48A
17	00b 18a 38A 58Aa	03a 23A 43Aa	08A 28Aa 45b	13Aa 30b 48a	15b 33Aa 53A	17	00 38	08A 48A	15 58	23A	30	17	08A 58	18	28A	38	48A
18	00b 18a 38A 58Aa	03a 23A 43Aa	08A 28Aa 45b	13Aa 30b 48a	15b 33Aa 53A	18	08A 58	18	28A	38	48A	18	08A 58	18	28A	38	48A

HAFAS/PF 3.2 Alle Angaben ohne Gewähr © Rhein-Main-Verkehrsverbund

Hotline (0,14 €/Minute)* 01805/768 4636 (0180 5 RMV INFO)
*aus dem deutschen Festnetz, Mobilfunkpreise anbieterabhängig

Internet www.rmv.de

WAP-Service wap.rmv.de

Beratung vor Ort Mobilitätszentralen

Abbildung 1.4.: Fahrplan der U4 an der Haltestelle „Bockenheimer Warte“

1.4. Übungsaufgaben zu Kapitel 1

Aufgabe 1.1. Gegeben seien drei Stapel mit Büchern. Der erste besteht aus vier Büchern, der zweite aus sechs Büchern und der dritte aus 14 Büchern. Die Stapel sollen nun ausgeglichen werden, so dass auf jedem Stapel acht Bücher liegen. Allerdings dürfen in jedem Schritt nur Bücher zwischen genau zwei Stapeln umgeschichtet werden. Zudem können auf jeden Stapel immer nur so viele Bücher gelegt werden, wie bereits darauf liegen.

(a) Lassen sich die Stapel wie gewünscht ausgleichen? Modellieren Sie zur Beantwortung dieser Frage das Problem analog zum Beispiel 1.2.

(b) Nehmen wir nun an, dass der ersten Stapel aus vier Büchern, der zweite aus sechs Büchern und der dritte aus acht Büchern besteht. Lassen sich die Stapel so ausgleichen, dass auf jedem Stapel sechs Bücher liegen?

Hinweis: Es brauchen nur diejenigen Zustände betrachtet zu werden, die man vom Startzustand aus durch geeignete Zustandsübergänge erreichen kann.

Aufgabe 1.2. In dieser Aufgabe betrachten wir eine Variante des unter dem Namen *Nim* bekannten Spiels, die wie folgt definiert ist: Es gibt zwei Spieler namens Alice und Bob. Zu Beginn des Spiels liegen fünf Hölzer auf dem Tisch. Die beiden Spieler sind abwechselnd am Zug. In jedem Zug kann der Spieler, der gerade an der Reihe ist, entscheiden, ob er ein Holz oder zwei Hölzer vom Tisch wegnimmt. Der Spieler, der das letzte Holz vom Tisch nimmt, verliert das Spiel. Zu Beginn ist Alice am Zug.

Modellieren Sie zur Beantwortung der folgenden Fragen das Spiel analog zum Beispiel 1.1 aus der Vorlesung durch ein Transitionssystem. Überlegen Sie sich zunächst, welche Zustände und Zustandsübergänge auftreten können.
Hinweis: Jeder Zustand des Transitionssystems sollte Informationen darüber enthalten, welcher Spieler am Zug ist und wie viele Hölzer noch auf dem Tisch liegen.

(a) Ist es eine gute Idee für Alice, im ersten Zug zwei Hölzer zu nehmen?

(b) Eine Gewinnstrategie für einen Spieler in diesem Spiel ist eine Vorschrift, welche ihm sagt, welchen Zug er als nächstes tätigen soll. Hält sich der Spieler an diese Vorschrift, so gewinnt er auf jeden Fall. Existiert in diesem Spiel eine Gewinnstrategie für Alice?

(c) Existiert eine Gewinnstrategie für Bob?

Aufgabe 1.3. In dieser Aufgabe betrachten wir das Spiel *NimHalbe*, das wie folgt definiert ist: Zwei Spieler, Alice und Bob, spielen gegeneinander. Zu Beginn des Spiels liegen 1336 Hölzer auf dem Tisch. Die beiden Spieler sind abwechselnd am Zug, Alice beginnt. In jedem Zug kann der Spieler, der gerade an der Reihe ist, entweder drei Hölzer vom Tisch entfernen oder, falls eine gerade Anzahl an Hölzern auf dem Tisch liegt, den Haufen halbieren, also die Hälfte der Hölzer vom Stapel nehmen.

So kann Alice im ersten Zug drei Hölzer vom Tisch nehmen (es verbleiben 1333) oder den Stapel halbieren (es verbleiben 668). Hiernach ist Bob am Zug und kann bei 668 Hölzern wieder zwischen beiden Optionen wählen; bei 1333 Hölzern ist er gezwungen, drei davon wegzunehmen. Es gewinnt der Spieler, der eine Anzahl von Hölzern auf dem Tisch hinterlässt, die eine Primzahl ist. (Achtung: 1 ist keine Primzahl.)

Modellieren Sie zur Beantwortung der folgenden Fragen das Spiel analog zum Beispiel 1.1 aus der Vorlesung durch ein Transitionssystem.

(a) Ist es eine gute Idee für Alice, im ersten Zug gleich den Stapel zu halbieren ?

(b) Eine Gewinnstrategie für einen Spieler in diesem Spiel ist eine Vorschrift, die ihm sagt, welchen Zug er als nächstes tätigen soll. Hält sich der Spieler an diese Vorschrift, so gewinnt er auf jeden Fall. Existiert in diesem Spiel eine Gewinnstrategie für Alice?

(c) Existiert eine Gewinnstrategie für Bob?

Aufgabe 1.4. Die Piraten Jack Sparrow und Barbossa sind auf einer einsamen Insel gestrandet. Von ihrem sinkenden Schiff konnten sie ein Fass mit Rum (8 Liter) sowie zwei leere Fässer der Größen 3 Liter und 5 Liter retten. Da keiner dem anderen richtig traut, wollen sie den Rum aufteilen, so dass sich jeder mit seinem Anteil an ein Ende der Insel zurückziehen kann.

Ziel ist es, in zwei Fässern jeweils genau die Menge von vier Litern Rum zu haben. Da sich an den Fässern keine Markierungen befinden, können die beiden Piraten ihr Ziel nur erreichen, indem sie schrittweise eines der Fässer in das andere kippen bis eines der Fässer voll oder leer ist. Da Piraten faul sind, werden sie außerdem darauf achten, niemals in den Startzustand mit dem vollen 8-Liter Fass zurückkehren. Zusätzlich dazu vermeiden sie den Zustand, in dem sowohl das 3-Liter Fass als auch das 5-Liter Fass komplett voll sind, da dieser genauso ungünstig ist.

Modellieren Sie zur Beantwortung der folgenden Fragen das Problem durch ein Transitionssystem analog zum „Murmelbeispiel“ aus der Vorlesung.

(a) Können Jack und Barbossa ihr Ziel erreichen?

(b) Für Jack ist es unerträglich, nicht voran zu kommen. Er möchte deshalb niemals eine gerade getätigte Aktion direkt wieder rückgängig machen. Nehmen Sie nun an, dass Jack keine Aktion direkt wieder rückgängig macht und ansonsten (unter Berücksichtigung der Regeln) wahllos vorgeht. Wird er dann zwangsläufig irgendwann in den Zustand kommen, in dem er sein Ziel erreicht hat?

(c) Nehmen wir nun an, Jack wäre nicht mit Barbossa, sondern mit zwei namenlosen Handlangern auf der Insel gestrandet. Da acht Liter Rum nicht gut durch drei zu teilen sind, beschließt Jack, sich selbst sechs Liter und den anderen beiden großzügigerweise jeweils einen Liter zuzuteilen. Ist dies unter Berücksichtigung der Regeln möglich?

Teil I.

Grundlagen

2. Der Kalkül der Mengen

2.1. Warum „sprechen“ wir Mathematik?

In der diskreten Modellierung benutzen wir, soweit möglich, eine mathematische Sprache. Warum? In der Informatik müssen wichtige Systeme fehlerfrei funktionieren: Wir müssen ihre Korrektheit beweisen! Die Umgangssprache verführt aber häufig dazu, ungenau oder sogar widersprüchlich zu argumentieren oder zu beschreiben.

2.1.1. Das ist aber paradox!

(Homer Simpson im Dialog mit Lisa)
Lisa: „Hier gibt es keine Tiger...und hier liegt ein Stein... nach deiner Logik müsste dieser Stein Tiger vertreiben!“
Homer: „Lisa, ich möchte dir den Stein abkaufen.“

Beispiel 2.1. (Das Henne-Ei-Problem). Wer war zuerst da, die Henne oder das Ei? Dass es sich hier um kein Paradoxon handelt, zeigt ein entsprechendes Problem in der Informatik. Ein Compiler ist ein Programm, das den Programmcode eines anderen Programms in ein ausführbares Programm verwandelt. Wie kompiliert man aber den Programmcode für einen Compiler, wenn noch kein Compiler zur Verfügung steht?

Mit Hilfe des Bootstrapping[1] baut man eine Hierarchie immer mächtigerer Compiler C_i, wobei C_0 in direkt ausführbarem Code geschrieben ist und C_{i+1} mit C_i kompiliert wird.

Beispiel 2.2. (Wer flunkert denn da?) Epimenides der Kreter sagt, dass alle Kreter lügen. Ist diese Aussage richtig?

Hier handelt es sich um die von dem britischen Philosophen und Mathematiker Bertrand Russel vorgeschlagene Kurzform des folgenden Verses[2]: „Es hat einer von ihnen gesagt, ihr eigener Prophet: Die Kreter sind immer Lügner, böse Tiere und faule Bäuche.“

Wenn die Aussage richtig ist, dann lügen alle Kreter immer und damit lügt auch Epimenides – im Widerspruch zur Fallannahme. Wenn die Aussage hingegen falsch ist, dann gibt es mindestens einen Kreter, der nicht immer lügt oder kein böses Tier ist oder kein fauler Bauch ist: Diese Fallannahme führt auf keinen Widerspruch.

In den beiden nächsten Beispielen lernen wir Aussagen kennen, die sich auf sich selbst beziehen. Aussagen von diesem Typ können nicht nur wahr oder falsch sein, sondern auch widersprüchlich.

Beispiel 2.3. (**Selbstreferenzen I**)

[1] Sich am Stiefelriemen über den Zaun ziehen
[2] Aus einem Brief des Apostels Paulus an Titus

(a) Pinocchios Nase wächst bekanntlich genau dann, wenn er lügt. Was passiert, wenn Pinocchio sagt, dass seine Nase wächst?

- Sagt Pinocchio die Wahrheit, dann wächst seine Nase gerade und er lügt somit – im Widerspruch zur Annahme.
- Lügt Pinocchio, dann wächst seine Nase nicht. Wenn seine Nase aber nicht wächst, dann sagt er die Wahrheit – im Widerspruch zur Annahme.

(b) Ich lüge gerade.

- Angenommen, die Aussage „Ich lüge gerade“ ist wahr. Dann lüge ich gerade. Wenn ich aber gerade lüge, dann ist die gerade von mir gemachte Aussage gelogen und ich sage also tatsächlich die Wahrheit – im Widerspruch zur Annahme.
- Angenommen, die Aussage „Ich lüge gerade“ ist falsch, dann sage ich gerade die Wahrheit. Also ist die gemachte Aussage „Ich lüge gerade“ wahr. Also lüge ich doch – im Widerspruch zur Annahme.

(c) „Dieser Satz ist unwahr“

- Angenommen, die Aussage „Dieser Satz ist unwahr“ ist unwahr. Dann ist dieser Satz wahr – im Widerspruch zur Annahme.
- Angenommen, die Aussage „Dieser Satz ist unwahr“ ist wahr. Dann ist die dieser Satz unwahr – im Widerspruch zur Annahme.

Beispiel 2.4. (**Selbstreferenzen II**) Eine Implikation, also eine Aussage der Form

Wenn Eigenschaft E_1, dann Eigenschaft E_2

ist genau dann falsch, wenn die Voraussetzung, also die Eigenschaft E_1, wahr ist und der Schluss, also die Eigenschaft E_2 falsch ist.

Wir „beweisen“, dass der Mond aus grünem Käse besteht. Dazu beachten wir zuerst, dass eine Aussage entweder wahr oder falsch ist. Also ist auch der folgende Satz entweder wahr oder falsch:

Wenn dieser Satz wahr ist, dann besteht der Mond aus grünem Käse!

Angenommen, dieser Satz ist falsch. Der Satz ist eine Implikation und kann deshalb nur dann falsch sein, wenn die Voraussetzung wahr ist. Die Voraussetzung ist aber „dieser Satz ist wahr“ und widerspricht der Annahme.

Also ist dieser Satz wahr. Aber dieser Satz besteht aus einer Implikation, deren Voraussetzung wahr ist: Der Schluss muss wahr sein: Der Mond besteht also tatsächlich aus grünem Käse!

Beispiel 2.5. Jura

Euathlos wurde von Protagoras von Abdera (ca. 485–415 vor Chr.), einem berühmten Rhetoriker und Lehrer der Sophistik ausgebildet. Sie vereinbarten, dass Euathlos seine Ausbildung erst dann bezahlen muss, wenn er seinen ersten Gerichtsprozess als Anwalt gewinnt. Nun wählte Euathlos aber einen anderen Beruf und fühlte sich deshalb an die Vereinbarung nicht gebunden. Daraufhin drohte Protagoras ihm mit Klage und behauptete, dass Euathlos, unabhängig vom Ausgang des Prozesses auf jeden Fall zahlen müsse. Hat er recht?

Ja, aber nur wenn Euathlos so dumm ist, sich selbst zu verteidigen. Warum?

Beispiel 2.6. Achilles und die Schildkröte

Achilles und die Schildkröte laufen ein Wettrennen. Achilles gewährt der Schildkröte einen Vorsprung.

Zenon von Elea (490 bis 425 v.Chr.) argumentiert, dass Achilles die Schildkröte niemals einholen kann, wie groß (oder klein) der Vorsprung auch immer ist. Zenon gibt folgende Begründung: Zu dem Zeitpunkt, an dem Achilles den Startpunkt der Schildkröte erreicht, ist die Schildkröte schon ein Stück weiter. Etwas später erreicht Achilles diesen Punkt, aber die Schildkröte ist schon etwas weiter. Erreicht Achilles diesen Punkt, ist die Schildkröte natürlich längst weiter gekrochen. So kommt Achilles zwar immer näher an die Schildkröte heran, holt sie aber niemals ein.

Beispiel 2.7. Das Berry-Paradoxon[3]

Jede natürliche Zahl ist mit höchstens dreizehn Worten des Dudens definierbar?! Warum? Sei n die kleinste natürliche Zahl, die nicht mit höchstens dreizehn Worten definierbar ist. Aber dann ist n definierbar durch die dreizehn Worte „ist die kleinste natürliche Zahl, die nicht mit höchstens dreizehn Worten definierbar ist“.

Aber was bedeutet „definierbar“? Kann man tatsächlich jedem aus Worten des Dudens gebauten Satz zweifelsfrei entweder die leere Menge oder eine durch den Satz definierte Zahl zuweisen? Wenn das der Fall wäre, warum braucht man selbst für sorgfältig formulierte Gesetzestexte ein Bundesverfassungsgericht, bzw. einen Europäischen Gerichtshof, um Gesetze auszulegen und warum gibt es häufig Gerichtsbeschlüsse, die nicht einstimmig erfolgen?

Beispiel 2.8. Sandhaufen oder kein Sandhaufen?

Offensichtlich kommt es bei einem Haufen Sand auf ein Korn mehr oder weniger nicht an. Ein Korn ist sicherlich kein Haufen, auch zwei Körner nicht, geben wir noch eins hinzu, so ist es immer noch kein Haufen, usw. Da es Sandhaufen gibt, muss es ein bestimmtes Korn gegeben haben, durch dessen Hinzunahme der Haufen entstanden ist. Also kommt es doch auf ein Korn mehr oder weniger an!?

Beispiel 2.9. Korrektes Schließen I

„Wenn man zwei beliebige Sätze auswählt, dann ist stets der erste eine Folgerung des zweiten oder es ist stets der zweite eine Folgerung des ersten.“ Ist diese Aussage wahr oder falsch?

Und was ist zu halten von der Aussage „Wenn man zwei beliebige Sätze auswählt, dann ist stets der erste eine Folgerung des zweiten oder der zweite eine Folgerung des ersten.“

Und wieso unterscheiden sich die beiden Aussagen?

Beispiel 2.10. Korrektes Schließen II

Eine Lehrerin sagt zu ihrer Klasse: „In der nächsten Woche schreibt ihr einen völlig überraschenden Test über dieses Thema!“ Ein Schüler hält das für unmöglich. Er sagt: „Die Klasse hat dieses Fach montags, donnerstags und freitags. Wenn der Test am Freitag geschrieben wird, so ist er nicht überraschend, sondern bereits am Donnerstag nach der Stunde vorhersehbar. Findet der Test am Donnerstag statt? Nein, denn ich habe den Freitag bereits ausgeschlossen und der Montag ist dann bereits vorbei und kann ebenfalls ausgeschlossen werden. Der Test muss also am Montag sein und wäre dann aber nicht überraschend. Also gibt es keinen überraschenden Test!

Natürlich wurde die Klasse in der nächsten Woche von dem Test völlig überrascht!

[3] George Godfrey Berry (1867-1928) war ein Bibliothekar der Bodleian Library Oxfords

Beispiel 2.11. Korrektes Schließen III

Welche der folgenden Schlüsse sind richtig, welche sind falsch? Man beachte, dass nicht die Richtigkeit der Folgerung beurteilt werden soll, sondern nur die Richtigkeit des Schlusses.

(a) Hier ist das erste Beispiel:

1. Annahme: Es gibt Vögel, die fliegen können.
2. Annahme: Vögel sind Tiere.
3. Annahme: Es gibt keine fliegenden Tiere, die sich nur unter Wasser aufhalten.

Folgerung: Es gibt keine Vögel, die sich nur unter Wasser aufhalten.

(b) Jetzt das zweite Beispiel:

1. Annahme: Es gibt Menschen, die stumm sind.
2. Annahme: Menschen sind Lebewesen.
3. Annahme: Es gibt keine stummen Lebewesen, die sprechen können.

Folgerung: Es gibt keine Menschen, die sprechen können.

(c) Und das letzte Beispiel:

1. Annahme: Erdbeeren schmecken gut.
2. Annahme: Sahne schmeckt gut.

Folgerung: Erdbeeren mit Sahne schmecken gut.

Beispiel 2.12. Konvergenz unendlicher Reihen

Wir betrachten die unendliche Reihe

$$S = \sum_{i=0}^{\infty} a_i = (1-1) + (1-1) + (1-1) + (1-1) \cdots$$

mit $a_i = (-1)^i$. Da $a_{2i} + a_{2i+1} = 0$ gilt, wird die Null unendlich mal addiert und wir erhalten $S = 0$. Andererseits ist aber

$$S = 1 + \sum_{i=1}^{\infty} a_i = 1 + (-1+1) + (-1+1) + (-1+1) \cdots = 1,$$

also $S = 1$, denn $a_{2i-1} + a_{2i} = 0$ gilt. Gleichzeitig, und das ist beruhigend, gilt $2 \cdot S = 1$, also $S = 1/2$. Warum? Weil

$$\begin{aligned} 2S &= (2-2) + (2-2) + (2-2) + (2-2) + \cdots \\ &= (1-1) + (1-1) + (1-1) + (1-1) + \cdots \\ &= 1 + (-1+1) + (-1+1) + (-1+1) + \cdots = 1. \end{aligned}$$

Was ist hier des Rätsels Lösung? Die Zahl S ist der Grenzwert der divergenten Folge $(\sum_{i=0}^{n} a_i : n \in \mathbb{N})$. Diesen Grenzwert, und damit auch die Zahl S, gibt es gar nicht! Über Zahlen, die es nicht gibt, kann man natürlich herrliche Theorien bilden.

Beispiel 2.13. Das kann doch nicht wahr sein!

(a) Es kann gezeigt werden, dass es *raumfüllende Kurven* gibt, also Funktionen

$$K : [0, 1] \to [0, 1]^2,$$

die jeden Punkt im Einheitsquadrat durchlaufen. Darüberhinaus kann man sogar fordern, dass K eine stetige Funktion ist!

(b) Kann man eine Kugel im Dreidimensionalen so in endlich viele Teile zerlegen, dass man die Bestandteile zu zwei Kugeln desselben ursprünglichen Durchmessers zusammensetzen kann? Natürlich, wie das Banach-Tarski-Paradoxon erklärt.

(c) Wir betrachten das Ziegenproblem: In einer Spieleshow haben wir die Wahl zwischen drei Toren. Hinter einem der Tore ist ein Auto, hinter den beiden anderen befinden sich Ziegen. Wir wählen ein Tor, der Showmaster, der weiß, was hinter den Toren ist, öffnet ein anderes Tor, hinter dem eine Ziege steht. Sollen wir unsere Wahl ändern, also das verbleibende ungeöffnete Tor wählen?

Ja, das sollten wir definitiv tun. Aber warum?

(d) Der britische Astronom Bentley hat beobachtet, dass das Universum gemäß den Newtonschen Gravitationsgesetzen gar nicht existieren dürfte: Die Sterne sollten sich mit ihren Gravitationskräften gegenseitig anziehen und aufeinander zu bewegen, um dann zu einem großen Klumpen zu verschmelzen.

Tatsächlich hat man festgestellt, dass die Sterne sich (schwach) aufeinander zu bewegen, gleichzeitig aber expandiert das Universum. Einige Galaxien verschmelzen, andere entfernen sich sogar voneinander.

(e) Wie groß ist die Wahrscheinlichkeit, dass in einer Gruppe von sich zufällig treffenden 23 Personen, zwei Personen denselben Geburtstag besitzen? Größer als 50%!

(f) Meine Freunde haben mehr Freunde als ich!

Diese Beobachtung ist tatsächlich im folgenden Sinne fundiert. Wählt man eine Person P zufällig aus und fragt sie nach dem Namen Q einer befreundeten Person, dann ist die erwartete Zahl von Freundinnen/Freunden von Q größer als die von P.

Wie kann man dieses Phänomen erklären? Sehr populäre Personen haben eine größere Wahrscheinlichkeit als befreundete Person Q aufzutreten und haben wegen ihrer Popularität wahrscheinlich mehr Freundinnen/Freunde. Wir wählen also einerseits eine Ausgangsperson P und andererseits eine mit P befreundete Person Q zufällig aus: Zwei völlig verschiedene Zufallsprozesse.

(g) Und warum taucht die Ziffer 1 in „real-world data“ häufiger als die Ziffer 9 auf? Auf dieses Phänomen aufbauend hat man eine Software entworfen, die verdächtige Bilanzen in einem ersten Schnelltest überprüft.

(h) In jeder Gruppe von sechs Leuten gibt es *stets* drei, von denen je zwei miteinander befreundet sind, oder drei, von denen keine zwei miteinander befreundet sind.

2.1.2. Mathematische Notationen:

Symbol	Bedeutung
$:=$	Definition eines Wertes, z.B. $x := 5$, $M := \{1, 2, 3\}$
$:\Longleftrightarrow$	Definition einer Eigenschaft oder einer Schreibweise z.B. $m \in M :\Longleftrightarrow m$ ist Element von M
ex.	Abkürzung für „es gibt“, „es existiert“
f.a.	Abkürzung für „für alle“, „für jedes“
s.d.	Abkürzung für „so, dass“
$\Longrightarrow$	Abkürzung für „impliziert“ z.B.: Regen $\Longrightarrow$ nasse Straße
$\Longleftrightarrow$	Abkürzung für „genau dann, wenn“ z.B.: Klausur bestanden $\Longleftrightarrow$ die erreichte Prozentzahl z ist $\geqslant 50\%$
$\square$	markiert das Ende eines Beweises

Modellierung und Wertebereiche

In der Modellierung von Systemen, Aufgaben, Problemen oder Lösungen kommen **Objekte unterschiedlicher Art und Zusammensetzung** vor. Für Teile des Modells wird angegeben, aus welchem Wertebereich sie stammen, es wird zumeist aber offen gelassen, welchen konkreten Wert sie annehmen.

Ein **Wertebereich** ist eine Menge gleichartiger Werte. Wertebereiche werden aus Mengen und Strukturen darüber gebildet. Wertebereich

Beispiel 2.14 (Modellierung der Karten eines (Skat-)Kartenspiels).
Die Karten eines Skat-Kartenspiels lassen sich durch folgende Wertebereiche darstellen:

$$\begin{aligned}
\text{KartenArten} &:= \{\ \text{Kreuz}, \text{Pik}, \text{Herz}, \text{Karo}\ \} \\
\text{KartenSymbole} &:= \{\ 7, 8, 9, 10, \text{Bube}, \text{Dame}, \text{König}, \text{Ass}\ \} \\
\text{Karten} &:= \{\ (\text{Kreuz}, 7), (\text{Kreuz}, 8), \ldots, (\text{Kreuz}, \text{Ass}), \\
&\quad\ (\text{Pik}, 7), (\text{Pik}, 8), \ldots, (\text{Pik}, \text{Ass}), \\
&\quad\ (\text{Herz}, 7), (\text{Herz}, 8), \ldots, (\text{Herz}, \text{Ass}), \\
&\quad\ (\text{Karo}, 7), (\text{Karo}, 8), \ldots, (\text{Karo}, \text{Ass})\ \}.
\end{aligned}$$

$\square$Ende Beispiel 2.14

Wertebereiche sind u.a. wichtig

- zur Modellierung von Strukturen und Zusammenhängen,
- als Grundlage für alle anderen formalen Kalküle und
- als abstrakte Grundlage für Typen in Programmiersprachen.

Der grundlegende Kalkül zur Handhabung von Wertebereichen ist die **Mengenlehre**, bei der Mengen und Mengenoperationen betrachtet werden. Zur Modellierung von „zusammengesetzten Wertebereichen“ kann man z.B.

- Potenzmengen,
- kartesische Produkte und Tupel,
- Relationen,
- Folgen bzw. Worte und
- Funktionen

nutzen. Wir präzisieren diese Begriffe und lernen, wie sie in der Modellierung eingesetzt werden können.

2.2. Mengen

2.2.1. Was ist eine Menge?

Cantors naiver Mengenbegriff: (Georg Cantor, 1845–1918)
Cantors naiver Mengenbegriff besagt Folgendes: Eine Menge M ist eine Zusammenfassung von bestimmten, wohlunterschiedenen Objekten unserer Anschauung oder unseres Denkens, welche „Elemente der Menge M“ genannt werden, zu einem Ganzen.

Wir schreiben

$$m \in M$$

um auszusagen, dass M eine Menge ist und dass m Element der Menge M ist.
Wir schreiben

$$m \notin M$$

um auszusagen, dass m kein Element der Menge M ist.
Künftig werden wir solche Notationen festlegen, indem wir kurz Folgendes schreiben:

Notation:

$m \in M$ $\quad m \in M \quad :\Longleftrightarrow \quad m$ ist Element der Menge M.
$m \notin M$ $\quad m \notin M \quad :\Longleftrightarrow \quad m$ ist kein Element der Menge M.

Cantors Mengenbegriff ist problematisch und führt zu Widersprüchen. Russell gab folgendes Beispiel:

Die Russellsche Antinomie: (Bertrand Russell, 1872–1970)

> Sei N die Menge aller Mengen M, die sich nicht selbst enthalten
> (d.h.: $M \in N :\Longleftrightarrow M$ ist eine Menge, für die gilt: $M \notin M$).

Frage: Enthält N sich selbst (d.h. gilt $N \in N$)?

Klar: Entweder es gilt $N \in N$ oder es gilt $N \notin N$.

Fall 1: $N \notin N$. Gemäß Definition der Menge N gilt dann, dass $N \in N$.
Das ist ein Widerspruch.

Fall 2: $N \in N$. Gemäß Definition der Menge N gilt dann, dass $N \notin N$.
Das ist ein Widerspruch.

Somit führen beide Fälle zu einem Widerspruch, obwohl wir wissen, dass einer der beiden Fälle zutreffen müsste. Irgendetwas stimmt nicht mit Cantors naivem Mengenbegriff!

Um Russells Beispiel und den daraus resultierenden Widerspruch besser zu verstehen, betrachte man folgende Geschichte vom Barbier von Sonnenthal.

Der Barbier von Sonnenthal:

Im Städtchen Sonnenthal (in dem bekanntlich viele seltsame Dinge passieren) wohnt ein Barbier, der genau diejenigen männlichen Einwohner von Sonnenthal rasiert, die sich nicht selbst rasieren.

Frage: Rasiert der Barbier sich selbst?

Bemerkung 2.15. Um die Russellsche Antinomie zu vermeiden, hat man die Mengenlehre sehr sorgfältig axiomatisch aufgebaut (siehe z.B. [5]). Um ein allererstes Gefühl für einen solchen axiomatischen Zugang zu erhalten, beschreiben wir einige Axiome der **Zermelo-Fraenkel-Mengenlehre** in allerdings stark vereinfachter Form.

Zermelo-Fraenkel-Mengenlehre

Wir benutzen hier erst später eingeführte Notationen, deshalb ist es sinnvoll, diese Bemerkung beim ersten Lesen nur zu überfliegen. Die exakte Formulierung der Axiome geben wir in Abschnitt 9.6.2 an.

(a) Eine Reihe von Axiomen erlaubt das Bilden endlicher Mengen beginnend mit der leeren Menge.

- Im **Nullmengenaxiom** fordert man, dass die leere Menge $\emptyset$, also die Menge ohne Elemente, eine Menge ist.
- Im **Paarmengenaxiom** dürfen wir aus zwei Mengen A und B die „Paarmenge“ $\{A, B\}$ bauen. Beachte, dass die Paarmenge genau aus den Elementen A und B besteht. Das **Vereinigungsaxiom** erlaubt dann die Bildung der Vereinigungsmenge $A \cup B$, die genau aus den Elementen besteht, die Elemente von A oder Elemente von B sind.

(b) Mit den bisherigen Axiomen können wir bereits jede natürliche Zahlen kodieren: Die Null kodieren wir mit der leeren Menge $\emptyset$. Haben wir die Zahl n mit der Menge x kodiert, dann kodieren wir die Zahl $n+1$ mit der Menge $x \cup \{x\}$. Das gibt die kodierte Zahlenfolge

$$\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \{\emptyset, \{\emptyset\}, \{\{\emptyset, \{\emptyset\}\}\}\}, \dots$$

Irgendwie schräg, aber wenn man alle Mengen aus der leeren Menge aufbauen will, dann ist das schon vernünftig. Allerdings hat man zwar jede natürliche Zahl darstellen können, nicht aber die Menge $\mathbb{N}$ aller natürlichen Zahlen. Dies gelingt mit dem **Unendlichkeitsaxiom** (siehe Abschnitt 9.6.2) und dem **Aussonderungsaxiom**.

Aussonderungsaxiom

- Das **Aussonderungsaxiom** ist für uns besonders wichtig. Für eine Menge A und eine Eigenschaft E wird gefordert, dass es eine Menge gibt, die aus allen Elementen von A besteht, die die Eigenschaft E besitzen. Wir schreiben

$$\{x \in A : x \text{ hat Eigenschaft } E\}$$

für diese „Teilmenge“ von A.

Potenzmengenaxiom

- Das **Potenzmengenaxiom** garantiert, dass für jede Menge A auch die Potenzmenge

$$\mathcal{P}(A)$$

eine Menge ist. Die Potenzmenge $\mathcal{P}(A)$ besitzt genau die Teilmengen von A als Elemente. Die Potenzmenge $\mathcal{P}(\mathbb{N})$ können wir mit der Menge der reellen Zahlen im Intervall $[0,1]$ in Verbindung bringen: Repräsentiere eine Zahl $x \in [0,1]$ durch ihre Binärdarstellung, bzw. durch die Menge aller Positionen mit „Bit“ 1.

Fundierungsaxiom

(c) Das **Fundierungsaxiom** (siehe Abschnitt 9.6.2) verhindert zum Beispiel, dass eine Menge sich selbst als Element enthält. Ist das ein sinnvolles Axiom? Vom Standpunkt der Informatik definitiv, denn wenn wir eine Menge A durch die Menge ihrer Elemente beschreiben wollen, dann werden wir sonst in eine nicht-endende Rekursion gezwungen.

Übrigens, die „Menge“ N der Russelschen Antinomie stimmt mit dem Fundierungsaxiom mit der „Menge“ aller Mengen überein. Aber N enthält sich nicht selbst als Element und damit kann N nicht alle Mengen enthalten: Das Fundierungsaxiom „verbietet“ die Russelsche Antinomie.

Extensionalitätsaxiom

(d) Das **Extensionalitätsaxiom** besagt, dass zwei Mengen A und B genau dann gleich sind, wenn sie dieselben Elemente besitzen:

$$A = B :\Longleftrightarrow \Big(\text{f.a. } x \in A \text{ gilt } x \in B\Big) \text{ und } \Big(\text{f.a. } x \in B \text{ gilt } x \in A\Big).$$

Unvollständigkeitssatz

Sind mit der Zermelo-Fraenkel-Mengenlehre alle möglichen Widersprüche ausgeschlossen? Nein! Kurt Gödel (1906-1978) hat in seinem zweiten **Unvollständigkeitssatz** unter anderem gezeigt, dass die Widerspruchsfreiheit der Zermelo-Fraenkel-Mengenlehre nicht innerhalb der Zermelo-Fraenkel-Mengenlehre gezeigt werden kann!

Sollten Sie deshalb Schlafprobleme entwickeln? Nicht wirklich, die Zermelo-Fraenkel-Mengenlehre „steht“ seit über 90 Jahren und ihre Widerspruchsfreiheit wird nicht in Frage gestellt. Allerdings, die letzte Sicherheit werden wir nie bekommen!

Sofern man sich der Problematik bewusst ist, kann man den Umgang mit kritischen „Mengen“ im „täglichen Informatik-Gebrauch“ leicht umgehen. Wir arbeiten daher weiter mit einem naiven Mengenbegriff und wenden dabei die folgenden Grundsätze an.

Beschreibung bzw. Definition von Mengen:

Wir beschreiben bzw. definieren Mengen

- *extensional* (bzw. konstruktiv), durch Aufzählen der Elemente, z.B.

$$M_1 := \{0,1,2,3,4,5\} = \{0,1,2,\ldots,5\}$$

oder

- *intensional* (bzw. deskriptiv), durch Angabe charakteristischer Eigenschaften der Elemente der Menge, z.B.

$$\begin{aligned} M_2 &:= \{x : x \in M_1 \text{ und } x \text{ ist gerade}\} \\ &= \{x \in M_1 : x \text{ ist gerade}\} \\ &= \{x : x \text{ ist eine natürliche Zahl und } x \text{ ist gerade und } 0 \leqslant x \leqslant 5\}. \end{aligned}$$

Extensional lässt sich die Menge M_2 folgendermaßen beschreiben:

$$M_2 = \{0, 2, 4\}.$$

Oft schreibt man statt „:“ auch „|“ und statt „und“ einfach ein „Komma“, also

$$M_2 = \{x \mid x \in M_1, x \text{ gerade}\}.$$

Vorsicht:

- $\{x : 0 \leqslant x \leqslant 5\}$ definiert nicht eindeutig eine Menge, weil nicht festgelegt ist, ob x beispielsweise eine ganze Zahl oder eine reelle Zahl ist.
- $\{M : M \text{ ist eine Menge}, M \notin M\}$ führt zur Russellschen Antinomie.

Um solche Probleme zu vermeiden, sollte man bei intensionalen Mengendefinitionen immer angeben, aus welcher anderen Menge die ausgewählten Elemente kommen sollen, also:

$$\{x \in M : x \text{ hat Eigenschaft(en) } E\},$$

wobei M eine Menge und E eine Eigenschaft oder eine Liste von Eigenschaften ist, die jedes einzelne Element aus M haben kann oder nicht. Beachte, dass wir hier das Aussonderungsaxiom der Zermelo-Fraenkel-Mengenlehre anwenden (siehe Bemerkung 2.15).

Wichtige grundsätzliche Eigenschaften von Mengen:

- Alle Elemente einer Menge sind verschieden. D.h. ein Wert ist entweder Element der Menge oder eben nicht – aber er kann nicht „mehrfach“ in der Menge vorkommen.
- Die Elemente einer Menge haben keine feste Reihenfolge.
- Dieselbe Menge kann auf verschiedene Weisen beschrieben werden, z.B.

$$\{1, 2, 3\} = \{1, 2, 2, 3\} = \{2, 1, 3\} = \{i : i \text{ ist eine ganze Zahl}, 0 < i \leqslant 3\}.$$

- Mengen können aus atomaren oder aus zusammengesetzten Elementen gebildet werden. Menge kann auch „verschiedenartige“ Elemente enthalten. Zum Beispiel besteht die Menge

$$M := \{\ 1,\ (\text{Pik}, 8),\ \{\text{rot}, \text{blau}\},\ 5,\ 1\ \}$$

aus vier Elementen: dem atomaren Wert 1, dem Tupel $(\text{Pik}, 8)$, der Menge $\{\text{rot}, \text{blau}\}$ und dem atomaren Wert 5.

Notationen für bestimmte Zahlenmengen:

$\mathbb{N}$	:=	Menge der natürlichen Zahlen := $\{0, 1, 2, 3, ...\}$
$\mathbb{N}_{>0}$	:=	Menge der positiven natürlichen Zahlen := $\{1, 2, 3, ...\}$
$\mathbb{Z}$	:=	Menge der ganzen Zahlen := $\{0, 1, -1, 2, -2, 3, -3, ...\}$
$\mathbb{Q}$	:=	Menge der rationalen Zahlen := $\{\frac{a}{b} : a, b \in \mathbb{Z},\ b \neq 0\}$
$\mathbb{R}$	:=	Menge der reellen Zahlen

Frage 2.16: Gibt es eine „Menge aller Mengen“?

Antwort: Nein! Denn wäre U die Menge aller Mengen, so wäre auch $N := \{M \in U : M \notin M\}$ eine Menge. Dies führt aber wieder zur Russellschen Antinomie, da die Frage „Ist $N \in N$?” nicht geklärt werden kann.

Kann man mit Mengen in der Programmiersprache Python arbeiten?

Python

Bemerkung 2.17. (Mengen in Python)

1. Extensional definierte Mengen wie etwa $M = \{1, a, A, 1234\}$ werden mit

   ```
   M = { 1, 'a', 'A', 1234 }
   ```

   in Python eingeführt. M besitzt den Datentyp `set` (Menge).

2. Für ***iterierbare*** Python-Objekte I (Listen, Tupel, Strings, Dictionaries, Ranges etc.), deren „Elemente“ hashbar sind, ist

   ```
   M = set(I)
   ```

   ein Objekt vom Datentyp `set`, das genau die „Elemente“ von I als Elemente besitzt. Hier sind zwei Beispiele für die Datentypen `list` (Liste) und `str` (String):

   - `set(['Frankfurt', 'Offenbach'])` führt auf die Menge $\{\text{Offenbach}, \text{Frankfurt}\}$,
   - `set('Frankfurt')` führt auf die Menge $\{k, u, n, a, r, F, f, t\}$.

3. Ist M ein Objekt vom Datentyp `set`, dann liefert der Funktionsaufruf

   ```
   x in M
   ```

   den Wert `True`, wenn x ein Element von M ist und ansonsten `False`. Mit der Schleife

   ```
   for x in M
   ```

   werden die Elemente von M in irgendeiner Reihenfolge verarbeitet.

4. Um das Element e zur Menge M hinzuzufügen, benutze die Anweisung

   ```
   M.add(e)
   ```

   Um das Element e – falls enthalten – aus M zu entfernen, benutze die Anweisung

   ```
   M.remove(e)
   ```

(Eine Fehlernachricht wird gegeben, falls e nicht in M enthalten ist.) Um das Element e aus M zu entfernen, selbst wenn e kein Element von M ist, benutze die Anweisung

```
M.discard(e)
```

Falls e nicht in M enthalten ist, erfolgt diesmal keine Fehlernachricht.

5. Für eine Python-Funktion `E`() und eine bereits in Python definierte Menge `M` möchten wir die intensional definierte Menge

$$N = \{\ x \in M\ :\ \texttt{E}(x)\ \}$$

aller Elemente in M bestimmen, die die „Eigenschaft“ E besitzen und benutzen dazu den Code

```
N = set()
for x in M:
   if E(x):
      N.add(x)
```

6. Aber Achtung bei Mengenzuweisungen! Nach den Print-Anweisungen in

```
M = { 1,2,3,4 }
N = M
M.discard(1)
print(N)
N.discard(2)
print(M)
```

erhalten wir zuerst $N = \{2, 3, 4\}$ und danach $M = \{3, 4\}$, denn nach der Zuweisung `N = M` zeigen M und N auf denselben Speicherplatz und damit auf dieselbe Menge! Soll nur eine Kopie der Menge M angelegt werden, benutze statt $N = M$ die Zuweisung

```
N = M.copy()
```

um eine Kopie von M auf einem nicht belegten Speicherbereich anzulegen.

2.2.2. Mengenalgebra

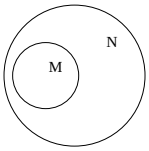
In diesem Abschnitt betrachten wir grundlegende Operationen auf Mengen und geben einige, wenn auch sehr einfache Beispiele von mathematischen Beweisen.

Zuerst führen wir den Begriff von Teil- und Obermengen ein.

Definition 2.18 (Teilmengen). Seien M, N Mengen.

(a) M ist eine **Teilmenge** von N (kurz: $M \subseteq N$), wenn f.a. $x \in M$ gilt $x \in N$. Teilmenge $M \subseteq N$

Skizze:



echte Teilmenge
$M \subsetneq N$

(b) M ist eine **echte Teilmenge** von N (kurz: $M \subsetneq N$), wenn $M \subseteq N$ und $M \neq N$.

Obermenge
$M \supseteq N$

(c) M ist eine **Obermenge** von N (kurz: $M \supseteq N$), wenn $N \subseteq M$.

echte Obermenge
$M \supsetneq N$

(d) M ist eine **echte Obermenge** von N (kurz: $M \supsetneq N$), wenn $M \supseteq N$ und $M \neq N$.

Also ist M genau dann eine Teilmenge von N, wenn alle Elemente von M auch Elemente von N sind. Wann sollten wir zwei Mengen gleich nennen?

Definition 2.19 (Mengengleichheit). Seien M, N Mengen.

$M = N$

Die Mengen M und N heißen gleich (kurz: $M = N$), falls gilt:

$$M \subseteq N \text{ und } N \subseteq M.$$

Es gibt genau eine leere Menge.

Satz 2.20. Es gibt genau eine Menge, die keine Elemente enthält.

Beweis: Seien M und N zwei Mengen, die keine Elemente besitzen. Wir zeigen, dass $M \subseteq N$ gilt. Warum? Es ist zu zeigen, dass f.a. $x \in M$ gilt $x \in N$. Die Menge M besitzt aber keine Elemente x und die Implikation „wenn $x \in M$, dann $x \in N$“ ist wahr, weil die Bedingung „$x \in M$“ der Implikation immer falsch ist. Aus genau demselben Grund gilt $N \subseteq M$. Die Gleichheit $M = N$ folgt mit Definition 2.19.

Aber warum gibt es denn überhaupt eine Menge ohne Elemente? Um diese anscheinend triviale Frage beantworten zu können, wird das Nullmengenaxiom (siehe Bemerkung 2.15) in die Zermelo-Fraenkel-Mengenlehre aufgenommen: Die Existenz einer Menge ohne Elemente muss gefordert werden. □

Definition 2.21 (leere Menge).

leere Menge
$\emptyset$

Die **leere Menge** ist die eindeutig bestimmte Menge, die keine Elemente enthält. Wir bezeichnen sie mit $\emptyset$.

Beachte: $\emptyset \neq \{\emptyset\}$, denn $\emptyset$ ist die Menge, die keine Elemente enthält, während $\{\emptyset\}$ eine Menge ist, die ein Element, nämlich $\emptyset$ enthält.

Bemerkung 2.22. Die leere Menge M wird in Python durch

```
M = set()
```

definiert. Achtung: Python interpretiert `M = {}` als das leere Dictionary und nicht als die leere Menge.

Wir müssen vom Begriff der Gleichheit von Mengen verlangen, dass zwei Mengen genau dann gleich sind, wenn sie dieselben Elemente besitzen. Haben wir dies erreicht?

Satz 2.23. Seien M, N, P Mengen. Dann gilt:

(a) $M = N \iff$ (f.a. $x \in M$ gilt $x \in N$) und (f.a. $x \in N$ gilt $x \in M$).

(b) $M \subseteq N$ und $N \subseteq P \implies M \subseteq P$.

Beweis:

(a) Die Mengengleichheit haben wir in Definition 2.19 eingeführt. Danach gilt

$$\begin{aligned} M = N &\overset{\text{Def. 2.19}}{\iff} M \subseteq N \text{ und } N \subseteq M \\ &\overset{\text{Def. 2.18}}{\iff} (\text{ f.a. } x \in M \text{ gilt } x \in N) \text{ und } (\text{ f.a. } x \in N \text{ gilt } x \in M), \end{aligned}$$

und das war zu zeigen.

(b) Es gelte $M \subseteq N$ und $N \subseteq P$.

Behauptung: $M \subseteq P$, d.h. f.a. $m \in M$ gilt $m \in P$.

Beweis: Sei $m \in M$ beliebig. Wir zeigen, dass $m \in P$:

$$m \in M \overset{\text{nach Vor.: } M \subseteq N}{\implies} m \in N \overset{\text{nach Vor.: } N \subseteq P}{\implies} m \in P.$$

□

Wie weist man nach, dass zwei Mengen gleich sind?

Bemerkung 2.24. Seien X und Y Mengen.

(a) Um die Inklusion $X \subseteq Y$ nachzuweisen, genügt es, für ein *beliebiges* Element x der Menge X zu zeigen, dass x auch ein Element der Menge Y ist.

(b) Um die Mengengleichheit $X = Y$ nachzuweisen, genügt nach Definition 2.19 der Nachweis der beiden Inklusionen $X \subseteq Y$ und $Y \subseteq X$.

Bemerkung 2.25. (Python: Teilmengen, Obermengen und Mengengleichheit) Python

M und N seien Python-Objekte vom Datentyp `set`.

1. M ist genau dann eine Teilmenge von N, wenn die Python-Methode

```
M.issubset(N)
```

den Wert `True` ausgibt. Äquivalent dazu kann man auch den Ausdruck

```
M <= N
```

benutzen. Soll festgestellt werden, ob M eine echte Teilmenge von N ist, benutze den Test

```
M < N
```

2. M ist genau dann eine Obermenge von N ist, wenn die Python-Methode

```
M.issupset(N)
```

den Wert `True` ausgibt. Äquivalent dazu kann man auch den Ausdruck

```
M >= N
```

benutzen. Soll festgestellt werden, ob M eine echte Obermenge von N ist, benutze den Test

```
M > N
```

3. M und N sind genau dann gleich, wenn der Python-Test

```
M == N
```

die Ausgabe `True` gibt.

Mit Mengen-Operationen erhalten wir neue Mengen aus bereits konstruierten Mengen. Die wichtigsten Operationen werden in der nächsten Definition eingeführt.

Definition 2.26. Seien M und N Mengen.

Durchschnitt $M \cap N$

(a) Der **Durchschnitt** von M und N ist die Menge

$$M \cap N \ := \ \{x : x \in M \text{ und } x \in N\}.$$

Vereinigung $M \cup N$

(b) Die **Vereinigung** von M und N ist die Menge

$$M \cup N \ := \ \{x : x \in M \text{ oder } x \in N\}.$$

Differenz $M \setminus N$

(c) Die **Differenz** von M und N ist die Menge

$$M \setminus N \ := \ M - N := \{x : x \in M \text{ und } x \notin N\}.$$

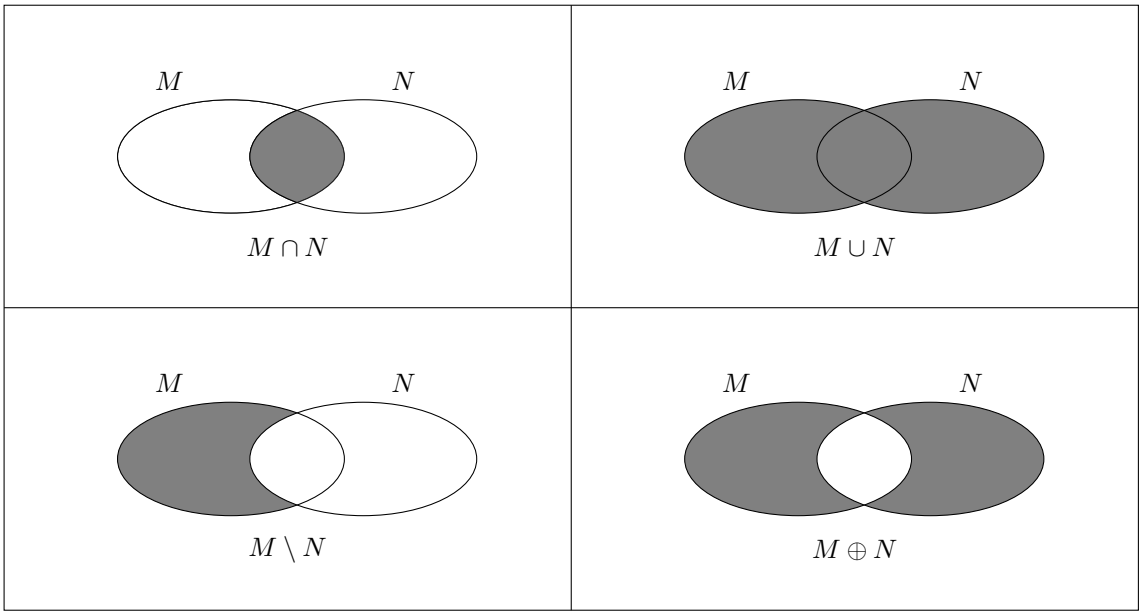
symmetrische Differenz $M \oplus N$

(d) Die **symmetrische Differenz** von M und N ist die Menge

$$M \oplus N \ := \ (M \setminus N) \cup (N \setminus M).$$

Manchmal schreibt man auch $M \triangle N$.

Veranschaulichung durch Venn-Diagramme:



Notation 2.27 (disjunkte Mengen).
Zwei Mengen M und N heißen **disjunkt**, falls $M \cap N = \emptyset$, d.h. falls sie keine gemeinsamen Elemente besitzen. Manchmal schreiben wir $M \dot\cup N$

$$M \dot\cup N,$$

um die Menge $M \cup N$ zu bezeichnen und gleichzeitig auszudrücken, dass $M \cap N = \emptyset$.

Bemerkung 2.28. (Python: Operationen auf Mengen) Python
M und N seien Python-Objekte vom Datentyp `set`.

1. M und N sind disjunkt, wenn die Python-Methode

   `M.isdisjoint(N)`

   den Wert `True` ausgibt.

2. Der Durchschnitt $K = M \cap N$ von M und N wird durch

   `K = M.intersection(N)` bzw. `K = M & N`

   beschrieben.

3. Die Vereinigung $K = M \cup N$ von M und N wird durch

   `K = M.union(N)` bzw. `K = M | N`

   beschrieben.

4. Die Mengendifferenz $K = M \setminus N$ wird durch

$$K = \texttt{M.difference(N)} \text{ bzw. } \texttt{K = M - N}$$

beschrieben.

5. Die symmetrische Differenz $K = M \oplus N$ von M und N wird durch

$$\texttt{K = M.symmetric\_difference(N)} \text{ bzw. } \texttt{K = M \^{} N}$$

beschrieben.

Rechenregeln für Durchschnitt und Vereinigung:

Satz 2.29. Seien M, N, P Mengen. Dann gelten:

Idempotenz (a) **Idempotenz**:

$$M \cap M = M \quad \text{und} \quad M \cup M = M.$$

Kommutativität (b) **Kommutativität**:

$$M \cap N = N \cap M \quad \text{und} \quad M \cup N = N \cup M.$$

Assoziativität (c) **Assoziativität**:

$$M \cap (N \cap P) = (M \cap N) \cap P \quad \text{und} \quad M \cup (N \cup P) = (M \cup N) \cup P.$$

Absorption (d) **Absorption**:

$$M \cap (M \cup N) = M \quad \text{und} \quad M \cup (M \cap N) = M.$$

Distributivität (e) **Distributivität**:

$$M \cap (N \cup P) = (M \cap N) \cup (M \cap P) \quad \text{und} \quad M \cup (N \cap P) = (M \cup N) \cap (M \cup P).$$

Beweis:

(a)

$$\begin{aligned} M \cap M &\overset{\text{Def. 2.26((a))}}{=} \{x : x \in M \text{ und } x \in M\} \\ &= \{x : x \in M\} \\ &= M. \end{aligned}$$

Analog: $M \cup M = M$.

(b)

$$\begin{aligned} M \cap N &\overset{\text{Def. 2.26((a))}}{=} \{x : x \in M \text{ und } x \in N\} \\ &= \{x : x \in N \text{ und } x \in M\} \\ &\overset{\text{Def. 2.26((a))}}{=} N \cap M. \end{aligned}$$

Analog: $M \cup N = N \cup M$.

(c)

$$\begin{aligned} M \cap (N \cap P) &\overset{\text{Def. 2.26((a))}}{=} \{x : x \in M \text{ und } x \in N \cap P\} \\ &\overset{\text{Def. 2.26((a))}}{=} \{x : x \in M \text{ und } (x \in N \text{ und } x \in P)\} \\ &= \{x : (x \in M \text{ und } x \in N) \text{ und } x \in P\} \\ &\overset{\text{Def. 2.26((a))}}{=} \{x : x \in M \cap N \text{ und } x \in P\} \\ &\overset{\text{Def. 2.26((a))}}{=} (M \cap N) \cap P. \end{aligned}$$

Analog: $M \cup (N \cup P) = (M \cup N) \cup P$.

(d) Wir beweisen, dass $M \cap (M \cup N) = M$ in zwei Schritten:

Schritt 1: Zeige, dass $M \cap (M \cup N) \supseteq M$.

Schritt 2: Zeige, dass $M \cap (M \cup N) \subseteq M$.

Aus Satz 2.23((a)) folgt dann, dass $M \cap (M \cup N) = M$.

„$\supseteq$“ Zu Schritt 1:

Behauptung: $M \cap (M \cup N) \supseteq M$, d.h. f.a. $m \in M$ gilt $m \in M \cap (M \cup N)$.

Beweis: Sei $m \in M$ beliebig. Zu zeigen: $m \in M \cap (M \cup N)$. Wegen $m \in M$ gilt auch $m \in M \cup N$ (gemäß Definition 2.26((b))). Wegen $m \in M$ und $m \in M \cup N$ gilt gemäß Definition 2.26((a)), dass $m \in M \cap (M \cup N)$.

„$\subseteq$“ Zu Schritt 2:

Behauptung: $M \cap (M \cup N) \subseteq M$, d.h. f.a. $m \in M \cap (M \cup N)$ gilt $m \in M$.

Beweis: Sei $m \in M \cap (M \cup N)$ beliebig. Zu zeigen: $m \in M$. Wegen $m \in M \cap (M \cup N)$ gilt gemäß Definition 2.26((a)), dass $m \in M$ und $m \in M \cup N$. Insbesondere ist also $m \in M$.

Insgesamt haben wir damit gezeigt, dass $M \cap (M \cup N) = M$. Analog: $M \cup (M \cap N) = M$.

(e) Analog; Details: Übung.

□

Im nächsten Resultat wird eine wichtige Darstellung der symmetrischen Differenz gezeigt. Wir benutzen im Beweis Fallannahmen. Für den Nachweis der Mengengleichheit wenden wir die in Bemerkung 2.24 skizzierte Methode an.

Satz 2.30. Für alle Mengen M und N gilt:

$$M \oplus N = (M \cup N) \setminus (M \cap N).$$

Beweis: Um die behauptete Gleichheit nachzuweisen, genügt der Nachweis der beiden Teilmengenbeziehungen (oder „Inklusionen“)

$$M \oplus N \subseteq (M \cup N) \setminus (M \cap N) \text{ und } M \oplus N \supseteq (M \cup N) \setminus (M \cap N).$$

„$\subseteq$“ Wir zeigen die erste Inklusion, also $M \oplus N \subseteq (M \cup N) \setminus (M \cap N)$. Sei $x \in M \oplus N$ beliebig. Wir müssen zeigen, dass $x \in (M \cup N) \setminus (M \cap N)$ gilt.

Nach Definition der symmetrischen Differenz gilt $M \oplus N := (M \setminus N) \cup (N \setminus M)$. Also gilt $x \in M \setminus N$ oder $x \in N \setminus M$. Wir unterscheiden zwischen den beiden Fällen.

Fall 1: Es gilt $x \in M \setminus N$. Nach Definition der Mengendifferenz ist $x \in M$ und $x \notin N$. Wegen $x \in M$ gilt $x \in M \cup N$, denn M ist eine Teilmenge von $M \cup N$. Wegen $x \notin N$ gilt $x \notin M \cap N$, denn $M \cap N$ ist eine Teilmenge von N. Nach Definition der Mengendifferenz folgt $x \in (M \cup N) \setminus (M \cap N)$ und die Inklusion ist in diesem Fall nachgewiesen.

Fall 2: Es gilt $x \in N \setminus M$. Die Argumentation verläuft wie in Fall 1, wenn die Rollen von M und N vertauscht werden.

Damit sind alle möglichen Fälle abgedeckt.

„$\supseteq$“ Wir zeigen die zweite Inklusion, also $M \oplus N \supseteq (M \cup N) \setminus (M \cap N)$. Sei $x \in (M \cup N) \setminus (M \cap N)$ beliebig. Wir müssen zeigen, dass $x \in M \oplus N$ gilt.

Nach Definition der Mengendifferenz folgt $x \in M \cup N$ und $x \notin M \cap N$. Wir unterscheiden, ob $x \in M$ oder $x \in N$ gilt.

Fall 1: Es gilt $x \in M$. Da $x \in M$ und $x \notin M \cap N$ gilt, ist x kein Element von N. Nach Definition der Mengendifferenz folgt $x \in M \setminus N$ und insbesondere $x \in (M \setminus N) \cup (N \setminus M)$. Die Inklusion ist in diesem Fall nachgewiesen, denn $M \oplus N := (M \setminus N) \cup (N \setminus M)$.

Fall 2: Es gilt $x \in N$. Die Argumentation verläuft wie in Fall 1, wenn die Rollen von M und N vertauscht werden.

Damit sind alle möglichen Fälle abgedeckt.

□

2.2.3. Das Komplement einer Menge

Komplement $\overline{M}$

Das **Komplement** einer Menge M (kurz: $\overline{M}$) soll die Menge aller Elemente sein, die **nicht** zu M gehören. Bei der präzisen Definition von $\overline{M}$ ist allerdings wieder Vorsicht geboten. Denn wenn wir einfach

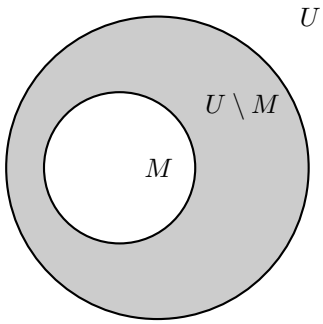
$$\overline{M} := \{x : x \notin M\}$$

setzen, so gilt für die leere Menge $\emptyset$, dass ihr Komplement $\overline{\emptyset}$ einfach **alles** enthält – und dann wäre

$$\{M : M \in \overline{\emptyset} \text{ und } M \text{ ist eine Menge}\}$$

die „Menge aller Mengen”, und dass es die nicht geben kann, haben wir bereits bei der Beantwortung von Frage 2.16 gesehen.

Daher betrachten wir Mengen stets innerhalb eines festen Universums U, das selbst eine Menge ist (die wir jeweils im Kontext angeben müssen). Für $M \subseteq U$ setzen wir dann $\overline{M} := U \setminus M$ und bezeichnen $\overline{M}$ als das Komplement von M in U.



Rechenregeln für Komplemente:

Satz 2.31. Sei U unser festes Universum, das selbst eine Menge ist, und seien $M, N \subseteq U$. Dann gelten:

(a) **Doppelte Negation**: Doppelte Negation

$$\overline{(\overline{M})} = M.$$

(b) **De Morgansche Regeln**: De Morgansche Regeln

$$\overline{M \cap N} = \overline{M} \cup \overline{N} \qquad \text{und} \qquad \overline{M \cup N} = \overline{M} \cap \overline{N}.$$

(c) **Inversionsregeln**: Inversionsregeln

$$M \cap \overline{M} = \emptyset \qquad \text{und} \qquad M \cup \overline{M} = U.$$

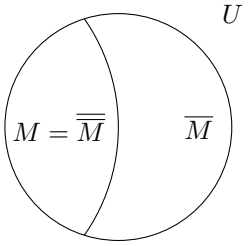
(d) **Identitätsregeln**: Identitätsregeln

$$M \cap U = M \qquad \text{und} \qquad M \cup \emptyset = M.$$

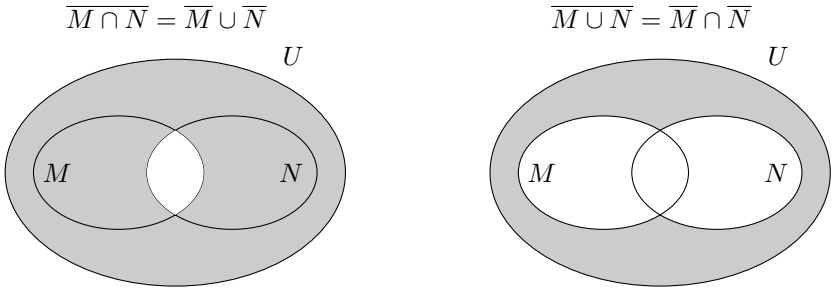
Beweis: Übung. □

Veranschaulichung durch Venn-Diagramme:

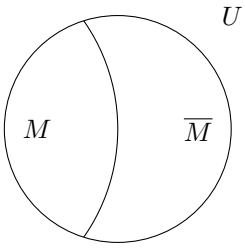
- Doppelte Negation:



- De Morgansche Regeln:



- Inversionsregel:



2.2.4. Die Potenzmenge

Definition 2.32.

Potenzmenge
$\mathcal{P}(M)$

Die **Potenzmenge** (engl.: power set) einer Menge M (kurz: $\mathcal{P}(M)$) ist die Menge aller Teilmengen von M. D.h.:

$$\mathcal{P}(M) \;=\; \{X : X \subseteq M\}.$$

Beispiel 2.33.

- $\mathcal{P}(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$.
- $\mathcal{P}(\{1, 2, 3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$.
- $\mathcal{P}(\emptyset) = \{\emptyset\}$.
  Insbesondere gilt: $\mathcal{P}(\emptyset) \neq \emptyset$.

Notation 2.34.

Pow(M)
2^M

In manchen Büchern wird $\mathcal{P}(M)$ auch mit Pow(M) (für „power set“) oder mit 2^M bezeichnet. Später, in Folgerung 2.58, werden wir nachweisen, dass für jede endliche Menge M gilt:

$$|\mathcal{P}(M)| \;=\; 2^{|M|}.$$

2.3. Kartesische Produkte und Relationen

2.3.1. Paare, Tupel und kartesische Produkte

In Mengen spielt die Reihenfolge, in der Elemente aufgeführt werden, keine Rolle. In Paaren oder allgemeiner in Tupeln hingegen ist die Reihenfolge wichtig.

Definition 2.35 (Paare und Tupel).

(a) Für beliebige Objekte a, b bezeichnet (a, b) das geordnete **Paar** mit Komponenten a und b. — Paar

(b) Für $k \in \mathbb{N}$ und beliebige Objekte $a_1, \ldots, a_k$ bezeichnet $(a_1, \ldots, a_k)$ das **k-Tupel** mit Komponenten $a_1, \ldots, a_k$. — k-Tupel

(c) Die **Gleichheit** zweier Tupel ist wie folgt definiert:
F.a. $k, \ell \in \mathbb{N}$ und $a_1, \ldots, a_k, b_1, \ldots, b_\ell$ gilt:

$$(a_1, \ldots, a_k) = (b_1, \ldots, b_\ell) \ :\Longleftrightarrow \ k = \ell \text{ und } a_1 = b_1 \text{ und } a_2 = b_2 \text{ und } \cdots \text{ und } a_k = b_k.$$

Notation 2.36. Tupel bezeichnet man auch manchmal als Vektoren oder Folgen.

Bemerkung 2.37.

(a) Für $k = 0$ gibt es genau ein k-Tupel, nämlich das **leere Tupel** $()$, das keine Komponente(n) hat. — leeres Tupel $()$

(b) Man beachte den Unterschied zwischen Tupeln und Mengen: z.B.
- $(1, 2) \neq (2, 1)$, aber $\{1, 2\} = \{2, 1\}$.
- $(1, 1, 2) \neq (1, 2)$, aber $\{1, 1, 2\} = \{1, 2\}$.

Definition 2.38.

(a) Sei $k \geqslant 2$ eine natürliche Zahl und sei M eine Menge. Die **k-te Potenz** von M ist die Menge — k-te Potenz M^k

$$M^k \ := \ \{(m_1, \ldots, m_k) : m_1 \in M, \ldots, m_k \in M\}.$$

Wir setzen $M^0 := \{()\}$ und $M^1 := M$. Also besteht M^0 aus genau einem Element, dem leeren Tupel.

(b) Das **kartesische Produkt** (bzw. **Kreuzprodukt**) zweier Mengen M, N ist die Menge — kartesisches Produkt, Kreuzprodukt $M \times N$

$$M \times N \ := \ \{(m, n) : m \in M, \, n \in N\}.$$

(c) Sei $k \in \mathbb{N}_{>0}$ und seien $M_1, \ldots, M_k$ Mengen. Das kartesische Produkt von $M_1, \ldots, M_k$ ist die Menge

$$M_1 \times \cdots \times M_k \ := \ \{(m_1, \ldots, m_k) : m_1 \in M_1, \ldots, m_k \in M_k\}.$$

Beispiel 2.39. Sei $M = \{a, b\}$ und $N = \{1, 2, 3\}$. Dann gilt:

- $M \times N = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$.
- $M \times \{1\} = \{(a, 1), (b, 1)\}$.
- $M \times \emptyset = \emptyset$.
- $M^2 = \{(a, a), (a, b), (b, a), (b, b)\}$.
- $M^1 = \{(a), (b)\}$.

- $M^0 = \{()\}$.
- $\emptyset^2 = \emptyset$.
- $\emptyset^1 = \emptyset$.
- $\emptyset^0 = \{()\}$.
- In Beispiel 2.14 hatten wir die Karten eines Skat-Kartenspiels durch folgende Wertebereiche modelliert:

$$\begin{aligned} \text{KartenArten} &= \{\text{Kreuz}, \text{Pik}, \text{Herz}, \text{Karo}\}, \\ \text{KartenSymbole} &= \{7, 8, 9, 10, \text{Bube}, \text{Dame}, \text{König}, \text{Ass}\}, \\ \text{Karten} &= \text{KartenArten} \times \text{KartenSymbole}. \end{aligned}$$

- Uhrzeiten kann man repräsentieren durch Elemente der Menge

$$\text{Uhrzeiten} := \text{Stunden} \times \text{Minuten} \times \text{Sekunden},$$

wobei

$$\begin{aligned} \text{Stunden} &:= \{0, 1, 2, ..., 23\}, \\ \text{Minuten} &:= \{0, 1, 2, ..., 59\}, \\ \text{Sekunden} &:= \{0, 1, 2, ..., 59\}. \end{aligned}$$

Das Tupel $(9, 45, 0)$ repräsentiert dann die Uhrzeit „9 Uhr, 45 Minuten und 0 Sekunden“.

Python

Bemerkung 2.40. (Python: Tupel und kartesische Produkte)
Tupel werden in Python durch den Datentyp `tuple` (*Tupel*) repräsentiert.

1. Ein Tupel wie etwa $T = (1, 2, a)$ wird mit

```
T = (1,2,'a')
```

in Python eingeführt. T besitzt den Datentyp `tuple`.

2. Für iterierbare Objekte I ist

```
T = tuple(I)
```

ein Objekt vom Datentyp `tuple`. Das leere Tupel wird repräsentiert durch

`T = tuple()` bzw. `T = ()`

3. Für Mengen $M1, \ldots, Mk$ möchten wir das kartesische Produkt $M = M1 \times \cdots \times Mk$ bestimmen. Das gelingt mit dem Submodul `product` des Python-Moduls `itertools`[4] und der Anweisung

```
M = set( itertools.product( M1, ..., Mk ))
```

Notation 2.41.

[4] Benutze die Anweisung `import itertools`

(a) Ist $k \in \mathbb{N}_{>0}$ und sind $z_1, \ldots, z_k$ Zahlen, so schreiben wir

$$\sum_{i=1}^{k} z_i \qquad \text{bzw.} \qquad \sum_{i \in \{1,\ldots,k\}} z_i$$

um die Summe der Zahlen $z_1, \ldots, z_k$ zu bezeichnen (d.h. die Zahl $z_1 + \cdots + z_k$).
Wir schreiben

$$\prod_{i=1}^{k} z_i \qquad \text{bzw.} \qquad \prod_{i \in \{1,\ldots,k\}} z_i$$

um das Produkt der Zahlen $z_1, \ldots, z_k$ zu bezeichnen (d.h. die Zahl $z_1 \cdot \cdots \cdot z_k$).

(b) Sind $M_1, \ldots, M_k$ Mengen, so schreiben wir

$$\bigcup_{i=1}^{k} M_i \qquad \text{bzw.} \qquad \bigcup_{i \in \{1,\ldots,k\}} M_i$$

um die Vereinigung der Mengen $M_1, \ldots, M_k$ zu bezeichnen (d.h. die Menge $M_1 \cup \cdots \cup M_k$).
Wir schreiben

$$\bigcap_{i=1}^{k} M_i \qquad \text{bzw.} \qquad \bigcap_{i \in \{1,\ldots,k\}} M_i$$

um den Durchschnitt der Mengen $M_1, \ldots, M_k$ zu bezeichnen (d.h. die Menge $M_1 \cap \cdots \cap M_k$).

(c) Ist K eine Menge, deren Elemente Teilmengen einer Menge U sind (d.h.: $K \subseteq \mathcal{P}(U)$), so ist

$$\bigcup_{M \in K} M \quad := \quad \{x \in U \ : \ \text{ex. } M \in K \text{ s.d. } x \in M\}$$

die Vereinigung aller Mengen $M \in K$ (d.h. die Menge aller Elemente x, die in mindestens einer Menge $M \in K$ liegen).
Analog ist

$$\bigcap_{M \in K} M \quad := \quad \{x \in U \ : \ \text{f.a. } M \in K \text{ gilt } x \in M\}$$

der Durchschnitt aller Mengen $M \in K$ (d.h. die Menge aller Elemente x, die in jeder Menge $M \in K$ liegen).

2.3.2. Relationen

Relationen sind Teilmengen von kartesischen Produkten. Präzise:

Definition 2.42.

(a) Sei $k \in \mathbb{N}_{>0}$ und seien $M_1, \ldots, M_k$ Mengen. Eine **Relation** auf $M_1, \ldots, M_k$ ist eine Teilmenge von $M_1 \times \cdots \times M_k$. Die **Stelligkeit** einer solchen Relation ist k. Relation Stelligkeit

(b) Sei M eine Menge und sei $k \in \mathbb{N}$. Eine **k-stellige Relation über M** ist eine Teilmenge von M^k.

Beispiel 2.43. Um Datumsangaben im Format (Tag, Monat, Jahr) anzugeben, nutzen wir die Wertebereiche

$$\begin{aligned} \text{TagWerte} &:= \{1, 2, \ldots, 31\} \\ \text{MonatsWerte} &:= \{1, 2, \ldots, 12\} \\ \text{JahresWerte} &:= \mathbb{Z}. \end{aligned}$$

Die Menge „Gültig“ aller **gültigen** Daten ist dann eine **Teilmenge** von

$$\text{TagWerte} \times \text{MonatsWerte} \times \text{JahresWerte},$$

d.h. eine **Relation** auf TagWerte, MonatsWerte, JahresWerte, zu der beispielsweise das Tupel $(23, 6, 1912)$ gehört,[5] nicht aber das Tupel $(30, 2, 1912)$.

Notation 2.44.

- Ist R eine Relation von M nach N (für zwei Mengen M, N), so schreiben wir oft

  $$mRn \quad \text{statt} \quad (m, n) \in R.$$

  Beispiel:
  - $m \leqslant n$, für natürliche Zahlen m, n
  - $m \neq n$
- Ist R eine Relation auf $M_1, \ldots, M_k$, so schreiben wir manchmal

  $$R(m_1, \ldots, m_k) \quad \text{statt} \quad (m_1, \ldots, m_k) \in R.$$

  Das soll verdeutlichen, dass R eine „Eigenschaft“ ist, die ein Tupel aus $M_1 \times \cdots \times M_k$ haben kann – oder eben nicht haben kann.

  Im Datums-Beispiel gilt: Gültig$(23, 6, 1912)$, aber es gilt **nicht**: Gültig$(30, 2, 1912)$.

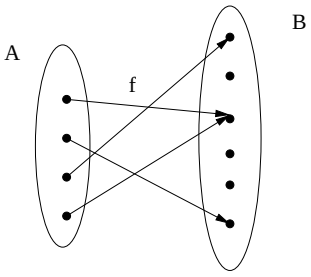
2.4. Funktionen

2.4.1. Totale und partielle Funktionen

Funktion
Abbildung

Definition 2.45. Seien A, B Mengen. Eine **Funktion** (oder **Abbildung**) von A nach B ist eine zwei-stellige Relation f auf A und B (d.h. $f \subseteq A \times B$) mit der Eigenschaft, dass für jedes $a \in A$ **genau ein** $b \in B$ mit $(a, b) \in f$ existiert.

Anschaulich:



[5] Der 23. Juni 1912 ist der Geburtstag von **Alan M. Turing**, einem der einflussreichsten Pioniere der Informatik.

Notation 2.46.

(a) Wir schreiben $f\colon A \to B$, um auszudrücken, dass f eine Funktion von A nach B ist.

(b) Ist $f\colon A \to B$ und ist $a \in A$, so bezeichnet $f(a)$ das (eindeutig bestimmte) $b \in B$ mit $(a, b) \in f$. Insbesondere schreiben wir meistens

$$f(a) = b \text{ bzw. } a \mapsto b$$

an Stelle von $(a, b) \in f$.

(c) Für $f\colon A \to B$ und $A' \subseteq A$ sei

$$f(A') \;:=\; \{f(a) : a \in A'\}.$$

(d) Die Menge aller Funktionen von A nach B bezeichnen wir mit $\mathrm{Abb}(A, B)$. $\mathrm{Abb}(A, B)$

(e) In manchen Büchern wird $\mathrm{Abb}(A, B)$ auch mit $A \to B$ oder mit B^A bezeichnet. B^A
Später, in Folgerung 2.58, werden wir sehen, dass

$$|\mathrm{Abb}(A, B)| \;=\; |B|^{|A|}$$

für endliche Mengen A, B gilt.

Definition 2.47.
Zwei Funktionen $f\colon A \to B$ und $g\colon A \to B$ sind **gleich** (kurz: $f = g$), falls f.a. $a \in A$ gilt: $f(a) = g(a)$.

Definition 2.48 (Definitionsbereich, Bildbereich, Bild, Urbild). Sei $f\colon A \to B$ eine Funktion.

(a) Der **Definitionsbereich** von f ist die Menge $\mathrm{Def}(f) := A$. Definitionsbereich $\mathrm{Def}(f)$

(b) Der **Bildbereich** (bzw. Wertebereich) von f ist die Menge B. Bildbereich

(c) Das **Bild** von f (genauer: das Bild von A unter f) ist die Menge $\mathrm{Bild}(f)$

$$\mathrm{Bild}(f) \;:=\; f(A) \;=\; \{f(a) : a \in A\}.$$

(d) Das **Urbild** $f^{-1}(B')$ einer Teilmenge $B' \subseteq B$ ist die Menge Urbild

$$f^{-1}(B') = \{\, a \in A \,:\, f(a) \in B' \,\}.$$

Definition 2.49 (Restriktionen).
Sei $f\colon A \to B$ eine Funktion und sei $A' \subseteq A$. Die **Restriktion** (oder **Einschränkung**) von f auf A' ist die Funktion Restriktion Einschränkung

$$f|_{A'}\colon A' \to B,$$

die folgendermaßen definiert ist: f.a. $a \in A'$ ist $f|_{A'}(a) := f(a)$.

Definition 2.50.
Eine **partielle Funktion** von einer Menge A in eine Menge B ist eine Funktion f mit $\mathrm{Def}(f) \subseteq A$ und $\mathrm{Bild}(f) \subseteq B$. partielle Funktion

Bemerkung 2.51.

(a) Im Gegensatz zu partiellen Funktionen nennt man Funktionen, wie wir sie in Definition 2.45 definiert haben, auch **totale Funktionen**.

totale Funktion

Sprechen wir von „Funktionen“, ohne sie explizit als „partiell“ zu bezeichnen, so meinen wir in dieser Vorlesung immer „totale“ Funktionen.

(b) Jede partielle Funktion von einer Menge A in eine Menge B lässt sich auch als totale Funktion von A nach $B \dot\cup \{\bot\}$ auffassen, wobei $\bot$ ein spezielles Zeichen ist, das für „undefiniert“ steht und nicht zur Menge B gehört.

2.4.2. Eigenschaften von Funktionen

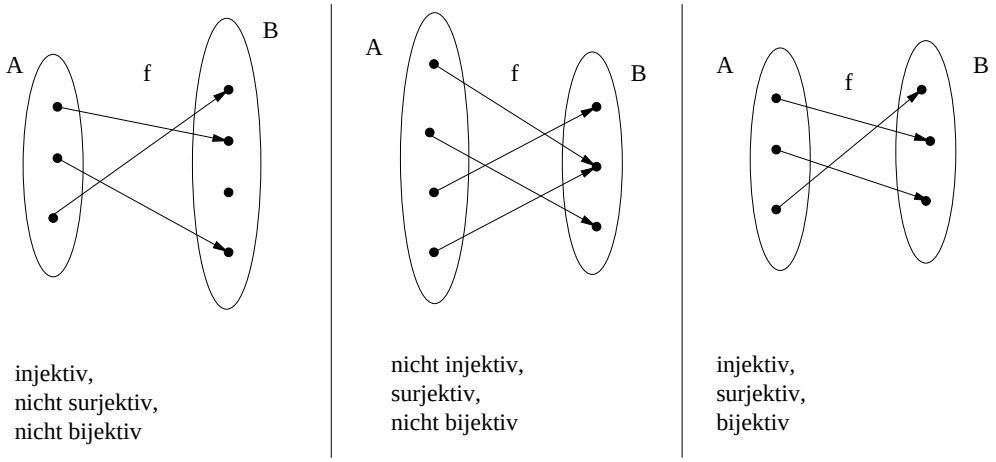
Definition 2.52. Sei $f\colon A \to B$.

injektiv (a) f heißt **injektiv**, falls es für jedes $b \in B$ höchstens ein $a \in A$ mit $f(a) = b$ gibt.

surjektiv (b) f heißt **surjektiv**, falls es für jedes $b \in B$ mindestens ein $a \in A$ mit $f(a) = b$ gibt.

bijektiv (c) f heißt **bijektiv**, falls es für jedes $b \in B$ genau ein $a \in A$ mit $f(a) = b$ gibt.

Anschaulich:



Beobachtung 2.53.

(a) Für jede Funktion $f\colon A \to B$ gilt:

$$f \text{ ist bijektiv} \iff f \text{ ist injektiv und surjektiv.}$$

Identitätsfunktion id_M

(b) Die **Identitätsfunktion** auf einer Menge M ist die Funktion

$$\mathrm{id}_M\colon M \to M$$

mit $\mathrm{id}_M(m) := m$ bzw. $m \mapsto m$ f.a. $m \in M$. Die Identitätsfunktion id_M ist eine bijektive Funktion.

2.4.3. Die Mächtigkeit bzw. Kardinalität einer Menge

Definition 2.54. (Die Mächtigkeit von Mengen).
Seien M und N Mengen.

(a) M heißt **endlich**, wenn M nur endlich viele Elemente enthält, d.h. wenn es eine Zahl $n \in \mathbb{N}$ gibt, so dass die Menge genau n Elemente enthält. Wir sagen, dass M die Mächtigkeit n (kurz: $|M| := n$) besitzt. endlich $|M|$

(b) Ist eine Menge M nicht endlich bezeichnen wir sie als **unendlich**. unendlich

(c) M und N heißen **gleichmächtig** : $\Longleftrightarrow$ Es gibt eine bijektive Funktion von A nach B. gleichmächtig

- M heißt **abzählbar unendlich**, wenn M und $\mathbb{N}$ gleichmächtig sind. (Eine bijektive Funktion $f : \mathbb{N} \to M$ „zählt die Elemente von M ab“.) abzählbar unendlich
- M heißt **überabzählbar**, wenn M weder endlich noch abzählbar unendlich ist. überabzählbar

In Python kann man die Mächtigkeit einer Menge M durch

```
len( M )
```

bestimmen.

Beispiel 2.55.

- $|\{2, 4, 6\}| = 3$
- $|\emptyset| = 0$
- $|\{\emptyset\}| = 1$
- $|\{2, 4, 6, 4\}| = 3$
- $|\{2, \{a, b\}\}| = 2$.

□ Ende von Beispiel 2.55

Wie viele Elemente besitzt das kartesische Produkt $M_1 \times \cdots \times M_k$?

Satz 2.56. (Die Mächtigkeit kartesischer Produkte).

(a) Seien M und N zwei endliche Mengen. Dann gilt:

$$|M \times N| \;=\; |M| \cdot |N|.$$

(b) Sei $k \in \mathbb{N}_{>0}$ und seien $M_1, \ldots, M_k$ endliche Mengen. Dann gilt:

$$|M_1 \times \cdots \times M_k| \;=\; \prod_{i=1}^{k} |M_i|.$$

(c) Sei $k \in \mathbb{N}$ und sei M eine endliche Menge. Dann gilt:

$$|M^k| \;=\; |M|^k.$$

Beweis:

(a) Es gilt:

$$M \times N \;=\; \{(m,n) : m \in M,\ n \in N\} \;=\; \bigcup_{m \in M} \{(m,n) : n \in N\} \;=\; \bigcup_{m \in M} \big(\{m\} \times N\big).$$

Außerdem gilt für alle $m, m' \in M$ mit $m \neq m'$, dass die Mengen $\{m\} \times N$ und $\{m'\} \times N$ disjunkt sind. Ferner gilt für beliebige disjunkte endliche Mengen A und B, dass $|A \cup B| = |A| + |B|$ ist. Insgesamt folgt daraus, dass

$$\begin{aligned} |M \times N| \;&=\; \Big|\bigcup_{m \in M} \big(\{m\} \times N\big)\Big| \;=\; \sum_{m \in M} |\{m\} \times N| \\ &=\; \sum_{m \in M} |N| \;=\; \underbrace{|N| + \cdots + |N|}_{|M|\text{-mal}} \;=\; |M| \cdot |N|. \end{aligned}$$

(b) Der Beweis verläuft analog. Für eine elegante Argumentation sollte man aber die vollständige Induktion anwenden, die wir erst in Kapitel 4.3 einführen.

(c)

$$|M^k| \;=\; \underbrace{|M \times \cdots \times M|}_{k\text{-mal}} \overset{((b))}{=} \prod_{i=1}^{k} |M| \;=\; \underbrace{|M| \cdot \cdots \cdot |M|}_{k\text{-mal}} \;=\; |M|^k.$$

□

Wie viele Abbildungen mit Definitionsbereich A und Bildbereich B gibt es? Wie viele Elemente besitzt die Potenzmenge $\mathcal{P}(M)$ einer Menge M? Wir geben eine Antwort in Folgerung 2.58. Zur Vorbereitung benötigen wir den folgenden Satz.

Satz 2.57.

(a) Für jede Menge M gibt es eine bijektive Funktion von $\mathcal{P}(M)$ nach $\mathrm{Abb}(M, \{0,1\})$.

(b) Sei B eine Menge, sei A eine endliche Menge und sei $k := |A|$. Dann gibt es eine bijektive Funktion von $\mathrm{Abb}(A, B)$ nach B^k.

Beweis:

charakteristische Funktion

(a) Repräsentiere jede Menge $A \in \mathcal{P}(M)$ (d.h. $A \subseteq M$) durch die so genannte **charakteristische Funktion** $\chi_A \colon M \to \{0,1\}$ mit

$$\chi_A(m) \;:=\; \begin{cases} 1, & \text{falls } m \in A \\ 0, & \text{sonst.} \end{cases} \qquad (*)$$

Sei nun $f \colon \mathcal{P}(M) \to \mathrm{Abb}(M, \{0,1\})$ definiert durch

$$f(A) \;:=\; \chi_A, \quad \text{für jedes } A \in \mathcal{P}(M). \qquad (**)$$

Behauptung: f ist bijektiv.

Wir zeigen dies in zwei Schritten (und nutzen Beobachtung 2.53(2.53)).

Schritt 1: f ist injektiv:

Seien $A, A' \in \mathcal{P}(M)$ mit $f(A) = f(A')$.

Ziel: Zeige, dass $A = A'$.

Wegen $f(A) = f(A')$ gilt gemäß (**), dass $\chi_A = \chi_{A'}$. D.h. f.a. $m \in M$ gilt $\chi_A(m) = \chi_{A'}(m)$. Gemäß (*) gilt daher f.a. $m \in M$, dass

$$m \in A \iff m \in A'.$$

Somit ist $A = A'$.

Schritt 2: f ist surjektiv:

Sei $h \in \mathrm{Abb}(M, \{0,1\})$, d.h. $h \colon M \to \{0,1\}$.

Ziel: Finde ein $A \in \mathcal{P}(M)$ mit $f(A) = h$.

Wir wählen

$$A \;:=\; \{m \in M \,:\, h(m) = 1\}.$$

Dann ist klar: $A \in \mathcal{P}(M)$. Gemäß (*) gilt $\chi_A = h$. Gemäß (**) ist daher $f(A) = h$.

(b) *Idee:* Sei $a_1, \ldots, a_k$ eine Liste aller Elemente in A. Repräsentiere jede Funktion $h \in \mathrm{Abb}(A, B)$ durch das k-Tupel $t_h := \big(h(a_1), \ldots, h(a_k)\big)$.

Rest: Übung.

□

Folgerung 2.58. Seien A, B, M endliche Mengen. Dann gilt:

(a) $|\mathrm{Abb}(A, B)| = |B|^{|A|}$.

(b) $|\mathcal{P}(M)| = 2^{|M|}$.

Beweis:

(a) Gemäß Satz 2.57((b)) und Beobachtung 2.53((c)) gilt für $k := |A|$, dass

$$|\mathrm{Abb}(A, B)| \;=\; |B^k|.$$

Laut Satz 2.56((c)) ist $|B^k| = |B|^k$. Somit $|\mathrm{Abb}(A, B)| = |B|^k = |B|^{|A|}$.

(b) Gemäß Satz 2.57((a)) und Beobachtung 2.53((c)) ist

$$|\mathcal{P}(M)| \;=\; |\mathrm{Abb}(M, \{0,1\})|.$$

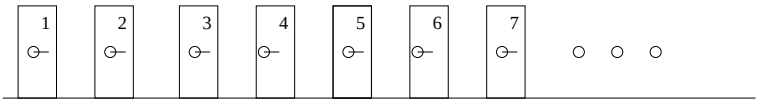
Gemäß ((a)) ist

$$|\mathrm{Abb}(M, \{0,1\})| \;=\; |\{0,1\}|^{|M|} \;=\; 2^{|M|}.$$

□

Bemerkung 2.59. (Hilberts Hotel). Der Mathematiker David Hilbert (1862–1943) hat sich unter Anderem auch mit den Grundlagen der Mathematik beschäftigt.

Hilberts Hotel hat unendlich viele Zimmer, die fortlaufend mit $1, 2, 3, \ldots$ (also mit allen Zahlen aus $\mathbb{N}_{>0}$) nummeriert sind. Ein neuer Gast kommt an. Obwohl alle Zimmer belegt sind, schafft es der Angestellte an der Rezeption, Platz zu schaffen.



Wie? – Er bittet alle Gäste, in das Zimmer mit der nächsthöheren Nummer umzuziehen und gibt dem neuen Gast das Zimmer mit der Nummer 1. Fügt man also zu einer unendlichen Menge ein Element hinzu, so erhält man keine „wirklich größere“ Menge. Zum Beispiel ist die Funktion $f : \mathbb{N} \to \mathbb{N}_{>0}$ mit $f(n) = n + 1$ eine Bijektion und die Mengen $\mathbb{N}, \mathbb{N}_{>0}$ sind gleichmächtig.

Es ist nicht schwer zu sehen, dass im vollbesetzten Hotel sogar unendlich viele neue Gäste, die mit den Zahlen 1, 2, 3, ... durchnummeriert sind, einquartiert werden können. Dazu muss einfach jeder der bisherigen Gäste in das Zimmer umziehen, dessen Nummer das Doppelte der bisherigen Zimmernummer ist. Danach sind alle „alten“ Gäste in den Zimmern mit geraden Zimmernummern untergebracht, und die neuen Gäste können in die Zimmer mit ungeraden Zimmernummern einziehen.

Und was passiert, wenn es unendlich viele vollbesetzte Hilbert-Hotels $H_1, H_2, H_3, \ldots$ gibt, die alle aus bautechnischen Gründen von jetzt auf nachher geschlossen werden müssen? Nur das leider auch vollbesetzte Hilbert-Hotel H_0 kann geöffnet bleiben. Ist es möglich, alle Gäste der geschlossenen Hotels im Hotel H_0 aufzunehmen?

Aber natürlich, bloß wie soll das funktionieren?

Man kann zum Beispiel zeigen, dass die Mengen $\mathbb{N}, \mathbb{Z}$ und $\mathbb{Q}$ gleichmächtig sind. Also sind $\mathbb{Z}$ und $\mathbb{Q}$ abzählbare Mengen. Genau diese Eigenschaft haben wir in Bemerkung 2.59 am Beispiel von Hilberts Hotel ausgenutzt.

Ebenfalls kann man zeigen, dass das Intervall $[0, 1]$, die Menge $\mathbb{R}$ der reellen Zahlen und die Potenzmenge $\mathcal{P}(\mathbb{N})$ gleichmächtig sind. In Satz 4.5 wird gezeigt, dass es keine surjektive Funktion $g : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ gibt. Insbesondere kann es keine bijektive Funktion geben und folglich sind die Mengen $[0, 1]$, $\mathbb{R}$ und $\mathcal{P}(\mathbb{N})$ überabzählbar groß: In der Welt der unendlich großen Mengen gibt es Mengen unterschiedlichster Mächtigkeiten.

2.5. Ein Beispiel zur Modellierung mit Wertebereichen

Beispiel 2.60 (Arbeitskreise der EU).

In der EU-Kommission sollen drei Arbeitskreise gebildet werden. Dazu entsendet jede der Nationen Deutschland, Frankreich, Österreich und Spanien drei Delegierte. Die Arbeitskreise sollen so gebildet werden, dass in jedem Arbeitskreis jede Nation vertreten ist und dass es unter Berücksichtigung der Fremdsprachenkenntnisse der Delegierten in jedem Arbeitskreis eine gemeinsame Sprache gibt, die alle beherrschen.

Aufgabe: Es soll nur die Situation modelliert werden – ein Lösungsverfahren wird hier zunächst nicht gesucht.

Formale Modellierung:

- Menge der **Nationen**:
$$\text{Nationen} \;:=\; \{\text{D}, \text{F}, \text{ö}, \text{S}\},$$
wobei D für Deutschland, F für Frankreich, Ö für Österreich und S für Spanien steht.
- Die **Delegierten** können wir repräsentieren als Paare, die aus einer Nation und einem Element aus $\{1, 2, 3\}$ bestehen, so dass beispielsweise die drei Delegierten aus Deutschland durch die Paare $(\text{D}, 1)$, $(\text{D}, 2)$ und $(\text{D}, 3)$ modelliert werden. Also:
$$\text{Delegierte} \;:=\; \text{Nationen} \times \text{DelegiertenNummer},$$
wobei $\text{DelegiertenNummer} := \{1, 2, 3\}$.
- Wir nutzen eine Funktion „spricht“, die jedem Delegierten die Menge von **Sprachen** zuordnet, die er beherrscht. Formal:
$$\text{spricht}\colon \text{Delegierte} \to \mathcal{P}(\text{Sprachen}),$$
wobei
$$\text{Sprachen} \;:=\; \{\text{deutsch}, \text{französisch}, \text{spanisch}, \text{englisch}, \text{italienisch}, \text{chinesisch}, \dots\}.$$
- Die drei Arbeitskreise bezeichnen wir mit AK1, AK2, AK3 und setzen
$$\text{Arbeitskreise} \;:=\; \{\text{AK1}, \text{AK2}, \text{AK3}\}.$$
- Eine konkrete Besetzung der drei Arbeitskreise repräsentieren wir durch eine Funktion
$$\text{AK-Besetzung}\colon \text{Arbeitskreise} \to \mathcal{P}(\text{Delegierte}),$$
die jedem der 3 Arbeitskreise die Menge der Delegierten zuordnet, die Mitglied des Arbeitskreises sind.
- Die Bedingung, dass jede Nation in jedem Arbeitskreis vertreten ist, lässt sich folgendermaßen formulieren:
$$\text{f.a.}\;\; a \in \text{Arbeitskreise} \;\; \text{ist} \;\; \text{Vertretene\_Nationen\_in\_}a \;=\; \text{Nationen},$$
wobei
$$\text{Vertretene\_Nationen\_in\_}a \;:=\; \Big\{\; n \in \text{Nationen} \;:\; \text{es ex. ein } i \in \text{DelegiertenNummer} \text{ s.d. } (n, i) \in \text{AK-Besetzung}(a) \;\Big\}.$$
- Die Bedingung, dass es für jeden Arbeitskreis eine Sprache gibt, die alle Mitglieder des Arbeitskreises beherrschen, lässt sich folgendermaßen formulieren:
$$\text{f.a.}\;\; a \in \text{Arbeitskreise} \;\; \text{ist} \;\; \text{Gemeinsame\_Sprachen\_in\_}a \;\neq\; \emptyset,$$
wobei
$$\text{Gemeinsame\_Sprachen\_in\_}a \;:=\; \Big\{\; \text{sp} \in \text{Sprachen} \;:\; \text{f.a. } d \in \text{AK-Besetzung}(a) \text{ ist sp} \in \text{spricht}(d) \;\Big\}.$$

□Ende Beispiel 2.60

2.6. Zusammenfassung und Ausblick

Die Umgangssprache ist unscharf, und wir arbeiten deshalb ausgiebig mit mathematischen Begriffen. Der Begriff der *Menge* ist ein Grundbaustein einer mathematisch ausgerichteten Sprache. Wir haben verschiedene wichtige Mengenoperationen, nämlich Durchschnitt, Vereinigung, Differenz, symmetrische Differenz und Komplementbildung eingeführt. Rechenregeln erklären, wie man mit den Operationen umzugehen hat. Zum Beispiel „sagen“ uns die De Morganschen Regeln $\overline{M \cap N} = \overline{M} \cup \overline{N}$ und $\overline{M \cup N} = \overline{M} \cap \overline{N}$, wie man mit dem Mengenkomplement arbeitet.

Mit den Mengenoperationen lassen sich komplexe Mengen mit Hilfe von Mengenausdrücken beschreiben. Um eine Inklusion oder eine Mengengleichheit für zwei Mengenausdrücke nachzuweisen, kann man Venn-Diagramme benutzen. Allerdings werden Venn-Diagramme unübersichtlich, wenn die Ausdrücke aus zu vielen Mengen bestehen, und ein Beweis ist notwendig. Um beispielsweise die Inklusion $M_1 \subseteq M_2$ zu zeigen, muss man für ein *beliebiges* Element $x \in M_1$ nachweisen, dass x auch ein Element von M_2 ist.

$$\text{Wenn f.a. } x \in M_1 \text{ auch stets } x \in M_2 \text{ gilt, dann folgt } M_1 \subseteq M_2.$$

Schließlich, um die Mengengleichheit $M_1 = M_2$ zu zeigen, genügt es, die beiden Inklusionen $M_1 \subseteq M_2$ und $M_2 \subseteq M_1$ zu verifizieren:

$$\text{Wenn } M_1 \subseteq M_2 \text{ und } M_2 \subseteq M_1, \text{ dann folgt } M_1 = M_2.$$

Nach Einführung des kartesischen Produkts können wir über Paare, Tupel, Folgen, Relationen und Funktionen reden: Diese Begriffe helfen uns in den folgenden Kapiteln, komplexe Sachverhalte kompakt zu beschreiben. Ein erstes Beispiel haben wir in der Modellierung mit Wertebereichen kennengelernt.

Wir haben die Größe kartesischer Produkte bestimmt und konnten dann auch die Größe der Potenzmenge einer endlichen Menge ableiten. In Satz 4.5 wird gezeigt, dass die Potenzmenge $\mathcal{P}(\mathbb{N})$ der natürlichen Zahlen „sehr viel“ größer als die Menge der natürlichen Zahlen ist.

Die Russelsche Antinomie zwingt uns, sorgfältig darüber nachzudenken, die Existenz welcher Mengen wir fordern „sollten“. Die Zermelo-Fraenkel-Mengenlehre ist eine erfolgreiche Axiomatisierung der Mengenlehre, allerdings kann ihre Widerspruchsfreiheit nicht ohne weitere Annahmen gezeigt werden. Dies ist ein erstes Anzeichen, dass die „Welt da draußen“ selbst für mathematische Argumente zu komplex ist. Der Gödelsche Unvollständigkeitssatz besagt genau dies, seine allerdings noch recht ungenaue Formulierung geben wir in Abschnitt 9.6.1 an.

2.7. Literaturhinweise zu Kapitel 2

Als vertiefende Lektüre seien die Kapitel 3, 6 und 7 in [21] empfohlen. Eine umfassende Einführung in die Mengenlehre gibt das Lehrbuch [5].

Quellennachweis: Teile der Abschnitte 2.2 und 2.4 orientieren sich an [9]. Das in Abschnitt 2.5 betrachtete Beispiel ist aus [14] entnommen. Die folgende Aufgabe 2.23 ist aus [14] entnommen.

2.8. Übungsaufgaben zu Kapitel 2

Aufgabe 2.1. Wir setzen eine Ameise auf einem Gummiband ab und lassen sie zum rechten Ende des Bandes laufen. Nach einiger Zeit ziehen wir am Band mit gleichbleibender Geschwindigkeit. Wird die Ameise das rechte Ende des Bands erreichen, wenn auch sie ihre allerdings vielfach kleinere Geschwindigkeit beibehält?

Aufgabe 2.2. Der Richter verkündet dem Angeklagten: „Sie werden morgen entweder durch Erhängen oder durch Enthauptung hingerichtet. Heute dürfen Sie eine letzte Aussage über den Ablauf des morgigen Tages treffen. Ist diese Aussage wahr, werden Sie erhängt. Ist die Aussage falsch, werden Sie enthauptet."

Natürlich möchte der Angeklagte überleben. Was sagt er? ;-)

Aufgabe 2.3.

(a) Was ist von dieser Aussage zu halten?

„Ich weiß, dass ich nichts weiß."

(b) Ist die Aussage „Es gibt keine wahren Aussagen" wahr?

(c) Im „Russelschen Club" darf nur Mitglied sein, wer in keinem Club Mitglied ist. Wie viele Mitglieder hat der Russelsche Club?

Aufgabe 2.4.

(a) Beschreiben Sie die folgenden Mengen sprachlich, beispielsweise wie in Aufgabenteil (b).

(i) $\{3n : n \in \mathbb{N}\}$

(ii) $\{\sqrt{k} : k \in \mathbb{N}_{>0}\}$

(iii) $\{x \in \mathbb{N} : x + 7 = x + 11\}$

(iv) $\{m \cdot n \colon m, n \in \mathbb{N}, m, n \geqslant 2\}$

(v) $\{(M, N) \colon M \subseteq N, N \subseteq \mathbb{N}\}$

(vi) $\{x^3 \colon x \in \mathbb{Z}, x^2 > 2\}$

(b) Geben Sie die folgenden Mengen in intensionaler Form, also wie in Aufgabenteil (a) an.

(i) Die Menge aller ganzen Zahlen, die größer als -12 und kleiner als 12 sind.

(ii) Die Menge aller natürlichen Zahlen, die nicht durch 3 teilbar sind.

(iii) Die Menge aller Zahlen, die das Quadrat einer natürlichen Zahl sind.

(iv) Die Menge aller ganzen Zahlen die Lösung der Ungleichung $3x + 2 < 1$ sind.

(v) Die Menge aller Tupel (x, y) reeller Zahlen, so dass der Betrag der Differenz zwischen x und y höchstens 5 ist.

(vi) Die Menge aller Produkte zweier ungerader Zahlen natürlicher Zahlen $\geqslant 3$.

(vii) Die Menge aller Teilmengen der natürlichen Zahlen, die nur gerade Zahlen enthalten.

Aufgabe 2.5. Es sei $M := \{2, 5, 8\}$ und $N := \{3, 5, 7, 11\}$. Schreiben Sie die folgenden Mengen in extensionaler Form auf und geben Sie ihre Kardinalität an.

(a) $M \cup N$ (b) $M \setminus N$ (c) $\mathcal{P}(M)$

(d) $\mathcal{P}(\{\emptyset\})$ (e) $M \times \{a, b\}$ (f) $\{M\} \times \{a, b\}$

(g) $\{P : P \subseteq N \text{ und } |P| = 2\}$ (h) $N^2 \setminus \{(x, x) : x \in N\}$

Aufgabe 2.6. Gegeben sei das Universum $U := \mathbb{N}$ sowie die Menge $M := \{2, 8, 17, 37, 47, 97\}$, die Menge $N := \{1, 3, 5, 7, \dots\}$ der ungeraden natürlichen Zahlen und die Menge $P := \{1, 2, 4, 8, 16, \dots\} = \{2^i : i \in \mathbb{N}\}$ der Zweierpotenzen. Geben Sie jede der folgenden Mengen in extensionaler Form an.

a) $M^2 \cap U$

b) $(P \cap N) \times M$

c) $\bigcup_{x \in N \cap M} \{1, x\}$

d) $\mathcal{P}\big(\{2^3, 3^2\}^2 \setminus P^2\big)$

Aufgabe 2.7. Gegeben seien die Menge $M = \{3, 14, 15, 16, 25\}$, die Menge $U := \{2x + 1 : x \in \mathbb{N}\} = \{1, 3, 5, \dots\}$ aller ungeraden natürlichen Zahlen und die Menge $Q := \{x^2 : x \in \mathbb{N}_{>0}\} = \{1, 4, 9, 16, \dots\}$ aller positiven Quadratzahlen. Geben Sie jede der folgenden Mengen in extensionaler Form an.

a) $(Q \setminus U) \times \emptyset$

b) $\mathcal{P}(U \cap M \cap Q)$

c) $\bigcap_{i=1}^{3} M^i$

d) $\mathcal{P}\big(\{2^3, 3^2\}^2 \setminus Q^2\big)$

e) $\big(M \times \{25\}\big) \cup \big(\{25\} \times M\big)$

f) $\mathcal{P}(\mathcal{P}(\mathcal{P}(\emptyset)))$

g) $\big\{x \in \mathcal{P}(M \cap U) : |x| = 2\big\}$

h) $\big(Q \oplus (M \oplus Q)\big)$

i) $\bigcup_{i=1}^{2} \big\{(2i-1, 3i+1)\big\} \setminus U^2$

Aufgabe 2.8.

a) Berechnen Sie den Wert der folgenden Ausdrücke

(i) $\sum_{i=1}^{5} 2i$

(ii) $\prod_{j=4}^{6} (3j + 1)$

(iii) $\sum_{k=2}^{4} \prod_{l=1}^{3} kl$

b) Es seien die Mengen $A_1 := \{1, 2, 3\}$, $A_2 := \{2, 3, 4\}$, $A_3 := \{3, 4, 5\}$ und $A_4 := \{4, 5, 6\}$ gegeben. Beschreiben Sie die folgenden Mengen in extensionaler Form.

(i) $\bigcup_{i=1}^{3} A_i$

(ii) $\bigcap_{j=1}^{4} A_j$

(iii) $\bigcup_{k=2}^{4} (A_k \setminus A_{k-1})$

Aufgabe 2.9. Geben Sie an, welche der folgenden Aussagen richtig und welche falsch sind.

a) $\{1, 3\} \in \{1, 3, \{1, 2\}\}$

b) $\{1, \{\emptyset\}\} \subseteq \{1, 3, \{\emptyset\}\}$

c) $\emptyset \subseteq \{1, 3, \{\emptyset\}\}$

d) $\{\emptyset\} \subseteq \{1, 3, \{\emptyset\}\}$

e) $\{\emptyset\} \in \{1, 3, \{\emptyset\}\}$

f) $\emptyset \supseteq \emptyset$

g) $\{\emptyset, \{2, 3\}\} \subseteq \mathcal{P}(\{1, 2, 3\})$

h) $\{1, \{1, 3\}\} \setminus \{1, 3\} = \{1\}$

i) $\{1, 1, 3, 4\} \setminus \{1, 3\} = \{1, 4\}$

j) $\{\emptyset, 5\} \setminus \emptyset = \{5, \emptyset\}$

k) $\{1, 2\} \oplus \{2, 3\} = \{3, 5\}$

l) $\emptyset \subsetneq \{\{1, 2\}\} \cap \{2, 1\}$

Aufgabe 2.10.

a) Für jede der folgenden Behauptungen beweisen Sie, dass die Behauptung für alle Mengen M, N, P gilt, oder widerlegen Sie die Behauptung, indem Sie Mengen M, N, P angeben und zeigen, dass die Behauptung für diese Mengen nicht gilt:

i) Falls $M \subseteq N$ und $N \subsetneq P$, dann $M \subsetneq P$.

ii) Falls $M \subseteq N$ und $N \not\subseteq P$, dann $M \not\subseteq P$.

iii) Falls $M \cup N \subseteq P$, dann $M \subseteq P$ und $N \subseteq P$.

iv) Falls $M \cap N \subseteq P$, dann $M \subseteq P$ oder $N \subseteq P$.

v) Falls $M \in N$ und $N \in P$, dann $M \in P$.

b) Gegeben sei ein Universum U sowie Mengen $A, B \subseteq U$. Begründen Sie jeweils kurz Ihre Antwort, z.B. mit Hilfe eines Venn-Diagramms, einer Definition oder eines Gegenbeispiels.

a) Folgt aus $A \times B = \emptyset$, dass $A = B = \emptyset$ gilt?

b) Folgt aus $A = \emptyset$ oder $B = \emptyset$, dass $A \times B = \emptyset$ gilt?

c) Gilt $\mathcal{P}(A \times B) = \mathcal{P}(A) \times \mathcal{P}(B)$?

d) Gilt $\overline{A \oplus B} = (A \oplus B) \oplus U$?

Aufgabe 2.11.

(a) Bestimmen Sie mit Hilfe von Venn-Diagrammen, welche der folgenden Behauptungen für alle Mengen M, N, P gilt, und welche nicht für alle Mengen M, N, P gilt:

i) $(M \cap N) \setminus P = (M \setminus P) \cup (N \setminus P)$

ii) $(M \cap N) \cup P = (M \cup P) \cap (N \cup P)$

iii) $(M \cup N) \setminus (M \cap N) = (M \setminus N) \cap (N \setminus M)$

iv) $M \cap (N \cup P) = (M \cap N) \cup (M \cap P)$

v) $\overline{M \cap N} = \overline{M} \cup \overline{N}$

vi) $M \cap N = M \setminus (M \setminus N)$

vii) $(M \cup N) \setminus M = M \cup (N \setminus M)$

viii) $(M \cap N) \cup M = M \cap (N \cup M)$

(b) Beweisen Sie Ihre Antworten aus Teil (a).

Aufgabe 2.12. Seien A, B und C Mengen. Zeigen Sie: Falls $(A \setminus B) \subseteq C$, so ist $(A \setminus C) \subseteq B$.

Aufgabe 2.13. Die Geschwister Konnie, Katie und Peter haben sehr unterschiedliche Hobbys. Während sich Konnie leidenschaftlich für Mathematik interessiert und Katie am liebsten die ganze Welt programmieren würde, ist ihr jüngerer Bruder Peter, den sie auch „der kleene[6] Piet“ nennen, ein begnadeter Astronom. Am liebsten zählt der kleene Piet Himmelskörper wie Sterne, Planeten, Monde oder Kometen und erstellt für diesen Zweck lange Listen. Mit der Zeit hat er sich eine ausgeklügelte Notation ausgedacht. Jeder Himmelskörpertyp erhält ein spezielles Symbol:

- `g` für Galaxie,
- `m` für Mond,
- `s` für Stern und
- `k` für Komet,
- `p` für Planet,
- `N` für Supernova.

Wenn er lange wach bleiben darf, sitzt er im Garten, zählt jedes leuchtende Objekt am Himmel und notiert seine Beobachtungen in seinem DisMond-Logbuch. Beispielsweise steht die Zeichenkette `mpNggssssm` dafür, dass der kleene Piet zuerst einen Mond, dann einen Planeten, eine Supernova, zwei Galaxien, fünf Sterne und letztlich einen weiteren Mond entdeckt hat. In

[6] plattdeutsch für „klein“

manchen Nächten zählt der kleene Piet fast ausschließlich Sterne und hat dementsprechend lange Listen der Form `ssss...sss`. Als Katie die Vorliebe ihres Bruders für Wiederholungen und Zeichenketten bemerkt, beginnt sie sogleich mit der Planung des computergestützten DisMond-Logbuchs *cgdl*. Konnie schaltet sich ebenfalls ein und schlägt vor, dass man zur einfacheren Eingabe und Verwaltung der langen Zeichenketten mathematische Operatoren verwenden könne. Zusammen einigen sich die drei Geschwister, dass in der ersten Version von cgdl zwei Operatoren zur Verfügung stehen sollen:

Definition: Der KonKat-Operator $\oplus$, benannt nach Konnie und Katie, bildet zwei Zeichenketten u und v auf uv ab:

$$(u \oplus v) := uv$$

Definition: Der kleene Stern $*$, benannt nach dem kleenen Piet und seiner Vorliebe für Sterne, bildet eine Zeichenkette u und eine positive natürliche Zahl k auf die k-fache Wiederholung von u ab:

$$(u * k) := \underbrace{uu \ldots u}_{k\text{-mal}}$$

So kann in cgdl beispielsweise die Zeichenkette `sNsNsNsNsNsNm` kurz als

$$((\texttt{sN} * 6) \oplus \texttt{m}) = (\texttt{sNsNsNsNsNsN} \oplus \texttt{m}) = \texttt{sNsNsNsNsNsNm} \tag{2.1}$$

dargestellt werden. Wir nennen `sNsNsNsNsNsNm` die *Auswertung* von $((\texttt{sN} * 6) \oplus \texttt{m})$.

a) Werten Sie folgende Ausdrücke wie in Gleichung (2.1) aus. Geben Sie alle Zwischenschritte an.

i) $((\texttt{gms} * 3) \oplus \texttt{ppN})$ ii) $((\texttt{kp} * 3) * 2)$ iii) $((\texttt{m} * 1) \oplus (\texttt{N} * 2))$

b) Seien u und v Zeichenketten und $k, l \in \mathbb{N}_{>0}$. Sind die folgenden Aussagen für jede Wahl von u, v, k und l wahr? Begründen Sie jeweils Ihre Antwort. Für eine falsche Aussage genügt die Angabe eines Gegenbeispiels.

i) $((u * k) \oplus (u * l)) = (u * (k + l))$ ii) $((u * k) \oplus (v * k)) = ((u \oplus v) * k)$

c) Piet benutzt nun auch die Ziffern $0, 1, 2$ und 3 als Symbole für Himmelsobjekte. Werten Sie auch die folgenden Ausdrücke aus:

i) $((2 * 2) \oplus 2)$ ii) $((3 \oplus 1) * 4)$ iii) $(((0 * 4) \oplus 1) * 2)$

Aufgabe 2.14. Abgeordnete des Deutschen Bundestages bilden Ausschüsse, die sich mit einem bestimmten Thema befassen. Sei A die Menge der Abgeordneten, die im Aussschuss Arbeit/Soziales sind und F die Menge der Abgeordneten die sich im Ausschuss Finanzen befinden. Außerdem sei S die Menge der Abgeordneten, die im Sport-Ausschuss sind. Es sind folgende Informationen über die Anzahl der Abgeordneten in den verschiedenen Aussschüssen bekannt:

$$|A| = 17, \quad |F| = 18, \quad |S| = 15, \quad |A \cap F| = 8, \quad |A \cap S| = 7, \quad |F \cap S| = 9, \quad |A \cap F \cap S| = 5$$

a) Wie viele Abgeordnete sind in mindestens einem der Ausschüsse Mitglied?
D.h. berechnen Sie $|A \cup F \cup S|$.

b) Wie viele der Abgeordneten sind in genau zwei Ausschüssen?
D.h. berechnen Sie $|((A \cap F) \cup (A \cap S) \cup (F \cap S)) \setminus (A \cap F \cap S)|$.

c) Es soll ein Unterausschuss gebildet werden, dem alle Abgeordneten des Sport-Aussschusses angehören und zusätzlich alle Abgeodneten, die im Arbeit/Soziales- aber nicht im Finanz-Ausschuss sitzen. Wie viele Mitglieder hat dieser Unterausschuss?
D.h. berechnen Sie $|S \cup (A \setminus F)|$.

Hinweis: Überlegen Sie sich zunächst anhand von Venn-Diagrammen, wie man die Kardinalitäten der Mengen berechnen kann.

Aufgabe 2.15. Schon lange steht die *Föderative Allianz Für Informatik (FAFI)* im Verdacht, an dubiosen Geschäften beteiligt zu sein. Daher ermitteln Polizeibehörden aus *Dismodien*, *Progistan* und *Mathibia* gegen Funktionäre der FAFI. Jeder Behörde liegt je eine Liste mit den Namen aller Personen vor, die aufgrund entsprechender *Beweise* als *tatverdächtig* gelten.

Dismodien hat Beweise gegen 170 Personen gesammelt. In Progistan stehen insgesamt 130 Funktionäre unter dringendem Tatverdacht. Auf der Liste der Polizei Mathibia stehen 480 FAFI-Mitglieder. Außerdem liegen den drei Polizeibehörden die folgenden Zahlen vor:

1) 90 Personen sind in Dismodien, aber weder in Progistan noch in Mathibia tatverdächtig.
2) 60 Personen stehen sowohl auf der Liste Progistans als auch auf der Liste Mathibias.
3) 20 Funktionäre gelten in Dismodien und Progistan, aber nicht in Mathibia als tatverdächtig.
4) Insgesamt 10 FAFI-Vertreter stehen sogar auf allen drei Listen.

Um die Zahlen weiter zu konsolidieren, verwenden die drei Polizeibehörden die Mengen D, P und M. Beispielsweise kann Aussage 1) durch die Gleichung $|D \setminus (P \cup M)| = 90$ dargestellt werden.

a) Formulieren Sie die Aussagen 2), 3) und 4) mithilfe von Mengen.
b) Wie viele Funktionäre sind in genau zweien der drei Länder tatverdächtig?
c) Wie groß ist die Anzahl der tatverdächtigen FAFI-Funktionäre insgesamt?

Ein Venn-Diagramm könnte sich als hilfreich erweisen.

Aufgabe 2.16.

a) Geben Sie alle Relationen von $A := \{x, y\}$ nach $B := \{c, d\}$ an. Geben Sie für jede Relation an, ob sie eine Funktion von A nach B oder eine partielle Funktion von A nach B oder keines von beiden ist. Geben Sie außerdem für jede Funktion an, ob sie injektiv, surjektiv und/oder bijektiv ist.

b) Seien M und N beliebige endliche Mengen. Wie viele Relationen von M nach N gibt es?

c) Geben Sie für jede der folgenden Funktionen f an, ob die Funktion injektiv, surjektiv und/oder bijektiv ist. Geben Sie jeweils auch das Bild von f an.
   a) $f\colon \mathbb{Z} \to \mathbb{Z}$ mit $x \mapsto x - 4$
   b) $f\colon \mathbb{Z} \to \mathbb{Z}$ mit $x \mapsto 2 \cdot x$
   c) $f\colon \mathbb{Z} \to \{-1, 1\}$ mit $x \mapsto (-1)^x$
   d) $f\colon \mathbb{Z} \to \mathbb{N}$ mit $x \mapsto= x^4$
   e) $f\colon \mathbb{N} \to \mathbb{Z}$ mit $x \mapsto (-1)^x x^2$

f) $f\colon \mathbb{Z} \to \mathbb{N}_{>0}$ mit $f(x) := \begin{cases} 2|x|+1 & \text{für } x \in \mathbb{Z},\ x \geqslant 0 \\ 2|x| & \text{für } x \in \mathbb{Z},\ x < 0 \end{cases}$

d) Wie viele Möglichkeiten gibt es,

i) zwei Bälle B_1, B_2 so auf drei Körbe K_1, K_2, K_3 zu verteilen, dass jeder Ball in einem anderen Korb landet? D.h. wie viele injektive Funktionen von $\{B_1, B_2\}$ nach $\{K_1, K_2, K_3\}$ gibt es?

ii) drei Bälle B_1, B_2, B_3 so auf zwei Körbe K_1, K_2 zu verteilen, dass kein Korb leer bleibt? D.h. wie viele surjektive Funktionen von $\{B_1, B_2, B_3\}$ nach $\{K_1, K_2\}$ gibt es?

iii) drei Bälle B_1, B_2, B_3 so auf drei Körbe K_1, K_2, K_3 zu verteilen, dass mindestens ein Korb leer bleibt? D.h. wie viele nicht surjektive Funktionen von $\{B_1, B_2, B_3\}$ nach $\{K_1, K_2, K_3\}$ gibt es?

Aufgabe 2.17. Beweisen Sie Satz 2.57((b)), d.h.:

Sei B eine Menge, sei A eine endliche Menge und sei $k := |A|$. Zeigen Sie, dass es eine bijektive Funktion von $\mathrm{Abb}(A, B)$ nach B^k gibt.

Aufgabe 2.18.

a) Seien A, B und C endliche Mengen und sei $f : A \to B$ eine Funktion von A nach B und $g : B \to C$ eine Funktion von B nach C. Wir definieren die Funktion $h : A \to C$ als *Verkettung*, d.h. Hintereinanderausführung, von f und g als $h(x) := g(f(x))$ f.a. $x \in A$. Beweisen Sie die folgenden Aussagen:

i) Wenn f und g surjektiv sind, so ist auch h surjektiv.

ii) Wenn f und g injektiv sind, so ist auch h injektiv.

iii) Falls h bijektiv ist, dann ist g surjektiv.

iv) Falls h bijektiv ist, dann ist f injektiv.

b) Seien X und Y endliche Mengen und $f \subseteq X \times Y$ eine Relation von X nach Y. Wir definieren die Relation $\widetilde{f}$ als Relation von Y nach X wie folgt:

$$\text{Für alle } x \in X,\ y \in Y \text{ gilt:} \quad (y, x) \in \widetilde{f} :\Longleftrightarrow (x, y) \in f,$$

Beweisen Sie, dass die folgende Aussage korrekt ist: f ist genau dann eine bijektive Funktion, wenn $\widetilde{f}$ eine bijektive Funktion ist.[7]

Aufgabe 2.19.

a) Betrachten Sie folgende Funktionen:

i) $f_1 : \{a\}^* \to \mathbb{N}$ mit $f_1(w) := |w|$ für alle $w \in \{a\}^*$

ii) $f_2 : \{b, c\}^+ \to \mathbb{N}$ mit $f_2(w) := |w|$ für alle $w \in \{b, c\}^+$

iii) $f_3 : \mathbb{Z} \to \mathbb{N}$ mit $f_3(z) := \frac{1}{2}(|z| + z)$ für alle $z \in \mathbb{Z}$

iv) $f_4 : \mathcal{P}(\mathbb{Z}) \to \mathcal{P}(\mathbb{N})$ mit $f_4(x) := x \cap \mathbb{N}$ für alle $x \in \mathcal{P}(\mathbb{Z})$

[7] In diesem Falle wird $\widetilde{f}$ die *Umkehrfunktion* von f genannt und üblicherweise mit f^{-1} bezeichnet.

v) $f_5 : \mathcal{P}(\{1,2,4,8\}) \to \{0,1,\ldots,15\}$ mit $f_5(A) := \sum_{a \in A} a$ für alle $A \in \mathcal{P}(\{1,2,4,8\})$.

Geben Sie für jede der obigen Funktionen f_i an, ob sie injektiv, surjektiv und/oder bijektiv ist. Geben Sie jeweils auch Bild(f_i) an. Geben Sie für jede nicht-injektive Funktion f_i zwei Elemente $x, y \in \text{Def}(f_i)$ an, sodass $x \neq y$ und $f_i(x) = f_i(y)$ gilt. Geben Sie für jede nicht-surjektive Funktion f_i ein Element x aus dem Bildbereich an, sodass $x \notin \text{Bild}(f_i)$ gilt.

b) Wir betrachten erneut die Funktion f_2 aus Teil a). Geben Sie eine Menge P an, sodass das Urbild $f_2^{-1}(P)$ genau 10 Elemente enthält.

Aufgabe 2.20. Es seien m endliche Mengen $M_1, \ldots, M_m$ für ein $m \in \mathbb{N}_{>0}$ gegeben. Beweisen Sie die folgende Aussage:

> Falls die Summe der Kardinalitäten der Mengen $M_1, \ldots, M_m$ größer als $n \in \mathbb{N}$ ist, so existiert eine Menge $M \in \{M_1, M_2, \ldots, M_m\}$ deren Kardinalität größer als $\frac{n}{m}$ ist.

Aufgabe 2.21. Beweisen Sie:

a) Falls M eine endliche Teilmenge einer unendlichen Menge U ist, so ist das Komplement von M in U unendlich.

b) Für beliebige Mengen A und B gilt: Falls $A \cup B$ unendlich ist, so ist A oder B unendlich.

Aufgabe 2.22. Das Reiseunternehmen ModTours transportiert Reisende mit Hilbert-Omnibussen. Ein Hilbert-Omnibus hat unendlich viele Sitzplätze, die fortlaufend mit allen Zahlen aus $\mathbb{N}_{>0}$ nummeriert sind. ModTours hat unendlich viele dieser Omnibusse, die ebenfalls fortlaufend mit allen Zahlen aus $\mathbb{N}_{>0}$ nummeriert sind.

a) Aus betrieblichen Gründen muss ModTours die Reisenden aus den drei Hilbert-Omnibussen Bus 1, Bus 2 und Bus 3 in einen neuen, anfänglich leeren, Hilbert-Omnibus zusammenlegen.
   - Geben Sie dazu eine Funktion f an, die einem Reisenden in Bus i auf Platz j den Sitzplatz $f(i,j)$ in dem neuen Bus zuweist.
   - Natürlich darf kein Platz im neuen Bus mehr als einem Reisenden zugewiesen werden. Welche Eigenschaft muss f haben, damit sich keine zwei Reisenden einen Platz teilen müssen?

b) Aus schwerwiegenden betrieblichen Gründen muss ModTours die Reisenden aus allen Hilbert-Omnibussen in einen neuen Hilbert-Omnibus zusammenlegen. Auch hier darf kein Platz doppelt vergeben werden. Geben Sie dazu eine Funktion f an, die einem Reisenden in Bus i auf Platz j den Sitzplatz $f(i,j)$ in dem neuen Bus zuweist.

   Hinweis: Die Sitzplätze im neuen Bus müssen nicht lückenlos vergeben werden.

Aufgabe 2.23. In den folgenden Teilaufgaben sollen einige Aspekte einer Variante des Spiels Monopoly mit Wertebereichen modelliert werden. Setzen Sie dabei nur die Menge $\mathbb{N}$ als vordefiniert voraus.

a) Auf dem Spielbrett gibt es 40 Felder, wobei 22 von diesen Feldern Straßen und 18 Felder Plätze sind. Die Straßen und Plätze sind von 1 bis 22 bzw. von 1 bis 18 durchnummeriert.

   Definieren Sie drei Mengen Strassen, Plätze und Felder, deren Elemente Straßen, Plätze bzw. Felder repräsentieren.

b) Auf ein Feld vom Typ 'Straße' können beliebig viele Häuser und Hotels platziert werden, deren Anordnung aber keine Rolle spielt.

   i) Definieren Sie eine Menge Bebauungszustände, von der jedes Element den Bebauungszustand einer einzelnen Straße (d.h. die Anzahl der Häuser und die Anzahl der Hotels) repräsentiert.

   ii) Welches Element von Bebauungszustände beschreibt, dass sich drei Häuser und vier Hotels auf der Straße befinden?

c) Der Zustand eines Spielers ist zu jedem Zeitpunkt bestimmt durch den Geldbetrag, der ihm zur Verfügung steht, der Menge der Straßen, die er besitzt, und dem Feld, auf dem er sich gerade befindet.

   i) Definieren Sie eine Menge Spielerzustände, von der jedes Element den Zustand eines Spielers repräsentiert.

   ii) Welches Element von Spielerzustände beschreibt, dass dem Spieler 1000 Euro zur Verfügung stehen, dass er die Straßen 4, 6 und 7 besitzt, und dass er gerade auf der 17. Straße steht?

d) Ein Spieler, der eine Straße betritt, die bereits einem anderen Spieler gehört, muss Miete an den Besitzer der Straße entrichten. Die Höhe der Miete hängt von der Straße und deren Bebauungszustand ab.

   Geben Sie Mengen A und B an, so dass der oben beschriebene Zusammenhang durch eine Funktion *miete*: $A \to B$ modelliert werden kann, d.h. *miete* soll die Miete für die Straße in Abhängigkeit von der Straße selbst und deren Bebauungszustand angeben.

Aufgabe 2.24. Derzeit sieht man in Frankfurt viele Menschen orientierungslos durch die Gegend laufen, während sie auf ihre Smartphone starren. Diese Menschen spielen *Pokémon Yalla*. In dem Spiel versucht jeder Spieler, von allen Pokémon-Arten ein Exemplar zu fangen und es durch Kämpfe zu trainieren.

Wir wollen einige Aspekte von Pokémon Yalla mit Mengen modellieren. Sie können die Menge $\mathbb{N}$ sowie die Menge **PA** aller im Spiel vorkommenden *Pokémon-Arten* (z. B. Pikachu, Nidoran, etc.) als gegeben voraussetzen.

a) Mithilfe des GPS ihres Smartphones können die Spieler 100 Kampf-Arenen und 4747 kleine Geschäfte, sogenannte Pokéstops, in Pokémon Yalla finden. Gegeben seien die Menge **A** aller Arenen und die Menge **PS** aller Pokéstops. Sowohl Arenen als auch Pokéstops werden durch ihre geographischen Koordinaten beschrieben. Definieren Sie die Menge **APS** aller Arenen, an deren Ort sich ebenfalls ein Pokéstop befindet.

b) Ein *Pokémon-Exemplar* ist spezifiziert durch seine Pokémon-Art sowie seinen *Level*. Der Level eines Pokémons kann die Werte 1 bis 99 annehmen.

   (i) Definieren Sie die Menge **PE** aller möglichen Pokémon-Exemplare.

   (ii) Welches Element von **PE** steht für ein Pikachu mit Level 37?

c) Jede Pokémon-Art besitzt außerdem *mindestens* einen *Typen*. Es gibt beispielsweise die Typen *Elektro*, *Gift* und *Normal*. Gegeben sei die Menge **T** aller im Spiel vorkommenden Typen und die Relation $\mathbf{R} \subseteq \mathbf{PA} \times \mathbf{T}$, die jeder Pokémon-Art ihre Typen zuordnet.

   (i) Welches Element von **R** besagt, dass der Art Pikachu der Typ Elektro zugeordnet ist?

   (ii) Definieren Sie die Menge **PA-Elektro** aller Pokémon-Arten des Typs Elektro.

   (iii) Definieren Sie die Menge **U** aller Pokémon-Arten, die *genau einen* Typ haben.

d) Der *Spielerzustand* eines Pokémon-Trainers wird bestimmt durch die Menge der mitgeführten Pokémon-Exemplare, die Anzahl mitgeführter Pokébälle und die Anzahl der Pokédollar im Pokémonnaie. Es können höchstens sechs Pokémon-Exemplare mitgenommen werden, aber kein Exemplar mehrfach; z. B. dürfen nicht zwei Pikachus desselben Levels mitgenommen werden.

   (i) Definieren Sie die Menge **SZ** aller möglichen Spielerzustände.

   (ii) Welches Element von **SZ** besagt, dass ein Trainer ein Nidoran mit Level 12, ein Pikachu mit Level 7 und ein Tauros mit Level 8, vier Pokébälle und 971 Pokédollar bei sich trägt?

e) Besiegt ein Trainer einen Gegner in einem Arena-Kampf, so erhält er als Belohnung eine bestimmte Anzahl an Pokédollar und Pokébällen. Die jeweilige Anzahl hängt vom eigenen und vom gegnerischen Spielerzustand sowie von der Arena ab, in welcher der Kampf stattfindet. Um diesen Zusammenhang durch eine Funktion $\textbf{belohnung} : \mathbf{D} \to \mathbf{B}$ zu modellieren, definieren Sie die Mengen **D** und **B**. Die Funktion **belohnung** gibt also an, wie viele Pokédollar und wie viele Pokébälle der Spieler gewinnt.

Aufgabe 2.25. Seit einigen Jahren sieht man in Frankfurt viele Leute konzentriert bunte Kügelchen auf ihren Smartphones herumwischen. Man spielt *Candy Crush*[8]. Verschiedenfarbige Bonbons liegen auf einem quadratischem Spielbrett. Ziel ist es, durch Vertauschen zweier benachbarter Bonbons drei oder mehr gleichfarbige Bonbons in einer Reihe zu erhalten. Dafür erhält man Punkte, die Bonbons lösen sich auf und lassen neue nachrücken. Wir wollen hier einige Aspekte des Spiels mithilfe von Mengen modellieren. Sie dürfen die Menge

$$\mathbf{F} = \{1, 2, \ldots, 9\} \times \{1, 2, \ldots, 9\}$$

aller Felder des Spielbretts als gegeben voraussetzen. Dabei bezeichnet $(i, j) \in \mathbf{F}$ das Feld in Zeile i und Spalte j.

a) Welches Element aus **F** bezeichnet das zweitunterste Feld ganz links?

b) Jedes Feld enthält genau ein *Candy*, dabei handelt es sich entweder um ein *Bonbon* oder eine *Spezialsüßigkeit* (z. B. Farbbombe, Kokoskonfekt, etc.). Jedes Bonbon wird charakterisiert durch seine *Farbe* (gelb, rot, blau, grün, orange, lila) und seine *Art* (normal, gestreift oder eingewickelt). Sei **Farb** die Menge aller Farben, **Ar** die Menge aller Arten und **Spez** die Menge aller Spezialsüßigkeiten. Definieren Sie die Menge **Bon** aller Bonbons sowie die Menge **Can** aller Candys.

c) *Zusätzlich* zu einem Candy kann sich auf einem Feld auch Gelee befinden. (Gelee zählt nicht als Candy!) Eine Funktion $\textbf{hier\_liegt} : \mathbf{X} \to \mathbf{Y}$ gebe für jedes Feld an, welches Candy sich darauf befindet und ob das Feld Gelee enthält.

   i) Geben Sie für diese Funktion geeignete Mengen **X** und **Y** an.

   ii) Welcher Funktionswert drückt aus, dass sich auf dem Feld $(2, 2)$ ein gelbes normales Bonbon, aber kein Gelee befindet?

d) In einem *Spielzug* dürfen die Candys zweier horizontal oder vertikal benachbarter Felder vertauscht werden. Wir charakterisieren einen solchen Zug durch die beiden Felder, deren Candys vertauscht werden.

[8] Hier können Sie selbst in das Abenteuer eintauchen: https://king.com/de/play/candycrush

i) Definieren Sie die Menge **Z** aller Spielzüge.

ii) Welches Element aus **Zug** gibt an, dass das Candy auf dem Feld ganz oben rechts mit seinem linken Nachbarn vertauscht wird?

e) Manche Candys haben einen bestimmten Effekt, wenn man sie aktiviert:

i) Ein *gestreiftes Bonbon* lässt alle Candys in derselben Zeile verschwinden.
Geben Sie die Menge $\mathbf{Zei}_i$ aller Felder in Zeile i an.

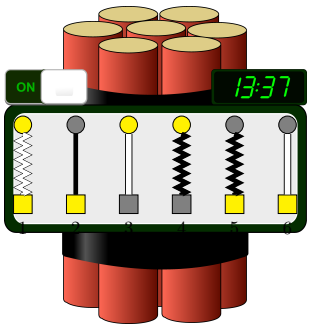
ii) Eine *Farbbombe* mit Farbe f lässt alle Bonbons auf dem Spielbrett mit derselben Farbe verschwinden.
Definieren Sie mithilfe der Funktion **hier_liegt** die Menge $\mathbf{F_{rot}}$ aller Felder auf dem Spielbrett, die ein rotes Bonbon enthalten.

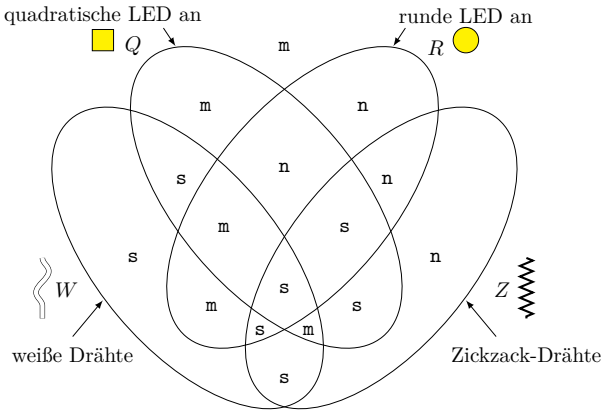
f) Im Laufe des Spiels lassen sich *Booster* freischalten (z. B. der Lollipop-Hammer oder der Kaugummi-Troll). Sei **Boost** die Menge aller im Spiel vorkommender Booster. Der *Spielzustand* wird charakterisiert durch die Anzahl der Punkte, die Menge der freigeschalteten Booster sowie das höchste absolvierte Level (1 bis 2855).

i) Definieren Sie die Menge **SZ** aller Spielzustände.

ii) Welches Element aus **SZ** gibt an, dass 110 110 Punkte erreicht wurden, ein Lollipop-Hammer freigeschaltet und das Level 20 absolviert wurde?

Aufgabe 2.26. In Frankfurt findet man auf dem Heimweg öfters tickende Zeitbomben. Diesmal geht die Bombe in 13 Minuten und 37 Sekunden[9] hoch. Die Bombe hat *sechs Drähte* sowie *sechs runde LEDs* über bzw. *sechs quadratische LEDs* unter den Drähten, die entweder an (bzw.) oder aus (bzw.) sind. Jeder Draht hat eine *Farbe (weiß oder schwarz)* und eine *Form (zickzack oder gerade)*.





Glücklicherweise besitzen Sie eine universelle Bombenentschärfungsanleitung, dargestellt durch ein Venn-Diagramm, das für jeden der sechs Drähte genau beschreibt, ob Sie ihn durchschneiden müssen bzw. nicht durchschneiden dürfen.

`s`: Draht durchschneiden

`n`: Draht nicht durchschneiden

[9] 36, 35, 34, 33, 32, 31, ... Los, Beeilung!

`m`: Draht durchschneiden, wenn Ihre Matrikelnummer gerade ist.

Rennen Sie nicht weg! Entschärfen Sie die Bombe! Geben Sie für jeden Draht Ihre Aktion an und markieren Sie den entsprechenden Bereich im Venn-Diagramm. Welche Drähte sind am Ende durchgeschnitten, welche nicht?

Aufgabe 2.27. *Relationale Datenbanken* sind in der Praxis weit verbreitet. Intuitiv gesprochen werden dabei Daten in Tabellen gespeichert, wobei jede Zeile einer Tabelle einem Datensatz entspricht. Mithilfe von *SQL-Befehlen* (Structured Query Language) können die Inhalte einer Datenbank abgefragt werden. Formal handelt es sich bei den Tabellen um Relationen und bei den Datensätzen um alle Tupel, die zur Relation gehören.

Definition. Für ein Tupel $x := (x_1, \dots, x_n)$ bezeichne x_i die i-te Komponente von x. Für eine Teilmenge $I \subseteq \{1, \dots, n\}$ entsteht das Tupel $(x_i : i \in I)$ aus x, indem alle Komponenten x_j mit $j \notin I$ gelöscht werden.

Seien R_1, R_2 und R_3 Relationen mit Stelligkeiten k, ℓ bzw. ℓ. Wir betrachten folgende *Operatoren*:

$$S_E(R_1) := \{x \in R_1 \ : \ E(x) \text{ ist wahr}\} \subseteq R_1 \qquad \text{„Selektion nach Eigenschaft } E\text{“}$$

$$R_1 \otimes R_2 := \{(x_1, \dots, x_k, x_{k+1}, \dots, x_{k+\ell}) \ : \ (x_1, \dots, x_k) \in R_1 \text{ und } (x_{k+1}, \dots, x_{k+\ell}) \in R_2\} \qquad \text{„Kartesisches Produkt“}[10]$$

$$R_2 \cup R_3 := \{x \ : \ x \in R_2 \text{ oder } x \in R_3\} \qquad \text{„Vereinigung“}$$

$$\pi_I(R_1) := \{(x_i : i \in I) \ : \ x \in R_1\} \qquad \text{„Projektion auf } I \subseteq \{1, \dots, k\}\text{“}$$

a) Die Relationen *User*, *Follower* und *Messages* sind unten gegeben. Bestimmen Sie die Relationen, die durch die folgenden Ausdrücke gegeben sind, in extensionaler Notation.

i) $S_{x_2=\text{fakeblues}}(\text{Follower})$

ii) $\pi_{\{1,3\}}\big(S_{x_2=\text{20.10.2017}}(\text{Messages}) \cup S_{x_1=\text{admin}}(\text{Messages})\big)$

iii) $\pi_{\emptyset}(\text{User})$

iv) $\pi_{\{6\}}\big(S_{x_1=\text{schniddyGee und } x_1=x_4}(\text{User} \otimes \text{Messages})\big)$

b) Seien R_1 und R_2 beliebige 2-stellige Relationen. Wie können Sie den Schnitt

$$R_1 \cap R_2 := \{x : x \in R_1 \text{ und } x \in R_2\}$$

von R_1 und R_2 mithilfe von Selektion, kartesischem Produkt und Projektion ausdrücken? *Hinweis:* Wenden Sie geeignete Selektionen und Projektionen auf $R_1 \otimes R_2$ an.

Kommentar: In SQL werden die Operatoren S, $\otimes$, $\cup$ und π durch die Schlüsselwörter WHERE (Selektion), FROM (kartesisches Produkt), UNION (Vereinigung) und SELECT (Projektion) dargestellt.

Relation „User“

1: Username	2: Passwort	3: E-Mail-Adresse
admin	Ea4%!3x2*	admin@dismodder.com
realdonaldduck	12345	thedonald@duck.com
cybert	sfdakl23	cybert@jmail.com
helldog	geheim123	helldog@jmx.de
schniddyGee	plsplspls	schn@iddy.com
fakeblues	12345	thedonald@duck.com
stud2017	stud2017	stu@d2017.org

Relation „Follower“

1: Username	2: folgt_Username
cybert	schniddyGee
cybert	stud2017
helldog	cybert
helldog	realdonaldduck
stud2017	admin
admin	fakeblues
realdonaldduck	fakeblues

[10] Wir haben hier ein anderes Symbol für das kartesische Produkt $\otimes$ verwandt. Beachten Sie den formalen Unterschied: $R_1 \times R_2 = \{((x_1, \dots, x_k), (x_{k+1}, \dots, x_{k+\ell})) \ : \ (x_1, \dots, x_k) \in R_1 \text{ und } (x_{k+1}, \dots, x_{k+\ell}) \in R_2\} \neq R_1 \otimes R_2$.

Relation „Messages“

1: Username	2: Datum	3: Text
admin	01.01.1970	test test test
admin	01.01.1970	test2
realdonaldduck	04.03.2010	Wo ist meine ....
realdonaldduck	04.03.2010	.. Hose? #fakeblues
cybert	20.06.2013	Gravitationswellenreiten #urlaub
stud2017	20.10.2017	Wololoooo!
stud2017	21.10.2017	Regenschirmstand 3 ist sein Geld wirklich wert!
stud2017	22.10.2017	Wir brauchen Silos!
schniddyGee	17.10.2017	BITTE BITTE BITTE #übungsbetrieb
schniddyGee	17.10.2017	Plan 2018: 50% und mehr!

3. Aussagenlogik

3.1. Wozu „Logik“ im Informatik-Studium?

Logik (nach dem Altgriechischen „Logos“: „Vernunft“) ist „die Lehre des vernünftigen Schlussfolgerns“. Logik ist ein Teilgebiet der Disziplinen Philosophie, Mathematik, Informatik und Linguistik. Eine zentrale Frage, mit dem sich die Logik beschäftigt, ist: Logik

> Wie kann man Aussagen miteinander verknüpfen, und auf welche Weise kann man formal Schlüsse ziehen und Beweise durchführen?

In einem gewissen Sinn spielt die Logik in der Informatik eine ähnlich wichtige Rolle wie die Differential- und Integralrechnung in der Physik [19, 10]. Logik wird in der Informatik u.a. genutzt

- zur Repräsentation von statischem Wissen (z.B. im Bereich der künstlichen Intelligenz),
- zur Verifikation von
  - Schaltkreisen (*Ziel*: beweise, dass ein Schaltkreis bzw. Chip „richtig“ funktioniert),
  - Programmen (*Ziel*: beweise, dass ein Programm gewisse wünschenswerte Eigenschaften hat),
  - Protokollen (*Ziel*: beweise, dass die Kommunikation zwischen zwei „Agenten“, die nach einem gewissen „Protokoll“ abläuft, „sicher“ ist — etwa gegen Abhören oder Manipulation durch dritte; Anwendungsbeispiel: Internet-Banking).
- zur automatischen Generierung von Beweisen (sogenannte „Theorembeweiser“),
- als Grundlage für Datenbank-Anfragesprachen,
- als Grundlage der Theorie der Programmiersprachen (Logikprogrammierung u.a.).

Aussagenlogik

Aussagen im Sinne der Aussagenlogik sind sprachliche Gebilde, die entweder **wahr** oder **falsch** sind. Aussagen können mit **Junktoren** wie „nicht“, „und“, „oder“, „wenn ... dann“ etc. zu komplexeren Aussagen verknüpft werden. Die **Aussagenlogik** beschäftigt sich mit allgemeinen Prinzipien des korrekten Argumentierens und Schließens mit Aussagen und Kombinationen von Aussagen. Aussagen
Junktoren
Aussagenlogik

Beispiel 3.1 („Geburtstagsfeier“).
Fred möchte mit möglichst vielen seiner Freunde Anne, Bernd, Christine, Dirk und Eva seinen Geburtstag feiern. Er weiß, dass Eva nur dann kommt, wenn Christine und Dirk kommen. Weiterhin kommt Christine nur dann, wenn auch Anne kommt; und Dirk wird auf keinen Fall kommen, wenn Bernd und Eva beide zur Feier kommen. Anne wiederum wird nur dann kommen, wenn auch Bernd oder Christine dabei sind. Wenn allerdings Bernd und Anne beide zur Party kommen, dann wird Eva auf keinen Fall dabei sein.

Frage: Wie viele Freunde kommen bestenfalls?

Das Wissen, das im obigen Text wiedergegeben ist, lässt sich in „atomare Aussagen“ zerlegen, die mit Junktoren verknüpft werden können. Die „atomaren Aussagen“, um die sich der Text dreht, kürzen wir folgendermaßen ab:

$$\begin{array}{lcl} A & \widehat{=} & \text{Anne kommt zur Feier} \\ B & \widehat{=} & \text{Bernd kommt zur Feier} \\ C & \widehat{=} & \text{Christine kommt zur Feier} \\ D & \widehat{=} & \text{Dirk kommt zur Feier} \\ E & \widehat{=} & \text{Eva kommt zur Feier} \end{array}$$

Das im Text zusammengefasste „Wissen“ lässt sich wie folgt repräsentieren:

(Wenn E, dann (C und D))	Eva kommt nur dann, wenn Christine und Dirk kommen,
und (wenn C, dann A)	Christine kommt nur dann, wenn auch Anne kommt,
und (wenn (B und E), dann nicht D)	Dirk wird auf keinen Fall kommen, wenn Bernd und Eva beide kommen,
und (wenn A, dann (B oder C))	Anne kommt nur dann, wenn auch Bernd oder Christine dabei sind,
und (wenn (B und A), dann nicht E)	wenn Bernd und Anne beide kommen, dann wird Eva auf keinen Fall dabei sein.

Die Aussagenlogik liefert einen Formalismus, mit dessen Hilfe man solches „Wissen“ modellieren und Schlüsse daraus ziehen kann — insbesondere z.B. um die Frage, mit wie vielen (und welchen) Gästen Fred bei seiner Feier rechnen kann, zu beantworten. □Ende von Beispiel 3.1

Syntax
Semantik

Die **Syntax** legt fest, welche Zeichenketten Formeln der Aussagenlogik sind. Die **Semantik** legt fest, welche „Bedeutung“ einzelne Formeln haben.

Man beachte, dass dies analog zur „Syntax“ und „Semantik“ von JAVA-Programmen ist: Die Syntax legt fest, welche Zeichenketten JAVA-Programme sind, während die Semantik bestimmt, was das Programm tut. Wir beginnen mit einer Beschreibung der Syntax.

3.2. Die Syntax der Aussagenlogik

Wir legen zuerst fest, welche Variablen in Formeln der Aussagenlogik vorkommen dürfen.

Definition 3.2 (Aussagenvariablen und Alphabet der Aussagenlogik).

