Goethe-Universität Frankfurt am Main
Institut für Informatik
Theorie komplexer Systeme

Diskrete Modellierung

Skript zur Vorlesung

Prof. Dr. Nicole Schweikardt

Version vom 6. März 2009

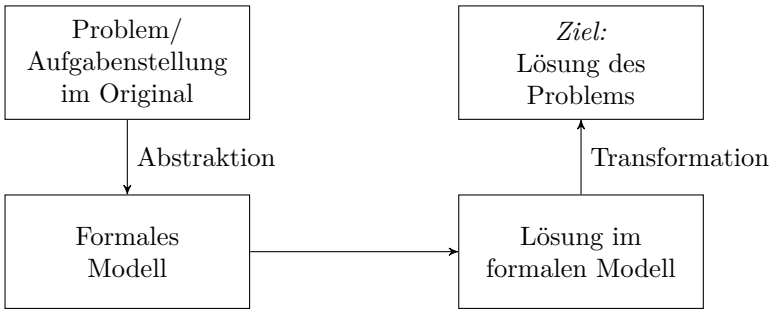
Inhaltsverzeichnis

1 Einführung ins Thema "Diskrete Modellierung"

1.1 Wozu diskrete Modellierung im Informatik-Studium?

Typische Arbeitsmethode in vielen Bereichen der Informatik: Modellierung von Problemen mittels diskreter Strukturen $\Longrightarrow$ präzise Beschreibung von Problemen – dies ist Grundvoraussetzung dafür, Probleme systematisch mit Methoden der Informatik lösen zu können.

Generelle Vorgehensweise in der Informatik:



Beispiel 1.1 (Problem "Flussüberquerung")**.**

Ein Mann steht mit einem Wolf, einer Ziege und einem Kohlkopf am linken Ufer eines Flusses, den er 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 einer gefressen. Ist es möglich, den Fluss zu überqueren, ohne dass die Ziege oder der Kohlkopf gefressen wird?

Lösungsansätze:

1. Knobeln → Lösung per "Geistesblitz"

2. Systematisches Vorgehen unter Verwendung von Informatik-Kalkülen

Hier: 2. Ansatz

Erste Analyse des Problems:

- relevante Objekte: Mann, Wolf, Ziege, Kohlkopf, Boot, Fluss, Ufer (links und rechts)

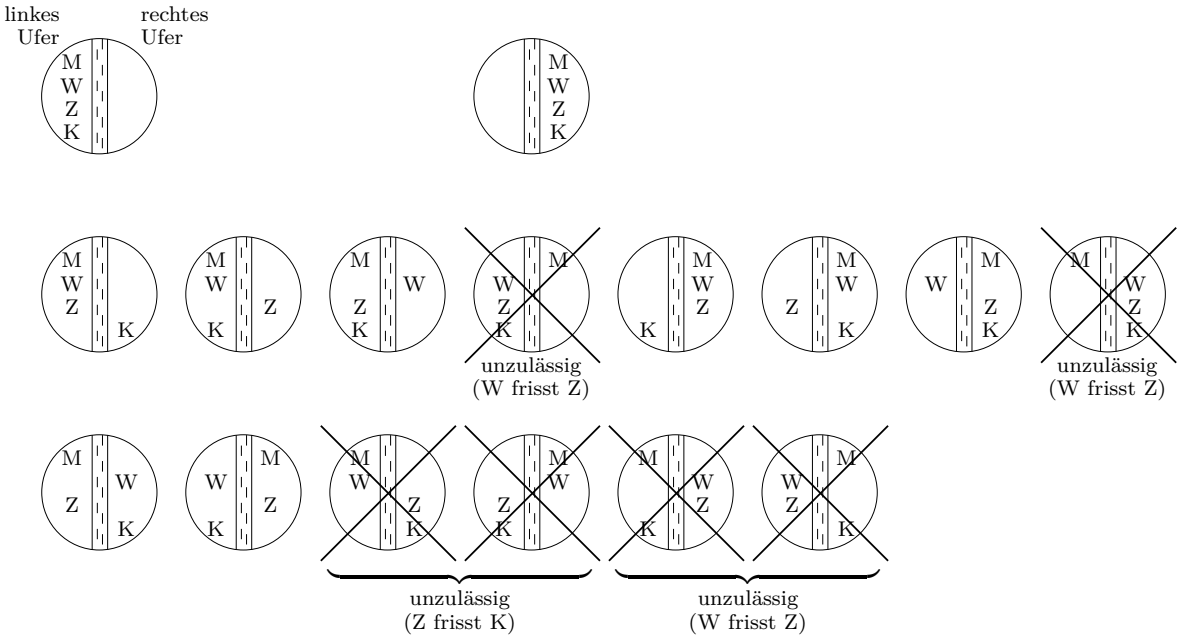
- Eigenschaften/Beziehungen:
  - Boot trägt Mann und zusätzlich maximal ein weiteres Objekt
  - Unbewacht am gleichen Ufer $\Longrightarrow$ Wolf frisst Ziege, Ziege frisst Kohlkopf
- Tätigkeit: Überqueren des Flusses
- Start: Mann, Wolf, Ziege, Kohlkopf, (Boot) am linken Ufer
- Ziel: Mann, Wolf, Ziege, Kohlkopf, (Boot) am rechten Ufer

Abstraktionen:

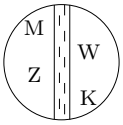
1. Nutze Abkürzungen:

   $M \quad \widehat{=} \quad$ Mann
   $W \quad \widehat{=} \quad$ Wolf
   $Z \quad \widehat{=} \quad$ Ziege
   $K \quad \widehat{=} \quad$ Kohlkopf

2. Betrachte die möglichen "Zustände", die auftreten dürfen:



3. Formale Modellierung der "Zustände": Repräsentiere den "Zustand"



   durch das Tupel $(\{M, Z\}, \{W, K\})$.

   Allgemein wird ein Zustand repräsentiert durch ein Tupel (l, r) mit $l \subseteq \{M, Z, W, K\}$ und $r \subseteq \{M, Z, W, K\}$, für das folgende Bedingungen erfüllt sind:

- $l \cup r = \{M, Z, W, K\}$
- $l \cap r = \varnothing$
- falls $Z, K \in l$, so auch $M \in l$ (um zu verhindern, dass K von Z gefressen wird)
- falls $Z, K \in r$, so auch $M \in r$ (um zu verhindern, dass K von Z gefressen wird)
- falls $W, Z \in l$, so auch $M \in l$ (um zu verhindert, dass Z von W gefressen wird)

(*)
- falls $W, Z \in r$, so auch $M \in r$ (um zu verhindert, dass Z von W gefressen wird)

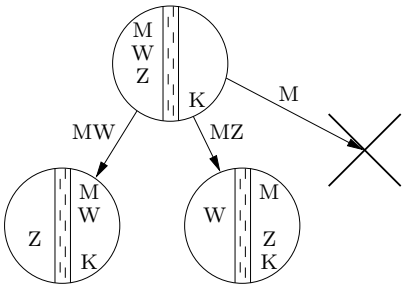
Übergänge von einem Zustand in einen anderen Zustand:

Vom Zustand $(\{M, W, Z\}, \{K\})$ aus kann man durch eine einzige Flussüberquerung in folgende Zustände gelangen:

- $(\{Z\}, \{M, W, K\})$, indem M und W im Boot fahren
- $(\{W\}, \{M, Z, K\})$, indem M und Z im Boot fahren

Beachte: wenn M allein fährt, tritt die Situation $(\{W, Z\}, \{M, K\})$ auf – dies ist aber laut (*) kein zulässiger Zustand.

Graphische Darstellung:



Insgesamt ergibt sich das in Abbildung 1.1 dargestellte Bild aus Zuständen und Zustandsübergängen.

Lösung des Problems "Flussüberquerung":

An diesem Bild lässt sich unser ursprüngliches Problem "Flussüberquerung" (Frage: Ist es möglich, den Fluss zu überqueren, ohne dass die Ziege oder der Kohlkopf gefressen werden?) ganz leicht lösen, indem man einfach einen Weg vom "Startzustand" zum "Zielzustand" sucht. Im Bild gibt es zwei verschiedene solche Wege; jeder davon kommt mit 7 Überfahrten aus.

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: in Kapitel 7.

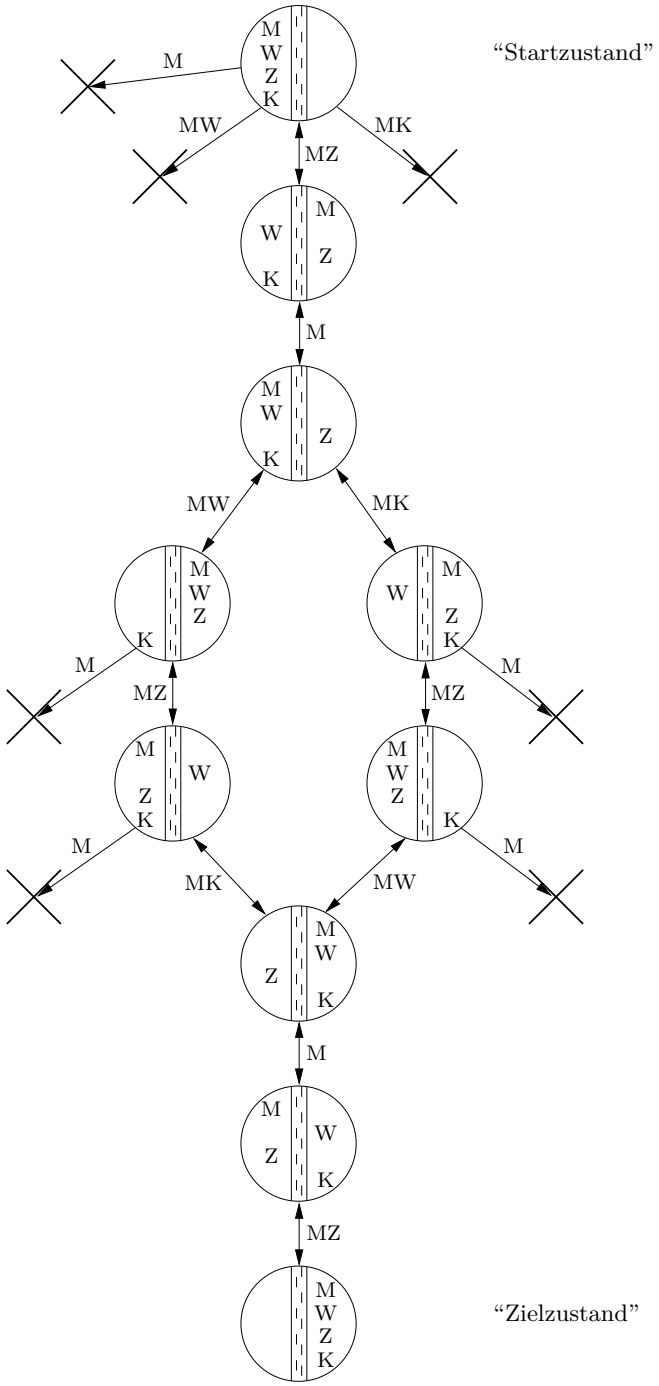


Abbildung 1.1: Übergänge zwischen den Zuständen beim Flussüberquerungsproblem

Diskussion dieses Modellierungsbeispiel:

- **Modellierung von Abläufen**/Folgen von Schritten durch ein Zustandsübergangsdiagramm
- **Abstraktion**: nur die Zustände und Übergänge interessieren
- **Relevante Objekte benannt**: M, W, Z, K
- Jeden **Zustand** haben wir hier **repräsentiert** durch **ein Paar** $(\mathbf{l}, \mathbf{r})$ **von Mengen** der Objekte (die sich am linken bzw. rechten Ufer befinden).
- Einteilung in **zulässige Zustände** und **unzulässige Zustände**
- Übergänge werden mit den transportierten Objekten beschriftet.

Besonders wichtig ist, was **nicht** modelliert wurde, weil es zur Lösung der Aufgabe irrelevant ist (z.B. Name, Breite, Tiefe des Flusses oder Länge, Geschwindigkeit des Boots etc.).

"Kreative Leistung": den Kalkül der Zustandsübergangsdiagramme wählen und die Bedeutung der Zustände und Übergänge festlegen.

Systematische Tätigkeit ("Routine-Arbeit"): das konkrete Zustandsübergangsdiagramm aufstellen und einen Weg vom Start- zum Zielzustand finden.

□Ende Beispiel 1.1

1.2 Ziele der Veranstaltung "Diskrete Modellierung"

- Überblick über grundlegende Modellierungsmethoden und -kalküle
- Verständnis des konzeptionellen Kerns der Kalküle
- Fähigkeit, Kalküle an typischen Beispielen anzuwenden
- Fähigkeit zur präzisen und formalen Ausdrucksweise bei der Analyse von Problemen
- Erkenntnis des praktischen Wertes präziser Beschreibungen

1.3 Allgemeiner Modellbegriff

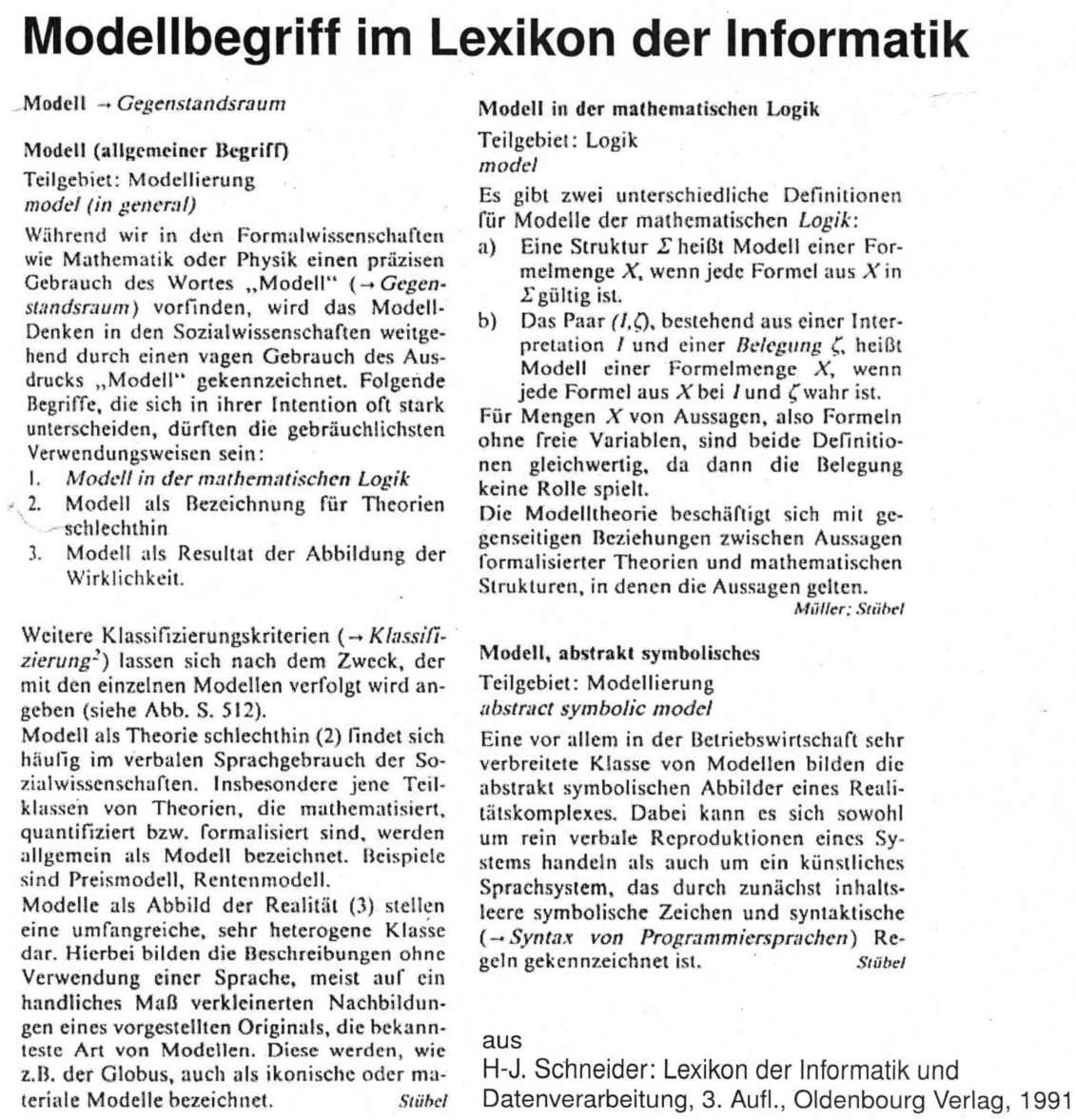
Modell (lat.: modulus, "Maß, Maßstab"): allgemeines Muster, Vorbild, Entwurf Modell

Beispielweise:

- Abbild eines vorhandenen Originals (z.B. Schiffsmodell)
- Vorbild für ein herzustellendes Original (z.B. Gelände in kleinem Maßstab; Vorbild in der Kunst)
- Konkretes Modell (z.B. Schiffsmodell) oder abstraktes Modell (z.B. Rentenmodell)

Beachte:

- Modelle sind **absichtlich** nicht originalgetreu. Sie heben bestimmte Eigenschaften hervor und lassen andere weg.



Modellbegriff im Lexikon der Informatik

Modell → *Gegenstandsraum*

Modell (allgemeiner Begriff)
Teilgebiet: Modellierung
model (in general)

Während wir in den Formalwissenschaften wie Mathematik oder Physik einen präzisen Gebrauch des Wortes „Modell" (→ *Gegenstandsraum*) vorfinden, wird das Modell-Denken in den Sozialwissenschaften weitgehend durch einen vagen Gebrauch des Ausdrucks „Modell" gekennzeichnet. Folgende Begriffe, die sich in ihrer Intention oft stark unterscheiden, dürften die gebräuchlichsten Verwendungsweisen sein:

1. *Modell in der mathematischen Logik*
2. Modell als Bezeichnung für Theorien schlechthin
3. Modell als Resultat der Abbildung der Wirklichkeit.

Weitere Klassifizierungskriterien (→ *Klassifizierung*[2]) lassen sich nach dem Zweck, der mit den einzelnen Modellen verfolgt wird angeben (siehe Abb. S. 512).
Modell als Theorie schlechthin (2) findet sich häufig im verbalen Sprachgebrauch der Sozialwissenschaften. Insbesondere jene Teilklassen von Theorien, die mathematisiert, quantifiziert bzw. formalisiert sind, werden allgemein als Modell bezeichnet. Beispiele sind Preismodell, Rentenmodell.
Modelle als Abbild der Realität (3) stellen eine umfangreiche, sehr heterogene Klasse dar. Hierbei bilden die Beschreibungen ohne Verwendung einer Sprache, meist auf ein handliches Maß verkleinerten Nachbildungen eines vorgestellten Originals, die bekannteste Art von Modellen. Diese werden, wie z.B. der Globus, auch als ikonische oder materiale Modelle bezeichnet. *Stübel*

Modell in der mathematischen Logik
Teilgebiet: Logik
model

Es gibt zwei unterschiedliche Definitionen für Modelle der mathematischen *Logik*:

a) Eine Struktur Σ heißt Modell einer Formelmenge X, wenn jede Formel aus X in Σ gültig ist.
b) Das Paar (I,ζ), bestehend aus einer Interpretation I und einer *Belegung* ζ, heißt Modell einer Formelmenge X, wenn jede Formel aus X bei I und ζ wahr ist.

Für Mengen X von Aussagen, also Formeln ohne freie Variablen, sind beide Definitionen gleichwertig, da dann die Belegung keine Rolle spielt.
Die Modelltheorie beschäftigt sich mit gegenseitigen Beziehungen zwischen Aussagen formalisierter Theorien und mathematischen Strukturen, in denen die Aussagen gelten.
Müller; Stübel

Modell, abstrakt symbolisches
Teilgebiet: Modellierung
abstract symbolic model

Eine vor allem in der Betriebswirtschaft sehr verbreitete Klasse von Modellen bilden die abstrakt symbolischen Abbilder eines Realitätskomplexes. Dabei kann es sich sowohl um rein verbale Reproduktionen eines Systems handeln als auch um ein künstliches Sprachsystem, das durch zunächst inhaltsleere symbolische Zeichen und syntaktische (→ *Syntax von Programmiersprachen*) Regeln gekennzeichnet ist. *Stübel*

aus
H-J. Schneider: Lexikon der Informatik und Datenverarbeitung, 3. Aufl., Oldenbourg Verlag, 1991

Abbildung 1.2: Modellbegriff im allgemeinen Lexikon

- Der intendierte Verwendungszweck des Modells bestimmt, welche Eigenschaften modelliert werden und welcher Kalkül dafür besonders geeignet ist.

Beispiel 1.2.

(a) Unterschiedliche Modelle, die beim Bau eines Hauses verwendet werden:

- Gebäudemodell: zur Vermittlung eines optischen Eindrucks

Modell [italien., zu lat. modulus „Maß, Maßstab"], allg. Muster, Vorbild, Entwurf.
– Mensch (auch Tier), der (das) als Vorbild für künstler. Studien oder Kunstwerke dient („sitzt").
– in der Bildhauerei meist in verkleinerter Form ausgeführter Entwurf einer Plastik oder Tonarbeit, die in Bronze gegossen werden soll. - †Architekturmodell.
– in der Modebranche Bez. für 1. ein nur einmal oder in eng begrenzter Anzahl hergestelltes Kleidungsstück. (*M.kleid*); 2. die Vorlage für eine Vervielfältigung; 3. svw. Mannequin.
– im Sprachgebrauch verschiedener Wiss. (Philosophie, Naturwiss., Soziologie, Psychologie, Wirtschaftswiss., Politikwiss., Kybernetik u.a.) ein Objekt materieller oder ideeller (Gedanken-M.) Natur, das von einem Subjekt auf der Grundlage einer Struktur-, Funktions- oder Verhaltensanalogie für ein anderes Objekt (*Original*) eingesetzt und genutzt wird, um Aufgaben zu lösen, deren Durchführung unmittelbar am Original selbst nicht möglich bzw. zu aufwendig ist (z. B. Flugzeug-M. im Windkanal). Die **Modellmethode** vollzieht sich in vier Schritten: 1. Auswahl (Herstellung eines dem [geplanten] Original entsprechenden M.; 2. Bearbeitung des M., um neue Informationen über das M. zu gewinnen (**Modellversuch**; †Ähnlichkeitsgesetze); 3. Schluss auf Informationen über das Original (meist Analogieschluß); ggf. 4. Durchführung der Aufgabe am Original. Infolge der Relationen zw. Subjekt, Original und M. (**Modellsystem**) ist ein M. einsetzbar u. a. zur Gewinnung neuer Informationen über das Original (z. B. Atom-M.), zur Demonstration und Erklärung (z. B. Planetarium), zur Optimierung des Originals (z. B. Netzplan), zur Überprüfung einer Hypothese oder einer techn. Konstruktion (z. B. Laborversuch). - Abweichend von diesem M.begriff versteht die *mathemat. Logik* unter M. eine Interpretation eines Axiomensystems, bei der alle Axiome dieses Systems wahre Aussagen darstellen. Diese **Modelltheorie** liefert grundlegende Verfahren zur Behandlung von Fragen der Vollständigkeit, Widerspruchsfreiheit und Definierbarkeit.

Abbildung 1.3: Modellbegriff im Lexikon der Informatik

- Grundriss: zur Einteilung der Räume und des Grundstückes
- Kostenplan: zur Finanzierung

(b) Frankfurter S- und U-Bahn Netzplan: siehe Abbildung 1.4

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

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

(c) Fahrplan der U4 an der Haltestelle "Bockenheimer Warte": siehe Abbildung 1.5

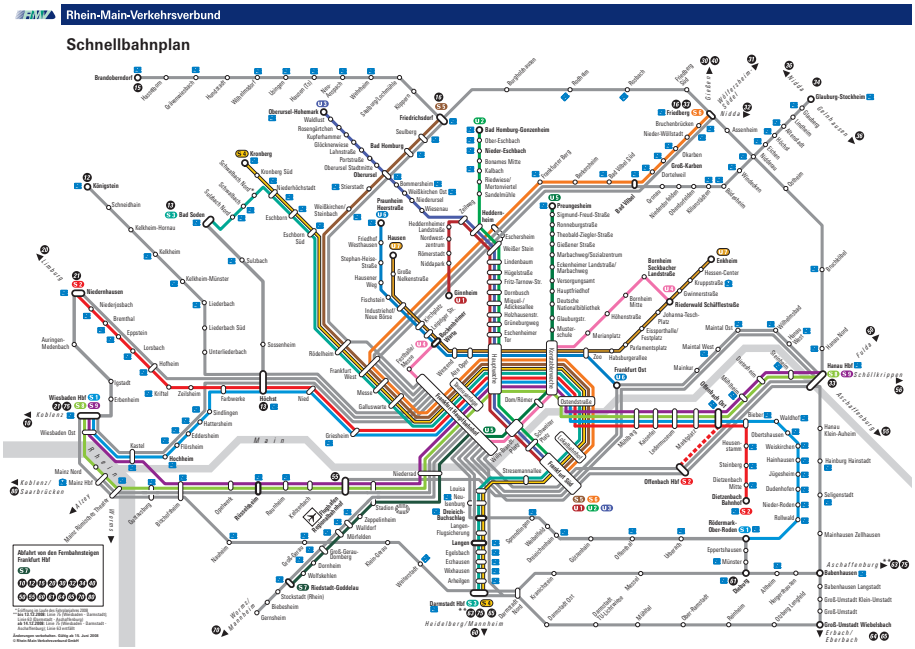


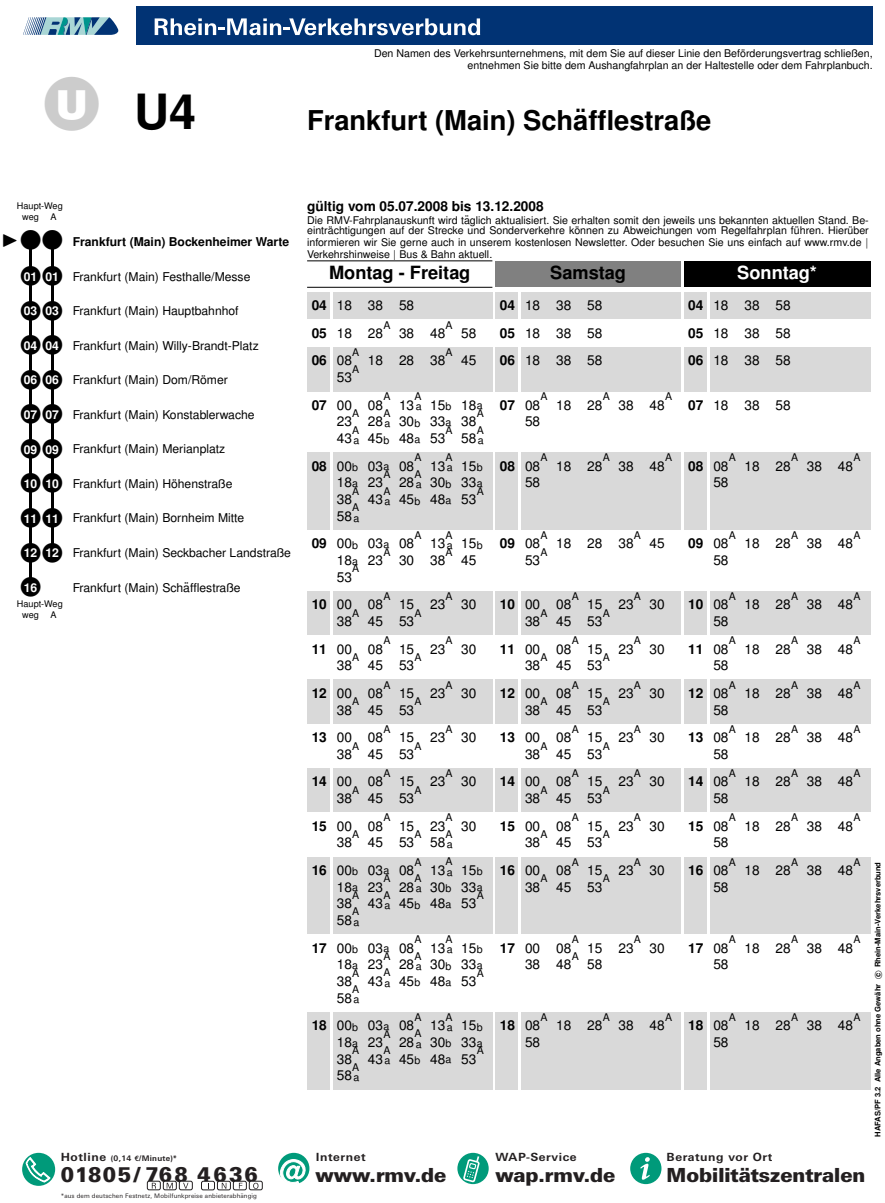
Abbildung 1.4: Schnellbahnplan des Rhein-Main-Verkehrsverbundes (RMV)

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 1.2

1.3.1 Arbeiten mit dem Modell

- Nutzung des Modells bei der Lösung eines Problems (z.B. Beispiel 1.1 “Flussüberquerung”)
- Bestimmte Aspekte eines komplexen Gebildes untersuchen und verstehen (z.B. Geschäftsabläufe in einer Firma)
- Verständigung zwischen Auftraggeber und Hersteller des Originals (z.B. Hausbau, Software-Entwicklung)
- Fixieren von Anforderungen für die Herstellung des Originals (z.B. Software: Spezifikation, Anforderungen)
- Durchführungen von Operationen, die man am Original nicht durchführen kann (z.B. Erprobung einer neuen Flügelform im Windkanal oder durch Computer-Simulation)
- Validierung des Modells (engl. Model Checking); Nachweis, dass die relevanten Eigenschaften des Originals korrekt und vollständig im Modell erfasst sind (z.B. Prüfung, ob ein Finanzplan alle Kosten erfasst, sie korrekt aufsummiert und die vorgegebene Kostengrenze eingehalten wird)



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	00 23A 43a	08A 28a 45b	13a 30b 48a	15b 33a 53A	18a 38A 58a	07	08A 58	18	28A	38	48A	07	18	38	58		
08	00b 18a 38A 58a	03a 23A 43a	08A 28a 45b	13a 30b 48a	15b 33a 53A	08	08A 58	18	28A	38	48A	08	08A 58	18	28A	38	48A
09	00b 18a 53A	03a 23A	08A 30	13a 38A	15b 45	09	08A 53A	18	28	38A	45	09	08A 58	18	28A	38	48A
10	00 38A	08A 45	15 53A	23A	30	10	00 38A	08A 45	15 53A	23A	30	10	08A 58	18	28A	38	48A
11	00 38A	08A 45	15 53A	23A	30	11	00 38A	08A 45	15 53A	23A	30	11	08A 58	18	28A	38	48A
12	00 38A	08A 45	15 53A	23A	30	12	00 38A	08A 45	15 53A	23A	30	12	08A 58	18	28A	38	48A
13	00 38A	08A 45	15 53A	23A	30	13	00 38A	08A 45	15 53A	23A	30	13	08A 58	18	28A	38	48A
14	00 38A	08A 45	15 53A	23A	30	14	00 38A	08A 45	15 53A	23A	30	14	08A 58	18	28A	38	48A
15	00 38A	08A 45	15 53A	23A 58a	30	15	00 38A	08A 45	15 53A	23A	30	15	08A 58	18	28A	38	48A
16	00b 18a 38A 58a	03a 23A 43a	08A 28a 45b	13a 30b 48a	15b 33a 53A	16	00 38A	08A 45	15 53A	23A	30	16	08A 58	18	28A	38	48A
17	00b 18a 38A 58a	03a 23A 43a	08A 28a 45b	13a 30b 48a	15b 33a 53A	17	00 38	08A 48A	15 58	23A	30	17	08A 58	18	28A	38	48A
18	00b 18a 38A 58a	03a 23A 43a	08A 28a 45b	13a 30b 48a	15b 33a 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 (R MV) (T N F G)
*aus dem deutschen Festnetz, Mobilfunkpreise anbieterabhängig

Internet www.rmv.de

WAP-Service wap.rmv.de

Beratung vor Ort Mobilitätszentralen

Abbildung 1.5: Fahrplan der U4 an der Haltestelle “Bockenheimer Warte”

1.3.2 Modellierte Aspekte

Ein Modell beschreibt nur bestimmte Aspekte des Originals, z.B.

- Struktur/Zusammensetzung des Originals (z.B. Organisationsschema einer Firma)

Dafür geeignete Kalküle: Wertebereiche, Entity-Relationship-Modell, Bäume

- Eigenschaften von Teilen des Originals (z.B. Farbe und Wert einer Spielkarte)

  Dafür geeignete Kalküle: Wertebereiche, Logik, Entity-Relationship-Modell

- Beziehungen zwischen Teilen des Originals (z.B. "Wolf frisst Ziege, Ziege frisst Kohlkopf")

  Dafür geeignete Kalküle: Graphen, Logik, Entity-Relationship-Modell

- Verhalten des Originals unter Operationen (z.B. aufeinanderfolgende Zustände bei wiederholter Flussüberquerung)

  Dafür geeignete Kalküle: Zustandsübergangsdiagramme, Petri-Netze, Graphen

1.4 Übungsaufgaben zu Kapitel 1

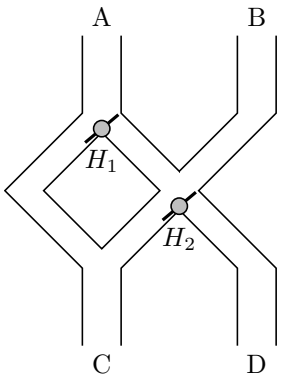
Aufgabe 1.1. Gegeben seien drei Stapel mit Büchern. Der erste besteht aus vier Büchern, der zweite aus sechs Büchern und der dritte aus 14 Büchern. Die Stapel sollen nun ausgeglichen werden, so dass auf jedem Stapel acht Bücher liegen. Allerdings dürfen in jedem Schritt nur Bücher zwischen genau zwei 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 1.1.

(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. 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.

Aus welcher Öffnung fällt die letzte Murmel, wenn

(a) fünf Murmeln fallen gelassen werden, wobei die erste und die vierte Murmel bei A und alle anderen Murmeln bei B fallen gelassen werden?

(b) sieben Murmeln fallen gelassen werden, wobei die erste, zweite, vierte und letzte Murmel bei A und alle anderen Murmeln bei B fallen gelassen werden?

Modellieren Sie zur Beantwortung der Fragen das Spiel analog zu Beispiel 1.1 durch ein Transitionssystem. Überlegen Sie sich zunächst die möglichen Zustände und Zustandsübergänge, die auftreten können.

Hinweis: Jeder Zustand sollte Informationen darüber enthalten, in welche Richtung die nächste Murmel rollt, wenn sie auf H_1 trifft, in welche Richtung die nächste Murmel rollt, wenn sie auf H_2 trifft und aus welcher Öffnung die zuvor fallen gelassene Murmel herausgerollt ist.

2 Modellierung mit Wertebereichen – mathematische Grundlagen und Beweistechniken

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 $\Longleftrightarrow$ $z \geq 50\%$
$\square$	markiert das Ende eines Beweises

Modellierung und Wertebereiche

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

Beispiel. Gegeben 3 Karten aus einem Kartenspiel, welches ist die höchste Karte?

Wertebereich

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

Beispiel 2.1 (Modellierung der Karten eines (Skat-)Kartenspiels)**.**
Wertebereiche:

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

Übersicht über Begriffe, die in Kapitel 2 genauer betrachtet werden:

- Wertebereich: eine Menge gleichartiger Werte
- Grundlegender Kalkül: Mengenlehre (Mengen und Mengenoperationen)
- Strukturen über Mengen zur Bildung von zusammengesetzten Wertebereichen:
  - Potenzmengen
  - Kartesische Produkte, Tupel
  - Relationen
  - Folgen, Wörter
  - Funktionen
- Verwendung dieses Kalküls
  - Modellierung von Strukturen und Zusammenhängen
  - Grundlage für alle anderen formalen Kalküle
  - abstrakte Grundlage für Typen in Programmiersprachen

Ziel von Kapitel 2 ist, diese Begriffe genauer zu betrachten und abgesehen davon einige wichtige mathematische Grundlagen und Beweistechniken zu erklären.

2.1 Mengen, Relationen und Funktionen

2.1.1 Was ist eine Menge?

Cantors naiver Mengenbegriff (Georg Cantor, 1845–1918)
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.

Notation. $m \in M :\Longleftrightarrow m$ ist Element der Menge M

Die Russellsche Antinomie (Bertrand Russell, 1872–1970)
Cantors Mengenbegriff ist problematisch und führt zu Widersprüchen. Russell gab folgendes Beispiel:

> 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 gilt $N \in N$ oder $N \notin N$.

Fall1: $N \notin N$. Gemäß Definition der Menge N gilt dann, dass $N \in N$. Das ist ein Widerspruch.

Fall2: $N \in N$. 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. $\Longrightarrow$ 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?

Um die Russellsche Antinomie zu vermeiden, muss man die Mengenlehre sehr vorsichtig axiomatisch aufbauen – dies sprengt allerdings den Rahmen dieser Vorlesung. Sofern man sich der Problematik aber bewusst ist, kann man sie im "täglichen Gebrauch" von Mengen vermeiden. Wir arbeiten daher weiter mit einem naiven Mengenbegriff.

2.1.2 Beschreibung/Definition von Mengen

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

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

- durch Angabe von charakteristischen Eigenschaften der Elemente der Menge (intensional), z.B.

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

Extensional lässt sich 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:

(a) $\{x : 0 \leq x \leq 5\}$ definiert nicht eindeutig eine Menge, weil nicht festgelegt ist, ob x beispielsweise eine ganze oder eine reelle Zahl ist.

(b) $\{M : M \text{ ist eine Menge}, M \notin M\}$ führt zur Russellschen Antinomie.

Fazit: 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.

2.1.3 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.

$$\{1,2,3\} \;=\; \{1,2,2,3\} \;=\; \{2,1,3\} \;=\; \{i : i \text{ ist eine ganze Zahl, } 0 \leq x \leq 3\}.$$

Insbesondere können Mengen aus atomaren oder aus zusammengesetzten Elementen gebildet werden, und eine Menge kann auch verschiedenartige Elemente enthalten.

Beispiel. Die Menge $M := \{1, (\text{Pik}, 8), \{\text{rot}, \text{blau}\}, 5\}$ besteht aus 4 Elementen:

- den atomaren Werten 1 und 5
- dem Tupel $(\text{Pik}, 8)$
- der Menge $\{\text{rot}, \text{blau}\}$

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

Beobachtung 2.2. Es gibt genau eine Menge, die keine Elemente enthält.

Definition 2.3 (leere Menge)**.** Die **leere Menge** ist die (eindeutig bestimmte) Menge, die kein(e) Element(e) enthält. Wir bezeichnen sie mit $\varnothing$. leere Menge

2.4 Frage: Gibt es eine "Menge aller Mengen"?

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).

2.1.4 Mengenalgebra

Definition 2.5 (Gleichheit von Mengen)**.** Zwei Mengen M und N sind gleich (kurz: $M = N$), falls sie dieselben Elemente enthalten, d.h. falls gilt:

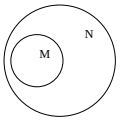
- f.a. $x \in M$ gilt $x \in N$, und
- f.a. $x \in N$ gilt $x \in M$.

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

Definition 2.6 (Teilmengen)**.** Seien M, N Mengen.

Teilmenge

(a) M ist eine **Teilmenge** von N (kurz: $M \subseteq N$), wenn jedes Element von M auch ein Element von N ist.



echte Teilmenge

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

Obermenge

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

echte Obermenge

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

Satz 2.7. *Seien M, N, P Mengen. Dann gilt:*

(a) $M = N \iff M \subseteq N$ *und* $M \supseteq N$.

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

Beweis:

(a)

$$
\begin{aligned}
M = N \quad &\overset{\text{Def. 2.5}}{\iff} \quad M \text{ und } N \text{ enthalten dieselben Elemente} \\
&\overset{\text{Def. 2.5}}{\iff} \quad \text{f.a. } x \in M \text{ gilt } x \in N \text{ und} \\
&\qquad\qquad\quad\ \text{f.a. } x \in N \text{ gilt } x \in M \\
&\iff \quad \text{jedes Element von } M \text{ ist auch ein Element von } N \text{ und} \\
&\qquad\qquad\quad\ \text{jedes Element von } N \text{ ist auch ein Element von } M \\
&\overset{\text{Def. 2.6(a)}}{\iff} \quad M \subseteq N \text{ und } N \subseteq M \\
&\overset{\text{Def. 2.6(c)}}{\iff} \quad M \subseteq N \text{ und } M \supseteq N
\end{aligned}
$$

(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 $m \in P$:

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

□

Definition 2.8. Seien M und N Mengen.

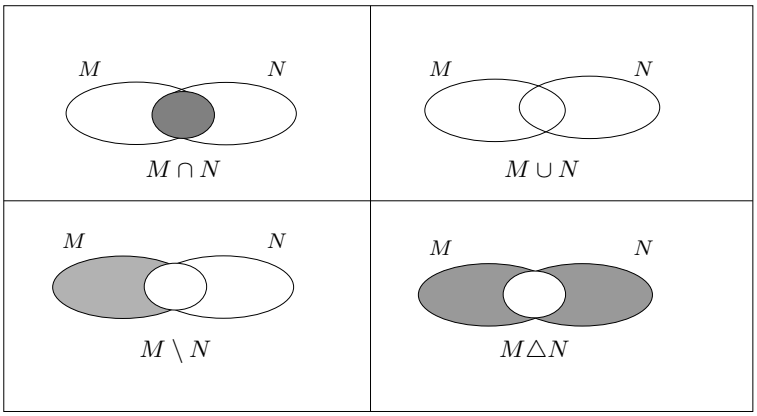
(a) (**Durchschnitt**) Durchschnitt
$M \cap N := \{x : x \in M \text{ und } x \in N\}$

(b) (**Vereinigung**) Vereinigung
$M \cup N := \{x : x \in M \text{ oder } x \in N\}$

(c) (**Differenz**) Differenz
$M \setminus N := M - N := \{x : x \in M \text{ und } x \notin N\}$

(d) (**Symmetrische Differenz**) Symmetrische Differenz
$M \triangle N := (M \setminus N) \cup (N \setminus M)$

Veranschaulichung durch Venn-Diagramme:



Notation 2.9. Zwei Mengen M und N heißen **disjunkt**, falls $M \cap N = \varnothing$, d.h. falls sie keine gemeinsamen Elemente besitzen. Manchmal schreiben wir $M \dot\cup N$, um $M \cup N$ zu bezeichnen und gleichzeitig auszudrücken, dass $M \cap N = \varnothing$.