Aussagenvariable

(a) Eine **Aussagenvariable** (kurz: Variable) hat die Form V_i, für $i \in \mathbb{N}$. Die Menge aller Aussagenvariablen bezeichnen wir mit AVAR. D.h.:

$$\text{AVAR} := \{V_i : i \in \mathbb{N}\} = \{V_0, V_1, V_2, V_3, \ldots\}.$$

(b) Das **Alphabet** der Aussagenlogik ist

$$A_{\text{AL}} := \text{AVAR} \cup \{\mathbf{0}, \mathbf{1}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \oplus, (,)\}.$$

Definition 3.3 (aussagenlogische Formeln: Syntax).
Die Menge AL der aussagenlogischen Formeln (kurz: Formeln) ist die folgendermaßen rekursiv definierte Teilmenge von A^*_{AL}:

Basisregeln:

(B0) $\mathbf{0} \in \mathrm{AL}$.

(B1) $\mathbf{1} \in \mathrm{AL}$.

(BV) Für jede Variable $X \in \textsc{Avar}$ gilt: $X \in \mathrm{AL}$ (kurz: $\textsc{Avar} \subseteq \mathrm{AL}$).

Rekursive Regeln:

(R1) Ist $\varphi \in \mathrm{AL}$, so ist auch $\neg\varphi \in \mathrm{AL}$.

(R2) Ist $\varphi \in \mathrm{AL}$ und $\psi \in \mathrm{AL}$, so ist auch

- $(\varphi \wedge \psi) \in \mathrm{AL}$
- $(\varphi \vee \psi) \in \mathrm{AL}$
- $(\varphi \to \psi) \in \mathrm{AL}$
- $(\varphi \leftrightarrow \psi) \in \mathrm{AL}$.
- $(\varphi \oplus \psi) \in \mathrm{AL}$.

Anmerkung 3.4 (griechische Buchstaben).
In der Literatur werden Formeln einer Logik traditionell meistens mit griechischen Buchstaben bezeichnet. Hier eine Liste der gebräuchlichsten Buchstaben:

Buchstabe	φ	ψ	χ	θ bzw. ϑ	λ	μ	ν	τ	κ
Aussprache	phi	psi	chi	theta	lambda	mü	nü	tau	kappa

Buchstabe	σ	ρ	ξ	ζ	α	β	γ	δ	ω
Aussprache	sigma	rho	xi	zeta	alpha	beta	gamma	delta	omega

Buchstabe	ε	ι	π	Δ	Γ	Σ	Π	Φ	Ψ
Aussprache	epsilon	iota	pi	Delta	Gamma	Sigma	Pi	Phi	Psi

Beispiel 3.5.
Die folgenden Zeichenketten sind Formeln, d.h. gehören zur Menge AL:

- $\big(\neg V_0 \vee (V_5 \to V_1)\big)$
- $\neg\big((V_0 \wedge \mathbf{0}) \leftrightarrow \neg V_3\big)$

Die folgenden Zeichenketten sind keine Formeln, d.h. gehören nicht zur Menge AL:

- $V_1 \vee V_2 \wedge V_3$ (da die Klammern fehlen),
- $(\neg V_1)$ (da die Klammern „zu viel“ sind).

□Ende von Beispiel 3.5

Notation 3.6.

atomare Formel
Atom

(a) **0**, **1** und die Variablen (d.h. die Elemente aus AVAR) bezeichnen wir als **atomare Formeln** bzw. **Atome**.

Junktor

(b) Die Symbole $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\oplus$ heißen **Junktoren**.

(c) Sind φ und ψ Formeln (d.h. $\varphi \in \mathrm{AL}$ und $\psi \in \mathrm{AL}$), so heißt:

Konjunktion
- $(\varphi \wedge \psi)$ **Konjunktion** (bzw. Ver*und*ung) von φ und ψ,

Disjunktion
- $(\varphi \vee \psi)$ **Disjunktion** (bzw. Ver*oder*ung) von φ und ψ,

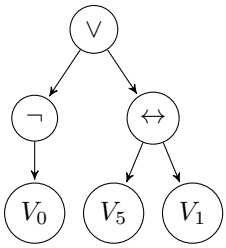
Implikation
- $\varphi \rightarrow \psi$ **Implikation** (bzw. Subjunktion) von φ und ψ,

Biimplikation
- $\varphi \leftrightarrow \psi$ **Biimplikation** (bzw. Bijunktion oder **Äquivalenz**) von φ und ψ,

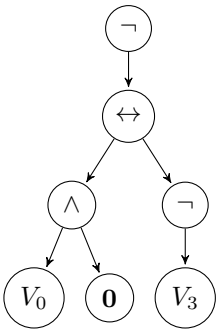
Äquivalenz
- $\varphi \oplus \psi$ **XOR** (bzw. exklusives ODER) von φ und ψ,

XOR
Negation
- $\neg\varphi$ **Negation** (bzw. Ver*nein*ung) von φ.

Bemerkung 3.7 (Syntaxbäume zur graphischen Darstellung von Formeln).
Die Struktur einer Formel lässt sich bequem durch einen **Syntaxbaum** (englisch: **parse tree**) darstellen. Zum Beispiel ist



der Syntaxbaum der Formel $\big(\neg V_0 \vee (V_5 \leftrightarrow V_1)\big)$ und



der Syntaxbaum der Formel $\neg\big((V_0 \wedge \mathbf{0}) \leftrightarrow \neg V_3\big)$.

Um umgangssprachlich formuliertes Wissen (vgl. Beispiel 3.1 „Geburtstagsfeier“) durch aussagenlogische Formeln zu repräsentieren, helfen folgende Konventionen:

Notation 3.8.

- Statt $V_0, V_1, V_2, \ldots$ bezeichnen wir Variablen oft auch mit $A, B, C, \ldots, X, Y, Z, \ldots$ oder mit Variablen wie $X', Y_1, \ldots$
- Die äußeren Klammern einer Formel lassen wir manchmal weg und schreiben zum Beispiel $(A \wedge B) \to C$ an Stelle des (formal korrekten) $\big((A \wedge B) \to C\big)$.

Beispiel 3.9.
Das in Beispiel 3.1 („Geburtstagsfeier“) aufgelistete Wissen kann folgendermaßen repräsentiert werden.

Atomare Aussagen:

- A: Anne kommt zur Feier,
- B: Bernd kommt zur Feier,
- C: Christine kommt zur Feier,
- D: Dirk kommt zur Feier,
- E: Eva kommt zur Feier.

Die Aussage des gesamten Textes aus Beispiel 3.1 wird durch folgende Formel repräsentiert:

$$\varphi := \big(E \to (C \wedge D)\big) \wedge (C \to A) \wedge \big((B \wedge E) \to \neg D\big) \wedge \big(A \to (B \vee C)\big) \wedge \big((B \wedge A) \to \neg E\big)$$

Beispiel 3.10.
Die Zeugenaussage „Das Fluchtauto war rot oder grün und hatte weder vorne noch hinten ein Nummernschild“ lässt sich durch die aussagenlogische Formel

$$\big((X_R \oplus X_G) \wedge (\neg X_V \wedge \neg X_H)\big)$$

repräsentieren, die die folgenden atomaren Aussagen nutzt:

- X_R: das Fluchtauto war rot,
- X_G: das Fluchtauto war grün,
- X_V: das Fluchtauto hatte vorne ein Nummernschild,
- X_H: das Fluchtauto hatte hinten ein Nummernschild.

Wir haben das exklusive Oder benutzt, weil wir davon ausgehen, dass der Zeuge kein rot-grünes Autos gesehen hat.

Aussagenlogische Formeln definieren Bedingungen in Python.

Bemerkung 3.11. (Python: Aussagenlogik) Python

Mit Python-Objekten vom Datentyp `bool` können Aussagenvariablen modelliert werden. Die Funktion `bool()` überführt ein Python-Objekte in ein Objekt vom Datentyp `bool`: Das Objekt x in

```
x = bool(y)
```

hat also den Datentyp `bool`, und es ist

```
bool ({}) = bool (()) = bool ([]) = bool (‘‘) = ···= bool (0) = False
```

Für „alle anderen“ Argumente nimmt bool den Wert `True` an. Die folgenden Python-Operatoren entsprechen Junktoren:

1. `not x` entspricht der Negation $\neg x$,
2. `x and y`, bzw. das bitweise-Und `x & y` entspricht der Konjunktion $x \wedge y$,
3. `x or y`, bzw. das bitweise-Oder `x | y` entspricht der Disjunktion $x \vee y$,
4. der „Vergleich“ `x <= y` entspricht der Implikation $x \to y$,
5. der Gleichheitstest `x == y` entspricht der Äquivalenz $x \leftrightarrow y$,
6. die Ungleichheit `x != y` entspricht dem Xor $x \oplus y$.

Mit Hilfe dieser Python-Operatoren und der runden Klammern können aussagenlogische Python-Formeln φ „gebaut“ werden. Allerdings wird Python sofort versuchen, φ zu evaluieren und einen Fehler melden, wenn dies nicht gelingt. Für die wirkliche Definition aussagenlogischer Formeln – also ohne unmittelbare Evaluierung – werden wir später (siehe Bemerkung 3.27) mit SymPy arbeiten.

3.3. Die Semantik der Aussagenlogik

Wir wissen nun, welche Zeichenketten (über dem Alphabet A_{AL}) **Formeln** genannt werden. Um festlegen zu können, welche Bedeutung (d.h. Semantik) solche Formeln haben, brauchen wir folgende Definition:

Definition 3.12.

Variablenmenge $\mathrm{Var}(\varphi)$

Die **Variablenmenge** einer aussagenlogischen Formel φ (kurz: $\mathrm{Var}(\varphi)$) ist die Menge aller Variablen $X \in \textsc{Avar}$, die in φ vorkommen.

Beispiele:

- $\mathrm{Var}\Big(\big(\neg V_0 \vee (V_5 \to V_1)\big)\Big) = \{V_0, V_1, V_5\}$,
- $\mathrm{Var}\Big(\neg\big((V_0 \wedge \mathbf{0}) \leftrightarrow \neg V_3\big)\Big) = \{V_0, V_3\}$,
- $\mathrm{Var}\Big(\big(\mathbf{0} \vee \mathbf{1}\big)\Big) = \emptyset$.

Definition 3.13.

(a) Eine **Belegung** (bzw. **Wahrheitsbelegung**) ist eine partielle Funktion von AVAR nach $\{0,1\}$. Dabei steht der Wert 1 für den Wert „wahr“ und 0 für den Wert „falsch“. (Manchmal verwendet man auch „wahr“, bzw. „w“ an Stelle der 1 und „falsch“ bzw. „f“ an Stelle der 0.) Belegung, Wahrheitsbelegung

(b) Eine Belegung $\mathcal{B}$ ist eine Belegung **für** die Formel φ (bzw. **passend zu** φ), wenn passend zu φ

$$\mathrm{Def}(\mathcal{B}) \supseteq \mathrm{Var}(\varphi).$$

Definition 3.14 (Semantik der Aussagenlogik).
Rekursiv über den Aufbau von AL definieren wir eine Funktion $[\![\cdot]\!]^{\cdot}$, die jeder Formel $\varphi \in \mathrm{AL}$ und jeder zu φ passenden Belegung $\mathcal{B}$ einen **Wahrheitswert** (kurz: **Wert**) $[\![\varphi]\!]^{\mathcal{B}} \in \{0,1\}$ zuordnet: Wahrheitswert

Rekursionsanfang:

- $[\![\mathbf{0}]\!]^{\mathcal{B}} := 0$.
- $[\![\mathbf{1}]\!]^{\mathcal{B}} := 1$.
- F.a. $X \in \mathrm{AVAR}$, für die $\mathcal{B}$ definiert ist, gilt: $[\![X]\!]^{\mathcal{B}} := \mathcal{B}(X)$.

Rekursionsschritt:

- Ist $\varphi \in \mathrm{AL}$, so ist $[\![\neg\varphi]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 0 \\ 0, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 1. \end{cases}$
- Ist $\varphi \in \mathrm{AL}$ und $\psi \in \mathrm{AL}$, so ist
  - $[\![(\varphi \wedge \psi)]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 1 \text{ und } [\![\psi]\!]^{\mathcal{B}} = 1 \\ 0, & \text{sonst} \end{cases}$
  - $[\![(\varphi \vee \psi)]\!]^{\mathcal{B}} := \begin{cases} 0, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 0 \text{ und } [\![\psi]\!]^{\mathcal{B}} = 0 \\ 1, & \text{sonst} \end{cases}$
  - $[\![(\varphi \to \psi)]\!]^{\mathcal{B}} := \begin{cases} 0, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 1 \text{ und } [\![\psi]\!]^{\mathcal{B}} = 0 \\ 1, & \text{sonst} \end{cases}$
  - $[\![(\varphi \leftrightarrow \psi)]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}} \\ 0, & \text{sonst} \end{cases}$
  - $[\![(\varphi \oplus \psi)]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} \neq [\![\psi]\!]^{\mathcal{B}} \\ 0, & \text{sonst.} \end{cases}$

Die **intuitive Bedeutung der Semantik** lässt sich wie folgt beschreiben:

- **Atome: 1** und **0** bedeuten einfach „wahr“ und „falsch“.

  Die Variablen $X \in \mathrm{AVAR}$ stehen für irgendwelche Aussagen. Uns interessiert hier nur, ob diese Aussagen „wahr“ oder „falsch“ sind — und dies wird durch eine Belegung $\mathcal{B}$ angegeben.

- **Negation:** $\neg\varphi$ bedeutet „nicht φ“.

  D.h.: $\neg\varphi$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ φ ist falsch (unter Belegung $\mathcal{B}$). Durch eine sogenannte **Wahrheitstafel** lässt sich dies wie folgt darstellen:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\neg\varphi]\!]^{\mathcal{B}}$
0	1
1	0

- **Konjunktion:** $(\varphi \wedge \psi)$ bedeutet „φ und ψ“.

  D.h.: $(\varphi\wedge\psi)$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ φ ist wahr und ψ ist wahr (unter Belegung $\mathcal{B}$). Zugehörige Wahrheitstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \wedge \psi)]\!]^{\mathcal{B}}$
0	0	0
0	1	0
1	0	0
1	1	1

- **Disjunktion, das inklusive „Oder“:** $(\varphi \vee \psi)$ bedeutet „φ oder ψ“.

  D.h.: $(\varphi \vee \psi)$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ φ ist wahr oder ψ ist wahr (unter Belegung $\mathcal{B}$). Zugehörige Wahrheitstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \vee \psi)]\!]^{\mathcal{B}}$
0	0	0
0	1	1
1	0	1
1	1	1

- **Implikation:** $(\varphi \to \psi)$ bedeutet „φ impliziert ψ“, d.h. „wenn φ, dann auch ψ“.

  D.h.: $(\varphi \to \psi)$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ wenn φ wahr ist, dann ist auch ψ wahr (unter Belegung $\mathcal{B}$). Zugehörige Wahrheitstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \to \psi)]\!]^{\mathcal{B}}$
0	0	1
0	1	1
1	0	0
1	1	1

- **Biimplikation:** $(\varphi \leftrightarrow \psi)$ bedeutet „φ genau dann, wenn ψ“.

  D.h.: $(\varphi \leftrightarrow \psi)$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ φ ist genau dann wahr, wenn ψ wahr ist (unter Belegung $\mathcal{B}$). Zugehörige Wahrheitstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \leftrightarrow \psi)]\!]^{\mathcal{B}}$
0	0	1
0	1	0
1	0	0
1	1	1

- **Xor, das exklusive „Oder“:** $(\varphi \oplus \psi)$ bedeutet „genau eine von φ oder ψ“.

  D.h.: $(\varphi \oplus \psi)$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ φ ist genau dann wahr, wenn ψ falsch ist (unter Belegung $\mathcal{B}$). Zugehörige Wahrheitstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \oplus \psi)]\!]^{\mathcal{B}}$
0	0	0
0	1	1
1	0	1
1	1	0

Bemerkung 3.15.

(a) Der Sprachgebrauch für „oder“ ist unscharf und meint manchmal das „inklusive Oder“ (wie in „Anna oder Xaver helfen beim Umzug“) und manchmal das „exklusive Oder“ (wie in „Ansgar oder Xenia fahren den Laster“).

(b) Man überzeuge sich, dass die Junktoren $\vee, \wedge, \leftrightarrow, \oplus$ kommutativ und assoziativ sind, d.h. für $\circ \in \{ \vee, \wedge, \leftrightarrow, \oplus \}$, für alle Belegungen $\mathcal{B}$ und für alle aussagenlogischen Formeln φ, χ und ψ gilt:

- **Kommutativität**: $\varphi \circ \psi$ ist genau dann wahr, wenn $\psi \circ \varphi$ wahr ist. kommutativ
- **Assoziativität**: $\varphi \circ (\chi \circ \psi)$ ist genau dann wahr, wenn $(\varphi \circ \chi) \circ \psi$ wahr ist. Der Wahrheitswert hängt also nicht von der Klammerung ab. assoziativ
- Wir nutzen die Assoziativität der Junktoren $\vee, \wedge, \leftrightarrow, \oplus$ und schreiben $\bigwedge_{i=1}^{n} \varphi_i$ bzw. $\varphi_1 \wedge \cdots \wedge \varphi_n$ an Stelle von $\Big(\big((\varphi_1 \wedge \varphi_2) \wedge \varphi_3 \big) \wedge \cdots \wedge \varphi_n \Big)$ (analog für „$\vee, \leftrightarrow, \oplus$“).

Man überzeuge sich aber auch, dass die Implikation $\rightarrow$ weder kommutativ noch assoziativ ist.

Beobachtung 3.16.
Sind $\mathcal{B}$ und $\mathcal{B}'$ zwei Belegungen für eine Formel φ, die auf $\mathrm{Var}(\varphi)$ übereinstimmen (d.h.: f.a. $X \in \mathrm{Var}(\varphi)$ ist $\mathcal{B}(X) = \mathcal{B}'(X)$), so ist $[\![\varphi]\!]^{\mathcal{B}} = [\![\varphi]\!]^{\mathcal{B}'}$.

In der Literatur wird diese Beobachtung oft unter dem Namen **Koinzidenzlemma** geführt. Koinzidenzlemma
Intuitiv ist die Beobachtung „offensichtlich richtig“, denn in der Definition von $[\![\varphi]\!]^{\mathcal{B}}$ werden ja nur diejenigen Variablen verwendet, die in φ vorkommen (also zu $\mathrm{Var}(\varphi)$ gehören). Aufgrund der Beobachtung des Koinzidenzlemmas werden wir uns im Folgenden, wenn wir Belegungen $\mathcal{B}$ für eine Formel φ betrachten, nur für diejenigen Werte $\mathcal{B}(X)$ interessieren, für die $X \in \mathrm{Var}(\varphi)$ ist.

3.3.1. Erfüllbarkeit, Allgemeingültigkeit und Widersprüchlichkeit

Definition 3.17. Sei φ eine Formel und $\mathcal{B}$ eine Belegung für φ.

(a) $\mathcal{B}$ **erfüllt** φ (bzw. $\mathcal{B}$ ist eine **erfüllende Belegung** für φ), falls $[\![\varphi]\!]^{\mathcal{B}} = 1$. erfüllen

(b) $\mathcal{B}$ **falsifiziert** φ (bzw. $\mathcal{B}$ ist eine **falsifizierende Belegung** für φ), falls $[\![\varphi]\!]^{\mathcal{B}} = 0$. falsifizieren

Beispiel 3.18.
Betrachte die Formel

$$\varphi \;:=\; \big(\neg V_0 \vee (V_5 \rightarrow V_1)\big).$$

Dann ist beispielsweise die Funktion $\mathcal{B}\colon \{V_0, V_1, V_5\} \to \{0,1\}$ mit $\mathcal{B}(V_0) := 1$, $\mathcal{B}(V_1) := 1$ und $\mathcal{B}(V_5) := 0$ eine Belegung für φ. Der Wahrheitswert von φ unter Belegung $\mathcal{B}$ ist der Wert

$$
\begin{aligned}
[\![\varphi]\!]^{\mathcal{B}} &\overset{\text{Def. 3.14}}{=} \begin{cases} 1, & \text{falls } [\![\neg V_0]\!]^{\mathcal{B}} = 1 \text{ oder } [\![(V_5 \to V_1)]\!]^{\mathcal{B}} = 1 \\ 0, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 3.14}}{=} \begin{cases} 1, & \text{falls } [\![V_0]\!]^{\mathcal{B}} = 0 \text{ oder } \big([\![V_5]\!]^{\mathcal{B}} = 0 \text{ oder } [\![V_1]\!]^{\mathcal{B}} = 1\big) \\ 0, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 3.14}}{=} \begin{cases} 1, & \text{falls } \mathcal{B}(V_0) = 0 \text{ oder } \mathcal{B}(V_5) = 0 \text{ oder } \mathcal{B}(V_1) = 1 \\ 0, & \text{sonst} \end{cases} \\
&= \quad 1 \qquad \text{(denn gemäß obiger Wahl von } \mathcal{B} \text{ gilt } \mathcal{B}(V_5) = 0).
\end{aligned}
$$

Die Belegung $\mathcal{B}$ erfüllt φ. Wie sind wir gerade vorgegangen, um $[\![\varphi]\!]^{\mathcal{B}}$ zu bestimmen? Wir haben φ von „innen nach außen“ ausgewertet und zuerst die Teilformeln $\neg V_0$ und $(V_5 \to V_1)$ ausgewertet.

Ist es denn immer klar, was die auszuwertenden Teilformeln sind? Ja natürlich, denn die Knoten des Syntaxbaums der Formel entsprechen den Teilformeln!

Jetzt kommen wir zu den für die Anwendung der Aussagenlogik wichtigsten Begriffe, nämlich zu Eigenschaften aussagenlogischer Formeln.

Definition 3.19. Sei φ eine aussagenlogische Formel.

erfüllbar

(a) φ heißt **erfüllbar**, wenn es (mindestens) eine erfüllende Belegung für φ gibt, d.h. wenn es (mindestens) eine zu φ passende Belegung $\mathcal{B}$ mit $[\![\varphi]\!]^{\mathcal{B}} = 1$ gibt.

falsifizierbar

(b) φ heißt **falsifizierbar**, wenn es (mindestens) eine falsifizierende Belegung für φ gibt, d.h. wenn es (mindestens) eine zu φ passende Belegung $\mathcal{B}$ mit $[\![\varphi]\!]^{\mathcal{B}} = 0$ gibt.

unerfüllbar
widersprüchlich

(c) φ heißt **unerfüllbar** (bzw. **widersprüchlich**), wenn **alle** zu φ passenden Belegungen φ falsifizieren.

allgemeingültig
Tautologie

(d) φ heißt **allgemeingültig** (bzw. **Tautologie**), wenn **alle** zu φ passenden Belegungen φ erfüllen.

Beispiel 3.20.

(a) Die Formel $\big((X \vee Y) \wedge (\neg X \vee Y)\big)$ ist

- **erfüllbar**, da z.B. die Belegung $\mathcal{B}$ mit $\mathcal{B}(X) = 0$ und $\mathcal{B}(Y) = 1$ die Formel erfüllt,
- **falsifizierbar** und deshalb **nicht allgemeingültig**, da z.B. die Belegung $\mathcal{B}$ mit $\mathcal{B}(X) = 0$ und $\mathcal{B}(Y) = 0$ die Formel falsifiziert.

Die Formel ist also weder unerfüllbar noch allgemeingültig.

(b) Die Formel $(X \wedge \neg X)$ ist **unerfüllbar**, da für jede zur Formel passenden Belegung $\mathcal{B}$ entweder $\mathcal{B}(X) = 1$ oder $\mathcal{B}(X) = 0$ gilt. Beachte:

Falls $\mathcal{B}(X) = 1$, so gilt:

$$\begin{aligned}[\![(X \wedge \neg X)]\!]^{\mathcal{B}} &= \begin{cases} 1, & \text{falls } \mathcal{B}(X) = 1 \text{ und } \mathcal{B}(X) = 0 \\ 0, & \text{sonst} \end{cases} \\ &= 0 \quad (\text{da } \mathcal{B}(X) = 1 \neq 0).\end{aligned}$$

Falls $\mathcal{B}(X) = 0$, so gilt:

$$\begin{aligned}[\![(X \wedge \neg X)]\!]^{\mathcal{B}} &= \begin{cases} 1, & \text{falls } \mathcal{B}(X) = 1 \text{ und } \mathcal{B}(X) = 0 \\ 0, & \text{sonst} \end{cases} \\ &= 0 \quad (\text{da } \mathcal{B}(X) = 0 \neq 1).\end{aligned}$$

(c) Die Formel $(X \vee \neg X)$ ist **allgemeingültig**, da für jede zur Formel passenden Belegung $\mathcal{B}$ entweder $\mathcal{B}(X) = 1$ oder $\mathcal{B}(X) = 0$ gilt. Somit gilt für alle zur Formel passenden Belegungen $\mathcal{B}$, dass $[\![(X \vee \neg X)]\!]^{\mathcal{B}} = 1$.

Beispiel 3.21. Das Schubfachprinzip: Wenn n Bälle in $n-1$ Schubfächer verteilt werden, dann erhält ein Schubfach mindestens zwei Bälle.

Wir formulieren das Schubfachprinzip als eine aussagenlogische Formel und verwenden dazu die Variablen $B_{i,j}$ für alle Bälle i $(1 \leqslant i \leqslant n)$ und Schubfächer j $(1 \leqslant j \leqslant n-1)$. Wenn $B_{i,j}$ den Wahrheitswert 1 erhält, dann denken wir uns den Ball i in das Schubfach j gelegt.

(1) Die Formel $\varphi_i := B_{i,1} \vee \cdots \vee B_{i,n-1}$ ist genau dann wahr, wenn Ball i in mindestens ein Schubfach gelegt wird.

(2) Die Formel $\varphi_{i,j}^k := \neg B_{i,k} \vee \neg B_{j,k}$ ist genau dann wahr, wenn Schubfach k nicht gleichzeitig die Bälle i und j erhält. Die Konjunktion

$$\psi_k := \bigwedge_{1 \leqslant i \neq j \leqslant n} \varphi_{i,j}^k$$

ist also genau dann wahr, wenn Schubfach k höchstens einen Ball erhält.

Die Formel

$$\alpha := \left(\bigwedge_{i=1}^{n} \varphi_i \right) \wedge \left(\bigwedge_{k=1}^{n-1} \psi_k \right)$$

ist unsere Formulierung des Schubfachprinzips als aussagenlogische Formel: α ist **unerfüllbar**. Beachte, dass α als eine Konjunktion von Disjunktionen eine besonders einfache Struktur besitzt.

3.3.2. Wahrheitstafeln

In Beispiel 3.1 haben wir gefragt, wie viele Freunde bestenfalls zur Geburtstagsfeier kommen und haben bisher diese Frage nicht beantworten können. In der „Sprache der Belegungen“ ausgedrückt suchen wir also eine Belegung $\mathcal{B}$ für φ, so dass

- φ von $\mathcal{B}$ erfüllt wird, d.h. $[\![\varphi]\!]^{\mathcal{B}} = 1$, und
- $\mathcal{B}$ möglichst viele Variablen erfüllt.

Auch haben wir uns bisher nicht überlegt, wie man denn systematisch feststellen kann, ob eine aussagenlogische Formel erfüllbar, allgemeingültig oder widerspruchsfrei ist.

Um solche Aufgaben zu lösen, versuchen wir einen Überblick über alle möglichen Belegungen zu erhalten: Wir benutzen für jede Formel φ eine Wahrheitstafel, um die Wahrheitswerte von φ unter allen möglichen Belegungen übersichtlich darstellen zu können. Für jede Belegung $\mathcal{B} : \mathrm{Var}(\varphi) \to \{0, 1\}$ hat die Wahrheitstafel eine Zeile, die die Werte $\mathcal{B}(X)$ f.a. $X \in \mathrm{Var}(\varphi)$ und den Wert $[\![\varphi]\!]^{\mathcal{B}}$ enthält. Um die Wahrheitstafel für φ auszufüllen, ist es bequem, auch Spalten für (alle oder einige) „Teilformeln“ von φ einzufügen.

Beispiel 3.22. (a) Wahrheitstafel für $\varphi := \big(\neg V_0 \vee (V_5 \to V_1)\big)$:

V_0	V_1	V_5	$\neg V_0$	$(V_5 \to V_1)$	φ
0	0	0	1	1	1
0	0	1	1	0	1
0	1	0	1	1	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	1	1

(b) Wahrheitstafel für $\varphi := \Big(X \wedge \big((\mathbf{1} \to \mathbf{0}) \to \mathbf{0}\big)\Big)$:

X	$\mathbf{1}$	$\mathbf{0}$	$(\mathbf{1} \to \mathbf{0})$	$((\mathbf{1} \to \mathbf{0}) \to \mathbf{0})$	φ
0	1	0	0	1	0
1	1	0	0	1	1

Die **erfüllenden** Belegungen für eine Formel φ entsprechen gerade denjenigen Zeilen der Wahrheitstafel für φ, in denen in der mit „φ“ beschrifteten Spalte der Wert 1 steht. Das liefert uns ein Werkzeug, um die in Beispiel 3.9 beschriebene Aufgabe zur „Geburtstagsfeier“ zu lösen.

Beispiel 3.23. Sei φ die Formel aus Beispiel 3.9. Die Frage „Wie viele (und welche) Freunde werden bestenfalls zur Party kommen?“ können wir lösen, in dem wir

(1) die Wahrheitstafel für φ ermitteln,

(2) alle Zeilen heraussuchen, in denen in der mit „φ“ beschrifteten Spalte der Wert 1 steht und

(3) aus diesen Zeilen all jene raussuchen, bei denen in den mit A, B, C, D, E beschrifteten Spalten möglichst viele Einsen stehen. Jede dieser Zeilen repräsentiert dann eine größtmögliche Konstellation von gleichzeitigen Partybesuchern.

Prinzipiell führt diese Vorgehensweise zum Ziel. Leider ist das Verfahren aber recht aufwendig, da die Wahrheitstafel, die man dabei aufstellen muss, sehr groß wird, wie man am Beispiel der Wahrheitstafel für die Formel φ (siehe Abbildung 3.1) sieht. **Erfüllende Belegungen** für φ werden in Abbildung 3.1 durch Zeilen repräsentiert, die grau unterlegt sind.

In der Wahrheitstafel sieht man, dass es **keine** erfüllende Belegung gibt, bei der in den mit A bis E beschrifteten Spalten insgesamt 5 Einsen stehen, und dass es genau **zwei** erfüllende

A	B	C	D	E	$E \to (C \wedge D)$	$C \to A$	$(B \wedge E) \to \neg D$	$A \to (B \vee C)$	$(B \wedge A) \to \neg E$	φ
0	0	0	0	0	1	1	1	1	1	1
0	0	0	0	1	0	1	1	1	1	0
0	0	0	1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	0
0	0	1	0	0	1	0	1	1	1	0
0	0	1	0	1	0	0	1	1	1	0
0	0	1	1	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1	1	1	0
0	1	0	0	0	1	1	1	1	1	1
0	1	0	0	1	0	1	1	1	1	0
0	1	0	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1	1	1	0
0	1	1	0	1	0	0	1	1	1	0
0	1	1	1	0	1	0	1	1	1	0
0	1	1	1	1	1	0	0	1	1	0
1	0	0	0	0	1	1	1	0	1	0
1	0	0	0	1	0	1	1	0	1	0
1	0	0	1	0	1	1	1	0	1	0
1	0	0	1	1	0	1	1	0	1	0
1	0	1	0	0	1	1	1	1	1	1
1	0	1	0	1	0	1	1	1	1	0
1	0	1	1	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	0	1	1	1	1	1	1
1	1	0	1	1	0	1	0	1	0	0
1	1	1	0	0	1	1	1	1	1	1
1	1	1	0	1	0	1	1	1	0	0
1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	0	0

Abbildung 3.1.: Wahrheitstafel für die Formel φ aus Beispiel 3.9

Belegungen gibt, bei denen in den mit A bis E beschrifteten Spalten insgesamt 4 Einsen stehen, nämlich die Belegungen $\mathcal{B}_1$ und $\mathcal{B}_2$ mit

$$\mathcal{B}_1(A) = \mathcal{B}_1(C) = \mathcal{B}_1(D) = \mathcal{B}_1(E) = 1 \quad \text{und} \quad \mathcal{B}_1(B) = 0$$

und

$$\mathcal{B}_2(A) = \mathcal{B}_2(B) = \mathcal{B}_2(C) = \mathcal{B}_2(D) = 1 \quad \text{und} \quad \mathcal{B}_2(E) = 0.$$

Die Antwort auf die Frage „Wie viele (und welche) Freunde werden bestenfalls zur Party kommen?“ lautet also: Bestenfalls werden 4 der 5 Freunde kommen, und dafür gibt es zwei Möglichkeiten, nämlich

(1) dass alle außer Bernd kommen, und

(2) dass alle außer Eva kommen.

□Ende Beispiel 3.23

Bemerkung 3.24. (Müssen wir immer die vollständige Wahrheitstafel ausrechnen?)

Wenn wir uns zuerst überzeugt hätten, dass alle fünf Freunde nicht zugleich zur Party kommen werden und dann die fünf Belegungen untersucht hätten, in denen genau ein Freund nicht kommt, hätten wir unser Ziel mit viel geringerem Aufwand erreicht. Auch ist es in unserem Beispiel nicht sinnvoll, stur die letzten sechs Spalten zu berechnen: Hat eine der Spalten 6-10 den Wert „falsch“ erhalten, ist φ falsch und der Wert der verbleibenden Spalten ist uninteressant.

Angesichts der Wahrheitstafel aus Abbildung 3.1 stellt sich die Frage, wie groß die Wahrheitstafel für eine gegebene Formel φ ist. Die Antwort darauf gibt der folgende Satz.

Satz 3.25. [1]
Sei φ eine aussagenlogische Formel und sei $n := |\mathrm{Var}(\varphi)|$ die Anzahl der in φ vorkommenden Variablen. Dann gibt es 2^n verschiedene zu φ passende Belegungen $\mathcal{B}$ mit $\mathrm{Def}(\mathcal{B}) = \mathrm{Var}(\varphi)$.

Beweis: Es gilt:

$$\begin{aligned} &\{\mathcal{B} : \mathcal{B} \text{ ist eine zu } \varphi \text{ passende Belegung mit } \mathrm{Def}(\mathcal{B}) = \mathrm{Var}(\varphi)\} \\ \overset{\text{Def. 3.13}}{=}\ &\{\mathcal{B} : \mathcal{B}\colon\ \mathrm{Var}(\varphi) \to \{0,1\} \text{ ist eine Funktion}\} \\ \overset{\text{Not. 2.46}}{=}\ &\mathrm{Abb}(\mathrm{Var}(\varphi), \{0,1\}). \end{aligned}$$

Wir wissen außerdem, dass

$$|\mathrm{Abb}(\mathrm{Var}(\varphi), \{0,1\})| \overset{\text{Fol. 2.58((a))}}{=} |\{0,1\}|^{|\mathrm{Var}(\varphi)|} \overset{n=|\mathrm{Var}(\varphi)|}{=} 2^n.$$

□

Satz 3.25 besagt, dass die Wahrheitstafel einer Formel mit n Variablen genau 2^n Zeilen hat. Warum ist dieses Ergebnis auch intuitiv klar? Sei a_n die Anzahl aller Belegungen für n Variable. Dann gilt $a_1 = 2$, denn eine einzige Variable V_1 besitzt die beiden Belegungen $B_0, B_1 : \{V_1\} \to \{0,1\}$ mit $\mathcal{B}_b(V_1) := b$ für $b \in \{0,1\}$. Kennen wir die Anzahl a_n der Belegungen für n Variable, dann ist $a_{n+1} = 2 \cdot a_n$, denn jede Belegung $\mathcal{B} : \{V_1, \ldots, V_n\} \to \{0,1\}$ der n Variablen erzeugt zwei „Kinder-Belegungen $\mathcal{B}_0, \mathcal{B}_1 : \{V_1, \ldots, V_n, V_{n+1}\} \to \{0,1\}$ mit $\mathcal{B}_b(V_i) := \mathcal{B}(V_i)$ für $1 \leqslant i \leqslant n$ und $\mathcal{B}_b(V_{n+1}) = b$. Aus den Rekursionsgleichungen $a_1 = 2$ und $a_{n+1} = 2 \cdot a_n$ folgt mit vollständiger Induktion, dass $a_n = 2^n$ gilt.

Wie die folgende Tabelle zeigt, ergibt das bereits bei relativ kleinen Werten von n schon riesige Wahrheitstafeln:

n (Anzahl Variablen)	2^n (Anzahl Zeilen der Wahrheitstafel)	
10	$2^{10} =$ 1.024	$\approx 10^3$
20	$2^{20} =$ 1.048.576	$\approx 10^6$
30	$2^{30} =$ 1.073.741.824	$\approx 10^9$
40	$2^{40} =$ 1.099.511.627.776	$\approx 10^{12}$
50	$2^{50} =$ 1.125.899.906.842.624	$\approx 10^{15}$
60	$2^{60} =$ 1.152.921.504.606.846.976	$\approx 10^{18}$

[1] Wenn Sie eine Wahrheitstafel mit 7 Zeilen als Lösung einer Übungsaufgabe einreichen, werden Sie wohl einen Fehler gemacht haben!

Zum Vergleich: Das Alter des Universums wird auf $13,8$ Milliarden Jahre (das sind ungefähr 10^{18} Sekunden) geschätzt.

Es ist deshalb sinnvoll, nur wirklich benötigte Teile der Wahrheitstafel zu bestimmen (vgl. Bemerkung 3.24), bzw. mit partiellen Belegungen zu arbeiten.

Beobachtung 3.26. Für jede aussagenlogische Formel φ gilt:

(a)	φ ist erfüllbar	$\iff$	in der Wahrheitstafel für φ steht in der mit „φ“ beschrifteten Spalte mindestens eine 1.
(b)	φ ist unerfüllbar	$\iff$	in der Wahrheitstafel für φ stehen in der mit „φ“ beschrifteten Spalte nur Nullen.
(c)	φ ist allgemeingültig	$\iff$	in der Wahrheitstafel für φ stehen in der mit „φ“ beschrifteten Spalte nur Einsen.
		$\iff$	$\neg\varphi$ ist unerfüllbar.
(d)	φ ist falsifizierbar	$\iff$	in der Wahrheitstafel für φ steht in der mit „φ“ beschrifteten Spalte mindestens eine 0.
		$\iff$	φ ist nicht allgemeingültig.

Bemerkung 3.27. (SymPy)

SymPy ist eine vollständig in Python implementierte Bibliothek für die Durchführung von Aufgaben in der symbolischen Mathematik. Sympy lässt sich problemlos in Python-Anwendungen einbetten, wir gehen hier aber den einfacheren Weg und benutzen die „App“ ***SymPy Live*** (`http://live.sympy.org/`).

Sympy Live stellt eine Shell zur Verfügung, in der aussagenlogische Formeln definiert und untersucht werden können. Zuerst müssen die Namen der aussagenlogischen Variablen deklariert werden. Dies geschieht mit einer Anweisung wie

```
u,v,mu = symbols(‘u,v,mu‘)
```

Standardmäßig sind in SymPy live `x,y,z,t` als Variablennamen deklariert. SymPy arbeitet mit den aussagenlogischen Konstanten

```
True, False
```

Atomare SymPy-Formeln sind `True`, `False` und die deklarierten aussagenlogischen Variablen. Man erhält alle SymPy-Formeln, wenn bereits konstruierte SymPy-Formeln geklammert oder mit den folgenden „SymPy-Junktoren“ verbunden werden. (Im Folgenden dürfen $x, y, x1, \ldots, xk$ durch bereits konstruierte SymPy-Formeln ersetzt werden.) Die Bedeutung der SymPy-Junktoren:

(a) `Not(x)` entspricht der Negation $\neg x$,

(b) `And(x1, ..., xk)` bzw. das „bitweise-Und“ `x1` $\&\cdots\&$ `xk` entspricht der Konjunktion $x1 \wedge \cdots \wedge xk$,

(c) `Or(x1, ..., xk)` bzw. das „bitweise-Oder“ `x1` $|\cdots|$ `xk` entspricht der Disjunktion $x1 \vee \cdots \vee xk$,

(d) `Implies(x,y)` entspricht der Implikation $x \to y$,

(e) `Equivalent(x1, x2)` entspricht der Äquivalenz $x1 \leftrightarrow x2$

(f) und `Xor(x1, ..., xk)` bzw. `x1 ^ ··· ^ xk` entspricht dem Xor $x1 \oplus \cdots \oplus xk$.

Formeln können mit der Zuweisung (=) gespeichert werden. Möchten wir zum Beispiel die aussagenlogische Formel

$$(x \oplus y) \leftrightarrow ((x \wedge \neg y) \vee (\neg x \wedge y))$$

als SymPy-Formel schreiben, gelingt das in SymPy Live mit der SymPy-Konstruktion

```
>>> phi = (x ^ y)
>>> psi = ((x & Not(y)) | (Not(x) & y))
>>> mu = Equivalent(phi,psi)
```

Natürlich hätten wir auch schreiben können

```
>>> phi = Xor(x, y)
>>> psi = Or( And(x, Not(y)) , And(Not(x),y))
>>> mu = Equivalent(phi,psi)
```

oder könnten `mu` in „einem Schlag“ definieren, d.h.

```
>>> mu = Equivalent( Xor(x, y) , Or( And(x, Not(y)) , And(Not(x),y)) )
```

Aber warum ist SymPy für uns interessant? Für eine SymPy-Formel `alpha` können wir die Funktion `satisfiable()` mit `alpha` als Argument aufrufen: Ist `alpha` erfüllbar, wird SymPy irgendeine erfüllende Belegung angeben, ist `alpha` unerfüllbar, antwortet SymPy mit `False`. Mit der Formel `mu` von oben erhalten wir

```
>>> satisfiable( mu )
                    {x:False, y:False}
```

bzw.

```
>>> satisfiable( Not(mu) )
                    False
```

Möchten wir alle erfüllenden Belegungen der Formel `mu` bestimmen, dann können wir dies mit der folgenden Python-Funktion tun:

```
>>> def erfuellen(models):
...     for model in models:
...         if model:
...              print(model)
...         else:
...              print('Die Formel ist unerfuellbar')
```

Der Aufruf

```
>>> erfuellen( satisfiable(mu, all_models=True))
```

gibt alle erfüllenden Belegungen aus bzw. erkennt dass die Formel unerfüllbar ist.

Alternativ können Sie den von `satisfiable(mu, all_models=True)` zurückgegebenen Generator mit

```
>>> list(satisfiable (mu, all_models=True))
```

in eine Liste umwandeln. Ist `mu` erfüllbar, so enthält die Liste alle erfüllenden Belegungen, wobei jede Belegung durch ein Wörterbuch repräsentiert wird. Ist `mu` unerfüllbar, enthält die Liste nur den Wert `False`.

3.3.3. Folgerung und Äquivalenz

Wir möchten uns mit allgemeingültigen Formeln beschäftigen und zwar insbesondere mit allgemeingültigen Implikationen (Folgerungen) und Biimplikationen (Äquivalenzen). Wir zeigen zum Beispiel in Satz 3.34 die beiden De Morganschen Regeln. Die erste De Morgansche Regel besagt, dass die Formel

$$((\neg(\varphi \wedge \psi)) \leftrightarrow (\neg\varphi \vee \neg\psi)) \tag{3.1}$$

allgemeingültig ist. Wir möchten herausstellen, dass hier letztlich die Aussage getroffen wird, dass die Formeln $\neg(\varphi \wedge \psi)$ und $(\neg\varphi \vee \neg\psi)$ äquivalent sind.

Definition 3.28. Seien φ, ψ aussagenlogische Formeln und Φ eine Menge aussagenlogischer Formeln.

(a) **(Semantische Folgerung für Formeln).**
Wir sagen ψ **folgt aus** φ (kurz: $\varphi \models \psi$), bzw. φ impliziert ψ, falls für jede zu φ und ψ passende Belegung $\mathcal{B}$ gilt:

ψ folgt aus φ
$\varphi \models \psi$

$$\text{Falls } [\![\varphi]\!]^{\mathcal{B}} = 1, \text{ so auch } [\![\psi]\!]^{\mathcal{B}} = 1.$$

Somit gilt:

$\varphi \models \psi \iff$ Jede Belegung, die zu φ und ψ passt und die φ erfüllt, erfüllt auch ψ.

(b) **(Semantische Folgerung für Mengen von Formeln).**
Wir sagen ψ **folgt aus** Φ (kurz: $\Phi \models \psi$), bzw. die Formeln in Φ implizieren ψ, falls für jede zu ψ und zu den Formeln in Φ passende Belegung $\mathcal{B}$ gilt:

ψ folgt aus Φ
$\Phi \models \psi$

$$\text{Falls } [\![\varphi]\!]^{\mathcal{B}} = 1 \text{ für alle } \varphi \in \Phi, \text{ so auch } [\![\psi]\!]^{\mathcal{B}} = 1.$$

Somit gilt:

$\Phi \models \psi \iff$ Jede Belegung, die zu allen Formeln in Φ und zu ψ passt und die alle Formeln in Φ erfüllt, erfüllt auch ψ.

(c) Wir schreiben $\models \psi$ an Stelle von $\emptyset \models \psi$. Wir schreiben einelementige Formelmengen Φ ohne Mengenklammern, wir schreiben also z.B. $\varphi \models \psi$ statt $\{\varphi\} \models \psi$.

(d) **(Semantische Äquivalenz).**
Die Formelmengen Φ und Ψ heißen (semantisch) **äquivalent** (kurz: $\Phi \equiv \Psi$), wenn jede Formel in Ψ von Φ und jede Formel in Φ von Ψ impliziert wird. D.h. es gilt

Äquivalenz
$\Phi \equiv \Psi$

- $\Phi \models \psi$ für alle Formeln $\psi \in \Psi$ und
- $\Psi \models \varphi$ für alle Formeln $\varphi \in \Phi$.

Die erste De Morgansche Regel lässt sich jetzt für alle Formeln φ, ψ kompakt als

$$\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi) \tag{3.2}$$

aufschreiben. Es wird sofort klar, dass wir eine Negation in eine Konjunktion „hineinschieben“ dürfen, wenn wir die Konjunktion durch eine Disjunktion ersetzen. Man vergleiche die neue Schreibweise in (3.2) mit der alten unübersichtlichen Darstellung in (3.1).

Beispiel 3.29.
Sei $\varphi := \big((X \vee Y) \wedge (\neg X \vee Y)\big)$ und $\psi := \big(Y \vee (\neg X \wedge \neg Y)\big)$.
Dann gilt $\varphi \models \psi$, aber es gilt **nicht** $\psi \models \varphi$ (kurz: $\psi \not\models \varphi$), denn:

X	Y	$(X \vee Y)$	$(\neg X \vee Y)$	φ	ψ
0	0	0	1	0	1
0	1	1	1	1	1
1	0	1	0	0	0
1	1	1	1	1	1

Hier repräsentiert jede Zeile eine zu φ und ψ passende Belegung. In jeder Zeile, in der in der mit „φ“ beschrifteten Spalte eine 1 steht, steht auch in der mit „ψ“ beschrifteten Spalte eine 1. Somit gilt $\varphi \models \psi$.

Andererseits steht in Zeile 1 in der mit „ψ“ beschrifteten Spalte eine 1 und in der mit „φ“ beschrifteten Spalte eine 0. Für die entsprechende Belegung $\mathcal{B}$ (mit $\mathcal{B}(X) = 0$ und $\mathcal{B}(Y) = 0$) gilt also $[\![\psi]\!]^{\mathcal{B}} = 1$ und $[\![\varphi]\!]^{\mathcal{B}} = 0$. Daher gilt $\psi \not\models \varphi$.

Beobachtung 3.30. Seien φ und ψ beliebige aussagenlogische Formeln. Dann gilt:

(a) $\mathbf{0} \models \varphi$ und $\varphi \models \mathbf{1}$.

(b) $\mathbf{1} \models \varphi \iff \models \varphi \iff \varphi$ ist allgemeingültig.

(c) $\varphi \models \mathbf{0} \iff \varphi$ ist unerfüllbar.

(d) $\varphi \models \psi \iff (\varphi \to \psi)$ ist allgemeingültig $\iff (\varphi \wedge \neg\psi)$ ist unerfüllbar.

Beweis: Übung. □

Beispiel 3.31. Sei $\varphi := \big(X \wedge (X \vee Y)\big)$ und $\psi := X$. Dann ist $\varphi \equiv \psi$, denn

X	Y	$(X \vee Y)$	φ	ψ
0	0	0	0	0
0	1	1	0	0
1	0	1	1	1
1	1	1	1	1

Hier ist die mit „φ“ beschriftete Spalte identisch zur mit „ψ“ beschrifteten Spalte. D.h. für alle zu φ und ψ passenden Belegungen $\mathcal{B}$ gilt $[\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}}$. Somit gilt $\varphi \equiv \psi$.

Beobachtung 3.32. Seien φ und ψ aussagenlogische Formeln. Dann gilt:

(a) $\varphi \equiv \psi \iff (\varphi \leftrightarrow \psi)$ ist allgemeingültig $\iff \varphi \models \psi$ und $\psi \models \varphi$.

(b) φ ist allgemeingültig $\iff \varphi \equiv \mathbf{1}$.

(c) φ ist erfüllbar $\iff \varphi \not\equiv \mathbf{0}$ (d.h. „$\varphi \equiv \mathbf{0}$“ gilt nicht).

Beweis: Übung. □

Bemerkung 3.33. (SymPy)
Seien φ, ψ aussagenlogische Formeln und seien `phi, psi` SymPy-Formeln, die zu φ bzw. ψ äquivalent sind. Wir benutzen SymPy zuerst, um festzustellen ob die semantische Folgerung

$$\varphi \models \psi$$

gilt. Dazu definieren wir die SymPy-Funktion `implication` in SymPy Live[2] wie folgt

```
>>> def implication( phi, psi ):
...     return satisfiable( Not(Implies( phi, psi ))) == False
```

$\varphi \models \psi$ gilt genau dann, wenn `implication( phi, psi )` den Wert `True` ausgibt. Im zweiten Schritt überprüfen wir mit SymPy, ob die Äquivalenz

$$\varphi \equiv \psi$$

gilt: Diese Überprüfung gelingt mit der Definition

```
>>> def equivalence( phi, psi ):
...     return implication( phi, psi ) & implication( psi, phi )
```

$\varphi \equiv \psi$ gilt genau dann, wenn `equivalence( phi, psi )` den Wert `True` ausgibt.

Fundamentale Äquivalenzen der Aussagenlogik:

Satz 3.34. Seien φ, ψ und χ aussagenlogische Formeln. Dann gilt:

(a) **Idempotenz:**
- $(\varphi \wedge \varphi) \equiv \varphi$
- $(\varphi \vee \varphi) \equiv \varphi$

(b) **Kommutativität:** Für alle Junktoren $\circ \in \{\vee, \wedge, \leftrightarrow, \oplus\}$ gilt
- $(\varphi \circ \psi) \equiv (\psi \circ \varphi)$

(c) **Assoziativität:** Für alle Junktoren $\circ \in \{\vee, \wedge, \leftrightarrow, \oplus\}$ gilt
- $((\varphi \circ \psi) \circ \chi) \equiv (\varphi \circ (\psi \circ \chi))$

(d) **Absorption:**
- $(\varphi \wedge (\varphi \vee \psi)) \equiv \varphi$
- $(\varphi \vee (\varphi \wedge \psi)) \equiv \varphi$

(e) **Distributivität:**
- $(\varphi \wedge (\psi \vee \chi)) \equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \chi))$
- $(\varphi \vee (\psi \wedge \chi)) \equiv ((\varphi \vee \psi) \wedge (\varphi \vee \chi))$

(f) **doppelte Negation:**
- $\neg\neg\varphi \equiv \varphi$

(g) **De Morgansche Regeln:**

[2] Benutze das Setting "Submit with Shift-Enter"

- $\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$
- $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$

(h) **Tertium non datur:**

- $(\varphi \wedge \neg\varphi) \equiv \mathbf{0}$
- $(\varphi \vee \neg\varphi) \equiv \mathbf{1}$

(i)

- $(\varphi \wedge \mathbf{1}) \equiv \varphi$
- $(\varphi \wedge \mathbf{0}) \equiv \mathbf{0}$
- $(\varphi \vee \mathbf{1}) \equiv \mathbf{1}$
- $(\varphi \vee \mathbf{0}) \equiv \varphi$

(j)

- $\mathbf{1} \equiv \neg\mathbf{0}$
- $\mathbf{0} \equiv \neg\mathbf{1}$

(k) **Elimination der Implikation:**

- $(\varphi \to \psi) \equiv (\neg\varphi \vee \psi)$

(l) **Elimination der Biimplikation:**

- $(\varphi \leftrightarrow \psi) \equiv ((\varphi \to \psi) \wedge (\psi \to \varphi)) \equiv \neg(\varphi \oplus \psi)$

(m) **Kontraktion**:

- $(\varphi \to (\varphi \to \psi)) \equiv (\varphi \to \psi)$

Beweis: Durch einfaches Nachrechnen. Details: Übung. □

Bemerkung 3.35. (Aussagenlogik und Beziehungen zwischen Mengen)
Viele der in Satz 3.34 festgestellten Äquivalenzen haben wir in ähnlicher Form als Mengengleichheiten in Satz 2.29 und Satz 2.31 kennengelernt. Tatsächlich ist dies kein Zufall, denn es gibt einen engen Zusammenhang zwischen semantischen Äquivalenzen und Mengengleichheiten: Sei nämlich $X = Y$ eine behauptete Mengengleichheit zwischen Mengen X und Y, wobei X und Y aus „Grundmengen“ $M_1, \ldots, M_k$ mit Hilfe der Mengenoperationen $\cap, \cup, \oplus$ sowie der Mengendifferenz $\setminus$ gebildet werden. Wir überführen die Mengengleichheit $X = Y$ am Beispiel des Distributivgesetzes

$$X := M_1 \cap (M_2 \cup M_3) = (M_1 \cap M_2) \cup (M_1 \cap M_3) =: Y$$

in eine semantische Äquivalenz $\varphi_X \equiv \varphi_Y$ wie folgt.

1. Ersetze jedes Vorkommen einer Menge M_i in X oder Y durch die aussagenlogische Variable V_i. Wir erhalten
$$V_1 \cap (V_2 \cup V_3) = (V_1 \cap V_2) \cup (V_1 \cap V_3)$$
2. Jetzt ersetze in beiden Mengenausdrücken die Mengenoperationen $\cap, \cup, \oplus$ durch die Junktoren $\wedge, \vee, \oplus$. Wir erhalten
$$V_1 \wedge (V_2 \vee V_3) = (V_1 \wedge V_2) \vee (V_1 \wedge V_3)$$
Übrigens, wie ist die Mengendifferenz $U \setminus W$ zu behandeln?

3. Im letzten Schritt ersetze das Gleichheitssymbol durch das Symbol $\equiv$ für die semantische Äquivalenz. Wir erhalten die (behauptete!) semantische Äquivalenz

$$V_1 \wedge (V_2 \vee V_3) \equiv (V_1 \wedge V_2) \vee (V_1 \wedge V_3)$$

Teil (e) in Satz 3.34 besagt, dass die Äquivalenz tatsächlich wahr ist.

Man kann allgemein zeigen, dass

$$X = Y \text{ gilt für alle Grundmengen } M_1, \ldots, M_n \iff \varphi_X \equiv \varphi_Y$$

stets gilt. Für den Nachweis müssten wir zuerst mit einer rekursiven Definition genau sagen, was ein Mengenausdruck ist. Den Nachweis führt man dann mit einer Induktion über den Aufbau eines Mengenausdrucks. Da wir die zugrundeliegende Beweismethode, nämlich die vollständige Induktion, erst später behandeln, lassen wir es bei der folgenden Intuition bewenden:

(a) Um die Äquivalenz $\varphi_X \equiv \varphi_Y$ bei n Variablen zu überprüfen, müssen alle möglichen Belegungen $b \in \{0,1\}^n$ der beteiligten aussagenlogischen Variablen betrachtet werden.

(b) Ähnliches passiert in der Mengengleichheit $X = Y$ bei dann n Mengenvariablen: Wir können für jedes $b \in \{0,1\}^n$ annehmen, dass es ein Element x_b gibt so dass für alle i:

$$x_b \in M_i \iff b_i = 1.$$

(Die Mengen M_i befinden sich in „allgemeiner Lage“.)

3.4. Normalformen

Bisher haben wir gesehen, wie man für eine gegebene aussagenlogische Formel φ eine Wahrheitstafel aufstellen kann.

Frage: Wie kann man umgekehrt zu einer gegebenen Wahrheitstafel eine „passende“ Formel φ finden?

3.4.1. Die disjunktive Normalform

Wir möchten Formeln einer besonders einfachen Form konstruieren, nämlich Disjunktionen von Formeln, die aus Konjunktionen von Variablen oder negierten Variablen bestehen. Formeln, die diese spezielle Struktur besitzen, nennt man auch Formeln in **disjunktiver Normalform** (kurz: **DNF**).

Definition 3.36 (disjunktive Normalform).

(a) Ein **Literal** (zur Variablen $X \in \text{AVAR}$) ist eine Formel der Form X oder $\neg X$. Man nennt X auch **positives Literal** und $\neg X$ **negatives Literal**. — Literal, positives Literal, negatives Literal

(b) Eine Formel der Form $\bigwedge_{j=1}^{m} \ell_j$ mit Literalen $\ell_1, \ldots, \ell_m$ heißt **Konjunktionsterm** (bzw. **Monom**). Ein leerer Konjunktionsterm besitzt keine Literale und wird als allgemeingültig definiert. — Konjunktionsterm, Monom

DNF

(c) Eine aussagenlogische Formel ψ ist in **disjunktiver Normalform** (**DNF**), wenn ψ eine Disjunktion von Konjunktionstermen ist, d.h. wenn ψ die Form

$$\psi = \bigvee_{i=1}^{k} \left(\bigwedge_{j=1}^{m_i} \ell_{i,j} \right)$$

hat, wobei $k, m_1, \ldots, m_k \in \mathbb{N}_{>0}$ und $\ell_{i,j}$ ein Literal ist (für jedes $i \in \{1, \ldots, k\}$ und $j \in \{1, \ldots, m_i\}$). Gilt $\varphi \equiv \psi$ für eine Formel φ, dann nennen wir ψ auch eine DNF für φ, bzw. sagen, dass φ die DNF ψ besitzt.

Normalformen spielen in vielen Anwendungsgebieten eine wichtige Rolle. Beispielsweise geht man in der Schaltungstechnik (Hardware-Entwurf) oft von DNF-Formeln aus. In Abschnitt 3.4.2 führen wir die Normalform der KNF-Formeln als Konjunktionen von Disjunktionen ein. KNF-Formeln eignen sich für die aussagenlogische Modellbildung, da sich eine Sammlung einfach strukturierter Aussagen häufig gut durch eine Konjunktion von Disjunktionen ausdrücken lässt.

Definition 3.37. Sei T eine Wahrheitstafel einer Formel φ und z eine Zeile von T.

1-Zeile
0-Zeile

(a) z heißt eine **1-Zeile**, wenn in der Spalte von φ eine „1“ steht. Steht in der Spalte von φ eine „0“, nennen wir z eine **0-Zeile**.

Vollkonjunktion

(b) $\mathrm{Var}(T)$ ist die Menge aller aussagenlogischen Variablen von T. Für jede 1-Zeile z mit Belegung $\mathcal{B}_z : \mathrm{Var}(T) \to \{0, 1\}$ definieren wir eine **Vollkonjunktion** Konj_z von **z** als einen Konjunktionsterm

- in dem genau die Literale zu den Variablen aus $\mathrm{Var}(T)$ vorkommen und
- der von $\mathcal{B}_z$ erfüllt, aber von Belegungen zu anderen Zeilen falsifiziert wird.

Die Vollkonjunktion der 1-Zeile z hat also die Form

$$\mathrm{Konj}_z = \Big(\bigwedge_{V \in \mathrm{Var}(T) : \mathcal{B}_z(V) = 1} V \Big) \wedge \Big(\bigwedge_{W \in \mathrm{Var}(T) : \mathcal{B}_z(W) = 0} \neg W \Big).$$

(c) Eine Disjunktion

$$\psi := \bigvee_{z:\ z \text{ ist eine 1-Zeile}} \mathrm{Konj}_z$$

kanonische DNF

heißt **kanonische DNF** der Wahrheitstafel T.

Beispiel 3.38. Betrachte die Wahrheitstafel T:

X	Y	Z	φ
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Es gibt genau drei 1-Zeilen, nämlich

X	Y	Z	φ	zur Belegung der Zeile gehörende Vollkonjunktion:
0	0	0	1	$\neg X \wedge \neg Y \wedge \neg Z$
1	0	0	1	$X \wedge \neg Y \wedge \neg Z$
1	0	1	1	$X \wedge \neg Y \wedge Z$

Die kanonische DNF der Wahrheitstafel T ist die Formel ψ mit

$$\psi \;:=\; (\neg X \wedge \neg Y \wedge \neg Z) \;\vee\; (X \wedge \neg Y \wedge \neg Z) \;\vee\; (X \wedge \neg Y \wedge Z).$$

Satz 3.39. Sei φ eine aussagenlogische Formel. Dann gibt es eine DNF ψ für φ.

Beweis: Sei T die Wahrheitstafel von φ und sei ψ die in Definition 3.37 beschriebene kanonische DNF von T. Wir behaupten, dass ψ die Anforderungen der Behauptung erfüllt. Warum? Für verschiedene Zeilen $u \neq v$ von T gilt

$$[\![\mathrm{Konj}_u]\!]^{\mathcal{B}_u} = 1 \text{ und } [\![\mathrm{Konj}_u]\!]^{\mathcal{B}_v} = 0.$$

(a) Ist u eine 1-Zeile von T, dann ist $[\![\mathrm{Konj}_u]\!]^{\mathcal{B}_u} = 1$ für die Vollkonjunktion Konj_u der Zeile u.

$$\Longrightarrow \qquad [\![\psi]\!]^{\mathcal{B}_u} = 1.$$

(b) Ist u eine 0-Zeile von T, dann ist $[\![\mathrm{Konj}_z]\!]^{\mathcal{B}_u} = 0$ für alle 1-Zeilen z.

$$\Longrightarrow \qquad [\![\psi]\!]^{\mathcal{B}_u} = 0.$$

Aus (a) und (b) folgt somit $[\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}}$ für alle Belegungen $\mathcal{B}$. Also gilt $\varphi \equiv \psi$. □

Bemerkung 3.40. (SymPy: Berechnung einer DNF)
Sei φ eine aussagenlogische Formel und sei `phi` eine zu φ äquivalente SymPy-Formel. Dann erhalten wir eine DNF für φ mit der SymPy-Funktion `to_dnf` und dem Aufruf

```
>>> to_dnf( phi )
```

Allerdings ist die Ausgabe einer möglichst kurzen DNF vorzuziehen. Dazu wähle den Aufruf

```
>>> to_dnf( phi , simplify=True )
```

3.4.1.1. Die Größe von DNFs

Die kanonische DNF kann unnötig lang sein, denn jede 1-Zeile trägt mit ihrer Vollkonjunktion zur Länge der kanonischen DNF bei. Zum Beispiel hat die kanonische DNF für die Wahrheitstafel der trivialen Formel $\varphi = \mathbf{1} \vee x_1 \vee \cdots \vee x_n$ bei n Variablen 2^n Vollkonjunktionen! Um mit DNFs effizient arbeiten zu können, versuchen wir DNFs mit möglichst wenigen Konjunktionstermen zu erhalten.

Definition 3.41. Sei φ eine aussagenlogische Formel.

(a) Ein Konjunktionsterm K heißt **Implikant** von φ, wenn Implikant

$$K \models \varphi.$$

Primimplikant

(b) Ein Implikant K heißt **Primimplikant** von φ, wenn K nicht verkürzbar ist, wenn also K nach Streichen irgendeines Literals kein Implikant von φ mehr ist.

Bemerkung 3.42. In der Literatur herrscht eine babylonische Sprachverwirrung. Eine „Vollkonjunktion“ wird häufig auch als „Minterm“ bezeichnet. Manchmal wird aber „Minterm“ als Synonym zu „Primimplikant“ benutzt.

Konjunktionsterme einer DNF für φ „implizieren“ φ. Also besteht eine DNF für φ nur aus Implikanten von φ wie wir gleich sehen werden.

Eine DNF für φ lässt sich auffassen als eine Sammlung von Gründen: Tritt ein solcher Grund, d.h. tritt ein Kojunktionsterm der DNF ein, muss φ gelten. Natürlich sollte man an den allgemeinsten Gründen, d.h. den kürzesten Implikanten interessiert sein. Aber das sind gerade die nichtverkürzbaren Implikanten, nämlich die Primimplikanten.

Satz von Quine

Satz 3.43. (**Satz von Quine**). Sei φ eine aussagenlogische Formel.

(a) Jede DNF für φ ist eine Disjunktion von Implikanten von φ.

(b) Sei K ein Konjunktionsterm und sei V eine Variable von φ, die in K nicht vorkomme. Dann gilt

$$K \text{ ist ein Implikant von } \varphi \iff K \wedge V \text{ und } K \wedge \neg V \text{ sind Implikanten von } \varphi.$$

(c) Es gibt eine DNF für φ mit einer kleinstmöglichen Anzahl von Konjunktionstermen, und diese DNF besteht nur aus einer Auswahl von Primimplikanten von φ.

(d) Wenn alle Primimplikanten von φ Vollkonjunktionen sind, dann ist die kanonische DNF die, bis auf die Reihenfolge der Primimplikanten einzige DNF für φ, die nur aus Primimplikanten besteht. Insbesondere besitzt die kanonische DNF in diesem Fall die kleinstmögliche Anzahl von Konjunktionstermen.

Beweis: **(a)** Sei D eine DNF für φ und K ein Konjunktionsterm in D. Wenn K kein Implikant von φ ist, dann gibt eine Belegung, die K (und damit auch D) erfüllt, aber φ falsifiziert. Dann ist aber D nicht äquivalent zu φ. Widerspruch!

(b) Wir benutzen Definition 3.41:

$$\begin{aligned}
K \text{ ist Implikant von } \varphi &\iff K \models \varphi \\
&\iff [\![K \to \varphi]\!]^{\mathcal{B}} = 1 \text{ für alle Belegungen } \mathcal{B} \\
&\iff [\![(K \wedge V) \to \varphi]\!]^{\mathcal{B}} = 1 = [\![(K \wedge \neg V) \to \varphi]\!]^{\mathcal{B}} \text{ für alle Belegungen } \mathcal{B} \\
&\iff (K \wedge V) \models \varphi \text{ und } (K \wedge \neg V) \models \varphi \\
&\iff K \wedge V,\ K \wedge \neg V \text{ sind Implikanten von } \varphi.
\end{aligned}$$

(c) Angenommen, D ist eine DNF für φ und D enthält einen Implikanten K, der kein Primimplikant ist. Dann gibt es einen Implikanten K' von φ, der durch Streichen eines Literals aus K resultiert. Sei D' die DNF, die man aus D erhält, wenn K durch K' ersetzt wird. Dann folgt

$$D \models D' \models \varphi \models D, \tag{3.3}$$

denn K impliziert K' und damit impliziert D die DNF D'. Aber D (und damit auch D') besteht nach Teil (a) nur aus Implikanten von φ und $D' \models \varphi$ folgt. Da nach Voraussetzung D eine DNF für φ ist, müssen, als Konsequenz von (3.3), φ, D und D' alle äquivalent sein. Wir können dieses Verfahren solange wiederholen bis die DNF nur aus Primimplikanten besteht.

(d) Sei D eine DNF für φ, die nur aus Primimplikanten besteht. Wenn alle Primimplikanten Vollkonjunktionen sind, dann muss muss D aus *allen* Vollkonjunktionen zu 1-Zeilen bestehen und D stimmt deshalb mit der kanonischen DNF überein. □

In der Veranstaltung „Hardwarearchitekturen und Rechensysteme“ wird das Quine-McCluskey-Verfahren vorgestellt, das alle Primplikanten bestimmt und dann versucht eine möglichst kleine DNF zu „bauen“.

Sei φ eine aussagenlogische Formel mit n Variablen. Wir beschreiben das **Primimplikanten-Verfahren**, um alle Primimplikanten von φ zu bestimmen.

Primimplikanten-Verfahren

1. Zuerst bestimme alle Implikanten von φ mit n Variablen.

   /* Jeder Implikant der „Länge“ n entspricht einer 1-Zeile der Wahrheitstafel von φ und umgekehrt. Beachte, dass die Implikanten der Länge n genau den Konjunktionstermen der kanonischen DNF entsprechen. */

2. **Verkürzung**: Unterscheiden sich zwei Implikanten $K \wedge V, K \wedge \neg V$ nur in der Variable V, dann ist auch K ein Implikant (mit $n-1$ Variablen).

   /* Als Konsequenz des Satzes von Quine, Teil (b) folgt, dass man genau die Menge der Implikanten (mit $n-1$ Variablen) auf diese Weise erhält. */

3. Wende das Verfahren aus Schritt (2) rekursiv auf alle Implikanten von φ mit genau $n-1$ Variablen an.

Man erhält die Menge aller Primimplikanten von φ als die Menge aller nicht weiter verkürzbaren Implikanten von φ.

Beispiel 3.44. In Beispiel 3.38 haben wir die kanonische DNF

$$\psi \quad := \quad (\neg X \wedge \neg Y \wedge \neg Z) \ \vee \ (X \wedge \neg Y \wedge \neg Z) \ \vee \ (X \wedge \neg Y \wedge Z). \tag{3.4}$$

gefunden.

Wir wenden das Primimplikanten-Verfahren an, um alle Primimplikanten von ψ zu berechnen. Da wir ψ als kanonische DNF bestimmt haben, sind ihre Konjunktionsterme, also $\neg X \wedge \neg Y \wedge \neg Z$, $X \wedge \neg Y \wedge \neg Z$ und $X \wedge \neg Y \wedge Z$, genau die Implikanten der Länge 3.

Welche Implikanten der Länge 2 hat ψ? Wir erhalten den Implikanten

$$\neg Y \wedge \neg Z,$$

wenn wir einen Verkürzungsschritt auf die Implikanten $\neg X \wedge \neg Y \wedge \neg Z$ und $X \wedge \neg Y \wedge \neg Z$ anwenden. (Beachte $(\neg X \wedge \neg Y \wedge \neg Z) \equiv (\neg Y \wedge \neg Z) \wedge \neg X$ und $(X \wedge \neg Y \wedge \neg Z) \equiv (\neg Y \wedge \neg Z) \wedge X$. Also ist $\neg Y \wedge \neg Z$ tatsächlich das Resultat des Verkürzungsschritts.)

Wir können den Verkürzungsschritt ein weiteres Mal ausführen und zwar für die Implikanten $X \wedge \neg Y \wedge \neg Z$ und $X \wedge \neg Y \wedge Z$. Diesmal erhalten wir den Implikanten

$$X \wedge \neg Y.$$

Eine weitere Anwendung des Verkürzungsschrittes ist nicht möglich. Daher sind $\neg Y \wedge \neg Z,\ X \wedge \neg Y$ die beiden einzigen Implikanten von ψ der Länge 2.

Wir haben alle Implikanten der Länge 3 verkürzen können. Wir können aber einen Verkürzungsschritt auf die beiden gerade gefundenen Implikanten der Länge 2 nicht mehr anwenden: Unser „Primimplikanten-Verfahren“ terminiert mit den Primimplikanten $\neg Y \wedge \neg Z,\ X \wedge \neg Y$.

Gibt es eine kleinere mit ψ äquivalente DNF? Ja, die DNF

$$\psi' := (\neg Y \wedge \neg Z) \vee (X \wedge \neg Y),$$

die aus den beiden einzigen Primimplikanten besteht. □ Ende von Beispiel 3.44

Beispiel 3.45. $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ seien aussagenlogische Variablen. Betrachte die Formel

$$\psi_n \quad := \quad \bigvee_{i=1}^{n} (X_i \oplus Y_i).$$

Wie viele 0-Zeilen hat die Wahrheitstafel für ψ_n? Eine Belegung $\mathcal{B}$ gehört genau dann zu einer 0-Zeile, wenn $\mathcal{B}(X_i) = \mathcal{B}(Y_i)$ für alle $i = 1, \ldots, n$ gilt. Also gibt es zu jeder der 2^n Belegungen $\mathcal{B} : \{X_1, \ldots, X_n\} \to \{0,1\}$ genau eine Belegung $\mathcal{B}' : \{Y_1, \ldots, Y_n\} \to \{0,1\}$ die zu einer 0-Zeile führt: Es gibt genau 2^n 0-Zeilen und damit genau $2^{2n} - 2^n$ 1-Zeilen. Die kanonische DNF für ψ_n hat deshalb $2^{2n} - 2^n$ Konjunktionsterme.

ψ_n hat aber auch die DNF

$$\Big(\bigvee_{i=1}^{n} \big((X_i \wedge \neg Y_i) \vee (\neg X_i \wedge Y_i)\big) \Big) \equiv \Big(\bigvee_{i=1}^{n} (X_i \wedge \neg Y_i) \vee \bigvee_{i=1}^{n} (\neg X_i \wedge Y_i) \Big)$$

mit $2n$ Konjunktionstermen. Wir haben also eine DNF gefunden, die viel, viel kleiner als die kanonische DNF ist. □ Ende von Beispiel 3.45

Gibt es aussagenlogische Formeln, die keine „kleinen“ DNFs besitzen?

Satz 3.46. $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ seien aussagenlogische Variablen. Dann hat jede zu

$$\varphi_n := \quad \bigwedge_{i=1}^{n} (X_i \leftrightarrow Y_i)$$

äquivalente DNF mindestens $\mathbf{2^n}$ Konjunktionsterme.

(Beachte, dass $\varphi_n = \bigwedge_{i=1}^{n} (X_i \leftrightarrow Y_i) \equiv \neg \bigvee_{i=1}^{n} (X_i \oplus Y_i) = \neg\psi_n$ gilt.)

Beweis: Beachte zuerst, dass alle Primimplikanten von φ_n Vollkonjunktionen sind. Warum? Jeder Konjunktionsterm mit höchstens $2n-1$ Literalen wird von zwei Belegungen erfüllt, die sich nur in dem Wahrheitswert einer Variablen unterscheiden. Dann wird aber eine der beiden Belegungen φ_n falsifizieren. Der Konjunktionsterm kann kein Implikant und erst recht kein Primimplikant sein.

Wir wenden den Satz von Quine (Satz 3.43 (c)) an und erhalten, dass die kanonische DNF für φ_n eine kleinstmögliche Anzahl von Konjunktionstermen besitzt. Aber die kanonische DNF besitzt genauso viele Vollkonjunktionen wie φ_n 1-Zeilen besitzt, nämlich 2^n viele. □

3.4.2. Die konjunktive Normalform

Definition 3.47 (konjunktive Normalform).

(a) Ein **Disjunktionsterm** (oder eine **Klausel**) ist eine Disjunktion $\bigvee_{i=1}^{n} l_i$ von Literalen $l_1, \ldots, l_n$. Ein leerer Disjunktionsterm besitzt keine Literale und wird als unerfüllbar definiert. Disjunktionsterm, Klausel

(b) Eine aussagenlogische Formel ψ ist in konjunktiver Normalform (**KNF**), wenn ψ eine Konjunktion von Disjunktionstermen ist, d.h. wenn ψ die Form KNF

$$\psi = \bigwedge_{i=1}^{k} \left(\bigvee_{j=1}^{m_i} \ell_{i,j} \right)$$

hat, wobei $k, m_1, \ldots, m_k \in \mathbb{N}_{>0}$ und $\ell_{i,j}$ für jedes $i \in \{1, \ldots, k\}, j \in \{1, \ldots, m_i\}$ ein Literal ist. Gilt für eine Formel φ die Äquivalenz $\varphi \equiv \psi$, dann nennen wir ψ eine KNF für φ, bzw. sagen, dass φ die KNF ψ besitzt.

KNFs bieten sich in der Modellierung an, wenn das zu modellierende System gleichzeitig einfache Eigenschaften erfüllen muss. Die gleichzeitige Erfüllung der Eigenschaften entspricht einer Konjunktion. Wenn wir Glück haben, lassen sich die Eigenschaften als Disjunktionsterme oder zumindest als kurze KNFs schreiben.

Später müssen wir uns allerdings fragen, ob, bzw. in welchem Umfang wir KNFs auch beherrschen können.

Beispiel 3.48. (Sudoku).

In einem Sudoku-Rätsel ist jede Zelle einer 9×9-Matrix mit einer der Ziffern $1, \ldots, 9$ zu füllen, so dass keine Ziffer zweimal in einer Zeile oder zweimal in einer Spalte oder zweimal in einer der neun disjunkten 3×3-Teilmatrizen vorkommt. Wir nehmen an, dass einige Zellen bereits Ziffern erhalten haben und fragen uns, ob wir alle verbleibenden Zellen entsprechend den Sudoku-Regeln füllen können.

						9		
7	9			6				
		5		3			1	4
	4				6			7
		3	2	1		5		
	7						8	
					1		9	2
			5			7		
	1			9	4		5	

ungelöstes Sudoku

Wir können diese Frage als eine Frage nach der Erfüllbarkeit einer KNF-Formel auffassen. Dazu definieren wir die $9^3 = 729$ aussagenlogischen Variablen $Z_{i,j}^k$ für $1 \leqslant i, j, k \leqslant 9$, wobei $Z_{i,j}^k$ aussagen soll, dass in Zeile i und Spalte j die Zahl k steht. Wir formulieren die Sudoku-Regeln als Konjunktion von Disjunktionstermen.

1. Für jede Zeile i und alle Ziffern k fordern wir,

- dass Ziffer k mindestens einmal erscheint und erreichen dies mit dem Disjunktionsterm

$$D_i^k := \bigvee_{j=1}^{9} Z_{i,j}^k,$$

- dass k nicht zweimal vergeben wird, d.h. dass gilt

$$Z_{i,j}^k \to \neg Z_{i,j^*}^k$$

für alle Spalten $j \neq j^*$ mit $1 \leqslant j \leqslant j^* \leqslant 9$ und alle Ziffern k. (Warum dürfen wir $j \leqslant j^*$ fordern?)

- Wir beachten die Äquivalenz $(Z_{i,j}^k \to \neg Z_{i,j^*}^k) \equiv (\neg Z_{i,j}^k \vee \neg Z_{i,j^*}^k)$. Wir drücken mit einer KNF aus, dass die Ziffer k genau einmal in Zeile i erscheint:

$$\text{Zeile}_i^k := D_i^k \wedge \bigwedge_{j=1}^{8} \bigwedge_{j^*=j+1}^{9} (\neg Z_{i,j}^k \vee \neg Z_{i,j^*}^k)$$

aus.

2. Wie drücken wir aus, dass jede Ziffer genau einmal in jeder Zeile vorkommt? Wir definieren die KNF

$$\text{Zeile} := \bigwedge_{i=1}^{9} \bigwedge_{k=1}^{9} \text{Zeile}_i^k.$$

3. Analog definieren wir KNFs

Spalte und Teilmatrix

für die Spalten und 3×3 Teilmatrizen, diesmal um auszudrücken, dass jede Ziffer in jeder Spalte, bzw. in jeder Teilmatrix genau einmal vorkommt.

4. Weiterhin müssen wir fordern, dass keine Zelle (i, j) zwei oder mehr Ziffern erhält, d.h.

$$Z_{i,j}^k \to \neg Z_{i,j}^{k^*}$$

gilt für alle $1 \leqslant i, j \leqslant 9$ und alle verschiedenen Ziffern k und k^*. Mit der folgenden KNF fordern wir, dass in **jeder** Zelle höchstens eine Ziffer steht

$$\text{Einfach} := \bigwedge_{i=1}^{9} \bigwedge_{j=1}^{9} \bigwedge_{k=1}^{8} \bigwedge_{k^*=k+1}^{9} (\neg Z_{i,j}^k \vee \neg Z_{i,j}^{k^*}).$$

5. Ist die Zelle in Zeile i und Spalte j mit der Ziffer k bereits gesetzt, dann fügen wir den Disjunktionsterm $Z_{i,j}^k$ hinzu. Und definieren die KNF

Schon_da

als Konjunktion dieser Disjunktionsterme für alle bereits gesetzten Zellen.

In der KNF

$$\text{Sudoku} := \text{Zeile} \wedge \text{Spalte} \wedge \text{Teilmatrix} \wedge \text{Einfach} \wedge \text{Schon\_da}$$

haben wir alle Sudoku-Regeln umgesetzt: Erfüllende Belegungen lösen das Sudoku-Rätsel. (Warum haben wir nicht fordern müssen, dass jede Zelle mindestens eine Ziffer erhält?)

4	3	8	1	5	2	9	7	6
7	9	1	4	6	8	2	3	5
6	2	5	7	3	9	8	1	4
5	4	9	3	8	6	1	2	7
8	6	3	2	1	7	5	4	9
1	7	2	9	4	5	6	8	3
3	5	6	8	7	1	4	9	2
9	8	4	5	2	3	7	6	1
2	1	7	6	9	4	3	5	8

Lösung

□ Ende von Beispiel 3.48

Es gibt Formeln, die nur sehr große DNFs aber kleine KNFs besitzen.

Beispiel 3.49. In Satz 3.46 wurde gezeigt, dass DNFs für

$$\varphi_n = \bigwedge_{i=1}^{n} (X_i \leftrightarrow Y_i)$$

mindestens 2^n Konjunktionsterme benötigen. Aber φ_n hat eine „kleine“ KNF, nämlich die KNF

$$K_{\varphi_n} := \bigwedge_{i=1}^{n} \Big((X_i \vee \neg Y_i) \wedge (\neg X_i \vee Y_i) \Big)$$

mit nur $2n$ Disjunktionstermen. (Beachte, dass $(X_i \leftrightarrow Y_i) \equiv \Big((X_i \vee \neg Y_i) \wedge (\neg X_i \vee Y_i) \Big)$ gilt.)

Hat vielleicht jede Formel eine „kleine“ DNF *oder* eine „kleine“ KNF? Wir geben eine negative Antwort in Satz 3.67.

□ Ende von Beispiel 3.49

Beobachtung 3.50. Wie erhält man eine zu einer Wahrheitstafel T passende KNF?. Wir wenden die De Morganschen Regeln an und erhalten

$$\neg \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{m_i} \ell_{i,j} \right) \equiv \bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{m_i} \neg \ell_{i,j} \right).$$

Um eine mit der Formel φ äquivalente KNF zu erhalten, genügt somit die Bestimmung einer mit der Formel $\neg\varphi$ äquivalenten DNF D, denn dann ist $\neg D$ eine mit der Formel $\neg\neg\varphi \equiv \varphi$ äquivalente KNF.

Also besitzt jede aussagenlogische Formel auch eine äquivalente KNF.

Satz 3.51. Für jede aussagenlogische Formel φ gibt es eine DNF D und eine KNF K mit $D \equiv \varphi \equiv K$.

Statt den Umweg über die Konstruktion einer DNF für die negierte Formel zu gehen, können wir natürlich auch direkt vorgehen. Hier ist die Idee:

> Für jede 0-Zeile z von φ konstruiere einen Disjunktionsterm D_z, der von der Belegung von z falsifiziert, aber von den Belegungen der verbleibenden Zeilen erfüllt wird: Der gesuchte Disjunktionsterm ist die negierte Vollkonjunktion von z.

Definition 3.52. Sei T eine Wahrheitstafel.

Volldisjunktion

Für jede 0-Zeile z von T konstruiere die **Volldisjunktion** Dis_z der Zeile z, wobei gilt

$$\mathrm{Dis}_z \equiv \neg\big(\mathrm{Konj}_z\big).$$

kanonische KNF

Wir erhalten die **kanonische KNF** nach Ver*und*ung der Volldisjunktionen zu allen 0-Zeilen!

Die kanonische KNF besitzt für jede 0-Zeile z eine Volldisjunktion, die nur von z nicht erfüllt wird. Da jede Volldisjunktion der kanonischen KNF von allen 1-Zeilen der Wahrheitstafel erfüllt wird, haben wir eine zur Wahrheitstafel passende KNF gefunden!

Bemerkung 3.53. (SymPy: Berechnung einer KNF)
Sei φ eine aussagenlogische Formel und sei `phi` eine zu φ äquivalente SymPy-Formel. Dann erhalten wir eine KNF für φ mit der SymPy-Funktion `to_cnf` und dem Aufruf

```
>>> to_cnf( phi )
```

Allerdings ist die Ausgabe einer möglichst kurzen KNF vorzuziehen. Dazu wähle den Aufruf

```
>>> to_cnf( phi , simplify=True )
```

3.4.2.1. Das KNF-Erfüllbarkeitsproblem

Definition 3.54. (**KNF-Erfüllbarkeitsproblem**).

KNF-SAT

Im KNF-Erfüllbarkeitsproblem (kurz: **KNF-SAT** für die englische Beschreibung Satisfiability) ist für eine KNF-Formel φ zu entscheiden, ob φ erfüllbar ist.

Frage: Wie „schwierig“ ist KNF-SAT?

Natürlich kann man das Erfüllbarkeitsproblem mit einer Wahrheitstafel lösen, indem man testet, ob es in der mit „φ“ beschrifteten Spalte mindestens eine 1 gibt. Wir erinnern uns aber an Satz 3.25: Die Wahrheitstafel einer aussagenlogischen Formel φ besitzt $2^{|\,\mathrm{Var}(\varphi)|}$ verschiedene Belegungen, ein verheerendes Ergebnis.

In der „Theoretischen Informatik 1“ wird gezeigt

Satz von Cook

Satz 3.55. (**Satz von Cook**).
KNF-SAT ist NP-vollständig.

Eine präzise Definition des Begriffs „NP-vollständig“ wird in der Veranstaltung „Theoretische Informatik 1“ gegeben. Grob gesagt bedeutet die NP-Vollständigkeit von KNF-SAT, dass es vermutlich kein *effizientes* Verfahren gibt, das KNF-SAT für alle Eingabeformeln löst. :-((

Ein Verfahren wird effizient genannt, wenn seine Laufzeit nur moderat mit der Länge der Formel wächst. Genauer: Die Laufzeit muss polynomiell in der Eingabelänge sein!

Die besten der so genannten SAT-Solver lösen KNF-SAT trotzdem für viele Eingabe-Formeln erstaunlich schnell.

Ich muss hoffen, dass ich für meine Eingabe-Formel ein Verfahren finde, dass meine Formel schnell genug löst. Aber es gibt vermutlich kein „einigermaßen“ schnelles Verfahren für alle Formeln!

Wie arbeiten denn diese SAT-Solver? Sie basieren auf dem Beweisverfahren der Resolution.

3.4.2.2. Resolution

In diesem Abschnitt stellen wir einen Disjunktionsterm $D = \ell_1 \vee \cdots \vee \ell_m$ als die Menge $\{\ell_1, \ldots, \ell_m\}$ seiner Literale dar.

Mit Hilfe der Resolutionsregel kann gezeigt werden, dass eine KNF nicht erfüllbar ist. In einigen Fällen gelingt ein wesentlich schnellerer Nachweis als mit Wahrheitstafeln möglich ist.

Definition 3.56. (**Resolution**). Resolution

(a) Sei X eine Variable, α', β' seinen Disjunktionsterme mit $\alpha' = \alpha \cup \{X\}$ und $\beta' = \beta \cup \{\neg X\}$. Dann besagt die **Resolutionsregel**, dass der Disjunktionsterm $\alpha \vee \beta$ abgeleitet werden darf. Resolutionsregel Man schreibt auch kurz

$$\frac{\alpha \cup \{X\},\ \beta \cup \{\neg X\}}{\alpha \cup \beta}$$

Wir sagen auch, dass $\alpha \cup \beta$ eine Anwendung der Resolutionsregel zu X ist. (Beachte, dass $\{\, \alpha \cup \{X\},\ \beta \cup \{\neg X\} \,\} \models \alpha \cup \beta$ gilt, die Resolutionsregel erlaubt also nur die Ableitung von semantischen Folgerungen.)

(b) Das Beweissystem $\mathfrak{R} = (A_{\mathfrak{R}}, S_{\mathfrak{R}})$ der Resolution besitzt keine Axiome, d.h. es gilt $A_{\mathfrak{R}} = \emptyset$. Die Resolutionsregel ist die einzige Schlussregel in $S_{\mathfrak{R}}$.

(c) Für eine Menge Φ von Disjunktionstermen und einen Disjunktionsterm χ führt man den Begriff

$$\Phi \vdash_{\mathfrak{R}} \chi$$

eines Resolutionsbeweises rekursiv wie folgt ein:

Basisregel: Für jedes $\alpha \in \Phi$ ist $\Phi \vdash_{\mathfrak{R}} \alpha$.

Rekursive Regel: Wenn $\Phi \vdash_{\mathfrak{R}} \alpha \cup \{X\}$ und $\Phi \vdash_{\mathfrak{R}} \beta \cup \{\neg X\}$, dann gilt auch $\Phi \vdash_{\mathfrak{R}} \alpha \cup \beta$.

(d) Ein Resolutionsbeweis für die Ableitung $\Phi \vdash_{\mathfrak{R}} \chi$ ist ein Tupel $(\chi_1, \ldots, \chi_i, \ldots, \chi_n)$ von Disjunktionstermen. Es muss gelten:

  (a) χ_n ist die Zielformel, d.h. es gilt $\chi_n = \chi$.

  (b) Für jedes i mit $1 \leqslant i \leqslant n$ ist entweder

  - χ_i eine Formel aus Φ oder
  - es gibt $1 \leqslant i_1 < i_2 < i$ und χ_i folgt aus χ_{i_1}, χ_{i_2} durch Anwendung der Resolutionsregel.

Beispiel 3.57. Wir möchten die „Transitivität“ der Implikation, also

$$\{X \to Y,\, Y \to Z\} \vdash_{\mathrm{ABS}} \{\neg X, Z\}$$

mit Hilfe der Resolution nachweisen. Zuerst beachten wir die Äquivalenzen $(X \to Y) \equiv \{\neg X, Y\}$ und $(Y \to Z) \equiv \{\neg Y, Z\}$. Wir setzen

$$\Phi := \{\, \{\neg X, Y\},\ \{\neg Y, Z\}\,\}$$

und müssen zeigen, dass $\Phi \vdash_{\mathfrak{R}} \chi$ mit $\chi := \{\neg X, Z\}$ gilt.

1. Die Formel $\chi_1 := \{\neg X, Y\}$ gehört zu Φ und ist deshalb in $\mathfrak{R}$ ableitbar.
2. Gleiches gilt für die Formel $\chi_2 := \{\neg Y, Z\}$ und deshalb ist auch χ_2 in $\mathfrak{R}$ ableitbar.
3. Jetzt erhalten wir $\chi_3 := \chi = \{\neg X, Z\}$ nach Anwendung der Resolutionsregel
$$\frac{\{\neg X, Y\},\ \{\neg Y, Z\}}{\{\neg X, Z\}}$$
auf χ_1 und χ_2.

□ Ende von Beispiel 3.57

Sei ϵ der leere Disjunktionsterm. In Definition 3.47 haben wir festgelegt, dass ϵ unerfüllbar ist. Diese Definition ist sinnvoll, denn keine Belegung kann einen Disjunktionsterm ohne Variablen erfüllen.

Wie kann man mit einem Resolutionsbeweis zeigen, dass eine Menge Φ von Disjunktionstermen unerfüllbar ist? Indem man die Ableitung $\Phi \vdash_{\mathfrak{R}} \epsilon$ herleitet.

Beispiel 3.58. Die Deutsche Bahn soll sich entscheiden, ob der Frankfurter Kopfbahnhof in einen Durchgangsbahnhof umgebaut werden soll. Dabei muss sie ihre Entscheidung auf die folgenden Fakten basieren:

1. Die Kunden der Bahn sind nicht **z**ufrieden, wenn
   - sich die **P**reise erhöhen: Wir erhalten die Implikation $P \to \neg Z$, bzw. den Disjunktionsterm $\{\neg P, \neg Z\}$,
   - oder sich die **F**ahrzeiten verlängern: Wir erhalten die Implikation $F \to \neg Z$, bzw. den Disjunktionsterm $\{\neg F, \neg Z\}$.
2. Wenn der Frankfurter Kopfbahnhof nicht in einen Durchgangsbahnhof umge**b**aut wird, verlängern sich die Fahrzeiten: Diesmal erhalten wir die Implikation $\neg B \to F$, bzw. den Disjunktionsterm $\{B, F\}$.
3. Der Bahnhof kann nur dann umgebaut werden, wenn die Fahrpreise erhöht werden: Also folgt $B \to P$, bzw. der Disjunktionsterm $\{\neg B, P\}$ folgt.

Die Bahn kann es niemandem recht machen, denn die Menge

$$\Phi := \Big\{\{\neg P, \neg Z\}\ \{\neg F, \neg Z\}, \{B, F\},\ \{\neg B, P\},\ \{Z\}\ \Big\}$$

ist unerfüllbar. Wie sieht ein Resolutionsbeweis aus?

1. Der Disjunktionsterm $\chi_1 := \{\neg P, \neg Z\}$ liegt in Φ genauso wie der Disjunktionsterm $\chi_2 := \{Z\}$. Folgere $\chi_3 := \{\neg P\}$ in einer Anwendung der Resolutionsregel.

2. Der Disjunktionsterm $\chi_4 := \{\neg F, \neg Z\}$ gehört zur Menge Φ. Zusammen mit $\chi_2 = \{Z\}$ folgt $\chi_5 := \{\neg F\}$ mit einer Anwendung der Resolutionsregel.

3. Der Disjunktionsterm $\chi_6 := \{\neg B, P\}$ gehört zur Menge Φ. Also folgt $\chi_7 := \{\neg B\}$ aus χ_6 und χ_3 durch eine Anwendung der Resolutionsregel.

4. Der Disjunktionsterm $\chi_8 := \{B, F\}$ gehört zur Menge Φ. Also folgt $\chi_9 := \{B\}$ durch eine Anwendung der Resolutionsregel aus χ_8 und χ_5.

5. Der leere Disjunktionsterm folgt durch Anwendung der Resolutionsregel aus χ_7 und χ_9: Die Menge Φ ist unerfüllbar!

□ Ende von Beispiel 3.58

Angenommen, Φ ist eine beliebige unerfüllbare Menge von Disjunktionstermen. Gibt es dann auch einen Resolutionsbeweis $\Phi \vdash_{\mathfrak{R}} \epsilon$?

Satz 3.59. (**Vollständigkeit der Resolution**)
Sei Φ eine Menge von Disjunktionstermen. Dann gilt

$$\bigwedge_{D \in \Phi} D \text{ ist unerfüllbar} \iff \Phi \vdash_{\mathfrak{R}} \epsilon.$$

Beweis: Sei Φ eine Menge von Disjunktionstermen. Wir geben ein Verfahren an, dass sukzessive Variablen entfernt, bis Erfüllbarkeit festgestellt wird oder der leere Disjunktionsterm abgeleitet wird.

(a) Setze $k := 0$ und $\Phi_0 := \Phi$.

(b) Wiederhole:

1. Entferne alle allgemeingültigen Disjunktionsterme D aus Φ_k.
(D heißt genau dann allgemeingültig, wenn $D = D' \cup \{X, \neg X\}$ gilt.)
   - Wenn $\Phi_k = \emptyset$, dann halte mit der Antwort „Φ ist erfüllbar“.
2. Wähle eine aussagenlogische Variable X, die in mindestens einem Disjunktionsterm in Φ_k vorkommt.
   - Wende, wann immer möglich, die Resolutionsregel zu X an, d.h. bestimme
     – $\Psi_1 := \{\alpha \cup \beta : \alpha \cup \{X\} \in \Phi_k, \beta \cup \{\neg X\} \in \Phi_k\}$
   - Nimm all diese Anwendungen der Resolutionsregel in Φ_k auf und entferne aus Φ_k alle Disjunktionsterme, die das Literal X, bzw. das Literal $\neg X$ enthalten. D.h. bestimme
     – $\Psi_2 := \{\alpha : X \in \mathrm{Var}(\alpha) \text{ und } \alpha \in \Phi_k\}$
     und setze $\Phi_{k+1} := (\Phi_k \cup \Psi_1) \setminus \Psi_2$.
     Achtung: Die Menge Φ_{k+1} nimmt möglicherweise gewaltig an Größe zu!
   - Wenn Φ_{k+1} jetzt die leere Klausel ϵ enthält, dann halte mit der Antwort „Φ ist unerfüllbar“. Sonst setze $k := k + 1$.

Das Verfahren terminiert, da nacheinander alle Variablen entfernt werden. Also zeige

$$\Phi_k \text{ erfüllbar} \iff \Phi_{k+1} \text{ erfüllbar.}$$

„$\Longrightarrow$“ $\mathcal{B}$ erfülle alle Disjunktionsterme in Φ_k. Ein Disjunktionsterm $D \in \Phi_{k+1}$ gehört entweder zu Φ_k, und wird dann natürlich von $\mathcal{B}$ erfüllt, oder wurde neu hinzugefügt. Im letzten Fall ist $D = \alpha \cup \beta$ und $\alpha \cup \{X\}$ wie auch $\beta \cup \{\neg X\}$ gehören zu Φ_k. Nach Annahme gilt $[\![\alpha \cup \{X\}]\!]^{\mathcal{B}} = [\![\beta \cup \neg X]\!]^{\mathcal{B}} = 1$ und $[\![\alpha \vee \beta]\!]^{\mathcal{B}} = 1$ folgt. Also erfüllt $\mathcal{B}$ alle Disjunktionsterme in Φ_{k+1}.

„$\Longleftarrow$“ $\mathcal{B}$ erfülle alle Disjunktionsterme in Φ_{k+1}. Es gelte $[\![X]\!]^{\mathcal{B}} = 0$, der Fall $[\![X]\!]^{\mathcal{B}} = 1$ verläuft analog. Sei $\mathcal{B}'$ die „Geschwisterbelegung“ mit $[\![X]\!]^{\mathcal{B}'} = 1$, aber $\mathcal{B}'$ stimme sonst mit $\mathcal{B}$ überein. Ein Disjunktionsterm $D \in \Phi_k$ gehört entweder zu Φ_{k+1}, und wird dann natürlich von $\mathcal{B}$ wie auch von $\mathcal{B}'$ erfüllt, oder D wurde aus Φ_k entfernt. Wenn $\mathcal{B}$ alle entfernten Disjunktionsterme erfüllt, dann ist Φ_k erfüllbar und das war zu zeigen. Also falsifiziere $\mathcal{B}$ den entfernten Disjunktionsterm D und $D = \alpha \cup \{X\}$ folgt.

Dann muss $\mathcal{B}$ (und damit auch $\mathcal{B}'$) alle Disjunktionsterme β mit $\beta \cup \{\neg X\} \in \Phi_k$ erfüllen, denn sonst gilt $[\![\alpha \cup \beta]\!]^{\mathcal{B}} = 0$ im Gegensatz zur Annahme. Aber dann folgt $[\![\alpha \cup \{X\}]\!]^{\mathcal{B}'} = 1 = [\![\beta \cup \{\neg X\}]\!]^{\mathcal{B}'}$ für alle entfernten Disjunktionsterme: $\mathcal{B}'$ erfüllt alle Terme in Φ_k und Φ_k ist erfüllbar. □

DPLL-Verfahren

Das im Beweis von Satz 3.59 vorgestellte Verfahren führt zu einem viel zu hohem Aufwand, da die Mengen Φ_k in Größe explodieren. Das **DPLL-Verfahren** von Davis, Putnam, Logemann und Loveland geht deshalb wie folgt vor.

1. Wenn Erfüllbarkeit festgestellt wird, dann halte mit der Antwort „Φ ist erfüllbar“.

2. Wenn Φ den leeren Disjunktionsterm enthält, dann halte mit der Antwort „Φ ist unerfüllbar“.

Unit-Resolution

3. „**Unit-Resolution**“: Wenn Φ für eine Variable X den Term $D = X$ enthält, dann entferne alle Disjunktionsterme aus Φ, die X enthalten und entferne jedes Auftreten von $\neg X$ in einem Term von Φ. (Die Setzung $X = 1$ ist erzwungen.)
Behandele den Fall $D = \neg X$ analog.

Pure Literal Rule

4. „**Pure Literal Rule**“: Wenn es eine Variable X gibt, so dass X nur positiv in Termen von Φ vorkommt, dann entferne alle Terme aus Φ, in denen X vorkommt. (Die Setzung $X = 1$ kann nicht falsch sein.)
Behandele den Fall, dass X nur negiert vorkommt, analog.

Choose Literal

5. „**Choose Literal**“. Eine Variable X wird ausgewählt.

Backtracking

6. „**Backtracking**“.
   - Rufe das Verfahren rekursiv für $\Phi' := \Phi \cup \{X\}$ auf.
   - Wenn die Antwort „unerfüllbar“ ist, dann rufe $\Phi' := \Phi \cup \{\neg X\}$ auf.

Zu den erfolgreichsten Implementierungen des DPLL-Verfahrens gehören Chaff (http://www.princeton.edu/~chaff/) und zChaff (http://www.princeton.edu/~chaff/zchaff.html). Die verschiedenen Implementierungen unterscheiden sich vor Allem in der Umsetzung der Schritte „Choose Literal“ und „Backtracking“.

3.5. Boolesche Funktionen

Was „steckt eigentlich hinter“ Formeln und Wahrheitstafeln?

Definition 3.60. Eine Funktion $f : \{0,1\}^n \to \{0,1\}$ heißt eine **boolesche Funktion**. boolesche Funktion

Hier sind Beispiele einfacher boolescher Funktionen.

Beispiel 3.61. Die boolesche Funktion

$$\mathrm{Und}_n : \{0,1\}^n \to \{0,1\}$$

nimmt genau dann den Wert 1 für Eingabe x an, wenn also x aus n Einsen besteht.

Beispiel 3.62. Die Paritätsfunktion

$$p_n : \{0,1\}^n \to \{0,1\}$$

nimmt genau dann den Wert 1 für Eingabe x an, wenn x eine ungerade Anzahl von Einsen besitzt. Das „Paritätsbit“ $p_n(x_1, \ldots, x_n)$ ändert sich also, wenn genau ein Bit „geflippt“ wird und wird deshalb zur Fehlererkennung eingesetzt. In Satz 4.13 wird gezeigt, dass

$$p_n(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$$

gilt. Da das XOR assoziativ ist – siehe Satz 3.34 – haben wir in der Darstellung von p_n keine Klammern einsetzen müssen.

Beispiel 3.63. Die Addition zweier n-Bit Zahlen $x = \sum_{i=0}^{n-1} x_i 2^i$ und $y = \sum_{i=0}^{n-1} y_i 2^i$ kann durch ein Tupel $(f_0, \ldots, f_n)$ boolescher Funktionen $f_i : \{0,1\}^{2n} \to \{0,1\}$ modelliert werden, wobei $f_i(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1})$ das $i$te Bit der Binärdarstellung der Summe $x + y$ ist. Analoges gilt natürlich auch für die Multiplikation. Der Entwurf von Schaltungen für Tupel boolescher Funktionen ist ein wichtiges Ziel der technischen Informatik.

Und was ist der Zusammenhang zwischen Formeln, Wahrheitstafeln und booleschen Funktionen?

Definition 3.64. (**Formeln und boolesche Funktionen**). Sei φ eine aussagenlogische Formel mit $\mathrm{Var}(\varphi) = \{X_1, \ldots, X_n\}$ und sei $f : \{0,1\}^n \to \{0,1\}$ eine boolesche Funktion.

Wir sagen, dass f die boolesche Funktion zu φ ist, bzw. dass φ zu f passt, genau dann, wenn

$$f(\,[\![X_1]\!]^{\mathcal{B}}, \ldots, [\![X_n]\!]^{\mathcal{B}}\,) = [\![\varphi]\!]^{\mathcal{B}}$$

für alle zu φ passenden Belegungen gilt.

Beispiel 3.65. Für eine Formel φ mit $\mathrm{Var}(\varphi) = \{\, X_1, \ldots, X_n \,\}$ definieren wir die boolesche Funktion $f_\varphi : \{0,1\}^n \to \{0,1\}$ durch

$$f_\varphi(x_1, \ldots, x_n) := [\![\varphi]\!]^{\mathcal{B}},$$

wobei $\mathcal{B} : \mathrm{Var}(\varphi) \to \{0,1\}$ die Belegung mit $[\![X_i]\!]^{\mathcal{B}} = x_i$ ist. Dann ist f_φ die boolesche Funktion zu φ.

Gibt es umgekehrt auch stets eine passende Formel φ_f für eine boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$? Zu f bauen wir eine Wahrheitstafel, die in Zeile $(x_1, \ldots, x_n) \in \{0,1\}^n$ den Wahrheitswert $f(x_1, \ldots, x_n)$ erhält. Für diese Wahrheitstafel konstruieren wir dann die kanonische DNF φ_f.

Zu jeder Formel φ gibt es somit eine boolesche Funktion f_φ zu φ und jede boolesche Funktion f hat eine passende Formel!

Satz 3.66 (Boolesche Funktionen und Formeln: Semantik und Syntax).

(a) Für jede boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ und für alle Variablen $X_1, \ldots, X_n$ gibt es eine passende aussagenlogische Formel φ_f mit $\mathrm{Var}(\varphi_f) = \{X_1, \ldots, X_n\}$

(b) Für jede aussagenlogische Formel φ mit $\mathrm{Var}(\varphi) = \{X_1, \ldots, X_n\}$ gibt es eine boolesche Funktion $f_\varphi : \{0,1\}^n \to \{0,1\}$ zu φ.

Und was ist der Unterschied zwischen einer Wahrheitstafel und einer booleschen Funktion? In einer Wahrheitstafel werden zuerst die aussagenlogischen Variablen benannt (z.B. seien dies $X_1, \ldots, X_n$) und ihre Reihenfolge festgelegt. Dann wird für jede Belegung $\mathcal{B} : \{X_1, \ldots, X_n\} \to \{0,1\}$ eine eigene Zeile angelegt und der Wahrheitswert

$$f([\![X_1]\!]^{\mathcal{B}}, \ldots, [\![X_n]\!]^{\mathcal{B}})$$

für $\mathcal{B}$ eingetragen. Wir nennen f die boolesche Funktion der Wahrheitstafel.

Die boolesche Funktion f weist jeder Zeile der Wahrheitstafel den Wahrheitswert der Zeile zu und modelliert damit die Wahrheitstafel. Die Wahrheitstafel tut allerdings ein klitzeklein wenig mehr, denn sie benennt auch die aussagenlogischen Variablen und gibt ihnen eine Reihenfolge. Die wesentlichen Eigenschaften einer Wahrheitstafel werden aber durch ihre boolesche Funktion ausgedrückt.

Zusammengefasst: Man kann boolesche Funktionen als die Bedeutung von Formeln auffassen und Formeln als eine spezielle Form der Darstellung einer booleschen Funktion. Schaltungen (bzw. Schaltkreise) in der technischen Informatik liefern eine weitere Darstellungsform. Wahrheitstafeln und boolesche Funktionen sind „im Wesentlichen“ identische Konzepte.

Bisher kennen wir nur die Formel $\varphi_n = \bigwedge_{i=1}^{n}(X_i \leftrightarrow Y_i)$, die DNFs mit 2^n, also mit exponentiell vielen Konjunktionstermen besitzt. Aber φ_n hat eine KNF mit nur $2n$ Disjunktionstermen. Haben Formeln immer eine „kleine“ DNF *oder* eine „kleine“ KNF?

Satz 3.67. Sei φ_n eine zur Paritätsfunktion p_n passende Formel. Jede DNF für φ_n hat mindestens $\mathbf{2^{n-1}}$ Konjunktionsterme, jede KNF hat mindestens $\mathbf{2^{n-1}}$ Disjunktionsterme.

Beweis: Angenommen wir wissen, dass alle Primimplikanten von φ_n Vollkonjunktionen sind. Dann kann Teil (c) des Satzes von Quine (Satz 3.43) angewandt werden: Die kanonische DNF für φ_n ist eine DNF mit der kleinstmöglichen Anzahl von Primimplikanten. Die kanonische DNF für φ_n hat einen Primimplikanten für jede 1-Zeile von φ_n. Es ist $p_n(x_1, \ldots, x_n) = 1$ genau dann, wenn $x = (x_1, \ldots, x_n)$ eine ungerade Anzahl von Einsen besitzt: Also hat φ_n genau 2^{n-1} 1-Zeilen und die kanonische DNF für φ_n hat genau 2^{n-1} Konjunktionsterme.

Warum hat jede KNF für φ_n mindestens 2^{n-1} Disjunktionsterme? Weil jede DNF für die Negation $\neg\varphi_n$ mindestens 2^{n-1} Primimplikanten besitzt.

Warum sind aber alle Primimplikanten von φ_n Vollkonjunktionen? Jeder Konjunktionsterm mit höchstens $n-1$ Literalen wird von zwei Belegungen erfüllt, die sich nur in dem Wahrheitswert

einer einzigen Variablen unterscheiden. Dann wird aber eine der beiden Belegungen eine gerade Anzahl von Einsen besitzen: Der Konjunktionsterm kann kein Implikant und erst recht kein Primimplikant von φ_n sein. □

Es gibt also Formeln, die sich weder durch DNFs noch durch KNFs kurz beschreiben lassen!

3.6. Zusammenfassung und Ausblick

Die Aussagenlogik spielt eine wichtige Rolle zum Beispiel in der künstlichen Intelligenz, um Schlüsse aus dem gesammelten Wissen ziehen zu können, oder in der technischen Informatik, um boolesche Funktionen beschreiben zu können.

Wir haben aussagenlogische Formeln rekursiv eingeführt und ihre Bedeutung festgelegt, wir haben also Syntax und Semantik aussagenlogischer Formeln definiert. Die Erfüllbarkeit oder Allgemeingültigkeit wie auch die semantische Folgerung oder Äquivalenz von Formeln lässt sich mit Wahrheitstafeln überprüfen. Allerdings besitzt die Wahrheitstafel einer Formel mit n Variablen 2^n Zeilen: Wahrheitstafeln für große Variablenzahlen lassen sich auch maschinell nicht mehr in den Griff bekommen.

Jede Formel φ besitzt eine disjunktive Normalform (DNF). Die kanonische DNF ist in vielen Fällen viel zu groß. Deshalb haben wir Implikanten und Primimplikanten eingeführt und im Satz von Quine festgestellt, dass es kleinstmögliche DNFs nur aus Primimplikanten gibt.

Möchten wir eine konjunktive Normalform (KNF) für φ erhalten, genügt es eine DNF D für die Negation von f zu bestimmen: $\neg D$ ist dann eine KNF für f.

In KNF-SAT wird gefragt, ob eine Eingabe-KNF erfüllbar ist. In der Veranstaltung „Theoretische Informatik 1“ wird gezeigt, dass KNF-SAT ein NP-vollständiges Problem ist und deshalb vermutlich nicht effizient gelöst werden kann. Wir haben aber mit dem Resolutionsverfahren und ihren Varianten eine Methode kennengelernt, die zwar nicht für alle KNFs Erfüllbarkeit in annehmbarer Rechenzeit feststellt, aber für viele interessante KNFs erfolgreich ist.

Wie gut haben Sie die Konzepte dieses Kapitels verstanden? Das folgende, möglicherweise auf Albert Einstein zurückgehende Rätsel ist ein guter Testfall. Hier ist die Beschreibung des „**Einsteinrätsels**“: Fünf Häuser mit fünf unterschiedlichen Farben stehen nebeneinander. In ihnen wohnen Menschen von fünf unterschiedlichen Nationalitäten, die fünf unterschiedliche Getränke trinken, fünf unterschiedliche Zigarettenmarken rauchen und fünf unterschiedliche Haustiere besitzen. Folgendes ist bekannt:

1) Der Brite lebt im roten Haus.
2) Die Schwedin hält sich einen Hund.
3) Der Däne trinkt gern Tee.
4) Das grüne Haus steht (direkt) links neben dem weißen Haus.
5) Die Person, die im grünen Haus wohnt, trinkt Kaffee.
6) Die Person, die Pall Mall raucht, hat einen Vogel.
7) Die Person, die im mittleren Haus wohnt, trinkt Milch.
8) Die Person, die im gelben Haus wohnt, raucht Dunhill.
9) Die Norwegerin lebt im ersten Haus.
10) Die Person, die Marlboro raucht, wohnt neben der Person mit der Katze.

11) Die Person mit dem Pferd lebt neben der Person, die Dunhill raucht.

12) Die Person, die Winfield raucht, trinkt gern Bier.

13) Die Norwegerin wohnt neben dem blauen Haus.

14) Der Deutsche raucht Rothmanns.

15) Die Person, die Marlboro raucht, lebt neben der Person, die Wasser trinkt.

Frage: Wem gehört der Fisch?

Sollte man das Rätsel als aussagenlogische Formel φ modellieren? Wenn ja, welche aussagenlogische Variablen sollte man verwenden? Hat φ eine besondere Form? Oder sollte man vielmehr auf die eigene Kreativität setzen?

Übrigens, Einstein soll der Ansicht gewesen sein, dass nur 2% der Bevölkerung dieses Rätsel lösen könnten.

3.7. Literaturhinweise zu Kapitel 3

Als vertiefende Lektüre seien die Kapitel 1 und 2 in [16] sowie die Einleitung und Kapitel 1 in [24] empfohlen. Einen umfassenden Überblick über die Rolle der Logik in der Informatik gibt [10]. Details zum Thema NP-Vollständigkeit und zum Satz von Cook finden sich in den Büchern [23, 27].

Quellennachweis: Teile dieses Kapitels orientieren sich an [9]. Die folgende Aufgabe 3.9 ist aus [16] entnommen. Beispiel 3.1 ist eine Variante einer Übungsaufgabe aus [16].

3.8. Übungsaufgaben zu Kapitel 3

Aufgabe 3.1. Betrachten Sie die folgenden Worte und beweisen Sie jeweils, dass das Wort gemäß Definition 3.3 zur Sprache AL gehört oder begründen Sie, warum das Wort nicht zu AL gehört.

(i) $\neg\Big((V_3 \wedge \neg\mathbf{0}) \to \big(V_0 \vee (\neg\neg V_1 \wedge V_4)\big)\Big)$

(ii) $(V_5 \leftrightarrow X) \wedge \big(V_{23} \to (V_1 \wedge \mathbf{0})\big)$

(iii) $\Big((V_{11} \leftarrow V_7) \vee \neg\neg V_5\Big)$

(iv) $\big((V_9 \vee \neg(\neg V_{42}) \vee \neg V_2) \to \mathbf{1}\big)$

(v) $((V_1 \wedge V_3) \leftrightarrow (\neg V_2 \vee V_3))$

(vi) $(V_3 \leftarrow (V_1 \to V_2))$

Aufgabe 3.2. Es sei die aussagenlogische Formel $\varphi := \neg(((V_1 \leftrightarrow \neg V_2) \vee (V_3 \wedge V_2)) \to V_1)$ gegeben.

(a) Beweisen Sie, dass $\varphi \in \mathrm{AL}$ gilt.

(b) Geben Sie $\mathrm{Var}(\varphi)$ an.

(c) Berechnen Sie für die Belegung $\mathcal{B} : \mathrm{Var}(\varphi) \to \{0,1\}$ mit $\mathcal{B}(V_1) = 0$, $\mathcal{B}(V_2) = 1$ und $\mathcal{B}(V_3) = 0$ den Wert $[\![\varphi]\!]^{\mathcal{B}}$ in nachvollziehbaren Schritten analog zu Beispiel (3.18) aus der Vorlesung.

(d) Geben Sie den Syntaxbaum der Formel φ an.

Aufgabe 3.3. Oftmals stolpert man in der Literatur über eine aussagenlogische Formel, die in einer vereinfachten Notation angegeben wird, d.h. die Syntax der Formel entspricht nicht der im Skript definierten Syntax. Wenn eine vereinfachte Notation verwendet wird, muss natürlich sichergestellt werden, dass die Bedeutung einer Formel „eindeutig“ ist – andernfalls ist die Formel unbrauchbar.
In dieser Aufgabe erforschen wir, wie weit man bei der Vereinfachung der Notation gehen darf. Wir nennen eine Zeichenkette φ

- eine *Formel*, wenn $\varphi \in \mathrm{AL}$ gilt, wobei AL wie in Kapitel 3.2 definiert ist.
- eine *Fast-Formel*, wenn alle Klammerungen von φ semantisch äquivalent sind: Eine *Klammerung* $k(\varphi)$ ist eine Formel, die aus φ erzeugt werden kann, indem zusätzliche Klammern (aber keine Variablen, Junktoren, etc.) in φ eingesetzt werden.
- eine *Nicht-Formel*, wenn φ keine Formel und keine Fast-Formel ist.

Klassifizieren Sie die folgenden Zeichenketten jeweils als Formel, als Fast-Formel bzw. als Nicht-Formel. Geben Sie für jede Formel den Syntaxbaum an. Weisen Sie für jede Fast-Formel die Äquivalenz aller Klammerungen nach. Zeigen Sie auch für jede Nicht-Formel, dass Ihre Antwort korrekt ist.

(a) $(V_3 \vee ((\neg V_1 \vee V_2) \to (\neg V_2 \wedge V_3)))$

(b) $V_1 V_2$

(c) $V_1 \to V_2 \to V_3$

(d) $(V_1 \vee V_2) \wedge (V_1 \vee V_3) \wedge (V_2 \vee V_3)$

Aufgabe 3.4. Geben Sie für jede der folgenden aussagenlogischen Formeln an, ob sie erfüllbar, unerfüllbar und/oder allgemeingültig ist.

(a) $(V_0 \vee \mathbf{0})$

(b) $(V_0 \wedge \neg V_1)$

(c) $\big(V_0 \leftrightarrow (\mathbf{1} \to V_0)\big)$

(d) $\big(V_0 \leftrightarrow (V_0 \to \mathbf{0})\big)$

(e) $\Big(V_1 \vee \big((V_0 \wedge V_1) \to V_2\big)\Big)$

(f) $\big((V_0 \to V_1) \leftrightarrow (\neg V_1 \to \neg V_0)\big)$

(g) $\big((V_0 \vee \neg V_1) \leftrightarrow V_2\big)$

(h) $\big(\neg V_0 \to (V_0 \to V_1)\big)$

(i) $\big(V_0 \wedge (V_0 \to \neg V_0)\big)$

(j) $\Big((V_0 \to V_1) \leftrightarrow \big((V_0 \wedge \neg V_1) \to \mathbf{0}\big)\Big)$

(k) $((V_2 \wedge \mathbf{1}) \vee \mathbf{0})$

(l) $(V_1 \to (V_2 \to V_1))$

(m) $(\neg(V_1 \to \mathbf{0}) \to V_1)$

(n) $((\mathbf{1} \to V_1) \wedge (V_1 \to \mathbf{0}))$

(o) $\psi_n := \bigwedge_{i=1}^{n} (V_i \leftrightarrow V_{2i})$ für $n \in \mathbb{N}$ mit $n \geqslant 2$

Aufgabe 3.5. Geben Sie für jede der folgenden aussagenlogischen Formeln φ_i an, ob sie

- allgemeingültig,
- unerfüllbar oder
- sowohl erfüllbar als auch falsifizierbar

ist. Geben Sie für jede Formel, die erfüllbar und falsifizierbar ist, sowohl eine erfüllende als auch eine falsifizierende Belegung an.

(a) $\varphi_1 := (\mathbf{1} \to P) \wedge (P \to \mathbf{0})$

(b) $\varphi_2 := (A \to B) \vee (B \to A)$

(c) $\varphi_3 := (\mathbf{1} \vee \mathbf{0}) \oplus (\mathbf{1} \wedge \mathbf{0})$

(d) $\varphi_4 := (\neg A \leftrightarrow B) \oplus \neg(A \leftrightarrow B)$

(e) $\varphi_5 := \big(A \to (A \to B)\big) \to (A \to B)$

(f) $\varphi_6 := \big(A \to \big(A \to (B \to A)\big)\big) \to B$

(g) $\varphi_7 := (B \wedge B) \oplus (B \wedge R)$

(h) $\varphi_8 := \mathbf{1} \to \big((X \to Y) \leftrightarrow (\neg Y \to \neg X)\big)$

(i) $\varphi_9 := \neg(x \vee y) \vee (\neg\neg x \wedge y)$

(j) $\varphi_{10} := \neg\Big((X \to Y) \leftrightarrow \big((X \wedge \neg Y) \to \mathbf{0}\big)\Big)$

(k) $\varphi_{11} := (\neg A \leftrightarrow B) \oplus \neg(A \leftrightarrow B)$

(l) $\varphi_{12} := \neg(X \to Z) \leftrightarrow (\neg Z \to \neg X)$

Aufgabe 3.6. Bestimmen Sie für jede der folgenden aussagenlogischen Formeln ψ_i die Menge aller erfüllenden Belegungen $\mathcal{B}$ mit $\mathrm{Def}(\mathcal{B}) = \mathrm{Var}(\psi_i)$.

(a) $\psi_1 := (\mathbf{1} \wedge B) \leftrightarrow (\mathbf{0} \vee A)$

(b) $\psi_2 := X \leftrightarrow (Y \leftrightarrow (Z \leftrightarrow W))$

(c) $\psi_3 := (A \to B) \wedge (B \to C) \wedge (C \to A)$

(d) $\psi_4 := \bigwedge_{i=1}^{n} \big(V_i \to V_{i+1}\big)$ für $n \in \mathbb{N}_{>0}$

Aufgabe 3.7. Oft genügen zur Evaluierung bzw. zum Überprüfen der Erfüllbarkeit einer Formel bereits einige wenige belegte Variablen. Aus diesem Grund definieren wir *partielle Evaluierungen* von Belegungen:

Definition. Seien $\mathcal{B}_1, \mathcal{B}_2$ Belegungen mit $\mathrm{Def}(\mathcal{B}_1) \subseteq \mathrm{Def}(\mathcal{B}_2)$ und $\mathcal{B}_1(V) = \mathcal{B}_2(V)$ f.a. $V \in \mathrm{Def}(\mathcal{B}_1)$. Dann heißt $\mathcal{B}_2$ *Erweiterung* von $\mathcal{B}_1$. Für eine Formel $\varphi \in \mathrm{AL}$ definieren wir den *partiellen Wahrheitswert* $[\![\varphi]\!]^{\mathcal{B}}$ als

$$[\![\varphi]\!]^{\mathcal{B}} = \begin{cases} 1 & \text{falls } [\![\varphi]\!]^{\mathcal{B}'} = 1 \text{ für alle zu } \varphi \text{ passenden Erweiterungen } \mathcal{B}' \text{ von } \mathcal{B}, \\ 0 & \text{falls } [\![\varphi]\!]^{\mathcal{B}'} = 0 \text{ für alle zu } \varphi \text{ passenden Erweiterungen } \mathcal{B}' \text{ von } \mathcal{B}, \\ * & \text{sonst.} \end{cases}$$

Dabei steht der Stern $*$ intuitiv für „noch nicht entschieden“. Beachten Sie, dass wir hier – anders als in Def 3.11 – nicht verlangen, dass die Belegung $\mathcal{B}$ passend zu φ ist. Wie gewohnt nennen wir eine Belegung $\mathcal{B}$ mit $[\![\varphi]\!]^{\mathcal{B}} = 1$ *erfüllend* und eine Belegung mit $[\![\varphi]\!]^{\mathcal{B}} = 0$ *falsifizierend*.

Beispiel: Betrachte die Belegung $\mathcal{B} : \{V_1\} \to \{0, 1\}$ mit $\mathcal{B}(V_1) = 1$. Dann ist

$$[\![(V_1 \vee V_2)]\!]^{\mathcal{B}} = 1, \ [\![(\neg V_1 \wedge V_3)]\!]^{\mathcal{B}} = 0 \ \text{ und } \ [\![(V_1 \oplus V_4)]\!]^{\mathcal{B}} = *$$

Bestimmen Sie für die folgenden Formeln ψ_i jeweils erfüllende Belegungen $\mathcal{B}$ mit möglichst kleinem Definitionsbereich (d. h. möglichst wenigen belegten Variablen).

(a) $\psi_1 := (A \vee B \vee C \vee \neg D) \wedge B$

(b) $\psi_2 := \big((K \to L) \to M\big) \to N$

(c) $\psi_3 := \neg(E \leftrightarrow F \leftrightarrow G) \vee \neg H \vee (E \oplus F \oplus G)$

(d) $\psi_4 := P \oplus Q \oplus R \oplus S$

(e) $\psi_5 := (X \wedge \neg Z) \to \big((Y \vee Z) \oplus \neg(\neg W \leftrightarrow Y)\big)$

Aufgabe 3.8. Vor Ihnen stehen drei Kisten, von denen Sie eine auswählen dürfen. In genau einer der Kisten befindet sich Gold, in den beiden anderen befindet sich nur Stroh. Die Kisten haben die folgenden Beschriftungen:

- Kiste 1: „In dieser Kiste befindet sich Stroh.“
- Kiste 2: „In dieser Kiste befindet sich Gold.“
- Kiste 3: „In Kiste 2 befindet sich Stroh.“

Sie dürfen nur eine Kiste öffnen. Welche Kiste öffnen Sie, um das Gold zu erlangen?

Aufgabe 3.9. Schon kurz nach der Geburt von Herakles und Eurystheus entstand ein Streit, wer von den beiden der rechtmäßige Herrscher sei. Dazu wurden die drei bekanntesten Orakel Griechenlands befragt.

Das Ammonion gab bekannt, dass die Orakelsprüche aus Klaros grundsätzlich falsch seien. Ebenso ließ das Orakel aus Klaros verlauten, dass die Orakelsprüche aus Delphi samt und sonders unzutreffend seien. Das Orakel aus Delphi jedoch behauptete, sowohl die Sprüche des Ammonions als auch die des Orakels in Klaros seien unwahr.

Wem sollen die Griechen nun glauben?

(a) Zerlegen Sie den obigen Text in atomare Aussagen und geben Sie eine aussagenlogische Formel φ an, die das im Text zusammengefasste Wissen repräsentiert (ähnlich wie in den Beispielen 3.1, 3.9 und 3.23).

(b) Geben Sie für Ihre Formel φ aus (a) eine Belegung $\mathcal{B}$ an, die besagt, dass das Ammonion die Wahrheit sagt und die beiden anderen Orakel lügen. Erfüllt $\mathcal{B}$ die Formel φ?

(c) Welchen Orakeln können die Griechen glauben, welchen nicht? Falls es mehrere Möglichkeiten gibt, geben Sie alle an.

Aufgabe 3.10. USA, 4. November 2008. Vor einem Wahllokal befragt ein Journalist vier Freunde A, B, C und D, die gerade das Wahllokal verlassen haben, wie sie gewählt haben. A sagt: „Falls B für Obama gestimmt hat, dann haben auch C und D für Obama gestimmt.“ B sagt: „A hat auf keinen Fall für Obama gestimmt, aber D.“ C sagt: „B hat nur dann für McCain gestimmt, wenn A für Obama gestimmt hat.“ D sagt schließlich: „Wenn C für Obama gestimmt hat, dann hat A für McCain oder B für Obama gestimmt.“ Wir nehmen an, dass jeder die Wahrheit gesagt und entweder Obama oder McCain gewählt hat.

(a) Zerlegen Sie den obigen Text in atomare Aussagen und geben Sie eine aussagenlogische Formel φ an, die das im Text zusammengefasste Wissen repräsentiert (ähnlich wie in den Beispielen 3.1, 3.9 und 3.23).

(b) Geben Sie für Ihre Formel φ aus (a) eine Belegung $\mathcal{B}$ an, die besagt, dass A, B und C Obama gewählt haben und D für McCain gestimmt hat. Erfüllt $\mathcal{B}$ die Formel φ?

(c) Wen haben A, B, C und D jeweils gewählt? Falls es mehrere Möglichkeiten gibt, geben Sie alle an.

Aufgabe 3.11. Auf der Insel Wafa leben zwei Stämme: Die Was, die immer die Wahrheit sagen, und die Fas, die immer lügen. Ein Reisender besucht die Insel und kommt mit drei Einwohnern A, B, C ins Gespräch. Der Reisende schreibt daraufhin folgende atomare Aussagen in sein Notizbuch:

- X_A: A sagt die Wahrheit
- X_B: B sagt die Wahrheit

- X_C: C sagt die Wahrheit

(a) Sei $\mathcal{B}\colon \{X_A, X_B, X_C\} \to \{0,1\}$ die Belegung mit $\mathcal{B}(X_A) = 1$, $\mathcal{B}(X_B) = 0$ und $\mathcal{B}(X_C) = 0$. Beschreiben Sie umgangssprachlich, welcher Sachverhalt durch die Belegung $\mathcal{B}$ ausgedrückt wird. Was folgt daraus über die Stammesangehörigkeit der drei Einwohner A, B und C?

Die Informationen, die der Reisende im Gespräch erhalten hat, fasst er durch folgende aussagenlogische Formeln zusammen:

- $\varphi_A := \big(X_A \leftrightarrow (\neg X_B \vee \neg X_C)\big)$
- $\varphi_B := \big(X_B \leftrightarrow (X_A \to X_C)\big)$
- $\varphi_C := \big(X_C \leftrightarrow (\neg X_B \to X_A)\big)$

Er merkt an, dass die durch $\varphi_A, \varphi_B, \varphi_C$ formalisierten Aussagen der Wahrheit entsprechen.

(b) Beschreiben Sie umgangssprachlich, was jede der Formeln $\varphi_A, \varphi_B, \varphi_C$ aussagt.

(c) Zu welchen Stämmen gehören A, B und C?

Aufgabe 3.12. Die Zwergenseherin Bh'èrtlân hatte eine Vision, in der ihr die Bedeutung der geheimen Runen ihrer Ahnen ($\neg, \vee, \wedge, \to, \leftrightarrow$ und $\oplus$) offenbart wurde. Somit war sie in der Lage, die Legende einer uralten Karte zu entziffern, auf der der Weg zu einem unermesslichen Zwergenschatz eingezeichnet ist. Leider liegt der Schatz in einer Höhle weit unter dem mächtigen Berg Erebor und wird von einem gefährlichen Drachen bewacht. Manchmal verlässt der Drache jedoch die Höhle, um Feuer zu spucken, tapfere Krieger zu fressen und weitere Schätze zu rauben. Bh'èrtlâns Übersetzung der uralten Karte ergab die folgenden Aussagen:

1. „Wenn der Drache in der Höhle liegt, dann wird der Schatz nicht erobert."
2. Der Sage nach bringen feuerfeste Schilde Glück oder Gefahr. „Tragt ihr feuerfeste Schilde, so gilt: Ihr erobert den Schatz oder der Drache liegt in der Höhle, aber auf keinen Fall wird beides eintreten."
3. „Der Schatz wird erobert oder der Drache liegt in der Höhle."

(a) Übersetzen Sie die Aussagen 1 bis 3 in aussagenlogische Formeln φ_1, φ_2 und φ_3. Benutzen Sie dazu die aussagenlogischen Variablen **D**, **F** und **S** mit der Bedeutung, dass der **D**rache in der Höhle liegt, **F**euerfeste Schilde getragen werden und der **S**chatz erobert wird.

(b) Geben Sie eine aussagenlogische Formel φ an, die ausdrückt, dass die Aussagen 1 bis 3 gelten.

(c) Stellen Sie eine Wahrheitstafel für φ auf. Benutzen Sie für die Kopfzeile der Wahrheitstafel folgende Vorlage:

D	**F**	**S**	φ_1	φ_2	φ_3	φ
⋮	⋮	⋮	⋮	⋮	⋮	⋮

(d) Geben Sie für Ihre Formel φ aus b) alle erfüllenden Belegungen $\mathcal{B} : \{\mathbf{D}, \mathbf{F}, \mathbf{S}\} \to \{0,1\}$ an, für die der Schatz erobert werden kann.

(e) Gammli fragt: „Feuerfeste Schilde spielen doch überhaupt keine Rolle, oder? Bh'èrtlân hat wohl zu lange am Hochofen geschnüffelt!"

Zeigen Sie, dass die Vermutung des Zwerges Gammli richtig ist, indem Sie eine aussagenlogische Formel ψ mit $\psi \equiv \varphi$ und $\mathrm{Var}(\psi) \subseteq \{\mathbf{D}, \mathbf{S}\}$ angeben.

Aufgabe 3.13. An einem verregneten Morgen im August wird der Schlossherr Graf von Modenburg leblos mit tiefen Stichwunden in seinem Schlafgemach aufgefunden. Dem eilig herbeigeeilten Kriminalinspektor ist sofort klar, dass es sich um einen kaltblütigen Mord handeln muss. Da die im Kampf zertrümmerte Taschenuhr des ermordeten Grafen um 19:03 Uhr stehen geblieben ist, kann der Inspektor rekonstruieren, dass der Mord am vorigen Abend gegen 19 Uhr geschehen ist. Der Kreis der Verdächtigen lässt sich schnell auf die vier Bediensteten reduzieren, die sich zur Tatzeit auf dem Schlossgelände aufhielten: der Gärtner, die Köchin, der Butler und die Magd.

Noch ist unklar, wer die Tat begangen hat. Oder sind vielleicht sogar mehrere Personen in den Mord verstrickt? Der Inspektor setzt seine Ermittlungen fort und fördert die folgenden Fakten zu Tage:

I. Die Köchin behauptet, sie habe zwischen 18 und 20 Uhr in der Küche das Essen für den kommenden Tag vorbereitet. Währenddessen habe sie mit absoluter Sicherheit den Gärtner beim Schneiden der Rosen im Garten vor dem Küchenfenster beobachtet. Falls sie unschuldig ist – und ihre Aussage somit glaubwürdig ist –, muss auch der Gärtner unschuldig sein.

II. Auf dem Flur vor dem Schlafgemach wurden schlammige Fußspuren entdeckt, die nur zum Gärtner oder zur Köchin gehören können. (Schlammige Füße holt man sich nur im Garten oder im Kartoffelkeller!) Mindestens eine(r) der beiden muss am Mord beteiligt gewesen sein.

III. Eine Nachbarin hat aus ihrem Fenster beobachtet, wie um 18:30 Uhr genau zwei Personen den Schlossherren in seinem Schlafgemach aufgesucht haben, und zwar ein Mann und eine Frau. Bei diesen beiden Personen muss es sich um die Täter handeln.

IV. Ein weiterer Zeuge sagt aus, zur Tatzeit genau eine Gestalt im Salon beim Abstauben der Vorhänge und Polieren des Silberbestecks gesehen zu haben. Somit kann diese Person den Mord nicht begangen haben. Da sonst niemand Zugang zum Salon hat, muss es sich bei dieser Person um den Butler oder die Magd handeln. Allerdings behaupten beide, zur fraglichen Zeit im Salon gewesen sein. Eine der beiden Personen lügt und ist in den Mord verstrickt.

Lösen Sie den Fall!

a) Stellen Sie aussagenlogische Formeln $\varphi_{\mathrm{I}}, \ldots, \varphi_{\mathrm{IV}}$ auf, welche die ermittelten Fakten widerspiegeln. Benutzen Sie die aussagenlogischen Variablen **B** („der Butler ist schuldig“), **G** („der Gärtner ist schuldig“), **K** („die Köchin ist schuldig“) und **M** („die Magd ist schuldig“).

b) Konstruieren Sie dann eine Formel φ, die ausdrückt, dass alle vier Fakten gelten.

c) Bestimmen Sie alle erfüllenden Belegungen von φ. Wer kommt als Mörder in Frage?

Erklären Sie, warum $\varphi_{\mathrm{I}}, \ldots, \varphi_{\mathrm{IV}}$ die Faktenlage exakt wiedergeben. Überprüfen Sie am Ende, ob Ihre Antwort wirklich plausibel ist. Schließlich wollen Sie wohl kaum Unschuldige verurteilen oder Mörder frei herumlaufen lassen!

Aufgabe 3.14. Im Koprolu-Sektor des Shylmagoghnar-Systems liegt P3X-888, Heimatplanet der Gelgameks. Um ihre Unobtanium-Exportwirtschaft ausdehnen zu können, beantragt die planetare Regierung der Gelgameks eine Aufnahme in die Merkantile Allianz für Fortschritt und Entwicklung im All (MAFEA). Nachdem die Inspekteure der MAFEA den Planeten P3X-888 ausgiebig unter die Lupe genommen haben, wird ein Katalog von Auflagen formuliert:

A. Der Sklavenaufstand in den Naquadah-Minen von Vendor muss niedergeschlagen werden oder die usurpatorischen Umtriebe des Raumpatrouillen-Offiziers Gorn müssen beendet werden.

B. Mit den Namdalorianern muss ein Waffenstillstand ausgehandelt werden oder die Quintronen-Technologie muss erforscht werden.

C. Nur wenn die usurpatorischen Umtriebe Gorns beendet sind, darf die Quintronen-Technologie erforscht werden.

Die Gelgameks müssen *mindestens zwei* der drei Auflagen erfüllen, um in die MAFEA aufgenommen zu werden.

Verwenden Sie im Folgenden die aussagenlogischen Variablen

N für „die Namdalorianer willigen in einen Waffenstillstand ein“,

U für „die usurpatorischen Umtriebe Gorns werden beendet“,

S für „der Sklavenaufstand in den Naquadah-Minen von Vendor wird niedergeschlagen“ ,

Q für „die Quintronen-Technologie wird erforscht“.

a) Stellen Sie Formeln $\varphi_{\mathrm{A}}, \varphi_{\mathrm{B}}, \varphi_{\mathrm{C}}$ auf, welche die drei Auflagen A, B, C widerspiegeln. Stellen Sie dann eine Formel φ_{MAFEA} auf, die ausdrückt, dass die Gelgameks in die MAFEA aufgenommen werden.

Um in Erfahrung zu bringen, wie die Auflagen am besten zu erfüllen sind, beruft die Gelgamekanische Regierung die fünf W'illders der Glondog von Gaggen-Thor in den Forschungsrat. Nach Analyse aller vorliegenden Daten kommt der Rat zu folgendem Ergebnis:

I. Der Sklavenaufstand kann genau dann niedergeschlagen werden, wenn entweder die Namdalorianer einem Waffenstillstand zustimmen oder die usurpatorischen Umtriebe Gorns beendet werden.

II. Wenn der Sklavenaufstand nicht niedergeschlagen wird, kann die Quintronen-Technologie nicht erforscht werden[3].

III. Entweder beendet Gorn seine ursupatorischen Umtriebe oder die Quintronen-Technologie wird erforscht.

IV. Die Namdalorianer werden einem Waffenstillstand niemals zustimmen.

b) Stellen Sie aussagenlogische Formeln $\psi_{\mathrm{I}}, \ldots, \psi_{\mathrm{IV}}$ auf, welche die vom Forschungsrat ermittelten Fakten widerspiegeln.

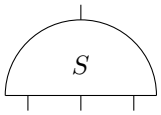
c) Was drückt die Formel $\chi := \varphi_{\mathrm{MAFEA}} \wedge \psi_{\mathrm{I}} \wedge \psi_{\mathrm{II}} \wedge \psi_{\mathrm{III}} \wedge \psi_{\mathrm{IV}}$ aus? Eine umgangssprachliche Erklärung genügt.

d) Bestimmen Sie alle erfüllenden Belegungen von χ, falls solche existieren. Können die Gelgameks der MAFEA beitreten? Falls ja, wie können sie dies bewerkstelligen? Falls nein, begründen Sie! Sie können beispielsweise eine Wahrheitstafel zur Begründung verwenden.

Aufgabe 3.15. Ein Schaltkreis mit n Eingängen und einem Ausgang ist ein elektronischer Baustein, der eine Funktion $f : \{0,1\}^n \to \{0,1\}$ berechnet. Dabei gilt für jede Komponente i der n Komponenten des Eingabetupels, dass sie genau dann 1 ist, falls am Schaltkreis am i-ten Eingang Spannung anliegt, und der Funktionswert von f ist genau dann 1, falls am Ausgang Spannung anliegt.

[3] Denn Quintronen bestehen zu mindestens 40% aus Naquadah!

Wir betrachten den Schaltkreis S mit 3 Eingängen. Die Eingänge des Schaltkreises sind der Reihe nach von 1 bis 3 durchnummeriert. Über den Schaltkreis S ist nun Folgendes bekannt:



Am Ausgang liegt Spannung an, falls jede der folgenden drei Eigenschaften erfüllt ist:

I. Aus der Tatsache, dass an dem ersten Eingang genau dann Spannung anliegt wenn am zweiten Eingang Spannung anliegt, folgt die Tatsache, dass am dritten Eingang Spannung anliegt,

II. Es gilt, am dritten Eingang liegt keine Spannung an oder am zweiten Eingang liegt keine Spannung an oder am ersten Eingang liegt Spannung an.

III. Es gilt, am dritten Eingang liegt keine Spannung an, oder falls am ersten Eingang Spannung anliegt, dann auch am zweiten.

(a) Modellieren Sie das Verhalten des Schaltkreises S als aussagenlogische Formel φ, welche genau dann wahr ist, falls am Schaltkreis am Ausgang Spannung anliegt. Benutzen Sie hierfür die drei atomaren Aussagen X_1, X_2 und X_3, wobei die atomare Aussage X_i genau dann erfüllt ist, falls am i-ten Eingang Spannung anliegt. Weiterhin soll die Formel φ eine Konjunktion von drei Teilformeln φ_1, φ_2 und φ_3 sein, wobei φ_1 die erste Eigenschaft des Schaltkreises modelliert. φ_2 die zweite und φ_3 die dritte.

(b) Stellen Sie eine Wahrheitstabelle für φ auf.

(c) Welche Funktion berechnet der Schaltkreis? D.h., für welche Art von Eingaben ist die Ausgabe positiv bzw. liegt Spannung am Ausgang an?

Aufgabe 3.16. Es ist ein bisher gut gehütetes Geheimnis, dass Lummerland Teil der Euro-Zone ist.[4] Aber ähnlich wie bei anderen Ländern auch haben die Turbulenzen an den Finanzmärkten den Staatshaushalt Lummerlands in arge Bedrängnis gebracht. Die Situation ist so ernst, dass Staatsoberhaupt König Alfons der Viertel-vor-Zwölfte die Europäische Zentralbank EZB um einen Kredit bitten muss. Doch die EZB ist streng und vergibt den Kredit nur, wenn die folgenden Anforderungen erfüllt sind:

I: Wenn die Ausgaben für Bildung nicht erhöht werden, müssen die Banken stärker kontrolliert werden.

II: Wenn Staatseigentum verkauft wird oder die Steuern gesenkt werden, dann dürfen die Ausgaben für Bildung nicht erhöht werden.

III: Die Banken werden genau dann stärker kontrolliert, wenn die Ausgaben für Bildung erhöht werden und die Steuern nicht gesenkt werden.

(a) Geben Sie für jede der Anforderungen I, II und III eine aussagenlogische Formel an, die die jeweilige Anforderung widerspiegelt (ähnlich zu Beispiel 3.10 und 3.9). Benutzen Sie dafür die atomaren Aussagen S (die Steuern werden gesenkt), B (die Ausgaben für Bildung werden erhöht), V (Staatseigentum wird verkauft) und K (die Banken werden stärker kontrolliert).

[4] Dies ist so natürlich nicht ganz richtig. Richtig ist, dass uns Lummerland und König Alfons der Viertel-vor-Zwölfte aus dem Kinderbuch „Jim Knopf und Lukas der Lokomotivführer“ von Michael Ende bekannt sind. Uns ist auch nicht bekannt, ob Rating-Agenturen die Kreditwürdigkeit Lummerlands anzweifeln.

(b) Stellen Sie eine aussagenlogische Formel φ auf, die die atomaren Aussagen S, B, V und K benutzt und die widerspiegelt, dass alle Anforderungen gleichzeitig erfüllt sein müssen.

(c) Geben Sie für Ihre Formel φ aus (4.4) eine Belegung an, die besagt, dass die Steuern gesenkt werden, die Ausgaben für Bildung sich nicht erhöhen, Staatseigentum verkauft wird und die Banken stärker kontrolliert werden. Erfüllt diese Belegung die Formel φ?

(d) Welche Maßnahmen genau muss König Alfons der Viertel-vor-Zwölfte aus den Möglichkeiten Steuersenkung, Bildungsausgabenerhöhung, Verkauf des Staatseigentums und Verstärkung der Bankenkontrolle treffen und welche muss er unterlassen, um allen Anforderungen der EZB gerecht zu werden?

Aufgabe 3.17. Oft ist es nicht praktikabel, (Eigenschaften von) Formeln mithilfe von Wahrheitstafeln zu erkunden: Die Wahrheitstafel einer gegebenen Formel ist mitunter so groß, dass sie per Hand – oder sogar maschinell – nicht schnell genug verarbeitet werden kann.

Beantworten Sie die Fragen ohne Verwendung von Wahrheitstafeln. Begründen Sie jeweils Ihre Antwort, indem Sie z.B. eine äquivalente Umformung vornehmen oder logische Schlüsse ziehen.

(a) Sind die folgenden Formeln erfüllbar?

a) $\Big(\big((A \to B) \to (C \to D)\big) \to C\Big) \wedge C \wedge D \wedge E \wedge F$

b) $\Big(\big((A \to B) \to (C \to D)\big) \to C\Big) \wedge \neg C \wedge D \wedge E \wedge F$

(b) Sei $n \in \mathbb{N}$ mit $n \geqslant 3$ beliebig. Wir suchen nach erfüllenden Belegungen $\mathcal{B} : \{V_1, \ldots, V_n\} \to \{0, 1\}$ für die folgenden Formeln auf den Variablen $V_1, \ldots, V_n$.

a) Geben Sie alle erfüllenden Belegungen der Formel $\bigwedge_{i=1}^{n-1}(V_i \oplus V_{i+1})$ an.

b) Geben Sie alle erfüllenden Belegungen der Formel $\bigwedge_{i,j\in\{1,\ldots,n\}}(V_i \oplus V_j)$ an.

(c) Gilt die folgende Äquivalenz? Begründen Sie Ihre Antwort.

$$(V_n \to V_1) \wedge \left(\bigwedge_{i=1}^{n-1}\Big(\neg V_i \vee V_{i+1}\Big)\right) \stackrel{?}{\equiv} \bigwedge_{j,k\in\{1,\ldots,n\}} (V_j \leftrightarrow V_k)$$

Aufgabe 3.18. Sei $\mathcal{O} := \{\mathbf{1}, \mathbf{0}, \neg, \vee, \wedge, \to, \leftrightarrow, \oplus\}$ die Menge aller aussagenlogischen *Konstanten* und *Junktoren*. Für jedes $S \subsetneq \mathcal{O}$ sei $\mathrm{AL}_S \subseteq \mathrm{AL}$ die Menge aller syntaktisch korrekten aussagenlogischen Formeln, in denen – neben Klammern und den Variablen – nur die Konstanten und Junktoren aus S vorkommen. Wir nennen S *vollständig*, falls wir für jedes $\varphi \in \mathrm{AL}$ ein $\varphi' \in \mathrm{AL}_S$ mit $\varphi \equiv \varphi'$ finden können.

Beispielsweise ist $S = \{\neg, \vee, \wedge\}$ vollständig[5]. Um die Vollständigkeit einer Teilmenge $R \subsetneq \mathcal{O}$ nachzuweisen, müssen also nur die Junktoren $\neg, \vee$ und $\wedge$ mithilfe der Konstanten und Junktoren in R ausgedrückt werden.

Zeigen Sie:

(a) $T = \{\mathbf{0}, \to\}$ ist vollständig. *Hinweis:* Wie können Sie die Negation $\neg\varphi$ ausdrücken?

[5] Den Beweis dafür liefert bereits Satz 3.33 aus dem Skript. In Teil (h) wird gezeigt, wie die beiden Konstanten **1** und **0** durch $\neg, \vee, \wedge$ ausgedrückt werden können und in Teil (k) bzw. (l) dasselbe für die Implikation, Biimplikation und XOR.

(b) $U = \{\mathbf{1}, \oplus\}$ ist nicht vollständig. *Hinweis:* Wie sehen Formeln aus AL_U aus? Benutzen Sie Assoziativität und Kommutativität für $\oplus$.

(c) Finden Sie eine möglichst kleine Obermenge $\widetilde{U} \supseteq U$, sodass $\widetilde{U}$ vollständig ist, und weisen Sie die Vollständigkeit nach.

Aufgabe 3.19. Betrachten Sie die Formeln

$$\varphi_1 := \neg V_3 \vee V_2 \vee \neg V_1, \qquad \varphi_2 := \neg V_3 \vee V_2 \vee (\neg V_1 \wedge V_3), \qquad \varphi_3 := V_2 \vee (\neg V_1 \wedge V_3)$$

und geben Sie an, welche dieser Formeln logisch äquivalent zueinander sind, beziehungsweise welche aus welcher semantisch folgt.

Aufgabe 3.20.

(a) Für jedes $n \in \mathbb{N}$ sei die aussagenlogische Formel φ_n definiert durch

$$\varphi_n \;:=\; \begin{cases} (V_n \vee V_{n+1}), & \text{falls } n \text{ gerade} \\ (V_n \to \neg V_{n+1}), & \text{falls } n \text{ ungerade.} \end{cases}$$

Es gilt also

$$\varphi_0 = (V_0 \vee V_1), \quad \varphi_1 = (V_1 \to \neg V_2), \quad \varphi_2 = (V_2 \vee V_3), \quad \varphi_3 = (V_3 \to \neg V_4), \quad \cdots$$

Geben Sie eine Belegung $\mathcal{B}$ an, so dass für alle $n \in \mathbb{N}$ gilt: $\mathcal{B}$ erfüllt φ_n.

(b) Für jedes $n \in \mathbb{N}$ sei die aussagenlogische Formel ψ_n definiert durch

$$\psi_n \;:=\; \begin{cases} (V_n \leftrightarrow V_{n+2}), & \text{falls } n \text{ gerade} \\ (V_n \leftrightarrow \neg V_{n-1}), & \text{falls } n \text{ ungerade.} \end{cases}$$

Es gilt also

$$\psi_0 = (V_0 \leftrightarrow V_2), \quad \psi_1 = (V_1 \leftrightarrow \neg V_0), \quad \psi_2 = (V_2 \leftrightarrow V_4), \quad \psi_3 = (V_3 \leftrightarrow \neg V_2), \quad \cdots$$

Geben Sie eine Belegung $\mathcal{B}\colon \textsc{Avar} \to \{0,1\}$ an, so dass für alle $n \in \mathbb{N}$ gilt: $\mathcal{B}$ erfüllt ψ_n.

Aufgabe 3.21. Beweisen oder widerlegen Sie die folgenden Behauptungen:

(a) $\big((\neg V_0 \vee V_2) \wedge (V_1 \to \neg V_2)\big) \;\models\; \neg\big((V_0 \wedge \neg V_1) \to \neg(V_0 \to V_2)\big)$

(b) $\big(\neg(V_0 \leftrightarrow V_1) \wedge (\neg V_2 \vee V_0)\big) \;\models\; \big(V_0 \vee (V_1 \wedge \neg V_2)\big)$

(c) $\big(\neg(V_0 \leftrightarrow V_1) \wedge (\neg V_2 \vee V_0)\big) \;\equiv\; \big(V_0 \vee (V_1 \wedge \neg V_2)\big)$

(d) Wenn $\varphi \models \psi$, dann $(\psi \to \chi) \models (\varphi \to \chi)$ f.a.$\chi \in \mathrm{AL}$.

(e) $\varphi \equiv \psi \equiv \chi$ gilt genau dann, wenn $(\varphi \leftrightarrow \psi \leftrightarrow \chi)$ allgemeingültig ist.

Aufgabe 3.22. Beweisen Sie Beobachtung 3.30 (b) und (d), d.h. beweisen Sie, dass für alle aussagenlogischen Formeln φ und ψ gilt:

(a) $\varphi \models \mathbf{0} \iff \varphi$ ist unerfüllbar.

(b) $\varphi \models \psi \iff (\varphi \wedge \neg\psi)$ ist unerfüllbar.

Aufgabe 3.23. Betrachten Sie die folgenden beiden Aussagen:

(1) Wenn sich der Rechner ein Virus eingefangen hat oder nicht mehr funktioniert, und wenn der Administrator erreichbar ist, dann rufen wir den Administrator.

(2) Wenn sich der Rechner ein Virus eingefangen hat, so rufen wir den Administrator, falls wir ihn erreichen; und wenn der Administrator erreichbar ist und der Rechner nicht funktioniert, so rufen wir den Administrator.

(a) Formalisieren Sie jede der beiden Aussagen (1), (2) durch eine aussagenlogische Formel.

(b) Zeigen Sie, dass die beiden Aussagen (1) und (2) äquivalent sind.

Aufgabe 3.24. Einer Ihrer Bekannten berichtet von seiner Zimmersuche in Frankfurt und äußert Ihnen gegenüber folgende Aussagen, die auf alle der von ihm besichtigten Wohnungen zutreffen:

- Wenn es sich um eine 1-Zimmer-Wohnung handelt, dann stehen höchstens 26 m^2 Wohnraum zur Verfügung oder der Mietpreis ist höher als 400 €.
- Wenn sich das Zimmer nicht in einer 1-Zimmer-Wohnung befindet, dann ist das Zimmer in einer WG.
- Wenn mehr als 26 m^2 Wohnraum zur Verfügung stehen, dann liegt das Zimmer nicht in einer WG.
- Wenn mehr als 26 m^2 Wohnraum zur Verfügung stehen und der Mietpreis höher als 400 € ist, dann handelt es sich nicht um eine 1-Zimmer-Wohnung.

(a) Zerlegen Sie den obigen Text in atomare Aussagen und geben Sie eine aussagenlogische Formel φ an, die das im Text zusammengefasste Wissen repräsentiert.

Betrachten Sie nun die nachfolgenden Aussagen:

- In jeder besichtigten Wohnung stehen Ihrem Bekannten maximal 26 m^2 zur Verfügung.
- Für jede besichtigte Wohnung gilt: Wenn die Wohnung in einer WG liegt, dann beträgt der Mietpreis höchstens 400 €.
- Für jede besichtigte Wohnung gilt: Wenn der verlangte Mietpreis höchstens 400 € beträgt, dann handelt es sich um eine WG oder um eine 1-Zimmer-Wohnung.

(b) Geben Sie für jede der drei Aussagen eine Formel an, die die Aussage repräsentiert.

(c) Entscheiden Sie für jede der Formeln aus (b), ob sie aus der Formel φ in (a) folgt.

Aufgabe 3.25.

(a) Es sei $\varphi := ((V_1 \wedge V_2) \vee \neg(V_1 \leftrightarrow \neg V_3))$. Konstruieren Sie die Wahrheitstafel zu φ. Erzeugen Sie aus dieser Wahrheitstafel eine zu φ logisch äquivalente aussagenlogische Formel φ' in DNF.

(b) Geben Sie für jede der folgenden Formeln an, ob sie in DNF und/oder KNF ist.

(i) $((V_0 \wedge \neg V_1) \vee (\neg V_1 \wedge \neg V_0))$

(ii) $((\mathbf{0} \vee \neg V_2) \rightarrow V_2)$

(iii) $\neg V_2 \wedge (V_5 \vee \neg V_1)$

(iv) $(V_3 \wedge V_5)$

(v) $\neg V_2 \wedge ((V_1 \wedge \neg V_0) \vee V_3)$

(vi) $(\neg(\neg V_3 \vee \neg V_3) \wedge \neg V_3)$

(vii) $\big((V_4 \wedge V_3) \wedge \neg V_2\big)$

(viii) $(\neg\neg V_0 \wedge V_0)$

Aufgabe 3.26.

(a) Sei ψ eine Formel mit $\mathrm{Var}(\psi) = \{X_1, X_2, X_3\}$, für die gelte:

$$[\![\psi]\!]^{\mathcal{B}} = 1 \iff \mathcal{B}(X_1) + \mathcal{B}(X_2) + \mathcal{B}(X_3) \geqslant 2$$

i) Bestimmen Sie die kanonische DNF für ψ.

ii) Weisen Sie nach, dass

- $(X_1 \wedge X_2 \wedge X_3)$ ein Implikant von ψ ist,
- X_2 *kein* Implikant von ψ ist,
- $(X_1 \wedge X_2)$ ein Primimplikant von ψ ist.

iii) Bestimmen Sie alle Primimplikanten von ψ und finden Sie eine möglichst kurze DNF für ψ, d.h. eine DNF mit möglichst wenigen Konjunktionstermen.

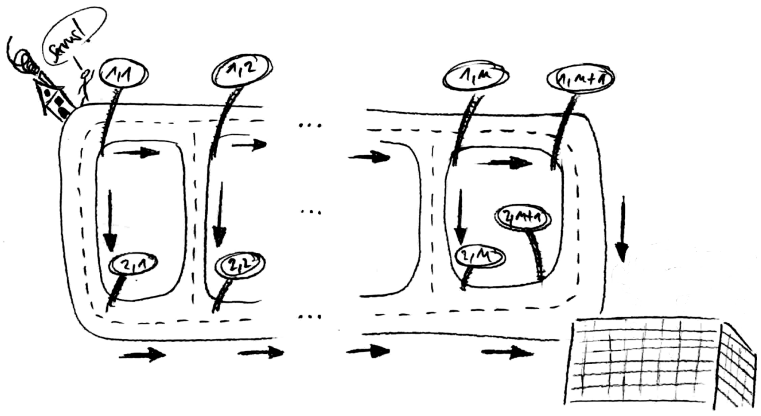
(b) Betrachten Sie die Formel

$$\eta = (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C)$$

i) Bestimmen Sie alle Primimplikanten von η.

ii) Bestimmen Sie eine **minimale** DNF für η, d.h. eine DNF mit möglichst wenigen Konjunktionstermen.

Aufgabe 3.27. Ivan hat große Probleme, sich in der Stadt zu orientieren. Das Straßennetz besteht aus einer Menge von Kreuzungen $K := \{1,2\}\times\{1,2,\ldots,n{+}1\}$. Zwischen je zwei Kreuzungen $(1,b),(2,b) \in K$ verläuft eine (vertikale) Straße, zwischen je zwei Kreuzungen $(a,b),(a,b{+}1) \in K$ verläuft eine (horizontale) Straße. Alle Straße sind Einbahnstraßen: Sie können nur von oben nach unten bzw. von links nach rechts befahren werden. Ivan wohnt an Kreuzung $(1,1)$, der Eingang zum Einkaufszentrum befindet sich an Kreuzung $(2,n{+}1)$. Dort hat er sich ein neues Navi gekauft. Um weiteres Zubehör zu erwerben, möchte er mit dem Auto zum Einkaufszentrum fahren.



Das Verkehrsamt muss allerdings einige der horizontal verlaufenden Straßenabschnitte erneuern lassen und richtet dazu Baustellen ein. Ein Straßenabschnitt mit Baustelle wird voll gesperrt, die Straße ist also nicht mehr befahrbar. Glücklicherweise kann das Navi über das Internet auf den neuesten Aussagenlogik-Cloud-Service (ALCS) zugreifen und so herausfinden, ob ein Ziel erreichbar ist.

ALCS verwendet die aussagenlogischen Variablen $X_1, \ldots, X_n, Y_1, \ldots, Y_n$, wobei Variable X_i ausdrückt, dass der Straßenabschnitt zwischen den Kreuzungen $(1, i)$ und $(1, i{+}1)$ gesperrt ist, und Y_i ausdrückt, dass der Straßenabschnitt zwischen $(2, i)$ und $(2, i{+}1)$ gesperrt ist.

(a) Bestimmen Sie für allgemeines $n \geqslant 3$ eine Formel φ_n in DNF, die ausdrückt, dass Ivan das Einkaufszentrum trotz der Baustellen erreichen kann, dass es also einen baustellenfreien Weg von Ivans Wohnung zum Einkaufszentrum gibt.
Beschreiben Sie zuerst Ihre **Idee** bzw. skizzieren Sie Ihren Lösungsweg.

Hinweis: Wie sehen die Primimplikanten von φ_n aus?

(b) Bestimmen Sie für allgemeines $n \geqslant 3$ eine Formel ψ_n in KNF, die dasselbe wie φ_n ausdrückt.
Beschreiben Sie zuerst Ihre **Idee** bzw. skizzieren Sie Ihren Lösungsweg.

Aufgabe 3.28. Das n-Damen-Problem ist wie folgt definiert: Für ein gegebenes $n \in \mathbb{N}_{>0}$ sollen n Damen auf einem $n \times n$-Schachbrett so platziert werden, dass sie sich nicht gegenseitig *bedrohen*. Eine solche Platzierung bezeichnen wir als *Lösung* des n-Damen-Problems. (Eine Dame darf beliebig viele Felder in horizontaler, vertikaler und diagonaler Richtung ziehen.)

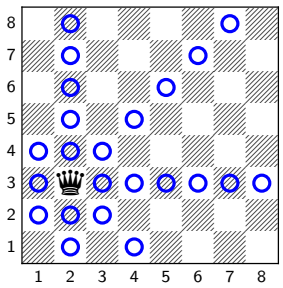


Abbildung 3.2.: Ein 8×8-Schachbrett. Die Dame auf $(3, 2)$ bedroht alle blau markierten Felder. Dort darf keine andere Dame stehen.

(a) Besitzt das n-Damen-Problem für $n = 3$ bzw. $n = 4$ eine Lösung?

(b) Modellieren Sie das n-Damen-Problem durch aussagenlogische Formeln. Verwenden Sie im Folgenden für alle $i, j \in \{1, \ldots, n\}$ die aussagenlogische Variable $D_{i,j}$ mit der Bedeutung „auf dem Feld (i, j) in Zeile i und Spalte j steht eine Dame“. Die Teilaufgaben (i) bis (iv) bauen nicht aufeinander auf. Geben Sie jeweils auch eine kurze ErläuterungIhrer Formeln an.

(i) Konstruieren Sie eine Formel $\mathit{zeile}_{i,j}$, die aussagt: „Wenn eine Dame auf dem Feld (i, j) steht, darf in der i-ten Zeile keine weitere Dame stehen“.

(ii) Stellen Sie eine entsprechende Formel $\mathit{spalte}_{i,j}$ für das Feld (i, j) und die j-te Spalte auf.

(iii) Konstruieren Sie eine Formel $\mathit{diagonalen}_{i,j}$, die das gleiche für die beiden Diagonalen des Feldes (i,j) aussagt.

(iv) Konstruieren Sie eine Formel *mindestens*, die aussagt, dass in jeder Zeile mindestens eine Dame steht.
Hinweis: Zusammen mit den Formeln aus (i), (ii) und (iii) folgt, dass genau n Damen auf dem Schachbrett stehen.

(c) Geben Sie eine Formel φ_n an, sodass die erfüllenden Belegungen von φ_n genau den Lösungen des n-Damen-Problems entsprechen.

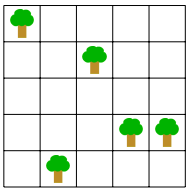
Aufgabe 3.29. Sei $k \in \mathbb{N}_{>0}$. Sie machen mit Ihren Kumpels und Kumpelinen einen Camping-Ausflug auf einem $k \times k$-Gitter mit einer Wald- und Wiesenlandschaft wie aus einem Bilderbuch. Sei $G_k := \{1, 2, \ldots, k\}^2$ die Menge aller Zellen des Gitters und sei **Bäume** $\subseteq G_k$ die Menge der Zellen, in denen ein Baum steht.

Wo sollten Sie am besten die Zelte aufbauen? Da Sie optimal vorbereitet sind, haben Sie die Ratgeber-Broschüre „Survival-Guide: Informatiknahes Camping für Dummies“ gelesen. Dort werden wichtige Regeln gegeben, die Sie beim Camping einhalten müssen:

1) Ein Zelt kann nur dort platziert werden, wo kein Baum steht.
2) Ein Zelt muss horizontal oder vertikal mit einem Baum benachbart sein.
3) Zelte dürfen nicht horizontal, nicht vertikal und auch nicht diagonal benachbart sein.
4) In jeder Spalte muss *mindestens* ein Zelt stehen.
5) In jeder Zeile darf *höchstens* ein Zelt stehen.

Verwenden Sie im Folgenden für alle $(i,j) \in G_k$ die Variablen $Z_{i,j}$ mit der Bedeutung „in Zelle (i,j) steht ein Zelt“ und die Variablen $B_{i,j}$ mit der Bedeutung „in Zelle (i,j) steht ein Baum“.

(a) Platzieren Sie auf dem folgenden 5×5-Gitter fünf Zelte und halten Sie dabei die Regeln aus dem Survival-Guide ein.



(b) Für das allgemeine Camping-Puzzle, also für beliebiges $k \in \mathbb{N}_{>0}$ und eine beliebige Teilmenge **Bäume** $\subseteq G_k$ wollen wir die Positionen der Bäume auf dem Gitter und die fünf Regeln durch aussagenlogische Formeln ausdrücken. Die korrekte Platzierung der Bäume erzwingen wir mit folgender Formel

$$\varphi_{\text{Baum}} := \bigwedge_{(i,j) \in \textbf{Bäume}} B_{i,j} \wedge \bigwedge_{(i,j) \in G_k \setminus \textbf{Bäume}} \neg B_{i,j} \ .$$

Alle Formeln sind in **konjunktiver Normalform** (KNF) anzugeben. Erläutern Sie außerdem jeweils kurz die Idee, die Ihrer Formel zugrunde liegt. Gehen Sie analog zur KNF-Modellierung des Sudoku-Spiels (Beispiel 3.48) vor.

(i) Geben Sie eine Formel φ_1 an, die Regel 1 formalisiert.

(ii) Geben Sie eine Formel φ_2 an, die Regel 2 formalisiert. *Hinweis:* Für jede Zelle (i,j) ist $HV(i,j) := \big\{(i{-}1,j),(i{+}1,j),(i,j{-}1),(i,j{+}1)\big\} \cap G_k$ die Menge aller mit (i,j) horizontal oder vertikal benachbarten Zellen auf dem Gitter G_k. Nutzen Sie die Mengen $HV(1,1), HV(1,2), \ldots, HV(k,k)$ für Ihre Formel φ_2.

(iii) Geben Sie eine Formel φ_3 an, die Regel 3 formalisiert.
Hinweis: Definieren Sie analog zu ii) geeignete Indexmengen.

(iv) Geben Sie eine Formel φ_4 an, die Regel 4 formalisiert.

(v) Geben Sie eine Formel φ_5 an, die Regel 5 formalisiert.

Hinweis: Notationen wie

$\bigwedge_{i=1}^{k} \ldots$ oder $\bigwedge_{(i,j)\in M} \ldots$ (für eine Menge M) oder $\bigwedge_{(i,j)\in G_k} \bigvee_{(i',j')\in HV(i,j)} \cdots$

sind hier hilfreich.

Aufgabe 3.30. Gegeben sei ein $2n \times 2n$-Schachbrett $\{1,\ldots,2n\}^2$ identifiziert. Wir nehmen an, dass eine Teilmenge $G \subseteq \{1,\ldots,2n\}^2$ von Zellen gesperrt ist. Im Domino-Problem ist zu entscheiden, ob alle nicht-gesperrten Zellen mit Dominosteinen überdeckt werden können, wobei ein Dominostein zwei aufeinanderfolgende Zellen einer Zeile oder Spalte überdecken darf. Jede nicht-gesperrte Zelle muss von genau einem Dominostein überdeckt werden und keine gesperrte Zelle darf überdeckt werden.

(a) Die aussagenlogische Variable $G_{i,j}$ soll aussagen, dass Zelle $G_{i,j}$ gesperrt ist, die aussagenlogische Variable $X_{(i,j),(i',j')}$ soll ausdrücken, dass die Zellen (i,j) und (i',j) von genau einem Dominostein überdeckt werden.

Welche X-Variablen sollte man sinnvollerweise auswählen?

(b) Beschreiben Sie eine KNF-Formel φ_G, die genau dann erfüllbar ist, wenn das Domino-Problem lösbar ist.

Aufgabe 3.31. Seien $m,n \in \mathbb{N}_{>0}$. Das Spiel *Minesweeper*[6] wird auf einem Spielfeld $F_{m,n} := \{1,\ldots,m\} \times \{1,\ldots n\}$ mit m Zeilen und n Spalten gespielt. Jede Zelle $(i,j) \in F_{m,n}$ enthält entweder genau eine *Mine* oder ist *frei*. Für eine Zelle $(i,j) \in F_{m,n}$ bezeichnen wir alle horizontal, vertikal oder diagonal angrenzenden Zellen als *Nachbarn* von (i,j). Sei $N(i,j)$ die Menge aller Nachbarn von (i,j).
Zu Beginn des Spiels ist nicht bekannt, welche der Zellen Minen enthalten. Um dies herausfinden, erhält der Spieler Hinweise: Eine Teilmenge HINTS $\subsetneq F_{m,n}$ der Zellen wird bekanntgegeben („aufgedeckt"), in denen jeweils eine Zahl aus der Menge $\{0,1\ldots,8\}$ steht: Wenn in einer Zelle (i,j) die Zahl z steht, dann

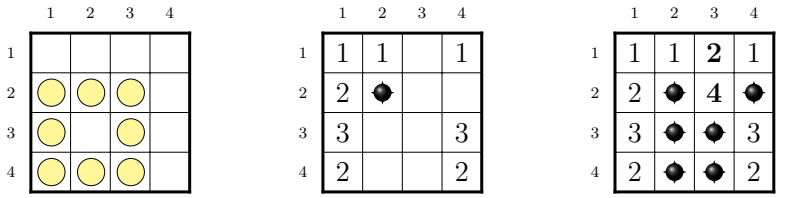
- ist die Zelle (i,j) frei und
- es befinden sich auf den Nachbarzellen von (i,j) **genau** z Minen.

Mithilfe dieser Hinweise muss der Spieler herausfinden, welche der Zellen Minen enthalten, und welche nicht. Ziel des Spiels ist es, alle freien Zellen „aufzudecken", ohne dabei eine Mine zu erwischen.

Wir wollen das Spiel mithilfe von Aussagenlogik modellieren. Verwenden Sie im Folgenden die aussagenlogischen Variablen $X_{i,j}$ mit der Bedeutung „die Zelle $(i,j) \in F_{m,n}$ enthält eine Mine".

(a) Sei $(i,j) \in$ HINTS.

[6] Hier können Sie Minesweeper spielen: http://minesweeperonline.com/.



(a) $N(3,2)$ (b) teilweise gelöst (c) vollständig gelöst

Abbildung 3.3.: Ein 4×4-Minesweeper-Spielfeld, links die Menge $N(3,2)$ aller Nachbarzellen von $(3,2)$, Mitte teilweise gelöst, rechts vollständig gelöst: Die Beschriftung der Zelle $(2,1)$ in der zweiten Zeile und ersten Spalte sagt aus, dass genau zwei seiner Nachbarzellen Minen enthalten.

(i) Entwerfen Sie eine Formel $\varphi_{i,j}^{(0)}$, die aussagt, dass (i,j) und alle Nachbarn von (i,j) frei sind.

(ii) Entwerfen Sie eine Formel $\varphi_{i,j}^{(1)}$, die aussagt, dass (i,j) frei ist und genau eine Nachbarzelle eine Mine enthält.

(b) Angenommen, für alle Zellen $(i,j) \in \textsc{Hints}$ ist (i,j) frei, genau $z_{i,j}$ viele der Nachbarn enthalten eine Mine und die Formel $\varphi_{i,j}^{(z_{i,j})}$ drückt dies jeweils aus.

Geben Sie eine Formel ψ an, die all diese Bedingungen wiedergibt.

(c) Sei $(i,j) \in F_{m,n}$ beliebig. Wann kann (i,j) gefahrlos aufgedeckt werden? Geben Sie eine Formel ξ an, die genau dann **unerfüllbar** ist, wenn (i,j) gefahrlos aufgedeckt werden kann.
Kommentar: Wozu ist das gut? Wir können ξ in eine Menge von Disjunktionstermen umwandeln und dann mit Resolution auf Unerfüllbarkeit überprüfen.

Aufgabe 3.32.

(a) Zeigen Sie mit Resolution, dass die KNF-Formel

$$\omega := (A \vee B \vee \neg C) \wedge (\neg A) \wedge (A \vee \neg B) \wedge (A \vee B \vee C)$$

unerfüllbar ist, indem Sie den leeren Disjunktionsterm ϵ herleiten. Zu zeigen ist also:

$$\Big\{\{A,B,\neg C\},\ \{\neg A\},\ \{A,\neg B\},\ \{A,B,C\}\Big\} \vdash_{\mathfrak{R}} \epsilon$$

(b) Gegeben sei die Menge

$$\Phi := \Big\{\{A,B,C\},\{A,B,\neg C\},\{A,\neg B,C\},\{A,\neg B,\neg C\},$$
$$\{\neg A,B,C\},\{\neg A,B,\neg C\},\{\neg A,\neg B,C\},\{\neg A,\neg B,\neg C\}\Big\}$$

von Disjunktionstermen. Leiten Sie mit Resolution den leeren Disjuktionsterm ϵ aus Φ her.

Aufgabe 3.33.

(a) Leiten Sie den leeren Disjunktionsterm ϵ mittels Resolution aus den Mengen K_1, K_2 bzw. K_3 her.

(i) $K_1 := \big\{\{Q, R\}, \{\neg Q, R, \neg T\}, \{\neg Q, \neg R, \neg T\}, \{T\}, \{Q, \neg R\}\big\}$

(ii) $K_2 := \big\{\{A, B\}, \{B, C\}, \{\neg B, C\}, \{\neg A, \neg C\}, \{\neg B, \neg C\}\big\}$

(iii) $K_3 := \big\{\{X, Y\}, \{X, Z\}, \{Y, Z\}, \{\neg X, \neg Y\}, \{\neg X, \neg Z\}, \{\neg Y, \neg Z\}\big\}$

(b) Zeigen Sie mit Resolution, dass die KNF-Formel

$$\psi := \big(A \vee B \vee D\big) \wedge \big(\neg A\big) \wedge \big(A \vee B \vee C \vee \neg D\big) \wedge \big(A \vee \neg B\big) \wedge \big(A \vee B \vee \neg C \vee \neg D\big)$$

unerfüllbar ist. Wandeln Sie die Formel zunächst in eine Klauselmenge um.

(c) Erläutern Sie, warum der folgende „Resolutionsschritt“ **falsch** ist:

$$\frac{\{A, B\}, \{\neg A, \neg B\}}{\epsilon}$$

Aufgabe 3.34. Das Beweissystem ABS* entsteht aus dem Beweissystem ABS, wenn der Modus Ponens durch den Modus Tollens ersetzt wird. Zeige:

$$\{\, \alpha_1, \alpha_1 \to \alpha_2 \,\} \vdash_{\mathrm{ABS}^*} \alpha_2$$

Also „gilt“ der Modus Ponens auch im neuen Beweissystem ABS*.

Aufgabe 3.35. Sei $n \in \mathbb{N}_{>0}$ und seien $\varphi, \psi \in \mathrm{AL}$ mit $\mathrm{Var}(\varphi) = \mathrm{Var}(\psi) = \{V_1, \ldots, V_n\}$ und $f_\varphi, f_\psi : \{0,1\}^n \to \{0,1\}$ dazu passende boolesche Funktionen. Zeigen Sie:

(a) $\varphi \models \psi \iff f_\varphi(x) \leqslant f_\psi(x)$ für alle $x \in \{0,1\}^n$

(b) $\varphi \equiv \psi \iff f_\varphi(x) = f_\psi(x)$ für alle $x \in \{0,1\}^n$

4. Beweise verstehen, Beweise führen

Angenommen, wir möchten Aussagen über ein Phänomen machen. In vielen Fällen ist das Phänomen so komplex, dass eine vollständige Erklärung nicht zu erwarten ist, man muss und sollte sich mit partiellen Erklärungen, wie etwa mit Erfahrungswerten und Ergebnissen experimenteller Arbeit bescheiden. In einigen Fällen aber sind unumstößlich wahre Aussagen nicht nur möglich, sondern werden sogar verlangt: Es genügt nicht, dass eine sicherheitssensitive Software funktionieren sollte, sie muss funktionieren!

Aber wie überzeugt man sich davon, dass eine Aussage tatsächlich immer, also in jeder denkbaren Situation wahr ist? Wir benutzen die Sprache der Mathematik. Und wie begründet man in der Sprache der Mathematik, dass es sich zweifelsfrei um eine wahre Aussage handelt? Man gibt einen Beweis an.

Ziel dieses Abschnitts ist deshalb ein kurzer Überblick über grundlegende Beweistechniken, und zwar insbesondere über die folgenden Beweismethoden:

(1) **direkter Beweis** — direkter Beweis

(2) Beweis durch **Kontraposition** — Kontraposition

(3) **Beweis durch Widerspruch** (indirekter Beweis) — Beweis durch Widerspruch

(4) **vollständige Induktion**. — vollständige Induktion

Einige einfach zu beschreibende Beweismethoden können wir sofort ansprechen.

- Eine existentielle Aussage, also eine Aussage der Form

  $$\textit{Es gibt ein Objekt mit der Eigenschaft } E \tag{4.1}$$

  lässt sich manchmal mit der Methode der **Konstruktion** zeigen: Man beschreibt ein spezifisches Objekt und weist nach, dass dieses Objekt die Eigenschaft E besitzt. — Konstruktion

  Beispielsweise können wir die existentielle Aussage „*das quadratische Polynom* $x^2 - 1$ *besitzt eine reellwertige Nullstelle*“ beweisen, indem wir feststellen, dass die reelle Zahl 1 eine Nullstelle ist.

- Eine All-Aussage der Form

  $$\textit{Alle fraglichen Objekte besitzen die Eigenschaft } E \tag{4.2}$$

  lässt sich mit einem **Gegenbeispiel** widerlegen. Es genügt in diesem Fall ein Objekt zu konstruieren, dass die Eigenschaft E nicht besitzt. Warum? Die existentielle Aussage „*Es gibt ein Objekt mit der Eigenschaft* $\neg E$“ ist die Negation der All-Aussage (4.2). Beispielsweise kann die All-Aussage „*jedes quadratische Polynom besitzt eine reellwertige Nullstelle*“ widerlegt werden, indem gezeigt wird, dass das quadratische Polynom x^2+1 keine reellwertige Nullstelle besitzt. — Gegenbeispiel

  Möchten wir hingegen die All-Aussage (4.2) beweisen, ist zu zeigen, dass ein *beliebiges* Objekt die Eigenschaft E besitzt.

Ringbeweis

- Wie zeigt man möglichst kräfteschonend, dass drei Aussagen A_1, A_2, A_3 äquivalent sind? Ein **Ringbeweis** wird zuerst die Implikation $A_1 \to A_2$ und dann die Implikationen $A_2 \to A_3$ sowie $A_3 \to A_1$ nachweisen. Und warum funktioniert das? Weil die semantische Folgerung

$$\{ A_1 \to A_2,\ A_2 \to A_3,\ A_3 \to A_1 \} \models (A_i \leftrightarrow A_j)$$

für alle i, j mit $1 \leqslant i, j \leqslant 3$ gilt.

4.1. Was sind „Sätze“ und „Beweise“?

Satz
Theorem
Beweis

Wir haben Sätze bereits ausgiebig benutzt. Ein **Satz** (bzw. **Theorem**) besteht aus Voraussetzungen und einer Behauptung. Wenn alle Voraussetzungen erfüllt sind, dann muss die Behauptung wahr sein. Der **Beweis** eines Satzes muss nachweisen, dass die Behauptung des Satzes wahr ist und kann dabei verwenden:

- die Voraussetzungen des Satzes,
- Definitionen und bereits bekannte Tatsachen und Sätze,
- im Beweis selbst oder anderswo bereits als wahr bewiesene Aussagen,
- logische Schlussregeln.

In der Aussagenlogik haben wir eine ähnliche, aber häufig einfachere Situation angetroffen, wenn – mit der Menge Φ als Menge der Voraussetzungen und der aussagenlogischen Formel φ als Behauptung – eine semantische Folgerung $\Phi \models \varphi$ zu beweisen ist.

Im Allgemeineren sind aber kompliziertere Aussagen zu beweisen, denn es werden vielfach nicht Aussagen über Wahrheitswerte, sondern Aussagen über gänzlich andere Objekte gemacht. Trotzdem helfen uns die mit der Aussagenlogik gemachten Erfahrungen weiter, denn viele logische Schlussregeln sind semantische Folgerungen bzw. logische Äquivalenzen der Aussagenlogik. Ein einfaches Beispiel ist der Modus Ponens:

> Folgere ψ, wenn φ und $\varphi \to \psi$ bereits abgeleitet wurden.

Hier ist es nicht notwendig, dass φ und ψ aussagenlogische Formeln sind, sondern es reicht völlig aus, dass φ und ψ Eigenschaften sind, die entweder wahr oder falsch sind. Die Beweismethode der **Kontraposition** ist ein weiteres Beispiel. Die Kontraposition beruht auf der Äquivalenz

$$(\varphi \to \psi) \ \equiv (\neg\psi \to \neg\varphi).$$

Um die Implikation $\varphi \to \psi$ zu beweisen, genügt es also anzunehmen, dass $\neg\psi$ gilt und dann $\neg\psi$ zu zeigen. Im **Beweis durch Widerspruch** wird die Äquivalenz

$$(\varphi \to \psi) \equiv ((\varphi \wedge \neg\psi) \to \mathbf{0})$$

ausgenutzt: Um die Implikation $\varphi \to \psi$ zu beweisen, genügt es also anzunehmen, dass die Implikation falsch ist und eine falsche Aussage, einen Widerspruch herzuleiten.

Wir sind also gut gerüstet, sollten aber **typische Fehler** bei der Formulierung von Beweisen vermeiden:

- Wenn eine All-Aussage zu zeigen ist, dann genügt ein „Beweis durch Beispiel“ natürlich nicht: Die Aussage ist nicht nur für einige Beispiele zu verifizieren, sondern ist für alle Instanzen zu zeigen.

- Die Notation kann uns einen Streich spielen, wenn gleiche Symbole zur Bezeichnung verschiedener Dinge verwendet werden.
- Die Bedeutung eingeführter Begriffe muss klar sein und darf nicht vom Kontext abhängen.
- Unzulässige Gedankensprünge beim Schlussfolgern bedeuten, dass das Argument unvollständig ist und
- das Ausnutzen von bis dahin noch unbewiesenen Behauptungen ist ein Fehler im Argument.

4.2. Beweistechniken

Im Folgenden besprechen wir „direkte Beweise“, „Beweise durch Kontraposition“ und „Beweise durch Widerspruch“. Die vollständige Induktion ist aufgrund ihrer Bedeutung für die Informatik alleiniger Inhalt des nächsten Abschnitts.

4.2.1. Beweistechnik „direkter Beweis“

Bei einem *direkten Beweis* wird die Behauptung eines Satzes „direkt“, d.h. ohne „Umwege“, bewiesen.

Wir haben bereits einige direkte Beweise geführt. Zum Beispiel haben wir im Beweis von Satz 2.57 und Folgerung 2.58 gezeigt, dass eine endliche Menge M genau $2^{|M|}$ Teilmengen besitzt. Wie sahen die Beweisschritte aus?

1. Zuerst haben wir eine bijektive Funktion

   $$f : \mathcal{P}(M) \to \mathrm{Abb}(M, \{0,1\})$$

   von der Potenzmenge $\mathcal{P}(M)$ nach $\mathrm{Abb}(M, \{0,1\})$ konstruiert: $\mathcal{P}(M)$ und $\mathrm{Abb}(M, \{0,1\})$ sind also gleichgroß. (Wir haben die Beweismethode der Konstruktion angewendet.)

2. Dann haben wir eine bijektive Funktion

   $$g : \mathrm{Abb}(A, B) \to B^A$$

   sogar für beliebige endliche Mengen A, B „gebaut“ und damit

   $$|\mathrm{Abb}(A, B)| = |B|^{|A|}$$

   gezeigt.

3. Jetzt müssen wir nur noch $A = M$ und $B = \{0,1\}$ setzen. Es folgt

   $$|\mathcal{P}(M)| \overset{1.}{=} |\mathrm{Abb}(M, \{0,1\})| \overset{2.}{=} 2^{|M|}.$$

In unserem zweiten Beispiel zeigen wir, dass das arithmetische Mittel stets mindestens so groß wie das geometrische Mittel ist.

Satz 4.1. Für alle reellen Zahlen $a, b \geqslant 0$ gilt

$$\frac{a+b}{2} \geqslant \sqrt{a \cdot b}.$$

Zuerst versuchen wir eine Beweisidee zu erhalten.

1. Die Wurzel stört und wir quadrieren:
   - Statt $\frac{a+b}{2} \geqslant \sqrt{a \cdot b}$ zeige die Ungleichung $(\frac{a+b}{2})^2 \geqslant a \cdot b$.
2. Wir multiplizieren aus und erhalten die Ungleichung $\frac{a^2+2a\cdot b+b^2}{4} \geqslant a \cdot b$.
3. Wir multiplizieren mit 4 und erhalten $a^2 + 2a \cdot b + b^2 \geqslant 4a \cdot b$.
4. Wenn wir $4a \cdot b$ nach links „bringen“, ist $a^2 - 2a \cdot b + b^2 \geqslant 0$ zu zeigen.
   - $a^2 - 2a \cdot b + b^2 = (a - b)^2$ gilt.
   - Jedes Quadrat ist nicht-negativ und die Ungleichung stimmt!?!

Das ist leider **kein** Beweis, weil wir aus der Ungleichung $\frac{a+b}{2} \geqslant \sqrt{a \cdot b}$ eine wahre Aussage folgern. Hoffentlich haben wir nur mit äquivalenten Umformungen gearbeitet.

Beweis:

1. $a^2 - 2a \cdot b + b^2 = (a - b)^2$ gilt und $a^2 - 2a \cdot b + b^2 \geqslant 0$ folgt.
2. Wir addieren $4a \cdot b$ auf beide Seiten: $a^2 + 2a \cdot b + b^2 \geqslant 4a \cdot b$ gilt ebenfalls.
3. Die linke Seite der Ungleichung stimmt mit $(a + b)^2$ überein: Es gilt also
$$(a + b)^2 \geqslant 4a \cdot b.$$
4. Wir dividieren beide Seiten durch 4 und ziehen die Wurzel:
$$\frac{a + b}{2} \geqslant \sqrt{ab}$$
folgt und das war zu zeigen.

□

4.2.2. Beweistechnik „Beweis durch Kontraposition“

Der *Beweis durch Kontraposition* beruht auf der semantischen Äquivalenz

$$\big(\varphi \to \psi\big) \equiv \big(\neg\psi \to \neg\varphi\big).$$

Beachte, dass φ, ψ keine aussagenlogischen Formeln sein müssen, sondern beliebige Aussagen darstellen, die entweder wahr oder falsch sind.
Beim *Beweis durch Kontraposition* wird also ein Satz der Form

„Falls Aussage φ gilt, so gilt auch Aussage ψ“

dadurch bewiesen, dass man zeigt:

„Falls Aussage ψ **nicht** gilt, so kann auch Aussage φ **nicht** gelten.“

Als Beispiel für einen Beweis durch Kontraposition betrachten wir folgenden Satz.

Satz 4.2. Für jedes $n \in \mathbb{N}$ gilt: Falls n^2 eine gerade Zahl ist, so ist auch n eine gerade Zahl.

Beweis: Wir wenden die Beweismethode der Kontraposition an. Sei $n \in \mathbb{N}$ beliebig. Wir müssen zeigen: Falls n **keine** gerade Zahl ist, so ist auch n^2 **keine** gerade Zahl.

$n \in \mathbb{N}$ war beliebig gewählt. Falls n gerade ist, so ist nichts weiter zu beweisen. Wir betrachten daher nur den Fall, dass n **keine** gerade Zahl ist (d.h. n ist ungerade). Wir müssen zeigen, dass dann auch n^2 **keine** gerade Zahl ist (d.h. n^2 ist eine ungerade Zahl).

Beachte: Eine natürliche Zahl m ist genau dann **ungerade**, wenn es ein $k \in \mathbb{N}$ gibt, s.d. $m = 2k + 1$. Aber dann folgt $n^2 = (2k + 1)^2 = 4k^2 + 4k + 1 = 2 \cdot (2k^2 + 2k) + 1$. Also ist auch n^2 ungerade und n^2 ist keine gerade Zahl. □

4.2.3. Beweistechnik „Beweis durch Widerspruch“ (indirekter Beweis)

Der *Beweis durch Widerspruch* basiert auf der semantischen Äquivalenz

$$(\varphi \to \psi) \equiv \big((\varphi \wedge \neg\psi) \to \mathbf{0}\big).$$

Beachte, dass auch hier φ und ψ keine aussagenlogischen Formeln sein müssen, sondern beliebige Aussagen darstellen, die entweder wahr oder falsch sind.

Man beweist einen Satz der Form

> „Falls die Voraussetzungen φ erfüllt sind, so gilt Aussage ψ“

durch Widerspruch indem man

- annimmt, dass die Voraussetzungen φ erfüllt sind, aber die Aussage ψ **nicht** gilt und
- daraus einen Widerspruch herleitet.

Nicht nur Implikationen lassen sich durch Widerspruch beweisen. Möchten wir zum Beispiel die Aussage ψ durch Widerspruch beweisen, dann stellen wir uns vor, dass die Implikation $\mathbf{1} \to \psi$ zu zeigen ist: Nimm an, dass die Aussage ψ **nicht** gilt, und leite daraus einen Widerspruch her.

Als ein erstes Beispiel für einen Beweis durch Widerspruch betrachten wir folgenden Satz:

Satz 4.3. $\sqrt{2}$ ist irrational, also keine rationale Zahl.

Beweis: Im Beweis durch Widerspruch nehmen wir an, dass $\sqrt{2}$ eine rationale Zahl ist und müssen einen Widerspruch herleiten.

Wenn $\sqrt{2}$ eine rationale Zahl ist, dann gibt es natürliche Zahlen p', q' mit $\sqrt{2} = p'/q'$. Wir kürzen p', q' und erhalten teilerfremde Zahlen $p, q \in \mathbb{N}$ mit

$$\sqrt{2} = \frac{p}{q}.$$

Wir quadrieren und erhalten die Gleichung

$$p^2 = 2 \cdot q^2.$$

Also ist p^2 eine gerade Zahl. Wir haben aber in Satz 4.2 gezeigt, dass dann auch p gerade ist. Also gibt es eine Zahl $r \in \mathbb{N}$ mit $p = 2r$. Dann folgt $p^2 = 4r^2 = 2q^2$ und damit $2r^2 = q^2$.

Dann ist aber q^2 gerade. Wir wenden wieder Satz 4.2 an und erhalten, dass auch q gerade ist: Die Zahlen p und q haben den gemeinsamen Teiler 2 im Widerspruch zur Teilerfremdheit ↯ (Man verwendet das Zeichen ↯, um anzudeuten, dass man einen Widerspruch erhalten hat.) □

Der griechische Mathematiker Euklid (300 v. Chr.) hat gezeigt, dass es unendlich viele Primzahlen[1] gibt. Dieses Resultat ist Grundlage für viele Verfahren der Public-Key-Kryptographie. Mehr über die Public-Key-Kryptographie erfahren Sie in der Veranstaltung „Mathematik 2“.

Satz 4.4 (Satz von Euklid). Es gibt unendlich viele Primzahlen.

Beweis: Wir benutzen, dass sich jede natürliche Zahl als Produkt von Primzahlen schreiben lässt.

Wir nehmen an, dass es nur endlich viele Primzahlen gibt, und dass dies die Primzahlen $p_1, \ldots, p_n$ sind. Definiere die Zahl

$$N = p_1 \cdot p_2 \cdots p_n + 1.$$

Dann ist $N - 1$ durch alle Primzahlen teilbar und N kann durch keine Primzahl teilbar sein. Also ist N eine Primzahl und N ist eine neue Primzahl. $\lightning$ □

Wir zeigen jetzt ein fundamentales Ergebnis des Mathematikers Georg Cantor (1845-1918): Es gibt keine surjektive Funktion $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$. Insbesondere gibt es also keine bijektive Funktion $g : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ und die Potenzmenge der natürlichen Zahlen ist „größer“ als die Menge $\mathbb{N}$ der natürlichen Zahlen (vgl. Definition 2.54). Es kann gezeigt werden, dass es eine Bijektion $f : \mathcal{P}(\mathbb{N}) \to \mathbb{R}$ gibt, und wir erhalten als Konsequenz, dass die Menge der reellen Zahlen überabzählbar groß ist.

Satz 4.5 (Die Potenzmenge der natürlichen Zahlen ist nicht abzählbar). Es gibt keine surjektive Funktion von $\mathbb{N}$ nach $\mathcal{P}(\mathbb{N})$.

Beweis: Im Beweis durch Widerspruch nehmen wir an, dass die Funktion

$$f \colon \mathbb{N} \to \mathcal{P}(\mathbb{N})$$

surjektiv ist. Die zentrale Idee: Wir definieren die Menge

$$M \;:=\; \{\, n \in \mathbb{N} \;:\; n \notin f(n) \,\},$$

die offensichtlich eine Teilmenge von $\mathbb{N}$ ist: Also gilt $M \in \mathcal{P}(\mathbb{N})$. Da f surjektiv ist, muss es ein $m \in \mathbb{N}$ geben mit $f(m) = M$. Natürlich ist klar, dass entweder $m \in M$ oder $m \notin M$ gilt.

Fall 1: $m \notin M$:

- Nach Definition der Menge M folgt $m \in f(m)$.
- Es ist $f(m) = M$ und deshalb folgt $m \in f(m) = M$. $\lightning$

Fall 2: $m \in M$:

- Nach Definition der Menge M folgt $m \notin f(m)$.
- Es ist $f(m) = M$ und deshalb folgt $m \notin f(m) = M$. $\lightning$

In jedem Fall haben wir einen Widerspruch erhalten: Es kann keine Zahl m mit $f(m) = M$ geben und f ist im Gegensatz zur Annahme nicht surjektiv. □

Unser Beweis verifiziert die Aussage durch die Konstruktion einer Menge M, die nicht im Bild der Funktion f enthalten ist. Die Menge M scheint vom Himmel zu fallen, aber tatsächlich ist hier eine sehr wichtige Beweistechnik, die **Diagonalisierung**, „am Werke“ mit der man weitere fundamentale Einsichten erzielen kann: Diagonalisierung

Eine Funktion $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$ können wir durch folgende, unendlich große Tabelle repräsentieren

	0	**1**	**2**	**3**	**4**	**5**	$\cdots$
0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$	$a_{0,5}$	$\cdots$
1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$	$a_{1,5}$	$\cdots$
2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	$a_{2,5}$	$\cdots$
3	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$	$a_{3,5}$	$\cdots$
4	$a_{4,0}$	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$	$a_{4,5}$	$\cdots$
5	$a_{5,0}$	$a_{5,1}$	$a_{5,2}$	$a_{5,3}$	$a_{5,4}$	$a_{5,5}$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

wobei $a_{i,j} := \begin{cases} 1 & \text{falls } j \in f(i) \\ 0 & \text{falls } j \notin f(i). \end{cases}$ der Eintrag in Zeile i und Spalte j ist.

Wir erzwingen, dass sich M von $f(n)$ unterscheidet, da wir für jedes $n \in \mathbb{N}$ fordern:

$$n \in M \iff n \notin f(n). \tag{4.3}$$

Jetzt ist klar, dass M sich von allen Mengen $f(n)$ unterscheidet, d.h. dass $M \neq f(n)$ für alle natürlichen Zahlen $n \in \mathbb{N}$ gilt. Stellt man sich die unendlich lange „Zeile“ $(b_n : n \in \mathbb{N})$ vor, die M beschreibt, d.h. definiert man

$$b_n = 1 :\iff n \in M, \tag{4.4}$$

dann folgt

$$b_n = 1 \overset{(4.4)}{\iff} n \in M \overset{(4.3)}{\iff} n \notin f(n)$$

und b wird sich von der Zeile n in dem **Diagonalelement** $a_{n,n}$ unterscheiden: Kein Wunder, dass man von der Methode der Diagonalisierung spricht.

Bemerkung 4.6. Der Beweis von Satz 4.5 zeigt sogar die folgende, sehr viel stärkere Aussage:

> Es gibt keine surjektive Funktion einer Menge M nach $\mathcal{P}(M)$.

Die Potenzmenge ist also *immer* größer als die Ausgangsmenge.

Was kann man noch so alles mit der Methode der Diagonalisierung anstellen? In der Vorlesung „Theoretische Informatik 1“ wird gezeigt, dass es keine Super-Compiler Q gibt, die überprüfen, ob ein Anwenderprogramm P auf einer bestimmten Eingabe n_P korrekt rechnet. Wie macht man das?

- Ein C++-Programm P lässt sich als eine natürliche Zahl n_P auffassen. Dazu interpretiere die Binärdarstellung von P als die Binärdarstellung der Zahl n_P.
- Betrachte nur C++-Programme, die eine natürliche Zahl als Eingabe erwarten und dann *akzeptieren*, *verwerfen* oder *nicht halten*.

[1] Eine Primzahl ist eine natürliche Zahl größer als Eins, die nur durch sich selbst und durch die Eins teilbar ist.

Satz 4.7. Es gibt kein C++-Programm Q, so dass

$$Q \text{ akzeptiert } n_P \iff P \text{ akzeptiert } n_P \text{ nicht.} \tag{4.5}$$

Beweisskizze: Warum kann es ein solches C++-Programm Q nicht geben? Nimm an, dass Q doch existiert. Wir setzen $P := Q$ in (4.5) und erhalten

$$Q \text{ akzeptiert } n_Q \iff Q \text{ akzeptiert } n_Q \text{ nicht.}$$

Wir haben den gewünschten Widerspruch erhalten: Das Programm Q kann es nicht geben. □

Es gibt also keinen „Super-Compiler“ Q^*, der voraussagt, ob ein Programm P eine (beliebige) Eingabe $n \in \mathbb{N}$ akzeptiert!

Denn sonst könnte Q insbesondere voraussagen, dass P seine Kodierung n_P nicht akzeptiert.

In der „Theoretischen Informatik 1“ wird der vollständige Beweis von Satz 4.7 gezeigt. Eine Vielzahl weiterer Probleme wird behandelt, für die Rechner „chancenlos“ sind.

Bemerkung 4.8. Jede Aussage, die durch einen *Beweis durch Kontraposition* bewiesen werden kann, kann auch durch einen *Beweis durch Widerspruch* nachgewiesen werden. Um zu zeigen, dass die Aussage

„Falls Aussage φ gilt, so gilt auch Aussage ψ“

wahr ist, kann man in einem Beweis durch Widerspruch folgendermaßen vorgehen: Man nimmt an, dass Aussage φ gilt und Aussage ψ nicht gilt. Im Beweis durch Kontraposition kann man dann $\neg\varphi$ herleiten und erhält deshalb den Widerspruch $\varphi \wedge \neg\varphi$.

Übung: Beweisen Sie Satz 4.2 durch einen „Beweis durch Widerspruch“.

4.3. Vollständige Induktion

Um die Idee der vollständigen Induktion zu erklären, sei $A(n)$ eine Aussage über die natürliche Zahl n. Das Ziel ist, zu zeigen, dass die Aussage $A(n)$ für jedes $n \in \mathbb{N}$ wahr ist.

Induktionsprinzip

Wir benutzen das **Induktionsprinzip** folgendermaßen:

(1) Zuerst zeigt man, dass die Aussage $A(n)$ für die Zahl $n = 0$ gilt.

INDUKTIONSANFANG

Diesen Schritt nennt man **Induktionsanfang** bzw. Induktionsbasis.

(2) Danach zeigt man, dass für jede beliebige natürliche Zahl $n \in \mathbb{N}$ gilt: Falls die Aussage $A(n)$ wahr ist, so ist auch die Aussage $A(n+1)$ wahr.

INDUKTIONSSCHRITT

Diesen Schritt nennt man **Induktionsschritt**. Um den Induktionsschritt auszuführen, nimmt man an, dass die *Induktionsannahme* $A(n)$ wahr ist und muss $A(n+1)$ herleiten.

Beachte: Wenn man die Schritte (1) und (2) bewiesen hat, so weiß man, dass die folgenden Aussagen wahr sind:

(i) $A(0)$ ist wahr gemäß Schritt (1).

(ii) $A(1)$ ist wahr gemäß (i) und Schritt (2) für $n = 0$,

(iii) $A(2)$ ist wahr gemäß (ii) und Schritt (2) für $n = 1$,

(iv) $A(3)$ ist wahr gemäß (iii) und Schritt (2) für $n = 2$,

(v) $A(4)$ ist wahr gemäß (iv) und Schritt (2) für $n = 3$,

(vi) $A(5)$ ist wahr gemäß (v) und Schritt (2) für $n = 4$,

(vii) usw.

Insgesamt hat man damit gezeigt, dass **für alle** $n \in \mathbb{N}$ die Aussage $A(n)$ wahr ist. Warum?

Wir führen einen Beweis durch Widerspruch. Sei m die kleinste Zahl, so dass $A(m)$ nicht gilt. Aber dann muss auch $A(m-1)$ als Konsequenz des Induktionsschritts falsch sein und m ist nicht die kleinste Zahl gewesen. $\lightning$

Beispiel 4.9 (Es ist richtig dunkel). Wir befinden uns in einem stockdunklem Gang, der in einer Richtung unbeschränkt lang ist. Den Gang können wir nur in der anderen Richtung verlassen. Was tun, wenn wir noch nicht einmal die Länge **n** des Weges bis zum Ende des Ganges kennen? Wie können wir mit möglichst wenigen Schritten den Gang verlassen?

- Wie wär's mit: Einen Schritt nach „vorn", zwei zurück, drei nach vorn, vier zurück, ...
- Und wie viele Schritte benötigen wir, um den Gang zu verlassen?

Satz 4.10. $\displaystyle\sum_{i=1}^{n} i = 1 + 2 + \ldots + n = \frac{n \cdot (n+1)}{2}.$

Beweis: Wir geben zwei Argumente und beginnen mit der vollständigen Induktion nach n:

(a) Induktionsanfang für $n = 0$: Es ist $\sum_{i=1}^{0} i = 0$ und $\frac{0\cdot(0+1)}{2} = 0$. ✓

(b) Induktionsschritt von n nach $n+1$: Sei $n \in \mathbb{N}$ beliebig.

- Wir können die *Induktionsannahme* $\sum_{i=1}^{n} i = \frac{n\cdot(n+1)}{2}$ voraussetzen.
- $\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) \overset{\text{Ind.ann.}}{=} \frac{n\cdot(n+1)}{2} + (n+1) = \frac{n\cdot(n+1)+2\cdot(n+1)}{2} = \frac{(n+1)\cdot(n+2)}{2}$. ✓

Unser zweites Argument ist ein direkter Beweis: Wir betrachten ein Gitter mit n Zeilen und n Spalten. Das Gitter hat n^2 Gitterpunkte. Die Summe $\sum_{i=1}^{n} i$ stimmt überein mit der Anzahl der Gitterpunkte unterhalb der Hauptdiagonalen und auf der Hauptdiagonalen. Die Hauptdiagonale besitzt n Gitterpunkte und unterhalb der Hauptdiagonalen befindet sich die Hälfte der verbleibenden $n^2 - n$ Gitterpunkte. Also folgt

$$\sum_{i=1}^{n} i = n + \frac{n^2 - n}{2} = \frac{n \cdot (n+1)}{2} \checkmark$$

□

Wie viele Schritte müssen wir gehen, wenn n die Länge des Weges bis zum Ende des Gangs ist?

1. Nach $2k$ Wiederholungen sind wir insgesamt

$$(1-2)+(3-4)+\cdots+(2k-1-2k)=-k$$

Schritte nach vorn, also k Schritte zurückgegangen.

- Nach $2k+1$ Wiederholungen haben wir also $k+1$ Schritte nach vorn geschafft.
- Um den Gang zu verlassen, müssen wir insgesamt

$$1+2+\cdots+(2n-2)+(2n-1)=\frac{(2n-1)\cdot 2n}{2}=(2n-1)\cdot n$$

Schritte zurücklegen.

Bei quadratisch vielen Schritten werden wir mächtig erschöpft sein! Sind denn quadratisch viele Schritte wirklich notwendig? Alles auf eine Karte zu setzen, also nur in eine Richtung zu marschieren, ist Unfug. Aber können wir etwas mutiger sein, als immer nur einen weiteren Schritt zu wagen?

- Zum Beispiel, einen Schritt nach vorn, zwei zurück, vier nach vorn, acht zurück, ...,
- Wie viele Schritte brauchen wir diesmal?

Wir benötigen das folgende fundamentale Resultat über die geometrische Reihe. (Wir definieren $a^0 := 1$ für alle reellen Zahlen $a \in \mathbb{R}$.)

Satz 4.11 (Die geometrische Reihe). Für alle $n \in \mathbb{N}$ gilt $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$, falls $a \neq 1$ eine reelle Zahl ist.

Beweis: Sei $a \in \mathbb{R}$ von der Zahl 1 verschieden. Wir geben wieder zwei Argumente und beginnen mit der vollständigen Induktion nach n:

(a) Induktionsanfang für $n=0$: $\sum_{i=0}^{0} a^i = 1$ und $\frac{a^{0+1}-1}{a-1} = 1$. ✓

(b) Induktionsschritt von n auf $n+1$: Sei $n \in \mathbb{N}$ beliebig.

- Wir können die *Induktionsannahme* $\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1}$ voraussetzen. Dann ist
- $\sum_{i=0}^{n+1} a^i = \sum_{i=0}^{n} a^i + a^{n+1} \overset{\text{Ind.ann}}{=} \frac{a^{n+1}-1}{a-1} + a^{n+1} = \frac{a^{n+1}-1+a^{n+2}-a^{n+1}}{a-1} = \frac{a^{n+2}-1}{a-1}$ ✓

Auch hier ist unser zweites Argument ein direkter Beweis:

$$\begin{aligned}(a-1)\cdot\textstyle\sum_{i=0}^{n} a^i &= a\cdot\textstyle\sum_{i=0}^{n} a^i - \sum_{i=0}^{n} a^i \\ &= \textstyle\sum_{i=1}^{n+1} a^i - \sum_{i=0}^{n} a^i = a^{n+1} - a^0 = a^{n+1}-1\end{aligned}.$$

Jetzt dividiere durch $a-1$ und wir haben die Behauptung gezeigt. ✓ □

Der stockdunkle Gang: Vorher quadratisch viele Schritte, jetzt linear viele! Warum? Es gelte $2^{k-1} < n \leqslant 2^k$. Nach höchstens $1+2+\cdots+2^k+2^{k+1}+2^{k+2} = 2^{k+3}-1 \leqslant 16\cdot n - 1$ Schritten haben wir den rettenden Ausgang erreicht. □ Ende Beispiel 4.9

4.3.1. Rekursive Definitionen von Funktionen

Die Zahlen n_0 und k seien beliebig wie auch die Menge M. Das Induktionsprinzip lässt sich auch zur „induktiven“ (bzw. „rekursiven“) Definition von Funktionen $f\colon \mathbb{N} \to M$ nutzen, indem man folgendermaßen vorgeht:

(1) Definiere $f(0)$ (bzw. definiere $f(n_0), \ldots, f(n_0 + k)$).
Diesen Schritt bezeichnet man als **Rekursionsanfang**. REKURSIONSANFANG

(2) F.a. $n \in \mathbb{N}$ definiere $f(n+1)$ unter Verwendung des Werts $f(n)$ (bzw. f.a. $n \in \mathbb{N}$ mit $n \geqslant n_0 + k$ definiere $f(n+1)$ unter Verwendung der Werte $f(n_0), f(n_0+1), \ldots, f(n)$).
Diesen Schritt bezeichnet man als **Rekursionsschritt**. REKURSIONSSCHRITT

Wenn wir eine Aussage A über die Funktion f herleiten wollen, dann benötigen wir eine Variante der vollständigen Induktion, die an die rekursive Definition von f angepasst ist:

- INDUKTIONSANFANG für $n_0, \ldots, n_0+k$: Zeige, dass die Aussagen $A(n_0), A(n_0+1), \ldots, A(n_0+k)$ wahr sind.
- INDUKTIONSSCHRITT von n auf $n+1$: Zeige, dass $A(n+1)$ wahr ist, falls die Aussagen

  $$A(n_0), A(n_0+1), \ldots, A(n)$$

  wahr sind.

Beispiel 4.12. (Die XOR-Funktion). Wir geben eine rekursive Definition der Paritätsfunktion $p_n : \{0,1\}^n \to \{0,1\}$.

(a) REKURSIONSANFANG: Die Funktion $p_1 : \{0,1\} \to \{0,1\}$ wird definiert durch $p_1(x_1) := x_1$.

(b) REKURSIONSSCHRITT: Die Funktion $p_{n+1} : \{0,1\}^{n+1} \to \{0,1\}$ wird definiert durch

$$p_{n+1}(x_1, \ldots, x_{n+1}) := p_n(x_1, \ldots, x_n) \oplus x_{n+1}.$$

Wir zeigen, dass die Parität $p_n(x)$ von $x \in \{0,1\}^n$ genau dann 1 ist, wenn x eine ungerade Anzahl von Einsen besitzt. Diese Eigenschaft ist der Grund, dass die Parität im Entwurf fehlerkorrigierender Codes eingesetzt wird, denn das „Flippen“ irgendeines Bits ändert die Parität.

Satz 4.13. Für alle $n \in \mathbb{N}_{>0}$ und alle $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ gilt

$$p_n(x) = 1 \iff x \text{ hat ungerade viele Einsen.}$$

Beweis: Wir führen eine vollständige Induktion nach n aus. Sei $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ beliebig.

(a) INDUKTIONSANFANG für $n = 1$: Es ist $p_1(x) = p_1(x_1) = x_1$. Also gilt $p_1(x) = 1$ genau dann, wenn $x_1 = 1$, d.h. genau dann wenn $x = (x_1)$ eine ungerade Anzahl von Einsen hat. ✓

(b) INDUKTIONSSCHRITT von n nach $n+1$: Sei $n \in \mathbb{N}_{>0}$ beliebig. Zeige, dass für alle $x = (x_1, \ldots, x_n, x_{n+1}) \in \{0,1\}^{n+1}$ gilt: $p_{n+1}(x) = 1 \iff x$ hat ungerade viele Einsen.

– Wir können die *Induktionsannahme*

$$p_n(y_1, \ldots, y_n) = 1 \iff (y_1, \ldots, y_n) \text{ hat ungerade viele Einsen.}$$

für alle Tupel $(y_1, \ldots, y_n) \in \{0,1\}^n$ voraussetzen.

– Es ist $p_{n+1}(x_1, \ldots, x_{n+1}) = p_n(x_1, \ldots, x_n) \oplus x_{n+1}$. Also folgt

$$\begin{aligned} p_{n+1}(x) = 1 \iff & \big(p_n(x_1, \ldots x_n) = 1 \text{ und } x_{n+1} = 0\big) \text{ oder} \\ & \big(p_n(x_1, \ldots x_n) = 0 \text{ und } x_{n+1} = 1\big). \end{aligned}$$

Wir wenden die Induktionsvoraussetzung für

$$y_1 = x_1, \ldots, y_n = x_n$$

an und beachten, dass $p_n(x_1, \ldots, x_n) = 0$ genau dann gilt, wenn $(x_1, \ldots, x_n)$ keine ungerade Anzahl von Einsen, also eine gerade Anzahl von Einsen besitzt. Deshalb folgt

$$\begin{aligned} p_{n+1}(x) = 1 \iff & \big(\big(x_1, \ldots x_n\big) \text{ hat ungerade viele Einsen } \text{ und } x_{n+1} = 0\big) \text{ oder} \\ & \big(\big(x_1, \ldots x_n\big) \text{ hat gerade viele Einsen } \text{ und } x_{n+1} = 1\big) \\ \iff & x = (x_1, x_2, \ldots, x_{n+1}) \text{ hat ungerade viele Einsen} \end{aligned}$$

und das war zu zeigen. ✓

□

Beispiel 4.14 (Die Weizenkornlegende). [2]

Der Brahmane Sissa ibn Dahir lebte angeblich im dritten oder vierten Jahrhundert n. Chr. in Indien. Der indische Herrscher Shihram tyrannisierte damals seine Untertanen und stürzte sein Land in Not und Elend. Sissa erfand das Schachspiel (bzw. seine indische Urform Tschaturanga), um die Aufmerksamkeit von Shihram auf seine Fehler zu lenken, ohne ihn dabei zu erzürnen:

> Der König ist die wichtigste Figur, kann aber ohne Hilfe der anderen Figuren nichts ausrichten.

Als Dank für die anschauliche, aber auch zugleich unterhaltsame Lehre, gewährte Shihram dem Brahmanen einen freien Wunsch. Sissa wünschte sich Weizenkörner:

- Auf das erste Feld eines Schachbretts wollte er ein Korn,
- auf das zweite Feld das doppelte, also zwei,
- auf das dritte wiederum die doppelte Menge, also vier und so weiter.

Shihram lachte und war gleichzeitig erbost über die vermeintliche Bescheidenheit des Brahmanen. Als sich Shihram einige Tage später erkundigte, ob Sissa seine Belohnung in Empfang genommen habe, hatten die Rechenmeister die Menge der Weizenkörner noch nicht berechnet.

Der Vorsteher der Kornkammer meldete nach mehreren Tagen ununterbrochener Arbeit, dass er diese Menge Getreidekörner im ganzen Reich nicht aufbringen könne: Auf allen Feldern eines Schachbretts zusammen wären es 18.446.744.073.709.551.615 (≈ 18,45 Trillionen) Weizenkörner.

Nun stellte Shihram sich die Frage, wie das Versprechen eingelöst werden könne. Der Rechenmeister half dem Herrscher aus der Verlegenheit, indem er ihm empfahl, er solle Sissa ibn Dahir ganz einfach das Getreide Korn für Korn zählen lassen.

Statt auf einem Schachbrett betrachten wir das Wachstum auf einem unbeschränkt langen, eindimensionalem Brett und definieren die Funktion $f : \mathbb{N}_{>0} \to \mathbb{N}$ durch

$$f(n) = \text{Anzahl der Weizenkörner auf dem } n\text{'ten Feld}$$

Wir geben eine rekursive Definition von f an.

[2] Informationen aus Wikipedia

(a) Rekursionsanfang: Es ist $f(1) = 1$ und

(b) Rekursionsschritt: Für alle $n \in \mathbb{N}_{>0}$ ist $f(n+1) = 2 \cdot f(n)$.

Wir können einen einfachen Ausdruck für $f(n)$ finden: Die Anzahl der Weizenkörner auf dem n-ten Feld erhalten wir durch $(n-1)$-maliges Verdoppeln. Es „sollte“ $f(n) = 2^{n-1}$ gelten! Wir verifizieren „$f(n) = 2^{n-1}$“ durch vollständige Induktion.

(a) Induktionsanfang $n = 1$: Es ist $f(1) = 1 = 2^{1-1}$✓

(b) Induktionsschritt $n \to n+1$: Sei $n \in \mathbb{N}$ beliebig. $f(n+1) = 2 \cdot f(n) \overset{\text{Ind.ann}}{=} 2 \cdot 2^{n-1} = 2^n$ ✓

Satz 4.15. Die Anzahl der Sissa zustehenden Weizenkörner ist $1 + 2 + 4 + \cdots + 2^{63} = 2^{64} - 1$.

Beispiel 4.16 (Die Fakultät). Ein Rennen mit n Teilnehmern findet statt. Wie viele verschiedene Reihenfolgen gibt es für den Zieleinlauf? Wir beschreiben die Anzahl $\text{fak}(n)$ der verschiedenen Reihenfolgen mit einer rekursiven Definition.

- Rekursionsanfang für $n = 1$: Es ist $\text{fak}(1) = 1$.
- Wir überlegen uns zuerst, dass es bei $n+1$ Teilnehmern genau $n+1$ mögliche Gewinner des Rennens gibt. Wenn wir aber den Gewinner kennen, dann gibt es genau $\text{fak}(n)$ verschiedene Reihenfolgen für den Zieleinlauf der verbleibenden n Teilnehmer.

  Rekursionsschritt $n \to n+1$: Sei $n \in \mathbb{N}_{>0}$ beliebig. Definiere $\text{fak}(n+1) := (n+1) \cdot \text{fak}(n)$.

Wir behaupten, dass

$$\text{fak}(n) \quad = \quad n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1 \quad \overset{\text{Not.2.41}}{=} \quad \prod_{i=1}^{n} i$$

gilt und beweisen diese Behauptung mit vollständiger Induktion. Der Induktionsanfang für $n = 1$ ist klar, denn nach Definition ist $\text{fak}(1) = 1$.

Induktionsschritt $n \to n+1$. Sei $n \in \mathbb{N}_{>0}$ beliebig. Nach Induktionsannahme ist $\text{fak}(n) = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$ und die Behauptung $\text{fak}(n+1) = (n+1) \cdot n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1$ folgt aus der Definition der Fakultätsfunktion. □ Ende Beispiel 4.16

Notation:
Die Funktion $\text{fak} : \mathbb{N}_{>0} \to \mathbb{N}$ wird **Fakultät** genannt. Meistens schreibt man $n!$ statt $\text{fak}(n)$ und spricht $n!$ als „n Fakultät“ aus. Man definiert $0! := 1$.

Fakultät
$n!$

Definition 4.17. M sei eine endliche Mengen. Eine bijektive Funktion $f : M \to M$ wird auch eine **Permutation** der Menge M genannt.

Permutation

Wie viele Permutationen besitzt eine endliche Menge M?

Satz 4.18. M, M_1, M_2 seien Mengen mit jeweils n Elementen.

(a) Es gibt genau $n!$ bijektive Funktionen $f : M_1 \to M_2$.

(b) Die Menge M besitzt genau $n!$ Permutationen.

Wenn M_1 die Menge der n Teilnehmer eines Rennens ist, dann entspricht eine bijektive Funktion $f : M_1 \to \{1, \ldots, n\}$ einer möglichen Reihenfolge im Zieleinlauf. Teil (a) verallgemeinert also unser Ergebnis über die Anzahl verschiedener Zieleinläufe. Beachte, dass auch Teil (b) eine Konsequenz von Teil (a) ist, wenn wir nämlich $M_1 := M$ und $M_2 := M$ setzen.

Beweis: Wir können unsere Argumentation für die Anzahl der verschiedenen Reihenfolgen beim Zieleinlauf übernehmen, denn die Anzahl $b(n)$ der bijektiven Funktionen zwischen zwei n-elementigen Mengen besitzt die rekursive Definition

$$b(1) := 1,\ b(n+1) := (n+1) \cdot b(n)$$

und diese rekursive Definition stimmt mit der rekursiven Definition der Fakultät überein. □

Frage: Ein Handlungsreisender muss jede von n Städten genau einmal besuchen. Gesucht ist eine Rundreise minimaler Länge.

Wie viele verschiedene Rundreisen gibt es?

Beispiel 4.19 (Die Fibonacci-Zahlen). Ein Bauer züchtet Kaninchen. Jedes (weibliche) Kaninchen bringt im Alter von zwei Monaten ein (weibliches) Kaninchen zur Welt und danach jeden Monat ein weiteres.
Wie viele Kaninchen hat der Bauer am Ende des n-ten Monats, wenn er mit einem neu geborenen Kaninchen startet? Diese Anzahl bezeichnen wir mit $\mathrm{fib}(n)$.

Wie schnell wächst $\mathrm{fib}(n)$? Können wir sogar einen expliziten Ausdruck für $\mathrm{fib}(n)$ bestimmen? Um diese Fragen beantworten zu können, geben wir zuerst eine rekursive Definition.

- Rekursionsanfang für $n = 1$ und $n = 2$: Es ist $\mathrm{fib}(1) := 1$ und $\mathrm{fib}(2) := 1$.
- Rekursionsschritt von $n-1$ und n nach $n+1$: Es ist $\mathrm{fib}(n+1) := \mathrm{fib}(n) + \mathrm{fib}(n-1)$ f.a. $n \in \mathbb{N}$ mit $n \geqslant 2$.

  Warum? Genau die $\mathrm{fib}(n-1)$ Kaninchen, die sich im Monat $n-1$ im Besitz des Bauern befinden, haben jeweils einen Nachkommen im Monat $n+1$. Des Weiteren besitzt der Bauer $\mathrm{fib}(n)$ Kaninchen im Monat n und diese Kaninchen bleiben auch im Monat $n+1$ in seinem Besitz.

Somit gilt:

n	1	2	3	4	5	6	7	8	9	10	11	12
$\mathrm{fib}(n)$	1	1	2	3	5	8	13	21	34	55	89	144

Fibonacci-Folge

Die Funktion fib wird **Fibonacci-Folge** genannt; sie ist benannt nach dem italienischen Mathematiker Leonardo Fibonacci (13. Jh.). Die Zahl $\mathrm{fib}(n)$ heißt **n-te Fibonacci-Zahl**.

Satz 4.20. Sei $\mathrm{fib}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ die Fibonacci-Folge. Dann gilt f.a. $n \in \mathbb{N}_{>0}$: $\mathrm{fib}(n) \leqslant 2^n$.

Beweis: Per Induktion nach n.

INDUKTIONSANFANG: Betrachte $n = 1$ und $n = 2$.

Behauptung: $\text{fib}(1) \leqslant 2^1$ und $\text{fib}(2) \leqslant 2^2$.

Beweis: Es gilt: $\text{fib}(1) \overset{\text{Def.}}{=} 1 \leqslant 2 = 2^1$ und $\text{fib}(2) \overset{\text{Def.}}{=} 1 \leqslant 4 = 2^2$.

INDUKTIONSSCHRITT: $n-1, \to n+1$

Sei $n \in \mathbb{N}$ mit $n \geqslant 2$ beliebig.

Induktionsannahme: Es gilt $\text{fib}(n-1) \leqslant 2^{n-1}$ und $\text{fib}(n) \leqslant 2^n$.

Behauptung: $\text{fib}(n+1) \leqslant 2^{n+1}$.

Beweis: $\text{fib}(n+1) \overset{\text{Def.}}{=} \text{fib}(n) + \text{fib}(n-1) \overset{\text{Ind.ann.}}{\leqslant} 2^n + 2^{n-1} \leqslant 2 \cdot 2^n = 2^{n+1}$. □

Analog zeigt man auch $2^{n/2} \leqslant \text{fib}(n)$ für $n \geqslant 6$: Die Fibonacci-Folge hat also ein „exponentielles Wachstum“!

Bemerkung 4.21. Es gibt auch einen expliziten Ausdruck für die n-te Fibonacci-Zahl. F.a. $n \in \mathbb{N}_{>0}$ gilt nämlich:

$$\text{fib}(n) = \frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right).$$

Beweis: Man zeigt die Bemerkung durch vollständige Induktion nach n. Die zentrale Beobachtung ist, dass $\frac{1+\sqrt{5}}{2}$ und $\frac{1-\sqrt{5}}{2}$ die beiden Lösungen der quadratische Gleichung $x^2 = x + 1$ sind. □

4.3.2. Rekursive Definitionen von Mengen

Oft ist es sinnvoll, auch **Mengen** rekursiv zu definieren. Eine rekursive Definition einer Menge M besteht aus:

(a) **Basisregeln** der Form „$m \in M$“.

(D.h. die Basisregeln führen explizit genannte Elemente auf, die zur Menge M gehören.)

(b) **Rekursiven Regeln** der Form:

„Wenn $m_1, \ldots, m_k \in M$, dann $m \in M$“,

wobei m von $m_1, \ldots, m_k$ abhängt.

Die dadurch definierte Menge M ist dann die Menge aller Elemente, deren Zugehörigkeit zu M durch endlich-maliges Anwenden der Regeln gezeigt werden kann.

Wir haben bereits zahlreiche Beispiele rekursiver Definitionen kennengelernt. In Definition 3.3 wird die Menge AL der aussagenlogischen Formeln rekursiv eingeführt. Nachfolgend haben wir die Funktion $[\![\cdots]\!]^{\mathcal{B}}$ in Definition 3.14 rekursiv definiert.

Beispiel 4.22 (Die Menge PAL).
Betrachte das Alphabet $A := \{a, b\}$. Die Menge $\text{PAL} \subseteq A^*$ sei wie folgt rekursiv definiert:

Basisregeln:

(B1): $\varepsilon \in \text{PAL}$.

(B2): $a \in \text{PAL}$.

(B3): $b \in \text{PAL}$.

Rekursive Regeln:

(R1): Ist $w \in \text{PAL}$, so ist auch $awa \in \text{PAL}$.

(R2): Ist $w \in \text{PAL}$, so ist auch $bwb \in \text{PAL}$.

Beispiele für Worte, die zur Menge PAL gehören:

$$\underbrace{\varepsilon,\ a,\ b}_{\text{durch Basisregeln}} \qquad \underbrace{aa,\ bb}_{\text{durch rek. Regeln mit } w := \varepsilon} \qquad \underbrace{aaa,\ bab}_{\text{durch rek. Regeln mit } w := a} \qquad \underbrace{aba,\ bbb}_{\text{durch rek. Regeln mit } w := b}$$

Es gilt beispielsweise auch: $aababaa \in \text{PAL}$.

Beweis:

- $a \in \text{PAL}$ (gemäß Basisregel (B1)).
- Regel (R2) mit $w := a \implies bab \in \text{PAL}$.
- Regel (R1) mit $w := bab \implies ababa \in \text{PAL}$.
- Regel (R1) mit $w := ababa \implies aababaa \in \text{PAL}$.

□

Aber beispielsweise gilt

$$aab \notin \text{PAL},$$

denn aus den Basisregeln und den rekursiven Regeln folgt, dass für jedes Wort $w \in \text{PAL}$ der erste und der letzte Buchstabe von w identisch sind. □Ende Beispiel 4.22

Induktion über den Aufbau einer rekursiv definierten Menge

Sei M eine rekursiv definierte Menge. Dass eine Aussage $A(m)$ für alle $m \in M$ wahr ist, kann man mit einem induktiven Beweis über den Aufbau von M zeigen:

(1) Zuerst betrachtet man nacheinander jede Basisregel der Form „$m \in M$“ und zeigt, dass die Aussage $A(m)$ wahr ist.
Dieser Schritt heißt **Induktionsanfang**.

(2) Danach betrachtet man nacheinander jede rekursive Regel der Form „Wenn $m_1, \ldots, m_k \in M$, dann $m \in M$“ und zeigt Folgendes: Wenn die Aussagen $A(m_1), \ldots, A(m_k)$ wahr sind, dann ist auch die Aussage $A(m)$ wahr.
Dieser Schritt heißt **Induktionsschritt**.

Beachte: Man kann leicht sehen, dass Folgendes gilt: Wenn man die Schritte (1) und (2) bewiesen hat, so weiß man, dass die Aussage $A(m)$ für alle $m \in M$ wahr ist.

Beispiel 4.23 (Palindrome).
Sei $A := \{a, b\}$. Für jedes Wort $w \in A^*$ sei w^R das Wort, das durch „Rückwärtslesen“ von w entsteht, d.h.:

- Ist $w = \varepsilon$, so ist $w^R = \varepsilon$.
- Ist $w = w_1 \cdots w_k$ mit $k \in \mathbb{N}_{>0}$ und $w_1, \ldots, w_k \in A$, so ist $w^R := w_k \cdots w_1$.

Beispiel: $(aaab)^R = baaa$.

Sei PAL die im Beispiel 4.22 rekursiv definierte Teilmenge von A^*.

Behauptung 1: Für jedes Wort $w \in \text{PAL}$ gilt: $w = w^R$.

Beweis: Per Induktion über den Aufbau von PAL.

INDUKTIONSANFANG: Betrachte diejenigen Worte, die aufgrund von Basisregeln zur Menge PAL gehören.
Behauptung: $\varepsilon = \varepsilon^R$, $a = a^R$ und $b = b^R$.
Beweis: Gemäß der Definition von w^R gilt offensichtlich, dass $\varepsilon = \varepsilon^R$, $a = a^R$ und $b = b^R$.

INDUKTIONSSCHRITT: Betrachte die rekursiven Regeln.

- (R1): Sei $w \in \text{PAL}$ und sei $v := awa$. Gemäß (R1) ist $v \in \text{PAL}$.

  Induktionsannahme: $w = w^R$.

  Behauptung: $v = v^R$.

  Beweis: $v^R \overset{\text{Def. } v}{=} (awa)^R \overset{\text{Def. } (\cdot)^R}{=} aw^Ra \overset{\text{Ind.ann.: } w = w^R}{=} awa \overset{\text{Def. } v}{=} v$.
- (R2): verläuft analog zu (R1).

Behauptung 2: Für jedes $w \in A^*$ mit $w = w^R$ gilt: $w \in \text{PAL}$.

Beweisansatz: Zeige folgende Aussage per Induktion nach n:

> Für alle $n \in \mathbb{N}$ gilt: Ist $w \in A^*$ mit $w = w^R$ und $|w| \leqslant n$, so gilt $w \in \text{PAL}$.

Im Induktionsanfang werden $n = 0$ und $n = 1$ betrachtet; im Induktionsschritt $n \to n + 1$ werden alle $n \geqslant 1$ betrachtet.
Details: übung. □ Beh. 2

Aus Behauptung 1 und Behauptung 2 folgt $\text{PAL} = \{w \in A^* : w = w^R\}$.

□ Ende Beispiel 4.23

4.3.3. Analyse rekursiver Programme

Beispiel 4.24. Wir möchten die n-te Fibonacci-Zahl $\text{fib}(n)$ berechnen und tun dies mit zwei verschiedenen Algorithmen. Welcher Algorithmus ist schneller?

> Algo1(n):
> 1. Falls $n = 1$, dann gib Algo1(1) := 1 als Ergebnis zurück.
> 2. Falls $n = 2$, dann gib Algo1(2) := 1 als Ergebnis zurück.

3. Falls $n \geqslant 3$, dann gib Algo1(n) := Algo1$(n-1)$ + Algo1$(n-2)$ als Ergebnis zurück.

Wenn wir jede Addition, jeden Vergleich, und jedes Zurückgeben eines Ergebnisses als einen Schritt zählen, dann benötigt dieser rekursive Algorithmus bei Eingabe einer Zahl n genau $g_1(n)$ Schritte, wobei

$$\begin{aligned} g_1(1) &= 2 \quad \text{und} \quad g_1(2) = 3 \quad \text{und} \\ g_1(n) &= 3 + g_1(n-1) + g_1(n-2) + 2 \\ &= 5 + g_1(n-1) + g_2(n-2) \quad \text{für alle } n \in \mathbb{N} \text{ mit } n \geqslant 3. \end{aligned}$$

Ein anderer Algorithmus, der den Wert fib(n) berechnet, ist:

Algo2(n):

1. Falls $n = 1$ oder $n = 2$, dann gib 1 als Ergebnis zurück.
2. Seien $a_0 := 0$, $a_1 := 1$ und $a_2 := 1$.
3. Wiederhole für alle i von 3 bis n:
4. Ersetze a_0 durch a_1 und a_1 durch a_2.
5. Ersetze a_2 durch $a_0 + a_1$.
6. Gib den Wert a_2 als Ergebnis zurück.

Dieser Algorithmus benötigt bei Eingabe $n \in \mathbb{N}_{>0}$ genau $g_2(n) := 6 + 5 \cdot (n-2)$ Schritte. (ähnlich wie oben zählen wir jeden Vergleich, jedes Zurückgeben eines Werts und jedes Setzen eines Werts als einen Schritt. Für jeden Schleifendurchlauf berechnen wir zusätzlich zwei Schritte, um den Wert von i um eins zu erhöhen und zu testen, ob das Ergebnis kleiner oder gleich n ist).

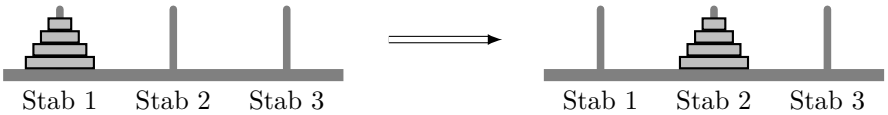
Frage: Welcher der beiden Algorithmen läuft im Allgemeinen schneller? D.h. welche der beiden Funktionen g_1 und g_2 liefert kleinere Funktionswerte?

Offensichtlich gilt $g_1(n) \geqslant \text{fib}(n)$. Wir wissen aber schon aus Beispiel 4.19, dass $\text{fib}(n) \geqslant 2^{n/2}$ für $n \geqslant 6$ gilt. Der elegante rekursive Algo1 mit exponentiell in n wachsender Laufzeit ist absolut gräßlich, während sein unscheinbarer Kollege Algo2 mit einer Laufzeit glänzt, die proportional zu n wächst.

Eine wichtige Anwendung der vollständigen Induktion ist die Verifikation rekursiver Programme. Hier ist ein erstes Beispiel.

Beispiel 4.25 (Die Türme von Hanoi). Wir haben drei Stäbe mit den Nummern 1, 2 und 3. Ursprünglich liegen N Ringe auf Stab 1, wobei die Ringe in absteigender Größe auf dem Stab aufgereiht sind: Der größte Ring von Stab 1 ist also der unterste Ring. Die Stäbe 2 und 3 sind zu Anfang leer.

In einem Zug können wir einen zuoberst liegenden Ring von einem Stab auf einen anderen bewegen. Der Zug ist nur dann erlaubt, wenn der Ring auf einen größeren Ring gelegt wird oder wenn der Stab leer ist. Alle Ringe sollen am Ende auf Stab 2 liegen.



Das folgende in Pseudocode geschriebene rekursive Programm soll dieses Ziel für $N \geqslant 1$ erreichen.

```
void Hanoi( int N, int stab1, int stab2, int stab3)
// Annahmen: Auf allen Stäben sind die Ringe der Größe nach geordnet.
// Jeder der oberen N Ringe auf Stab „stab1“ passt auf Stab „stab2“ und „stab3“.
// Die Folge (stab1,stab2,stab3) ist eine Permutation der Zahlen 1,2 und 3.
{if (N==1)
    bewege den obersten Ring von Stab „stab1“ nach Stab „stab2“;
else {
    Hanoi(N-1,stab1,stab3,stab2);
    bewege den obersten Ring von Stab „stab1“ nach Stab „stab2“;
    Hanoi(N-1,stab3,stab2,stab1); }}
```

Wir zeigen mit vollständiger Induktion nach N: Für jede Permutation (stab1,stab2,stab3) der drei Stäbe wird „Hanoi (N, stab1, stab2, stab3)“ die obersten N Ringe von stab1 auf stab2 bewegen ohne andere Ringe anzufassen. Vorausgesetzt wird, dass jeder der obersten N Ringe von stab1 auf die beiden anderen Stäbe passt.

Beachte, dass wir eine stärkere Aussage behaupten als auf den ersten Blick notwendig zu sein scheint: Wir behaupten nämlich die Richtigkeit für alle(!) Permutationen der drei Stäbe. Diese verschärfte Behauptung ist auch notwendig, um Aussagen über die beiden rekursiven Aufrufe machen zu können. Die Formulierung einer verschärften Aussage ist charakteristisch für viele Induktionsbeweise.

Induktionsanfang für $N = 1$. Richtigerweise wird der eine zuoberst liegende Ring von „stab1“ nach „stab2“ bewegt.

Induktionsschritt von N auf $N + 1$: Wir können in der *Induktionsannahme* voraussetzen, dass „Hanoi(N, stab1, stab2, stab3)“ – für jede Permutation (stab1,stab2,stab3) der drei Stäbe – die N obersten Ringe von „stab1“ nach „stab2“ bewegt ohne andere Ringe anzufassen.

Im ersten rekursiven Aufruf wird „Hanoi(N, stab1, stab3, stab2)“, nach Induktionsannahme, die obersten N Ringe von „stab1“ nach „stab3“ bewegen. Der jetzt zuoberst liegende Ring von „stab1“ wird auf „stab2“ gelegt: Nach Annahme passt dieser Ring auf „stab2“.

Da dieser Ring der größte der ursprünglichen $N + 1$ obersten Ringe von „stab1“ ist, passen alle jetzt auf „stab3“ hinzu gepackten Ringe auf „stab1“ und „stab2“. Der zweite und letzte rekursive Aufruf „Hanoi(N, stab3, stab2, stab1)“ wird deshalb nach Induktionsannahme alle durch den ersten rekursiven Aufruf auf „stab3“ bewegten Ringe erfolgreich auf „stab2“ bewegen: Damit liegen die ursprünglich obersten $N + 1$ Ringe von „stab1“ jetzt auf „stab2“.

Beispiel 4.26 (Türme von Hanoi: Die Anzahl der Ringbewegungen). Sei $T(N)$ die Anzahl der Ringbewegungen nach Aufruf des Programms Hanoi(N,stab1,stab2,stab3). (Beachte, dass diese Anzahl nicht von der Permutation (stab1,stab2,stab3) abhängt.) Wir geben eine rekursive Definition von $T(N)$ an.

(a) Rekursionsanfang: Es ist $T(1) = 1$ und

(b) Rekursionsschritt: Es ist $T(N) = 2 \cdot T(N - 1) + 1$.

Und wie sieht ein expliziter Ausdruck für $T(N)$ aus? Es ist $T(1) = 1, T(2) = 3, T(3) = 7, T(4) = 15$ und das sieht ganz so aus als ob $T(N) = 2^N - 1$ gilt. Wir verifizieren unsere Vermutung mit vollständiger Induktion.

(a) Induktionsanfang für $N = 1$: Unser Programm bewegt einen Ring und $1 = 2^1 - 1$. ✓

(b) Induktionsschritt von N auf $N+1$: Wir können in der *Induktionsannahme* voraussetzen, dass $T(N) = 2^N - 1$ gilt. Dann folgt

$$T(N+1) = 2 \cdot T(N) + 1 \overset{\text{Ind.ann}}{=} 2 \cdot (2^n - 1) + 1 = 2^{N+1} - 1.$$

Beispiel 4.27 (Binärsuche). Ein Array $A = (A[1], \ldots, A[n])$ von n Zahlen und eine Zahl x ist gegeben. Wir möchten wissen, ob und wenn ja wo die Zahl x in A vorkommt.

lineare Suche

Binärsuche

Wenn A nicht sortiert ist, dann bleibt uns nichts anderes übrig als uns alle Zellen von A auf der Suche nach x anzuschauen. Wir führen eine **lineare Suche** durch, die im schlimmsten Fall alle n Zellen des Arrays inspizieren muss. Wenn das Array aber aufsteigend sortiert ist, dann können wir **Binärsuche** anwenden. Hier ist ein C++ Programm für die Binärsuche:

```
void Binärsuche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
     std::cout « x « " wurde in Position "
      « mitte « " gefunden." « std::endl;
     else {
       if (x < A[mitte])
         Binärsuche(unten, mitte-1);
       else
         Binärsuche(mitte+1,oben);}}
```

Es sei $n = \mathit{oben} - \mathit{unten} + 1$. Wir fordern $n = 2^k - 1$ für eine Zahl $k \in \mathbb{N}$. Sei $T(n)$ die maximale Anzahl von Zellen, die Binärsuche für ein sortiertes Array von n Zahlen inspiziert.

Frage: Wie groß ist $T(n)$?

Hier ist eine rekursive Definition von $T(n)$:

(a) Rekursionsanfang: $T(0) = 0$.

(b) Rekursionsschritt: $T(n) = T(\frac{n-1}{2}) + 1$.

Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt. Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt und $T(2^k - 1) = k$ „sollte“ folgen. Wir zeigen $T(2^k - 1) = k$ mit vollständiger Induktion nach k.

(a) Induktionsanfang: $T(2^0 - 1) = T(0) = 0$. ✓

(b) Induktionsschritt: $T(2^{k+1} - 1) = T(2^k - 1) + 1 \overset{\text{Induktionsannahme}}{=} k + 1$. ✓

Und was bedeutet das jetzt? Binärsuche muss höchstens k Zahlen inspizieren gegenüber bis zu $2^k - 1$ Zahlen für die lineare Suche: Die lineare Suche ist exponentiell langsamer als Binärsuche.
□ Ende von Beispiel 4.27

4.3.4. Was so alles schiefgehen kann

Das folgende Beispiel zeigt, dass man beim Führen von Induktionsbeweisen sehr sorgfältig sein muss:

Beispiel 4.28.
Der folgende Satz ist offensichtlich nicht wahr, aber wo steckt der Fehler im Beweis?

„Satz“: *F.a. $n \in \mathbb{N}$ mit $n \geqslant 1$ gilt: Ist M eine Menge von Menschen mit $|M| = n$, so haben alle Menschen in M die gleiche Größe.*

„Beweis“: Durch vollständige Induktion nach n.

Induktionsanfang: $n = 1$

Behauptung: Ist M eine Menge von Menschen mit $|M| = 1$, so haben alle Menschen in M die gleiche Größe.

Beweis: Sei M eine Menge von Menschen mit $|M| = 1$. d.h. M besteht aus genau einem Menschen. Daher haben offensichtlich alle Menschen in M die gleiche Größe.

Induktionsschritt: $n \to n + 1$

Sei $n \in \mathbb{N}$ mit $n \geqslant 1$ beliebig.

Induktionsannahme: Ist M' eine Menge von Menschen mit $|M'| = n$, so haben alle Menschen in M' die gleiche Größe.

Behauptung: Ist M eine Menge von Menschen mit $|M| = n + 1$, so haben alle Menschen in M die gleiche Größe.

Beweis: Sei M eine Menge von Menschen mit $|M| = n+1$. Sei $a_1, a_2, \ldots, a_n, a_{n+1}$ eine Liste aller Menschen in M, d.h. $M = \{a_1, a_2, \ldots, a_n, a_{n+1}\}$. Sei

$$M' := \{a_1, a_2, \ldots, a_n\} \quad \text{und} \quad M'' := \{a_2, \ldots, a_n, a_{n+1}\}.$$

Offensichtlich sind M' und M'' Mengen von Menschen mit $|M'| = |M''| = n$. Nach Induktionsannahme gilt:

(1) Alle Menschen in M' haben die gleiche Größe, und

(2) alle Menschen in M'' haben die gleiche Größe.

Sei g' die Größe, die gemäß (1) jeder Mensch in M' hat, und sei g'' die Größe, die gemäß (2) jeder Mensch in M'' hat. Laut Definition von M' und M'' gilt: $a_2 \in M'$ und $a_2 \in M''$. Da jeder einzelne Mensch (und daher insbesondere der Mensch a_2) nur *eine* Größe haben kann, gilt: $g' = g''$. Wegen $M = M' \cup M''$ gilt daher, dass alle Menschen in M die gleiche Größe haben, nämlich die Größe $g := g' = g''$. „ □“

Frage: Wo steckt der Fehler im Beweis?

Beispiel 4.29. Und hier ist der Beweis einer zweiten sinnlosen Aussage.

„**Satz**“: Für alle natürlichen Zahlen a und b gilt $a = b$.

„*Beweis*“: Wir setzen $k = \max\{a, b\}$ und führen eine vollständige Induktion nach k aus.

Induktionsanfang: $k = 0$.

Offensichtlich ist $a = 0 = b$, und das war zu zeigen.

INDUKTIONSSCHRITT: $k \to k+1$

Sei $\max\{a, b\} = k+1$. Wir können die Induktionsannahme voraussetzen, dass für alle a', b' mit $\max\{a', b'\} \leqslant k$ gilt $a' = b'$. Wir wenden die Induktionsannahme für $a' = a - 1$ und $b' = b - 1$, denn $\max\{a-1, b-1\} = k$. Also ist $a' = b'$, bzw. $a - 1 = b - 1$ und die Behauptung $a = b$ folgt. „ □“

Frage: Wo steckt der Fehler diesmal?

Beispiel 4.30. „**Satz**“: Für alle natürlichen Zahlen n gilt $n = n + 1$.

„Beweis“: Wir zeigen die Behauptung mit vollständiger Induktion nach n.

INDUKTIONSSCHRITT: $n \to n+1$. Wir können die Induktionsannahme „$n = n + 1$“ anwenden und müssen die Behauptung „$n + 1 = n + 2$“ zeigen. Die Behauptung folgt sofort, wenn wir 1 auf die linke wie auch die rechte Seite der Induktionsannahme addieren. „ □“

Frage: Wo ist der Fehler begraben?

4.4. Zusammenfassung und Ausblick

Was ist ein korrektes Argument? Diese Frage hat uns in diesem Kapitel beschäftigt. Wir haben verschiedene Beweismethoden, nämlich direkte Beweise, Beweise durch Kontraposition und Widerspruchsbeweise betrachtet.

Von großer Wichtigkeit ist die Beweismethode der vollständigen Induktion, die aus Induktionsanfang und Induktionsschritt besteht. Was im Induktionsanfang und was im Induktionsschritt zu zeigen ist, hängt von der Zielstellung ab: Wenn wir die Gleichung $\sum_{i=0}^{k} a^i = (a^{k+1} - 1)/(a - 1)$ für die geometrische Reihe zeigen wollen, zeigt der Induktionsanfang sinnvollerweise die Gleichung für $k = 0$ und der Induktionsschritt zeigt die Gleichung für $k + 1$ unter der Induktionsannahme, nämlich, dass die Gleichung für k richtig ist.

Möchten wir aber zeigen, dass $\text{fib}(k) \leqslant 2^k$ für die k-te Fibonacci-Zahl $\text{fib}(k)$ gilt, dann zeigt man im Induktionsanfang die Ungleichung für $k = 1$ und $k = 2$ und im Induktionsschritt für $k + 1$ unter der Induktionsannahme, dass die Ungleichung für $k - 1$ und k gilt. Möchten wir Aussagen über rekursiv definierte Funktionen machen, dann muss sich der Induktionsbeweis der Struktur der rekursiven Definition anpassen. Im Fall der Fibonacci-Zahlen zwingt uns die rekursive Definition $\text{fib}(k + 1) = \text{fib}(k - 1) + \text{fib}(k)$ zu dem gerade beschriebenen Vorgehen.

Wie schreibt man ein korrektes rekursives Programm? Indem man schon beim Entwurf des Programms einen Korrektheitsbeweis mit Hilfe der vollständigen Induktion im Hinterkopf hat. Wie analysiert man den Aufwand eines rekursiven Programms? Indem man die rekursiv definierte Laufzeit analysiert.

Systeme zu entwickeln ohne ihre Korrektheit nachzuweisen wird zu einer beruflich sehr kurzen Karriere führen. Deshalb ist dieses Kapitel so wichtig.

4.5. Literaturhinweise zu Kapitel 4

Als vertiefende Lektüre seien die Kapitel 3, 6 und 7 in [21] empfohlen. Wertvolle Tipps und Tricks zur Formulierung mathematischer Gedanken und Beweise finden sich in [2]. Kapitel 2 und 3 in [8]

beschäftigen sich mit Rekursion und vollständiger Induktion, die Beweisführung ist Inhalt von Kapitel 7. Einen Crashkurs in diskreter Mathematik für die Informatik gibt das Buch [13].

Quellennachweis: Teile der Abschnitte 2.2–2.4 sowie 4.3.1 orientieren sich an [9]. Das in Abschnitt 2.5 betrachtete Beispiel ist aus [14] entnommen. Teile von Abschnitt 4 orientieren sich an [21]. Die folgende Aufgabe 2.23 ist aus [14] entnommen.

4.6. Übungsaufgaben zu Kapitel 4

Aufgabe 4.1.

(a) Sei $n \in \mathbb{N}_{>0}$. Auf dem Tisch liegen nebeneinander Karten $k_1, k_2, \ldots, k_n$, die jeweils auf einer Seite mit einem Buchstaben aus der Menge {A, B, C, ..., Z} und auf der anderen Seite mit einer natürlichen Zahl beschriftet sind. Wir können auf dem Tisch jeweils nur eine der beiden Beschriftungen sehen und wollen testen, ob folgende Aussage wahr ist:

Steht auf einer Seite der Karte eine gerade Zahl, dann steht auf der anderen Seite ein Vokal.

Natürlich können wir jede Karte $k_1, k_2, \ldots, k_n$ umdrehen, um den Wahrheitsgehalt der Aussage zu überprüfen, aber es geht oft auch einfacher.

(i) Ein konkretes Beispiel: Die Karten liegen wie in oben abgebildet auf dem Tisch. Welche Karten müssen wir umdrehen? Begründen Sie Ihre Antwort.

(ii) Verallgemeinern Sie Ihre Antwort für beliebige Kartenfolgen $k_1, k_2, \ldots, k_n$.

(b) Betrachten Sie eine Färbung $f : \mathbb{N} \to \{\text{rot}, \text{blau}\}$ der natürlichen Zahlen. Folgende Aussage über f sei wahr: *Für jede blau gefärbte Zahl gibt es eine* ***größere*** *rot gefärbte Zahl.*

Welche der folgenden Aussagen kann man folgern?

(i) Es gibt eine blau gefärbte natürliche Zahl.

(ii) Für jede rot gefärbte natürliche Zahl gibt es eine kleinere blau gefärbte.

(iii) $f^{-1}(\text{rot}) = \{x \in \mathbb{N} : f(x) = \text{rot}\}$ ist unendlich.

(iv) $f^{-1}(\text{blau}) = \{x \in \mathbb{N} : f(x) = \text{blau}\}$ ist endlich.

(v) Wenn $f(1) = \text{blau}$ gilt, dann gibt es ein $n \in \mathbb{N}$, sodass $f(n) = \text{blau}$ und $f(n+1) = \text{rot}$ gilt.

(c) Formulieren Sie die Negation der folgenden Aussagen jeweils umgangssprachlich.

(i) Alle Katzen sind grau.

(ii) Wenn es Einhörner gibt, dann können alle Elefanten fliegen.

(iii) Für jeden Jedi gilt: Die Macht ist genau dann stark in ihm, wenn in seinem Blut Midi-Chlorianer zu finden sind.

Aufgabe 4.2. Sei $n \in \mathbb{N}$ mit $n \geqslant 2$. Zu Freds Geburtstagsfeier im Vereinsheim des Dackelzüchterclubs Wily Werewolf sind n Personen (inklusive Fred) erschienen. Manche dieser Personen kennen einander, andere wiederum nicht. Die Bekanntschaften sind symmetrisch, d. h. Person

i kennt Person j genau dann, wenn Person j Person i kennt. Bekanntschaften außerhalb des Vereinsheims werden nicht berücksichtigt.

Zeigen Sie: Es gibt mindestens zwei Personen, die gleichviele andere Personen kennen.

Aufgabe 4.3.

(a) Seien $a, b \in \mathbb{N}$. Beweisen Sie: $a \cdot b$ ist gerade $\iff$ a ist gerade oder b ist gerade.

(b) Beweisen oder widerlegen Sie:

(i) Für jedes gerade $k \in \mathbb{N}_{>0}$ und jedes $a \in \mathbb{Z}$ gilt:
$a + (a+1) + (a+2) + \cdots + (a+k-1)$ ist durch k teilbar.

(ii) Für jedes ungerade $k \in \mathbb{N}_{>0}$ und jedes $a \in \mathbb{Z}$ gilt:
$a + (a+1) + (a+2) + \cdots + (a+k-1)$ ist durch k teilbar.

Hinweis: Die Identität $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ kann hilfreich sein.

(c) Seien $n \in \mathbb{N}_{>0}$ und $x_1, x_2, \ldots, x_n \in \mathbb{R}$. Beweisen Sie: Gilt $\sum_{i=1}^{n} x_i = b$, dann gibt es ein $i \in \{1, 2, \ldots, n\}$ mit $x_i \geqslant \frac{b}{n}$.

Aufgabe 4.4. Seien a, b, c irrationale Zahlen, d. h. es gelte $a, b, c \in \mathbb{R} \setminus \mathbb{Q}$. Zeigen Sie: Mindestens eine der drei Zahlen $a + b$, $b + c$ oder $c + a$ ist irrational. *Hinweis:* Beweis durch Kontraposition oder Beweis durch Widerspruch

Aufgabe 4.5. Um Speicherplatz bzw. Bandbreite zu sparen, werden Daten in vielen Anwendungen komprimiert. Dabei wird die Originaldatei durch ein Kompressionsverfahren *komprimiert*; vor der Verwendung wird die Datei wieder *dekomprimiert*. Man unterscheidet zwei Arten von Kompressionsverfahren: Bei der verlustbehafteten Kompression, z. B. bei den Formaten MP3 und JPEG, wird nur ein Teil der Information der Originaldatei gespeichert. Dadurch kann eine gute Kompression erreicht werden, allerdings lässt sich die Originaldatei nicht identisch wiederherstellen. Bei der verlustfreien Kompression hingegen, z. B. bei den Formaten ZIP und PNG, kann die Originaldatei exakt wiederhergestellt werden. Jedes Kompressionsverfahren lässt sich als eine Funktion $f : \Sigma^* \to \Sigma^*$ modellieren: Eine Originaldatei $x \in \Sigma^*$ wird durch das Verfahren f auf die komprimierte Datei $f(x) \in \Sigma^*$ abgebildet. Ein *verlustfreies* Kompressionsverfahren ist zudem injektiv: Für je zwei unterschiedliche Dateien x, y muss $f(x) \neq f(y)$ gelten, damit eine eindeutige Dekompression möglich ist.

Wir wollen in dieser Aufgabe die Grenzen verlustfreier Kompressionsverfahren erkunden.

(a) Beweisen Sie: Sei f eine injektive Funktion $f : \{0,1\}^* \to \{0,1\}^*$. Existiert ein Wort $x \in \{0,1\}^*$ mit $|f(x)| < |x|$, so existiert auch ein Wort $y \in \{0,1\}^*$ mit $|f(y)| > |y|$.

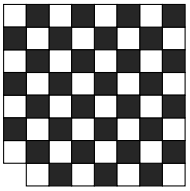
(b) Welche Erkenntnis zu verlustfreien Kompressionsverfahren können Sie aus der Aussage in Teilaufgabe (a) ableiten?

Aufgabe 4.6. Zeigen Sie: Die Menge $\mathrm{Abb}(\mathbb{N}, \mathbb{N})$ ist nicht abzählbar, d.h. es gibt keine surjektive Abbildung von $\mathbb{N}$ nach $\mathrm{Abb}(\mathbb{N}, \mathbb{N})$.

Hinweis: Passen Sie das Diagonalargument aus dem Beweis für die Nichtabzählbarkeit von $\mathcal{P}(\mathbb{N})$ (Satz 2.46) an.

Aufgabe 4.7. Gegeben ist ein 8×8-Schachbrett, bei dem eine Ecke entfernt worden ist. Ihnen stehen Kacheln der Form $\square\square\square$ zur Verfügung. Kacheln dürfen gedreht werden, dürfen sich aber nicht überlappen. Beweisen Sie: Es gibt *keine* Kachelung des Schachbrettes.

Hinweis: Färben Sie die Felder in geeigneter Weise mit drei verschiedenen Farben.



Aufgabe 4.8. Sei $f_0 = 0, f_1 = 1, f_2 = 1, \ldots$ die Folge der Fibonaccizahlen. Zeigen Sie mit vollständiger Induktion:

(a) Für alle $n \in \mathbb{N}_{>0}$ gilt: $f_{2n} = f_n \cdot (f_{n+1} + f_{n-1})$.

(b) Für alle $n \in \mathbb{N}_{>0}$ gilt: $\sum_{i=1}^{n} f_i^2 = f_n \cdot f_{n+1}$.

Aufgabe 4.9.

Wir definieren die aussagenlogische Formel φ_n für alle $n \in \mathbb{N}_{>0}$ wie folgt:

$$\begin{aligned} \varphi_1 &:= V_1 \\ \varphi_{n+1} &:= \big(\varphi_n \to V_{n+1}\big) \text{ für alle } n \in \mathbb{N}_{>0} \end{aligned}$$

Betrachten Sie die Belegung $\mathcal{B} : \textsc{Avar} \to \{0,1\}$ mit $\mathcal{B}(V_i) := \begin{cases} 1, & i \text{ ist ungerade,} \\ 0, & i \text{ ist gerade.} \end{cases}$

(i) Bestimmen Sie die Formel φ_4.

(ii) Berechnen Sie $[\![\varphi_3]\!]^{\mathcal{B}}$.

(iii) Zeigen Sie mit vollständiger Induktion über n: Für alle $n \in \mathbb{N}_{>0}$ gilt $[\![\varphi_n]\!]^{\mathcal{B}} = \mathcal{B}(V_n)$.

Aufgabe 4.10. Beweisen Sie Folgendes durch vollständige Induktion nach n.

(a) Für alle $n \in \mathbb{N}$ mit $n \geqslant 5$ gilt: $2^n > n(n+1)$.

(b) Für alle $n \in \mathbb{N}$ mit $n \geqslant 1$ gilt: $\sum_{i=1}^{n} (2i-1) = n^2$.

(c) Für alle $n \in \mathbb{N}$ mit $n \geqslant 1$ gilt: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

(d) Für alle $n \in \mathbb{N}$ mit $n \geqslant 1$ gilt: $\sum_{i=1}^{n} (4i-1) = 2n^2 + n$.

(e) Für alle $n \in \mathbb{N}$ mit $n \geqslant 2$ gilt: $\prod_{i=1}^{n} i^{\,i} \leqslant n^{n(n-1)}$

(f) Für alle $x \in \mathbb{R}$ mit $x \geqslant -1$ und alle $n \in \mathbb{N}$ mit $n \geqslant 1$ gilt: $1 + n \cdot x \leqslant (1+x)^n$.

Aufgabe 4.11. Wir betrachten die Funktionen $f, g\colon \mathbb{N}_{>0} \times \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ mit

$$f(n,k) := \Big(\sum_{i=1}^{n} i\Big)^k = (1 + \cdots + n)^k \qquad \text{und} \qquad g(n,k) := \sum_{i=1}^{n} i^{k+1} = 1^{k+1} + \cdots + n^{k+1}.$$

(a) Beweisen Sie mit Hilfe des Induktionsprinzips, dass $f(n,2) = g(n,2)$ für alle $n \in \mathbb{N}_{>0}$.

(b) Gilt $f(n,k) = g(n,k)$ für alle $n, k \in \mathbb{N}_{>0}$? Beweisen Sie die Aussage oder geben Sie ein Gegenbeispiel an.

Zur Erinnerung: Für alle $n \in \mathbb{N}_{>0}$ gilt: $\sum_{i=1}^{n} i = \frac{n\cdot(n+1)}{2}$.

Aufgabe 4.12. Gegeben sei folgende rekursiv definierte Funktion:

$$\text{Für allle } n \in \mathbb{N} \text{ sei} \qquad g_s(n) := \begin{cases} s & , n = 0 \\ \frac{1}{2} \cdot g_s(n-1) & , \text{falls } g_s(n-1) \text{ gerade und } n \geqslant 1 \\ 3 \cdot g_s(n-1) + 1 & , \text{falls } g_s(n-1) \text{ ungerade und } n \geqslant 1 \end{cases}$$

Hierbei bezeichnet $s \in \mathbb{N}_{>0}$ den Startwert der Funktion. Berechnen Sie $g_5(5)$ und $g_{23}(15)$.[3]

Aufgabe 4.13. Sei $G(n)$ ein Gitter bestehend aus einer Zeile und n Spalten. Formal definieren wir $G(n)$ als Menge von Kreuzungspunkten (x, y) in $\mathbb{N} \times \mathbb{N}$ und Linien, die diese Punkte verbinden, und zwar wie folgt: Sei $\{(x,y) : x, y \in \mathbb{N},\ 0 \leqslant x \leqslant n,\ 0 \leqslant y \leqslant 1\}$ die Menge von Kreuzungspunkten von $G(n)$. Zwischen je zwei Kreuzungspunkten k_1 und k_2 verläuft eine Linie genau dann, wenn sich k_1 und k_2 in genau einer Koordinate um genau den Betrag eins unterscheiden.

Sei $R(n)$ die Anzahl der verschiedenen Rechtecke mit nicht-leerem Flächeninhalt, die ins Gitter $G(n)$ so gezeichnet werden können, dass jedes Rechteck sich aus Linien von $G(n)$ zusammensetzt. Die folgende Abbildung zeigt alle möglichen Rechtecke, die in $G(3)$ gezeichnet werden können. Insbesondere ist $R(3) = 6$.

Beweisen Sie durch vollständige Induktion nach n, dass f. a. $n \in \mathbb{N}_{>0}$ gilt: $R(n) = n(n+1)/2$.

Aufgabe 4.14. Gegeben sei ein quadratisches $2^n \times 2^n$-Schachbrett mit $n \in \mathbb{N}_{>0}$, bei dem ein Eckfeld fehlt. Wir fragen uns, ob wir alle Felder mit *L-Kacheln* auslegen können. Dabei überdeckt jede L-Kachel genau drei Felder, hat die Form wie links in Abbildung 4.1 dargestellt und darf rotiert werden. In einer Kachelung dürfen sich keine L-Kacheln überlappen.

Zeigen durch Induktion Sie für alle $n \in \mathbb{N}_{>0}$: Eine Kachelung ist für jedes $2^n \times 2^n$-Schachbrett möglich, bei dem genau ein Feld entfernt wurde.

Aufgabe 4.15.

[3] Bei dieser Funktion handelt es sich um die sogenannte Collatz-Funktion für den Startwert $s \in \mathbb{N}_{>0}$. Es ist kein Startwert s bekannt, für den g_s nicht irgendwann den Wert 1 erreicht, d.h. es ist unbekannt, ob für jedes $s \in \mathbb{N}_{>0}$ ein $n_0 \in \mathbb{N}$ existiert, so dass $g_s(n_0) = 1$.

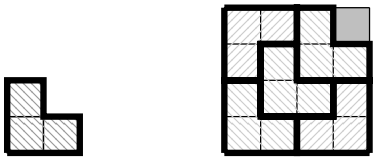


Abbildung 4.1.: Links: Eine L-Kachel. Rechts: Beispielkachelung eines $2^2 \times 2^2$-Schachbretts mit fehlender Ecke oben rechts (grau markiert).

(a) Sei $n \in \mathbb{N}_{>0}$. Wie viele verschiedene boolesche Funktionen $f : \{0,1\}^n \to \{0,1\}$ gibt es, d.h. wie groß ist die Menge $\mathrm{Abb}(A, B)$ für $A = \{0,1\}^n$ und $B = \{0,1\}$? Beweisen Sie Ihre Antwort.

(b) Zeigen Sie: Es gibt überabzählbar viele Funktionen $f : \{0,1\}^* \to \{0,1\}$, d.h. es existiert keine surjektive Funktion von $\mathbb{N}$ nach $\mathrm{Abb}\big(\{0,1\}^*, \{0,1\}\big)$.

Hinweis: Passen Sie das Diagonalargument aus dem Beweis für die Nichtabzählbarkeit von $\mathcal{P}(\mathbb{N})$ an. Für eine beliebige Funktion $F : \mathbb{N} \to \mathrm{Abb}\big(\{0,1\}^*, \{0,1\}\big)$ konstruiere eine Funktion $g : \{0,1\}^* \to \{0,1\}$ sodass $g \neq F(n)$ f.a. $n \in \mathbb{N}$.

(c) Vergleichen Sie die Größen der beiden Mengen

$$\bigcup_{n\in\mathbb{N}} \mathrm{Abb}\big(\{0,1\}^n, \{0,1\}\big) \quad \text{und} \quad \mathrm{Abb}\big(\{0,1\}^*, \{0,1\}\big)\,.$$

Welche ist „größer“? Oder sind beide gleichmächtig? (Eine Menge A ist „größer“ als eine Menge B, wenn es keine surjektive Funktion $s : B \to A$ gibt.)

Aufgabe 4.16. Lösen Sie die folgenden Rekursionsgleichungen auf, d. h. finden Sie jeweils einen (möglichst einfachen) geschlossenen Ausdruck für a_n und geben Sie eine kurze Herleitung an.

(a) $a_1 := 0$ und $a_{n+1} := n + a_n$ für alle $n \in \mathbb{N}_{>0}$.

(b) $a_1 := 1$ und $a_{n+1} := 3 - a_n$ für alle $n \in \mathbb{N}_{>0}$.

(c) $a_0 := 1$ und $a_{n+1} := \frac{a_n}{2} + 2^n$ für alle $n \in \mathbb{N}$.

(d) $a_0 := 0$, $a_1 := 2$ und $a_{n+1} := 2 \cdot a_{n-1}$ für alle $n \in \mathbb{N}_{>0}$

(e) $a_1 := 1$ und $a_{n+1} := \left(1 - \frac{1}{n+1}\right) \cdot a_n$ für alle $n \in \mathbb{N}_{>0}$.

(f) $a_1 := 2$ und $a_{n+1} := (a_n)^2$ für alle $n \in \mathbb{N}_{>0}$.

(g) $a_2 := 1$ und $a_{n+1} := a_n \cdot \frac{n+1}{n-1}$ für alle $n \in \mathbb{N}_{>0}$ mit $n \geqslant 2$

Beispiel. Es gelte $a_1 := 1$ und $a_{n+1} := a_n + 1$ für alle $n \in \mathbb{N}_{>0}$. Dann lautet die *Lösung* der Rekursionsgleichung:

$$a_n = n$$

Herleitung:

$$a_n = a_{n-1} + 1 = a_{n-2} + 1 + 1 = \cdots = a_{n-(n-1)} \underbrace{+1 + 1 + \cdots + 1}_{(n-1)\text{-mal}} = \underbrace{a_1}_{=1} + n - 1 = n$$

Aufgabe 4.17. Die beiden Trolle Höllga Knochenbeißer und Trolgar Keulenhau sind frisch gebackene Eltern eines mit allen drei Zahnreihen grinsenden 94 kg Säuglings namens Sulpur. Nach alter Troll-Tradition wird Sulpur von der ganzen Troll-Familie reich beschenkt.

(a) Onkel Raudishor Grubentod schenkt Sulpur eine kleine Modelleisenbahn mit kleinen und großen Wagen, welche 1 Meter bzw. 2 Meter lang sind. Leider kann Sulpur sein neues Geschenk überhaupt nicht ausprobieren, da zwischen den erwachsenen Trollen sofort eine heftige Diskussion über die Anzahl möglicher Züge der Länge $n \in \mathbb{N}_{>0}$ entbrannt ist[4]. Raudishor hat sich einen 1 Meter langen und Höllga einen 2 Meter langen Wagen geschnappt.

- Raudishor sagt: „Wenn ich nur wüsste, wie viele verschiedene Züge der Länge $n-1$ es gibt, dann wüsste ich auch, wie viele Züge der Länge n es gibt, die mit einem 1 Meter langen Wagen beginnen."
- Höllga sagt: „Und ich kann dieselbe Frage für die Züge beantworten, die mit einem 2 Meter langen Wagen beginnen."

Wir repräsentieren einen Zug mit genau k Wagen durch das Tupel $z = (z_1, \ldots, z_k) \in \{1,2\}^k$, wobei z_i die Länge des i-ten Wagens ist. Die Länge des Zugs ist durch $\text{länge}(z) = \sum_{i=1}^{k} z_i$ gegeben[5]. Ein Beispiel können Sie Abbildung 4.2 entnehmen.

Abbildung 4.2.: Ein Zug der Länge 5 mit k=4 Wagen und $(z_1, z_2, z_3, z_4) = (1, 2, 1, 1)$.

Für alle $n \in \mathbb{N}_{>0}$ bezeichne a_n die Anzahl möglicher Züge der Länge genau n, d. h. es gilt $a_n = |\{z \in \{1,2\}^* : \text{länge}(z) = n\}|$.

i) Skizzieren Sie alle möglichen Züge der Längen $1, 2, 3$ und 4 und bestimmen Sie a_1, a_2, a_3 und a_4.

ii) Formulieren Sie eine Rekursionsgleichung für a_{n+1}, indem Sie Raudishors und Höllgas Gedanken weiterentwickeln. Stellen Sie sicher, dass Ihre Rekursionsgleichung konsistent zu Ihren Ergebnissen aus Teil i) ist.

iii) Angenommen, jetzt stehen Wagen der Längen 1, 2 und 3 zur Verfügung. Wie lautet nun die Rekursionsgleichung für a_{n+1}?

(b) Sei $n \in \mathbb{N}_{>0}$ die Anzahl von Sulpurs Geschenken. Laut Troll-Tradition muss Sulpur sich von den n vorhandenen Geschenken k Stück aussuchen ($0 \leqslant k \leqslant n$). Wie groß ist die Qual der Wahl, d. h. wie viele k-elementige Teilmengen der Menge $\{1, 2, \ldots, n\}$ gibt es?

- Papa Trolgar rät Sulpur: „Wenn du die Modelleisenbahn behältst, dann musst du nur noch $k-1$ aus $n-1$ Geschenken auswählen." Sulpurs Frage „Und wenn ich die Bahn nicht behalten möchte?" bleibt unbeantwortet.

Sei $T(n,k) = |\{A : A \subseteq \{1, 2, \ldots, n\}, |A| = k\}|$ die Anzahl k-elementiger Teilmengen von $\{1, 2, \ldots, n\}$.

(i) Bestimmen Sie die Potenzmenge $\mathcal{P}(\{1,2,3\})$ sowie $T(3,0)$, $T(3,1)$, $T(3,2)$ und $T(3,3)$.

(ii) Formulieren Sie eine Rekursionsgleichung für $T(n,k)$, indem Sie Trolgars Idee weiterentwickeln. Bestimmen Sie zunächst die beiden *Rekursionsanker* $T(n,n)$ und $T(n,0)$ direkt.

[4] Das weitverbreitete Vorurteil, dass sich Trolle nicht für mathematische Fragestellungen interessieren, ist falsch!
[5] Die Lokomotive bleibt hier der Einfachheit halber unberücksichtigt.

Aufgabe 4.18. (a) Wir betrachten die Python-Funktion `hoch` : $\mathbb{N}_{>0} \times \mathbb{N} \to \mathbb{N}$:

```
def hoch(b, n):
    if n == 0:
        return 1
    elif n%2 == 0:  # n ist gerade
        return hoch(b*b, n/2)
    else:
        return b * hoch(b, n-1)
```

Zeigen Sie: Für alle $b \in \mathbb{N}_{>0}$ und $n \in \mathbb{N}$ liefert der Aufruf `hoch(b,n)` den Rückgabewert b^n.

Kommentar: Natürlich können wir die n-te Potenz von b auch naiv durch die n-fache Multiplikation von b mit sich selbst berechnen, d.h. durch $b^n = b \cdot b \cdot \ldots \cdot b$. Sehr viel weniger Multiplikationen sind aber ausreichend: Für $n = 2^k$ genügen zur Berechnung von b^n bereits k Quadrierungen, denn es gilt $b^n = b^{2^k} = b^{2 \cdot 2 \cdot \ldots \cdot 2} = (\cdots(b^2)^{2\cdots})^2$. Diesen Umstand macht sich die Funktion `hoch` zunutze.

(b) Wir betrachten die *Russische Bauernmultiplikation*[6] zum Berechnen eines Produktes $x \cdot k$, wobei $x \in \mathbb{R}$ und $k \in \mathbb{N}$.

```
def prod(x,k):
    if k == 0:
        return 0
    elif k % 2 == 0:                # k ist gerade und groesser 0
        return prod(2*x, k/2)
    else:                           # k ist ungerade
        return prod(2*x, k//2) + x  # k//2 entspricht der
                                    # ganzzahligen Division (k-1)/2
```

Hierbei wird ausgenutzt, dass eine *Verdopplung* von x (bzw. eine Halbierung von k) in binärer Darstellung relativ einfach ist: ein Bitshift genügt.

Zeigen Sie mit vollständiger Induktion nach k: Für alle $x \in \mathbb{R}$ und alle $k \in \mathbb{N}$ gilt: $x \cdot k =$ `prod(x,k)`

(c) Gegeben sei die folgende rekursive Funktion `f` : $\mathbb{Z} \to \mathbb{Z}$.

```
def f(z):
    if z > 11:
        return z-2
    else:
        return f(f(z+3))
```

Zeigen Sie mit vollständiger Induktion:

Für alle $z \in \mathbb{Z}$ mit $z \leqslant 12$ gibt die Funktion `f` den Wert 10 zurück.

Hinweis: Verwenden Sie als Induktionsschritt $z \to z - 3$. Was müssen Sie dann beim Induktionsanfang beachten?

Kommentar: Bei der obigen Funktion handelt es sich um eine Variante der *McCarthy-91-Funktion*, die als schwieriger Testfall bei der formalen Verifikation von Programmen eingesetzt wird.

Aufgabe 4.19. Sei die Sprache L über dem Alphabet $A := \{(,)\}$ wie folgt rekursiv definiert:

[6] Diese Methode war sogar bereits vor über 3500 Jahren im Alten Ägypten bekannt: https://en.wikipedia.org/wiki/Rhind_Mathematical_Papyrus

Basisregel: (B) $\varepsilon \in L$

Rekursive Regeln: (R1) Ist $w \in L$, so ist auch $(w) \in L$.

(R2) Sind $w_1, w_2 \in L$, so ist auch $w_1 w_2 \in L$.

(a) Für jedes Symbol $s \in A$ und jedes Wort $w \in A^*$ bezeichne $|w|_s$ die Anzahl der Vorkommen des Symbols s in w. Beweisen Sie durch Induktion, dass für alle Wörter $w \in L$ gilt: $|w|_(= |w|_)$.

(b) Beweisen Sie, dass $()(()() \notin L$ ist.

Aufgabe 4.20. Im Folgenden wird die Syntax einer sehr einfachen Programmiersprache definiert, der so genannten WHILE-Programme. Die Menge L, die hier definiert wird, ist die Menge aller Zeichenketten über dem Alphabet A, die syntaktisch korrekte WHILE-Programme sind. Hierbei ist $A := \{\mathbf{x}, \mathbf{:=}, +, -, \neq, \mathbf{;}, \mathbf{while}, \mathbf{do}, \mathbf{end}\} \cup \mathbb{N}$, und L ist die folgendermaßen rekursiv definierte Menge:

Basisregeln: (B1) Für Zahlen $i, j, c \in \mathbb{N}$ gilt: $\mathbf{x}i \mathbf{:=} \mathbf{x}j + c \in L$.

(B2) Für Zahlen $i, j, c \in \mathbb{N}$ gilt: $\mathbf{x}i \mathbf{:=} \mathbf{x}j - c \in L$.

Rekursive Regeln: (R1) Sind $w_1 \in L$ und $w_2 \in L$, so ist auch $w_1 \mathbf{;}\ w_2 \in L$.

(R2) Ist $w \in L$ und $i \in \mathbb{N}$, so ist $\mathbf{while}\ \mathbf{x}i \neq 0\ \mathbf{do}\ w\ \mathbf{end} \in L$.

(a) Welche der folgenden Wörter aus A^* gehören zu L und welche nicht? Begründen Sie jeweils Ihre Antwort.

(i) $\mathbf{x}3 \mathbf{:=} \mathbf{x}7 - 2$

(ii) $\mathbf{x}3 \mathbf{:=} 1\mathbf{;}\ \mathbf{x}2 \mathbf{:=} \mathbf{x}3 + 5$

(iii) $\mathbf{while}\ \mathbf{x}1 \neq 0\ \mathbf{do}\ \mathbf{x}0 \mathbf{:=} \mathbf{x}0 + 1\mathbf{;}\ \mathbf{x}1 \mathbf{:=} \mathbf{x}1 - 1\ \mathbf{end}$

(iv) $\mathbf{x}1 \mathbf{:=} \mathbf{x}1 + 42\mathbf{;}\ \mathbf{while}\ \mathbf{x}1 \neq 0\ \mathbf{do}\ \mathbf{x}1 \mathbf{:=} \mathbf{x}1 - 1$

(b) Für jedes Wort $w \in A^*$ und jedes Symbol $s \in A$ bezeichne $|w|_s$ die Anzahl der Vorkommen des Symbols s in w. Beweisen Sie durch Induktion, dass für alle Wörter $w \in L$ gilt: $|w|_{\mathbf{do}} = |w|_{\mathbf{end}}$.

Aufgabe 4.21. Sei $A = \{a, b, c, \langle, \rangle, [,], \circ\}$. Die Sprache $\mathrm{TL} \subseteq A^*$ sei wie folgt rekursiv definiert:

Basisregel: (B) Es gilt: $a, b, c \in \mathrm{TL}$.

Rekursive Regeln: (R1) Ist $w \in \mathrm{TL}$, so ist auch $[w] \in \mathrm{TL}$.

(R2) Sind $w_1 \in \mathrm{TL}$ und $w_2 \in \mathrm{TL}$, so ist auch $\langle w_1 \circ w_2 \rangle \in \mathrm{TL}$.

(a) Welche der folgenden Aussagen gelten, welche nicht?

a) $\langle [\langle c \circ b \rangle] \circ a \rangle \in \mathrm{TL}$

b) $\langle [a \circ b] \rangle \in \mathrm{TL}$

c) $\varepsilon \in \mathrm{TL}$

d) $\varepsilon \in \mathrm{TL}^*$

(b) Die Funktion $f : \mathrm{TL} \to \mathbb{N}$ sei wie folgt definiert:

a) $f(a) := 3$, $f(b) := 36$, $f(c) := 303$.

b) Für jedes $w \in \mathrm{TL}$ sei $f([w]) := f(w) + 6$.

c) Für alle $w_1, w_2 \in \mathrm{TL}$ sei $f(\langle w_1 \circ w_2 \rangle) := f(w_1) + f(w_2)$.

Berechnen Sie $f([[[a]]])$ und $f([\langle b \circ [c] \rangle])$.
Beweisen Sie außerdem durch Induktion über den Aufbau von TL, dass Folgendes gilt:

Für alle $w \in \mathrm{TL}$ ist $f(w)$ durch 3 teilbar.

Aufgabe 4.22.
In Beispiel 4.28 ist der „Induktionsschritt $n \to n+1$“ für den Wert $n = 1$ nicht schlüssig, denn in diesem Fall gilt $n + 1 = 2$ und

- $M = \{a_1, a_2\}$,
- $M' = \{a_1\}$,
- $M'' = \{a_2\}$.

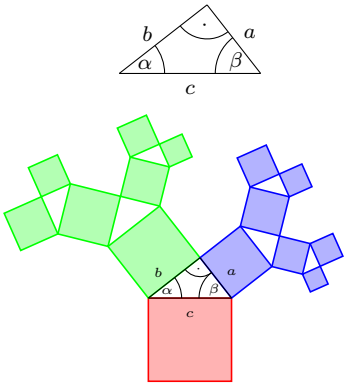
Insbesondere gilt also zwar, dass $a_2 \in M''$, aber es gilt nicht, dass $a_2 \in M'$.

(a) Wo steckt der Fehler im Argument aus Beispiel 4.29, dass je zwei natürliche Zahlen gleich sind?

(b) Auch die Behauptung aus Beispiel 4.30, dass „$n = n + 1$ für alle natürlichen Zahlen gilt“ ist natürlich falsch. Aber warum ist der Beweis falsch?

Aufgabe 4.23. Gegeben sei ein Winkel α zwischen 1° und 89°. Für ein $n \in \mathbb{N}$ wird der n-te *Pythagoras-Baum* P_n folgendermaßen konstruiert:

- P_0 ist ein Quadrat mit Kantenlänge 1.
- Um P_{n+1} zu erhalten, gehe wie folgt vor (vgl. die Skizze rechts).
  - Zeichne ein Dreieck mit den Winkeln 90°, α und $\beta := 90° - \alpha$ sowie den Kantenlängen $c = 1$, $b = \cos(\alpha)$ und $a = \sin(\alpha)$.
  - Setze an die Kante c ein Quadrat mit Seitenlänge 1.
  - Setze an die Kante b einen Pythagoras-Baum P_n, dessen Kantenlängen um den Faktor $\cos(\alpha)$ verkleinert sind.
  - Setze an die Kante a einen Pythagoras-Baum P_n, dessen Kantenlängen um den Faktor $\sin(\alpha)$ verkleinert sind.



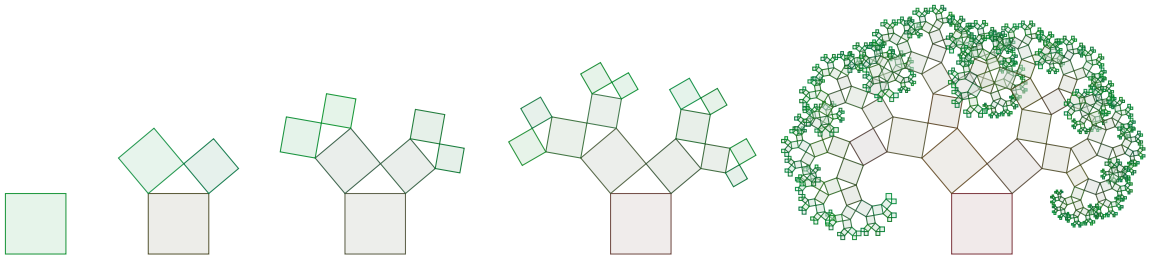


Abbildung 4.3.: von links nach rechts: P_0, P_1, P_2, P_3 und P_9, jeweils mit $\alpha = 40°$.

(a) Sei E_n die Anzahl der Ecken des n-ten Pythagoras-Baums. (Z.B. gilt E_0=4, E_1=9, E_2=19.)

(i) Geben Sie eine Rekursionsgleichung für E_n an.

(ii) Lösen Sie die Rekursionsgleichung aus Teil i), d. h. geben Sie einen geschlossenen (nicht-rekursiven) Ausdruck für E_n an.

Hinweis: Überprüfen Sie die Korrektheit Ihrer Lösung für kleine n. Die geometrische Reihe könnte sich als hilfreich erweisen.

(b) Sei A_n der Flächeninhalt des n-ten Pythagoras-Baumes, wobei wir nur die Flächen in den Quadraten, nicht aber in den Dreiecken zählen.

(i) Geben Sie eine Rekursionsgleichung für A_n an.

(ii) Lösen Sie die Rekursionsgleichung aus Teil i).

Hinweis: Für beliebige Winkel α gilt der Satz des Pythagoras: $(\cos(\alpha))^2 + (\sin(\alpha))^2 = 1$.

Für große n kann es dazu kommen, dass sich Ecken oder Flächen der Bäume überschneiden. Dieser Umstand soll in allen Teilaufgaben ignoriert werden.

Aufgabe 4.24.

a) Für die Folge $T_0, T_1, \ldots$ der Sierpinski-Dreiecke sind in Abbildung 4.4 die ersten vier Folgenglieder dargestellt. Beginnend bei einem gleichseitigen Dreieck T_0 mit dem Flächeninhalt A, entsteht T_1 aus T_0, indem T_0 wie abgebildet in vier gleichseitige Dreiecke mit identischem Flächeninhalt zerlegt und das mittlere Dreieck „gelöscht“ wird (dargestellt durch eine weiße Fläche). Für die drei übrigen Dreiecke (oben, links unten und rechts unten) wird dieser Prozess rekursiv wiederholt.

(a) T_0 (b) T_1 (c) T_2 (d) T_3

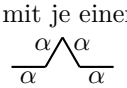
Abbildung 4.4.: Die Sierpinski-Dreiecke T_0, T_1, T_2 und T_3. Die weißen Dreiecke wurden gelöscht.

Sei A_n die Fläche des n-ten Sierpinski-Dreiecks T_n – die weißen Dreiecke gehen also nicht in die Fläche ein.

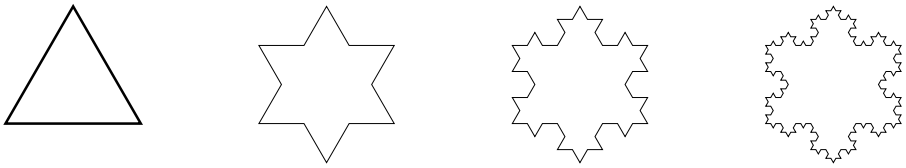
i) Stellen Sie eine Rekursionsgleichung für A_n auf.

ii) Zeigen Sie mit vollständiger Induktion, dass $A_n = \left(\frac{3}{4}\right)^n \cdot A$ gilt.

iii) Was passiert im Grenzwert für $n \to \infty$? Bestimmen Sie $\lim_{n\to\infty} A_n$.

b) Die Folge $S_0, S_1, \ldots$ der Kochschen Schneeflocken ist in Abbildung 4.5 dargestellt. Beginnend mit einem gleichseitigen Dreieck S_0 mit der Seitenlänge a, entsteht S_{n+1} aus S_n, indem jede Seite von S_n durch vier Seiten mit je einem Drittel ihrer Länge ersetzt wird, mit folgender Form: wird zu 

Bestimmen Sie für jedes $n \in \mathbb{N}$ den Umfang U_n der Kochschen Schneeflocke S_n. Eine kurze Begründung ist ausreichend.



(a) S_0 (b) S_1 (c) S_2 (d) S_3

Abbildung 4.5.: Die Kochschen Schneeflocken S_0, S_1, S_2 und S_3

Kommentar: Für $n \to \infty$ erhalten wir auf diese Weise eine Schneeflocke mit endlicher Fläche (sie passt vollständig in Quadrat mit der Seitenlänge $\frac{2a\sqrt{5}}{3}$), aber unbeschränktem Umfang.

c) Wir definieren die Folge $C_0, C_1, \ldots$ von Mengen: Wir beginnen mit dem reellen Intervall $C_0 = [0, 1]$ und erhalten $C_1 = [0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$, indem wir das mittlere Drittel entfernen. Auf den beiden anderen Intervallen setzen wir diesen Prozess rekursiv fort und entfernen in jedem Schritt jeweils das mittlere Drittel (siehe Abbildung 4.6). Den Schnitt aller Folgenglieder bezeichnen wir als die Cantor-Menge.

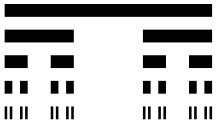


Abbildung 4.6.: $C_0, C_1, \ldots, C_4$

i) Bestimmen Sie die Länge der maximalen Intervalle in C_n. Wie viele solcher Intervalle gibt es in C_n?

ii) Zeigen Sie, dass die Cantor-Menge $C := \bigcap_{n \in \mathbb{N}} C_n$ überabzählbar ist.

Teil II.

Werkzeugkasten: Graphen

5. Graphen und Bäume

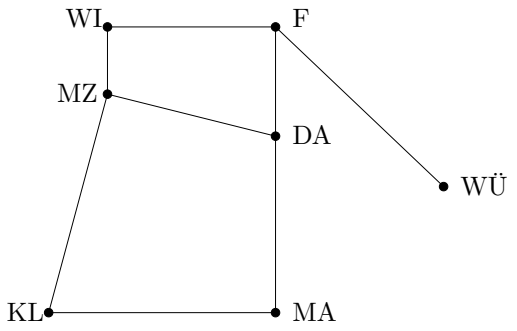
Bei Modellierungsaufgaben geht es oft darum, **Objekte** sowie **Beziehungen** zwischen Objekten zu beschreiben. Graphen und Bäume eignen sich dazu oft besonders gut.
Anschaulich besteht ein Graph aus **Knoten** und **Kanten:**

- „Knoten“ repräsentieren „Objekte“.
- „Kanten“ repräsentieren Beziehungen zwischen je zwei „Objekten“.

Je nach Aufgabenstellung werden **ungerichtete** Graphen oder **gerichtete** Graphen verwendet. Bäume sind Graphen mit bestimmten Eigenschaften.

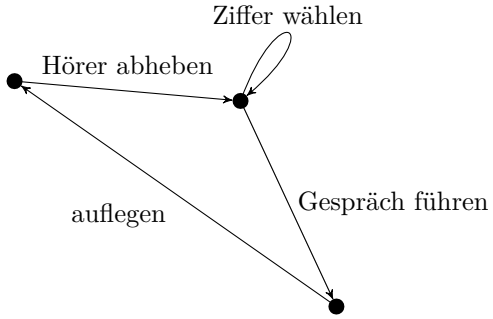
Beispiel 5.1.

(a) Skizze eines **ungerichteten** Graphen, der die Autobahnverbindungen zwischen einigen Städten darstellt:



DA	$\widehat{=}$	Darmstadt
F	$\widehat{=}$	Frankfurt
KL	$\widehat{=}$	Kaiserslautern
MA	$\widehat{=}$	Mannheim
MZ	$\widehat{=}$	Mainz
WI	$\widehat{=}$	Wiesbaden
WÜ	$\widehat{=}$	Würzburg

(b) Skizze eines **gerichteten** Graphen, der den prinzipiellen Ablauf eines Telefonats darstellt:



5.1. Graphen

5.1.1. Grundlegende Definitionen

Definition 5.2 (ungerichteter Graph).

ungerichteter Graph

Ein **ungerichteter Graph** $G = (V, E)$ besteht aus einer endlichen Menge V, die **Knotenmenge** von G genannt wird, und einer Menge

$$E \subseteq \big\{\{i, j\} : i \in V, j \in V, i \neq j\big\},$$

Knoten
Kanten

die **Kantenmenge** von G genannt wird. Die Elemente aus V heißen **Knoten** von G (auch: „Ecken“; englisch: **vertices**, singular: vertex); die Elemente aus E heißen **Kanten** von G (englisch: **edges**, singular: edge).

Beispiel 5.3. $G = (V, E)$ mit

$$\begin{aligned} V &:= \big\{\mathrm{MZ}, \mathrm{WI}, \mathrm{MA}, \mathrm{DA}, \mathrm{KL}, \mathrm{F}, \mathrm{W\ddot{U}}\big\} \text{ und} \\ E &:= \big\{\{\mathrm{MZ}, \mathrm{WI}\}, \{\mathrm{WI}, \mathrm{F}\}, \{\mathrm{F}, \mathrm{DA}\}, \{\mathrm{F}, \mathrm{W\ddot{U}}\}, \{\mathrm{MZ}, \mathrm{DA}\}, \{\mathrm{MZ}, \mathrm{KL}\}, \{\mathrm{KL}, \mathrm{MA}\}, \{\mathrm{DA}, \mathrm{MA}\}\big\} \end{aligned}$$

ist ein ungerichteter Graph, der die Autobahnverbindungen zwischen Mainz (MZ), Wiesbaden (WI), Mannheim (MA), Darmstadt (DA), Kaiserslautern (KL), Frankfurt (F) und Würzburg (WÜ) repräsentiert.

Beispiel 5.1((a)) zeigt diesen Graphen G in **graphischer Darstellung**: Knoten werden als Punkte dargestellt, Kanten als Verbindungslinien zwischen Punkten.

Beachte: Laut Definition 5.2 gibt es zwischen zwei Knoten i und j aus V

- **höchstens** eine Kante; diese wird mit $\{i, j\}$ bezeichnet und graphisch dargestellt als

- **keine** Kante, falls $i = j$ ist. In der graphischen Darstellung eines ungerichteten Graphen sind also „Schleifen“ der Form

nicht erlaubt.

Jede Kante $\{i, j\}$ eines ungerichteten Graphen ist also eine 2-elementige Menge von Knoten des Graphen.

Bemerkung: In der Literatur wird zumeist die oben genannte Definition von ungerichteten Graphen verwendet. Davon abweichend erlauben einige Bücher in ungerichteten Graphen aber auch „Schleifen“ der Form

Notation 5.4. Sei $G = (V, E)$ ein ungerichteter Graph.

inzident

- Ein Knoten $v \in V$ heißt **inzident** mit einer Kante $e \in E$, falls $v \in e$.

- Die beiden mit einer Kante $e \in E$ inzidenten Knoten nennen wir die **Endknoten** von e, und wir sagen, dass e diese beiden Knoten **verbindet**. Endknoten
- Zwei Knoten $v, v' \in V$ heißen **benachbart** (bzw. **adjazent**), falls es eine Kante $e \in E$ gibt, deren Endknoten v und v' sind (d.h. $e = \{v, v'\}$). benachbart adjazent
- Falls v und v' zwei benachbarte Knoten sind, so sagen wir auch: v' ist ein **Nachbar** von Knoten v. Nachbar

Definition 5.5 (Grad).
Sei $G = (V, E)$ ein ungerichteter Graph und sei $v \in V$ ein Knoten von G. Der **Grad von v in G** (engl.: degree), kurz: $\text{Grad}_G(v)$, ist die Anzahl der Kanten, die v als Endknoten haben. D.h. Grad $\text{Grad}_G(v)$

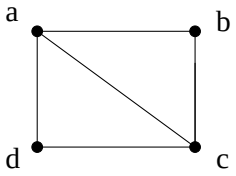
$$\text{Grad}_G(v) \;=\; |\{e \in E : v \in e\}|.$$

Der **Grad von G** ist

$$\text{Grad}(G) \;:=\; \max\{\text{Grad}_G(v) : v \in V\}, \qquad \text{Grad}(G)$$

d.h. $\text{Grad}(G)$ gibt den maximalen Grad eines Knotens von G an.[1]

Beispiel:

Für den Graphen $G =$ 

gilt: $\text{Grad}_G(a) = 3$, $\text{Grad}_G(b) = 2$, $\text{Grad}_G(c) = 3$, $\text{Grad}_G(d) = 2$ und $\text{Grad}(G) = 3$.