Rechenregeln für Durchschnitt und Vereinigung

Satz 2.10. *Seien M, N, P Mengen. Dann gelten:*

(a) ***(Idempotenz)*** Idempotenz
$M \cap M = M$
$M \cup M = M$

(b) ***(Kommutativität)*** Kommutativität
$M \cap N = N \cap M$
$M \cup N = N \cup M$

(c) ***(Assoziativität)*** Assoziativität
$M \cap (N \cap P) = (M \cap N) \cap P$
$M \cup (N \cup P) = (M \cup N) \cup P$

(d) ***(Absorption)*** Absorption
$M \cap (M \cup N) = M$
$M \cup (M \cap N) = M$

Distributivität

(e) ***(Distributivität)***
$M \cap (N \cup P) = (M \cap N) \cup (M \cap P)$
$M \cup (N \cap P) = (M \cup N) \cap (M \cup P)$

Beweis:

(a)

$$\begin{aligned} M \cap M &\overset{\text{Def. 2.8(a)}}{=} \{x : x \in M \text{ und } x \in M\} \\ &= \{x : x \in M\} \\ &= M. \end{aligned}$$

Analog: $M \cup M = M$.

(b)

$$\begin{aligned} M \cap N &\overset{\text{Def. 2.8(a)}}{=} \{x : x \in M \text{ und } x \in N\} \\ &= \{x : x \in N \text{ und } x \in M\} \\ &\overset{\text{Def. 2.8(a)}}{=} N \cap M. \end{aligned}$$

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

(c)

$$\begin{aligned} M \cap (N \cap P) &\overset{\text{Def. 2.8(a)}}{=} \{x : x \in M \text{ und } x \in N \cap P\} \\ &\overset{\text{Def. 2.8(a)}}{=} \{x : x \in M \text{ und } (x \in N \text{ und } x \in P)\} \\ &= \{x : (x \in M \text{ und } x \in N) \text{ und } x \in P\} \\ &\overset{\text{Def. 2.8(a)}}{=} \{x : x \in M \cap N \text{ und } x \in P\} \\ &\overset{\text{Def. 2.8(a)}}{=} (M \cap N) \cap P. \end{aligned}$$

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

(d) Wir beweisen, dass $M \cap (M \cup N) = M$ in 2 Schritten:

Schritt 1: Zeige, dass $M \subseteq M \cap (M \cup N)$.
Schritt 2: Zeige, dass $M \cap (M \cup N) \subseteq M$.

Aus Satz 2.7(a) folgt dann, dass $M \cap (M \cup N) = M$.

Zu Schritt 1:

Behauptung: $M \subseteq M \cap (M \cup N)$, 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 2.8(b)). Wegen $m \in M$ und $m \in M \cup N$ gilt gemäß Definition 2.8(a), dass $m \in M \cap (M \cup N)$.

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 2.8(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.

□

2.1.5 Komplemente

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

Komplement

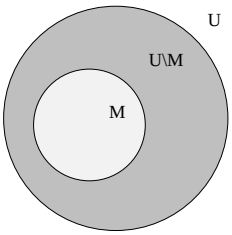
$$\overline{M} \;:=\; \{x : x \notin M\}$$

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

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

die "Menge aller Mengen" ... und dass es die nicht geben kann, haben wir in Frage 2.4 gesehen.

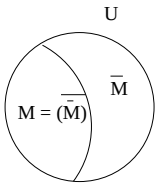
Daher betrachten wir Mengen stets innerhalb eines festen Universums U, das selber eine Menge ist. Für $M \subseteq U$ setzen wir dann $\overline{M} := U \setminus M$ und bezeichnen $\overline{M}$ als das Komplement von M in U.



Rechenregeln für Komplemente

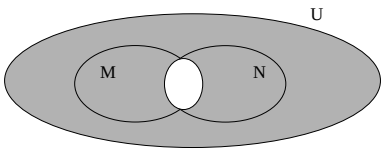
Satz 2.11. *Sei U unser festes Universum, das selbst eine Menge ist, und seien $M, N \subseteq U$. Dann gelten:*

(a) ***(Doppelte Negation)*** Doppelte Negation
$\overline{(\overline{M})} = M$

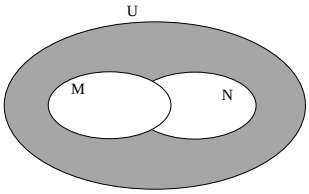


(b) ***(De Morgansche Regeln)*** De Morgansche Regeln

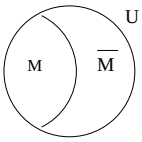
$\overline{M \cap N} = \overline{M} \cup \overline{N}$



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



(c) ***(Inversionsregeln)*** Inversionsregeln
$M \cap \overline{M} = \varnothing$
$M \cup \overline{M} = U$



Identitätsregeln

(d) ***(Identitätsregeln)***
$M \cap \overline{U} = M$
$M \cup \varnothing = M$

Beweis: Übung. □

2.1.6 Mächtigkeit/Kardinalität

Definition 2.12.

endlich

(a) Eine Menge heißt **endlich**, wenn sie nur endlich viele Elemente enthält, d.h. wenn es eine Zahl $n \in \mathbb{N}$ gibt, so dass die Menge genau n Elemente enthält.

Mächtigkeit
Kardinalität

(b) Die **Mächtigkeit** (oder **Kardinalität**) einer Menge M ist

$$|M| := \begin{cases} \text{Anzahl der Elemente in } M, & \text{falls } M \text{ endlich ist} \\ \infty \text{ (unendlich)}, & \text{sonst.} \end{cases}$$

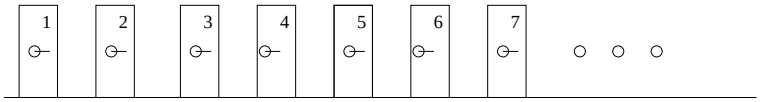
Notation 2.13. $\mathrm{Card}(M) := |M|$.

Beispiel 2.14.

- $|\{2, 4, 6\}| = 3$
- $|\varnothing| = 0$
- $|\{\varnothing\}| = 1$
- $|\mathbb{N}| = \infty$
- $|\mathbb{Z}| = \infty$
- $|\{2, 4, 6, 4\}| = 3$
- $|\{2, \{a, b\}\}| = 2$

Vorsicht beim Vergleich der Mächtigkeit unendlicher Mengen:

Hilberts Hotel (David Hilbert, 1862–1943)
Hilberts Hotel hat unendlich viele Zimmer, die fortlaufend mit $1, 2, 3, \ldots$ (also mit allen Zahlen aus $\mathbb{N}_{>0}$) nummeriert sind. Obwohl alle Zimmer belegt sind, schafft der Angestellte an der Rezeption es, für jeden neuen Gast 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.

2.1.7 Die Potenzmenge

Definition 2.15. Die **Potenzmenge** (engl.: power set) einer Menge M (kurz: $\mathcal{P}(M)$) ist die Menge aller Teilmengen von M. D.h.: Potenzmenge

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

Notation 2.16. In manchen Büchern wird $\mathcal{P}(M)$ auch mit $\mathrm{Pow}(M)$ (für "power set") oder mit 2^M bezeichnet.

Beispiel 2.17.

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

2.1.8 Paare, Tupel und kartesisches Produkt

Definition 2.18 (Paare und Tupel)**.**

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

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

(c) (Gleichheit zweier Tupel) F.a. $k, l \in \mathbb{N}$ und $a_1, \ldots, a_k, b_1, \ldots, b_l$ gilt:

$$(a_1, \ldots, a_k) = (b_1, \ldots, b_l) \;:\Longleftrightarrow\; k = l \text{ und } a_1 = b_1 \text{ und } a_2 = b_2 \text{ und } \ldots \text{ und } a_k = b_k.$$

Bemerkung 2.19.

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

(b) 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 2.20.

(a) Sei $k \in \mathbb{N}$ und sei M eine Menge. Die **k-te Potenz** von M ist die Menge k-te Potenz

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

Insbesondere: $M^0 = \{()\}$ besteht genau aus einem Element, dem leeren Tupel.

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

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

(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 2.21. Sei $M = \{a, b\}$ und $N = \{1, 2, 3\}$.

- $M \times N = \{(a,1), (a,2), (a,3), (b,1), (b,2), (b,3)\}$
- $M \times \{1\} = \{(a,1), (b,1)\}$
- $M \times \varnothing = \varnothing$
- $M^2 = \{(a,a), (a,b), (b,a), (b,b)\}$
- $M^1 = \{(a), (b)\}$
- $M^0 = \{()\}$
- $\varnothing^2 = \varnothing$
- $\varnothing^1 = \varnothing$
- $\varnothing^0 = \{()\}$
- Im Beispiel 2.1 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".

Die Mächtigkeit von kartesischen Produkten

Satz 2.22.

(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:

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

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

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

Beweis:

(a)

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

(b) analog

(c)

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

□

2.1.9 Worte bzw. endliche Folgen

Bemerkung 2.23. Sei A eine Menge.

- Gelegentlich fassen wir ein Tupel $(a_1, \ldots, a_k) \in A^k$ als **Wort** auf, dessen "Buchstaben" $a_1, \ldots, a_k$ sind. Um diese Sichtweise zu betonen, schreiben wir oft $a_1 \cdots a_k$.
  Beispiel: Das Tupel $(\mathrm{m}, \mathrm{o}, \mathrm{d}, \mathrm{e}, \mathrm{l}, \mathrm{l})$ identifizieren wir mit dem Wort modell. — Wort
- A ist dann das **Alphabet**, über dem die Worte gebildet werden, und $a_1 \cdots a_k$ wird "Wort über A" genannt. — Alphabet
- Das **leere Tupel** $() \in A^0$ heißt auch **leeres Wort** und wird oft als ε (epsilon) bezeichnet. — leeres Wort (ε)
- Die **Länge** eines Wortes $a_1 \cdots a_k$ ist die Zahl — Länge

  $$|a_1 \cdots a_k| \;:=\; k.$$

  Insbesondere ist $|\varepsilon| = 0$, d.h. das leere Wort hat die Länge 0.
- Sind $v = a_1 \cdots a_k$ und $w = b_1 \cdots b_l$ zwei Worte über A, so ist die **Konkatenation** von v und w das Wort — Konkatenation

  $$vw \;:=\; a_1 \cdots a_k b_1 \cdots b_l.$$
- Manchmal wird ein Wort $a_1 \cdots a_k$ auch als **Folge** der Länge k aufgefasst.

Definition 2.24 (A^*, A^+, Sprache)**.** Sei A ein Alphabet (d.h. eine Menge).

(a) Die **Menge aller Worte über** A (von beliebiger endlicher Länge) bezeichnen wir mit A^*. Es gilt also: — Menge aller Worte über A (A^*)

$$A^* \;=\; \bigcup_{k\in\mathbb{N}} A^k \;=\; \{a_1 \cdots a_k : k \in \mathbb{N},\, a_1, \ldots, a_k \in A\}.$$

Beachte: Wegen $0 \in \mathbb{N}$ und $A^0 = \{()\} = \{\varepsilon\}$ enthält A^* insbesondere das leere Wort.

A^+

(b) Die Menge aller **nicht-leeren** Worte über A (von beliebiger endlicher Länge) bezeichnen wir mit A^+. Es gilt:

$$A^+ \;=\; A^* \setminus \{\varepsilon\} \;=\; \{a_1 \cdots a_k : k \in \mathbb{N}_{>0},\, a_1, \ldots, a_k \in A\}.$$

Sprache

(c) Eine **Sprache** über A ist eine Teilmenge von A^*.

Bemerkung. In vielen Büchern werden Sprachen mit dem Buchstaben L (für **L**anguage) oder Varianten wie L' oder L_1 bezeichnet.

Beispiel 2.25 (Natürliche Sprachen)**.**
Alphabet:

$$A_{\text{deutsch}} \;:=\; \{\text{A, B, } \ldots, \text{ Z, Ä, Ö, Ü, a, b, } \ldots, \text{ z, ä, ö, ü, ß, ., ,, :, ;, !, ?, -, \_}\}$$

Sprachen über $A_{deutsch}$:

- $L_1 :=$ Menge aller grammatikalisch korrekten Sätze der deutschen Sprache (aufgefasst als Zeichenketten über A_{deutsch})
- $L_2 :=$ Menge aller Wörter der deutschen Sprache

Beispiel 2.26 (Programmiersprachen)**.**
Alphabet:

$$\text{ASCII} \;:=\; \text{Menge aller ASCII-Symbole}$$

Sprachen:

- $L_1 :=$ Menge aller JAVA-Schlüsselwörter
- $L_2 :=$ Menge aller erlaubten Variablennamen in JAVA
- $L_3 :=$ Menge aller syntaktisch korrekten JAVA-Programme

2.1.10 Relationen

Relationen sind Teilmengen von kartesischen Produkten. Präzise:

Definition 2.27.

Relation

(a) Seien M, N Mengen. Eine **Relation** von M nach N ist eine Teilmenge von $M \times N$.

Stelligkeit

(b) 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.

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

Beispiel 2.28. 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 $(24, 12, 2007)$ gehört, nicht aber das Tupel $(30, 2, 2008)$.

Notation 2.29.

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

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

  Beispiel:
  - $m \leq n$, für natürliche Zahlen m, n
  - $m \neq n$

- Ist R eine Relation auf $M_1, \ldots, M_k$, so schreiben wir manchmal

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

  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.

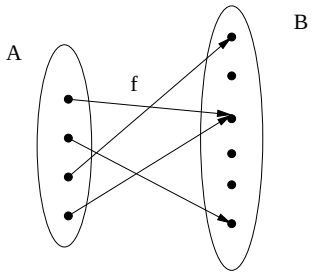
  Im Datums-Beispiel gilt: Gültig$(24, 12, 2007)$, aber es gilt **nicht**: Gültig$(30, 2, 2008)$.

2.1.11 Funktionen

Definition 2.30. Seien A, B Mengen. Eine **Funktion** (oder **Abbildung**) von A nach B ist eine Relation f von A nach 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.

Funktion
Abbildung

Anschaulich:



Notation 2.31.

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

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

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

Abb(A, B)

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

Beachte: In manchen Büchern wird $\mathrm{Abb}(A, B)$ auch mit $A \to B$ oder mit B^A bezeichnet.

Bemerkung. 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: $f(a) = g(a)$.

Definitionsbereich
Bildbereich
Bild

Definition 2.32 (Definitionsbereich, Bildbereich, Bild)**.** Sei $f\colon A \to B$. Der **Definitionsbereich** von f ist die Menge $\mathrm{Def}(f) := A$. Der **Bildbereich** von f ist die Menge B. Das **Bild** von f (genauer: das Bild von A unter f) ist die Menge $\mathrm{Bild}(f) := f(A) \overset{(\mathrm{Def})}{=} \{f(a) : a \in A\} \subseteq B$.

Restriktion
Einschränkung

Definition 2.33 (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

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

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

2.1.12 Partielle Funktionen

partielle Funktion

Definition 2.34. Eine **partielle Funktion** von einer Menge A in eine Menge B ist eine Funktion f mit $\mathrm{Def}(f) \subseteq A$ und $\mathrm{Bild}(f) \subseteq B$.

Bemerkung 2.35.

totale Funktion

(a) Im Gegensatz zu partiellen Funktionen nennt man Funktionen, wie wir sie in Definition 2.30 definiert haben, auch **totale Funktionen**.

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 das nicht zur Menge B gehört).

2.1.13 Eigenschaften von Funktionen

Definition 2.36. 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.

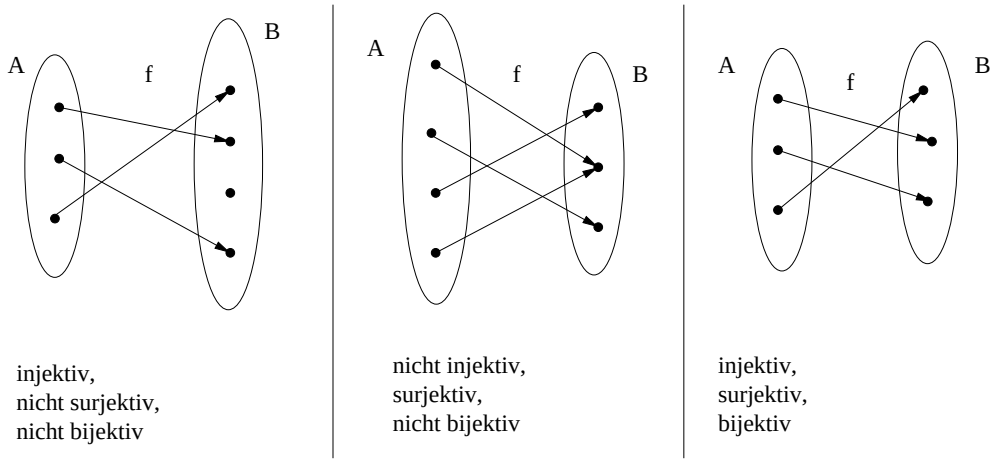
surjektiv

(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 2.37.

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

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

(b) Seien A und B **endliche** Mengen. Dann gilt:

$$|A| = |B| \iff \text{es gibt eine bijektive Funktion von } A \text{ nach } B.$$

Satz 2.38.

(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:

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

charakteristische Funktion

$$\chi_{M'}(m) := \begin{cases} 1, & \text{falls } m \in M' \\ 0, & \text{sonst.} \end{cases} \qquad (*)$$

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

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

Behauptung. f ist bijektiv.

Wir zeigen dies in 2 Schritten (und nutzen Beobachtung 2.37(a)).

Schritt 1: f ist injektiv:

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

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

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

$$m \in M' \iff m \in M''.$$

Somit ist $M' = M''$.

Schritt 2: f ist surjektiv:

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

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

Wir wählen: $M' := \{m \in M : h(m) = 1\}$. Dann ist klar: $M' \in \mathcal{P}(M)$. Gemäß (*) gilt $\chi_{M'} = h$. Gemäß (**) ist daher $f(M') = h$.

(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 2.39. 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 2.38(b) und Beobachtung 2.37(b) gilt für $k := |A|$, dass

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

Laut Satz 2.22(c) ist $|B^k| = |B|^k$. Somit $|\mathrm{Abb}(A, B)| = |B|^k = |B|^{|A|}$.

(b) Gemäß Satz 2.38(a) und Beobachtung 2.37(b) ist

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

Gemäß (a) ist

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

□

2.1.14 Spezielle Funktionen

Identitätsfunktion

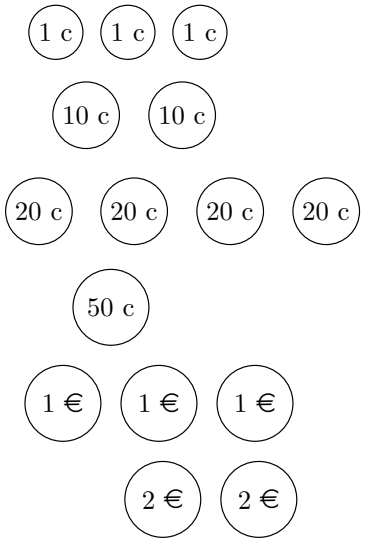
Definition 2.40. Die **Identitätsfunktion** auf einer Menge M ist die Funktion $\mathrm{id}_M\colon M \to M$ mit $\mathrm{id}_M(m) := m$, f.a. $m \in M$.

Multimenge

Definition 2.41 (Multimenge, engl.: bag)**.** Eine **Multimenge** über einer Menge M ist eine Funktion $f\colon M \to \mathbb{N}$.

Mit solchen Funktionen kann man "Mengen" beschreiben, in denen einzelne Elemente mehrfach vorkommen können: Für jedes $m \in M$ gibt $f(m)$ an, wie oft m in der "Multimenge" vorkommt.

Beispiel 2.42. Ein Geldbeutel mit



- 3 1-Cent-Münzen
- 0 2-Cent-Münzen
- 0 5-Cent-Münzen
- 2 10-Cent-Münzen
- 4 20-Cent-Münzen
- 1 50-Cent-Münzen
- 3 1-Euro-Münzen
- 2 2-Euro-Münzen

kann repräsentiert werden durch die Multimenge

$$\text{Geldbeutelinhalt}\colon \text{MünzenArten} \to \mathbb{N},$$

wobei

$$\text{MünzenArten} \;:=\; \{1\text{c}, 2\text{c}, 5\text{c}, 10\text{c}, 20\text{c}, 50\text{c}, 1€, 2€\}$$

und

$$\begin{array}{lcl}
\text{GeldbeutelInhalt}(1\text{c}) & := & 3 \\
\text{GeldbeutelInhalt}(2\text{c}) & := & 0 \\
\text{GeldbeutelInhalt}(5\text{c}) & := & 0 \\
\text{GeldbeutelInhalt}(10\text{c}) & := & 2 \\
\text{GeldbeutelInhalt}(20\text{c}) & := & 4 \\
\text{GeldbeutelInhalt}(50\text{c}) & := & 1 \\
\text{GeldbeutelInhalt}(1€) & := & 3 \\
\text{GeldbeutelInhalt}(2€) & := & 2.
\end{array}$$

Bequemere Schreibweise:

$$\text{GeldbeutelInhalt} \;:=\; \{(1\text{c}, 3), (2\text{c}, 0), (5\text{c}, 0), (10\text{c}, 2), (20\text{c}, 4), (50\text{c}, 1), (1€, 3), (2€, 2)\}.$$

2.2 Ein Beispiel zur Modellierung mit Wertebereichen

Beispiel 2.43 (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 erst mal 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 Index 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:
$$\text{Delegierte} \;:=\; \text{Nationen} \times \text{DelegiertenIndex},$$
wobei $\text{DelegiertenIndex} := \{1, 2, 3\}$.
- 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
$$\text{Vertretene\_Nationen\_in\_}a \;:=\; \{n \in \text{Nationen} : \text{es ex. ein } i \in \text{DelegiertenIndex} \\ \text{s.d. } (n, i) \in \text{AK-Besetzung}(a)\}.$$
- Die Bedingung, dass es für jeden Arbeitskreis eine Sprache gibt, die alle Mitglieder des Arbeitskreises beherrschen, lässt sich folgendermaßen formulieren:
$$\text{f.a.}\;\; a \in \text{Arbeitskreise} \;\; \text{ist} \;\; \text{Gemeinsame\_Sprachen\_in\_}a \neq \varnothing,$$
wobei
$$\text{Gemeinsame\_Sprachen\_in\_}a \;:=\; \{\text{sp} \in \text{Sprachen} : \text{f.a. } d \in \text{AK-Besetzung}(a) \\ \text{ist sp} \in \text{spricht}(d)\}.$$

□Ende Beispiel 2.43

2.3 Beweise verstehen und selbst formulieren

Ziel dieses Abschnitts ist, einen kurzen Überblick über grundlegende Beweistechniken zu geben, insbesondere:

- direkter Beweis
- Beweis durch Kontraposition
- Beweis durch Widerspruch (indirekter Beweis)
- vollständige Induktion.

2.3.1 Was sind "Sätze" und "Beweise"?

Satz
Theorem
Beweis

Ein **Satz** (bzw. **Theorem**) besteht aus Voraussetzungen und einer Behauptung. Voraussetzungen und Behauptung sind Aussagen, so dass folgendes gilt: Wenn alle Voraussetzungen erfüllt sind, dann muss auch 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

Typische Fehler, die man beim Versuch, Beweise zu formulieren, **vermeiden** sollte, sind:

- unzulässiges Argumentieren mit Beispielen
- Verwendung gleicher Symbole zur Bezeichnung verschiedener Dinge
- Hantieren mit nicht exakt oder gar widersprüchlich definierten Begriffsbildungen
- unzulässige Gedankensprünge beim Schlussfolgern
- Ausnutzung von bis dahin noch unbewiesenen Behauptungen zur Begründung von einzelnen Beweisschritten.

2.3.2 Beweistechnik "direkter Beweis"

Ansatz: die Behauptung eines Satzes wird "direkt" (d.h. ohne "Umwege") bewiesen.

Beispiele für direkte Beweise haben wir in dieser Vorlesung bereits kennengelernt, z.B.

- der Beweis von Satz 2.7
- der Beweis von Satz 2.22
- der Beweis von Satz 2.38
- der Beweis von Folgerung 2.39.

2.3.3 Beweistechnik "Beweis durch Kontraposition"

Beim Beweis durch Kontraposition wird ein Satz der Form "Falls Aussage A gilt, so gilt auch Aussage B" dadurch bewiesen, dass man zeigt: "Falls Aussage B **nicht** gilt, so kann auch Aussage A **nicht** gelten." Als Beispiel für einen Beweis durch Kontraposition betrachten wir folgenden Satz.

Satz 2.44. *Für jedes $n \in \mathbb{N}$ gilt: Falls n^2 eine ungerade Zahl ist, so ist auch n eine ungerade Zahl.*

Beweis: Durch Kontraposition. Sei $n \in \mathbb{N}$ beliebig.

Wir zeigen: Falls n **keine** ungerade Zahl ist, so ist auch n^2 **keine** ungerade Zahl.

$n \in \mathbb{N}$ war beliebig gewählt. Falls n ungerade ist, so ist nichts weiter zu beweisen. Wir betrachten daher nun den Fall, dass n **nicht** ungerade ist (d.h. n ist gerade) und zeigen, dass dann auch n^2 gerade ist.

Beachte: Per Definition ist eine natürliche Zahl m genau dann **gerade**, wenn es ein $k \in \mathbb{N}$ gibt, s.d. $m = 2 \cdot k$.

Daher gilt:

$$\begin{aligned}
n \text{ ist gerade} &\implies \text{es ex. } k \in \mathbb{N} \text{ s.d. } n = 2 \cdot k \quad \text{(gemäß Def. von "gerade")} \\
&\implies \text{es ex. } k \in \mathbb{N} \text{ s.d. } n^2 = n \cdot (2 \cdot k) \\
&\implies \text{es ex. } k \in \mathbb{N} \text{ s.d. } n^2 = 2 \cdot (n \cdot k) \\
&\implies \text{es ex. } k' \in \mathbb{N} \text{ s.d. } n^2 = 2 \cdot k' \\
&\implies n^2 \text{ ist gerade.} \quad \text{(gemäß Def. von "gerade")}
\end{aligned}$$

Somit ist n^2 gerade, d.h. n^2 ist keine ungerade Zahl. □

2.3.4 Beweistechnik "Beweis durch Widerspruch" (indirekter Beweis)

Beim Beweis durch Widerspruch wird ein Satz der Form "Falls die Voraussetzungen A erfüllt sind, so gilt Aussage B" dadurch bewiesen, dass man

- annimmt, dass die Voraussetzungen A erfüllt sind, aber die Aussage B **nicht** gilt, und
- daraus einen Widerspruch herleitet.

Als Beispiel für einen Beweis durch Widerspruch betrachten wir folgenden Satz:

Satz 2.45. *Für alle geraden natürlichen Zahlen a und b gilt: $a \cdot b$ ist gerade.*

Beweis: Durch Widerspruch. Angenommen, a und b sind gerade natürlichen Zahlen, so dass $a \cdot b$ **nicht** gerade ist. Da a und b gerade sind, gibt es $k, l \in \mathbb{N}$ s.d. $a = 2 \cdot k$ und $b = 2 \cdot l$. Dann ist $a \cdot b = (2 \cdot k) \cdot (2 \cdot l)$. Insbesondere gibt es also ein $k' \in \mathbb{N}$, s.d. $a \cdot b = 2 \cdot k'$. Gemäß der Definition von "gerade" ist also $a \cdot b$ gerade. Dies ist ein Widerspruch zur Annahme, dass $a \cdot b$ **nicht** gerade ist. □

Ein weiteres, etwas anspruchsvolleres Beispiel für einen Beweis durch Widerspruch ist der Beweis des folgenden Satzes, der "anschaulich" besagt, dass die Potenzmenge von $\mathbb{N}$ **viel** größer ist als die Menge $\mathbb{N}$ selbst.

Satz 2.46 ("$\mathcal{P}(\mathbb{N})$ ist nicht abzählbar"). *Es gibt keine surjektive Funktion von $\mathbb{N}$ nach $\mathcal{P}(\mathbb{N})$.*

Beweis: Durch Widerspruch. Angenommen, $f\colon \mathbb{N} \to \mathcal{P}(\mathbb{N})$ ist surjektiv. Sei

$$M := \{n \in \mathbb{N} : n \notin f(n)\}. \tag{*}$$

Klar: $M \in \mathcal{P}(\mathbb{N})$. Da f surjektiv ist, muss es ein $m \in \mathbb{N}$ geben mit $f(m) = M$. Klar: Entweder gilt $m \in M$ oder es gilt $m \notin M$.

Fall 1: $m \notin M$:
Wegen $f(m) = M$ gilt also $m \notin f(m)$. Gemäß (*) für $n := m$ gilt also $m \in M$. $\lightning$ (Widerspruch zu "Fall 1: $m \notin M$").

Fall 2: $m \in M$:
Wegen $f(m) = M$ gilt also: $m \in f(m)$. Gemäß (*) für $n := m$ gilt also $m \notin M$. $\lightning$ (Widerspruch zu "Fall 2: $m \in M$").

Somit führen beide Fälle zu einem Widerspruch. Daher kann es keine surjektive Funktion f von $\mathbb{N}$ nach $\mathcal{P}(\mathbb{N})$ geben. □

Ein weiteres, sehr ähnliches Beispiel für einen Beweis durch Widerspruch haben wir bereits im Zusammenhang mit der Russellschen Antinomie kennengelernt.

Satz 2.47 ("Es gibt keine Menge aller Mengen"). *Es gibt keine Menge U, so dass für jede Menge M gilt: $M \in U$.*

Beweis: Durch Widerspruch. Angenommen, U ist eine Menge, so dass für jede Menge M gilt: $M \in U$. Dann ist auch

$$N := \{M \in U : M \text{ ist eine Menge und } M \notin M\} \tag{*}$$

eine Menge. Insbesondere gilt entweder $N \in N$ oder $N \notin N$.

Fall 1: $N \notin N$: Wir wissen: N ist eine Menge, also insbesondere $N \in U$. Und da wir in Fall 1 sind, gilt außerdem: $N \notin N$. Gemäß (*) (für $M := N$) muss dann aber gelten: $N \in N$. $\lightning$ (Widerspruch zu "Fall 1: $N \notin N$").

Fall 2: $N \in N$: Wegen $N \in N$ gilt gemäß (*) für $M := N$, dass $N \in U$ ist, dass N eine Menge ist, und dass $N \notin N$ ist. $\lightning$ (Widerspruch zu "Fall 2: $N \in N$").

Somit führen beide Fälle zu einem Widerspruch. Daher kann es keine Menge U geben, so dass für jede Menge M gilt: $M \in U$. □

2.3.5 Beweistechnik "Beweis durch vollständige Induktion"

Grundidee der vollständigen Induktion

Für $n \in \mathbb{N}$ sei $A(n)$ eine Aussage über die natürliche Zahl n.

Ziel: Zeige, dass für alle $n \in \mathbb{N}$ die Aussage $A(n)$ wahr ist.

Eine Möglichkeit, dies zu zeigen ist, sich das so genannte Induktionsprinzip zu Nutze zu machen.

Induktionsprinzip

Man zeigt, dass eine Aussage $A(n)$ für alle $n \in \mathbb{N}$ wahr ist, indem man folgendermaßen vorgeht:

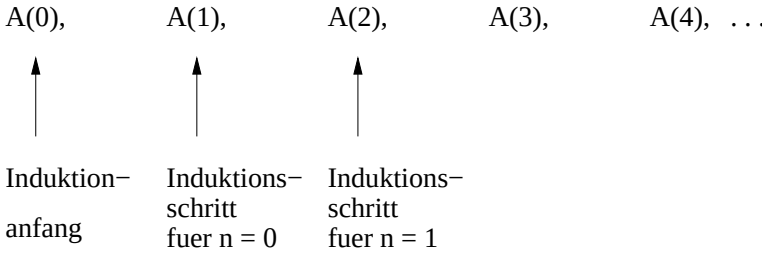
(1) Zuerst zeigt man, dass die Aussage $A(n)$ für die Zahl $n = 0$ gilt.

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.

Diesen Schritt nennt man "Induktionsschritt".

Beachte: Wenn man die Schritte (1) und (2) bewiesen hat, so weiß man, dass die folgenden Aussagen wahr sind:



d.h. man hat gezeigt, dass **für alle** $n \in \mathbb{N}$ die Aussage $A(n)$ wahr ist.

Beispiel für einen Beweis durch vollständige Induktion

Satz 2.48. *F.a.* $n \in \mathbb{N}$ *gilt:* $\sum_{i=0}^{n} 2^i = 2^{(n+1)} - 1$.

(Bemerkung: Die "Aussage $A(n)$*", deren Gültigkeit hier f.a.* $n \in \mathbb{N}$ *bewiesen werden soll, besagt also: "*$\sum_{i=0}^{n} 2^i = 2^{(n+1)} - 1$*".*

Zur Erinnerung: $\sum_{i=0}^{n} 2^i$ *ist eine abkürzende Schreibweise für* $2^0 + 2^1 + 2^2 + \cdots + 2^n$*.)*

Beweis: Per Induktion nach n.

INDUKTIONSANFANG: $n = 0$

Behauptung: $\sum_{i=0}^{0} 2^i = 2^{0+1} - 1$.

Beweis:

- $\sum_{i=0}^{0} 2^i = 2^0 = 1$
- $2^{0+1} - 1 = 2^1 - 1 = 2 - 1 = 1$

Also: $\sum_{i=0}^{0} 2^i = 1 = 2^{0+1} - 1$.

INDUKTIONSSCHRITT: $n \to n+1$

Sei $n \in \mathbb{N}$ beliebig.

Induktionsannahme: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

(D.h. wir gehen davon aus, dass die Aussage $A(n)$ wahr ist.)

Behauptung: $\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1$

(D.h. wir müssen zeigen, dass dann auch die Aussage $A(n+1)$ wahr ist.)

Beweis:

$$\begin{aligned}
\sum_{i=0}^{n+1} 2^i &\overset{\text{Def.}\sum}{=} \left(\sum_{i=0}^{n} 2^i\right) + 2^{(n+1)} \\
&\overset{\text{Ind.ann.}}{=} 2^{n+1} - 1 + 2^{n+1} \\
&= 2 \cdot 2^{n+1} - 1 \\
&= 2^{(n+1)+1} - 1.
\end{aligned}$$

□

Zwei nützliche Varianten des Induktionsprinzips

Um zu zeigen, dass eine Aussage $A(n)$ für alle $n \in \mathbb{N}$ mit $n \geq n_0$ wahr ist (wobei n_0 eine geeignete natürliche Zahl ist), kann man nach einem der beiden folgenden Schemata vorgehen:

Variante 1:

Induktionsanfang: $n = n_0$

Behauptung: Die Aussage $A(n_0)$ ist wahr.

Beweis: ...

Induktionsschritt: $n \to n+1$

Sei $n \in \mathbb{N}$ mit $n \geq n_0$ beliebig.

Induktionsannahme: Die Aussage $A(n)$ ist wahr.

Behauptung: Die Aussage $A(n+1)$ ist wahr.

Beweis: ...

Variante 2:

Induktionsanfang: $n = n_0$

Behauptung: Die Aussage $A(n_0)$ ist wahr.

Beweis: ...

Induktionsschritt: $n \to n+1$

Sei $n \in \mathbb{N}$ mit $n \geq n_0$ beliebig.

Induktionsannahme: Die Aussagen $A(n_0), A(n_0+1), \ldots, A(n)$ sind wahr.

Behauptung: Die Aussage $A(n+1)$ ist wahr.

Beweis: ...

Beispiel 2.49. Welche der Funktionen $f\colon \mathbb{N} \to \mathbb{Z}$ und $g\colon \mathbb{N} \to \mathbb{Z}$ mit $f(n) := n^2 - 7$ und $g(n) := 4 \cdot n$ (f.a. $n \in \mathbb{N}$) liefert größere Funktionswerte?

n	0	1	2	3	4	5	6	7	8	9
$f(n)$	-7	-6	-3	2	9	18	29	42	57	74
$g(n)$	0	4	8	12	16	20	24	28	32	36

Vermutung. F.a. $n \in \mathbb{N}$ mit $n \geq 6$ gilt: $f(n) > g(n)$.

Beweis: per Induktion nach n.

INDUKTIONSANFANG: $n = 6$
Behauptung: $f(6) > g(6)$
Beweis:

- $f(6) = 6^2 - 7 = 29$
- $g(6) = 4 \cdot 6 = 24$

Also: $f(6) = 29 > 24 = g(6)$.

INDUKTIONSSCHRITT: $n \to n + 1$
Sei $n \in \mathbb{N}$ mit $n \geq 6$ beliebig.
Induktionsannahme: $f(n) > g(n)$, d.h. $n^2 - 7 > 4 \cdot n$.
Behauptung: $f(n+1) > g(n+1)$, d.h. $(n+1)^2 - 7 > 4 \cdot (n+1)$.
Beweis:

$$\begin{aligned}
(n+1)^2 - 7 &= n^2 + 2n + 1 - 7 \\
&= (n^2 - 7) + 2n + 1 \\
&\overset{\text{Ind.ann}}{>} 4n + 2n + 1 \\
&\overset{n \geq 6, \text{ also } 2n+1 \geq 13 > 4}{\geq} 4n + 4 \\
&= 4(n+1).
\end{aligned}$$

□

Auf ähnliche Weise kann man per Induktion auch Folgendes beweisen.

Satz 2.50.

(a) F.a. $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$.

(b) F.a. $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\sum_{i=1}^{n} (2i - 1) = n^2$
(d.h. die Summe der ersten n ungeraden Zahlen ergibt gerade die Zahl n^2).

(c) F.a. $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\sum_{i=1}^{n} i^2 = \frac{n \cdot (n+1) \cdot (2n+1)}{6}$.

Beweis: Übung. □

Das folgende Beispiel zeigt, dass man beim Führen von Induktionsbeweisen vorsichtig bzw. sehr sorgfältig sein sollte:

Beispiel 2.51. Der folgende Satz ist offensichtlich nicht wahr – aber wo steckt der Fehler im Beweis?

"Satz": *F.a.* $n \in \mathbb{N}$ *mit* $n \geq 1$ *gilt: Ist* M *eine Menge von Menschen mit* $|M| = n$*, so haben alle Menschen in* M *die gleiche Größe.*

"Beweis": Per 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 $n \geq 1$ 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 $|M'| = n$ und $|M''| = n$. Gemäß der Induktionsannahme gilt daher:

(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 insbes. 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? □Ende Beispiel 2.51

2.4 Rekursive Definitionen von Funktionen und Mengen

2.4.1 Rekursive Definitionen von Funktionen

Das Induktionsprinzip lässt sich auch zur "induktiven" (bzw. "rekursiven") Definition von Funktionen $f\colon \mathbb{N} \to M$ (wobei M eine beliebige Menge ist) nutzen, indem man folgendermaßen vorgeht:

(1) Definiere $f(0)$. ("Rekursionsanfang")

(2) Definiere, f.a. $n \in \mathbb{N}$, $f(n+1)$ unter Verwendung des Werts $f(n)$ (bzw. unter Verwendung der Werte $f(n), f(n-1), \ldots, f(1), f(0)$). ("Rekursionsschritt")

Auch hier ist wieder eine Reihe von Varianten möglich.

Beispiel 2.52.

Fakultätsfunktion

(a) Rekursive Definition der **Fakultätsfunktion**:

Aufgabe: n Studenten $s_1, \ldots, s_n$ sollen so auf n PCs $c_1, \ldots, c_n$ verteilt werden, dass an jedem PC genau ein Student sitzt.

Frage: Wie viele Möglichkeiten gibt es?

Antwort: $\mathrm{fak}(n)$, wobei

- $\mathrm{fak}(1) \;=\; 1$ und
- $\mathrm{fak}(n+1) \;=\; (n+1) \cdot \mathrm{fak}(n)$ (für alle $n \in \mathbb{N}_{>0}$).

(Insbesondere ist $\mathrm{fak}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$.)

Beachte:

$$\mathrm{fak}(4) \;=\; 4 \cdot \mathrm{fak}(3) \;=\; 4 \cdot 3 \cdot \mathrm{fak}(2) \;=\; 4 \cdot 3 \cdot 2 \cdot \mathrm{fak}(1) \;=\; 4 \cdot 3 \cdot 2 \cdot 1.$$

Allgemein gilt f.a. $n \in \mathbb{N}_{>0}$:

$$\mathrm{fak}(n) \;=\; n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1 \;=\; \prod_{i=1}^{n} i.$$

Notation. $n! \;:=\; \mathrm{fak}(n)$

Fibonacci-Folge

(b) Rekursive Definition der so genannten **Fibonacci-Folge**:
(Leonardo Fibonacci, ital. Mathematiker, 13. Jh.)

Fragestellung: 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 weibliche Kaninchen hat der Bauer am Ende des n-ten Monats, wenn er mit einem neu geborenen weiblichen Kaninchen startet?

Antwort: $\mathrm{fib}(n)$

Rekursive Definition der Fibonacci-Folge: $\mathrm{fib}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ mit

- $\mathrm{fib}(1) \;:=\; 1$,
- $\mathrm{fib}(2) \;:=\; 1$ und
- $\mathrm{fib}(n+1) \;:=\; \mathrm{fib}(n) + \mathrm{fib}(n-1)$ (f.a. $n \in \mathbb{N}$, $n \geq 2$)

Somit:

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

Um Aussagen über rekursiv definierte Funktionen zu beweisen, kann man wieder das Induktionsprinzip nutzen. Beispiel:

Satz 2.53. *Sei* $\text{fib}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ *die Fibonacci-Folge. Dann gilt f.a.* $n \in \mathbb{N}_{>0}$*:* $\text{fib}(n) \leq 2^n$.

Beweis: Per Induktion nach n.

INDUKTIONSANFANG: betrachte $n = 1$ und $n = 2$

Behauptung: $\text{fib}(1) \leq 2^1$ und $\text{fib}(2) \leq 2^2$.

Beweis: $\text{fib}(1) \overset{\text{Def.}}{=} 1 \leq 2 = 2^1, \quad \text{fib}(2) \overset{\text{Def.}}{=} 1 \leq 4 = 2^2$.

INDUKTIONSSCHRITT: $n \to n+1$

Sei $n \in \mathbb{N}$ mit $n \geq 2$ beliebig.

Induktionsannahme: F.a. $i \in \mathbb{N}_{>0}$ mit $i \leq n$ gilt: $\text{fib}(i) \leq 2^i$.

Behauptung: $\text{fib}(n+1) \leq 2^{n+1}$.

Beweis: $\text{fib}(n+1) \overset{\text{Def.}}{=} \text{fib}(n) + \text{fib}(n-1) \overset{\text{Ind.ann.}}{\leq} 2^n + 2^{n-1} \leq 2 \cdot 2^n = 2^{n+1}$. □

Bemerkung 2.54. Es gibt auch eine "geschlossene Formel", mit der man den n-ten Wert der Fibonacci-Folge, d.h. die Zahl $\text{fib}(n)$ direkt ausrechnen kann, ohne dafür sämtliche Werte $\text{fib}(0), \text{fib}(1), \ldots, \text{fib}(n-1)$ ausrechnen zu müssen:
F.a. $n \in \mathbb{N}_{>0}$ gilt:

$$\text{fib}(n) = \frac{1}{\sqrt{5}} \cdot \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right).$$

Beweis: Übung (... per Induktion nach n).
(Details: siehe Buch von Meinel und Mundhenk) □

2.4.2 Rekursive Definitionen von Mengen

Oft ist es nützlich, auch **Mengen** rekursiv (bzw. induktiv) zu definieren. Eine rekursive Definition einer Menge M besteht aus:

(a) **Basisregeln** der Form "$m \in M$".
(D.h. die Basisregeln listen explizit bestimmte 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.

Beispiel 2.55 ("Palindrome")**.** 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:

- $a \in \text{PAL}$ (Basisregel (B1))
- Rek. Regel (R2) mit $w := a \implies bab \in \text{PAL}$
- Rek. Regel (R1) mit $w := bab \implies ababa \in \text{PAL}$
- Rek. Regel (R1) mit $w := ababa \implies aababaa \in \text{PAL}$

□

Aber beispielsweise $aab \notin \text{PAL}$, denn gemäß Basisregeln und rekursiven Regeln gilt für jedes Wort $w \in \text{PAL}$: der erste und der letzte Buchstabe von w sind identisch. □Ende Beispiel 2.55

Induktionsprinzip für rekursiv definierte Mengen

Sei M eine rekursiv definierte Menge. Dass eine Aussage $A(m)$ für alle $m \in M$ wahr ist, kann man folgendermaßen zeigen:

(1) Zuerst betrachtet man nacheinander jede Basisregel der Form "$m \in M$" und zeigt, dass die Aussage $A(m)$ wahr ist. ("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. ("Induktionsschritt")

Beachte: 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 2.56. 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 2.55 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 $v := awa$. Nach (R1) ist damit auch $v \in \text{PAL}$.

  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): Sei $w \in \text{PAL}$ und sei $v := bwb$. Nach (R2) ist damit auch $v \in \text{PAL}$.

  Induktionsannahme: $w = w^R$

  Behauptung: $v = v^R$

  Beweis: $v^R \overset{\text{Def. } v}{=} (bwb)^R \overset{\text{Def. } (\cdot)^R}{=} bw^Rb \overset{\text{Ind.ann.: } w = w^R}{=} bwb \overset{\text{Def. } v}{=} v.$

□

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 $|w| \leq n$, so gilt $w \in \text{PAL}$.

Im Induktionsanfang werden $n = 0$ und $n = 1$ betrachtet; im Induktionsschritt $n \to n+1$ werden alle $n \geq 1$ betrachtet.

Details: Übung. □

Aus Behauptung 1 und Behauptung 2 folgt: $\text{PAL} = \{w \in A^* : w = w^R\}$. □Ende Beispiel 2.56

Antwort auf die Frage aus Beispiel 2.51:

Der "Induktionsschritt $n \to n + 1$" ist für den Wert $n = 1$ nicht schlüssig, denn in diesem Fall gilt $n + 1 = 2$ und

- $M = \{a_1, a_2\}$
- $M' = \{a1\}$
- $M'' = \{a2\}$

Insbesondere gilt also zwar, dass $a_2 \in M''$, aber es gilt nicht, dass $a_2 \in M'$!

2.5 Übungsaufgaben zu Kapitel 2

Aufgabe 2.1. 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) $\mathcal{P}(\{\varnothing\})$ (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 2.2. Sei $U := \{1, 2, \ldots, 10\}$ ein festes Universum, und seien $M := \{1, 3, 5\}$, $N := \{2, 3, 5, 7\}$ und $P := \{1, 4, 9\}$. Schreiben Sie jede der folgenden Mengen in extensionaler Form auf und geben Sie ihre Kardinalität an.

(a) $M \setminus (N \cup P)$ (d) $(M \cap \overline{P}) \cup (N \cap \overline{P})$ (g) $M \times P \times \{a, b\}$
(b) $(M \setminus N) \cup (M \setminus P)$ (e) $M^2 \setminus (N \times P)$ (h) $\{Q : Q \subseteq N, |Q| = 3\}$
(c) $(M \cup N) \cap \overline{P}$ (f) $\mathcal{P}(N)$

Aufgabe 2.3. 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:

(a) Falls $M \subseteq N$ und $N \subsetneqq P$, dann $M \subsetneqq P$.

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

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

Aufgabe 2.4.

(a) Welche der Gleichungen stimmt, welche stimmt nicht?
   a) $(M \cap N) \setminus P = (M \setminus P) \cap (N \setminus P)$
   b) $(M \cap N) \setminus P = (M \setminus P) \cup (N \setminus P)$

(b) Begründen Sie Ihre Antwort aus (a) durch Betrachtung von Venn-Diagrammen.

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

Aufgabe 2.5.

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

(b) Beweisen Sie, dass Ihre Antworten aus (a) korrekt sind.

Aufgabe 2.6. Seien A, B, C, D, E Teilmengen von $\mathbb{N}$, die wie folgt definiert sind:

$$A = \{3n : n \in \mathbb{N}\} \quad B = \{5n : n \in \mathbb{N}\} \quad C = \{15n : n \in \mathbb{N}\}$$
$$D = \{6n : n \in \mathbb{N}\} \quad E = \{12n : n \in \mathbb{N}\}$$

(a) Welche der folgenden Aussagen sind richtig und welche sind falsch?

$$\text{(i) } E \subseteq D \subseteq A \quad \text{(ii) } E \subseteq C \quad \text{(iii) } A \cap B \subseteq C \quad \text{(iv) } A \cup B \subseteq C$$

(b) Berechnen Sie die folgenden Mengen:

$$\text{(i) } A \cup C \quad \text{(ii) } A \cap E \quad \text{(iii) } B \cap D \quad \text{(iv) } C \setminus B$$

Aufgabe 2.7. Ein Informatikstudent hat 30 Informatikbücher von der Bibliothek ausgeliehen, die sich u.a. mit den Gebieten Algorithmik, Betriebssysteme und Compilerbau beschäftigen. Sei A die Menge der Bücher, die sich u.a. mit Algorithmik beschäftigen, B die Menge der Bücher, die sich u.a. mit Betriebssystemen beschäftigen und C die Menge der Bücher, die sich u.a. mit Compilerbau beschäftigen. Folgende Information über die Anzahl der Bücher und die von ihnen behandelten Themen ist bekannt:

$$|A| = 14, \quad |B| = 18, \quad |C| = 16, \quad |A \cap B| = 8, \quad |A \cap C| = 7, \quad |B \cap C| = 10, \quad |A \cap B \cap C| = 3.$$

(a) Wie viele der Bücher enthalten Material aus mindestens einem der genannten Gebiete? D.h. berechnen Sie $|A \cup B \cup C|$.

(b) Wie viele der Bücher enthalten Material aus mindestens zwei der genannten Gebiete? D.h. berechnen Sie $|D|$, wobei $D := (A \cap B) \cup (A \cap C) \cup (B \cap C)$.

(c) Wie viele der Bücher enthalten Material aus genau einem der genannten Gebiete? D.h. berechnen Sie $|(A \cup B \cup C) \setminus D|$, wobei D die Menge aus (b) ist.

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

Aufgabe 2.8.

(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. Wieviele 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 $f(x) := x - 4$ für alle $x \in \mathbb{Z}$
   b) $f\colon \mathbb{Z} \to \mathbb{Z}$ mit $f(x) := 2 \cdot x$ für alle $x \in \mathbb{Z}$
   c) $f\colon \mathbb{Z} \to \mathbb{Z}$ mit $f(x) := x^2$ für alle $x \in \mathbb{Z}$
   d) $f\colon A^* \to \mathbb{N}$ für eine beliebige Menge A mit $|A| \geq 2$ und $f(w) := |w|$ für alle $w \in A^*$

(d) Wie viele Möglichkeiten gibt es,

   a) 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?
   b) 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?

Aufgabe 2.9. Beweisen Sie Satz 2.38(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 2.10. 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 2.11. Beweisen Sie: Falls M eine endliche Teilmenge einer unendlichen Menge U ist, so ist das Komplement von M in U unendlich.

Aufgabe 2.12. Beweisen Sie, dass für alle Mengen A, B, C mit $A = B \cup C$ gilt: Falls A unendlich ist, so ist B oder C unendlich.

Aufgabe 2.13. Beweisen Sie folgendes durch vollständige Induktion nach n.

(a) Für alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\displaystyle\sum_{i=1}^{n} (2i - 1) = n^2$.

(b) Für alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\displaystyle\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$.

(c) Für alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $\sum_{i=1}^{n}(4i-1) = 2n^2 + n$.

(d) Für alle $x \in \mathbb{R}$ mit $x \geq -1$ und alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt: $1 + n \cdot x \leq (1+x)^n$.

Aufgabe 2.14. Ein möglicher Algorithmus, um für eine Zahl $n \in \mathbb{N}_{>0}$ den Wert $\text{fib}(n)$ der Fibonacci-Folge zu berechnen, ist:

Algo 1 (bei Eingabe einer Zahl $n \in \mathbb{N}_{>0}$):

1. Falls $n = 1$ oder $n = 2$, dann gib 1 als Ergebnis zurück.
2. Falls $n \geq 3$, dann:
3. Sei x_1 die Ausgabe von *Algo 1* bei Eingabe der Zahl $n-1$.
4. Sei x_2 die Ausgabe von *Algo 1* bei Eingabe der Zahl $n-2$.
5. Gib den Wert $(x_1 + x_2)$ als Ergebnis zurück.

Der Algorithmus benötigt bei Eingabe einer Zahl n höchstens $g_1(n)$ Schritte, wobei

$$\begin{aligned} g_1(1) &= 1 \quad \text{und} \quad g_1(2) = 1 \quad \text{und} \\ g_1(n) &= 3 + g_1(n-1) + g_1(n-2) \quad \text{für alle } n \in \mathbb{N} \text{ mit } n \geq 3 \end{aligned}$$

(der Einfachheit halber zählen die Zeilen 1, 2 und 5 hier jeweils nur als ein Schritt).

Ein anderer Algorithmus, um für eine Zahl $n \in \mathbb{N}_{>0}$ den Wert $\text{fib}(n)$ zu berechnen, ist:

Algo 2 (bei Eingabe einer Zahl $n \in \mathbb{N}_{>0}$):

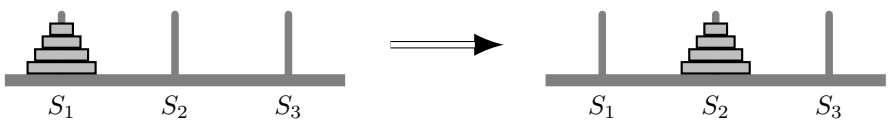
1. Falls $n = 1$ oder $n = 2$, dann gib 1 als Ergebnis zurück.
2. Seien $a_0 := 0$, $a_1 := 1$ und $a_2 := 1$.
3. Wiederhole für alle i von 3 bis n:
4. Ersetze a_0 durch a_1 und a_1 durch a_2.
5. Ersetze a_2 durch $a_0 + a_1$.
6. Gib den Wert a_2 als Ergebnis zurück.

Dieser Algorithmus benötigt bei Eingabe $n \in \mathbb{N}_{>0}$ höchstens $g_2(n) := 2 \cdot (n-2) + 3$ Schritte (wie oben zählen wir die Zeilen 1, 2, 4, 5 und 6 jeweils als nur einen Schritt).

(a) Welcher der beiden Algorithmen läuft im Allgemeinen schneller? D.h. welche der beiden Funktionen g_1 und g_2 liefert kleinere Funktionswerte?

(b) Beweisen Sie, dass Ihre Antwort aus (a) korrekt ist. D.h. falls Sie in (a) geantwortet haben, dass *Algo i* im Allgemeinen schneller als *Algo j* ist, dann finden Sie eine Zahl $n_0 \in \mathbb{N}_{>0}$ und beweisen Sie per Induktion nach n, dass für alle $n \in \mathbb{N}$ mit $n \geq n_0$ gilt: $g_i(n) < g_j(n)$.

Aufgabe 2.15 (Türme von Hanoi)**.** Ein Turm aus $n \in \mathbb{N}_{>0}$ unterschiedlich großen gelochten Scheiben soll von einem Stab (S_1) auf einen zweiten Stab (S_2) unter Zuhilfenahme eines Hilfsstabes (S_3) verschoben werden (das folgende Bild zeigt die Situation für den Fall $n = 4$).



Dabei müssen die folgenden Regeln beachtet werden:

- Pro Zug darf nur eine Scheibe bewegt werden. Es kann also immer nur die oberste Scheibe eines Turmes bewegt werden.
- Es darf nie eine größere Scheibe auf einer kleineren Scheibe liegen.

(a) Beschreiben Sie, wie der Turm im Fall $n = 4$ von S_1 nach S_2 verschoben werden kann.

(b) Beweisen Sie, dass es für alle $n \in \mathbb{N}_{>0}$ möglich ist, die n Scheiben von S_1 nach S_2 zu verschieben.

Hinweis: Beweisen Sie zuerst durch vollständige Induktion nach n, dass die folgende Aussage für alle $n \in \mathbb{N}$ mit $n \geq 1$ gilt:

A(n): Seien $i, j \in \{1, 2, 3\}$ mit $i \neq j$, sei $m \in \mathbb{N}$ mit $m \geq n$, und seien m Scheiben so auf die drei Stäbe verteilt, dass gilt:

- Auf S_i liegen mindestens n Scheiben.
- Die Scheiben auf den beiden anderen Stäben sind größer als die obersten n Scheiben auf S_i.

Dann lassen sich die obersten n Scheiben von S_i so nach S_j verschieben, dass keine der anderen Scheiben bewegt wird.

Aufgabe 2.16. Sei die Sprache L über dem Alphabet $A := \{(,)\}$ wie folgt rekursiv definiert:

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) Welche der folgenden Wörter gehören zu L und welche nicht?

- ()
- ()()
- ((
- (())
- ())(
- (((())()

(b) Beweisen Sie, dass (()(()())) $\in L$ ist.

(c) 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: $|w|_(= |w|_)$.

(d) Beweisen Sie, dass ()(()() $\notin L$ ist.

Aufgabe 2.17. 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: $\mathbf{x}i := \mathbf{x}j + c \ \in L$.
	(B2)	Für Zahlen $i, j, c \in \mathbb{N}$ gilt: $\mathbf{x}i := \mathbf{x}j - c \ \in L$.
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) $\mathbf{x}3 := \mathbf{x}7 - 2$

(ii) $\mathbf{x}3 := 1\mathbf{;}\ \mathbf{x}2 := \mathbf{x}3 + 5$

(iii) $\mathbf{while}\ \mathbf{x}1 \neq 0\ \mathbf{do}\ \mathbf{x}0 := \mathbf{x}0 + 1\mathbf{;}\ \mathbf{x}1 := \mathbf{x}1 - 1\ \mathbf{end}$

(iv) $\mathbf{x}1 := \mathbf{x}1 + 42\mathbf{;}\ \mathbf{while}\ \mathbf{x}1 \neq 0\ \mathbf{do}\ \mathbf{x}1 := \mathbf{x}1 - 1$

(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: $|w|_{\mathbf{do}} = |w|_{\mathbf{end}}$.

3 Aussagenlogik

3.1 Wozu "Logik" im Informatik-Studium?

Logik

Logik (altgriechisch: "Logos": "Vernunft")

- "die Lehre des vernünftigen Schlussfolgerns"
- Teilgebiete der
  - Philosophie
  - Mathematik
  - Informatik
  - Linguistik
- zentrale Frage: "Wie kann man Aussagen miteinander verknüpfen, und auf welche Weise kann man formal Schlüsse ziehen und Beweise durchführen?"
- These: Logik spielt für die Informatik eine ähnlich wichtige Rolle wie die Differentialrechnung für die Physik
- Anwendungsbereiche der Logik in der Informatik: z.B.
  - 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 2 "Agenten", die nach einem gewissen "Protokoll" abläuft, "sicher" ist – etwa gegen Abhören, Anwendungsbeispiel: Internet-Banking)
  - als Grundlage für Datenbank-Anfragesprachen
  - als Bestandteil von Programmiersprachen (z.B. um "Bedingungen" in "IF-Anweisungen" zu formulieren)
  - zur automatischen Generierung von Beweisen (so genannte "Theorembeweiser")

Aussagenlogik

Aussagen
Junktoren
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.

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. Andererseits 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 (und welche) werden im besten Fall zur Party kommen?

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

3.2 Syntax und Semantik der Aussagenlogik

Syntax: legt fest, welche Zeichenketten (Worte) Formeln der Aussagenlogik sind — Syntax

Semantik: legt fest, welche "Bedeutung" einzelne Formeln haben — Semantik

(vgl. "Syntax" und "Semantik" von JAVA-Programmen: die Syntax legt fest, welche Zeichenketten JAVA-Programme sind; Semantik bestimmt, was das Programm tut)

Definition 3.2 (Aussagenvariablen und Alphabet der Aussagenlogik)**.**

(a) Eine **Aussagenvariable** (kurz: Variable) hat die Form V_i, für $i \in \mathbb{N}$. Die Menge aller Variablen bezeichnen wir mit $\textsc{Avar}$, d.h. $\textsc{Avar} := \{V_i : i \in \mathbb{N}\}$. — Aussagenvariable

(b) Das Alphabet der Aussagenlogik ist

$$A_{\mathrm{AL}} \;:=\; \textsc{Avar} \cup \{\mathbf{0}, \mathbf{1}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}.$$

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: $X \in \mathrm{AL}$.

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

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

Notation 3.4.

atomare Formel
Atom

(a) $\mathbf{0}$, $\mathbf{1}$ und die Variablen (d.h. die Elemente aus $\textsc{Avar}$) bezeichnen wir als **atomare Formeln** bzw. **Atome**.

Junktor

(b) Die Symbole $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ 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

Negation
- $\neg\varphi$ **Negation** (bzw. ver-NEIN-ung) von φ

Beispiel 3.5. Beispiele für Formeln (d.h. Worte über A_{AL}, die zur Menge AL gehören):

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

Aber beispielsweise ist

$$V_1 \vee V_2 \wedge V_3$$

keine Formel (d.h. kein Element in AL), da die Klammern fehlen. Auch $(\neg V_1)$ ist **keine** Formel, da die Klammern "zu viel sind". □Ende von Beispiel 3.5

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:

Variablenmenge

Definition 3.6. Die **Variablenmenge** einer aussagenlogischen Formel φ (kurz: $\mathrm{Var}(\varphi)$) ist die Menge aller Variablen $X \in \textsc{Avar}$, die in φ vorkommen.

Beispiel.

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

Definition 3.7.

(a) Eine **Belegung** (bzw. **Wahrheitsbelegung**) ist eine partielle Funktion von AVAR nach $\{0,1\}$. — Belegung, Wahrheitsbelegung

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

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

Intuitive Bedeutung: 1 steht für "wahr" und 0 steht für "falsch".

Definition 3.8 (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. $X \in \mathrm{AVAR}$ 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
  - $[\![(\varphi \wedge \psi)]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 1 \text{ und } [\![\psi]\!]^{\mathcal{B}} = 1 \\ 0, & \text{sonst} \end{cases}$
  - $[\![(\varphi \vee \psi)]\!]^{\mathcal{B}} := \begin{cases} 0, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 0 \text{ und } [\![\psi]\!]^{\mathcal{B}} = 0 \\ 1, & \text{sonst} \end{cases}$
  - $[\![(\varphi \to \psi)]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = 0 \text{ oder } [\![\psi]\!]^{\mathcal{B}} = 1 \\ 0, & \text{sonst} \end{cases}$
  - $[\![(\varphi \leftrightarrow \psi)]\!]^{\mathcal{B}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}} \\ 0, & \text{sonst} \end{cases}$

Intuitive Bedeutung der Semantik:

- **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 $\mathcal{B}$). Darstellung durch eine so genannte **Verknüpfungstafel** (bzw. **Wahrheitstafel**):

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\neg\varphi]\!]^{\mathcal{B}}$
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 Verknüpfungstafel:

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

- **Disjunktion:** $(\varphi \vee \psi)$ 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 Verknüpfungstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \vee \psi)]\!]^{\mathcal{B}}$
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 von $\mathcal{B}$).

  Zugehörige Verknüpfungstafel:

$[\![\varphi]\!]^{\mathcal{B}}$	$[\![\psi]\!]^{\mathcal{B}}$	$[\![(\varphi \to \psi)]\!]^{\mathcal{B}}$
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 von $\mathcal{B}$).

  Zugehörige Verknüpfungstafel:

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

Beispiel 3.9. Betrachte die Formel $\varphi := \big(\neg V_0 \vee (V_5 \to V_1)\big)$. 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

$$
\begin{aligned}
[\![\varphi]\!]^{\mathcal{B}} &\overset{\text{Def. 3.8}}{=} \begin{cases} 1, & \text{falls } [\![\neg V_0]\!]^{\mathcal{B}} = 1 \text{ oder } [\![(V_5 \to V_1)]\!]^{\mathcal{B}} = 1 \\ 0, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 3.8}}{=} \begin{cases} 1, & \text{falls } [\![V_0]\!]^{\mathcal{B}} = 0 \text{ oder } ([\![V_5]\!]^{\mathcal{B}} = 0 \text{ oder } [\![V_1]\!]^{\mathcal{B}} = 1) \\ 0, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 3.8}}{=} \begin{cases} 1, & \text{falls } \mathcal{B}(V_0) = 0 \text{ oder } \mathcal{B}(V_5) = 0 \text{ oder } \mathcal{B}(V_1) = 1 \\ 0, & \text{sonst} \end{cases} \\
&= \quad 1 \qquad \text{(denn gemäß obiger Wahl von } \mathcal{B} \text{ gilt } \mathcal{B}(V_5) = 0).
\end{aligned}
$$

Beobachtung 3.10. Sind $\mathcal{B}$ und $\mathcal{B}'$ zwei Belegungen für eine Formel φ, die auf $\mathrm{Var}(\varphi)$ übereinstimmen (d.h. f.a. $X \in \mathrm{Var}(\varphi)$ gilt $\mathcal{B}(X) = \mathcal{B}'(X)$), so ist $[\![\varphi]\!]^{\mathcal{B}} = [\![\varphi]\!]^{\mathcal{B}'}$.

In der Literatur wird diese Beobachtung oft unter dem Namen "**Koinzidenzlemma**" geführt. 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). Einen formalen Beweis der Beobachtung kann man leicht per Induktion über den Aufbau von AL führen. Aufgrund der Beobachtung des "Koinzidenzlemmas" werden wir im Folgenden, wenn wir Belegungen $\mathcal{B}$ für eine Formel φ betrachten, uns meistens nur für diejenigen Werte $\mathcal{B}(X)$ interessieren, für die $X \in \mathrm{Var}(\varphi)$ ist.

Anmerkung 3.11 (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
Aussprache	epsilon	iota	pi	Delta	Gamma	Sigma	Pi	Phi

Um umgangssprachlich formuliertes Wissen (vgl. Beispiel 3.1 "Geburtstagsfeier") durch aussagenlogische Formeln zu repräsentieren, sind folgende Konventionen bequem:

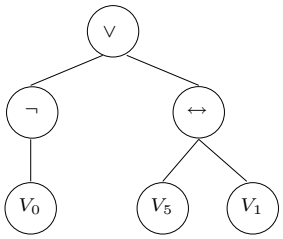
Notation 3.12.

- 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 $X', Y_1, \ldots$
- Wir 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 "$\vee$" an Stelle von "$\wedge$").
- Die äußeren Klammern einer Formel lassen wir manchmal weg und schreiben z.B. $(A \wedge B) \to C$ an Stelle des (formal korrekten) $\big((A \wedge B) \to C\big)$.
- Ist φ eine Formel und $\mathcal{B}$ eine Belegung für φ, so sagen wir "$\mathcal{B}$ **erfüllt** φ" (bzw. "$\mathcal{B}$ ist eine **erfüllende Belegung** für φ"), falls $[\![\varphi]\!]^{\mathcal{B}} = 1$.

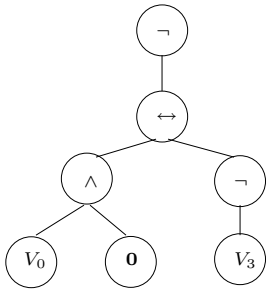
Bemerkung 3.13 (Syntaxbäume zur graphischen Darstellung von Formeln)**.** Die Struktur einer Formel lässt sich bequem durch einen **Syntaxbaum** (englisch: **parse tree**) darstellen.

Beispiele:

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



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



Computerlesbare Darstellung von Formeln:

Definition 3.14 (ASCII-Syntax für die Aussagenlogik)**.**

(a) Wir betrachten das folgende Alphabet:

$$\text{ASCII} \;:=\; \text{Menge aller ASCII-Symbole.}$$

(b) Die Menge $\text{AVAR}_{\text{ASCII}}$ aller ASCII-Repräsentationen von Aussagenvariablen ist wie folgt definiert:

$$\text{AVAR}_{\text{ASCII}} := \big\{ w \in \text{ASCII}^+ : \text{das erste Symbol in } w \text{ ist ein Buchstabe, alle weiteren Symbole in } w \text{ sind Buchstaben oder Ziffern} \big\}.$$

(c) Die Menge AL_{ASCII} aller ASCII-Repräsentationen von aussagenlogischen Formeln ist die rekursiv wie folgt definierte Teilmenge von ASCII^*:

Basisregeln:

- `0` $\in \text{AL}_{\text{ASCII}}$
- `1` $\in \text{AL}_{\text{ASCII}}$
- Für alle $w \in \text{AVAR}_{\text{ASCII}}$ gilt: $w \in \text{AL}_{\text{ASCII}}$.

Rekursive Regeln:

- Ist $\varphi \in \text{AL}_{\text{ASCII}}$, so ist auch `~`$\varphi \in \text{AL}_{\text{ASCII}}$.
- Ist $\varphi \in \text{AL}_{\text{ASCII}}$ und $\psi \in \text{AL}_{\text{ASCII}}$, so ist auch
  - $(\varphi$ `/\` $\psi) \in \text{AL}_{\text{ASCII}}$
  - $(\varphi$ `\/` $\psi) \in \text{AL}_{\text{ASCII}}$
  - $(\varphi$ `->` $\psi) \in \text{AL}_{\text{ASCII}}$
  - $(\varphi$ `<->` $\psi) \in \text{AL}_{\text{ASCII}}$.

Bemerkung 3.15. Es ist offensichtlich, wie man Formeln aus AL in ihre entsprechende ASCII-Repräsentation übersetzt und umgekehrt.

Beispiel:

Formel in AL:	$\big((V_0 \wedge \mathbf{0}) \rightarrow \neg V_{13}\big)$
entsprechende Formel in AL_{ASCII}:	`( (V0 /\ 0) -> ~V13 )`

In der Vorlesung werden wir nur mit der "abstrakten Syntax", d.h. mit AL, arbeiten. Um aber Formeln in Computer-Programmen eingeben zu können (siehe Webseite der Vorlesung) werden wir die ASCII-Repräsentation verwenden.

Umgangssprachliche Aussagen lassen sich wie folgt durch aussagenlogische Formeln repräsentieren:

Beispiel 3.16.
"Das Fluchtauto war rot oder grün und hatte weder vorne noch hinten ein Nummernschild."
Atomare Aussagen:

- 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.

Obige Aussage wird dann durch folgende aussagenlogische Formel repräsentiert:

$$\big((X_R \vee X_G) \wedge (\neg X_V \wedge \neg X_H)\big).$$

Beispiel 3.17.
Das in Beispiel 3.1 ("Geburtstagsfeier") aufgelistete Wissen kann folgendemaßen 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 in Beispiel 3.1 wird durch folgende Formel repräsentiert:

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

Die Frage "Wie viele (und welche) Freunde werden im besten Fall zur Party kommen?" kann dann durch Lösen der folgenden Aufgabe beantwortet werden: Finde eine Belegung $\mathcal{B}$ für φ, so dass

(1) φ von $\mathcal{B}$ erfüllt wird, d.h. $[\![\varphi]\!]^{\mathcal{B}} = 1$ und

(2) $|\{X \in \{A, B, C, D, E\} : \mathcal{B}(X) = 1\}|$ so groß wie möglich ist.

Um Aufgaben solcher Art lösen zu können, brauchen wir also eine Methode zum Finden der erfüllenden Belegungen für eine Formel. Eine Möglichkeit dafür ist, so genannte Wahrheitstafeln zu benutzen.

Wahrheitstafeln:

Für jede Formel φ kann man die Werte unter allen möglichen Belegungen in einer Wahrheitstafel darstellen. 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 3.18. (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 also 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 3.17 beschriebene Aufgabe zur "Geburtstagsfeier" zu lösen.

Beispiel 3.19. Sei φ die Formel aus Beispiel 3.17. Die Frage "Wie viel (und welche) Freunde werden bestenfalls zur Party kommen?" können wir lösen, in dem wir

(1) die Wahrheitstafel für φ ermitteln,

(2) alle Zeilen raussuchen, 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 Tabelle 3.1) sieht. **Erfüllende Belegungen** für φ werden in Tabelle 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 Belegung gibt, bei denen in den mit A bis E beschrifteten Spalten insgesamt 4 Einsen stehen, nämlich die beiden 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 Antworten auf die Frage "Wie viel (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 3.19

Angesichts der Wahrheitstafel aus Beispiel 3.19 stellt sich die Frage, wie groß die Wahrheitstafel für eine gegebene Formel φ ist. Die Antwort darauf gibt der folgende Satz.

Satz 3.20. *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

$$\begin{aligned} &\{\mathcal{B} : \mathcal{B} \text{ ist eine zu } \varphi \text{ passende Belegung mit } \mathrm{Def}(\mathcal{B}) = \mathrm{Var}(\varphi)\} \\ \overset{\text{Def. 3.7}}{=}\ &\{\mathcal{B} : \mathcal{B}\colon\ \mathrm{Var}(\varphi) \to \{0,1\} \text{ ist eine Funktion}\} \\ \overset{\text{Not. 2.31}}{=}\ &\mathrm{Abb}(\mathrm{Var}(\varphi), \{0,1\}). \end{aligned}$$

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

Tabelle 3.1: Wahrheitstafel für die Formel φ aus Beispiel 3.17

Wir wissen außerdem, dass

$$|\mathrm{Abb}(\mathrm{Var}(\varphi), \{0,1\})| \overset{\text{Fol. 2.39(a)}}{=} |\{0,1\}|^{|\mathrm{Var}(\varphi)|} \overset{n=|\mathrm{Var}(\varphi)|}{=} 2^n.$$

□

Satz 3.20 besagt, dass die Wahrheitstafel einer Formel mit n Variablen genau 2^n Zeilen hat. 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	$2^{60} =$	$1.152.921.504.606.846.976 \approx 10^{18}$

Zum Vergleich: Das Alter des Universums wird auf 10^{10} Jahre $< 10^{18}$ Sekunden geschätzt.

Ein ganzer Zweig der Informatik (z.B. unter dem Stichwort "SAT-Solving") und viele internationale Forschungsgruppen beschäftigen sich mit der Aufgabe, Verfahren zu entwickeln, die die erfüllenden Belegungen von aussagenlogischen Formeln ermitteln und dabei wesentlich effizienter sind als das vorgestellte Wahrheitstafel-Verfahren. Ein relativ ernüchterndes Resultat, das Sie in der "Algorithmentheorie"-Vorlesung kennenlernen werden, ist der folgende Satz:

Satz. *Das aussagenlogische Erfüllbarkeitsproblem ist NP-vollständig.*

Das **aussagenlogische Erfüllbarkeitsproblem** (Kurz: **SAT**, für englisch: "satisfiability") ist dabei das folgende Berechnungsproblem:

aussagenlogisches Erfüllbarkeitsproblem (SAT)

AUSSAGENLOGISCHES ERFÜLLBARKEITSPROBLEM (SAT)
Eingabe: eine aussagenlogische Formel φ
Frage: Gibt es eine erfüllende Belegung für φ?

Was der Begriff "NP-vollständig" **genau** bedeutet, werden Sie in der "Algorithmentheorie"-Vorlesung lernen; grob gesagt bedeutet "NP-vollständig", dass es "Wahrscheinlich keinen **effizienten** Algorithmus gibt, der das aussagenlogische Erfüllbarkeitsproblem löst." Andererseits wurden (besonders in den letzten Jahren) Heuristiken und randomisierte Algorithmen entwickelt, die das aussagenlogische Erfüllbarkeitsproblem trotzdem in vielen Fällen erstaunlich effizient lösen können.

Die folgenden Begriffe werden Ihnen in späteren Vorlesungen immer wieder begegnen:

Definition 3.21. Sei φ eine aussagenlogische Formel.

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

(b) φ heißt **unerfüllbar**, wenn es **keine** erfüllende Belegung für φ gibt. unerfüllbar

(c) φ heißt **allgemeingültig** (bzw. **Tautologie**), wenn **jede** zu φ passende Belegung φ erfüllt, d.h. wenn für jede zu φ passende Belegung $\mathcal{B}$ gilt: $[\![\varphi]\!]^{\mathcal{B}} = 1$. allgemeingültig, Tautologie

Beispiel 3.22.

(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.
- **nicht allgemeingültig**, da z.B. die Belegung $\mathcal{B}'$ mit $\mathcal{B}'(X) = 0$ und $\mathcal{B}'(Y) = 0$ die Formel nicht erfüllt.

(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.

Fall 1: $\mathcal{B}(X) = 1$*:*

$$\begin{aligned}[\![(X \wedge \neg X)]\!]^{\mathcal{B}} &= \begin{cases} 1, & \text{falls } \mathcal{B}(X) = 1 \text{ und } \mathcal{B}(X) = 0 \\ 0, & \text{sonst} \end{cases} \\ &= 0, \text{ da } \mathcal{B}(X) = 1 \neq 0.\end{aligned}$$

Fall 2: $\mathcal{B}(X) = 0$*:*

$$\begin{aligned}[\![(X \wedge \neg X)]\!]^{\mathcal{B}} &= \begin{cases} 1, & \text{falls } \mathcal{B}(X) = 1 \text{ und } \mathcal{B}(X) = 0 \\ 0, & \text{sonst} \end{cases} \\ &= 0, \text{ da } \mathcal{B}(X) = 0 \neq 1.\end{aligned}$$

(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 $[\![(X \wedge \neg X)]\!]^{\mathcal{B}} = 1$, für alle zur Formel passenden Belegungen $\mathcal{B}$.

Beobachtung 3.23. Sei φ eine aussagenlogische Formel.

(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.

Folgerung 3.24. Für alle aussagenlogischen Formeln φ gilt:

$$\varphi \text{ ist allgemeingültig} \iff \neg\varphi \text{ ist unerfüllbar.}$$

3.3 Folgerung und Äquivalenz

folgt aus

Definition 3.25 (semantische Folgerung)**.** Seien φ und ψ zwei aussagenlogische Formeln. Wir sagen ψ **folgt aus** φ (kurz: $\varphi \models \psi$, "φ impliziert ψ"), falls für jede zu φ und ψ passende Belegung $\mathcal{B}$ gilt:

$$\text{Falls } [\![\varphi]\!]^{\mathcal{B}} = 1, \text{ so auch } [\![\psi]\!]^{\mathcal{B}} = 1.$$

D.h.: $\varphi \models \psi \iff$ jede Belegung, die zu φ und ψ passt und die φ erfüllt, erfüllt auch ψ.

Beispiel 3.26. 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 **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 steht in jeder Zeile (d.h. jede zu φ und ψ passende Belegung), in der in der mit "φ" beschrifteten Spalte eine 1 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 3.27. Seien φ und ψ aussagenlogische Formeln. Dann gilt:

(a) $\mathbf{1} \models \varphi \iff \varphi$ ist allgemeingültig.

(b) $\varphi \models \mathbf{0} \iff \varphi$ ist unerfüllbar.

(c) $\varphi \models \psi \iff (\varphi \to \psi)$ ist allgemeingültig.

(d) $\varphi \models \psi \iff (\varphi \wedge \neg\psi)$ ist unerfüllbar.

Beweis: Übung. □

Definition 3.28 (logische Äquivalenz)**.** Zwei aussagenlogische Formeln φ und ψ heißen **äquivalent** (kurz: $\varphi \equiv \psi$), wenn für alle zu φ und ψ passenden Belegungen $\mathcal{B}$ gilt: $[\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}}$. äquivalent

Beispiel 3.29. 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 $[\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}}$. D.h.: $\varphi \equiv \psi$.

Beobachtung 3.30. 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. □

Fundamentale Äquivalenzen der Aussagenlogik

Satz 3.31. *Seien φ, ψ und χ aussagenlogische Formeln. Dann gilt:*

(a) (Idempotenz)

- $(\varphi \wedge \varphi) \equiv \varphi$
- $(\varphi \vee \varphi) \equiv \varphi$

(b) (Kommutativität)

- $(\varphi \wedge \psi) \equiv (\psi \wedge \varphi)$
- $(\varphi \vee \psi) \equiv (\psi \vee \varphi)$

(c) (Assoziativität)

- $((\varphi \wedge \psi) \wedge \chi) \equiv (\varphi \wedge (\psi \wedge \chi))$
- $((\varphi \vee \psi) \vee \chi) \equiv (\varphi \vee (\psi \vee \chi))$

(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)

- $\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$
- $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$

(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)

- $(\varphi \leftrightarrow \psi) \equiv ((\varphi \to \psi) \wedge (\psi \to \varphi))$

Beweis: Durch einfaches Nachrechnen. Details: Übung. □

Bemerkung 3.32. Durch schrittweises Anwenden der in Satz 3.31 aufgelisteten Äquivalenzen kann man eine gegebene aussagenlogische Formel in eine zu ihr äquivalente Formel umformen.

Beispiel. Sind φ und ψ aussagenlogische Formeln, so gilt:

$$\begin{aligned}(\varphi \leftrightarrow \psi) &\equiv \big((\varphi \to \psi) \wedge (\psi \to \varphi)\big) && \text{(Satz 3.31(l))}\\ &\equiv \big((\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)\big) && \text{(Satz 3.31(k))}\end{aligned}$$

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 Formel φ finden, zu der die Wahrheitstafel passt?

Beispiel 3.33. 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

Eine Formel φ, so dass T die Wahrheitstafel für φ ist, kann man folgendermaßen erzeugen:

- Betrachte alle Zeilen von T, in denen in der mit "φ" beschrifteten Spalte eine 1 steht.
- Für jede solche Zeile konstruiere eine Formel, die genau von der zur Zeile gehörenden Belegung erfüllt wird.
- Bilde die Disjunktion (d.h. Ver-ODER-ung) über all diese Formeln. Dies liefert die gesuchte Formel φ.

In unserer Beispiel-Wahrheitstafel T gibt es genau 3 Zeilen, in denen in der mit φ beschrifteten Spalte eine 1 steht, nämlich die Zeilen

X	Y	Z	φ	zur Belegung der jeweiligen Zeile gehörende Formel:
0	0	0	1	$(\neg X \wedge \neg Y \wedge \neg Z)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	
1	0	0	1	$(\ X \wedge \neg Y \wedge \neg Z)$
1	0	1	1	$(\ X \wedge \neg Y \wedge \ Z)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

$\Longrightarrow$ zur Wahrheitstafel T passende Formel:

$$\varphi \ := \ (\neg X \wedge \neg Y \wedge \neg Z) \ \vee \ (X \wedge \neg Y \wedge \neg Z) \ \vee \ (X \wedge \neg Y \wedge Z).$$

Generell kann man auf die beschriebene Art zu jeder beliebigen Wahrheitstafel eine aussagenlogische Formel konstruieren, die zur Wahrheitstafel passt. Die so konstruierten Formeln haben eine besonders einfache Form. Sie sind 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 3.34 (disjunktive Normalform, konjunktive Normalform)**.**

(a) Ein **Literal** ist eine Formel der Form X oder $\neg X$, wobei $X \in \text{AVAR}$ (d.h. X ist eine Aussagenvariable). Ein Literal der Form X, mit $X \in \text{AVAR}$, wird auch **positives Literal** genannt; eine Formel der Form $\neg X$, mit $x \in \text{AVAR}$, **negatives Literal**.

Literal
positives Literal
negatives Literal

(b) Eine aussagenlogische Formel ist in **disjunktiver Normalform** (**DNF**), wenn sie eine Disjunktion von Konjunktionen von Literalen ist, d.h. wenn sie die Gestalt

$$\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right)$$

hat, wobei $n, m_1, \ldots, m_n \in \mathbb{N}_{>0}$ und $l_{i,j}$ ein Literal ist (für jedes $i \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m_i\}$).

Die Teilformeln $\kappa_i := \bigwedge_{j=1}^{m_i} l_{i,j}$ (für $i \in \{1, \ldots, n\}$) heißen **konjunktive Klauseln**.

disjunktive Normalform
DNF

konjunktive Klausel

(c) Eine aussagenlogische Formel ist in **konjunktiver Normalform** (**KNF**), wenn sie eine Konjunktion von Disjunktionen von Literalen ist, d.h. wenn sie die Gestalt

$$\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} l_{i,j} \right)$$

konjunktive Normalform
KNF

hat, wobei $n, m_1, \ldots, m_n \in \mathbb{N}_{>0}$ und $l_{i,j}$ ein Literal ist (für jedes $i \in \{1, \ldots, n\}$ und $j \in \{1, \ldots, m_i\}$).

disjunktive Klausel

Die Teilformeln $\kappa_i := \bigvee_{j=1}^{m_i} l_{i,j}$ (für $i \in \{1, \ldots, n\}$) heißen **disjunktive Klauseln**.