Definition 5.6 (gerichteter Graph).
Ein **gerichteter Graph** $G = (V, E)$ besteht aus einer endlichen Menge V, die **Knotenmenge** von G genannt wird, und einer Menge gerichteter Graph

$$E \;\subseteq\; V \times V = \big\{(i, j) \;:\; i \in V, j \in V\big\},$$

die **Kantenmenge** von G genannt wird. Die Elemente aus V heißen **Knoten** (bzw. „Ecken“), die Elemente aus E heißen (gerichtete) **Kanten** von G. Knoten (gerichtete) Kante

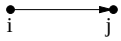
Beispiel 5.7. $G = (V, E)$ mit

$$\begin{aligned} V &:= \{a, b, c\} \text{ und} \\ E &:= \big\{(a, b), (b, b), (b, c), (c, a), (a, c)\big\} \end{aligned}$$

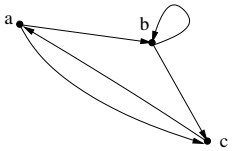
ist ein gerichteter Graph.

In der **graphischen Darstellung** eines gerichteten Graphen werden Knoten werden als Punkte dargestellt. Eine Kante der Form (i, j) wird als Pfeil von Knoten i nach Knoten j dargestellt, also

[1] Ist M eine endliche, nicht-leere Menge von Zahlen, so bezeichnet $\max M$ das größte Element von M.



Der gerichtete Graph aus Beispiel 5.7 lässt sich graphisch also wie folgt darstellen:



Notation 5.8. Sei $G = (V, E)$ ein gerichteter Graph.

Ausgangsknoten
Endknoten

- Ist $e = (i, j) \in E$, so heißt i der **Ausgangsknoten** von e und j der **Endknoten** von e, und wir sagen, dass e **von** i **nach** j **verläuft**.

inzident

- Ein Knoten $v \in V$ heißt **inzident** mit einer Kante $e \in E$, falls v der Ausgangs- oder der Endknoten von e ist.

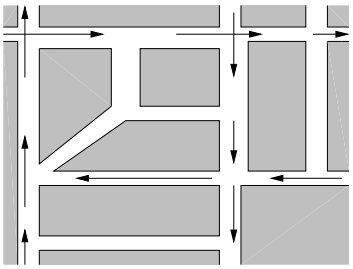
benachbart
adjazent

- Zwei Knoten $v, v' \in V$ heißen **benachbart** (bzw. **adjazent**), falls $(v, v') \in E$ oder $(v', v) \in E$.

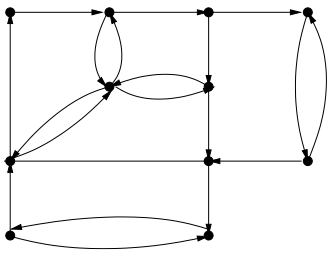
Schleife

- Eine Kante der Form (v, v) wird **Schleife** genannt. Eine Schleife ist also eine Kante, deren Ausgangs- und Endknoten identisch sind.

Beispiel 5.9 (Modellierung durch gerichtete Graphen).
In der folgenden Straßenkarte sind Einbahnstraßen durch Pfeile markiert.



Diese Straßenkarte können wir durch einen gerichteten Graphen repräsentieren, der für jede Straßenkreuzung einen Knoten enthält, und in dem es eine Kante von „Kreuzung“ i zu „Kreuzung“ j gibt, falls man von i nach j fahren kann, ohne zwischendurch eine weitere Kreuzung zu passieren. Graphisch lässt sich dieser gerichtete Graph folgendermaßen darstellen:



Welche Beispiele zur Modellierung durch Graphen haben wir bereits gesehen und welche weiteren wichtigen Beispiele gibt es noch?

- Wir haben die in einer Straßenkarte relevante Information durch einen gerichteten Graphen (mit Zusatzinformation) repräsentiert. In ähnlicher Weise lassen sich
  - das Schienennetz der deutschen Bahn oder
  - städtische S- und U-Bahn Netze

  modellieren.

  Graphen und Algorithmen für das kürzeste-Wege Problem spielen eine zentrale Rolle in Routenplanern.

- In einem Computer-Netzwerk werden Computer durch Knoten und Netzwerkverbindungen durch ungerichtete Kanten repräsentiert.

  In einem Strom-Netzwerk entsprechen die Knoten Erzeugern, Transformatoren oder Verbrauchern. Kanten entsprechen Direktverbindungen.

- Im „Webgraphen“ repräsentieren die Knoten Webseiten und die gerichteten Kanten Hyperlinks.

- Binäre Entscheidungsgraphen (BDDs) werden in der technischen Informatik als Datenstruktur für die kompakte Darstellung und effiziente Handhabung boolescher Funktionen eingesetzt. (Siehe z.B. die Vorlesung „Hardwarearchitekturen und Rechensysteme“.)

Definition 5.10.
Sei $G = (V, E)$ ein gerichteter Graph und sei $v \in V$ ein Knoten von G.

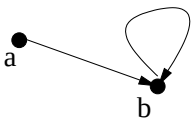
- Der **Ausgangsgrad** von v in G (engl.: out-degree), kurz: $\text{Aus-Grad}_G(v)$, ist die Anzahl der Kanten, die v als Ausgangsknoten haben. D.h.: Ausgangsgrad $\text{Aus-Grad}_G(v)$

$$\text{Aus-Grad}_G(v) \;=\; |\{e \in E : \text{ex. } v' \in V \text{ s.d. } e = (v, v')\}|.$$

- Der **Eingangsgrad** von v in G (engl.: in-degree), kurz: $\text{Ein-Grad}_G(v)$, ist die Anzahl der Kanten, die v als Eingangsknoten haben. D.h.: Eingangsgrad $\text{Ein-Grad}_G(v)$

$$\text{Ein-Grad}_G(v) \;=\; |\{e \in E : \text{ex. } v' \in V \text{ s.d. } e = (v', v)\}|.$$

Beispiel:

Für den Graphen $G =$ 

gilt:

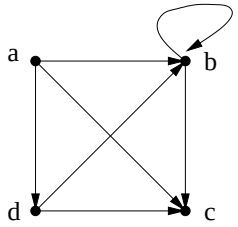
$$\begin{aligned} \text{Ein-Grad}_G(a) &= 0 \\ \text{Ein-Grad}_G(b) &= 2 \\ \text{Aus-Grad}_G(a) &= 1 \\ \text{Aus-Grad}_G(b) &= 1 \end{aligned}$$

Bemerkung 5.11 (Verschiedene Arten der Darstellung von Graphen).
Es gibt mehrere Arten Graphen darzustellen, zum Beispiel

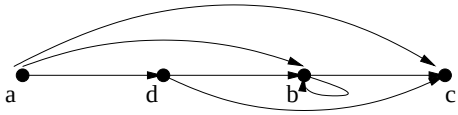
- **abstrakt**, durch Angabe der Knotenmenge V und der Kantenmenge E.
  Beispiel: $G_1 = (V_1, E_1)$ mit

  $$V_1 := \{a, b, c, d\} \quad \text{und} \quad E_1 := \{(a,b), (a,c), (a,d), (b,b), (b,c), (d,b), (d,c)\}.$$

- **graphisch** (bzw. **anschaulich**): Der obige Beispiel-Graph G_1 kann graphisch dargestellt werden durch

  oder, äquivalent dazu, durch

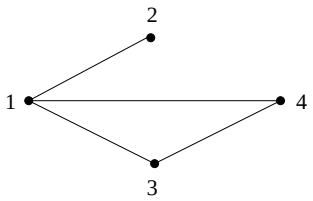
  

Adjazenzliste

- durch eine **Adjazenzliste**, die zu jedem Knoten i eine Liste aller Knoten angibt, zu denen eine von i ausgehende Kante führt. Der Beispiel-Graph G_1 wird durch folgende Adjazenzliste repräsentiert:

Knoten	Nachfolger
a	b, c, d
b	b, c
c	
d	b, c

  Auf die gleiche Art können auch **ungerichtete** Graphen durch eine Adjazenzliste repräsentiert werden. Beispielsweise der Graph

  $G_2 :=$ 

  durch die Adjazenzliste

Knoten	Nachbarn
1	2, 3, 4
2	1
3	1, 4
4	1, 3

Adjazenzmatrix

- durch eine **Adjazenzmatrix**, d.h. eine Tabelle, deren Zeilen und Spalten mit Knoten beschriftet sind, und die in der mit Knoten i beschrifteten Zeile und der mit Knoten j beschrifteten Spalte

– den Eintrag 1 hat, falls es eine Kante von Knoten i nach Knoten j gibt, und

– den Eintrag 0 hat, falls es keine Kante von i nach j gibt.

Beispielsweise sieht die Adjazenzmatrix des gerichteten Graphen G_1 wie folgt aus:

	a	b	c	d
a	0	1	1	1
b	0	1	1	0
c	0	0	0	0
d	0	1	1	0

Die Adjazenzmatrix des ungerichteten Graphen G_2 ist:

	1	2	3	4
1	0	1	1	1
2	1	0	0	0
3	1	0	0	1
4	1	0	1	0

In der Veranstaltung „Datenstrukturen“ werden die Vor- und Nachteile der Adjazenzlisten- und Adjazenzmatrixdarstellung besprochen.

Markierte Graphen

Bemerkung 5.12.
Viele Modellierungsaufgaben erfordern, dass den Knoten oder den Kanten eines Graphen weitere Informationen zugeordnet werden. Dies wird durch so genannte **Markierungsfunktionen** für Knoten oder Kanten formalisiert:

(a) Eine **Knotenmarkierung** eines (gerichteten oder ungerichteten) Graphen $G = (V, E)$ ist eine Abbildung Knoten-markierung

$$m\colon V \to W,$$

wobei W ein geeigneter Wertebereich ist. In dem Graph aus Beispiel 5.1((a)) könnte man beispielsweise eine Knotenmarkierung Einwohnerzahl$\colon V \to \mathbb{N}$ einführen, die jedem Knoten die Einwohnerzahl der zugehörigen Stadt zuordnet.

(b) Eine **Kantenmarkierung** von G ist eine Abbildung Kanten-markierung

$$m\colon E \to W,$$

wobei W ein geeigneter Wertebereich ist. Routenplaner (siehe Beispiel 5.1.2.2) benutzen vielfältige Kantenmarkierungen, aus denen die Länge des entsprechenden Straßenstücks hervorgeht wie auch die durchschnittliche Dauer für das Durchfahren der durch die Kante repräsentierten Strecke.

Kantenmarkierungen kann man auch dazu verwenden, um auszudrücken, dass es zwischen zwei Knoten mehr als eine Kante gibt. Die Markierungsfunktion gibt dann an, für wie viele Verbindungen die eine Kante des Graphen steht:

Definition 5.13 (Multigraph).

Multigraph

Ein **Multigraph** (G, m) besteht aus einem (gerichteten oder ungerichteten) Graphen $G = (V, E)$ und einer Kantenmarkierung $m\colon E \to \mathbb{N}$.

Beispiel: Sei $G = (V, E)$ der Graph mit $V = \{a, b, c\}$ und $E = \{\{a, b\}, \{b, c\}, \{c, a\}\}$. Sei $m : E \to \mathbb{N}$ mit $m(\{a, b\}) = 1$, $m(\{b, c\}) = 1$ und $m(\{c, a\}) = 2$. Dann ist (G, m) ein Multigraph, der graphisch wie folgt dargestellt werden kann:

5.1.1.1. Wege in Graphen

Definition 5.14 (Wege und Kreise).
Sei $G = (V, E)$ ein (gerichteter oder ungerichteter) Graph.

Weg

(a) Ein **Weg** in G ist ein Tupel

$$(v_0, \ldots, v_\ell) \in V^{\ell+1},$$

für ein $\ell \in \mathbb{N}$, so dass für alle $i \in \mathbb{N}$ mit $0 \leqslant i < \ell$ gilt:

- falls G ein gerichteter Graph ist, so ist $(v_i, v_{i+1}) \in E$,
- falls G ein ungerichteter Graph ist, so ist $\{v_i, v_{i+1}\} \in E$.

Weglänge

Das Tupel $(v_0, \ldots, v_\ell)$ wird dann **ein Weg von** v_0 **nach** v_ℓ genannt. ℓ ist die **Länge des Weges**. D.h.: Die **Länge** des Weges gibt gerade an, wie viele **Kanten** auf dem Weg durchlaufen werden.

Beachte: Gemäß dieser Definition ist für jedes $v \in V$ das Tupel (v) ein Weg der Länge 0 von v nach v. In der Literatur werden Wege manchmal auch als Tupel „aufeinanderfolgender" Kanten definiert.

einfacher Weg

(b) Ein Weg $(v_0, \ldots, v_\ell)$ heißt **einfach**, wenn kein Knoten mehr als einmal in dem Weg vorkommt (d.h. die Knoten $v_0, \ldots, v_\ell$ sind paarweise verschieden, lso $|\{v_0, \ldots, v_\ell\}| = \ell + 1$).

Kreis

(c) Ein Weg $(v_0, \ldots, v_\ell)$ heißt **Kreis**, wenn $\ell \geqslant 1$ und $v_\ell = v_0$ ist.

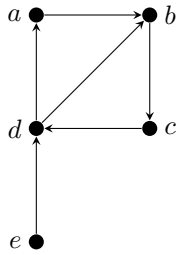
einfacher Kreis

(d) Ein Kreis $(v_0, \ldots, v_\ell)$ heißt **einfach**, wenn keine Kante mehrfach durchlaufen wird und – abgesehen vom Start- und Endknoten – kein Knoten mehrfach besucht wird. D.h.:

- In einem **gerichteten** Graphen G sind **einfache** Kreise genau die Wege der Form $(v_0, \ldots, v_\ell)$, für die gilt: $\ell \geqslant 1$ und $v_\ell = v_0$ und $|\{v_0, \ldots, v_{\ell-1}\}| = \ell$.
- In einem **ungerichteten** Graphen G sind **einfache** Kreise genau die Wege der Form $(v_0, \ldots, v_\ell)$, für die gilt: $\ell \geqslant 3$ und $v_\ell = v_0$ und $|\{v_0, \ldots, v_{\ell-1}\}| = \ell$.

Beispiel 5.15.

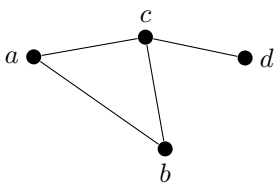
(a) Für den Graphen



gilt:

- (e, d, b, c, d) ist ein Weg der Länge 4, aber kein einfacher Weg.
- (d, b, c, d) ist ein einfacher Kreis.
- (e, d, a, b) ist ein einfacher Weg.
- (b, d, a) ist kein Weg.
- (a, b, c, d, b, c, d, a) ist ein Kreis, aber kein einfacher Kreis.

(b) Für den Graphen



gilt:

- (a, b, c, a) ist ein einfacher Kreis.
- (c, d, c) ist ein Kreis, aber kein einfacher Kreis.
- (a, c, d) ist ein einfacher Weg.
- (c, b, a, c, d) ist ein Weg, aber kein einfacher Weg.

Definition 5.16. (**Zusammenhängende und stark zusammenhängende Graphen**).
Sei $G = (V, E)$ ein Graph und $v \in V$ ein Knoten von G.

(a) Wenn G ungerichtet ist, dann besteht die **Zusammenhangskomponente** von v aus allen, von v aus durch einen Weg erreichbaren Knoten. — Zusammenhangskomponente

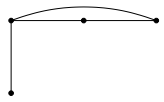
Der Graph G heißt **zusammenhängend**, wenn die Zusammenhangskomponente irgendeines Knotens aus allen Knoten von G besteht. — zusammenhängend

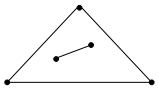
(b) Wenn G gerichtet ist, dann besteht die **starke Zusammenhangskomponente** von v aus allen Knoten w, — starke Zusammenhangskomponente

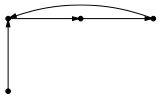
- die sowohl von v aus erreichbar sind,
- die aber auch v selbst durch einen in w beginnenden Weg erreichen.

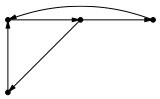
G heißt **stark zusammenhängend**, wenn die starke Zusammenhangskomponente irgendeines Knotens aus allen Knoten von G besteht. — stark zusammenhängend

Beispiel 5.17.

Der Graph  ist zusammenhängend,

 ist nicht zusammenhängend und besteht aus zwei Zusammenhangskomponenten.

Der Graph  ist nicht stark zusammenhängend, da es z.B. keinen Weg vom Knoten links oben zum Knoten links unten gibt.

 ist stark zusammenhängend.

5.1.1.2. Ähnlichkeit von Graphen

Die folgende Definition formalisiert, wann ein Graph G' in einem Graphen G „enthalten“ ist.

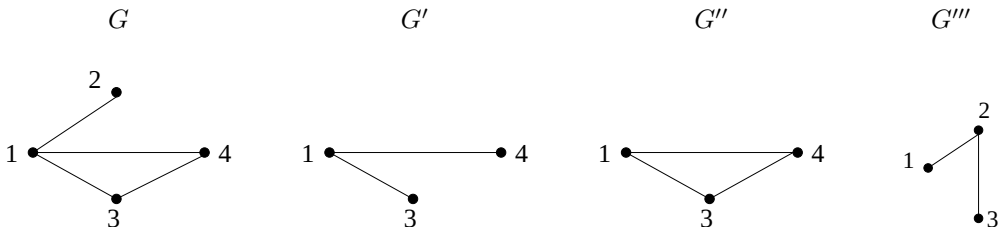
Definition 5.18 (Teilgraph).
Seien $G = (V, E)$ und $G' = (V', E')$ zwei (gerichtete oder ungerichtete) Graphen.

Teilgraph (a) G' heißt **Teilgraph von** G, falls $V' \subseteq V$ und $E' \subseteq E$.

$G_{|W}$ (b) Sei $W \subseteq V$. Der von W induzierte Teilgraph von G ist der Graph $G_{|W}$ mit Knotenmenge W und Kantenmenge $E_{|W} := \{e \in E : \text{alle mit } e \text{ inzidenten Knoten liegen in } W\}$.

induzierter Teilgraph (c) $G' = (V', E')$ heißt **induzierter Teilgraph von** G, falls $V' \subseteq V$ und $G' = G_{|V'}$.

Beispiel 5.19. Wir betrachten die folgenden Graphen:



Dann ist

- G' ein Teilgraph von G, aber kein induzierter Teilgraph von G.
- G'' ein induzierter Teilgraph von G.
- G''' kein Teilgraph von G.

Definition 5.20 (Gleichheit von Graphen).
Zwei Graphen $G = (V, E)$ und $G' = (V', E')$ sind **gleich** (kurz: $G = G'$), falls sie dieselbe Knotenmenge und dieselbe Kantenmenge besitzen. D.h.: $G = G'$

$$G = G' \ :\Longleftrightarrow\ V = V' \text{ und } E = E'.$$

Beispielsweise sind die beiden Graphen

nicht gleich, da sie unterschiedliche Knotenmengen besitzen. Intuitiv sind die beiden Graphen aber *„prinzipiell gleich“* (Fachbegriff: **isomorph**, kurz: $G \cong G'$), da der zweite Graph aus dem ersten durch Umbenennung der Knoten entsteht. Der Begriff der Isomorphie wird durch die folgende Definition präzisiert:

Definition 5.21. (Isomorphie von Graphen).
Seien $G = (V, E)$ und $G' = (V', E')$ zwei (gerichtete oder ungerichtete) Graphen. G und G' heißen **isomorph** (kurz: $G \cong G'$, in Worten: G ist isomorph zu G'), falls es eine bijektive Abbildung isomorph $G \cong G'$

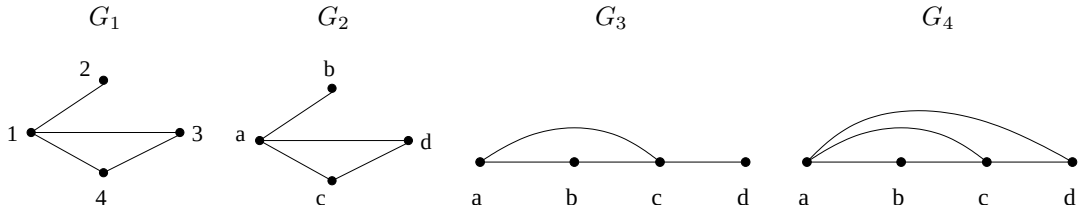
$$f : V \to V'$$

gibt, so dass für alle Knoten $i \in V$ und $j \in V$ gilt:

$$\begin{aligned} (i,j) \in E \quad &\Longleftrightarrow \quad \big(f(i), f(j)\big) \in E' \ \text{(für gerichtete Graphen)} \\ \{i,j\} \in E \quad &\Longleftrightarrow \quad \{f(i), f(j)\} \in E' \ \text{(für ungerichtete Graphen.)} \end{aligned}$$

Eine solche Abbildung f wird **Isomorphismus von G nach G'** genannt. Isomorphismus

Beispiel 5.22. Es seien:



Dann gilt:

- $G_1 \cong G_2$ via $f\colon \{1,2,3,4\} \to \{a,b,c,d\}$ mit $f(1) = a$, $f(2) = b$, $f(3) = d$, $f(4) = c$.
- $G_1 \cong G_3$ via $f\colon \{1,2,3,4\} \to \{a,b,c,d\}$ mit $f(1) = c$, $f(2) = d$, $f(3) = a$, $f(4) = b$.
- G_3 ist nicht isomorph zu G_4, kurz: $G_3 \ncong G_4$, da G_4 **mehr** Kanten als G_3 hat.

5.1.1.3. Spezielle Graphklassen

Die folgenden Graphklassen sollte man kennen.

1. Kreisfreie Graphen

Definition 5.23. (Graphen ohne Kreise).

Wald
Baum

(a) Ein ungerichteter Graph ohne Kreise heißt ein **Wald**. Ein ungerichteter, zusammenhängender Graph ohne Kreise heißt ein **Baum**.

DAG

(b) Ein gerichteter Graph ohne Kreise heißt ein azyklischer Graph. Im Englischen spricht man von einem **directed acyclic graph**, kurz: DAG.

Bäume und Wälder besprechen wir detailliert in Abschnitt 5.2.

2. Vollständige Graphen

Definition 5.24. Sei $n \in \mathbb{N}_{>0}$.

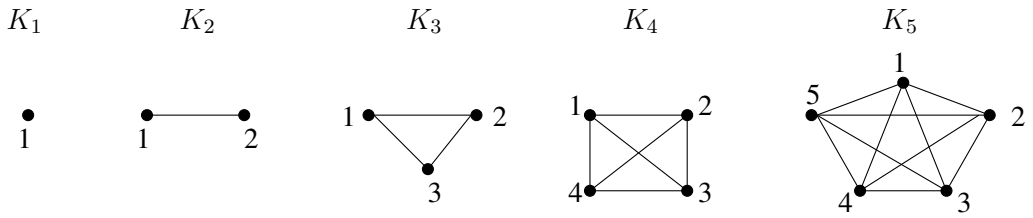
vollständiger ungerichteter Graph K_n

Der **vollständige ungerichtete Graph** K_n hat die Knotenmenge $\{1, \ldots, n\}$ und die Kantenmenge $\big\{\{i, j\} : i, j \in \{1, \ldots, n\},\, i \neq j\big\}$.

vollständiger gerichteter Graph $\vec{K_n}$

Der **vollständige gerichtete Graph** $\vec{K_n}$ hat ebenfalls die Knotenmenge $\{1, \ldots, n\}$. Seine Kantenmenge besteht aus allen Paaren (i, j) mit $i, j \in V$.

Beispiele:



Beobachtung 5.25. Der Graph K_n hat n Knoten und $\dfrac{n \cdot (n-1)}{2}$ Kanten. Der Graph $\vec{K_n}$ hat ebenfalls n Knoten, aber n^2 Kanten.

3. Bipartite Graphen

Bipartite Graphen spielen eine wichtige Rolle in Zuordnungsproblemen (siehe Abschnitt 5.1.2.3).

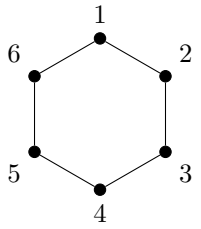
Definition 5.26 (bipartiter Graph).

bipartit

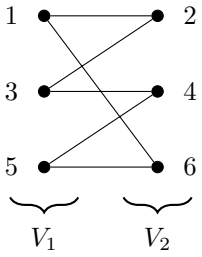
Ein ungerichteter Graph $G = (V, E)$ heißt **bipartit**, wenn seine Knotenmenge V so in zwei disjunkte Teilmengen V_1 und V_2 zerlegt werden kann (d.h. $V = V_1 \dot\cup V_2$), dass jede Kante aus E einen Endknoten in V_1 und einen Endknoten in V_2 hat.

Wann ist ein ungerichteter Graph bipartit?

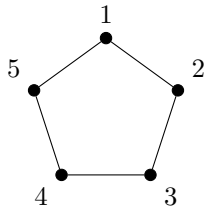
1. Der Graph



ist bipartit mit der Zerlegung $V_1 = \{1, 3, 5\}$ und $V_2 = \{2, 4, 6\}$. Der Graph lässt sich auch wie folgt graphisch darstellen:



2. Der Graph



ist **nicht** bipartit.

Beweis: Durch Widerspruch. Angenommen, er ist doch bipartit. Dann seien V_1 und V_2 die beiden disjunkten Teilmengen der Knotenmenge, so dass jede Kante des Graphen einen Endknoten in V_1 und einen Endknoten in V_2 hat. Wir können ohne Beschränkung der Allgemeinheit annehmen, dass $1 \in V_1$ ist (falls nicht, vertauschen wir einfach V_1 und V_2). Dann muss aber gelten: $2 \in V_2$, $3 \in V_1$, $4 \in V_2$ und $5 \in V_1$, also $V_1 = \{1, 3, 5\}$ und $V_2 = \{2, 4\}$. Im Graphen gibt es aber auch eine Kante zwischen 1 und 5, und beide Knoten gehören zu V_1. Dies ist ein Widerspruch zu der Annahme, dass jede Kante einen Endpunkt in V_1 und einen Endpunkt in V_2 hat. □

Allgemein kann man zeigen (siehe Übungen):

Ein Graph G ist genau dann bipartit, wenn G keinen Kreis ungerader Länge besitzt.

Definition 5.27. Seien $m, n \in \mathbb{N}_{>0}$. Der **vollständige bipartite Graph** $K_{m,n}$ hat die Knotenmenge

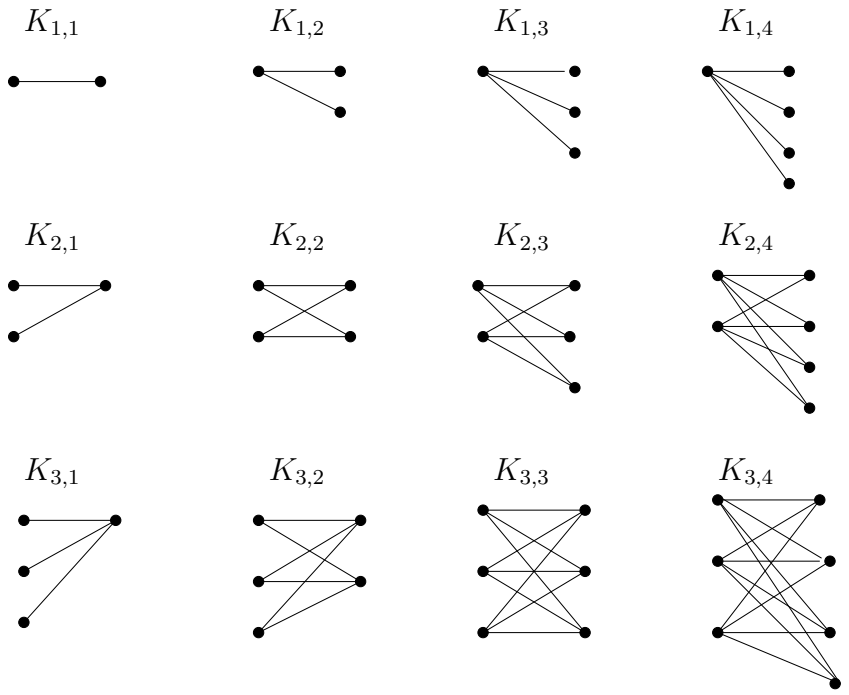
$$\big\{\, (1,i) : i \in \{1,\dots,m\} \,\big\} \;\cup\; \big\{\, (2,j) : j \in \{1,\dots,n\} \,\big\}$$

und die Kantenmenge

$$\big\{\, \{\,(1,i),\,(2,j)\,\} \;:\; i \in \{1,\dots,m\},\; j \in \{1,\dots,n\} \,\big\}.$$

vollständiger bipartiter Graph $K_{m,n}$

Beispiele:



Beobachtung 5.28. Der Graph $K_{m,n}$ hat $m + n$ Knoten und $m \cdot n$ Kanten.

Notation 5.29. Ein ungerichteter Graph G mit nicht-leerer Knotenmenge heißt

vollständig

(a) **vollständig**, falls es ein $n \in \mathbb{N}_{>0}$ gibt, so dass $G \cong K_n$ (d.h. G ist isomorph zu K_n).

vollständig bipartit

(b) **vollständig bipartit**, falls es Zahlen $m, n \in \mathbb{N}_{>0}$ gibt, so dass $G \cong K_{m,n}$.

4. Der d-dimensionale Würfel

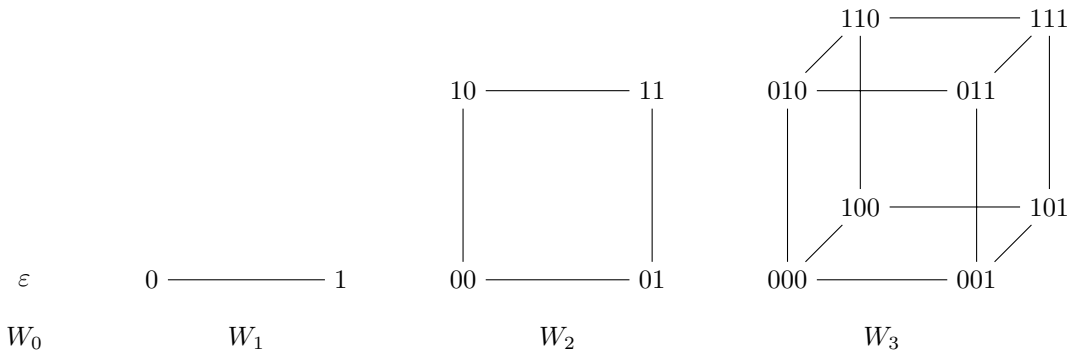
d-dimensionaler Würfel W_d

Definition 5.30. Der **d-dimensionale Würfel** W_d besitzt die Knotenmenge $V_d = \{0,1\}^d$ und die Kantenmenge

$$E_d = \{\, \{u, v\} \,:\, u, v \in \{0,1\}^d \,,\ u \text{ und } v \text{ unterscheiden sich in genau einem Bit } \}.$$

Man modelliert ein verteiltes System von Rechnern durch einen ungerichteten Graphen, der die „Kommunikationsstruktur“ des Systems beschreibt. Die Knoten des Graphen entsprechen den Rechnern, die Kanten Direktverbindungen zwischen Rechnern. Man wählt häufig einen Würfel als Kommunikationsstruktur. Warum?

W_d hat 2^d Knoten und „nur“ $d \cdot 2^{d-1}$ Kanten. Im Vergleich zur Kantenzahl $\Theta(2^{2d})$ des vollständigen Graphen K_{2^d} ist die Kantenzahl sehr viel kleiner. Nachrichten lassen sich schnell zwischen Knoten austauschen, denn der größte Abstand zwischen zwei Knoten ist d. Man kann zeigen, dass W_d keinen „Flaschenhals“ hat: Mindestens 2^d Kanten verbinden zwei disjunkte Knotenmengen der Größe 2^{d-1}.



Der Würfel W_d für $d \in \{0, \ldots, 3\}$.

Frage: Der Würfel W_d ist für jede Dimension d bipartit. Warum?

5.1.2. Schnell lösbare Probleme für Graphen

Wie findet man schnell aus einem Labyrinth? Was ist die grundlegende algorithmische Idee für Navis? Wie lassen sich Zuordnungsprobleme optimal lösen?

All diese Fragen lassen sich effizient lösen. Die hierzu notwendigen algorithmische Ideen lernen Sie in den Veranstaltungen „Datenstrukturen“, „Theoretische Informatik 1“ und „Approximationsalgorithmen“ kennen, aber einige algorithmische Ideen werden wir hier skizzieren.

5.1.2.1. Suche in Graphen

Wie findet man den Ausgang in einem Labyrinth?

Ein Auszug aus *Umbert Eco's „Der Name der Rose“*. William von Baskerville und sein Schüler Adson van Melk sind heimlich in die als Labyrinth gebaute Bibliothek eines hochmittelalterlichen Klosters irgendwo im heutigen Norditalien eingedrungen. Fasziniert von den geistigen Schätzen, die sie beherbergt, haben sie sich nicht die Mühe gemacht, sich ihren Weg zu merken. Erst zu spät erkennen sie, dass die Räume unregelmäßig und scheinbar wirr miteinander verbunden sind. Man sitzt fest. William erinnert sich:

> „Um den Ausgang aus einem Labyrinth zu finden,“, dozierte William, „gibt es nur ein Mittel. An jedem Kreuzungspunkt wird der Durchgang, durch den man gekommen ist, mit drei Zeichen markiert. Erkennt man an den bereits vorhandenen Zeichen auf einem der Durchgänge, dass man an der betreffenden Kreuzung schon einmal gewesen ist, bringt man an dem Durchgang, durch den man gekommen ist, nur ein Zeichen an. Sind alle Durchgänge schon mit Zeichen versehen, so muss man umkehren und zurückgehen. Sind aber einer oder zwei Durchgänge der Kreuzung noch nicht mit Zeichen versehen, so wählt man einen davon und bringt zwei Zeichen an. Durchschreitet man einen Durchgang, der nur ein Zeichen trägt, so markiert man ihn mit zwei weiteren, so dass er nun drei Zeichen trägt. Alle Teile des Labyrinthes müssten durchlaufen worden sein,wenn man, sobald man an eine Kreuzung gelangt, niemals den Durchgang mit drei Zeichen nimmt, sofern noch einer der anderen Durchgänge frei von Zeichen ist.
>
> Woher wisst Ihr das? Seid Ihr ein Experte in Labyrinthen?
> Nein, ich rezitiere nur einen alten Text, den ich einmal gelesen habe.
> Und nach dieser Regel gelangt man hinaus?
> Nicht dass ich wüsste. Aber wir probieren es trotzdem.[...]“

Geht das denn nicht viel einfacher? Prinzessin Ariadne, Tochter des Königs Minos, hat Theseus den „**Ariadne-Faden**“ geschenkt. Theseus hat den Ariadne-Faden während der Suche im Ariadne-Faden

Labyrinth abgerollt. Nachdem er den Minotauros aufgespürt und getötet hat, braucht er nur den Faden zurückverfolgen, um das Labyrinth wieder verlassen zu können.

Tiefensuche

Aber wie durchsucht man das Labyrinth systematisch mit Hilfe eines Fadens? Der Algorithmus **Tiefensuche** implementiert und erweitert die Methode des Ariadne-Fadens mit Hilfe eines Farbeimers: Wir nehmen an, dass der ungerichtete Graph $G = (V, E)$ sowie ein Startknoten $s \in V$ gegeben ist und dass alle Knoten in V unmarkiert sind. Jetzt wird das rekursive Programm „Tiefensuche(s)“ aufgerufen:

Algorithmus 5.31. Tiefensuche(u)

1. Der Knoten u wird markiert.

   /* Der Knoten u wird sofort angepinselt. */

2. Alle Nachbarn v von u werden in beliebiger Reihenfolge bearbeitet:
   (a) Wenn v markiert ist, tue nichts.
   (b) Wenn v unmarkiert ist, rufe „Tiefensuche(v)“ auf.

3. Der Aufruf „Tiefensuche(u)“ endet, wenn alle Nachbarn von u bearbeitet wurden.

   /* Für jeden unmarkierten Nachbarn v von u endet der Aufruf „Tiefensuche(v)“ irgendwann und die Kontrolle wird mit Hilfe des Ariadne-Fadens an „Tiefensuche(u)“ zurückgegeben, bis „Tiefensuche(u)“ schließlich selbst endet. */

Tiefensuche ist schnell, denn jede Kante $\{u, v\}$ wird höchstens zweimal „angefasst“, nämlich wenn einer der beiden Knoten zuerst und nachfolgend sein Nachbar besucht wird. Aber „funktioniert“ Tiefensuche und was bedeutet das überhaupt? Es sollte bedeuten, dass Tiefensuche alle vom Startknoten s aus erreichbaren Knoten besucht und dass dies die einzigen Knoten sind, die besucht werden.

Satz 5.32. Tiefensuche besucht alle Knoten der Zusammenhangskomponente von s und besucht sonst keine weiteren Knoten.

Beweis: Angenommen, Tiefensuche besucht alle Knoten im Abstand d von s. Der Knoten w habe den Abstand $d + 1$ von s.

Dann gibt es einen Knoten v im Abstand d von s und w ist ein Nachbar von v. Der Knoten w wird inspiziert bevor Tiefensuche für den Knoten v terminiert:

- Entweder ist w zum Zeitpunkt der Inspektion unmarkiert und wird besucht
- oder w wurde zwischenzeitlich besucht.

Wenn alle Knoten im Abstand d von s besucht werden, dann auch alle Knoten im Abstand $d + 1$ von s. Ein induktives Argument zeigt jetzt, dass alle Knoten der Zusammenhangskomponente von s besucht werden.

Warum können keine weiteren Knoten besucht werden? □

In den Vorlesungen „Datenstrukturen“ und „Theoretische Informatik 1“ wird gezeigt, wie man Tiefensuche für die Lösung vieler wichtiger algorithmischer Probleme anwenden kann.

5.1.2.2. Das kürzeste-Wege Problem

Wie gehen Routenplaner vor, um schnellste Verbindungen zwischen einem Start und einem Ziel zu berechnen? Die relevanten Informationen werden in einen gerichteten Graphen gepackt.

(a) Die Knoten des gerichteten Graphen entsprechen den Kreuzungen oder Abfahrten, gerichtete Kanten entsprechen Straßenabschnitten, die Knoten direkt miteinander verbinden.

(b) Jede Kante besitzt geographische Information.

- Für die Navigation ist wichtig, in welchem Stadtteil, Stadt, Bundesland oder Staat der entsprechende Straßenabschnitt liegt und welche Hausnummern vorkommen.
- Um schnellste Verbindungen zu berechnen wird einer Kante die Dauer zugewiesen, wenn der entsprechende Straßenabschnitt in Regelgeschwindigkeit gemäß den Verkehrsregeln durchfahren wird.

Definition 5.33. (a) Für den gerichteten Graphen $G = (V, E)$ weist eine Funktion

$$\text{länge} : E \to \mathbb{R}_{\geqslant 0}$$

jeder Kante eine nicht-negative Länge zu.

(b) Im **Problem der kürzesten Wege** ist für einen „Start"-Knoten $s \in V$ und einen „Ziel"-Knoten $t \in V$ ein Weg $W = (s, v_1, v_2, \ldots, v_k, t)$ von s nach t kürzester Länge zu bestimmen, wobei

$$\text{länge}(s, v_1) + \text{länge}(v_1, v_2) + \cdots + \text{länge}(v_k, t)$$

die Länge von W ist.

Problem der kürzesten Wege

In den Veranstaltungen „Datenstrukturen" und „Theoretische Informatik 1" wird eine schnelle Lösung dieses „Kürzesten-Wege-Problems" mit *dem Algorithmus von Dijkstra* beschrieben.

5.1.2.3. Zuordnungsprobleme

Wir betrachten zunächst zwei typische Beispiele von Zuordnungsproblemen.

Beispiel 5.34. (a) In einem Tennisverein sollen die Vereinsmitglieder für ein Turnier zu Doppelpaarungen zusammengestellt werden. Dabei möchte man jeweils nur befreundete Personen als „Doppel" zusammen spielen lassen.

Um diese Aufgabe zu lösen, modellieren wir die Situation durch den ungerichteten Graphen $G_T := (V_T, E_T)$ mit

$$\begin{aligned} V_T &:= \{x : x \text{ ist ein Vereinsmitglied}\} \\ E_T &:= \{\{x, y\} : x \text{ und } y \text{ sind miteinander befreundete Vereinsmitglieder}\}. \end{aligned}$$

Das **Ziel** ist, eine größtmögliche Anzahl von Doppelpaarungen zu finden.

(b) Eine Gruppe unterschiedlich ausgebildeter Piloten soll so auf Flugzeuge verteilt werden, dass jeder das ihm zugeteilte Flugzeug fliegen kann.

Auch hier modellieren wir die Situation durch einen ungerichteten Graphen $G_F := (V_F, E_F)$ mit

$$\begin{aligned} V_F &:= \{x : x \text{ ist ein Pilot}\} \dot\cup \{y : y \text{ ist ein Flugzeug}\}, \\ E_F &:= \{\{x, y\} : \text{Pilot } x \text{ kann Flugzeug } y \text{ fliegen}\}. \end{aligned}$$

Das **Ziel** ist, einen Flugplan aufzustellen, so dass jeder Pilot das ihm zugeteilte Flugzeug fliegen kann.

Die gesuchten Kantenmengen E' aus ((a)) und ((b)) werden **Matching** (bzw. **Paarung** oder **Menge unabhängiger Kanten**) genannt:

Definition 5.35.
Sei $G = (V, E)$ ein ungerichteter Graph.

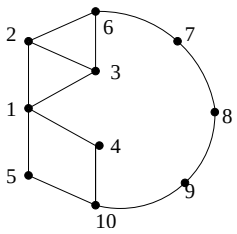
Matching
Paarung
(a) Eine Kantenmenge $E' \subseteq E$ heißt **Matching** (bzw. **Paarung** bzw. **Menge unabhängiger Kanten**), falls kein Knoten aus V Endpunkt von mehr als einer Kante aus E' ist.

perfekt
(b) Ein Matching heißt **perfekt**, falls jeder Knoten aus V Endpunkt einer Kante aus M ist.

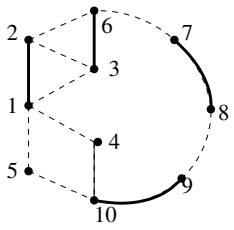
Ziel in Beispiel 5.34 ((a)) und ((b)) ist die Bestimmung eines Matchings maximaler Größe, also eines Matching mit möglichst vielen Kanten. Beachte, dass ein perfektes Matching stets größtmöglich ist.

Beispiel 5.36. In einem Tennisverein mit 10 Mitgliedern und „Freundschaftsgraph“

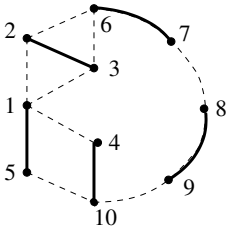
G_T =



sind z.B. die folgenden beiden Kantenmengen Matchings:



und

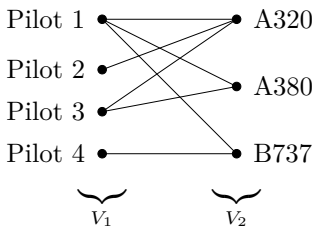


$E' = \{\{1,2\},\{3,6\},\{7,8\},\{9,10\}\}$ $E'' = \{\{1,5\},\{4,10\},\{8,9\},\{6,7\},\{2,3\}\}$.

In Beispiel 5.34((b)) sollten Piloten auf Flugzeuge verteilt werden. Die Knotenmenge des zugehörigen Graphen G_F bestand aus zwei verschiedenen Arten von Objekten (nämlich einerseits

Piloten und andererseits Flugzeuge), und Kanten konnten jeweils nur zwischen Objekten unterschiedlicher Art verlaufen (also zwischen Piloten und Flugzeugen, nicht aber zwischen Piloten und Piloten bzw. Flugzeugen und Flugzeugen).

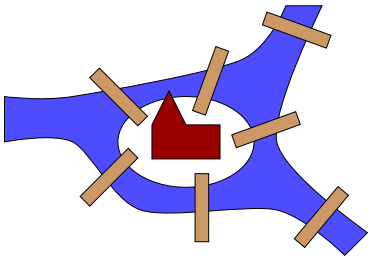
Der Graph



ist bipartit mit $V_1 = \{\text{Pilot}\,1, \text{Pilot}\,2, \text{Pilot}\,3, \text{Pilot}\,4\}$ und $V_2 = \{\text{A320}, \text{A380}, \text{B737}\}$.

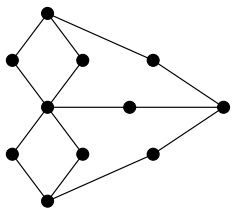
5.1.2.4. Euler-Kreise

Beispiel 5.37 (Königsberger Brückenproblem).
Die Stadt Königsberg wird durch die Pregel in vier Ufergebiete, bzw. Inseln aufgeteilt. Im 18. Jahrhundert gab es sieben Brücken, die die Ufer und zwei Inseln in der folgenden Weise miteinander verbanden.



Frage: Gibt es einen Spaziergang, der jede der sieben Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt?

Die obige Skizze lässt sich durch einen ungerichteten Graphen modellieren: Für jedes Ufer, jede Insel und jede Brücke gibt es einen Knoten; Kanten zeigen direkte Verbindungen an. Die Skizze wird also durch folgenden Graphen repräsentiert:

$G_{\text{Königsberg}} :=$ 

Die Frage nach dem „Spaziergang“ entspricht dann gerade der Frage:

Gibt es in $G_{\text{Königsberg}}$ einen Euler-Kreis?

Definition 5.38 (Euler-Kreise und Euler-Wege).
Sei $G = (V, E)$ ein ungerichteter Graph.

Euler-Weg

(a) Ein Weg $W = (v_0, \ldots, v_\ell)$ heißt **Euler-Weg**, wenn W jede Kante aus E genau einmal durchläuft, d.h. wenn es für jedes $e \in E$ genau ein $i \in \{0, \ldots, \ell - 1\}$ gibt, so dass $e = \{v_i, v_{i+1}\}$.

Euler-Kreis

(b) Ein Weg $W = (v_0, \ldots, v_\ell)$ heißt **Euler-Kreis**, wenn W ein Euler-Weg ist und $v_0 = v_\ell$ ist.

Satz 5.39 (Existenz von Euler-Kreisen und Euler-Wegen).
Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph. Dann gilt:

(a) G besitzt einen Euler-Kreis $\iff$ jeder Knoten von G hat einen geraden Grad (d.h. ist mit einer geraden Anzahl von Kanten inzident).

(b) G besitzt einen Euler-Weg, $\iff$ es gibt in G genau zwei Knoten mit ungeradem Grad. der kein Euler-Kreis ist

Beweis:

(a) „$\Longrightarrow$“: Sei $K = (v_0, \ldots, v_\ell)$ ein Euler-Kreis. Insbesondere gilt: $v_0 = v_\ell$.

Schritt 1: Jeder Knoten $v \in \{v_0, \ldots, v_{\ell-1}\}$ hat geraden Grad, denn:
Sei $v \in \{v_0, \ldots, v_{\ell-1}\}$ beliebig. Zu jedem $i \in \{0, \ldots, \ell - 1\}$ mit $v = v_i$ gibt es im Euler-Kreis K zwei verschiedene Kanten, nämlich

- $\{v_{i-1}, v_i\}$ und $\{v_i, v_{i+1}\}$, falls $i \neq 0$, bzw.
- $\{v_0, v_1\}$ und $\{v_{\ell-1}, v_0\}$, falls $i = 0$ (beachte: $v_0 = v_\ell$).

Da der Euler-Kreis K jede Kante von G genau einmal enthält, gilt somit Folgendes: Ist $k = |\{i \in \{0, \ldots, \ell - 1\} : v = v_i\}|$ (d.h. k gibt an, wie oft v im Tupel $(v_0, \ldots, v_{\ell-1})$ vorkommt), so ist $\text{Grad}_G(v) = 2 \cdot k$. Daher hat jeder Knoten $v \in \{v_0, \ldots, v_{\ell-1}\}$ geraden Grad.

Schritt 2: $\{v_0, \ldots, v_{\ell-1}\} = V$, denn:
Laut Voraussetzung ist G zusammenhängend. Für beliebige Knoten $v, w \in V$ gilt daher: es gibt in G einen Weg von v nach w. Da K ein Euler-Kreis ist, enthält K sämtliche Kanten, die auf dem Weg von v nach w vorkommen. Insbesondere gilt also f.a. $v, w \in V$, dass $v, w \in \{v_0, \ldots, v_{\ell-1}\}$.

Zusammenfassung: Aus den Schritten 1 und 2 folgt direkt, dass jeder Knoten von G geraden Grad hat.

„$\Longleftarrow$“: Sei G ein zusammenhängender ungerichteter Graph, in dem jeder Knoten geraden Grad hat. Es sei

$$W \;=\; (v_0, \ldots, v_\ell)$$

ein Weg **maximaler Länge** in G, der **keine Kante(n) mehrfach** enthält. Da wir W nicht mehr verlängern können, liegen alle mit v_ℓ inzidenten Kanten auf W. Da laut unserer Voraussetzung die Anzahl dieser Kanten gerade ist, folgt $v_\ell = v_0$.

Zu zeigen: W ist ein Euler-Kreis.

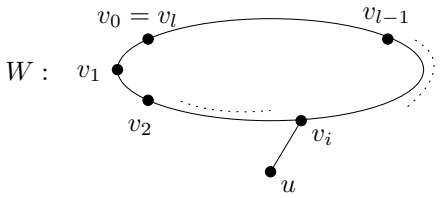
Angenommen, W ist **kein** Euler-Kreis.

Dann gibt es in G eine Kante e', die nicht auf W liegt. Da G zusammenhängend ist, gibt es einen Weg, der von einem Endknoten von e' zu einem zu W gehörenden Knoten führt. Sei e die erste Kante auf diesem Weg, die einen Endpunkt in W hat. Sei v_i der zu e inzidente Knoten aus W und sei $u \in V$ der andere zu e inzidente Knoten, d.h. $e = \{u, v_i\}$. Dann ist der Weg

$$W' := (u, v_i, v_{i+1}, \ldots, v_{\ell-1}, v_0, v_1, \ldots, v_i)$$

ein Weg der Länge $\ell + 1$, der keine Kante(n) mehrfach enthält.

Skizze:



Dies widerspricht aber der Tatsache, dass W ein Weg **maximaler** Länge ist.

(b) Die Richtung „$\Longrightarrow$“ folgt analog zu ((a)): Sei $K = (v_0, \ldots, v_\ell)$ ein Euler-Weg, der kein Euler-Kreis ist. Man sieht leicht, dass die beiden Endknoten von K ungeraden Grad haben, und dass alle anderen Knoten geraden Grad haben.

Zum Beweis der Richtung „$\Longleftarrow$“ kann man ((a)) verwenden: Seien x und y die beiden Knoten von G, die ungeraden Grad haben. Wir betrachten den Graphen $G' := (V', E')$ mit $V' := V \cup \{z\}$ und $E' := E \cup \{\{x, z\},\ \{y, z\}\}$, wobei z ein „neuer“ Knoten ist, der nicht zu V gehört.

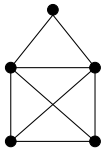
Offensichtlich hat jeder Knoten in G' geraden Grad. Außerdem ist G' zusammenhängend (da G zusammenhängend ist). Aus ((a)) folgt, dass G' einen Euler-Kreis besitzt. Wegen $\mathrm{Grad}_{G'}(z) = 2$ wird z auf diesem Kreis genau einmal besucht. Durch Entfernen der Kanten $\{x, z\}$ und $\{z, y\}$ erhält man einen Euler-Weg in G, der die beiden Knoten x und y als Anfangs- und Endpunkt hat. □

Beispiel 5.40 (Lösung des Königsberger Brückenproblems).
Mit Hilfe von Satz 5.39 können wir das Königsberger Brückenproblem aus Beispiel 5.37 leicht lösen: Es gibt **keinen** Spaziergang, der jede der sieben Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt.

Beweis: Ein solcher Spaziergang würde gerade einem Euler-Kreis im Graphen $G_{\text{Königsberg}}$ entsprechen. Dieser Graph besitzt aber vier Knoten von ungeradem Grad und kann daher laut Satz 5.39((a)) keinen Euler-Kreis besitzen. □

Beispiel 5.41.
Unter Verwendung von Satz 5.39 kann man auch die folgende Frage leicht lösen.

Frage: Kann man das „Haus vom Nikolaus“



in einem Zug nachzeichnen? D.h: Besitzt der Graph einen Euler-Weg?

Der obige Graph besitzt genau 2 Knoten von ungeradem Grad. Nach Satz 5.39 gibt es also einen Euler-Weg, aber keinen Euler-Kreis.

5.1.3. Schwierige Graph-Probleme

5.1.3.1. Hamilton-Kreise

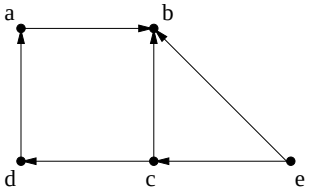
Im Gegensatz zu Euler-Wegen, bei denen es darum geht, einen Weg zu finden, der jede **Kante** des Graphen genau einmal besucht, geht es bei den im Folgenden betrachteten **Hamilton-Wegen** darum, einen Weg zu finden, der jeden **Knoten** des Graphen genau einmal besucht.

Definition 5.42 (Hamilton-Kreise und Hamilton-Wege).
Sei $G = (V, E)$ ein (gerichteter oder ein ungerichteter) Graph.

Hamilton-Weg (a) Ein Weg $W = (v_0, \ldots, v_\ell)$ heißt **Hamilton-Weg**, wenn jeder **Knoten** aus V genau einmal in W vorkommt.

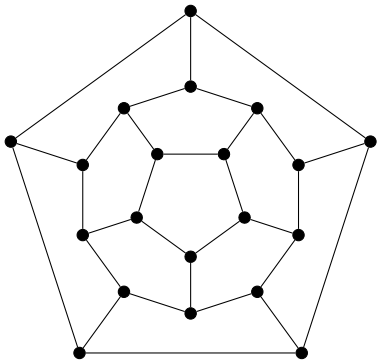
Hamilton-Kreis (b) Ein Weg $W = (v_0, \ldots, v_\ell)$ heißt **Hamilton-Kreis**, wenn $\ell \geqslant 1$ und $v_\ell = v_0$ und $(v_0, \ldots, v_{\ell-1})$ ein Hamilton-Weg ist.

Beispiel: Im Graphen G



gibt es einen Hamilton-Weg, nämlich (e, c, d, a, b), aber keinen Hamilton-Kreis (da Aus-Grad$_G(b) = 0$ ist).

Beispiel 5.43. Im Graphen

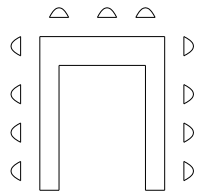


gibt es ebenfalls einen Hamilton-Kreis. Können Sie ihn finden?

Bemerkung 5.44. Das Problem für einen gegebenen Graphen G zu entscheiden, ob G einen Hamilton-Kreis besitzt, ist schwierig: Man kann zeigen, dass es genau wie das KNF-Erfüllbarkeitsproblem (siehe Definition 3.54) NP-vollständig ist.

In der Vorlesung „Theoretische Informatik 1“ wird gezeigt, dass wir ein beliebiges NP-vollständiges Problem effizient lösen können, wenn irgendein anderes NP-vollständiges effizient lösbar ist. Wenn wir also die Erfüllbarkeit von KNF-Formeln effizient entscheiden könnten, dann können wir auch effizient entscheiden, ob ein Graph einen Hamilton-Kreis besitzt –und umgekehrt.

Beispiel 5.45 (Sitzordnung bei einer Familienfeier).
Die Gäste einer Familienfeier sollen so an einer hufeisenförmigen Tafel



platziert werden, dass niemand neben jemanden sitzt, den er nicht leiden kann.

Lösungsansatz:

Schritt 1: Stelle den **Konfliktgraphen** $G = (V, E)$ auf, wobei

$$\begin{aligned} V &:= \{x : \text{Person } x \text{ soll zur Feier kommen}\} \text{ und} \\ E &:= \{\ \{x, y\} : \text{ Person } x \text{ kann Person } y \text{ nicht leiden }\ \} \end{aligned}$$

d.h. Kanten im Konfliktgraphen zeigen auf, wer im Konflikt mit wem steht.

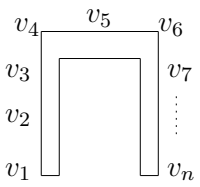
Schritt 2: Bilde das Komplement des Konfliktgraphen, d.h. den Graphen $\overline{G} = (\overline{V}, \overline{E})$ mit

$$\begin{aligned} \overline{V} &:= V \quad \text{und} \\ \overline{E} &:= \big\{\{x, y\} : x, y \in V,\, x \neq y,\, \{x, y\} \notin E\big\}, \end{aligned}$$

Die Kanten in $\overline{G}$ zeigen an, welche Gäste prinzipiell nebeneinander platziert werden könnten.

Schritt 3: Suche einen Hamilton-Weg in $\overline{G}$.

Wenn $(v_1, \ldots, v_n)$ (mit $n = |\overline{V}|$) ein Hamilton-Weg in $\overline{G}$ ist, dann kann man die Sitzordnung folgendermaßen festlegen:



Angenommen, es gibt in $\overline{G}$ keinen Hamilton-Weg. Dann ist es nicht möglich, die geladenen Gäste so an einer hufeisenförmigen Tafel zu platzieren, dass niemand neben jemandem sitzt, den sie/er nicht leiden kann. □ Ende von Beispiel 5.45

TSP

Beispiel 5.46. Beim Problem des Handlungsreisenden (engl.: **Travelling Salesman Problem**, kurz: **TSP**) ist eine Rundreise minimaler Länge gesucht, sodass jede Stadt genau einmal besucht wird. Es geht also darum, einen Hamilton-Kreis kürzester Länge zu finden. □ Ende von Beispiel 5.46

5.1.3.2. Die chromatische Zahl

Wir möchten die Knoten eines ungerichteten Graphen so färben, dass Kanten nur verschieden gefärbte Knoten miteinander verbinden.

Definition 5.47 (Färbungen).
Sei $G = (V, E)$ ein ungerichteter Graph und F eine Menge.

Färbung
(a) Eine Funktion $m\colon V \to F$ heißt eine **Färbung** von G, wenn für jede Kante $\{x, y\} \in E$ gilt: $m(x) \neq m(y)$.

k-Färbung
(b) Eine Färbung $m\colon V \to F$ von G heißt **k-Färbung**, wenn F genau k Elemete besitzt.

k-färbbar
chromatische Zahl
(c) G heißt **k-färbbar**, wenn G eine k-Färbung besitzt. Ist G k-färbbar, aber nicht $(k-1)$-färbbar, dann heißt $\chi(G) := k$ die **chromatische Zahl** von G.

Färbungsproblem
(d) Im **Färbungsproblem** für G ist eine Färbung von G mit $\chi(G)$ Farben zu bestimmen.

Beispiel 5.48. Sei G ein Graph mit n Knoten und sei $m : V \to \{1, \ldots, n\}$ eine bijektive Funktion. Dann ist m eine Färbung von G.

Frage: Also hat jeder Graph mit n Knoten auch eine chromatische Zahl von höchstens n. Aber werden so viele Farben auch wirklich benötigt, d.h. gibt es einen Graphen G mit n Knoten für den $\chi(G) = n$ gilt?

Beispiel 5.49. Für einen bipartiten Graphen $G = (V, E)$ kann die Knotenmenge V so in disjunkte Teilmengen V_1 und V_2 zerlegt werden, dass alle Kanten einen Endpunkt in V_1 und einen Endpunkt in V_2 besitzen. Dann ist die Funktion $m : V \to \{1, 2\}$, die alle Knoten in V_1 mit der Farbe „1“ und alle Knoten in V_2 mit der Farbe „2“ färbt eine 2-Färbung von G.

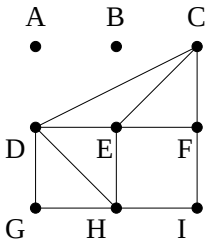
Frage: Für einen bipartiten Graphen G gilt somit $\chi(G) \leqslant 2$. Und andersherum: Ist jeder Graph mit chromatischer Zahl höchstens zwei bipartit?

In Beispiel 5.45 haben wir eine „konfliktfreie“ Sitzordnung an einem hufeisenförmigen Tisch entworfen. Diesmal stellen wir mehrere Tische zur Verfügung.

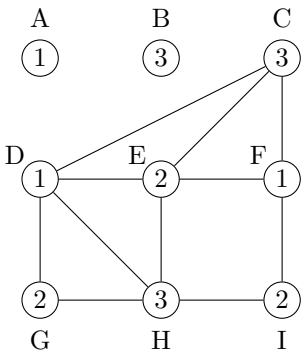
Beispiel 5.50 (Sitzordnung bei einer Familienfeier, Teil 2).
Die Gäste einer Familienfeier sollen so an möglichst wenigen Tischen platziert werden, dass Personen, die sich nicht leiden können, an verschiedenen Tischen sitzen.

Diese Aufgabe kann folgendermaßen modelliert werden: Die verfügbaren Tische werden mit den Zahlen $1, 2, 3, \ldots$ durchnummeriert. Die geladenen Gäste und die herrschenden Konflikte zwischen Gästen werden durch den in Beispiel 5.45 betrachteten Konfliktgraphen $G = (V, E)$ repräsentiert. Die Zuordnung, wer an welchem Tisch sitzen soll, wird durch eine Färbung $m\colon V \to F$ mit $F \subseteq \mathbb{N}$ repräsentiert, wobei $m(x) = i$ bedeutet, dass Person x am Tisch i sitzen soll.

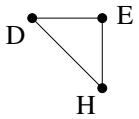
Das Ziel ist die Bestimmung einer Färbung $m\colon V \to F$ des Konfliktgraphen mit möglichst wenigen Farben – dies entspricht dem Ziel, die Gäste auf möglichst wenige Tische zu verteilen. Betrachten wir zum Beispiel eine Familienfeier mit Gästen A, B, C, D, E, F, G, H, I und folgendem Konfliktgraphen:



Die folgende Graphik gibt eine Färbung $m\colon V \to \{1,2,3\}$ des Konfliktgraphen an, wobei für jeden Knoten $v \in V$ der Wert $m(v)$ in den Kreis geschrieben ist, der den Knoten v repräsentiert.



Wir haben eine 3-Färbung gefunden, die Gäste werden also an 3 Tische verteilt. Dies ist optimal, da der Konfliktgraph ein Dreieck, z.B.



als Teilgraphen enthält: Jede Färbung des Konfliktgraphen muss mindestens drei Farben besitzen.

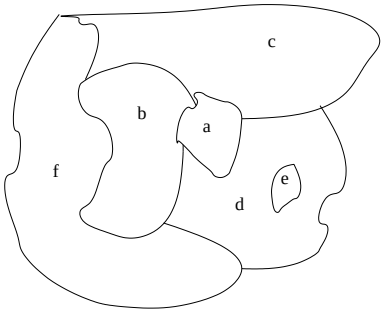
Bemerkung 5.51 (4-Farben-Problem).
Das **4-Farben-Problem** gehört zu den bekanntesten Färbungsproblemen. Hier handelt es sich um die Frage, wie viele verschiedene Farben nötig sind, um jede Landkarte so einzufärben, dass zwei Staaten[2], die ein Stück gemeinsamer Grenze haben, durch unterschiedliche Farben dargestellt werden. 1976 wurde bewiesen, dass vier Farben ausreichen. Der Beweis basiert auf einer Fallunterscheidung mit mehr als 1000 Fällen, die mit Hilfe eines Computerprogramms analysiert wurden.

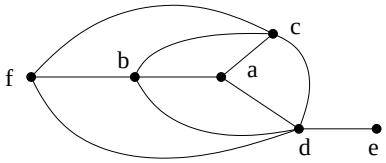
4-Farben-Problem

Eine Landkarte kann durch einen ungerichteten Graphen G modelliert werden, dessen Knoten die Staaten repräsentieren, und bei dem es eine Kante zwischen zwei Staaten gibt, falls diese ein gemeinsames Grenzstück besitzen. Ziel ist die Konstruktion einer Färbung von G mit möglichst wenigen Farben.

[2] Es wird angenommen, dass das Gebiet eines jeden Staats zusammenhängend ist.

Beispiel:
Wir betrachten eine kleine Landkarte und den zugehörigen Konfliktgraphen:





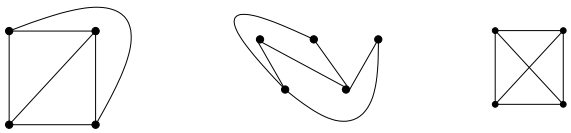
Knoten $\widehat{=}$ Staaten
Kanten $\widehat{=}$ Staaten mit gemeinsamer Grenze

Da bei den vier Knoten a, b, c, d paarweise jeder zu jedem benachbart ist, muss eine Färbung des Konfliktgraphen diesen vier Knoten vier verschiedene Farben zuordnen — für a, b, c, d etwa *rot*, *gelb*, *grün*, *blau*. Da f außerdem mit b, c, d benachbart ist, muss f dann wieder *rot* gefärbt sein; e kann jede Farbe außer *blau* erhalten.

Die aus Landkarten entstehenden Konfliktgraphen haben eine besondere Eigenschaft: Sie sind **planar**.

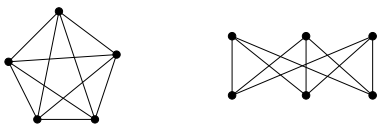
Definition 5.52 (planare Graphen).
planar Ein Graph G heißt **planar**, wenn er so in die Ebene gezeichnet werden kann, dass seine Kanten sich nicht kreuzen.

Beispiele für planare Graphen sind:



(Der dritte Graph ist planar, da er wie der erste Graph kreuzungsfrei in die Ebene gezeichnet werden kann.)

Beispiele für nicht-planare Graphen sind:



Bemerkung 5.53. (Anwendungen des Färbungsproblems).
Das Färbungsproblem besitzt zahlreiche Anwendungen. Viele dieser Anwendungen beinhalten eine optimale Auflösung von Konflikten.

Knoten	Kanten	Farbe/Markierung
Staat auf Karte	gemeinsame Grenze	Farbe
Gast auf Feier	können sich nicht leiden	Tischnummer
Vorlesung	gemeinsame Hörer	Termin
Prozess	benötigen dieselbe Ressource	Ausführungszeitpunkt

Das Färbungsproblem hat viele wichtige Anwendungen, aber leider ist schon die Frage, ob 3 Farben ausreichen, NP-vollständig, also vermutlich nicht effizient lösbar :-((.

Schlimmer noch, bis heute sind keine schnellen Algorithmen bekannt, die auf Graphen mit n Knoten eine Färbung mit beispielsweise $\sqrt{n}$ Farben finden, selbst wenn drei Farben ausreichen. Der Grund für diese schlechte Approximationsleistung“ in der großen Komplexität des Färbungsproblems zu sehen: In bahnbrechenden Arbeiten konnte mit der Methode der „Probabilistically Checkable Proofs“ in den 90er Jahren gezeigt werden, dass die „Fähigkeit der schnellen, guten Approximation“ bereits die „Fähigkeit der schnellen, exakten Lösung“ bedeutet. Diese Ergebnisse werden in der Veranstaltung „Komplexitätstheorie“ behandelt.

Aber schwierige Probleme wie das Färbungsproblem begründen die Wichtigkeit der Informatik, denn ein einzelnes Verfahren wird nicht auf allen Instanzen erfolgreich sein, sondern für viele Instanzen müssen maßgeschneiderte Verfahren entwickelt werden, die zumindest gut approximieren :-)).

Wir stellen ein einfaches Verfahren vor, das zwar im schlimmsten Fall kläglich versagt, aber für viele (wenn auch nicht für alle) Graphen G gute Approximationen der chromatischen Zahl $\chi(G)$ berechnet.

Algorithmus 5.54 (Eine Färbungsheuristik).

1. Setze $k := 1, F := \{1\}$ und Ungefärbt $:= \{1, \ldots, n\}$.

   /* Wir nehmen an, dass $G = (V, E)$ der Eingabegraph ist und dass $V = \{1, \ldots, n\}$ gilt. Anfänglich ist natürlich jeder Knoten in $\{1, \ldots, n\}$ ungefärbt. */

2. Solange Ungefärbt $\neq \emptyset$ wiederhole

   (a) Für jeden ungefärbten Knoten, also für jeden Knoten $v \in$ Ungefärbt, setze

   $$\text{Optionen}(v) := F \setminus \{\, f \in F : \text{ein Nachbar von } v \text{ wurde mit } f \text{ gefärbt} \,\}.$$

   /* Optionen(v) ist die Menge der Farben, die gegenwärtig für eine Färbung von v zur Verfügung stehen. */

   (b) Wähle einen Knoten $v \in$ Ungefärbt für den, unter allen Knoten in Ungefärbt, die Menge Optionen(v) die wenigsten Elemente besitzt.

   - Wenn $|\text{Optionen}(v)| \geqslant 1$, dann färbe v mit der kleinsten Farbe in Optionen(v).

     /* Wir färben stets einen Knoten mit den wenigsten Optionen. Warum? Wenn nur noch eine Option wählbar ist und wir bisher alles richtig gemacht haben, dann machen wir auch jetzt alles richtig. */

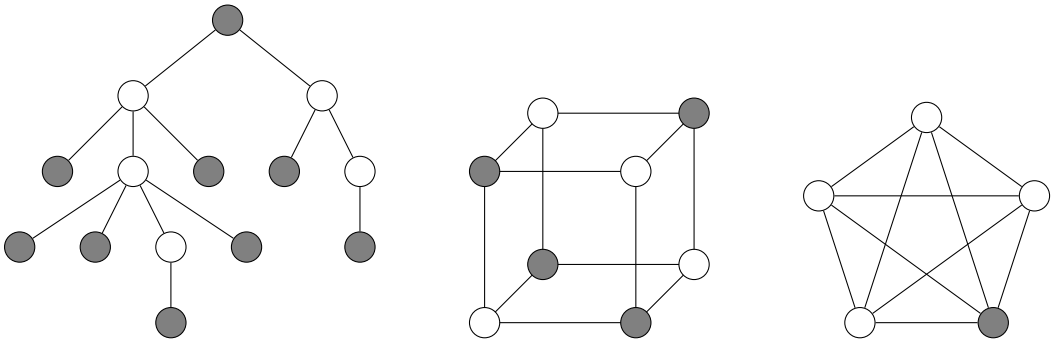


Abbildung 5.1.: Die grau markierten Knoten bilden jeweils eine größte unabhängige Menge.

- Sonst setze $F := F \cup \{k+1\}$, färbe v mit $k+1$ und setze $k := k+1$.

  /* Wenn Optionen(v) $= \emptyset$, dann muss v mit einer neuen Farbe gefärbt werden. */

(c) Setze Ungefärbt := Ungefärbt $\setminus \{v\}$.

Frage: Wie viele Farben benötigt Algorithmus 5.54 für planare Graphen?

5.1.3.3. Unabhängige Mengen, Cliquen und Knotenüberdeckungen

Wir möchten eine möglichst große Menge von Aufgaben gleichzeitig ausführen, dürfen aber keine zwei „in Konflikt stehende“ Aufgaben ausführen. Wir modellieren dieses Problem durch einen ungerichteten Graphen: Knoten entsprechen Aufgaben, Kanten verbinden in Konflikt stehende Aufgaben. Unser Ziel ist die Bestimmung einer möglichst großen unabhängigen Menge:

Definition 5.55. Sei $G = (V, E)$ ein ungerichteter Graph.

unabhängig
Independent Set

(a) Eine Teilmenge $W \subseteq V$ heißt **unabhängig** (engl. **independent** oder **stable**), wenn keine zwei Knoten in W durch eine Kante in E verbunden sind, d.h. es gilt $\{u, v\} \notin E$ für alle Knoten $u, v \in W$. Man nennt eine unabhängige Menge ein „**Independent Set**“.

Im Problem der unabhängigen Mengen (bzw. im Independent Set Problem) ist eine möglichst große unabhängige Menge zu bestimmen.

Clique

(b) Eine Teilmenge $W \subseteq V$ heißt eine **Clique**, wenn je zwei Knoten in W durch eine Kante in E verbunden sind, d.h. es gilt $\{u, v\} \in E$ für alle Knoten $u, v \in W$.

Im Clique-Problem ist eine möglichst große Clique zu bestimmen.

Das Problem der unabhängigen Mengen und das Clique-Problem sind „Zwillingsprobleme“: Ist $W \subseteq V$ eine unabhängige Menge im Graphen $G = (V, E)$, dann ist W eine Clique im Komplement-Graphen $\overline{G} = (V, \overline{E})$.

Um eine möglichst große unabhängige Menge $W \subsetneq V$ zu bestimmen, sollte das Komplement $U = V \setminus W$ möglichst klein sein. Beachte, dass alle Kanten in E mindestens einen Endpunkt in U besitzen müssen: Die Komplementmenge $U = V \setminus W$ ist eine Knotenüberdeckung.

Definition 5.56. Sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $U \subseteq V$ heißt eine **Knotenüberdeckung** (engl. **Vertex Cover**), wenn jede Kante mindestens einen Endpunkt in U besitzt, d.h. für alle Kanten $e \in E$ gilt $e \cap U \neq \emptyset$.

Knotenüberdeckung
Vertex Cover

Im Problem der Knotenüberdeckung (bzw. im Vertex Cover Problem) ist eine möglichst kleine Knotenüberdeckung zu bestimmen.

Achtung: Eine Knotenüberdeckung muss „Kanten überdecken"!

Die Bestimmung einer möglichst großen unabhängigen Menge ist also äquivalent zur Bestimmung einer möglichst kleinen Knotenüberdeckung. Das Problem der Knotenüberdeckung ist aber auch für sich genommen interessant:

Beispiel 5.57. Wir führen eine Reihe von Experimenten durch und versuchen „Ausreißer", also Experimente mit wahrscheinlich verfälschten Messergebnissen, auszusortieren. Dazu wählen wir die Menge der Experimente als Knotenmenge und verbinden zwei Experimente, wenn die Messergebnisse zu stark voneinander abweichen.

Wenn die Messergebnisse der weitaus meisten Experimente nicht verfälscht sind, dann sollte die Menge der Ausreißer einer kleinstmöglichen Knotenüberdeckung entsprechen. □ Ende von Beispiel 5.57

Leider sind aber die beiden Fragen „Gibt es eine unabhängige Menge der Größe mindestens k?", bzw. „Gibt es eine Knotenüberdeckung der Größe höchstens k?" NP-vollständig. Aber das Problem der Knotenüberdeckung besitzt schnelle Algorithmen, die relativ gute approximative Lösungen bestimmen. Hier ist ein Beispiel einer solchen „Heuristik".

Algorithmus 5.58 (Eine Heuristik zur Bestimmung kleiner Knotenüberdeckungen).

1. Der ungerichtete Graph $G = (V, E)$ ist gegeben.
2. Setze $W := \emptyset$.

   /* Die Kantenmenge E wird am Ende der Berechnung mit der Knotenmenge W überdeckt. */
3. Solange $E \neq \emptyset$ wiederhole
   (a) Wähle eine beliebige Kante $e \in E$.
   (b) Wenn $e = \{a, b\}$, dann
   - setze $W := W \cup \{a, b\}$

     /* Jede Überdeckung muss einen der beiden Endpunkte von e besitzen. Die Heuristik wählt beide Endpunkte für die Überdeckung. */
   - und entferne alle Kanten aus E, die a oder b als Endpunkt besitzen.

     /* Die Knoten in W überdecken alle gerade entfernten Kanten. */

Nur durch Knoten in W überdeckte Kanten werden entfernt. Wenn alle Kanten entfernt sind, muss W somit alle Kanten in E überdecken.

Frage: Um wie viel größer als eine kleinste Knotenüberdeckung ist W im schlimmsten Fall? Genauer, wie groß kann der Quotient

$$\frac{|W(G)|}{\text{optimum}(G)}$$

werden, wenn $W(G)$ die von unserer Heuristik für G bestimmte Knotenüberdeckung ist und „optimum(G)“ die Größe der kleinsten Knotenüberdeckung für G bezeichnet?

5.1.3.4. Feedback Vertex Set

Definition 5.59. Sei $G = (V, E)$ ein gerichteter oder ungerichteter Graph und $w : V \to \mathbb{R}$ eine Gewichtung der Knoten.

Feedback-Vertex-Set

(a) Eine Teilmenge $F \subseteq V$ heißt ein **Feedback-Vertex-Set**, wenn G nach Entfernen von F (und aller mit F inzidenten Kanten) keine Kreise mehr besitzt.

(b) Im **Feedback-Vertex-Set Problem** wird ein leichtestes Feedback-Vertex-Set $F \subseteq V$ gesucht, d.h. ein Feedback-Vertex-Set F, das

$$\sum_{v \in F} w(v)$$

unter allen Feedback-Vertex-Sets minimiert.

Für ungerichtete Graphen wird also eine leichteste Knotenmenge gesucht, nach deren Herausnahme der Graph zu einem Wald wird. Im Fall von gerichteten Graphen muss ein Feedback Vertex Set alle gerichteten Kreise treffen.

Eine wichtige Anwendung für gerichtete Graphen ist die Vermeidung von Deadlocks, wenn Prozesse auf nur individuell nutzbare Ressourcen zugreifen. Dazu führen wir einen Graphen G ein, der alle aktiven Prozesse als Knoten besitzt. Ein Kante $i \to j$ von Prozess i nach Prozess j wird eingeführt, wenn Prozess j eine Ressource nutzt, auf deren Freigabe Prozess i wartet. Jeder Kreis in G muss durch den Abbruch von mindestens einem Prozess aufgelöst werden: Wenn die Gewichtung $w : V \to \mathbb{R}$ die Wichtigkeit der Prozesse festhält, dann sollte natürlich eine möglichst leichte Menge von Prozessen abgebrochen werden.

Feedback Vertex Set hat weitere Anwendungen im Entwurf von VLSI-Chips und in der Programmverifizierung.

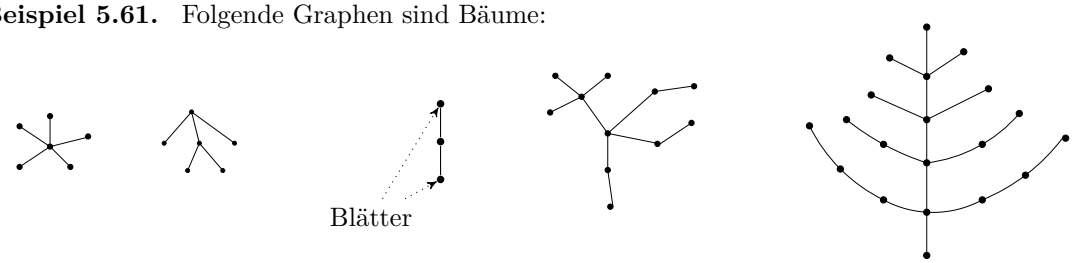
Es ist wie verhext, auch Feedback Vertex Set ist NP-vollständig und stellt somit eine große Herausforderung dar. Wenn aber eine leichteste, bzw. schwerste *Kantenmenge* für einen ungerichteten Graphen gesucht wird, deren Herausnahme alle Kreise zerstört, dann werden wir fündig. Mit der Bestimmung maximaler, bzw. minimaler Spannbäume erhalten wir ein wichtiges, schnell lösbares Problem (siehe Abschnitt 5.2.1).

5.2. Ungerichtete Bäume

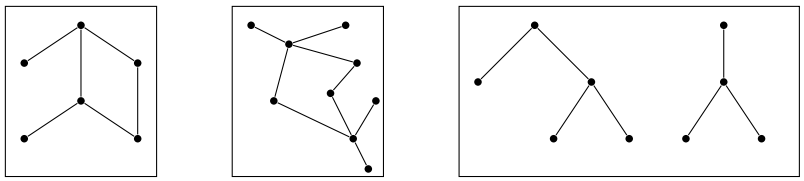
Eine für die Informatik besonders wichtige Art von Graphen sind **Bäume**. Wir betrachten im Folgenden zunächst ungerichtete Bäume und danach gewurzelte Bäume, eine Einschränkung der Klasse gerichteter Bäume.

Definition 5.60 (ungerichteter Baum).
Ein **ungerichteter Baum** ist ein ungerichteter, zusammenhängender Graph $G = (V, E)$, der keinen einfachen Kreis enthält. ungerichteter Baum
Diejenigen Knoten in V, die den Grad $\leqslant 1$ haben, heißen **Blätter** des Baums. Blätter

Beispiel 5.61. Folgende Graphen sind Bäume:



Folgende Graphen sind keine Bäume:

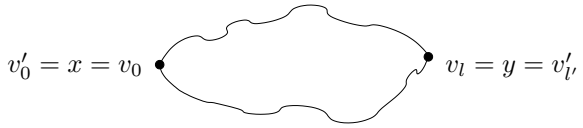


Beobachtung 5.62. Ist $B = (V, E)$ ein ungerichteter Baum, so gilt für alle Knoten $x, y \in V$ mit $x \neq y$:

Es gibt in B genau einen einfachen Weg von x nach y.

Denn: B ist ein ungerichteter Baum, d.h. B ist zusammenhängend und enthält keinen einfachen Kreis. Da B zusammenhängend ist, gibt es mindestens einen einfachen Weg von x nach y. Angenommen, $(v_0, \ldots, v_\ell)$ und $(v'_0, \ldots, v'_{\ell'})$ sind zwei verschiedene einfache Wege von x nach y. Insbesondere gilt dann $v_0 = x = v'_0$ und $v_\ell = y = v'_{\ell'}$.

Skizze:



Dann ist aber $(v_0, \ldots, v_\ell, v'_{\ell'-1}, \ldots, v'_0)$ ein Kreis. Dieser Kreis enthält einen einfachen Kreis. Dann kann B aber kein Baum sein. Widerspruch. □

Satz 5.63.
Jeder ungerichtete Baum $B = (V, E)$ mit $V \neq \emptyset$ besitzt mindestens ein Blatt.

Beweis: Sei $B = (V, E)$ ein ungerichteter Baum mit $V \neq \emptyset$.
Sei $W := (v_0, \ldots, v_\ell)$ ein einfacher Weg maximaler Länge in B. Dann ist $\ell \geqslant 0$, da $V \neq \emptyset$. Außerdem ist $\ell < |V|$, da W einfach ist (d.h. die in W vorkommenden Knoten $v_0, \ldots, v_\ell$ sind

paarweise verschieden — und in V gibt es nur $|V|$ viele verschiedene Knoten). Wir betrachten zwei Fälle:

Fall 1: $\ell = 0$.
Da B ein Baum ist (d.h. insbesondere zusammenhängend) und $W = (v_0)$ ein einfacher Weg maximaler Länge, muss dann $V = \{v_0\}$ und $E = \emptyset$ sein. Insbesondere ist v_0 ein Blatt von B.

Fall 2: $\ell \geqslant 1$.
Dann ist $v_{\ell-1}$ ein Nachbar von v_ℓ. Angenommen, v_ℓ ist kein Blatt. Dann hat v_ℓ einen weiteren Nachbarn, den wir im Folgenden u nennen.
Falls u nicht in W vorkommt, so ist $(v_0, \ldots, v_\ell, u)$ ein einfacher Weg, der länger ist als W. Dies widerspricht unserer Wahl von W als einfachem Weg maximaler Länge.
Falls u in W vorkommt, so gibt es ein i mit $u = v_i$ und $0 \leqslant i < \ell - 1$ (da u ein von den Knoten v_ℓ und $v_{\ell-1}$ verschiedener Knoten ist). Dann ist $(v_i, \ldots, v_{\ell-1}, v_\ell, v_i)$ ein einfacher Kreis in B. Dies widerspricht aber der Tatsache, dass B ein ungerichteter Baum ist.
Insgesamt kann es also keinen von $v_{\ell-1}$ verschiedenen Nachbarn u von v_ℓ geben. D.h. v_ℓ ist ein Blatt. □

Als Folgerung von Satz 5.63 können wir zeigen, dass die Anzahl der Kanten eines ungerichteten Baums um Eins kleiner ist als die Anzahl seiner Knoten.

Satz 5.64 (Anzahl der Kanten eines Baums).
Für jeden ungerichteten Baum $B = (V, E)$ mit $V \neq \emptyset$ gilt: $|E| = |V| - 1$.

Beweis: Per Induktion nach $n := |V|$.

Induktionsanfang: $n = 1$

Der einzige ungerichtete Baum $B = (V, E)$ mit $|V| = 1$ ist der Graph • mit $E = \emptyset$. Für diesen Graphen gilt: $|E| = 0 = 1 - 1 = |V| - 1$.

Induktionsschritt: $n \to n + 1$

Sei $n \in \mathbb{N}$ mit $n \geqslant 1$ beliebig.

Induktionsannahme:
Für jeden ungerichteten Baum $B' = (V', E')$ mit $V' \neq \emptyset$ und $|V'| \leqslant n$ gilt: $|E'| = |V'| - 1$.

Behauptung:
Für jeden ungerichteten Baum $B = (V, E)$ mit $|V| = n + 1$ gilt: $|E| = |V| - 1$.

Beweis: Sei $B = (V, E)$ ein ungerichteter Baum mit $|V| = n + 1$. Gemäß Satz 5.63 besitzt B (mindestens) ein Blatt, das wir im Folgenden u nennen. Da B zusammenhängend ist und $|V| = n + 1 \geqslant 2$ ist, besitzt u einen Nachbarn v in B — und da u ein Blatt ist, ist v der einzige Nachbar von u in B.

Sein nun B' der Graph, der aus B entsteht, indem wir den Knoten u und die von v zu u führende Kante löschen. D.h., $B' = (V', E')$ mit $V' := V \setminus \{u\}$ und $E' := E \setminus \{\{v, u\}\}$. Man sieht leicht, dass B' zusammenhängend ist und keinen einfachen Kreis enthält. D.h., B' ist ein Baum. Außerdem ist $|V'| = n$. Aus der Induktionsannahme folgt daher, dass $|E'| = |V'| - 1$ ist. Insgesamt gilt daher für den Baum B:

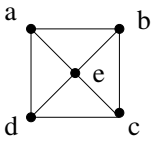
$$|E| = |E'| + 1 = (|V'| - 1) + 1 = |V'| = |V| - 1.$$

□

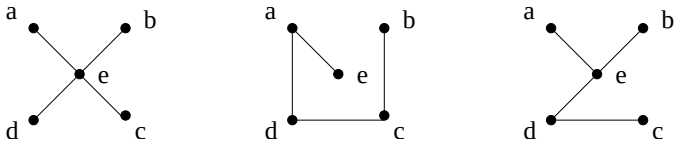
5.2.1. Spannbäume

Definition 5.65 (Spannbaum).
Sei $G = (V, E)$ ein ungerichteter Graph. Ein Graph $G' = (V', E')$ heißt **Spannbaum von** G, falls G' ein ungerichteter Baum mit $V' = V$ und $E' \subseteq E$ ist. Spannbaum

Beispiel 5.66. Der Graph



hat u.a. folgende Spannbäume:



Jeder zusammenhängende Graph besitzt einen Spannbaum. Genauer gilt:

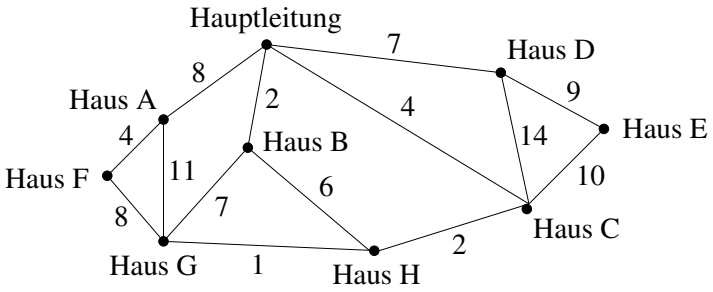
Satz 5.67. Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

$$\text{Es gibt (mindestens) einen Spannbaum von } G \iff G \text{ ist zusammenhängend.}$$

Beweis: „$\Longrightarrow$“: klar.
„$\Longleftarrow$“: Übung. □

Geht man von einem zusammenhängenden Graphen zu einem seiner Spannbäume über, so verkleinert man gemäß Satz 5.64 die Kantenmenge von $|E|$ auf $|V| - 1$ Kanten, ohne dabei den Zusammenhang des Graphen aufzugeben. Mit dem Begriff des Spannbaums wird also ein „bezüglich der Kantenzahl kostengünstigerer Zusammenhang“ modelliert.

Beispiel 5.68 (Kabelfernsehen). Eine Firma will Leitungen zum Empfang von Kabelfernsehen in einem neuen Wohngebiet verlegen. Der folgende Graph skizziert das Wohngebiet:



Knoten entsprechen dabei einzelnen Häusern bzw. der Hauptleitung, die aus einem bereits verkabelten Gebiet heranführt. Eine Kante zwischen zwei Knoten zeigt an, dass es prinzipiell möglich ist, eine direkte Leitung zwischen den beiden Häusern zu verlegen. Der Wert, mit dem die Kante markiert ist, beschreibt, wie teuer (in 1000 €) es ist, diese Leitung zu verlegen.

Ziel ist, Leitungen so zu verlegen, dass

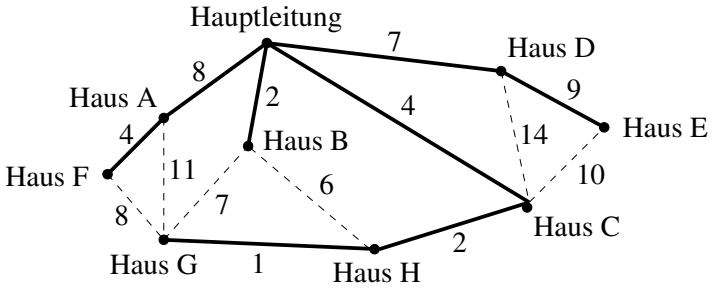
(1) jedes Haus ans Kabelfernsehen angeschlossen ist und

(2) die Kosten für das Verlegen der Leitungen so gering wie möglich sind.

minimaler Spannbaum

Es wird also ein Spannbaum gesucht, bei dem die Summe seiner Kantenmarkierungen so klein wie möglich ist. Ein solcher Spannbaum wird **minimaler Spannbaum** (engl.: **minimum spanning tree**) genannt.

Die im Folgenden **fett** gezeichneten Kanten geben die Kanten eines minimalen Spannbaums an:



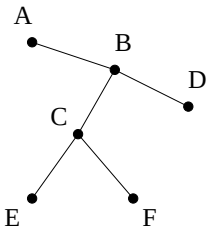
Verlegt die Firma genau diese Leitungen, so hat sie das neue Wohngebiet mit den geringstmöglichen Kosten ans Kabelfernsehen angeschlossen.

Bemerkung: Verfahren zum Finden minimaler Spannbäume werden Sie in der Vorlesung „Theoretische Informatik 1" kennenlernen.

5.3. Gewurzelte Bäume

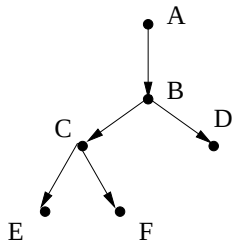
Einen **gewurzelten Baum** erhält man, indem man in einem ungerichteten Baum einen Knoten als „Wurzel" auswählt und alle Kanten in die Richtung orientiert, die von der Wurzel weg führt.

Beispiel 5.69. Ungerichteter Baum:



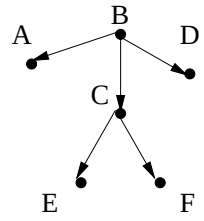
- Zugehöriger gewurzelter Baum mit Wurzel A:

$G_A :=$



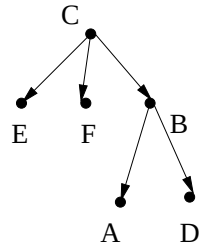
- Zugehöriger gewurzelter Baum mit Wurzel B:

$G_B :=$



- Zugehöriger gewurzelter Baum mit Wurzel C:

$G_C :=$



Wir definieren den Begriff „gewurzelter Baum“ wie folgt:

Definition 5.70 (gewurzelter Baum).
Ein gerichteter Graph $G = (V, E)$ heißt **gewurzelter Baum**, falls er folgende Eigenschaften hat: gewurzelter Baum

(1) G besitzt genau einen Knoten $w \in V$ mit Ein-Grad$_G(w) = 0$.
Dieser Knoten wird **Wurzel** genannt. Wurzel

(2) Für jeden Knoten $v \in V$ gilt: Es gibt in G einen Weg von der Wurzel zum Knoten v.

(3) Für jeden Knoten $v \in V$ gilt: Ein-Grad$_G(v) \leqslant 1$.

Definition 5.71 (Blätter, innere Knoten, Höhe).
Sei $B = (V, E)$ ein gewurzelter Baum.

(a) Knoten mit Aus-Grad 0 heißen **Blätter**. Knoten, die weder Wurzel noch Blätter sind, heißen **innere Knoten**. Blätter, innere Knoten

(b) Die **Höhe** eines Knotens v ist die Länge eines längsten Weges, der in v beginnt, die **Tiefe** von v ist die Länge eines längsten Weges, der in v endet. Höhe, Tiefe

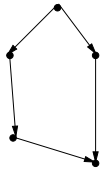
(c) Die Höhe, bzw. die Tiefe (engl.: height, depth) von B ist die Länge eines längsten Weges in B.

Beispiel: In Beispiel 5.69 hat G_A die Blätter D, E, F. G_B hat die Blätter A, D, E, F und G_C die Blätter A, D, E, F. Des weiteren hat G_A die Höhe 3, G_B die Höhe 2 und G_C die Höhe 2.

Beobachtung 5.72.

(a) Jeder gewurzelte Baum ist ein gerichteter azyklischer Graph (kurz: DAG, vgl. Definition 5.23). Aber es gibt gerichtete azyklische Graphen, die keine gewurzelten Bäume sind.

Beispiel:



ist ein DAG, aber kein gewurzelter Baum.

(b) Für jeden gewurzelten Baum $B = (V, E)$, dessen Knotenmenge nicht-leer ist, gilt:

$$|E| \;=\; |V| - 1.$$

Dies folgt unmittelbar aus Satz 5.64, da der ungerichtete Graph, der entsteht, indem man in B die Kantenorientierung „vergisst“ (d.h. jede gerichtete Kante (i, j) durch die ungerichtete Kante $\{i, j\}$ ersetzt), ein ungerichteter Baum ist.

Alternativ zu Definition 5.70 kann man die gewurzelten Bäume, deren Knotenmenge nicht-leer ist, auch folgendermaßen definieren:

Definition 5.73 (gewurzelte Bäume, rekursive Definition).
Die Klasse der gewurzelten Bäume mit nicht-leerer Knotenmenge ist rekursiv wie folgt definiert:

Basisregel: Ist V eine Menge mit $|V| = 1$, so ist $B := (V, \emptyset)$ ein gewurzelter Baum.

Skizze: $B :=$ ●

Der (eindeutig bestimmte) Knoten in V heißt **Wurzel** von B.
Die **Höhe** von B ist 0.

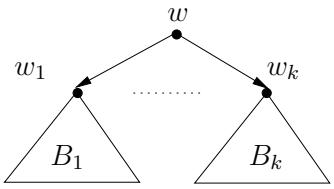
Rekursive Regel: Ist $k \in \mathbb{N}_{>0}$, sind $B_1 = (V_1, E_1), \ldots, B_k = (V_k, E_k)$ gewurzelte Bäume mit paarweise disjunkten Knotenmengen (d.h. $V_i \cap V_j = \emptyset$ f.a. $i, j \in \{1, \ldots, k\}$ mit $i \neq j$), sind $w_1 \in V_1, \ldots, w_k \in V_k$ die Wurzeln von $B_1, \ldots, B_k$, und ist w ein Element, das nicht in $V_1 \cup \cdots \cup V_k$ liegt, dann ist der Graph $B = (V, E)$ mit

$$V := \{w\} \cup V_1 \cup \cdots \cup V_k \qquad \text{und} \qquad E := E_1 \cup \cdots \cup E_k \cup \{(w, w_i) : i \in \{1, \ldots, k\}\}$$

ein gewurzelter Baum.

Skizze:

B :=

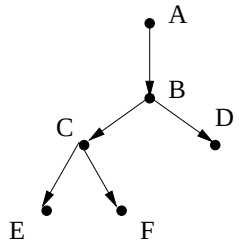


Der Knoten w heißt **Wurzel** von B.
Die **Höhe** von B ist $1 + \max\{h_1, \ldots, h_k\}$, wobei $h_1, \ldots, h_k \in \mathbb{N}$ die Höhen der gewurzelten Bäume $B_1, \ldots, B_k$ sind.

Notation 5.74 (Kinder eines Knotens).
Sei $B = (V, E)$ ein gewurzelter Baum und sei $v \in V$ ein beliebiger Knoten in B. Die Knoten $v' \in V$, zu denen von v aus eine Kante führt (d.h. $(v, v') \in E$), heißen **Kinder** von v. Kinder

Beispiel: Im Graphen

G_A =



aus Beispiel 5.69 gilt: Knoten A hat genau ein Kind, nämlich B; Knoten B hat genau zwei Kinder, nämlich C und D; Knoten C hat genau zwei Kinder, nämlich C und F; und die Knoten D, E, F haben keine Kinder.

Eine besondere Rolle bei der Modellierung spielen Bäume, bei denen jeder Knoten höchstens zwei Kinder hat. Mit solchen Bäumen kann man z.B. Binär-Codierung oder Kaskaden von JA-NEIN-Entscheidungen beschreiben.

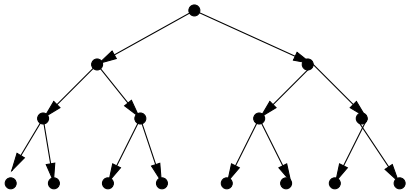
Definition 5.75 (Binärbaum, vollständiger Binärbaum).

(a) Ein gewurzelter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt: Aus-Grad$_B(v) \leqslant 2$. Binärbaum

(b) Ein Binärbaum $B = (V, E)$ heißt **voll**, falls jeder Knoten, der kein Blatt ist, den Aus-Grad 2 besitzt. voller Binärbaum

(c) Ein Binärbaum $B = (V, E)$ heißt **vollständiger Binärbaum**, falls gilt: vollständiger Binärbaum
   (1) Jeder Knoten, der kein Blatt ist, hat Aus-Grad 2.
   (2) Alle Blätter $v \in V$ haben dieselbe Tiefe.

Beispiel 5.76. Der Graph G_A aus Beispiel 5.69 ist ein Binärbaum, aber kein vollständiger Binärbaum. Der Graph G_B aus Beispiel 5.69 ist kein Binärbaum. Der folgende Graph B_3 ist ein **vollständiger Binärbaum** der Höhe 3:



Zwischen der Höhe, der Anzahl der Blätter und der Anzahl der Knoten eines Binärbaums besteht der folgende wichtige Zusammenhang:

Satz 5.77. Sei $h \in \mathbb{N}$.

(a) Jeder **vollständige Binärbaum der Höhe** h hat genau 2^h **Blätter** und genau $2^{h+1} - 1$ **Knoten**.

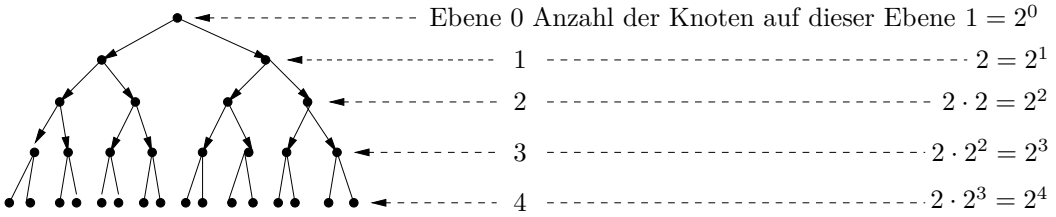
(b) Jeder **Binärbaum der Höhe** h hat **höchstens** 2^h **Blätter** und **höchstens** $2^{h+1} - 1$ **Knoten**.

(c) Für **volle Binärbäume** $B = (V, E)$ mit b Blättern gilt

$$b = \frac{|V| + 1}{2}.$$

Beweis:

(a) *Skizze:*



Anhand dieser Skizze sieht man leicht, dass ein vollständiger Binärbaum der Höhe h genau 2^h Blätter und

$$2^0 + 2^1 + 2^2 + \cdots + 2^h \overset{\text{Satz 4.11}}{=} 2^{h+1} - 1$$

Knoten besitzt.

Den formalen Beweis führen wir per Induktion nach h:

Induktionsanfang: $h = 0$:
Für jeden gewurzelten Baum $B = (V, E)$ der Höhe 0 gilt: $|V| = 1$ und $|E| = 0$. D.h. B besteht aus genau einem Knoten, der gleichzeitig Wurzel und (einziges) Blatt des Baums ist. D.h: B hat genau $1 = 2^0 = 2^h$ Blätter und genau $1 = 2 - 1 = 2^1 - 1 = 2^{h+1} - 1$ Knoten.

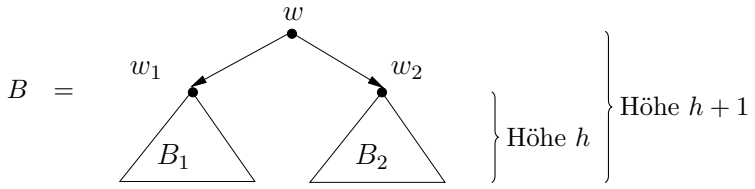
Induktionsschritt: $h \to h + 1$:
Sei $h \in \mathbb{N}$ beliebig.

Induktionsannahme: Jeder vollständige Binärbaum der Höhe h hat genau 2^h Blätter und genau $2^{h+1} - 1$ Knoten.

Behauptung: Jeder vollständige Binärbaum der Höhe $h+1$ hat genau 2^{h+1} Blätter und genau $2^{h+2}-1$ Knoten.

Beweis: Sei $B=(V,E)$ ein vollständiger Binärbaum der Höhe $h+1$, und sei $w \in V$ die Wurzel von B. Wegen $h+1 \geqslant 1$ hat w genau 2 Kinder. Seien $w_1 \in V$ und $w_2 \in V$ diese beiden Kinder von w. Für $i \in \{1,2\}$ sei V_i die Menge aller Knoten aus V, zu denen von w_i aus ein Weg führt; und sei $B_i := (V_i, E_i)$ der induzierte Teilgraph von B mit Knotenmenge V_i.

Skizze:



Offensichtlich ist sowohl B_1 als auch B_2 ein vollständiger Binärbaum der Höhe h. Gemäß Induktionsannahme hat jeder der beiden Bäume B_1 und B_2 genau 2^h Blätter und genau $2^{h+1}-1$ Knoten.

Der Baum B hat daher genau $2^h+2^h = 2^{h+1}$ Blätter und genau $1+(2^{h+1}-1)+(2^{h+1}-1) = 2 \cdot 2^{h+1} - 1 = 2^{h+2}-1$ Knoten.

(b) Analog. Details: Übung.

(c) Details: Übung.

□

Frage: Gibt es in jedem vollen Binärbaum stets kantendisjunkte Wege von allen Nicht-Blättern zu Blättern?

Gewurzelte Bäume werden eingesetzt, um

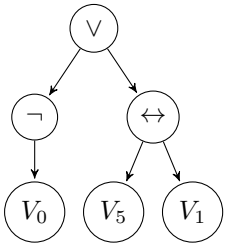
- Entwicklungen zu verfolgen:
  - Stammbäume (in der Genealogie) geben die Familiengeschichte wieder,
  - phylogenetische Bäume (in der Bioinformatik) zeigen evolutionäre Beziehungen zwischen verschiedenen Arten auf.
- hierarchische Strukturen zu repräsentieren wie etwa
  - eine Verzeichnisstruktur oder die
  - Organisationsstrukur einer Firma.
- Datenstrukturen zu entwickeln wie etwa
  - binäre Suchbäume, AVL-Bäume oder B-Bäume (siehe die gleichnamige Veranstaltung)
- rekursive Definitionen zu veranschaulichen, bzw. zu verstehen. Beispiele sind rekursiv definierte Mengen oder Funktionen wie etwa
  - (aussagenlogische) Formeln, arithmetische Ausdrücke, XML-Dokumente oder die Syntax einer Programmiersprache,
  - rekursive Programme.

5.3.1. Syntaxbäume

Wir haben aussagenlogische Formeln durch eine rekursive Definition über den Aufbau der Formel hergeleitet. Der Syntaxbaum (siehe Bemerkung 3.2) bildet die Rekursion nach: Die inneren Knoten wie auch die Wurzel sind jeweils mit einem Junktor markiert, die Blätter sind mit Variablen oder den Konstanten **0**, bzw **1** markiert. Für aussagenlogische Formeln ist der Syntaxbaum stets binär, weil wir nur ein- oder zweistellige Junktoren betrachten.

Syntaxbaum

Die Bedeutung der *aussagenlogischen Formel* $\big(\neg V_0 \vee (V_5 \leftrightarrow V_1)\big)$ wird sofort klar, wenn wir ihren **Syntaxbaum** betrachten:



Auf ähnliche Art werden markierte Bäume genutzt, um die Struktur vieler anderer Objekte (an Stelle von aussagenlogischen Formeln) zu beschreiben: Z.B. für arithmetische Terme, zur Darstellung von Klassen- und Objekthierarchien, zur Beschreibung der Struktur von Computerprogrammen oder umgangssprachlichen Texten.

Im Kapitel 8 und insbesondere in Abschnitt 8.2.2 beschäftigen wir uns intensiv mit Ableitungsbäumen, die als Ergebnis der Compilierung eines Programms erstellt werden. Genauso wie ein Syntaxbaum einer aussagenlogischen Formel die Semantik der Formel beschreibt, drückt ein Ableitungsbaum die Bedeutung des Programms aus.

5.3.2. Rekursive Programme und ihre Rekursionsbäume

Wir untersuchen ein rekursives Programm $R(\vec{p})$ (mit den Parametern $\vec{p}$). Während der Abarbeitung von $R(\vec{p})$ werden rekursive Aufrufe getätigt, die ihrerseits weitere rekursive Aufrufe starten und so weiter. Der Rekursionsbaum Baum$(\vec{p})$ modelliert die Struktur all dieser rekursiven Aufrufe. Baum$(\vec{p})$ wird nach den folgenden Regeln gebaut.

1. Beschrifte die Wurzel von Baum$(\vec{p})$ mit den Parametern $\vec{p}$ des Erstaufrufs.
   - Wenn innerhalb von $R(\vec{p})$ keine rekursiven Aufrufe getätigt werden, dann wird die Wurzel zu einem Blatt.
   - Ansonsten erhält die Wurzel für jeden rekursiven Aufruf in $R(\vec{p})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.
2. Wenn ein Knoten v von Baum$(\vec{p})$ mit den Parametern $\vec{q}$ beschriftet ist, gehen wir mit v genauso wie mit der Wurzel vor.
   - Wenn innerhalb von $R(\vec{q})$ keine rekursiven Aufrufe getätigt werden, wird v zu einem Blatt.
   - Ansonsten erhält v für jeden rekursiven Aufruf in $R(\vec{q})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.

Baum$(\vec{p})$ veranschaulicht die Arbeitsweise von $R(\vec{p})$. So stimmt etwa die Anzahl der Knoten von Baum$(\vec{p})$ überein mit der Anzahl aller rekursiven Aufrufe, die irgendwann getätigt werden bis $R(\vec{p})$ terminiert.

Beispiel 5.78 (Binärsuche). Wir betrachten die Binärsuche (siehe Beispiel 4.27). Wie sieht der Rekursionsbaum Baum(y, A) für ein Array A und einen Schlüssel y aus, der nicht in A vorkommt?

Wir nehmen wie in Beispiel 4.27 an, dass das Array A genau $n = 2^k - 1$ Schlüssel für eine Zahl $k \in \mathbb{N}$ besitzt. Beachte, dass ein Knoten von Baum(y, A) entweder ein Blatt ist oder `genau einen` rekursiven Aufruf verursacht. Also ist Baum(y, A) ein Weg.

Im Beispiel 4.27 haben wir gezeigt, dass Binärsuche im schlimmsten Fall k Zellen inspiziert. Der Rekursionsbaum B ist also ein Weg der Länge $k - 1$.

Binärsuche ist schnell, weil stets höchstens ein rekursiver Aufruf getätigt wird und weil die Länge des Array-Abschnitts in dem gesucht wird, mindestens halbiert wird.

Beispiel 5.79 (Türme von Hanoi). Wie sieht der Rekursionsbaum Baum$(N, 1, 2, 3)$ für das Programm Hanoi(N,1,2,3) aus, das wir in Beispiel 4.25 eingeführt haben?

Es werden zwei rekursive Aufrufe mit dem Parameter $N-1$ getätigt. Mit vollständiger Induktion über N zeigt man, das Baum$(N, 1, 2, 3)$ ein vollständiger Binärbaum der Höhe $N - 1$ ist: Trifft diese Aussage für die beiden rekursiven Aufrufe (mit entsprechenden Permutationen) zu, dann gilt sie auch für den „Master-Aufruf“.

Aber das sind ganz schlechte Nachrichten, denn wir wissen bereits, dass ein solcher Baum $2^{\text{Höhe}+1} - 1 = 2^N - 1$ Knoten besitzt. Da die Anzahl der Ringbewegungen mit der Anzahl der Knoten des Rekursionsbaums übereinstimmt, haben wir unsere Rechnung aus Beispiel 4.26 bestätigt.

Die Anzahl der Ringbewegungen ist so groß, weil für $N > 1$ stets zwei rekursive Aufrufe getätigt werden, wobei der Parameter N nur um Eins reduziert wird.

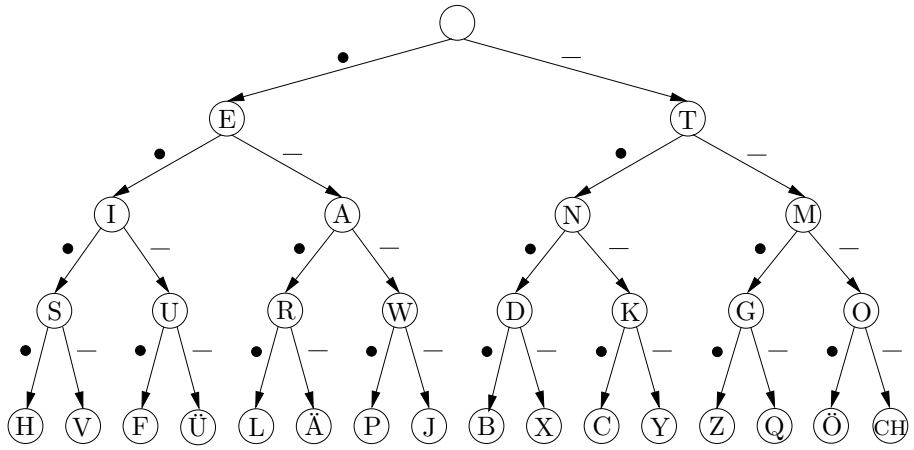
5.3.3. Entscheidungsbäume

Folgen von Entscheidungen können in vielen Zusammenhängen durch gewurzelte markierte Bäume modelliert werden. Solche Bäume heißen **Entscheidungsbäume**. Entscheidungsbaum

Beispiel 5.80 (Der Morse-Code). Durch einen solchen Entscheidungsbaum erhält man beispielsweise eine kompakte Darstellung des **Morse-Codes**, wenn man Sonder- und Gesprächszeichen vernachlässigt.

Im Morse-Code wird jeder Buchstabe durch eine Folge von kurzen und langen Signalen repräsentiert. Ein „kurzes Signal“ wird im folgenden Baum als Kantenmarkierung „•“ dargestellt; ein „langes Signal“ wird als „—“ dargestellt. Insgesamt wird der Morsecode durch folgenden Entscheidungsbaum repräsentiert:

M =

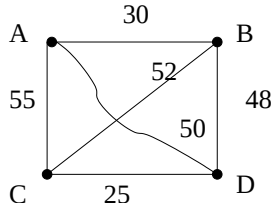


Eine eingehende Meldung aus kurzen und langen Signalen wird entschlüsselt, indem man an der Wurzel des Baums M beginnt und bei einem kurzen Signal nach links, bei einem langen nach rechts weitergeht. Eine längere Pause zeigt an, dass ein Buchstabe vollständig übermittelt ist.

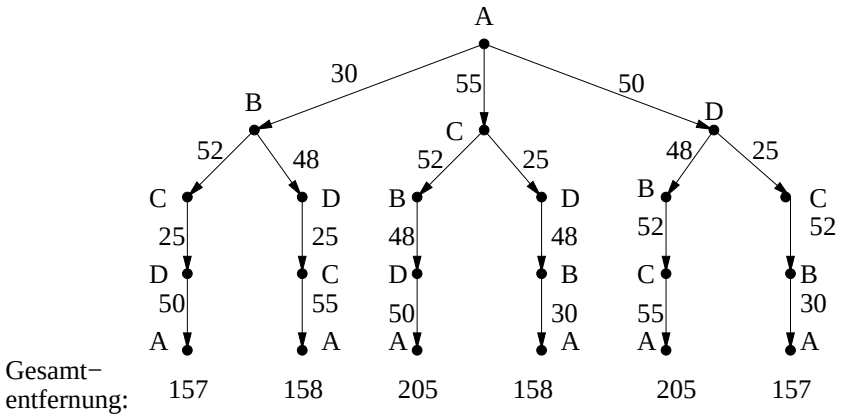
In jedem Entscheidungsbaum modellieren die Knoten einen Zwischenstand bei der Entscheidungsfindung. Sie können entsprechend markiert sein, z.B. mit dem kodierten Buchstaben des Morse-Codes. Die Kanten, die von einem Knoten ausgehen, modellieren die Alternativen, aus denen eine ausgewählt werden kann. Beim Morse-Code ist das jeweils ein kurzes oder ein langes Signal, das als Kantenmarkierung angegeben wird.

5.3.3.1. Das Problem des Handlungsreisenden

Markierte Bäume können auch genutzt werden, um den Lösungsraum kombinatorischer Probleme darzustellen. Als Beispiel betrachten wir einen Handlungsreisenden, der eine möglichst kurze Rundreise finden soll, auf dem er jede der Städte A, B, C, D besucht. Die Entfernungen (in km) zwischen den Städten sind als Kantenmarkierungen des folgenden Graphen gegeben:



Der folgende Baum repräsentiert alle möglichen in Stadt A startenden Rundreisen:



Jeder Weg von der Wurzel zu einem Blatt repräsentiert dabei eine Rundreise, auf dem jede Stadt genau einmal besucht wird. Die Kantenmarkierungen geben die Entfernungen zwischen einzelnen Städten wieder. Eine zusätzliche Knotenmarkierung an jedem Blatt gibt die Gesamtlänge des entsprechenden Rundwegs an. Die beiden kürzesten Rundreisen für unseren Handlungsreisenden sind also

$$(A, B, C, D, A) \quad \text{und} \quad (A, D, C, B, A).$$

5.3.3.2. Spielbäume

In einem Zwei-Personen-Spiel (wie zum Beispiel Schach, Dame, Mühle, Go, Othello) spielen Alice und Bob gegeneinander. Können wir in einer vorgegebenen Spielsituation einen gewinnenden Zug für den ziehenden Spieler bestimmen?

Wir bauen einen **Spielbaum**. Spielbaum

(a) Ein Knoten (S, X) im Spielbaum modelliert eine Spielsituation S und einen ziehenden Spieler $X \in \{\,\text{Alice}, \text{Bob}\,\}$. Von (S, X) aus wird für jeden möglichen Zug z des Spielers X eine mit z beschriftete Kante angelegt.

(b) Ein Blatt wird mit 1, bzw. -1 beschriftet, falls Alice das Spiel gewinnt oder verliert. (Wir nehmen an, dass es kein Unentschieden gibt.)

Der Minimax-Algorithmus bestimmt für jeden Knoten, welcher der Spieler eine Gewinnstrategie hat. Der Algorithmus beginnt mit den Blättern und arbeitet sich dann „aufwärts“ zur Wurzel.

1. Für einen inneren Knoten u wird der Spielwert

$$\text{wert}(u) = \begin{cases} 1 & \text{Alice hat eine Gewinnstrategie,} \\ -1 & \text{Bob hat eine Gewinnstrategie.} \end{cases}$$

   definiert.

2. Angenommen, Alice ist in v am Zug und „wert(w)“ ist für jedes Kind w von v bekannt. Dann setzt der Algorithmus

$$\text{wert}(v) = \max_{w \text{ ist ein Kind von } v} \text{wert}(w),$$

   denn eine intelligent spielende Alice wird den besten möglichen Zug auswählen.

3. Ist hingegen Bob in v am Zug, dann wird

$$\text{wert}(v) = \min_{w \text{ ist ein Kind von } v} \text{wert}(w),$$

gesetzt: ein intelligent spielender Bob wird seinerseits den besten möglichen Zug auswählen.

Der Spielbaum für nicht-triviale Spiele ist viel zu groß und kann durch Rechner auch nicht annähernd berechnet werden. Stattdessen beschränken sich clevere Strategien auf das Durchspielen „guter Züge“. Heuristiken werden eingesetzt, um zwischenzeitlich erhaltene Spielsituationen zu bewerten und „gute Züge“ zu bestimmen. Diese Heuristiken werden für Schach ergänzt um Eröffnungsbibliotheken, Endspiel-Datenbanken und Datenbanken vollständig gespielter Partien.

Mittlerweile sind Schachprogramme so ausgereift, dass selbst Schachgroßmeister große Probleme haben, mitzuhalten: Die Spielstärke eines Schachspielers wird mit der Elo-Zahl gemessen, die besten 10 Schachprogramme erzielen Elo-Zahlen von über 3350, während der Schachweltmeister eine Elo-Zahl von ca. 2850 „schafft“.

Das Othello-Programm Logistello hat in 1997 alle sechs Spiele gegen den damaligen Weltmeister gewonnen. Mittlerweile besteht Übereinstimmung, dass Othello-Programme dem menschlichen Spieler weit überlegen sind.

5.4. Zusammenfassung und Ausblick

Graphen können eingesetzt werden zur Modellierung von

(a) Straßenkarten,

(b) der städtischen S- und U-Bahn Netze oder des Schienennetzes der deutschen Bahn,

(c) von Computer-Netzen, wenn Computer durch Knoten und Netzwerkverbindungen durch ungerichtete Kanten repräsentiert werden,

(d) Strom-Netzen, wenn Knoten Erzeugern, Transformatoren oder Verbrauchern entsprechen und Kanten Direktverbindungen repräsentieren,

(e) Hyperlinks im World-Wide-Web.

Natürlich müssen wir mit den Modellen arbeiten können und wir haben deshalb eine Reihe fundamentaler Probleme für ungerichtete und gerichtete Graphen behandelt. Einige dieser Probleme stellen sich als einfach oder zumindest beherrschbar heraus, wie etwa

(a) die Lösung von Labyrinth-Problemen (mit Hilfe der Tiefensuche),

(b) die Bestimmung kürzester Wege (einem zentralen Problem für Navis),

(c) die Berechnung von Euler-Wegen und Euler-Kreisen,

(d) die Berechnung eines größtmöglichen Matchings

(e) oder die Bestimmung minimaler Spannbäume.

In den Veranstaltungen „Datenstrukturen“ und „Theoretische Informatik 1“ werden effiziente Algorithmen für viele dieser Probleme beschrieben. Wir haben aber auch schon schnelle Lösungen entworfen: Mit der Tiefensuche haben wir einen wichtigen Algorithmus analysiert, die vielen Anwendungen der Tiefensuche – wie etwa die Frage, ob ein gerichteter Graph azyklisch ist –

werden in der „Theoretischen Informatik 1“ untersucht. Der Beweis von Satz 5.39 enthält die wichtigsten Ideen für effiziente Algorithmen, die einen Euler-Kreis oder einen Euler-Weg, falls vorhanden, bestimmen.

Von ganz anderem Kaliber sind die folgenden Probleme: Bestimme

(a) einen Hamilton-Kreis oder einen Hamilton-Weg,

(b) eine kürzeste Rundreise im Problem des Handlungsreisenden,

(c) eine Färbung mit möglichst wenigen Farben,

(d) eine größte unabhängige Menge, eine größte Clique oder eine kleinste Knotenüberdeckung,

(e) ein leichtestes Feedback Vertex Set,

(f) einen gewinnenden Zugs in einem „nicht-trivialen“ Zwei-Personen Spiel, vorausgesetzt natürlich ein solcher Zug existiert.

Bis auf das letzte Problem sind alle Probleme NP-vollständig und sind somit genauso schwierig wie das KNF-Erfüllbarkeitsproblem. Das letzte Problem ist sogar noch schwieriger und führt auf ein PSPACE-vollständiges Problem: PSPACE und die Klasse der PSPACE vollständigen Probleme werden in der Vorlesung „Theoretische Informatik 2“ behandelt (siehe auch Abschnitt 8.4.2).

Wie können so anscheinend völlig verschiedenartige Probleme wie KNF-SAT und die Frage, ob ein gegebener Graph einen Hamilton-Kreis besitzt, gleich schwierig sein? Was heißt das überhaupt „gleich schwierig zu sein“? Warum ist die Bestimmung von Hamilton-Kreisen schwierig, die Bestimmung von Euler-Kreisen hingegen einfach? Hoffentlich ist Ihre Neugier geweckt!

5.5. Literaturhinweise zu Kapitel 5

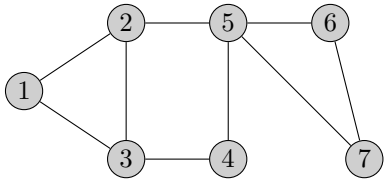
Als vertiefende Lektüre sei Kapitel 5 in [14], Kapitel 11 in [21], Teile der Kapitel 0–4 und 8 in [4], sowie Teile der Kapitel 7–10 und 13 in [18] empfohlen. Eine umfassende Einführung in die Graphentheorie gibt das Lehrbuch [4].

Quellennachweis: Viele der in diesem Kapitel angegebenen Modellierungsbeispiele sowie die folgenden Aufgaben **??**, 5.8, 5.9 und 5.34 sind dem Buch [14] entnommen.

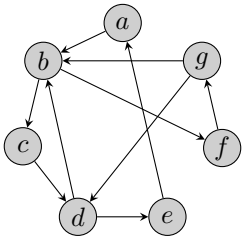
5.6. Übungsaufgaben zu Kapitel 5

Aufgabe 5.1. Es seien die folgenden beiden Graphen G_1 und G_2 gegeben:

G_1:



G_2:

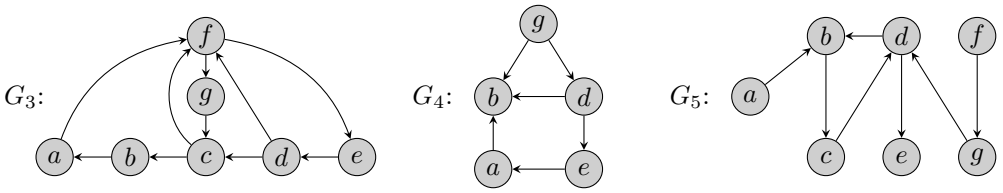


(a) Geben Sie für jeden der beiden Graphen G_1 und G_2 die Knotenmenge und die Kantenmenge an. Repräsentieren Sie außerdem jeden der beiden Graphen durch eine Adjazenzmatrix und eine Adjazenzliste.

(b) Geben Sie einen Weg von 2 nach 4 in G_1 an, der *nicht* einfach ist. Geben Sie außerdem einen Kreis in G_1 an, der *nicht* einfach ist und durch den Knoten 2 verläuft.

(c) Ist G_1 zusammenhängend? Ist G_2 stark zusammenhängend? Ist G_2 azyklisch?

(d) Überprüfen Sie für jeden der folgenden Graphen G, ob Folgendes gilt: (i) $G = G_2$, (ii) G ist ein Teilgraph von G_2, (iii) G ist ein induzierter Teilgraph von G_2, (iv) G ist isomorph zu G_2. Geben Sie bei (d) auch einen Isomorphismus von G nach G_2 an, falls dieser existiert.



Aufgabe 5.2.

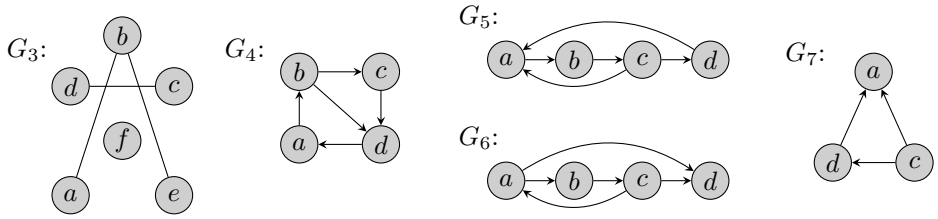
(a) Geben Sie die folgenden Graphen G_1 und G_2 in graphischer Darstellung an.

(i) $G_1 = \big(\{1,2,3,4,5,6\},\ \{\{x,y\} : x = 2 \cdot y\}\big)$

(ii) $G_2 = \big(\{x \in \mathbb{N}_{>0} : 1 \leqslant x \leqslant 6\},\ \{(x,y) : x = y+1\} \cup \{(x,y) : y = 3 \cdot x\}\big)$

Sind die Graphen G_1 und G_2 zusammenhängend bzw. stark zusammenhängend? Sind sie azyklisch?

(b) Seien G_3, G_4, G_5, G_6 und G_7 die folgenden Graphen:



(i) Gelten die folgenden Aussagen ?

(I) $G_3 \cong G_1$ (II) $G_4 \cong G_5$ (III) $G_5 \cong G_6$ (IV) $G_6 \cong G_7$

(V) G_7 ist ein induzierter Teilgraph von G_5.

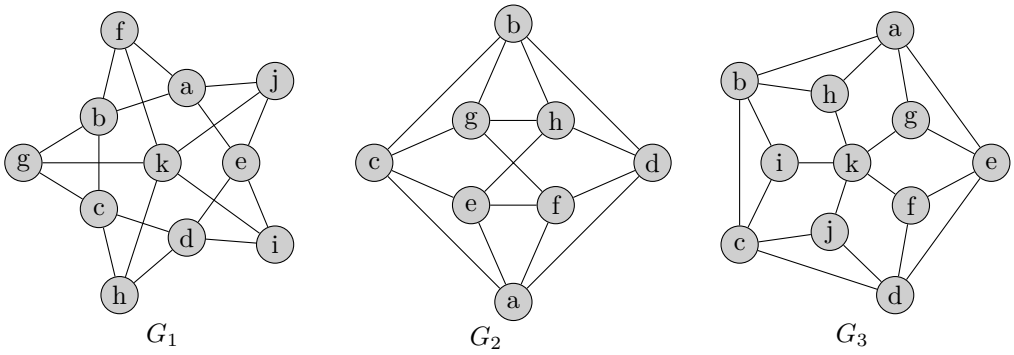
(VI) G_7 ist ein induzierter Teilgraph von G_6

Geben Sie für (I) bis (IV) jeweils einen Isomorphismus an, falls ein solcher existiert.

(ii) Geben Sie in G_3 und G_5 jeweils einen nicht einfachen Weg an, der kein Kreis ist.

(iii) Geben Sie in G_3 und G_5 jeweils einen nicht einfachen Kreis und einen einfachen Kreis an, falls ein solcher existiert.

Aufgabe 5.3. Es seien die folgenden drei ungerichteten Graphen G_1, G_2 und G_3 gegeben.



(a) Geben Sie für G_1, G_2 und G_3 jeweils einen Knoten maximalen Grades und einen Knoten minimalen Grades an.

(b) Geben Sie für G_1, G_2 und G_3 jeweils ein Matching maximaler Größe an.

(c) Enthalten die Graphen G_1, G_2 und G_3 jeweils einen Euler-Kreis?

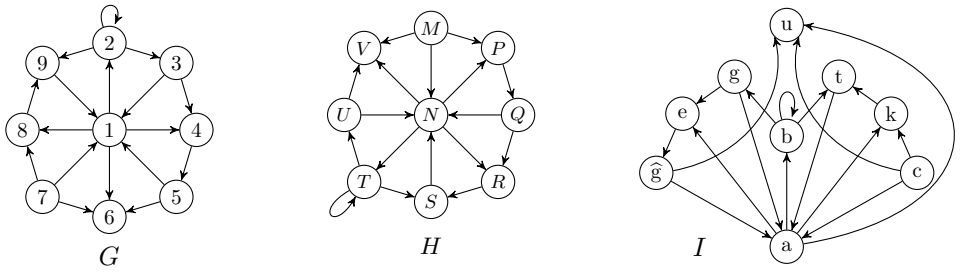
(d) Enthalten die Graphen G_1, G_2 und G_3 jeweils einen Hamilton-Kreis?

(e) Geben Sie für G_1, G_2 und G_3 jeweils eine Knotenfärbung mit möglichst wenigen Farben an. (Sie brauchen nicht zu begründen, warum die von Ihnen verwendete Anzahl von Farben jeweils minimal ist.)

(f) Gilt $G_1 \cong G_3$, gilt $G_2 \cong G_3$?

(g) Welche der Graphen G_1, G_2 und G_3 sind planar?

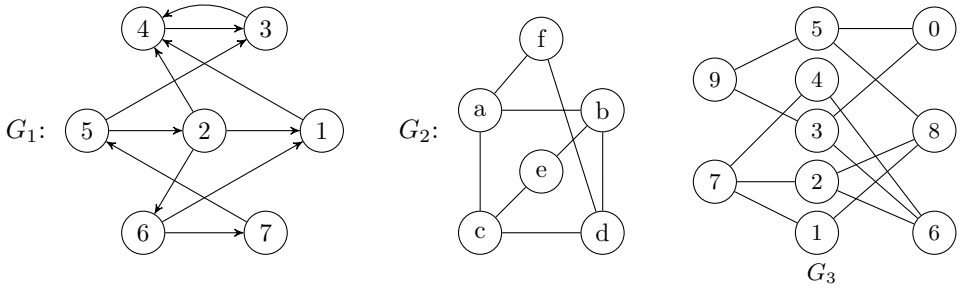
Aufgabe 5.4. Die Graphen G, H und I seien wie folgt in grafischer Darstellung gegeben.



(a) Zeigen Sie: G und H sind nicht isomorph.

(b) Geben Sie einen Isomorphismus von G nach I an.

Aufgabe 5.5. Die Graphen G_1, G_2 und G_3 seien wie folgt in grafischer Darstellung gegeben.

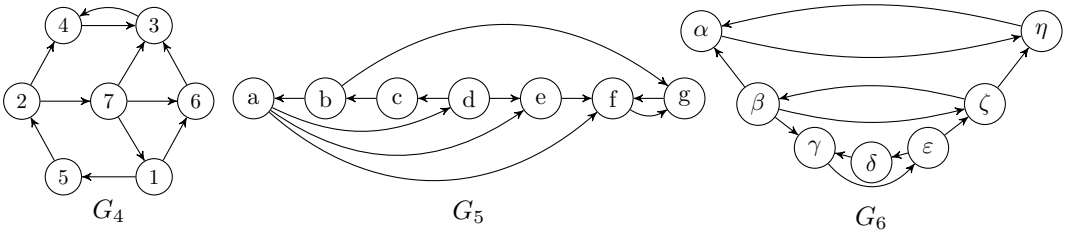


(a) Geben Sie alle starken Zusammenhangskomponenten von G_1 an.

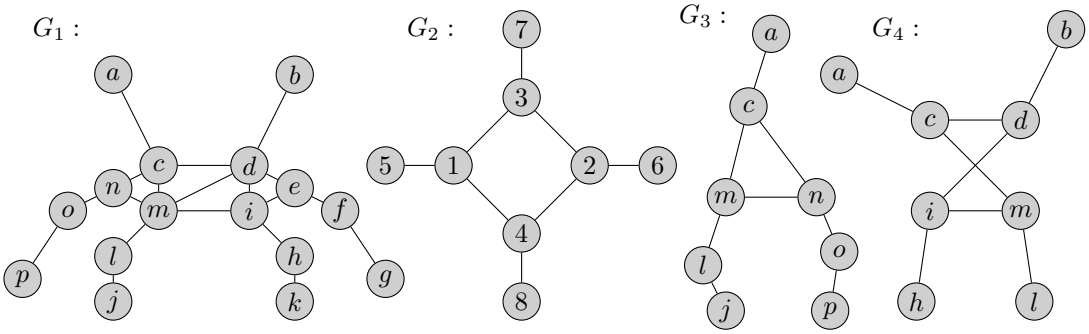
(b) Sind G_2 bzw. G_3 planar?

(c) Besitzen G_2 bzw. G_3 ein perfektes Matching?

(d) Welche der folgenden Graphen G_4, G_5, G_6 sind isomorph zu G_1? Geben Sie jeweils einen Isomorphismus an oder beweisen Sie, dass kein Isomorphismus existiert.



Aufgabe 5.6. Betrachten Sie die folgenden ungerichteten Graphen:



Für welche Zahlen $i, j \in \{1, \ldots, 4\}$ gelten die folgenden Aussagen?

(a) $G_i \cong G_j$.

(b) G_i ist ein Teilgraph von G_j.

(c) G_i ist ein induzierter Teilgraph von G_j.

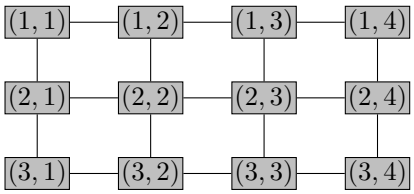
(d) Es existiert ein Teilgraph G'_j von G_j, der isomorph zu G_i ist.

Geben Sie für (a) jeweils einen Isomorphismus von G_i nach G_j an, falls ein solcher existiert. Geben Sie für (d) jeweils einen Teilgraphen G'_j von G_j und einen Isomorphismus von G_i nach G'_j an, wenn diese existieren.

Aufgabe 5.7. Für $m, n \in \mathbb{N}_{>0}$ sei das $m \times n$-*Gitter* der Graph $G_{m\times n} = (V_{m\times n}, E_{m\times n})$ mit

$$\begin{aligned} V_{m\times n} &:= \big\{\, (i,j) \,:\, 1 \leqslant i \leqslant m,\, 1 \leqslant j \leqslant n \,\big\}, \\ E_{m\times n} &:= \big\{\, \{(i,j),(i,j+1)\} \,:\, 1 \leqslant i \leqslant m,\, 1 \leqslant j < n \,\big\} \cup \\ &\quad\ \big\{\, \{(i,j),(i+1,j)\} \,:\, 1 \leqslant i < m,\, 1 \leqslant j \leqslant n \,\big\}. \end{aligned}$$

Das 3×4-Gitter $G_{3\times 4}$ sieht z.B. wie folgt aus:



(a) Überprüfen Sie, ob $G_{3\times 4}$ bipartit ist. Falls $G_{3\times 4}$ bipartit ist, so geben Sie zwei disjunkte Knotenmengen $V_1, V_2 \subseteq V_{3\times 4}$ mit $V_1 \cup V_2 = V_{3\times 4}$ an, so dass jede Kante aus $E_{3\times 4}$ einen Knoten aus V_1 und einen Knoten aus V_2 miteinander verbindet. Falls $G_{3\times 4}$ nicht bipartit ist, so begründen Sie dies.

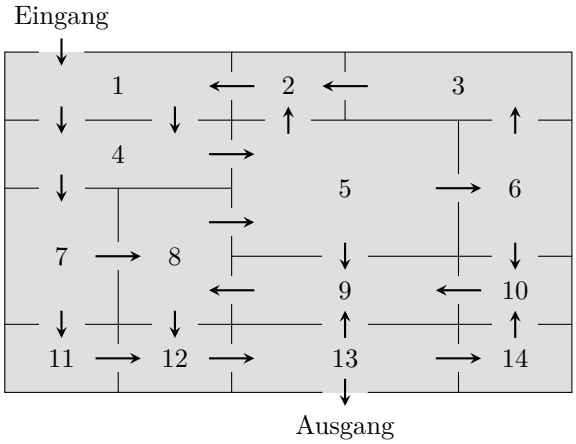
(b) Geben Sie ein Matching maximaler Größe in $G_{3\times 4}$ an.

(c) Geben Sie einen Hamilton-Kreis in $G_{3\times 4}$ an.

(d) Für welche $m, n \in \mathbb{N}_{>0}$ besitzt $G_{m\times n}$ einen Hamilton-Kreis, für welche nicht?

Hinweis: Stellen Sie sich vor, dass die Knoten des Gitters so mit den Farben rot oder blau eingefärbt sind, dass benachbarte Knoten unterschiedliche Farben besitzen.

Aufgabe 5.8. Die folgende Abbildung stellt den Grundriss eines Irrgartens dar.



Die Türen in diesem Irrgarten schwingen nur zu einer Seite auf und haben keine Klinken o.ä. Nachdem also ein Besucher die Eingangstür oder eine nachfolgende Tür durchschritten hat und die Tür hinter ihm zugefallen ist, kann der Besucher nicht mehr durch diese Tür zurück. Die Tür bleibt aber für weitere Durchgänge in der ursprünglichen Richtung benutzbar. Die allgemeinen Sicherheitsbestimmungen für Irrgärten schreiben vor, dass jeder Besucher, der den Irrgarten betritt, – egal wie er läuft – den Ausgang erreichen kann.

(a) Modellieren Sie den Irrgarten durch einen Graphen.

(b) Formulieren Sie die allgemeinen Sicherheitsbestimmungen für Irrgärten mit Begriffen der Graphentheorie.

(c) Überprüfen Sie anhand der Formulierungen aus (b), ob der angegebene Irrgarten den allgemeinen Sicherheitsbestimmungen entspricht.

Aufgabe 5.9. Sie bekommen die Aufgabe, $n \in \mathbb{N}_{>0}$ Rechner zu vernetzen. Ihr Auftraggeber verlangt folgende Eigenschaften des Netzwerkes:

(1) Von jedem Rechner muss jeder andere Rechner über einen Leitungsweg erreichbar sein.

(2) Auch wenn genau eine Leitung zwischen zwei Rechnern ausfällt, muss jeder Rechner über einen Leitungsweg mit jedem anderen Rechner verbunden sein.

(3) An jedem Rechner können maximal vier Leitungen angeschlossen werden.

Dabei können auf einer Leitung Daten in beide Richtungen gesendet werden. Ein solches Netzwerk lässt sich leicht als ungerichteter Graph darstellen: ein Knoten repräsentiert einen Rechner, und eine Kante repräsentiert eine Leitung.

(a) Formulieren Sie die Eigenschaften (1), (2) und (3) mit Begriffen der Graphentheorie.

(b) Untersuchen Sie die folgenden Graphen G_1, G_2 und G_3 auf Ihre Tauglichkeit bezüglich der Eigenschaften (1), (2) bzw. (3):

- $G_1 = (V_1, E_1)$ mit $V_1 = \{1, 2, \ldots, n\}$ und $E_1 = \{\{1, i\} : 2 \leqslant i \leqslant n\}$
- $G_2 = (V_2, E_2)$ mit $V_2 = V_1$ und $E_2 = \{\{i, i+1\} : 1 \leqslant i < n\}$
- $G_3 = (V_3, E_3)$ mit $V_3 = V_1$ und $E_3 = E_2 \cup \{\{n, 1\}\}$

Aufgabe 5.10. Poseidon möchte seinen Computer mit einer Wasserkühlung (Wakü) ausstatten. Eine Wakü besteht aus Gummischläuchen und zwei Arten von Steckverbindungen. Es gibt *einfache Verbindungsstücke* mit Anschlüssen für zwei Gummischlauch-Enden und *T-Stücke* mit Anschlüssen für drei Gummischlauch-Enden. Um die Elektronik nicht zu beschädigen, muss die Wakü *dicht* sein. Damit kein Wasser entweicht, muss jeder verwendete Schlauch mit beiden Enden an Steckverbindungen angeschlossen und jeder Anschluss einer Steckverbindung mit einem Schlauch versehen sein.

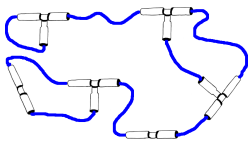
Definition: Ein Multigraph (G, m) (mit Eigenschleifen) besteht aus einem Tupel $G = (V, E)$ und einer Kantenmarkierung $m : E \to \mathbb{N}_{>0}$, wobei $V \neq \emptyset$ eine nicht-leere, endliche Menge von Knoten und $E \subseteq \{\{u, v\} : u, v \in V\}$ die Kantenmenge ist. Kanten der Form $\{u, u\}$ sind erlaubt und werden als *Eigenschleifen* bezeichnet. Eine Kante $e \in E$ mit $m(e) > 1$ heißt *Multikante* und kann als $m(e)$-fache Kopie einer gewöhnlichen Kante aufgefasst werden.

Sei $E^{(1)} \subseteq E$ die Menge der Eigenschleifen und $E^{(2)} \subseteq E$ die Menge der zwei-elementigen Kanten. Der Grad eines Knotens $v \in V$ ist durch

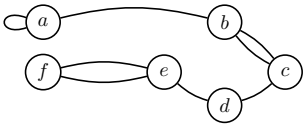
$$\mathrm{Grad}_{(G,m)}(v) := \begin{cases} 2 \cdot m(\{v, v\}) + \sum\limits_{u \in V, \{u,v\} \in E^{(2)}} m(\{u, v\}) & \text{falls } \{v, v\} \in E^{(1)}, \\ \sum\limits_{u \in V, \{u,v\} \in E^{(2)}} m(\{u, v\}) & \text{sonst} \end{cases}$$

definiert, d. h. jede Multikante $e = \{u, v\}$ zwischen verschiedenen Knoten u, v wird $m(e)$-mal gezählt und Eigenschleifen tragen doppelt zum Grad bei.[3]

Eine Wakü kann als Multigraph aufgefasst werden: Jedes einfache Verbindungsstück entspricht einem Knoten vom Grad 2. Jedes T-Stück entspricht einem Knoten vom Grad 3. Jeder verwendete Schlauch entspricht einer Kante.



(a) eine dichte Wakü



(b) und ihr Multigraph mit $m(\{a, a\}) = 1, m(\{b, c\}) = 2$ usw. Beispielsweise gilt $\text{Grad}_{(G,m)}(a) = 3$ und $\text{Grad}_{(G,m)}(e) = 3$.

(a) (i) Kann Poseidon aus zwei einfachen Verbindungsstücken eine dichte Wakü bauen? Ist dies auch mit zwei T-Stücken möglich?

(ii) In jedem ungerichteten Graphen $G = (V, E)$ gilt die Gleichung (I); in jedem Multigraphen (G', m) mit $G' = (V', E')$ gilt die Gleichung (II):

$$\text{(I)} \quad \sum_{v \in V} \text{Grad}_G(v) = 2 \cdot |E| \qquad \text{(II)} \quad \sum_{v \in V'} \text{Grad}_{(G',m)}(v) = 2 \cdot \sum_{e \in E'} m(e)$$

Zeigen Sie die Gleichung (I). Eine informelle Begründung ist hier ausreichend.

(b) (i) Sei $n \in \mathbb{N}_{>0}$ beliebig. Kann Poseidon aus n einfachen Verbindungsstücken und einem T-Stück eine dichte Wakü bauen?

(ii) Für welche $t \in \mathbb{N}_{>0}$ kann Poseidon dichte Waküs bauen, die aus genau t vielen T-Stücken[4] bestehen, d. h. für welche $t \in \mathbb{N}_{>0}$ existieren Multigraphen mit genau t Knoten vom Grad 3?

Bestimmen Sie die Menge

$$\textsc{Dicht} := \{t \in \mathbb{N}_{>0} : \text{Eine dichte Wakü kann mit genau } t \text{ vielen T-Stücken konstruiert werden}\}.$$

Begründen Sie für jedes $t \notin \textsc{Dicht}$, weshalb keine dichte Wakü mit t vielen T-Stücken konstruiert werden kann.

Hinweis: Verwenden Sie Ihre Erkenntnisse aus Aufgabenteil a).

Aufgabe 5.11. Sei $G = (V, E)$ ein ungerichteter planarer Graph mit $V \neq \emptyset$. Dann gilt, dass $|E| < 3 \cdot |V|$.

(a) Benutzen Sie diesen Zusammenhang, um zu beweisen, dass es in G einen Knoten $v \in V$ gibt mit $\text{Grad}_G(v) \leqslant 5$.

(b) Beweisen Sie, dass G mit sechs Farben gefärbt werden kann. D. h. beweisen Sie, dass G eine Färbung $m : V \to \{1, 2, 3, 4, 5, 6\}$ besitzt.[5]

[3] Ein Multigraph ohne Eigenschleifen mit der Kantenmarkierung $m(e) = 1$ für alle $e \in E$ kann als ungerichteter Graph aufgefasst werden.

[4] Es dürfen beliebig viele einfache Verbindungen verwendet werden, allerdings sind diese nicht notwendig.

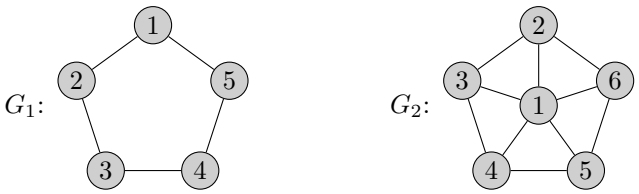
[5] Tatsächlich ist jeder planare Graph 4-färbbar. Der Beweis dieser als „Vier-Farben-Satz“ bekannten Aussage ist sehr aufwändig. Der hier geforderte Beweis der entsprechenden Aussage für sechs Farben ist wesentlich einfacher.