Normalformen spielen in vielen Anwendungsgebieten eine wichtige Rolle. Beispielsweise geht man in der Schaltungstechnik (Hardware-Entwurf) oft von DNF-Formeln aus, während bei der aussagenlogischen Modellbildung oftmals KNF-Formeln auftreten, da sich eine Sammlung von einfach strukturierten Aussagen sehr gut durch eine Konjunktion von Klauseln ausdrücken lässt.

Satz 3.35. *Für jede aussagenlogische Formel φ gibt es eine Formel ψ_D in DNF und eine Formel ψ_K in KNF, so dass $\varphi \equiv \psi_D \equiv \psi_K$.*
(D.h.: Jede Formel ist äquivalent zu einer Formel in DNF und zu einer Formel in KNF.)

Beweisidee:

- Zur Konstruktion einer zu φ äquivalenten Formel ψ_D in DNF stellen wir zunächst die Wahrheitstafel für φ auf. Falls diese in der mit "φ" beschrifteten Spalte nur Nullen hat (d.h. φ ist unerfüllbar), so setzen wir $\psi_D := (V_0 \wedge \neg V_0)$ – offensichtlich ist ψ_D in DNF und unerfüllbar, also äquivalent zu φ.

  Falls die mit "φ" beschriftete Spalte der Wahrheitstafel mindestens eine 1 enthält, so gehen wir wie in Beispiel 3.33 vor, um eine zu φ äquivalente Formel ψ_D in DNF zu konstruieren.

- Zur Konstruktion einer zu φ äquivalenten Formel ψ_K in KNF können wir folgendermaßen vorgehen:

  (1) Sei $\varphi' := \neg\varphi$.

  (2) Konstruiere eine zu φ' äquivalente Formel ψ'_D in DNF. Sei $\bigvee_{i=1}^{n}\left(\bigwedge_{j=1}^{m_i} l_{i,j}\right)$ die Gestalt von ψ'_D.

  (3) Für alle i, j sei $\tilde{l}_{i,j} := \begin{cases} \neg X, & \text{falls } l_{i,j} = X \text{ für ein } X \in \text{AVAR} \\ X, & \text{falls } l_{i,j} = \neg X \text{ für ein } X \in \text{AVAR}. \end{cases}$

  (4) Setze $\psi_K := \bigwedge_{i=1}^{n}\left(\bigvee_{j=1}^{m_i} \tilde{l}_{i,j}\right)$.

  Offensichtlich ist ψ_K eine Formel in KNF. Außerdem gilt:

$$
\begin{aligned}
\varphi &\equiv \neg\varphi' \\
&\equiv \neg\psi'_D \\
&\equiv \neg\left(\bigvee_{i=1}^{n}\left(\bigwedge_{j=1}^{m_i} l_{i,j}\right)\right) \\
&\overset{\text{Satz 3.31(g)}}{\equiv} \left(\bigwedge_{i=1}^{n}\neg\left(\bigwedge_{j=1}^{m_i} l_{i,j}\right)\right) \\
&\overset{\text{Satz 3.31(g)}}{\equiv} \bigwedge_{i=1}^{n}\left(\bigvee_{j=1}^{m_i}\neg l_{i,j}\right) \\
&\overset{\text{Def. } \tilde{l}_{i,j}}{\equiv} \bigwedge_{i=1}^{n}\left(\bigvee_{j=1}^{m_i}\tilde{l}_{i,j}\right) \\
&\overset{\text{Def.}}{=} \psi_K.
\end{aligned}
$$

□

Abgesehen von DNF und KNF gibt es noch eine weitere wichtige Normalform, die so genannte Negationsnormalform.

Definition 3.36. Eine aussagenlogische Formel ist in **Negationsnormalform** (**NNF**), wenn sie keines der Symbole $\rightarrow$, $\leftrightarrow$, $\mathbf{0}$, $\mathbf{1}$ enthält und Negationszeichen nur unmittelbar vor Variablen auftreten.

Negationsnormalform
NNF

Rekursiv lässt sich die Menge der Formeln in NNF folgendermaßen definieren.

Basisregeln: Für jedes $X \in \text{AVAR}$ ist sowohl X als auch $\neg X$ eine Formel in NNF.

Rekursive Regeln: Sind φ und ψ Formeln in NNF, so sind auch $(\varphi \wedge \psi)$ und $(\varphi \vee \psi)$ Formeln in NNF.

Beobachtung 3.37. Jede Formel, die in KNF oder in DNF ist, ist auch in NNF. Aus Satz 3.35 folgt also insbesondere, dass jede aussagenlogische Formel äquivalent zu einer Formel in NNF ist.

Beachte: Nicht jede Formel in NNF ist auch in KNF oder in DNF.

Beispiel: $\Big(\big((X \wedge \neg Y) \vee (\neg X \wedge Y)\big) \wedge \neg Z\Big)$ ist in NNF, aber weder in KNF noch in DNF.

Ein einfaches Verfahren zur Transformation einer gegebenen aussagenlogischen Formel in eine äquivalente Formel in NNF beruht auf der wiederholten Anwendung der De Morganschen Regeln (Satz 3.31(g)) und der Regel für "doppelte Negation" (Satz 3.31(f)): Mit den De Morganschen Regeln ($\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)$ bzw. $\neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)$) ziehen wir das Negationszeichen nach innen; mit der "doppelte Negation"-Regel ($\neg\neg\varphi \equiv \varphi$) können wir Schritt für Schritt mehrfach hintereinander vorkommende Negationszeichen eliminieren. Eventuell in der Formel vorkommende Implikationspfeile "$\rightarrow$" oder Biimplikationspfeile "$\leftrightarrow$"eliminieren wir durch Verwenden von Satz 3.31(k) und (l). Eventuelle Vorkommen der Symbole $\mathbf{0}$ bzw. $\mathbf{1}$ ersetzen wir durch die Formel $(V_0 \wedge \neg V_0)$ bzw. $(V_0 \vee \neg V_0)$.

Beispiel 3.38.

Ziel: Bringe die Formel $\Big(\big(\neg V_0 \wedge \neg\big((V_0 \vee V_1) \rightarrow V_0\big)\big) \rightarrow \mathbf{0}\Big)$ in NNF, d.h. finde eine zur gegebenen Formel äquivalente Formel in NNF.

Lösung:

$$
\begin{aligned}
\Big(\big(\neg V_0 \wedge \neg\big((V_0 \vee V_1) \rightarrow V_0\big)\big) \rightarrow \underline{\mathbf{0}}\Big) &\equiv \Big(\big(\neg V_0 \wedge \neg\big((V_0 \vee V_1) \rightarrow V_0\big)\big) \underline{\rightarrow} (V_0 \wedge \neg V_0)\Big) \\
&\equiv \Big(\neg\big(\neg V_0 \wedge \neg\big((V_0 \vee V_1) \underline{\rightarrow} V_0\big)\big) \vee (V_0 \wedge \neg V_0)\Big) \\
&\equiv \Big(\underline{\neg}\big(\neg V_0 \wedge \neg\big(\neg(V_0 \vee V_1) \vee V_0\big)\big) \vee (V_0 \wedge \neg V_0)\Big) \\
&\equiv \Big(\big(\underline{\neg\neg} V_0 \vee \underline{\neg\neg}\big(\neg(V_0 \vee V_1) \vee V_0\big)\big) \vee (V_0 \wedge \neg V_0)\Big) \\
&\equiv \Big(\big(V_0 \vee \big(\underline{\neg}(V_0 \vee V_1) \vee V_0\big)\big) \vee (V_0 \wedge \neg V_0)\Big) \\
&\equiv \Big(\big(V_0 \vee \big((\neg V_0 \wedge \neg V_1) \vee V_0\big)\big) \vee (V_0 \wedge \neg V_0)\Big) \quad \text{in NNF.}
\end{aligned}
$$

Unter zusätzlicher Verwendung der "Distributivitätsregel" (Satz 3.31(e)) erhält man Verfahren zur Transformation einer gegebenen Formel in eine äquivalente Formel in DNF bzw. KNF, bei

denen man nicht zuerst eine Wahrheitstafel aufstellen muss. Diese Verfahren sind vor allem dann ratsam, wenn die gegebene Formel sehr viele verschiedene Variablen enthält, die zugehörige Wahrheitstafel also sehr groß wird.

Algorithmus 3.39 (Ein KNF-Algorithmus)**.**

Eingabe: Eine aussagenlogische Formel φ.

Ausgabe: Eine zu φ äquivalente Formel φ' in KNF.

Verfahren:

(1) Konstruiere eine zu φ äquivalente Formel φ' in NNF (beispielsweise mit dem in Beobachtung 3.37 beschriebenen Verfahren).

(2) Wiederhole folgende Schritte:
   (i) Falls φ' in KNF ist, so halte mit Ausgabe φ'.
   (ii) Ersetze eine Teilformel von φ' der Gestalt $\big(\psi_1 \vee (\psi_2 \wedge \psi_3)\big)$ durch $\big((\psi_1 \vee \psi_2) \wedge (\psi_1 \vee \psi_3)\big)$ oder ersetze eine Teilformel von φ' der Gestalt $\big((\psi_2 \wedge \psi_3) \vee \psi_1\big)$ durch $\big((\psi_2 \vee \psi_1) \wedge (\psi_3 \vee \psi_1)\big)$. Sei φ'' die resultierende Formel.
   (iii) Setze $\varphi' := \varphi''$.

Algorithmus 3.40 (Ein DNF-Algorithmus)**.**

Eingabe: Eine aussagenlogische Formel φ

Ausgabe: Eine zu φ äquivalente Formel φ' in DNF.

Verfahren:

(1) Konstruiere eine zu φ äquivalente Formel φ' in NNF.

(2) Wiederhole folgende Schritte:
   (i) Falls φ' in DNF ist, so halte mit Ausgabe φ'.
   (ii) Ersetze eine Teilformel von φ' der Gestalt $\big(\psi_1 \wedge (\psi_2 \vee \psi_3)\big)$ durch $\big((\psi_1 \wedge \psi_2) \vee (\psi_1 \wedge \psi_3)\big)$ oder ersetze eine Teilformel von φ' der Gestalt $\big((\psi_2 \vee \psi_3) \wedge \psi_1\big)$ durch $\big((\psi_2 \wedge \psi_1) \vee (\psi_3 \wedge \psi_1)\big)$. Sei φ'' die resultierende Formel.
   (iii) Setze $\varphi' := \varphi''$.

Satz 3.41 (Korrektheit der Algorithmen 3.39 und 3.40)**.** *Für jede aussagenlogische Formel φ gilt:*

(a) Algorithmus 3.39 hält bei Eingabe der Formel φ nach endlich vielen Schritten an und gibt eine zu φ äquivalente Formel in KNF aus.

(b) Algorithmus 3.40 hält bei Eingabe der Formel φ nach endlich vielen Schritten an und gibt eine zu φ äquivalente Formel in DNF aus.

(hier ohne Beweis)

Beispiel 3.42. Sei $\varphi := \Big(\big(\neg V_0 \wedge (V_0 \to V_1)\big) \vee (V_2 \to V_3)\Big)$.

Transformation von φ in NNF:

$$\varphi \;=\; \Big(\big(\neg V_0 \wedge (V_0 \mathrel{\underline{\to}} V_1)\big) \vee (V_2 \mathrel{\underline{\to}} V_3)\Big) \;\equiv\; \Big(\big(\neg V_0 \wedge (\neg V_0 \vee V_1)\big) \vee (\neg V_2 \vee V_3)\Big) \;=:\; \varphi'.$$

Transformation von φ in DNF mittels Algorithmus 3.40:

(1) Transformation von φ in NNF $\rightsquigarrow$ liefert $\varphi' = \Big(\underline{(\neg V_0 \wedge (\neg V_0 \vee V_1))} \vee (\neg V_2 \vee V_3)\Big)$

(2) 1-maliges Anwenden von Zeile (ii) des Algorithmus auf φ' liefert:

$$\varphi'' \;:=\; \Big(\big(\underline{(\neg V_0 \wedge \neg V_0)} \vee \underline{(\neg V_0 \wedge V_1)}\big) \vee \underline{(\neg V_2 \vee V_3)}\Big).$$

Diese Formel ist die DNF-Formel, die von dem Algorithmus ausgegeben wird.

Transformation von φ in KNF mittels Algorithmus 3.39:

(1) Transformation von φ in NNF $\rightsquigarrow$ liefert $\varphi' = \Big((\neg V_0 \wedge (\neg V_0 \vee V_1)) \veebar (\neg V_2 \vee V_3)\Big)$

(2) 1-maliges Anwenden von Zeile (ii) des Algorithmus auf φ' liefert:

$$\varphi'' \;:=\; \Big(\underline{(\neg V_0 \vee (\neg V_2 \vee V_3))} \wedge \underline{((\neg V_0 \vee V_1) \vee (\neg V_2 \vee V_3))}\Big).$$

Dies ist die KNF-Formel, die von dem Algorithmus ausgegeben wird.

Unmittelbar vor Definition 3.21 wurde darauf hingewiesen, dass die Aufgabe, für eine gegebene Formel φ herauszufinden, ob sie erfüllbar ist, im Allgemeinen ein recht schwieriges Problem ist. Für den Spezialfall, dass φ eine Formel in DNF ist, lässt sich das Erfüllbarkeitsproblem allerdings sehr effizient lösen, wie die folgende Beobachtung zeigt.

Beobachtung 3.43 (effizienter Erfüllbarkeitstest für DNF-Formeln)**.** Sei φ eine Formel in DNF, d.h. φ ist von der Form

$$\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right), \quad \text{für Literale } l_{i,j}.$$

D.h. φ ist von der Form

$$\kappa_1 \vee \cdots \vee \kappa_n,$$

wobei für jedes $i \in \{1, \ldots, n\}$ κ_i die konjunktive Klausel

$$\kappa_i \;:=\; l_{i,1} \wedge \cdots \wedge l_{i,m_i}$$

ist.

Klar: φ ist erfüllbar $\iff$ für mindestens ein $i \in \{1, \ldots, n\}$ ist κ_i erfüllbar. Da κ_i eine Konjunktion von Literalen (d.h. von Variablen und/oder negierten Variablen) ist, gilt:

$$\kappa_i \text{ ist erfüllbar} \iff \text{es gibt keine } j, j' \in \{1, \ldots, m_i\}, \text{ so dass } l_{i,j} = \neg l_{i,j'}.$$

Daher ist der folgende Algorithmus dazu geeignet, zu testen, ob eine gegebene DNF-Formel erfüllbar ist.

Eingabe: Eine aussagenlogische Formel $\varphi = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} l_{i,j} \right)$ in DNF

Ziel: Entscheide, ob φ erfüllbar ist.

Verfahren:

(1) Für $i = 1, \ldots, n$

(2) Für $j = 1, \ldots, m_i$

(3) Für $j' = j + 1, \ldots, m_i$

(4) Falls $l_{i,j} = \neg l_{i,j'}$ oder $l_{i,j'} = \neg l_{i,j}$, dann:

(5) Falls $i = n$ ist, so mache in Zeile 7 weiter;
ansonsten setze $i := i + 1$ und mache in Zeile 2 weiter.

(6) Halte mit Ausgabe "φ ist erfüllbar".

(7) Halte mit Ausgabe "φ ist unerfüllbar".

Um aussagenlogische Formeln φ von **beliebiger** Form auf Erfüllbarkeit zu testen, kann man dann folgendermaßen vorgehen:

Schritt 1: Transformiere φ in eine äquivalente Formel φ' in DNF (z.B. mit Algorithmus 3.40).

Schritt 2: Entscheide, ob φ' erfüllbar ist (z.B. mit dem obigen Verfahren).

Das Ausführen von Schritt 1 kann dabei u.U. aber leider wieder sehr lange dauern, da es einige Formeln gibt, zu denen äquivalente Formeln in DNF zwangsläufig sehr groß sind. Dies wird durch den folgenden Satz präzisiert:

Satz 3.44. *Sei $n \in \mathbb{N}_{>0}$, seien $X_1, \ldots, X_n, Y_1, \ldots, Y_n$ genau $2 \cdot n$ verschiedene aussagenlogische Variablen, und sei*

$$\varphi_n \;:=\; \bigwedge_{i=1}^{n} (X_i \leftrightarrow Y_i).$$

Dann hat jede zu φ_n äquivalente Formel in DNF mindestens 2^n konjunktive Klauseln.

Beweis: Übung. □

3.5 Übungsaufgaben zu Kapitel 3

Aufgabe 3.1.

(a) Welche der folgenden Wörter gehören gemäß Definition 3.3 zur Sprache AL, welche nicht?

- $(V_1 \wedge \mathbf{1})$
- $(V_1 \wedge \mathbf{101})$
- $(\neg(V_1 \wedge V_2) \vee V_3)$
- $\neg(V_1 \wedge V_2) \vee V_3$
- $(V_1 \rightarrow V_2)$
- $(V_1 \rightarrow V_2 \rightarrow V_3)$
- $(V_1 \leftarrow V_2)$
- $(V_1 \leftrightarrow V_2)$

(b) Beweisen Sie, dass für die Formel $\varphi := \big((V_1 \leftrightarrow \mathbf{1}) \wedge (V_1 \rightarrow (V_2 \wedge \mathbf{0}))\big)$ gilt: $\varphi \in \mathrm{AL}$.

(c) Betrachten Sie die Formel φ aus (b) und die Belegung $\mathcal{B}\colon \mathrm{Var}(\varphi) \to \{0,1\}$ mit $\mathcal{B}(V_1) = 1$ und $\mathcal{B}(V_2) = 0$. Berechnen Sie den Wert $[\![\varphi]\!]^{\mathcal{B}}$.

(d) Geben Sie den Syntaxbaum der Formel φ aus (b) an.

Aufgabe 3.2.

(a) Betrachten Sie die folgenden Wörter 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) $\neg\Big((V_3 \wedge \neg\mathbf{0}) \rightarrow \big(V_0 \vee (\neg\neg V_1 \wedge V_4)\big)\Big)$

(ii) $(V_5 \leftrightarrow X) \wedge \big(V_{23} \rightarrow (V_1 \wedge \mathbf{0})\big)$

(iii) $\Big((V_{11} \leftarrow V_7) \vee \neg\neg V_5\Big)$

(iv) $\big((V_9 \vee \neg(\neg V_{42}) \vee \neg V_2) \rightarrow \mathbf{1}\big)$

(b) Betrachten Sie die aussagenlogische Formel

$$\varphi \;:=\; \Big((\neg V_0 \wedge V_1) \rightarrow \big(V_0 \wedge (V_1 \vee \neg V_2)\big)\Big)$$

und die Belegung $\mathcal{B}\colon \mathrm{Var}(\varphi) \to \{0,1\}$ mit $\mathcal{B}(V_0) = 1$ und $\mathcal{B}(V_1) = \mathcal{B}(V_2) = 0$. Berechnen Sie den Wert $[\![\varphi]\!]^{\mathcal{B}}$ in nachvollziehbaren Schritten analog zu Beispiel 3.9.

(c) Geben Sie den Syntaxbaum und die ASCII-Darstellung der Formel φ aus (b) an.

Aufgabe 3.3. 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 armen 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, 3.17 und 3.19).

(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 3.4. 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, 3.17 und 3.19).

(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 3.5. 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.

(b) Zu welchen Stämmen gehören A, B und C?

Aufgabe 3.6. Zwei Analysten streiten sich, wer von ihnen denn nun am besten Aktienkurse voraussagen kann. Dazu wollen sie drei zufällig anwesende Anleger A, B und C befragen. Das wäre nicht weiter schwierig, wenn sich A, B und C nicht folgendes (repräsentiert durch aussagenlogische Formeln) vorwerfen würden:

- A behauptet: $\varphi_A := (\neg B \vee \neg C)$
- B behauptet: $\varphi_B := \neg A$
- C behauptet: $\varphi_C := (A \wedge \neg B)$

Hierbei bedeuten die Aussagenvariablen:

- A: A sagt die Wahrheit.
- B: B sagt die Wahrheit.
- C: C sagt die Wahrheit.

(a) Beschreiben Sie umgangssprachlich, was jede der Formeln φ_A, φ_B, φ_C aussagt.

(b) Wem können die Analysten glauben und wem nicht? Falls es mehrere Möglichkeiten gibt, geben Sie alle an.

Aufgabe 3.7. Geben Sie für jede der folgenden aussagenlogischen Formeln eine Wahrheitstafel und alle erfüllenden Belegungen $\mathcal{B}\colon \mathrm{Var}(\varphi_1) \to \{0,1\}$ (für (a)) bzw. $\mathcal{B}\colon \mathrm{Var}(\varphi_2) \to \{0,1\}$ (für (b)) an.

(a) $\varphi_1 := \Big(\big((V_1 \leftrightarrow \neg V_2) \wedge (\neg V_2 \vee V_3)\big) \wedge (V_3 \to V_1)\Big)$

(b) $\varphi_2 := \Big(\big(V_1 \leftrightarrow \mathbf{1}\big) \wedge \big(V_1 \to (V_2 \wedge \mathbf{0})\big)\Big)$

Aufgabe 3.8.

(a) Geben Sie für jede der folgenden aussagenlogischen Formeln an, ob sie erfüllbar, unerfüllbar und/oder allgemeingültig ist.

- $(V_0 \wedge \neg V_1)$
- $\big(V_0 \leftrightarrow (\mathbf{1} \to V_0)\big)$
- $\big(V_0 \leftrightarrow (V_0 \to \mathbf{0})\big)$
- $\Big(V_1 \vee \big((V_0 \wedge V_1) \to V_2\big)\Big)$
- $\big((V_0 \to V_1) \leftrightarrow (\neg V_1 \to \neg V_0)\big)$

(b) 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.

(c) Beweisen oder widerlegen Sie die folgende Behauptung:

$$\big((\neg V_0 \vee V_2) \wedge (V_1 \to \neg V_2)\big) \;\models\; \neg\big((V_0 \wedge \neg V_1) \to \neg(V_0 \to V_2)\big)$$

Aufgabe 3.9.

(a) Geben Sie für jede der folgenden aussagenlogischen Formeln an, ob sie erfüllbar, unerfüllbar und/oder allgemeingültig ist.

- $\neg V_1$
- $\big((V_0 \vee \neg V_1) \leftrightarrow V_2\big)$
- $\big(\neg V_0 \to (V_0 \to V_1)\big)$
- $\big(V_0 \wedge (V_0 \to \neg V_0)\big)$
- $\Big(\big(V_0 \to V_1\big) \leftrightarrow \big((V_0 \wedge \neg V_1) \to \mathbf{0}\big)\Big)$

(b) Für jedes $n \in \mathbb{N}$ sei die aussagenlogische Formel φ_n definiert durch

$$\varphi_n \;:=\; \begin{cases} (V_n \leftrightarrow V_{n+2}), & \text{falls } n \text{ gerade} \\ (V_n \leftrightarrow \neg V_{n-1}), & \text{falls } n \text{ ungerade.} \end{cases}$$

Es gilt also

$$\varphi_0 = (V_0 \leftrightarrow V_2), \quad \varphi_1 = (V_1 \leftrightarrow \neg V_0), \quad \varphi_2 = (V_2 \leftrightarrow V_4), \quad \varphi_3 = (V_3 \leftrightarrow \neg V_2), \quad \cdots$$

Geben Sie eine Belegung $\mathcal{B}\colon \mathrm{Var} \to \{0,1\}$ an, so dass für alle $n \in \mathbb{N}$ gilt: $\mathcal{B}$ erfüllt φ_n.

(c) Beweisen oder widerlegen Sie die folgenden Behauptungen:

(i) $\big(\neg(V_0 \leftrightarrow V_1) \wedge (\neg V_2 \vee V_0)\big) \models \big(V_0 \vee (V_1 \wedge \neg V_2)\big)$

(ii) $\big(\neg(V_0 \leftrightarrow V_1) \wedge (\neg V_2 \vee V_0)\big) \equiv \big(V_0 \vee (V_1 \wedge \neg V_2)\big)$

Aufgabe 3.10. Beweisen Sie Beobachtung 3.27 (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 3.11. Betrachten Sie die folgenden beiden Aussagen:

(1) Wenn der Rechner einen Virus hat oder nicht mehr funktioniert, und wenn der Administrator erreichbar ist, dann rufen wir den Administrator.

(2) Wenn der Rechner einen Virus 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 3.12 (Modellierung und Folgerung)**.** 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 aussagenlogische Formel an, die die Aussage repräsentiert.

(c) Entscheiden Sie für jede der aussagenlogischen Formeln aus (b), ob sie aus der Formel φ in (a) folgt.

Aufgabe 3.13. Es sei $\varphi := \big((V_0 \vee \neg V_2) \to V_1\big)$.

(a) Wandeln Sie φ mittels Wahrheitstafel in eine äquivalente aussagenlogische Formel in DNF um.

(b) Wenden Sie Algorithmus 3.39 an, um eine zu φ äquivalente Formel in KNF zu finden.

Aufgabe 3.14. Betrachten Sie die aussagenlogische Formel

$$\varphi \;:=\; \big(\neg(V_0 \leftrightarrow V_1) \wedge (\neg V_2 \vee V_0)\big).$$

(a) Wandeln Sie φ mittels Wahrheitstafel in eine äquivalente aussagenlogische Formel in DNF um.

(b) Wenden Sie Algorithmus 3.39 an, um eine zu φ äquivalente Formel in KNF zu finden.

Aufgabe 3.15. Für jedes $n \in \mathbb{N}_{>0}$ sei die aussagenlogische Formel φ_n definiert durch

$$\varphi_n \;:=\; \bigwedge_{i=1}^{n} (X_i \leftrightarrow Y_i).$$

(a) Beschreiben Sie die erfüllenden Belegungen $\mathcal{B}\colon \mathrm{Var}(\varphi_n) \to \{0,1\}$ für φ_n. Wie viele solche Belegungen gibt es?

(b) Geben Sie eine zu φ_n äquivalente Formel in DNF an.

(c) Beweisen Sie Satz 3.44, d.h. zeigen Sie, dass jede zu φ_n äquivalente Formel in DNF mindestens 2^n konjunktive Klauseln hat.

Hinweis: Eine Möglichkeit, dies zu zeigen, ist einen Beweis durch Widerspruch zu führen. Nehmen Sie dazu an, dass ψ_n eine zu φ_n äquivalente Formel in DNF ist, die aus weniger als 2^n konjunktiven Klauseln besteht. D.h. es gibt eine natürliche Zahl $N < 2^n$ und N konjunktive Klauseln $\kappa_1, \ldots, \kappa_N$, so dass $\psi_n = \kappa_1 \vee \cdots \vee \kappa_N$. Folgern Sie aus Ihrer Antwort aus Teil (a), dass mindestens eine der Klauseln $\kappa_1, \ldots, \kappa_N$ von mindestens zwei verschiedenen die Formel φ_n erfüllenden Belegungen wahr gemacht wird. Leiten Sie daraus einen Widerspruch her.

4 Graphen und Bäume

Bei Modellierungsaufgaben geht es oft darum, **Objekte** sowie **Beziehungen zwischen Objekten** zu beschreiben. **Graphen** und **Bäume** (Bäume sind Graphen mit bestimmten Eigenschaften) eignen sich dazu oft besonders gut.

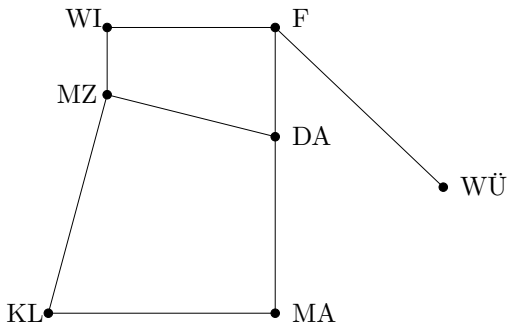
Anschaulich besteht ein Graph aus **Knoten** und **Kanten:**

- "Knoten" repräsentieren dabei "gleichartige Objekte".
- "Kanten" repräsentieren Beziehungen zwischen je zwei "Objekten".

Je nach Aufgabenstellung werden **ungerichtete Graphen** oder **gerichtete Graphen** verwendet.

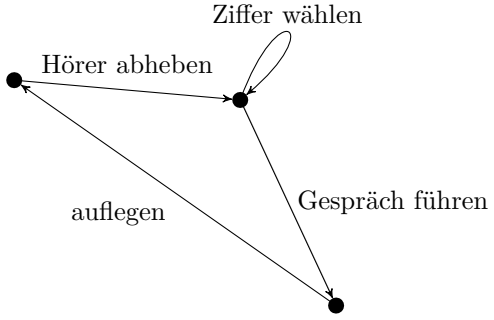
Beispiel 4.1.

(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:



4.1 Graphen

4.1.1 Grundlegende Definitionen

Definition 4.2 (ungerichteter Graph)**.** Ein **ungerichteter Graph** $G = (V, E)$ besteht aus einer Menge V, die **Knotenmenge von G** genannt wird, und einer Menge

ungerichteter Graph

$$E \subseteq \big\{\{i, j\} : i \in V,\, j \in V,\, i \neq j\big\},$$

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).

Knoten
Kanten

Beispiel 4.3. $G = (V, E)$ 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 4.1(a) zeigt diesen Graphen G in **graphischer Darstellung**: Knoten werden als Punkte dargestellt, Kanten als Verbindungslinien zwischen Punkten.

Beachte: Laut Definition 4.2 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 Graphs 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. Diese Definition ungerichteter Graphen wird in den meisten Büchern verwendet. In manchen Büchern werden davon abweichend in ungerichteten Graphen auch "Schleifen" der Form

erlaubt.

Notation 4.4. Sei $G = (V, E)$ ein ungerichteter Graph.

- Ein Knoten $v \in V$ heißt **inzident** mit einer Kante $e \in E$, falls $v \in e$. inzident
- 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

benachbart
adjazent

- 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. $e = \{v, v'\}$).

Grad
$\mathrm{Grad}_G(v)$

Definition 4.5 (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: $\mathrm{Grad}_G(v)$, ist die Anzahl der Kanten, die v als Endknoten haben. D.h.

$$\mathrm{Grad}_G(v) \;:=\; |\{e \in E : v \in e\}|.$$

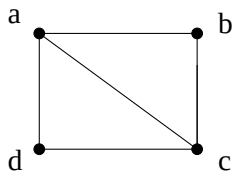
Der **Grad von G** ist

$\mathrm{Grad}(G)$

$$\mathrm{Grad}(G) \;:=\; \max\{\mathrm{Grad}_G(v) : v \in V\},$$

d.h. $\mathrm{Grad}(G)$ gibt den maximalen Grad eines Knotens von G an.

Beispiel.



$\mathrm{Grad}_G(a) = 3$
$\mathrm{Grad}_G(b) = 2$
$\mathrm{Grad}_G(c) = 3$
$\mathrm{Grad}_G(d) = 2$

$\mathrm{Grad}(G) = 3$

gerichteter Graph

Definition 4.6 (gerichteter Graph)**.** Ein **gerichteter Graph** $G = (V, E)$ besteht aus einer Menge V, die **Knotenmenge von G** genannt wird, und einer Menge

$$E \;\subseteq\; \big\{(i, j) : i \in V,\, j \in V\big\},$$

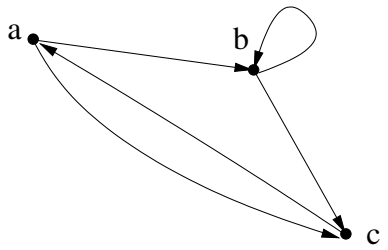
Knoten
(gerichtete) Kante

die **Kantenmenge von G** genannt wird. Die Elemente aus V heißen **Knoten** (bzw. "Ecken"), die Elemente aus E heißen (gerichtete) **Kanten** von G.

Beispiel 4.7. $G = (V, E)$ 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.
Graphische Darstellung:

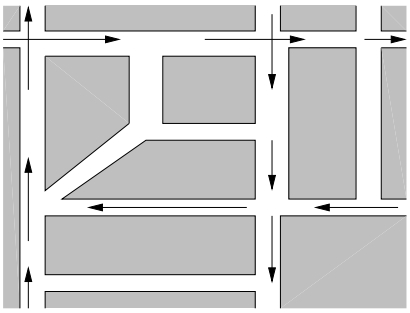


Knoten werden dabei als Punkte dargestellt; eine Kante der Form (i, j) wird als Pfeil von Knoten i nach Knoten j dargestellt, also

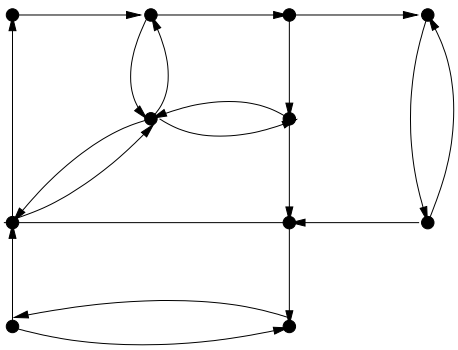
Notation 4.8. Sei $G = (V, E)$ ein gerichteter Graph.

- 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.
- Ein Knoten V heißt **inzident** mit einer Kante $e \in E$, falls v der Ausgangs- oder Endknoten von e ist. inzident
- Zwei Knoten $v, v' \in V$ heißen **benachbart** (bzw. **adjazent**), falls $(v, v') \in E$ oder $(v', v) \in E$. benachbart adjazent
- Eine Kante der Form (v, v) (d.h. deren Ausgangs- und Endpunkt identisch ist) wird **Schleife** bzw. **Schlinge** genannt. Schleife Schlinge

Beispiel 4.9 (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:



Weitere Beispiele zur Modellierung durch Graphen:

- Computer-Netzwerk:
  Knoten repräsentieren Computer; Kanten repräsentieren Netzwerkverbindungen.
- das World Wide Web:
  Knoten repräsentieren Webseiten; Kanten repräsentieren Hyperlinks.

Definition 4.10. Sei $G = (V, E)$ ein gerichteter Graph und sei $v \in V$ ein Knoten von G.

Ausgangsgrad
Aus-Grad$_G(v)$

- Der **Ausgangsgrad von v in G** (engl.: out-degree), kurz: Aus-Grad$_G(v)$, ist die Anzahl der Kanten, die v als Ausgangsknoten haben. D.h.:

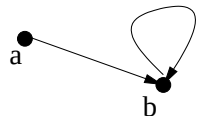
$$\text{Aus-Grad}_G(v) \ := \ |\{e \in E : \text{es ex. } v' \in V \text{ s.d. } e = (v, v')\}|$$

Eingangsgrad
Ein-Grad$_G(v)$

- Der **Eingangsgrad von v in G** (engl.: in-degree), kurz: Ein-Grad$_G(v)$, ist die Anzahl der Kanten, die v als Eingangsknoten haben. D.h.:

$$\text{Ein-Grad}_G(v) \ := \ |\{e \in E : \text{es ex. } v' \in V \text{ s.d. } e = (v', v)\}|.$$

Beispiel.

$G =$ 

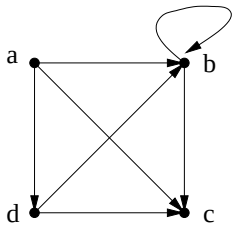
$\text{Ein-Grad}_G(a) = 0$
$\text{Ein-Grad}_G(b) = 2$
$\text{Aus-Grad}_G(a) = 1$
$\text{Aus-Grad}_G(b) = 1$

Bemerkung 4.11 (Darstellung von Graphen)**.** Es gibt mehrere Arten Graphen darzustellen, zum Beispiel

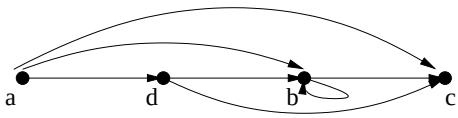
- **abstrakt**, durch Angabe der Knotenmenge V und der Kantenmenge E.
  Beispiel: $G_1 = (V_1, E_1)$ mit

$$V_1 = \{a, b, c, d\} \quad \text{und} \quad E_1 = \{(a, b), (a, c), (a, d), (b, b), (b, c), (d, b), (d, c)\}.$$

- **graphisch** (bzw. **anschaulich**). Der Beispiel-Graph G_1 wird graphisch dargestellt durch:



  oder, äquivalent dazu, durch

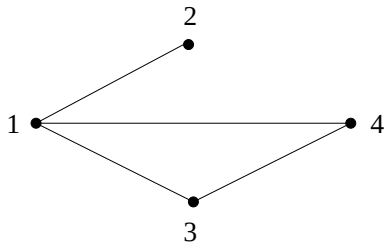


Adjazenzliste

- durch Angabe einer **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. Beispielweise der Graph $G_2 :=$



durch die Adjazenzliste

Knoten	Nachfolger
1	(2, 3, 4)
2	(1)
3	(1, 4)
4	(1, 3)

- durch Angabe einer **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, falls es keine Kante von i nach j gibt. Adjazenzmatrix

  Adjazenzmatrix von G_1:

	a	b	c	d
a	0	1	1	1
b	0	1	1	0
c	0	0	0	0
d	0	1	1	0

  Adjazenzmatrix von G_2:

	1	2	3	4
1	0	1	1	1
2	1	0	0	0
3	1	0	0	1
4	1	0	1	0

4.1.2 Wege in Graphen

Definition 4.12. Sei $G = (V, E)$ ein (gerichteter oder ungerichteter) Graph.

(a) Ein **Weg** in G ist ein Tupel Weg

$$(v_0, \ldots, v_l) \in V^{l+1},$$

für ein $l \in \mathbb{N}$, so dass f.a. i mit $0 \leq i \leq l$ 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 $(v_0, \ldots, v_l)$ 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.

einfacher Weg

(b) Ein Weg heißt **einfach**, wenn kein Knoten mehr als einmal in dem Weg vorkommt.

Kreis

(c) Ein Weg $(v_0, \ldots, v_l)$ heißt **Kreis**, wenn $l \geq 1$ und $v_l = v_0$ ist.

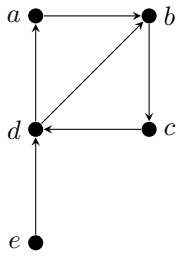
einfacher Kreis

(d) Ein Kreis $(v_0, \ldots, v_l)$ 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 $(v_0, \ldots, v_l)$, für die gilt: $l \geq 1$ und $v_l = v_0$ und $|\{v_0, \ldots, v_{l-1}\}| = l$.
- In einem **ungerichteten** Graphen G sind **einfache** Kreise genau die Wege der Form $(v_0, \ldots, v_l)$, für die gilt: $l \geq 3$ und $v_l = v_0$ und $|\{v_0, \ldots, v_{l-1}\}| = l$.

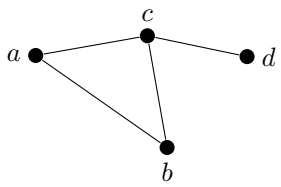
Beispiel 4.13.

(a) Wir betrachten den Graph



- (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) Wir betrachten den Graph



- (a, b, c, a) ist ein einfacher Kreis.
- (c, d, c) ist ein Kreis, aber kein einfacher Kreis.

Definition 4.14.

azyklisch

(a) Ein Graph heißt **azyklisch**, falls er keinen einfachen Kreis enthält.

DAG

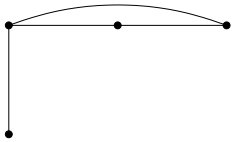
(b) Gerichtete azyklische Graphen werden im Englischen "directed acyclic graph", kurz: DAG, genannt.

Definition 4.15 (zusammenhängend, stark zusammenhängend)**.**

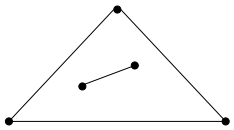
zusammenhängend

(a) Ein ungerichteter Graph $G = (V, E)$ heißt **zusammenhängend**, wenn für alle Knoten $v, w \in V$ gilt: Es gibt in G einen Weg von v nach w.

Beispiel.



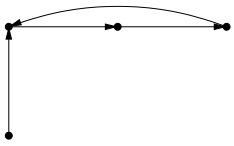
ist zusammenhängend.



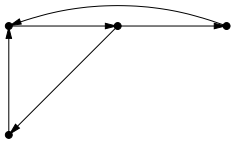
ist nicht zusammenhängend.

(b) Ein gerichteter Graph $G = (V, E)$ heißt **stark zusammenhängend**, wenn für alle Knoten $v, w \in V$ gilt: Es gibt in G einen Weg von v nach w. stark zusammenhängend

Beispiel.



ist **nicht** stark zusammenhängend (da es z.B. keinen Weg vom Knoten links oben zum Knoten links unten gibt).



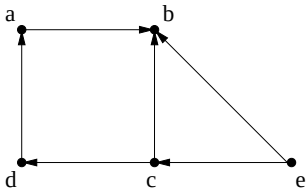
ist stark zusammenhängend.

Definition 4.16 (Hamilton-Kreis und Hamilton-Weg)**.** Sei $G = (V, E)$ ein (gerichteter oder ein ungerichteter) Graph.

(a) Ein Weg $W = (v_0, \ldots, v_l)$ heißt **Hamilton-Weg**, wenn jeder **Knoten** aus V genau einmal in W vorkommt. Hamilton-Weg

(b) Ein Weg $W = (v_0, \ldots, v_l)$ heißt **Hamilton-Kreis**, wenn $l \geq 1$ und $v_l = v_0$ und $(v_0, \ldots, v_{l-1})$ ein Hamilton-Weg ist. Hamilton-Kreis

Beispiel. Der Graph G



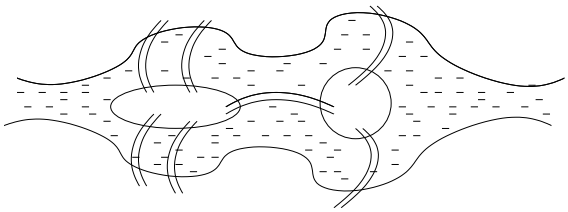
hat einen Hamilton-Weg, nämlich (e, c, d, a, b), aber keinen Hamilton-Kreis (da Aus-Grad$_G(b) = 0$ ist).

Ein Anwendungsbeispiel: Beim Problem des **Handlungsreisenden** (engl.: Travelling Salesman's Problem) geht es darum, eine Rundreise durch n Städte so durchzuführen, dass jede Stadt genau 1 mal besucht wird. Es geht also darum, einen Hamilton-Kreis zu finden. Das Problem, zu einem gegebenen Graphen zu entscheiden, ob er einen Hamilton-Kreis besitzt, ist algorithmisch ein Handlungsreisendenproblem

schwieriges Problem: man kann zeigen, dass es (genau wie das aussagenlogische Erfüllbarkeitsproblem) NP-vollständig ist.