Hinweis: Benutzen Sie eine vollständige Induktion nach $n := |V|$ und die Aussage in Teilaufgabe ((a)) (auch wenn Sie diese nicht selbst bewiesen haben). Verwenden Sie keine anderen Resultate zur chromatischen Zahl planarer Graphen.

Aufgabe 5.12. Zwei Personen A und B spielen ein Spiel auf einem zusammenhängenden ungerichteten Graphen $G = (V, E)$. Die Spieler wählen abwechselnd Knoten $v_1, v_2, v_3, \ldots$ aus V, so dass $v_1, v_2, v_3, \ldots$ verschiedene Knoten sind und jeweils gilt: $\{v_i, v_{i+1}\} \in E$. Den ersten Knoten wählt A. Der letzte Spieler, der einen Knoten wählen kann, gewinnt.

Ein Spieler hat eine *Gewinnstrategie* in dem Spiel genau dann, wenn der Spieler das Spiel, unabhängig davon wie der andere Spieler spielt, gewinnen kann.

(a) Geben Sie für jeden der beiden folgenden Graphen G_1 und G_2 ein Matching maximaler Größe an und entscheiden Sie, welcher der beiden Spieler in dem Spiel auf dem entsprechenden Graph eine Gewinnstrategie hat.



(b) Beweisen Sie, dass die beiden folgenden Aussagen äquivalent sind:

(i) G besitzt ein *perfektes Matching*, also ein Matching M, so dass jeder Knoten aus V Endpunkt einer Kante in M ist.

(ii) Spieler B hat eine Gewinnstrategie in dem oben beschriebenen Spiel auf G.

Aufgabe 5.13. Auf Freds Geburtstagsfeier soll Sirtaki getanzt werden. Damit der Tanz gelingt, muss jede am Tanz beteiligte Person die rechte und linke Nachbarperson des Tanzkreises kennen. Fred weiß, dass es auf seiner Feier mindestens genauso viele Bekanntschaften wie Personen gibt und der Sirtaki auf jeden Fall ein voller Erfolg wird. Fred hat tatsächlich Recht, und das ist kein Zufall! Sei $G=(V, E)$ ein ungerichteter Graph mit $|V| = n$ und $|E| \geqslant n$ für ein $n \in \mathbb{N}_{>0}$.
Zeigen Sie, dass G einen einfachen Kreis besitzt.

Aufgabe 5.14. Auf dem Weihnachtsmarkt von Großdorf sollen acht Stände rund um den Marktplatz, den sogenannten Großmarktplatz, arrangiert werden. Die Stände setzen sich folgendermaßen zusammen:

- Ein Stand, in dem die traditionelle Weihnachtskrippe aufgebaut ist.
- Zwei Stände, an denen Weihnachtsschmuck verkauft wird; einer bietet Holzschmuck aus dem Erzgebirge an, der andere dekorative Zombies im Weihnachtsmannkostüm.
- Zwei Glühweinstände (einer von Herrn Max, der andere von Frau Peters).
- Drei Essensstände; einer davon verkauft Kartoffelpuffer, ein anderer glutenfreie Tofuspieße und der dritte Bratwürste vom Holzkohlegrill.

Bei der Platzierung der acht Stände um den Großmarktplatz ist Folgendes zu beachten:

1) Der Zombiestand und der Tofustand sind vielen älteren Bewohnern von Großdorf ein wenig suspekt und dürfen daher nicht neben der Weihnachtskrippe platziert werden.

2) Frau Peters und Herr Max betrachten jede Art von Lebensmitteln als Konkurrenz zu ihrem Angebot — die Glühweinstände dürfen daher nicht nebeneinander stehen, außerdem darf kein Glühweinstand neben einem Essensstand stehen.

3) Der Besitzer des Holzschmuckstandes ist strenger Veganer und fordert deshalb, dass sein Stand auf keinen Fall neben dem Stand mit den Bratwürsten steht. Dafür weigert sich die Besitzerin des Bratwurststandes, ihren Stand neben den mit den Tofuspießen zu stellen.

4) Außerdem muss beachtet werden, dass Herr Max und die Betreiberin des Holzschmuckstandes ihre Stände nicht neben den Zombiestand stellen wollen, da sie diesen albern und unweihnachtlich finden.

5) Schließlich will die Besitzerin des Kartoffelpufferstands ihren Stand nicht neben dem Holzschmuckstand aufbauen, da sie sich daran stört, dass dessen Besitzerin permanent den Verfall der Traditionen beklagt.

Zur Problemstellung:

(a) Stellen Sie den Konfliktgraph auf, in dem die Stände durch Knoten repräsentiert werden und eine Kante zwischen zwei Knoten anzeigt, dass die entsprechenden Stände nicht nebeneinander platziert werden können. Verwenden Sie zur Beschriftung der Knoten jeweils die fett gedruckten Buchstaben (z. B. **P** für den Stand von Frau Peters).

(b) Um eine Platzierung der Stände zu bestimmen, gehen Sie wie folgt vor: (i) Geben Sie das Komplement des Konfliktgraphen an. (ii) Geben Sie einen Hamilton-Kreis im Komplement des Konfliktgraphen an. (iii) Geben Sie eine Platzierung der acht Stände rund um den Großmarktplatz an, mit der alle zufrieden sind.

(c) Die Erfahrungen der letzten Jahre haben gezeigt, dass es selbst bei einer solchen Platzierung immer wieder zu Streit kommt. Die Bürgermeisterin von Großdorf überlegt daher, den Weihnachtsmarkt auf mehrere Standorte in Großdorf zu verteilen. Statt die Stände, zwischen denen ein Konflikt vorliegt, nicht nebeneinander zu stellen, sollen diese an unterschiedlichen Standorten aufgestellt werden. Neben dem Großmarktplatz bieten sich in Großdorf noch der Kleinmarktplatz und der Restmarktplatz als Standorte an. Nach ein wenig Herumprobieren ist sich die Bürgermeisterin allerdings nicht sicher, ob drei Standorte für diese Aufgabe ausreichen.

Bestimmen Sie die chromatische Zahl des Konfliktgraphen. Wie viele Standorte sind mindestens notwendig, um alle beschriebenen Konflikte auszuschließen? Wie viele Standorte reichen aus?

Aufgabe 5.15. Sei $d \in \mathbb{N}_{>0}$. Zeigen Sie: der d-dimensionale Würfel $W_d = (V_d, E_d)$ ist bipartit.

Aufgabe 5.16. Die fünf Studenten Alice, Bob, Carol, Eike und Leo wohnen seit einiger Zeit zusammen in einer WG. Da einige übellaunige Nachbarn sich mehrfach über die WG beschwert haben, hat sich die Vermieterin angekündigt, um den Zustand der Wohnung zu überprüfen. Da die fünf ungern ihre Wohnung verlieren wollen, beschließen sie, diese doch mal ein wenig aufzuräumen und die schlimmsten Ärgernisse zu beseitigen. Dabei identifizieren sie die folgenden fünf Aufgaben als besonders dringend: das Abtragen des Altglasberges, die Entsorgung aller Pizzaschachteln, das Verstecken von verdächtigen Gewächsen, eine ausführliche Desinfektion des Bades und das Einfangen der Ratten in der Küche. Da alle diese Aufgaben ungefähr gleich anstrengend sind, einigen sie sich darauf, dass jeder Bewohner genau eine dieser Aufgaben übernimmt. Dabei äußern sie folgende Abneigungen:

- Alice hat eine Abneigung gegen jede Aufgabe außer Rattenfangen,
- Bob mag sich weder um die Pizzaschachteln, noch um das Bad, noch um die Gewächse kümmern,
- Carol weigert sich den Altglasberg abzutragen,
- Eike will weder das Bad noch den Altglasberg übernehmen,
- Leo will nicht das Bad desinfizieren und auch nicht die Pizzaschachteln entsorgen.

(a) Stellen Sie den Konfliktgraphen als *ungerichteten* Graphen auf, dessen Knotenmenge die Bewohner und die Aufgaben repräsentiert. Eine Kante in diesem Graphen zwischen BewohnerIn A und Aufgabe B soll dafür stehen, dass A nicht B übernehmen will.

(b) Geben Sie auf der Grundlage Ihres Konfliktgraphen einen *ungerichteten* „Zufriedenheitsgraphen“ mit der gleichen Knotenmenge an. Eine Kante in diesem „Zufriedenheitsgraphen“ zwischen BewohnerIn A und Aufgabe B soll bedeuten, dass A einverstanden wäre B zu übernehmen.

(c) Geben Sie ein Matching maximaler Größe in Ihrem ufriedenheitsgraphen an.

(d) Geben Sie eine Zuordnung zwischen Bewohnern und Aufgaben an, so dass jeder Bewohner genau eine Aufgabe erhält, jede Aufgabe genau einmal zugeordnet wird und alle Bewohner mit ihrer Zuordnung zufrieden sind.

Aufgabe 5.17. Es soll ein Klausurplan für 7 Klausuren A–G aufgestellt werden, bei dem kein Student mehr als eine Klausur pro Tag schreiben muss. Über die Teilnehmer an den Klausuren ist Folgendes bekannt:

- Für jede der Klausuren B, C, E und G gibt es mindestens einen Studenten, der sich für diese Klausur und A angemeldet hat.
- Es gibt Studenten, die sich für B und C angemeldet haben, als auch Studenten, die die Kombination B und E gewählt haben.
- Jeder Student, der D mitschreibt, hat sich auch für C und G angemeldet.
- Mindestens ein Teilnehmer der Klausur G nimmt an F teil.

(a) Stellen Sie den Konfliktgraphen auf.

(b) Geben Sie einen Klausurplan an, bei dem kein Student an mehr als einer Klausur pro Tag teilnimmt.

(c) Wie viele Tage werden für einen solchen Klausurplan benötigt?

Aufgabe 5.18. Beweisen Sie die Gültigkeit der folgenden Aussagen:

(a) Für jeden ungerichteten Baum $B = (V, E)$ mit $V \neq \emptyset$ gilt: B ist bipartit.

(b) Für jeden ungerichteten Graph G gibt es mindestens einen Spannbaum von G, falls G zusammenhängend ist. (Rückrichtung von Satz 5.67)

(c) (i) Für jeden ungerichteten Graphen $G = (V, E)$ gilt: Bei jeder Färbung von G mit $\chi(G)$ Farben muss es eine Menge von mindestens $\frac{|V|}{\chi(G)}$ Knoten in G geben, die mit der gleichen Farbe gefärbt sind.

(ii) Für jeden ungerichteten Graphen $G = (V, E)$ und sein Komplement $\tilde{G} = (\tilde{V}, \tilde{E})$ gilt: $\chi(G) \cdot \chi(\tilde{G}) \geqslant |V|$

Aufgabe 5.19. Zeigen Sie: Ein ungerichteter Graph $G = (V, E)$ ist genau dann bipartit, wenn es in G keine Kreise ungerader Länge gibt.

Hinweis: Sie können $V = \{1, 2 \ldots, n\}$ annehmen. Zeigen Sie dann per Induktion über n, dass $V_1 := \{v \in V :$ es gibt einen Weg gerader Länge von Knoten v nach Knoten 1.$\}$ und $V_1 := \{u \in V :$ es gibt einen Weg ungerader Länge von Knoten u nach Knoten 1$\}$ eine Partition von V (mit $V_1 \cap V_2 = \emptyset$) ist.

Aufgabe 5.20. Beweisen Sie die Gültigkeit der folgenden Aussagen:

(a) Für je zwei ungerichtete Graphen G_1 und G_2 mit $G_1 \cong G_2$ gilt: Wenn G_1 zusammenhängend ist, so ist auch G_2 zusammenhängend.

(b) Für jeden ungerichteten Graphen G gilt: G oder $\overline{G}$ ist zusammenhängend.

(c) Für einen ungerichteten Graphen $G = (V, E)$ sei der Graph $\overline{G} = (V, \overline{E})$ das *Komplement* von G, falls $\overline{E} := \{\{x, y\} : x, y \in V, x \neq y, \{x, y\} \notin E\}$. Ein ungerichteter Graph heißt *selbst-komplementär*, wenn er isomorph zu seinem Komplement ist.

Zeigen Sie: Jeder selbst-komplementäre Graph ist zusammenhängend.

Aufgabe 5.21.

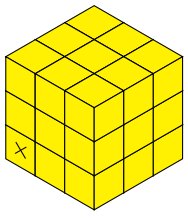
(a) Geben Sie für jedes $n \in \mathbb{N}_{>0}$ mit $n \leqslant 7$ einen selbst-komplementären Graphen an, der n Knoten hat, falls ein solcher Graph existiert.

(b) Beweisen Sie, dass für die Knotenanzahl $|V|$ jedes selbst-komplementären Graphen $G = (V, E)$ gilt: Es gibt ein $k \in \mathbb{N}$ so dass $|V| = 4k$ oder $|V| = 4k + 1$.

Aufgabe 5.22. Sei $G = (V, E)$ ein gerichteter Graph mit mindestens zwei Knoten, so dass für alle $v, v' \in V$ mit $v \neq v'$ entweder $(v, v') \in E$ oder $(v', v) \in E$. Beweisen Sie durch vollständige Induktion über die Anzahl der Knoten von G, dass G einen Hamilton-Weg besitzt.

Aufgabe 5.23. (a) Sei $G = (V, E)$ ein bipartiter Graph mit den Bipartitionsklassen V_1 und V_2, d.h. jede Kante von E hat einen Endknoten in V_1 und einen in V_2, außerdem ist $V = V_1 \dot{\cup} V_2$. Beweisen Sie die folgende Aussage: Falls $|V_1| > |V_2|$, so kann es keinen Hamilton-Weg in G geben, der in V_2 endet.

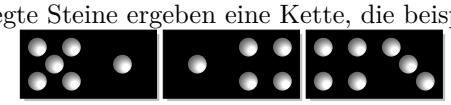
(b) Es sei ein Käsewürfel gegeben, der in $3 \times 3 \times 3$ kleinere Teilwürfel wie in der nebenstehenden Abbildung unterteilt ist. Zwei Teilwürfel sind *benachbart*, wenn sie sich entlang einer Fläche berühren. So ist beispielsweise der mit einem Kreuz markierte Teilwürfel mit 3 Teilwürfeln benachbart. Weiterhin sei eine Maus gegeben, die das Ziel hat, den großen Käsewürfel vollständig zu verspeisen. Sie kann dabei nur schrittweise vorgehen, indem sie in jedem Schritt einen Teilwürfel komplett verspeist und im nächsten Schritt mit einem Teilwürfel fortfährt, der mit dem gerade verspeisten



Teilwürfel benachbart ist. Außerdem macht die Maus keinen Schritt, ohne einen Teilwürfel zu fressen und natürlich kann jeder Teilwürfel nur genau einmal gefressen werden.[6]

Kann die Maus bei dem markierten Teilwürfel starten und alle Teilwürfel so verspeisen, dass sie als Letztes den Teilwürfel in der Mitte des Kasewürfels frisst? Beweisen Sie, dass Ihre Antwort korrekt ist. (Sie können dafür die Aussage aus Teilaufgabe ((a)) benutzen, auch wenn Sie diese nicht bewiesen haben.)

Aufgabe 5.24. Die Menge $Dom_n := \{\{x, y\} : x, y \in \mathbb{N}, x < y \leqslant n\}$ repräsentiert eine Teilmenge aller möglichen Dominosteine. Das Element $\{x, y\} = \{y, x\}$ steht für den Stein, der x Augen auf der einen und y Augen auf der anderen Hälfte zeigt, wobei $x \neq y$ für jeden Stein in Dom_n gilt. Zwei Steine können aneinander gelegt werden, wenn ihre benachbarten Augenzahlen gleich sind. Mehrere aneinander gelegte Steine ergeben eine Kette, die beispielsweise wie folgt beginnt:

 ...

Ist es möglich, eine einzelne Kette ohne Verzweigungen so zu legen, dass jeder Stein aus Dom_n genau einmal vorkommt?

(a) Modellieren Sie diese Frage als ein Problem für einen geeigneten Graphen.

(b) Gibt es eine Kette für $n = 18$? Beweisen Sie, dass Ihre Antwort korrekt ist.

(c) Für welche $n \in \mathbb{N}$ gibt es eine Kette, für welche nicht?

Aufgabe 5.25.

(a) Gegeben sei ein ungerichteter Graph mit Knotenmenge V und jeder Knoten habe höchstens drei Nachbarn. Zeigen Sie: Es gibt eine Partition $V = V_1 \cup V_2$ mit $V_1 \cap V_2 = \emptyset$ sodass für alle $i = 1, 2$ gilt: Jeder Knoten in V_i hat höchstens einen Nachbarn in V_i.

(b) Gegeben sei ein ungerichteter Graph. Zeigen Sie: Es gibt zwei Knoten mit demselben Grad.

(c) Gegeben sei ein ungerichteter Graph: Zeigen Sie, dass die Anzahl der Knoten mit ungeradem Grad gerade ist.

Aufgabe 5.26. Der Soziologe Norbert E. Twork befasst sich im Rahmen seiner Promotion mit sozialen Netzwerken. Norbert hat Kindergartengruppen untersucht und dabei festgestellt, dass es in jeder Gruppe stets drei Kinder gibt, die alle miteinander befreundet sind, oder drei Kinder, die alle nicht miteinander befreundet sind. Zunächst macht sich Norbert Hoffnung, dass seine Entdeckung auf eine soziologische Besonderheit von Kindergartengruppen hindeutet, die er im Rahmen seiner Dissertation weiter untersuchen könnte. Norberts Bekannte Dana Ismod macht ihn allerdings auf einen Schwachpunkt seines Vorhabens aufmerksam: Norbert hat nur Kindergartengruppen mit mindestens sechs Kindern untersucht. Dana erklärt Norbert, dass dieses Phänomen in allen sozialen Netzwerken von mindestens sechs Personen auftritt.

(a) Formulieren Sie Danas Behauptung mit Begriffen der Graphentheorie.

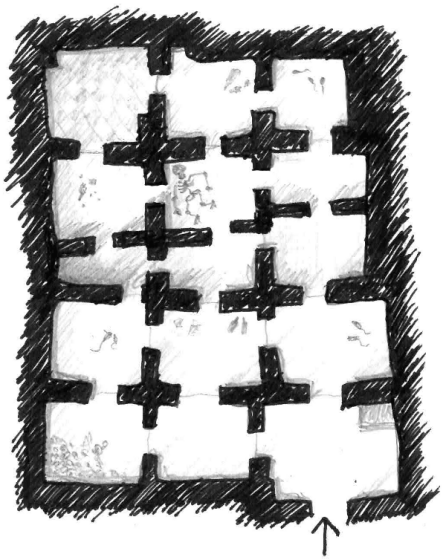
[6] Es ist noch zu erwähnen, dass sich die ganze Situation in der Schwerelosigkeit abspielt, d.h. die Maus kann sich in jede Richtung fressen und es besteht auch nicht die Gefahr, dass der Würfel umkippt oder herunterfällt, wenn die untere Ebene von Teilwürfeln teilweise oder komplett gefressen ist.

(b) Beweisen Sie, dass Dana Recht hat.

Hinweis Wählen Sie aus einem entsprechenden sozialen Netzwerk eine Person aus. Was passiert, wenn diese mit mindestens drei Personen befreundet ist? Was passiert ansonsten?

Aufgabe 5.27.

Tief versteckt in den nebligen Wäldern des Taunus steht am Rande eines kleinen Dorfes eine finstere alte Villa. Hier haust die unheimliche Gräfin D'Ismod mit ihrem buckligen Butler Igor und verbringt ihre Zeit damit, die Dorfbewohner zu beunruhigen und seltene Weine zu sammeln.



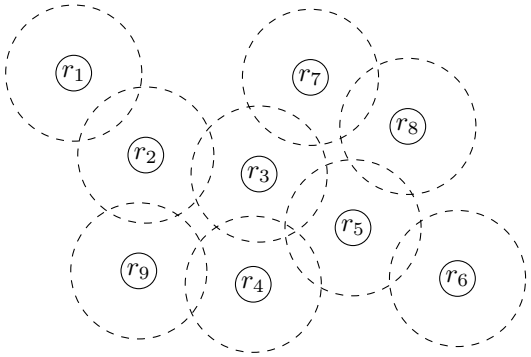
Eines Tages hat die Gräfin Lust auf einen besonders seltenen Wein und gibt Igor den Auftrag, eine Flasche *Königsberger Eulenkopf 1857* aus dem Keller zu holen. Igor ist sich zwar sicher, dass die letzte Flasche dieses Weines bereits vor ein paar Jahren verbraucht wurde, allerdings besteht die Gräfin darauf, dass er in jedem einzelnen Raum ihres ausgedehnten Weinkellers nachsieht.

Der Weinkeller besteht aus 15 Räumen, die in einem Rechteck (3 Räume in der Breite, 5 Räume in der Länge) angeordnet sind. Die Treppe, die die einzige Verbindung zwischen Keller und Außenwelt darstellt, befindet sich in der rechten unteren Ecke. Aus jedem Raum kann man entlang der vier Hauptrichtungen die benachbarten Räume betreten – Igor kann sich also einen Raum nach links, rechts, oben oder unten bewegen, aber nicht entlang der Diagonalen.

Igor sucht nun eine Route durch den Keller, die ihn vom Raum mit der Treppe zu jedem Raum des Kellers bringt und schließlich wieder im Raum mit der Treppe endet. Da der Auftrag schon mühsam genug ist, will Igor den Raum mit der Treppe nur zweimal betreten (nämlich zu Beginn und Ende seiner Route), und jeden der anderen Räume nur ein einziges Mal.

Kann Igor eine solche Route finden? Beweisen Sie, dass Ihre Antwort korrekt ist.

Aufgabe 5.28. Es seien die Radiostationen $r_1, \ldots, r_9$ gegeben, denen jeweils eine Sendefrequenz zugeordnet werden soll. Radiostationen, die zu dicht beieinander liegen, dürfen allerdings nicht die gleiche Frequenz erhalten. Das nebenstehende Diagramm stellt die Lage der einzelnen Radiostationen dar. Um jede Station ist ein gestrichelter Kreis eingezeichnet, der die Reichweite einer Radiostation repräsentiert. Schneiden sich die Kreise von zwei Radiostationen r_i und r_j, so liegen r_i und r_j zu dicht beieinander und dürfen nicht die gleiche Frequenz zugeordnet bekommen.



(a) Geben Sie den Konfliktgraphen an, der als Knotenmenge die Radiostationen besitzt und bei dem eine Kante zwischen zwei Radiostationen r_i und r_j anzeigt, dass r_i und r_j nicht die gleiche Frequenz benutzen dürfen.

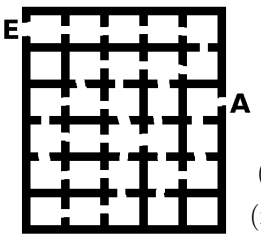
(b) Sei $G = (V, E)$ der Konfliktgraph aus Aufgabenteil ((a)). Geben Sie eine Färbung $m : V \to F$ für G mit möglichst wenigen Farben an.

(c) Weisen Sie jeder der Radiostationen $r_1, \ldots, r_9$ genau eine Frequenz zu, so dass Radiostationen, die zueinander in Konflikt stehen, nicht die gleiche Frequenz erhalten und möglichst wenige verschiedene Frequenzen benötigt werden.

(d) Wie viele verschiedene Frequenzen werden für die Radiostationen $r_1, \ldots, r_9$ mindestens benötigt, d.h. wie groß ist die chromatische Zahl des Konfliktgraphen?

Aufgabe 5.29. Ein kleine Gruppe düsterer Kultisten sucht in den Gewölben einer längst verlassenen Stadt nach alten Büchern. Bei ihrer Suche stoßen sie auf ein altes Labyrinth, das sie durchqueren müssen.

(a) In der folgenden Grafik ist der Grundriss des Labyrinthes abgebildet. Der Eingang und der Ausgang sind durch **E** und **A** markiert:



(i) Modellieren Sie das Labyrinth durch einen ungerichteten Graphen $G = (V, E)$. Repräsentieren Sie dazu den Eingang **E**, den Ausgang **A** und jeden **von E erreichbaren Raum** durch Knoten in V. Räume mit genau zwei Türen müssen Sie **nicht** berücksichtigen.

(ii) Bestimmen Sie einen Spannbaum T für G.

(iii) Geben Sie einen Weg in T an, der eine Route vom Eingang zum Ausgang beschreibt und durch das Labyrinth führt.

(b) In einem uralten verstaubten Buch finden die Kultisten Hinweise, wie sie den Dämonen Cthulhu erwecken können. Es müssen die folgenden acht Symbole in einem Kreis angeordnet werden:

Runde bzw. *eckige* Symbole werden jeweils durch einen Kreis bzw. ein Quadrat eingerahmt. *Licht-Symbole* sind schwarz auf weißem Hintergrund, während *Schatten-Symbole* weiß auf schwarzem Hintergrund sind. Für die Beschwörung Cthulhus ist die Anordnung der Symbole

im Kreis entscheidend. Dabei ist zu beachten, dass manche Symbole im Konflikt stehen und nicht (direkt) nebeneinander angeordnet werden dürfen. Folgende Konflikte gibt es:

- Ein eckiges Licht-Symbol darf nicht neben einem runden Licht-Symbol stehen.
- Schattensymbole mit der gleichen Form (rund bzw. eckig) dürfen nicht nebeneinander stehen.
- Schatten-Symbole dürfen nicht neben Licht-Symbolen stehen, wenn beide rund bzw. beide eckig sind.
- Die Symbole und dürfen nicht nebeneinander stehen, ebenso nicht die Symbole und .

(i) Stellen Sie gemäß den Anweisungen des Buches den Konfliktgraphen auf. Stellen Sie anschließend den Komplementgraphen des Konfliktgraphen auf.

(ii) Gelingt es den Kultisten, Cthulhu zu erwecken?

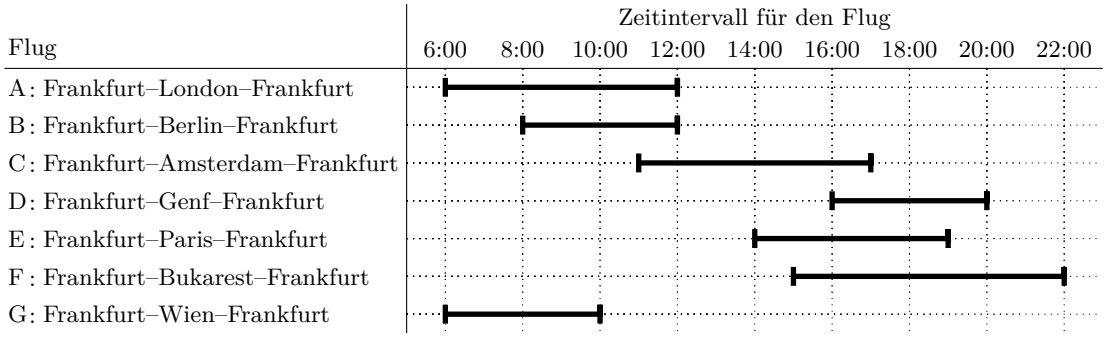
(iii) In einem anderen Kapitel des Buches steht geschrieben, wie die Entität Yog-Sothoth erweckt werden kann:

Die selben acht Symbole müssen auf drei verschiedene Steintafeln geschrieben werden. Dabei darf jedes Symbol nur auf einer einzigen Tafel stehen und zwei im Konflikt stehende Symbole müssen auf unterschiedliche Tafeln geschrieben werden. Ist es möglich, Yog-Sothoth zu erwecken?

(iv) Der Legende nach steht jedes der acht Symbole für einen Dämon und jeder Konflikt unter ihnen steht für ein Duell zwischen den beiden, das vor langer Zeit stattgefunden hat. Man sagt, es haben sogar vier Duelle gleichzeitig stattgefunden! Kann diese Legende wahr sein?

(v) Welche graphentheoretischen Probleme müssen die Kultisten in ii), iii) und iv) lösen?

Aufgabe 5.30. Die kleine Frankfurter Fluggesellschaft Air-Flight hat für den kommenden Freitag sieben Flüge geplant, die wir im Folgenden mit den Buchstaben A–G bezeichnen. Für jeden Flug ist ein Zeitintervall (Abflugzeit bis Ankunftszeit) vorgesehen, in dem der Flug stattfinden soll:



Nun muss jedem Flug eines von fünf Flugzeugen F1–F5 zugeordnet werden, das für den Flug eingesetzt wird. Natürlich dürfen zwei Flüge, die zueinander in Konflikt stehen (d.h. bei denen sich die Zeitintervalle überlappen), nicht dem selben Flugzeug zugeordnet werden.

(a) Geben Sie den Konfliktgraphen an, der als Knotenmenge die Flüge besitzt und bei dem eine Kante einen Konflikt zwischen zwei Flügen anzeigt.

(b) Sei $G = (V, E)$ der Konfliktgraph aus Aufgabenteil ((a)). Geben Sie eine Färbung $m : V \to F$ mit möglichst wenigen Farben an.

(c) Weisen Sie jedem der Flüge A–G genau eines der Flugzeuge F1–F5 zu, so dass Flüge, die zueinander in Konflikt stehen, nicht dem selben Flugzeug zugeordnet sind und möglichst wenige verschiedene Flugzeuge benötigt werden.

(d) Wie viele Flugzeuge werden für die Flüge A–G mindestens benötigt, d. h. wie groß ist die chromatische Zahl des Konfliktgraphen?

Aufgabe 5.31. Betrachten Sie die Operationen **Löschen** bzw. **Kontraktion** auf einem ungerichteten Graphen $G = (V, E)$.

(1) Beim **Löschen** darf eine Kante aus E entfernt werden oder ein Knoten aus V (mitsamt aller inzidenter Kanten) entfernt werden.

(2) Sei $\{u, v\} \in E$ und $w \notin V$. Eine **Kontraktion** der Kante $\{u, v\}$ erzeugt aus G einen neuen Graphen $G' = (V', E')$ mit

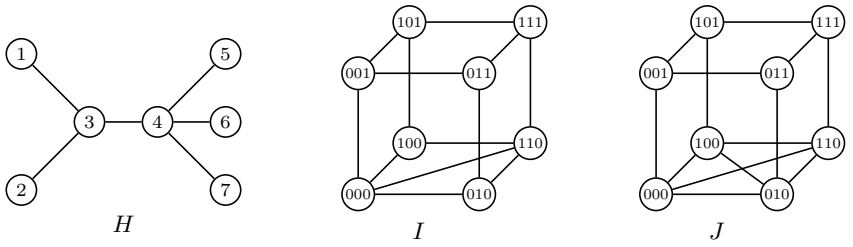
$$\begin{aligned} V' =& (V \setminus \{u, v\}) \cup \{w\} \\ E' =& \Big(E \setminus \{e \in E : u \in e \text{ oder } v \in e\}\Big) \\ & \cup \Big\{\{w, x\} : x \notin \{u, v\} \text{ und } \big(\{u, x\} \in E \text{ oder } \{v, x\} \in E\big)\Big\} \end{aligned}$$

Anschaulich: Eine Kontraktion verschmilzt eine Kante $\{u, v\}$ mitsamt ihrer beiden Endknoten zu einem neuen Knoten w. Alle Knoten (ausgenommen u und v), die in G mit u bzw. v benachbart waren, sind in G' mit w benachbart.

Mithilfe dieser beiden Operationen können wir zeigen, dass ein Graph nicht planar ist.

Satz von Wagner. Ein ungerichteter Graph G ist genau dann nicht planar, wenn es eine Folge von Kontraktionen, Knoten- bzw. Kantenlöschungen gibt, sodass aus G ein Graph erzeugt werden kann, der zu K_5 oder $K_{3,3}$ isomorph ist. ◇

Seien die Graphen H, I und J wie folgt in grafischer Darstellung gegeben:



(a) Führen Sie eine Kontraktion der Kante $\{3, 4\}$ im Graphen H durch.

(b) Zeigen oder widerlegen Sie: I ist planar.

(c) Zeigen oder widerlegen Sie: J ist planar.

Aufgabe 5.32.

(a) Betrachte die Funktion `fibonacci_naiv`, die auf naive Art die n-te Fibonacci-Zahl fib(n) (siehe Beispiel 4.19) rekursiv berechnet:

```
def fibonacci_naiv(n):
    if n <= 2:
        return 1
    else:
        return fibonacci_naiv(n-2) + fibonacci_naiv(n-1)
```

(i) Geben Sie den Rekursionsbaum Baum(7) für den Aufruf `fibonacci_naiv`(7) in grafischer Darstellung an. Beschriften Sie jeden Knoten, der einem Aufruf `fibonacci_naiv`(p) entspricht, mit dem Parameter p. Ist Baum(7) ein Binärbaum? Ist er ein voller Binärbaum? Ist er ein vollständiger Binärbaum?

(ii) Zeigen Sie durch vollständige Induktion: Für jedes $n \in \mathbb{N}_{>0}$ hat Baum(n) für den Aufruf `fibonacci_naiv`(n) genau fib(n) Blätter.

(b) Betrachten Sie die Funktion `prod` für die Russische Bauernmultiplikation:

```
def prod(x,k):
    if k == 0:
        return 0
    elif k % 2 == 0:                # k ist gerade und groesser 0
        return prod(2*x, k/2)
    else:                           # k ist ungerade
        return prod(2*x, k//2) + x  # k//2 entspricht der
                                    # ganzzahligen Division (k-1)/2
```

(i) Geben Sie den Rekursionsbaum Baum$(12, 21)$ für den Aufruf `prod`$(12, 21)$ in grafischer Darstellung an. Beschriften Sie jeden Knoten, der einem Aufruf `prod`(x, k) entspricht, mit dem Tupel (x, k). Handelt es sich um einen Binärbaum?

(ii) Sei $n \in \mathbb{N}$ und $x \in \mathbb{R}$. Bestimmen Sie die Tiefe des Rekursionsbaums Baum(x, 2^n).

Aufgabe 5.33. Sei $k \geqslant 2$. Ein gewurzelter Baum $B = (V, E)$ ist ein *vollständiger k-ärer Baum*, falls gilt:

- für jeden Knoten $v \in V$ gilt Aus-Grad$_B(v) = k$ oder v ist ein Blatt
- und alle Blätter von B haben dieselbe Tiefe.

Zeigen Sie mit vollständiger Induktion: Ein vollständiger k-ärer Baum der Tiefe t besitzt genau $\frac{k^{t+1}-1}{k-1}$ Knoten.

Aufgabe 5.34. Zwei Spieler A und B spielen das folgende Spiel. Das Spiel ist in Runden aufgeteilt, wobei Spieler A in den geraden Runden und Spieler B in den ungeraden Runden spielt. In der ersten Runde wählt Spieler B eine Zahl aus $\{1, 2\}$. In jeder der nachfolgenden Runden wählt der jeweilige Spieler eine Zahl aus $\{1, 2, 3\}$ mit der Einschränkung, dass die Zahl aus der vorhergehenden Runde nicht gewählt werden darf. Nach jeder Runde wird die Summe der bereits gewählten Zahlen berechnet. Nimmt diese Summe den Wert 6 an, so gewinnt der Spieler der jeweiligen Runde; übersteigt sie den Wert 6, so verliert er.

(a) Beschreiben Sie das Spiel durch einen Entscheidungsbaum.

(b) Wer gewinnt, wenn beide Spieler optimal spielen, d.h. wenn jeder Spieler immer nur diejenigen Zahlen wählt, mit denen er – falls dies noch möglich ist – gewinnen kann?

Aufgabe 5.35. Alice und Bob spielen das Spiel *Yag*[7] gegeneinander, das wie folgt definiert ist: Die Spieler schreiben gemeinsam eine Folge von Nullen und Einsen auf. Sie beginnen mit der leeren Zeile und sind abwechselnd am Zug, Alice beginnt. Der Spieler am Zug schreibt an das Ende der Zeile eine Null oder eine Eins. Ein Spieler gewinnt, wenn die von ihm hinzugefügte Ziffer einen Block der Länge zwei erzeugt, der in der Folge schon einmal vorkommt. (Dabei werden auch Blöcke betrachtet, die sich überlappen: Der Spieler, der die Folge 0111 erzeugt, gewinnt, da der Block 11 zweimal in der Folge vorkommt.) Haben die Spieler eine Folge der Länge vier erzeugt ohne dass es einen Gewinner gibt, so endet das Spiel unentschieden.

(a) Beschreiben Sie das Spiel durch einen Entscheidungsbaum.

(b) Ist der von Ihnen bei Teilaufgabe ((a)) aufgestellte Entscheidungsbaum ein Binärbaum? Ist er ein vollständiger Binärbaum?

(c) Was haben alle Spielsituationen gemeinsam, die den Blättern des Entscheidungsbaumes entsprechen? Was haben alle Situationen des Spiels gemeinsam, die durch innere Knoten repräsentiert werden? Welcher Spielsituation entspricht die Wurzel?

(d) Wie viele Runden dauert das Spiel höchstens, d. h. wie groß ist die Höhe des Entscheidungsbaumes? Wie viele Runden dauert das Spiel mindestens, d. h. was ist die kürzeste Länge eines Weges von der Wurzel zu einem Blatt?

(e) Wer gewinnt, wenn beide Spieler optimal spielen, d. h. wenn jeder Spieler immer nur die Ziffer wählt, mit der er – falls dies noch möglich ist – gewinnen kann?

Aufgabe 5.36. Für ein $n \in \mathbb{N}$ seien 2^n Münzen gegeben, die wir im Folgenden mit $M_1, \ldots, M_{2^n}$ bezeichnen. Genau eine der Münzen ist schwerer als alle anderen. Diese Münze lässt sich mit Hilfe einer Balkenwaage wie folgt finden:

(i.) Falls $n = 0$, ist die gesuchte Münze die einzige, die vorhanden ist.

(ii.) Ansonsten vergleiche das Gesamtgewicht der Münzen aus der Menge $A := \{M_1, \ldots, M_{2^{n-1}}\}$ mit dem Gesamtgewicht der Münzen aus der Menge $B := \{M_{2^{n-1}+1}, \ldots, M_{2^n}\}$. Ist das Gesamtgewicht von A größer als das von B, muss sich die gesuchte Münze in A befinden und das beschriebene Verfahren wird rekursiv auf die Menge A angewendet, andernfalls wird es rekursiv auf die Menge B angewendet.

(a) Beschreiben Sie das Verfahren für $n = 2$ durch einen Entscheidungsbaum. Wählen Sie hierfür geeignete Kanten- und Knotenbeschriftungen.

(b) Ist der von Ihnen in Teilaufgabe ((a)) aufgestellte Entscheidungsbaum ein Binärbaum? Ist er ein vollständiger Binärbaum?

(c) Welchen Situationen im Entscheidungsprozess entsprechen die inneren Knoten des Baumes? Welcher Situation entspricht ein Blatt?

(d) Wie viele Wiegevorgänge müssen für 2^n Münzen mindestens durchgeführt werden? Wie viele Wiegevorgänge sind im schlimmsten Fall, also höchstens, nötig?

[7] Yet another game.

6. Markov-Ketten und Googles Page-Rank

In diesem Kapitel geben wir einen kurzen Überblick über die Arbeitsweise von Suchmaschinen für das Internet. Wir betrachten hierbei eine Suchmaschine, die als Eingabe ein Stichwort oder eine Liste von Stichworten erhält und als Ausgabe eine Liste von Links auf Webseiten geben soll, deren Inhalt relevante Informationen zu den eingegebenen Stichworten enthält. Diese Liste soll so sortiert sein, dass die informativsten Links am weitesten oben stehen.

Die Herausforderungen, die sich beim Bau einer Suchmaschine stellen, sind enorm. Zum einen ist die Anzahl der Webseiten *sehr* groß: Bereits im Jahr 2012 gab es ca. 2,4 Milliarden Internetnutzer weltweit, 634 Millionen Websites und 3,5 Milliarden Webseiten. In einem Jahr gab es 1,2 Billionen, also 10^{12} Suchanfragen auf Google allein. [1] Desweiteren ändert sich das Web ständig, täglich kommen neue Webseiten hinzu, viele Webseiten werden täglich aktualisiert, und andere nach einiger Zeit auch wieder gelöscht. Eine Suchmaschine muss daher eine riesige Menge von Daten verarbeiten, die in kurzen Zeitabständen immer wieder aktualisiert werden. Trotzdem müssen Suchanfragen, die an eine Suchmaschine geschickt werden, in hoher Qualität in „Echtzeit“ beantwortet werden.

Um die Ergebnisse nach ihrer Relevanz für die jeweiligen Suchbegriffe sortieren zu können, benötigt man auch ein sinnvolles Maß dafür, welche Webseiten als besonders „informativ“ bewertet werden sollen. Die Beschreibung des Page-Ranks als ein solches Maß ist das erste wesentliche Ziel dieses Kapitels.

Um zu verstehen, warum der Page-Rank „funktioniert“, benutzt man fundamentale Ergebnisse aus der Theorie der Markov-Ketten. Wir werden nicht alle Details der Analyse ausführen können, doch werden wir ein ausreichendes Grundverständnis erreichen, um auch weitere Anwendungen von Markov-Ketten in Angriff nehmen zu können. Die Durchführung dieser Analyse wie auch die weiteren Anwendungsmöglichkeiten von Markov-Ketten sind das zweite wesentliche Ziel des Kapitels.

6.1. Die Architektur von Suchmaschinen

Die Herausforderung besteht darin, Anfragen für einen sich rasant ändernden Suchraum gigantischer Größe ohne merkliche Reaktionszeit zu beantworten. Um dies zu bewältigen, nutzen Suchmaschinen die folgenden Komponenten:

(1) **Web-Crawler:** Computerprogramme, die **Crawler** genannt werden, durchforsten das Internet, um neue oder veränderte Webseiten zu identifizieren. Die von den Crawlern gefundenen Informationen über Webseiten und deren Inhalt werden aufbereitet und gespeichert.

(2) **Indexierung:** Die Informationen werden in einer Datenstruktur gespeichert, mit deren Hilfe bei Eingabe eines Suchworts in „Echtzeit“ alle Webseiten ermittelt werden können, die das Suchwort enthalten.

[1] Quelle: http://royal.pingdom.com/2013/01/16/internet-2012-in-numbers/; zuletzt besucht am 01.12.2014.

(3) **Bewertung der Webseiten:** Die ausgewählten Webseiten werden im Hinblick auf ihren Informationsgehalt (hinsichtlich möglicher Suchworte sowie hinsichtlich ihrer generellen Bedeutung im Internet) bewertet.

Zu jeder vom Crawler gefundenen Webseite wird die URL, d.h. die Adresse sowie der Inhalt der Webseite gespeichert.

Der Inhalt der Webseite wird analysiert und es werden Informationen darüber gespeichert, welches Wort mit welcher Häufigkeit und an welchen Positionen (etwa: im Titel, als Überschrift, im Fließtext, mit welcher Schriftgröße etc.) in der Webseite vorkommt. Diese Informationen werden im so genannten **Index** gespeichert.

Index

Außerdem werden Links, die auf Webseiten angegeben sind, analysiert. Enthält Webseite i einen Link auf eine Webseite j, so wird der Text, mit dem der Link beschriftet ist, im zu j gehörenden Index-Eintrag abgelegt. Diese Linkbeschriftungen geben wertvolle Hinweise darüber, welche Informationen die Webseite j enthält.

invertierter Index

Aus dem Index wird der so genannte **invertierte Index** generiert. Dies ist eine Datenstruktur, die zu jedem möglichen Suchwort eine Liste aller Webseiten angibt, die dieses Suchwort enthalten. Dabei werden jeweils auch Zusatzinformationen gespeichert, die die Wichtigkeit des Suchworts innerhalb der Webseite beschreiben, z.B. die Häufigkeit des Stichworts, seine Position und Schriftgröße innerhalb der Webseite sowie das Vorkommen des Stichworts in Beschriftungen von Links *auf* die Webseite.

Web-Graph

Die **Link-Struktur** des Webs wird durch den **Web-Graphen** modelliert. Der Web-Graph ist ein gerichteter Graph, bei dem jede Webseite (d.h. jede URL) durch einen Knoten repräsentiert wird, und bei dem es eine Kante von Knoten i zu Knoten j gibt, wenn Webseite i einen Link auf Webseite j enthält.

Bearbeitung von Suchanfragen:

Bei Eingabe einer Liste von Such-Stichworten soll die Suchmaschine die hinsichtlich dieser Stichworte informativsten Webseiten finden und diese sortiert nach ihrer Relevanz anzeigen. Dabei werden folgende Kriterien berücksichtigt:

(1) syntaktische Kriterien wie die Häufigkeit und Positionierung der Suchbegriffe auf der jeweiligen Webseite sowie in der Beschriftung von Links, die auf diese Webseite verweisen, und

(2) die grundlegende Bedeutung einer Webseite.

Für ((1)) können Methoden aus dem Bereich des **Information Retrieval** verwendet werden; Details dazu finden sich z.B. in Kapitel 6 von [20].

Ausschlaggebend für ((2)) ist die Link-Struktur des Webs, also der Web-Graph. Insbesondere geht man von der folgenden Annahme aus: Wenn eine Webseite i einen Link auf eine Webseite j enthält, dann

- gibt es eine inhaltliche Beziehung zwischen beiden Webseiten, und
- der Autor der Webseite i hält die Informationen auf Webseite j für wertvoll.

Es gibt verschiedene Verfahren, die Maße für die grundlegende Bedeutung einer Webseite liefern, beispielsweise das von *Google* genutzte **Page-Rank** Verfahren von Brin und Page [3] oder die **HITS** Methode (Hypertext Induced Topic Search) von Kleinberg [15]. Beide Ansätze versuchen, die in der Link-Struktur manifestierte „relative Wertschätzung“ zwischen einzelnen Webseiten in

eine „grundlegende Bedeutung“ der Webseiten umzurechnen. Details zu den beiden Verfahren finden sich in dem Buch [17].

Bei der Bearbeitung einer Suchanfrage, bei der eine Liste s von Such-Stichworten eingegeben wird, wird dann unter Verwendung von ((1)) und ((2)) jeder Webseite i ein Wert $\mathit{Score}(i, s)$ zugeordnet, der als Maß für die Relevanz der Webseite i hinsichtlich der Suchanfrage s dient. Als Trefferliste gibt die Suchmaschine dann eine Liste aller Webseiten aus, deren Score über einer bestimmten Schranke liegt und sortiert die Liste so, dass die Webseiten mit dem höchsten Score am weitesten oben stehen. Wie der Wert $\mathit{Score}(i, s)$ gewählt wird, ist Betriebsgeheimnis der einzelnen Betreiber von Suchmaschinen.

Im Rest dieses Kapitels werden wir uns anhand des Page-Rank-Verfahrens genauer ansehen, wie die „grundlegende Bedeutung“ einer Webseite modelliert und berechnet werden kann.

6.2. Der Page-Rank einer Webseite

Der Page-Rank liefert ein Maß für die „grundlegende Bedeutung“ einer Webseite, das allein durch die Link-Struktur des Webs bestimmt wird, ohne dass der textuelle Inhalt einer Webseite dabei berücksichtigt wird.

Sei $\text{WEB} = (V, E)$ der Web-Graph. Der Einfachheit halber nehmen wir an, dass die Webseiten mit den Zahlen $1, \ldots, n$ durchnummeriert sind, es gilt also $V = \{1, 2, \ldots, n\}$. Jeder Knoten von WEB repräsentiert eine Webseite und jede Kante $(i, j) \in E$ modelliert einen Hyperlink von Webseite i auf Webseite j. Für jeden Knoten $i \in V$ sei

$$a_i \;:=\; \text{Aus-Grad}_{\text{WEB}}(i)$$

der Aus-Grad von i in WEB, d.h. a_i ist die Anzahl der Hyperlinks, die von der Webseite i auf andere Webseiten verweisen. Für eine Webseite $j \in V$ bezeichnen wir mit $\text{Vor}_{\text{WEB}}(j)$ die Menge aller Webseiten, die einen Link auf j enthalten, d.h.

$$\text{Vor}_{\text{WEB}}(j) \;:=\; \{i \in V : (i, j) \in E\}.$$

Die Elemente in $\text{Vor}_{\text{WEB}}(j)$ werden **Vorgänger** von j genannt.

Die „grundlegende Bedeutung“ einer Webseite i wird im Folgenden durch eine **nicht-negative** reelle Zahl PR_i modelliert, dem so genannten **Page-Rank** von i. Der Wert PR_i soll die Qualität (im Sinne von „Renommee“ oder „Ansehen“) der Webseite i widerspiegeln und umso größer sein, je höher das Renommee der Webseite i ist. Der **Peer-Review** Ansatz wird verfolgt:

PR_i
Page-Rank
Peer-Review

> Das Renommee (und damit der Wert PR_j) einer Webseite j wird als hoch bewertet, wenn viele Webseiten i mit hohem Page-Rank PR_i einen Link auf die Seite j enthalten.

Für einen hohen Page-Rank PR_i ist es im Allgemeinen nicht ausreichend, wenn viele Webseiten einen Hyperlink auf die Seite i besitzen, vielmehr müssen unter den „Peers“ der Seite i viele Seiten *mit hohem Renommee* auf i zeigen. Die Werte PR_i werden daher so gewählt, dass Folgendes gilt:

> Eine Webseite i mit a_i ausgehenden Links „vererbt“’ ihren Page-Rank an alle Webseiten j mit $(i, j) \in E$ zu gleichen Teilen, d.h. gibt den Anteil $\frac{\text{PR}_i}{a_i}$ ab.

Demgemäß erfüllt der Page-Rank PR_j (aus Sicht des Peer-Review) die Gleichungen

$$\text{PR}_j \;=\; \sum_{i \in \text{Vor}_{\text{WEB}}(j)} \frac{\text{PR}_i}{a_i} \quad \text{und} \quad \text{PR}_j \geqslant 0 \tag{6.1}$$

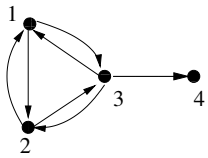
für alle $j \in V$.

Wir müssen uns im nächsten Abschnitt überlegen, ob das lineare Gleichungssystem (6.1) mit den Nebenbedingungen $\mathrm{PR}_1 \geqslant 0, \ldots, \mathrm{PR}_n \geqslant 0$ eine sinnvolle „Definition“ von Renommee erreicht. Man beachte auch, dass wir zum jetzigen Zeitpunkt noch nicht einmal wissen, ob das Gleichungssystem lösbar ist und wenn ja, ob es nur eindeutige Lösungen gibt und ob die Komponenten der Lösung nicht-negative Komponenten besitzen: Der „Page-Rank-Ansatz“ scheint interessant zu sein, aber wir müssen noch viele Fragen beantworten.

6.2.1. Der Dämpfungsfaktor

Angenommen, das lineare Gleichungssystem (6.1) besitzt eine eindeutige Lösung PR. Lässt sich PR als ein Maß für Renommee auffassen?

Ein erstes Problem sind Knoten mit Aus-Grad 0, da solche Knoten ihren Page-Rank nicht an andere Knoten weitervererben und daher zu Werten PR_i führen können, die kein sinnvolles Maß für die Bedeutung einer Webseite liefern. Betrachte beispielsweise den folgenden Graphen $\textsc{Web} = (V, E)$:

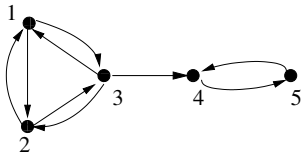


Die einzigen Werte $\mathrm{PR}_1, \mathrm{PR}_2, \mathrm{PR}_3, \mathrm{PR}_4 \in \mathbb{R}$, welche das Gleichungssystem (6.1) erfüllen, sind $\mathrm{PR}_1 = \mathrm{PR}_2 = \mathrm{PR}_3 = \mathrm{PR}_4 = 0$. Diese Werte sind aber leider schlicht nutzlos.

Senke

Im Folgenden werden **Knoten vom Aus-Grad 0** auch **Senken** genannt. Zur Bestimmung des Page-Ranks betrachtet man in der Regel nur **Graphen ohne Senken**, d.h. gerichtete Graphen, bei denen jeder Knoten einen Aus-Grad $\geqslant 1$ hat. Natürlich gibt es keine Garantie, dass der Web-Graph keine Senken besitzt. Die Autoren von [3, 22] schlagen zwei Möglichkeiten vor, den Web-Graphen in einen Graphen ohne Senken zu transformieren: Die eine Möglichkeit ist, von jeder Senke Kanten zu *allen* Knoten hinzuzufügen. Die andere Möglichkeit ist, alle Senken zu löschen und dies rekursiv so lange zu tun, bis ein Graph übrig bleibt, der keine Senke besitzt. Wir nehmen im Folgenden an, dass eine dieser beiden Transformationen durchgeführt wurde und der Web-Graph keine Senke besitzt.

Zwar können wir annehmen, dass der Webgraph keine Senken besitzt, aber auch eine Menge $U \subseteq V$ von Webseiten, die keinen Hyperlink zu einer Seite außerhalb von U besitzen, bleibt für den Vererbungsansatz von (6.1) problematisch. Betrachte zum Beispiel den folgenden Graphen $\textsc{Web} = (V, E)$:



Man kann sich leicht davon überzeugen, dass Werte $\mathrm{PR}_1, \mathrm{PR}_2, \mathrm{PR}_3, \mathrm{PR}_4, \mathrm{PR}_5 \in \mathbb{R}$ genau dann die Gleichung (6.1) erfüllen, wenn $\mathrm{PR}_1 = \mathrm{PR}_2 = \mathrm{PR}_3 = 0$ und $\mathrm{PR}_4 = \mathrm{PR}_5$ ist. Wie auch im vorigen Beispiel spiegeln diese Werte nicht das Renommee der einzelnen Webseiten wider.

Um dieses Problem zu vermeiden, wird die Vererbung von PR_i auf die Nachfolgeseiten j mit $(i, j) \in E$ um einen **Dämpfungsfaktor** d mit $0 \leqslant d < 1$ abgeschwächt. Dies wird in der folgenden

Definition präzisiert. Hier fordern wir auch eine anscheinend selbstverständliche Eigenschaft, nämlich dass alle Page-Rank-Werte PR_i nicht-negative reelle Zahlen sind: Negative Werte taugen nicht zur Messung des Renommees.

Definition 6.1 (Page-Rank-Eigenschaft). Sei $\textsc{Web} = (V, E)$ ein gerichteter Graph ohne Senken. Es gelte $V = \{1, \ldots, n\}$ für eine positive natürliche Zahl n.

(a) Sei d eine reelle Zahl mit $0 \leqslant d < 1$. Die Zahl d wird im Folgenden **Dämpfungsfaktor** genannt. Dämpfungsfaktor

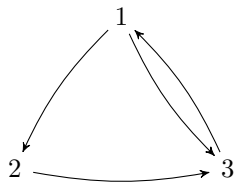
(b) Ein Zeilenvektor $\mathrm{PR} = (\mathrm{PR}_1, \ldots, \mathrm{PR}_n) \in \mathbb{R}^n$ hat die **Page-Rank-Eigenschaft** (bzgl. d), wenn für alle $j \in V$ gilt: Page-Rank-Eigenschaft

$$\mathrm{PR}_j \;=\; \frac{1-d}{n} \;+\; d \cdot \sum_{i \in \mathrm{Vor}_{\textsc{Web}}(j)} \frac{\mathrm{PR}_i}{a_i} \quad \text{und} \quad \mathrm{PR}_j \geqslant 0. \tag{6.2}$$

Wir erinnern daran, dass $a_i = \text{Aus-Grad}_{\textsc{Web}}(i)$ und $\mathrm{Vor}_{\textsc{Web}}(j) = \{i \in V : (i,j) \in E\}$.

Beachte: Für den (nicht erlaubten) Dämpfungsfaktor $d = 1$ erhält man das Gleichungssystem (6.1). Für den Dämpfungsfaktor $d = 0$ ist $\mathrm{PR}_1 = \mathrm{PR}_2 = \ldots = \mathrm{PR}_n = \frac{1}{n}$.
In [3] wird empfohlen, den Wert $d = 0.85 = \frac{17}{20}$ zu wählen.

Beispiel 6.2. Zur Veranschaulichung der Page-Rank-Eigenschaft betrachten wir den Dämpfungsfaktor $d = \frac{1}{2}$ und den folgenden Graphen $\textsc{Web} = (V, E)$:



Wir suchen einen Vektor $\mathrm{PR} = (\mathrm{PR}_1, \mathrm{PR}_2, \mathrm{PR}_3)$ von reellen Zahlen, der die Page-Rank-Eigenschaft bzgl. $d = \frac{1}{2}$ hat, d.h. es gilt:

(1) $\mathrm{PR}_1 = \frac{1}{2\cdot 3} + \frac{1}{2} \cdot \frac{\mathrm{PR}_3}{1}$

(2) $\mathrm{PR}_2 = \frac{1}{2\cdot 3} + \frac{1}{2} \cdot \frac{\mathrm{PR}_1}{2}$

(3) $\mathrm{PR}_3 = \frac{1}{2\cdot 3} + \frac{1}{2} \cdot \left(\frac{\mathrm{PR}_1}{2} + \frac{\mathrm{PR}_2}{1}\right)$.

Wir können Werte für PR_1, PR_2 und PR_3 finden, indem wir das lineare Gleichungssystem lösen, das aus den folgenden drei Gleichungen besteht:

(1) $1 \cdot \mathrm{PR}_1 \;-\; \frac{1}{2} \cdot \mathrm{PR}_3 \;=\; \frac{1}{6}$

(2) $-\frac{1}{4} \cdot \mathrm{PR}_1 \;+\; 1 \cdot \mathrm{PR}_2 \;=\; \frac{1}{6}$

(3) $-\frac{1}{4} \cdot \mathrm{PR}_1 \;-\; \frac{1}{2} \cdot \mathrm{PR}_2 \;+\; 1 \cdot \mathrm{PR}_3 \;=\; \frac{1}{6}$

Die Auflösung dieses linearen Gleichungssystems (z.B. mittels **Gauß-Elimination**) liefert die eindeutige Lösung

$$\mathrm{PR}_1 = \tfrac{14}{39}, \quad \mathrm{PR}_2 = \tfrac{10}{39}, \quad \mathrm{PR}_3 = \tfrac{15}{39}.$$

□Ende Beispiel 6.2

6.2.2. Die Page-Rank Matrix $P_d(\mathbf{Web})$

Die Situation zum jetzigen Zeitpunkt ist alles andere als übersichtlich:

Fragen 6.3:

(?) Ist das Gleichungssystem (6.2) mit den Nebenbedingungen $\mathrm{PR}_1 \geqslant 0, \ldots, \mathrm{PR}_n \geqslant 0$ überhaupt lösbar, besitzt es eine eindeutige Lösung?

(?) Wenn wir Glück haben, dann gibt es eindeutige Lösungen. Aber wie berechnet man Lösungen für ein System von mehreren Milliarden Gleichungen mit mehreren Milliarden Unbekannten? Die Gaußsche Eliminierung ist für ein Gleichungssystem dieser Dimension viel zu langsam!

(?) Ist der Page-Rank überhaupt ein sinnvolles Maß für das Renommee von Webseiten?

Irrfahrt
Random Walk
neuer Page-Rank

Fangen wir mit der letzten Frage an: Warum beobachten wir nicht einfach einen „Zufallssurfer“, also einen zufällig im Web herumspringender Surfer und notieren, wie häufig die Webseite i in einer unbeschränkt langen **Irrfahrt** (engl: **Random Walk**) erreicht wird. Dann ist es sinnvoll, PR_i^* als die relative Häufigkeit zu definieren, mit der Seite i besucht wird. Wir erhalten ein zweites Maß für das Renommee, nämlich den „**neuen Page-Rank-Vektor**“

$$\mathrm{PR}^* := (\mathrm{PR}_1^*, \ldots, \mathrm{PR}_n^*).$$

Wir stellen dem „alten Page-Rank“ PR ein durchaus sinnvolles Konkurrenzmaß für Renommee gegenüber. Damit scheinen wir nur weitere Fragen zu produzieren, wenn aber alter und neuer Page-Rank übereinstimmen, dann haben wir sehr starke Indizien erhalten, dass PR sinnvoll ist. Und vielleicht liefert die (hoffentlich zutreffende) Gleichheit $\mathrm{PR} = \mathrm{PR}^*$ auch die Chance einer schnellen Berechnung ...

Schauen wir uns den neuen Page-Rank genauer an. Wie soll sich der Zufallssurfer verhalten, wenn er den Knoten i in seiner Irrfahrt erreicht hat? Wir legen die Wahrscheinlichkeit $p_{i,j}$ fest mit der unser Surfer vom Knoten i auf den Knoten j springt und zwar soll der Zufallssurfer

- die Option „Webgraph“ mit Wahrscheinlichkeit d wählen.
  - Darauffolgend ist einer der $a_i = \text{Aus-Grad}_{\mathrm{WEB}}(i)$ ausgehenden Hyperlinks mit Wahrscheinlichkeit $\frac{1}{a_i}$ auszuwürfeln.
- Die Option „Hüpfen“ soll mit Wahrscheinlichkeit $(1-d)$ gewählt werden.
  - Darauffolgend ist eine der n Webseiten mit Wahrscheinlichkeit $\frac{1}{n}$ auszuwürfeln.

Für alle $i, j \in V$ gibt also

$$p_{i,j} := \begin{cases} \dfrac{1-d}{n} + \dfrac{d}{a_i} & \text{falls } (i,j) \in E, \\ \dfrac{1-d}{n} & \text{falls } (i,j) \notin E \end{cases} \tag{6.3}$$

die Wahrscheinlichkeit an, mit der der Zufallssurfer in einem Schritt von Seite i zu Seite j wechselt. (Zur Erinnerung: Wir haben angenommen, dass der Webgraph keine Senke besitzt, es gilt also stets $a_i > 0$.)

Definition 6.4. Sei $\pi \in \mathbb{R}^n$ ein Vektor. Dann heißt π heißt eine **Verteilung** (auf $\{1,\dots,n\}$), falls gilt Verteilung

$$\sum_{i=1}^{n} \pi_i = 1 \text{ und } \pi_i \geqslant 0 \text{ für alle } i \in \{1,\dots,n\}.$$

Der Vektor $\pi := (p_{i,1},\dots,p_{i,n})$ beschreibt somit, wie es weitergeht, wenn der Surfer den Knoten i erreicht hat. Beachte, dass π eine Verteilung ist, denn es ist stets $p_{i,j} \geqslant 0$ und

$$\sum_{j=1}^{n} p_{i,j} = \sum_{j=1}^{n} \frac{1-d}{n} + \sum_{j:(i,j)\in E} \frac{d}{a_i} = 1 - d + a_i \cdot \frac{d}{a_i} = 1.$$

Die Wahrscheinlichkeiten $p_{i,j}$, mit denen sich der Zufallssurfer von Knoten zu Knoten bewegt, lassen sich kompakt durch die Page-Rank-Matrix darstellen.

Definition 6.5 (Die Page-Rank-Matrix P_d(Web)).
Sei $d \in \mathbb{R}$ mit $0 \leqslant d < 1$, sei $n \in \mathbb{N}_{>0}$ und sei $\text{WEB} = (V,E)$ mit $V = \{1,\dots,n\}$ ein gerichteter Graph ohne Senke. Für jedes $i \in V$ sei $a_i := \text{Aus-Grad}_{\text{WEB}}(i)$. Die **Page-Rank-Matrix** ist die $n\times n$-Matrix Page-Rank-Matrix $P_d(\text{WEB})$

$$P_d(\text{WEB}) \;:=\; \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix},$$

wobei für alle $i,j \in V$ der Eintrag in Zeile i und Spalte j der in Gleichung (6.3) festgelegte Wert $p_{i,j}$ ist. Wir schreiben auch kurz $(p_{i,j})_{i,j=1,\dots,n}$, um die Matrix $P_d(\text{WEB})$ zu bezeichnen.

Beispiel 6.6. Für den Dämpfungsfaktor $d = \frac{1}{2}$ und den Graphen WEB aus Beispiel 6.2 ist beispielsweise $p_{1,1} = \frac{1}{6}$, $p_{1,2} = \frac{1}{6} + \frac{1}{4} = \frac{5}{12}$, $p_{2,3} = \frac{1}{6} + \frac{1}{2} = \frac{2}{3}$ und insgesamt

$$P_d(\text{WEB}) \;=\; \begin{pmatrix} \frac{1}{6} & \frac{5}{12} & \frac{5}{12} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}.$$

Definition 6.7. Eine $n \times n$-Matrix P heißt **stochastisch**, wenn stochastisch

(1) $P_{i,j} \geqslant 0$ für alle $i,j \in \{1,\dots,n\}$, und

(2) für jede Zeile $i \in \{1,\dots,n\}$ gilt: $\sum_{j=1}^{n} P_{i,j} = 1$.

Die Matrix P ist also genau dann stochastisch, wenn jede Zeile eine Verteilung ist. Insbesondere ist die Matrix $P_d(\text{WEB})$ der Übergangswahrscheinlichkeiten eine stochastische Matrix.

6.3. Markov-Ketten

Wie hängen alter und neuer Page-Rank zusammen? Um diese Frage zu beantworten, konzentrieren wir uns zuerst auf die wesentlichen Eigenschaften von Irrfahrten, nämlich den Graphen, in dem

die Irrfahrten unternommen werden, und die Matrix der Übergangswahrscheinlichkeiten.

Markov-Kette

Definition 6.8. $G = (V, E)$ sei ein gerichteter Graph mit Knotenmenge $V = \{1, \ldots, n\}$. Eine (homogene) **Markov-Kette** wird durch das Paar (G, P) beschrieben, falls folgendes gilt:

(a) G hat keine Senke, d.h. Aus-Grad$_G(v) > 0$ gilt für alle Knoten v von G.

(b) Die Matrix P ist eine stochastische Matrix mit n Zeilen und n Spalten und $P_{i,j} > 0$ gilt genau dann, wenn (i, j) eine Kante in G ist.

Graph der Kette
Übergangsmatrix
Zustand

Der **Graph der Kette** ist G und P ihre **Übergangsmatrix**, die Knoten von G nennt man auch **Zustände**.

Beispiel 6.9. (Der Page-Rank). Sei WEB der Webgraph und sei $P_d(\text{WEB})$ die Übergangsmatrix der Webkette. Der Graph der Webkette ist der vollständige, gerichtete Graph $\vec{K}_n = (V, E_n)$ mit Knotenmenge $V = \{1, \ldots, n\}$ und Kantenmenge $E_n = \{\, (u, v) \;:\; u, v \in \{1, \ldots, n\} \,\}$. Die

Webkette $\mathcal{W}$

Webkette $\mathcal{W}$ wird also durch das Paar

$$\mathcal{W} := (\vec{K}_n, P_d(\text{WEB}))$$

beschrieben. Wir möchten die Perspektive der Markov-Ketten einnehmen, um die in 6.3 aufgeworfenen Fragen zu beantworten. □ Ende von Beispiel 6.9

Sei $\mathcal{M} = (G, P)$ eine Markov-Kette mit dem Graphen $G = (V, E)$. Wir nehmen an, dass die von $\mathcal{M}$ erzeugte Irrfahrt mit Wahrscheinlichkeit π_i im Zustand $i \in V$ beginnt. Nach k Schritten der Irrfahrt befindet sich die Kette dann im Zustand $X_k(\pi)$. Die Zufallsvariable $X_k(\pi)$ stimmt also mit dem Zustand überein, den wir nach einer Irrfahrt der Länge k erreichen, wenn wir im Knoten i mit Wahrscheinlichkeit π_i beginnen. Insbesondere gilt $X_0(\pi) = i$ mit Wahrscheinlichkeit π_i.

Die Kette (G, P) erzeugt somit eine unendlich lange Irrfahrt durch den Graphen G, die durch die unendliche Folge

$$X_0(\pi) := \pi, X_1(\pi), X_2(\pi), \ldots, X_k(\pi), \ldots$$

beschrieben wird.

Welche Eigenschaften einer Markov-Kette sind bemerkenswert?

1. Eine Markov-Kette erinnert sich nur an den letzten Schritt: Der zu jedem Zeitpunkt $k > 0$ angenommene Zustand $X_k(\pi)$ hängt nur vom Vorgänger-Zustand $X_{k-1}(\pi)$ ab. Markov-Ketten modellieren somit Phänomene, für die die Zukunft von der Gegenwart, aber nicht von der Vergangenheit abhängt.

   Ordnung

   Man sagt deshalb auch, dass Markov-Ketten die **Ordnung** Eins besitzen. Man kann auch den Begriff einer Markov-Kette der Ordnung j definieren. In einer solchen Kette hängt $X_k(\pi)$ von den j Vorgängerzuständen $X_{k-1}(\pi), \ldots, X_{k-j}(\pi)$ ab. Wir arbeiten aber nur mit Ketten der Ordnung 1.

2. Die Übergangswahrscheinlichkeiten ändern sich *nicht* mit der Zeit, sondern werden allein von der zeit-unabhängigen Übergangsmatrix P bestimmt. Man sagt, dass die Kette **homogen** ist.

   homogen

6.3.1. Beispiele

Beispiel 6.10. (**Irrfahrten auf ungerichteten Graphen**.)
Viele Mähroboter, die ohne GPS-Ortung arbeiten, mähen den Rasen nach dem Zufallsprinzip. Der Roboter fährt gerade Strecken über den Rasen und wendet dann am Rand zufällig in einem von endlich vielen Winkeln. Wird tatsächlich jede Stelle des Rasens hochwahrscheinlich gemäht?

Dazu führen wir eine Irrfahrt auf einem ungerichteten, zusammenhängenden Graphen $G = (V, E)$ durch: Sei d_v die Anzahl der Nachbarn von v. Hat die Irrfahrt den Knoten v erreicht, dann wird die Irrfahrt mit einem Nachbarn v_i ($1 \leqslant i \leqslant d_v$) fortgesetzt, wobei v_i mit Wahrscheinlichkeit $1/d_v$ gewählt wird. Die Markov-Kette (G', P) dieser Irrfahrt besitzt

1. den Graphen $G' = (V, E')$, wobei
   - G und G' dieselbe Knotenmenge V besitzen,
   - eine gerichtete Kante (i, j) genau dann in G' vorkommt, wenn $\{i, j\}$ eine (ungerichtete) Kante von G ist,
2. und die Übergangsmatrix P mit

$$P_{i,j} := \begin{cases} \frac{1}{d_i} & \text{falls } \{i,j\} \text{ eine Kante von } G \text{ ist,} \\ 0 & \text{sonst.} \end{cases}$$

Was möchte man gern wissen?

(?) Ab welcher Länge wird eine Irrfahrt wahrscheinlich alle Knoten von G besucht haben?
- Die erwartete Länge einer Irrfahrt, die alle Knoten des Graphen $G = (V, E)$ besucht, ist höchstens $2|V| \cdot |E|$. (Dies wird zum Beispiel in der Veranstaltung „Effiziente Algorithmen“ gezeigt.) Man sagt auch, dass die „Cover-Time“ des Graphen durch $2|V| \cdot |E|$ beschränkt ist.
- Für reguläre Graphen, also für Graphen in denen jeder Knoten dieselbe Anzahl von Nachbarn besitzt, beträgt die Cover-Time höchstens $4 \cdot |V|^2$.

(?) Mit welcher relativen Häufigkeit wird ein Knoten $i \in V$ in einer genügend langen Irrfahrt besucht? Siehe dazu Beispiel 6.39.

Beispiel 6.11. (**Gambler's Ruin**) Wir modellieren ein faires Glücksspiel eines Spielers gegen eine Spielbank: In jeder Runde des Glücksspiels beträgt der Einsatz 1 €, der Einsatz des Verlierers geht an den Gewinner. Die Spielbank hat ein Kapital von N €, das Startkapital des Spielers ist K. Wir setzen $M := K + N$, verwenden die Zustände $0, \ldots, M$ und erlauben Übergänge vom Zustand $0 < i < M$ zu den Zuständen $i - 1$ und $i + 1$ mit der Wahrscheinlichkeit von jeweils $1/2$. Während der Zustand 0 für den Ruin des Spielers steht, nehmen wir im Zustand M an, dass der Spieler die Bank gesprengt hat: In den Zuständen 0 und M verbleibt man deshalb mit Wahrscheinlichkeit 1. Solche Zustände nennt man auch **absorbierend**. Unsere Markov-Kette (G, P) besitzt also

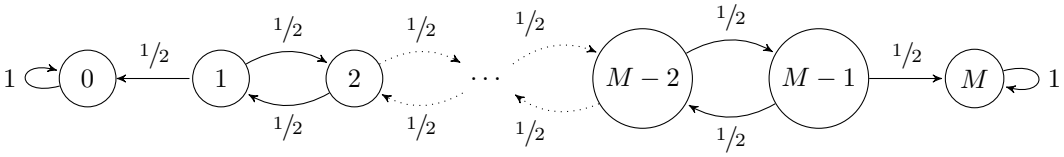
Gambler's Ruin

absorbierend

1. einen Graphen $G = (V, E)$ mit
   - Knotenmenge $V = \{0, 1, \ldots, M\}$,
   - den gerichteten Kanten $(i, i + 1), (i, i - 1)$ für $i = 1, \ldots, M - 1$ sowie den Kanten $(0, 0), (M, M)$,

2. und die Übergangsmatrix P mit

$$P_{i,j} := \begin{cases} 1 & \text{falls } i = j = 0 \text{ oder } i = j = M, \\ \frac{1}{2} & \text{falls } i \in \{1, \dots, M-1\} \text{ und } |i-j| = 1, \\ 0 & \text{sonst.} \end{cases}$$



Wie groß ist die Wahrscheinlichkeit s_K, dass der Spieler die Bank sprengt, wenn Casino und Spieler bis zum bitteren Ende spielen und wenn der Spieler das Startkapital K besitzt? Man kann zeigen, dass $s_K = K/M$ gilt und erstaunlicherweise sind die Chancen die Bank zu sprengen also selbst dann durchaus beträchtlich, wenn K sehr viel kleiner als N ist.

Kein Casino kann sich derartig große Misserfolgswahrscheinlichkeiten leisten und deshalb sind alle Spiele zwangsläufig unfair. Ist $p < 1/2$ die Wahrscheinlichkeit, dass der Spieler eine Runde gewinnt und $q := 1 - p$ die Komplementärwahrscheinlichkeit, dann kann gezeigt werden, dass die Bank mit Wahrscheinlichkeit

$$s_K = \frac{\left(\frac{q}{p}\right)^K - 1}{\left(\frac{q}{p}\right)^M - 1}$$

gesprengt wird. Beachte, dass $q = 1 - p > p$, denn $p < 1/2$ nach Annahme und

$$s_K = \frac{\left(\frac{q}{p}\right)^K - 1}{\left(\frac{q}{p}\right)^M - 1} \leqslant \frac{\left(\frac{q}{p}\right)^K}{\left(\frac{q}{p}\right)^M} = \left(\frac{q}{p}\right)^{-N}.$$

Die Erfolgswahrscheinlichkeit s_K für den Spieler fällt exponentiell mit N: Machen Sie einen großen Bogen um „Glücksspiele“!

Im nächsten Beispiel wird ein würfelnder Algorithmus für 2-SAT beschrieben. Auch hier muss eine Irrfahrt auf der eindimensionalen Kette untersucht werden.

Beispiel 6.12 (Ein effizienter Algorithmus für 2-SAT). Die KNF-Formel

$$\alpha = \bigwedge_{i=1}^{m} \left(\ell_{i,1} \vee \ell_{i,2}\right)$$

mit höchstens 2 Literalen pro Disjunktionsterm sei gegeben. Ist α erfüllbar? Wir nehmen an, dass α genau n Variablen besitzt und dass α von der Belegung $x^* \in \{0,1\}^n$ erfüllt wird. Das Erfüllbarkeitsproblem für k-KNF-Formeln (KNF-Formeln mit höchstens k Literalen pro Disjunktionsterm) ist vermutlich sehr schwierig für $k \geqslant 3$, für $k = 2$ hingegen gibt es effiziente Algorithmen.

Ein würfelnder Algorithmus für 2-SAT:

1. Bestimme eine zufällige Belegung $x \in \{0,1\}^n$.

   # Selbst wenn α erfüllbar ist, wird x i.A. keine erfüllende Belegung sein.

2. Wiederhole genügend oft:

   (a) Bestimme irgendeinen Disjunktionsterm $D = \ell \vee \ell'$ von α, der von x falsifiziert wird.

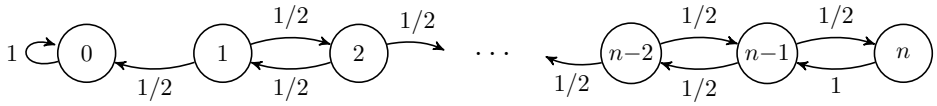
   (b) Wähle $L \in \{\ell, \ell'\}$ zufällig aus: Modifiziere x, so dass L erfüllt wird, d.h. „flippe“ die zu L gehörende Variable.

   # x^* erfüllt D und deshalb auch ℓ oder ℓ'. Deshalb gilt mit Wahrscheinlichkeit $\geqslant \frac{1}{2}$
   # (bzw. $\leqslant \frac{1}{2}$), dass x und x^* in einer Variablen mehr (bzw. weniger) übereinstimmen als vorher.

Frage: Wie häufig muss Schritt 2 wiederholt werden, bis eine erfüllende Belegung gefunden wird? Um diese Frage zu beantworten, verfolgen wir den Hammingabstand $H(x, x^*)$ zwischen der aktuellen Belegung x und der erfüllenden Belegung x^*, wobei

$$H(x, x^*) := \text{die Anzahl der Positionen } i \text{ mit } x_i \neq x_i^*$$

ist. Beachte, dass $H(x, x^*) = 0$ genau dann gilt, wenn $x = x^*$ gilt. Wir analysieren die folgende Markov-Kette:



- Zustand i entspricht dem aktuellen Hammingabstand i. Insbesondere haben wir x^* im Zustand 0 gefunden.
- Zustandsübergänge protokollieren den Verlauf des Algorithmus.

In den Übungsaufgaben wird gezeigt, dass der Algorithmus die erfüllende Belegung x^* nach einer erwarteten Anzahl von höchstens n^2 Wiederholungen findet, es sei denn, dass zwischenzeitlich eine andere erfüllende Belegung gefunden wird.

Beispiel 6.13. (Die Ehrenfest-Kette). Zwei Substanzen sind durch eine Membran getrennt, aber Moleküle wandern über die Membran zwischen den beiden Substanzen. Gibt es so etwas wie einen Gleichgewichtszustand? Ehrenfest-Kette

Der Physiker Paul Ehrenfest (1880-1933) hat die folgende Modellierung vorgeschlagen. Anfänglich befinden sich l_0 Partikel in einer linken Urne und $r_0 = N - l_0$ Partikel in einer rechten Urne. In jedem Schritt wird ein Partikel gleichverteilt, also mit Wahrscheinlichkeit $1/N$ ausgewählt und in die jeweils andere Urne verschoben.

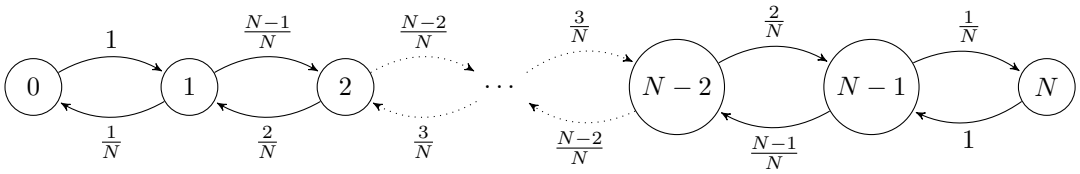
Diesmal besitzt unsere Markov-Kette (G, P) deshalb

1. einen Graphen $G = (V, E)$ mit
   - Knotenmenge $V = \{0, 1, \ldots, N\}$: für jede mögliche Partikelzahl ℓ in der linken Urne gibt es den Zustand $\ell \in V$,

- den Kanten $i \to i+1$ und $i \to i-1$ für $i = 1, \ldots, N-1$ sowie den Kanten $0 \to 1$, $N \to N-1$

2. und die Übergangsmatrix P mit

$$P_{i,j} := \begin{cases} \frac{i}{N} & \text{falls } j = i-1, \\ 1 - \frac{i}{N} & \text{falls } j = i+1, \\ 0 & \text{sonst.} \end{cases}$$



Angenommen, wir lassen Partikel „hinreichend oft“ wandern: Mit welcher Wahrscheinlichkeit besitzt die linke Urne genau ℓ Partikel? Wir kommen auf diese Frage in Beispiel 6.25 zurück.

Warteschlangen

Beispiel 6.14. (Warteschlangen). Wir möchten die Länge der Warteschlange an einer Supermarktkasse simulieren. Dazu wählen wir die Zustandsmenge $V = \mathbb{N}$, wobei der Zustand $i \in V$ bedeutet, dass i Kunden warten. Wir nehmen an, dass zu jedem Zeitpunkt genau k neue Kunden mit Wahrscheinlichkeit $2^{-(k+1)}$ erscheinen und dass mit Wahrscheinlichkeit 1 genau ein Kunde eine nicht-leere Schlange verlässt[2].

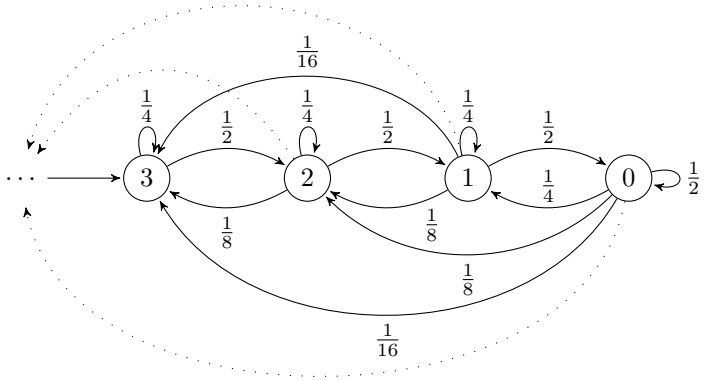
Mit Wahrscheinlichkeit $1/2$ nimmt also die Länge einer nicht-leeren Schlange um Eins ab, denn ein Kunde wird definitiv abkassiert und mit Wahrscheinlichkeit $1/2$ kommen keine neuen Kunden. Mit Wahrscheinlichkeit $1/4$ bleibt die Länge einer nicht-leeren Schlange unverändert, denn ein neuer Kunde kommt mit eben dieser Wahrscheinlichkeit an.

Die Markov-Kette (G, P) besitzt

1. den Graphen $G = (V, E)$ mit
   - Knotenmenge $V = \mathbb{N}$,
   - Kanten führen von i nach j für alle $j \geqslant \max\{0, i-1\}$,
2. und die Übergangsmatrix P mit

$$P_{i,j} := \begin{cases} 2^{-(k+2)} & \text{falls } i \geqslant 1, j \geqslant i-1, k = j-i, \\ 2^{-(k+1)} & \text{falls } i = 0, j \geqslant i, k = j-i, \\ 0 & \text{sonst.} \end{cases}$$

[2]Wenn mehrere Kassen simuliert werden sollen, dann muss natürlich die Zahl abkassierter Kunden entsprechend erhöht werden.



Man beachte, dass die Kette unendlich viele Zustände besitzt und deshalb keine Markov-Kette gemäß Definition 6.8 ist. Natürlich passen nur beschränkt viele Kunden in einen Supermarkt, aber diese Kette wird die wesentlichen Eigenschaften einer realen Warteschlange zumindest angenähert wiedergeben, solange die Übergangsmatrix P empirisch festgestellte Übergangswahrscheinlichkeiten beschreibt.

Angenommen, wir beobachten eine Warteschlange mit den obigen Parametern genügend lange: Mit welcher Wahrscheinlichkeit warten genau k Kunden? Siehe dazu Beispiel 6.40.

Beispiel 6.15. (Zeitreihen). Eine Zeitreihe ist eine zeitabhängige Folge Z_t von Datentupeln. Typische Beispiele sind Börsenkurse, die Arbeitslosenquote, das Bruttosozialprodukt, Wetterbeobachtungen, Wahlabsichtsbefragungen und verschiedenes mehr. Um Zeitreihen vorauszusagen, versucht man häufig eine Markov-Kette zu konstruieren, die die beobachtete Zeitreihe gut approximiert. Zeitreihen

Bemerkung 6.16. (Unendlich große Zustandsräume). Wir behandeln in Definition 6.8 nur Markov-Ketten mit endlich vielen Zuständen, haben aber in Beispiel 6.14 schon eine erste „illegale“ Markov-Kette mit unendlich vielen Zuständen kennengelernt. Wir besprechen hier einige interessante Eigenschaften von Irrfahrten auf dem d-dimensionalen Gitter $\mathbb{Z}^d$. Diese Bemerkung zeigt überraschende Eigenschaften von Irrfahrten auf unendlich großen Graphen, kann aber „gefahrlos“ übersprungen werden.

(a) Wir nehmen an, dass eine Irrfahrt auf dem d-dimensionalen Gitter $\mathbb{Z}^d$ stets im Ursprung beginnt. Hat die Irrfahrt einen Gitterpunkt erreicht, wechselt sie zu einem bestimmten der $2d$ benachbarten[3] Gitterpunkte mit Wahrscheinlichkeit $1/(2d)$. Für welche Dimensionen d ist der Ursprung $(0,\dots,0)$ **rekurrent**, d.h. wann wird eine unbeschränkt lange Irrfahrt hochwahrscheinlich unendlich oft zum Ursprung zurückkehren, und wann ist der Ursprung **transient**, wann kehrt die Irrfahrt hochwahrscheinlich also nur endlich oft zurück? Der Mathematiker George Polya hat gezeigt, dass der Ursprung für die Dimension 1 und 2 rekurrent ist und für alle anderen Dimensionen transient ist: Im Dreidimensionalen sieht man den Ursprung nur endlich oft und dann heißt es „Tschüss“. rekurrent transient

[3] Zwei Gitterpunkte $u, v \in \mathbb{Z}^d$ sind benachbart, wenn $\sum_{i=1}^{d} |u_i - v_i| = 1$.

(b) Wir betrachten Irrfahrten auf dem Gitter $\mathbb{Z}$, also den Fall $d = 1$ aus (a). Wieder nehmen wir an, dass die Irrfahrt im Nullpunkt beginnt.

- Sei $p_m(a, b)$ die Wahrscheinlichkeit, in genau m Schritten vom Punkt a aus den Punkt b zu erreichen. Man kann zeigen, dass

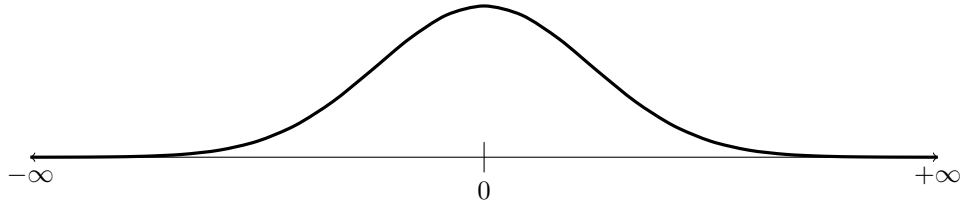
$$p_{2n}(0,0) \sim \frac{1}{\sqrt{\pi \cdot n}}$$

gilt[4].

- Die Position einer Irrfahrt zur Zeit n ist in einem Intervall der Länge $\sqrt{n}$ um den Nullpunkt fast gleichverteilt. Außerhalb dieses Intervalls fällt die Wahrscheinlichkeit exponentiell. Warum? Man kann zeigen, dass

$$p_n(0,a) \leqslant \exp(-a^2/(2n))$$

gilt.



Die Funktion $p_n(0, a) = \exp(-a^2/(2n))$.

- Sei q_n die Wahrscheinlichkeit, dass der Punkt 1 erst nach mehr als $2n$ Schritten besucht wird. Dann kann gezeigt werden, dass

$$q_n \sim \frac{1}{\sqrt{\pi \cdot n}}$$

gilt. Als Konsequenz überlegt man sich, dass die erwartete Zeit bis zum ersten Besuch der 1 *unendlich groß* ist, obwohl die 1 unendlich oft besucht wird: Zwar ist die Irrfahrt mit Wahrscheinlichkeit 1/2 bereits nach einem Schritt im Punkt 1, mit derselben Wahrscheinlichkeit entfernt sie sich aber auch!

6.3.2. Ein Schritt einer Markov-Kette

Wir möchten den neuem Page-Rank PR^* verstehen. Wir haben PR^*_i als die relative Häufigkeit definiert, mit der der Zufallssurfer die Seite i besucht. Was genau sind relative Häufigkeiten?

Anfangsverteilung

Sei (G, P) eine beliebige Markov-Kette. Angenommen, wir beginnen im Zustand i mit Wahrscheinlichkeit π_i. Wir sagen, dass (G, P) die **Anfangsverteilung** $\pi^{(0)} := \pi$ auf den Zuständen in $\{1, \ldots, n\}$ *besitzt*. Nach einem Schritt der Irrfahrt besitzt die Kette eine neue Verteilung $\pi^{(1)}$, nach k Schritten die Verteilung $\pi^{(k)}$. Die von uns gesuchten relativen Häufigkeiten stimmen dann überein mit der Verteilung

$$\pi^{(\infty)} := \lim_{k \to \infty} \pi^{(k)}.$$

Grenzverteilung zu π

Wenn Konvergenz vorliegt, sagen wir dass $\pi^{(\infty)}$ die **Grenzverteilung zu** π ist. Wir möchten den neuen Page-Rank durch $\mathrm{PR}^* := \pi^{(\infty)}$ definieren, müssen uns aber fragen, ob der Grenzwert

[4] Es gilt $a_n \sim b_n$ genau dann, wenn $\lim_{n\to\infty} a_n/b_n = 1$.

$\lim_{k\to\infty} \pi^{(k)}$ überhaupt existiert und ob verschiedene Anfangsverteilungen zu verschiedenen Grenzverteilungen führen. (Ist die Antwort auf die letzte Frage für die Webkette positiv, dann gehört der neue Page-Rank „in die Tonne“.)

Um den Grenzwert $\lim_{k\to\infty} \pi^{(k)}$ zu analysieren, müssen wir zuerst wissen wie die k-Schritt-Verteilungen $\pi^{(k)}$ aussehen. Wie bestimmt man $\pi^{(1)}$? Um diese Frage zu beantworten, erinnern wir an die Definition des Matrizenprodukts und an das Vektor-Matrix-Produkt.

Definition 6.17. (**Matrizenprodukt**). $X = (x_{i,j})$ und $Y = (y_{k,l})$ seien $N \times M$, bzw. $M \times R$ Matrizen reeller Zahlen. Dann ist die Produktmatrix $Z = X \cdot Y$ eine $N \times R$ Matrix $Z = (z_{i,l})$ mit

$$z_{i,l} = \sum_{j=1}^{M} x_{i,j} \cdot y_{j,l}.$$

Wie ist das Vektor-Matrix-Produkt $z = x \cdot Y$ definiert, wenn $x \in \mathbb{R}^n$ ein Zeilenvektor und $Y = (y_{k,l})$ eine $n \times n$-Matrix reeller Zahlen ist? Interpretiere den Vektor x als eine $1 \times n$-Matrix: Dann ist x nacheinander mit allen Spalten von Y zu multiplizieren. Es ist also

$$z_l = \sum_{j=1}^{n} x_j \cdot y_{j,l}.$$

Beispiel 6.18. Sei $P := P_d(\text{WEB})$ die Matrix aus Beispiel 6.6 und sei $x := (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Dann gilt:

$$\begin{aligned} x \cdot P &= \left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right) \cdot \begin{pmatrix} \frac{1}{6} & \frac{5}{12} & \frac{5}{12} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix} \\ &= \left(\tfrac{1}{3}\cdot\tfrac{1}{6} + \tfrac{1}{3}\cdot\tfrac{1}{6} + \tfrac{1}{3}\cdot\tfrac{2}{3}, \quad \tfrac{1}{3}\cdot\tfrac{5}{12} + \tfrac{1}{3}\cdot\tfrac{1}{6} + \tfrac{1}{3}\cdot\tfrac{1}{6}, \quad \tfrac{1}{3}\cdot\tfrac{5}{12} + \tfrac{1}{3}\cdot\tfrac{2}{3} + \tfrac{1}{3}\cdot\tfrac{1}{6}\right) \\ &= \left(\tfrac{1}{3}, \tfrac{1}{4}, \tfrac{5}{12}\right). \end{aligned}$$

Sei (G, P) eine Markov-Kette mit $G = (\{1, \ldots, n\}, E)$. Wir wählen einen Knoten gemäß der Anfangsverteilung π aus, wählen also den Knoten i mit Wahrscheinlichkeit $\pi_i^{(0)} := \pi_i$. Mit welcher Wahrscheinlichkeit befinden wir uns im Knoten j nach einem Schritt der Markov-Kette? Die Frage ist schnell beantwortet, denn wir befinden uns im Knoten j mit Wahrscheinlichkeit

$$\pi_j^{(1)} := \sum_{i=1}^{n} \pi_i^{(0)} \cdot P_{i,j} \stackrel{\text{toll!}}{=} (\pi^{(0)} \cdot P)_j. \tag{6.4}$$

Die Kette besitzt also nach einem Schritt die Verteilung $\pi^{(1)} = \pi^{(0)} \cdot P$. Und warum, bitte schön, ist das so toll? Weil wir auf das Vektor-Matrix Produkt geführt werden und mit Leichtigkeit sagen können, welche Verteilung die Kette nach k Schritten besitzt.

(a) Rekursionsanfang: Anfänglich besitzt die Kette die Verteilung $\pi^{(0)} := \pi$.

(b) Rekursionsschritt: Wenn die Kette nach k Schritten die Verteilung $\pi^{(k)}$ besitzt, dann besitzt sie nach $k+1$ Schritten die Verteilung $\pi^{(k+1)} := \pi^{(k)} \cdot P$.

Können wir die Verteilungen $\pi^{(k)}$ durch einen geschlossenen Ausdruck beschreiben?

Satz 6.19. (G, P) sei eine Markov-Kette. Wenn die Kette (G, P) anfänglich die Verteilung π besitzt, dann besitzt sie nach k Schritten die Verteilung

$$\pi^{(k)} = \pi \cdot P^k.$$

Wenn der Zufallssurfer also mit Wahrscheinlichkeit π_i im Knoten i beginnt, dann befindet er sich nach einer Irrfahrt der Länge k im Knoten j mit Wahrscheinlichkeit $(\pi \cdot P^k)_j$.

Beweis: Wir beweisen die Behauptung durch vollständige Induktion nach k. Der Induktionsanfang für $k = 0$ folgt aus $\pi^{(0)} := \pi = \pi \cdot P^0$. (Man beachte, dass P^0 die Einheitsmatrix ist.)

Induktionsschritt von k auf $k + 1$. Sei $k \in \mathbb{N}$ beliebig. Wir können die Induktionsannahme $\pi^{(k)} = \pi \cdot P^k$ voraussetzen. Der Rekursionsschritt besagt, dass $\pi^{(k+1)} = \pi^{(k)} \cdot P$ gilt. Also folgt

$$\pi^{(k+1)} = \pi^{(k)} \cdot P \overset{\text{Induktionsannahme}}{=} (\pi \cdot P^k) \cdot P \overset{\text{Assoziativität}}{=} \pi \cdot (P^k \cdot P) = \pi \cdot P^{k+1},$$

denn das Matrizenprodukt ist assoziativ. □

Bemerkung 6.20. Angenommen eine Markov-Kette (G, P) befindet sich anfänglich in der Verteilung π. Wie schnell – wenn überhaupt – konvergieren die Verteilungen

$$\pi^{(k)} = \pi \cdot P^k$$

Um solche Fragen beantworten zu können, brauchen wir Werkzeuge, die das Matrizenprodukt P^k und das Matrix-Vektorprodukt $\pi \cdot P^k$ schnell (und für uns komfortabel) berechnen. Für kleine Markov-Ketten ist das in https://matrixcalc.org/de/ bereitgestellte Online-Werkzeug völlig ausreichend. Für größere Ketten ist SymPy (http://docs.sympy.org/latest/tutorial/matrices.html) eine gute Wahl.

6.3.3. Die Grenzverteilung einer ergodischen Kette

Was ist der Zusammenhang zwischen altem und neuem Page-Rank? Diese Frage beschäftigt uns weiterhin.

Wir haben in Satz 6.19 festgestellt, dass die Kette (G, P) nach k Schritten die Verteilung $\pi \cdot P^k$ besitzt, wenn π die Anfangsverteilung der Kette ist. Dann ist klar, dass die Kette, wenn in π gestartet, zur Grenzverteilung $\lim_{k\to\infty} \pi \cdot P^k$ konvergiert, wenn denn der Grenzwert existiert.

Der neue Page-Rank ist aber nur dann sinnvoll, wenn die Kette dieselbe Grenzverteilung für alle Anfangsverteilungen π besitzt. Ketten mit dieser Eigenschaft schauen wir uns deshalb genauer an.

Grenzverteilung $\mathcal{G}(\mathcal{M})$ der Kette

Definition 6.21. $\mathcal{M} = (G, P)$ sei eine Markov-Kette. Dann heißt die Verteilung $\mathcal{G}(\mathcal{M})$ die **Grenzverteilung der Kette**, wenn für alle Anfangsverteilungen π gilt:

$$\lim_{k\to\infty} \pi \cdot P^k = \mathcal{G}(\mathcal{M}).$$

$\mathcal{G}(\mathcal{M})$ ist somit Grenzverteilung zu allen Anfangsverteilungen.

Schauen wir uns die linke Seite $\lim_{k\to\infty} \pi \cdot P^k$ genauer an. Wenn alle Grenzwerte existieren, dann können wir den Vektor π nach vorne ziehen:

$$\lim_{k\to\infty} \pi \cdot P^k = \pi \cdot \lim_{k\to\infty} P^k.$$

Wenn(!) zusätzlich alle Zeilen der „Grenzmatrix“ $\lim_{k\to\infty} P^k$ mit dem Vektor z übereinstimmen, dann gilt

$$\lim_{k\to\infty} \pi \cdot P^k = \pi \cdot \lim_{k\to\infty} P^k = \pi \cdot \begin{pmatrix} z \\ \vdots \\ z \end{pmatrix} = z \tag{6.5}$$

für jede Anfangsverteilung π.

Satz 6.22. Sei $\mathcal{M} = (G, P)$ eine Markov-Kette. Wenn die Grenzmatrix $\lim_{k\to\infty} P^k$ existiert und alle Zeilen übereinstimmen, dann gilt für jede Verteilung σ

$$\lim_{k\to\infty} \sigma \cdot P^k = \left(\lim_{k\to\infty} \left(P^k\right)_{1,1}, \ldots, \lim_{k\to\infty} \left(P^k\right)_{1,n} \right).$$

Insbesondere ist $\mathcal{G}(\mathcal{M}) = (\lim_{k\to\infty} \left(P^k\right)_{1,1}, \ldots, \lim_{k\to\infty} \left(P^k\right)_{1,n})$ die Grenzverteilung der Kette.

Die Zeilen der Grenzmatrix stimmen genau dann überein, wenn

$$\lim_{k\to\infty} \left(P^k\right)_{i,j} = \lim_{k\to\infty} \left(P^k\right)_{i',j} \tag{6.6}$$

für alle Zustände i, i' und j gilt. Was besagt diese Eigenschaft? Eine genügend lange Irrfahrt „vergisst“, ob sie im Zustand i oder im Zustand i' begonnen hat. Und wenn die Kette die Anfangszustände nicht vergisst? Dann gibt es verschiedene Grenzverteilungen zu verschiedenen Anfangsverteilungen! Aber längst nicht alle Markov-Ketten vergessen den Anfangszustand unendlich langer Irrfahrten, aber schöne Ketten wie unsere Webkette $\mathcal{W}$ könnten, oder besser sollten vergesslich sein. Wir fordern deshalb „Vergesslichkeit“ von einer „schönen“, sprich ergodischen Kette.

Definition 6.23. Eine Markov-Kette (G, P) mit $G = (V, E)$ ist **ergodisch**, wenn ergodisch

(a) die Grenzwahrscheinlichkeiten $\lim_{k\to\infty} \left(P^k\right)_{i,j}$ und $\lim_{k\to\infty} \left(P^k\right)_{i',j}$ für alle Knoten i, i' und j existieren und übereinstimmen sowie

(b) $\lim_{k\to\infty} \left(P^k\right)_{i,j} > 0$ für alle Knoten i, j gilt.

Wir haben schon argumentiert, dass Eigenschaft (a), also Vergesslichkeit, sinnvoll ist, um letztlich eine Grenzverteilung für alle Anfangsverteilungen zu erhalten. Warum aber haben wir Eigenschaft (b) gefordert? Ist diese Eigenschaft verletzt, dann gilt $\lim_{k\to\infty} \left(P^k\right)_{i,j} = 0$ für zwei Zustände i und j. Aus Eigenschaft (a) folgt dann, dass j von keinem Zustand mit positiver Wahrscheinlichkeit erreichbar ist: Der Zustand j ist bedeutungslos. Wir fordern also, dass ergodische Ketten keine bedeutungslosen Zustände besitzen.

Für ergodische Ketten existiert die Grenzmatrix $\lim_{k\to\infty} P^k$ und ihre Zeilen sind identisch. Also erhalten wir aus Satz 6.22:

Folgerung 6.24. Die Markov-Kette $\mathcal{M} = (G, P)$ sei ergodisch. Dann gilt für jede Verteilung π

$$\lim_{k\to\infty} \pi \cdot P^k = (\lim_{k\to\infty} \left(P^k\right)_{1,1}, \ldots, \lim_{k\to\infty} \left(P^k\right)_{1,n}).$$

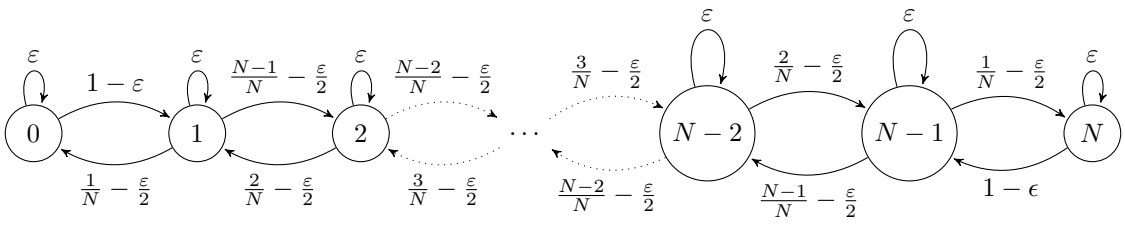
Insbesondere ist $\mathcal{G}(\mathcal{M}) = (\lim_{k\to\infty} \left(P^k\right)_{1,1}, \ldots, \lim_{k\to\infty} \left(P^k\right)_{1,n})$ die Grenzverteilung der Kette.

In ergodischen Markov-Ketten können wir somit $\mathrm{PR}^* := \mathcal{G}(\mathcal{M})$ setzen und der neue Page-Rank erscheint als ein sinnvolles Maß für Renommee. Aber ist die Webkette ergodisch? Zum Beispiel, wann ist Eigenschaft (a) in Definition 6.23 erfüllt?

Beispiel 6.25. (Grenzverteilung der Ehrenfest-Kette, vgl. Beispiel 6.13) Die Zustände $\lfloor N/2 \rfloor$ und $\lceil N/2 \rceil$ „sollten" in der Grenzverteilung die größte Wahrscheinlichkeit besitzen, die Wahrscheinlichkeiten für andere Partikelzahlen „sollten" innerhalb der Standardabweichung $\sqrt{N}$ langsam, außerhalb aber exponentiell schnell fallen. Mit anderen Worten, die Ehrenfest-Kette „sollte" eine von der Anfangsverteilung unabhängige Grenzverteilung besitzen.

Leider, leider liegt keine Konvergenz vor, denn in jedem Schritt findet ein Zustandswechsel von einem geraden zu einem ungeraden Zustand statt und umgekehrt. Gilt für eine Anfangsverteilung π zum Beispiel $\pi_{2i} = 1$ für irgendein i, dann befindet sich die Kette nach einer ungeraden Anzahl $2k+1$ von Schritten garantiert nicht im Zustand $2i$ und es ist $p_{2i}^{(2k+1)} = 0$ (für die Wahrscheinlichkeit $p_{2i}^{(2k+1)}$ nach $2k+1$ Schritten im Zustand $2i$ zu sein) : Die Folge $(p_{2i}^{(m)} : m \in \mathbb{N})$ besitzt unendlich viele Nullen und kann nicht gegen eine positive Zahl konvergieren.

Mutter Natur mag Konvergenz, deshalb ändern wir die Ehrenfest-Kette leicht: Mit einer kleinen Wahrscheinlichkeit $\varepsilon > 0$ wird *kein einziges* Partikel aus „seiner" Urne verschoben.



(Es muss $1/N - \varepsilon/2 \geqslant 0$ gelten, d.h. es ist $\varepsilon \leqslant 2/N$ zu fordern.) In Folgerung 6.38 erhalten wir später gute Nachrichten: Die neue Kette $\mathcal{M}$ besitzt eine Grenzverteilung $\mathcal{G}(\mathcal{M})$, die nicht von der Anfangsverteilung π abhängt und zwar gilt für alle $l \in \{0, \ldots, N\}$

$$\mathcal{G}(\mathcal{M})_l = \binom{N}{l} \cdot 2^{-N}. \tag{6.7}$$

Man sagt auch, dass $\mathcal{G}(\mathcal{M})$ „binomialverteilt" ist. □ Ende von Beispiel 6.25

Sei (G, P) eine ergodische Markov-Kette. Wenn G nicht stark zusammenhängend ist, dann gibt es zwei Knoten $i, j \in V$, so dass *kein* Weg im Graphen G von i nach j führt und $\lim_{k\to\infty} \left(P^k\right)_{i,j} = 0$ folgt, im Widerspruch zur Eigenschaft (b) einer ergodischen Markov-Kette. Also ist G stark zusammenhängend. Wie wir in Beispiel 6.25 gesehen haben genügt die Forderung nach starkem Zusammenhang nicht, um Konvergenz zu garantieren: Die Eigenschaft der Aperiodizität muß gefordert werden.

Definition 6.26. Sei G ein gerichteter Graph.

(a) G heißt genau dann **irreduzibel**, wenn G stark zusammenhängend ist. irreduzibel

(b) Ein Zustand $i \in V$ hat die **Periode** p, wenn die Längen aller Wege von i nach i durch p teilbar sind und p größtmöglich ist. Periode

(c) G heißt genau dann **aperiodisch**, wenn kein Zustand eine Periode $p > 1$ besitzt. aperiodisch

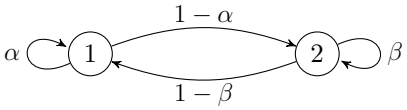
Wir haben uns gerade überlegt, dass ergodische Ketten (G, P) einen irreduziblen und aperiodischen Graphen besitzen. Überraschenderweise sind diese beiden Graph-Eigenschaften auch „hinreichend“, d.h. Ketten (G, P) mit einem irreduziblen und aperiodischen Graphen G sind ergodisch!

Satz 6.27. Eine Markov-Kette (G, P) ist genau dann ergodisch, wenn G irreduzibel und aperiodisch ist.

Wir beweisen Satz 6.27 nicht und verweisen stattdessen auf die Veranstaltung „Mathematik 3“.

Beispiel 6.28. (**Ergodisch oder nicht?**).

(a) Die Kette



ist ergodisch, wenn $0 < \alpha < 1$ und $0 < \beta < 1$ gilt. In diesem Fall ist die Periode beider Zustände nämlich 1 und die Kette ist irreduzibel. Die Kette ist sicherlich nicht ergodisch, wenn $\alpha = 1$ oder $\beta = 1$, denn dann ist Zustand 1 bzw. Zustand 2 isoliert. Wenn $\alpha = 0 = \beta$, dann hat jeder Zustand die Periode zwei und die Kette ist wiederum nicht ergodisch. Und wenn $\alpha = 0,\ 0 < \beta < 1$ oder $0 < \alpha < 1,\ \beta = 0$?

(b) Der Graph $\vec{K}_n$ der Webkette $\mathcal{W} = (\vec{K}_n, P_d(\text{WEB}))$ (siehe Beispiel 6.9) ist vollständig, besitzt also eine Kante (i, j) für alle Knoten i, j. Dann ist $\vec{K}_n$ sicherlich irreduzibel, aber auch aperiodisch, da wir ja in einem Schritt von i zu i zurückkehren können. Also ist die Webkette $\mathcal{W}$ ergodisch und besitzt eine Grenzverteilung $\mathcal{G}(\mathcal{W})$.

(c) Die Markov-Ketten $\mathcal{M} = (G', P)$ zu Irrfahrten auf einem zusammenhängenden, nichtbipartiten Graphen $G = (V, E)$ (siehe Beispiel 6.10) sind ergodisch. Warum? Zur Erinnerung: Es ist $G' = (V, \{(i, j) : \{i, j\} \in E\})$.

- Da der Graph G zusammenhängend ist, ist der Graph G' der Kette irreduzibel.
- Jeder Knoten u ist Teil eines Kreises in G' der Länge 2, wenn wir nämlich von u zu einem Nachbarn laufen und dann zurückkehren. Man kann zeigen, dass G genau dann nicht bipartit ist, wenn G einen Kreis ungerader Länge hat. Da G zusammenhängend, aber nicht bipartit ist, ist u auch Teil eines Kreises in H ungerader Länge. Also ist H aperiodisch.

Wie sieht die Grenzverteilung $\mathcal{G}(\mathcal{M})$ aus?

(d) Die Kette aus dem Casino-Beispiel 6.11 ist nicht irreduzibel, denn die Zustände 0 und $K+N$ sind absorbierend! Die Casino-Kette ist also nicht ergodisch.

(e) Jeder Knoten der ersten Ehrenfest-Kette in Beispiel 6.13 hat Periode 2 und die Kette ist nicht ergodisch. In der zweiten Ehrenfest-Kette hingegen hat jeder Knoten die Periode 1 aufgrund der Eigenschleifen: Die zweite Kette ist also irreduzibel wie auch aperiodisch und deshalb ergodisch. Wie sieht ihre Grenzverteilung aus?

(f) Die „Warteschlangenkette“ aus Beispiel 6.14 ist zwar irreduzibel und aperiodisch, aber sie besitzt unendlich viele Zustände: In diesem Fall gilt Satz 6.27 leider nicht.

6.3.4. Stationäre Verteilungen

Der neue Page-Rank PR^* stimmt mit der Grenzverteilung der Webkette überein und scheint ein sehr sinnvolles Maß für Renommee zu sein. Können wir auch den alten Page-Rank PR mit Begriffen aus der Theorie der Markov-Ketten charakterisieren, um dann PR und PR^* vergleichen zu können? Wir schreiben die Page-Rank-Eigenschaft in Matrizen-Notation auf.

Satz 6.29. Sei $0 \leqslant d < 1$ und der gerichtete Graph WEB besitze keine Senke.

(a) Der Vektor PR erfülle die Page-Rank-Eigenschaft (6.2). Dann ist PR eine Verteilung.

(b) Für jede Verteilung σ gilt

$$\sigma \text{ erfüllt die Page-Rank-Eigenschaft (6.2)} \iff \sigma = \sigma \cdot P_d(\mathrm{WEB}). \tag{6.8}$$

Beweis: (a) Wir zeigen, dass PR eine Verteilung ist. Wir dürfen annehmen, dass PR die Page-Rank-Eigenschaft (6.2) besitzt.

$$\begin{aligned}
\sum_{j=1}^{n} \mathrm{PR}_j &\overset{(6.2)}{=} \sum_{j=1}^{n} \Big(\tfrac{1-d}{n} + d \cdot \sum_{i \in \mathrm{Vor}_{\mathrm{WEB}}(j)} \tfrac{\mathrm{PR}_i}{a_i} \Big) &&= n \cdot \tfrac{1-d}{n} + d \cdot \sum_{j=1}^{n} \sum_{i \in \mathrm{Vor}_{\mathrm{WEB}}(j)} \tfrac{\mathrm{PR}_i}{a_i} \\
&= (1-d) + d \cdot \sum_{(i,j) \in E} \tfrac{\mathrm{PR}_i}{a_i} &&= (1-d) + d \cdot \sum_{i=1}^{n} \sum_{j:(i,j) \in E} \tfrac{\mathrm{PR}_i}{a_i} \\
&\overset{\substack{G \text{ ohne} \\ \text{Senke}}}{=} (1-d) + d \cdot \sum_{i=1}^{n} \big(a_i \cdot \tfrac{\mathrm{PR}_i}{a_i} \big) &&= (1-d) + d \cdot \sum_{i=1}^{n} \mathrm{PR}_i \\
&= (1-d) + d \cdot \sum_{j=1}^{n} \mathrm{PR}_j \,.
\end{aligned}$$

Insbesondere gilt also:

$$(1-d) \cdot \sum_{j=1}^{n} \mathrm{PR}_j \quad = \quad (1-d).$$

Wir haben gefordert, dass der Dämpfungsfaktor d von Eins verschieden ist und erhalten daher

$$\sum_{j=1}^{n} \mathrm{PR}_j = 1.$$

PR ist also eine Verteilung, denn in der Page-Rank-Bedingung (6.2) wird auch die Nicht-Negativität $\mathrm{PR}_j \geqslant 0$ gefordert.

(b) Angenommen die Verteilung σ erfüllt die Page-Rank-Eigenschaft (6.2). Wir nutzen aus, dass σ eine Verteilung ist:

$$\begin{aligned}
\sigma_j &\overset{(6.2)}{=} \frac{1-d}{n} + d \cdot \sum_{i \in \mathrm{Vor}_{\text{WEB}}(j)} \frac{\sigma_i}{a_i} \\
&\overset{\sigma \text{ ist eine Verteilung}}{=} \sum_{i \in \{1,\dots,n\}} \frac{1-d}{n} \cdot \sigma_i + d \cdot \sum_{i \in \mathrm{Vor}_{\text{WEB}}(j)} \frac{\sigma_i}{a_i} \\
&\overset{\text{Definition von } P_d(\text{WEB})}{=} \sum_{i \in \{1,\dots,n\}} \sigma_i \cdot \big(P_d(\text{WEB})\big)_{i,j} = (\sigma \cdot P_d(\text{WEB}))_j
\end{aligned}$$

und $\sigma = \sigma \cdot P_d(\text{WEB})$ folgt. Gilt umgekehrt $\sigma = \sigma \cdot P_d(\text{WEB})$ für die Verteilung σ, dann lesen wir die obigen Gleichungen rückwärts und erhalten

$$\begin{aligned}
(\sigma \cdot P_d(\text{WEB}))_j &= \sum_{i \in \{1,\dots,n\}} \sigma_i \cdot \big(P_d(\text{WEB})\big)_{i,j} \\
&\overset{\text{Definition von } P_d(\text{WEB})}{=} \sum_{i \in \{1,\dots,n\}} \frac{1-d}{n} \cdot \sigma_i + d \cdot \sum_{i \in \mathrm{Vor}_{\text{WEB}}(j)} \frac{\sigma_i}{a_i} \\
&\overset{\sigma \text{ ist eine Verteilung}}{=} \frac{1-d}{n} + d \cdot \sum_{i \in \mathrm{Vor}_{\text{WEB}}(j)} \frac{\sigma_i}{a_i} \\
&\overset{\sigma \cdot P_d(\text{WEB}) = \sigma}{=} \sigma_j.
\end{aligned}$$

Also erfüllt σ die Page-Rank-Eigenschaft (6.2). □

Definition 6.30. Sei (G, P) eine Markov-Kette mit n Zuständen. Eine Verteilung $\sigma \in \mathbb{R}^n$ heißt **stationär** für die Markov-Kette (G, P), falls gilt

stationär

$$\sigma = \sigma \cdot P.$$

Stationäre Verteilungen sind von besonderem Interesse, denn besitzt ein Vektor die Page-Rank-Eigenschaft, dann ist der Vektor eine Verteilung nach Teil (a) und sogar eine stationäre Verteilung nach Teil (b). Umgekehrt besitzt jede stationäre Verteilung der Page-Rank-Matrix die Page-Rank-Eigenschaft. Mit anderen Worten:

Folgerung 6.31. Sei $0 \leqslant d < 1$ und der gerichtete Graph WEB besitze keine Senke. Dann gilt

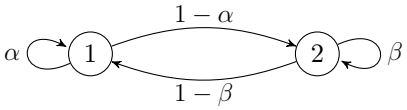
Der Vektor σ besitzt die Page-Rank-Eigenschaft $\Longleftrightarrow$

σ ist eine stationäre Verteilung der Page-Rank-Matrix $P_d(\text{WEB})$.

PR ist eine stationäre Verteilung. Ja, wir wissen sogar mehr, denn umgekehrt erfüllt eine stationäre Verteilung der Page-Rank-Matrix die Page-Rank-Eigenschaft.

Wenn die Kette (G, P) eine stationäre Verteilung σ als Anfangsverteilung besitzt, dann gilt $\sigma = \sigma \cdot P$ und die Kette verbleibt in σ im nächsten und allen weiteren Schritten: Kein Wunder, dass die Verteilung σ „stationär“ genannt wird.

Beispiel 6.32. Wir haben uns in Beispiel 6.28 bereits mit der Kette



beschäftigt und beobachtet, dass für eine ergodische Kette $\alpha \neq 1 \neq \beta$ gilt. Hier rechnen wir die stationäre Verteilung aus. Eine Verteilung $\sigma = (x, y)$ ist genau dann stationär, wenn $\sigma \cdot P = \sigma$ gilt.

$$\begin{aligned} \sigma \cdot P &= (\,x\,,\,y\,) \cdot \begin{pmatrix} \alpha & 1-\alpha \\ 1-\beta & \beta \end{pmatrix} \\ &= (\; x \cdot \alpha + y \cdot (1-\beta), \quad x \cdot (1-\alpha) + y \cdot \beta \;) \\ &\stackrel{!}{=} \sigma = (\,\alpha, \beta\,). \end{aligned}$$

Also führt die Forderung $\sigma P \stackrel{!}{=} \sigma$ auf das Gleichungssystem

$$\begin{aligned} x \cdot \alpha + y \cdot (1-\beta) &\stackrel{!}{=} x \\ x \cdot (1-\alpha) + y \cdot \beta &\stackrel{!}{=} y \end{aligned}$$

und dieses Gleichungssystem ist äquivalent mit

$$x \cdot (1-\alpha) = y \cdot (1-\beta).$$

Wir erhalten einen ein-dimensionalen Lösungsraum, was ist passiert? Wir müssen fordern, dass σ eine Verteilung ist, d.h. das $x + y = 1$ gilt. Wir erhalten das neue Gleichungssystem

$$\begin{aligned} x \cdot (1-\alpha) &= y \cdot (1-\beta) \\ y &= 1 - x \end{aligned}$$

mit der Lösung

$$x = \frac{1-\beta}{2-\alpha-\beta} \text{ und } y = \frac{1-\alpha}{2-\alpha-\beta}.$$

Die Wahrscheinlichkeit einen Zustand zu erreichen ist proportional zu der Komplementärwahrscheinlichkeit im anderen Zustand zu verbleiben.

Beispiel 6.33. (**Symmetrische Ketten**). Sei (G, P) eine Markov-Kette und die Matrix P sei symmetrisch, d.h. es gelte $P_{i,j} = P_{j,i}$ für alle Knoten i, j. Wie üblich sei $V = \{1, \ldots, n\}$ die Knotenmenge. Wir behaupten, dass die **Gleichverteilung** $\sigma = (1/n, \ldots, 1/n)$ eine stationäre Verteilung. Wir werten das Vektor-Matrix Produkt $\sigma \cdot P$ aus:

Gleichverteilung

$$\begin{aligned} (\sigma \cdot P)_j &= \sum_{i=1}^{n} \sigma_i \cdot P_{i,j} = \sum_{i=1}^{n} \frac{1}{n} \cdot P_{i,j} = \frac{1}{n} \cdot \sum_{i=1}^{n} P_{i,j} \\ &\stackrel{P \text{ ist symmetrisch}}{=} \frac{1}{n} \cdot \sum_{i=1}^{n} P_{j,i} \stackrel{P \text{ ist stochastisch}}{=} \frac{1}{n} = \sigma_j. \end{aligned}$$

Also gilt $\sigma \cdot P = \sigma$ und die Gleichverteilung ist tatsächlich stationär.

Betrachten wir Irrfahrten auf einem ungerichteten, regulären[5] Graphen G. Die Übergangsmatrix P der Irrfahrten ist eine symmetrische Matrix, denn $P_{i,j} = P_{j,i}$ für alle Knoten i und j. Irrfahrten auf regulären Graphen besitzen die Gleichverteilung als stationäre Verteilung!

Beispiel 6.34. („**Gambler's Ruin**"). Wir betrachten die Markov-Kette (G, P) aus dem „Casino-Beispiel" 6.11. Die Verteilungen $\sigma_1 = (1, 0, \ldots, 0)$ und $\sigma_2 = (0, \ldots, 0, 1)$, die den Ruin des Spielers, bzw. der Bank ausdrücken, sind stationär. Beachte aber, dass Konvexkombinationen[6] stationärer Verteilungen wieder stationär sind. Im Casino-Beispiel sind also alle Verteilungen $(\lambda, 0, \ldots, 0, 1-\lambda)$ mit $0 \leqslant \lambda \leqslant 1$ stationär. Insbesondere haben wir gelernt, dass eine Markov-Kette unendlich viele stationäre Verteilungen besitzen kann.

Und jetzt **der Hammer**: der Hammer

Satz 6.35. Sei $\mathcal{M} = (G, P)$ eine ergodische Markov-Kette. Dann besitzt die Kette genau eine stationäre Verteilung σ und es gilt

$$\sigma = \left(\lim_{k\to\infty} \left(P^k\right)_{1,1}, \ldots, \lim_{k\to\infty} \left(P^k\right)_{1,n} \right) = \mathcal{G}(\mathcal{M}).$$

Also ist die einzige stationäre Verteilung der Kette auch die Grenzverteilung der Kette.

Beweis: Die Kette $\mathcal{M} = (G, P)$ sei ergodisch und σ sei eine stationäre Verteilung. Nach Folgerung 6.24 gilt

$$\lim_{k\to\infty} \sigma P^k = \left(\lim_{k\to\infty} \left(P^k\right)_{1,1}, \ldots, \lim_{k\to\infty} \left(P^k\right)_{1,n} \right) = \mathcal{G}(\mathcal{M}) \tag{6.9}$$

für die Grenzverteilung $\mathcal{G}(\mathcal{M})$ der Kette. Um zu zeigen, dass $\mathcal{G}(\mathcal{M})$ stationär ist, werten wir das Vektor-Matrix-Produkt $\mathcal{G}(\mathcal{M}) \cdot P$ aus. Wir dürfen den Grenzwert mit einer endlichen Summe vertauschen und erhalten

$$\begin{aligned} (\mathcal{G}(\mathcal{M}) \cdot P)_j &= \sum_{i=1}^{n} \lim_{k\to\infty} \left(P^k\right)_{1,i} \cdot P_{i,j} = \lim_{k\to\infty} \sum_{i=1}^{n} \left(P^k\right)_{1,i} \cdot P_{i,j} \\ &= \lim_{k\to\infty} \left(P^{k+1}\right)_{1,j} = \lim_{k\to\infty} \left(P^k\right)_{1,j} = \mathcal{G}(\mathcal{M})_j. \end{aligned}$$

Also ist die Grenzverteilung, wie behauptet, stationär.

σ ist eine stationäre Verteilung. Also gilt $\sigma = \sigma \cdot P$ und deshalb folgt auch $\sigma = \sigma \cdot P^k$ für jede natürliche Zahl k. Dann gilt aber auch $\sigma = \lim_{k\to\infty} \sigma \cdot P^k$. Also folgt

$$\sigma = \lim_{k\to\infty} \sigma P^k \overset{(6.9)}{=} \mathcal{G}(\mathcal{M})$$

für jede stationäre Verteilung σ. Alle stationären Verteilungen stimmen mit der Grenzverteilung überein: Es gibt genau eine stationäre Verteilung und das ist die Grenzverteilung. □

[5] Ein Graph ist regulär, wenn alle Knoten dieselbe Anzahl von Nachbarn besitzen.

[6] Für Vektoren $x_1, \ldots, x_k \in \mathbb{R}^n$ ist eine Linearkombination $\sum_{i=1}^{k} \lambda_i \cdot x_i$ eine Konvexkombination, falls $(\lambda_1, \ldots, \lambda_k)$ eine Verteilung ist.

Folgerung 6.36. (Webkette). Alter und neuer Page-Rank stimmen überein.

Beweis: Die Webkette $\mathcal{W}$ ist ergodisch. Nach Folgerung 6.24 besitzt die Webkette eine Grenzverteilung, nämlich den neuen Page-Rank PR^*. Nach Satz 6.35 ist die Grenzverteilung eine stationäre Verteilung der Page-Rank-Matrix und sie ist die einzige. Nach Folgerung 6.31 besitzen genau die stationären Verteilungen die Page-Rank-Eigenschaft. Also ist die Grenzverteilung der einzige Vektor, der die Page-Rank-Eigenschaft besitzt: Es ist $\mathrm{PR} = \mathrm{PR}^*$. □

Wir haben sehr starke Indizien erhalten, dass der Page-Rank ein fundamentales Maß für das Renommee von Webseiten ist. Eine letzte Frage müssen wir später beantworten: Wie lässt sich die Grenzverteilung, bzw. der Page-Rank effizient berechnen?

Folgerung 6.37. (**Symmetrische, ergodische Ketten**) Wenn $\mathcal{M} = (G, P)$ eine ergodische Markov-Kette ist und wenn die Übergangsmatrix P symmetrisch ist, dann ist $\mathcal{G}(\mathcal{M})$ die Gleichverteilung: Alle Zustände haben die gleiche Wahrscheinlichkeit besucht zu werden.

Beweis: Wir wissen aus Beispiel 6.33, dass die Gleichverteilung eine stationäre Verteilung ist. Wenn also (G, P) eine ergodische Kette ist, dann folgt die Behauptung aus Satz 6.35. □

Folgerung 6.38. (Die Ehrenfest-Kette). Für die ergodische Ehrenfest-Kette $\mathcal{E}$ ist die Verteilung π mit $\pi_i = \binom{n}{i}/2^n$ für $i = 0, \ldots, n$ die einzige stationäre Verteilung und stimmt mit der Grenzverteilung $\mathcal{G}(\mathcal{E})$ überein: Die linke Urne besitzt i Partikel mit Wahrscheinlichkeit $\mathcal{G}(\mathcal{E})_i = \pi_i$.

Beweis: Nach Satz 6.35 genügt der Nachweis, dass die Verteilung π mit $\pi_i = \binom{n}{i}/2^n$ stationär ist. Dies wird in den Übungen gezeigt. □

Folgerung 6.39. (Irrfahrten auf Graphen).
Sei G ein ungerichteter, zusammenhängenden und nicht-bipartiten Graph $G = (V, E)$. Dann besucht eine Irrfahrt den Knoten v mit Wahrscheinlichkeit

$$\pi_v = \frac{d_v}{2|E|}.$$

Eine Irrfahrt besucht also jeden Knoten mit Wahrscheinlichkeit proportional zu seinem Grad!

Beweis: Sei $G = (V, E)$ ein ungerichteter zusammenhängender und nicht-bipartiter Graph. In Beispiel 6.28 (b) wird gezeigt, dass die Markov-Kette aus Beispiel 6.10 ergodisch ist. Nach Satz 6.35 genügt der Nachweis, dass die Verteilung π mit

$$\pi_v := \frac{d_v}{2|E|}$$

stationär ist, wobei d_v der Grad des Knotens $v \in V$ ist, also mit der Anzahl der Nachbarn von v übereinstimmt. Es ist $\sum_{v \in V} d_v = 2|E|$, denn in der Summe der Grade wird jede Kante zweimal, einmal für jeden Endpunkt gezählt. Wir können jetzt feststellen, dass π eine Verteilung ist, denn $\sum_{v \in V} \pi_v = \sum_{v \in V} \frac{d_v}{2|E|} = \frac{2|E|}{2|E|} = 1$.

Zuletzt ist zu zeigen, dass $\pi \cdot P = \pi$ für die Übergangsmatrix P gilt. Für jeden Knoten v gilt

$$(\pi \cdot P)_v = \sum_{u \in V} \pi_u \cdot P_{u,v} = \sum_{u \in V} \frac{d_u}{2|E|} \cdot P_{u,v} = \sum_{u \in V, \{u,v\} \in E} \frac{d_u}{2|E|} \frac{1}{d_u} = \sum_{u \in V, \{u,v\} \in E} \frac{1}{2|E|} = \frac{d_v}{2|E|} = \pi_v.$$

Also ist π stationär und die Grenzverteilung der Kette. □

Beispiel 6.40. (Warteschlangen). Leider gilt Satz 6.27 nur für Markov-Ketten nach unserer eingeschränkten Definition, also für Markov-Ketten mit endlich vielen Zuständen. Auch Satz 6.35 ist leider nicht mehr anwendbar. Was kann im Fall von abzählbar unendlich vielen Zuständen noch gesagt werden?

(a) Eine irreduzible Markov-Kette besitzt höchstens eine stationäre Verteilung.

(b) Besitzt die Kette $\mathcal{M}$ eine Grenzverteilung $\mathcal{G}(\mathcal{M})$, dann ist $\mathcal{G}(\mathcal{M})$ stationär.

Zwar ist die Kette aus Beispiel 6.14 irreduzibel und sogar aperiodisch, aber in den Übungen wird gezeigt, dass es keine stationären Verteilungen gibt und damit auch keine Grenzverteilung. Es stellt sich für jede Zahl $s \in \mathbb{N}$ heraus, dass die Schlangenlänge s die Wahrscheinlichkeit Null besitzt: Insbesondere ist die erwartete Schlangenlänge unendlich. □ Ende von Beispiel 6.40

In Markov-Chain-Monte-Carlo Verfahren versucht man eine Markov-Kette zu bauen, die eine vorgegebene Verteilung π als stationäre Verteilung besitzt. Ist dies gelungen, kann man die Kette benutzen, um Beispiele gemäß π zufällig zu ziehen.

Beispiel 6.41. (Markov-Chain-Monte-Carlo-Verfahren, kurz MCMC-Verfahren). MCMC-Verfahren

Angenommen, wir möchten eine größte (oder zumindest eine sehr große) unabhängige Menge in einem ungerichteten Graphen $G = (V, E)$ bestimmen. Sei

$$\mathcal{U}(G) = \{\, U \subseteq V \,:\, U \text{ ist unabhängig} \,\}$$

die Menge aller unabhängigen Mengen von G. Sei $h : \mathbb{N} \to \mathbb{R}_{>0}$ eine beliebige, aber strikt monoton wachsende Funktion, d.h. für alle Zahlen $n, m \in N$ mit $n < m$ möge $h(n) < h(m)$ gelten. Dann wäre es natürlich hervorragend, wenn wir unabhängige Teilmengen $U \in \mathcal{U}(G)$ mit einer „Zielwahrscheinlichkeit“ π_U proportional[7] zu $h(|U|)$ ziehen könnten: Zum Beispiel erhalten wir für jedes $\lambda > 0$ mit $h(n) := \exp(\lambda n)$ große unabhängige Mengen mit größerer Wahrscheinlichkeit als kleine. Wenn wir „genügend viele“ unabhängige Mengen ziehen und die größte ausgeben, haben wir gute Chancen auf eine zufriedenstellende Lösung.

Aber wie würfelt man unabhängige Mengen aus? Wir entwerfen eine Markov-Kette $\mathcal{M}$.

(a) Die Zustände von $\mathcal{M}$ sind die unabhängigen Teilmengen in $\mathcal{U}(\mathcal{G})$. Ein Zustandsübergang zwischen Teilmengen $U, U' \in \mathcal{U}(G)$ ist erlaubt, falls $|U \oplus U'| = 1$, d.h. falls sich U und U' in genau einem Element unterscheiden. (Andere Definitionen der Zustandsübergänge sind möglich.)

(b) Die Wahrscheinlichkeit $P_{U,U'}$ eines Zustandsübergangs von $U \in \mathcal{U}(G)$ zu $U' \in \mathcal{U}(G)$ ist definiert durch

$$P_{U,U'} = \frac{1}{|\mathcal{U}(G)|} \cdot \min\left\{1, \frac{h(|U'|)}{h(|U|)}\right\}.$$

[7] Wenn π proportional zu h ist (kurz: $\pi \sim h$), dann gilt $\pi_U = C \cdot h(U)$ für eine Konstante C und alle unabhängigen Mengen U.

Wie groß sind die Wahrscheinlichkeiten $P_{U,U'}$?

- Gilt $|U'| > |U|$, d.h. ist U' eine bessere Lösung, dann ist $h(|U'|) > h(|U|)$ und U' wird mit Wahrscheinlichkeit $1/|\mathcal{U}(G)|$ gewählt. Gilt hingegen $|U'| < |U|$, dann toleriert die Kette die schlechtere Lösung U' nur widerstrebend mit einer Wahrscheinlichkeit möglicherweise sehr viel kleiner als $1/|\mathcal{U}(G)|$.
- Schließlich definiere $P_{U,U}$ so, dass $P_{U,U} + \sum_{U',U \neq U'} P_{U,U'} = 1$ gilt.

In der Veranstaltung „Effiziente Algorithmen“ wird gezeigt, dass die stationäre Verteilung der Kette proportional zu h ist und deshalb mit der Zielverteilung π übereinstimmt. Man kann zeigen, dass die Kette ergodisch ist, also stimmen Grenzverteilung und stationäre Verteilung π überein: Die Kette, wenn genügend lange(!) durchlaufen, liefert Stichproben $U \in \mathcal{U}(G)$ mit Wahrscheinlichkeit ungefähr π_U.

Das hier beschriebene Verfahren lässt sich für viele Optimierungsprobleme anwenden. Während für das NP-vollständige Problem der unabhängigen Mengen gute Lösungen nur beobachtet werden, wenn die Kette sehr, sehr lange durchlaufen wird, ist die „Konvergenzgeschwindigkeit“ bei anderen Problemen wie etwa der approximativen Auswertung bestimmter mehrdimensionaler Integrale wesentlich besser.

Fazit: Zu einer beliebigen Zielverteilung π kann eine Markov-Kette konstruiert werden, die π als stationäre Verteilung besitzt. Die Kenntnis von π ist nicht notwendig, die Kenntnis irgendeiner Funktion h mit $\pi \sim h$ ist ausreichend. Kritisch ist die Geschwindigkeit mit der eine scharfe Approximation von π erreicht wird. Das skizzierte Verfahren wird auch als **Metropolis-Algorithmus** bezeichnet.

Metropolis-Algorithmus

6.3.5. Eine effiziente Approximation der Grenzverteilung

Wir nehmen in diesem Abschnitt an, dass die Kette $\mathcal{M} = (G, P)$ ergodisch ist. Unser Ziel ist die schnelle, approximative Bestimmung der stationären Verteilung von $\mathcal{M}$. Eine Lösung des entsprechenden Gleichungssystems ist aufgrund der Dimension des Systems völlig unrealistisch, dennoch wird uns eine approximative Bestimmung für bestimmte Ketten und insbesondere auch für die Webkette $\mathcal{W}$ gelingen.

In Folgerung 6.24 haben wir für eine beliebige Verteilung π die Beziehung

$$\lim_{k \to \infty} \pi \cdot P^k = \mathcal{G}(\mathcal{M})$$

gezeigt. Um die Grenzverteilung $\mathcal{G}(\mathcal{M})$ zu approximieren liegt es also nahe,

$$\pi \cdot P^k$$

für ein genügend großes k zu berechnen und genau dies tun wir.

Für die Webkette $\mathcal{W}$ genügen bereits kleine Werte von k für eine scharfe Approximation. Woran liegt das? Der Webgraph zeigt einen hohen Grad an Zusammenhang, der sich etwa im „small-world-Phänomen“ zeigt: Die durchschnittliche Distanz zwischen zwei Webseiten ist sehr klein. Für andere Ketten hingegen mag die Konvergenzgeschwindigkeit sehr viel langsamer sein und die notwendigen Werte für k sind entsprechend groß.

Wie geht man vor? Sollen wir zuerst die Matrixpotenz P^k mit $P = P_d(\text{WEB})$ berechnen und dann π mit P^k multiplizieren? Bitte, bitte nicht! Wir ersetzen stattdessen eine teure Matrizenmultiplikation mit bis zu n^3 Operationen durch die sehr viel billigere Vektor-Matrix-Multiplikation

$$\pi^{(k+1)} := \pi^{(k)} \cdot P, \tag{6.10}$$

mit höchstens n^2 Operationen, wobei wir zum Beispiel für $\pi^{(0)} := \pi$ die Gleichverteilung $(1/n, \ldots, 1/n)$ wählen können.

Für eine schnelle Berechnung des Vektor-Matrix-Produkts (6.10) wird ausgenutzt, dass $P_d(\text{WEB})$ viele identische Einträge der Form $\frac{1-d}{n}$ hat. Außerdem ist die Berechnung des Vektor-Matrix-Produkts hochgradig parallelisierbar. Details hierzu finden sich in [17].

Derzeit werden mehrere Tausend Rechner eingesetzt, die mehrere Stunden zur Berechnung des Page-Ranks benötigen — in Anbetracht der mehreren Milliarden Webseiten ein erstaunlich geringer Aufwand.

6.4. Zusammenfassung und Ausblick

Um zu verstehen, warum der von Google eingeführte Page-Rank so erfolgreich ist, haben wir Markov-Ketten betrachtet.

Zuerst haben wir für jede Webseite neue Verbindungen zu jeder anderen Webseite hinzugefügt. Wenn der Zufallssurfer in der neuen Webkette $\mathcal{W}$ unterwegs ist, dann entscheidet er für jede besuchte Seite zuerst, ob er eine neue Verbindung (mit Wahrscheinlichkeit $1 - d$) oder einen Hyperlink des ursprünglichen Webgraphen (mit Wahrscheinlichkeit d) verfolgen möchte. In jedem der beiden Fälle wählt er darauffolgend eine der jeweils zur Verfügung stehenden Nachbarseiten gemäß der Gleichverteilung aus.

Die derart modifizierte Markov-Kette $\mathcal{W} = (\vec{K}_n, P_d(\text{WEB}))$ ist jetzt irreduzibel, natürlich auch aperiodisch und deshalb ergodisch. In einer ergodischen Markov-Kette „vergisst“ die Kette die Anfangsverteilung und besitzt eine Grenzverteilung $\mathcal{G}(\mathcal{W})$: Die relativen Häufigkeiten, mit der Knoten in einer zufälligen, beliebig langen Irrfahrt besucht werden, konvergieren somit unabhängig von der Anfangsverteilung.

Warum erklärt die Theorie der Markov-Ketten, dass der Page-Rank ein vernünftiges Maß für das Renommee einer Webseite ist? In der Definition des Page-Rank wird gefordert, dass sich das Renommee einer Webseite zu gleichen Anteilen an alle direkten Nachfolger der Webseite vererbt. Der Page-Rank stellt sich deshalb als stationäre Verteilung der Webkette $\mathcal{W}$ heraus. Für ergodische Markov-Ketten stimmen stationäre Verteilung und Grenzverteilung überein und damit stimmen auch die Perspektiven von Page-Rank (bzw. die Verteilung PR) und Random Surfer (bzw. die Verteilung PR^*) überein.

Statt den Page-Rank als Lösung des viel zu großen linearen Gleichungssystems $\pi = \pi \cdot P_d(\text{WEB})$ zu bestimmen, wird die Grenzverteilung approximiert: Der Webgraph besitzt einen „hohen Grad“ an Zusammenhang und die Grenzverteilung lässt sich scharf approximieren, selbst wenn die Webkette nur für relativ wenige Schritte simuliert wird.

Markov-Ketten haben viele Anwendungen, unter anderen in der Analyse von Glücksspielen, Irrfahrten oder Warteschlangen, in der Modellierung physikalischer Prozesse, in der Approximation von Zeitreihen, in der numerischen Approximation mehrdimensionaler Integrale, im Entwurf von würfelnden Algorithmen und ...

Viel, viel mehr über Markov-Ketten erfahren Sie in der Vorlesung „Mathematik 3“.

6.5. Literaturhinweise zu Kapitel 6

Zur vertiefenden Lektüre seien Kapitel 2 von [25] sowie das Buch [11] empfohlen, das eine Algorithmen-orientierte Einführung in die Theorie der Markov-Ketten gibt. Einen Überblick über die Architektur von Suchmaschinen gibt der Artikel [1]; Details zum Page-Rank und zum HITS Verfahren finden sich in dem Buch [17] sowie in den Originalarbeiten [3, 22, 15, 6]. Als Einführung ins Thema *Information Retrieval* sei das Buch [20] empfohlen. Das Buch [7] ist ein „Klassiker“,

der eine umfassende Einführung in die Wahrscheinlichkeitstheorie (und insbesondere auch ins Thema Markov-Ketten) gibt.

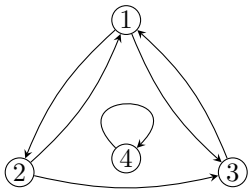
Viele Informationen und Literaturhinweise zum Thema *Suchmaschinen* finden sich auf der Webseite von Martin Sauerhoffs Vorlesung *Internet Algorithmen* an der TU Dortmund; siehe http://ls2-www.cs.uni-dortmund.de/lehre/winter200910/IntAlg/. Ein kurzer und allgemein verständlicher Überblick über das Page-Rank Verfahren wird in dem Spiegel-Online Artikel *Wie Google mit Milliarden Unbekannten rechnet* von Holger Dambeck gegeben; siehe http://www.spiegel.de/wissenschaft/mensch/0,1518,646448,00.html.

Quellennachweis: Teile dieses Kapitels orientieren sich an [25].

6.6. Übungsaufgaben zu Kapitel 6

Aufgabe 6.1.

Betrachten Sie den Web-Graphen $G = (V, E)$, der aus den vier Webseiten 1, 2, 3 und 4 besteht, die wie in der nebenstehenden Abbildung miteinander verlinkt sind. Benutzen Sie für die folgenden Aufgaben den Dämpfungsfaktor $d := \frac{1}{2}$.



(a) Berechnen Sie ähnlich wie in Beispiel 6.2 die Page-Ranks PR_1, PR_2, PR_3 und PR_4 der vier Webseiten von G bezüglich des Dämpfungsfaktors d.

(b) Stellen Sie für den angegebenen Web-Graphen G und den Dämpfungsfaktor d die Page-Rank-Matrix $P_d(\text{WEB})$ auf.

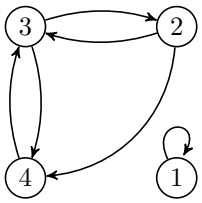
(c) Sei P die Page-Rank-Matrix $P_d(\text{WEB})$ aus Teilaufgabe ((b)). Wir nehmen an, der Zufallssurfer startet auf einer der vier Webseiten von G, wobei er jede Webseite gleichwahrscheinlich als Startpunkt wählen kann. Das bedeutet, dass die Anfangsverteilung für den Zufallssurfer durch $X^{(0)} := (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ beschrieben wird. Berechnen Sie die Wahrscheinlichkeitsverteilung des Zufallssurfers auf den Knoten von G nach einem Schritt (d.h. $X^{(1)}$), nach zwei Schritten (d.h. $X^{(2)}$) und nach drei Schritten (d.h. $X^{(3)}$). Dabei ist $X^{(1)} := X^{(0)} \cdot P$, $X^{(2)} := X^{(1)} \cdot P$ und $X^{(3)} := X^{(2)} \cdot P$.

(d) Gesucht ist ein Web-Graph $G' = (V', E')$ mit vier Webseiten, in dem jede Webseite auf mindestens eine Webseite verlinkt, die nicht sie selber ist. Zusätzlich soll der Zufallssurfer mit der Anfangsverteilung $X^{(0)} := (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ nach einem Schritt in G' genau dieselbe Wahrscheinlichkeitsverteilung erreichen, es soll also $X^{(0)} \cdot P(G', d) = X^{(0)}$ gelten. Geben Sie einen solchen Graphen G' an und weisen Sie nach, dass $X^{(0)} \cdot P(G', d) = X^{(0)}$ gilt.

Aufgabe 6.2.

(a) i) Betrachten Sie den rechts dargestellten Webgraphen WEB.
Bestimmen Sie die Übergangsmatrix $P_d(\text{WEB})$ für den Dämpfungsfaktor $d = \frac{3}{4}$.

WEB:



ii) Zeigen Sie, dass die Verteilung $\text{PR} := \frac{1}{44}(11, 8, 14, 11)$ die Page-Rank-Eigenschaft (bzgl. $d = \frac{3}{4}$) besitzt.

iii) Wie ändern sich die Page-Ranks PR_i der Seiten $i = 1, 2, 3, 4$, wenn dem Webgraphen ein Link von Webseite 2 auf sich selbst hinzugefügt wird? Welche steigen, welche sinken,

welche bleiben gleich?
Eine kurze, begründete Antwort genügt, eine Rechnung ist nicht erforderlich.

(b) Wir betrachten nun einen anderen Webgraphen WEB.

i) Gegeben sei die Übergangsmatrix einer Webkette $\mathcal{W} = (G, P)$ auf einem Webgraphen WEB′

$$P := P_d(\text{WEB}') = \frac{1}{12}\begin{pmatrix} 2 & 5 & 5 \\ 5 & 5 & 2 \\ 2 & 8 & 2 \end{pmatrix}$$

mit dem Dämpfungsfaktor $d = \frac{1}{2}$.

Geben Sie den zugrundeliegenden **Webgraphen** WEB′ sowie den **Graphen** G der dazugehörigen Markov-Kette $\mathcal{W}$ an.

ii) Ein Zufallssurfer starte in Knoten 1, d.h. für die Anfangsverteilung gelte $\pi^{(0)} = (1, 0, 0)$. Berechnen Sie, wo sich der Surfer mit welcher Wahrscheinlichkeit nach einem Schritt und nach zwei Schritten aufhält, d.h. berechnen Sie $\pi^{(1)}$ und $\pi^{(2)}$.

iii) Berechnen Sie den Page-Rank-Vektor von P, d.h. bestimmen Sie eine Verteilung PR mit der Page-Rank-Eigenschaft.

Hinweis: Stellen Sie ein lineares Gleichungssystem auf und lösen Sie es.