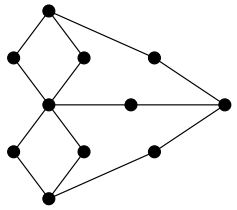
Im Gegensatz zu Hamilton-Wegen (bei denen es darum geht, einen Weg zu finden, der jeden **Knoten** des Graphen genau einmal besucht), geht es bei den im Folgenden betrachteten **Euler-Wegen** darum, einen Weg zu finden, der jede **Kante** des Graphen genau einmal besucht.

Beispiel 4.17 (Königsberger Brückenproblem)**.** In der Stadt Königsberg gab es im 18 Jahrhundert 7 Brücken über den Fluss Pregel, die die Ufer und 2 Inseln miteinander verbanden. Skizze:



Frage: Gibt es einen Spaziergang, der jede der 7 Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt?

Die obige Skizze lässt sich folgendermaßen 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 4.18 (Euler-Kreise und Euler-Wege)**.** Sei $G = (V, E)$ ein ungerichteter Graph.

Euler-Weg (a) Ein Weg $W = (v_0, \ldots, v_l)$ heißt **Euler-Weg**, wenn W jede Kante aus E genau einmal enthält, d.h. wenn es für jedes $e \in E$ genau ein $i \in \{0, \ldots, l-1\}$ gibt, so dass $e = \{v_i, v_{i+1}\}$.

Euler-Kreis (b) Ein Weg $w = (v_0, \ldots, v_l)$ heißt **Euler-Kreis**, wenn W ein Euler-Weg ist und $v_0 = v_l$ ist.

Satz 4.19 (Existenz von Euler-Kreisen und Euler-Wegen)**.** *Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph, dessen Knotenmenge endlich ist. 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, der kein Euler-Kreis ist $\iff$ *es gibt in G genau zwei Knoten mit ungeradem Grad.*

Beweis:

(a) "$\Longrightarrow$" Sei $K = (v_0, \ldots, v_l)$ ein Euler-Kreis. Insbesondere: $v_0 = v_l$.

Schritt 1: Jeder Knoten $v \in \{v_0, \ldots, v_{l-1}\}$ hat geraden Grad, denn: Sei $v \in \{v_0, \ldots, v_{l-1}\}$ beliebig. Zu jedem $i \in \{0, \ldots, l-1\}$ 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. falls $i = 0$, $\{v_0, v_1\}$ und $\{v_{l-1}, v_0\}$ (beachte: $v_0 = v_l$).

Da der Euler-Kreis K jede Kante von G genau einmal enthält, gilt somit folgendes: Ist $k = |\{i \in \{0, \ldots, l-1\} : v = v_i\}|$ (d.h. k gibt an, wie oft v im Tupel $(v_0, \ldots, v_{l-1})$ vorkommt), so ist $\text{Grad}_G(v) = 2 \cdot k$. Daher hat jeder Knoten $v \in \{v_0, \ldots, v_{l-1}\}$ geraden Grad.

Schritt 2: $\{v_0, \ldots, v_{l-1}\} = V$, 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 $v, w \in \{v_0, \ldots, v_{l-1}\}$.

Schritt 3: Aus Schritt 1 und Schritt 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

$$W \;=\; (v_0, \ldots, v_l)$$

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 $v_l = v_0$.

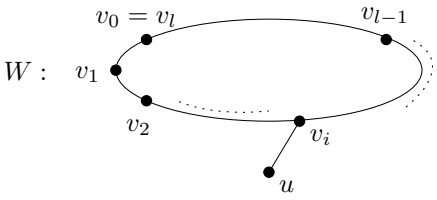
Zu zeigen: W ist ein Euler-Kreis.

Angenommen, W ist **kein** Euler-Kreis. Dann gibt es in G eine Kante e, die nicht auf W liegt, die aber mit mindestens einem Knoten auf W inzident ist (um dies zu sehen, nutzen wir, dass G zusammenhängend ist). 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

$$W' \;:=\; (u, v_i, v_{i+1}, \ldots, v_{l-1}, v_0, v_1, \ldots, v_i)$$

ein Weg der Länge $l + 1$, der keine Kante(n) mehrfach enthält.

Skizze:



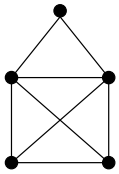
Dies widerspricht aber der Tatsache, dass W ein Weg **maximaler** Länge ist.

(b) Folgt leicht aus (a). Details: Übung. □

Beispiel 4.20. Mit Hilfe von Satz 4.19 können wir das Königsberger Brückenproblem aus Beispiel 4.17 leicht lösen: Es gibt **keinen** Spaziergang, der jede der 7 Brücken genau einmal überquert und zum Ausgangspunkt zurückkehrt, denn: ein solcher Spaziergang würde gerade einem Euler-Kreis im Graphen $G_{\text{Königsberg}}$ entsprechen. Dieser Graph besitzt aber 4 Knoten von ungeradem Grad und kann daher laut Satz 4.19(a) keinen Euler-Kreis besitzen.

Beispiel 4.21.

Frage: Kann man die Figur



in einem Zug nachzeichnen? D.h: Besitzt dieser Graph einen Euler-Weg?

Unter Verwendung von Satz 4.19 kann man die Frage leicht beantworten, indem man nachzählt, wie viele Knoten von ungeradem Grad es gibt. Im obigen Graphen gibt es genau 2 Knoten von ungeradem Grad. Gemäß Satz 4.19 besitzt G also einen Euler-Weg, der kein Euler-Kreis ist.

4.1.3 Ähnlichkeit zweier Graphen

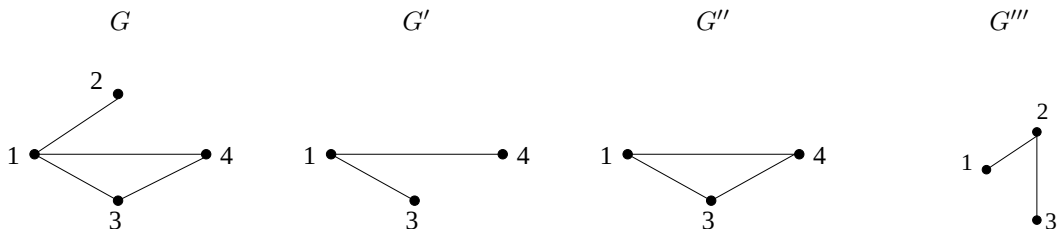
Die folgende Definition formalisiert, wann ein Graph G' in einem Graphen G "enthalten" ist.

Teilgraph
induzierter Teilgraph

Definition 4.22 (Teilgraph)**.** Seien $G = (V, E)$ und $G' = (V', E')$ zwei (gerichtete oder ungerichtete) Graphen. G' heißt **Teilgraph von** G, falls $V' \subseteq V$ und $E' \subseteq E$. G' heißt **induzierter Teilgraph von** G, falls

$$V' \subseteq V \quad \text{und} \quad E' = \{e \in E : \text{die mit } e \text{ inzidenten Knoten liegen in } V'\}.$$

Beispiel 4.23. 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.

Bemerkung 4.24. Zwei Graphen $G = (V, E)$ und $G' = (V', E')$ sind **gleich** (kurz: $G = G'$), falls sie dieselbe Knotenmenge und dieselbe Kantenmenge besitzen. D.h.:

$$G = G' \ :\Longleftrightarrow \ V = V' \text{ und } E = E'.$$

Zwei Graphen G und G' sind *"prinzipiell gleich"* (Fachbegriff: **isomorph**, kurz: $G \cong G'$), falls G' aus G durch Umbenennung der Knoten entsteht. Dies wird durch die folgende Definition präzisiert:

Definition 4.25 (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 $f : V \to V'$ gibt, so dass für alle Knoten $i \in V$ und $j \in V$ gilt: isomorph $G \cong G'$

- falls G und G' gerichtet sind:

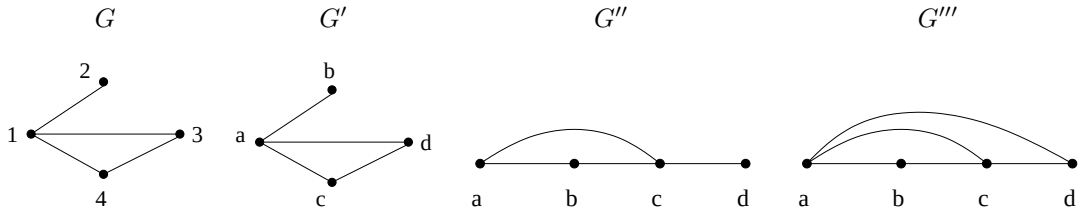
$$(i, j) \in E \iff \big(f(i), f(j)\big) \in E'$$

- falls G und G' ungerichtet sind:

$$\{i, j\} \in E \iff \{f(i), f(j)\} \in E'$$

Eine solche Abbildung f wird **Isomorphismus von G nach G′** genannt. Isomorphismus

Beispiel 4.26. Es seien:



Dann gilt:

- $G \cong G'$ via $f\colon \{1, 2, 3, 4\} \to \{a, b, c, d\}$ mit $f(1) = a$, $f(2) = b$, $f(3) = d$, $f(4) = c$.
- $G \cong G''$ 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'' ist nicht isomorph zu G''', kurz: $G'' \ncong G'''$, da G''' **mehr** Kanten als G'' hat.

4.1.4 Markierte Graphen

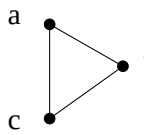
Bemerkung 4.27. Viele Modellierungsaufgaben erfordern, dass den Knoten oder den Kanten eines Graphen weitere Informationen zugeordnet werden. Dies wird durch so genannte **Markierungsfunktionen** für Knoten oder Kanten formalisiert.

Eine **Knotenmarkierung** eines (gerichteten oder ungerichteten) Graphen $G = (V, E)$ ist eine Abbildung $\mathrm{MV}\colon V \to \mathrm{WV}$, wobei WV ein geeigneter Wertebereich ist. In dem Graph aus Beispiel 4.1(a) könnte man beispielweise eine Knotenmarkierung Einwohnerzahl$\colon V \to \mathbb{N}$ einführen, die jedem Knoten die Einwohnerzahl der zugehörigen Stadt zuordnet. Eine **Kantenmarkierung** von G ist eine Abbildung $\mathrm{ME}\colon E \to \mathrm{WE}$, wobei WE ein geeigneter Wertebereich ist. In dem Graph aus Beispiel 4.1(a) könnte man beispielweise eine Kantenmarkierung Entfernung$\colon E \to \mathbb{N}$ einführen, die jeder Kante die Länge (in km) des von der Kante repräsentierten Autobahnteilstücks zuordnet. Knotenmarkierung Kantenmarkierung

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.

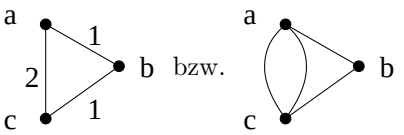
Definition 4.28 (Multigraph)**.** Ein **Multigraph** (G, m) besteht aus einem (gerichteten oder ungerichteten) Graphen $G = (V, E)$ und einer Kantenmarkierung $m\colon E \to \mathbb{N}$. Multigraph

Beispiel. Multigraph (G, m) mit



und $m(\{a,b\}) = 1$, $m(\{b,c\}) = 1$, $m(\{c,a\}) = 2$

graphische Darstellung von (G, m):



4.1.5 Zuordnungsprobleme

Beispiel 4.29. Typische Aufgabenstellungen:

(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.

(b) Eine Gruppe unterschiedlich ausgebildeter Piloten soll so auf Flugzeuge verteilt werden, dass jeder das ihm zugeteilte Flugzeug fliegen kann.

Beide Situationen lassen sich gut durch ungerichtete Graphen modellieren:

Zu (a): Modelliere die Situation durch den Graphen $G_T := (V_T, E_T)$ mit

$$\begin{aligned} V_T &:= \{x : x \text{ ist ein Vereinsmitglied}\} \\ E_T &:= \{\{x, y\} : x \text{ und } y \text{ sind befreundete Vereinsmitglieder}\}. \end{aligned}$$

Ziel: Finde eine größtmögliche Anzahl von Doppelpaarungen, d.h. finde eine möglichst große Menge $E' \subseteq E_T$, so dass kein Vereinsmitglied Endpunkt von mehr als einer Kante aus E' ist.

Zu (b): Modelliere die Situation durch den Graphen $G_F := (V_F, E_F)$ mit

$$\begin{aligned} V_F &:= \{x : x \text{ ist ein Pilot}\} \dot\cup \{y : y \text{ ist ein Flugzeug}\}, \\ E_F &:= \{\{x, y\} : \text{Pilot } x \text{ kann Flugzeug } y \text{ fliegen}\}. \end{aligned}$$

Ziel: Stelle einen Flugplan auf, so dass jeder Pilot das ihm zugeteilte Flugzeug fliegen kann, d.h. finde eine möglichst große Menge $E' \subseteq E_F$, so dass kein Element aus V_F Endpunkt von mehr als einer Kante in E' ist.

Die gesuchten Kantenmengen E' aus (a) und (b) werden auch **Matching** genannt:

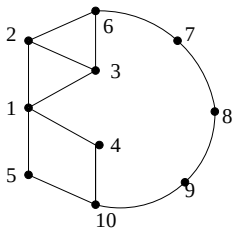
Matching

Definition 4.30. Sei $G = (V, E)$ ein ungerichteter Graph. Eine Kantenmenge $E' \subseteq E$ heißt **Matching** (bzw. **Menge unabhängiger Kanten**), falls gilt: kein Knoten aus V ist Endpunkt von mehr als einer Kante aus E'.

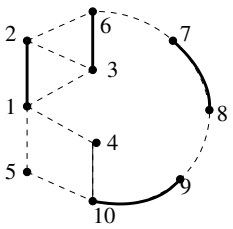
Ziel in Beispiel 4.29 (a) und (b) ist es, ein Matching maximaler Größe (d.h. eins, das so viele Kanten wie möglich enthält) zu finden.

Beispiel 4.31. In einem Tennisverein mit 10 Mitgliedern und "Freundschaftsgraph"

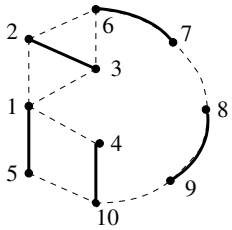
G_T =



sind z.B. die folgenden beiden Kantenmengen Matchings:



und



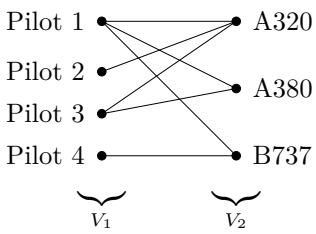
$E' = \big\{\{1,2\},\{3,6\},\{7,8\},\{9,10\}\big\}$ $E'' = \big\{\{1,5\},\{4,10\},\{8,9\},\{6,7\},\{2,3\}\big\}$.

In Beispiel 4.29(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). Solche Graphen werden *bipartite Graphen* genannt:

Definition 4.32. 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, dass jede Kante aus E einen Endknoten in V_1 und einen Endknoten in V_2 hat. bipartit

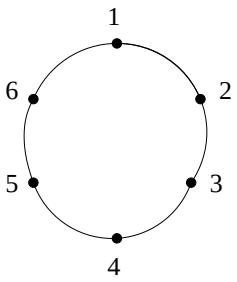
Beispiel 4.33.

(a) Der Graph

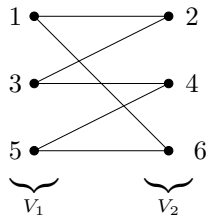


ist bipartit.

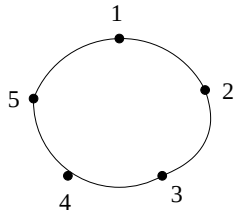
(b) Der Graph



ist bipartit, denn wähle $V_1 = \{1,3,5\}$ und $V_2 = \{2,4,6\}$; andere graphische Darstellung des Graphen:



(c) Der Graph



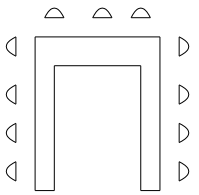
ist **nicht** bipartit (denn angenommen doch, seien V_1 und V_2 die beiden disjunkten Teilmengen der Knotenmenge, so dass jede Kante des Graphen einen Endknoten in V_1 und einen Endknoten in V_2 hat; o.B.d.A. nehmen wir an, dass $1 \in V_1$ ist. Dann muss aber gelten: $2 \in V_2$, $3 \in V_1$, $4 \in V_2$, $5 \in V_1$, also $V_1 = \{1, 3, 5\}$ und $V_2 = \{2, 4\}$. **Aber:** es gibt eine Kante zwischen 1 und 5, und beide Knoten gehören zu V_1. $\lightning$)

Allgemein gilt: Ist $n \in \mathbb{N}_{>0}$ und ist G ein Kreis auf n Knoten (wie in (b) für $n = 6$ und in (c) für $n = 5$), so gilt:

$$G \text{ ist bipartit } \iff n \text{ ist gerade.}$$

Ein weiteres typisches Beispiel für ein Zuordnungsproblem:

Beispiel 4.34. 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 mit

$$\begin{aligned} V &= \{x : \text{Person } x \text{ soll zur Feier eingeladen werden}\} \text{ und} \\ E &= \big\{\{x, y\} : \text{Person } x \text{ kann Person } y \text{ nicht leiden oder} \\ &\qquad \text{Person } y \text{ kann Person } x \text{ nicht leiden}\big\}, \end{aligned}$$

d.h. Kanten im Konfliktgraphen zeigen auf, wer im Konflikt mit wem steht.

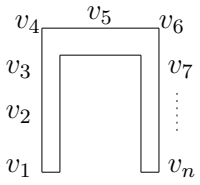
Schritt 2: Bilde das Komplement $\tilde{G} = (\tilde{V}, \tilde{E})$ des Konfliktgraphen, d.h. betrachte

$$\begin{aligned} \tilde{V} &:= V, \\ \tilde{E} &:= \big\{\{x, y\} : x, y \in V,\, x \neq y,\, \{x, y\} \notin E\big\}, \end{aligned}$$

d.h. Kanten in $\tilde{G}$ zeigen an, wer prinzipiell neben wem platziert werden könnte.

Schritt 3: Suche einen Hamilton-Weg in $\tilde{G}$.

Wenn $(v_1, \ldots, v_n)$ (mit $n = |\tilde{V}|$) ein Hamilton-Weg in $\tilde{G}$ ist, dann kann man die Sitzordnung folgendermaßen festlegen:



Falls es in $\tilde{G}$ keinen Hamilton-Weg gibt, so weiß man, dass es **keine Möglichkeit gibt**, die geladenen Gäste so an einer hufeisenförmigen Tafel zu platzieren, dass niemand neben jemandem sitzt, den er nicht leiden kann.

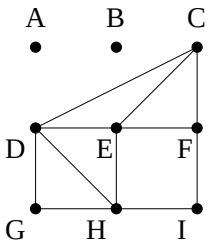
Ein möglicher Ausweg: Verteile die Gäste auf **mehrere** Tische:

Beispiel 4.35. Die Gäste einer Familienfeier sollen so an mehreren (möglichst wenigen) Tischen platziert werden, dass Personen, die sich nicht ausstehen können, an verschiedenen Tischen sitzen. Diese Aufgabe kann folgendermaßen modelliert werden. Die verfügbaren Tische werden durchnummeriert mit den Zahlen $1, 2, 3, \ldots$. Die geladenen Gäste und die herrschenden Konflikte zwischen Gästen werden durch den in Beispiel 4.34 betrachteten Konfliktgraphen $G = (V, E)$ repräsentiert. Die Zuordnung, wer an welchem Tisch sitzen soll, wird durch eine Knotenmarkierung $m\colon V \to \mathbb{N}_{>0}$ repräsentiert ($m(x) = i$ bedeutet dann, dass Person x am Tisch i sitzen soll).

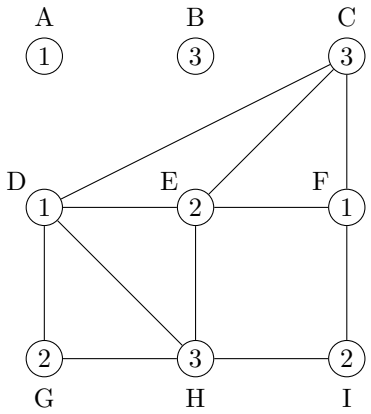
Ziel: Finde eine **konfliktfreie Knotenmarkierung** $m\colon V \to \mathbb{N}_{>0}$, d.h. eine Knotenmarkierung, so dass für jede Kante $\{x, y\} \in E$ gilt: $m(x) \neq m(y)$. Dabei soll $|\text{Bild}(m)|$ möglichst klein sein (dies entspricht dem Ziel, die Gäste auf möglichst **wenige** Tische zu verteilen).

Definition 4.36. Sei $G = (V, E)$ ein ungerichteter Graph. Eine Funktion $m\colon V \to \mathbb{N}$ heißt **konfliktfreie Knotenmarkierung** (oder: konfliktfreie **Färbung**), wenn für jede Kante $\{x, y\} \in E$ gilt: $m(x) \neq m(y)$. konfliktfreie Knotenmarkierung

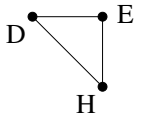
Beispiel 4.37. Familienfeier mit Gästen A, B, C, D, E, F, G, H, I und folgendem Konfliktgraphen



Eine konfliktfreie Knotenmarkierung (d.h. Platzierung an verschiedene Tische) $m\colon V \to \mathbb{N}$:



Hier ist für jeden Knoten $v \in V$ der Wert $m(v)$ in den Kreis geschrieben, der den Knoten v repräsentiert. Für die hier gegebene Markierung m gilt $|\text{Bild}(m)| = 3$, die Gäste werden also auf 3 Tische verteilt. Dies ist "optimal", da der Konfliktgraph ein Dreieck, z.B.



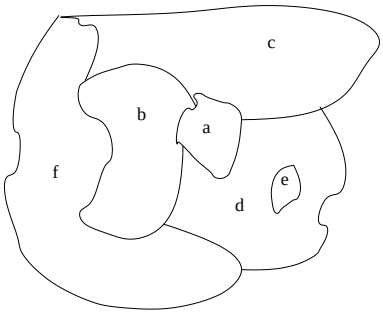
als Teilgraph enthält – deshalb muss für jede konfliktfreie Knotenmarkierung m' gelten: $|\text{Bild}(m')| \geq 3$.

4-Farben-Problem

Bemerkung 4.38. Die wohl berühmteste Aufgabe dieser Art von Markierungs- oder Färbungsaufgaben ist das so genannte **4-Farben-Problem**. Dabei handelt es sich um die Hypothese, dass vier verschiedene Farben ausreichen, um eine Landkarte so einzufärben, dass zwei Staaten, die ein Stück gemeinsamer Grenze haben, durch unterschiedliche Farben dargestellt werden. Erst 1976 wurde diese Hypothese bewiesen, und zwar durch eine Fallunterscheidung mit mehr als 1000 Fällen, die mit Hilfe eines Computerprogramms gelöst wurde.

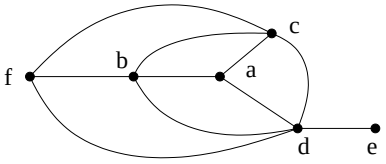
Beispiel.

Eine (kleine) Landkarte:



zugehöriger Konfliktgraph:

Knoten $\widehat{=}$ Staaten
Kanten $\widehat{=}$ Staaten mit gemeinsamer Grenze

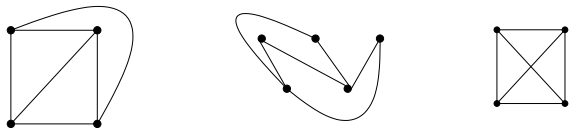


Da bei den vier Knoten a, b, c, d paarweise jeder zu jedem benachbart ist, muss eine konfliktfreie Färbung diesen 4 Knoten 4 verschiedene Farben zuordnen – für a, b, c, d etwa rot, gelb, grün, blau. Da f außerdem mit b, c, d benachbart ist, muss f dann wieder rot gefärbt sein; e kann jede Farbe außer blau erhalten.

Die zu Landkarten gehörenden Konfliktgraphen haben eine besondere Eigenschaft: Sie sind planar.

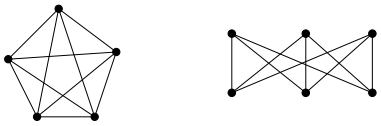
Definition 4.39. Ein Graph G heißt **planar**, wenn er so in die Ebene gezeichnet werden kann, dass seine Kanten sich nicht kreuzen. planar

Beispiel. Planare Graphen sind:



(Der dritte Graph kann wie der erste Graph kreuzungsfrei in die Ebene gezeichnet werden.)

Nicht-planare Graphen sind:



Beispiel 4.40. Weitere Beispiele von Anwendungen, die durch Finden konfliktfreier Färbungen im Konfliktgraphen gelöst werden können:

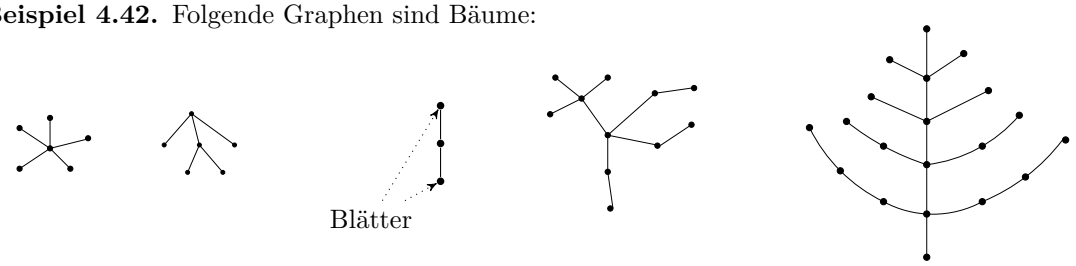
Knoten	Kante zwischen x und y	Farbe bzw. Markierung
Staat auf Karte	haben gemeinsame Grenze	Farbe
Gast auf Familienfeier	können sich nicht leiden	Tischnummer
Vorlesung	haben gemeinsame Teilnehmer	Termin
Variable im Programm	ihre Werte werden gleichzeitig benötigt	Registerspeicher
Prozess	benötigt dieselben Ressourcen	Ausführungstermin

4.2 Bäume

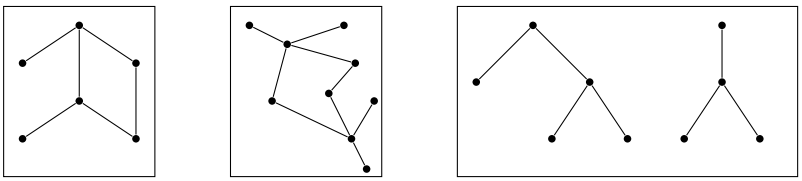
4.2.1 Ungerichtete Bäume

Definition 4.41. Ein **ungerichteter Baum** ist ein ungerichteter, zusammenhängender Graph $G = (V, E)$, der keinen einfachen Kreis enthält. Diejenigen Knoten in V, die den Grad 1 haben, heißen **Blätter** des Baums. ungerichteter Baum, Blätter

Beispiel 4.42. Folgende Graphen sind Bäume:



Folgende Graphen sind keine Bäume:

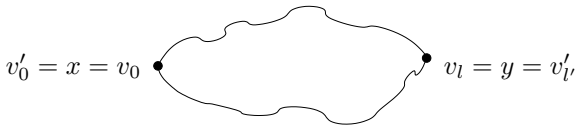


Beobachtung 4.43. Sei $B = (V, E)$ ein ungerichteter Baum. Da B zusammenhängend ist und keinen einfachen Kreis enthält, gilt f.a. Knoten $x, y \in V$:

es gibt in B genau einen einfachen Weg von x nach y

(denn: Da B zusammenhängend ist, gibt es mindestens einen einfachen Weg von x nach y. Angenommen, $(v_0, \ldots, v_l)$ und $(v'_0, \ldots, v'_{l'})$ sind zwei verschiedene einfache Wege von x nach y. Insbesondere gilt dann $v_0 = x = v'_0$ und $v_l = y = v'_{l'}$.

Skizze:

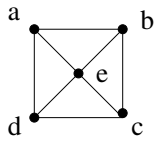


Dann ist aber $(v_0, \ldots, v_l, v'_{l'-1}, \ldots, v'_0)$ ein Kreis. Dieser Kreis enthält einen einfachen Kreis. Dann kann B aber kein Baum sein. $\lightning$)

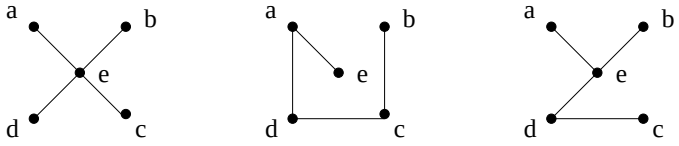
Spannbaum

Definition 4.44. 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.

Beispiel 4.45. Der Graph



hat z.B. folgende Spannbäume:



Satz 4.46. *Sei $G = (V, E)$ ein ungerichteter Graph, dessen Knotenmenge endlich ist. Dann gilt:*

Es gibt (mindestens) einen Spannbaum von G $\iff$ G ist zusammenhängend.

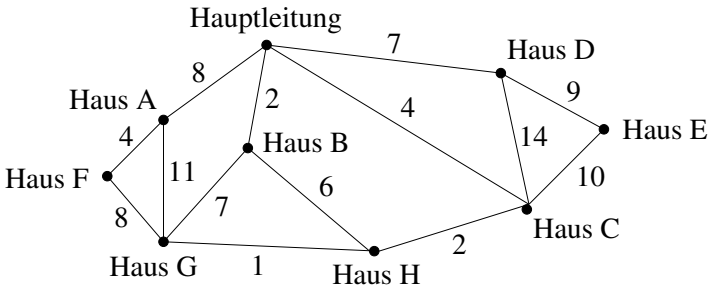
Beweis: "$\Longrightarrow$" klar, "$\Longleftarrow$" Übung. □

Geht man von einem zusammenhängenden Graphen zu einem seiner Spannbäume über, so verkleinert man die Kantenmenge von $|E|$ auf – gemäß dem folgenden Satz 4.48 – $|V|-1$ Kanten, ohne dabei den Zusammenhang des Graphen aufzugeben. Mit dem Begriff des Spannbaums wird also ein "bezüglich der Kantenzahl kostengünstigerer Zusammenhang" modelliert.

Manche konkreten Probleme lassen sich durch Graphen modellieren, deren Kanten mit bestimmten Werten markiert sind, so dass zur Lösung des Problems ein Spannbaum gesucht wird, bei dem die Summe seiner Kantenmarkierungen so klein wie möglich ist (engl: "**minimum spanning tree** problem").

Beispiel 4.47 (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, und
- der Wert, mit dem die Kante markiert ist, beschreibt, wie teuer (in 1000 €) es ist, diese Leitung unterirdisch zu verlegen.



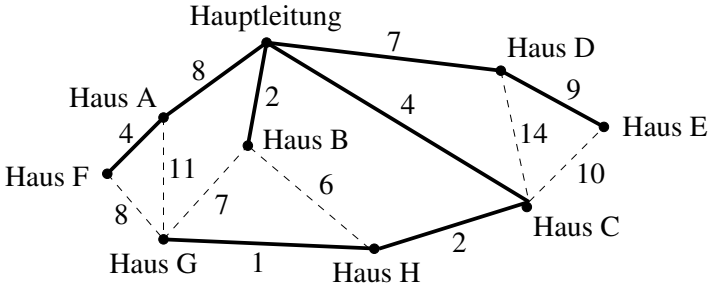
Ziel ist, Leitungen so zu verlegen, dass

(1) jedes Haus ans Kabelfernsehen angeschlossen ist und

(2) die Kosten für das Verlegen der Leitungen so gering wie möglich sind.

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. minimaler Spannbaum

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. Effiziente Verfahren zum Finden minimaler Spannbäume werden Sie in der Vorlesung "Algorithmentheorie" (GL-1) kennenlernen.

Satz 4.48 (Anzahl der Kanten eines Baums)**.** *Sei $B = (V, E)$ ein* ***ungerichteter Baum****, dessen Knotenmenge endlich und nicht-leer ist. Dann 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 $\bullet$ mit $E = \varnothing$. Für diesen Graphen gilt:

$$|E| \;=\; 0 \;=\; 1 - 1 \;=\; |V| - 1.$$

Induktionsschritt: $n \to n + 1$

Sei $n \in \mathbb{N}$ mit $n \geq 1$ beliebig.

Induktionsannahme: Für jeden ungerichteten Baum $B' = (V', E')$ mit $0 \neq |V'| \leq n$ gilt:

$$|E'| \;=\; |V'| - 1.$$

Behauptung: Für jeden ungerichteten Baum $B = (V, E)$ mit $0 \neq |V| = n+1$ gilt: $|E| \;=\; |V|-1$.

Beweis: Sei $B = (V, E)$ ein ungerichteter Baum mit $|V| = n + 1$. Da B zusammenhängend ist und $|V| = n + 1 \geq 1 + 1 = 2$ ist, muss E mindestens eine Kante enthalten. Sei $\{x, y\}$ eine Kante in E.

Sei $\tilde{G} = (\tilde{V}, \tilde{E})$ der Graph, der aus B durch Löschen der Kante $\{x, y\}$ entsteht. D.h.:

$$\tilde{V} \;=\; V \qquad \text{und} \qquad \tilde{E} \;=\; E \setminus \{\{x, y\}\}.$$

Sei

$$V_x \;:=\; \{v \in V : \text{in } \tilde{G} \text{ gibt es einen Weg von } x \text{ nach } v\}$$

und

$$V_y \;:=\; \{v \in V : \text{in } \tilde{G} \text{ gibt es einen Weg von } y \text{ nach } v\}.$$

Da B zusammenhängend ist, gilt $V = V_x \cup V_y$. Da B keinen einfachen Kreis enthält, gilt $V_x \cap V_y = \varnothing$. Seien $B_x := (V_x, E_x)$ und $B_y := (V_y, E_y)$ die durch V_x bzw. V_y induzierten Teilgraphen von $\tilde{G}$, d.h.: $E_x := \tilde{E} \cap \{\{u, v\} : u, v \in V_x,\ u \neq v\}$ und $E_y := \tilde{E} \cap \{\{u, v\} : u, v \in V_y,\ u \neq v\}$. Da B ein ungerichteter Baum ist, sind auch B_x und B_y ungerichtete Bäume. Es gilt:

(1) $V = V_x \,\dot\cup\, V_y,\ V_x \neq \varnothing$ und $V_y \neq \varnothing$.

(2) $E = E_x \,\dot\cup\, E_y \,\dot\cup\, \{\{x, y\}\}$.

(3) B_x ist ein ungerichteter Baum.

(4) B_y ist ein ungerichteter Baum.

Wegen (1) und $|V| = n + 1$ gilt insbesondere, dass $0 \neq |V_x| \leq n$ und $0 \neq |V_y| \leq n$. Wegen (3) und (4) gilt daher gemäß Induktionsannahme:

(5) $|E_x| = |V_x| - 1$ und $|E_y| = |V_y| - 1$.

Aus (1) und (2) folgt daher:

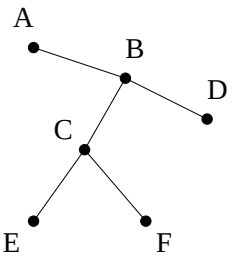
$$|E| \;\overset{(2)}{=}\; |E_x| + |E_y| + 1 \;\overset{(5)}{=}\; |V_x| - 1 + |V_y| - 1 + 1 \;\overset{(1)}{=}\; |V| - 1.$$

□

4.2.2 Gerichtete Bäume

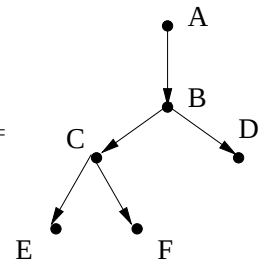
Einen **gerichteten Baum** erhält man, indem man in einem ungerichteten Baum einen Knoten als "Wurzel" auswählt und alle Kanten in die Richtung orientiert, die von der Wurzel weg führt. gerichteten Baum

Beispiel 4.49. Ungerichteter Baum:



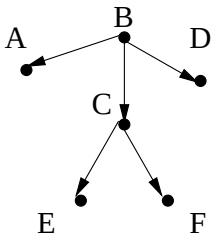
- Zugehöriger gerichteter Baum mit Wurzel A:

$G_A :=$



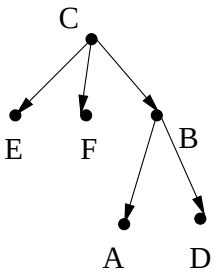
- Zugehöriger gerichteter Baum mit Wurzel B:

$G_B :=$



- Zugehöriger gerichteter Baum mit Wurzel C:

$G_C :=$



Definition 4.50.

gerichteter Baum

(a) Ein gerichteter Graph $G = (V, E)$ heißt **gerichteter Baum**, falls er folgende Eigenschaften hat:

Wurzel

(1) G besitzt genau einen Knoten $w \in V$ mit Ein-Grad$_G(w) = 0$. Dieser Knoten wird **Wurzel** genannt.

(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$_G(v) \leq 1$.

Höhe
Tiefe

(b) Sei $B = (V, E)$ ein gerichteter Baum. Die **Höhe** (bzw. **Tiefe**, engl.: height, depth) von B ist die Länge eines längsten Weges in B.

Beispiel: In Beispiel 4.49 hat G_A die Höhe 3, G_B die Höhe 2 und G_C die Höhe 2.

Blätter

(c) Sei $B = (V, E)$ ein gerichteter Baum. Diejenigen Knoten, deren Aus-Grad 0 ist, heißen **Blätter**.

Beispiel: In Beispiel 4.49 hat G_A die Blätter D, E, F; G_B die Blätter A, D, E, F und G_C die Blätter A, D, E, F.

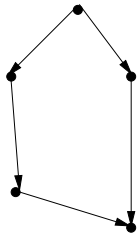
innere Knoten

(d) Diejenigen Knoten eines gerichteten Baums, die weder Wurzel noch Blätter sind, heißen **innere Knoten**.

Beobachtung 4.51.

(a) Jeder gerichtete Baum ist ein gerichteter azyklischer Graph (kurz: DAG, vgl. Definition 4.14). Aber es gibt gerichtete Graphen, die keine gerichteten Bäume sind.

Beispiel:



ist ein DAG, aber kein gerichteter Baum.

(b) Für jeden gerichteten Baum $B = (V, E)$, dessen Knotenmenge endlich und nicht-leer ist, gilt:

$$|E| \;=\; |V| - 1.$$

Dies folgt unmittelbar aus Satz 4.48, 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 4.50 kann man die gerichteten Bäume, deren Knotenmenge endlich und nicht-leer ist, auch folgendermaßen definieren:

Definition 4.52. Die Klasse der gerichteten Bäume mit endlicher, nicht-leerer Knotenmenge ist rekursiv wie folgt definiert:

Basisregel: Ist V eine Menge mit $|V| = 1$, so ist $B := (V, \varnothing)$ ein gerichteter Baum.

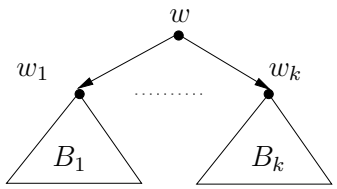
Skizze: $B :=$ ●

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 $B_1 = (V_1, E_1), \ldots, B_k = (V_k, E_k)$ gerichtete Bäume mit paarweise disjunkten Knotenmengen (d.h. $V_i \cap V_j = \varnothing$ 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 $V := \{w\} \cup V_1 \cup \cdots \cup V_k$ und $E := E_1 \cup \cdots \cup E_k \cup \{(w, w_i) : i \in \{1, \ldots, k\}\}$ ein gerichteter Baum.

Skizze:

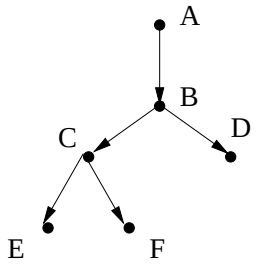
B :=



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 gerichteten Bäume $B_1, \ldots, B_k$ sind.

Notation 4.53. Sei $B = (V, E)$ ein gerichteter 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 4.49 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 2 Kinder hat. Mit solchen Bäumen kann man z.B. Kaskaden von JA-NEIN-Entscheidungen oder Binär-Codierung beschreiben.

Definition 4.54 (Binärbaum, vollständiger Binärbaum)**.**

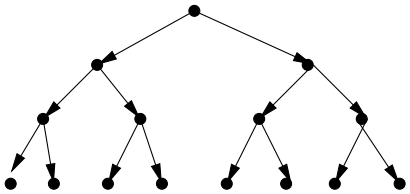
(a) Ein gerichteter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt: Aus-Grad$_B(v) \leq 2$. Binärbaum

(b) Ein Binärbaum $B = (V, E)$ heißt **vollständiger Binärbaum**, falls gilt: vollständiger Binärbaum

(1) Jeder Knoten, der kein Blatt ist, hat Aus-Grad 2.

(2) Es gibt eine Zahl $h \in \mathbb{N}$, so dass für jedes Blatt $v \in V$ gilt: Der Weg von der Wurzel zum Blatt v hat die Länge h.

Beispiel 4.55. Der Graph G_A aus Beispiel 4.49 ist ein Binärbaum, aber kein vollständiger Binärbaum. Der Graph G_B aus Beispiel 4.49 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:

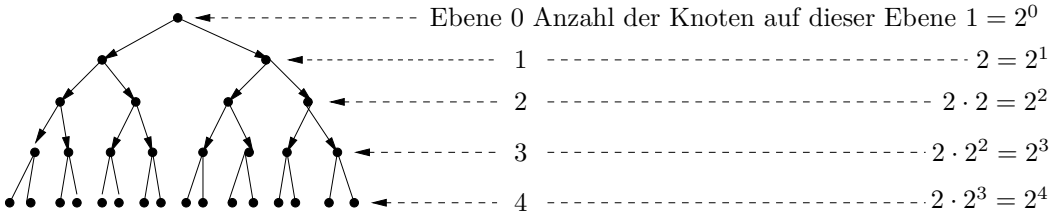
Satz 4.56. *Sei $h \in \mathbb{N}$.*

*(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**.*

Beweis:

(a) **Anschaulich:**



$\rightsquigarrow$ vollständiger Binärbaum der Höhe h hat 2^h Blätter und

$$2^0 + 2^1 + 2^2 + \cdots + 2^h \overset{\text{Satz 2.48}}{=} 2^{h+1} - 1$$

Knoten.

Formaler Beweis: Per Induktion nach h:

Induktionsanfang: $h = 0$:
Für jeden gerichteten 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.