Aufgabe 6.3. Zwei Roboter R_1 und R_2 sollen eine schwere Kiste bewegen. Dies gelingt ihnen nur, wenn beide gemeinsam ziehen oder gemeinsam drücken. Andernfalls bewegt sich die Kiste nicht. Wenn sich die Kiste bewegt, bekommen beide Roboter positives Feedback (+1) und behalten jeweils ihre aktuelle Aktion bei. Ansonsten erhalten sie negatives Feedback (–1) und wählen für den nächsten Zug zufällig und unabhängig voneinander eine neue Aktion, d. h. beide Roboter werfen jeweils eine faire Münze und entscheiden sich bei Kopf fürs Drücken und bei Zahl fürs Ziehen.

(a) Modellieren Sie das Verhalten der Roboter als Markov-Kette.

(b) Angenommen, anfangs drückt der Roboter R_1 und der andere Roboter zieht. Wie hoch ist die Wahrscheinlichkeit, dass sich die Kiste nach k Schritten ($k \in \mathbb{N}$) immer noch am selben Platz wie zu Beginn befindet?

(c) Angenommen, wir kennen die Anfangsverteilung der Markov-Kette nicht. Wie hoch ist die Wahrscheinlichkeit, dass sich die Kiste nach „unendlich vielen“ Schritten immer noch am selben Platz wie zu Beginn befindet?

(d) Sei $m \in \mathbb{N}$ mit $m \geqslant 2$. Jetzt stehen die Roboter $R_1, \ldots, R_m$ zur Verfügung. Auch sie können die Kiste nur bewegen, wenn *alle* Roboter dieselbe Aktion ausführen. Modellieren Sie auch diese Situation als Markov-Kette.

Aufgabe 6.4. Das Quidditch-Spiel[8] Gryffindor gegen Ravenclaw steht bevor. Der Mannschaftskapitän der Gryffindors hat beim Training der gegnerischen Mannschaft zugeschaut und vom Passspiel der drei Ravenclaw-Jägern Roger Davies, Jeremy Stretton und Randolph Burrow folgende Beobachtungen festgehalten.

[8] Quidditch ist die Lieblingssportart des Zauberers Harry Potter.

- Davies spielt nur in der Hälfte aller Schritte den Quaffel[9] ab, und zwar doppelt so oft zu Burrow wie zu Stretton.
- Stretton behält den Quaffel mit Wahrscheinlichkeit 2/3, spielt mit Wahrscheinlichkeit 2/9 zu Davies ab, und ansonsten zu Burrow.
- Burrow spielt den Quaffel immer sofort weiter, wobei er doppelt so oft zu Stretton wie zu Davies abspielt.

(a) Modellieren Sie das Passspiel der Ravenclaw-Jäger als Markov-Kette, indem Sie eine Irrfahrt des Quaffels auf den Zuständen 1 := Davies, 2 := Stretton, und 3 := Burrow gemäß den oben beschriebenen Wahrscheinlichkeiten annehmen. Geben Sie den Graphen der Kette mit den Übergangswahrscheinlichkeiten sowie die Übergangsmatrix an.

(b) Am Anfang eines Spielzugs sei Stretton im Ballbesitz, d.h. es gelte $X^{(0)} = (0, 1, 0)$. Zeigen Sie durch vollständige Induktion, dass für alle $t \in \mathbb{N}$ gilt:

$$X^{(t)} = \left(\tfrac{1}{3}\left(1 - 3^{-t}\right),\ \tfrac{1}{2}\left(1 + 3^{-t}\right),\ \tfrac{1}{6}\left(1 - 3^{-t}\right)\right)$$

(c) Mit welcher Wahrscheinlichkeit befindet jeder der Jäger im Ballbesitz, wenn der Spielzug unendlich lange dauert?

Aufgabe 6.5. Carla betreibt einen Car-Sharing-Service in Frankfurt am Main mit drei Standorten Bockenheim (B), Hauptbahnhof (H) und Riedberg (R). An diesen drei Verleih-Stationen können Fahrzeuge ausgeliehen und wieder abgegeben werden. Die Auswertung ihrer Kundendaten ergibt die folgende Statistik:

(B) 75% aller in Bockenheim ausgeliehenen Fahrzeuge werden wieder in Bockenheim abgegeben, aber nur ein Sechstel der in Bockenheim ausgeliehen Wagen werden am Hauptbahnhof abgegeben.

(H) Zwei von drei Kunden, die am Hauptbahnhof ein Fahrzeug ausleihen, geben es am Hauptbahnhof oder am Riedberg wieder ab, und zwar fünfmal so häufig am Hauptbahnhof wie am Riedberg.

(R) Leiht ein Kunde ein Fahrzeug am Riedberg aus, so gibt er es in 19 von 36 Fällen dort wieder ab, aber nur in einem von 12 Fällen in Bockenheim.

Da alle Autos zwischen 22 Uhr abends und 6 Uhr morgens auf einem Parkplatz an den drei Standorten stehen müssen, überlegt Carla, wie groß diese Parkplätze auf lange Sicht sein sollten.

(a) Modellieren Sie Carlas Statistik als Markov-Kette $\mathcal{M} = (G, P)$, indem Sie eine Irrfahrt eines Fahrzeugs zwischen den Zuständen **B**, **H** und **R** annehmen.

Geben Sie den Graphen G in grafischer Darstellung an und beschriften Sie jede Kante mit der entsprechenden Übergangswahrscheinlichkeit. Geben Sie auch die Übergangsmatrix P an.

(b) Angenommen, zu Beginn der Irrfahrt befindet sich ein Fahrzeug in Bockenheim mit Wahrscheinlichkeit 1, d. h. die Anfangsverteilung sei $\pi^{(0)} = \left(\pi_B^{(0)}, \pi_H^{(0)}, \pi_R^{(0)}\right) = (1, 0, 0)$.

[9] Der Quaffel ist ein großer, roter Lederball, der von den Jägern in eines der drei gegnerischen Tore befördert werden muss.

(i) Zeigen Sie durch vollständige Induktion: Für alle $k \in \mathbb{N}$ gilt

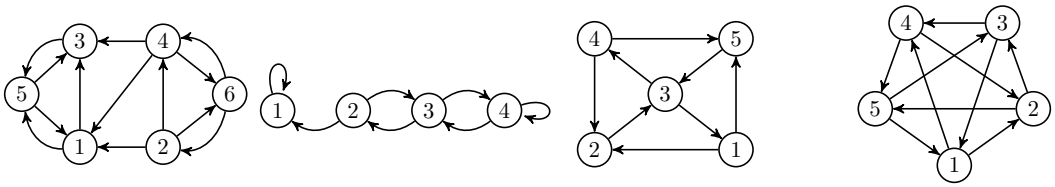
$$\pi^{(k)} = \left(\tfrac{1}{2}(1+2^{-k}) \ , \ \tfrac{1}{3}(1-2^{-k}) \ , \ \tfrac{1}{6}(1-2^{-k})\right).$$

(ii) Berechnen Sie, wo sich das Fahrzeug mit welcher Wahrscheinlichkeit nach 10 Schritten aufhält.

(iii) Mit welcher relativen Häufigkeit befindet sich das Fahrzeug nach einer „unendlich langen“ Irrfahrt in Bockenheim, am Hauptbahnhof bzw. am Riedberg?

(c) Angenommen, Carlas Car-Sharing-Service verfügt über 24 Fahrzeuge und ist schon seit über einem Jahr im Geschäft. Wie viele Fahrzeuge parken erwartungsgemäß am Abend auf jedem der drei Parkplätze Bockenheim, Hauptbahnhof und Riedberg?

Aufgabe 6.6. Betrachten Sie die folgenden Graphen G_1, G_2, G_3 und G_4. Bestimmen Sie (mit kurzer Begründung), welche der Graphen irreduzibel und/oder aperiodisch sind, und welche nicht.



Aufgabe 6.7.

(a) Betrachten Sie die folgenden Graphen G_1, G_2, G_3 und G_4. Bestimmen Sie (mit kurzer Begründung), welche der Graphen i) irreduzibel, ii) aperiodisch sind, und welche nicht.

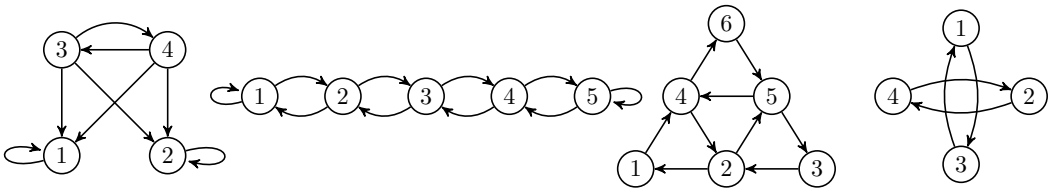


Abbildung 6.1.: Links: G_1, Mitte links: G_2, Mitte rechts: G_3, rechts: G_4

(b) Wir modellieren die Bewegung einer einzelnen Schachfigur auf einem Schachbrett als Markov-Kette mit Zuständen $V := \{\mathrm{a}, \ldots, \mathrm{h}\} \times \{1, \ldots, 8\}$. In jedem Schritt führe die Figur einen der ihr nach den Schachregeln möglichen Züge aus, wobei jeder Zug mit derselben Wahrscheinlichkeit gewählt werde. Es ist nicht zulässig, dass die Figur auf ihrem Feld stehen bleibt, sie *muss* sich bewegen.

Geben Sie jeweils an, ob die so beschriebene Markov-Kette irreduzibel bzw. aperiodisch ist, wenn es sich bei der Figur um einen

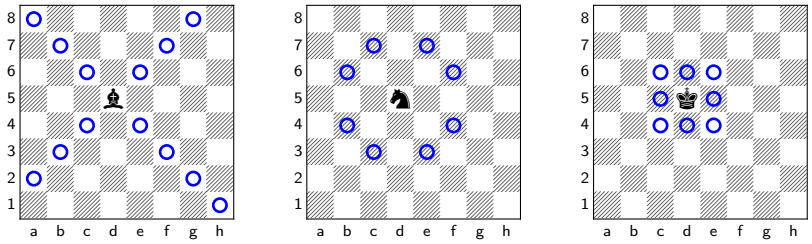


Abbildung 6.2.: Schachfeld mit Spielfigur. Links: Läufer; Mitte: Springer; rechts: König. Die möglichen Züge jeder Figur sind mit blauen Kreisen markiert.

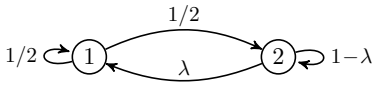
a) Läufer b) Springer c) König

handelt. Begründen Sie die Korrektheit Ihrer Antwort. Sie brauchen die Markov-Kette bzw. ihren Graphen und ihre Übergangsmatrix nicht explizit zu bestimmen!

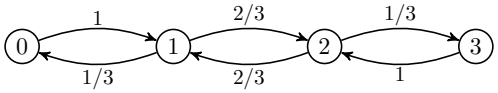
Aufgabe 6.8.

(a) Berechnen Sie die stationären Verteilungen für die Markov-Ketten $\mathcal{M}_1$ und $\mathcal{M}_2$.

$\mathcal{M}_1$ mit $0 \leqslant \lambda \leqslant 1$

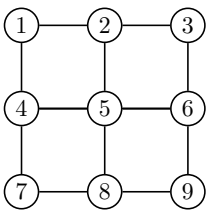


$\mathcal{M}_2$



(b) Ein Rasenmähroboter bewege sich zufällig auf einem Rasen, der durch den ungerichteten Graphen $R = (V_R, E_R)$ repräsentiert werde. Von einem Knoten v aus besuche der Roboter in jedem Schritt einen der Nachbarknoten mit jeweils gleicher Wahrscheinlichkeit.

Rasen R:



(i) Modellieren Sie die Irrfahrt des Roboters als Markov-Kette (G, P), indem Sie den Graphen G angeben und seine Kanten mit den Übergangswahrscheinlichkeiten beschriften.

(ii) Bestimmen Sie die stationäre Verteilung, indem Sie Resultate aus der Vorlesung verwenden. *Hinweis*: Sie müssen dazu kein lineares Gleichungssystem lösen.

Aufgabe 6.9. Alice besitzt vier Regenschirme, zwei davon zu Hause und zwei auf der Arbeit. Alice geht morgens von zu Hause zur Arbeit und abends von der Arbeit zurück nach Hause. Es regnet unabhängig voneinander morgens mit Wahrscheinlichkeit $m > 0$ bzw. abends mit Wahrscheinlichkeit $a > 0$.

Falls an dem jeweiligen Ort ein Schirm vorhanden ist und es regnet, so nimmt Alice einen Schirm mit. Andernfalls nimmt Alice keinen Schirm mit.

(a) Modellieren Sie das Geschehen als Markov-Kette. Ihre Zustände sollten ausdrücken, wie viele Schirme sich morgens (vor der Arbeit) bei Alice zu Hause befinden. Geben Sie auch die Übergangsmatrix an.

(b) Ist die Markov-Kette ergodisch? Begründen Sie!

(c) Bestimmen Sie die Grenzverteilung für $m = a = 1/4$.

(d) Bestimmen Sie die Grenzverteilung für $m = 1/4$ und $a = 1/3$.

(e) Für die Parameterwahlen für a und m aus Aufgabenteill (c) und (d): Wie hoch ist jeweils die Wahrscheinlichkeit, dass Alice auf dem Weg zur Arbeit nass wird? Wie hoch ist die Wahrscheinlichkeit, dass Alice auf dem Weg nach Hause nass wird?

Aufgabe 6.10. James hat sein gesamtes Vermögen bis auf 3 Dollar im Casino verspielt. Um seine Verluste besser zu verkraften, muss James sich einen Martini gönnen. Dieser kostet jedoch 8 Dollar.
Am Roulette-Tisch kann James Geld auf „gerade“ oder „ungerade“ setzen und dabei den gesetzten Betrag verdoppeln oder verlieren. James' Gewinnwahrscheinlichkeit für eine Partie sei $p \in (0, 1)$. James möchte sich das Geld für den Martini erspielen und erwägt dabei die folgenden Strategien:

- Bei der *vorsichtigen* Strategie setzt er stets einen Dollar und verlässt den Tisch, sobald er 8 Dollar hat oder pleite ist.
- Bei der *aggressiven* Strategie setzt er stets soviel wie möglich, *aber nicht mehr als nötig*, um den Tisch mit 8 Dollar zu verlassen. Er verlässt den Tisch, sobald er 8 Dollar hat oder pleite ist.

(a) Modellieren Sie jede der beiden Strategien als Markov-Kette.

(b) Bestimmen Sie für beide Strategien die Wahrscheinlichkeit w_{Martini}, dass James sich einen Martini gönnen kann.

Hinweis: Für die vorsichtige Strategie können Sie die Ergebnisse zum Gambler's-Ruin-Problem aus der Vorlesung (Beispiel 6.11 im Skript) verwenden. Unterscheiden Sie dabei die Fälle $p = 1/2$ und $p \neq 1/2$.

(c) Berechnen Sie für beide Strategien jeweils die Wahrscheinlichkeit w_{Martini} für die drei Fälle

(i) $p = \frac{1}{3}$, (ii) $p = \frac{1}{2}$, (iii) $p = \frac{2}{3}$.

und diskutieren Sie die Ergebnisse.

(d) Skizzieren Sie den Funktionsgraphen der Wahrscheinlichkeiten w_{Martini} für beide Strategien in Abhängigkeit von p und diskutieren Sie das Ergebnis. Sie können den Funktionsgraphen plotten lassen, z. B. mit diesem Online-Tool: https://rechneronline.de/funktionsgraphen/

Aufgabe 6.11. Wir betrachten die Gambler's-Ruin-Kette aus der Vorlesung (Beispiel 6.11) und wollen die Wahrscheinlichkeit herleiten, dass der Spieler die Bank sprengt. Sei K das Startkapital des Spielers, N das Kapital des Casinos und $M := K + N$. Die Gewinnwahrscheinlichkeit des Spielers für eine Partie sei $p \in (0, 1)$ und es sei $q := 1 - p$. Es sei s_K die Wahrscheinlichkeit, die Bank zu sprengen, d. h. vom Zustand K aus den Zustand M zu erreichen.

(a) Stellen Sie eine Rekursionsgleichung für s_K in Abhängigkeit von s_{K-1} und s_{K+1} auf. Welche Werte haben s_0 bzw. s_M?

(b) Zeigen Sie, dass für $0 < K < M$ gilt: $s_{K+1} - s_K = \frac{q}{p}(s_K - s_{K-1})$.

(c) Zeigen Sie: $s_K = \begin{cases} \frac{1-\left(\frac{q}{p}\right)^K}{1-\frac{q}{p}} \cdot s_1 & \text{falls } p \neq q, \\ K \cdot s_1 & \text{falls } p = q\,. \end{cases}$

Hinweis: Formen Sie die Rekursionsgleichung aus a) nach $s_{K+1} - s_K$ um und expandieren Sie sie, sodass Sie $s_{K+1} - s_K$ in Abhängigkeit von s_1 und s_0 ausdrücken. Verwenden Sie anschließend eine Teleskopsumme.

(d) Zeigen Sie: $s_K = \begin{cases} \frac{1-\left(\frac{q}{p}\right)^K}{1-\left(\frac{q}{p}\right)^M} & \text{falls } p \neq q, \\ \frac{K}{M} & \text{falls } p = q\,. \end{cases}$

Für die folgenden Aufgaben können Sie einen Matrizenrechner (z.B. https://matrixcalc.org/de) als Hilfsmittel verwenden.

Aufgabe 6.12. Julian möchte den sagenhaften Yeti aufspüren, der sich bekanntermaßen vorzüglich auf den höheren Gipfeln des Himalayas aufhält. Julian besitzt keine Karte des Gebietes. Um den Yeti dennoch zu finden, verfolgt er einen randomisierten Ansatz und geht zufällig bergauf oder bergab.

Das Gebirge ist als eine Menge von Knoten $V := \{1, \ldots, n\}$ gegeben. Jedem Knoten ist durch eine Funktion $H\colon V \to \mathbb{N}$ seine Höhe zugeordnet. Zwischen je zwei benachbarten Knoten $i, i{-}1 \in V$ beträgt der Höhenunterschied genau 1 oder –1, d.h. $|H(i) - H(i{-}1)| = 1$.

Wir modellieren Julians Tour als Markov-Kette mit Zustandsmenge V: Von Zustand i aus wird ein Nachbarzustand $j \in \{i{-}1, i{+}1\}$ mit Wahrscheinlichkeit u (*upwards*) besucht, falls $H(j) > H(i)$; und mit Wahrscheinlichkeit d (*downwards*) besucht, falls $H(j) < H(i)$. Dabei gilt $0 < d \leqslant u \leqslant \frac{1}{2}$. Mit der restlichen Wahrscheinlichkeit verbleibt die Kette in Zustand i. Formal:

$$\text{Für } i \in \{2, \ldots, n{-}1\} \text{ gilt } P_{i,j} = \begin{cases} u & \text{falls } j \in \{i{-}1, i{+}1\} \text{ und } H(j) > H(i), \\ d & \text{falls } j \in \{i{-}1, i{+}1\} \text{ und } H(j) < H(i), \\ 1 - P_{i,i-1} - P_{i,i+1} & \text{falls } j = i, \\ 0 & \text{sonst.} \end{cases}$$

Für die Zustände 1 und n gelten entsprechende Übergangswahrscheinlichkeiten, wobei hier jeweils nur ein Nachbarzustand existiert und die Verbleibe-Wahrscheinlichkeiten $P_{1,1}$ bzw. $P_{n,n}$ größer sind.

(a) Betrachten Sie das Bergmassiv *Annapurna Himal* in Abbildung 6.3 mit $n{=}7$ und $H(1){=}H(3){=}H(7){=}0$, $H(2){=}H(4){=}H(6){=}1$, $H(5){=}2$.

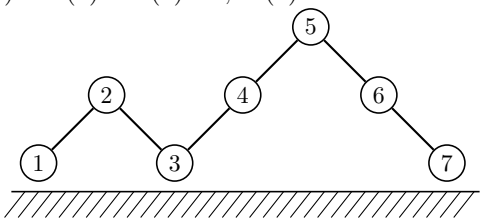


Abbildung 6.3.: Bergmassiv *Annapurna Himal*

Geben Sie für die folgende Parameterwahl von u und d jeweils den Graphen der Markov-Kette an und beschriften Sie die Kanten mit den Übergangswahrscheinlichkeiten. Berechnen Sie anschließend die Grenzverteilung, z.B. indem Sie die Potenzen P^k für hinreichend großes k berechnen.

Julian starte am linken Fuß des Berges, d.h. in Knoten 1. In welcher der beiden Ketten – d.h. in i) oder in ii) – sind mehr Schritte nötig, um die Grenzverteilung näherungsweise zu erreichen? Vergleichen Sie die Verteilungen $\pi^{(k)}$ beider Ketten jeweils nach $k = 20$, 60, 100, 200 Schritten. Welche Kette konvergiert schneller gegen ihre stationäre Verteilung?

a) $u = \frac{1}{2}$, $d = \frac{1}{4}$ b) $u = \frac{1}{2}$, $d = \frac{1}{20}$

(b) Zeigen Sie, dass in allgemeinen Gebirgen für jede Wahl von n, u, d und H die Verteilung $\mu := (\mu_1, \ldots, \mu_n)$

$$\text{mit } \mu_i := \frac{1}{\mathcal{C}} \cdot \left(\frac{u}{d}\right)^{H(i)} \text{ für alle } i \in V$$

stationär ist, wobei $\mathcal{C} := \sum_{i=1}^{n} (u/d)^{H(i)} > 0$ eine Konstante[10] ist.

Hinweis: Unterscheiden Sie die drei Fälle

- i ist ein Gipfel, d.h. $H(i{-}1) < H(i)$ und $H(i) > H(i{+}1)$.
- i ist ein Tal, d.h. $H(i{-}1) > H(i)$ und $H(i) < H(i{+}1)$.
- i ist ein Hang, d.h. $H(i{-}1) < H(i) < H(i{+}1)$ oder $H(i{-}1) > H(i) > H(i{+}1)$.

(c) Wie würden Sie u und d wählen, um nach einer unendlich langen Wanderung mit möglichst großer Wahrscheinlichkeit auf einem hohen Gipfel zu stehen? (Diese Frage betrifft also die Grenzverteilung.)

Hinweis: Nutzen Sie die Ergodizität und Teil b) aus.

Aufgabe 6.13. Alice und Bob spielen Karten mit einem Kartenstapel bestehend aus den drei Karten 1, 2, 3. Die beiden verwenden jeweils ein eigenes Mischverfahren:

- **Alice** zieht mit jeweils gleicher Wahrscheinlichkeit eine der drei Karten aus dem Stapel und legt sie in die Mitte zwischen die anderen beiden, ohne deren Reihenfolge zu verändern.
- **Bob** zieht mit jeweils gleicher Wahrscheinlichkeit eine der drei Karten aus dem Stapel und legt sie oben drauf.

(a) Modellieren Sie beide Mischverfahren als Markov-Ketten, indem Sie jeweils den Graphen angeben und seine Kanten mit den Übergangswahrscheinlichkeiten beschriften. Benutzen Sie dazu die Zustände

$$(1,2,3),\ (1,3,2),\ (3,1,2),\ (3,2,1),\ (2,3,1),\ (2,1,3),$$

wobei das Tripel (i, j, k) ausdrückt, dass die Karte i oben, die Karte j in der Mitte und die Karte k unten liegt. (Ordnen Sie dabei die Zustände in der obigen Reihenfolge kreisförmig an.)

(b) Wir definieren ein Maß[11] m für die „Zufälligkeit“ einer Verteilung $\mu = (\mu_1, \ldots, \mu_6)$:

$$m(\mu) := \max\left\{|\mu_i - \mu_j| : 1 \leqslant i < j \leqslant 6\right\}$$

[10] Die Konstante dient lediglich dazu, die Wahrscheinlichkeiten auf 1 zu normieren, spielt aber für die Modellierung hier keine Rolle.

[11] Bessere Maße für die „Zufälligkeit“ einer Verteilung sind im Allgemeinen die quadratische Abweichung $\sum_i (\mu_i - 1/n)^2$ oder die Shannon-Entropie $\sum_i \mu_i \log_2(1/\mu_i)$, die für diese Aufgabe aber unnötig kompliziert sind.

Je kleiner $m(\mu)$, desto „zufälliger“ (d.h. besser gemischt) ist die Verteilung μ.

Vergleichen Sie die Qualität der Mischverfahren von Alice und Bob. Angenommen, anfangs liegen die Karten in der Reihenfolge $(1, 2, 3)$ auf dem Stapel. Mit welchem Verfahren sind die Karten besser gemischt, nach

a) einem Schritt?
b) zwei Schritten?
c) fünf Schritten?
d) unendlich vielen Schritten?

Hinweis: Bei gegebener Anfangsverteilung $\pi^{(0)}$ und Übergangsmatrix P können Sie die Verteilung nach k Schritten durch $\pi^{(k)} = \pi^{(0)} \cdot P^k$ berechnen.
Eine andere Möglichkeit: Der Eintrag $(P^k)_{i,j}$ gibt die Wahrscheinlichkeit dafür an, dass wir von Zustand i nach Zustand j in genau k Schritten gelangen.

Aufgabe 6.14. Seien A und B stochastische $n \times n$-Matrizen. Zeigen Sie, dass $A \cdot B$ eine stochastische Matrix ist.

Aufgabe 6.15. Die deutsche Post unterhält eine Filiale mit drei Kundendienstarbeitern. Jeder Mitarbeiter kann einen Kunden betreuen (und ist dann *beschäftigt*) oder hat gerade nichts zu tun (und ist dann *frei*). Es spielen sich abwechselnd zwei verschiedene Prozesse ab:

(x) **Die Kundenabfertigung:** In jedem Schritt fertigt jeder beschäftigte Mitarbeiter unabhängig von den anderen seinen Kunden mit Wahrscheinlichkeit 1/2 ab. Mit Wahrscheinlichkeit 1/2 bleibt der Kunde und der Mitarbeiter ist weiterhin beschäftigt. Freie Mitarbeiter bleiben natürlich weiterhin frei.

(y) **Das Eintreffen neuer Kunden:** In jedem Schritt treffen mit Wahrscheinlichkeit 1/10 drei neue Kunden, mit Wahrscheinlichkeit 2/10 zwei neue Kunden, mit Wahrscheinlichkeit 3/10 ein neuer Kunde und mit Wahrscheinlichkeit 4/10 gar keine neuen Kunden ein.
Die Neuankömmlinge werden an die freien Mitarbeiter verteilt, die anschließend beschäftigt sind. Stehen nicht genügend freie Mitarbeiter für alle neuen Kunden zur Verfügung, so verlassen die überzähligen, nicht bedienten Kunden die Filiale wieder und gehen heim.

Wir wollen das Geschehen als Markov-Kette modellieren, mit den Zuständen $0, 1, 2, 3$, wobei Zustand i bedeute, dass genau i Mitarbeiter beschäftigt und $3-i$ Mitarbeiter frei sind. (Insbesondere spielt es keine Rolle, *welche* Mitarbeiter gerade beschäftigt oder frei sind.)

(a) Stellen Sie die Übergangsmatrix X auf, welche **nur die Kundenabfertigung** (x) beschreibt. Geben Sie auch den dazugehörigen Graphen G_X an.

(b) Stellen Sie die Übergangsmatrix Y auf, welche **nur das Eintreffen neuer Kunden** (y) beschreibt. Geben Sie auch den dazugehörigen Graphen G_Y an.

(c) Berechnen Sie die Matrizen $P := X \cdot Y$ und $Q := Y \cdot X$. Was drückt die zu P gehörige Markovkette (G_P, P) aus? Was drückt die zu Q gehörige Markov-Kette (G_Q, Q) aus?

(d) Berechnen Sie für (G_P, P) und (G_Q, Q) jeweils (näherungsweise) die stationäre Verteilung, z.B. indem Sie die Matrizen hinreichend oft potenzieren.

(e) Beantworten Sie folgende Fragen mit Hilfe der vorherigen Teilaufgaben.

- Wie oft kommt es vor, dass beim Eintreffen neuer Kunden alle Mitarbeiter beschäftigt sind?

- Wie oft kommt es vor, dass nach dem Eintreffen neuer Kunden alle Mitarbeiter frei sind?
- Würden Sie dazu raten, einen weiteren Mitarbeiter einzustellen?

Sie können einen Matrizenrechner als Hilfsmittel verwenden.

Aufgabe 6.16. Alice und Bob fahren gemeinsam in den Urlaub nach Dismodetien, ein kleines Land, welches bekannt für seine weißen Sandstrände, seine Museen und seine hochmodernen Wolkenkratzer ist. Da sich beide nicht einigen können, wie ihre Reise ablaufen soll, kommen sie zu der folgenden Vereinbarung: An ungeraden Tagen der Urlaubsreise darf Alice bestimmen, welche Sehenswürdigkeit sie besuchen, an geraden Tagen hat Bob das Sagen.

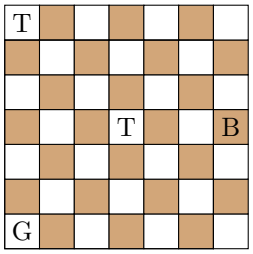
- Alice bestimmt, dass jeder ungerade (also der erste, dritte, fünfte,...) Tag am Strand verbracht werden muss, unabhängig davon, wo beide am Vortag waren.
- Bob hingegen ist kulturell interessiert: Wenn beide am Vortag am Strand waren, dann wählt er nur mit der Wahrscheinlichkeit $\frac{1}{4}$ einen weiteren Strand-Tag und mit jeweils gleicher Wahrscheinlichkeit p die Besichtigung eines Museums bzw. eines hochmodernen Wolkenkratzers. Wenn beide am Vortag im Museum waren, wählt er nun auf jeden Fall einen Wolkenkratzer und umgekehrt.

(a) Modellieren Sie die Urlaubspräferenzen von Alice und Bob zunächst unabhängig voneinander durch zwei Markov-Ketten $\mathcal{M}_A = (G_A, P_A)$ und $\mathcal{M}_B = (G_B, P_B)$, d. h. $\mathcal{M}_A$ und $\mathcal{M}_B$ beschreiben, wie ein Urlaub ablaufen würde, wenn nur Alice bzw. nur Bob das Sagen hätte. Verwenden Sie hierfür jeweils die Zustände **M** (wie Museum), **S** (wie Strand) und **W** (wie Wolkenkratzer).

(b) Berücksichtigen Sie nun die Vereinbarung, dass Alice und Bob abwechselnd ihre Urlaubstage gestalten. Modellieren Sie dies durch eine Markov-Kette $\mathcal{M}_{AB} = (G_{AB}, P_{AB})$, wobei die Zustände **M**, **S** und **W** für die Urlaubsaktivitäten an geraden Tagen (also am nullten, zweiten, vierten Tag usw.) stehen.

(c) Angenommen, $\mathcal{M}_1 = (G_1, P_1)$ und $\mathcal{M}_2 = (G_2, P_2)$ sind zwei Markov-Ketten auf einer gemeinsamen Menge $V = \{1, \ldots, n\}$ von Zuständen. Geben Sie die Übergangsmatrix P einer Kette $\mathcal{M} = (G, P)$ an, sodass P_{ij} die Wahrscheinlichkeit ist, dass ein Zufallssurfer in zwei Schritten von i nach j gelangt, wobei der erste Schritt gemäß $\mathcal{M}_1$ und der zweite Schritt gemäß $\mathcal{M}_2$ erfolgt.

Aufgabe 6.17. Eine größere Gruppe gar geldgieriger Gauner geht auf Goldsuche in ein großes Gewölbe, das von einem gefährlichen Gebirgsgolem bewacht wird.



Gewölbe mit Golem (G), Goldbarren (B) und Schatztruhen (T)

Das Gewölbe hat den Grundriss eines 7×7-Gitters. Der Golem patrouilliert zufällig durch das Gewölbe: In jeder Sekunde bewegt er sich zufällig genau drei Felder weit entweder in horizontale oder in vertikale Richtung.
Dem Golem möchten die Gauner lieber nicht in die Quere kommen. Um also möglichst unbeschadet das Gold plündern zu können, müssen sie wissen, mit welcher Wahrscheinlichkeit sich der Golem wo befindet.

Die Gauner versprechen, Ihnen (ja, genau: Sie sind gemeint!) ein Vierzehntel der Beute abzugeben, wenn Sie ihnen helfen die Wahrscheinlichkeiten berechnen.

(a) Angenommen, der Golem startet auf dem Feld ganz unten links.

Modellieren Sie die Bewegung des Golems als eine Irrfahrt auf einem ungerichteten Graphen U. Berücksichtigen Sie dabei **nur** die vom Golem erreichbaren Felder. Wie sieht die dazugehörige Markov -Kette (G, P) aus? Besitzt diese Markov-Kette eine Grenzverteilung?

(b) Nehmen Sie nun an, dass der Golem stets **zwei Bewegungen** auf einmal ausführt. Wie sehen der Übergangsgraph G' und die Matrix P' der neuen Markov-Kette (G', P') aus? Welche Felder sind von der Anfangsposition des Golems aus erreichbar?

(c) Bestimmen Sie für die beiden Felder mit den Schatztruhen (T), mit welcher Wahrscheinlichkeit sich der Golem dort nach sehr, sehr langer Zeit in einem **geraden** Zeitschritt befindet.

Hinweis: Berechnen Sie (näherungsweise) $\lim_{k\to\infty} P^{2k} = \lim_{k\to\infty}(P')^k$.

(d) Bestimmen Sie für das Feld mit den Goldbarren (B), mit welcher Wahrscheinlichkeit sich der Golem dort nach sehr, sehr langer Zeit in einem **ungeraden** Zeitschritt befindet.

Hinweis: Berechnen Sie (näherungsweise) $\lim_{k\to\infty} P^{2k+1} = \lim_{k\to\infty} P \cdot (P')^k$.

Aufgabe 6.18.

(a) Jede *nicht*-aperiodische Markov-Kette lässt sich durch Hinzufügen von Eigenschleifen in eine aperiodische Kette überführen, welche dieselben stationären Verteilungen besitzt.

Sei $\mathcal{M} = (G, P)$ eine beliebige Markov-Kette, $G = (V, E)$, sei I die Einheitsmatrix und $0 < \varepsilon < 1$. Betrachte die Markov-Kette $\mathcal{M}' = (G', P')$ mit $G' = (V, E')$, $E' = E \cup \{(i, i) : i \in V\}$ und $P' = (1-\varepsilon) \cdot P + \varepsilon \cdot I$.

Zeigen Sie:

$\mathcal{M}$ und $\mathcal{M}'$ besitzen dieselben stationären Verteilungen, d.h. für jede Verteilung σ gilt:

$$\sigma \text{ ist eine stationäre Verteilung für } \mathcal{M} \iff \sigma \text{ ist eine stationäre Verteilung für } \mathcal{M}'$$

(b) Sei $\mathcal{M} = (G, P)$ eine Markov-Kette und G irreduzibel. Zeigen Sie: $\mathcal{M}$ besitzt genau eine stationäre Verteilung.

Aufgabe 6.19. Sei (G, P) eine Markov-Kette mit Zustandsmenge V und sei G irreduzibel. Zeigen Sie: Wenn es einen Zustand mit Periode 1 gibt, dann ist G aperiodisch.

Hinweis: Sie dürfen folgenden Satz verwenden: Seien $a_1, \ldots, a_k \in \mathbb{N}_{>0}$ und $\mathrm{ggT}(a_1, \ldots, a_n) = 1$, dann lässt sich jede natürliche Zahl bis auf endlich viele Zahlen als Linearkombination $\lambda_1 a_1 + \cdots + \lambda_k a_k$ mit $\lambda_1, \ldots, \lambda_n \in \mathbb{N}$ darstellen.

Teil III.

Werkzeugkasten: Reguläre und kontextfreie Sprachen

7. Endliche Automaten

In diesem Kapitel geht es darum, das dynamische Verhalten von Systemen zu beschreiben, wie etwa

- die Funktionsweise wichtiger Teile eines Steuerwerks,
- die Wirkung von Bedienoperationen auf reale Automaten (z.B. einen Geldautomaten bei der Bank) oder auf die Benutzungsoberflächen von Software-Systemen,
- Schaltfolgen von Ampelanlagen,
- Abläufe von Geschäftsprozessen in Firmen oder
- die Steuerung von Produktionsanlagen.

Solche Abläufe werden modelliert, indem man die Zustände angibt, die ein System annehmen kann, und beschreibt, unter welchen Bedingungen das System aus einem Zustand in einen anderen übergehen kann.

Endliche Automaten (bzw. **Transitionssysteme**) werden als ein grundlegender Kalkül zur Modellierung sequentieller Abläufe wie auch zur Spezifikation realer oder abstrakter Maschinen vorgestellt. Das Einsatzgebiet endlicher Automaten ist vielfältig, nämlich:

- in der Definition regulärer Sprachen

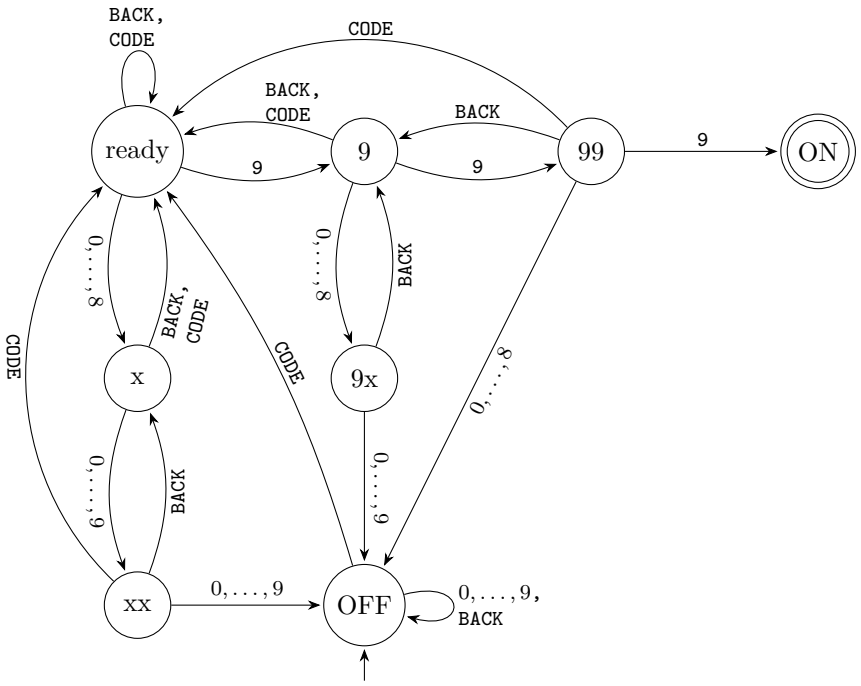
  also der Menge aller Folgen von Ereignissen, die von einem Startzustand in einen gewünschten Zustand führen (vgl. das Murmel-Problem aus Beispiel 1.1),

- in der Entwicklung digitaler Schaltungen
- in der Softwaretechnik (z. B. in der Modellierung des Applikationsverhaltens)
- in der lexikalischen Analyse, einer Vorstufe der Kompilierung
- im Algorithmenentwurf für String Probleme
- in der Abstraktion tatsächlicher Automaten (wie Bank- und Getränkeautomaten, Fahrstühle etc.).

Beispiel 7.1. (**Freischaltung eines Fernsehers**). Um die Kindersicherung des Fernsehers über die Fernbedienung freizuschalten, muss ein dreistelliger Code korrekt eingegeben werden. Dabei sind die folgenden Tasten relevant:

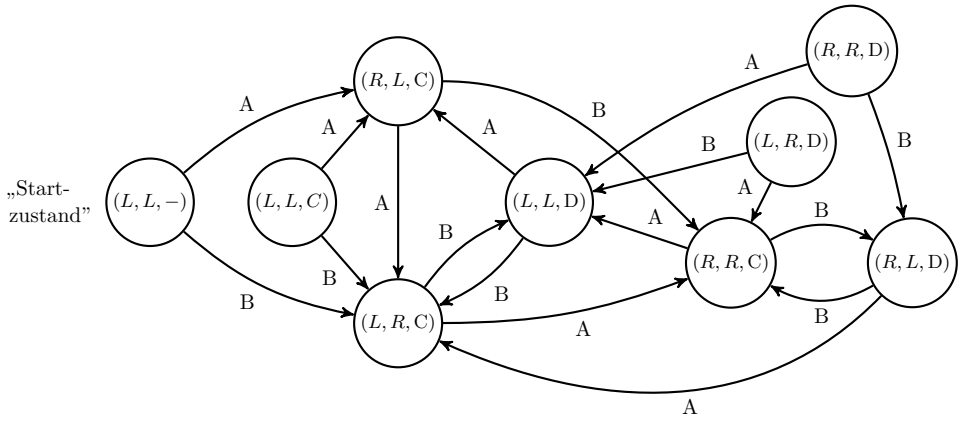
- Die Tasten $0, \ldots, 9$,
- die Taste `CODE` sowie
- die Taste `BACK`.

Die Taste `CODE` muss vor Eingabe des Codes gedrückt werden. Wird `CODE` während der Codeeingabe nochmals gedrückt, so wird die Eingabe neu begonnen. Schließlich, wird `BACK` gedrückt, so wird die zuletzt eingegebene Zahl zurückgenommen. Der Code zum Entsperren ist `999`.



Der Automat „akzeptiert“ alle Folgen von Bedienoperationen, die vom Zustand „**ready**“ in den Zustand „**ON**“ führen.

Beispiel 7.2. (**Das Murmelspiel als Transitionssystem**). In Beispiel 1.1 haben wir das Murmelspiel vorgestellt und mit dem folgenden Transitionssystem modelliert:



Die Wirkung aller Folgen von in A, bzw. B eingeworfenen Murmeln lässt sich jetzt schnell voraussagen.

7.1. Alphabete, Worte und Sprachen

Ein endlicher Automat reagiert auf ein Tupel von Eingaben durch eine Folge von Zustandswechseln. Die Komponenten der Eingabetupel gehören stets zu einer endlichen Menge Σ, die wir auch das
Alphabet **Alphabet** des Automaten nennen: Alphabete und endliche Mengen sind also Synonyme. Statt von einem Eingabetupel

$$(a_1, \ldots, a_k) \in \Sigma^k$$

spricht man von einem Wort

$$w = a_1 \cdots a_k$$

(über dem Alphabet Σ).

Definition 7.3. (Alphabete und Worte)
Sei Σ ein Alphabet, also eine endliche Menge.

(a) Für jede natürliche Zahl k wird Σ^k – also die Menge der Tupel mit k Komponenten aus der
Worte Menge Σ – als die Menge der **Worte** der **Länge** k (über dem Alphabet Σ) bezeichnet. Die
Länge Länge eines Wortes $a_1 \cdots a_k$ kennzeichnet man durch

$$|a_1 \cdots a_k| \ := \ k.$$

das leere Wort
- Das **leere Tupel** $() \in \Sigma^0$ heißt auch **das leere Wort** und wird mit ε (epsilon) bezeichnet.

Σ^*
- Die Menge aller Worte über Σ (von beliebiger endlicher Länge) bezeichnen wir mit Σ^*. Es gilt also:

$$\Sigma^* \ = \ \bigcup_{k \in \mathbb{N}} \Sigma^k \ = \ \{\, a_1 \cdots a_k \ : \ k \in \mathbb{N},\, a_1, \ldots, a_k \in \Sigma \,\}.$$

Σ^+
- Die Menge aller **nicht-leeren** Worte über Σ (von beliebiger endlicher Länge) bezeichnen wir mit Σ^+. Es gilt:

$$\Sigma^+ \ = \ \Sigma^* \setminus \{\varepsilon\} \ = \ \{\, a_1 \cdots a_k \ : \ k \in \mathbb{N}_{>0},\, a_1, \ldots, a_k \in \Sigma \,\}.$$

Konkatenation (b) Sind $v = a_1 \cdots a_k$ und $w = b_1 \cdots b_\ell$ zwei Worte über Σ, so ist die **Konkatenation** von v und w das Wort

$$vw \ := \ a_1 \cdots a_k b_1 \cdots b_\ell.$$

Präfix
Suffix
Teilwort (c) Ist $x = uvw$ für Worte u, v, w über Σ, dann heißt u ein **Präfix**, w ein **Suffix** und v ein **Teilwort** des Wortes x.

Beachte: Wegen $0 \in \mathbb{N}$ und $A^0 = \{()\} = \{\varepsilon\}$ enthält A^* insbesondere das leere Wort. Weiterhin ist $|\varepsilon| = 0$, d.h. das leere Wort hat die Länge 0.

Definition 7.4 (Sprachen). Sei Σ ein Alphabet.

Sprache (a) Eine **Sprache** L (über Σ) ist eine Teilmenge von A^*.

Konkatenation (b) Die **Konkatenation** $L = L_1 \cdot L_2$ von Sprachen L_1 und L_2 ist die Sprache

$$L = \{u \cdot v \ : \ u \in L_1,\, v \in L_2 \,\}.$$

Man verwendet häufig den Buchstaben L (Abkürzung für **L**anguage), um Sprachen zu bezeichnen.

Beispiel 7.5 (Natürliche Sprachen).
Wir betrachten das Alphabet

$$\Sigma_{deutsch} := \{\text{A, B, } \ldots\text{, Z, ä, ö, ü, a, b, } \ldots\text{, z, ä, ö, ü, ß, ., ,, :, ;, !, ?, -, \_}\}.$$

Beispiele für Sprachen über $\Sigma_{deutsch}$ sind:

- $L_1 :=$ Menge aller grammatikalisch korrekten Sätze der deutschen Sprache (aufgefasst als, Zeichenketten über $\Sigma_{deutsch}$)
- $L_2 :=$ Menge aller Worte der deutschen Sprache.

Beispiel 7.6 (Programmiersprachen).
Wir betrachten das Alphabet

$$\text{ASCII} := \text{die Menge aller ASCII-Symbole}$$

Beispiele für Sprachen über Alphabet ASCII sind:

- $L_1 :=$ die Menge aller Python-Schlüsselworte,
- $L_2 :=$ die Menge aller erlaubten Variablennamen in Python,
- $L_3 :=$ die Menge aller syntaktisch korrekten Python-Programme.

Im Folgenden interessieren wir uns vor Allem für Sprachen, die von Automaten definiert werden. So interessieren wir uns in Beispiel 7.1 etwa für die Sprache aller Worte über dem Alphabet $\Sigma := \{0, 1, \ldots, 9\} \cup \{\texttt{CODE}, \texttt{BACK}\}$, die zu einer Freischaltung des Fernsehers führen.

7.2. Deterministische endliche Automaten

Definition 7.7 (DFA). Ein **deterministischer endlicher Automat**[1] (kurz: **DFA**) DFA

$$A = (\Sigma, Q, \delta, q_0, F)$$

besteht aus:

- einer endlichen Menge Σ, dem so genannten **Eingabealphabet**, Eingabealphabet
- einer endlichen Menge Q, der so genannten **Zustandsmenge** (die Elemente aus Q werden **Zustände** genannt), Zustandsmenge
- einer Funktion δ von $Q \times \Sigma$ nach Q, dem Programm (oder **Übergangsfunktion**, bzw. **Überführungsfunktion**), Übergangsfunktion
- einem Zustand $q_0 \in Q$, dem so genannten **Startzustand**, Startzustand
- einer Menge $F \subseteq Q$, der so genannten Menge der **Endzustände** bzw. **akzeptierenden Zustände** (der Buchstabe F steht für „final states“, also „Endzustände“). Endzustand

deterministisch

Die in Definition 7.7 eingeführten DFAs $A = (\Sigma, Q, \delta, q_0, F)$ heißen **deterministisch**, weil es zu jedem Paar $(q, a) \in Q \times \Sigma$ genau einen „Nachfolgezustand“ $\delta(q, a)$ gibt: Der Nachfolgezustand von q ist durch den nächsten Buchstaben „determiniert“.

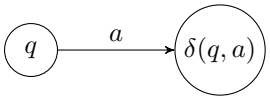
Graphische Darstellung endlicher Automaten:

Endliche Automaten lassen sich anschaulich durch beschriftete Graphen darstellen (vgl. das Murmelspiel, bzw. Beispiel 7.1):

- Für jeden Zustand $q \in Q$ gibt es einen durch  dargestellten Knoten.
- Der Startzustand q_0 wird durch einen in ihn hineinführenden Pfeil markiert, d.h.:

- Jeder akzeptierende Zustand $q \in F$ wird durch eine doppelte Umrandung markiert, d.h.:

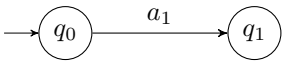
- Ist $q \in Q$ ein Zustand und $a \in \Sigma$ ein Symbol aus dem Alphabet, so gibt es in der graphischen Darstellung von Σ einen mit dem Symbol a beschrifteten Pfeil von Knoten  zu Knoten  , d.h.:



Die Verarbeitung eines Eingabeworts durch einen DFA:

Ein DFA $A = (\Sigma, Q, \delta, q_0, F)$ erhält als Eingabe ein Wort $w \in \Sigma^*$, das eine Folge von „Aktionen“ oder „Bedienoperationen“ repräsentiert. Bei Eingabe eines Worts w wird A im Startzustand q_0 gestartet. Falls w das leere Wort ist, d.h. $w = \varepsilon$, so passiert nichts weiter. Falls w ein Wort der Form $a_1 \cdots a_n$ mit $n \in \mathbb{N}_{>0}$ und $a_1, \ldots, a_n \in \Sigma$ ist, so geschieht bei der Verarbeitung von w durch A Folgendes: Durch Lesen der ersten Buchstabens von w, also a_1, geht der Automat in den Zustand $q_1 := \delta(q_0, a_1)$ über. In der graphischen Darstellung von A wird der Zustand

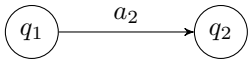
durch die mit a_1 beschriftete Kante verlassen, und q_1 ist der Endknoten dieser Kante, d.h.



In der graphischen Darstellung wird

[1] engl.: deterministic finite automaton

durch die mit a_2 beschriftete Kante verlassen



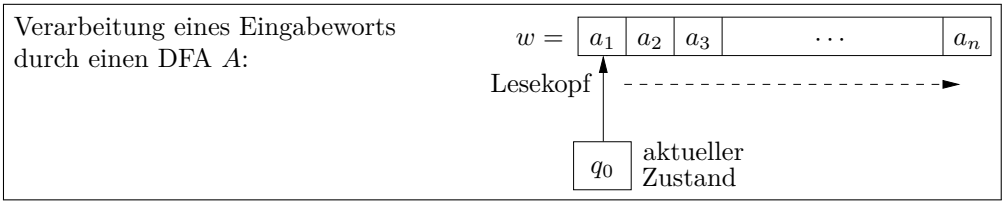
und der Automat landet in Zustand q_2, wobei $q_2 := \delta(q_1, a_2)$ der Endknoten dieser Kante ist.

Auf diese Weise wird nach und nach das gesamte Eingabewort $w = a_1 \cdots a_n$ abgearbeitet. Ausgehend vom Startzustand q_0 werden dabei nacheinander Zustände $q_1, \ldots, q_n$ erreicht. In der graphischen Darstellung von A entspricht dies gerade dem Durchlaufen eines Weges der Länge n, der im Knoten

startet und dessen Kanten mit den Buchstaben $a_1, \ldots, a_n$ beschriftet sind. Der Knoten

der am Ende dieses Weges erreicht wird, ist **der von A bei Eingabe w erreichte Zustand**, kurz: $q_n = \widehat{\delta}(q_0, w)$.

Ein DFA als Maschine mit Eingabeband und Lesekopf



Definition 7.8 (Die erweiterte Übergangsfunktion $\widehat{\delta}$ eines DFA).
Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein DFA. Die Funktion

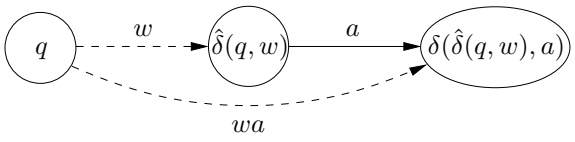
$$\widehat{\delta} \ : \ Q \times \Sigma^* \ \to \ Q$$

ist rekursiv wie folgt definiert:

- F.a. $q \in Q$ ist $\widehat{\delta}(q, \varepsilon) \ := \ q$.
- F.a. $q \in Q$, $w \in \Sigma^*$ und $a \in \Sigma$ gilt für $q' := \widehat{\delta}(q, w)$:

  $$\widehat{\delta}(q, wa) \ := \ \delta(q', a).$$

  Graphische Darstellung:



Insgesamt gilt: $\hat{\delta}(q_0, w)$ ist der Zustand, der durch Verarbeiten des Worts w erreicht wird.

Die von einem DFA akzeptierte Sprache:

Das Eingabewort w wird vom DFA A **akzeptiert**, falls

$$\widehat{\delta}(q_0, w) \in F,$$

d.h., der durch Verarbeiten des Worts w erreichte Zustand gehört zur Menge F der akzeptierenden Zustände.

In der graphischen Darstellung von A heißt das für ein Eingabewort $w = a_1 \cdots a_n$, dass es einen in

startenden Weg der Länge n gibt, dessen Kanten mit den Symbolen $a_1, \ldots, a_n$ beschriftet sind, und der in einem akzeptierenden Zustand  endet.

Definition 7.9 (Die von einem DFA A akzeptierte Sprache $L(A)$).
Die von einem DFA $A = (\Sigma, Q, \delta, q_0, F)$ akzeptierte Sprache $L(A)$ ist

$$L(A) \;:=\; \{\, w \in \Sigma^* \;:\; \hat{\delta}(q_0, w) \in F \,\}.$$

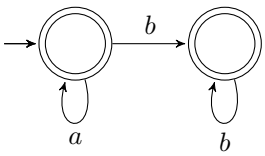
D.h.: Ein Wort $w \in \Sigma^*$ gehört genau dann zur Sprache $L(A)$, wenn es vom DFA A akzeptiert wird.

Beispiel 7.10. Wir betrachten das Eingabealphabet $\Sigma := \{a, b\}$.

(a) Manchmal geben wir nicht alle Zustandsübergänge an. Besitzt z.B. ein Zustand q keinen Übergang für einen Buchstaben $x \in \Sigma$, dann soll die Berechnung des DFA für alle Worte wx als „abgestürzt“ gelten, wenn $w \in \Sigma^*$ den Zustand q erreicht, wenn also $\widehat{\delta}(q_0, w) = q$ gilt. Stürzt die Berechnung für wx ab, dann stürzen auch Berechnungen für irgendwelche Fortsetzungen wxu ab: Der DFA kann kein solches Wort akzeptieren.

Durch das Weglassen von Zustandsübergängen wird die graphische Darstellung eines DFA übersichtlicher: Wir sparen einen „Trap-Zustand“, auf den fehlende Zustandsübergänge gerichtet werden müssten.

Schauen wir uns ein Beispiel an. Der DFA A_1 besitze die folgende graphische Darstellung:



- A_1 akzeptiert z.B. folgende Worte: ε, a, b, aaa, $aaab$, $aaaabbbb$, bbb, ...
- A_1 „stürzt ab“ z.B. bei Eingabe von ba, $aabbba$, diese Eingaben werden verworfen.

Insgesamt gilt:

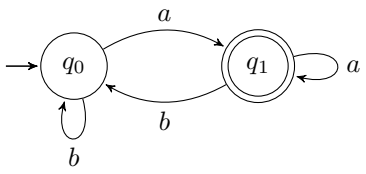
$$L(A_1) \;=\; \{a^n b^m : n \in \mathbb{N},\, m \in \mathbb{N}\}$$

(**Notation:** $a^n b^m$ bezeichnet das Wort $a \cdots a b \cdots b$ der Länge $n + m$, das aus n a's gefolgt von m b's besteht, z.B. ist $a^3 b^4$ das Wort $aaabbbb$.)

(b) Die graphische Darstellung eines DFA A_2 mit

$$L(A_2) \;=\; \{w \in \{a,b\}^* : \text{der letzte Buchstabe von } w \text{ ist ein } a\}$$

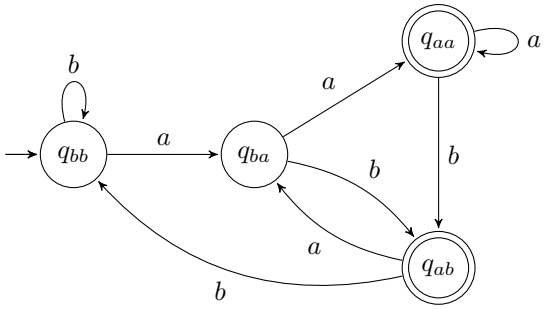
ist



(c) Die graphische Darstellung eines DFA A_3 mit

$$L(A_3) \;=\; \{w \in \{a,b\}^* : \text{der } \textbf{vorletzte} \text{ Buchstabe von } w \text{ ist ein } a\}$$

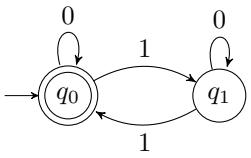
ist



Ein Anwendungsbeispiel: Paritätscheck durch einen DFA

Bei der Speicherung von Daten auf einem Speichermedium eines Computers werden Informationen durch Worte über dem Alphabet $\{0,1\}$ kodiert. Um eventuelle Fehler beim Übertragen der Daten erkennen zu können, wird der Kodierung oft ein so genanntes **Paritätsbit** angehängt, das bewirkt, dass die Summe der Einsen im resultierenden Wort gerade ist.

Für ein beliebiges Wort $w \in \{0,1\}^*$ sagen wir „w besteht den Paritätscheck“, falls die Anzahl der Einsen in w gerade ist. Zur effizienten Durchführung eines Paritätschecks für ein gegebenes Wort $w \in \{0,1\}^*$ kann man den folgenden DFA A benutzen:



Für diesen Automaten gilt: $L(A) = \{w \in \{0,1\}^* : w \text{ besteht den Paritätscheck}\}$.

7.2.1. Moore-Automaten

Endliche Automaten spielen auch in der technischen Informatik eine wichtige Rolle und zwar im Entwurfsprozess von Schaltwerken. Bevor wir das Konzept der Schaltwerke einführen, beschreiben wir das einfachere Konzept eines Schaltnetzes.

Schaltnetz

Ein **Schaltnetz** ist ein „Netz“ aus elementaren logischen Gattern, aufgebaut aus Transistoren. Wir begnügen uns mit dieser sehr oberflächlichen Beschreibung und verweisen für eine ausführlichere Darstellung auf die Veranstaltung „Hardwarearchitekturen und Rechensysteme“ im zweiten Semester. Für uns genügt das Wissen, dass ein Schaltnetz ein Tupel

$$f : \{0,1\}^n \to \{0,1\}^m$$

boolescher Funktionen berechnet; boolesche Funktionen haben wir in Definition 3.60 eingeführt. Schaltnetze können z.B. arithmetische Funktionen berechnen wie etwa die Addition oder Multiplikation von zwei Zahlen, wenn beide Eingabezahlen eine Binärdarstellung der Länge $n/2$ besitzen.

Betrachten wir zuerst eine einzige boolesche Funktion, wir nehmen also $m = 1$ an und das Schaltnetz berechnet die boolesche Funktion

$$f : \{0,1\}^n \to \{0,1\}.$$

Wir berechnen f mit einem endlichen Automaten

$$A_f = (\Sigma, Q, \delta, q_0, F).$$

Wir machen uns das Leben einfach und wählen das Eingabealphabet $\Sigma = \{0,1\}^n$. Der (partiell definierte) Automat A_f startet im Zustand ε, liest die gesamte Eingabe x in einem Schritt und wechselt dann entweder in den Zustand 0 (es ist $f(x) = 0$) oder in den Zustand 1 (es ist $f(x) = 1$). Wir beschreiben die einzelnen Komponenten von A_f:

1. A_f besitzt das Eingabealphabet $\Sigma = \{0,1\}^n$,
2. die Zustandsmenge $Q = \{\varepsilon, 0, 1\}$ mit Startzustand $q_0 = \varepsilon$,
3. die (partiell definierte) Übergangsfunktion $\delta : Q \times \Sigma \to Q$ mit
$$\delta(\varepsilon, x) = f(x)$$
4. und die Menge $F = \{1\}$ akzeptierender Zustände.

Endliche Automaten können „nur“ akzeptieren oder verwerfen. Um ein Tupel boolescher Funktionen zu berechnen, benötigen wir ein Automatenmodell mit Ausgabe.

Moore-Automat

Definition 7.11. Ein **Moore-Automat**

$$(\Sigma, Q, \delta, q_0, \lambda, \Omega)$$

ist wie ein DFA aufgebaut, besitzt aber zusätzlich

- ein Ausgabealphabet Ω und
- eine Ausgabefunktion $\lambda : Q \to \Omega$.

Die Menge F akzeptierender Zustände fehlt, da Moore-Automaten nicht akzeptieren, sondern eine Ausgabe berechnen sollen. Des Weiteren verwenden wir häufig ein nur partiell definiertes Programm δ: Wir verabreden die Konvention, dass nicht definierte Zustandsübergänge zu einem Trap-Zustand führen, der das leere Wort als Ausgabe produziert.

Um ein Zustandsdiagramm eines Moore-Automaten $M = (\Sigma, Q, \delta, q_0, \lambda, \Omega)$ zu erhalten, ist jeder Zustand $q \in Q$ zusätzlich mit der Ausgabe $\lambda(q)$ zu beschriften. Der Moore-Automat M durchläuft für eine Eingabe $w = a_1 a_2 \cdots a_n$ einen Weg

$$(q_0, q_1, \ldots, q_n) \in Q^{n+1}$$

in seinem Zustandsdiagramm, wobei die Zustandsübergänge genau wie im Fall von DFAs durch die Übergangsfunktion δ definiert sind, d.h. es gilt $q_{i+1} = \delta(q_i, a_{i+1})$ für $i = 0, \ldots, n-1$. Im wesentlichen Unterschied zu DFAs kann der Moore-Automat aber eine Folge von Ausgaben produzieren, nämlich für Eingabe w das Tupel

$$\mu = (\lambda(q_1), \ldots, \lambda(q_n)) \in \Omega^n,$$

und wir sagen, dass der Moore-Automat die Ausgabe μ berechnet.

Wir nehmen jetzt an, dass ein Schaltnetz das Tupel

$$f : \{0,1\}^n \to \{0,1\}^m$$

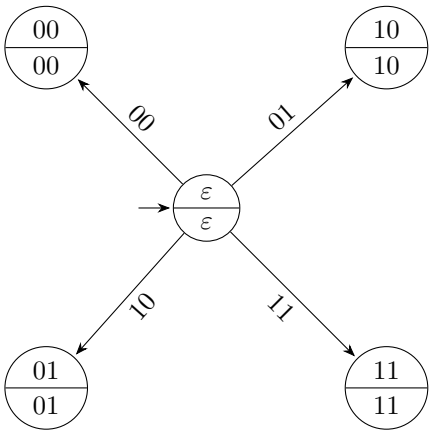
boolescher Funktionen für eine beliebige natürliche Zahl $m \in \mathbb{N}$ berechnet und entwerfen einen Moore-Automaten

$$A_f = (\Sigma, Q, \delta, q_0, \lambda, \Omega),$$

der die Ausgabe $f(x)$ für jede Eingabe $x \in \{0,1\}^n$ berechnet. Auch jetzt machen wir uns das Leben einfach und wählen $\{0,1\}^m$ als Ausgabealphabet. Unser Moore-Automat A_f wechselt für Eingabe $x \in \Sigma$ vom Startzustand ε in den Zustand $f(x)$ und kann dann die Ausgabe $f(x)$ geben. Die einzelnen Komponenten von A_f haben die folgende Form:

1. $\Sigma = \{0,1\}^n$ ist das Eingabealphabet,
2. $Q = \{\varepsilon\} \cup \{0,1\}^m$ ist die Zustandsmenge und $q_0 = \varepsilon$ der Startzustand,
3. $\delta : Q \times \Sigma \to Q$ mit
$$\delta(\varepsilon, x) = f(x)$$
ist die partielle Übergangsfunktion. Beachte, dass δ nur für von ε ausgehende Übergänge definiert ist.
4. $\Omega = \{0,1\}^m$ ist das Ausgabealphabet und
5. $\lambda : Q \to \Omega$ mit $\lambda(q) = q$ ist die Ausgabefunktion.

Man überzeuge sich, dass A_f die gewünschte Funktion f berechnet. Zum Beispiel, für $n = m = 2$ und $f(x_1, x_2) := x_2 x_1$ hat der Moore-Automat die folgende Form:



Beachte, dass für Eingabe x_1x_2 der Zustand $f(x_1, x_2) = x_2x_1$ erreicht wird und dass $f(x_1, x_2)$ ausgegeben wird.

7.2.2. Mealy-Automaten

Schaltwerk

Bisher haben wir nur eine einzige Ausgabe $f(z) \in \Omega$ für ein Eingabealphabet Σ gegeben. Wenn wir aber ein **Schaltwerk** durch einen Automaten modellieren möchten, müssen wir das ändern. Wie arbeitet ein Schaltwerk? Wie im Fall von Schaltnetzen nehmen wir an, dass $\Sigma = \{0,1\}^n$ das Eingabealphabet und $\Omega = \{0,1\}^m$ das Ausgabealphabet ist.

Wenn ein Schaltwerk ein Eingabetupel $(x_1, \ldots, x_k) \in \Sigma^k$ verarbeitet, dann wird mit Rückkopplung gerechnet. Was genau bedeutet das? Angenommen, r Bits werden rückgekoppelt. Um die Berechnung eines Schaltwerks formal zu definieren, benötigen wir zwei Funktionen, eine

Ausgabefunktion

Ausgabefunktion

$$f_A : \Sigma \times \{0,1\}^r \to \Omega$$

Rückkopplung

und eine **Rückkopplungsfunktion**

$$f_R : \Sigma \times \{0,1\}^r \to \{0,1\}^r.$$

Wenn das Schaltwerk nacheinander die Eingaben $x_1, \ldots, x_k \in \{0,1\}^n$ verarbeitet, dann werden die

$$\text{Ausgaben } y_1, \ldots, y_k \in \Omega \text{ und Rückkopplungen } r_0 = 0^r, r_1, \ldots, r_{k-1}$$

berechnet, wobei

$$y_i = f_A(x_i, r_{i-1}) \text{ und } r_i = f_R(x_i, r_{i-1})$$

gilt. (Die anfängliche Rückkopplung ist also $r_0 = 0^r$.)

Das Schaltwerk berechnet also im $i$ten Schritt die Ausgabe $(f_A(x_i, r_{i-1}), f_R(x_i, r_{i-1}))$ und benutzt den Teil $f_R(x_i, r_{i-1})$ der Ausgabe als Rückkopplung r_i für den $(i+1)$ten Schritt. Das Schaltwerk beschreiben wir durch das Tupel $(\Sigma, r, \Omega, f_A, f_R)$.

Steuerwerk

Bemerkung 7.12. Beispielsweise kann ein **Steuerwerk** durch ein Schaltwerk berechnet werden. Wie arbeitet ein Steuerwerk?

1. Ein Steuerwerk bestimmt den nächsten Befehl in Abhängigkeit vom gegenwärtigen Systemzustand r_i und interpretiert den Befehl. Insbesondere werden Operanden, auf die sich der Befehl bezieht, geladen und Steuersignale an andere Funktionseinheiten (wie etwa das Rechenwerk) erstellt.

2. Der Systemzustand wird aktualisiert, d.h. r_{i+1} wird berechnet.

Mehr über Schaltwerke erfahren Sie in der Vorlesung „Hardwarearchitekturen und Rechensysteme“.

Können wir ein Schaltwerk $(\Sigma, r, \Omega, f_A, f_R)$ mit einem Moore-Automaten modellieren? Ja, aber die Modellierung wird einfacher, wenn wir mit Mealy-Automaten arbeiten.

Definition 7.13. Ein **Mealy-Automat** Mealy-Automat

$$(\Sigma, Q, \delta, q_0, \lambda, \Omega)$$

ist wie ein DFA aufgebaut, besitzt aber zusätzlich

- ein Ausgabealphabet Ω und
- eine Ausgabefunktion $\lambda : Q \times \Sigma \to \Omega$.

Wie für Moore-Automaten besitzt auch ein Mealy-Automat keine Menge F akzeptierender Zustände. Im Fall eines nur partiell definierten Programms verwenden wir wieder einen Trap-Zustand: Als Konvention verlangen wir, dass die auf den Trap-Zustand führenden Übergänge das leere Wort als Ausgabe produzieren.

Ein Mealy-Automat bestimmt seine nächste Ausgabe also in Abhängigkeit vom aktuellen Zustand *und* dem aktuellen Eingabebuchstaben. Um ein Zustandsdiagramm eines Mealy-Automaten $M = (\Sigma, Q, \delta, q_0, \lambda, \Omega)$ zu erhalten, erstellen wir wieder zuerst das Zustandsdiagramm des DFA $(\Sigma, Q, \delta, q_0, F)$ und beschriften zusätzlich die Übergange: Wenn $\delta(p, a) = q$, dann beschriften wir die Kante $p \to q$ wie üblich mit der Eingabe a und *zusätzlich* mit der Ausgabe $\lambda(p, a)$.

Der Mealy-Automat M wird für eine Eingabe $w = a_1 a_2 \cdots a_n$ eine Zustandsfolge $(q_0, q_1, \ldots, q_n) \in Q^{n+1}$ durchlaufen. Wir sagen, dass M die Ausgabefolge $(\lambda(q_0, a_1), \lambda(q_1, a_2), \ldots, \lambda(q_{n-1}, a_n))$ berechnet.

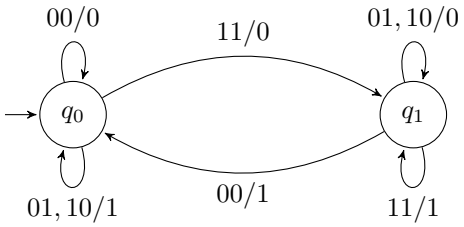
Beispiel 7.14. Wir können jeden Moore-Automaten $(\Sigma, Q, \delta, q_0, \lambda_{\text{Moore}}, \Omega)$ auch als Mealy-Automaten $(\Sigma, Q, \delta, q_0, \lambda_{\text{Mealy}}, \Omega)$ auffassen, wenn wir die Ausgabefunktion $\lambda_{\text{Mealy}} : Q \times \Sigma \to \Omega$ durch

$$\lambda_{\text{Mealy}}(p, a) := \lambda_{\text{Moore}}(p)$$

definieren: Im Mealy-Automaten benutzen wir die Abhängigkeit vom Eingabesymbol nicht.

Beispiel 7.15. Wir bauen einen Mealy-Automaten für die Addition zweier Binärzahlen $x = 0x_n \cdots x_1$ und $y = 0y_n \cdots y_1$. Dazu nehmen wir an, dass der Mealy-Automat die Eingabe $x_i y_i$ im i-ten Schritt erhält. (Beachte, dass 00 die Eingabe im letzten Schritt ist.) Wir wählen deshalb das Eingabealphabet $\Sigma := \{0, 1\}^2$ sowie das Ausgabealphabet $\Omega := \{0, 1\}$.

Das i-te Ausgabe-Bit hängt nur ab von den beiden Eingabebits x_{i-1}, y_{i-1} und dem im vorherigen Schritt evtl. erzeugten Übertrag. Wir „merken uns den Übertrag im Zustand“ (q_0 entspricht dem Übertrag 0, q_1 dem Übertrag 1) und können dann mit Hilfe der beiden neuen Eingabebits das entsprechende Ausgabebit der Summe berechnen. Hier ist das Zustandsdiagramm. (Lese ab/c als: Für Eingabebits ab wird das Bit c ausgegeben.)



Wir möchten ein Schaltwerk $S = (\Sigma, r, \Omega, f_A, f_R)$ durch einen Mealy-Automaten simulieren. Die Idee ist sehr einfach, denn das Schaltwerk ist schon ein „verkappter“ Mealy-Automat mit Zustandsmenge $Q = \{0,1\}^r$. Insbesondere sollten wir die Rückkopplungsfunktion f_R des Schaltwerks S als Übergangsfunktion unseres Mealy-Automaten verwenden, denn der „Systemzustand“ von S wird durch f_R aktualisiert. Die Ausgabefunktion f_A von S berechnet die Ausgabe in Abhängigkeit vom Systemzustand und der aktuellen Eingabe. Wir können also f_A auch als Ausgabefunktion des Mealy-Automaten verwenden. Unser Mealy-Automat $(\Sigma, Q, \delta, q_0, \lambda, \Omega)$ hat die folgenden Komponenten:

1. Das Eingabealphabet Σ und das Ausgabealphabet Ω werden vom Schaltwerk S übernommen.

2. Die Zustandsmenge ist $Q := \{0,1\}^r$ und $q_0 = 0^r$ ist der Anfangszustand,

3. die Übergangsfunktion $\delta : Q \times \Sigma \to Q$ wird durch $\delta = f_R$ und

4. die Ausgabefunktion $\lambda : Q \times \Sigma \to \Omega$ durch $\lambda := f_A$ definiert.

Wir haben ausgenutzt, dass Mealy-Automaten die Rückkopplung mit Hilfe ihrer Zustände auf natürliche Art und Weise simulieren können.

Bemerkung 7.16. Der Entwurfsprozess eines Schaltwerks wird grob gesprochen in den folgenden Schritten durchgeführt.

1. Ein Mealy-Automat wird zur Lösung eines Teils der Problemstellung entwickelt.

2. Der Automat wird minimiert, d.h. ein äquivalenter Automat mit kleinster Zustandsmenge wird berechnet und seine Automatentabelle wird bestimmt.

3. Mit Verfahren der Logik-Optimierung, wie etwa mit dem Quine/McCluskey Verfahren bestimmt man die minimale Gatterzahl für den restlichen Teil des Schaltwerks. (Das Verfahren von Quine/McCluskey haben wir im Kapitel „Aussagenlogik“ erwähnt.)

Alle Details und viel, viel mehr in der Vorlesung

„Hardwarearchitekturen und Rechensysteme“.

Bemerkung 7.17. Können wir heutige Rechner durch Mealy-Automaten modellieren?

„Im Prinzip“ ja: Heutige Rechner besitzen zwar einen modifizierbaren Speicher, aber dieser Speicher ist beschränkt. Was immer ein moderner Rechner berechnen kann, lässt sich auch durch Moore- oder Mealy-Automaten berechnen.

Aber, um k Flip-Flops zu simulieren, brauchen wir Mealy-Automaten mit bis zu 2^k Zuständen. Eine Modellierung moderner Rechner durch Mealy-Automaten ist völlig unsinnig, weil der notwendige Speicher im schlimmsten Fall exponentiell ($\lightning$) anwächst.

Aber Mealy-Automaten bieten sich zum Beispiel an, um Steuerwerke zu simulieren. Und dann sollten wir auch mit ihnen arbeiten, denn wir können Mealy-Automaten minimieren, wie wir gleich sehen werden.

7.3. Minimierung

Definition 7.18. Seien $A = (\Sigma, Q^A, \delta^A, q_0^A, F^A)$ und $B = (\Sigma, Q^B, \delta^B, q_0^B, F^B)$ DFAs.

(a) Wir nennen A und B **äquivalent**, wenn gilt äquivalent

$$L(A) = L(B).$$

(b) A heißt **minimal**, wenn kein mit A äquivalenter DFA eine kleinere Zustandszahl besitzt. minimal

Äquivalente DFAs haben dasselbe Ausgabeverhalten, können aber völlig unterschiedliche Zustandszahlen besitzen. Für einen gegebenen DFA $A = (\Sigma, Q, \delta, q_0, F)$ suchen wir einen „kleinsten“ mit A äquivalenten DFA. Das Tolle ist die genial einfache Idee der Minimierung:

> Wir sollten doch zwei Zustände $p, q \in Q$ zu einem einzigen Zustand verschmelzen dürfen, wenn p und q „äquivalent“ sind,
>
> also dasselbe Ausgabeverhalten besitzen.

Aber was bedeutet es, dasselbe Ausgabeverhalten zu besitzen?

Definition 7.19. Der DFA $A = (\Sigma, Q, \delta, q_0, F)$ sei gegeben. Die **Verschmelzungsrelation** $\equiv_A$ ist eine 2-stellige Relation über der Zustandsmenge Q. Wir sagen, dass Zustände $p, q \in Q$ äquivalent bzgl. A sind (kurz $p \equiv_A q$), wenn Verschmelzungsrelation

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \iff \widehat{\delta}(q, w) \in F.$$

Wir nennen $\equiv_A$ Verschmelzungsrelation, da wir bzgl. $\equiv_A$ äquivalente Zustände in einen Zustand verschmelzen möchten. Die Verschmelzungsrelation hat eine sehr schöne Eigenschaft, sie ist nämlich eine Äquivalenzrelation. Was es damit auf sich hat, sehen wir im nächsten Abschnitt.

7.3.1. Äquivalenzrelationen

Jeder gerichtete Graph $G = (V, E)$ lässt sich als eine 2-stellige Relation über der Knotenmenge V auffassen, da die Kantenmenge E von G ja gerade eine Teilmenge von $V^2 = V \times V$ ist. Umgekehrt können wir natürlich auch jede 2-stellige Relation R über einer Menge V als gerichteten Graphen mit Knotenmenge V und Kantenmenge R auffassen. Gerichtete Graphen mit Knotenmenge V sind also dasselbe wie 2-stellige Relationen über einer Menge V.

Von besonderem Interesse sind Äquivalenzrelationen:

Definition 7.20 (Äquivalenzrelation).
Sei E eine 2-stellige Relation über einer Menge V (d.h. $G = (V, E)$ ist ein gerichteter Graph).

(a) E heißt **reflexiv**, falls für alle $v \in V$ gilt: reflexiv

$$(v, v) \in E. \qquad \text{(Skizze: } v \text{)}$$

(b) E heißt **symmetrisch**, falls f.a. $v, w \in V$ gilt: symmetrisch

$$\text{Wenn } (v, w) \in E, \text{ dann auch } (w, v) \in E.$$

(d.h. zu jeder Kante 

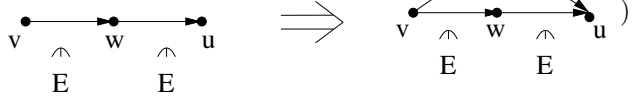
gibt es auch eine “Rückwärtskante” 

)

transitiv

(c) E heißt **transitiv**, falls f.a. $v, w, u \in V$ gilt:

$$\text{Ist } (v,w) \in E \text{ und } (w,u) \in E, \text{ so auch } (v,u) \in E.$$

(Skizze: 

)

Äquivalenzrelation

(d) E heißt eine **Äquivalenzrelation**, falls E **reflexiv**, **transitiv** und **symmetrisch** ist.

Beispiel 7.21. Beispiele für Äquivalenzrelationen:

(a) **Gleichheit:** Für jede Menge M ist

$$E \;:=\; \{\, (m,m) : m \in M \,\}$$

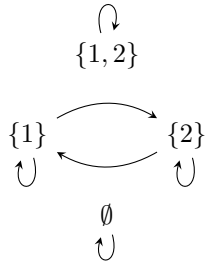
eine Äquivalenzrelation. Die Aussage "$(x,y) \in E$" entspricht gerade der Aussage "$x = y$".

(b) **Gleichmächtigkeit:** Für jede endliche Menge M ist

$$E \;:=\; \{\, (A,B) : A \subseteq M,\, B \subseteq M,\, |A| = |B| \,\}$$

eine Äquivalenzrelation über der Potenzmenge $\mathcal{P}(M)$.

Skizze für $M = \{1,2\}$:



(c) **Logische Äquivalenz:** Die Relation

$$E \;:=\; \{\, (\varphi,\psi) : \varphi, \psi \in \mathrm{AL},\, \varphi \equiv \psi \,\}$$

ist eine Äquivalenzrelation über der Menge AL aller aussagenlogischen Formeln.

Bemerkung 7.22 (Äquivalenzklassen). Sei E eine Äquivalenzrelation über einer Menge V. Für jedes $v \in V$ bezeichnet

$$[v]_E \;:=\; \{\, v' \in V \,:\, (v,v') \in E \,\}$$

$[v]_E$
Äquivalenzklasse

die **Äquivalenzklasse** von v bezüglich E. D.h.: Die Äquivalenzklasse $[v]_E$ besteht aus allen Elementen von V, die laut E "äquivalent" zu v sind. Eine Menge $W \subseteq V$ heißt *Äquivalenzklasse* (bzgl. E), falls es ein $v \in V$ mit $W = [v]_E$ gibt. Das Element v wird dann ein *Vertreter*, bzw. *Repräsentant* seiner Äquivalenzklasse W genannt. Wir kommen zu der herausragenden Eigenschaft einer Äquivalenzrelation (bzgl. E):

Satz 7.23. Sei E eine Äquivalenzrelation über der Menge V und $[v]_E, [w]_E$ seien zwei beliebige Äquivalenzklassen. Dann stimmen die beiden Äquivalenzklassen entweder überein (d.h. $[v]_E = [w]_E$) oder die Klassen sind disjunkt (d.h. $[v]_E \cap [w]_E = \emptyset$).

Beweis: Wir nehmen an, dass die Äquivalenzklassen $[v]_E$ und $[w]_E$ ein gemeinsames Element u besitzen. Dann gilt

1. $(v, u) \in E$ und $(w, u) \in E$ nach Definition der Äquivalenzklassen,
2. Es folgt $(u, w) \in E$, denn E ist symmetrisch.
3. E ist transitiv und wir erhalten $(v, w) \in E$ aus $(v, u) \in E$ und $(u, w) \in E$.
4. Mit dem gleichen Argument folgt für einen beliebigen mit v äquivalenten Knoten x aus $(v, x) \in E$ (d.h. $(x, v) \in E$) und $(v, w) \in E$ auch $(x, w) \in E$, bzw. $(w, x) \in E$. Wir erhalten $[v]_E \subseteq [w]_E$.
5. Ebenso erhalten wir $[v]_E \subseteq [w]_E$ und wir haben die Gleichheit $[v]_E = [w]_E$ nachgewiesen.

□

Falls V endlich und nicht leer ist, folgt daraus, dass es eine Zahl $k \in \mathbb{N}_{>0}$ und Äquivalenzklassen $W_1, \ldots, W_k$ geben muss, so dass V eine disjunkte Vereinigung der verschiedenen Äquivalenzklassen ist, d.h. $V = W_1 \dot\cup \cdots \dot\cup W_k$ gilt.

Die Zahl k wird auch **Index** von E genannt. Der Index der Äquivalenzrelation E gibt an, wie viele verschiedene Äquivalenzklassen E besitzt. Index

Beispielsweise hat die Gleichmächtigkeits-Relation aus Beispiel 7.21 ((b)) den Index $|M| + 1$.

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA. Wir erinnern an die Definition der Verschmelzungsrelation: Zwei Zustände $p, q \in Q$ sind äquivalent, kurz $p \equiv_A q$, wenn

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \iff \widehat{\delta}(q, w) \in F. \tag{7.1}$$

Satz 7.24. Sei A ein DFA. Dann ist die Verschmelzungsrelation $\equiv_A$ eine Äquivalenzrelation.

Insbesondere ist die Zustandsmenge Q eine disjunkte Vereinigung von Äquivalenzklassen und unsere Idee, äquivalente Zustände zu einem einzigen Zustand zu verschmelzen, erscheint sinnvoll. Wenn alles klappt, wenn wir also die Zustandsübergänge nach Verschmelzung „vernünftig“ setzen können, dann ist der „Äquivalenzklassenautomat“, also der DFA nach Verschmelzung mit A äquivalent. Natürlich stimmt die Anzahl seiner Zustände mit dem Index der Verschmelzungsrelation überein. Ach wär das toll, wenn der Index die minimale Zustandszahl wäre.

Beweis: Wenn wir $q = p$ in (7.2) annehmen, erhalten wir $p \equiv_A p$ als Konsequenz und die Verschmelzungsrelation ist reflexiv. Die Symmetrie folgt auch sofort, denn aus $p \equiv_A q$, d.h

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p, w) \in F \iff \widehat{\delta}(q, w) \in F.$$

folgt $q \equiv_A p$, d.h.

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(q, w) \in F \iff \widehat{\delta}(p, w) \in F.$$

Seien p, q, r beliebige Zustände. Um die Transitivität nachzuweisen, müssen wir $p \equiv_A q, q \equiv_A r$ annehmen und dann $p \equiv_A r$ zeigen. Nach Annahme gilt also

$$\text{f.a. Worte } w \in \Sigma^* : \big(\widehat{\delta}(p,w) \in F \iff \widehat{\delta}(q,w) \in F\big) \wedge \big(\widehat{\delta}(q,w) \in F \iff \widehat{\delta}(r,w) \in F\big).$$

und damit folgt

$$\text{f.a. Worte } w \in \Sigma^* : \widehat{\delta}(p,w) \in F \iff \widehat{\delta}(r,w) \in F.$$

Wir haben die Behauptung $p \equiv_A r$ gezeigt. □

7.3.2. Die Verschmelzungsrelation

Sei der DFA $A = (\Sigma, Q, \delta, q_0, F)$ gegeben. Um die Zustandszahl von A zu reduzieren, führen wir die folgenden Schritte durch:

1. Zuerst bestimmen wir die Äquivalenzklassen der Verschmelzungsrelation $\equiv_A$.
2. Für jede Äquivalenzklasse von $\equiv_A$ verschmelzen wir alle Zustände der Klasse zu einem einzigen Zustand und fügen „entsprechende“ Übergänge ein. Den neuen Automaten nennen wir $\mathbf{A'}$ und bezeichnen ihn als den **Äquivalenzklassenautomaten** von A.
   - Wie sollen wir die Zustandsübergänge von A' definieren, so dass A und A' dieselbe Sprache berechnen?

Äquivalenzklassen-automaten

Wenn der Index von $\equiv_A$, und damit die Zustandszahl von A', mit der minimalen Zustandszahl übereinstimmt, dann ist A' **minimal**! Natürlich müssen wir uns auch fragen, ob wir $\equiv_A$ und A' effizient berechnen können.

Wie können wir die Äquivalenzklassen der Verschmelzungsrelation berechnen?

Wir bestimmen alle Paare **inäquivalenter** Zustände.

Definition 7.25. Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA mit Zuständen $p, q \in Q$.
Wir sagen, dass das Wort $w \in \Sigma^*$ die Zustände p und q trennt, bzw. ein **Zeuge** für die Inäquivalenz von p und q ist, wenn

Zeuge

$$\Big(\widehat{\delta}(p,w) \in F \wedge \widehat{\delta}(q,w) \notin F\Big) \vee \Big(\widehat{\delta}(p,w) \notin F \wedge \widehat{\delta}(q,w) \in F\Big).$$

Um alle Paare inäquivalenter Zustände zu bestimmen, wenden wir den **Verschmelzungs-algorithmus** an.

Verschmelzungs-algorithmus

1. **Markiere** alle Paarmengen $\{p, q\}$ mit $p \in F$ und $q \notin F$ (als inäquivalent).
   - Es ist $\delta(p, \varepsilon) \in F$ und $\delta(q, \varepsilon) \notin F$.
   - $w = \varepsilon$ ist Zeuge für die Inäquivalenz von p und q.
2. Wenn die Paarmenge $\{p, q\}$ bereits markiert wurde und

   $$\text{wenn } \delta(r,a) = p \text{ sowie } \delta(s,a) = q$$

   für ein $a \in \Sigma$, dann **markiere** $\{r, s\}$.

- Da $p \not\equiv_A q$, gibt es einen **Zeugen** w mit

$$\big(\widehat{\delta}(p,w) \in F \text{ und } \widehat{\delta}(q,w) \notin F\big) \text{ oder } \big(\widehat{\delta}(p,w) \notin F \text{ und } \widehat{\delta}(q,w) \in F\big).$$

- Das Wort aw bezeugt, dass r und s inäquivalent sind.

3. Halte, wenn keine neuen Paarmengen $\{r,s\}$ markiert werden können.
   - Unsere Hoffnung ist, dass $p \not\equiv_A q$ genau dann gilt, wenn die Paarmenge $\{p,q\}$ markiert wurde.

Der Verschmelzungsalgorithmus funktioniert!

Satz 7.26. Der Verschmelzungsalgorithmus findet alle Paare inäquivalenter Zustände.

Beweis: Sei P die Menge aller Paare $\{r,s\}$ inäquivalenter Zustände, die aber von unserem Verfahren *nicht* gefunden werden. Wir müssen zeigen, dass P leer ist! Wir führen einen Beweis durch Widerspruch und nehmen an, dass P nicht-leer ist. $\{p,q\} \in P$ habe unter allen Paaren in P einen *kürzesten* Zeugen w.
Fall 1: Wenn $w = \varepsilon$, dann ist

- $\big(\delta(p,\varepsilon) \in F \text{ und } \delta(q,\varepsilon) \notin F\big)$ oder $\big(\delta(p,\varepsilon) \notin F \text{ und } \delta(q,\varepsilon) \in F\big)$,
- bzw. $\big(p \in F \text{ und } q \notin F\big)$ oder $\big(p \notin F \text{ und } q \in F\big)$.

Aber dann haben wir $\{p,q\}$ im Schritt 1. markiert. $\lightning$
Fall 2: Wenn $w = au$ für den Buchstaben $a \in \Sigma$, dann ist

- $\big(\widehat{\delta}(p,au) \in F \text{ und } \widehat{\delta}(q,au) \notin F\big)$ oder $\big(\widehat{\delta}(p,au) \notin F \text{ und } \widehat{\delta}(q,au) \in F\big)$,
- bzw. $\big(\widehat{\delta}(\delta(p,a),u) \in F \text{ und } \widehat{\delta}(\delta(q,a),u) \notin F\big)$ oder $\big(\widehat{\delta}(\delta(p,a),u) \notin F \text{ und } \widehat{\delta}(\delta(q,a),u) \in F\big)$.

Dann gilt $\delta(p,a) \not\equiv_A \delta(q,a)$ mit dem kürzeren Zeugen u: Aber $\{\delta(p,a),\delta(q,a)\}$ muss nach Annahme bereits markiert sein, und wir werden darauffolgend $\{p,q\}$ markieren. $\lightning$ □

Ist unser Algorithmus effizient? Die Anzahl aller Zustandspaare ist durch $|Q|^2$ nach oben beschränkt. Wir werden deshalb höchstens $|Q|^2$ Markierungsschritte benötigen.
Wir haben die Verschmelzungsrelation erfolgreich berechnet und bestimmen jetzt den Äquivalenzklassenautomaten.

7.3.3. Der Aquivalenzklassenautomat

Für Zustand $p \in Q$ bezeichnet

$$[p]_A := \big\{q \in Q \mid p \equiv_A q\big\}$$

die Äquivalenzklasse von p. Der **Äquivalenzklassenautomat**

$$A' = (\Sigma, Q', \delta', q_0', F')$$

für den DFA $A = (\Sigma, Q, \delta, q_0, F)$ besitzt

- die Zustandsmenge
$$Q' := \{\, [p]_A \mid p \in Q \,\},$$
- den Anfangszustand $q_0' := [q_0]_A$,
- die Menge $F' := \{\, [p]_A \mid p \in F \,\}$ der akzeptierenden Zustände und
- das Programm δ' mit
$$\delta'([p]_A, a) := [\ \delta(p,a)\ ]_A$$
für alle $q \in Q$ und $a \in \Sigma$.

Wir haben alle Zustände einer Äquivalenzklasse $[p]_A$ zu einem einzigen Zustand zusammengefasst und haben diesen Zustand $[p]_A$ genannt. Folgerichtig arbeiten wir deshalb mit der Menge Q' aller Äquivalenzklassen von $\equiv_A$.

Als Menge der akzeptierenden Zustände des Äquivalenzklassenautomaten haben wir alle mit einem akzeptierenden Zustand von A äquivalenten Zustände gewählt: Das scheint vernünftig, aber ist denn wirklich jeder mit einem akzeptierenden Zustand von A äquivalenter Zustand auch selbst akzeptierend?

Um die Übergangsfunktion δ' für eine Äquivalenzklasse $[p]_A$ und einen Buchstaben $a \in \Sigma$ zu definieren, sind wir fast gezwungen, die Äquivalenzklasse des Zustands $\delta(p,a)$ zu wählen, und genau das haben wir getan. Aber vorsichtig, gilt denn $\delta(p,a) \equiv_A \delta(q,a)$ für je zwei äquivalente Zustände p, q? Wenn nicht, dann wäre unsere Definition von δ' abhängig vom Vertreter der Äquivalenzklasse und das würde nichts Gutes verheißen!

Wir zeigen zuerst, dass die Definition von δ' unabhängig vom Vertreter der Äquivalenzklasse ist. Danach können wir beweisen, dass A und A' äquivalente DFAs sind.

Satz 7.27. *$A = (\Sigma, Q, \delta, q_0, F)$ sei ein DFA. Für alle Zustände $p, q \in Q$ mit $[p]_A = [q]_A$ gilt*
$$\delta(p,a) \equiv_A \delta(q,a).$$

Beweis: Seien also $p, q \in Q$ äquivalente Zustände (d.h. es gilt $p \equiv_A q$). Dann folgt:
$$\begin{aligned} p \equiv_A q \quad &\Longrightarrow \quad \text{für alle } w' \in \Sigma^* \text{ gilt: } \widehat{\delta}(p,w') \in F \iff \widehat{\delta}(q,w') \in F \\ &\Longrightarrow \quad \text{für alle } w \in \Sigma^* \text{ gilt: } \widehat{\delta}(p,aw) \in F \iff \widehat{\delta}(q,aw) \in F. \end{aligned} \tag{7.2}$$
Für die letzte Implikation haben wir die Aussage der ersten Implikation auf alle w' der Form $w' = aw$ eingeschränkt. Jetzt beachte $\widehat{\delta}(p,aw) = \widehat{\delta}(\delta(p,a),w)$ und $\widehat{\delta}(q,aw) = \widehat{\delta}(\delta(q,a),w)$:
$$\begin{aligned} (7.2) \quad &\Longrightarrow \quad \text{für alle } w \in \Sigma^* \text{ gilt: } \widehat{\delta}\big(\delta(p,a),w\big) \in F \iff \widehat{\delta}\big(\delta(q,a),w\big) \in F \\ &\Longrightarrow \quad \delta(p,a) \equiv_A \delta(q,a). \end{aligned}$$
Somit hängt die Definition von δ' nicht ab von der speziellen Wahl eines Vertreters der Äquivalenzklasse. □

Wir können jetzt nachweisen, dass A und A' äquivalente DFAs sind!

Satz 7.28. *$A = (\Sigma, Q, \delta, q_0, F)$ sei ein DFA. Dann sind A und A' äquivalente DFAs, d.h. es gilt*
$$L(A) = L(A').$$

Beweis: Wir zeigen die Behauptung in drei Schritten.

1. Schritt 1: Zeige, dass
$$\widehat{\delta'}([q_0]_A, w) = [\widehat{\delta}(q_0, w)]_A$$
für alle $w \in \Sigma^*$ gilt. Wenn wir also wissen möchten, in welchem Zustand sich A' nach Lesen des Worts w befindet, dann bestimmen wir den Zustand $p = \delta(q_0, w)$ von A nach Lesen von w. A' befindet sich dann im Zustand $[p]_A$ und das ist alles andere als überraschend.

2. In den Schritten 2 und 3 zeigen wir die Inklusionen $L(A) \subseteq L(A')$ und $L(A') \subseteq L(A)$.

Beweis von Schritt 1: Durch vollständige Induktion über die Länge von w.

Induktionsanfang:
Betrachte $w = \varepsilon$. Nach Definition von δ' gilt: $\widehat{\delta'}([q_0]_A, \varepsilon) = [q_0]_A = [\widehat{\delta}(q_0, \varepsilon)]_A$.

Induktionsschritt:
Betrachte $w = ua$ mit $u \in \Sigma^*$ und $a \in \Sigma$. Nach Induktionsannahme gilt $\widehat{\delta'}([q_0]_A, u) = [\widehat{\delta}(q_0, u)]_A$.
Wir müssen zeigen
Beh.: $\widehat{\delta'}([q_0]_A, ua) = [\widehat{\delta}(q_0, ua)]_A$.
Beweis:

$$\begin{aligned}
\widehat{\delta'}([q_0]_A, ua) &= \delta'\big(\widehat{\delta'}([q_0]_A, u), a\big) \\
&\overset{\text{Induktionsannahme}}{=} \delta'\big([\widehat{\delta}(q_0, u)]_A, a\big) \\
&= \delta'([p]_A, a), \quad \texttt{für } p := \widehat{\delta}(q_0, u) \\
&\overset{\text{Definition von } \delta'}{=} [\delta(p, a)]_A \;=\; [\delta\big(\widehat{\delta}(q_0, u), a\big)]_A \;=\; [\widehat{\delta}(q_0, ua)]_A.
\end{aligned}$$

$\square_{\text{Schritt 1}}$

Beweis von Schritt 2: Zeige, dass $L(A) \subseteq L(A')$ gilt. Sei also w ein beliebiges Wort in $L(A)$.

1. A akzeptiert die Eingabe w, d.h. es gilt: $p := \widehat{\delta}(q_0, w) \in F$.
2. Nach Schritt 1 gilt: $\widehat{\delta'}([q_0]_A, w) = [\widehat{\delta}(q_0, w)]_A = [p]_A$.
3. Nach Definition von F' gilt: $[p]_A \in F'$, denn $p \in F$.

Somit wird w von A' akzeptiert, d.h. es gilt $w \in L(A')$.

$\square_{\text{Schritt 2}}$

Beweis von Schritt 3: Zeige, dass $L(A') \subseteq L(A)$ gilt. Wir zeigen eine Zwischenbehauptung.

Beh: Wenn $p \equiv_A q$ und $p \in F$, dann auch $q \in F$.
Beweis: Wenn $p \equiv_A q$, dann folgt aus der Definition der Verschmelzungsrelation $\widehat{\delta}(p, w) \in F \iff \widehat{\delta}(q, w) \in F$ für alle Worte $w \in \Sigma^*$. Also gilt insbesondere: $\delta(p, \varepsilon) = p \in F \iff \delta(q, \varepsilon) = q \in F$.

Zurück zu Schritt 3. Wir nehmen an, dass A' die Eingabe w akzeptiert, d.h. es gilt: $\widehat{\delta'}([q_0]_A, w) \in F'$. Wir müssen $w \in L(A)$ zeigen, d.h., dass $\widehat{\delta}(q_0, w) \in F$ gilt.

Nach Schritt 1 gilt:
$$\widehat{\delta'}([q_0]_A, w) = [\widehat{\delta}(q_0, w)]_A.$$

Nach Annahme ist $\widehat{\delta'}([q_0]_A, w) \in F'$. Nach Definition von F' muss es einen Zustand $p \in F$ mit $p \equiv_A \widehat{\delta}(q_0, w)$ geben. Also folgt aus der Zwischenbehauptung $\widehat{\delta}(q_0, w) \in F$, d.h. $w \in L(A)$.

$\square_{\text{Schritt 3}}$

$L(A) \subseteq L(A')$ gilt nach Schritt 2 und $L(A') \subseteq L(A)$ nach Schritt 3. Es folgt $L(A) = L(A')$ und die Behauptung von Satz 7.28 ist gezeigt.

□

Der Äquivalenzklassenautomat tut genau das, was wir von ihm verlangen. Allerdings müssen wir später noch zeigen, dass A' der kleinste zu A äquivalente DFA ist.

Bemerkung 7.29. Wenn wir den Äquivalenzklassenautomat für einen Moore- oder Mealy-Automaten bestimmen möchten, dann müssen wir die Verschmelzungsrelation dem Ausgabeverhalten des Automaten anpassen: Für die Ausgabefunktion λ führen wir die Erweiterung $\widehat{\lambda}$ für Worte $w \in \Sigma^*$ ein und definieren dann die Äquivalenz von Zuständen p, q durch

$$p \equiv_A q \iff \text{f.a. Worte } w \in \Sigma^* : \widehat{\lambda}(p, w) = \widehat{\lambda}(r, w).$$

7.3.4. Der Minimierungsalgorithmus

Um einen DFA A zu minimieren, entfernen wir zuerst alle vom Startzustand aus nicht erreichbaren Zustände, berechnen dann die Äquivalenzklassen von $\equiv_A$ und bestimmen den Äquivalenzklassenautomaten A' mit Hilfe der Klassen.

Wir bestimmen die Äquivalenzklassen von $\equiv_A$ iterativ. Zuerst beobachte, dass Zustände $p \in F$ und $q \notin F$ nicht in derselben Klasse liegen können. Wichtig ist jetzt die Feststellung, dass auch $p' \neq q'$ nicht zur selben Klasse gehören, wenn $p = \delta(p', x)$ und $q = \delta(q', x)$ gilt und bekannt ist, dass p, q inäquivalent sind.

Minimierungsalgorithmus

Hier ist unser **Minimierungsalgorithmus**.

Eingabe: Ein DFA $A = (Q, \Sigma, \delta, q_0, F)$.

Schritt 1: Entferne aus A alle **überflüssigen** Zustände, d.h. alle Zustände, die nicht von q_0 aus erreichbar sind.

Schritt 2: Bestimme alle Paare $\{p, q\}$ mit $p, q \in Q$ und $p \not\equiv_A q$:

1. Markiere alle Paarmengen in $M_0 := \{\ \{p, q\}\ :\ p \in F,\ q \in Q \setminus F\ \}$; setze $i := 0$
2. Wiederhole bis $M_i = \emptyset$
   (a) Für alle $\{p, q\} \in M_i$ und für alle $x \in \Sigma$ tue folgendes:
   (b) Markiere $\{p', q'\}$ für alle $p' \neq q'$ mit $\delta(p', x) = p$ und $\delta(q', x) = q$.
   (c) Sei M_{i+1} die Menge aller hierbei **neu** markierten Paarmengen.
   (d) $i := i + 1$
3. **Ausgabe:** $M := M_0 \cup \cdots \cup M_{i-1}$.

Schritt 3: Konstruiere $A' := (Q', \Sigma, \delta', q_0', F')$:

$$Q' := \{\ [q]_A\ :\ q \in Q\},\ \text{ wobei }\ [q]_A = \{\ p \in Q\ :\ \{p, q\} \notin M\ \}$$

$$q_0' := [q_0]_A,\quad F' := \{\ [q]_A\ :\ q \in F\ \}$$

$$\delta' : Q' \times \Sigma \to Q' \ \text{ mit }\ \delta'\big([q]_A, a\big) := [\delta(q, a)]_A \ \text{ für alle } q \in Q \text{ und } x \in \Sigma.$$

Wir haben eine Beschreibung des Minmierungsalgorithmus gewählt, so dass eine Ausführung „per Hand“ leicht gelingt. Eine effiziente Implementierung für den Rechner nutzt aus, dass ein Paar $\{p, q\}$ genau dann zu M_{i+1} gehört, wenn p, q nicht von Zeugen der Länge höchstens i, wohl aber von einem Zeugen der Länge $i + 1$ getrennt wird. Um diese Beobachtung zu motivieren, führen wir die Äquivalenzrelationen $\equiv_A^i$ ein.

Definition 7.30. Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DFA. Für jede natürliche Zahl i definieren wir

$$p \equiv_A^i q \quad :\Longleftrightarrow \quad \text{für alle Worte } w \text{ der } \underline{\text{Länge höchstens } i} \text{ gilt} \left(\hat{\delta}(p, w) \in F \Longleftrightarrow \hat{\delta}(q, w) \in F \right).$$

Es gilt also $p \equiv_A^i q$ genau dann, wenn p und q von keinem Zeugen der Länge höchstens i getrennt werden können.

Bemerkung 7.31. (Die Mengen M_i und Zeugenlängen)
Es ist $p \equiv_A^0 q \Longleftrightarrow \Big(\big(p \in F \text{ und } q \in F\big) \text{ oder } \big(p \notin F \text{ und } q \notin F\big) \Big)$. Also folgt

$$\{p, q\} \notin M_0 \Longleftrightarrow p \equiv_A^0 q.$$

Andererseits folgt

$$p \equiv_A^{i+1} q \iff p \equiv_A^i q \text{ und für alle } a \in \Sigma \text{ gilt: } \delta(p, a) \equiv_A^i \delta(q, a)$$

aus der Definition von $\equiv_A^{i+1}$, denn $p \equiv_A^{i+1} q$ gilt genau dann, wenn p, q nicht von Zeugen der Länge höchstens i und auch nicht durch Zeugen aw für ein Wort $w \in \Sigma^i$ trennbar sind. Jetzt zeige mit vollständiger Induktion nach i, dass

$$\{p, q\} \notin M_0 \cup \cdots \cup M_i \iff p \equiv_A^i q$$

gilt. Also wird $\{p, q\}$ genau dann in einer der Iterationen $0, \ldots, i$ markiert, wenn $p \not\equiv_A^i q$, d. h. wenn p und q von einem Zeugen der Länge höchstens i getrennt werden.

Was „tut“ der Minimierungsalgorithmus? Er versucht, aus den Äquivalenzklassen von $\equiv_A^i$ einen DFA zu bauen. Dieser Versuch scheitert genau dann, wenn äquivalente Zustände keine äquivalenten Nachfolger besitzen, d.h. wenn gilt

$$\delta(p, a) \not\equiv_A^i \delta(q, a) \text{ für einen Buchstaben } a \in \Sigma \text{ und Zustände } p, q \text{ mit } p \equiv_A^i q.$$

Aber dann wird $\{p, q\}$ zur Menge M_{i+1} gehören und p, q werden getrennt.

Bemerkung 7.32. (Wie viele Iterationen gibt es?)
Entweder stimmen $\equiv_A^i$ und $\equiv_A^{i+1}$ überein – und unser Algorithmus terminiert – oder mindestens eine Äquivalenzklasse von $\equiv_A^i$ wird in zwei oder mehr Klassen zerlegt.

- $\equiv_A^{i+1}$ besitzt also mindestens eine Äquivalenzklasse mehr als $\equiv_A^i$, falls $\equiv_A^i$ und $\equiv_A^{i+1}$ noch nicht übereinstimmen.

- Aber dann wird die Schleife in Schritt 2 des Algorithmus höchstens $|Q| - 1$ mal wiederholt, denn erstens besitzt $\equiv_A^0$ ja bereits zwei Äquivalenzklassen und zweitens kann es nicht mehr als $|Q|$ Äquivalenzklassen geben.

Schließlich kann man zeigen, dass die Anzahl der Operationen pro Iteration von Schritt (2) höchstens proportional zur Anzahl $|\Sigma| \cdot |Q|$ der Zustandübergänge ist.

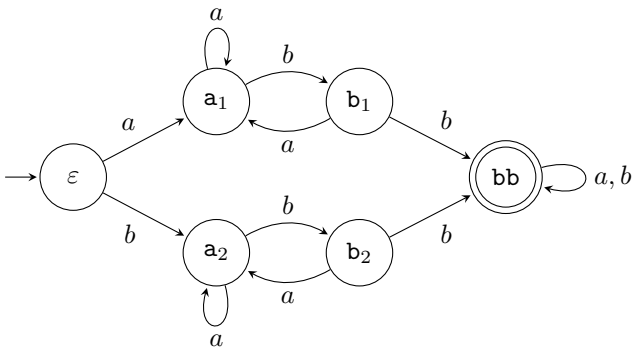
Aber dann ist unser Algorithmus **effizient** und benötigt eine Anzahl von Operationen, die höchstens proportional zu $|\Sigma| \cdot |Q|^2$ ist. Wir haben somit eine relativ schnelle Implementierung mit (fast-)quadratischer Laufzeit erhalten. Es gibt sogar Minimierungsalgorithmen wie den Algorithmus von Hopcroft, die eine (fast-)lineare Laufzeit besitzen. Mehr erfahren Sie in der Veranstaltung „Theoretische Informatik 2“.

Die kritische Frage

„Ist A' minimal?“

muss noch geklärt werden. Aber zuerst rechnen wir einige Beispiele durch.

Beispiel 7.33. Wir möchten den folgenden DFA minimieren:



Die Menge $\mathbf{M_0}$: bb ist der einzige akzeptierende Zustand. Deshalb hat die Menge M_0 die Form

$$M_0 = \{\{\varepsilon, \mathtt{bb}\}, \{\mathtt{a}_1, \mathtt{bb}\}, \{\mathtt{a}_2, \mathtt{bb}\}, \{\mathtt{b}_1, \mathtt{bb}\}, \{\mathtt{b}_2, \mathtt{bb}\}\}.$$

Was haben wir gelernt? $\{\mathtt{bb}\}$ ist eine Klasse der Verschmelzungsrelation, die restlichen Klassen sind disjunkte Teilmengen von $\{\varepsilon, \mathtt{a}_1, \mathtt{a}_2, \mathtt{b}_1, \mathtt{b}_2\}$.

Wir veranschaulichen die Menge aller Paare von Zuständen durch eine zweidimensionale Tabelle und vermerken alle zu M_0 gehörenden Paare.

$\mathtt{a}_1$					
$\mathtt{a}_2$					
$\mathtt{b}_1$					
$\mathtt{b}_2$					
bb	M_0	M_0	M_0	M_0	M_0
	ε	$\mathtt{a}_1$	$\mathtt{a}_2$	$\mathtt{b}_1$	$\mathtt{b}_2$

Die Menge $\mathbf{M_1}$: Wir inspizieren alle nicht mit M_0 markierten Positionen der Tabelle und überprüfen, ob wir eine mit M_0 markierte Position erhalten, wenn der Buchstabe a (bzw. der

Buchstabe b) gelesen wird. Dies ist der Fall zum Beispiel für die Position in Zeile $\mathtt{b_1}$ und Spalte ε, die zur Paarmenge $\{\mathtt{b_1}, \varepsilon\}$ gehört: Es ist $\delta(\mathtt{b_1}, \mathtt{b}) = \mathtt{bb}$ und $\delta(\varepsilon, \mathtt{b}) = \mathtt{a_2}$. Also gehört $\{\mathtt{b_1}, \varepsilon\}$ zur Menge M_1, denn $\{\mathtt{bb}, \mathtt{a_2}\}$ gehört zur Menge M_0.

$\mathtt{a_1}$					
$\mathtt{a_2}$					
$\mathtt{b_1}$	M_1	M_1	M_1		
$\mathtt{b_2}$	M_1	M_1	M_1		
$\mathtt{bb}$	M_0	M_0	M_0	M_0	M_0
	ε	$\mathtt{a_1}$	$\mathtt{a_2}$	$\mathtt{b_1}$	$\mathtt{b_2}$

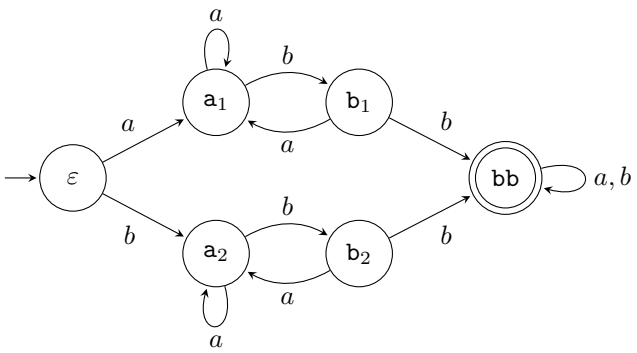
Die Menge $\mathbf{M_2}$: Diesmal inspizieren wir alle nicht mit M_0 oder M_1 markierten Positionen der Tabelle und überprüfen, ob wir eine mit M_1 markierte Position erhalten, wenn der Buchstabe a (bzw. der Buchstabe b) gelesen wird. Dies ist der Fall zum Beispiel für die Position in Zeile $\mathtt{a_1}$ und Spalte ε, die zur Paarmenge $\{\mathtt{a_1}, \varepsilon\}$ gehört: Es ist $\delta(\mathtt{a_1}, \mathtt{b}) = \mathtt{b_1}$ und $\delta(\varepsilon, \mathtt{b}) = \mathtt{a_2}$. Also gehört $\{\mathtt{a_1}, \varepsilon\}$ zur Menge M_2, denn $\{\mathtt{b_1}, \mathtt{a_2}\}$ gehört zur Menge M_1.

$\mathtt{a_1}$	M_2				
$\mathtt{a_2}$	M_2				
$\mathtt{b_1}$	M_1	M_1	M_1		
$\mathtt{b_2}$	M_1	M_1	M_1		
$\mathtt{bb}$	M_0	M_0	M_0	M_0	M_0
	ε	$\mathtt{a_1}$	$\mathtt{a_2}$	$\mathtt{b_1}$	$\mathtt{b_2}$

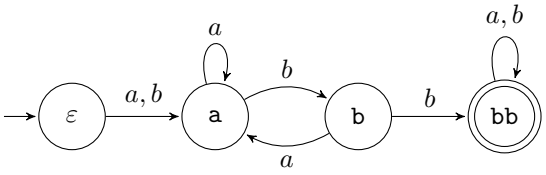
Die Menge $\mathbf{M_3}$: Wir müssen nur die unmarkierten Positionen (mit den jeweiligen Paarmengen $\{\mathtt{a_2}, \mathtt{a_1}\}$ und $\{\mathtt{b_2}, \mathtt{b_1}\}$) inspizieren. Wir beginnen mit $\{\mathtt{a_2}, \mathtt{a_1}\}$ und erhalten die unmarkierte Paarmenge $\{\mathtt{a_2}, \mathtt{a_1}\}$ für den Buchstaben a, bzw. die unmarkierte Paarmenge $\{\mathtt{b_2}, \mathtt{b_1}\}$ für den Buchstaben b: Die Paarmenge $\{\mathtt{a_2}, \mathtt{a_1}\}$ bleibt unmarkiert.

Schließlich betrachten wir die Paarmenge $\{\mathtt{b_2}, \mathtt{b_1}\}$. Für den Buchstaben a erhalten wir die unmarkierte Paarmenge $\{\mathtt{a_2}, \mathtt{a_1}\}$ und für den Buchstaben b die Menge $\{\mathtt{bb}, \mathtt{bb}\}$, die natürlich nicht markiert werden kann.

Fazit: Es ist $M_3 = \emptyset$ und wir kennen die Verschmelzungsrelation: Die einzigen nicht-trivialen Äquivalenzbeziehungen sind $\mathtt{a_1} \equiv_{\mathtt{A}} \mathtt{a_2}$ und $\mathtt{b_1} \equiv_{\mathtt{A}} \mathtt{b_2}$, die weiteren Zustände ε und $\mathtt{bb}$ bilden jeweils ihre eigene Äquivalenzklasse. Wir erhalten nach Verschmelzung der Zustände $\mathtt{a_1}, \mathtt{a_2}$, bzw. $\mathtt{b_1}, \mathtt{b_2}$ im Ausgangsautomaten



den Äquivalenzklassenautomaten



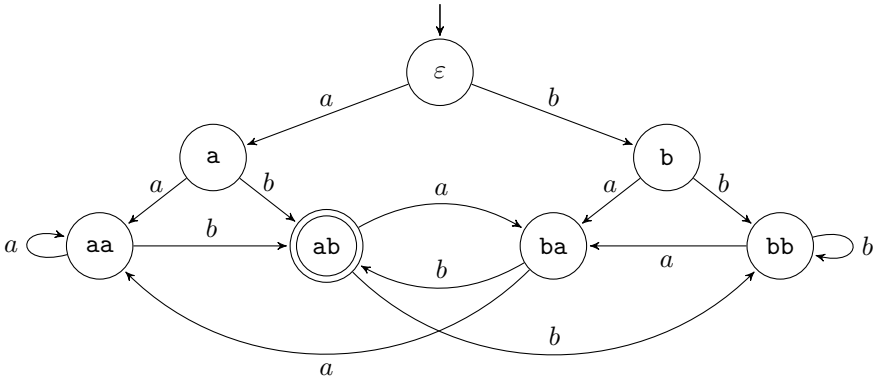
Genügt es, wenn wir nur Zustände mit identischen Nachfolgezuständen verschmelzen? Nein, der Automat aus dem letzten Beispiel ist ein Gegenbeispiel.

Beispiel 7.34. Wie stellt man fest, ob ein Wort das Suffix ab besitzt? Wir arbeiten mit dem Alphabet $\Sigma = \{a, b\}$ und speichern in unserem ersten Ansatz die beiden zuletzt gelesenen Buchstaben im aktuellen Zustand ab. Wir benutzen deshalb die Zustände

$$Q = \{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ab}, \mathtt{ba}, \mathtt{bb}\}$$

- mit Startzustand $q_0 = \varepsilon$ („wir haben noch nichts gelesen“ und
- dem akzeptierenden Zustand ab: Die beiden letzten Buchstaben sind ab.

Hier ist unser erster Automat.



Die Menge $\mathbf{M_0}$: ab ist der einzige akzeptierende Zustand. Deshalb hat die Menge M_0 die Form

$$M_0 = \{\{\varepsilon, \mathtt{ab}\}, \{\mathtt{a}, \mathtt{ab}\}, \{\mathtt{b}, \mathtt{ab}\}, \{\mathtt{aa}, \mathtt{ab}\}, \{\mathtt{ba}, \mathtt{ab}\}, \{\mathtt{bb}, \mathtt{ab}\}\}.$$

Was haben wir gelernt? {ab} ist eine Klasse der Verschmelzungsrelation, die restlichen Klassen sind disjunkte Teilmengen von $\{\varepsilon, \mathtt{a}, \mathtt{b}, \mathtt{aa}, \mathtt{ba}, \mathtt{bb}\}$.

Wir veranschaulichen die Menge aller Paare von Zuständen wieder durch eine zweidimensionale Tabelle und vermerken alle zu M_0 gehörenden Paare.

a						
b						
aa						
ab	M_0	M_0	M_0	M_0		
ba					M_0	
bb					M_0	
	ε	a	b	aa	ab	ba

Die Menge $\mathbf{M_1}$: Dazu inspizieren wir alle nicht mit M_0 markierten Positionen der Tabelle und überprüfen, ob wir eine mit M_0 markierte Position erhalten, wenn der Buchstabe a (bzw. der Buchstabe b) gelesen wird. Dies ist der Fall zum Beispiel für die Position oben links, die die Paarmenge $\{\mathtt{a}, \varepsilon\}$ repräsentiert. Es ist $\delta(\mathtt{a}, \mathtt{b}) = \mathtt{ab}$ und $\delta(\varepsilon, \mathtt{b}) = \mathtt{b}$: $\{\mathtt{a}, \varepsilon\}$ gehört zur Menge M_1, weil $\{\mathtt{ab}, \mathtt{b}\}$ zur Menge M_0 gehört.

Um nicht jede nicht markierte Position der Tabelle gesondert zu diskutieren, machen wir die folgende Beobachtung. Für $x = a$ oder $x = b$ gilt: Wenn $\delta(p', x) = p, \delta(q', x) = q$ und $\{p, q\} \in M_0$, dann ist $\{p', q'\} \in M_1$. Der Zustand $\mathtt{ab}$ wird aber genau dann erreicht, wenn der Buchstabe $x = b$ in den Zuständen $\mathtt{a}, \mathtt{aa}$ oder $\mathtt{ba}$ gelesen wird. Demgemäß erhalten wir für M_1 die Tabelle

a	M_1					
b		M_1				
aa	M_1		M_1			
ab	M_0	M_0	M_0	M_0		
ba	M_1		M_1		M_0	
bb		M_1		M_1	M_0	M_1
	ε	a	b	aa	ab	ba

Was haben wir gelernt?

1. Kein Zustand in der Menge $\mathtt{a}, \mathtt{aa}, \mathtt{ba}$ kann mit einem Zustand in der Menge $\varepsilon, \mathtt{b}, \mathtt{bb}$ äquivalent sein.

2. Die weiteren Äquivalenzklassen der Verschmelzungsrelation sind disjunkte Teilmengen entweder von $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ oder von $\{\varepsilon, \mathtt{b}, \mathtt{bb}\}$.

Lassen sich die Zustände in $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ (bzw. in $\{\varepsilon, \mathtt{b}, \mathtt{bb}\}$) voneinander trennen?

Die Menge $\mathbf{M_2}$:

1. Die Zustände $\mathtt{a}, \mathtt{aa}, \mathtt{ba}$ können nicht voneinander getrennt werden, da sie alle unter a auf den Zustand $\mathtt{aa}$ und unter b auf den Zustand $\mathtt{ab}$ führen.

2. $\mathtt{b}$ und $\mathtt{bb}$ lassen sich ebenfalls nicht mehr trennen. Beide führen unter a auf $\mathtt{ba}$ und unter b auf $\mathtt{bb}$.

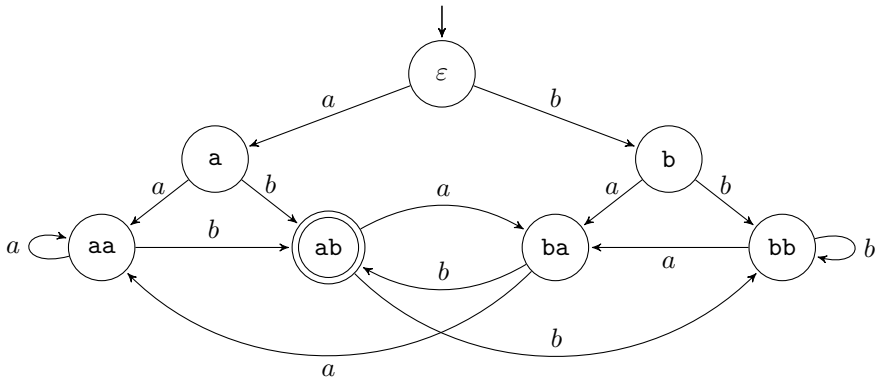
3. Aber auch ε führt unter a auf die Klasse $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ und unter b auf $\mathtt{b}$.

Es ist $M_2 = \emptyset$, denn ε lässt sich nicht von $\mathtt{b}, \mathtt{bb}$ trennen. Wie sehen die verschiedenen Äquivalenzklassen aus?

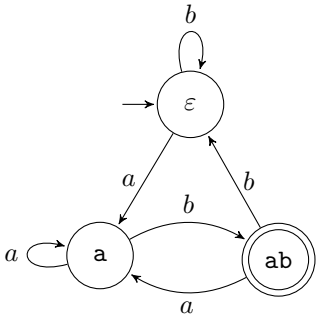
(a) $\{\mathtt{ab}\}$ ist die Klasse des einzigen akzeptierenden Zustands,

(b) $\{\mathtt{a}, \mathtt{aa}, \mathtt{ba}\}$ und $\{\varepsilon, \mathtt{b}, \mathtt{bb}\}$ sind die beiden verbleibenden Klassen.

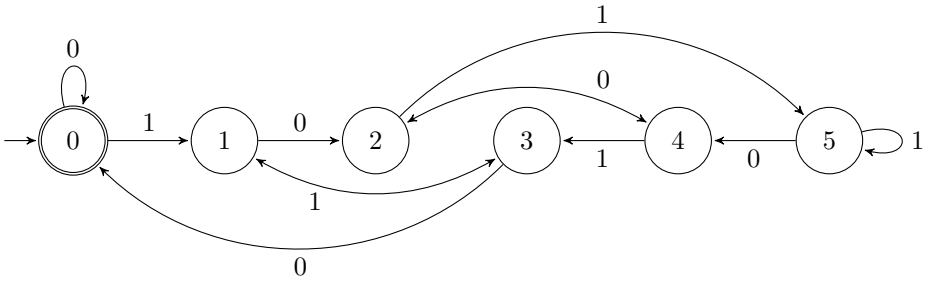
Und die Verschmelzung führt vom Ausgangsautomaten



auf den Äquivalenzklassenautomaten



Beispiel 7.35. Der DFA A mit Zustandsdiagramm



akzeptiert die Binärdarstellungen aller durch 6 teilbaren Zahlen, also die Sprache

$$L(A) \quad = \quad \Big\{ w \in \{0,1\}^* \ : \ \sum_{i=1}^{|w|} w_i \, 2^{|w|-i} \equiv 0 \ (\text{mod } 6) \Big\}. \tag{7.3}$$

Wir möchten wieder den Äquivalenzklassenautomaten A' bestimmen und stellen zuerst die Tabelle aller Paarmengen auf:

1	M_0				
2	M_0	M_2			
3	M_0	M_1	M_1		
4	M_0		M_2	M_1	
5	M_0	M_2		M_1	M_2
	0	1	2	3	4

Wir können somit die Zustände 1 und 4, aber auch die Zustände 2 und 5 verschmelzen.

7.3.5. Die Nerode-Relation

Was haben wir bisher gelernt? Der ursprüngliche Automat $A = (Q, \Sigma, \delta, q_0, F)$ und sein Äquivalenzklassenautomat A' sind äquivalent. Wir können A' effizient berechnen. Aber hat A' unter allen mit A äquivalenten DFAs die kleinste Zustandszahl?

Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA. Wenn zwei Worte $u, v \in \Sigma^*$ auf denselben Zustand von A führen (d.h. wenn $\widehat{\delta}(q_0, u) = \widehat{\delta}(q_0, v)$ gilt), dann folgt

$$\text{f.a. } w \in \Sigma^* : (uw \in L(A) \iff vw \in L(A)). \tag{7.4}$$

Wir interpretieren Eigenschaft (7.4) als Definition der Äquivalenz der Worte u und v und führen deshalb die Nerode-Relation ein.

Definition 7.36. Sei Σ eine endliche Menge und L eine Sprache über Σ (d.h. es gilt $L \subseteq \Sigma^*$).

(a) Die **Nerode-Relation** $\equiv_L$ für L ist eine 2-stellige Relation über Σ^*. Für alle Worte $u, v \in \Sigma^*$ definiere Nerode-Relation

$$u \equiv_L v \ :\iff \text{f.a. } w \in \Sigma^* \text{ gilt: } \big(uw \in L \iff vw \in L\big).$$

(b) Die Äquivalenzklasse eines Wortes $u \in \Sigma^*$ wird mit $[u]_L$ bezeichnet. Es ist also $[u]_L$

$$[u]_L = \{\, w \,:\, u \equiv_L w \,\}.$$

(c) Wir sagen, dass das Wort $v \in \Sigma^*$ die Worte $u, w \in \Sigma^*$ trennt, bzw. dass v ein **Zeuge** für die Inäquivalenz von u und w ist, wenn Zeuge

$$\Big(uv \in L \wedge wv \notin L\Big) \vee \Big(uv \notin L \wedge wv \in L\Big)$$

(d) **Index**(L) ist die Anzahl der Äquivalenzklassen von $\equiv_L$. Index(L)

Die Nerode-Relation erlaubt, dass wir die Perspektive der DFAs aufgeben können und uns nur noch auf die Struktur der Sprache L konzentrieren dürfen.

Beachte die Unterschiede zur Verschmelzungsrelation $\equiv_A$. Während $\equiv_A$ auf dem DFA A basiert, ist die Sprache L maßgeblich für die Nerode-Relation $\equiv_L$. Kurz gefasst: $\equiv_A$ spricht über die Äquivalenz von Zuständen, $\equiv_L$ über die Äquivalenz von Worten.

Satz 7.37. Sei L eine Sprache. Dann ist die Nerode-Relation für L eine Äquivalenzrelation.

Beweis: Die Nerode-Relation $\equiv_L$ ist

1. reflexiv, denn f.a. $z \in \Sigma^*$ gilt $\big(uz \in L \iff uz \in L\big)$ und deshalb gilt $u \equiv_L u$ für alle Worte $u \in \Sigma^*$.

2. symmetrisch, denn aus $u \equiv_L v$ folgt $\big(uz \in L \iff vz \in L\big)$ f.a. $z \in \Sigma^*$ und damit auch $\big(vz \in L \iff uz \in L\big)$ f.a. $z \in \Sigma^*$. Die letzte Aussage ist aber äquivalent zu $v \equiv_L u$.

3. transitiv, denn aus $u \equiv_L v$ und $v \equiv_L w$ folgt f.a. $z \in \Sigma^*$ $\big(uz \in L \iff vz \in L\big) \wedge \big(vz \in L \iff wz \in L\big)$. Also folgt $\big(uz \in L \iff wz \in L\big)$ f.a. $z \in \Sigma^*$ und damit $u \equiv_L w$.

□

Beispiel 7.38. (**Die Parität**). Wir betrachten die Sprache

$$P = \{w \in \{0,1\}^* \,:\, w \text{ hat eine gerade Anzahl von Einsen }\}.$$

Es ist $\varepsilon \not\equiv_P 1$ mit dem leeren Wort als Zeugen. Sei u ein beliebiges Wort mit gerade vielen Einsen und v ein beliebiges Wort mit ungerade vielen Einsen. Dann gilt

$$\Big(\varepsilon w \in P \iff uw \in P\Big) \text{ und } \Big(1w \in P \iff vw \in P\Big)$$

für alle $w \in \Sigma^*$. Mit anderen Worte, die Nerode-Relation für P hat genau zwei Äquivalenzklassen –es ist $\text{Index}(P) = 2$– mit den jeweiligen Vertretern ε und 1. Zusammengefasst: Es gilt

$$\begin{aligned} [\epsilon]_P &= \{\, w \in \{0,1\}^* \,:\, w \text{ hat gerade viele Einsen }\} \text{ und} \\ [1]_P &= \{\, w \in \{0,1\}^* \,:\, w \text{ hat ungerade viele Einsen }\}. \end{aligned}$$

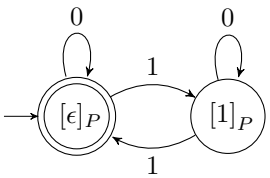
Wir bauen aus diesen beiden Äquivalenzklassen den sogenannten Nerode-Automaten N_P. Hier ist unser Rezept: Für jedes Wort $w \in \{0,1\}^*$ soll N_L nach Lesen von w den Zustand $[w]_P$ erreichen.

- Damit ist schon klar, dass $q_0 = [\epsilon]_P$ der Anfangszustand sein muss und q_0 ist ein akzeptierender Zustand, denn $\epsilon \in P$.

- Der Nachfolgezustand unter 0 muss die Äquivalenzklasse von $\epsilon 0 = 0$ sein und das ist die Klasse $[\epsilon]_P$, denn $0 \equiv_L \epsilon$. Der Nachfolgezustand unter 1 ist dann die Klasse $[1]_P$.

- Der Zustand $[1]_P$ ist nicht akzeptierend, denn $1 \notin P$. Lesen wir im Zustand $[1]_P$ eine 0, dann müssen wir die Äquivalenzklasse von 10 erreichen und das bleibt die Klasse $[1]_P$. Lesen wir eine 1, ist die Klasse von 11, also die Klasse $[\epsilon]_P$ unser Ziel.

Hier ist der Nerode-Automat N_L:



□Ende von Beispiel 7.38

Wir sind im Beispiel 7.38 sehr unvorsichtig gewesen, denn wir haben nur die Vertreter ϵ und 1 der Nerode-Klassen betrachtet. Wir müssen aber beispielsweise für jeden Buchstaben $a \in \{0,1\}$ garantieren, dass aus $u \equiv_P 1$ auch $ua \equiv_P 1a$ folgt, denn nur dann erreicht N_P nach Lesen von ua den Zustand $[ua]$. Kein Problem, es genügt wenn wir nur mit den Vertretern arbeiten:

Satz 7.39. Sei L eine beliebige Sprache über dem Alphabet Σ und $a \in \Sigma$ ein beliebiger Buchstabe. Dann gilt für beliebige Worte $u, w \in \Sigma^*$

$$u \equiv_L w \implies ua \equiv_L wa.$$

Beweis: Angenommen, es ist $u \equiv_L w$, aber $ua \not\equiv_L wa$. Dann gibt es einen Zeugen $v \in \Sigma^*$, so dass zum Beispiel $uav \in L$, aber $wav \notin L$. Aber dann folgt $u \not\equiv_L w$ mit dem Zeugen av, im Widerspruch zur Annahme $\lightning$ □

Beispiel 7.40. In Beispiel 7.34 haben wir einen DFA für die „Suffixsprache“ $S = \{a,b\}^* \cdot ab$ minimiert. Hier fragen wir uns wie die Nerodeklassen $[w]_S$ aussehen und ob wir wieder den Nerode-Automaten N_S bauen können. Fangen wir gleich mit dem Nerode-Automaten an.

- Der Startzustand ist die Klasse des leeren Worts, also $q_0 = [\epsilon]_S$. Hier gibt es nichts zum Nachdenken.

  Wenn $w \equiv_S \epsilon$, dann kann w nicht mit einem a enden, denn sonst trennt der Zeuge $v = b$ das Wort w von ϵ. Auch kann w nicht mit ab enden, denn dann trennt der Zeuge ε. Also folgt

  $$[\epsilon]_S = \{\, ubb \,:\, u \in \{a,b\}^* \,\} \cup \{\epsilon,\, b\}. \tag{7.5}$$

  - Lesen wir den Buchstaben a haben wir einen Übergang zur Äquivalenzklasse von $\epsilon a = a$. Wir haben eine neue Äquivalenzklasse gefunden, denn $\epsilon \not\equiv_S a$ mit dem Zeugen $v = b$.
  - Lesen wir den Buchstaben b erreichen wir die Äquivalenzklasse von b. Aber $\epsilon \equiv_S b$ gilt (vgl. (7.5) und wir kehren unter b zum Startzustand zurück.

- Wann gilt $w \equiv_S a$ für ein Wort $w \in \{a,b\}^*$? Wenn w mit einem b endet, dann folgt $w \not\equiv_S a$ mit dem Zeugen b. Man überzeuge sich, dass $ua \equiv_S a$ für alle Worte u gilt und wir erhalten

  $$[a]_S = \{\, ua \,:\, u \in \{a,b\}^* \,\}. \tag{7.6}$$

  - Es ist $aa \equiv_S a$ und N_S verbleibt im Zustand $[a]_S$, wenn a gelesen wird.
  - Es gilt $ab \not\equiv_S a$ und $ab \not\equiv_S \epsilon$ jeweils mit dem Zeugen ϵ: Wird der Buchstabe b gelesen, dann wechselt N_S vom Zustand $[a]_S$ in den neuen Zustand $[ab]_S$.

- Wann gilt $w \equiv_S ab$? Es muss $w \in S$ gelten, denn ansonsten trennt das leere Wort w von ab. Für jedes Wort $w \in S$ gilt

  $$abv \in S \iff ((v \in S) \vee (v = \epsilon)) \iff wv \in S$$

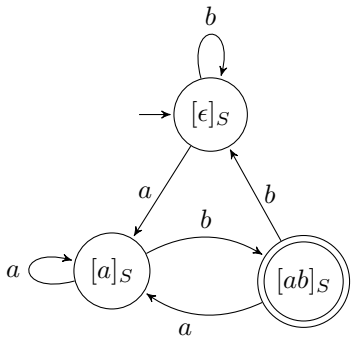
  und als Konsequenz folgt

  $$[ab]_S = S.$$

  - Wenn N_S den Buchstaben a im Zustand $[ab]_S$ liest, dann gilt $aba \equiv_S a$ und N_S wechselt in den Zustand $[a]_S$.

– Liest N_S hingegen den Buchstaben b, dann ist $abb \equiv_S b$ und N_S wechselt in den Anfangszustand $[\epsilon]_S$.

Wir haben keine neue Äquivalenzklasse gefunden, die Konstruktion des Nerode-Automaten N_S ist abgeschlossen und damit haben wir auch alle Nerodeklassen gefunden. Insbesondere gilt $\text{Index}(S) = 3$.



□Ende von Beispiel 7.40

Kann man den Nerode-Automaten für jede Sprache L einführen? Na klar:

Nerode-Automat

Definition 7.41. (**Der Nerode-Automat**). Sei L eine Sprache über dem Alphabet Σ. Dann hat der **Nerode-Automat** $N_L = (Q, \Sigma, \delta, q_0, F)$ die folgenden Komponenten:

- Die Zustandsmenge Q besteht aus allen Äquivalenzklassen der Nerode-Relation $\equiv_L$.
- Es ist $\delta([w]_L, a) := [wa]_L$ für jedes Wort $w \in \Sigma^*$ und jeden Buchstaben $a \in \Sigma$.
- $q_0 := [\epsilon]_L$ ist der Anfangszustand.
- $F := \{\, [w]_L \,:\, w \in L \,\}$ ist die Menge der akzeptierenden Zustände.

Ist der Nerode-Automat denn auch „wohl-definiert“? Schauen wir uns das Programm δ an: Wenn $u \equiv_L w$ gilt, darf $ua \not\equiv_L va$ nicht passieren, denn sonst ist $\delta([u]_L, a) = [ua]_L \neq [va]_L = \delta[v]_L, a) = \delta([u]_L, a)$ ↯ Aber keine Sorge, Satz 7.39 lässt nichts anbrennen.

Satz 7.42. Sei L eine Sprache über dem Alphabet Σ.

(a) Für jedes Wort $w \in \Sigma^*$ erreicht der Nerode-Automat N_L den Zustand $[w]_L$.

(b) N_L akzeptiert die Sprache L mit $\text{Index}(L)$ vielen Zuständen.

Beweis: Teil (a) zeigt man durch vollständige Induktion über die Länge von w.
Induktionsanfang: Für $w = \epsilon$ ist $\widehat{\delta}(q_0, w) = q_0 := [\epsilon]_L$. ✓
Induktionsschritt: Für $w = ua$ ist

$$\widehat{\delta}(q_0, w) = \widehat{\delta}(q_0, ua) = (\delta(\widehat{\delta}(q_0, u), a) = (\delta([u], a) := [ua]_L. \checkmark$$

Für Teil (b) betrachten wir die Menge F. Gibt es ein Wort $u \in L$ und ein Wort $w \notin L$ mit $u \equiv_L w$? Kann nicht passieren, denn dann trennt der Zeuge $v = \epsilon$ die Worte u und w.

Zusammengefasst, N_L erreicht den Zustand $[u]_L$ für das Wort u und $[u]_L \in F$ gilt genau dann wenn $u \in L$. □

Der Nerode-Automat N_L akzeptiert die Sprache mit Index(L) vielen Zuständen. Die Intuition sollte sein, dass N_L mit Zuständen äußerst geizig umgeht. Kann es einen DFA $A = (\Sigma, Q, \delta, q_0, F)$ für L geben, der weniger Zustände besitzt?

Wenn $k = \text{Index}(L)$, dann besitzen die k verschiedenen Äquivalenzklassen der Nerode-Relation k Vertreter $u_1, \ldots, u_k$, und diese Vertreter sind natürlich paarweise inäquivalent.

Angenommen, zwei dieser Vertreter, z.B. u_i und u_j (mit $i \neq j$), führen auf denselben Zustand in Q. Dann gilt

$$\widehat{\delta}(q_0, u_i) = \widehat{\delta}(q_0, u_j)$$

und es ist $\widehat{\delta}(q_0, u_i w) = \widehat{\delta}(q_0, u_j w)$ für alle Worte $w \in \Sigma^*$. Aber A akzeptiert die Sprache L und es ist $u_i w \in L \iff u_j w \in L$: Wir erhalten $u_i \equiv_L u_j$ im Widerspruch zur Inäquivalenz von u_i und u_j.

Jetzt können wir die Ernte einbringen und es gibt reichlich einzubringen!

Satz 7.43. (**Der Satz von Myhill-Nerode, I**). Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA.

(a) Index$(L(A))$ stimmt überein mit der minimalen Zustandszahl eines DFA für die Sprache $L(A)$.

(b) Der Äquivalenzklassenautomat A' ist minimal.

Beweis: Für Teil (a) beachten wir mit Satz 7.42, dass der Nerode-Automat nur Index(L) Zustände besitzt. Gerade aber haben wir gesehen, dass jeder DFA für L mindestens Index(L) Zustände benötigt: Der Nerode-Automat ist der Champion!

Zum Beweis von Teil (b) nehmen wir an, dass Worte $u, v \in \Sigma^*$ im Äquivalenzklassenautomaten A' zu verschiedenen Zuständen führen. Dann folgt $\widehat{\delta'}(q_0', u) \neq \widehat{\delta'}(q_0', v)$. Was passiert im Ausgangsautomaten A mit den Worten u und v?

Für A sei $p = \widehat{\delta}(q_0, u)$ und $q = \widehat{\delta}(q_0, v)$. Dann folgt $[p]_A = [\widehat{\delta}(q_0, u)]_A = \widehat{\delta'}(q_0', u) \neq \widehat{\delta'}(q_0', v) = [\widehat{\delta}(q_0, v)]_A = [q]_A$.

Es ist also $p \not\equiv_A q$ und es gibt einen Zeugen $w \in \Sigma^*$ für die Inäquivalenz. Also:

$$\Big(\widehat{\delta}(p, w) \in F \text{ und } \widehat{\delta}(q, w) \notin F\Big) \text{ oder } \Big(\widehat{\delta}(p, w) \notin F \text{ und } \widehat{\delta}(q, w) \in F\Big),$$

bzw.

$$\Big(\widehat{\delta}(q_0, uw) \in F \text{ und } \widehat{\delta}(q_0, vw) \notin F\Big) \text{ oder } \Big(\widehat{\delta}(q_0, uw) \notin F \text{ und } \widehat{\delta}(q_0, vw) \in F\Big)$$

bzw.

$$\Big(uw \in L(A) \text{ und } vw \notin L(A)\Big) \text{ oder } \Big(uw \notin L(A) \text{ und } vw \in L(A)\Big).$$

Wenn u und v in A' zu verschiedenen Zuständen führen, dann folgt somit $u \not\equiv_{L(A)} v$: Die Anzahl der Zustände von A' ist höchstens so groß wie der Index von $L(A)$. Und nach Teil (a) ist A' minimal. □

Mit anderen Worten: Wenn Index$(L(A)) = |Q|$, dann ist A ein minimaler Automat!

Beispiel 7.44. Sei Σ ein beliebiges Alphabet und u ein Wort über Σ. Wir möchten das Teilwort-Problem studieren und betrachten deshalb die Sprache

$$L_u = \{ w \in \Sigma^* \,:\, u \text{ ist ein Teilwort von } w \}.$$

Sei $u = u_1 \cdots u_k$ mit den Buchstaben $u_1, \ldots, u_k \in \Sigma$. Wir behaupten, dass der Index von L_u mindestens $k+1$ ist. Wir betrachten dazu die $k+1$ Präfixe $p_0, \ldots, p_k$ mit

$$p_0 = \varepsilon, p_1 = u_1, p_2 = u_1u_2, \ldots, p_i = u_1 \cdots u_i, \ldots, p_k = u_1 \cdots u_k. \tag{7.7}$$

Zwei verschiedene Präfixe $p_i = u_1 \cdots u_i$ und $p_j u_1 \cdots u_j$ (mit $i < j$) lassen sich mit dem Zeugen $u_{j+1} \cdots u_k$ trennen, denn $u_1 \cdots u_j \cdot u_{j+1} \cdots u_k = u \in L_u$, aber $u_1 \cdots u_i u_{j+1} \cdots u_k$ kann als ein Wort mit weniger als k Buchstaben nicht in L_u liegen.

Editoren müssen das Teilwort-Problem blitzschnell lösen. Lässt sich ein kleiner DFA konstruieren, der genau alle Worte mit Teilwort u akzeptiert? Bei einer positiven Antwort können wir das Teilwort-Problem tatsächlich blitzschnell auf kleinem Speicherplatz lösen!

Tatsächlich ist die Antwort positiv, denn $\text{Index}(L) = k+1$. Dazu genügt es zu zeigen, dass jedes Wort $w \in \Sigma^*$ zu einem der Präfixe p_i äquivalent ist.

- Wenn $w \in L_u$, dann ist $w \equiv_{L_u} u_1 \cdots u_k$.
- Ansonsten ist u kein Teilwort von w und w ist mit dem Suffix w' seiner letzten k Buchstaben äquivalent. Zeige, dass w' mit einem der $k+1$ Präfixe p_i äquivalent ist.

Beispiel 7.45. In Beispiel 7.35 haben wir einen DFA mit vier Zuständen konstruiert, der alle Binärdarstellungen durch sechs teilbarer Zahlen akzeptiert. Sei $L = L(A)$ die in (7.3) definierte Sprache. Wir zeigen, dass der Index von L mindestens vier ist.

Betrachte dazu die Worte $00, 01, 10, 11$. 00 lässt sich durch das leere Wort von $01, 10, 11$ trennen. Des weiteren können wir 11 von 01 und 10 trennen, wenn wir 0 als Zeugen verwenden. Schließlich folgt $01 \not\equiv_L 10$ mit dem Zeugen 10, denn 0110 stellt die Zahl sechs dar, während 1010 die nicht durch sechs teilbare Zahl zehn darstellt.

Der DFA nach Verschmelzen der Zustände 1 und 4, bzw. 2 und 5 ist somit minimal.

7.4. Reguläre Sprachen

reguläre Sprache

Definition 7.46. Sei Σ eine endliche Menge. Eine Sprache $L \subseteq \Sigma^*$ heißt eine **reguläre Sprache**, wenn es einen DFA A gibt mit

$$L = L(A).$$

Klar: Um zu zeigen, dass eine Sprache $L \subseteq \Sigma^*$ regulär ist, reicht es, einen DFA A mit $L(A) = L$ zu finden.

Frage: Wie kann man nachweisen, dass eine bestimmte Sprache $L \subseteq \Sigma^*$ **nicht** regulär ist?

Versuchen wir den Nerode-Automaten N_L für eine vermutlich nicht-reguläre Sprache L zu konstruieren. Wenn der minimale Automat N_L unendlich viele Zustände hat, dann muss doch jeder Automat für L unendlich viele Zustände besitzen und L kann nicht regulär sein.

Beispiel 7.47. (Die Sprache $L = \{ a^n b^n \,:\, n \in \mathbb{N} \}$).

Wir zeigen später, dass die Sprache L von keinem DFA akzeptiert werden kann. Wir möchten die Äquivalenzklassen der Nerode-Relation $\equiv_L$ für jedes Wort $u \in \{a,b\}^*$ beschreiben. Dazu führen wir eine Fallunterscheidung durch.

Fall 1: Es gibt kein Wort $v \in \{a,b\}^*$ mit $u \cdot v \in L$.

Dann gilt $u \equiv_L b$. Warum? Für alle Worte $v \in \{a,b\}^*$ ist $uv \notin L$ genauso wie $bv \notin L$ gilt, denn auch b ist nicht mehr „zu retten".

Fall 2: Es gibt ein Wort $v \in \{a,b\}^*$ mit $u \cdot v \in L$.