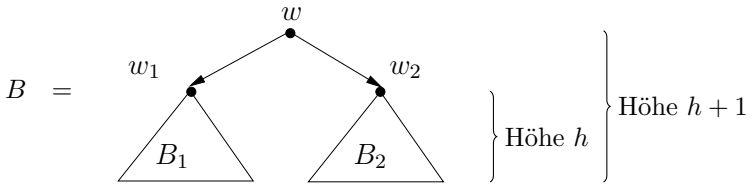
Induktionsschritt: $h \to h + 1$:
Sei $h \in \mathbb{N}$ beliebig.

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 $h + 1 \geq 1$ 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.

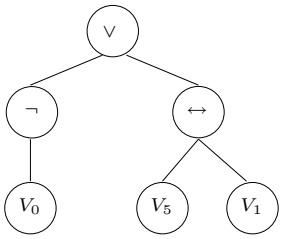
□

4.2.3 Modellierungsbeispiele

Gerichtete Bäume mit zahlreichen Knoten- und/oder Kantenmarkierungen können auf vielfältige Arten zur Modellierung genutzt werden:

Beispiel 4.57. In Kapitel 3 (Seite 60) haben wir Bäume bereits genutzt, um die Struktur einer aussagenlogischen Formel übersichtlich darzustellen (der entsprechende Baum heißt **Syntaxbaum** der Formel).

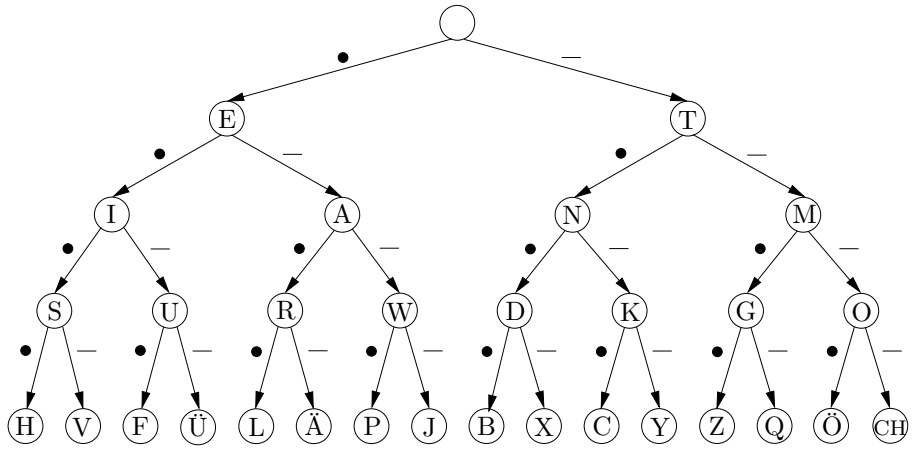
Beispiel: Syntaxbaum der Formel $\big(\neg V_0 \vee (V_5 \leftrightarrow V_1)\big)$:



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 oder zur Beschreibung der Struktur von Computerprogrammen oder umgangssprachlichen Texten.

Beispiel 4.58. Folgen von Entscheidungen können in vielen Zusammenhängen durch gerichtete markierte Bäume modelliert werden – so genannte *Entscheidungsbäume*. Durch einen solchen Entscheidungsbaum erhält man beispielsweise eine kompakte Darstellung des **Morse-Codes:**

M =

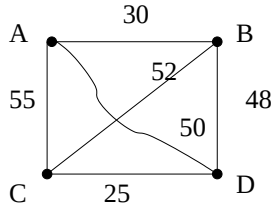


Im Morse-Code wird jeder Buchstabe durch eine Folge von kurzen und langen Signalen (• $\widehat{=}$ kurz, — $\widehat{=}$ lang) repräsentiert. Ein eingehende Meldung aus kurzen und langen Signalen wird entschlüsselt, indem man an der Wurzel des Baums M beginnt und bei einem kurzen Signal nach links, bei einem langen nach rechts weitergeht. Eine längere Pause zeigt an, dass ein Buchstabe vollständig übermittelt ist.

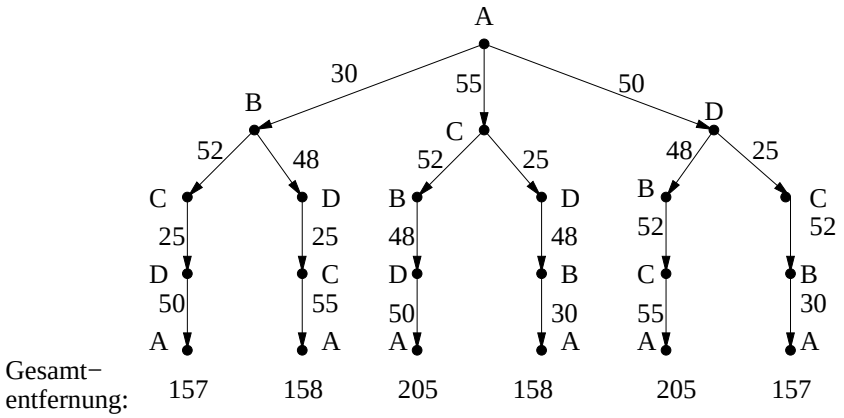
In jedem Entscheidungsbaum modellieren die Knoten einen Zwischenstand bei der Entscheidungsfindung. Sie können entsprechend markiert sein, z.B. mit dem codierten Buchstaben des Morse-Codes. Die Kanten, die von einem Knoten ausgehen, modellieren die Alternativen, aus denen in dem durch den Knoten repräsentierten „Zustand“ eine ausgewählt werden kann – im Morse-Code ist das jeweils ein kurzes oder ein langes Signal, das als Kantenmarkierung angegeben wird.

Beispiel 4.59. Markierte Bäume können auch genutzt werden, um den Lösungsraum kombinatorischer Probleme darzustellen.

Beispiel: Ein Handlungsreisender soll einen möglichst kurzen Rundweg finden, 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:



Alle möglichen in Stadt A startenden Rundwege werden durch den folgenden Baum dargestellt:



Jeder Weg von der Wurzel zu einem Blatt repräsentiert dabei einen Rundweg, auf dem jede der Städte 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 Rundwege für unseren Handlungsreisenden sind also

$$(A, B, C, D, A) \quad \text{und} \quad (A, D, C, B, A).$$

Bemerkung 4.60. Nach dem gleichen Schema kann man auch Zugfolgen in Spielen modellieren: Jeder Knoten des Entscheidungsbaums modelliert einen Spielzustand. Die von dort ausgehenden Kanten geben an, welche Möglichkeiten für den nächsten Zug bestehen. Solche Darstellungen werden z.B. in Schachprogrammen verwendet, um die Folgen der anstehenden Entscheidung zu analysieren und zu bewerten.

Beachte: Bei der Modellierung von Spielabläufen können manche "Spielzustände" (z.B. Konfigurationen eines Schachbretts) auf unterschiedlichen Wegen (d.h. Spielverläufen) erreicht werden, und trotzdem "im Sinne des Spiels" den selben Zustand beschreiben. In solchen Fällen könnte man im Entscheidungsbaum die zugehörigen Knoten zu einem einzigen Knoten zusammenfassen. Damit geht dann allerdings die Baum-Eigenschaft verloren, und es entsteht ein allgemeiner gerichteter Graph, der auch Kreise enthalten kann. Ein Kreis entspricht dann der Situation, dass eine Folge von Spielzügen in einen Zustand zurückführt, der früher schon einmal durchlaufen wurde.

4.3 Einige spezielle Arten von Graphen

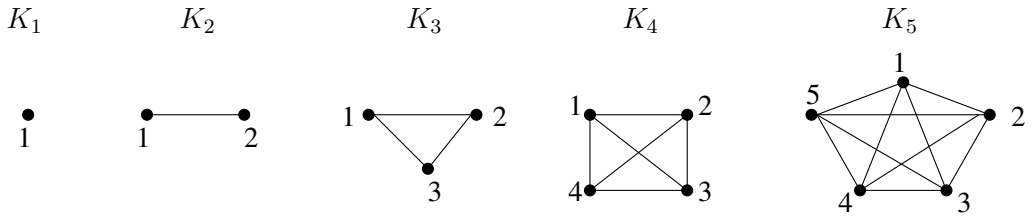
In diesem Abschnitt werden einige spezielle Arten von Graphen vorgestellt, die Ihnen im Laufe der nächsten Semester immer wieder begegnen werden.

4.3.1 Spezielle ungerichtete Graphen

Definition 4.61 (Der vollständige Graph K_n)**.** Sei $n \in \mathbb{N}_{>0}$. Der **vollständige ungerichtete Graph** K_n hat Knotenmenge $\{1, \ldots, n\}$ und Kantenmenge $\big\{\{i, j\} : i, j \in \{1, \ldots, n\},\, i \neq j\big\}$.

vollständiger ungerichteter Graph

Beispiele.



Beobachtung. Der vollständige Graph K_n hat n Knoten und $\frac{n(n-1)}{2}$ Kanten.

vollständiger ungerichteter bipartiter Graph

Definition 4.62 (Der vollständige bipartite Graph $K_{m,n}$)**.** Seien $m, n \in \mathbb{N}_{>0}$. Der **vollständige ungerichtete bipartite Graph** $K_{m,n}$ hat Knotenmenge

$$\big\{(1,i) : i \in \{1,\dots,m\}\big\} \ \dot\cup\ \big\{(2,j) : j \in \{1,\dots,n\}\big\}$$

und Kantenmenge

$$\big\{\{(1,i),(2,j)\} \ : \ i \in \{1,\dots,m\},\ j \in \{1,\dots,n\}\big\}.$$

Beispiele. Siehe Abbildung 4.1.

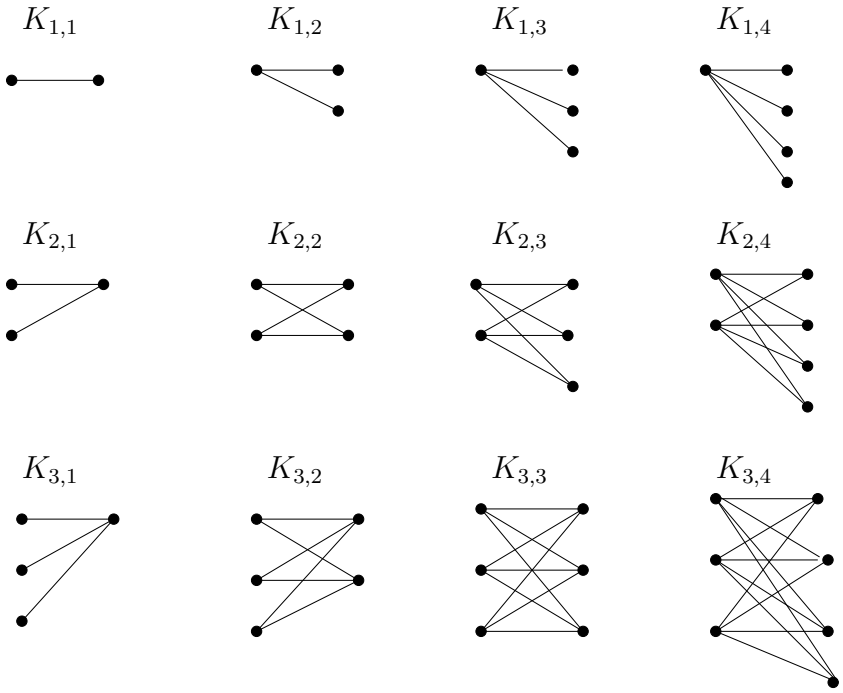


Abbildung 4.1: einige vollständige bipartite Graphen

Beobachtung. Der Graph $K_{m,n}$ hat $m+n$ Knoten und $m \cdot n$ Kanten.

Notation 4.63. Ein ungerichteter Graph G mit endlicher, 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 $G \cong K_{m,n}$.

4.3.2 Spezielle gerichtete Graphen

Gemäß Definition 4.6 ("gerichteter Graph") und Definition 2.27(c) ("k-stellige Relation") kann jeder gerichtete Graph $G = (V, E)$ als eine 2-stellige Relation über V aufgefasst werden, 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 M als gerichteten Graph auffassen, dessen Knotenmenge M und dessen Kantenmenge R ist. Gerichtete Graphen sind also dasselbe wie 2-stellige Relationen über einer Menge V.

Von besonderem Interesse sind 2-stellige Relationen, die eine oder mehrere der folgenden Eigenschaften besitzen:

Definition 4.64. Sei E eine 2-stellige Relation über einer Menge V (d.h. $E \subseteq V \times V$, 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 v w gibt es auch eine "Rückwärtskante" v w)

(c) E heißt **antisymmetrisch**, falls f.a. $v, w \in V$ gilt: antisymmetrisch

$$\text{Wenn } (v, w) \in E \text{ und } (w, v) \in E, \text{ dann } v = w.$$

(d.h: Ist $v \neq w$, so gibt es in E allenfalls eine der beiden Kanten v w und v w)

(d) E heißt **konnex**, falls f.a. $v, w \in V$ gilt: konnex

$$(v, w) \in E \text{ oder } (w, v) \in E.$$

(d.h. mindestens eine der beiden Kanten v w und v w liegt in E)

(e) E heißt **transitiv**, falls f.a. $v, w, u \in V$ gilt: transitiv

$$\text{Ist } (v, w) \in E \text{ und } (w, u) \in E, \text{ so auch } (v, u) \in E.$$

(Skizze: v E w E u ⇒ v E w E u E)

Äquivalenzrelationen

Definition 4.65. Eine **Äquivalenzrelation** ist eine 2-stellige Relation, die **reflexiv**, **symmetrisch** und **transitiv** ist. Äquivalenzrelation

Beispiel 4.66. Beispiele für Äquivalenzrelationen:

(a) **Gleichheit:** Für jede Menge M ist

$$E := \{(m, m) : m \in M\}$$

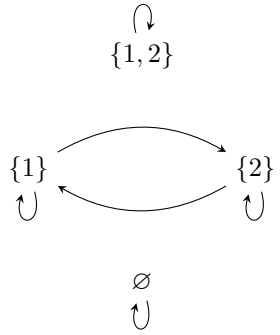
eine Äquivalenzrelation. Die Aussage "$(x, y) \in E$" entspricht gerade der Aussage "$x = y$".

(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

$$E := \{(\varphi, \psi) : \varphi \in \mathrm{AL},\, \varphi \equiv \psi\}$$

ist eine Äquivalenzrelation über der Menge AL aller aussagenlogischen Formeln.

Ordnungsrelationen

Definition 4.67 (Ordnungen)**.** Sei E eine 2-stellige Relation über einer Menge V.

Präordnung (a) E heißt **Präordnung**, falls E **reflexiv und transitiv** ist.

partielle Ordnung (b) E heißt **partielle Ordnung**, falls E **reflexiv, transitiv und antisymmetrisch** ist.

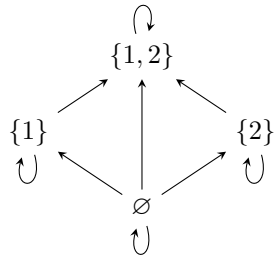
lineare Ordnung
totale Ordnung (c) E heißt **lineare Ordnung** oder **totale Ordnung**, falls E **reflexiv, transitiv, antisymmetrisch und konnex** ist.

Beispiel 4.68.

(a) $\leq$ ist eine **lineare Ordnung** auf $\mathbb{N}$ (und $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$). Ebenso ist $\geq$ eine lineare Ordnung auf $\mathbb{N}$ (und $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$).

(b) Für jede Menge M sind $\subseteq$ und $\supseteq$ **partielle Ordnungen** auf der Potenzmenge $\mathcal{P}(M)$.

Skizze für $M = \{1, 2\}$:



(c) Die "Folgerungsrelation"

$$E \;:=\; \{(\varphi, \psi) : \varphi \in \mathrm{AL},\, \psi \in \mathrm{AL},\, \varphi \models \psi\}$$

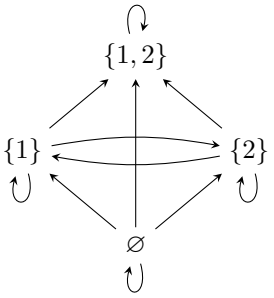
ist eine **Präordnung auf AL**.

(d) Für jede endliche Menge M ist

$$E \;:=\; \{(A, B) : A \subseteq M,\, B \subseteq M,\, |A| \leq |B|\}$$

eine **Präordnung** auf $\mathcal{P}(M)$.

Skizze für $M = \{1, 2\}$:



Die reflexive transitive Hülle einer Relation

reflexive transitive Hülle
reflexiver transitiver Abschluss

Definition 4.69. Sei $G = (V, E)$ ein gerichteter Graph. Die **reflexive transitive Hülle** (bzw. der **reflexive transitive Abschluss**) von E auf V ist die rekursiv wie folgt definierte Relation $E^* \subseteq V \times V$:

Basisregeln:

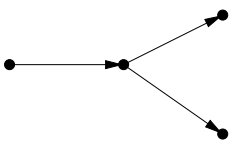
- F.a. $v \in V$ ist $(v, v) \in E^*$.
- F.a. $(v, w) \in E$ ist $(v, w) \in E^*$

Rekursive Regel:

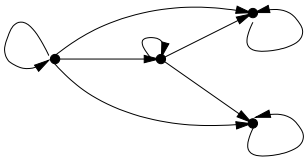
- Sind $(v, w) \in E^*$ und $(w, u) \in E^*$, so ist auch $(v, u) \in E^*$.

Beispiel.

$G = (V, E)$:=



$G^* = (V, E^*)$:



Beobachtung 4.70. Sei $G = (V, E)$ ein gerichteter Graph und seien $v, w \in V$. Dann sind die beiden folgenden Aussagen äquivalent:

(a) $(v, w) \in E^*$, wobei E^* die reflexive transitive Hülle von E auf V ist.

(b) Es gibt in G einen Weg von v nach w.

Beweis: Übung. □

4.4 Übungsaufgaben zu Kapitel 4

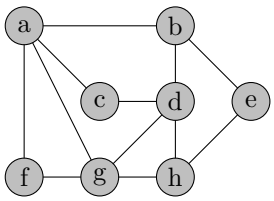
Aufgabe 4.1. Betrachten Sie den ungerichteten Graphen G auf der rechten Seite.

(a) Geben Sie die Knotenmenge V und die Kantenmenge E des Graphen G an. Repräsentieren Sie G außerdem durch eine Adjazenzmatrix und eine Adjazenzliste.

(b) Geben Sie einen Euler-Weg in G an. Besitzt G auch einen Euler-Kreis?

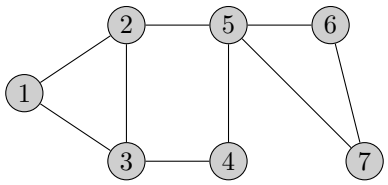
(c) Geben Sie einen Hamilton-Kreis in G an.

(d) Geben Sie einen Spannbaum von G an, den man so wurzeln kann, dass er die Höhe 2 hat. Kennzeichnen Sie die Wurzel in Ihrer Lösung.

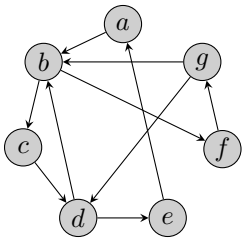


Aufgabe 4.2. Es seien die folgenden beiden Graphen G_1 und G_2 gegeben:

G_1:



G_2:



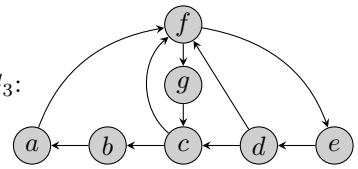
(a) Geben Sie für jeden der beiden Graphen G_1 und G_2 die Knotenmenge und die Kantenmenge an. Repräsentieren Sie außerdem jeden der beiden Graphen durch eine Adjazenzmatrix und eine Adjazenzliste.

(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.

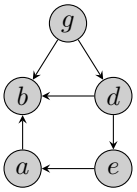
(c) Ist G_1 zusammenhängend? Ist G_2 stark zusammenhängend? Ist G_2 azyklisch?

(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.

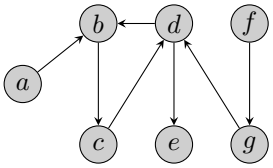
G_3:



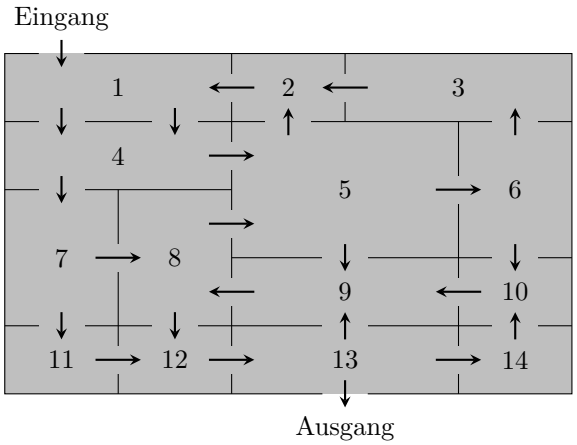
G_4:



G_5:



Aufgabe 4.3. 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 4.4. 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.

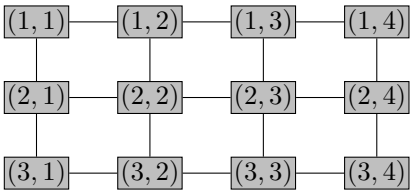
(b) Untersuchen Sie die folgenden Graphen G_1, G_2 und G_3 auf Ihre Tauglichkeit bezüglich der Eigenschaften (1), (2) bzw. (3):

- $G_1 = (V_1, E_1)$ mit $V_1 = \{1, 2, \ldots, n\}$ und $E_1 = \{\{1, i\} : 2 \leq i \leq n\}$
- $G_2 = (V_2, E_2)$ mit $V_2 = V_1$ und $E_2 = \{\{i, i+1\} : 1 \leq i < n\}$
- $G_3 = (V_3, E_3)$ mit $V_3 = V_1$ und $E_3 = E_2 \cup \{\{n, 1\}\}$

Aufgabe 4.5. 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

$$\begin{aligned} V_{m\times n} &:= \big\{\, (i,j) \,:\, 1 \le i \le m,\, 1 \le j \le n \,\big\}, \\ E_{m\times n} &:= \big\{\, \{(i,j),(i,j+1)\} \,:\, 1 \le i \le m,\, 1 \le j < n \,\big\} \cup \\ &\quad\ \big\{\, \{(i,j),(i+1,j)\} \,:\, 1 \le i < m,\, 1 \le j \le n \,\big\}. \end{aligned}$$

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.

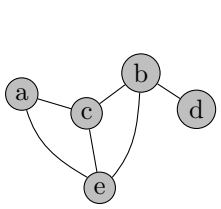
(b) Geben Sie ein Matching maximaler Größe in $G_{3\times 4}$ an.

(c) Geben Sie einen Hamilton-Kreis in $G_{3\times 4}$ an.

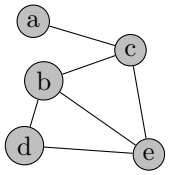
(d) Für welche $m, n \in \mathbb{N}_{>0}$ besitzt $G_{m\times n}$ einen Hamilton-Kreis, für welche nicht?

Hinweis: Verallgemeinern Sie die Argumentation auf Seite 136 (oben) im Buch „Modellierung – Grundlagen und formale Methoden“ von Uwe Kastens und Hans Kleine Büning. Das Buch befindet sich unter Anderem im Semesterapparat zur Vorlesung in der Informatik-Bibliothek.

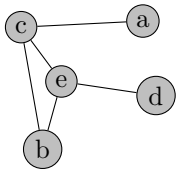
Aufgabe 4.6. Es seien die folgenden ungerichteten Graphen G_1, G_2, G_3 und G_4 gegeben:



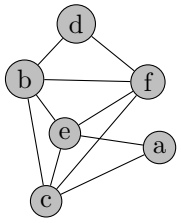
G_1



G_2



G_3



G_4

(a) Überprüfen Sie für alle $i, j \in \{1, 2, 3, 4\}$ mit $i \neq j$, ob Folgendes gilt:

- $G_i = G_j$
- G_i ist ein Teilgraph von G_j
- G_i ist ein induzierter Teilgraph von G_j

(b) Überprüfen Sie, welche der Graphen isomorph zueinander sind. Falls zwei Graphen G_i und G_j isomorph sind, so geben Sie einen Isomorphismus von G_i nach G_j an. Falls hingegen G_i und G_j nicht isomorph sind, so begründen Sie dies.

(c) Welche der Graphen kann man nachzeichnen, ohne den Stift abzusetzen oder eine Kante doppelt zu ziehen?

Aufgabe 4.7. König Artus will für die Tafelrunde eine Sitzordnung für sich und neun seiner Ritter festlegen, bei der er und die neun Ritter im Kreis an einem runden Tisch sitzen. Das wäre nicht schwer, gäbe es nicht diese Rivalitäten und Eifersüchteleien zwischen den Rittern. König Artus möchte, dass Lancelot zu seiner Rechten und Kay zu seiner Linken sitzt. Erec weigert sich, neben jemand anderem als Lyonel oder Tristan zu sitzen. Galahad will weder neben Tristan noch neben Lancelot oder Lyonel sitzen. Parzival lehnt es ab, neben Gawain, Lancelot oder Lyonel zu sitzen. Gaheris möchte auf keinen Fall neben Gawain, Lancelot oder Kay sitzen. Tristan weigert sich, neben Lancelot, Parzival oder Kay zu sitzen. Gawain würde sich neben jeden anderen setzen, aber nicht neben Galahad oder Kay. Und Lyonel ist dagegen, neben Gawain zu sitzen.

(a) Stellen Sie den Konfliktgraphen auf.

(b) Verwenden Sie den Konfliktgraphen aus (a), um eine Tischordnung aufzustellen, die von allen akzeptiert wird. Zeichnen Sie den entsprechenden Graph und die Sitzordnung.

Aufgabe 4.8. Auf dem Weihnachtsmarkt von Großdorf sollen insgesamt 8 Stände rund um den Marktplatz arrangiert werden. Die 8 Stände setzen sich folgendermaßen zusammen:

- Ein Stand, in dem die traditionelle Weihnachtskrippe aufgebaut ist.
- Zwei Stände, an denen Kunsthandwerk verkauft wird: einer der beiden Stände ist die Töpferei, der andere bietet Holzschmuck aus dem Erzgebirge an.
- Zwei Glühweinstände; einer davon wird von Herrn Max, der andere von Frau Peters betrieben.
- Drei Essensstände; einer davon verkauft Crêpes, der andere Waffeln und der dritte Steaks vom Holzkohlegrill.

Bei der Platzierung der 8 Stände um den Marktplatz ist folgendes zu beachten: Neben der Weihnachtskrippe darf keiner der Glühweinstände platziert werden. Essensstände dürfen nicht nebeneinander stehen, die beiden Glühweinstände dürfen nicht nebeneinander stehen, und die beiden Kunsthandwerkstände dürfen nicht nebeneinander stehen. Aus Sicherheitsgründen darf der Holzkohlegrill weder neben der Weihnachtskrippe noch neben dem Stand mit dem Holzschmuck aus dem Erzgebirge stehen. Herr Max ist mit den Besitzern des Holzkohlegrills und der Töpferei befreundet und möchte daher unbedingt die beiden als Nachbarn haben. Außerdem ist zu beachten, dass sich der Betreiber des Waffelstands weder mit Frau Peters noch mit dem Besitzer der Töpferei verträgt und daher auf keinen Fall neben einem der beiden platziert werden will.

(a) Stellen Sie den Konfliktgraphen und das Komplement des Konfliktgraphen auf.

(b) Gibt es im Komplement des Konfliktgraphen einen Hamiltonkreis? Falls ja, dann geben Sie einen solchen Hamiltonkreis an. Falls nein, dann begründen Sie, warum es keinen gibt.

(c) Geben Sie eine Platzierung der 8 Stände rund um den Marktplatz an, mit der alle zufrieden sind.

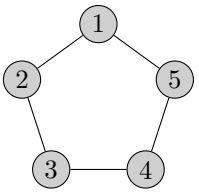
Aufgabe 4.9. Sei $G = (V, E)$ ein ungerichteter, zusammenhängender Graph, dessen Knotenmenge endlich ist. Beweisen Sie, dass G genau dann einen Euler-Weg besitzt, der kein Euler-Kreis ist, wenn es in G genau zwei Knoten mit ungeradem Grad gibt.

Aufgabe 4.10. 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.

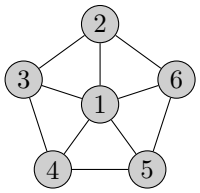
Ein Spieler hat eine *Gewinnstrategie* in dem Spiel genau dann, wenn der Spieler das Spiel, unabhängig davon wie der andere Spieler spielt, gewinnen kann.

(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.

G_1:



G_2:



(b) Beweisen Sie, dass die beiden folgenden Aussagen äquivalent sind:

(i) G besitzt ein Matching M, so dass jeder Knoten aus V zu mindestens einer Kante aus M inzident ist.

(ii) Spieler B hat eine Gewinnstrategie in dem oben beschriebenen Spiel auf G.

Aufgabe 4.11. 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 4.12. Beweisen Sie, dass jeder ungerichtete, zusammenhängende Graph $G = (V, E)$, dessen Knotenmenge V endlich ist, einen Spannbaum besitzt.

Hinweis: Gehen Sie per Induktion nach $n := |E|$ vor.

Aufgabe 4.13. 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 4.14. Sei $\mathcal{G}$ die Menge der ungerichteten Graphen $G = (V, E)$ mit $V \subseteq \mathbb{N}$.

(a) Unter (i) bis (iii) sind zweistellige Relationen über $\mathcal{G}$ gegeben. Überprüfen Sie für jede dieser Relationen, ob sie reflexiv, symmetrisch, antisymmetrisch, konnex bzw. transitiv ist.

(i) $\{ (G, G') \in \mathcal{G}^2 : G \cong G' \}$

(ii) $\{ (G, G') \in \mathcal{G}^2 : G$ hat höchstens so viele Knoten wie $G' \}$

(iii) $\{ (G, G') \in \mathcal{G}^2 : G'$ besitzt einen Teilgraph G'' mit $G'' \cong G \}$

(b) Zeigen Sie, dass die zweistellige Relation

$$R := \{ (G, G') \in \mathcal{G}^2 : G \text{ ist ein induzierter Teilgraph von } G' \}$$

eine partielle Ordnung, aber keine lineare Ordnung, auf $\mathcal{G}$ ist.

5 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.

5.1 Motivation zur Logik erster Stufe

5.1.1 Grenzen der Aussagenlogik

Beispiel 5.1 (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 5.2 (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."

5.1.2 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 festgelegt, 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 5.1) oder den natürlichen Zahlen und deren Addition und Multiplikation (Beispiel 5.2)) 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.

5.2 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 $G = (V, E)$
- Bäume $B = (V, E)$
- 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

usw. Die im Folgenden definierten **Signaturen** legen den "Typ" (bzw. das "Format") der entsprechenden Strukturen fest.

Definition 5.3. Eine **Signatur** (bzw. ein **Vokabular** bzw. eine **Symbolmenge**; englisch: signature, vocabulary) ist eine Menge σ (in Worten: sigma) 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 5.4.

- In diesem Kapitel bezeichnet der griechische Buchstabe σ 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

$$\begin{aligned} \dot{\leq} \quad & \text{(2-stelliges Relationssymbol)}, \\ \dot{+}, \dot{\times} \quad & \text{(2-stellige Funktionssymbole)}, \\ \dot{0}, \dot{1} \quad & \text{(Konstantensymbole)}. \end{aligned}$$

Struktur
σ-Struktur

Definition 5.5. 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 5.6.

- 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 $\mathfrak{A} = (A, (\dot{S}^{\mathfrak{A}})_{\dot{S}\in\sigma})$, oder falls $\sigma = \{\dot{R}_1, \ldots, \dot{R}_k, \dot{f}_1, \ldots, \dot{f}_l, \dot{c}_1, \ldots, \dot{c}_m\}$ ist,

  $$\mathfrak{A} \;=\; (A, \dot{R}_1^{\mathfrak{A}}, \ldots, \dot{R}_k^{\mathfrak{A}}, \dot{f}_1^{\mathfrak{A}}, \ldots, \dot{f}_l^{\mathfrak{A}}, \dot{c}_1^{\mathfrak{A}}, \ldots, \dot{c}_m^{\mathfrak{A}}).$$

Beispiel 5.7 (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 5.8 (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 $\mathfrak{A} = (A, E^{\mathfrak{A}})$ mit Universum $A := V$ und Relation $\dot{E}^{\mathfrak{A}} := E$ auffassen.

Beispiel 5.9 (Ordnungen)**.** Sei $\sigma_{\mathrm{Ord}} := \{\dot{\leq}\}$, wobei $\dot{\leq}$ ein 2-stelliges Relationssymbol ist. Jeder Präordnung, partiellen Ordnung oder linearen Ordnung $\leq$ auf einer Menge A entspricht eine σ_{Ord}-Struktur

$$\mathfrak{A} \;=\; (A, \dot{\leq}^{\mathfrak{A}})$$

mit $\dot{\leq}^{\mathfrak{A}} := \leq$.

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 4.25) wird dies durch folgende Definition präzisiert:

Definition 5.10. 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 $r := \mathrm{ar}(\dot{R})$ 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 $r := \mathrm{ar}(\dot{f})$ 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 5.11.

(a) Ist $A = \{1, 2, 3, 4\}, B = \{6, 7, 8, 9\}$, und sind $\dot{\leq}^{\mathfrak{A}}$ und $\dot{\leq}^{\mathfrak{B}}$ die natürlichen linearen Ordnungen auf A und B, so sind die beiden σ_{Ord}-Strukturen $\mathfrak{A} = (A, \dot{\leq}^{\mathfrak{A}})$ und $\mathfrak{B} = (B, \dot{\leq}^{\mathfrak{B}})$ isomorph.
Skizze:

Allgemein gilt: Sind A und B endliche Mengen mit $|A| = |B|$ und sind $\dot{\leq}^{\mathfrak{A}}$ und $\dot{\leq}^{\mathfrak{B}}$ lineare Ordnungen auf $\mathfrak{A} = (A, \dot{\leq}^{\mathfrak{A}})$ und $\mathfrak{B} = (B, \dot{\leq}^{\mathfrak{B}})$, so ist $\mathfrak{A} \cong \mathfrak{B}$, und die Abbildung π, die das (bzgl. $\dot{\leq}^{\mathfrak{A}}$) kleinste Element in A auf das (bzgl. $\dot{\leq}^{\mathfrak{B}}$) kleinste Element von $\mathfrak{B}$ abbildet und, allgemein, für jedes $i \in \{1, \ldots, |A|\}$ das i-kleinste Element in A (bzgl. $\dot{\leq}^{\mathfrak{A}}$) auf das i-kleinste Element in B (bzgl. $\dot{\leq}^{\mathfrak{B}}$) abbildet, ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$.

(b) Sind $\dot{\leq}^{\mathcal{N}}$ und $\dot{\leq}^{\mathcal{Z}}$ die natürlichen linearen Ordnungen auf $\mathbb{N}$ und $\mathbb{Z}$, so sind die σ_{Ord}-Strukturen $\mathcal{N} := (\mathbb{N}, \dot{\leq}^{\mathcal{N}})$ und $\mathcal{Z} := (\mathbb{Z}, \dot{\leq}^{\mathcal{Z}})$ **nicht isomorph** (kurz: $\mathcal{N} \ncong \mathcal{Z}$).
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.

5.3 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").
  - Es gibt Symbole für Elemente aus der Signatur σ.

Definition 5.12 (Variablen und Alphabet der Logik erster Stufe)**.**

(a) Eine **Individuenvariable** (kurz: **Variable**) hat die Form v_i, für $i \in \mathbb{N}$. Die Menge aller Variablen bezeichnen wir mit VAR, d.h. $\text{VAR} = \{v_i : i \in \mathbb{N}\}$.

Individuenvariable
Variable

(b) Sei σ eine Signatur. Das **Alphabet** A_σ der **Logik erster Stufe über** σ besteht aus:

Alphabet A_σ

- den Variablen in VAR
- den Symbolen in σ
- den Quantoren $\exists$ (Existenzquantor) und $\forall$ (Allquantor)
- dem Gleichheitssymbol $\doteq$
- den Junktoren $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- den Klammern (,) und dem Komma ,

D.h.

$$A_\sigma \;=\; \text{VAR} \;\cup\; \sigma \;\cup\; \{\exists, \forall\} \;\cup\; \{\doteq\} \;\cup\; \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\} \;\cup\; \{,\}.$$

Definition 5.13 (Terme der Logik erster Stufe)**.** Sei σ eine Signatur. Die Menge T_σ der **σ-Terme** ist die folgendermaßen rekursiv definierte Teilmenge von A_σ^*:

σ-Terme

Basisregeln:

- Für jedes Konstantensymbol $\dot{c} \in \sigma$ ist $\dot{c} \in T_\sigma$.
- Für jede Variable $x \in \text{VAR}$ ist $x \in T_\sigma$.

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 5.14. Sei $\sigma = \{\dot{f}, \dot{c}\}$ die Signatur aus Beispiel 5.11(c), die aus einem 2-stelligen Funktionssymbol $\dot{f}$ und einem Konstantensymbol $\dot{c}$ besteht.

Folgende Worte aus A_σ^* 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:

$$\mathbf{0}, \quad \dot{f}(\mathbf{0}, \dot{c}), \quad \dot{f}(v_0, \dot{c}, v_1), \quad f^{\mathfrak{A}}(2, 3).$$

Definition 5.15 (Formeln der Logik erster Stufe)**.** Sei σ eine Signatur. Die Menge $\text{FO}[\sigma]$ 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^*:

FO[σ]-Formeln

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 $\varphi \in \mathrm{FO}[\sigma]$, so auch $\neg\varphi \in \mathrm{FO}[\sigma]$.
- Ist $\varphi \in \mathrm{FO}[\sigma]$ und $\psi \in \mathrm{FO}[\sigma]$, so ist auch
  - $(\varphi \wedge \psi) \in \mathrm{FO}[\sigma]$
  - $(\varphi \vee \psi) \in \mathrm{FO}[\sigma]$
  - $(\varphi \to \psi) \in \mathrm{FO}[\sigma]$
  - $(\varphi \leftrightarrow \psi) \in \mathrm{FO}[\sigma]$.
- Ist $\varphi \in \mathrm{FO}[\sigma]$ und ist $x \in \textsc{Var}$, so ist auch
  - $\exists x\, \varphi \in \mathrm{FO}[\sigma]$
  - $\forall x\, \varphi \in \mathrm{FO}[\sigma]$.

Beispiel 5.16.

(a) Sei $\sigma = \{\dot{f}, \dot{c}\}$ die Signatur aus Beispiel 5.11(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 $\sigma_{\mathrm{Graph}} = \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol besteht. Folgendes ist eine FO[σ_{Graph}]-Formel:

$$\forall v_0 \forall v_1 \Big(\big(\dot{E}(v_0, v_1) \wedge \dot{E}(v_1, v_0)\big) \to v_0 \doteq v_1 \Big).$$

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 $(a_1, a_0) \in \dot{E}^{\mathfrak{A}}$, so 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 (vgl. Definition 4.64). 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 5.17.

- 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 Funktionssymbole wie $\dot{+}, \dot{\times} \in \sigma_{\mathrm{Ar}}$ oder $\dot{\leq} \in \sigma_{\mathrm{Ord}}$ verwenden wir **Infix- statt Präfixschreibweise** und setzen Klammern dabei auf natürliche Weise, um die eindeutige Lesbarkeit zu gewährleisten.

  Beispiel: An Stelle des (formal korrekten) Terms $\dot{\times}(\dot{+}(x, y), z)$ schreiben wir $(x \dot{+} y) \dot{\times} z$. An Stelle der (formal korrekten) atomaren Formel $\dot{\leq}(x, y)$ schreiben wir $x \dot{\leq} y$.

Wir wissen nun, welche Zeichenketten (über dem Alphabet A_σ) **FO**$[\sigma]$**-Formeln** genannt werden (Syntax). Bevor wir die formale Definition der **Semantik** angeben, betrachten wir zunächst einige Beispiele, um ein "intuitives Verständnis" der Semantik zu bekommen.

5.4 Beispiele zur Semantik der Logik erster Stufe

Beispiel 5.18 (gerichtete Graphen)**.** Sei $\sigma_{\mathrm{Graph}} = \{\dot{E}\}$, wobei $\dot{E}$ ein 2-stelliges Relationssymbol ist.

(a) Die FO$[\sigma_{\mathrm{Graph}}]$-Formel

$$\varphi \;:=\; \forall x \forall y \big(\dot{E}(x, y) \to \dot{E}(y, x)\big)$$

besagt:

> "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 $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ gilt daher:

$$\mathfrak{A} \text{ erfüllt } \varphi \iff \dot{E}^{\mathfrak{A}} \text{ ist symmetrisch.}$$

Umgangssprachlich sagen wir auch: "Die Formel φ sagt **in einem Graphen $\mathfrak{A}$ aus, dass** dessen Kantenrelation symmetrisch ist."

(b) Die folgende FO$[\sigma_{\mathrm{Graph}}]$-Formel drückt aus, dass es von Knoten x zu Knoten y einen Weg der Länge 3 gibt:

$$\varphi(x, y) \;:=\; \exists z_1 \exists z_2 \Big(\big(\dot{E}(x, z_1) \wedge \dot{E}(z_1, z_2)\big) \wedge \dot{E}(z_2, y) \Big).$$

(c) Die FO$[\sigma_{\mathrm{Graph}}]$-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 5.19 (Verwandtschaftsbeziehungen)**.** Um Verwandtschaftsbeziehungen zu modellieren, können wir die Symbolmenge σ benutzen, die aus den folgenden Symbolen besteht:

- 1-stellige Funktionen $\dot{Vater}$, $\dot{Mutter}$

  (Bedeutung: $x \doteq \dot{Vater}(y)$ besagt "x ist der Vater von y".)

- 2-stellige Relationen $\dot{Geschwister}$, $\dot{Vorfahr}$

  (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":

$$\forall x \forall y \Big(\big(\dot{Vater}(x) \doteq \dot{Vater}(y) \wedge \dot{Mutter}(x) \doteq \dot{Mutter}(y) \big) \rightarrow \dot{Geschwister}(x, y) \Big).$$

- "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:

$$\psi(x) \;:=\; \exists y_1 \exists y_2 \Big(\Big(\big(x \doteq \dot{Vater}(y_1) \wedge x \doteq \dot{Vater}(y_2) \big) \wedge \neg y_1 \doteq y_2 \Big) \wedge \forall z \big(x \doteq \dot{Vater}(z) \rightarrow (z \doteq y_1 \vee z \doteq y_2) \big) \Big).$$

5.5 Semantik der Logik erster Stufe

Um die formale Definition der Semantik der Logik erster Stufe angeben zu können, benötigen wir noch folgende Begriffe:

Notation 5.20.