Dann gilt $u = a^n b^m$ für $n \geqslant m$. Wir behandeln zwei Unterfälle

Fall 2.1 : Es ist $u = a^n$.

Wir behaupten, dass $a^{n_1} \not\equiv_L a^{n_2}$ falls $n_1 \neq n_2$. Warum? Für den Zeugen $v = b^{n_1}$ ist $a^{n_1} v \in L$, aber $a^{n_2} v \notin L$.

Zwischenstand: Die Worte a^n sind also paarweise inäquivalent und natürlich ist auch $a^n \not\equiv_L b$ mit dem Zeugen $v = b^n$.

Fall 2.2 : Es ist $u = a^n b^m$ mit $1 \leqslant m \leqslant n$.

Für alle Worte $v \in \{a,b\}^*$ ist

$$\begin{aligned} uv \in L &\iff a^n b^m v \in L \iff v = b^{n-m}, \\ a^{n-m+1} bv \in L &\iff v = b^{n-m}. \end{aligned}$$

Also gilt $a^n b^m \equiv_L a^{n-m+1} b$. Wir erhalten also die Vertreter $a^k b$ und beobachten zuerst, dass $a^{n_1} \not\equiv_L a^{n_2} b$ für alle n_1, n_2 gilt. Warum? Benutze den Zeugen $v = a b^{n_1+1}$.

Die Vertreter a^{n_1} aus Fall 2.1 und $a^{n_2} b$ aus Fall 2.2 sind also paarweise inäquivalent. Jetzt haben wir ein vollständiges Bild erhalten, denn es ist $a_{n_1} b \not\equiv_L a^{n_2} b$ falls $n_1 \neq n_2$ mit dem Zeugen $v = b^{n_1 - 1}$.

Fazit: Wir haben die folgenden Äquivalenzklassen erhalten.

(a) Die große Äquivalenzklasse $[b]_L$ mit

$$[b]_L = \{ w \in \{a,b\}^* \,:\, \text{für alle Worte } v \text{ ist } wv \notin L \}.$$

(b) Die kleinen Äquivalenzklassen $[a^n]_L$ mit

$$[a^n]_L = \{ a^n \}$$

(c) und die größeren Klassen $[a^n b]_L$ mit

$$[a^n b]_L = \{ a^{n+k} b^{k+1} \,:\, k \in \mathbb{N} \}.$$

Diesmal gibt es somit unendlich viele Äquivalenzklassen. Gleich werden wir sehen, dass L keine reguläre Sprache ist. □Ende von Beispiel 7.47

7.4.1. Der Satz von Myhill und Nerode

Mit Hilfe der Nerode-Relation können wir genau sagen, wann eine Sprache regulär ist.

Satz 7.48. (**Satz von Myhill-Nerode, II**): Sei $L \subseteq \Sigma^*$ eine Sprache über dem Alphabet Σ Dann gilt

$$L \text{ ist regulär} \iff \text{Index}(L) \text{ ist endlich.}$$

Beweis: $\Longrightarrow$ Die Sprache $L \subseteq \Sigma^*$ sei regulär. Dann gibt es einen DFA A mit $L = L(A)$. A hat mindestens Index(L) viele Zustände und Index(L) ist endlich.

$\Longleftarrow$ Index(L) sei endlich. Dann wird L von seinem Nerode-Automaten N_L akzeptiert. Aber N_L ist ein DFA und L ist eine reguläre Sprache. □

Satz 7.49. $L = \{a^n b^n \ : \ n \in \mathbb{N}\}$ ist nicht regulär.

Beweis: Wir müssen unendlich viele Worte aus Σ^* bestimmen, die paarweise inäquivalent bzgl. der Nerode-Relation sind. Wir wählen die Worte aus $\{\, a^i \,:\, i \in \mathbb{N} \,\}$.

Für $i \neq j$ gilt $a^i \not\equiv_L a^j$ mit dem Zeugen b^i, denn $a^i b^i \in L$, aber $a^j b^i \notin L$. (Alternativ hätten wir natürlich auch den Zeugen b^j benutzen können. □

Satz 7.50. $L = \{ww \ : \ w \in \{a,b\}^*\}$ ist nicht regulär: DFAs können sich nur endlich viele Dinge merken.

Beweis: Wir zeigen, dass alle Worte in $\{\, a^i b \,:\, i \in \mathbb{N} \,\}$ paarweise inäquivalent bzgl. der Nerode-Relation sind.

Für $i \neq j$ trenne $a^i b$ von $a^j b$ mit Hilfe des Zeugens $v = a^i b$: Es ist $a^i bv = a^i b a^i b \in L$, aber $a^j bv = a^j b a^i b \notin L$. Wir erhalten Index($L$) $= \infty$ und L ist nicht regulär. □

Weitere nicht-reguläre Sprachen L erhalten wir, wenn wir zeigen, dass Index(L) unendlich groß ist.

Satz 7.51. Keine der folgenden Sprachen ist regulär.

(a) $L_1 = \{\, a^n b^m \,:\, n, m \in \mathbb{N},\ n \leqslant m \,\}$,

(b) $L_2 = \{\, a^n b^m c^{n+m} \,:\, n, m \in \mathbb{N} \,\}$,

(c) $L_3 = \{\, a^{n^2} \,:\, n \in \mathbb{N} \,\}$,

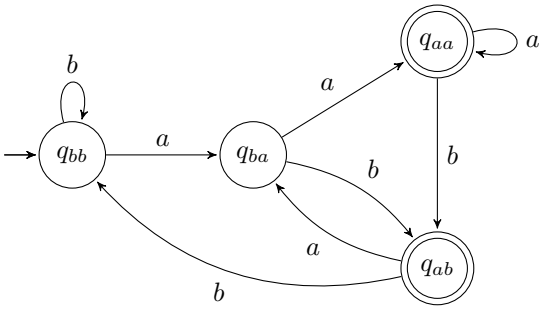
(d) $L_4 = \{\, w \in \{a,b\}^* \,:\, w \text{ ist ein Palindrom} \,\}$.

Beweis: Übung. □

7.4.2. Das Pumping-Lemma für reguläre Sprachen

Ein weiteres nützliches Werkzeug zum Nachweis der Nicht-Regularität ist der folgende Satz 7.53, der unter dem Namen **Pumping-Lemma** bekannt ist. Bevor wir den Satz präzise angeben, betrachten wir zunächst ein Beispiel:

Beispiel 7.52. Sei A_3 der endliche Automat aus Beispiel 7.10((c)), d.h.



A_3 akzeptiert beispielsweise das Eingabewort

$$x \;=\; babaa,$$

indem er nacheinander die Zustände

$$q_{bb} \xrightarrow{b} q_{bb} \xrightarrow{a} q_{ba} \xrightarrow{b} q_{ab} \xrightarrow{a} q_{ba} \xrightarrow{a} q_{aa}$$

besucht. Dieser Weg durch die graphische Darstellung von A_3 enthält einen Kreis

$$q_{ba} \xrightarrow{b} q_{ab} \xrightarrow{a} q_{ba},$$

der beliebig oft durchlaufen werden kann, so dass man (egal ob der Kreis 0-mal, 1-mal, 2-mal, 3-mal, ... durchlaufen wird) jedesmal ein Eingabewort erhält, das von A_3 akzeptiert wird, nämlich für jede Zahl $i \geqslant 0$ das Eingabewort $ba(ba)^i a$.

Der folgende Satz 7.53 beruht auf demselben Prinzip sowie der Tatsache, dass in jedem Graph auf z Knoten gilt: Jeder Weg der Länge $\geqslant z$ enthält einen Kreis (d.h. mindestens ein Knoten wird auf dem Weg mehr als 1-mal besucht).

Satz 7.53 (Pumping-Lemma). Sei Σ ein endliches Alphabet. Pumping-Lemma
Für jede **reguläre** Sprache $L \subseteq \Sigma^*$ gibt es eine Zahl $n \in \mathbb{N}_{>0}$ (die so genannte **Pumpingkonstante**), so dass für jedes Wort $z \in L$ der Länge $|z| \geqslant n$ gilt: Es gibt eine Zerlegung von z in Worte $u, v, w \in \Sigma^*$, so dass die folgenden Bedingungen erfüllt sind: Pumpingkonstante

(1) $z = uvw$

(2) $|uv| \leqslant n$

(3) $|v| \geqslant 1$

(4) für jedes $i \in \mathbb{N}$ gilt: $uv^i w \in L$.
(d.h.: $uw \in L$, $uvw \in L$, $uvvw \in L$, $uvvvw \in L$, ...)

Beweis: Da L regulär ist, gibt es einen DFA $A = (\Sigma, Q, \delta, q_0, F)$ mit $L(A) = L$. Sei $n := |Q|$ die Anzahl der Zustände von A.

Sei nun $z \in \Sigma^*$ ein beliebiges Wort der Länge $|z| \geqslant n$, das in L liegt, d.h. das von A akzeptiert wird. Sei $q_0, q_1, \ldots, q_{|z|}$ die Folge von Zuständen, die A beim Verarbeiten von z durchläuft. Da $|z| \geqslant n = |Q|$ ist, können die Zustände $q_0, q_1, \ldots, q_{n_L}$ nicht alle verschieden sein. Daher gibt es ein $k \geqslant 0$ und ein $\ell \geqslant 1$, so dass $q_k = q_{k+\ell}$ und $k + \ell \leqslant n$. Wir wählen folgende Zerlegung von z in Worte $u, v, w \in \Sigma^*$:

- u besteht aus den ersten k Buchstaben von z.
- v besteht aus den nächsten ℓ Buchstaben von z.
- w besteht aus den restlichen Buchstaben von z.

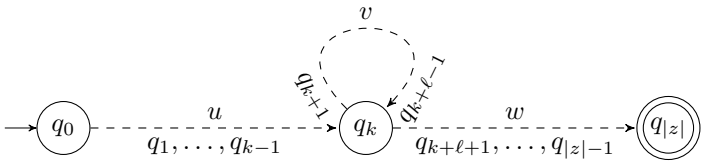
Offensichtlich gilt:

(1) $z = uvw$,

(2) $|uv| = k + \ell \leqslant n$,

(3) $|v| = \ell \geqslant 1$.

Skizze:



Daher gilt für jedes $i \geqslant 0$: A akzeptiert das Eingabewort uv^iw, d.h. $uv^iw \in L$. □

Beobachtung 7.54. Unter Verwendung des Pumping-Lemmas kann man nachweisen, dass gewisse Sprachen L **nicht** regulär sind. Dies lässt sich gut durch ein Zwei-Personen-Spiel illustrieren, bei dem „wir“ gegen einen „Gegner“ spielen. Das Spiel zum Nachweis, dass L nicht regulär ist, wird nach folgenden Regeln gespielt:

(1) Der Gegner wählt eine Pumpingkonstante $n_L \geqslant 1$.

(2) Wir wählen ein Wort $z \in L$ der Länge $|z| \geqslant n_L$.

(3) Der Gegner zerlegt z in Worte u, v, w, so dass gilt: $|uv| \leqslant n_L$ und $|v| \geqslant 1$.

(4) Wir pumpen auf oder ab, d.h. wir wählen eine Zahl $i \geqslant 2$ oder $i = 0$.

(5) Wir haben gewonnen, wenn $uv^iw \notin L$ ist; ansonsten hat der Gegner gewonnen.

Aus dem Pumping-Lemma folgt: Wenn wir eine Gewinnstrategie in diesem Spiel haben, dann ist die Sprache L nicht regulär.

Beispiel 7.55. Sei $\Sigma := \{a, b\}$. Die Sprache $L := \{a^nb^n : n \in \mathbb{N}\}$ ist **nicht** regulär.

Vergleiche die Argumentation mit dem Beweis von Satz 7.49. Beachte auch, dass gemäß Lemma 7.10((a)) die Sprache $L_1 := \{a^nb^m : n \in \mathbb{N}, m \in \mathbb{N}\}$ regulär ist.

Beweisidee: Wir versuchen, eine Gewinnstrategie im Spiel zu finden.

(1) Der Gegner wählt eine Pumpingkonstante $n \geqslant 1$.

(2) Wir wählen das Wort $z := a^nb^n$ (beachte: $z \in L$ und $|z| \geqslant n$, d.h. wir haben regelkonform gespielt).

(3) Der Gegner zerlegt z in Worte u, v, w, so dass gilt: $|uv| \leqslant n$ und $|v| \geqslant 1$.

(4) Wir beobachten, dass uv kein b enthalten kann, da $uvw = z = a^{n_L} b^{n_L}$ und $|uv| \leqslant n_L$ ist. Außerdem muss es eine Zahl $\ell \geqslant 1$ geben, so dass $v = a^\ell$ ist. Wir entscheiden uns daher, die Zahl „$i = 0$“ zu wählen (also „abzupumpen“).

(5) Es gilt nun: $uv^i w = uw$. Dieses Wort sieht wie z aus, nur dass ein Teilstück der Form a^ℓ gelöscht wurde. D.h. $uw = a^{n_L - \ell} b^{n_L}$. Wegen $\ell \geqslant 1$ liegt dieses Wort nicht in der Menge $L = \{a^n b^n : n \in \mathbb{N}\}$.

Die hier gefundene Beweisidee schreiben wir nun noch als formalen Beweis auf.

Beweis: Durch Widerspruch.
Angenommen, L ist regulär. Dann sei $n \in \mathbb{N}_{>0}$ die gemäß dem Pumping-Lemma (Satz 7.53) gewählte Pumpingkonstante. Betrachte das Wort $z := a^n b^n$. Klar: $z \in L$ und $|z| \geqslant n$. Gemäß dem Pumping-Lemma gibt es eine Zerlegung von z in Worte $u, v, w \in \{a, b\}^*$, so dass

(1) $z = uvw$

(2) $|uv| \leqslant n$

(3) $|v| \geqslant 1$

(4) f.a. $i \in \mathbb{N}$ gilt: $uv^i w \in L$.

Wegen $z = a^n b^n = uvw$ und $|uv| \leqslant n$ und $|v| \geqslant 1$ gibt es eine Zahl $\ell \in \mathbb{N}_{>0}$, so dass $v = a^\ell$. Wegen ((4)) gilt insbesondere für $i = 0$, dass $uw \in L$. Wegen $z = uvw = a^n b^n$ und $v = a^\ell$ mit $\ell \geqslant 1$ gilt

$$uw \;=\; a^{n-\ell} b^n \;\overset{\ell \geqslant 1}{\notin}\; \{a^i b^i : i \in \mathbb{N}\} \;=\; L.$$

↯ □

Beispiel 7.56. Sei $\Sigma := \{a\}$. Die Sprache $L := \{w \in \{a\}^* : |w| \text{ ist eine Primzahl}\}$ ist **nicht** regulär.

Beweis: Durch Widerspruch.
Angenommen, L ist regulär. Dann sei $n \in \mathbb{N}_{>0}$ die gemäß dem Pumping-Lemma (Satz 7.53) gewählte Pumpingkonstante. Da es unendlich viele Primzahlen gibt, gibt es auch eine Primzahl p mit $p \geqslant n + 2$. Sei p solch eine Primzahl. Betrachte nun das Wort $z := a^p$. Klar: $z \in L$ und $|z| \geqslant n_L$. Gemäß dem Pumping-Lemma gibt es also eine Zerlegung von z in Worte $u, v, w \in \{a\}^*$, so dass

(1) $z = uvw$,

(2) $|uv| \leqslant n$,

(3) $|v| \geqslant 1$,

(4) f.a. $i \in \mathbb{N}$ gilt: $uv^i w \in L$.

Wegen ((4)) gilt insbesondere für $i := |uw|$, dass $uv^i w \in L$. Es gilt dann

$$\left|uv^i w\right| \;=\; |uw| + i \cdot |v| \;=\; |uw| + |uw| \cdot |v| \;=\; |uw| \cdot (1 + |v|).$$

Wegen

$$|uw| \;\geqslant\; |w| \;=\; |z| - |uv| \;\overset{|uv| \leqslant n}{\geqslant}\; p - n \;\overset{p \geqslant n+2}{\geqslant}\; 2 \qquad \text{und} \qquad 1 + |v| \;\overset{|v| \geqslant 1}{\geqslant}\; 2$$

ist $\left|uv^i w\right|$ daher keine Primzahl, d.h. $uv^i w \notin L$. ↯ □

Bemerkung 7.57. Wir haben zwei Methoden kennengelernt, um zu zeigen, dass Sprachen nicht-regulär sind: Die „Index-Methode“ und das Pumping-Lemma. Welche Methode vorzuziehen ist, ist letztlich eine Frage des persönlichen Geschmacks. Aber:

(a) Es gibt nicht-reguläre Sprachen auf die das Pumping-Lemma nicht anwendbar ist.

(b) Die Index-Methode ist auf alle nicht-regulären Sprachen anwendbar, aber in einigen Fällen(wie etwa für die „Primzahlsprache“) kann ihre Handhabung schwerfälliger als die des Pumping-Lemmas sein.

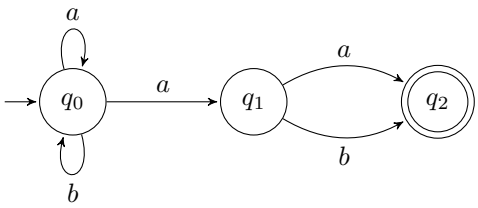
7.5. Nichtdeterministische endliche Automaten

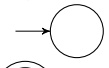
Für manche Modellierungsaufgaben ist die Forderung, dass es für jeden Zustand q und jedes Eingabesymbol a höchstens einen Nachfolgezustand $\delta(q, a)$ gibt, zu restriktiv, da man in manchen Zuständen für den Übergang mit einem Symbol a mehrere Möglichkeiten angeben will, ohne festzulegen, welche davon gewählt wird. Solche Entscheidungsfreiheiten in der Modellierung von Abläufen bezeichnet man als **nichtdeterministisch**. Nichtdeterministische Modelle sind oft einfacher aufzustellen und leichter zu verstehen als deterministische.

Beispiel 7.58. In Beispiel 7.10((c)) haben wir einen (recht komplizierten) DFA A_3 mit

$$L(A_3) \;=\; \{w \in \{a,b\}^* : \text{der } \textbf{vorletzte} \text{ Buchstabe von } w \text{ ist ein } a\}$$

kennengelernt. Dieselbe Sprache wird auch von dem deutlich einfacheren **nichtdeterministischen endlichen Automaten** (kurz: NFA) A_4 mit der folgenden graphischen Darstellung akzeptiert:



Ein Eingabewort $w \in \{a,b\}^*$ wird von dem NFA A_4 genau dann akzeptiert, wenn es in der graphischen Darstellung mindestens einen Weg gibt, der im Startzustand  beginnt, dessen Kanten mit w beschriftet sind, und der im akzeptierenden Zustand  endet.

Hier ist die präzise Beschreibung.

NFA

Definition 7.59 (NFA). Ein **nichtdeterministischer endlicher Automat**[2] (kurz: **NFA**) $A = (\Sigma, Q, \delta, q_0, F)$ besteht aus:

Eingabealphabet

- einer endlichen Menge Σ, dem so genannten **Eingabealphabet**,

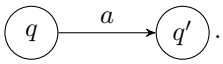
Zustandsmenge

- einer endlichen Menge Q, der so genannten **Zustandsmenge**,

- einer Funktion[3] $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$, der so genannten **Übergangsfunktion**, die jedem Zustand $q \in Q$ und jedem Symbol $a \in \Sigma$ eine **Menge $\delta(q,a)$ von möglichen Nachfolgezuständen** zuordnet (beachte: möglicherweise ist $\delta(q,a) = \emptyset$ – dann „stürzt" der Automat ab, wenn er im Zustand q ist und das Symbol a liest), Übergangsfunktion
- einem Zustand $q_0 \in Q$, dem so genannten **Startzustand**, Startzustand
- einer Menge $F \subseteq Q$, der so genannten Menge der **Endzustände** bzw. **akzeptierenden Zustände**. Endzustand

Graphische Darstellung von NFAs:

Die graphische Darstellung von NFAs erfolgt analog der graphischen Darstellung von DFAs. Ist $q \in Q$ ein Zustand und ist $a \in \Sigma$ ein Eingabesymbol, so gibt es **für jeden Zustand $q' \in \delta(q,a)$** in der graphischen Darstellung des NFAs einen mit dem Symbol a beschrifteten Pfeil von Knoten (q) zu Knoten (q'), d.h.



Die von einem NFA A akzeptierte Sprache $L(A)$:

Definition 7.60. Sei $A = (\Sigma, Q, \delta, q_0, F)$ ein NFA.

(a) Sei $n \in \mathbb{N}$ und sei $w = a_1 \cdots a_n$ ein Eingabewort der Länge n. Das Wort w wird genau dann vom NFA A akzeptiert, wenn es in der graphischen Darstellung von A einen im Startzustand

beginnenden Weg der Länge n gibt, dessen Kanten mit den Symbolen $a_1, \ldots, a_n$ beschriftet sind und der in einem akzeptierenden Zustand endet.

(b) Die von A akzeptierte Sprache $L(A)$ ist

$$L(A) \;:=\; \{w \in \Sigma^* : A \text{ akzeptiert } w\}.$$

Ähnlich wie bei DFAs können wir auch für NFAs eine erweiterte Überführungsfunktion $\widehat{\delta}$ definieren. Für einen Zustand q und ein Eingabewort w gibt $\widehat{\delta}(q,w)$ die **Menge** aller Zustände an, die durch Verarbeiten des Worts w erreicht werden können, wenn der Automat im Zustand q beginnt. Dies wird durch die folgende Definition präzisiert.

Definition 7.61 (Die erweiterte Übergangsfunktion $\widehat{\delta}$ eines NFAs).
Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein NFA. Die Funktion

$$\widehat{\delta} \;:\; Q \times \Sigma^* \;\to\; \mathcal{P}(Q)$$

ist rekursiv wie folgt definiert:

- F.a. $q \in Q$ ist $\widehat{\delta}(q, \varepsilon) \;:=\; \{q\}$.

[2] engl.: non-deterministic finite automaton, kurz: NFA
[3] Zur Erinnerung: $\mathcal{P}(Q)$ bezeichnet die **Potenzmenge** von Q.

- F.a. $q \in Q$, $w \in \Sigma^*$ und $a \in \Sigma$ ist $\quad \widehat{\delta}(q, wa) \;:=\; \bigcup_{q' \in \widehat{\delta}(q,w)} \delta(q', a).$

Beachte, dass ein Wort w genau dann von einem NFA $A := (\Sigma, Q, \delta, q_0, F)$ akzeptiert wird, wenn gilt:

$$\widehat{\delta}(q_0, w) \cap F \;\neq\; \emptyset.$$

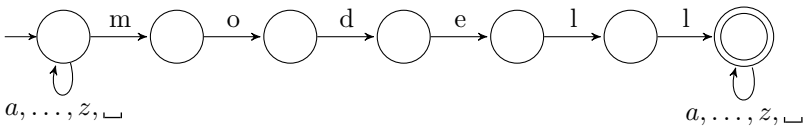
Somit ist

$$L(A) \;=\; \{\, w \in \Sigma^* \,:\, \widehat{\delta}(q_0, w) \cap F \;\neq\; \emptyset \,\}.$$

Ein Anwendungsbeispiel: Stichwort-Suche in Texten

Gegeben:	Ein Stichwort, z.B. „modell“
Eingabe:	Ein Text, der aus den Buchstaben „a“ bis „z“ sowie dem Leerzeichen „$\sqcup$“ besteht
Frage:	Kommt das Stichwort „modell“ irgendwo im Eingabetext vor? Der Eingabetext soll genau dann akzeptiert werden, wenn er das Stichwort „modell“ enthält.

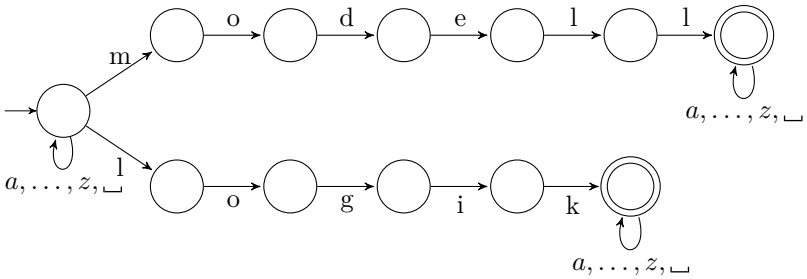
Graphische Darstellung eines NFAs, der dies bewerkstelligt:



Auf ähnliche Art können auch Varianten dieser Stichwortsuche behandelt werden, zum Beispiel die Frage

> Kommt mindestens eins der Stichworte „modell“ bzw. „logik“ im Eingabetext vor?

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



7.5.1. Äquivalenz von NFAs und DFAs

Wir haben mit NFAs und DFAs zwei Sorten von endlichen Automaten kennen gelernt. Sicherlich ist jeder DFA auch ein NFA. Die Frage drängt sich aber auf, ob NFAs möglicherweise „mehr“ können als DFAs.

Der folgende Satz beantwortet diese Frage mit „nein“.

Satz 7.62. Für jeden NFA $A = (\Sigma, Q, \delta, q_0, F)$ gibt es einen DFA $A' = (\Sigma, Q', \delta', q_0', F')$ mit $L(A') = L(A)$. D.h.: NFAs und DFAs modellieren dieselbe Klasse von Sprachen.

Beweis: Sei $A = (\Sigma, Q, \delta, q_0, F)$ der gegebene NFA.
Idee: Wir konstruieren einen DFA $A' = (\Sigma, Q', \delta', q_0', F')$, der in seinem aktuellen Zustand $q' \in Q'$ die Menge aller Zustände abspeichert, in denen der Automat A in der aktuellen Situation sein könnte. Wir definieren die Komponenten von A' daher wie folgt:

- Eingabealphabet Σ,
- Zustandsmenge $Q' := \mathcal{P}(Q)$,
- Startzustand $q_0' := \{q_0\}$,
- Endzustandsmenge $F' := \{X \in Q' \,:\, X \cap F \neq \emptyset\}$,
- Übergangsfunktion $\delta' : Q' \times \Sigma \to Q'$, wobei für alle $X \in Q'$ und alle $a \in \Sigma$ gilt:

$$\delta'(X, a) \;:=\; \bigcup_{q \in X} \delta(q, a).$$

Mit vollständiger Induktion nach n kann man leicht nachweisen (Details: **Übung**), dass für alle $n \in \mathbb{N}$ und jedes $w \in \Sigma^*$ mit $|w| = n$ gilt:

$$\widehat{\delta}'(q_0', w) \;=\; \widehat{\delta}(q_0, w).$$

Daraus folgt, dass für jedes Eingabewort $w \in \Sigma^*$ gilt:

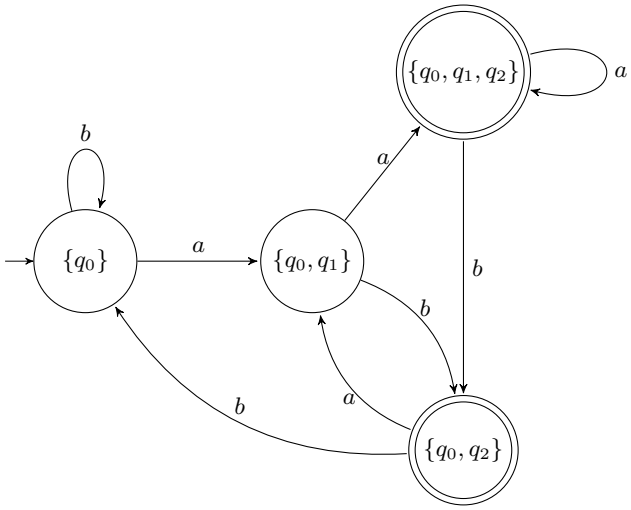
$$\begin{aligned} w \in L(A') \quad &\iff \quad \widehat{\delta}'(q_0', w) \in F' \\ &\iff \quad \widehat{\delta}'(q_0', w) \cap F \neq \emptyset \\ &\iff \quad \widehat{\delta}(q_0, w) \cap F \neq \emptyset \\ &\iff \quad w \in L(A). \end{aligned}$$

Insbesondere ist daher $L(A') = L(A)$. □

Die im obigen Beweis durchgeführte Konstruktion eines DFAs A' wird auch **Potenzmengenkonstruktion** (engl: **subset construction**) genannt.

Potenzmengenkonstruktion

Beispiel 7.63. Wir führen die Potenzmengenkonstruktion an dem NFA A_3 aus Beispiel 7.58 durch, wobei wir die Zustände von A_3, von links nach rechts gelesen, mit q_0, q_1 und q_2 bezeichnen. Im folgenden ist die graphische Darstellung des aus der Potenzmengen-Konstruktion resultierenden DFAs A_3' angegeben, wobei nur solche Zustände aus $\mathcal{P}(\{q_0, q_1, q_2\})$ berücksichtigt werden, die vom Startzustand $q_0' := \{q_0\}$ aus erreicht werden können:



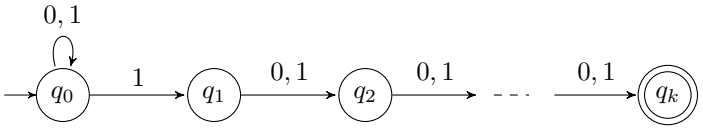
Zwar akzeptieren DFAs und NFAs dieselbe Klasse von Sprachen, nämlich die Klasse der regulären Sprachen, ihre Größen können aber höchst unterschiedlich sein.

Satz 7.64. Die Sprache

$$L_k = \{0,1\}^* \cdot 1 \cdot \{0,1\}^{k-1}$$

kann durch einen NFA mit $k+1$ Zuständen akzeptiert werden. Jeder DFA, der L_k akzeptiert benötigt hingegen mindestens 2^k Zustände.

Beweis: Wir entwerfen einen NFA für L_k, der die „richtige“ Position der Eins rät und anschließend mit $k-1$ Zuständen verifiziert, dass $k-1$ weitere Buchstaben folgen.



Also sind für NFAs $k+1$ Zustände ausreichend. Wie groß muss ein DFA für L_k sein? Wir zeigen

$$\text{Index}(L_k) \geqslant 2^k.$$

Dazu genügt der Nachweis, dass je zwei verschiedene Worte $u, w \in \{0,1\}^k$ inäquivalent bzgl. der Nerode-Relation sind: Da $u \neq w$ gibt es eine Position i mit z.B.$u_i = 1$ und $w_i = 0$. Für $v = 0^{i-1}$ ist $uv \in L_k$, aber $wv \notin L_k$. □ □

Die Potenzmengenkonstruktion scheint auf den ersten Blick viele unnütze Zustände in den DFA einzubauen, aber wie wir gerade gesehen haben lässt sich manchmal eine riesige Zustandsmenge für DFAs nicht vermeiden.

7.6. Reguläre Ausdrücke

Reguläre Ausdrücke beschreiben Mengen von Worten, die nach bestimmten Regeln bzw. „Mustern“ aufgebaut sind.

Beispiel 7.65. Die Menge aller Worte über dem Alphabet $\{a, b\}$, deren vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a \mid b)^* \, a \, (a \mid b)$$

Definition 7.66 (Reguläre Ausdrücke: Syntax). Sei Σ ein endliches Alphabet.
Die Menge aller **regulären Ausdrücke über Σ** ist rekursiv wie folgt definiert:

reguläre Ausdrücke

Basisregeln:

- $\emptyset$ ist ein regulärer Ausdruck über Σ („leere Menge“).
- ε ist ein regulärer Ausdruck über Σ („leeres Wort“).
- Für jedes $a \in \Sigma$ gilt: a ist ein regulärer Ausdruck über Σ.

Rekursive Regeln:

- Ist R ein regulärer Ausdruck über Σ, so ist auch R^* ein regulärer Ausdruck über Σ („Kleene-Stern“).
- Sind R und S reguläre Ausdrücke über Σ, so ist auch
  - $(R \cdot S)$ ein regulärer Ausdruck über Σ („Konkatenation“).
  - $(R \mid S)$ ein regulärer Ausdruck über Σ („Vereinigung“).

Definition 7.67 (Reguläre Ausdrücke: Semantik). Sei Σ ein endliches Alphabet.
Jeder reguläre Ausdruck R über Σ **beschreibt** (oder definiert) eine Sprache $L(R) \subseteq \Sigma^*$, die induktiv wie folgt definiert ist:

- $L(\emptyset) \;:=\; \emptyset$.
- $L(\varepsilon) \;:=\; \{\varepsilon\}$.
- Für jedes $a \in \Sigma$ gilt: $L(a) \;:=\; \{a\}$.
- Ist R ein regulärer Ausdruck über Σ, so ist

$$L(R^*) \;:=\; (L(R))^* = \{\varepsilon\} \;\cup\; \{w_1 \cdots w_k : k \in \mathbb{N}_{>0},\, w_1 \in L(R), \ldots, w_k \in L(R)\}.$$

- Sind R und S reguläre Ausdrücke über Σ, so ist
  - $L\big((R \cdot S)\big) \;:=\; L(R) \cdot L(S) = \{wu : w \in L(R),\, u \in L(S)\}$.
  - $L\big((R \mid S)\big) \;:=\; L(R) \cup L(S)$.

Notation 7.68. Zur vereinfachten Schreibweise und besseren Lesbarkeit von regulären Ausdrücken vereinbaren wir folgende Konventionen:

- Den „Punkt“ bei der Konkatenation $(R \cdot S)$ darf man weglassen.
- Bei Ketten gleichartiger Operatoren darf man Klammern weglassen: z.B. schreiben wir kurz $(R_1 \mid R_2 \mid R_3 \mid R_4)$ statt $\Big(\big((R_1 \mid R_2) \mid R_3\big) \mid R_4\Big)$ und $(R_1 R_2 R_3 R_4)$ statt $\Big(\big((R_1 R_2) R_3\big) R_4\Big)$.

- „Präzedenzregeln“: (1): * bindet stärker als $\cdot$
  (2): $\cdot$ bindet stärker als $|$
- Äußere Klammern, die einen regulären Ausdruck umschließen, dürfen weggelassen werden.
- Zur besseren Lesbarkeit dürfen zusätzliche Klammern und auch Klammern von neuem Typ (z.B. eckige Klammern) benutzt werden.

Beispiel 7.69.

(a) $a \,|\, bc^*$ ist eine verkürzte Schreibweise für den regulären Ausdruck $(a \,|\, (b \cdot c^*))$.

Die von diesem regulären Ausdruck beschriebene Sprache ist

$$L(a \,|\, bc^*) \;=\; \{a\} \;\cup\; \{w \in \{a,b,c\}^* : \text{der erste Buchstabe von } w \text{ ist ein } b \text{ und alle weiteren Buchstaben von } w \text{ sind } c\text{'s}\}.$$

(b) $L((a \,|\, b)^*) \;=\; \{a,b\}^*$.

(c) Die Menge aller Worte über $\{a,b,c\}$, in denen abb als Teilwort vorkommt, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a \,|\, b \,|\, c)^* abb\, (a \,|\, b \,|\, c)^*.$$

(d) Die Menge aller Worte über $\{a,b,c\}$, deren letzter oder vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a \,|\, b \,|\, c)^* a\, (\varepsilon \,|\, a \,|\, b \,|\, c).$$

Beispiel 7.70.

(a) Wir wollen einen regulären Ausdruck angeben, der die Sprache all jener Worte über dem Alphabet $\Sigma = \{0,1,\ldots,9,/\}$ definiert, die Telefonnummern der Form

$$\textit{Vorwahl}/\textit{Nummer}$$

kodieren, wobei *Vorwahl* und *Nummer* nicht-leere Ziffernfolgen sind, *Vorwahl* mit einer Null beginnt und *Nummer nicht* mit einer Null beginnt. Worte dieser Sprache sind z.B. 069/7980 und 06131/3923378, aber *nicht* die Worte 069/798-0, 0697980, 69/7980 und 069/07980.

Der folgende Ausdruck definiert die gewünschte Sprache:

$$0\,(0 \,|\, 1 \,|\, \cdots \,|\, 9)^* \,/\, (1 \,|\, \cdots \,|\, 9)\,(0 \,|\, 1 \,|\, \cdots \,|\, 9)^*$$

(b) Es sei R der folgende reguläre Ausdruck:

$$\big(\varepsilon \,|\, 069/\big)\;\; 798\;\; \big(\varepsilon \,|\, \text{-}\big)\;\; \big(\;0 \;\;|\;\; (1 \,|\, \cdots \,|\, 9)(0 \,|\, 1 \,|\, \cdots \,|\, 9)^*\;\big)$$

R definiert eine Sprache, die beispielsweise die Worte 069/798-0 und 7980 enthält, aber nicht das Wort 069/798-028362.

7.7. Zusammenfassung und Ausblick

Deterministische endliche Automaten (DFAs) bieten eine Methode für die Modellierung von Handlungsabläufen. Mit der erweiterten Übergangsfunktion $\widehat{\delta}$ wird die Arbeitsweise eines DFAs beschrieben und die Sprache des DFAs definiert. Moore- und Mealy-Automaten als DFAs mit Ausgabe werden in der technischen Informatik für die (partielle) Modellierung von Schaltwerken eingeführt.

Die Minimierung von DFAs (oder Moore-, bzw. Mealy-Automaten) ist wichtig, um mit DFAs effizient arbeiten zu können. Die Verschmelzungsrelation $\equiv_A$ für einen DFA A definiert die Äquivalenz von Zuständen von A: Wir erhalten den Äquivalenklassenautomaten, wenn äquivalente Zustände von A verschmolzen werden. Mit Hilfe der Nerode-Relation haben wir im Satz von Myhill, Nerode I gezeigt, dass der Äquivalenzklassenautomat minimal ist: Der Index einer Sprache stimmt überein mit der minimalen Zustandszahl eines DFA für die Sprache und der wiederum mit der Zustandszahl (irgend-)eines Äquivalenzklassenautomaten.
Verschmelzungsrelation und Nerode-Relation sind Äquivalenzrelationen: Die Äquivalenzklassen einer Äquivalenzrelation über V bilden eine disjunkte Vereinigung von V.

Die Klasse der regulären Sprachen ist eine der wichtigsten Sprachklassen. Reguläre Sprachen können aus verschiedensten Perspektiven definiert werden: Als Klasse der von DFAs akzeptierten Sprachen, als Klasse der von NFAs akzeptierten Sprachen oder als Klasse der von regulären Ausdrücken beschreibbaren Sprachen. Abgesehen von DFAs, NFAs und regulären Ausdrücken kann man reguläre Sprachen auch durch bestimmte Grammatiken erzeugen, so genannte **reguläre Grammatiken** — das sind kontextfreie Grammatiken (vgl. Kapitel 8), die von einer besonders einfachen Form sind. In der Veranstaltung „Theoretische Informatik 2“ wird die Äquivalenz der verschiedenen Formalismen –DFAs, NFAs, reguläre Ausdrücke, reguläre Grammatiken– nachgewiesen und ergänzt. Wir zeigen in Satz 8.10, dass es für jeden NFA eine äquivalente reguläre Grammatik gibt.

Der Satz von Myhill, Nerode II gibt eine exakte Charakterisierung der Klasse regulärer Sprachen an: Eine Sprache ist genau dann regulär, wenn ihr Index endlich ist. Um zu zeigen, dass eine Sprache L nicht regulär ist, genügt die Angabe von unendlich vielen Worten, die paarweise nicht äquivalent in Bezug auf die Nerode-Relation $\equiv_L$ sind. Das Pumping-Lemma bietet eine zweite Möglichkeit für den Nachweis der Nicht-Regularität.

Fragen über Automaten

Eine typische Fragestellung bzgl. DFAs oder NFAs ist das so genannte **Leerheitsproblem**:

> Gegeben sei ein DFA oder NFA A. Wie kann man herausfinden, ob $L(A) \neq \emptyset$ ist, d.h. ob es (mindestens) ein Eingabewort gibt, das von A akzeptiert wird?

Durch Betrachtung der graphischen Darstellung von A kann diese Frage leicht mit Hilfe der Tiefensuche (Kapitel 5.1.2.1) beantwortet werden:

> Teste, ob es in der graphischen Darstellung von A einen Weg gibt, der vom Startzustand zu einem akzeptierenden Zustand führt.

Wir können einen DFA effizient minimieren und wir können effizient feststellen, ob zwei DFAs dieselbe Sprache akzeptieren. Beide Fragen stellen sich für NFAs als äußerst kompliziert heraus. Mehr erfahren Sie in der Veranstaltung „Theoretische Informatik 2“.

7.8. Literaturhinweise zu Kapitel 7

[14] Kapitel 7.1

[12] Kapitel 2 und 3

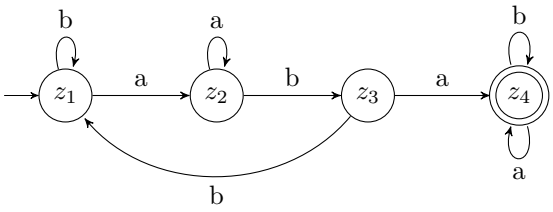
[26] Kapitel 4.1 und 4.2

[27] Kapitel 4.1

[23] Kapitel 1.2

7.9. Übungsaufgaben zu Kapitel 7

Aufgabe 7.1. Sei A der folgende endliche Automat über dem Alphabet $\Sigma = \{a, b\}$:



(i) Geben Sie die Menge der Zustände, den Startzustand, die Menge der akzeptierenden Zustände und die Übergangsfunktion von A an.

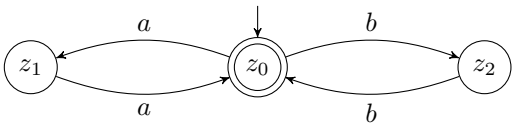
(ii) Welche der folgenden Worte werden von A akzeptiert, welche nicht?

- *bbaabba*
- *abbaaababbba*
- *aabbaab*

(iii) Geben Sie ein möglichst kurzes Wort an, das von A akzeptiert wird.

(iv) Beschreiben Sie umgangssprachlich, welche Sprache $L(A)$ von A akzeptiert wird.

Aufgabe 7.2. Sei A der folgende endliche Automat über dem Alphabet $\Sigma = \{a, b\}$:



(a) Geben Sie Folgendes für A an:

- die Menge der Zustände,
- den Startzustand,
- die Menge der Endzustände und
- die Übergangsfunktion.

(b) Welche der folgenden Worte werden von A akzeptiert, welche nicht?

- $aaabb$
- $aabz_0aa$
- $bbaaaabbbb$
- $bbbbaab$

(c) Geben Sie die sieben kürzesten Worte an, die A akzeptiert.

(d) Beschreiben Sie umgangssprachlich, aus welchen Worten die von A akzeptierte Sprache $L(A)$ besteht.

Aufgabe 7.3. Der DFA A sei durch das folgende Python-Programm gegeben:

```
def dfa(word):                # word bestehe nur aus den Symbolen 'a', 'b', 'c'
    state = 0                 # Anfangszustand
    for symbol in word:       # Lies jedes Symbol des Eingabewortes
        if state == 0:        # aktueller Zustand: 0
            if symbol == 'a':
                state = 1     # Uebergang von 0 nach 1: delta(0,a) = 1
            elif symbol == 'b':
                state = 0
            else:
                state = 2
        elif state == 1:
            if symbol == 'a':
                state = state - 1
            if symbol == 'c':
                state = 2
        elif state == 2:
            state = 2

    if state == 0:
        return (state, True)   # Eingabewort akzeptieren
    else:
        return (state, False)  # Eingabewort verwerfen
```

Kommentar: Wir haben diese nicht sehr elegante Implementierung gewählt, um verschiedene Implementierungsmöglichkeiten von DFAs darzustellen.

a) Geben Sie A (mit dem Alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$) als Zustandsdiagramm an.

b) Welche der folgenden Wörter liegen in $L(A)$, welche nicht?

- $w_1 := \varepsilon$
- $w_2 := \mathtt{baba}$
- $w_3 := \mathtt{abaa}$
- $w_4 := \mathtt{acbab}$

c) Beschreiben Sie die vom DFA A akzeptierte Sprache $L(A) \subseteq \{\mathtt{a}, \mathtt{b}, \mathtt{c}\}^*$ mathematisch oder umgangssprachlich.

Aufgabe 7.4. Der Radiosender Radio Mod! führt zum Beginn des neuen Jahres eine 2015er-Retro-Woche durch. Teil dieser Woche ist das Rückblick-Marathon-Gewinnspiel. Dieses läuft folgendermaßen ab: Durch eine Internet-Abstimmung wurden die drei beliebtesten Lieder des Vorjahres bestimmt, nämlich „Automatenlos durch die Nacht“, sowie „Beweist Du (wieviel

Sternlein stehen (Dancefloor Remix))“ und „Un-Markov-Chain My Heart“. Zwischen Mitternacht und sechs Uhr morgens werden nun ausschließlich diese drei Lieder gespielt; dabei müssen die Zuhörer die Reihenfolge der Lieder aufmerksam verfolgen. Zu Beginn der Sendung wird eine Reihenfolge von Liedern mitgeteilt. Sobald Lieder in dieser Reihenfolge abgespielt wurden, gewinnt der erste Anruf[4] bei Radio Mod! einen Preis, nämlich 10000 Euro oder einen Elefanten. Lester wünscht sich schon lange einen Elefanten – allerdings sind die Bedingungen des Gewinnspiels doch recht qualvoll (nicht nur wegen der Uhrzeit, sondern auch weil die Lieder seinem Musikgeschmack überhaupt nicht entsprechen). Daher fragt er seine Schwester Eliza um Rat. Um Lester das Verfolgen der Lieder einfacher zu machen, erstellt Eliza ihm jeweils einen deterministischen endlichen Automaten über dem Alphabet $\Sigma = \{A, B, U\}$ (die Buchstaben stehen dafür jeweils für die Anfangsbuchstaben der Lieder) -– so muss sich Lester wenigstens nicht die bisher gehörten Lieder merken, sondern nur den aktuellen Zustand des DFAs (das geht zur Not mit einem Finger und ist gerade zu später Stunde deutlich einfacher).

(a) In der ersten Nacht führt die folgende Reihenfolge von Liedern zu einer Gewinnchance: Zuerst „Automatenlos“, dann „Beweist Du“ und abschließend „Un-Markov-Chain My Heart“.

Konstruieren Sie einen deterministischen endlichen Automaten für die Sprache $L_1 := \{ w \in \Sigma^* : w \text{ endet auf } ABU \}$. Verwenden Sie so wenige Zustände wie möglich. Sie müssen die Korrektheit nicht beweisen.

(b) In der zweiten Nacht wird die Schwierigkeit ein wenig erhöht. Nun führt die folgende Reihenfolge von Liedern zu einer Gewinnchance: Zuerst „Automatenlos“, dann „Beweist Du“, dann wieder „Automatenlos“ und abschließend „Un-Markov-Chain My Heart“. Konstruieren Sie einen deterministischen endlichen Automaten für die Sprache $L_2 := \{ w \in \Sigma^* : w \text{ endet auf } ABAU \}$. Verwenden Sie so wenige Zustände wie möglich. Sie müssen die Korrektheit nicht beweisen.

Aufgabe 7.5. Der deutsch-amerikanische Komponist Gershon Kingsley hat 1969 das Instrumentalmusikstück *Popcorn* komponiert. Bis heute sind über 500 verschiedene Cover-Versionen davon entstanden, im letzten Jahr erst gab es Interpretationen von *Faith No More* und *Muse*.

Nehmen Sie an, es existiert eine elektronische Sammlung mit Notenschriften von allen jemals produzierten Musikstücken, die Kingsley nach Cover-Versionen seines Stücks durchsuchen möchte. Eine solche Cover-Version ist für Kingsley jedes Stück, das wie *Popcorn* in der Tonart h-Moll geschrieben ist bzw. nur die Noten h, $c^\sharp$, d, e ,$f^\sharp$, g oder a benutzt. Außerdem muss natürlich das berühmte *Popcorn*-Thema $hahf^\sharp df^\sharp h$ als Teilstück enthalten sein.

Geben Sie einen DFA A in graphischer Darstellung an, der eine Notenschrift in der Sammlung genau dann akzeptiert, wenn es sich für Kingsley um eine Cover-Versionen seines Stücks *Popcorn* handelt.

Hinweis: Lassen Sie im Zustandsdiagramm Übergänge der Form $\delta(\cdot, \cdot) = q_0$ der Übersichtlichkeit halber weg.

Aufgabe 7.6. Jede natürliche Zahl n lässt sich als *Dualzahl*, d.h., in der Form $[n]_2 = z_l z_{l-1} \cdots z_0$ darstellen, so dass $z_i \in \{0, 1\}$ für $0 \leqslant i \leqslant l$ mit $l \in \mathbb{N}$ ist und $n = \sum_{i=0}^{l} z_i \cdot 2^i$ gilt. Die Zahl $[n]_2$ wird als die *Dualdarstellung* der Zahl n bezeichnet. Dualzahlen können auf herkömmliche Weise schriftlich addiert werden, wobei der Übertrag bei der Zwei erfolgt.

[4] Tarifhinweis: 50 Cent/Anruf aus dem Festnetz, ggf. abweichende Kosten aus dem Mobilnetz.

Gegeben sei das folgende Eingabealphabet

$$\Sigma := \left\{ \begin{array}{|c|}\hline 0\\0\\\hline 0\\\hline\end{array}, \begin{array}{|c|}\hline 0\\0\\\hline 1\\\hline\end{array}, \begin{array}{|c|}\hline 0\\1\\\hline 0\\\hline\end{array}, \begin{array}{|c|}\hline 0\\1\\\hline 1\\\hline\end{array}, \begin{array}{|c|}\hline 1\\0\\\hline 0\\\hline\end{array}, \begin{array}{|c|}\hline 1\\0\\\hline 1\\\hline\end{array}, \begin{array}{|c|}\hline 1\\1\\\hline 0\\\hline\end{array}, \begin{array}{|c|}\hline 1\\1\\\hline 1\\\hline\end{array} \right\}.$$

Geben Sie einen DFA A an, der ein Wort w aus Σ^* genau dann akzeptiert, wenn w eine korrekte Addition zweier Dualzahlen $[n]_2$ und $[m]_2$ mit $n, m \in \mathbb{N}$ darstellt. So ist beispielsweise $w \in L(A)$ für

$$w = \begin{array}{|c|c|c|c|}\hline 0&1&0&1\\0&1&1&1\\\hline 1&1&0&0\\\hline\end{array} \quad \text{, weil} \quad \begin{array}{rrcr} & 0101 & = & [5]_2 \\ + & 0111 & = & [7]_2 \\ \hline & 1100 & = & [12]_2 \end{array} .$$

Hinweis: Beachten Sie, dass ein endlicher Automat jedes Eingabewort von links nach rechts liest. Begründen Sie kurz, warum der von Ihnen angegebene DFA die verlangte Sprache akzeptiert.

Aufgabe 7.7. Das *Transmission Control Protocol* (kurz: *TCP*) gehört zu den wichtigsten Protokollen in vernetzten IT-Systemen. Kurz gesagt steuert TCP die Kommunikation zwischen zwei Rechnern, indem es *Sitzungen* zwischen den beiden Rechnern eröffnet. Ist eine Sitzung eröffnet, können beide Rechner beliebig viele Datenpakete austauschen und beispielsweise fehlerhaft erhaltene Nachrichten erneut anfordern. In dieser Aufgabe betrachten wir eine vereinfachte Variante von TCP, hier *Simple-TCP* genannt.

Seien r_1 und r_2 zwei Rechner, die mittels Simple-TCP eine Sitzung eröffnen möchten. Hierfür stehen ihnen die *Befehle* `SYN` (wie *synchronize*) und `ACK` (wie *acknowledge*) zur Verfügung. Damit r_1 erfolgreich eine Sitzung mit r_2 eröffnen kann, führen die Rechner die folgenden drei Schritte der Reihe nach aus. Man spricht daher auch einem *Drei-Wege-Handschlag*:

1. Zunächst sendet r_1 ein `SYN` an r_2.
2. Daraufhin antwortet r_2 mit `SYN` und `ACK` (oder mit `ACK` und `SYN` in umgekehrter Reihenfolge).
3. Schließlich bestätigt r_1, indem er ein `ACK` an r_2 sendet.

Vor und nach jedem der drei Schritte dürfen von beiden Rechnern `SYN`- und `ACK`-Befehle gesendet werden. Auch während r_2 den Schritt 2 ausführt, darf r_1 beliebige `SYN`- und `ACK`-Sequenzen senden.

Modellieren Sie die Eröffnung einer Sitzung druch Rechner r_1 mittels eines DFAs $A_{\text{Simple-TCP}}$. Verwenden Sie das Alphabet $\Sigma = \{(\texttt{SYN},\text{-}), (\texttt{ACK},\text{-}), (\text{-},\texttt{SYN}), (\text{-},\texttt{ACK})\}$, um die Kommunikation der beiden Rechner zu beschreiben. Beispielsweise repräsentiert das Symbol (`SYN`,-), dass r_1 ein `SYN`an r_2 sendet (und r_2 schweigt).

Der DFA $A_{\text{Simple-TCP}}$ soll genau die Wörter über Σ^* akzeptieren, die einen Drei-Wege-Handschlag enthalten, der durch Rechner r_1 eröffnet wird.

Aufgabe 7.8. Eine Bande besonders berüchtigter Banditen befindet sich nach einem beispiellosen Banküberfall bei Baden-Baden auf der Flucht. Nach Zeugenaussagen benutzt die Bande vier verschiedene Fluchtwagen, davon zwei Pkws der Marke Audi, einen Citroën 2CV und einen Dacia Duster. Außerdem liegen der Polizei Hinweise vor, dass sich der Fluchtwagen-Konvoi in der Reihenfolge (von vorne nach hinten) Audi, Dacia, Audi, Citroën auf der Autobahn A 5 in Richtung Norden bewegt.

Die neue Cyber-Einheit der Bundespolizei beteiligt sich im Rahmen des Projekts „Fahndung 4.0“ und setzt dabei auf allerneueste Technik: Mithilfe der Verkehrsüberwachungskameras und der automatischen Automarken-Erkennung soll der Konvoi aufgespürt werden.

Die Verkehrsüberwachungskameras nehmen alle vorbeifahrenden Autos auf und leiten das Bildmaterial an die Automarken-Erkennung weiter, die daraus eine Autofolge $a_1 a_2 \dots a_n \in \{\texttt{A}, \texttt{C}, \texttt{D}, \texttt{X}\}^n$ erstellt. Dabei steht a_i für die Automarke des i-ten gesichteten Autos und `A`, `C` bzw. `D` bezeichnen die Automarken Audi, Citroën bzw. Dacia, während `X` für jede andere Marke außer den drei genannten steht.

(a) Helfen Sie bei der Fahndung, indem Sie einen DFA A_1 mit möglichst wenigen Zuständen konstruieren, welcher die erfasste Autofolge liest und genau dann „Alarm schlägt“, sobald ein Konvoi der Form Audi, Dacia, Audi, Citroën gesichtet wird. Eine grafische Darstellung des Automaten ohne Begründung ist ausreichend.

(b) Einer brandaktuellen Zeugenaussage zufolge haben die Banditen auf einem Rastplatz ihre Fluchtfahrzeuge gewechselt und sind nun in einem großen Konvoi bestehend aus 177 Audis und 1 Citroën unterwegs. Es ist bekannt, dass sich der Citroën entweder am Anfang oder am Ende des Konvois befindet, und außerdem, dass sich kein anderes Auto zwischen den Fluchtfahrzeugen aufhält.

Konstruieren Sie einen DFA A_2, der „Alarm schlägt“, sobald dieser neue Konvoi gesichtet wird. Beschreiben Sie dazu insb. die Zustandsmenge und die Übergangsfunktion und erläutern Sie Ihre Idee dahinter.

(c) Lösen Sie Aufgabenteil b) in der folgenden, schwierigeren Variante: Es ist unbekannt, an welcher Stelle im Konvoi sich der Citroën befindet.

Hinweis: Es reicht **nicht** aus, in jedem Zustand die Anzahl der (möglicherweise) zum Konvoi gehörenden Audis und die Anzahl der (möglicherweise) zum Konvoi gehörenden Citroëns zu zählen.

Aufgabe 7.9. Für $b \in \mathbb{N}$ mit $b \geqslant 2$ betrachten wir das Eingabealphabet $\Sigma_b := \{0, 1, ..., b-1\}$. Wir identifizieren jedes Wort $w = w_0 w_1 w_2 \cdots w_{|w|-1} \in \Sigma_b^*$ mit der natürlichen Zahl

$$z_b(w) := \begin{cases} 0 & \text{, falls } w = \varepsilon \\ \sum_{i=0}^{|w|-1} w_i \cdot b^{|w|-(i+1)} & \text{, sonst.} \end{cases}$$

Zum Beispiel ist $z_2(101) = 5$, $z_3(101) = 10$ und $z_4(101) = 17$, sowie $z_3(12) = 5$ und $z_4(12) = 6$.

Geben Sie für jede der folgenden drei Sprachen jeweils einen endlichen Automaten mit möglichst wenigen Zuständen an, der genau diese Sprache akzeptiert.

$$\begin{aligned} L_1 &:= \{w \in \Sigma_2^* : z_2(w) \text{ ist durch 4 teilbar}\} = \{\, w \in \{0,1\}^* : \text{ex. } k \in \mathbb{N} \text{ mit } z_2(w) = 4 \cdot k \,\} \\ L_2 &:= \{w \in \Sigma_3^* : z_3(w) \text{ ist durch 2 teilbar}\} = \{\, w \in \{0,1,2\}^* : \text{ex. } k \in \mathbb{N} \text{ mit } z_3(w) = 2 \cdot k \,\} \\ L_3 &:= \{w \in \Sigma_4^* : z_4(w) \text{ ist durch 3 teilbar}\} = \{\, w \in \{0,1,2,3\}^* : \text{ex. } k \in \mathbb{N} \text{ mit } z_4(w) = 3 \cdot k \,\}. \end{aligned}$$

Aufgabe 7.10. Sei Σ ein endliches Alphabet und seien $L, L_1, L_2 \subseteq \Sigma^*$ reguläre Sprachen, d. h. es existieren DFAs A, A_1 und A_2 mit $L(A) = L$, $L(A_1) = L_1$ und $L(A_2) = L_2$.

Beweisen Sie:

(a) $\overline{L} := \Sigma^* \setminus L$ ist eine reguläre Sprache, d. h. es gibt einen DFA $\overline{A}$ mit $L(\overline{A}) = \overline{L}$.

(b) $L_1 \cap L_2$ ist eine reguläre Sprache.

Hinweis: Konstruieren Sie einen DFA A der den Durchschnitt $L_1 \cap L_2$ akzeptiert. A sollte die Automaten A_1 und A_2 „gleichzeitig laufen“ lassen. Wie müssen die Übergangsfunktion δ, der Startzustand q_0 und die Menge F der akzeptierenden Zustände gewählt werden?

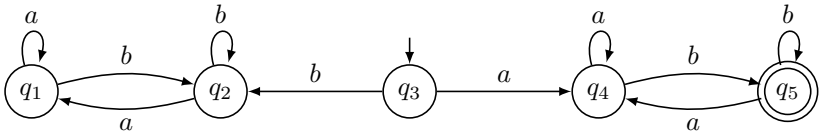
Aufgabe 7.11. Sei Σ ein Alphabet, $w \in \Sigma^+$ ein Wort und $L \subsetneq \Sigma^*$ eine reguläre Sprache. Zeigen Sie:

(a) Die Sprache $w^{-1}L := \{x \in \Sigma^* : wx \in L\}$ ist regulär.

(b) Die Sprache $\text{Präfix}(L) := \{x \in \Sigma^* : \text{es gibt ein } y \in \Sigma^* \text{ mit } xy \in L\}$ ist regulär.

Hinweis: Konstruieren Sie jeweils einen DFA für die beiden Sprachen.

Aufgabe 7.12. Betrachten Sie den folgenden DFA A über dem Alphabet $\Sigma = \{a, b\}$.

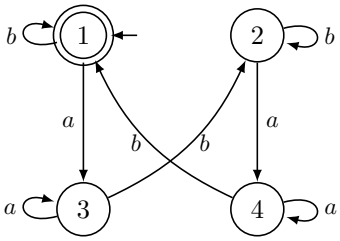


(a) Weisen Sie die folgenden Inäquivalenzen bzgl. der Verschmelzungsrelation $\equiv_A$ nach, indem Sie geeignete Zeugen $z \in \Sigma^*$ angeben.

- $q_3 \not\equiv_A q_4$
- $q_2 \not\equiv_A q_3$
- $q_3 \not\equiv_A q_5$
- $q_1 \not\equiv_A q_4$.

(b) Gibt es in A zwei verschiedene Zustände q_i und q_j, sodass $q_i \equiv_A q_j$ gilt? Begründen Sie Ihre Antwort.

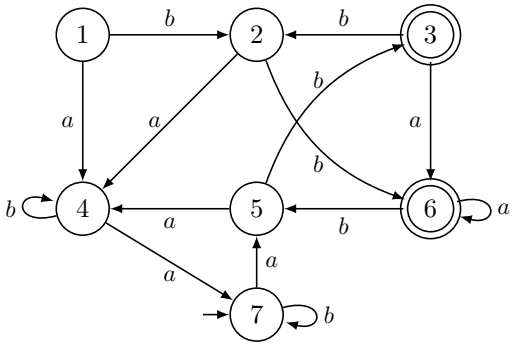
Aufgabe 7.13. Der DFA $M = (\Sigma, Q, \delta, q_0, F)$ mit $\Sigma = \{a, b\}$, $Q = \{1, 2, 3, 4\}$, $q_0 = 1$ und $F = \{1\}$ sei durch folgende Grafik gegeben:



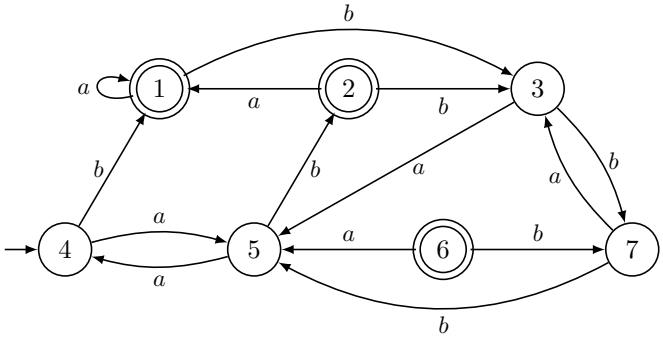
Zeigen Sie, dass M minimal ist.

Aufgabe 7.14. Minimieren Sie die folgenden DFAs, d. h. bestimmen Sie jeweils den Äquivalenzklassenautomaten A_1' bzw. A_2'. Verwenden Sie dazu den Algorithmus aus der Vorlesung und geben Sie auch die Tabelle an, in der Paare inäquivalenter Zustände mit den Mengen M_i markiert sind.

(a) Der DFA $A_1 = (\Sigma, Q, \delta, q_0, F)$ mit $\Sigma = \{a, b\}$, $Q = \{1, 2, 3, 4, 5, 6, 7\}$, $q_0 = 7$ und $F = \{3, 6\}$ sei durch folgende Grafik gegeben:



(b) Der DFA $A_2 = (\Sigma, Q, \delta, q_0, F)$ mit $\Sigma = \{a, b\}$, $Q = \{1, 2, 3, 4, 5, 6, 7\}$, $q_0 = 4$ und $F = \{1, 2, 6\}$ sei durch folgende Grafik gegeben:



Aufgabe 7.15.

(a) Bestimmen Sie für die Sprache

$$H = \Sigma^* \cdot \{aaba\} \cdot \Sigma^* \text{ über dem Alphabet } \Sigma = \{a, b, c\}$$

den Nerode-Automaten in folgenden Schritten:

(i) Geben Sie alle Äquivalenzklassen der Nerode-Relation an und beschreiben Sie jede der Klassen durch Angabe aller in ihr enthaltenen Elemente. (Z. B. $[\varepsilon]_H = \{\dots\}$)

(ii) Wählen Sie für jede Klasse einen (möglichst kurzen) Vertreter und trennen Sie die Vertreter verschiedener Klassen jeweils durch einen Zeugen.

(iii) Geben Sie den Nerode-Automaten in grafischer Darstellung an.

Hinweis: Um alle Äquivalenzklassen zu bestimmen, beginnen Sie mit der Äquivalenzklasse $[\varepsilon]_F$ des leeren Wortes. Was passiert im Nerode-Automaten, wenn nun ein a, b oder c gelesen wird? Sind die Wörter a, b, c äquivalent zu ε? Welche weiteren Äquivalenzklassen treten auf und welche Elemente enthalten sie?

(b) Konstruieren Sie den Nerode-Automaten für die Sprache $S = \Sigma^* \cdot \{aaba\}$ über dem Alphabet $\Sigma = \{a, b, c\}$.

(c) Konstruieren Sie den Nerode-Automaten für die Sprache $P = \{abb\} \cdot \Sigma^*$ über dem Alphabet $\Sigma = \{a, b\}$.

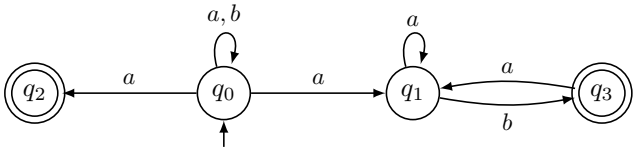
(d) Konstruieren Sie den Nerode-Automaten für die Sprache $L = \Sigma^* \cdot \{aaa, ba, bb\}$ über dem Alphabet $\Sigma = \{a, b\}$.

(e) Konstruieren Sie den Nerode-Automaten für die Sprache

$$V = \{w \in \Sigma^* : \text{ die letzen beiden Buchstaben von } w \text{ sind verschieden. }\}$$

über dem Alphabet $\Sigma = \{a, b, c\}$.

Aufgabe 7.16. Sei A der folgende NFA über dem Alphabet $\Sigma = \{a, b\}$:



(a) Welche der folgenden Wörter liegen in $L(A)$, welche nicht?

$w_1 := a$ $\qquad$ $w_2 := aba$ $\qquad$ $w_3 := bbab$

(b) Beschreiben Sie (mathematisch oder umgangssprachlich) die Sprache $L(A)$.

(c) Konstruieren Sie mithilfe der Potenzmengenkonstruktion einen DFA D, der dieselbe Sprache wie der NFA A akzeptiert. Berücksichtigen Sie in D nur solche Zustände, die vom Startzustand $\{q_0\}$ aus erreichbar sind.

Aufgabe 7.17. Die fünf Studierenden Leo, Uli, Robin, Kaya und Sascha wohnen seit einiger Zeit zusammen in einer WG und ernähren sich ausschließlich von Nudelsuppe (kurz: n), Pizza (kurz: p), Spaghetti mit Tomatensauce (kurz: s) und Tintenfisch-Burgern (kurz: t). Dabei kommt es oft zu leidenschaftlichen Diskussionen, wie häufig die Küche geputzt werden muss.

Da die vier Gerichte die Küche unterschiedlich stark verschmutzen und weil niemand gerne putzen will, einigen sich die Fünf auf das folgende Verfahren: Die Folge der seit dem letzten Putzen gekochten Gerichte wird als ein Wort w über dem Alphabet $\Sigma := \{n, p, s, t\}$ aufgefasst. Jeden Sonntag Nachmittag überprüfen sie das entstandene Wort w auf die folgenden drei Bedingungen:

1) Es ist mindestens dreimal Pizza zubereitet worden.

2) Es ist mindestens zweimal Spaghetti zubereitet worden.

3) Es wurden mindestens einmal Tintenfisch-Burger zubereitet.

Wenn das Wort w mindestens eine dieser drei Bedingungen erfüllt, muss die Küche geputzt werden. Da die Fünf in ihrem Studium gelernt haben, dass NFAs oft kompaktere Definitionen erlauben als DFAs, möchten sie w durch einen NFA überprüfen.

Konstruieren Sie einen NFA A, so dass $L(A)$ genau die Wörter aus Σ^* enthält, die mindestens eine der drei obigen Bedingungen erfüllen. Dabei darf A nicht mehr als fünf Zustände haben.

Aufgabe 7.18. Der *Konami-Code* ist eine Tastenkombination, die in verschiedenen Computerspielen versteckte Extras freischaltet. Auch wenn sich inzwischen unterschiedliche Varianten davon entwickelt haben, so ist doch die folgende klassische Form am weitesten verbreitet:

Diese Zeichenfolge wurde zum ersten Mal im Jahr 1986 in einem Computerspiel der Firma Konami verwendet und führt seitdem nicht nur in Spielen, sondern auch auf Webseiten[5] zu mehr oder weniger nützlichen oder originellen Überraschungen. Auch in Filmen und auf T-Shirts ist sie gelegentlich anzutreffen. Sei $\Sigma = \{$,  ,  , , ,  $\}$.

(a) Geben Sie einen nichtdeterministischen endlichen Automaten A_1 in graphischer Darstellung an, der eine Zeichenfolge über Σ genau dann akzeptiert, wenn diese den Konami-Code enthält (und zwar in der oben dargestellten klassischen Form).

(b) Nicht ganz so bekannt ist die folgende Zeichenfolge, die wir als den *DisMod-Code* bezeichen:

Geben Sie einen nichtdeterministischen endlichen Automaten A_2 in graphischer Darstellung an, der eine Zeichenfolge über Σ genau dann akzeptiert, wenn diese den Konami-Code (wie gehabt in der klassischen Form) oder den DisMod-Code enthält. Elf Zustände sind ausreichend.

Aufgabe 7.19. Der Morsecode basiert auf der Zeichenmenge $\Sigma := \{$▬ ,• ,␣$\}$, wobei ▬ für einen langen Impuls, • für einen kurzen Impuls und ␣ für eine Pause steht. Jeder Buchstabe unseres Alphabets wird durch eine Folge von kurzen und langen Impulsen codiert, der Buchstabe `j` beispielsweise durch •▬▬▬ und `a` durch •▬ . Ein ganzes Wort wird ohne Berücksichtigung von Groß/Kleinschreibung einfach durch die Folge seiner Buchstaben kodiert, jeder Buchstabe wird mit einer Pause abgeschlossen. Auf diese Weise wird das Wort `ja` durch •▬▬▬␣•▬ ␣ kodiert.

In der Anfangszeit der Mobiltelefone und damit vor der Einführung polyphoner Klingeltöne hat der Mobiltelefonhersteller Nokia für alle seine Modelle den Standardton ▬▬␣ für `m` wie message gewählt, um den Empfang einer Nachricht zu signalisieren. Mit der zunehmenden Beliebtheit von Kurznachrichten hat Nokia seine Modelle mit einem weiteren Klingelton in Morsecode ausgestattet: Der „Special“ genannte Ton signalisiert •••␣▬▬␣•••␣ für `SMS`. [6]

Wir stellen uns eine deutsche Version des Standardtons für den Empfang einer Nachricht vor, die ▬•␣ für `N` wie Nachricht ausgibt.

Geben Sie einen nichtdeterministischen endlichen Automaten A in graphischer Darstellung an, der eine beliebige Zeichenfolge aus Σ^* entgegennimmt und diese genau dann akzeptiert, wenn darin beide Zeichenketten ▬•␣ und •••␣▬▬␣•••␣ enthalten sind.

Aufgabe 7.20. Im Beweis von Satz 7.62 wird aus einem NFA $A = (\Sigma, Q, \delta, q_0, F)$ ein DFA $A' = (\Sigma, Q', \delta', q_0', F')$ konstruiert mit $L(A) = L(A')$. Beweisen Sie per Induktion nach n, dass für alle $n \in \mathbb{N}$ und jedes $w \in \Sigma^*$ mit $|w| = n$ gilt:

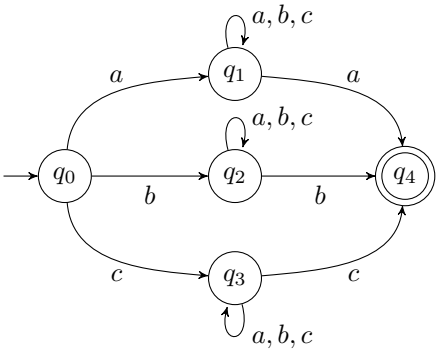
$$\widehat{\delta'}(q_0', w) = \widehat{\delta}(q_0, w).$$

Aufgabe 7.21.

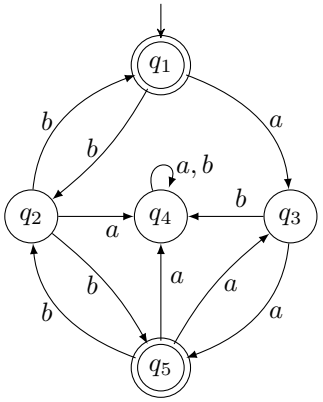
[5] Beispiele finden Sie auf der Seite http://konamicodesites.com/ oder mit der Suchmaschine Ihrer Wahl.

[6] Nokias Einsatz von Morsecodes als Klingelzeichen endet an dieser Stelle nicht: Der „Ascending“ genannte Ton erzeugt ▬•▬•␣▬▬▬␣▬•␣▬•␣•␣▬•▬•␣▬␣••␣▬•␣▬▬•␣•▬▬•␣•␣▬▬▬␣•▬▬•␣ •▬••␣•␣ und morst damit den Slogan der Firma.

Betrachten Sie den abgebildeten NFA A über dem Eingabealphabet $\Sigma = \{a, b, c\}$.
Geben Sie einen DFA A' in graphischer Darstellung an, mit $L(A') = L(A)$. Wandeln sie dazu den NFA A mit Hilfe der Potenzmengenkonstruktion in einen DFA A' um. Berücksichtigen Sie dabei nur solche Zustände von A', die vom Startzustand $q_0' := \{q_0\}$ aus erreicht werden können.



Aufgabe 7.22. Sei A der abgebildete endliche Automat über dem Alphabet $\Sigma = \{a, b, c\}$.



(a) Geben Sie Folgendes für A an:
  (i) die Menge der Zustände,
  (ii) den Startzustand,
  (iii) die Menge der Endzustände und
  (iv) die Übergangsfunktion.

(b) Ist A ein deterministischer Automat? Ist A ein nichtdeterministischer Automat?

(c) Welche der folgenden Worte werden von A akzeptiert, welche nicht? Begründen Sie Ihre Antworten.

$w_1 = q_1aa$ $\quad$ $w_2 = bbbbaa$ $\quad$ $w_3 = abaabbbb$
$w_4 = aababbaa$ $\quad$ $w_5 = aaaabb$ $\quad$ $w_6 = bbabaa$

(d) Geben Sie eine mathematische oder umgangssprachliche Beschreibung der Sprache $L(A)$ an.

Aufgabe 7.23. Sei $n \in \mathbb{N}$. Betrachten Sie das Eingabealphabet $\Sigma := \{a, b\}$ und die Sprache

$$L_n := \{ww \,:\, w \in \Sigma^* \text{ und } |w| = n\}.$$

Beweisen Sie, dass jeder NFA, der L_n akzeptiert, mindestens 2^n Zustände besitzt.

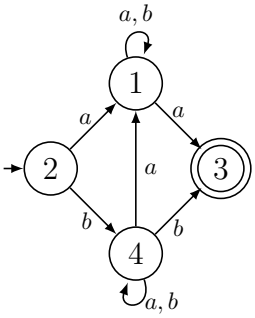
Aufgabe 7.24. Sei Σ ein Alphabet und seien $L_1, L_2 \subseteq \Sigma^*$ reguläre Sprachen. Zeigen Sie mithilfe von NFAs:

(a) Die Vereinigung $L_v := L_1 \cup L_2$ ist regulär.

(b) Die Konkatenation $L_k := L_1 \cdot L_2$ ist regulär.

Aufgabe 7.25.

Sei N der rechts abgebildete NFA über dem Alphabet $\Sigma := \{a, b\}$.



(i) Welche der folgenden Worte liegen in $L(N)$, welche nicht?

$$w_1 = aaa \qquad w_2 = b \qquad w_3 = babb \qquad w_4 = bba$$

(ii) Konstruieren Sie mittels Potenzmengenkonstruktion einen DFA D, der dieselbe Sprache wie N akzeptiert. Berücksichtigen Sie in D nur Zustände, die vom Startzustand von D aus erreichbar sind.

(iii) Geben Sie einen regulären Ausdruck für $L(N)$ an.

Aufgabe 7.26. Sei $\Sigma := \{a\}$. Zeigen oder widerlegen Sie: Für jeden NFA $N := (\Sigma, Q, \delta, q_0, F)$ akzeptiert der NFA $N' := (\Sigma, Q, \delta, q_0, Q \backslash F)$ die Sprache $\Sigma^* \backslash L(N)$.

Aufgabe 7.27. Sei $n \in \mathbb{N}$ und $\Sigma = \{a, b\}$. Betrachte die Sprache $L_n := \{xy : x, y \in \Sigma^n, x \neq y\}$.

(a) Zeigen Sie: Jeder DFA für L_n benötigt mindestens 2^n Zustände.

Hineweis: Zeigen Sie $\text{Index}(L_n) \geqslant 2^n$.

(b) Konstruieren Sie einen DFA D_n für die Sprache L_n mit möglichst wenigen Zuständen.

(c) Konstruieren Sie einen NFA N_n für die Sprache L_n mit möglichst wenigen Zuständen.

Aufgabe 7.28. Welche der Sprachen L_1, L_2 und L_3 sind regulär, welche nicht? Beweisen Sie Ihre Antwort jeweils mithilfe der Index-Methode, des Pumping-Lemmas oder durch die Angabe eines endlichen Automaten.

(a) $L_1 = \{b^n a^{(2n)} : n \in \mathbb{N}\}$

(b) $L_2 = \{a^{(2n)} : n \in \mathbb{N}\}$

(c) $L_3 = \{a^{(2^n)} : n \in \mathbb{N}\}$

(d) $L_4 := \{a^i a a^i : i \in \mathbb{N}\}$

(e) $L_5 := \{a^i b a^i : i \in \mathbb{N}\}$

(f) $L_6 := \{a a^{i \cdot i} : i \in \mathbb{N}\}$

(g) $L_8 := \{b a^i b a^i : i \in \mathbb{N}\}$

(h) $L_{10} := \{a^i a^j a^i : i, j \in \mathbb{N}\}$

Aufgabe 7.29. Für ein Alphabet Σ, einen Buchstaben $\sigma \in \Sigma$ und ein Wort $w \in \Sigma^*$ bezeichnet $|w|_\sigma$ die Anzahl der Vorkommen des Buchstabens σ im Wort w.
Zum Beispiel gilt $|aababa|_a = 4$, $|aababa|_b = 2$ und $|aababa|_y = 0$.

(a) Zeigen Sie mit Hilfes des Satzes von Myhill-Nerode, dass die folgenden Sprachen nicht regulär sind.

(i) $L_1 = \{\, a^n b^m c^{n+m} \;:\; n, m \in \mathbb{N} \,\}$,

(ii) $L_2 = \{a^{(n^2)} : n \in \mathbb{N}\,\}$.

(iii) $L_3 := \big\{ w \in \{a, b\}^* : |w|_a \geqslant |w|_b \big\}$ über $\Sigma = \{a, b\}$

(iv) $L_4 := \big\{ w w^{\text{reverse}} : w \in \{c, d\}^* \big\}$ über $\Sigma = \{c, d\}$

(v) $L_5 := \big\{ a b a^2 b a^3 b \cdots a^n b : n \in \mathbb{N}_{>0} \big\}$ über $\Sigma = \{a, b\}$

Hinweis: Finden Sie jeweils eine unendliche Menge von Wörtern, die paarweise nicht-äquivalent bzgl. der Nerode-Relation sind, und weisen Sie die Nicht-Äquivalenz durch Angabe geeigneter Zeugen nach.

(b) Sei $\Sigma := \{a, b\}$. Welche der folgenden Sprachen sind regulär, welche nicht? Beweisen Sie jeweils Ihre Antwort.

a) $L_6 := \left\{ a^n w : w \in \Sigma^*,\ |w|_a \geqslant n,\ n \in \mathbb{N} \right\}$

b) $L_7 := \left\{ a^n w : w \in \Sigma^*,\ |w|_b \leqslant n,\ n \in \mathbb{N} \right\}$

c) $L_8 := \{\ a^n w b^n : m, n \in \mathbb{N}, m \geqslant n, w \in \Sigma^m\ \}$.

Aufgabe 7.30. Finden Sie den Fehler im folgenden „Beweis“:

Behauptung: $L = \{w \in \{a, b\}^* : |w| \text{ ist gerade}\}$ ist nicht regulär.

Beweis mit dem Satz von Myhill-Nerode II:

Für jedes $n \in \mathbb{N}$ wähle den Vertreter a^n. Wir zeigen die Inäquivalenz $a^n \not\equiv_L a^m$ für beliebige $n, m \in \mathbb{N}$ mit $m = n+1$ durch den Zeugen b^n:

- $a^n b^n \in L$, da $|a^n b^n| = 2n$ gerade ist, aber
- $a^m b^n \notin L$, da $|a^m b^n| = m + n = 2n + 1$ ungerade ist.

Also sind die Wörter $\epsilon, a, aa, aaa, \ldots$ inäquivalent und Index(L) ist unendlich. □

Aufgabe 7.31. Sei $\Sigma := \{a, b\}$. Welche der folgenden Sprachen sind regulär, welche nicht? Beweisen Sie jeweils Ihre Antwort.

(a) $L_1 := \left\{ (ab)^n \cdot a \cdot (ab)^m :\ n, m \in \mathbb{N}, n \neq m \right\}$

(b) $L_2 := \left\{ (ab)^n \cdot a \cdot (ba)^m :\ n, m \in \mathbb{N}, n = m \right\}$

(c) $L_3 := \{w \in \Sigma^* : |w|_a > |w|_b\} \cup \{w \in \Sigma^* : |w|_a < |w|_b\}$

(d) $L_4 := \{w \in \Sigma^* : |w|_a \geqslant |w|_b\} \cap \{w \in \Sigma^* : |w|_a \leqslant |w|_b\}$

Aufgabe 7.32. Gegeben seien die folgenden regulären Ausdrücke über dem Alphabet $A = \{a, b\}$:

$$R_1 = a(ba)^*b \qquad R_2 = (a|b)(a|b)\left((a|b)(a|b)\right)^*(a|bb|\varepsilon) \qquad R_3 = (ab)(ab)^*(a|b)^*$$

(a) Gehören die folgenden Worte zur Sprache $L(R_1)$, $L(R_2)$ bzw. $L(R_3)$?

$$w_1 = bb \qquad w_2 = abab \qquad w_3 = abb \qquad w_4 = aba$$

(b) Geben Sie das kürzeste Wort w an, so dass $w \in L(R_3)$ und $w \notin L(R_2)$.

(c) Geben Sie einen NFA mit möglichst wenigen Zuständen in graphischer Darstellung an, der die Sprache $L(R_3)$ akzeptiert.

Aufgabe 7.33.

(a) Gegeben seien die folgenden regulären Ausdrücke:

$R_1 := a(\varepsilon|a|b)^*b$ $\qquad$ $R_2 := (a|b)\big((a|b)(a|b)\big)^*$ $\qquad$ $R_3 := (a^*b^*)^*$

(i) Welche der folgenden Wörter liegen in $L(R_1), L(R_2)$ bzw. $L(R_3)$, welche nicht?

$w_1 := \varepsilon$ $\qquad$ $w_2 := ab$ $\qquad$ $w_2 := aab$ $\qquad$ $w_3 := bb$

(ii) Geben Sie ein kürzestes Wort w an, sodass $w \in L(R_2)$ und $w \notin L(R_1)$ gilt.

(iii) Beschreiben Sie die Sprachen $L(R_1)$, $L(R_2)$ und $L(R_3)$ umgangssprachlich.

(iv) Bestimmen Sie einen DFA mit möglichst wenigen Zuständen, der die Sprache $L(R_3)$ akzeptiert.

(b) Geben Sie für die folgenden Sprachen je einen möglichst kurzen regulären Ausdruck an, der die Sprache beschreibt.

(i) $L_1 := \{\varepsilon, a, aa, aaa\}$

(ii) $L_2 := \{w \in \{a,b\}^* : |w|_b \geqslant 2\}$

(iii) $L_3 := \{w \in \{a,b\}^* : w$ enthält nicht das Teilwort $bb\}$

(iv) $L_4 := \big\{w \in \{a,b\}^* : w$ beginnt mit bb und $|w|_b$ ist durch 3 teilbar$\big\}$

(v) $L_5 := \big\{w \in \{a,b\}^* : w$ enthält nicht das Teilwort $aab\big\}$

Aufgabe 7.34. Geben Sie für die folgenden Sprachen je einen möglichst kurzen regulären Ausdruck an, der die Sprache beschreibt.

(a) $L_1 := \{w \in \{a,b\}^* : w$ beginnt mit b und hat höchstens die Länge fünf$\}$

(b) $L_2 := \{w \in \{a,b\}^* : w$ besteht nur a's oder nur aus b's, zusätzlich ist $|w|$ gerade$\}$

(c) $L_3 := \{w \in \{a,b\}^* :$ in w kommen keine drei a's hintereinander vor$\}$

(d) $L_4 := \{w = w_0w_1w_2\ldots \in \{a,b\}^* : w_i = b$, f.a. $i = 2n, n \in \mathbb{N}\}$

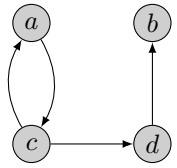
(e) $L_5 := \{w = w_0w_1w_2\ldots \in \{a,b\}^* :$ Falls $w_i = b$ so $w_{i+1} = a$ oder $w_{i+2} = a$, $i \in \mathbb{N}\}$

(f) $L_6 := \{w \in \{a,b,c\}^*:$ der erste und der letzte Buchstabe von w sind verschieden$\}$

(g) $L_7 := \left\{ \begin{array}{c} w \in \{a,b,c\}^*: w \neq \varepsilon \text{ und } w \text{ enthält beliebig viele Vorkommen der} \\ \text{Buchstaben } a,b,c, \text{ diese aber in alphabetischer Reihenfolge} \end{array} \right\}$

Aufgabe 7.35.

(a) Betrachten Sie den dargestellten gerichteten Graphen $G := (V, E)$.



Welche Kanten $(x, y) \in V \times V$ müssen zur Kantenrelation E mindestens hinzugefügt werden, um eine Kantenrelation zu erhalten, die jeweils

(i) reflexiv ist? (ii) symmetrisch ist? (iii) transitiv ist?

(b) Betrachten Sie die folgenden Relationen R_i über der jeweiligen Menge M_i.

(i) $M_1 := \{1,2,3,4,5\}$, $R_1 := \{(1,1),(1,2),(2,2),(2,3),(3,3),(4,4),(4,5),(5,4),(5,5)\}$

(ii) $M_2 := \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$, $R_2 := \{(\clubsuit,\spadesuit),(\clubsuit,\heartsuit),(\clubsuit,\diamondsuit),(\spadesuit,\heartsuit),(\spadesuit,\diamondsuit),(\heartsuit,\diamondsuit)\}$

(iii) $M_3 := \{-4,-3,-2,-1,0,1,2\}$, $R_3 := \{(x,y) \in M_3 \times M_3 : x \cdot y \leqslant 3\}$

(iv) $M_4 := \mathbb{N}_{>0}$, $R_4 := \{(a,b) \in M_4 \times M_4 : ggT(a,b) > 1\}$

Stellen Sie R_1 und R_2 durch einen gerichteten Graphen in graphischer Darstellung dar. Geben Sie für jedes $i \in \{1,2,3,4\}$ an, welche Eigenschaften (reflexiv, symmetrisch, transitiv) die Relation R_i jeweils besitzt.

Aufgabe 7.36.

(a) Betrachten Sie die Relation $R = \{(a,a),(b,c),(c,d),(d,c)\}$ über der Menge $A = \{a,b,c,d\}$. Welche Paare $(x,y) \in A \times A$ müssen zu R mindestens hinzugefügt werden, um Relationen R_r, R_s, R_t, R_e zu erhalten, sodass

(i) R_r reflexiv ist?

(ii) R_s symmetrisch ist?

(iii) R_t transitiv ist?

(iv) R_e eine Äquivalenzrelation ist? Geben Sie auch den Index und die Äquivalenzklassen von R_e an.

(b) Geben Sie für jede der folgenden Relationen R_i über der Menge A_i an, ob es sich um eine Äquivalenzrelation handelt.

(i) $A_1 = \mathcal{P}(\mathbb{N})$ und $R_1 = \{(a,b) \in A_1 \times A_1 : a \cap b = \emptyset\}$

(ii) $A_2 = \mathbb{N}$ und $R_2 = \{(a,b) \in A_2 \times A_2 : a \cap b \neq \emptyset\}$

(iii) $A_3 = \mathbb{Z}$ und $R_3 = \{(x,y) \in A_3 \times A_3 : x - y$ ist durch 3 teilbar $\}$.

Aufgabe 7.37. Zeigen Sie, dass die im Folgenden angegebenen Relationen R_i jeweils Äquivalenzrelationen sind. Geben Sie jeweils jede Äquivalenzklasse von R_i sowie jeweils den Index von R_i an. Geben Sie für jede Äquivalenzklasse von R_i jeweils einen Vertreter an.

(a) Sei $G = (V,E)$ ein gerichteter Graph. Wir definieren $R_1 = \{(u,v) \in V^2 : u$ und v liegen in derselben starken Zusamm als Relation über V.

(b) Sei $A := \{a,b\}$ ein Alphabet und sei A^n die Menge aller Worte der Länge n, die sich mit Buchstaben aus A bilden lassen. Betrachten Sie

$$R_2 := \{(w_1,w_2) \in A^n \times A^n : w_2 \text{ entsteht aus } w_1 \text{ durch Umsortierung der Buchstaben}\}$$

(c) Wir definieren $\mathrm{AL}_{|V_0} := \{\varphi \in \mathrm{AL} : \mathrm{Var}(\varphi) = \{V_0\}\}$ als die Menge aller syntaktisch korrekten aussagenlogischen Formeln, die außer V_0 keine weiteren Aussgenvariablen enthalten. Dann sei

$$R_3 := \{(\varphi,\psi) \in \mathrm{AL}_{|V_0} \times \mathrm{AL}_{|V_0} : \varphi \equiv \psi\}.$$

Aufgabe 7.38. Sei $\mathfrak{G}_3$ die Menge aller Graphen $G = (V, E)$ mit $V \subseteq \mathbb{N}$ und $|V| = 3$, und es gelte für zwei Graphen $G_1, G_2 \in \mathfrak{G}_3$, dass $G_1 \cong G_2$ genau dann, wenn G_1 isomorph zu G_2 ist.

(a) Zeigen Sie, dass die Relation $\cong$ eine Äquivalenzrelation über $\mathfrak{G}_3$ ist.

(b) Geben Sie für jede Äquivalenzklasse einen Vertreter in graphischer Darstellung an.

(c) Wie groß ist der Index von $\cong$ auf $\mathfrak{G}_3$?

(d) Welche Kardinalität besitzen die Äquivalenzklassen?

Aufgabe 7.39. Für jedes $n \in \mathbb{N}_{>0}$ sei die Relation $\equiv_n$ (*„Kongruenz modulo n“*) über $\mathbb{N}$ definiert indem $x \equiv_n y$ genau dann gelte, wenn Zahlen $i, j, k \in \mathbb{N}$ existieren, so dass $x = in + k$ und $y = jn + k$. (Man erhält also beim Teilen von x durch n den gleichen Rest k wie beim Teilen von y durch n.)

(a) Zeigen Sie, dass die Relation $\equiv_n$ für jedes $n \in \mathbb{N}$ eine Äquivalenzrelation über $\mathbb{N}$ ist.

(b) Geben Sie den Index von $\equiv_n$ in Abhängigkeit von n an.

(c) Geben Sie alle Äquivalenzklassen von $\equiv_5$ an. Geben Sie für jede dieser Äquivalenzklassen zwei unterschiedliche Vertreter an.

8. Kontextfreie Grammatiken und rekursiv definierte Strukturen

In diesem Kapitel lernen wir einen Kalkül kennen, mit dem man strukturelle Eigenschaften von rekursiv definierten Systemen beschreiben kann: **Kontextfreie Grammatiken**.

Kontextfreie Grammatiken (kurz: KFGs) eignen sich besonders gut zur Modellierung von beliebig tief geschachtelten baumartigen Strukturen. KFGs können gleichzeitig

- hierarchische Baumstrukturen und
- Sprachen und deren textuelle Notation

spezifizieren. KFGs werden z.B. angewendet zur Definition von:

- Programmen einer Programmiersprache und deren Struktur, z.B. Java, C, Pascal (KFGs spielen z.B. beim „Compilerbau“ eine wichtige Rolle).
- Datenaustauschformaten, d.h. Sprachen als Schnittstelle zwischen Software-Werkzeugen, z.B. HTML, XML.
- Bäumen zur Repräsentation strukturierter Daten, z.B. XML.
- Strukturen von Protokollen beim Austausch von Nachrichten zwischen Prozessen oder Geräten.

Wir stellen nur die Grundbegriffe und einige Beispiele vor, im Detail werden KFGs in der Veranstaltung „GL-2: Theoretische Informatik 2“ behandelt.

8.1. Was ist eine kontextfreie Grammatik?

Es gibt zwei Sichtweisen auf KFGs:

(1) Eine KFG ist ein spezielles **Ersetzungssystem**. Seine Regeln geben an, auf welche Art man ein Symbol durch eine Folge von Symbolen ersetzen kann. Auf diese Weise definiert eine KFG eine **Sprache**, d.h. eine **Menge von Worten** über einem bestimmten Alphabet, die mit dem durch die KFG gegebenen Regeln erzeugt werden können.

(2) Gleichzeitig definiert eine KFG eine **Menge von Baumstrukturen**, die sich durch schrittweises Anwenden der Regeln erzeugen lassen.

Für die Modellierung rekursiv definierter Strukturen ist die zweite Sichtweise besonderes interessant. Aber es ist oft sehr nützlich, dass derselbe Kalkül auch gleichzeitig eine textuelle Notation für die Baumstrukturen liefern kann und dass Eigenschaften der zugehörigen Sprache untersucht werden können.

Definition 8.1 (KFG). Eine kontextfreie Grammatik $G = (\Sigma, V, S, P)$ besteht aus

Terminalsymbole
Terminale

- einer endlichen Menge Σ, der so genannten Menge der **Terminalsymbole**
  (die Elemente aus Σ werden auch **Terminale** genannt).

Nichtterminalsymbole
Variablen
Nichtterminale

- einer endlichen Menge V, der so genannten Menge der **Nichtterminalsymbole** (oder **Variablen**)
  (die Elemente aus V werden auch **Nichtterminale** genannt).

  Die Mengen Σ und V sind disjunkt, d.h. $\Sigma \cap V = \emptyset$.

Vokabular
Symbole

  Die Menge $W := \Sigma \cup V$ heißt **Vokabular** (die Elemente in W nennt man auch **Symbole**).

Startsymbol

- einem Symbol $S \in V$, dem so genannten **Startsymbol**.

Produktionen

- einer endlichen Menge $P \subseteq V \times W^*$, der so genannten Menge der **Produktionen**.
  Für eine Produktion $(A, x) \in P$ schreiben wir meistens $A \to x$.

In der Literatur und in den Übungsaufgaben in Abschnitt 8.7 werden an Stelle der Buchstaben Σ und V oft auch die Buchstaben T und N verwendet, um die Menge der Terminalsymbole bzw. der Nichtterminalsymbole zu bezeichnen.

Beispiel 8.2. Als erstes Beispiel betrachten wir eine KFG, die arithmetische Ausdrücke erzeugt, die über den Zahlen $1, 2, 3$ gebildet sind und die Operatoren $+, -, \cdot$ sowie Klammern $(,)$ benutzt. Ein Beispiel für einen zulässigen solchen arithmetischen Ausdruck ist $(1 + 3) \cdot (2 + 2 + 3) - 1$.
Wir betrachten die KFG $G_{AA} := (\Sigma, V, S, P)$ mit

- Terminalalphabet $\Sigma := \{1, 2, 3, +, -, \cdot, (,)\}$
- Nichtterminalalphabet $V := \{\mathit{Ausdruck}, \mathit{Operator}\}$
- Startsymbol $S := \mathit{Ausdruck}$
- Produktionsmenge

$$
P := \Big\{ \begin{array}{lcl}
\mathit{Ausdruck} & \to & 1\ , \\
\mathit{Ausdruck} & \to & 2\ , \\
\mathit{Ausdruck} & \to & 3\ , \\
\mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ , \\
\mathit{Ausdruck} & \to & (\ \mathit{Ausdruck}\ )\ , \\
\mathit{Operator} & \to & +\ , \\
\mathit{Operator} & \to & -\ , \\
\mathit{Operator} & \to & \cdot
\end{array} \Big\}.
$$

Um Schreibarbeit zu sparen, werden wir bei der Angabe der Produktionsmenge einer KFG oft einige Zeilen, die das gleiche Nichtterminal auf der linken Seite des Pfeils aufweisen, zu einer einzigen Zeile zusammenfassen, bei der die Möglichkeiten der rechten Seite des Pfeils durch das „oder“-Zeichen | getrennt sind.
Damit können wir die Produktionsmenge P auch kurz wie folgt beschreiben:

$$
P = \Big\{ \begin{array}{lcl}
\mathit{Ausdruck} & \to & 1 \mid 2 \mid 3\ , \\
\mathit{Ausdruck} & \to & \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \mid (\ \mathit{Ausdruck}\ )\ , \\
\mathit{Operator} & \to & + \mid - \mid \cdot
\end{array} \Big\}.
$$

8.2. Produktionen: Die Semantik von KFGs

Jede Produktion einer KFG, etwa die Produktion *Ausdruck* $\rightarrow$ *Ausdruck Operator Ausdruck*, kann man auffassen als:

- eine *Strukturregel*, die besagt „Ein *Ausdruck* besteht aus einem *Ausdruck*, gefolgt von einem *Operator*, gefolgt von einem *Ausdruck* — oder als
- eine *Ersetzungsregel*, die besagt „Das Symbol *Ausdruck* kann man durch das Wort *Ausdruck Operator Ausdruck* ersetzen."

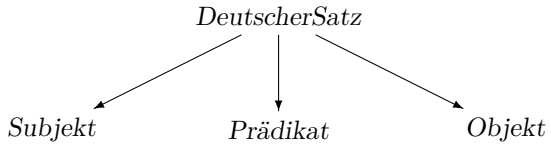
Beispielsweise kann man die Produktion

$$\textit{DeutscherSatz} \quad \rightarrow \quad \textit{Subjekt Prädikat Objekt}$$

verstehen als Aussage, die besagt:

> „Ein deutscher Satz ist aufgebaut aus Subjekt Prädikat Objekt."

Graphische Darstellung:



Das Grundkonzept für die Anwendung von Produktionen einer KFG ist die „Ableitung":

Definition 8.3 (Ableitung). Sei $G = (\Sigma, V, S, P)$ eine KFG.

- Falls $A \rightarrow x$ eine Produktion in P ist und $u \in W^*$ und $v \in W^*$ beliebige Worte über dem Vokabular $W = \Sigma \cup V$ sind, so schreiben wir

  $$uAv \implies_G uxv \quad \text{(bzw. kurz:} \quad uAv \implies uxv\text{)}$$

  und sagen, dass uAv in einem **Ableitungsschritt** zu uxv umgeformt werden kann. Ableitungsschritt
- Eine **Ableitung** ist eine endliche Folge von hintereinander angewendeten Ableitungsschritten. Für Worte $w \in W^*$ und $w' \in W^*$ schreiben wir Ableitung

  $$w \implies^*_G w' \quad \text{(bzw. kurz:} \quad w \implies^* w'\text{)},$$

  um auszusagen, dass es eine endliche Folge von Ableitungsschritten gibt, die w zu w' umformt.

  Spezialfall:
  Diese Folge darf auch aus 0 Ableitungsschritten bestehen, d.h. f.a. $w \in W^*$ gilt: $w \implies^* w$.

Beispiel 8.4. Sei G_{AA} die Grammatik für arithmetische Ausdrücke aus Beispiel 8.2.
Beispiele für einzelne Ableitungsschritte:

- (*Ausdruck*) $\implies$ (*Ausdruck Operator Ausdruck*)

- (*Ausdruck Operator Ausdruck*) $\Longrightarrow$ (*Ausdruck* + *Ausdruck*)

Ein Beispiel für eine Ableitung in G_{AA}:

$$\begin{aligned}
\textit{Ausdruck} &\Longrightarrow \textit{Ausdruck Operator Ausdruck}\\
&\Longrightarrow (\ \textit{Ausdruck}\ )\ \textit{Operator Ausdruck}\\
&\Longrightarrow (\ \textit{Ausdruck Operator Ausdruck}\ )\ \textit{Operator Ausdruck}\\
&\Longrightarrow (\ \textit{Ausdruck}\ +\ \textit{Ausdruck}\ )\ \textit{Operator Ausdruck}\\
&\Longrightarrow (\ \textit{Ausdruck}\ +\ \textit{Ausdruck}\ )\ \cdot\ \textit{Ausdruck}\\
&\Longrightarrow (\ 1\ +\ \textit{Ausdruck}\ )\ \cdot\ \textit{Ausdruck}\\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ \textit{Ausdruck}\\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ 2\ ,
\end{aligned}$$

In jedem Schritt wird jeweils eine Produktion auf ein Nichtterminal der vorangehenden Symbolfolge angewandt. Die obige Kette von Ableitungsschritten zeigt die Ableitung

$$\textit{Ausdruck} \Longrightarrow^*_{G_{AA}} (\ 1\ +\ 3\ )\ \cdot\ 2.$$

8.2.1. Kontextfreie Sprachen

Definition 8.5 (Grammatiken und Sprachen). Sei Σ ein Alphabet.

Sprache einer KFG, $L(G)$

(a) Sei $G = (\Sigma, V, S, P)$ eine KFG. Die **von G erzeugte Sprache $L(G)$** ist die Menge aller Worte über dem Terminalalphabet Σ, die aus dem Startsymbol S abgeleitet werden können. D.h.:

$$L(G) := \{w \in \Sigma^* : S \Longrightarrow^*_G w\}.$$

kontextfreie Sprache

(b) Wir nennen $L \subseteq L^*$ eine **kontextfreie Sprache**, wenn es eine KFG G mit $L = L(G)$ gibt.

Man beachte, dass $L(G) \subseteq \Sigma^*$ ist. Daher kommen in Worten aus $L(G)$ keine Nichtterminale vor, sondern nur die aus dem Startsymbol S ableitbaren Worte über dem Terminalalphabet Σ!

Beispiel 8.6. Die KFG G_{AA} aus Beispiel 8.2 definiert die Sprache $L(G_{AA})$, die aus allen über den Zahlen $1, 2, 3$, den Operatoren $+, -, \cdot$ und den Klammersymbolen $(,)$ korrekt geformten arithmetischen Ausdrücken besteht. Beispielsweise gehören folgende Worte zu $L(G_{AA})$:

$$3, \quad (3+1), \quad 1+2\cdot 3, \quad (3+1)\cdot(2+2+3)-1, \quad 2\cdot((3+1)\cdot(2+2+3)-1), \quad ((3+1)).$$

Aber die Worte

$$(), \quad 4, \quad (3+1, \quad (\ \textit{Ausdruck Operator Ausdruck}\ )$$

gehören **nicht** zu $L(G_{AA})$. Beachte, dass $L(G_{AA})$ eine kontextfreie Sprache ist.

Beispiel 8.7. Die Sprache $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ wird von der KFG $G_1 = (\Sigma, V, S, P)$ mit $\Sigma = \{a, b\}$, $V = \{S\}$ und $P = \{\ S \to aSb \mid \varepsilon\ \}$ erzeugt, denn $L_1 = L(G_1)$. Also ist die Sprache L_1 kontextfrei aber nicht regulär. (Zur Erinnerung: In Satz 7.49 haben wir gezeigt, dass L_1 keine reguläre Sprache ist.)

Notation: $a^n b^n$ ist eine Abkürzung für $\underbrace{aa\cdots a}_{n \text{ mal}}\underbrace{bb\cdots b}_{n \text{ mal}}$.

Beispiel 8.8 (Wohlgeformte Klammerausdrücke). Betrachte die Sprache D aller wohlgeformten Klammerausdrücke über $\Sigma = \{ (,) \}$. Wohlgeformte Klammerausdrücke sind

- (), ((())), (())()()((())), (()())

Nicht wohlgeformt sind

- (()(), ((())

Es ist $D = L(G)$ für die kontextfreie Grammatik $G = (\Sigma, \{S\}, S, P)$ mit den Produktionen

$$S \to SS \mid (S) \mid \epsilon.$$

Also ist auch D eine kontextfreie Sprache.

Frage: Ist jede reguläre Sprache auch kontextfrei?

Um diese Frage zu beantworten, betrachten wir einen beliebigen NFA

$$A = (\Sigma, Q, \delta, q_0, F).$$

Wir konstruieren eine kontextfreie Grammatik

$$G = (\Sigma, V, S, P)$$

mit $L(A) = L(G)$ und haben damit nachgewiesen, dass die reguläre Sprache $L(A)$ kontextfrei ist. Die Grammatik G simuliert die Arbeitsweise des NFAs Schritt für Schritt. Wir beschreiben die einzelnen Komponenten von G.

- $V := Q$,
- $S := q_0$ und
- $P = \{q \to aq' \ : \ q, q' \in Q, a \in \Sigma \text{ und } \delta(q, a) = q' \} \cup \{q \to \varepsilon \ : \ q \in F \}$.

Die Grammatik „äfft“ A tatsächlich nach: Wenn A den Buchstaben a im Zustand q liest und dann in den Zustand q' wechselt, dann darf G die Variable q durch aq' ersetzen; mit anderen Worten, G spaltet das Nichtterminal a ab und wechselt in den Zustand q'. Wir zeigen, dass $L(A) = L(G)$ gilt.

$L(A) \subseteq L(G)$: Angenommen $w \in L(A)$. Dann gibt es eine akzeptierende Berechnung mit der Zustandsfolge $(q_0, q_1, \ldots, q_{|w|})$ für $q_{|w|} \in F$ und den Übergängen $q_{i+1} = \delta(q_i, w_{i+1})$. Dann zeigt man mit vollständiger Induktion nach der Länge $|w|$ von w, dass es die Ableitung

$$q_0 \Longrightarrow w_1 q_1 \Longrightarrow \cdots \Longrightarrow w_1 \cdots w_i q_i \Longrightarrow w_1 \cdots w_i w_{i+1} q_{i+1} \Longrightarrow \cdots \Longrightarrow w q_{|w|} \tag{8.1}$$

gibt. Da die Zustandsfolge eine akzeptierende Berechnung von A beschreibt, ist der Zustand $q_{|w|}$ akzeptierend und wir erhalten

$$q_0 \Longrightarrow_G^* w q_{|w|} \Longrightarrow w.$$

$L(G) \subseteq L(A)$: Das Argument ist analog zur ersten Inklusion, nur beginnen wir diesmal mit einer Ableitung $q_0 \Longrightarrow_G^* w$ der Form (8.11), von der wir dann die Zustandsfolge einer akzeptierenden Berechnung von A mit vollständiger Induktion ableiten können.

Nicht nur ist die Grammatik G kontextfrei, sondern ihre Produktionen sind auch von eingeschränkter Struktur: G ist eine rechtsreguläre Grammatik.

Definition 8.9. Sei $G = (\Sigma, V, S, P)$ eine kontextfreie Grammatik.

rechtsregulär
linksregulär

(a) Dann heißt G **rechtsregulär** (bzw. **linksregulär**), wenn alle Produktionen $A \to w$ von der Form

$$w \in \Sigma \cdot \big(V \cup \{\varepsilon\}\big) \quad (\text{bzw. } w \in V \cdot \big(\Sigma \cup \{\varepsilon\}\big)$$

sind.

(b) G wird eine reguläre Grammatik genannt, wenn G rechts- oder linksregulär ist.

Satz 8.10. Für jeden NFA A gibt es eine rechtsreguläre Grammatik G_R und eine linksreguläre Grammatik G_L mit $L(G_L) = L(A) = L(G_R)$.

Beweis: Die rechtsreguläre Grammatik G_R haben wir gerade konstruiert. Wir erhalten die linksreguläre Grammatik G_L, wenn wir für jeden Übergang $q' = \delta(q, a)$ die Produktion $q \Longrightarrow q'a$ aufnehmen. □

Die Klasse der kontextfreien Sprachen ist also eine echte Obermenge der Klasse der regulären Sprachen.

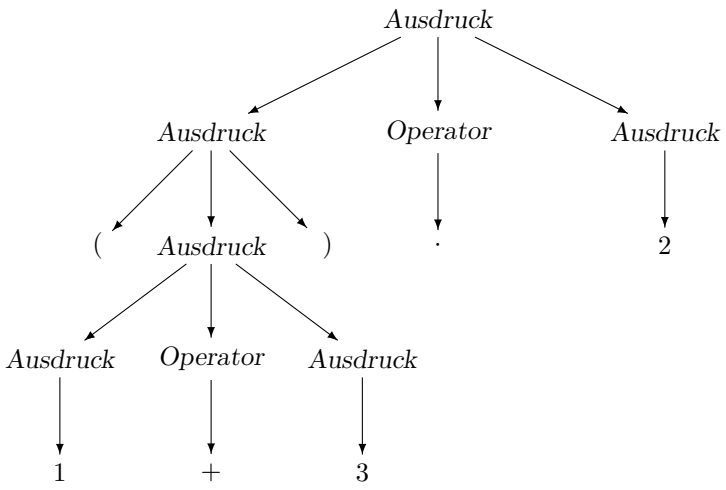
Folgerung 8.11. Jede reguläre Sprache ist auch kontextfrei. Die Sprache $\{\, a^n b^n \;:\; n \in \mathbb{N} \,\}$ ist kontextfrei, aber nicht regulär.

8.2.2. Ableitungsbäume

Beispiel 8.12. Sei G_{AA} die Grammatik für arithmetische Ausdrücke aus Beispiel 8.2.
Die Ableitung

$$\begin{aligned}
\mathit{Ausdruck} &\Longrightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow (\ \mathit{Ausdruck}\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\
&\Longrightarrow (\ 1\ +\ \mathit{Ausdruck}\ )\ \cdot\ \mathit{Ausdruck} \\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ \mathit{Ausdruck} \\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ 2\ ,
\end{aligned}$$

wird durch den folgenden „Ableitungsbaum“ dargestellt:



Wir führen den Begriff eines Ableitungsbaums formal ein. Sei $G = (\Sigma, V, S, P)$ eine KFG. Jede Ableitung $S \Longrightarrow_G^* w$ lässt sich als gerichteter Baum darstellen, bei dem

- jeder Knoten mit einem Symbol aus $\Sigma \cup V \cup \{\varepsilon\}$ markiert ist und
- die Kinder jedes Knotens eine festgelegte Reihenfolge haben. In der Zeichnung eines Ableitungsbaums werden von links nach rechts zunächst das „erste Kind“ dargestellt, dann das zweite, dritte etc.

Die Wurzel des Baums ist mit dem Startsymbol S markiert. Jeder Knoten mit seinen Kindern repräsentiert die Anwendung einer Produktion aus P:

- Die Anwendung einer Produktion der Form $A \to x$ mit $A \in V$ und $x \in W^+$ wird im Ableitungsbaum repräsentiert durch einen Knoten, der mit dem Symbol A markiert ist und der $|x|$ viele Kinder hat, so dass das i-te Kind mit dem i-ten Symbol von x markiert ist (f.a. $i \in \{1, \ldots, |x|\}$).
- Die Anwendung einer Produktion der Form $A \to \varepsilon$ mit $A \in V$ wird im Ableitungsbaum repräsentiert durch einen Knoten, der mit dem Symbol A markiert ist und der genau ein Kind hat, das mit ε markiert ist.

Beachte: Ein Ableitungsbaum kann mehrere Ableitungen repräsentieren. Beispielsweise repräsentiert der Ableitungsbaum aus Beispiel 8.12 auch die Ableitung

$$
\begin{aligned}
\mathit{Ausdruck} &\Longrightarrow \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck} \\
&\Longrightarrow \mathit{Ausdruck}\ \mathit{Operator}\ 2 \\
&\Longrightarrow \mathit{Ausdruck}\ \cdot\ 2 \\
&\Longrightarrow (\ \mathit{Ausdruck}\ )\ \cdot\ 2 \\
&\Longrightarrow (\ \mathit{Ausdruck}\ \mathit{Operator}\ \mathit{Ausdruck}\ )\ \cdot\ 2 \\
&\Longrightarrow (\ \mathit{Ausdruck}\ \mathit{Operator}\ 3\ )\ \cdot\ 2 \\
&\Longrightarrow (\ \mathit{Ausdruck}\ +\ 3\ )\ \cdot\ 2 \\
&\Longrightarrow (\ 1\ +\ 3\ )\ \cdot\ 2
\end{aligned}
$$

in der gegenüber der ursprünglichen Ableitung aus Beispiel 8.12 einige Ableitungsschritte vertauscht sind. Im Ableitungsbaum wird von der konkreten Reihenfolge, in der die einzelnen Ableitungsschritte vorkommen, abstrahiert.

8.3. Beispiele

Im Folgenden betrachten wir einige weitere Beispiele für kontextfreie Grammatiken.

8.3.1. Die Syntax von Programmiersprachen

Wir beginnen mit einer der wichtigsten Anwendungen kontextfreier Grammatiken, nämlich in der Definition der Syntax von Programmiersprachen.

Wir beschreiben einen allerdings sehr kleinen Ausschnitt von Pascal durch eine kontextfreie Grammatik $G = (\Sigma, V, S, P)$. Dazu benutzen wir das Alphabet

$$\Sigma = \{\texttt{a}, \ldots, \texttt{z}, ;, :=, \texttt{begin}, \texttt{end}, \texttt{while}, \texttt{do}\},$$

die Variablenmenge

$$\begin{aligned} V \quad = \quad & \{S, \text{ statements, statement, assignment-statement, while-statement,} \\ & \text{variable, boolean, expression } \} \end{aligned}$$

und das Startsymbol S. Die folgenden Produktionen werden benutzt, wobei wir die Variablen „variable, boolean, expression“ im Folgenden nicht weiter ersetzen:

$$\begin{aligned} S &\to \texttt{begin}\ \text{statements}\ \texttt{end} \\ \text{statements} &\to \text{statement} \mid \text{statement ; statements} \\ \text{statement} &\to \text{assign-statement} \mid \text{while-statement} \\ \text{assign-statement} &\to \text{variable} := \text{expression} \\ \text{while-statement} &\to \texttt{while}\ \text{boolean}\ \texttt{do}\ \text{statements} \end{aligned}$$

Frage: Lassen sich die syntaktisch korrekten Pascal-Programme durch eine kontextfreie Sprache definieren?

1. Antwort: Nein. In Pascal muss zum Beispiel sichergestellt werden, dass Anzahl und Typen der formalen und aktuellen Parameter übereinstimmen. Die Sprache

$$\{ww \ : \ w \in \Sigma^*\}$$

kann aber als nicht kontextfrei nachgewiesen werden.

2. Antwort: Im Wesentlichen ja, wenn man „Details“ wie Typ-Deklarationen und Typ-Überprüfungen ausklammert: Man beschreibt die Syntax durch eine kontextfreie Grammatik, die alle syntaktisch korrekten Programme erzeugt und stellt mit anderen Methoden sicher, dass Typen korrekt behandelt werden.

Beispiel 8.13 (Die Backus-Naur-Form). In der Syntax-Definition von Algol, Java[1] oder Pascal wird die Backus-Naur-Form (BNF), ein „Dialekt“ der kontextfreien Grammatiken, benutzt.

[1] Vergleiche http://docs.oracle.com/javase/specs/jls/se8/html/index.html, zuletzt besucht am 29.01.2015

Zum Beispiel werden in der BNF Nichtterminale von spitzen Klammern eingeschlossen und der Pfeil in einer Produktion wird durch das Symbol ::= ersetzt. Produktionen haben also die Form

$$< \text{Nichtterminal} > ::= \ w$$

für ein Wort w über dem Vokabular $W = \Sigma \cup V$. Mittlerweile gibt es viele Modifikation der BNF.

8.3.2. Aussagenlogische Formeln

Wir konstruieren eine KFG

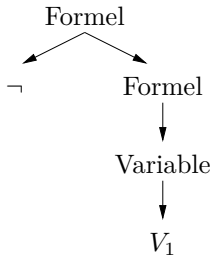
$$G_{\mathrm{AL}} \ = \ (\Sigma, V, S, P),$$

deren Sprache $L(G_{\mathrm{AL}})$ gerade die Menge aller aussagenlogischen Formeln ist, in denen nur Variablen aus $\{V_0, V_1, V_2\}$ vorkommen:

- Terminalsymbole $\Sigma := \{ V_0,\ V_1,\ V_2,\ \mathbf{0},\ \mathbf{1},\ \neg,\ \wedge,\ \vee,\ \rightarrow,\ \leftrightarrow,\ (,\ ) \}$
- Nichtterminalsymbole $V := \{ \mathit{Formel},\ \mathit{Variable},\ \mathit{Junktor} \}$
- Startsymbol $S := \mathit{Formel}$
- Produktionsmenge $P :=$

$$\begin{array}{rcl} \mathit{Formel} & \rightarrow & \mathbf{0} \mid \mathbf{1} \mid \mathit{Variable}\ , \\ \mathit{Formel} & \rightarrow & \neg\, \mathit{Formel} \mid (\ \mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}\ )\ , \\ \mathit{Variable} & \rightarrow & V_0 \mid V_1 \mid V_2\ , \\ \mathit{Junktor} & \rightarrow & \wedge \mid \vee \mid \rightarrow \mid \leftrightarrow \end{array} \Big\}.$$

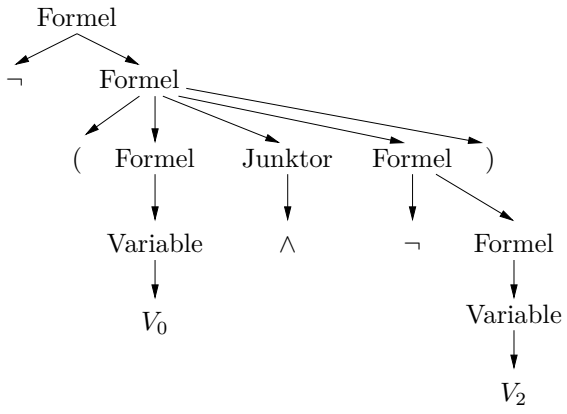
Beispiele für Ableitungsbäume:



Dieser Ableitungsbaum repräsentiert die Ableitung

$$\begin{aligned} \mathit{Formel} &\Longrightarrow \neg\, \mathit{Formel} \\ &\Longrightarrow \neg\, \mathit{Variable} \\ &\Longrightarrow \neg V_1 \end{aligned}$$

Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{\mathrm{AL}})$ ist die Formel $\neg V_1$.



Dieser Ableitungsbaum repräsentiert die Ableitung

$$\begin{aligned} \mathit{Formel} &\Longrightarrow \neg\, \mathit{Formel} \\ &\Longrightarrow \neg\, (\mathit{Formel}\ \mathit{Junktor}\ \mathit{Formel}) \\ &\Longrightarrow \neg\, (\mathit{Variable}\ \mathit{Junktor}\ \mathit{Formel}) \\ &\Longrightarrow \neg\, (V_0\ \mathit{Junktor}\ \mathit{Formel}) \\ &\Longrightarrow \neg\, (V_0 \wedge \mathit{Formel}) \\ &\Longrightarrow \neg\, (V_0 \wedge \neg \mathit{Formel}) \\ &\Longrightarrow \neg\, (V_0 \wedge \neg \mathit{Variable}) \\ &\Longrightarrow \neg\, (V_0 \wedge \neg V_2). \end{aligned}$$

Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{\mathrm{AL}})$ ist die Formel $\neg\, (V_0 \wedge \neg V_2)$.

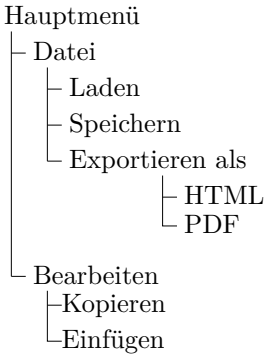
8.3.3. Reguläre Ausdrücke

Für jedes Alphabet Σ definieren die syntaktisch korrekt aufgebauten reguläre Ausdrücke über Σ eine kontextfreie Sprache. Siehe dazu Aufgabe 8.2.

8.3.4. Menü-Struktur in Benutzungsoberflächen

In der graphischen Benutzungsoberfläche von vielen Software-Systemen werden oftmals „Menüs“ verwendet. Ein Menü besteht aus einem Menünamen und einer Folge von Einträgen. Jeder einzelne Eintrag besteht dabei aus einem Operationsnamen oder selbst wieder einem Menü.

Beispiel:



Zur Spezifizierung der Grundstruktur solcher Menüs kann man folgende Grammatik $G_{\text{Menü}}$ verwenden:

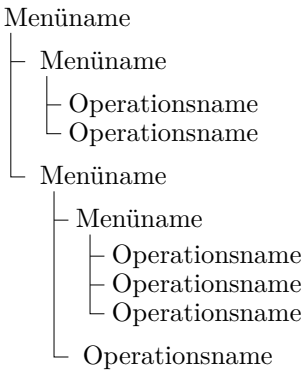
$$G_{\text{Menü}} = (\Sigma, V, S, P)$$

mit

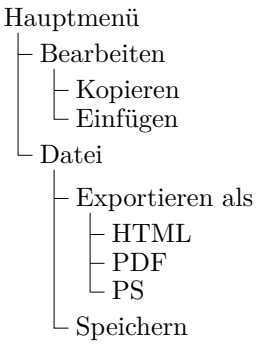
- $\Sigma := \{$ `Menüname`, `Operationsname` $\}$
- $V := \{$ *Menü*, *Eintragsfolge*, *Eintrag* $\}$
- $S :=$ *Menü*
- $P :=$

$$\left\{ \begin{array}{rcl} \textit{Menü} & \rightarrow & \texttt{Menüname}\ \textit{Eintragsfolge}\ , \\ \textit{Eintragsfolge} & \rightarrow & \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Eintragsfolge}\ , \\ \textit{Eintrag} & \rightarrow & \texttt{Operationsname} \mid \textit{Menü} \end{array} \right\}.$$

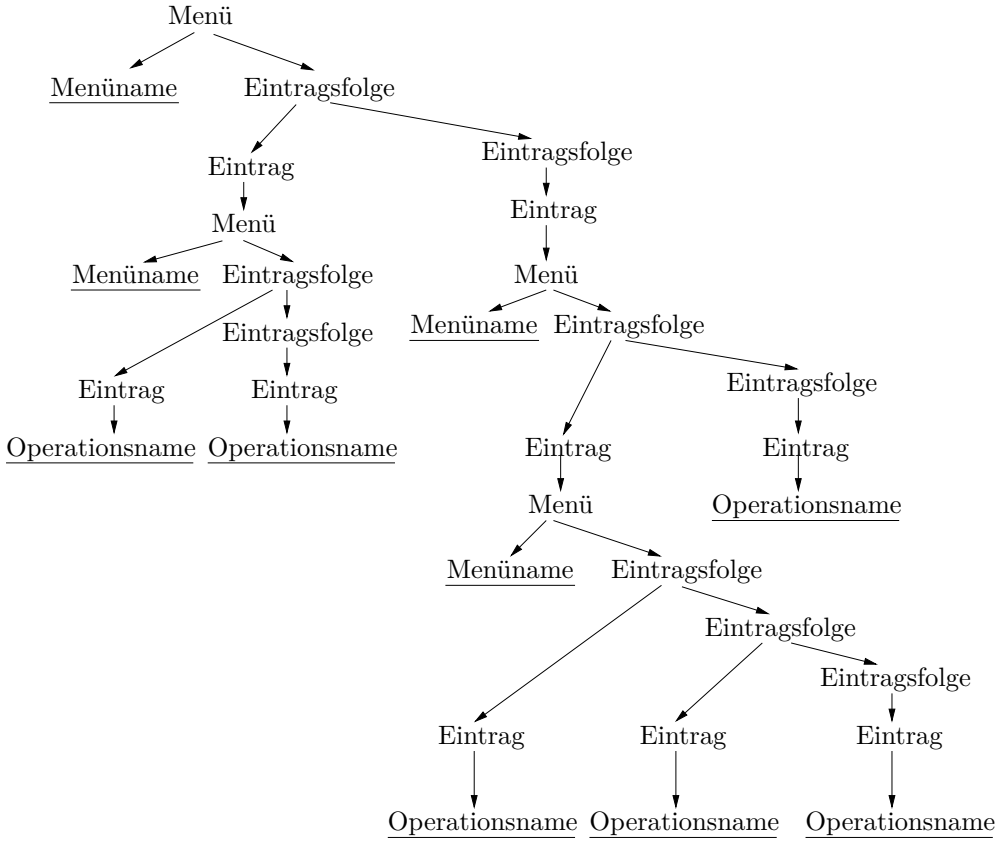
Jeder Ableitungsbaum repräsentiert die Struktur eines Menüs. Ein Menü der Struktur



Beispiel:



wird z.B. von folgendem Ableitungsbaum repräsentiert:



8.3.5. HTML-Tabellen

HTML (HyperText Markup Language) ist ein Format zur Beschreibung von verzweigten Dokumenten im Internet. Ein Bestandteil, der oft im Quell-Code von Internet-Seiten vorkommt, sind Tabellen. Z.B. wird der Eintrag

Tag	Zeit	Raum
Di	8:00-10:00	Hörsaal VI
Do	8:00-10:00	Hörsaal VI

durch HTML-Quelltext der folgenden Form erzeugt:

```
<table>
  <tr>
    <td> Tag </td>
    <td> Zeit </td>
    <td> Raum </td>
  </tr>
  <tr>
    <td> Di </td>
```

```
    <td> 8:00-10:00 </td>
    <td> Hörsaal VI </td>
  </tr>
  <tr>
    <td> Do </td>
    <td> 8:00-10:00 </td>
    <td> Hörsaal VI </td>
  </tr>
</table>
```

erzeugt.

Das Symbol `<table>` steht hier für den Anfang einer Tabelle, `</table>` steht für das Ende einer Tabelle. Die Symbole `<tr>` und `</tr>` stehen für den Anfang bzw. das Ende einer Zeile der Tabelle. Die Symbole `<td>` und `</td>` stehen für den Anfang bzw. das Ende eines Eintrags in einer Zelle der Tabelle. Als Einträge in einzelnen Zellen kann z.B. Text stehen oder eine weitere Tabelle.

Im Folgenden konstruieren wir eine Grammatik

$$G_{\text{HTML-Tabellen}} \;=\; (\Sigma, V, S, P),$$

so dass die von $G_{\text{HTML-Tabellen}}$ erzeugte Sprache aus (möglicherweise geschachtelten) HTML-Tabellen besteht:

- $\Sigma \;:=\; \{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`, a, …, z, A, …, Z, 0, 1, …, 9, :, -, ␣, ä, ö, ü, ß, Ä, Ö, Ü $\}$
- $V \;:=\; \{$ *Tabelle*, *Zeile*, *Eintrag*, *Text*, *Zeilen*, *Einträge* $\}$
- $S \;:=\;$ *Tabelle*
- $P \;:=\; \{$

$$\begin{aligned}
\textit{Tabelle} &\to \texttt{<table>}\ \textit{Zeilen}\ \texttt{</table>}\ , \\
\textit{Zeilen} &\to \textit{Zeile} \mid \textit{Zeile}\ \textit{Zeilen}\ , \\
\textit{Zeile} &\to \texttt{<tr>}\ \text{Einträge}\ \texttt{</tr>}\ , \\
\textit{Einträge} &\to \textit{Eintrag} \mid \textit{Eintrag}\ \textit{Einträge}\ , \\
\textit{Eintrag} &\to \texttt{<td>}\ \textit{Text}\ \texttt{</td>} \mid \texttt{<td>}\ \textit{Tabelle}\ \texttt{</td>}\ , \\
\textit{Text} &\to a \mid b \mid \cdots \mid z \mid A \mid B \mid \cdots \mid Z \mid \cdots \mid \text{Ü}\ , \\
\textit{Text} &\to a\ \textit{Text} \mid b\ \textit{Text} \mid \cdots \mid z\ \textit{Text} \mid A\ \textit{Text} \mid B\ \textit{Text} \mid \cdots \mid Z\ \textit{Text} \mid \cdots \mid \text{Ü}\ \textit{Text}
\end{aligned} \quad \Big\}.$$

Die oben angegebene Beispiel-HTML-Tabelle wird z.B. durch eine Ableitung erzeugt, die durch den Ableitungsbaum in Abbildung 8.1 repräsentiert wird.

8.4. Ausdruckskraft und Berechnungskomplexität

Welche Sprachen lassen sich leicht beschreiben, welche Sprachen erlauben nur komplexere Beschreibungen und welche Sprachen lassen sich überhaupt nicht beschreiben? Und wie sollen Beschreibungen überhaupt aussehen? Diese Fragen sind für die Spezifikation von Programmiersprachen von großer Wichtigkeit. Da sich Grammatiken in der Spezifikation von Programmiersprachen als sehr erfolgreich herausgestellt haben, wählen wir Grammatiken um Beschreibungen zu formulieren.

Wir beginnen mit der Chomsky-Hierarchie, in der wichtige Klassen von Grammatiken miteinander verglichen werden. Natürlich muss man mit einer Grammatik auch arbeiten können und eine Minimalvoraussetzung hierzu ist eine Lösung des Wortproblems.

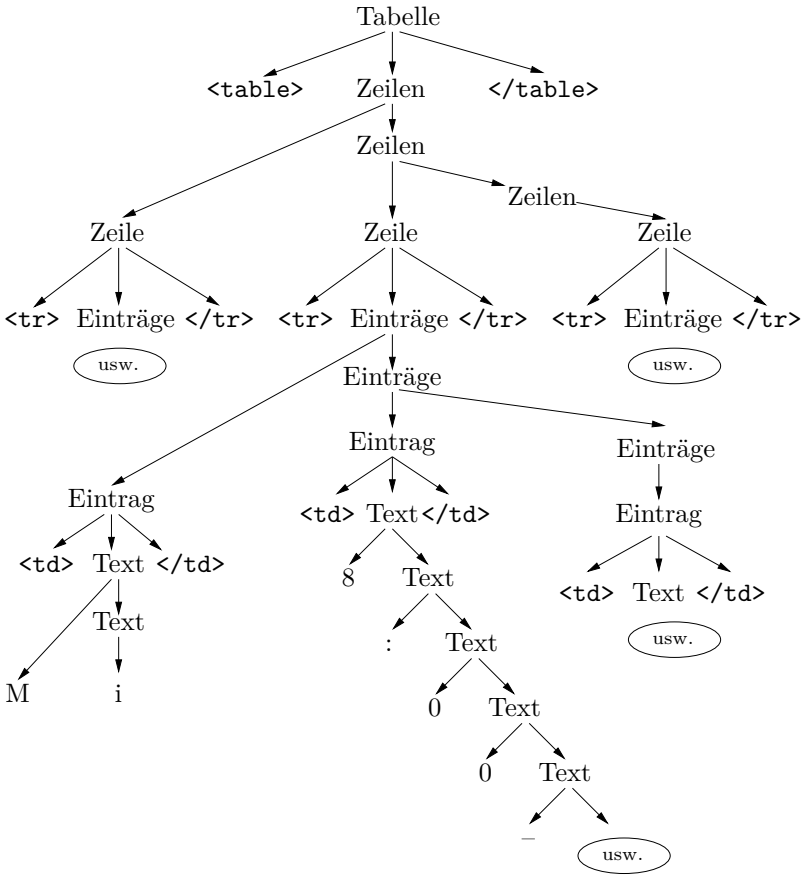


Abbildung 8.1.: Ableitungsbaum für eine Beispiel-HTML-Tabelle

Definition 8.14. Im **Wortproblem** für eine Grammatik $G = (\Sigma, V, S, P)$ und ein Wort $w \in \Sigma^*$, ist zu überprüfen, ob $w \in L(G)$ gilt. Wortproblem

Wenn die Grammatik G eine Programmiersprache beschreibt und wenn w ein vermeintliches Programm dieser Programmiersprache ist, dann muss schnell geklärt werden, ob denn nun w wirklich ein korrektes Programm ist. Mit anderen Worten, wir müssen die „Komplexität“ des Wortproblems untersuchen. Eine Diskussion von Beschreibungsmechanismen ohne Bezug zur Berechnungskomplexität ist sinnlos!

In der Berechnungskomplexität wird untersucht, welche Probleme viele Ressourcen für ihre Lösung, welche Probleme einfach zu lösen sind und welche Probleme überhaupt keine automatisierten Lösungen erlauben. Natürlich wird man ein solche vollständige Katalogisierung von Problemen in Schwierigkeitsklassen nie und nimmer abschließend durchführen können, aber man muss Begriffe zur Verfügung stellen, um den hinreichend und notwendigen Ressourcenaufwand für ein konkretes Problem untersuchen zu können. Genau diese Aufgabe erfüllen Komplexitätsklassen, die wir in Abschnitt 8.4.2 einführen.

8.4.1. Die Chomsky-Hierarchie

Wir haben in Folgerung 8.11 nachgewiesen, dass die Klasse der kontextfreien Sprachen eine echte Obermenge der Klasse der regulären Sprachen ist. Es gibt weitere wichtige Sprachklassen wie die Klasse der kontextsensitiven Sprachen, bzw. die Klasse der rekursiv aufzählbaren Sprachen, die mit Hilfe der kontextsensitiven und allgemeinen Grammatiken beschrieben werden.

Definition 8.15. Sei $G = (\Sigma, V, S, P)$ eine Grammatik.

allgemeine Grammatik

(a) G heißt eine **allgemeine Grammatik**, wenn alle Produktionen von G die Form

$$\alpha \to \beta$$

haben, wobei $\alpha \in (\Sigma \cup V)^* \cdot V \cdot (\Sigma \cup V)^*$ mindestens eine Variable besitzt und $\beta \in (\Sigma \cup V)^*$ ein beliebiges Wort ist.

kontextsensitive Grammatik

(b) G heißt eine **kontextsensitive Grammatik**, wenn G wie eine allgemeine Grammatik aufgebaut ist, allerdings müssen alle Produktionen $\alpha \to \beta$ in P längenerhaltend sein, d.h. es muss $|\alpha| \leqslant |\beta|$ gelten.

erzeugte Sprache

Analog zur Definition kontextfreier Sprachen (vgl. Definition 8.5) kann man auch die von einer allgemeinen Grammatik G **erzeugte Sprache** $L(G)$ einführen:

Definition 8.16. Sei L eine Sprache.

rekursiv aufzählbare Sprache
$\mathcal{L}_0$

(a) L ist eine **rekursiv aufzählbare Sprache**, wenn es eine allgemeine Grammatik G mit $L = L(G)$ gibt. Die Klasse aller rekursiv aufzählbaren Sprachen wird mit $\mathcal{L}_0$ bezeichnet.

kontextsensitive Sprache

(b) L ist eine **kontextsensitive Sprache**, wenn es eine kontextsensitive Grammatik G mit $L = L(G)$ gibt.

$\mathcal{L}_1$

Beachte, dass kontextsensitive Grammatiken G nur Sprachen $L(G)$ erzeugen, die das leere Wort nicht enthalten –denn eine Produktion $A \to \epsilon$ für eine Variable A ist nicht längenerhaltend. Man definiert deshalb die Klasse $\mathcal{L}_1$ der kontextsensitiven Sprachen durch

$$\mathcal{L}_1 := \{\, L(G) \,:\, G \text{ ist kontextsensitiv } \} \cup \{\, L(G) \cup \{\, \epsilon \,\} \,:\, G \text{ ist kontextsensitiv } \}.$$

$\mathcal{L}_2$
$\mathcal{L}_3$

(c) Die Klasse aller kontextfreien Sprachen wird mit $\mathcal{L}_2$, die Klasse aller regulären Sprachen wird mit $\mathcal{L}_3$ bezeichnet.

Satz 8.17. Die Sprachklassen bilden eine Hierarchie, die sogenannte Chomsky-Hierarchie, denn es ist

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0.$$

Beweis: Die echte Inklusion $\mathcal{L}_3 \subsetneq \mathcal{L}_2$ ist eine direkte Konsequenz der Folgerung 8.11. Die restlichen Inklusionen werden in der Veranstaltung „Theoretische Informatik 2“ gezeigt. □ □

Warum wählt man denn nicht gleich allgemeine Grammatiken? Weil das Wortproblem viel zu schwierig ist!

8.4.2. Komplexitätsklassen

Wenn wir davon sprechen ein Problem L zu lösen, meinen wir damit für eine Eingabe (oder eine Instanz) w korrekt festzustellen, ob w zu L gehört. Wir haben viele Beispiele bereits kennengelernt:

(a) Im KNF-Erfüllbarkeitsproblem ist festzustellen, ob eine aussagenlogische Formel w in konjunktiver Normalform erfüllbar ist.

(b) Im Problem der Hamilton-Kreise (bzw. Euler-Kreise) ist für einen ungerichteten Graphen w zu entscheiden, ob w einen einfachen Kreis besitzt, der alle Knoten besucht (bzw. ob w einen Kreis besitzt, der alle Kanten genau einmal durchläuft).

(c) Das Clique-Problem ist „wirklich“ ein Optimierungsproblem: Für einen ungerichteten Graphen w ist die maximale Größe eines vollständigen Teilgraphs von w zu bestimmen. Wir können aber auch Optimierungsprobleme als Sprachen auffassen: Eingaben sind von der Form $W = (w, \ell)$ und es ist festzustellen, ob w einen vollständigen Graphen der Größe mindestens ℓ besitzt.

Komplexitätsklassen, die Probleme nach der für ihre Lösung hinreichenden Laufzeit, bzw. hinreichendem Speicherplatz aufführen, werden in den Veranstaltungen „Theoretische Informatik 1“ und „Theoretische Informatik 2“ eingeführt.

(a) Probleme, deren Lösung mit „schnellen“ Algorithmen berechnet werden können, werden in der Komplexitätsklasse P gesammelt. Insbesondere fordert man von einem schnellen Algorithmus, dass seine Laufzeit „polynomiell in der Länge der Eingabe“ ist: P ist ein Akronym für *polynomielle Laufzeit*. P

Wir haben bereits mehrere Probleme in P betrachtet. (Der Nachweis, dass diese Probleme zur Klasse P gehören wird für die meisten Probleme in der Veranstaltung „Theoretische Informatik 1“ geführt.)

- Labyrinth-Probleme („gibt es einen Weg von Knoten a nach Knoten b?“) gehören genauso wie
- das Problem der Euler-Kreise zur Klasse P.
- Auch die Zuordnungsprobleme aus Abschnitt 5.1.2.3 können „effizient“ gelöst werden (siehe die Vorlesung „Approximationslagorithmen“) und gehören zur Klasse P.
- Natürlich kann jede reguläre Sprache von einem DFA akzeptiert werden. Das Wortproblem für reguläre Sprachen kann damit sogar in linearer Zeit durch eine einfache for-Schleife gelöst werden.
- Kontextfreie Sprachen sind ungleich komplizierter als reguläre Sprachen, können aber noch in polynomieller Zeit akzeptiert werden (siehe die Veranstaltung „Theoretische Informatik 2“). Damit gehört auch das Wortproblem für kontextfreie Sprachen zur Klasse P.

(b) Probleme, die schnelle nichtdeterministische Algorithmen besitzen, werden in der Komplexitätsklasse NP gesammelt. NP ist ein Akronym für *nichtdeterministisch polynomielle Laufzeit*. NP

Was bedeutet es, dass ein nichtdeterministischer Algorithmus A eine Eingabe w akzeptiert? A akzeptiert w genau dann, wenn mindestens eine Berechnung von A die Eingabe w akzeptiert. (Diese Definition ist völlig analog zur Definition der Akzeptanz durch NFAs.)

Beachte, dass jeder (deterministische) Algorithmus sich auch als nichtdeterministischer Algorithmus auffassen lässt: Es gilt also $\mathrm{P} \subseteq \mathrm{NP}$.

NP-vollständig

Unter den Problemen in NP sind NP-vollständige Probleme die schwierigsten: Ist ein Problem L NP**-vollständig** und kann L schnell, also in polynomieller Zeit gelöst werden, dann lassen sich alle Probleme in NP schnell lösen. (Zur Begründung siehe die Veranstaltung „Theoretische Informatik 1“.)

$P \stackrel{?}{=} NP$

Die $P \stackrel{?}{=} NP$ Frage ist eine der wichtigsten Fragen der theoretischen Informatik. Man erwartet eine negative Antwort: Kein einziges NP-vollständiges Problem wird dann einen schnellen Algorithmus besitzen.

Wir haben zahlreiche NP-vollständige Probleme kennengelernt:

- das KNF-Erfüllbarkeitsproblem,
- das Problem der Hamilton-Kreise (für gerichtete und ungerichtete Graphen),
- das Färbungsproblem,
- das Problem des Feedback Vertex Set (für gerichtete und ungerichtete Graphen)
- und die stark miteinander verwandten Probleme maximale Clique, größte unabhängige Mengen und kleinste Knotenüberdeckungen (jeweils in Sprachenversion).

PSPACE

(c) Die Komplexitätsklasse PSPACE besteht aus allen Problemen, die von deterministischen Algorithmen mit nur polynomiellen Speicherplatzverbrauch lösbar sind. PSPACE ist die Kurzform von *polynomial Space*.

Beachte, dass die Laufzeit eines Algorithmus exponentiell groß im Speicherplatzverbrauch sein kann: In PSPACE liegen deshalb auch sehr schwierige Probleme. Die schwierigsten Probleme in PSPACE nennt man PSPACE**-vollständig**.

PSPACE-vollständig

- Das Wortproblem für kontextsensitive Sprachen liegt in PSPACE und es gibt kontextsensitive Sprachen, deren Wortproblem PSPACE-vollständig ist.
- Die „unschuldige“ Frage, ob ein NFA alle Worte seines Alphabets akzeptiert ist PSPACE-vollständig. Beachte, dass die entsprechende Frage für DFAs einfach ist: Minimiere den DFA und überprüfe, ob der minimierte Automat nur aus einem Zustand und zwar aus einem akzeptierenden Zustand besteht. Die Minimierung von NFAs führt also auf ein PSPACE-vollständiges Problem.
- Für viele Zwei-Personen Spiele (mit Spielern Alice und Bob) ist die Frage, ob Alice in einer gegebenen Spielsituation eine Gewinnstrategie hat, in polynomiellen Speicherplatz beantwortbar. Dame, Go, Hex, Schach, Sokoban ... führen bei entsprechender Verallgemeinerung auf PSPACE-vollständige Probleme.

  Für interessante Spiele sollte die Bestimmung einer Gewinnstrategie schwierig sein: Die PSPACE-Vollständigkeit ist in diesem Fall sogar ein Gütesiegel.

E

entscheidbar
unentscheidbare Sprachen

(d) Die Klasse E besteht aus allen Sprachen, die sich von einem deterministischen Algorithmus ohne Einschränkung von Laufzeit oder Speicherplatz lösen lassen. E ist ein Akronym für *entscheidbar*. Man sagt auch, dass Sprachen in E **entscheidbar** sind, **unentscheidbare Sprachen**, also Sprachen außerhalb der Klasse E, können selbst bei uneingeschränkten Ressourcen nicht von Rechnern gelöst werden!

In Satz 4.7 wird die Existenz eines speziellen unentscheidbaren Problems gezeigt. Dieses Problem erscheint sehr künstlich, aber als Konsequenz wird in der „Theoretischen Informatik 1“ und „Theoretischen Informatik 2“ eine Vielzahl wichtiger unentscheidbarer Probleme nachgewiesen. Hier sind einige Beispiele.

- Das Halteproblem besteht aus allen Paaren (P, w), wobei das C++ Programm P auf der Eingabe w hält. Das Halteproblem ist unentscheidbar. Es kann also keinen Super-Compiler geben, der zusätzlich bestätigt oder widerlegt, dass ein Eingabeprogramm auf einer bestimmten Eingabe überhaupt hält.
- Es gibt allgemeine Grammatiken G, so dass das Wortproblem für $L(G)$ unentscheidbar ist: Finger weg von allgemeinen Grammatiken!
- Das Problem zu entscheiden, ob zwei kontextfreie Grammatiken dieselbe Sprache erzeugen, ist ebenfalls unentscheidbar.

In der „Theoretischen Informatik 2“ werden die Inklusionsketten

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathrm{P} \subseteq \mathrm{NP} \subseteq \textsc{Pspace} \subsetneq \mathtt{E} \subsetneq \mathcal{L}_0$$

sowie

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \textsc{Pspace}$$

gezeigt. Für Probleme, die sich nicht in der Klasse P befinden, kann man auf eine Lösung nur für Teilmengen von Eingaben hoffen.

Mit Hilfe weiterer Komplexitätsklassen wird die notwendige Laufzeit bei parallelen Rechnern untersucht, bzw. die Laufzeit für randomisierte Algorithmen (Algorithmen, die bei mehreren Optionen eine Option zufällig auswürfeln) und Quantenalgorithmen studiert. Diese Themen und mehr –wie etwa die Nachbildung „echten Zufalls“ durch Pseudorandom-Generatoren oder Ansätze zur Untersuchung der $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ Frage– werden in der Veranstaltung „Komplexitätstheorie“ aufgegriffen.

8.5. Zusammenfassung und Ausblick

Kontextfreie Grammatiken eignen sich für die Modellierung rekursiv definierter Strukturen. Wir haben viele Beispiele gesehen, aber viele Fragen bleiben offen:

Welche Sprachen sind kontextfrei und welche nicht?

Die Sprache $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ haben wir bereits in Beispiel 8.7 als kontextfrei nachgewiesen.

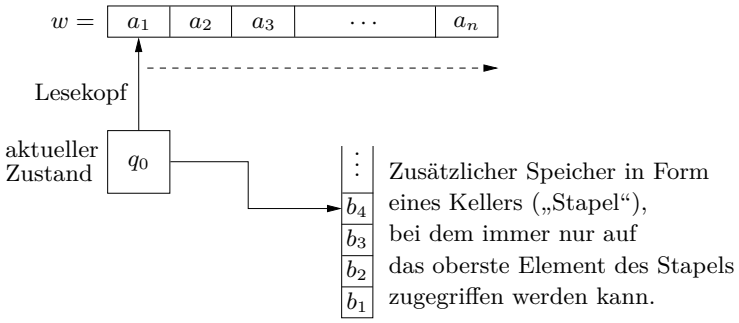
Satz: Es gibt keine KFG, die die Sprache $L_2 := \{a^n b^n c^n : n \in \mathbb{N}\}$ erzeugt. Die Sprache L_2 ist also nicht kontextfrei.