Teilformel

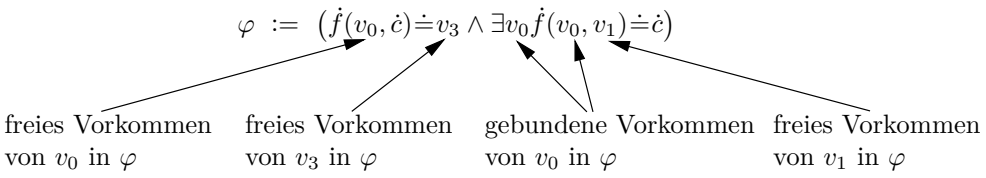
1. Eine Formel ψ ist **Teilformel** einer Formel φ, wenn ψ als Teil-Wort in φ vorkommt.

   Beispiel: $\psi := \dot{f}(v_0, v_1) \doteq \dot{c}$ ist Teilformel der Formel $\exists v_0 \, \dot{f}(v_0, v_1) \doteq \dot{c}$.

gebunden
frei

2. 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**.

Beispiel:



3. Die Menge $\mathrm{frei}(\varphi)$ aller **freien Variablen** einer FO[σ]-Formel φ besteht aus allen Variablen, die mindestens einmal frei in φ vorkommen. — $\mathrm{frei}(\varphi)$, freien Variablen

   Beispiel:
   - $\mathrm{frei}\big(\dot f(v_0,\dot c)\doteq v_3\big) = \{v_0, v_3\}$
   - $\mathrm{frei}\big(\exists v_0\, \dot f(v_0,v_1)\doteq \dot c\big) = \{v_1\}$
   - $\mathrm{frei}\big(\dot f(v_0,\dot c)\doteq v_3 \wedge \exists v_0\, \dot f(v_0,v_1)\doteq \dot c\big) = \{v_0, v_3, v_1\}$

4. Eine FO[σ]-Formel φ heißt **Satz** (genauer: FO[σ]-Satz), falls sie keine freien Variablen besitzt, d.h. falls $\mathrm{frei}(\varphi) = \varnothing$. — Satz

Definition 5.21 (Belegung und Interpretation)**.**

(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). — Belegung

(b) Eine Belegung β ist eine **Belegung für eine FO[σ]-Formel** φ (bzw. **passend zu** φ), wenn $\mathrm{frei}(\varphi) \subseteq \mathrm{Def}(\beta)$. — Belegung für eine FO[σ]-Formel φ, passend zu φ

(c) Eine **σ-Interpretation** ist ein Paar — σ-Interpretation

$$\mathcal{I} \;=\; (\mathfrak{A}, \beta)$$

bestehend aus einer σ-Struktur $\mathfrak{A}$ und einer Belegung β in $\mathfrak{A}$. $\mathcal{I} = (\mathfrak{A}, \beta)$ 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

Definition 5.22 (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 σ-Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$, so dass $\mathrm{Def}(\beta)$ jede in t vorkommende Variable enthält, einen Wert $[\![t]\!]^{\mathcal{I}} \in A$ zuordnet:

- Für alle $x \in$ 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:

$$[\![\dot f(t_1, \ldots, t_r)]\!]^{\mathcal{I}} \;:=\; \dot f([\![t_1]\!]^{\mathcal{I}}, \ldots, [\![t_r]\!]^{\mathcal{I}}).$$

Beispiel 5.23. 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 5.11(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:

$$\begin{aligned}
[\![t]\!]^{\mathcal{I}} &= [\![\dot{f}\big(v_2, \dot{f}(v_1, \dot{c})\big)]\!]^{\mathcal{I}} \\
&= \dot{f}^{\mathfrak{A}}\big([\![v_2]\!]^{\mathcal{I}}, [\![\dot{f}(v_1, \dot{c})]\!]^{\mathcal{I}}\big) \\
&\overset{\dot{f}^{\mathfrak{A}} = \text{Addition auf } \mathbb{N}}{=} [\![v_2]\!]^{\mathcal{I}} + [\![\dot{f}(v_1, \dot{c})]\!]^{\mathcal{I}} \\
&= \beta(v_2) + \dot{f}^{\mathfrak{A}}([\![v_1]\!]^{\mathcal{I}}, [\![\dot{c}]\!]^{\mathcal{I}}) \\
&= \beta(v_2) + ([\![v_1]\!]^{\mathcal{I}} + [\![\dot{c}]\!]^{\mathcal{I}}) \\
&= \beta(v_2) + (\beta(v_1) + \dot{c}^{\mathfrak{A}}) \\
&\overset{\text{Def. } \beta \text{ und } \dot{c}^{\mathfrak{A}}}{=} 7 + (1 + 0) \\
&= 8.
\end{aligned}$$

Notation 5.24.

- Ist β eine Belegung in einer σ-Struktur $\mathfrak{A}$, ist $x \in \text{VAR}$ und ist $a \in A$, so sei

  $$\beta\frac{a}{x}$$

  die Belegung mit $\text{Def}(\beta\frac{a}{x}) := \text{Def}(\beta) \cup \{x\}$, die für alle $y \in \text{Def}(\beta\frac{a}{x})$ definiert ist durch

  $$\beta\frac{a}{x}(y) \;:=\; \begin{cases} a, & \text{falls } y = x \\ \beta(y) & \text{sonst.} \end{cases}$$

- Ist $\mathcal{I} = (\mathfrak{A}, \beta)$ eine σ-Interpretation, ist $x \in \text{VAR}$ und ist $a \in A$, so sei

  $$\mathcal{I}\frac{a}{x} \;:=\; (\mathfrak{A}, \beta\frac{a}{x}).$$

Wir können nun (endlich) die formale Semantik der Logik erster Stufe festlegen.

Definition 5.25 (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 $r := \text{ar}(\dot{R})$ und für alle σ-Terme $t_1, \ldots, t_r$ in T_σ gilt:

  $$[\![\dot{R}(t_1, \ldots, t_r)]\!]^{\mathcal{I}} \;:=\; \begin{cases} 1, & \text{falls } ([\![t_1]\!]^{\mathcal{I}}, \ldots, [\![t_r]\!]^{\mathcal{I}}) \in \dot{R}^{\mathfrak{A}} \\ 0, & \text{sonst.} \end{cases}$$

Rekursionsschritt:

- Die Semantik der Junktoren $\neg$, $\wedge$, $\vee$, $\rightarrow, \leftrightarrow$ ist wie in der Aussagenlogik definiert – beispielsweise ist für $\varphi \in \mathrm{FO}[\sigma]$ und $\psi \in \mathrm{FO}[\sigma]$:

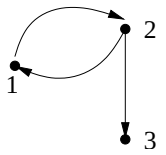
$$[\![(\varphi \wedge \psi)]\!]^{\mathcal{I}} := \begin{cases} 1, & \text{falls } [\![\varphi]\!]^{\mathcal{I}} = 1 \text{ und } [\![\psi]\!]^{\mathcal{I}} = 1 \\ 0, & \text{sonst.} \end{cases}$$

- Ist $\varphi \in \mathrm{FO}[\sigma]$ und ist $x \in \textsc{Var}$, 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 5.26. Sei $\sigma_{\mathrm{Graph}} = \dot{E}$ die Signatur, die aus einem 2-stelligen Relationssymbol $\dot{E}$ besteht. Betrachte die $\mathrm{FO}[\sigma]$-Formel

$$\varphi := \forall x \forall y \big(\dot{E}(x,y) \rightarrow \dot{E}(y,x)\big).$$

Sei $\mathfrak{A}$ 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:

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{I}} = 1 &\iff \text{für alle } a \in A \text{ gilt: } [\![\forall y\big(\dot{E}(x,y) \rightarrow \dot{E}(y,x)\big)]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\
&\iff \text{für alle } a \in A \text{ gilt:} \\
&\qquad \text{für alle } b \in A \text{ gilt: } [\![\big(\dot{E}(x,y) \rightarrow \dot{E}(y,x)\big)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1 \\
&\iff \text{für alle } a \in A \text{ und alle } b \in A \text{ gilt:} \\
&\qquad \text{falls } [\![\dot{E}(x,y)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1, \text{ so auch } [\![\dot{E}(y,x)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1 \\
&\iff \text{für alle } a \in A \text{ und alle } b \in A \text{ gilt:} \\
&\qquad \text{falls } (a,b) \in \dot{E}^{\mathfrak{A}}, \text{ so auch } (b,a) \in \dot{E}^{\mathfrak{A}} \\
&\iff \dot{E}^{\mathfrak{A}} \text{ ist symmetrisch, vgl. Def. 4.64.}
\end{aligned}$$

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 hier $[\![\varphi]\!]^{\mathcal{I}} = 0$.

Notation 5.27. Sei σ eine Signatur und sei φ eine $\mathrm{FO}[\sigma]$-Formel.

- Ist $\mathcal{I} = (\mathfrak{A}, \beta)$ eine zu φ passende σ-Interpretation, so sagen wir "$\boldsymbol{\mathcal{I}}$ **erfüllt** $\boldsymbol{\varphi}$" (bzw. "$\boldsymbol{\mathcal{I}}$ **ist ein Modell von** $\boldsymbol{\varphi}$", kurz: $\boldsymbol{\mathcal{I} \models \varphi}$), falls $[\![\varphi]\!]^{\mathcal{I}} = 1$.
- 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."

5.6 Erfüllbarkeit, Allgemeingültigkeit, Folgerung und Äquivalenz

Definition 5.28. 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.

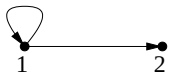
unerfüllbar

(b) φ heißt **unerfüllbar**, wenn φ nicht erfüllbar ist.

allgemeingültig

(c) φ heißt **allgemeingültig**, wenn jede zu φ passende σ-Interpretation φ erfüllt.

Beispiel 5.29. 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 5.30. 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 5.31 (semantische Folgerung)**.** Sei σ eine Signatur und seien φ und ψ zwei FO[σ]-Formeln. Wir sagen ψ **folgt aus** φ (kurz: $\varphi \models \psi$, "φ impliziert ψ"), falls für jede zu φ und ψ passende Interpretation $\mathcal{I}$ gilt:

ψ folgt aus φ

$$\text{Falls} \quad \underbrace{\mathcal{I} \models \varphi}_{\text{d.h. } [\![\varphi]\!]^{\mathcal{I}} = 1} \quad , \quad \text{so auch} \quad \underbrace{\mathcal{I} \models \psi.}_{\text{d.h. } [\![\psi]\!]^{\mathcal{I}} = 1}$$

Definition 5.32 (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

$$\mathcal{I} \text{ erfüllt } \varphi \iff \mathcal{I} \text{ erfüllt } \psi.$$

Beobachtung 5.33. 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) $\varphi \models \psi \iff (\varphi \rightarrow \psi)$ ist allgemeingültig.

Beweis: Übung. □

5.7 Grenzen der Logik erster Stufe

In Beispiel 5.18 und 5.19 haben wir viele Beispiele für umgangssprachliche Aussagen kennengelernt, die man durch Formeln der Logik erster Stufe beschreiben kann (siehe auch die Aufgaben zu diesem Kapitel). Es gibt allerdings auch Aussagen, die **nicht** in der Logik erster Stufe formalisiert werden können:

Satz 5.34. *Sei $\sigma_{Graph} := \{\dot{E}\}$ 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:*

$$\mathfrak{A} \text{ erfüllt } \varphi \iff \mathfrak{A} \text{ ist azyklisch (vgl. Definition 4.14).}$$

(b) 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.

5.8 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 '$\dot{Babylon}$', '$\dot{Casablanca}$', ..., '$\dot{Capote}$', '*Das Leben* $\dot{der}$ *Anderen*', ... usw. – d.h. für jeden Eintrag c in der Beispieldatenbank gibt es ein Konstantensymbol '$\dot{c}$'.

Die σ_{Kino}-Struktur $\mathfrak{A}_{\text{Kino}}$ hat das Universum

$$\begin{aligned} A_{\text{Kino}} := \{&\text{Babylon}, \text{Casablanca}, \text{Cinestar Cubix am Alexanderplatz}, \ldots, \\ &\text{Dresdner Str. } 2, \text{Friedenstr. } 12, \ldots, 61609693, \ldots, \text{Capote}, \ldots, 22{:}00\}, \end{aligned}$$

die 3-stelligen Relationen

$$\begin{aligned} \dot{Orte}^{\mathfrak{A}_{\text{Kino}}} := \{&(\text{Babylon}, \text{Dresdner Str. } 2, 61609693), \\ &(\text{Casablanca}, \text{Friedenstr. } 12, 6775752), \\ &\ldots, \\ &(\text{Moviemento}, \text{Kottbusser Damm } 22, 6924785)\}, \\ \dot{Filme}^{\mathfrak{A}_{\text{Kino}}} := \{&(\text{Capote}, \text{Bennet Miller}, \text{Philip Seymour Hoffman}), \\ &(\text{Capote}, \text{Bennet Miller}, \text{Catherine Keener}), \\ &\ldots, \\ &(\text{Walk the Line}, \text{James Mangold}, \text{Reese Witherspoon})\}, \\ \dot{Programm}^{\mathfrak{A}_{\text{Kino}}} := \{&(\text{Babylon}, \text{Capote}, 17{:}00), \end{aligned}$$

$$\begin{aligned}&(\text{Babylon}, \text{Capote}, 19{:}30),\\ &(\text{Kino in der Kulturbrauerei}, \text{Capote}, 17{:}30),\\ &\dots\}\end{aligned}$$

sowie für jedes in σ_{Kino} vorkommende Konstantensymbol '$\dot{c}$' die Konstante '$\dot{c}$'$^{\mathfrak{A}_{\text{Kino}}} := c$ (d.h.:

$$\begin{aligned}&\text{'}\dot{Babylon}\text{'}^{\mathfrak{A}_{\text{Kino}}} = \text{Babylon},\\ &\text{'}\dot{Capote}\text{'}^{\mathfrak{A}_{\text{Kino}}} = \text{Capote},\\ &\text{'}\dot{George}\ \dot{Clooney}\text{'}^{\mathfrak{A}_{\text{Kino}}} = \text{George Clooney}\end{aligned}$$

usw. Anfragen an die Kinodatenbank lassen sich auf unterschiedliche Art formulieren:

Beispiel 5.35. Eine Anfrage an unsere Kinodatenbank:

> "Gib die Titel aller Filme aus, die um 20:30 Uhr laufen."

In der Datenbankanfragesprache SQL lässt sich dies folgendermaßen formulieren:

```
SELECT   Titel
FROM     Programm
WHERE    Zeit = '20:30'
```

Dieselbe Anfrage lässt sich auch durch die folgende Formel der Logik erster Stufe beschreiben:

$$\varphi_{\text{Filme um 20:30 Uhr}}(x_{\text{T}}) := \exists x_{\text{K}}\, \dot{Programm}(x_{\text{K}}, x_{\text{T}}, \text{'}\dot{20{:}30}\text{'}).$$

Notation 5.36. Sei σ eine Signatur und seien $x_1, \dots, x_n$ Variablen.

- Die Notation $\varphi(x_1, \dots, x_n)$ deutet an, dass φ eine FO[σ]-Formel mit frei(φ) = $\{x_1, \dots, x_n\}$ ist, d.h. dass $x_1, \dots, x_n$ diejenigen Variablen sind, die in φ frei vorkommen.
- Ist $\varphi(x_1, \dots, x_n)$ eine FO[σ]-Formel, ist $\mathfrak{A}$ eine σ-Struktur und sind $a_1, \dots, a_n \in A$ Elemente im Universum von $\mathfrak{A}$, so schreiben wir

  $$\mathfrak{A} \models \varphi[a_1, \dots, a_n],$$

  um auszudrücken, dass für die Belegung $\beta\colon \{x_1, \dots, x_n\} \to A$ mit $\beta(x_1) = a_1, \dots, \beta(x_n) = a_n$ gilt:

  $$(\mathfrak{A}, \beta) \models \varphi.$$

Definition 5.37. Sei σ eine Signatur, $\varphi(x_1, \dots, x_n)$ eine FO[σ]-Formel und $\mathfrak{A}$ eine σ-Signatur. Die von φ in $\mathfrak{A}$ definierte n-stellige Relation ist

$$\varphi(\mathfrak{A}) := \{(a_1, \dots, a_n) \in A^n : \mathfrak{A} \models \varphi[a_1, \dots, a_n]\}.$$

Beispiel 5.38. Die FO[σ_{Kino}]-Formel $\varphi_{\text{Filme um 20:30}}(x_{\text{T}})$ aus Beispiel 5.35 definiert in unserer Kinodatenbank $\mathfrak{A}_{\text{Kino}}$ die 1-stellige Relation:

$$\begin{aligned}\varphi_{\text{Filme um 20:30}}(\mathfrak{A}_{\text{Kino}}) = \{&(\text{Das Leben der Anderen}),\\ &(\text{Good Night and Good Luck})\}.\end{aligned}$$

Darstellung als Tabelle:

Filme um 20:30 Uhr:

Titel
Das Leben der Anderen
Good Night and Good Luck

Beispiel 5.39. Die Anfrage

> "Gib Name und Adresse aller Kinos aus, in denen ein Film läuft, in dem George Clooney mitspielt oder Regie geführt hat."

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:

$$\varphi_{\text{Kinos mit George Clooney}}(x_{\text{K}}, x_{\text{A}}) :=$$
$$\exists x_{\text{Tel}} \exists x_{\text{T}} \exists x_{\text{Z}} \Big(\big(\dot{Orte}(x_{\text{K}}, x_{\text{A}}, x_{\text{Tel}}) \wedge \dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \big) \wedge$$
$$\big(\exists x_{\text{R}}\, \dot{Filme}(x_{\text{T}}, x_{\text{R}}, \text{'George } \dot{\text{C}}\text{looney'}) \vee$$
$$\exists x_{\text{S}}\, \dot{Filme}(x_{\text{T}}, \text{'George } \dot{\text{C}}\text{looney'}, x_{\text{S}}) \big) \Big)$$

In unserer konkreten Kinodatenbank $\mathfrak{A}_{\text{Kino}}$ liefert diese Formel die 2-stellige Relation:

$$\varphi_{\text{Kinos mit George Clooney}}(\mathfrak{A}_{\text{Kino}}) = \{(\text{Filmpalast Berlin}, \text{Kurfürstendamm 225}),$$
$$(\text{Kino in der Kulturbrauerei}, \text{Schönhauser Allee 36})\}.$$

Darstellung als Tabelle:

Kinos mit George Clooney:

Kino	Adresse
Filmpalast Berlin	Kurfürstendamm 225
Kino in der Kulturbrauerei	Schönhauser Allee 36

Datails zum Thema Datenbanken und Datenbankanfragesprachen können Sie in den Vorlesungen "Datenbanksysteme I und II" und "Logik und Datenbanken" kennenlernen.

5.9 Übungsaufgaben zu Kapitel 5

Aufgabe 5.1. 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 5.2. Sei $\sigma = \{\dot{f}, \dot{R}, \dot{c}\}$ eine Signatur mit einem 2-stelligen Funktionssymbol $\dot{f}$, einem 3-stelligen Relationsymbol $\dot{R}$ und einem Konstantensymbol $\dot{c}$. Betrachten Sie die σ-Struktur $\mathfrak{A} = (A, \dot{f}^{\mathfrak{A}}, \dot{R}^{\mathfrak{A}}, \dot{c}^{\mathfrak{A}})$, wobei $A = \{0, 1, 2, 3, 4\}$, $\dot{R}^{\mathfrak{A}} = \{(0, 3, 4), (1, 3, 0), (4, 2, 3)\}$, $\dot{c}^{\mathfrak{A}} = 3$ und die Funktion $\dot{f}^{\mathfrak{A}} \colon A \times A \to A$ definiert ist durch

$\dot{f}^{\mathfrak{A}}$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Zum Beispiel gilt $\dot{f}^{\mathfrak{A}}(2, 3) = 0$ und $\dot{f}^{\mathfrak{A}}(1, 3) = 4$.

Sei $\mathcal{I} = (\mathfrak{A}, \beta)$ die Interpretation mit der Belegung $\beta \colon \textsc{Var} \to A$, für die gilt: $\beta(v_0) = 2$, $\beta(v_1) = 0$, $\beta(v_2) = 1$, $\beta(v_3) = 4$, und $\beta(v_i) = 3$ für alle $i \geq 4$.

(a) Berechnen Sie $[\![t_1]\!]^{\mathcal{I}}$ und $[\![t_2]\!]^{\mathcal{I}}$ für

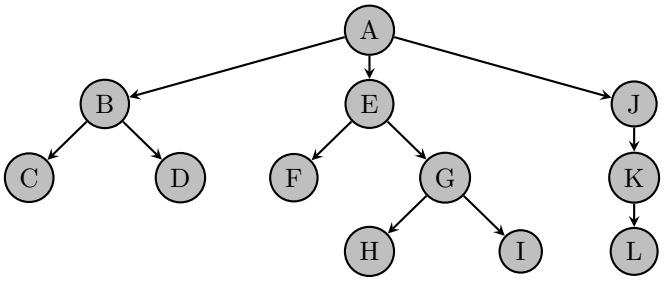
- $t_1 \ :=\ \dot{f}(\dot{f}(v_1, v_5), \dot{c})$
- $t_2 \ :=\ \dot{f}\big(\dot{f}(\dot{c}, \dot{f}(v_2, \dot{c})), v_1\big)$

(b) Berechnen Sie $[\![\varphi_1]\!]^{\mathcal{I}}$ und $[\![\varphi_2]\!]^{\mathcal{I}}$ für

- $\varphi_1 \ :=\ \big(\ \dot{R}(v_1, v_2, \dot{f}(v_0, v_2)) \vee \exists v_0\, \dot{R}(v_0, v_2, v_3)\ \big)$
- $\varphi_2 \ :=\ \forall v_1\Big(\ \dot{f}(v_1, \dot{c}) \doteq \dot{f}(v_2, \dot{c}) \ \rightarrow\ \exists v_3\big(\dot{R}(v_1, v_2, v_3) \vee \dot{f}(v_1, v_5) \doteq v_4\big)\ \Big)$

Aufgabe 5.3. 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 $B = (V, E)$ durch eine Struktur über der Signatur $\{\dot{Eltern}\dot{knoten}\}$ 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

(i) ein Blatt ist,

(ii) die Wurzel ist,

(iii) genau zwei Kinder hat.

Aufgabe 5.4. 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 5.5 (freie und gebundene Variablen)**.** Bestimmen Sie für jede der folgenden $\{\dot{P}, \dot{E}, \dot{R}, \dot{f}, \dot{g}, \dot{c}\}$-Formeln, welche Variablen gebunden und welche Variablen frei in der Formel vorkommen:

(a) $\Big(\dot{P}(x) \vee \neg\big(\dot{E}(y,z) \rightarrow \dot{f}(x) \doteq \dot{f}(y)\big) \Big)$

(b) $\forall x \Big(\dot{f}(x) \doteq \dot{g}(y,x) \vee \exists z\, \dot{E}\big(x, \dot{g}(x,z)\big) \Big)$

(c) $\Big(\forall y \neg\dot{E}(y,x) \wedge \exists z \big(\dot{E}(x,z) \wedge \dot{E}(z,y)\big) \Big)$

(d) $\exists x \forall y \exists z \Big(\dot{f}(y) \doteq \dot{g}(x,z) \vee \neg\dot{R}\big(\dot{c}, z, \dot{f}(y)\big) \Big)$

Aufgabe 5.6 (Äquivalenz, Folgerung)**.**

(a) Welche der folgenden Aussagen stimmen, welche stimmen nicht?

(i) $\forall x\, \varphi \equiv \neg\exists x\, \neg\varphi$

(ii) $\exists x(\varphi \wedge \psi) \models \big(\exists x\, \varphi \wedge \exists x\, \psi\big)$

(iii) $\big(\exists x\, \varphi \wedge \exists x\, \psi\big) \models \exists x(\varphi \wedge \psi)$

(iv) $\exists x(\varphi \wedge \psi) \equiv \big(\exists x\, \varphi \wedge \exists x\, \psi\big)$

(v) $\forall x(\varphi \wedge \psi) \equiv \big(\forall x\, \varphi \wedge \forall x\, \psi\big)$

(vi) $\forall x\,\varphi \models \exists x\,\varphi$

(b) Beweisen Sie, dass Ihre Antworten zu (ii), (iii) und (vi) aus (a) korrekt sind.

Aufgabe 5.7 (Datenbankanfragen)**.** 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:

$$\begin{aligned}
\varphi_1(x_{\text{K}}) &= \exists x_{\text{Z}}\, \dot{Programm}(x_{\text{K}}, \text{`}\dot{Capote}\text{'}, x_{\text{Z}}) \\
\varphi_2(x_{\text{S}}) &= \exists x_{\text{T}}\big(\, \exists x_{\text{R}}\, \dot{Filme}(x_{\text{T}}, x_{\text{R}}, x_{\text{S}}) \;\wedge\; \exists x_{\text{K}} \exists x_{\text{Z}}\, \dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \,\big) \\
\varphi_3(x_{\text{T}}) &= \exists x_{\text{K}} \exists x_{\text{Z}} \Big(\dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \;\wedge \\
&\qquad\qquad \forall y_{\text{K}} \forall y_{\text{Z}} \big(\dot{Programm}(y_{\text{K}}, x_{\text{T}}, y_{\text{Z}}) \rightarrow y_{\text{Z}} \dot{=} x_{\text{Z}} \big) \Big) \\
\varphi_4(x_{\text{T}}, x_{\text{K}}, x_{\text{A}}) &= \Big(\exists x_{\text{Z}}\, \exists x_{\text{Tel}} \big(\dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \wedge \dot{Orte}(x_{\text{K}}, x_{\text{A}}, x_{\text{Tel}}) \big) \;\wedge \\
&\qquad\qquad \exists x_{\text{S}}\, \dot{Filme}(x_{\text{T}}, \text{`}\dot{George}\ \dot{Clooney}\text{'}, x_{\text{S}}) \Big)
\end{aligned}$$

(b) Finden Sie Formeln der Logik erster Stufe, die die folgenden Anfragen beschreiben:

(i) Gib die Titel aller Filme aus, die in mindestens zwei Kinos laufen.

(ii) Gib 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.

(iii) Gib die Titel aller Filme aus, deren Schauspieler schon mal in einem Film von Stephen Spielberg mitgespielt haben.

6 Modellierung von Strukturen

In Kapitel 4 haben wir bereits **Graphen** und **Bäume** als Möglichkeiten kennengelernt, mit denen sich Objekte sowie Beziehungen zwischen je 2 Objekten gut modellieren lassen. In Kapitel 5.2 wurden Verallgemeinerungen davon eingeführt, die so genannten σ-Strukturen, wobei σ eine Signatur ist. Abgesehen von Graphen und Bäumen kann man damit beispielsweise auch die natürlichen (oder die rationalen) Zahlen mit arithmetischen Operationen $+$, $\times$ etc. modellieren oder – wie in Kapitel 5.8 gesehen – auch relationale Datenbanken, die z.B. Informationen über Kinofilme und das aktuelle Kinoprogramm enthalten. In Kapitel 6 werden wir nun zwei weitere Kalküle kennenlernen, mit denen man strukturelle Eigenschaften von Systemen beschreiben kann: das Entity-Relationship-Modell und kontextfreie Grammatiken.

6.1 Das Entity-Relationship-Modell

Das Entity-Relationship-Modell (kurz: **ER-Modell**) geht zurück auf einen grundlegenden Artikel von P.P. Chen aus dem Jahr 1976:

> P.P. Chen „The Entity-Relationship-Model – Towards a Unified View of Data.“ ACM Transactions on Database Systems, Band 1, Nr. 1, Seiten 9–36, 1976.

Es wird heute praktisch als Standardmodell für frühe Entwurfsphasen in der Datenbankentwicklung eingesetzt. Darüber hinaus basiert auch die Spezifikationssprache UML („Unified Modeling Language“), die zur Spezifikation von Strukturen und Beziehungen in Software-Systemen eingesetzt wird, auf dem ER-Modell.

In dieser Vorlesung werden nur einige grundlegende Züge des ER-Modells vorgestellt. Details können Sie z.B. in der Veranstaltung „Datenbanksysteme I“ kennenlernen.

Das ER-Modell basiert auf den 3 Grundkonzepten

- **Entity**: „zu modellierende Informationseinheit“ (deutsch: „Objekt“, „Ding“, „Entität“) — Entity
- **Relationship**: zur Modellierung von Beziehungen zwischen Entities (deutsch: „Beziehung“, „Relation“) — Relationship
- **Attribut**: Eigenschaft von einem Entity oder einer Beziehung. — Attribut

Genauer:

- **Entity**: Objekt der realen oder der Vorstellungswelt, über das Informationen zu speichern sind (z.B. eine Vorlesungsveranstaltung, ein Buch oder eine/n Dozent/in). Auch Informationen über Ereignisse wie Klausuren können Objekte im Sinne des ER-Modells sein. — Entity
- **Entity-Typ** (bzw. **Entity-Menge**): eine Zusammenfassung von Entities, die im Modell als „gleichartig“ angesehen werden (z.B. „Vorlesung“, „Buch“, „Dozent/in“). Im Modell steht ein Entity-Typ für die Menge aller in Frage kommenden Objekte dieser Art. — Entity-Typ, Entity-Menge

Relationship

- **Relationship**: Beziehung zwischen Entities (z.B. welche Dozenten/innen welche Vorlesungen halten).

Attribut

- **Attribut**: Eigenschaften von Entities oder Relationships (z.B. die ISBN eines Buchs, der Titel einer Vorlesung oder die Semester, in denen Vorlesung X von Dozent/in Y gehalten wird).

Beispiel 6.1. Abbildung 6.1 zeigt eine graphische Darstellung für eine Modellierung im ER-Modell – im Beispiel geht es darum, Vorlesungen, Dozenten und für die Vorlesungen empfohlene Bücher darzustellen.

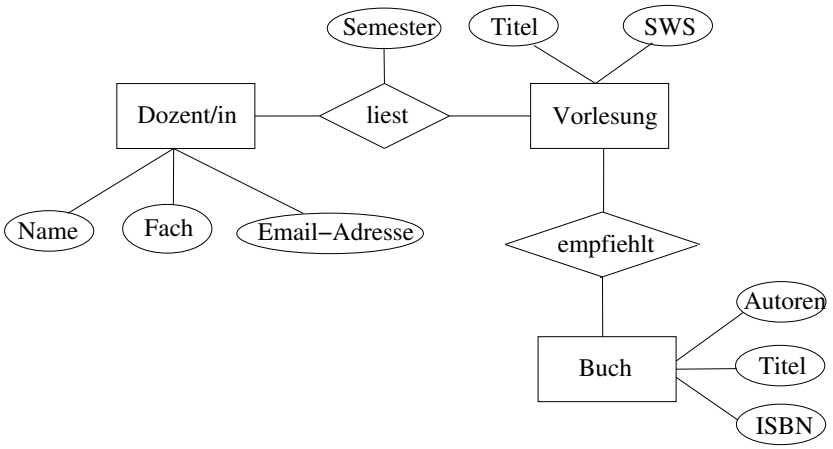
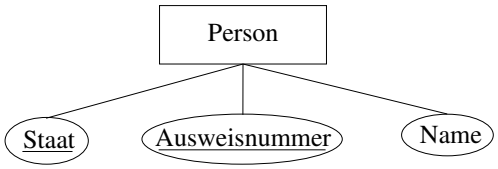


Abbildung 6.1: Ein ER-Modell, das Vorlesungen, Dozenten und für die Vorlesungen empfohlene Bücher darstellt

- **Entity-Typen** werden als Rechtecke dargestellt (hier: Dozent/in, Vorlesung, Buch).
- **Eigenschaften von Entities**, sog. **Attribute**, werden durch Ellipsen dargestellt, die mit dem Rechteck des zugehörigen Entity-Typs verbunden sind (im Beispiel hat jede/r Dozent/in die Attribute „Name“, „Fach“ und „Email-Adresse“).

  Ein Attribut ordnet jeder Entity des entsprechenden Entity-Typs einen Wert zu. Ein Attribut, dessen Wert jedes Entity eindeutig bestimmt (z.B. die ISBN von Büchern), heißt **Schlüsselattribut**. Um Schlüsselattribute im ER-Modell explizit zu kennzeichnen, werden sie unterstrichen. Auch mehrere Attribute zusammen können einen Schlüssel bilden, z.B.

Schlüsselattribut

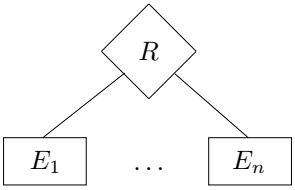


Relationen-Typen

- Typen von Relationships, sog. **Relationen-Typen**, werden durch Rauten dargestellt, die mit den betreffenden Entity-Typen durch Striche verbunden sind (z.B. ist in Beispiel 6.1 „liest“ ein Relationen-Typ, der angibt, welche/r Dozent/in welche Vorlesung liest).

Allgemein gilt: Ein Relationen-Typ modelliert Beziehungen zwischen den Entities der betroffenen Entity-Typen. Ein ***n*-stelliger Relationen-Typ *R*** (für $n \geq 2$) verknüpft Entities aus n Entity-Typen $E_1, \ldots, E_n$. Er wird graphisch repräsentiert durch

n-stelliger Relationen-Typ



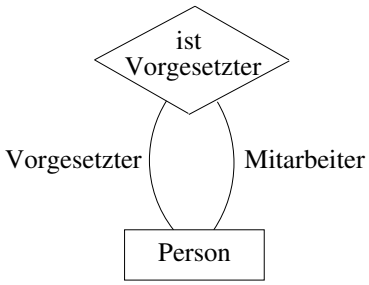
Eine **konkrete Ausprägung des Relationen-Typs** R ist eine Menge von n-Tupeln $(e_1, \ldots, e_n)$, wobei für jedes $i \in \{1, \ldots, n\}$ gilt: e_i ist ein Entity des Entity-Typs E_i.

- Auch Relationen-Typen können Attribute haben (z.B. hat der Relationen-Typ „liest“ in Beispiel 6.1 ein Attribut „Semester“, das angibt, in welchen Semestern Dozent/in X die Vorlesung Y hält).

  Allgemein gilt: Ein Attribut ordnet jedem Tupel des entsprechenden Relationen-Typs einen Wert zu. Beispielsweise ordnet das Attribut „Semester“ jedem Tupel (X, Y) der „liest“-Relationen die Liste aller Semester zu, in denen Dozent/in X die Vorlesung Y hält.

- Für manche Relationen-Typen wird aus ihrem Namen und der graphischen Darstellung zunächst nicht klar, welche Bedeutung die einzelnen Entity-Typen in der Relation haben – insbesondere, wenn ein Entity-Typ mehrfach am Relationen-Typ beteiligt ist. Es können dann **Rollennamen** vergeben werden, etwa um die Beziehung „Person X ist Vorgesetzter von Person Y“ darzustellen:

  Rollennamen



Beispiel 6.2. Man beachte die Auswirkung von Modellierungsentscheidungen beim Entwickeln eines ER-Modells: Nutzt ein Reisebüro das ER-Modell aus Abbildung 6.2, so besteht eine konkrete Ausprägung des Relationen-Typs „gebucht für“ aus einer Menge von Tupeln (X, Y), die angibt, dass Person X ein Ticket für Flug Y gebucht hat – und zwar zum Preis Preis(X, Y). Insbesondere heißt dies aber, dass Passagier X für Flug Y nicht zwei verschiedene Buchungen getätigt haben kann. Wenn man solche „Mehrfachbuchungen“ zulassen will, kann man das ER-Modell aus Abbildung 6.3 benutzen.

Ein weiterer Bestandteil von ER-Modellen

Kardinalität von Relationen-Typen:

Kardinalität

Relationen-Typen in der Form, wie wir sie bisher eingeführt haben, sagen über konkrete Ausprägungen nur aus, dass einige Entities aus den beteiligten Entity-Typen in der angegebenen

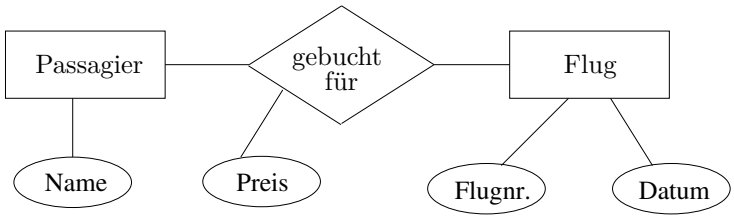


Abbildung 6.2: ER-Modell für ein Reisebüro

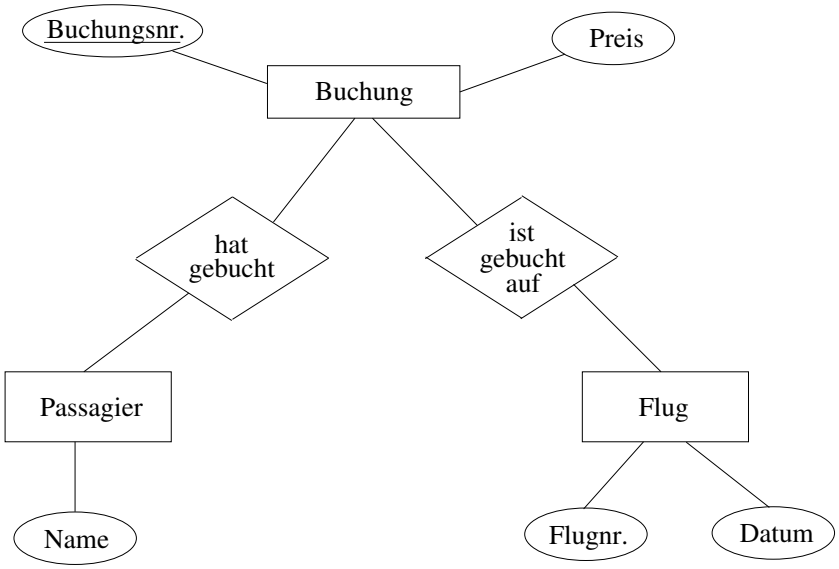
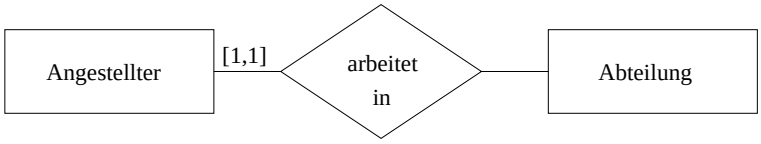
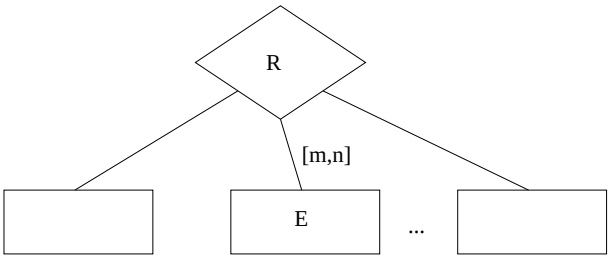


Abbildung 6.3: ER-Modell für ein Reisebüro mit „Mehrfachbuchungen“

Beziehung stehen können. Oft will man aber genauere Angaben (bzw. Einschränkungen) machen – z.B. dass jeder Angestellte durch eine Relation des Relationen-Typs „arbeitet in“ mit genau einer Abteilung verbunden ist. Dies kann graphisch folgendermaßen dargestellt werden:



Allgemein besagt ein Relationen-Typ der Form

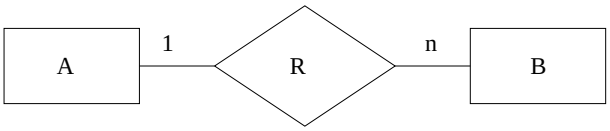


dass für jede konkrete Ausprägung dieses Typs gelten muss: Jedes Entity e der konkreten Ausprägung des Entity-Typs E kommt in mindestens m und höchstens n Tupeln vor.

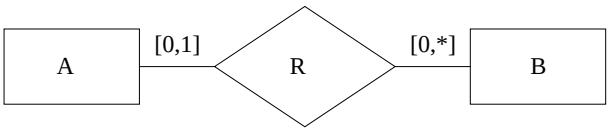
Spezialfälle für $[m, n]$:

- $[1, 1]$ bedeutet: „in genau einem Tupel“.
- $[0, 1]$ bedeutet: „in höchstens einem Tupel“.
- $[0, *]$ bedeutet: „in beliebig vielen Tupeln“.
  Die Angabe $[0, *]$ wird oft auch einfach weggelassen.

Kurznotation für 2-stellige Relationen-Typen:



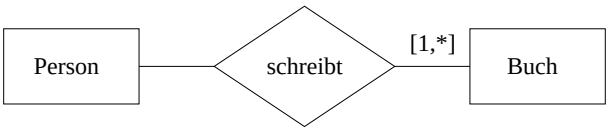
bedeutet



d.h.: „jedes Entity a des Typs A kommt in höchstens einem Tupel von R vor, und jedes Entity b des Typs B kommt in beliebig vielen Tupeln von R vor.“

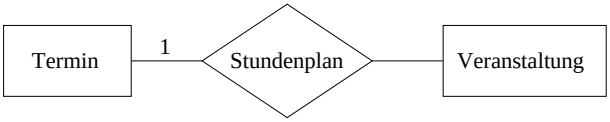
Beispiel 6.3.