Beweis: In der Vorlesung „Theoretische Informatik 2“ wird in Analogie zu Satz 7.53 ein „Pumping-Lemma für KFGs“ gezeigt. Eine Anwendung dieses Pumping-Lemmas beweist die Behauptung. □

Gibt es ein Automatenmodell für kontextfreie Sprachen?

Ja, es gibt ein Automatenmodell, das genau die kontextfreien Sprachen akzeptiert, nämlich das Modell der **Kellerautomaten**. Kellerautomaten

Schematische Darstellung der Verarbeitung eines Eingabeworts durch einen Kellerautomaten:



In der „Theoretischen Informatik 2“ wird gezeigt, dass kontextfreie Grammatiken und nichtdeterministische Kellerautomaten genau die Klasse der kontextfreien Sprachen erzeugen, bzw. akzeptieren.

Deterministisch kontextfreie Sprachen

Ein Parser muss das Wortproblem (siehe Definition 8.14) für jedes Eingabeprogramm w lösen, deshalb kommt dem Wortproblem eine große Bedeutung zu. Einen „effizienten“ Algorithmus zum Lösen des Wortproblems für KFGs werden Sie in der Vorlesung „Theoretische Informatik 2“ kennenlernen: den so genannten CYK-Algorithmus, der nach seinen Erfindern Cocke, Younger und Kasami benannt ist. Der CYK-Algorithmus löst das Wortproblem für eine Grammatik $G = (\Sigma, V, S, P)$ in Zeit proportional zu $|w|^3 \cdot |P|$ und berechnet sogar einen Ableitungsbaum. Allerdings ist kubische Laufzeit völlig inakzeptabel.

deterministisch kontextfreie Sprachen

Man beschränkt sich deshalb bei der Lösung des Wortproblems auf **deterministisch kontextfreie Sprachen**, nämlich die Sprachen, die von einem deterministischen Kellerautomaten akzeptiert werden. Deterministisch kontextfreie Sprache besitzen Parser, die das Wortproblem in Linearzeit lösen, also syntaktische Korrektheit blitzschnell verifizieren.

Die Syntax vieler wichtiger Programmiersprachen wird deshalb von deterministischen kontextfreien Sprachen definiert.

Eindeutige Grammatiken

Viele Parser übersetzen ein Anwenderprogramm in einen Ableitungsbaum, um damit die Semantik des Programms verstehen zu können. Es ist deshalb im Entwurf von kontextfreien Grammatiken wichtig, dass jedes von der Grammatik erzeugbare Wort genau einen Ableitungsbaum hat und deshalb auch genau eine Bedeutung besitzt.

deterministischer Kellerautomat
eindeutige Grammatik

Nenne eine Grammatik eindeutig, wenn jedes erzeugbare Wort genau einen Ableitungsbaum besitzt. Ein **deterministischer Kellerautomat** besitzt eine „äquivalente **eindeutige Grammatik**, deterministische Kellerautomaten sind deshalb von besonderer Wichtigkeit. Auch hier können Sie mehr in der Vorlesung „Theoretische Informatik 2“ erfahren.

8.6. Literaturhinweise zu Kapitel 8

[14] Kapitel 6.1 und 6.2

[26] Kapitel 6.1

[27] Kapitel 6.1

[23] Kapitel 1.3

8.7. Übungsaufgaben zu Kapitel 8

Aufgabe 8.1.

(a) Sei $L_1 = \{a^i b^j c^k : k = i + j\}$.
Geben Sie eine kontextfreie Grammatik G_1 an, sodass $L(G_1) = L_1$ gilt.

(b) Betrachten Sie die folgende Grammatik $G_2 = (\Sigma, V, S, P)$ mit

$$V = \{A\}, \quad \Sigma = \{0, 1\}, \quad S = A, \quad P = \{A \to AA \,|\, 0A1 \,|\, 1A0 \,|\, \varepsilon\}.$$

(i) Geben Sie eine Ableitung und einen Ableitungsbaum des Wortes $w = 00101101$ an.

(ii) Beschreiben Sie die von G_2 erzeugte Sprache $L(G_2)$ mathematisch oder umgangssprachlich.

Aufgabe 8.2. Die Sprache $\mathbf{REG_{ab}}$ der *Regulären Ausdrücke* über dem Alphabet $\Sigma = \{a, b\}$ ist die Menge der Worte über dem Alphabet $A = \{\emptyset, \varepsilon, a, b, |, \cdot, ^*, (,)\}$, die rekursiv wie folgt definiert ist:

Basisregel:	(B1)	Die Symbole $\emptyset$ und ε sind in $\mathbf{REG_{ab}}$.
	(B2)	Die Buchstaben a und b sind in $\mathbf{REG_{ab}}$.
Rekursive Regeln:	(R1)	Ist R in $\mathbf{REG_{ab}}$, so ist auch R^* in $\mathbf{REG_{ab}}$.
	(R2)	Sind R und S in $\mathbf{REG_{ab}}$, so ist auch $(R \cdot S)$ in $\mathbf{REG_{ab}}$.
	(R3)	Sind R und S in $\mathbf{REG_{ab}}$, so ist auch $(R\|S)$ in $\mathbf{REG_{ab}}$.

Geben Sie eine kontextfreie Grammatik G an, so dass $L(G) = \mathbf{REG_{ab}}$ ist. Stellen Sie einen Ableitungsbaum für das folgende Wort auf:

$$((a|b)^* \cdot \emptyset)$$

Geben Sie außerdem eine umgangssprachliche Beschreibung der gemäß Definition (7.67) durch diesen regulären Ausdruck beschriebenen Sprache $L(((a|b)^* \cdot \emptyset))$ an.

Aufgabe 8.3. Der Fernsehsender DiscoMod hat sich auf Sendungen zum Thema Tanzen spezialisiert. Markenzeichen dieses Senders sind der eigens dafür erfundene Tanz Mod-Flotttrott und der dazu gehörende Wettbewerb um den Tanzmeisterpokal Mod-Flotttrott-Pott. Der Mod-Flotttrott verwendet dabei die folgenden Grundschritte, die mit Buchstaben aus dem Alphabet $\Sigma := \{k, l, r, v, z\}$ bezeichnet werden:

- in die Hände klatschen (abgekürzt als k),
- einen Sprung nach links (abgekürzt als l),
- einen Sprung nach rechts (abgekürzt als r),
- einen Schritt vor (abgekürzt als v),
- einen Schritt zurück (abgekürzt als z).

Natürlich dürfen diese Grundschritte nicht frei kombiniert werden. Eine Folge von Grundschritten ist genau dann ein korrekter Mod-Flotttrott-Tanz, wenn sie folgenden Regeln genügt:

- Einmal in die Hände klatschen ist ein korrekter Tanz.

- Einmal in die Hände klatschen und dann einen korrekten Tanz tanzen ist ebenfalls ein korrekter Tanz.
- Einen Sprung nach links, einen korrekten Tanz tanzen und dann einen Sprung nach rechts ist ebenfalls ein korrekter Tanz.
- Einen Schritt vor, noch einen Schritt vor, einen Schritt zurück, einen korrekten Tanz tanzen und dann einen Schritt zurück ist ebenfalls ein korrekter Tanz.

Dabei sei L definiert als die Sprache aller Wörter aus Σ^*, die einem korrekten Tanz entsprechen. Beispielsweise gilt $k \in L$ und $vvzkkz \in L$, und außerdem auch $lr \notin L$ und $vzvz \notin L$.

Dieses Jahr wird das Team der Schiedsrichter für die Tanzmeisterschaft durch einen Roboter ergänzt, den Mod-Flotttrott-Bot, der die Tänze vollautomatisch bewertet (nämlich als Hot! oder Schrott!). Allerdings wurde der Mod-Flotttrott-Bot Opfer eines Mod-Flotttrott-Bot-Komplotts: Spione eines anderen Senders haben die Definition der korrekten Tänze gelöscht. Glücklicherweise wird der Mod-Flotttrott-Bot durch kontextfreie Grammatiken programmiert, so dass Sie den Leuten von DiscoMod helfen können (und diese so vor Mod-Flotttrott-Bot-Spott retten).

Geben Sie eine kontextfreie Grammatik G an, für die $L(G) = L$ gilt.

Aufgabe 8.4. Kritischen Stimmen zufolge lässt sich die Handlung der neueren Starwars-Filme als mehr oder weniger kontextfrei beschreiben. Nach länglichen Diskussionen hat sich die Fangemeinschaft darauf einigen können, dass sich ein Starwars-Film im Wesentlichen aus (bis zu) fünf Typen von Szenen zusammensetzt:

d	Laserschwert-**D**uelle
r	Schlachten oder Verfolgungsjagden mit **R**aumschiffen
j	altkluge **J**edi-Weisheiten, von einem Jedi-Meister vorgetragen
b	ein **B**ösewicht, der ein rotes Laserschwert präsentiert
t	Explosion des **T**odessterns

Jede solche Szene kann auch wiederholt in einem Film vorkommen. Ein Starwars-Film kann somit als Wort $f \in \Sigma^*$ mit $\Sigma := \{\mathrm{d}, \mathrm{r}, \mathrm{j}, \mathrm{b}, \mathrm{t}\}$ modelliert werden.

Intensive algorithmengestützte Marktforschung ergab, dass ein Film $f \in \Sigma^*$ genau dann erfolgreich wird, wenn er nach folgenden Regeln aufgebaut ist:

- Ein Film bestehend aus einer Bösewicht-Szene, anschließendem Laserschwert-Duell und der Explosion des Todessterns am Ende ist erfolgreich.

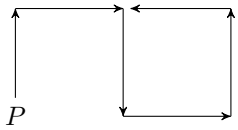
Außerdem sind folgende Ersetzungen erlaubt:

- Ein Laserschwert-Duell darf stets ersetzt werden durch eine Raumschiffschlacht gefolgt von einer Jedi-Weisheit und anschließendem Laserschwert-Duell.
- Eine Raumschiffschlacht darf stets ersetzt werden durch
  - zwei aufeinanderfolgende Laserschwert-Duelle oder
  - eine Jedi-Weisheit gefolgt von einer Raumschiffschlacht und anschließendem Laserschwert-Duell.
- Eine Jedi-Weisheit darf durch eine Bösewicht-Szene ersetzt oder ersatzlos gestrichen werden.

Sei $F \subseteq \Sigma^*$ die durch die obigen Regeln beschriebene Sprache aller möglichen erfolgreichen Starwars-Filme. Es gilt beispielsweise bdt $\in F$, brjdt $\in F$ und bdddt $\in F$.

Konstruieren Sie eine Grammatik $G = (\Sigma, V, S, P)$, die genau die Sprache aller erfolgreichen Starwars-Filme beschreibt, d.h. es soll $L(G) = F$ gelten. Erläutern Sie auch Ihre Notation.

Aufgabe 8.5. Der wahnsinnige Wissenschaftler Dr. Fo hat sein neuestes Geheimversteck auf einer unbewohnten Vulkaninsel in der einsamen Weite eines abgelegenen Teils des pazifischen Ozeans errichtet. Dort schmiedet er Rachepläne gegen seine ehemaligen Kollegen (die ihn ausgelacht haben), gegen Geheimagenten (die seine vorigen Geheimverstecke zerstört haben) und ganz besonders gegen seine früheren Studenten (die seine Übungsblätter nicht zu schätzen wussten).



Um sich vor ungebetenen Gästen zu schützen hat er den Flugroboter D.I.S.M.O.D.[2] gebaut, der entlang vorgegebener Routen hoch über dem Meer herumfliegt und Eindringlinge sucht. Eine Route besteht dabei aus einem Wort x über dem Alphabet $\Sigma = \{n, s, w, o\}$. Für jedes Zeichen in x (gelesen von links nach rechts) fliegt der Roboter dabei 100 Meter in eine bestimmte Richtung: Bei n nach Norden, bei s nach Süden, bei w nach Westen und bei o nach Osten. Zum Beispiel entspricht das Wort *nosonw* der im Bild links angegebenen Route (wobei P der Startpunkt ist).

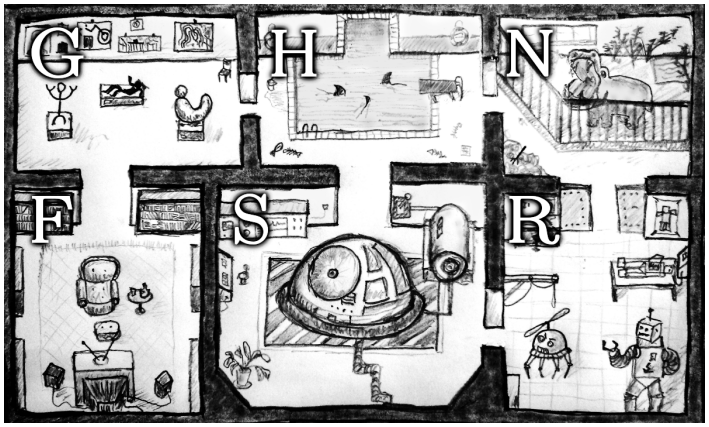
(a) Betrachten Sie die folgende Grammatik $G = (\Sigma, N, S, P)$ mit $N = \{S, A, W, O\}$, und

$$P = \{S \to nAs, A \to \epsilon, A \to wO, A \to oW, W \to wA, W \to oWW, O \to oA, O \to wOO\}$$

(i) Überprüfen Sie für jedes der folgenden Worte, ob es in $L(G)$ liegt. Wenn ja, geben Sie einen Ableitungsbaum für dieses Wort an; ansonsten begründen Sie, warum dieses Wort nicht zur Sprache gehört.

$$w_1 = nAs \qquad w_2 = nowwos \qquad w_3 = nwws \qquad w_4 = nwwowoos$$

(ii) Beschreiben Sie, welche Sprache $L(G)$ von G erzeugt wird. Was für ein Flugverhalten zeigt D.I.S.M.O.D., wenn er ein Wort aus $L(G)$ als Route erhält?



Um auch in seinem Versteck für Sicherheit zu sorgen, hat Dr. Fo den Laufroboter M.O.D.[3] gebaut. Dieser erhält Routen im gleichen Format wie D.I.S.M.O.D.; der einzige Unterschied ist, dass er in jedem Schritt nicht 100 Meter in die entsprechende Richtung fliegt, sondern einfach in den Nachbarraum in der entsprechenden Richtung läuft. Die aufgeführten Räume im angegebenen Lageplan des Verstecks sind Galerie (G), Haifischbecken (H), Nilpferdgehege (N), Fernsehzimmer (F), Superlaser (S) und Roboterraum (R).

[2] Durchaus Intelligenter Schwebender Meeres-Observations-Droide
[3] Misstrauischer Oniscidea-Droide

Angenommen, M.O.D. befindet sich in Raum S. Dann läuft er bei einem o nach R und bei n nach H. Allerdings kann er nicht mittels w nach F laufen, da sich dort kein Durchgang befindet. Auch s führt zu keinem Raum.

(b) Dr. Fo möchte nun alle Routen beschreiben, die bei R beginnen, nur die im Lageplan aufgeführten Durchgänge verwenden, und wieder bei R enden. Mögliche Worte wären daher $nwwsnoso$ und $wnos$, aber nicht ww oder no. Geben Sie eine kontextfreie Grammatik über Σ an, die genau die Worte über Σ erzeugt, die eine solche Route beschreiben.

Hinweis: Nutzen Sie für jeden der Räume ein eigenes Nichtterminalsymbol. Beschreiben Sie jede mögliche Richtung, die von jeder dieser Positionen aus jeweils möglich ist, durch eine Regel der Grammatik.

Aufgabe 8.6.

(a) Betrachten Sie die folgende Grammatik $G := (\Sigma, V, S, P)$ mit $V := \{S, A, B\},\ \Sigma := \{0, 1\}$ und

$$P := \big\{S \to A \,|\, B,\quad A \to 0A1 \,|\, 0A \,|\, 0,\quad B \to 0B1 \,|\, B1 \,|\, 1\big\}.$$

(i) Geben Sie eine Ableitung des Wortes $w_1 := 0001$ an.

(ii) Geben Sie einen Ableitungsbaum des Wortes $w_2 := 00111$ an.

(iii) Beschreiben Sie die von G erzeugte Sprache $L(G)$ mathematisch oder umgangssprachlich.

(b) Die Klammersprache K über dem Alphabet $\Sigma := \{\,[\,,\,],\ \langle\,,\,\rangle\,\}$ ist wie folgt rekursiv definiert:

(B) $\varepsilon \in K$

(R1) Ist $w \in K$, dann ist auch $[w] \in K$.

(R2) Ist $w \in K$, dann ist auch $\langle w \rangle \in K$.

(R3) Ist $u \in K$ und ist $v \in K$, dann ist auch $uv \in K$.

Konstruieren Sie eine kontextfreie Grammatik G_K für die Klammersprache K.

(c) Betrachten Sie die folgende Grammatik $G := (\Sigma, V, S, P)$ mit $V := \{S\},\ \Sigma := \{0, 1\}$ und

$$P := \big\{S \to \varepsilon \mid SS \mid 0S1 \mid 1S0\big\}.$$

(i) Geben Sie eine Ableitung und einen Ableitungsbaum für das Wort 1001 an.

(ii) Beschreiben Sie die von G erzeugte Sprache $L(G)$ mathematisch oder umgangssprachlich.

Aufgabe 8.7. Bei der Sportart Parkour sind mithilfe verschiedener Fortbewegungselemente Strecken in städtischer Umgebung möglichst elegant und effizient zurückzulegen. Ein Traceur (Parkour-Läufer) führt in einem Parkour-Lauf im Wesentlichen die folgenden Aktionen aus:

l **L**aufen, eine wenig spektakuläre und daher verpönte Art der Fortbewegung

r **R**ollen, in der Regel über die Schulter, um die Landung nach einem Sprung abzufedern

s **S**pringen, z. B. vom Dach eines Hauses auf einen Balkon eines Nachbarhauses

v **V**aulting, ein Hindernis durch einen Sprung unter Zuhilfenahme der Hände überwinden

w **W**allclimbing, eine Mauer emporklettern

Jede solche Aktion kann auch mehrfach in einem Lauf vorkommen. Ein Parkour-Lauf kann somit als Wort $p \in \Sigma^*$ mit $\Sigma := \{\mathrm{l}, \mathrm{r}, \mathrm{s}, \mathrm{v}, \mathrm{w}\}$ modelliert werden.

Zwar behaupten die Traceurs, dass ihre Aktionen sehr auf die jeweilige Umgebung abgestimmt sind, aber die Hersteller von Fitness-Armbändern haben durch algorithmengestützte Auswertung der gesammelten Daten herausgefunden, dass jeder Parkour-Lauf nur nach den folgenden Regeln aufgebaut ist:

1. Zu Beginn läuft der Traceur, springt anschließend, rollt über die Schulter und läuft weiter.

Außerdem sind folgende Ersetzungen erlaubt:

2. Rollen darf stets ersetzt werden durch zweimaliges, unmittelbar aufeinanderfolgendes Rollen.
3. Laufen darf stets ersetzt werden durch
   - Laufen, gefolgt von Springen und anschließendem Abrollen oder
   - Laufen, eine Vaulting-Aktion und weiteres Laufen.
4. Vaulting darf stets durch Wallclimbing ersetzt werden.
5. Rollen, Springen, Vaulting und Wallclimbing dürfen stets ersatzlos gestrichen oder durch Laufen ersetzt werden.

Sei PARKOUR $\subseteq \Sigma^*$ die durch die obigen Regeln beschriebene Sprache aller möglichen Parkour-Läufe. Es gilt beispielsweise lsrl $\in$ PARKOUR, lsrrl $\in$ PARKOUR und lwlsrl $\in$ PARKOUR.

Konstruieren Sie eine kontextfreie Grammatik G_{Parkour} für die Sprache PARKOUR. Erläutern Sie auch Ihre Notation.

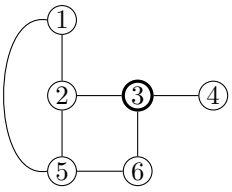
Aufgabe 8.8. Sei $\Sigma = \{a, c, g, t\}$. Die Sprache HN $\subseteq \Sigma^*$ (eine vereinfachte Form der Sprache aller DNA-Stränge, die eine so genannte *Haarnadelschleife* bilden) sei wie folgt definiert:

Basisregel: (B) Es gilt: $a, c, g, t \in$ HN.

Rekursive Regel: (R) Ist $w \in$ HN, so sind auch $awt, twa, cwg, gwc \in$ HN.

So ist z.B. das Wort *tcactga* in HN. Geben Sie eine kontextfreie Grammatik G an, so dass $L(G) =$ HN ist. Geben Sie außerdem einen Ableitungsbaum für das Wort *ctagggcctag* an.

Aufgabe 8.9. Das Wegenetz des kleinen Dorfes Dismodau ist durch den unten dargestellten Graphen gegeben, wobei die Knoten den Häusern und die Kanten den Wegen zwischen den Häusern entsprechen. Sie wohnen in Haus 3 und haben viel Zeit darauf verwendet, Ihrem Hund Marcus Aurelius II das selbständige Gassigehen mithilfe einer rechtsregulären Grammatik beizubringen.



Bevor Sie ihn aus dem Haus schicken, geben Sie ihm ein Gassi-Kommando, das seine Route durch Dismodau mithilfe der Himmelsrichtungen Norden (**N**), Süden (**S**), Osten (**O**) und Westen (**W**) beschreibt. Beispielsweise bedeutet das Kommando **WNWOWON**, dass Marcus Aurelius II

nach seinem Start in Haus 3 die Häuser 2, 1, 5, 6, 5 und 6 besucht und schließlich wieder zu Haus 3 zurückkehrt.

Modellieren Sie die Sprache aller Gassi-Routen[4], die

- in Haus 3 starten und in Haus 3 enden und
- nur entlang der Wege zwischen den Häusern in Dismodau verlaufen

durch eine rechtsreguläre Grammatik $G_{\text{assi}} := (\Sigma, V, S, P)$. Nehmen Sie an, dass auch das leere Wort ε eine Gassi-Route beschreibt.

Hinweis: Fassen Sie die Häuser als Nichtterminalsymbole auf, d. h. $V \supseteq \{1, 2, \ldots, 6\}$ und beschreiben jeden „Schritt“ des Hundes durch eine entsprechende Produktionsregel.

[4] Ja, Hunde sind so schlau, dass sie reguläre Sprachen lernen können! ;-)

Teil IV.

Werkzeugkasten: Logik

9. Logik erster Stufe (Prädikatenlogik)

In Kapitel 3 haben wir bereits die **Aussagenlogik** kennengelernt, die einen Formalismus darstellt, mit dessen Hilfe man „Wissen“ modellieren und Schlüsse aus dem Wissen ziehen kann. In diesem Kapitel werden wir die **Logik erster Stufe** (bzw. **Prädikatenlogik**) als einen weiteren solchen Formalismus kennenlernen. Im Vergleich zur Aussagenlogik hat die Prädikatenlogik den Vorteil, dass

- eine klare Trennung zwischen „Daten“ einerseits und „Logik“ andererseits besteht, und dass in der Prädikatenlogik
- wesentlich umfangreichere Ausdrucksmöglichkeiten zur Verfügung stehen.

Der Preis für diese Vorteile ist allerdings, dass die Prädikatenlogik **algorithmisch** deutlich schwerer zu handhaben ist als die Aussagenlogik.

9.1. Motivation zur Logik erster Stufe

Grenzen der Aussagenlogik:

Beispiel 9.1 (Verwandtschaftsbeziehungen). Die Aussagenlogik kann helfen, um Aussagen der Art

> „Anne und Bernd sind Geschwister. Wenn Christine Annes Tochter ist, dann ist Bernd Christines Onkel.“

zu modellieren und Schlüsse daraus zu ziehen. Für die Modellierung der folgenden Aussage ist die Aussagenlogik aber eher ungeeignet:

> „Es gibt in Frankfurt mindestens 2 Leute, die mehr als 3 Kinder, aber selbst keine Geschwister haben.“

Beispiel 9.2 (Arithmetische Aussagen). Die Aussagenlogik kann helfen, um Sätze der Art

> „Wenn eine Zahl gerade ist, dann ist sie nicht ungerade.“

zu formalisieren. Für viele andere Aussagen ist die Aussagenlogik aber eher ungeeignet, zum Beispiel:

> „Es gibt eine Zahl, die nicht Summe zweier Primzahlen ist.“

Ein Überblick über die Logik erster Stufe:

Die Logik erster Stufe ist ein Formalismus, mit dem man die in den beiden obigen Beispielen genannten Aussagen bequem beschreiben kann. Genau wie die Aussagenlogik besitzt die Logik erster Stufe:

- eine **Syntax**, die festlegt, welche Zeichenketten Formeln der Logik erster Stufe sind und
- eine **Semantik**, die festlegt, welche „Bedeutung“ einzelne Formeln haben.

Die Logik erster Stufe beschäftigt sich mit **Objekten** (z.B. den Einwohnern Frankfurts und deren Verwandtschaftsbeziehungen (Beispiel 9.1) oder den natürlichen Zahlen und deren Addition und Multiplikation (Beispiel 9.2)) und **Aussagen über deren Eigenschaften**. (Im Gegensatz dazu beschäftigt sich die Aussagenlogik nicht mit Objekten sondern lediglich mit „wahren“ und „falschen“ Aussagen und deren Kombination.)

Vor der Einführung in die Syntax und die Semantik der Logik erster Stufe wenden wir uns zunächst den Objekten zu, über die Formeln der Logik erster Stufe „reden“ können.

9.2. Strukturen

Die Objekte, über die Formeln der Logik erster Stufe Aussagen treffen können, heißen **Strukturen**. Viele Objekte lassen sich auf natürliche Weise durch solche Strukturen repräsentieren, beispielsweise

Strukturen

- Graphen $G = (V, E)$
- die natürlichen Zahlen mit Addition und Multiplikation, $(\mathbb{N}, +, \times)$
- die reellen Zahlen mit Addition, Multiplikation und den Konstanten 0 und 1, $(\mathbb{R}, +, \times, 0, 1)$
- Datenbanken ...

9.2.1. Signaturen

Die im Folgenden definierten **Signaturen** legen den „Typ“ (bzw. das „Format“) der entsprechenden Strukturen fest. Formal ist eine Signatur σ eine Menge von Relations-, Funktions- und Konstantensymbolen. Die „Formeln der Logik erster Stufe über σ“ dürfen die in σ aufgeführten Relationen, Funktionen und Konstanten verwenden.

Definition 9.3. Eine **Signatur** (bzw. ein **Vokabular** bzw. eine **Symbolmenge**; englisch: signature, vocabulary) ist eine Menge σ von Relationssymbolen, Funktionssymbolen und/oder Konstantensymbolen. Jedes Relationssymbol $\dot{R} \in \sigma$ und jedes Funktionssymbol $\dot{f} \in \sigma$ hat eine **Stelligkeit** (bzw. Arität, engl. arity)

Signatur
Vokabular
Symbolmenge
Stelligkeit

$$\mathrm{ar}(\dot{R}) \in \mathbb{N}_{>0} \quad \text{bzw.} \quad \mathrm{ar}(\dot{f}) \in \mathbb{N}_{>0}.$$

Notation 9.4.

- In diesem Kapitel bezeichnet der griechische Buchstabe σ (in Worten: sigma) immer eine Signatur.
- Wir kennzeichnen Symbole aus σ immer mit einem Punkt, wie in $\dot{R}$ bzw. $\dot{f}$.
- Für Relationssymbole verwenden wir meistens Großbuchstaben wie $\dot{R}, \dot{P}, \dot{E}, \dot{R}_1, \dot{R}_2, \ldots$,
  für Funktionssymbole verwenden wir meistens Kleinbuchstaben wie $\dot{f}, \dot{g}, \dot{h}, \ldots$,
  für Konstantensymbole verwenden wir meistens Kleinbuchstaben wie $\dot{c}, \dot{d}, \ldots$.
- Gelegentlich verwenden wir als Relations- und Funktionssymbole auch Zeichen wie

$\dot{\leqslant}$ (2-stelliges Relationssymbol),
$\dot{+}, \dot{\times}$ (2-stellige Funktionssymbole),
$\dot{0}, \dot{1}$ (Konstantensymbole).

9.2.2. Strukturen über einer Signatur

Struktur
σ-Struktur

Definition 9.5. Eine **Struktur über der Signatur** σ (kurz: **σ-Struktur**) ist ein Paar

$$\mathfrak{A} \;=\; (A, \alpha),$$

bestehend aus:

Universum
Träger

- einer nicht-leeren Menge A, dem so genannten **Universum** (bzw. **Träger**, engl.: universe, domain) von $\mathfrak{A}$, und
- einer auf σ definierten Abbildung α, die
  - jedem Relationssymbol $\dot{R} \in \sigma$ eine Relation $\alpha(\dot{R}) \subseteq A^{\mathrm{ar}(\dot{R})}$ der Stelligkeit $\mathrm{ar}(\dot{R})$ zuordnet,
  - jedem Funktionssymbol $\dot{f} \in \sigma$ eine Funktion $\alpha(\dot{f})\colon A^{\mathrm{ar}(\dot{f})} \to A$ zuordnet,
  - jedem Konstantensymbol $\dot{c} \in \sigma$ ein Element $\alpha(\dot{c}) \in A$ zuordnet.

Notation 9.6.

- Strukturen bezeichnen wir meistens mit Fraktur-Buchstaben $\mathfrak{A}, \mathfrak{B}, \mathfrak{G}, \ldots$; das Universum der Strukturen durch die entsprechenden lateinischen Großbuchstaben, also $A, B, G, \ldots$.
- Ist $\mathfrak{A} = (A, \alpha)$ eine σ-Struktur, so schreiben wir für jedes Symbol $\dot{S} \in \sigma$ oft
  $$\dot{S}^{\mathfrak{A}} \quad \text{an Stelle von} \quad \alpha(\dot{S}).$$
  An Stelle von $\mathfrak{A} = (A, \alpha)$ schreiben wir oft auch $\mathfrak{A} = (A, (\dot{S}^{\mathfrak{A}})_{\dot{S} \in \sigma})$.
  Falls $\sigma = \{\dot{R}_1, \ldots, \dot{R}_k, \dot{f}_1, \ldots, \dot{f}_l, \dot{c}_1, \ldots, \dot{c}_m\}$ ist, schreiben wir auch
  $$\mathfrak{A} \;=\; (A, \dot{R}_1^{\mathfrak{A}}, \ldots, \dot{R}_k^{\mathfrak{A}}, \dot{f}_1^{\mathfrak{A}}, \ldots, \dot{f}_l^{\mathfrak{A}}, \dot{c}_1^{\mathfrak{A}}, \ldots, \dot{c}_m^{\mathfrak{A}}).$$

Beispiel 9.7 (Arithmetische Strukturen). Sei $\sigma_{\mathrm{Ar}} := \{\dot{+}, \dot{\times}, \dot{0}, \dot{1}\}$, wobei $\dot{+}$ und $\dot{\times}$ 2-stellige Funktionssymbole und $\dot{0}$ und $\dot{1}$ Konstantensymbole sind. Wir betrachten die σ_{Ar}-Struktur

$$\mathcal{N} \;:=\; (\mathbb{N}, \dot{+}^{\mathcal{N}}, \dot{\times}^{\mathcal{N}}, \dot{0}^{\mathcal{N}}, \dot{1}^{\mathcal{N}}),$$

wobei $\dot{+}^{\mathcal{N}}$ und $\dot{\times}^{\mathcal{N}}$ die natürliche Addition bzw. Multiplikation auf $\mathbb{N}$ sind und $\dot{0}^{\mathcal{N}} := 0$, $\dot{1}^{\mathcal{N}} := 1$. Entsprechend können wir σ_{Ar}-Strukturen $\mathcal{Z}, \mathcal{Q}, \mathcal{R}$ mit Universum $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$ definieren.

Beispiel 9.8 (Graphen und Bäume). Sei $\sigma_{\mathrm{Graph}} := \{\dot{E}\}$, wobei $\dot{E}$ ein 2-stelliges Relationssymbol ist. Jeder gerichtete Graph bzw. gerichtete Baum (V, E) lässt sich als σ_{Graph}-Struktur $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ mit Universum $A := V$ und Relation $\dot{E}^{\mathfrak{A}} := E$ auffassen.

lineare Ordnung
totale Ordnung

Definition 9.9. Eine 2-stellige Ordnung E heißt **lineare Ordnung** oder **totale Ordnung** für ein Universum A, falls E reflexiv und transitiv ist (vgl. Definition 7.20). Zudem gilt $E(x, y)$ oder $E(y, x)$ für je zwei verschiedene Elemente $x, y \in A$.

Beispiel 9.10 (Lineare Ordnungen). Sei $\sigma_{\text{Ord}} := \{\dot{\leqslant}\}$ eine Signatur und sei $\leqslant$ eine lineare Ordnung auf dem Universum A. Dann erhalten wir die σ_{Ord}-Struktur

$$\mathfrak{A} \;=\; (A, \dot{\leqslant}^{\mathfrak{A}})$$

mit $\dot{\leqslant}^{\mathfrak{A}} := \leqslant$.

Frage: Wann sind zwei σ-Strukturen $\mathfrak{A}$ und $\mathfrak{B}$ „prinzipiell gleich“ (Fachbegriff: isomorph)?

Antwort: Falls $\mathfrak{B}$ aus $\mathfrak{A}$ entsteht, indem man die Elemente des Universums von $\mathfrak{A}$ umbenennt.

Analog zum Begriff der Isomorphie von Graphen (Definition 5.21) wird dies durch folgende Definition präzisiert:

Definition 9.11. Sei σ eine Signatur und seien $\mathfrak{A}$ und $\mathfrak{B}$ zwei σ-Strukturen. $\mathfrak{A}$ und $\mathfrak{B}$ heißen **isomorph** (kurz: $\mathfrak{A} \cong \mathfrak{B}$, in Worten: $\mathfrak{A}$ ist isomorph zu $\mathfrak{B}$), falls es eine **bijektive** Abbildung $\pi\colon A \to B$ gibt, für die gilt: isomorph

- für jedes Relationssymbol $\dot{R} \in \sigma$, für $r := \text{ar}(\dot{R})$ und für alle r-Tupel $(a_1, \ldots, a_r) \in A^r$ gilt:

$$(a_1, \ldots, a_r) \in \dot{R}^{\mathfrak{A}} \iff \big(\pi(a_1), \ldots, \pi(a_r)\big) \in \dot{R}^{\mathfrak{B}}.$$

- für jedes Konstantensymbol $\dot{c} \in \sigma$ gilt:

$$\pi(\dot{c}^{\mathfrak{A}}) \;=\; \dot{c}^{\mathfrak{B}}.$$

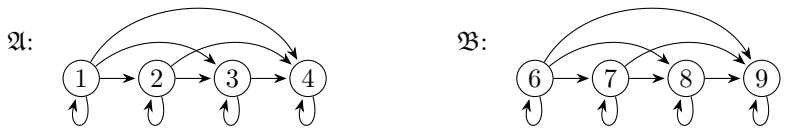
- für jedes Funktionssymbol $\dot{f} \in \sigma$, für $r := \text{ar}(\dot{f})$ und für alle r-Tupel $(a_1, \ldots, a_r) \in A^r$ gilt:

$$\pi\big(\dot{f}^{\mathfrak{A}}(a_1, \ldots, a_r)\big) \;=\; \dot{f}^{\mathfrak{B}}\big(\pi(a_1), \ldots, \pi(a_r)\big).$$

Eine solche Abbildung π wird **Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$** genannt. Isomorphismus

Beispiel 9.12.

(a) Ist $A = \{1, 2, 3, 4\}$, $B = \{6, 7, 8, 9\}$, und sind $\dot{\leqslant}^{\mathfrak{A}}$ und $\dot{\leqslant}^{\mathfrak{B}}$ die natürlichen linearen Ordnungen auf A und B, so sind die beiden σ_{Ord}-Strukturen $\mathfrak{A} = (A, \dot{\leqslant}^{\mathfrak{A}})$ und $\mathfrak{B} = (B, \dot{\leqslant}^{\mathfrak{B}})$ isomorph.
Skizze:



Allgemein gilt: Sind A und B endliche Mengen mit $|A| = |B|$ und sind $\dot{\leqslant}^{\mathfrak{A}}$ und $\dot{\leqslant}^{\mathfrak{B}}$ lineare Ordnungen auf den Universen A und B, so ist $\mathfrak{A} \cong \mathfrak{B}$, und die Abbildung π, die das (bzgl. $\dot{\leqslant}^{\mathfrak{A}}$) kleinste Element in A auf das (bzgl. $\dot{\leqslant}^{\mathfrak{B}}$) kleinste Element von $\mathfrak{B}$ abbildet und, allgemein, für jedes $i \in \{1, \ldots, |A|\}$ das i-kleinste Element in A (bzgl. $\dot{\leqslant}^{\mathfrak{A}}$) auf das i-kleinste Element in B (bzgl. $\dot{\leqslant}^{\mathfrak{B}}$) abbildet, ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$.

(b) Sind $\dot{\leqslant}^{\mathcal{N}}$ und $\dot{\leqslant}^{\mathcal{Z}}$ die natürlichen linearen Ordnungen auf $\mathbb{N}$ und $\mathbb{Z}$, so sind die σ_{Ord}-Strukturen $\mathcal{N} := (\mathbb{N}, \dot{\leqslant}^{\mathcal{N}})$ und $\mathcal{Z} := (\mathbb{Z}, \dot{\leqslant}^{\mathcal{Z}})$ **nicht isomorph** (kurz: $\mathcal{N} \not\cong \mathcal{Z}$).

Skizze:

(c) Sei $\sigma := \{\dot{f}, \dot{c}\}$, wobei $\dot{f}$ ein 2-stelliges Funktionssymbol und $\dot{c}$ ein Konstantensymbol ist. Sei $\mathfrak{A} := (A, \dot{f}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$, wobei

- $A := \mathbb{N}$
- $\dot{f}^{\mathfrak{A}} := \dot{+}^{\mathcal{N}}$ die Addition auf $\mathbb{N}$
- $\dot{c}^{\mathfrak{A}} := \dot{0}^{\mathcal{N}}$ die natürliche Zahl 0 ist

und sei $\mathfrak{B} := (B, \dot{f}^{\mathfrak{B}}, \dot{c}^{\mathfrak{B}})$, wobei

- $B := \{2^n : n \in \mathbb{N}\}$ die Menge aller Zweierpotenzen
- $\dot{f}^{\mathfrak{B}} \colon B \times B \to B$ die Funktion mit

$$\dot{f}^{\mathfrak{B}}(b_1, b_2) := b_1 \cdot b_2, \quad \text{f.a. } b_1, b_2 \in B$$

- $\dot{c}^{\mathfrak{B}} := 1 = 2^0 \in B$.

Dann gilt: $\mathfrak{A} \cong \mathfrak{B}$, und die Abbildung $\pi \colon A \to B$ mit $\pi(n) := 2^n$, f.a. $n \in \mathbb{N}$, ist ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$, denn:

- π ist eine bijektive Abbildung von A nach B.
- Für das Konstantensymbol $\dot{c} \in \sigma$ gilt:

$$\pi(\dot{c}^{\mathfrak{A}}) \overset{\text{Def. } \dot{c}^{\mathfrak{A}}}{=} \pi(0) \overset{\text{Def. } \pi}{=} 2^0 \overset{\text{Def. } \dot{c}^{\mathfrak{B}}}{=} \dot{c}^{\mathfrak{B}}.$$

- Für das Funktionssymbol $\dot{f} \in \sigma$ und für alle $(a_1, a_2) \in A^2$ gilt:

$$\pi\big(\dot{f}^{\mathfrak{A}}(a_1, a_2)\big) \overset{\text{Def. } \dot{f}^{\mathfrak{A}}}{=} \pi(a_1 + a_2) \overset{\text{Def. } \pi}{=} 2^{a_1 + a_2}$$

und

$$\dot{f}^{\mathfrak{B}}\big(\pi(a_1), \pi(a_2)\big) \overset{\text{Def. } \pi}{=} \dot{f}^{\mathfrak{B}}(2^{a_1}, 2^{a_2}) \overset{\text{Def. } \dot{f}^{\mathfrak{B}}}{=} 2^{a_1} \cdot 2^{a_2} = 2^{a_1 + a_2}.$$

Also: $\pi\big(\dot{f}^{\mathfrak{A}}(a_1, a_2)\big) = \dot{f}^{\mathfrak{B}}\big(\pi(a_1), \pi(a_2)\big)$. Somit ist π ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$.

Wir wissen nun, über welche Objekte Formeln der Logik erster Stufe „reden“ können: über σ-Strukturen, wobei σ eine Signatur ist. Als nächstes legen wir die Syntax der Logik erster Stufe fest.

9.3. Syntax der Logik erster Stufe

Die Logik erster Stufe übernimmt, verändert und erweitert die Syntax der Aussagenlogik.

- Was gleich bleibt:
  - Alle Junktoren $\neg, \wedge, \vee, \to, \leftrightarrow$ werden übernommen.

- Was sich verändert:
  - Variablen stehen nicht mehr für „wahre“ oder „falsche“ Aussagen, sondern für Elemente im Universum einer σ-Struktur.
  - Variablen sind keine atomaren Formeln mehr.
- Was neu hinzukommt:
  - Es gibt **Quantoren** $\exists$ (für „es existiert“) und $\forall$ (für „für alle“).
  - Die Relations-, Funktions- und Konstantensymbole der Signatur σ sind Teil des Formelalphabets.
  - Konstanten (und ineinander gesetzte) Funktionen bezeichnen Elemente im Universum einer σ-Struktur.

9.3.1. σ- Terme

Definition 9.13 (Variablen der Logik erster Stufe).
Eine **Individuenvariable** (kurz: **Variable**) hat die Form v_i, für $i \in \mathbb{N}$. Individuenvariable
Die Menge aller Variablen bezeichnen wir mit VAR, d.h. $\text{VAR} = \{v_i : i \in \mathbb{N}\}$. Variable

Was sind σ-Terme? Variablen- oder Konstantensymbole, aber auch Funktionssymbole, in die Variablen- oder Konstantensymbole und wieder Funktionssymbole eingesetzt werden.

Definition 9.14 (Terme der Logik erster Stufe).

(a) Für eine Signatur σ sei $A_{\sigma\text{-Terme}}$ das Alphabet, das aus allen Elementen in VAR, allen Konstanten- und Funktionssymbolen in σ, den Klammern (,) und dem Komma , besteht. $A_{\sigma\text{-Terme}}$

(b) Die Menge T_σ der **σ-Terme** ist die wie folgt rekursiv definierte Teilmenge von $A^*_{\sigma\text{-Terme}}$: σ-Terme

Basisregeln:

- Für jedes Konstantensymbol $\dot{c} \in \sigma$ ist $\dot{c} \in T_\sigma$.
- Für jede Variable $x \in \text{VAR}$ ist $x \in T_\sigma$, kurz: $\text{VAR} \subseteq T_\sigma$.

Rekursive Regeln:

- Für jedes Funktionssymbol $\dot{f} \in \sigma$ und für $r := \text{ar}(\dot{f})$ gilt: Sind $t_1 \in T_\sigma, \ldots, t_r \in T_\sigma$, so ist auch $\dot{f}(t_1, \ldots, t_r) \in T_\sigma$.

Beispiel 9.15. Sei $\sigma = \{\dot{f}, \dot{c}\}$ die Signatur aus Beispiel 9.12((c)), die aus einem 2-stelligen Funktionssymbol $\dot{f}$ und einem Konstantensymbol $\dot{c}$ besteht.

Folgende Worte sind σ-Terme:

$$\dot{c}, \quad v_4, \quad \dot{f}(\dot{c}, \dot{c}), \quad \dot{f}(\dot{c}, v_0), \quad \dot{f}\big(\dot{c}, \dot{f}(\dot{c}, v_0)\big).$$

Folgende Worte sind keine σ-Terme:

$$\mathbf{0}, \quad \dot{f}(\mathbf{0}, \dot{c}), \quad \dot{f}(v_0, \dot{c}, v_1), \quad f^{\mathfrak{A}}(2, 3).$$

9.3.2. Formeln der Logik erster Stufe

Definition 9.16 (Das Alphabet der Logik erster Stufe).

Alphabet A_σ

Sei σ eine Signatur. Das **Alphabet** A_σ der **Logik erster Stufe über** σ besteht aus:

- allen Symbolen in $A_{\sigma\text{-Terme}}$
- allen Relationssymbolen in σ
- den Quantoren $\exists$ (Existenzquantor) und $\forall$ (Allquantor)
- dem Gleichheitssymbol $\doteq$
- den Junktoren $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$.

D.h.

$$A_\sigma \;=\; \textsc{Var} \;\cup\; \sigma \;\cup\; \{\exists, \forall\} \;\cup\; \{\doteq\} \;\cup\; \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\} \;\cup\; \{,\}.$$

Definition 9.17 (Formeln der Logik erster Stufe).

FO[σ]-Formeln

Sei σ eine Signatur. Die Menge FO[σ] aller **Formeln der Logik erster Stufe über der Signatur** σ (kurz: **FO**[σ]**-Formeln**; FO steht für die englische Bezeichnung der Logik erster Stufe: first-order logic) ist die folgendermaßen rekursiv definierte Teilmenge von A_σ^*:

Basisregeln:

- Für alle σ-Terme t_1 und t_2 in T_σ gilt:

$$t_1 \doteq t_2 \;\in\; \text{FO}[\sigma].$$

- Für jedes Relationssymbol $\dot{R} \in \sigma$, für $r := \text{ar}(\dot{R})$ und für alle σ-Terme $t_1, \ldots, t_r$ in T_σ gilt:

$$\dot{R}(t_1, \ldots, t_r) \;\in\; \text{FO}[\sigma].$$

atomare σ-Formeln

Bemerkung: FO[σ]-Formeln der Form $t_1 \doteq t_2$ oder $\dot{R}(t_1, \ldots, t_r)$ heißen **atomare** σ**-Formeln**.

Rekursive Regeln:

- Ist $\varphi \in \text{FO}[\sigma]$, so auch $\neg\varphi \;\in\; \text{FO}[\sigma]$.
- Ist $\varphi \in \text{FO}[\sigma]$ und $\psi \in \text{FO}[\sigma]$, so ist auch
  - $(\varphi \wedge \psi) \;\in\; \text{FO}[\sigma]$
  - $(\varphi \vee \psi) \;\in\; \text{FO}[\sigma]$
  - $(\varphi \rightarrow \psi) \;\in\; \text{FO}[\sigma]$
  - $(\varphi \leftrightarrow \psi) \;\in\; \text{FO}[\sigma]$.
- Ist $\varphi \in \text{FO}[\sigma]$ und ist $x \in \textsc{Var}$, so ist auch
  - $\exists x\, \varphi \;\in\; \text{FO}[\sigma]$
  - $\forall x\, \varphi \;\in\; \text{FO}[\sigma]$.

Beispiel 9.18.

(a) Sei $\sigma = \{\dot{f}, \dot{c}\}$ die Signatur aus Beispiel 9.12((c)), die aus einem 2-stelligen Funktionssymbol $\dot{f}$ und einem Konstantensymbol $\dot{c}$ besteht.

Folgende Worte aus A_σ^* sind FO[σ]-Formeln:

- $\dot{f}(v_0, v_1) \doteq \dot{c}$ (atomare σ-Formel)
- $\forall v_2\, \dot{f}(v_2, \dot{c}) \doteq v_2$
- $\neg\exists v_3 \big(\dot{f}(v_3, v_3) \doteq v_3 \wedge \neg v_3 \doteq \dot{c}\big)$

Folgende Worte sind **keine** FO[σ]-Formeln:

- $\big(\dot{f}(v_0, v_1) \doteq \dot{c}\big)$
- $\Big(\forall v_2 \big(\dot{f}(v_2, \dot{c}) \doteq v_2\big)\Big)$
- $\exists \dot{c}\, \dot{f}(v_0, \dot{c}) \doteq v_0$

(b) Sei $\sigma_{\text{Graph}} = \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol besteht. Folgendes ist eine FO[σ_{Graph}]-Formel:

$$\forall v_0 \forall v_1 \Big(\big(\dot{E}(v_0, v_1) \wedge \dot{E}(v_1, v_0)\big) \to v_0 = v_1\Big)$$

Intuition zur Semantik: In einem Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ sagt diese Formel Folgendes aus:

„für alle Knoten $a_0 \in A$ und
für alle Knoten $a_1 \in A$ gilt:
falls $(a_0, a_1) \in \dot{E}^{\mathfrak{A}}$ und $(a_1, a_0) \in \dot{E}^{\mathfrak{A}}$, dann ist $a_0 = a_1$“

Die Formel sagt in einem Graph $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ also gerade aus, dass die Kantenrelation $\dot{E}^{\mathfrak{A}}$ „antisymmetrisch“ ist: Jede Kante (a_0, a_1) schließt die Kante in der Gegenrichtung aus. D.h.: Ein Graph $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ **erfüllt** die Formel genau dann, wenn die Kantenrelation $\dot{E}^{\mathfrak{A}}$ antisymmetrisch ist.

Notation 9.19.

- Statt mit $v_0, v_1, v_2, \ldots$ bezeichnen wir Variablen oft auch mit $x, y, z, \ldots$ oder mit Varianten wie $x', y_1, y_2, \ldots$.
- Für gewisse 2-stellige Relationssymbole wie z.B. $\dot{\leqslant} \in \sigma_{\text{Ord}}$ verwenden wir **Infix- statt Präfixschreibweise** und setzen Klammern dabei auf natürliche Weise, um die eindeutige Lesbarkeit zu gewährleisten.

  Beispiel:
  An Stelle der (formal korrekten) atomaren Formel $\dot{\leqslant}(x, y)$ schreiben wir $x \dot{\leqslant} y$.

Wir wissen nun, welche Zeichenketten (über dem Alphabet A_σ) **FO[σ]-Formeln** genannt werden.

9.4. Semantik der Logik erster Stufe

Was soll die Semantik leisten? Wir betrachten zunächst einige Beispiele, um ein intuitives Verständnis für die Anforderungen an die Semantik zu erlangen.

Beispiel 9.20 (gerichtete Graphen).
Sei $\sigma_{\text{Graph}} = \{\dot{E}\}$, wobei $\dot{E}$ ein 2-stelliges Relationssymbol ist.

(a) Die FO[σ_{Graph}]-Formel

$$\varphi \quad := \quad \forall x \forall y \big(\dot{E}(x,y) \to \dot{E}(y,x) \big)$$

besagt intuitiv:

> „Für alle Knoten x und für alle Knoten y gilt: Falls es eine Kante von x nach y gibt, so gibt es auch eine Kante von y nach x."

Für jeden Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ sollte daher gelten:

$$\mathfrak{A} \text{ erfüllt } \varphi \iff \dot{E}^{\mathfrak{A}} \text{ ist symmetrisch.}$$

Umgangssprachlich sagen wir auch: „Die Formel φ sagt **in einem Graphen $\mathfrak{A}$ aus, dass** dessen Kantenrelation symmetrisch ist."

(b) Die folgende FO[σ_{Graph}]-Formel drückt aus, dass es von Knoten x zu Knoten y einen Weg der Länge 3 gibt:

$$\varphi(x,y) \quad := \quad \exists z_1 \exists z_2 \Big(\big(\dot{E}(x,z_1) \wedge \dot{E}(z_1,z_2) \big) \wedge \dot{E}(z_2,y) \Big).$$

(c) Die FO[σ_{Graph}]-Formel

$$\forall x \forall y \exists z_1 \exists z_2 \Big(\big(\dot{E}(x,z_1) \wedge \dot{E}(z_1,z_2) \big) \wedge \dot{E}(z_2,y) \Big)$$

sagt in einem Graphen $\mathfrak{A}$ aus, dass es zwischen je 2 Knoten einen Weg der Länge 3 gibt.

Beispiel 9.21 (Verwandtschaftsbeziehungen). Um Verwandtschaftsbeziehungen zu modellieren, können wir die Signatur σ benutzen, die aus den folgenden Symbolen besteht:

- 1-stellige Funktionssymbole $\dot{Vater}$, $\dot{Mutter}$

  (Bedeutung: $x \doteq \dot{Vater}(y)$ besagt „x ist der Vater von y".)

- 2-stellige Relationssymbole $\dot{Geschwister}$, $\dot{Vorfahr}$

  (Bedeutung: $\dot{Geschwister}(x,y)$ besagt, dass x und y Geschwister sind; $\dot{Vorfahr}(x,y)$ besagt, dass x ein Vorfahr von y ist.)

Generelles Wissen über Verwandtschaftsbeziehungen lässt sich durch Formeln der Logik erster Stufe repräsentieren, beispielsweise:

- „Personen mit gleichem Vater und gleicher Mutter sind Geschwister":

$$\forall x \forall y \Big(\big(\dot{Vater}(x) \doteq \dot{Vater}(y) \wedge \dot{Mutter}(x) \doteq \dot{Mutter}(y) \big) \to \dot{Geschwister}(x,y) \Big).$$

- „Eltern sind gerade die unmittelbaren Vorfahren":

$$\forall x \forall y \Big(\big(x \doteq \dot{Vater}(y) \vee x \doteq \dot{Mutter}(y) \big) \leftrightarrow \Big(\dot{Vorfahr}(x,y) \wedge \neg \exists z \big(\dot{Vorfahr}(x,z) \wedge \dot{Vorfahr}(z,y) \big) \Big) \Big).$$

- „Die Relation $\dot{Vorfahr}$ ist transitiv“:

$$\forall x \forall y \forall z \Big(\big(\dot{Vorfahr}(x,y) \wedge \dot{Vorfahr}(y,z) \big) \rightarrow \dot{Vorfahr}(x,z) \Big).$$

- Die folgende Formel $\varphi(x,y)$ besagt, dass x Tante oder Onkel von y ist:

$$\varphi(x,y) \;:=\; \exists z \Big(\dot{Geschwister}(x,z) \wedge \big(z \doteq \dot{Vater}(y) \vee z \doteq \dot{Mutter}(y) \big) \Big).$$

- Die folgende Formel $\psi(x)$ besagt, dass x Vater von genau 2 Kindern ist:

$$\psi(x) \;:=\; \exists y_1 \exists y_2 \Bigg(\Big(\big(x \doteq \dot{Vater}(y_1) \wedge x \doteq \dot{Vater}(y_2) \big) \wedge \neg y_1 \doteq y_2 \Big) \wedge \forall z \big(x \doteq \dot{Vater}(z) \rightarrow (z \doteq y_1 \vee z \doteq y_2) \big) \Bigg).$$

9.4.1. Belegungen und Interpretationen

Welche Variablen einer Formel werden durch einen Quantor „gebunden“, welche Variablen sind „frei“?

Notation 9.22.

(a) Eine Formel ψ ist **Teilformel** einer Formel φ, wenn ψ als Teil-Wort in φ vorkommt. Teilformel
Beispiel: $\psi := \dot{f}(v_0, v_1) \doteq \dot{c}$ ist Teilformel der Formel $\exists v_0 \, \dot{f}(v_0, v_1) \doteq \dot{c}$.

(b) Ist φ eine Formel und x eine Variable, so heißt jedes Vorkommen von x in einer Teilformel der Form $\exists x \, \psi$ oder $\forall x \, \psi$ **gebunden**. Jedes andere Vorkommen von x in φ heißt **frei**. gebunden frei
Beispiel:

$$\varphi \;:=\; \big(\dot{f}(v_0, \dot{c}) \doteq v_3 \wedge \exists v_0 \dot{f}(v_0, v_1) \doteq \dot{c} \big)$$

freies Vorkommen von v_0 in φ — freies Vorkommen von v_3 in φ — gebundene Vorkommen von v_0 in φ — freies Vorkommen von v_1 in φ

(c) Die Menge frei(φ) aller **freien Variablen** einer FO[σ]-Formel φ besteht aus allen Variablen, die mindestens einmal frei in φ vorkommen. frei(φ) freien Variablen
Beispiele:
- $\text{frei}\big(\dot{f}(v_0, \dot{c}) \doteq v_3 \big) = \{v_0, v_3\}$
- $\text{frei}\big(\exists v_0 \, \dot{f}(v_0, v_1) \doteq \dot{c} \big) = \{v_1\}$
- $\text{frei}\big(\dot{f}(v_0, \dot{c}) \doteq v_3 \wedge \exists v_0 \, \dot{f}(v_0, v_1) \doteq \dot{c} \big) = \{v_0, v_3, v_1\}$

(d) Eine FO[σ]-Formel φ heißt **Satz** (genauer: FO[σ]-Satz), falls sie keine freien Variablen besitzt, d.h. falls frei(φ) = $\emptyset$. Satz

Intuitiv gesprochen sollte eine Struktur nur dann entscheiden, ob eine Formel wahr oder falsch ist, wenn die Formel ein Satz ist. Um freie Variablen „kümmern“ sich Belegungen, die den Variablen bestimmte Elemente des Universums zuweisen.

Definition 9.23 (Belegungen und Interpretationen).

Belegung

(a) Eine **Belegung** in einer σ-Struktur $\mathfrak{A} = (A, \alpha)$ ist eine partielle Funktion β von VAR nach A (d.h. β ordnet jeder Variablen $x \in \mathrm{Def}(\beta)$ ein Element $\beta(x)$ aus dem Universum von $\mathfrak{A}$ zu).

passend zu t

(b) Eine Belegung β ist eine **Belegung für einen σ-Term** t (bzw. **passend zu** t), wenn $\mathrm{Def}(\beta)$ alle in t vorkommenden Variablen enthält.

σ-Interpretation

(c) Eine **σ-Interpretation** ist ein Paar

$$\mathcal{I} \;=\; (\mathfrak{A}, \beta),$$

bestehend aus einer σ-Struktur $\mathfrak{A}$ und einer Belegung β in $\mathfrak{A}$.

(d) Eine σ-Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ ist eine **Interpretation für einen σ-Term** t (bzw. **passend zu** t), wenn β passend zu t ist.

Wir wollen Terme nun in Interpretationen „auswerten“. Die Auswertung von Term t in einer zu t passenden Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ soll dasjenige Element aus A liefern, das man erhält, wenn man die in t vorkommenden Variablen gemäß der Belegung β interpretiert, die in t vorkommenden Konstantensymbole gemäß ihrer Interpretation in $\mathfrak{A}$ belegt, und dann nach und nach den Term t gemäß den in $\mathfrak{A}$ gegebenen Interpretationen der Funktionssymbole berechnet.
Dies wird in der folgenden Definition präzisiert:

Definition 9.24 (Semantik von σ-Termen).
Sei σ eine Signatur. Rekursiv über den Aufbau von T_σ definieren wir eine Funktion $[\![\cdot]\!]^{\cdot}$, die jedem σ-Term $t \in T_\sigma$ und jeder zu t passenden σ-Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ einen Wert $[\![t]\!]^{\mathcal{I}} \in A$ zuordnet:

- Für alle $x \in \mathrm{VAR}$ ist $[\![x]\!]^{\mathcal{I}} \;:=\; \beta(x)$.
- Für alle Konstantensymbole $\dot{c} \in \sigma$ ist $[\![\dot{c}]\!]^{\mathcal{I}} \;:=\; \dot{c}^{\mathfrak{A}}$.
- Für alle Funktionssymbole $\dot{f} \in \sigma$, für $r := \mathrm{ar}(\dot{f})$ und für alle σ-Terme $t_1, \ldots, t_r \in T_\sigma$ gilt:

$$[\![\dot{f}(t_1, \ldots, t_r)]\!]^{\mathcal{I}} \;:=\; \dot{f}^{\mathfrak{A}}([\![t_1]\!]^{\mathcal{I}}, \ldots, [\![t_r]\!]^{\mathcal{I}}).$$

Beispiel 9.25. Sei $\sigma = \{\dot{f}, \dot{c}\}$ und sei $\mathfrak{A} = (A, \dot{f}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$ mit

- $A \;:=\; \mathbb{N}$
- $\dot{f}^{\mathfrak{A}} \;:=\; \dot{+}^{\mathcal{N}}$ (die Addition auf $\mathbb{N}$)
- $\dot{c}^{\mathfrak{A}} \;:=\; \dot{0}^{\mathcal{N}}$ (die natürliche Zahl 0)

wie im Beispiel 9.12((c)). Sei β eine Belegung mit $\beta(v_1) = 1$ und $\beta(v_2) = 7$. Und sei $\mathcal{I} := (\mathfrak{A}, \beta)$.

Sei t der Term $\dot{f}\big(v_2, \dot{f}(v_1, \dot{c})\big)$. Dann gilt:

$$\begin{aligned}
[\![t]\!]^{\mathcal{I}} &= [\![\dot{f}\big(v_2, \dot{f}(v_1, \dot{c})\big)]\!]^{\mathcal{I}} \\
&= \dot{f}^{\mathfrak{A}}\big(\beta(v_2), \dot{f}^{\mathfrak{A}}\big(\beta(v_1), \dot{c}^{\mathfrak{A}}\big)\big) \\
&= \dot{f}^{\mathfrak{A}}\big(7, \dot{f}^{\mathfrak{A}}(1, 0)\big) \\
&= \big(7 + \big(1 + 0\big)\big) \\
&= 8.
\end{aligned}$$

9.4.2. Formale Definition der Semantik

Um die formale Definition der Semantik der Logik erster Stufe angeben zu können, benötigen wir noch folgende Begriffe:

Definition 9.26 (passende Belegungen und Interpretationen).

(a) Eine Belegung β ist eine **Belegung für eine FO[σ]-Formel** φ (bzw. **passend zu** φ), wenn $\text{frei}(\varphi) \subseteq \text{Def}(\beta)$. — passend zu φ

(b) Eine σ-Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ ist eine **Interpretation für eine FO[σ]-Formel** φ (bzw. **passend zu** φ), wenn β passend zu φ ist. — Interpretation für eine FO[σ]-Formel

Notation 9.27.

(a) Ist β eine Belegung in einer σ-Struktur $\mathfrak{A}$, ist $x \in \text{VAR}$ und ist $a \in A$, so sei

$$\beta\tfrac{a}{x}$$

die Belegung mit $\text{Def}(\beta\tfrac{a}{x}) := \text{Def}(\beta) \cup \{x\}$ und

$$\beta\tfrac{a}{x}(x) \;=\; a \qquad \text{und} \qquad \beta\tfrac{a}{x}(y) \;=\; \beta(y) \;\text{ f.a. } y \in \text{Def}(\beta) \setminus \{x\}.$$

(b) Ist $\mathcal{I} = (\mathfrak{A}, \beta)$ eine σ-Interpretation, ist $x \in \text{VAR}$ und ist $a \in A$, so sei

$$\mathcal{I}\tfrac{a}{x} \;:=\; (\mathfrak{A}, \beta\tfrac{a}{x}).$$

Wir können nun (endlich) die formale Semantik der Logik erster Stufe festlegen.

Definition 9.28 (Semantik der Logik erster Stufe).
Sei σ eine Signatur. Rekursiv über den Aufbau von FO[σ] definieren wir eine Funktion $[\![\cdot]\!]^{\cdot}$, die jeder FO[σ]-Formel φ und jeder zu φ passenden Interpretationen $\mathcal{I} = (\mathfrak{A}, \beta)$ einen **Wahrheitswert** (kurz: **Wert**) $[\![\varphi]\!]^{\mathcal{I}} \in \{0, 1\}$ zuordnet: — Wahrheitswert

Rekursionsanfang:

- Für alle σ-Terme t_1 und t_2 in T_σ gilt:

$$[\![t_1 \doteq t_2]\!]^{\mathcal{I}} \;:=\; \begin{cases} 1, & \text{falls } [\![t_1]\!]^{\mathcal{I}} = [\![t_2]\!]^{\mathcal{I}} \\ 0, & \text{sonst.} \end{cases}$$

- Für jedes Relationssymbol $\dot{R} \in \sigma$, für $r := \mathrm{ar}(\dot{R})$ und für alle σ-Terme $t_1, \ldots, t_r$ in T_σ gilt:

$$[\![\dot{R}(t_1, \ldots, t_r)]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls } ([\![t_1]\!]^{\mathcal{I}}, \ldots, [\![t_r]\!]^{\mathcal{I}}) \in \dot{R}^{\mathfrak{A}} \\ 0, & \text{sonst.} \end{cases}$$

Rekursionsschritt:

- Die Semantik der Junktoren $\neg$, $\wedge$, $\vee$, $\rightarrow, \leftrightarrow$ ist wie in der Aussagenlogik definiert, d.h. für alle $\varphi \in \mathrm{FO}[\sigma]$ und $\psi \in \mathrm{FO}[\sigma]$ gilt:

$$[\![\neg\varphi]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 0 \\ 0, & \text{sonst.} \end{cases}$$

$$[\![(\varphi \wedge \psi)]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 1 \text{ und } [\![\psi]\!]^{\mathcal{I}} = 1 \\ 0, & \text{sonst.} \end{cases}$$

$$[\![(\varphi \vee \psi)]\!]^{\mathcal{I}} := \begin{cases} 0, & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 0 \text{ und } [\![\psi]\!]^{\mathcal{I}} = 0 \\ 1, & \text{sonst.} \end{cases}$$

$$[\![(\varphi \rightarrow \psi)]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 0 \text{ oder } [\![\psi]\!]^{\mathcal{I}} = 1 \\ 0, & \text{sonst.} \end{cases}$$

$$[\![(\varphi \leftrightarrow \psi)]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = [\![\psi]\!]^{\mathcal{I}} \\ 0, & \text{sonst.} \end{cases}$$

- Ist $\varphi \in \mathrm{FO}[\sigma]$ und ist $x \in \mathrm{VAR}$, so ist
  - $[\![\exists x\, \varphi]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls es (mindestens) ein } a \in A \text{ gibt, so dass } [\![\varphi]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\ 0, & \text{sonst.} \end{cases}$
  - $[\![\forall x\, \varphi]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls für alle } a \in A \text{ gilt: } [\![\varphi]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\ 0, & \text{sonst.} \end{cases}$

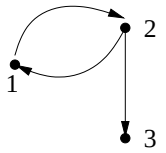
Beispiel 9.29. Sei $\sigma_{\mathrm{Graph}} = \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol $\dot{E}$ besteht. Betrachte die $\mathrm{FO}[\sigma_{\mathrm{Graph}}]$-Formel

$$\varphi := \forall x \forall y \big(\dot{E}(x, y) \rightarrow \dot{E}(y, x)\big).$$

Für jede zu φ passende σ_{Graph}-Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ gilt:

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{I}} = 1 &\iff \text{für alle } a \in A \text{ gilt: } [\![\forall y\big(\dot{E}(x,y) \to \dot{E}(y,x)\big)]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\
&\iff \text{für alle } a \in A \text{ gilt:} \\
&\qquad \text{für alle } b \in A \text{ gilt: } [\![\big(\dot{E}(x,y) \to \dot{E}(y,x)\big)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1 \\
&\iff \text{für alle } a \in A \text{ und alle } b \in A \text{ gilt:} \\
&\qquad \text{falls } [\![\dot{E}(x,y)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1, \text{ so auch } [\![\dot{E}(y,x)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1 \\
&\iff \text{für alle } a \in A \text{ und alle } b \in A \text{ gilt:} \\
&\qquad \text{falls } (a,b) \in \dot{E}^{\mathfrak{A}}, \text{ so auch } (b,a) \in \dot{E}^{\mathfrak{A}} \\
&\iff \dot{E}^{\mathfrak{A}} \text{ ist symmetrisch, (vgl. Def. 7.20).}
\end{aligned}$$

Sei nun $\mathfrak{A}$ die σ_{Graph}-Struktur, die den gerichteten Graphen



repräsentiert, d.h. $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ mit $A = \{1,2,3\}$ und $\dot{E}^{\mathfrak{A}} = \{(1,2),(2,1),(2,3)\}$. Sei β die Belegung mit leerem Definitionsbereich und sei $\mathcal{I} := (\mathfrak{A}, \beta)$. Dann gilt: Da in unserem konkreten Graphen $\mathfrak{A}$ für $a = 2$ und $b = 3$ gilt: $(a,b) \in \dot{E}^{\mathfrak{A}}$, aber $(b,a) \notin \dot{E}^{\mathfrak{A}}$, ist $\dot{E}^{\mathfrak{A}}$ nicht symmetrisch, und daher ist hier $[\![\varphi]\!]^{\mathcal{I}} = 0$.

Notation 9.30. Sei σ eine Signatur und sei φ eine FO[σ]-Formel.

(a) Sei $\mathcal{I} = (\mathfrak{A}, \beta)$ eine zu φ passende σ-Interpretation.

Wir sagen „**$\mathcal{I}$ erfüllt φ**“ (bzw. „**$\mathcal{I}$ ist ein Modell von φ**“, kurz: $\mathcal{I} \models \varphi$), falls $[\![\varphi]\!]^{\mathcal{I}} = 1$. $\mathcal{I} \models \varphi$

Wir sagen „**$\mathcal{I}$ erfüllt φ nicht**“ (bzw. „**$\mathcal{I}$ ist kein Modell von φ**“, kurz: $\mathcal{I} \not\models \varphi$), falls $[\![\varphi]\!]^{\mathcal{I}} = 0$. $\mathcal{I} \not\models \varphi$

(b) Ist φ ein **Satz** (d.h. φ hat keine freien Variablen), so hängt die Tatsache, ob φ von einer Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ erfüllt wird, nur von der Struktur $\mathfrak{A}$ und nicht von der Belegung β ab. An Stelle von „$\mathcal{I} \models \varphi$“ schreiben wir dann kurz „$\mathfrak{A} \models \varphi$“ und sagen „die σ-Struktur $\mathfrak{A}$ erfüllt den Satz φ.“

9.5. Ein Anwendungsbereich der Logik erster Stufe: Datenbanken

Relationale Datenbanken bestehen aus **Tabellen**, die sich als Relationen auffassen lassen. Datenbanken lassen sich daher als **Strukturen** über einer passenden Signatur auffassen. Die in der Praxis gebräuchlichste Datenbankanfragesprache ist **SQL**. Der „Kern“ von SQL basiert auf der Logik erster Stufe, die in der Datenbankterminologie oft auch „relationaler Kalkül“ (engl.: „relational calculus“) bezeichnet wird.

Zur Illustration von Anfragen verwenden wir eine kleine Datenbank mit Kinodaten, bestehend aus:

- einer Tabelle *Orte*, die Informationen über Kinos (Kino, Adresse, Telefonnummer) enthält,
- einer Tabelle *Filme*, die Informationen über Filme enthält (Titel, Regie, Schauspieler).
- eine Tabelle *Programm*, die Informationen zum aktuellen Kinoprogramm enthält (Kino, Titel, Zeit).

Orte-Tabelle:

Kino	Adresse	Telefon
Babylon	Dresdner Str. 2	61609693
Casablanca	Friedenstr. 12	6775752
Cinestar Cubix Alexanderplatz	Rathausstr. 1	2576110
Die Kurbel	Giesebrechtstr. 4	88915998
Filmpalast Berlin	Kurfürstendamm 225	8838551
International	Karl-Marx-Allee 33	24756011
Kino in der Kulturbrauerei	Schönhauser Allee 36	44354422
Moviemento	Kottbusser Damm 22	6924785

Filme-Tabelle:

Titel	Regie	Schauspieler
Capote	Bennet Miller	Philip Seymour Hoffman
Capote	Bennet Miller	Catherine Keener
Das Leben der Anderen	F. Henkel von Donnersmarck	Martina Gedeck
Das Leben der Anderen	F. Henkel von Donnersmarck	Ulrich Tukur
Der ewige Gärtner	Fernando Meirelles	Ralph Fiennes
Der ewige Gärtner	Fernando Meirelles	Rachel Weisz
Good Night and Good Luck	George Clooney	David Strathairn
Good Night and Good Luck	George Clooney	Patricia Clarkson
Knallhart	Detlev Buck	Jenny Elvers
Knallhart	Detlev Buck	Jan Henrik Stahlberg
Raumpatrouille Orion – Rücksturz ins Kino	Michael Braun	Dietmar Schönherr
Raumpatrouille Orion – Rücksturz ins Kino	Theo Mezger	Dietmar Schönherr
Raumpatrouille Orion – Rücksturz ins Kino	Michael Braun	Eva Pflug
Raumpatrouille Orion – Rücksturz ins Kino	Theo Mezger	Eva Pflug
Raumpatrouille Orion – Rücksturz ins Kino	Michael Braun	Wolfgang Völz
Raumpatrouille Orion – Rücksturz ins Kino	Theo Mezger	Wolfgang Völz
Requiem	Hans-Christian Schmid	Sandra Hüller
Sommer vorm Balkon	Andreas Dresen	Nadja Uhl
Sommer vorm Balkon	Andreas Dresen	Inka Friedrich
Sommer vorm Balkon	Andreas Dresen	Andreas Schmidt
Syriana	Stephen Gaghan	George Clooney
Syriana	Stephen Gaghan	Matt Damon
V wie Vendetta	James McTeigue	Natalie Portman
Walk the Line	James Mangold	Joaquin Phoenix
Walk the Line	James Mangold	Reese Witherspoon

Programm-Tabelle:

Kino	Titel	Zeit
Babylon	Capote	17:00
Babylon	Capote	19:30
Kino in der Kulturbrauerei	Capote	17:30
Kino in der Kulturbrauerei	Capote	20:15
International	Das Leben der Anderen	14:30
International	Das Leben der Anderen	17:30
International	Das Leben der Anderen	20:30
Filmpalast Berlin	Good Night and Good Luck	15:30
Filmpalast Berlin	Good Night and Good Luck	17:45
Filmpalast Berlin	Good Night and Good Luck	20:00
Kino in der Kulturbrauerei	Good Night and Good Luck	18:00
Kino in der Kulturbrauerei	Good Night and Good Luck	20:00
Kino in der Kulturbrauerei	Good Night and Good Luck	22:45
Babylon	Sommer vorm Balkon	21:45
Kino in der Kulturbrauerei	Sommer vorm Balkon	21:45
Filmmuseum Potsdam	Raumpatrouille Orion – Rücksturz ins Kino	22:00

Für eine geeignete Signatur σ_{Kino} können wir diese Datenbank durch eine σ_{Kino}-Struktur $\mathfrak{A}_{\text{Kino}}$ folgendermaßen modellieren: Die Signatur σ_{Kino} besteht aus:

- einem 3-stelligen Relationssymbol $\dot{Orte}$
- einem 3-stelligen Relationssymbol $\dot{Filme}$
- einem 3-stelligen Relationssymbol $\dot{Programm}$
- Konstantensymbolen '$\dot{c}$', die allen potentiellen Einträgen c der Datenbank entsprechen, also '$\dot{Babylon}$', '$\dot{Casablanca}$', ..., '$\dot{Capote}$', '$\dot{Das\ Leben\ der\ Anderen}$', ... usw., aber auch z.B. '$\dot{Stephen\ Spielberg}$' oder '$\dot{Lola\ rennt}$'. D.h.: Für jedes Wort c über dem ASCII-Alphabet gibt es ein Konstantensymbol '$\dot{c}$'.

Die σ_{Kino}-Struktur $\mathfrak{A}_{\text{Kino}}$ hat als Universum die Menge aller Worte über dem ASCII-Alphabet, d.h.

$$A_{\text{Kino}} := \text{ASCII}^*,$$

die 3-stelligen Relationen

$$\begin{aligned}
\dot{Orte}^{\mathfrak{A}_{\text{Kino}}} &:= \{(\text{Babylon}, \text{Dresdner Str. } 2, 61609693),\\
&\quad (\text{Casablanca}, \text{Friedenstr. } 12, 6775752),\\
&\quad \ldots,\\
&\quad (\text{Moviemento}, \text{Kottbusser Damm } 22, 6924785)\},\\
\dot{Filme}^{\mathfrak{A}_{\text{Kino}}} &:= \{(\text{Capote}, \text{Bennet Miller}, \text{Philip Seymour Hoffman}),\\
&\quad (\text{Capote}, \text{Bennet Miller}, \text{Catherine Keener}),\\
&\quad \ldots,\\
&\quad (\text{Walk the Line}, \text{James Mangold}, \text{Reese Witherspoon})\},\\
\dot{Programm}^{\mathfrak{A}_{\text{Kino}}} &:= \{(\text{Babylon}, \text{Capote}, 17{:}00),\\
&\quad (\text{Babylon}, \text{Capote}, 19{:}30),\\
&\quad (\text{Kino in der Kulturbrauerei}, \text{Capote}, 17{:}30),\\
&\quad \ldots\}
\end{aligned}$$

sowie für jedes in σ_{Kino} vorkommende Konstantensymbol '$\dot{c}$' die Konstante '$\dot{c}$'$^{\mathfrak{A}_{\text{Kino}}} := c$.
Zum Beispiel:

$$\begin{aligned}
&\text{'}\dot{Babylon}\text{'}^{\mathfrak{A}_{\text{Kino}}} = \text{Babylon},\\
&\text{'}\dot{Capote}\text{'}^{\mathfrak{A}_{\text{Kino}}} = \text{Capote},\\
&\text{'}\dot{George}\ \dot{Clooney}\text{'}^{\mathfrak{A}_{\text{Kino}}} = \text{George Clooney}.
\end{aligned}$$

Anfragen an die Kinodatenbank lassen sich auf unterschiedliche Art formulieren:

Beispiel 9.31. (a) Eine Anfrage an unsere Kinodatenbank:

„Gib die Titel aller Filme aus, die um 17:30 Uhr laufen."

In der Datenbankanfragesprache SQL lässt sich dies folgendermaßen formulieren:

```
SELECT  Titel
FROM    Programm
WHERE   Zeit = '17:30'
```

Dieselbe Anfrage lässt sich auch durch die folgende Formel der Logik erster Stufe beschreiben:

$$\varphi_{\text{Filme um 17:30 Uhr}}(x_{\text{T}}) := \exists x_{\text{K}}\, \dot{Programm}(x_{\text{K}}, x_{\text{T}}, \text{'}\dot{17}\dot{:}\dot{30}\text{'}).$$

(b) Die Anfrage

„Gib die Titel aller Filme aus, in denen George Clooney mitspielt oder Regie führt."

lässt sich in Logik erster Stufe wie folgt beschreiben:

$$\varphi_{\text{Filme mit George Clooney}}(x_{\text{T}}) := \big(\exists x_{\text{R}}\, \dot{Filme}(x_{\text{T}}, x_{\text{R}}, \text{'}\dot{George}\ \dot{Clooney}\text{'}) \vee \exists x_{\text{S}}\, \dot{Filme}(x_{\text{T}}, \text{'}\dot{George}\ \dot{Clooney}\text{'}, x_{\text{S}})\big).$$

Notation 9.32. Sei σ eine Signatur und seien $x_1, \ldots, x_n$ Variablen.

- Die Notation $\varphi(x_1, \ldots, x_n)$ deutet an, dass φ eine FO[σ]-Formel mit frei(φ) = $\{x_1, \ldots, x_n\}$ ist, d.h. dass $x_1, \ldots, x_n$ diejenigen Variablen sind, die in φ frei vorkommen.
- Ist $\varphi(x_1, \ldots, x_n)$ eine FO[σ]-Formel, ist $\mathfrak{A}$ eine σ-Struktur und sind $a_1, \ldots, a_n$ Elemente im Universum von $\mathfrak{A}$, so schreiben wir

  $$\mathfrak{A} \models \varphi[a_1, \ldots, a_n],$$

  um auszudrücken, dass für die Belegung $\beta\colon \{x_1, \ldots, x_n\} \to A$ mit $\beta(x_1) = a_1, \ldots, \beta(x_n) = a_n$ gilt:

  $$(\mathfrak{A}, \beta) \models \varphi.$$

Definition 9.33. Sei σ eine Signatur, $\varphi(x_1, \ldots, x_n)$ eine FO[σ]-Formel und $\mathfrak{A}$ eine σ-Struktur. Die von φ in $\mathfrak{A}$ definierte n-stellige Relation ist

$$\varphi(\mathfrak{A}) \quad := \quad \big\{ (a_1, \ldots, a_n) \in A^n \ : \ \mathfrak{A} \models \varphi[a_1, \ldots, a_n] \big\}.$$

Beispiel 9.34. Die FO[σ_{Kino}]-Formel $\varphi_{\text{Filme um 17:30}}(x_{\text{T}})$ aus Beispiel 9.31 definiert in unserer Kinodatenbank $\mathfrak{A}_{\text{Kino}}$ die 1-stellige Relation:

$$\varphi_{\text{Filme um 17:30}}(\mathfrak{A}_{\text{Kino}}) \ = \ \{ \text{ (Capote),} \\ \text{(Das Leben der Anderen) }\}.$$

Darstellung als Tabelle:

Filme um 17:30 Uhr:

Titel
Capote
Das Leben der Anderen

Beispiel 9.35. (a) Die Anfrage

> „Gib Name und Adresse aller Kinos aus, in denen ein Film läuft, in dem George Clooney mitspielt oder Regie führt."

lässt sich folgendermaßen formulieren:

In SQL:

```
SELECT  Orte.Kino, Orte.Adresse
FROM    Orte, Filme, Programm
WHERE   Orte.Kino = Programm.Kino AND
        Filme.Titel = Programm.Titel AND
        (Filme.Schauspieler = 'George Clooney' OR
         Filme.Regie = 'George Clooney')
```

In Logik erster Stufe:

$$\varphi_{\text{Kinos mit George Clooney}}(x_{\text{K}}, x_{\text{A}}) \ :=$$
$$\Big(\exists x_{\text{Tel}}\, \dot{Orte}(x_{\text{K}}, x_{\text{A}}, x_{\text{Tel}}) \ \wedge$$
$$\exists x_{\text{T}} \exists x_{\text{Z}} \, \big(\dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \ \wedge$$
$$\big(\exists x_{\text{R}}\, \dot{Filme}(x_{\text{T}}, x_{\text{R}}, \text{'George } \dot{\text{Clooney'}}) \ \vee \ \exists x_{\text{S}}\, \dot{Filme}(x_{\text{T}}, \text{'George } \dot{\text{Clooney'}}, x_{\text{S}}) \big) \big) \Big)$$

In unserer konkreten Kinodatenbank $\mathfrak{A}_{\text{Kino}}$ liefert diese Formel die 2-stellige Relation:

$$\varphi_{\text{Kinos mit George Clooney}}(\mathfrak{A}_{\text{Kino}}) = \{ \big(\text{Filmpalast Berlin, Kurfürstendamm 225}\big), \big(\text{Kino in der Kulturbrauerei, Schönhauser Allee 36}\big) \}.$$

Darstellung als Tabelle:

Kinos mit George Clooney:

Kino	Adresse
Filmpalast Berlin	Kurfürstendamm 225
Kino in der Kulturbrauerei	Schönhauser Allee 36

(b) Die Anfrage

„Gib die Titel aller Filme aus, in denen nur Schauspieler mitspielen, die schon mal mit Stephen Spielberg zusammengearbeitet haben."

lässt sich in Logik erster Stufe wie folgt formulieren:

$$\varphi_{\text{Filme mit Spielberg-Schauspielern}}(x_{\text{T}}) :=$$
$$\exists x_R \exists x_S \Big(\dot{Filme}(x_T, x_R, x_S) \wedge \forall y_S \big(\dot{Filme}(x_T, x_R, y_S) \rightarrow \exists z_T \dot{Filme}(z_T, \dot{\text{'Stephen Spielberg'}}, y_S)\big)\Big)$$

Details zum Thema Datenbanken und Datenbankanfragesprachen können Sie in den Vorlesungen „Datenbanksysteme I und II" und „Logik und Datenbanken" kennenlernen.

9.6. Erfüllbarkeit, Allgemeingültigkeit, Folgerung und Äquivalenz

Definition 9.36. Sei σ eine Signatur und sei φ eine FO[σ]-Formel.

erfüllbar (a) φ heißt **erfüllbar**, wenn es (mindestens) eine zu φ passende σ-Interpretation $\mathcal{I}$ gibt, die φ erfüllt.

unerfüllbar (b) φ heißt **unerfüllbar**, wenn φ nicht erfüllbar ist.

allgemeingültig (c) φ heißt **allgemeingültig**, wenn jede zu φ passende σ-Interpretation φ erfüllt.

Beispiel 9.37. Sei $\sigma_{\text{Graph}} := \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol $\dot{E}$ besteht. Die FO[σ_{Graph}]-Formel $\varphi := \forall y\, \dot{E}(x, y)$ ist erfüllbar, aber nicht allgemeingültig, denn: Sei $\mathfrak{A} := (A, \dot{E}^{\mathfrak{A}})$ der gerichtete Graph

und sei β die Belegung mit $\beta(x) = 1$. Dann erfüllt die Interpretation $(\mathfrak{A}, \beta)$ die Formel φ. Somit ist φ erfüllbar.

Andererseits gilt für den Graphen $\mathfrak{B} := (B, \dot{E}^{\mathfrak{B}})$

und die Belegung β mit $\beta(x) = 1$, dass die zu φ passende σ-Interpretation $\mathcal{I} := (\mathfrak{B}, \beta)$ die Formel φ **nicht** erfüllt (d.h. $[\![\varphi]\!]^{\mathcal{I}} = 0$), denn:

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{I}} = 1 &\iff \text{Für jedes } b \in B \text{ gilt: } [\![\dot{E}(x,y)]\!]^{\mathcal{I}\frac{b}{y}} = 1 \\
&\iff \text{Für jedes } b \in B \text{ gilt: } \big(\beta\tfrac{b}{y}(x), \beta\tfrac{b}{y}(y)\big) \in \dot{E}^{\mathfrak{B}} \\
&\iff \text{Für jedes } b \in B \text{ gilt: } (1,b) \in \dot{E}^{\mathfrak{B}}.
\end{aligned}$$

Aber für $b := 1$ gilt: $(1,1) \notin \dot{E}^{\mathfrak{B}}$, und daher ist $[\![\varphi]\!]^{\mathcal{I}} = 0$. $\mathcal{I}$ ist also eine zu φ passende σ-Interpretation, die φ **nicht** erfüllt. Somit ist φ nicht allgemeingültig.

Beobachtung 9.38. Für alle Formeln φ der Logik erster Stufe gilt:

(a) φ ist allgemeingültig $\iff$ $\neg\varphi$ ist unerfüllbar.

(b) φ ist erfüllbar $\iff$ $\neg\varphi$ ist nicht allgemeingültig.

Beweis: Übung. □

Definition 9.39 (semantische Folgerung).
Sei σ eine Signatur und seien Φ eine Menge von FO[σ]-Formel und ψ eine FO[σ]-Formel. Wir sagen ψ **folgt aus** Φ (kurz: $\Phi \models \psi$, „Φ impliziert ψ“), falls für jede zu den Formeln in Φ und ψ passende Interpretation $\mathcal{I}$ gilt: ψ folgt aus Φ

$$\text{Falls f.a. } \varphi \in \Phi \quad \underbrace{\mathcal{I} \models \varphi}_{\text{d.h. } [\![\varphi]\!]^{\mathcal{I}} = 1} \quad , \quad \text{so auch} \quad \underbrace{\mathcal{I} \models \psi}_{\text{d.h. } [\![\psi]\!]^{\mathcal{I}} = 1}.$$

Definition 9.40 (logische Äquivalenz).
Sei σ eine Signatur. Zwei FO[σ]-Formeln φ und ψ heißen **äquivalent** (kurz: $\varphi \equiv \psi$), wenn für jede zu φ und ψ passende σ-Interpretation $\mathcal{I}$ gilt: äquivalent

$$\mathcal{I} \models \varphi \iff \mathcal{I} \models \psi.$$

Beobachtung 9.41. Sei σ eine Signatur und seien φ und ψ zwei FO[σ]-Formeln. Es gilt:

(a) $\varphi \equiv \psi \iff \varphi \models \psi$ und $\psi \models \varphi$.

(b) $\varphi \equiv \psi \iff (\varphi \leftrightarrow \psi)$ ist allgemeingültig.

(c) $\varphi \models \psi \iff (\varphi \to \psi)$ ist allgemeingültig.

Beweis: Übung. □

9.6.1. Die Addition in den natürlichen Zahlen

Wir definieren die Signatur

$$\sigma^{+}_{\mathrm{Ar}} := \{\dot{+}, \dot{<}, \dot{0}, \dot{1}\},$$

wobei $\dot{+}$ ein 2-stelliges Funktionssymbol, $\dot{<}$ ein 2-stelliges Relationssymbol und $\dot{0}$ und $\dot{1}$ Konstantensymbole sind. In FO[σ^+_{Ar}]-Formeln können wir Aussagen über die Addition in den natürlichen Zahlen machen. Wir betrachten deshalb die σ^+_{Ar}-Struktur

$$\mathcal{N}_+ := (\mathbb{N}, \dot{+}^{\mathcal{N}}, \dot{<}^{\mathcal{N}}, \dot{0}^{\mathcal{N}}, \dot{1}^{\mathcal{N}}),$$

- wobei $\dot{+}^{\mathcal{N}}$ und $\dot{<}^{\mathcal{N}}$ die natürliche Addition bzw. die Kleiner-Relation auf $\mathbb{N}$ sind
- und $\dot{0}^{\mathcal{N}} := 0$, $\dot{1}^{\mathcal{N}} := 1$ gilt.

Frage: Gibt es ein Axiomensystem, aus dem sich genau die Sätze aus FO[σ^+_{Ar}] folgern lassen, die in der σ^+_{Ar}-Struktur $\mathcal{N}_+$ wahr sind?

Wir führen dazu das Axiomensystem der Presburger-Arithmetik ein:

$$\begin{array}{llll} (P1) & S(x) \neq 0, & (P2) & S(x) = S(y) \to x = y, \\ (P3) & x + 0 = x, & (P4) & x + S(y) = S(x + y), \\ (P5) & \neg(x < 0), & (P6) & x < S(y) \leftrightarrow x < y \vee x = y, \\ & & (P7) & x < y \vee x = y \vee y < x, \end{array}$$

Weiterhin wird das Induktionsaxiom

$$\varphi(0) \wedge \forall x(\varphi(x) \to \varphi(S(x))) \to \varphi$$

für jede FO[σ^+_{Ar}] Formel $\varphi(x)$ gefordert.

Satz 9.42. Sei $\mathcal{M}$ eine σ^+_{Ar}-Struktur, die alle Axiome der Presburger-Arithmetik erfüllt. Dann gilt für alle FO[σ^+_{Ar}]-Sätze φ :

$$\mathcal{N}_+ \models \varphi \;\Leftrightarrow\; \mathcal{M} \models \varphi.$$

Die Presburger-Arithmetik axiomatisiert also die Eigenschaften der Addition natürlicher Zahlen. Zusätzlich kann man zeigen, dass sich alle wahren FO[σ^+_{Ar}]-Sätze auch beweisen lassen: Wahrheit und Beweisbarkeit stimmen überein!

Frage: Können wir ein solches Axiomensystem auch finden, wenn wir neben der Addition auch die Multiplikation als Funktionssymbol zulassen, wenn wir also die Signatur

$$\sigma_{\mathrm{Ar}} := \{\dot{+}, \dot{\times}, \dot{0}, \dot{1}\}$$

verwenden?

1. Die erste Antwort ist **klar**, wenn wir alle wahren Aussagen auch als Axiome verwenden.
2. Aber Axiomensysteme sollte man „einfach beschreiben“ können! Zumindest sollte es (ein stets haltendes) Programm geben, das entscheidet, ob eine Formel ein Axiom ist oder nicht.

Gödelscher Unvollständigkeitssatz: Es gibt kein einfach beschreibbares Axiomensystem: Komplexe Realitäten wie die σ_{Ar}-Struktur

Gödelscher Unvollständigkeitssatz

$$\mathcal{N} := (\mathbb{N}, \dot{+}^{\mathcal{N}}, \dot{\times}^{\mathcal{N}}, \dot{0}^{\mathcal{N}}, \dot{1}^{\mathcal{N}})$$

lassen sich nicht beschreiben und lassen sich insbesondere nicht im Rechner modellieren!

9.6.2. Die Zermelo-Fraenkel Mengenlehre

Die Formeln der Mengenlehre benutzen die Signatur $\sigma := \{\dot{\in}\}$ mit dem 2-stelligen Relationssymbol $\dot{\in}$. Die Zermelo-Fraenkel Mengenlehre (siehe Bemerkung 2.15) besteht aus der Menge `ZF` aller Axiome und Axiomenschemata der Zermelo-Fraenkel Mengenlehre. Neben den Formeln der Menge `ZF` interessieren wir uns vor allem für die Folgerungen des Axiomensystems, also die Menge

$$\{\, \varphi \in \mathrm{FO}[\sigma] \,:\, \mathtt{ZF} \models \varphi \,\}.$$

1. Das `Extensionalitätsaxiom`: Zwei Mengen A, B sind genau gleich, wenn sie dieselben Elemente besitzen:
$$A \doteq B \leftrightarrow \forall x\big(x \mathrel{\dot{\in}} A \leftrightarrow x \mathrel{\dot{\in}} B\big).$$

2. Das `Nullmengenaxiom` fordert, dass die leere Menge eine Menge ist:
$$\exists A \forall x \neg(x \mathrel{\dot{\in}} A)$$
Als Folgerung des Extensionalitätsaxioms und des Nullmengenaxioms gibt es genau eine leere Menge, die wir natürlich mit $\emptyset$ bezeichnen.

3. Das `Paarmengenaxiom` fordert, dass für alle Mengen A, B auch $\{A, B\}$ eine Menge ist:
$$\forall A \forall B \exists C \forall x \Big(x \mathrel{\dot{\in}} C \leftrightarrow \big(x \doteq A \vee x \doteq B\big)\Big).$$
Das `Vereinigungsaxiom` besagt, dass mit jeder Menge A von Mengen auch die Vereinigung $\bigcup_{a \in A} a$ aller Elemente von A eine Menge ist:
$$\forall A \exists B \forall x \Big(x \mathrel{\dot{\in}} B \leftrightarrow \exists y\big(y \mathrel{\dot{\in}} A \wedge x \mathrel{\dot{\in}} y\big)\Big).$$
Paarmengenaxiom und Vereinigungsaxiom zusammen garantieren, dass auch die Vereinigung $A_1 \cup A_2$ von zwei Mengen A_1, A_2 eine Menge ist. Dazu bilden wir zuerst die Paarmenge $\{A_1, A_2\}$ und beachten $A_1 \cup A_2 = \bigcup_{a \in \{A_1, A_2\}} a$.

4. Das `Potenzmengenaxiom` fordert, dass die Menge aller Teilmengen einer Menge A eine Menge ist:
$$\forall A \exists B \forall x \Big(x \mathrel{\dot{\in}} B \leftrightarrow \forall y(y \mathrel{\dot{\in}} x \to y \mathrel{\dot{\in}} A)\Big).$$

5. Das `Aussonderungsaxiom` haben häufig benutzt, um neue Mengen aus alten Mengen zu konstruieren. Sei $\psi(x)$ eine beliebige $\mathrm{FO}[\sigma_{\dot{\in}}]$-Formel, in der die Variable B nicht als freie Variable vorkommt. Dann ist $\{x \in A \,:\, \psi(x)\}$ wieder eine Menge:
$$\forall A \exists B \forall x \Big(x \mathrel{\dot{\in}} B \leftrightarrow \big(x \mathrel{\dot{\in}} A \wedge \psi(x)\big)\Big).$$
Jetzt können wir sofort folgern, dass der Durchschnitt $A \cap B$ zweier Mengen A und B eine Menge ist, denn $A \cap B = \{\, x \mathrel{\dot{\in}} A \,:\, \psi(x) \,\}$ mit $\psi(x) := x \mathrel{\dot{\in}} B$.

6. Das `Unendlichkeitsaxiom` fordert die Existenz einer Menge A, die die leere Menge und mit jedem Element y auch die Menge $y \cup \{y\}$ enthält:

$$\exists A\big(\emptyset \dot{\in} A \wedge \forall y(y \dot{\in} A \to y \cup \{y\} \dot{\in} A)\big).$$

Wir kürzen das Unendlichkeitsaxiom durch $\exists A\psi(A)$ ab, $\psi(A)$ fordert also, dass A die leere Menge und mit jedem Element y auch die Menge $y \cup \{y\}$ enthält.

In der Mengenlehre repräsentiert man die Menge $\mathbb{N}$ der natürlichen Zahlen durch die Menge

$$N := \{\, \emptyset,\, \{\emptyset\},\, \{\emptyset, \{\emptyset\}\},\, \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}, \ldots\}.$$

Die Null wird durch die leere Menge (mit 0 Elementen), die Eins durch die Menge $\{\emptyset\}$ (mit 1 Element), die Zwei durch die Menge $\{\emptyset, \{\emptyset\}\}$ (mit 2 Elementen) usw. repräsentiert.

Das Unendlichkeitsaxiom garantiert aber nur, dass eine Obermenge von N existiert. Mit Hilfe des Aussonderungsaxioms können wir folgern, dass die Menge N existiert, denn wenn A eine beliebige Menge ist, deren Existenz vom Unendlichkeitsaxiom garantiert ist, dann folgt

$$N = \{\, x \dot{\in} A \,:\, \forall B\big(\psi(B) \to x \in B\big)\,\},$$

die Menge N ist also die kleinste Menge mit der Eigenschaft $\psi(N)$.

7. Das Fundierungsaxiom fordert, dass jede nichtleere Menge A ein Element B mit $A \cap B = \emptyset$ enthält:

$$\forall A\Big(\neg A \doteq \emptyset \to \exists B\big(B \dot{\in} A \wedge \neg\exists C(C \dot{\in} A \wedge C \dot{\in} B)\big)\Big)$$

Das Fundierungsaxiom verhindert, dass es eine Menge $A = \{\, x_1, \ldots, x_i, x_{i+1}, \ldots\,\}$ von ineinander enthaltenen Mengen gibt, dass also $x_{i+1} \in x_i$ für alle i gilt. Warum ist das so? Für jedes i sind A und x_i nicht disjunkt, da $x_{i+1} \in A \cap x_i$.

Hier ist eine weitere unmittelbare Folgerung: Es gibt keine Menge M, die sich selbst enthält. Denn die Menge $A = \{M\}$ nur das eine Element M, aber $M \in A \cap M$ wenn M sich selbst enthält. Die „Menge“ N der Russelschen Antinomie, also die „Menge“ aller Mengen, die sich nicht selbst enthalten, ist also die „Menge“ aller Mengen.

8. Angenommen wir ersetzen jedes Element x einer Menge A durch eine Menge y so dass $\psi(x, y)$ für die FO[σ]-Formel ψ gilt. Wenn die Formel ψ eine partielle Funktion beschreibt, dann fordert das `Ersetzungsaxiom`, dass wir wieder eine Menge erhalten.

$$\begin{aligned} & \forall x \forall y \forall z(\ \psi(x,y) \wedge \psi(x,z) \to y \doteq z\ ) \\ \to\ & \forall A \exists B \forall y\Big(y \dot{\in} B \leftrightarrow \exists x\big(x \dot{\in} A \wedge \psi(x,y)\big)\Big). \end{aligned}$$

Bemerkung 9.43. Wir haben in den Abschnitten 2.3 und 2.4 Relationen und Funktionen eingeführt. Kann man dies auch in der Zermelo-Fraenkel Mengenlehre tun? Natürlich! Wir setzen der Einfachheit halber voraus, dass die Mengen $M_1, \ldots, M_k$ disjunkt sind. Dann ist die Menge R genau dann eine **Relation** auf den Mengen $M_1, \ldots, M_k$, wenn die FO[σ]-Formel

$$\psi(R) := \forall x\Big(x \dot{\in} R \to \exists m_1 \cdots \exists m_r\big(m_1 \dot{\in} M_1 \wedge \cdots \wedge m_k \dot{\in} M_k \wedge x \doteq \{m_1, \ldots, m_r\}\big)\Big)$$

erfüllt ist.

Seien A und B, wieder der Einfachheit halber, disjunkte Mengen. Dann ist eine Menge f eine **Funktion** von A nach B genau dann, wenn f eine Relation auf A und B ist und die FO[σ]-Formel

$$\forall x\Big(x \mathrel{\dot\in} A \to \exists y\big(\{x,y\} \mathrel{\dot\in} f \wedge \forall z(\{x,z\} \mathrel{\dot\in} f \to y \mathrel{\dot=} z)\big)\Big).$$

Bemerkung 9.44. Nehmen wir uns einen beliebigen Satz $\varphi \in \mathrm{FO}[\sigma]$ vor. Gilt stets $\mathtt{ZF} \models \psi$ oder $\mathtt{ZF} \models \neg\psi$? Keineswegs!

1. Ein erstes Beispiel ist der Satz φ, der für alle Mengen A, deren Elemente paarweise disjunkte Mengen sind, die Existenz einer Menge B fordert, die genau ein Element aus jedem Element von A enthält.

   Dieser Satz „scheint“ offensichtlich zu sein, kann aber nicht aus dem Axiomensystem ZF gefolgert werden. Den Satz φ nimmt man auch deshalb als **Auswahlaxiom** in ZF auf und nennt das neue Axiomensystem ZFC, Zermelo-Fraenkel Mengenlehre mit dem Auswahlaxiom (englisch: axiom of choice). Auswahlaxiom

2. Die **Kontinuumshypothese** α besagt, dass jede überabzählbare Teilmenge der reellen Zahlen gleichmächtig mit der Menge der reellen Zahlen ist. Auch die Kontinuumshypothese kann mit ZF weder bewiesen noch widerlegt werden. Kontinuums-hypothese

Lässt sich denn ein Axiomensystem $\mathtt{ZF}^*$ finden, so dass für alle FO[σ]-Sätze φ entweder φ oder $\neg\varphi$ gefolgert werden kann? Aber auch hier gilt der Gödelsche Unvollständigkeitssatz: Für jedes solche Axiomensystem $\mathtt{ZF}^*$ gibt es kein stets haltendes Programm, das entscheidet ob eine FO[σ]-Formel zu $\mathtt{ZF}^*$ gehört oder nicht.

So gut wie alle beweisbaren mathematischen Aussagen lassen sich in der Zermelo-Fraenkel Mengenlehre (mit Auswahlaxiom) formulieren und beweisen [5]. Wahrscheinlich gibt es aber auch viele, in ihren jeweiligen Axiomensystemen nicht beweisbare Aussagen.

9.7. Zusammenfassung und Ausblick

Eine **Signatur** σ ist eine Menge von Relations-, Funktions- und Konstantensymbolen. Wir haben σ-**Strukturen** $\mathfrak{A}$ für eine Signatur σ eingeführt: $\mathfrak{A}$ besteht aus einem Universum A und „tatsächlichen“ Relationen und Funktionen für die entsprechenden Relations- und Funktionssymbole der Signatur. Auch müssen den Konstantensymbolen in σ Elemente des Universums zugewiesen werden.

Die Menge FO[σ] der Formeln der Logik erster Stufe über σ haben wir durch eine rekursive Definition eingeführt. Atomare FO[σ]-Formeln sind gleichgesetzte σ-Terme und Relationssymbole in die σ-Terme eingesetzt sind. Im Rekursionsschritt können bereits konstruierte FO[σ]-Formeln mit den aussagenlogischen Junktoren verknüpft werden, bzw. ein Existenz- oder Allquantor darf eingeführt werden.

Die Semantik einer FO[σ]-Formel φ haben wir mit Hilfe von σ-**Interpretationen** $\mathcal{I} = (\mathfrak{A}, \beta)$ festgelegt, wobei die **Belegung** β den freien Variablen von φ Elemente des Universums von $\mathfrak{A}$ zuweist.

In Beispiel 9.20 und 9.21 haben wir viele Beispiele für umgangssprachliche Aussagen kennengelernt, die man durch Formeln der Logik erster Stufe beschreiben kann (siehe auch die Übungsaufgaben am Ende dieses Kapitels). Es gibt allerdings auch Aussagen, die mit bestimmten Signatur **nicht** in der Logik erster Stufe formalisiert werden können:

Satz 9.45. Sei $\sigma_{\text{Graph}} := \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol $\dot{E}$ besteht. Es gilt:

(a) Es gibt keinen FO[σ_{Graph}]-Satz φ, so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ gilt:

$$\mathfrak{A} \text{ erfüllt } \varphi \iff \mathfrak{A} \text{ ist azyklisch (vgl. Definition 5.14.}$$

(b) Es gibt keinen FO[σ_{Graph}]-Satz φ', so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ gilt:

$$\mathfrak{A} \text{ erfüllt } \varphi' \iff \mathfrak{A} \text{ ist stark zusammenhängend (vgl. Definition 5.16).}$$

(c) Es gibt keine FO[σ_{Graph}]-Formel ψ mit freien Variablen x und y, so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ und jede zu ψ passende Belegung β in $\mathfrak{A}$ gilt:

$$(\mathfrak{A}, \beta) \text{ erfüllt } \psi \iff \text{es gibt in } \mathfrak{A} \text{ einen Weg von Knoten } \beta(x) \text{ zu Knoten } \beta(y).$$

Einen Beweis dieses Satzes können Sie in der Vorlesung „Logik in der Informatik“ kennenlernen.

Wir haben auch sehr ausdruckskräftige „Theorien“, nämlich die Zermelo-Fraenkel Mengenlehre ZF und ihre Folgerungen sowie alle in den natürlichen Zahlen wahren FO[σ_{Ar}]-Formeln mit $\sigma_{\text{Ar}} := \{\dot{+}, \dot{\times}, \dot{0}, \dot{1}\}$ betrachtet. In beiden Fällen wird in der „Logik in der Informatik“ gezeigt, dass es kein einfach beschreibbares Axiomensystem gibt, so dass alle Formeln der Theorie Folgerungen des jeweiligen Axiomensystems sind: Es gibt komplexe Realitäten, die sich nicht im Rechner angemessen modellieren lassen.

9.8. Literaturhinweise zu Kapitel 9

[24] Kapitel 2.1

[16] Kapitel 4.A

[14] Kapitel 4.2

Vorsicht: Jedes dieser Bücher verwendet unterschiedliche Notationen, die wiederum etwas von den in der Vorlesungen eingeführten Notationen abweichen.

9.9. Übungsaufgaben zu Kapitel 9

Aufgabe 9.1. Sei $\sigma := \{\dot{f}, \dot{Q}, \dot{R}, \dot{c}\}$ eine Signatur mit einem 1-stelligen Funktionssymbol $\dot{f}$, einem 3-stelligen Relationssymbol $\dot{Q}$, einem 2-stelligen Relationssymbol $\dot{R}$ und einem Konstantensymbol $\dot{c}$.

(a) Überprüfen Sie für jedes der folgenden Wörter, ob es sich jeweils um einen σ-Term (gemäß Definition 6.13), um eine atomare σ-Formel bzw. um eine FO[σ]-Formel (gemäß Definition 6.15) handelt. Begründen Sie gegebenenfalls, warum ein Wort kein σ-Term, keine atomare σ-Formel bzw. keine FO[σ]-Formel darstellt.

(i) $\dot{Q}\big(\dot{f}(v_1), v_4, \dot{c}\big)$

(ii) $\dot{R}\big(\dot{f}(v_1), v_4, \dot{c}\big)$

(iii) $\big(\dot{f}(v_7) \leftrightarrow \dot{R}(v_1, v_2)\big)$

(iv) $\exists v_9\, \dot{f}\big(\dot{f}(\dot{f}(\dot{c}))\big) \doteq \dot{f}(v_8)$

(v) $\forall v_9\, \dot{f}\big(\dot{f}(\dot{f}(\dot{c}))\big) \doteq \dot{f}(v_8 \wedge v_9)$

(vi) $\exists v_1 \forall v_2 \forall v_4 \big(\dot{c} \doteq \dot{f}(v_4) \vee \forall v_2 (\dot{R}(\dot{f}(v_1), v_4) \rightarrow \dot{Q}(v_1, v_3, v_4))\big)$

(b) Betrachten Sie die σ-Strukturen $\mathfrak{A} = (A, \dot{f}^{\mathfrak{A}}, \dot{Q}^{\mathfrak{A}}, \dot{R}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$ und $\mathfrak{B} = (B, \dot{f}^{\mathfrak{B}}, \dot{Q}^{\mathfrak{B}}, \dot{R}^{\mathfrak{B}}, \dot{c}^{\mathfrak{B}})$ wobei

- $A := \{1,2,3,4,5\}$, $\dot{Q}^{\mathfrak{A}} := \{(2,2,4),(5,3,1)\}$, $\dot{R}^{\mathfrak{A}} := \{(3,3),(5,4),(1,1)\}$, $\dot{c}^{\mathfrak{A}} := 2$
- $B := \{k,l,m,n,o\}$, $\dot{Q}^{\mathfrak{B}} := \{(k,m,o),(n,n,l)\}$, $\dot{R}^{\mathfrak{B}} := \{(o,o),(m,m),(k,l)\}$, $\dot{c}^{\mathfrak{B}} := n$

und die Funktionen $\dot{f}^{\mathfrak{A}} \colon A \to A$ und $\dot{f}^{\mathfrak{B}} \colon B \to B$ definiert sind durch

x	1	2	3	4	5
$\dot{f}^{\mathfrak{A}}(x)$	2	1	2	5	4

x	k	l	m	n	o
$\dot{f}^{\mathfrak{B}}(x)$	l	k	n	o	n

Überprüfen Sie, ob $\mathfrak{A} \cong \mathfrak{B}$ gilt. Falls ja, geben Sie einen Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$ an und begründen Sie, warum es sich um einen Isomorphismus handelt. Falls nein, begründen Sie, warum es keinen Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$ gibt.

Aufgabe 9.2. Sei $\sigma := \{\dot{f}, \dot{R}, \dot{S}, \dot{c}\}$ eine Signatur mit einem 1-stelligen Funktionssymbol $\dot{f}$, einem 2-stelligen Relationssymbol $\dot{R}$, einem 3-stelligen Relationssymbol $\dot{S}$ und einem Konstantensymbol $\dot{c}$.

(a) Überprüfen Sie für jedes der folgenden Wörter, ob es sich jeweils um einen σ-Term (gemäß Definition 6.13), um eine atomare σ-Formel bzw. um eine FO[σ]-Formel (gemäß Definition 6.19) handelt. Begründen Sie gegebenenfalls, warum ein Wort kein σ-Term, keine atomare σ-Formel bzw. keine FO[σ]-Formel darstellt.

(i) $\dot{f}(\dot{f}(\dot{f}(\dot{c})))$

(ii) $\big(\dot{f}(v_1) \wedge \dot{R}(v_1, v_2)\big)$

(iii) $\dot{S}(\dot{f}(\dot{c}), v_3)$

(iv) $\big(\forall v_1 \dot{f}(\dot{f}(v_1)) \wedge \dot{S}(\dot{f}(v_2), v_1, \dot{c})\big)$

(v) $\exists v_2 \big(\dot{f}(v_2) \doteq \dot{f}(\dot{f}(v_2)) \vee \neg \dot{R}(v_2, \dot{f}(v_2))\big)$

(vi) $\forall v_4 \exists v_5 \forall v_6 \big(\big(\dot{S}(\dot{f}(v_1), v_4, v_5) \wedge \dot{R}(v_1, v_6)\big) \rightarrow \dot{f}(v_3) \doteq \dot{c}\big)$

(b) Betrachten Sie die drei σ-Strukturen $\mathfrak{A} := (A, \dot{f}^{\mathfrak{A}}, \dot{R}^{\mathfrak{A}}, \dot{S}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$, $\mathfrak{B} := (B, \dot{f}^{\mathfrak{B}}, \dot{R}^{\mathfrak{B}}, \dot{S}^{\mathfrak{B}}, \dot{c}^{\mathfrak{B}})$ und $\mathfrak{C} := (C, \dot{f}^{\mathfrak{C}}, \dot{R}^{\mathfrak{C}}, \dot{S}^{\mathfrak{C}}, \dot{c}^{\mathfrak{C}})$ wobei

- $A := \{q,r,s,t,u\}$, $\dot{R}^{\mathfrak{A}} := \{(q,q),(r,t),(t,r),(u,q)\}$, $\dot{S}^{\mathfrak{A}} := \{(q,s,q),(u,t,r)\}$, $\dot{c}^{\mathfrak{A}} := s$,
- $B := \{1,2,3,4,5\}$, $\dot{R}^{\mathfrak{B}} := \{(5,5),(4,1),(1,4),(2,5)\}$, $\dot{S}^{\mathfrak{B}} := \{(5,3,5),(2,1,4)\}$, $\dot{c}^{\mathfrak{B}} := 3$,
- $C := \{v,w,x,y,z\}$, $\dot{R}^{\mathfrak{C}} := \{(v,v),(w,y),(y,w),(z,v)\}$, $\dot{S}^{\mathfrak{C}} := \{(v,z,v),(z,y,w)\}$, $\dot{c}^{\mathfrak{C}} := x$

und die Funktionen $\dot{f}^{\mathfrak{A}} \colon A \to A$, $\dot{f}^{\mathfrak{B}} \colon B \to B$ und $\dot{f}^{\mathfrak{C}} \colon C \to C$ definiert sind durch

x	q	r	s	t	u
$\dot{f}^{\mathfrak{A}}(x)$	t	s	t	u	q

x	1	2	3	4	5
$\dot{f}^{\mathfrak{B}}(x)$	2	5	1	3	1

x	v	w	x	y	z
$\dot{f}^{\mathfrak{C}}(x)$	y	x	y	z	v

Überprüfen Sie jeweils, ob $\mathfrak{A} \cong \mathfrak{B}$ und ob $\mathfrak{A} \cong \mathfrak{C}$ gilt. Falls ja, geben Sie einen entsprechenden Isomorphismus an und begründen Sie, warum es sich um einen Isomorphismus handelt. Falls nein, begründen Sie, warum es keinen entsprechenden Isomorphismus gibt.

Aufgabe 9.3. Sei $\sigma := \{\dot{R}, \dot{f}, \dot{c}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{R}$, einem 1-stelligen Funktionssymbol $\dot{f}$ und dem Konstantensymbol $\dot{c}$.

(a) Bestimmen Sie für jede der folgenden FO[σ]-Formeln, welche Variablen frei und welche Variablen gebunden in der Formel vorkommen (ohne Begründung). Entscheiden Sie außerdem für jede der FO[σ]-Formeln, ob es sich um einen FO[σ]-Satz handelt.

(i) $\dot{f}(x) \doteq \dot{f}(\dot{c})$

(ii) $\exists x\, \dot{R}(\dot{f}(\dot{f}(x)), x)$

(iii) $\big(\exists y\, \dot{R}(y,y) \;\vee\; \forall x\, \neg \dot{R}(x,y)\big)$

(iv) $\exists z\, \big(\dot{R}(x,z) \;\wedge\; \dot{R}(z,y)\big)$

(v) $\forall x \forall y\, \big(\exists z\, \dot{f}(z) \doteq x \;\rightarrow\; \exists z\, \dot{f}(z) \doteq y\big)$

(vi) $\big(\forall z \exists x\, \dot{R}(x, \dot{c}) \;\leftrightarrow\; \forall y\, \dot{R}(y,z)\big)$

(b) Betrachten Sie die σ-Struktur $\mathfrak{A} = (A, \dot{R}^{\mathfrak{A}}, \dot{f}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$ mit dem Universum $A = \{0,1,2,3,4\}$, $\dot{R}^{\mathfrak{A}} = \{(x,y) \in A^2 \colon x \leqslant y\}$ und $\dot{c}^{\mathfrak{A}} = 0$. Weiterhin sei $\dot{f}^{\mathfrak{A}} \colon A \to A$ für alle $x \in A$ definiert durch

$$\dot{f}^{\mathfrak{A}}(x) \;=\; \begin{cases} 2x & \text{wenn } x \leqslant 2, \\ 2x - 5 & \text{sonst.} \end{cases}$$

Sei $\mathcal{I} = (\mathfrak{A}, \beta)$ die σ-Interpretation mit der Belegung $\beta : \textsc{Var} \to A$, für die gilt:

$$\beta(v_0) = 1, \quad \beta(v_1) = 3, \quad \beta(v_2) = 1, \quad \beta(v_3) = 2, \quad \beta(v_i) = 4, \text{für alle } i > 3.$$

Berechnen Sie $[\![\varphi_i]\!]^{\mathcal{I}}$ für jede der FO[σ]-Formeln φ_i, $i \in \{1,2,3\}$ analog zu Beispiel 9.29.

(i) $\varphi_1 \;:=\; \big(\dot{f}(v_1) \doteq v_2 \;\wedge\; \dot{f}(v_2) \doteq v_3\big)$

(ii) $\varphi_2 \;:=\; \exists v_1\, \big(v_1 \doteq \dot{f}(v_4) \;\wedge\; \dot{R}(v_4, v_1)\big)$

(iii) $\varphi_3 \;:=\; \forall v_0 \forall v_1 \forall v_2\, \big((\dot{R}(v_0, v_1) \;\wedge\; \dot{R}(v_1, v_2)) \;\rightarrow\; \dot{R}(v_0, v_2)\big)$

Aufgabe 9.4. Sei $\sigma = \{\dot{B}, \dot{S}, \dot{F}, \dot{Nachfolger}, \dot{letzter}\}$ eine Signatur, wobei $\dot{B}, \dot{S}, \dot{F}$ 1-stellige Relationsymbole, $\dot{Nachfolger}$ ein 1-stelliges Funktionssymbol und $\dot{letzter}$ ein Konstantensymbol ist. Sei $\mathfrak{A}$ eine σ-Struktur mit $A = \{1,2,\ldots,34\}$ und $\dot{letzter}^{\mathfrak{A}} = 34$, so dass für alle $a \in A$ gilt:

- $a \in \dot{B}^{\mathfrak{A}} \iff$ FC Bayern München ist Tabellenführer an Spieltag a
- $a \in \dot{S}^{\mathfrak{A}} \iff$ FC Schalke 04 ist Tabellenführer an Spieltag a
- $a \in \dot{F}^{\mathfrak{A}} \iff$ Eintracht Frankfurt ist Tabellenführer an Spieltag a
- $\dot{Nachfolger}^{\mathfrak{A}}(a) = \begin{cases} a+1, & \text{falls } a \in \{1,2,\ldots,33\} \\ a, & \text{falls } a = 34. \end{cases}$

(a) Geben Sie FO[σ]-Formeln an, die in $\mathfrak{A}$ Folgendes aussagen:

(i) Eintracht Frankfurt ist mindestens einmal Tabellenführer.

(ii) Jede der drei Mannschaften ist mindestens einmal Tabellenführer.

(iii) Sind die Bayern an einem Spieltag Erster, so werden sie auch Meister.

(iv) Schalke holt nicht den Titel, wenn sie bereits am vorletzten Spieltag Tabellenführer sind.

(b) Beschreiben Sie umgangssprachlich, was jede der folgenden FO[σ]-Formeln in $\mathfrak{A}$ aussagt:

(i) $\forall x\big(\, \neg \dot{B}(x) \;\rightarrow\; (\dot{S}(x) \vee \dot{F}(x))\, \big)$

(ii) $\neg\exists x\Big(\ \dot{F}(x)\ \wedge\ \big(\ \dot{F}(\dot{Nachfolger}(x))\ \wedge\ \big(\ \dot{F}(\dot{Nachfolger}(\dot{Nachfolger}(x)))\ \wedge \neg \dot{Nachfolger}(x) \doteq \dot{letzter}\ \big)\ \big)\ \Big)$

(iii) $\Big(\ \neg\exists x\big(\dot{S}(x) \wedge \neg x \doteq \dot{letzter}\big)\ \rightarrow\ \neg\dot{S}(\dot{letzter})\ \Big)$

Aufgabe 9.5. Sophie, Marie und Lena nehmen im Internet an einer Auktion teil. Die Auktion dauert zehn Runden und in jeder Runde gibt es genau einen Höchstbietenden. Der Höchstbietende der zehnten Runde gewinnt die Auktion. Den Verlauf dieser Auktion modellieren wir wie folgt.

Sei $\sigma := \{\dot{S}, \dot{M}, \dot{L}, \dot{Nachfolger}, \dot{Endrunde}\}$ eine Signatur, wobei $\dot{S}, \dot{M}, \dot{L}$ 1-stellige Relationssymbole, $\dot{Nachfolger}$ ein 1-stelliges Funktionssymbol und $\dot{Endrunde}$ ein Konstantensymbol ist. Sei $\mathfrak{A}$ eine σ-Struktur mit $\mathfrak{A} = (A, \dot{S}^{\mathfrak{A}}, \dot{M}^{\mathfrak{A}}, \dot{L}^{\mathfrak{A}}, \dot{Nachfolger}^{\mathfrak{A}}, \dot{Endrunde}^{\mathfrak{A}})$ wobei $A := \{1, \ldots, 10\}$ und $\dot{Endrunde}^{\mathfrak{A}} := 10$. Außerdem gilt für alle $a \in A$:

- $a \in \dot{S}^{\mathfrak{A}} \iff$ Sophie ist in Runde a die Höchstbietende
- $a \in \dot{M}^{\mathfrak{A}} \iff$ Marie ist in Runde a die Höchstbietende
- $a \in \dot{L}^{\mathfrak{A}} \iff$ Lena ist in Runde a die Höchstbietende
- $\dot{Nachfolger}^{\mathfrak{A}}(a) := \begin{cases} a+1, & \text{falls } a \in \{1, \ldots, 9\} \\ a, & \text{falls } a = 10. \end{cases}$

So sagt beispielsweise die FO[σ]-Formel $\varphi = \forall x(\dot{S}(x) \vee \dot{M}(x))$ aus, dass in jeder Runde Sophie oder Marie die Höchstbietende ist.

(a) Geben Sie möglichst kurze FO[σ]-Formeln an, die in $\mathfrak{A}$ Folgendes aussagen:
   (i) Lena gewinnt die Auktion.
   (ii) Sophie ist in mindestens einer Runde die Höchstbietende.
   (iii) Lena ist in der neunten Runde die Höchstbietende.
   (iv) Ist Marie in keiner der ersten neun Runden die Höchstbietende, so gewinnt sie auch nicht die Auktion.

(b) Beschreiben Sie umgangssprachlich, was die folgenden FO[σ]-Formeln aussagen
   (i) $\forall x\big(\ \neg\big(\dot{L}(x) \vee \dot{S}(x)\big) \rightarrow \dot{M}(x)\ \big)$
   (ii) $\forall x\big(\ \big(\neg \dot{Nachfolger}(x) \doteq x \wedge \dot{S}(x)\big) \rightarrow \dot{L}(\dot{Nachfolger}(x))\ \big)$
   (iii) $\neg\exists x\Big(\ \dot{M}(x)\ \wedge\ \big(\ \dot{M}(\dot{Nachfolger}(x))\ \wedge\ \big(\ \dot{M}(\dot{Nachfolger}(\dot{Nachfolger}(x)))\ \wedge \neg \dot{Nachfolger}(x) \doteq \dot{Endrunde}\ \big)\ \big)\ \Big)$

Aufgabe 9.6. Sei $\sigma = \{\dot{R}, \dot{f}, \dot{c}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{R}$, einem 1-stelligen Funktionssymbol $\dot{f}$ und einem Konstantensymbol $\dot{c}$.

Betrachten Sie die σ-Struktur $\mathfrak{A} = (A, \dot{R}^{\mathfrak{A}}, \dot{f}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$, wobei gilt, dass $A = \{1, 3, 5, 7, 8, 9\}$, $\dot{R}^{\mathfrak{A}} = \{(1,3), (1,5), (5,1), (5,8), (9,9)\}$ und $\dot{c}^{\mathfrak{A}} = 1$. Weiterhin sei $\dot{f}^{\mathfrak{A}} : A \to A$, definiert durch

x	1	3	5	7	8	9
$\dot{f}^{\mathfrak{A}}(x)$	9	8	7	5	3	1

Sei $\mathcal{I} = (\mathfrak{A}, \beta)$ die σ-Interpretation mit der Belegung $\beta : \mathrm{Var} \to A$, für die gilt:

$$\beta(v_0) = 3,\ \beta(v_1) = 1,\ \beta(v_2) = 9,\ \text{ und } \beta(v_i) = 5 \text{ für alle } i \geqslant 3.$$

Berechnen Sie $[\![t_1]\!]^{\mathcal{I}}$ für den σ-Term t_1 analog zu Beispiel 9.21. Berechnen Sie weiterhin $[\![\varphi_1]\!]^{\mathcal{I}}$ und $[\![\varphi_2]\!]^{\mathcal{I}}$ für die FO[σ]-Formeln φ_1 und φ_2 analog zu Beispiel 9.29.

(i) $t_1 := \dot{f}(\dot{f}(v_0))$

(ii) $\varphi_1 := \big(\dot{R}(v_6, v_2) \to \dot{R}(v_2, v_2)\big)$

(iii) $\varphi_2 := \forall v_4\big((\dot{R}(\dot{c}, v_4) \vee \dot{R}(v_3, v_4)) \wedge \exists v_0\, \neg \dot{f}(v_0) \doteq v_4\big)$

Aufgabe 9.7.
Sei $\sigma := \{\dot{f}, \dot{c}\}$ eine Signatur mit einem 2-stelligen Funktionssymbol $\dot{f}$ und einem Konstantensymbol $\dot{c}$. Wir betrachten die σ-Struktur $\mathfrak{A} := (A, \dot{f}^{\mathfrak{A}}, \dot{C}^{\mathfrak{A}})$, wobei $A := \{\text{Stein}, \text{Schere}, \text{Papier}, \text{Echse}, \text{Spock}\}$ und $\dot{c}^{\mathfrak{A}} := \text{Spock}$. Der Wert $\dot{f}^{\mathfrak{A}}(x, y)$ für $x, y \in A$ findet sich in Zeile x und Spalte y der nebenstehenden Tabelle.[1]

$\dot{f}^{\mathfrak{A}}$	Stein	Schere	Papier	Echse	Spock
Stein	Stein	Stein	Papier	Stein	Spock
Schere	Stein	Schere	Schere	Schere	Spock
Papier	Papier	Schere	Papier	Echse	Papier
Echse	Stein	Schere	Echse	Echse	Echse
Spock	Spock	Spock	Papier	Echse	Spock

Sei $\mathcal{I} = (\mathfrak{A}, \beta)$ die σ-Interpretation mit der Belegung $\beta\colon \textsc{Var} \to A$, für die gilt:

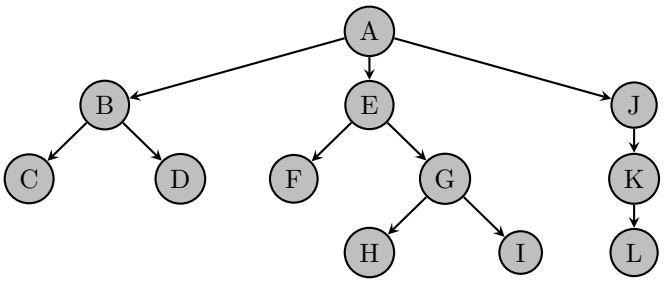
$$\beta(v_0) = \text{Stein},\ \beta(v_1) = \text{Spock},\ \beta(v_2) = \text{Schere},\ \text{ und } \beta(v_i) = \text{Papier} \ \text{ für alle } i \geqslant 3.$$

Berechnen Sie $[\![t_1]\!]^{\mathcal{I}}$, $[\![t_2]\!]^{\mathcal{I}}$, $[\![t_3]\!]^{\mathcal{I}}$ für die folgenden σ-Terme:

(a) $t_1 := \dot{f}(v_0, \dot{c})$ (b) $t_2 := \dot{f}(v_1, \dot{f}(v_0, v_2))$ (c) $t_3 := \dot{f}(\dot{f}(\dot{f}(v_0, v_0), \dot{c}), \dot{f}(v_3, \dot{f}(v_4, v_5)))$

Aufgabe 9.8. In dieser Aufgabe sollen gerichtete Bäume durch Strukturen über einer Signatur mit einem 1-stelligen Funktionssymbol *Elternknoten* repräsentiert werden.

(a) Beschreiben Sie, wie ein gegebener gerichteter Baum $B = (V, E)$ mit $V \neq \emptyset$ durch eine Struktur über der Signatur $\sigma = \{\dot{\textit{Elternknoten}}\}$ modelliert werden kann. Geben Sie die entsprechende Struktur für den folgenden Baum an:



(b) Geben Sie je eine Formel $\varphi(x)$ der Logik erster Stufe an, die ausdrückt, dass der Knoten x

[1] Die Struktur $\mathfrak{A}$ spiegelt das Spiel „Stein, Schere, Papier, Echse, Spock“ wider, dass Rajesh und Sheldon in Folge 25 der Serie *The Big Bang Theory* spielen, um ihre Meinungsverschiedenheiten auszuräumen – allerdings ohne Erfolg.

(i) ein Blatt ist,
(ii) die Wurzel ist,
(iii) ein innerer Knoten ist,
(iv) genau zwei Kinder hat.

(c) Geben Sie je eine Formel $\psi(x, y)$ der Logik erster Stufe an, die ausdrückt, dass es vom Knoten x zum Knoten y
(i) einen Weg der Länge zwei gibt,
(ii) einen einfachen Weg der Länge zwei gibt.

Aufgabe 9.9. Sei $\sigma := \{\dot{E}, \dot{P}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{E}$ und einem 1-stelligen Relationssymbol $\dot{P}$. Geben Sie für jede der folgenden Formeln je eine σ-Struktur an, die die Formel erfüllt, und eine, die die Formel nicht erfüllt:

(a) $\forall x \forall y \forall z \big(\big(\dot{E}(x,y) \wedge \dot{E}(y,z)\big) \rightarrow \dot{E}(x,z) \big)$

(b) $\forall x \forall y \Big(\dot{E}(x,y) \rightarrow \big(\big(\dot{P}(y) \wedge \neg\dot{P}(x)\big) \vee \big(\dot{P}(x) \wedge \neg\dot{P}(y)\big) \big) \Big)$

(c) $\Big(\forall x \forall y \big(\dot{E}(x,y) \vee \dot{E}(y,x)\big) \wedge \forall x \forall y \big((\dot{E}(x,y) \wedge \dot{E}(y,x)) \rightarrow x \doteq y \big) \Big)$

Aufgabe 9.10. Sei $\sigma := \{\dot{E}, \dot{g}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{E}$ und einem 1-stelligen Funktionssymbol $\dot{g}$. Geben Sie für jede der folgenden FO[σ]-Formeln je eine σ-Struktur an, die die Formel erfüllt, und eine, die die Formel nicht erfüllt.

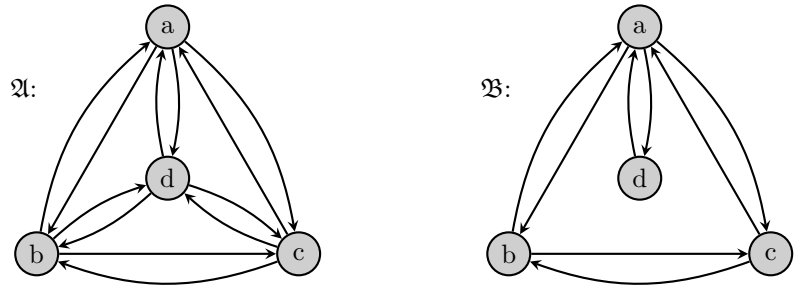
(a) $\forall x \forall y \big(\dot{E}(x,y) \rightarrow \dot{E}(y,x) \big)$

(b) $\forall x \forall y \big(\dot{E}(x,y) \rightarrow \neg\dot{g}(x) \doteq \dot{g}(y) \big)$

(c) $\Big(\forall x \forall y \forall z \big(\big(\dot{E}(x,y) \wedge \dot{E}(y,z)\big) \rightarrow \dot{E}(x,z) \big) \vee \forall x \forall y \big(\dot{E}(x,y) \leftrightarrow \neg\dot{E}(y,x) \big) \Big)$

Aufgabe 9.11.

(a) Sei $\sigma = \{\dot{E}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{E}$. Betrachten Sie die beiden σ-Strukturen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ und $\mathfrak{B} = (A, \dot{E}^{\mathfrak{B}})$, die durch die beiden folgenden Graphen repräsentiert werden.



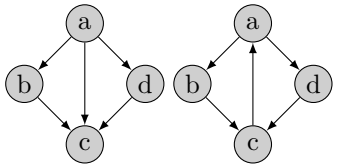
Geben Sie einen FO[σ]-Satz φ an, so dass $\mathfrak{A} \models \varphi$ und $\mathfrak{B} \models \neg\varphi$ gilt.

(b) Sei $\sigma = \{\dot{E}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{E}$. Geben Sie für die Formel

$$\varphi(x) := \forall y \exists z \big((\dot{E}(y,x) \rightarrow \dot{E}(x,z)) \vee x \doteq y \big)$$

eine Struktur $\mathfrak{A}$ und zwei Interpretationen $\mathcal{I}_1 = (\mathfrak{A}, \beta_1)$ und $\mathcal{I}_2 = (\mathfrak{A}, \beta_2)$ an, so dass $\mathcal{I}_1 \models \varphi$ und $\mathcal{I}_2 \models \neg\varphi$ gilt.

Aufgabe 9.12. (a) Betrachten Sie die σ_{Graph}-Strukturen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ und $\mathfrak{B} = (B, \dot{E}^{\mathfrak{B}})$, die durch die beiden Graphen in der folgenden Abbildung repräsentiert werden:



Geben Sie einen FO[σ_{Graph}]-Satz φ an, so dass $\mathfrak{A} \models \varphi$ und $\mathfrak{B} \models \neg\varphi$.

(b) Geben Sie für die Formel

$$\varphi(x) \;:=\; \forall y \,\big(x \doteq y \;\vee\; \neg \dot{E}(y,x) \;\vee\; \exists z\, \dot{E}(x,z)\big)$$

eine σ_{Graph}-Struktur $\mathfrak{C}$ und zwei Interpretationen $\mathcal{I}_1 = (\mathfrak{C}, \beta_1)$ und $\mathcal{I}_2 = (\mathfrak{C}, \beta_2)$ an, so dass $\mathcal{I}_1 \models \varphi$ und $\mathcal{I}_2 \models \neg\varphi$.

Aufgabe 9.13. Betrachten Sie die Kinodatenbank $\mathfrak{A}_{\text{Kino}}$ aus der Vorlesung.

(a) Berechnen Sie für jede der folgenden Formeln φ_i die Relation $\varphi_i(\mathfrak{A}_{\text{Kino}})$ und geben Sie umgangssprachlich an, welche Anfrage durch die Formel φ_i beschrieben wird:

$$\begin{aligned}
\varphi_1(x_{\text{K}}) &= \exists x_{\text{T}} \exists x_{\text{Z}}\, \dot{\mathit{Programm}}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \\
\varphi_2(x_{\text{K}}) &= \exists x_{\text{Z}}\, \dot{\mathit{Programm}}(x_{\text{K}}, \text{`}\dot{\mathit{Capote}}\text{'}, x_{\text{Z}}) \\
\varphi_3(x_{\text{S}}) &= \exists x_{\text{T}} \big(\, \exists x_{\text{R}}\, \dot{\mathit{Film}}(x_{\text{T}}, x_{\text{R}}, x_{\text{S}}) \;\wedge\; \exists x_{\text{K}} \exists x_{\text{Z}}\, \dot{\mathit{Programm}}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \,\big) \\
\varphi_4(x_1, x_2) &= \exists x_{\text{S}} \exists x_{\text{R}} \exists x_{\text{T}} \,\big(\dot{\mathit{Film}}(x_2, x_1, x_{\text{S}}) \;\wedge\; \dot{\mathit{Film}}(x_{\text{T}}, x_{\text{R}}, x_1)\big) \\
\varphi_5(x_{\text{T}}) &= \exists x_{\text{K}} \exists x_{\text{Z}} \Big(\; \dot{\mathit{Programm}}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \;\wedge \\
&\qquad\qquad \forall y_{\text{K}} \forall y_{\text{Z}} \big(\dot{\mathit{Programm}}(y_{\text{K}}, x_{\text{T}}, y_{\text{Z}}) \to y_{\text{Z}} \doteq x_{\text{Z}}\big) \;\Big) \\
\varphi_6(x_{\text{T}}, x_{\text{K}}, x_{\text{A}}) &= \Big(\; \exists x_{\text{Z}}\, \exists x_{\text{Tel}} \big(\dot{\mathit{Programm}}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \wedge \dot{\mathit{Orte}}(x_{\text{K}}, x_{\text{A}}, x_{\text{Tel}})\big) \;\wedge \\
&\qquad\qquad \exists x_{\text{S}}\, \dot{\mathit{Film}}(x_{\text{T}}, \text{`}\dot{\mathit{George}}\ \dot{\mathit{Clooney}}\text{'}, x_{\text{S}}) \;\Big) \\
\varphi_7(x_1, x_2, x_3) &= \Big(\exists x_{\text{T}}\, \dot{\mathit{Orte}}(x_1, x_2, x_{\text{T}}) \wedge \\
&\qquad \exists x_{\text{Z}_1} \big(\dot{\mathit{Programm}}(x_1, x_3, x_{\text{Z}_1}) \;\wedge\; \forall x_{\text{K}} \forall x_{\text{Z}_2} (\dot{\mathit{Programm}}(x_{\text{K}}, x_3, x_{\text{Z}_2}) \;\to\; x_1 \doteq x_{\text{K}})\big)\Big)
\end{aligned}$$

(b) Finden Sie Formeln der Logik erster Stufe, die die folgenden Anfragen beschreiben:

(i) Geben Sie alle Kinos aus, in denen um '21:45' ein Film läuft.

(ii) Geben Sie die Telefonnummern und Adressen aller Kinos aus, in denen der Film 'Capote' läuft.

(iii) Geben Sie die Telefonnummern der Kinos aus, die um 20:15 Uhr oder um 20:30 Uhr eine Vorstellung haben.

(iv) Geben Sie die Titel aller Filme aus, die in mindestens einem Kino laufen.

(v) Geben Sie die Titel aller Filme aus, die in mindestens zwei Kinos laufen.

(vi) Geben Sie die Titel aller Filme aus, die in genau einem Kino laufen.

(vii) Geben Sie die Titel aller Filme aus, deren Schauspieler schon mal in einem Film von Stephen Spielberg mitgespielt haben.

(viii) Geben Sie die Titel aller Filme aus, in denen George Clooney mitspielt, aber nicht selbst Regie führt.

Beachten Sie: Es kann sein, dass ein Film mehr als einen Regisseur hat, z.B. Raumpatrouille Orion – Rücksturz ins Kino.

(ix) Geben Sie alle Filme aus, die nur in einem einzelnen Kino gespielt werden, zusammen mit den Anfangszeiten des jeweiligen Filmes.

(x) Geben Sie die Adressen aller Kinos aus, in denen kein Film vom Regisseur 'George Clooney' und kein Film mit dem Schauspieler 'Philip Seymour' läuft.

Aufgabe 9.14.

(a) Welche der folgenden Aussagen stimmen, welche stimmen nicht?

(i) $\forall x\,\varphi \;\equiv\; \neg\exists x\,\neg\varphi$

(ii) $\exists x\,\varphi \;\models\; \forall x\,\varphi$

(iii) $\forall x\,\varphi \;\models\; \exists x\,\varphi$

(iv) $\exists x(\varphi \wedge \psi) \;\models\; \big(\exists x\,\varphi \;\wedge\; \exists x\,\psi\big)$

(v) $\big(\exists x\,\varphi \;\wedge\; \exists x\,\psi\big) \;\models\; \exists x(\varphi \wedge \psi)$

(vi) $\exists x(\varphi \wedge \psi) \;\equiv\; \big(\exists x\,\varphi \;\wedge\; \exists x\,\psi\big)$

(vii) $\forall x(\varphi \wedge \psi) \;\equiv\; \big(\forall x\,\varphi \;\wedge\; \forall x\,\psi\big)$

(viii) $(\forall x\,\varphi \;\wedge\; \forall x\,\psi) \;\equiv\; \forall x\,(\varphi \;\wedge\; \psi)$

(ix) $(\forall x\,\varphi \;\vee\; \forall x\,\psi) \;\equiv\; \forall x\,(\varphi \;\vee\; \psi)$

(x) $(\exists x\,\varphi \;\wedge\; \exists x\,\psi) \;\equiv\; \exists x\,(\varphi \;\wedge\; \psi)$

(xi) $(\exists x\,\varphi \;\vee\; \exists x\,\psi) \;\equiv\; \exists x\,(\varphi \;\vee\; \psi)$

(b) Beweisen Sie, dass Ihre Antworten zu ((iii)), ((iv)), ((v)), ((viii)) und ((x)) aus (a) korrekt sind.

(c) Zeigen Sie die Korrektheit der Beobachtung 9.41 (c), d.h. zeigen Sie, dass für jede Signatur σ und zwei beliebige FO[σ]-Formeln φ und ψ gilt:

$$\varphi \models \psi \quad\iff\quad (\varphi \to \psi) \text{ ist allgemeingültig.}$$

Aufgabe 9.15. Entscheiden Sie, ob FO[σ_{Graph}]-Formeln φ und ψ mit freien Variablen x, y und z existieren, so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ und jede zu φ und ψ passende Belegung β in $\mathfrak{A}$ gilt:

(a) $(\mathfrak{A}, \beta) \models \varphi \iff$ Es ex. in $\mathfrak{A}$ ein einfacher Weg von $\beta(x)$ über $\beta(y)$ nach $\beta(z)$ der Länge vier.

(b) $(\mathfrak{A}, \beta) \models \psi \iff$ Es ex. in $\mathfrak{A}$ ein einfacher Weg von $\beta(x)$ über $\beta(y)$ nach $\beta(z)$.

Aufgabe 9.16. Entscheiden Sie, ob FO[σ_{Graph}]-Formeln φ und ψ mit freien Variablen x und y existieren, so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ und jede zu φ und ψ passende Belegung β in $\mathfrak{A}$ gilt:

(a) $(\mathfrak{A}, \beta)$ erfüllt $\varphi \iff \beta(x)$ und $\beta(y)$ liegen zusammen auf einem Kreis in $\mathfrak{A}$

(b) $(\mathfrak{A}, \beta)$ erfüllt $\psi \iff \beta(x)$ und $\beta(y)$ liegen zusammen auf einem Kreis der Länge vier in $\mathfrak{A}$

(c) $(\mathfrak{A}, \beta)$ erfüllt $\varphi \iff$ in $\mathfrak{A}$ existiert ein Weg von $\beta(x)$ nach $\beta(y)$

(d) $(\mathfrak{A}, \beta)$ erfüllt $\psi \iff$ in $\mathfrak{A}$ existiert ein Weg der Länge 3 von $\beta(x)$ nach $\beta(y)$.

Literaturverzeichnis

[1] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Searching the web. *ACM Transactions on Internet Technology*, 1(1):2–43, 2001.

[2] Albrecht Beutelspacher. *"Das ist o.B.d.A. trivial!". Tipps und Tricks zur Formulierung mathematischer Gedanken.* Vieweg Studium, Braunschweig, 2002.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[4] Reinhard Diestel. *Graphentheorie.* Springer-Verlag, Berlin, 2006.

[5] Heinz-Dieter Ebbinghaus. *Einführung in die Mengenlehre.* Spektrum Akademischer Verlag, Heidelberg Berlin, 2003.

[6] Ayman Farahat, Thomas LoFaro, Joel C. Miller, Gregory Rae, and Lesley A. Ward. Authority rankings from HITS, PageRank, and SALSA: Existence, uniqueness, and effect of initialization. *SIAM Journal on Scientific Computing*, 27(4):1181–1201, 2006.

[7] William Feller. *An Introduction to Probability Theory and Its Applications: Volume I.* Wiley, 3rd edition, 1968. ISBN: 978-0-471-25708-0.

[8] Daniel Grieser. *Mathematisches Problemlösen und Beweisen: Eine Entdeckungsreise in die Mathematik (Bachelorkurs Mathematik).* Springer Vieweg, 2012.

[9] Martin Grohe. *Theoretische Informatik I.* Skript zur Vorlesung am Institut für Informatik, Humboldt-Universität zu Berlin, 2007.

[10] J. Y. Halpern, R. Harper, N. Immerman, P. G. Kolaitis, M. Y. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic*, 7(2):213–236, June 2001.

[11] Olle Häggström. *Finite Markov chains and algorithmic applications.* Number 52 in London Mathematical Society Student Texts. Cambridge University Press, 2002. ISBN-10: 0521890012.

[12] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[13] Stasys Jukna. *Crashkurs Mathematik für Informatiker.* Leitfäden der Informatik. Teubner Verlag, 2008. ISBN 978-3-8351-0216-3.

[14] Uwe Kastens and Hans Kleine Büning. *Modellierung. Grundlagen und formale Methoden.* Carl Hanser Verlag, München, 2005.

[15] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[16] Martin Kreuzer and Stefan Kühling. *Logik für Informatiker.* Pearson Studium, München, 2006.

[17] Amy N. Langville and Carl D. Meyer. *Google's Pagerank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[18] L. Lovász, J. Pelikán, and K. Vesztergombi. *Discrete Mathematics.* Springer Science+Business Media, LLC, New York, 2003.

[19] Zohar Manna and Richard Waldinger. *The logical basis for computer programming.* Addison-Wesley, 1985.

[20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[21] Christoph Meinel and Martin Mundhenk. *Mathematische Grundlagen der Informatik. Mathematisches Denken und Beweisen - Eine Einführung.* B.G. Teubner, Stuttgart, 2000.

[22] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66 (previous number: SIDL-WP-1999-0120), Stanford InfoLab, November 1999. The article is available from http://ilpubs.stanford.edu:8090/422/.

[23] Uwe Schöning. *Theoretische Informatik – kurzgefasst.* Spektrum Akademischer Verlag, Heidelberg, 2001.

[24] Uwe Schöning. *Logik für Informatiker.* Spektrum Akademischer Verlag, Heidelberg, 2005.

[25] Georg Schnitger. *Internet Algorithmen.* Skript zur Vorlesung am Institut für Informatik, Goethe-Universität Frankfurt am Main, 2014.

[26] Ingo Wegener. *Kompendium Theoretische Informatik – eine Ideensammlung.* B.G. Teubner, Stuttgart, 1996.

[27] Ingo Wegener. *Theoretische Informatik.* B.G. Teubner, Stuttgart, 1999.