(a)



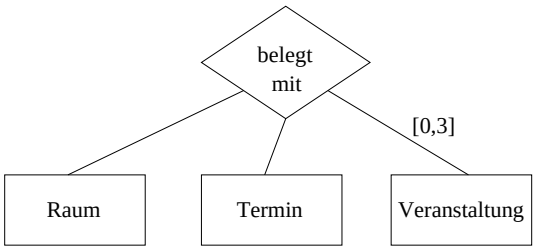
bedeutet: „Jedes Buch wird von mindestens einer Person geschrieben.“

(b)



bedeutet: „Jeder Termin im Stundenplan ist mit höchstens einer Veranstaltung belegt.“

(c)



bedeutet: „Jede Veranstaltung wird höchstens dreimal (pro Woche) angeboten.“

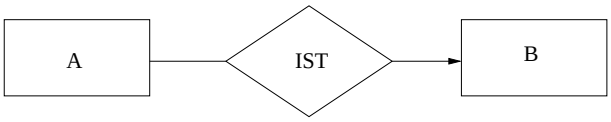
Noch ein Bestandteil von ER-Modellen:

IST-Beziehung

Die **IST-Beziehung** (englisch „is-a“):

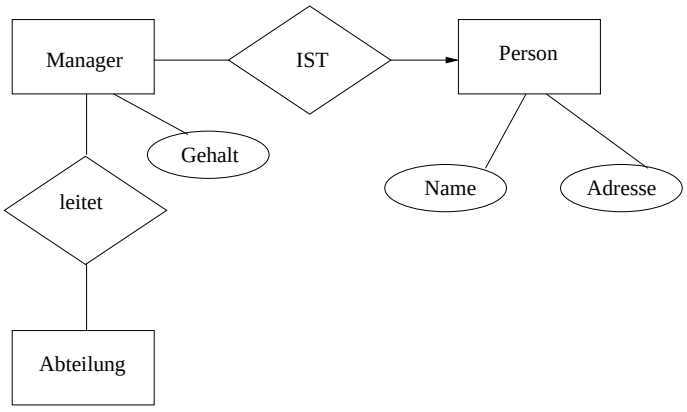
Der spezielle Relationen-Typ IST definiert eine Spezialisierungs-Hierarchie.

Graphische Darstellung:



Bedeutung: Jedes Entity des Typs A ist auch ein Entity des Typs B (d.h. A ist eine Spezialisierung des Typs B).

Beispiel:



Allgemein gilt: Die Entities des Typs A „erben“ alle Attribute von B und können außerdem noch weitere Attribute haben, die spezielle „A-Eigenschaften“ beschreiben. Auch Schlüsselattribute werden als solche geerbt.

Beispiel 6.4. Ein umfangreiches ER-Modell, das einige Aspekte einer Fluggesellschaft modelliert, ist in Abbildung 6.4 dargestellt. In diesem ER-Modell wird u.a. folgendes modelliert:

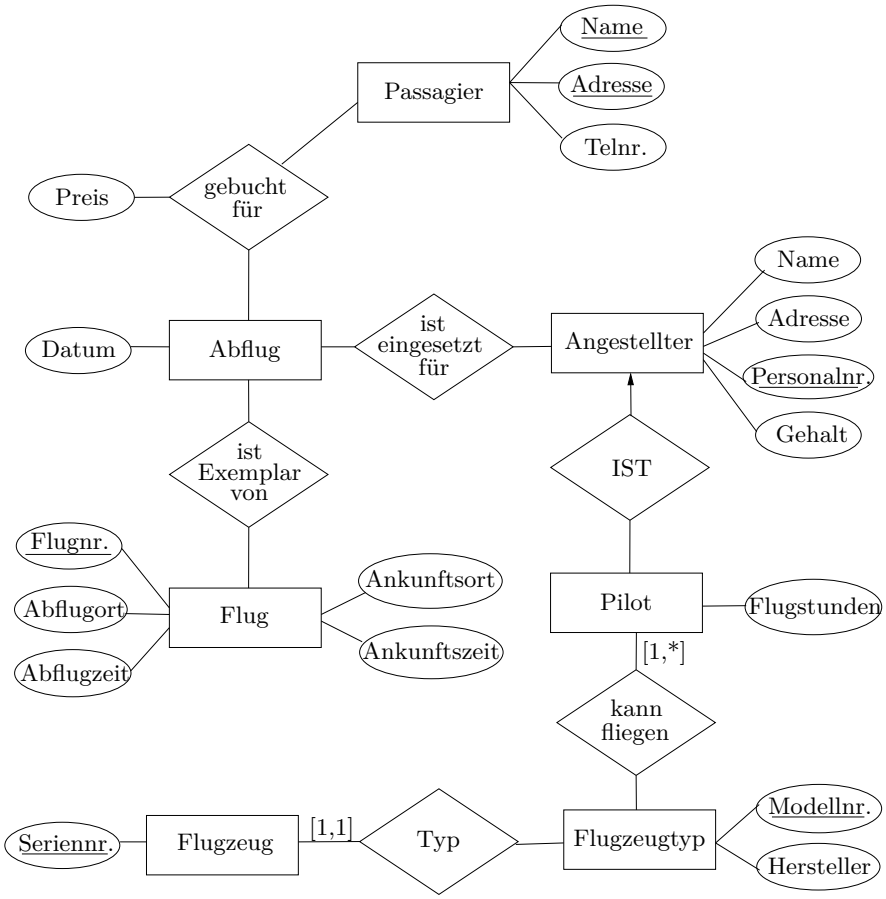


Abbildung 6.4: ER-Modell, das einige Aspekte einer Fluggesellschaft modelliert

(1) Es kommt eine IST-Spezialisierung vor, die besagt, dass Piloten spezielle Angestellte sind. Das wird insbesondere benötigt, um den Relationen-Typ „kann fliegen“ hinreichend präzise formulieren zu können und das Attribut „Flugstunden“ nicht allen Angestellten zuordnen zu müssen.

(2) Entities aller hier aufgeführten Typen (bis auf Abflug) werden durch Schlüsselattribute eindeutig identifiziert. Bei den Passagieren wird angenommen, dass Name und Adresse den jeweiligen Passagier eindeutig festlegen. Die Namen der übrigen Schlüsselattribute deuten an, dass man jeweils eine Nummerierung eingeführt hat, um die Eindeutigkeit zu erreichen (z.B. Personalnr., Flugnr., etc.).

(3) Der Relationen-Typ „Typ“ verbindet konkrete Flugzeuge mit Flugzeugtypen, in dem sie jedem Flugzeug genau einen Flugzeugtypen zuordnet. Solche Unterscheidungen zwischen „Typ“ und „Exemplar“ werden oft in Modellierungen verwendet und sind wichtig, damit Relationen und Attribute sachgerecht zugeordnet werden können. Beispielsweise sind die Fähigkeiten eines Piloten dadurch bestimmt, welche Flugzeugtypen er fliegen kann – und nicht durch die konkreten Flugzeuge (Exemplare), die er fliegen kann.

Vorsicht: Beim ersten Hinsehen mag es verlockend erscheinen, Typ-Exemplar-Beziehungen durch die IST-Spezialisierung zu modellieren. Dies ist aber ein schwerer Entwurfsfehler: „Typen“ und „Exemplare“ bezeichnen verschiedenartige Entity-Typen, zwischen denen i.d.R. keine Teilmengen-Beziehung (wie bei der IST-Spezialisierung) bestehen kann.

6.2 Kontextfreie Grammatik

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.

KFGs sind ein grundlegender Kalkül, der für die formale Definition von Sprachen eingesetzt wird.

In dieser Veranstaltung werden nur die Grundbegriffe und einige Beispiele vorgestellt: im Detail werden KFGs in der Veranstaltung „GL-2“ behandelt.

6.2.1 Definition des Begriffs „Kontextfreie Grammatik“

Es gibt 2 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 von 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 6.5 (KFG)**.** Eine kontextfreie Grammatik $G = (T, N, S, P)$ besteht aus:

Terminalsymbole
Terminale

- einer endlichen Menge T, der so genannten Menge der **Terminalsymbole** (die Elemente aus T werden auch **Terminale** genannt).

- einer endlichen Menge N, der so genannten Menge der **Nichtterminalsymbole** (die Elemente aus N werden auch **Nichtterminale** genannt).

  Die Mengen T und N sind disjunkt, d.h. $T \cap N = \varnothing$. Die Menge $V := T \cup N$ heißt **Vokabular**; die Elemente in V nennt man auch **Symbole**.

- einem Symbol $S \in N$, dem so genannten **Startsymbol**.

- einer endlichen Menge $P \subseteq N \times V^*$, der so genannten Menge der **Produktionen**. Für eine Produktion $(A, x) \in P$ schreiben wir meistens $A \to x$ (bzw. $A ::= x$).

Nichtterminalsymbole
Nichtterminale
Vokabular
Symbole
Startsymbol
Produktionen

Beispiel 6.6. Als erstes Beispiel betrachten wir eine KFG für „Wohlgeformte Klammerausdrücke“: $G_{\mathrm{K}} := (T, N, S, P)$ mit

- $T := \{\,(\,,\,)\,\}$

  D.h. G_{K} hat als Terminale die Symbole „(“ („öffnende Klammer“) und „)“ („schließende Klammer“).

- $N := \{\,\mathrm{Klammerung}\,,\,\mathrm{Liste}\,\}$

  D.h. G_{K} hat als Nichtterminale die Symbole „Klammerung“ und „Liste“.

- $S := \mathrm{Klammerung}$

  D.h. „Klammerung“ ist das Startsymbol.

- $P := \{\,\mathrm{Klammerung} \to (\mathrm{Liste}),$
  $\mathrm{Liste} \to \mathrm{Klammerung\ Liste},$
  $\mathrm{Liste} \to \varepsilon\,\}$

  (zur Erinnerung: ε bezeichnet das leere Wort).

6.2.2 Bedeutung der Produktionen/Semantik von KFGs

Jede Produktion einer KFG, etwa die Produktion $A \to x$, kann man auffassen als:

- „Strukturregel“, die besagt „Ein A besteht aus x“ oder als
- „Ersetzungsregel“, die besagt „A kann man durch x ersetzen.“

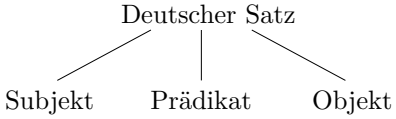
Beispielsweise kann man die Produktion

$$\mathrm{DeutscherSatz} \to \mathrm{Subjekt\ Prädikat\ Objekt}$$

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 6.7 (Ableitung)**.** Sei $G = (T, N, S, P)$ eine KFG.

- Falls $A \to x$ eine Produktion in P ist und $u \in V^*$ und $v \in V^*$ beliebige Worte über der Symbolmenge $V = T \cup N$ sind, so schreiben wir

$$uAv \Longrightarrow_G uxv \quad \text{(bzw. kurz: } uAv \Longrightarrow uxv\text{)}$$

Ableitungsschritt

und sagen, dass uAv in einem **Ableitungsschritt** zu uxv umgeformt werden kann.

Ableitung

- Eine **Ableitung** ist eine Folge von hintereinander angewendeten Ableitungsschritten. Für Worte $w \in V^*$ und $w' \in V^*$ schreiben wir

$$w \Longrightarrow_G^* w' \quad \text{(bzw. kurz: } w \Longrightarrow^* w'\text{)},$$

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. $w \in V^*$ gilt: $w \Longrightarrow^* w$.

Beispiel 6.8. Sei G_K die „Klammer-Grammatik“ aus Beispiel 6.6. Beispiele für einzelne Ableitungsschritte:

- (Liste) $\Longrightarrow$ (Klammerung Liste)
- ((Liste) Liste) $\Longrightarrow$ (() Liste)

Ein Beispiel für eine Ableitung in G_K:

$$\begin{aligned}
\text{Klammerung} &\Longrightarrow \text{(Liste)} \\
&\Longrightarrow \text{(Klammerung Liste)} \\
&\Longrightarrow \text{(Klammerung Klammerung Liste)} \\
&\Longrightarrow \text{(Klammerung (Liste) Liste)} \\
&\Longrightarrow \text{((Liste)(Liste) Liste)} \\
&\Longrightarrow \text{(()(Liste) Liste)} \\
&\Longrightarrow \text{(()()Liste)} \\
&\Longrightarrow \text{(()())}
\end{aligned}$$

In jedem Schritt wird jeweils eine Produktion auf ein Nichtterminal der vorangehenden Symbolfolge angewandt. Obige Kette von Ableitungsschritten zeigt, dass

$$\text{Klammerung} \Longrightarrow^* \text{(()())}$$

Sprache einer KFG, $L(G)$

Definition 6.9 (Sprache einer KFG)**.** Sei $G = (T, N, S, P)$ eine KFG. Die **von G erzeugte Sprache $L(G)$** ist die Menge aller Worte über dem Alphabet T, die aus dem Startsymbol S abgeleitet werden können. D.h.:

$$L(G) \;:=\; \left\{w \in T^* : S \Longrightarrow_G^* w\right\}.$$

Beispiel 6.10. Die KFG G_K aus Beispiel 6.6 definiert die Sprache $L(G_\mathrm{K})$, die aus allen „wohlgeformten Klammerausdrücken“ besteht, die von einem Klammerpaar umschlossen werden. Beispielsweise gehören folgende Worte zu $L(G_\mathrm{K})$:

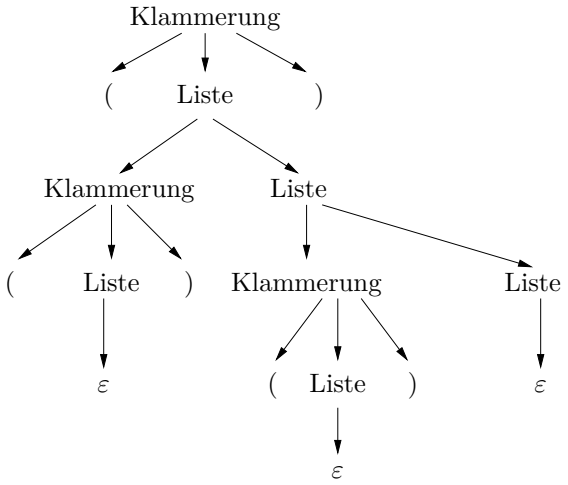
$$\text{() ,} \qquad \text{(()) ,} \qquad \text{(()()) ,} \qquad \text{(((()))()) ,}$$

aber die Worte)(, ((() , (()() , (Liste) gehören **nicht** zu $L(G_\mathrm{K})$.

Beispiel 6.11. Sei G_K die „Klammer-Grammatik“ aus Beispiel 6.6. Die Ableitung

$$\begin{aligned}
\text{Klammerung} &\Longrightarrow (\text{Liste}) \\
&\Longrightarrow (\text{Klammerung Liste}) \\
&\Longrightarrow ((\text{Liste}) \text{ Liste}) \\
&\Longrightarrow (() \text{ Liste}) \\
&\Longrightarrow (() \text{ Klammerung Liste}) \\
&\Longrightarrow (() \text{ Klammerung}) \\
&\Longrightarrow (()(\text{Liste})) \\
&\Longrightarrow (()())
\end{aligned}$$

wird durch den folgenden **Ableitungsbaum** dargestellt:



Ableitungsbäume

Sei $G = (T, N, S, P)$ eine KFG. Jede Ableitung $S \Longrightarrow_G^* w$ lässt sich als gerichteter Baum darstellen, bei dem

- jeder Knoten mit einem Symbol aus $T \cup N \cup \{\varepsilon\}$ 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 $A \in N$ und $x \in V^*$) 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. $i \in \{1, \ldots, |x|\}$). *Spezialfall:* Ist x das leere Wort ε, so wird eine Anwendung der Produktion $A \to \varepsilon$ repräsentiert durch einen mit A markierten Knoten, der genau ein Kind hat, das mit ε markiert ist.

Beachte: Ein Ableitungsbaum kann mehrere Ableitungen repräsentieren. Beispielsweise repräsentiert der obige Ableitungsbaum auch die Ableitung

$$\begin{aligned}
\text{Klammerung} &\Longrightarrow \text{(Liste)} \\
&\Longrightarrow \text{(Klammerung Liste)} \\
&\Longrightarrow \text{(Klammerung Klammerung Liste)} \\
&\Longrightarrow \text{((Liste) Klammerung Liste)} \\
&\Longrightarrow \text{((Liste)(Liste) Liste)} \\
&\Longrightarrow \text{(()(Liste) Liste)} \\
&\Longrightarrow \text{(()() Liste)} \\
&\Longrightarrow \text{(()())}\ ,
\end{aligned}$$

in der gegenüber der ursprünglichen Ableitung aus Beispiel 6.11 einige Ableitungsschritte vertauscht sind. Im Ableitungsbaum wird von der konkreten Reihenfolge, in der die einzelnen Ableitungsschritte vorkommen, abstrahiert.

Im Folgenden betrachten wir einige weitere Beispiele für kontextfreie Grammatiken.

Beispiel 6.12 (Aussagenlogik)**.** Wir konstruieren eine KFG

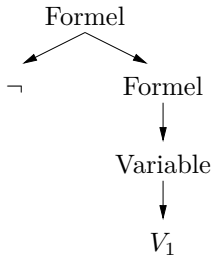
$$G_{\mathrm{AL}} \;=\; (T, N, S, P),$$

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 $T \;:=\; \big\{V_0, V_1, V_2, \mathbf{0}, \mathbf{1}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\big\}$
- Nichtterminalsymbole $N := \big\{\text{Formel}, \text{Variable}, \text{Junktor}\big\}$
- Startsymbol $S :=$ Formel
- Produktionen

$$\begin{aligned}
P \;:=\; \big\{&\text{Formel} \to \mathbf{0},\ \text{Formel} \to \mathbf{1},\ \text{Formel} \to \text{Variable}, \\
&\text{Formel} \to \neg\text{Formel}, \\
&\text{Formel} \to (\text{Formel Junktor Formel}), \\
&\text{Variable} \to V_0,\ \text{Variable} \to V_1,\ \text{Variable} \to V_2, \\
&\text{Junktor} \to \wedge,\ \text{Junktor} \to \vee,\ \text{Junktor} \to \rightarrow,\ \text{Junktor} \to \leftrightarrow\big\}
\end{aligned}$$

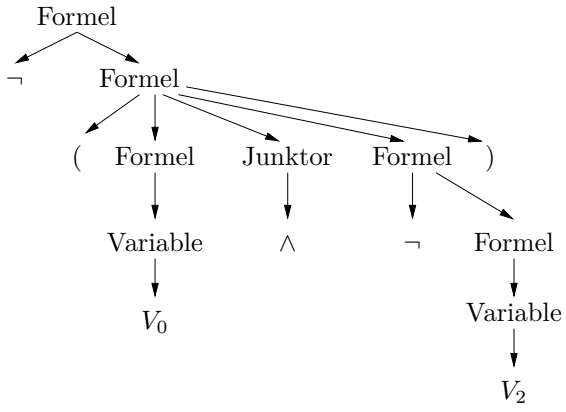
Beispiele für Ableitungsbäume:



Dieser Ableitungsbaum repräsentiert die Ableitung

$$\begin{aligned}
\text{Formel} &\Longrightarrow \neg\text{Formel} \\
&\Longrightarrow \neg\text{Variable} \\
&\Longrightarrow \neg V_1
\end{aligned}$$

Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{\mathrm{AL}})$ ist die Formel $\neg V_1$.



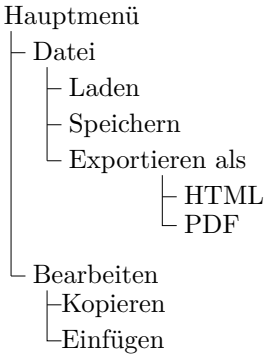
Dieser Ableitungsbaum repräsentiert die Ableitung

$$\begin{aligned}\text{Formel} &\Longrightarrow \neg\text{Formel}\\ &\Longrightarrow \neg(\text{Formel Junktor Formel})\\ &\Longrightarrow \neg(V_0 \text{ Junktor Formel})\\ &\Longrightarrow \neg(V_0 \wedge \text{Formel})\\ &\Longrightarrow \neg(V_0 \wedge \neg\text{Formel})\\ &\Longrightarrow \neg(V_0 \wedge \neg V_2).\end{aligned}$$

Das durch diese(n) Ableitung(sbaum) erzeugte Wort in der Sprache $L(G_{\text{AL}})$ ist die Formel $\neg(V_0 \wedge \neg V_2)$.

Beispiel 6.13 (Menü-Struktur)**.** 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:



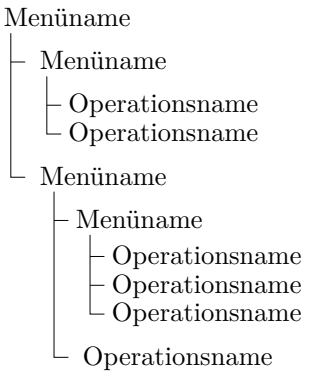
Zur Spezifizierung der Grund-Struktur solcher Menüs kann man folgende Grammatik $G_{\text{Menü}}$ verwenden:

$$G_{\text{Menü}} \;=\; (T, N, S, P)$$

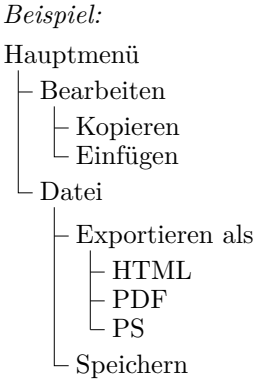
mit

- $T \;:=\; \{\text{Menüname}, \text{Operationsname}\}$
- $N \;:=\; \{\text{Menü}, \text{Eintragsfolge}, \text{Eintrag}\}$
- $S \;:=\; \text{Menü}$
- $P \;:=\; \{\, \text{Menü} \to \text{Menüname Eintragsfolge},$
  $\text{Eintragsfolge} \to \text{Eintrag},$
  $\text{Eintragsfolge} \to \text{Eintrag Eintragsfolge},$
  $\text{Eintrag} \to \text{Operationsname},$
  $\text{Eintrag} \to \text{Menü} \,\}$

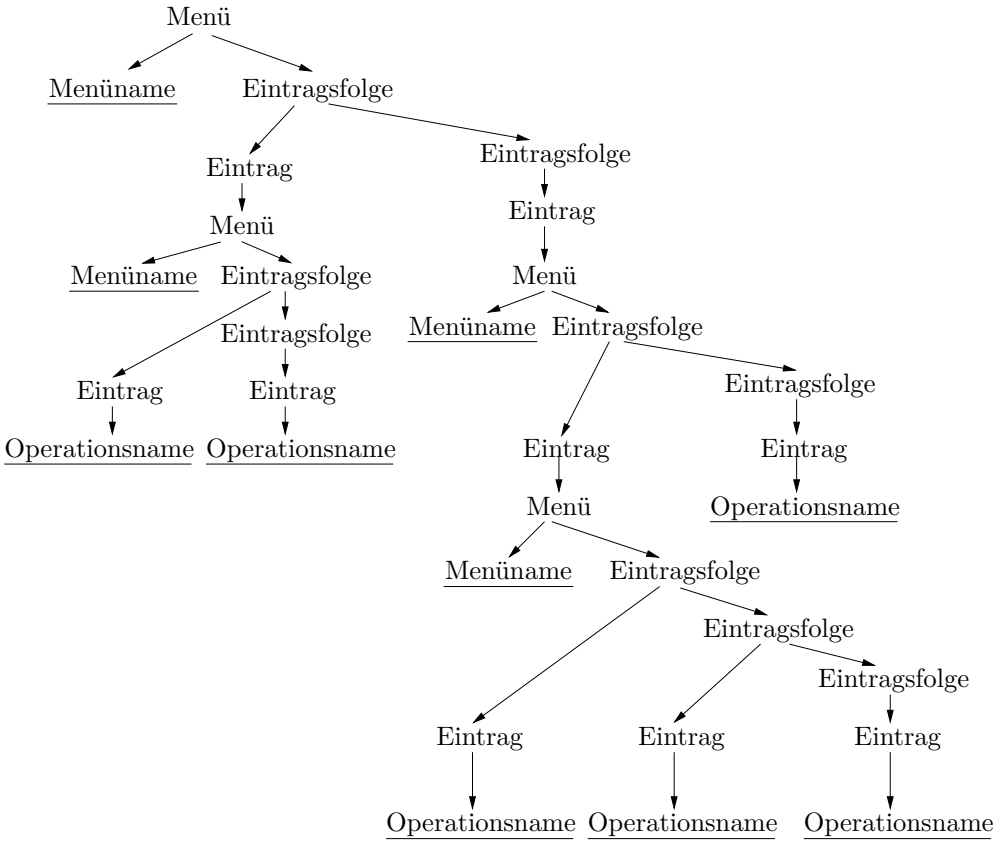
Jeder Ableitungsbaum repräsentiert die Struktur eines Menüs. Ein Menü der Struktur



Beispiel:



wird z.B. von folgendem Ableitungsbaum repräsentiert:



Beispiel 6.14 (HTML-Tabellen)**.**
HTML: Format zur Beschreibung von verzweigten Dokumenten im WWW.
Ein Bestandteil, der oft im Quell-Code von Internet-Seiten vorkommt, sind Tabellen. Z.B. wird der Eintrag

Tag	Zeit	Raum
Mi	8:00-11:00	Magnus-Hörsaal
Do	14:00-16:00	NM 10

durch HTML-Quelltext der folgenden Form erzeugt:

```
<table>
  <tr>
    <td> Tag </td>
    <td> Zeit </td>
    <td> Raum </td>
  </tr>
  <tr>
    <td> Mi </td>
    <td> 8:00-11:00 </td>
    <td> Magnus-Hörsaal </td>
  </tr>
  <tr>
    <td>Do </td>
    <td>14:00-16:00 </td>
    <td>NM10 </td>
  </tr>
</table>
```

<tr> steht für den Anfang einer Zeile der Tabelle,
</tr> steht für das Ende einer Zeile der Tabelle

<td> steht für den Anfang eines Eintrags,
</td> steht für das Ende eines Eintrags

Als Einträge in einzelnen Zellen der Tabelle kann z.B. Text stehen oder eine weitere Tabelle.

Im Folgenden konstruieren wir eine Grammatik

$$G_{\text{HTML-Tabellen}} \;=\; (T, N, S, P),$$

so dass die von $G_{\text{HTML-Tabellen}}$ erzeugte Sprache aus (möglicherweise geschachtelten) HTML-Tabellen besteht:

- $T \;:=\; \big\{$<table>, </table>, <tr>, </tr>, <td>, </td>, a, …, z, A, …, Z, 0, 1, …, 9, :, -, ␣, ä, ö, ü, ß, Ä, Ö, Ü$\big\}$
- $N \;:=\; \big\{$Tabelle, Zeile, Eintrag, Text, Zeilen, Einträge$\big\}$
- $S \;:=\;$ Tabelle
- $P \;:=\; \big\{$ Tabelle → <table>Zeilen</table>,
  Zeilen → Zeile, Zeilen → Zeile Zeilen,
  Zeile → <tr>Einträge</tr>,
  Einträge → Eintrag, Einträge → Eintrag Einträge,
  Eintrag → <td>Text</td>, Eintrag → Tabelle,
  Text → a, …, Text → z, Text → A, …, Text → Ü,
  Text → a Text, …, Text → z Text, Text → A Text, …, Text → Ü Text $\big\}$

Die oben angegebene Beispiel-HTML-Tabelle wird z.B. durch eine Ableitung erzeugt, die durch den Ableitungsbaum in Abbildung 6.5 repräsentiert wird.

Bemerkung 6.15. Typische Fragen bzgl. kontextfreien Grammatiken:

(a) Welche Sprachen können prinzipiell durch KFGs erzeugt werden, welche nicht?

Beispiel: Die Sprache $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ wird von der KFG $G_1 = (T, N, S, P)$ mit $T = \{a, b\}, N = \{S\}$ und $P = \{S \to aSb, S \to \varepsilon\}$ erzeugt, d.h. $L_1 = L(G_1)$.

Notation: $a^n b^n$ ist eine Abkürzung für $\underbrace{aa\cdots a}_{n \text{ Mal}}\underbrace{bb\cdots b}_{n \text{ Mal}}$.

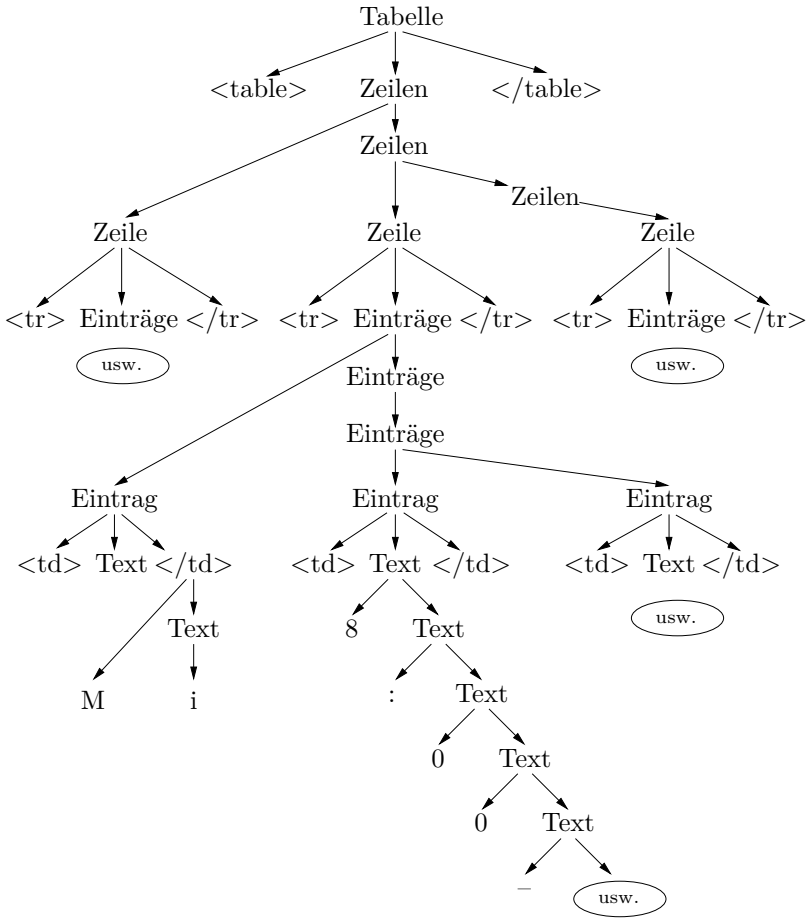


Abbildung 6.5: Ableitungsbaum für eine Beispiel-HTML-Tabelle

Satz. *Es gibt keine KFG, die genau die Sprache* $L_2 := \{a^n b^n c^n : n \in \mathbb{N}\}$ *erzeugt.*

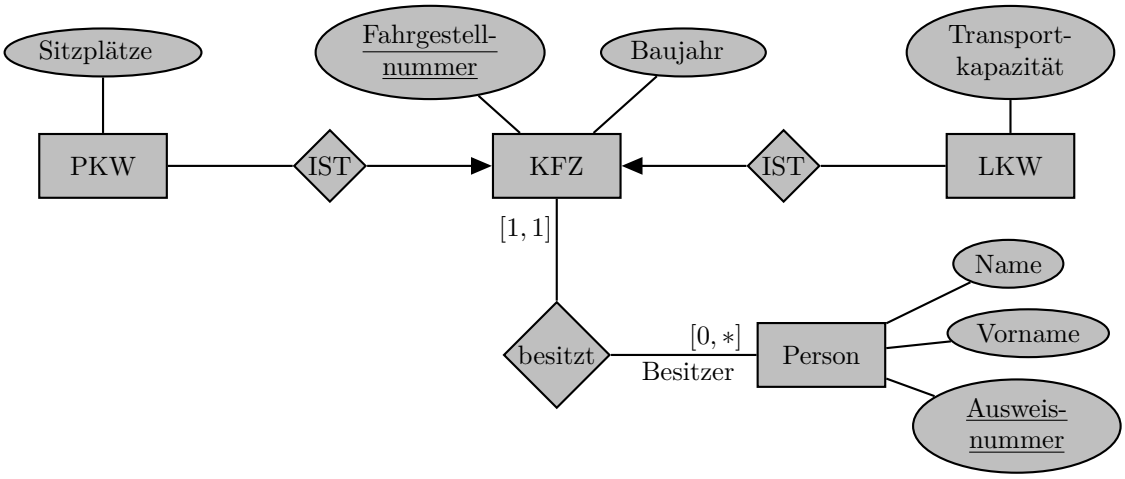
Beweis: In der Vorlesung GL-2.

(b) Gegeben sei eine KFG $G = (T, N, S, P)$ und ein Wort $w \in T^*$. Wie kann man herausfinden, ob $w \in L(G)$ ist, d.h. ob das Wort w zu der von G erzeugten Sprache gehört? Dies ist das so genannte **Wortproblem**.

Ein Algorithmus zum Lösen des Wortproblems für KFGs werden Sie in der Vorlesung GL-2 kennenlernen (den so genannten CYK-Algorithmus, der nach seinen Erfindern Cocke, Younger und Kasami benannt ist).

6.3 Übungsaufgaben zu Kapitel 6

Aufgabe 6.1. Es sei folgendes Entity-Relationship-Modell gegeben:



Weiterhin seien die folgenden konkreten Ausprägungen der einzelnen Entity-Typen PKW, LKW, KFZ und Person gegeben: PKW = $\{PKW1, PKW2\}$, LKW = $\{LKW\}$, KFZ = PKW ∪ LKW und Person = $\{Person1, Person2\}$ mit:

- *PKW1*: Fahrgestellnummer: 123421, Baujahr: 1999, Sitzplätze: 4
- *PKW2*: Fahrgestellnummer: 123123, Baujahr: 2003, Sitzplätze: 6
- *LKW*: Fahrgestellnummer: 123131, Baujahr: 1994, Transportkapazität: 7 Tonnen
- *Person1*: Ausweisnummer: 1234567890, Name: Meier, Vorname: Max
- *Person2*: Ausweisnummer: 9876543210, Name: Müller, Vorname: Martha

(a) Welche der folgenden Relationen sind zulässige Ausprägungen des Relationen-Typs „besitzt“?

a) $\{(Person1, PKW1), (Person2, PKW2), (Person1, LKW)\}$
b) $\{(Person1, PKW1), (Person1, PKW2), (Person1, LKW), (Person2, PKW2)\}$
c) $\{(Person1, PKW1), (Person1, PKW2), (Person1, LKW)\}$
d) $\{(Person1, PKW1), (Person2, LKW)\}$

(b) Würde es dem Modell widersprechen, wenn

(a) *Person1* und *Person2* den gleichen (Nach)Namen hätten?
(b) *PKW1* und *PKW2* die gleiche Fahrgestellnummer hätten?
(c) *PKW1* und *LKW* das gleiche Baujahr hätten?
(d) *PKW2* ein Entity vom Typ KFZ (d.h. $PKW2 \in$ KFZ), aber nicht vom Typ PKW (d.h. $PKW2 \notin$ PKW) wäre?

Begründen Sie jeweils Ihre Antworten!

Aufgabe 6.2. In einer Bibliothek gibt es verschiedene Exemplare von Büchern. Die Bücher sind eindeutig über Ihre ISBN gekennzeichnet und besitzen darüberhinaus noch einen Titel, eine Seitenanzahl und ein Erscheinungsjahr. Die Exemplare eines bestimmten Buches sind fortlaufend nummeriert und weiterhin durch eine Inventarnummer und den Standort charakterisiert. Für jedes Buch wird vermerkt, welche Exemplare dieses Buches in der Bibliothek vorhanden sind. Bücher sind in einem Verlag erschienen, von dem der Name und der Verlagsort registriert wird.

Für jeden Bibliotheksbenutzer ist der Name, Vorname, Wohnort und das Geburtsdatum gespeichert. Benutzer können einzelne Buchexemplare ausleihen, wobei jeweils das Datum der Ausleihe registriert wird. Weiterhin können Benutzer einzelne Bücher vorbestellen, wobei auch hier das Datum der Vorbestellung registriert wird.

Geben Sie ein Entity-Relationship-Modell für den oben beschriebenen Sachverhalt an! Geben Sie auch Kardinalitäten und Schlüsselattribute an!

Aufgabe 6.3. Gegeben sei folgende Grammatik $G = (T, N, S, P)$ mit

- $T = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}, \mathtt{@}, \mathtt{.}\}$
- $N = \{S, X, Y, Z\}$
- $P = \big\{S \to X\mathtt{@}Y,$
  $Y \to X\,\mathtt{.}\,Z,\ Z \to X\,\mathtt{.}\,Z,\ Z \to X,$
  $X \to \mathtt{a}X,\ X \to \mathtt{b}X,\ \ldots, X \to \mathtt{z}X,\ X \to \mathtt{a},\ X \to \mathtt{b},\ \ldots, X \to \mathtt{z}\big\}$

(a) Überprüfen Sie für jedes der folgenden Worte, ob es in der von G erzeugten Sprache liegt. Wenn ja, dann geben Sie einen Ableitungsbaum für dieses Wort an; ansonsten begründen Sie, warum das Wort nicht zur Sprache gehört.

(i) `meier@web.de`
(ii) `Meier@web.de`
c) `max.meier@web.de`
d) `meier@www.web.de`
e) `root@localhost`

(b) Beschreiben Sie in Worten, welche Sprache $L(G)$ von der Grammatik G erzeugt wird.

Aufgabe 6.4. Sei σ eine Signatur mit einem zweistelligen Relationssymbol $\dot{E}$, einem zweistelligen Funktionssymbol $\dot{f}$, einem einstelligen Funktionssymbol $\dot{g}$ und einem Konstantensymbol $\dot{c}$.

(a) Konstruieren Sie eine Grammatik G_{Terme}, so dass $L(G_{\text{Terme}})$ genau die Menge aller σ-Terme ist, in denen nur Variablen aus $\{v_0, v_1, v_2\}$ vorkommen. Geben Sie einen Ableitungsbaum für den Term $t := \dot{f}\big(v_0, \dot{g}(\dot{f}(v_2, \dot{c}))\big)$ an.

(b) Konstruieren Sie eine Grammatik $G_{\text{FO}[\sigma]}$, so dass $L(G_{\text{FO}[\sigma]})$ genau die Menge aller FO[σ]-Formeln ist, in denen nur Variablen aus $\{v_0, v_1, v_2\}$ vorkommen. Geben Sie einen Ableitungsbaum für die Formel $\varphi := \forall v_0\big(\dot{E}(v_0, \dot{f}(v_2, \dot{c})) \to \dot{g}(v_1) \doteq v_0\big)$ an.

7 Modellierung von Abläufen

In diesem Kapitel geht es darum, das dynamische Verhalten von Systemen zu beschreiben, z.B.

- die Wirkung von Bedienoperationen auf reale Automaten oder auf die Benutzungsoberflächen von Software-Systemen
- Schaltfolgen von Ampelanlagen
- Abläufe von Geschäftsprozessen in Firmen
- Steuerung von Produktionsanlagen.

Solche Abläufe werden modelliert, indem man die Zustände angibt, die ein System annehmen kann, und beschreibt, unter welchen Bedingungen es aus einem Zustand in einen anderen übergehen kann (vgl. das Flussüberquerungs-Problem aus Beispiel 1.1).

In diesem Kapitel werden zwei grundlegende Kalküle vorgestellt, mit denen man solche Abläufe beschreiben kann:

(1) **endliche Automaten**, die sich gut zur Modellierung sequentieller Abläufe eignen, und

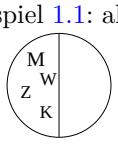
(2) **Petri-Netze**, mit denen nebenläufige Prozesse beschrieben werden können, bei denen Ereignisse gleichzeitig an mehreren Stellen des Systems Zustandsänderungen bewirken können.

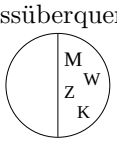
7.1 Endliche Automaten

Endliche Automaten sind ein formaler Kalkül, der zur Spezifikation von realen oder abstrakten Maschinen genutzt werden kann. Endliche Automaten

- reagieren auf äußere Ereignisse
- ändern ggf. ihren „inneren Zustand“
- produzieren ggf. eine Ausgabe.

Sie werden z.B. eingesetzt, um

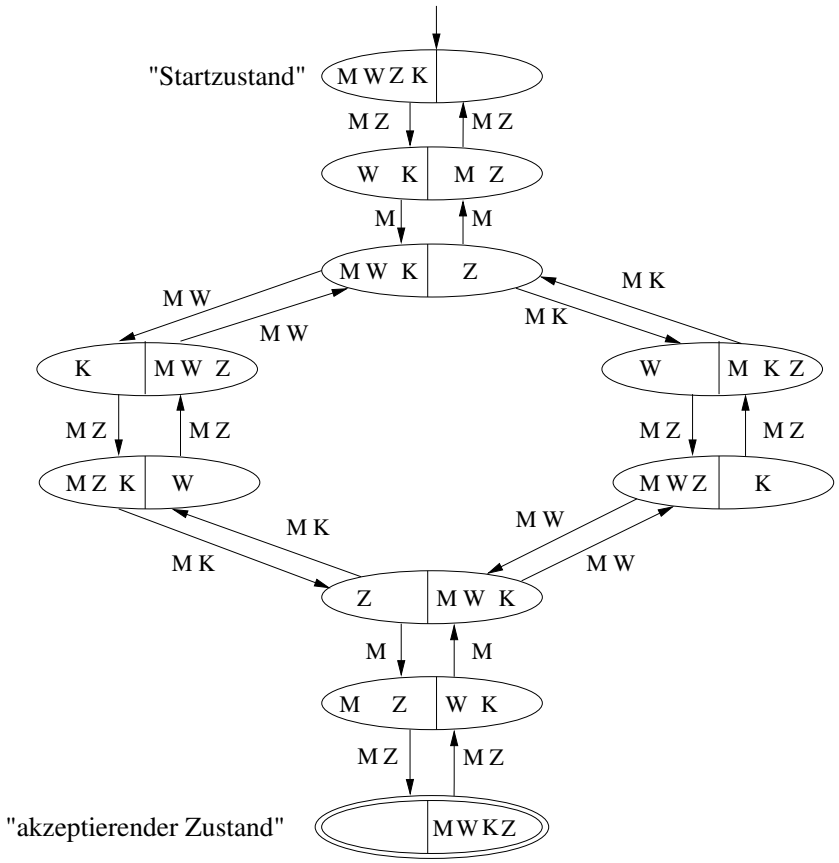
- das Verhalten realer Maschinen zu spezifizieren
  (Bsp.: Getränkeautomat)
- das Verhalten von Software-Komponenten zu beschreiben
  (Bsp.: das Wirken von Bedienoperationen auf Benutzungsoberflächen von Software-Systemen)
- Sprachen zu spezifizieren, d.h. die Menge aller Ereignisfolgen, die den Automat von seinem „Startzustand“ in einen „akzeptierenden Zustand“ überführen.
  (Bsp.: „Flussüberquerung“ aus Beispiel 1.1: alle Folgen von „Flussüberquerungsschritten“, mit denen man vom „Startzustand“ 

zum „Zielzustand“ 

gelangen kann).

Vor der formalen Definitionen endlicher Automaten zunächst zwei einführende Beispiele:

Beispiel 7.1. Graphische Darstellung eines endlichen Automaten zum Flussüberquerungs-Problem aus Beispiel 1.1:



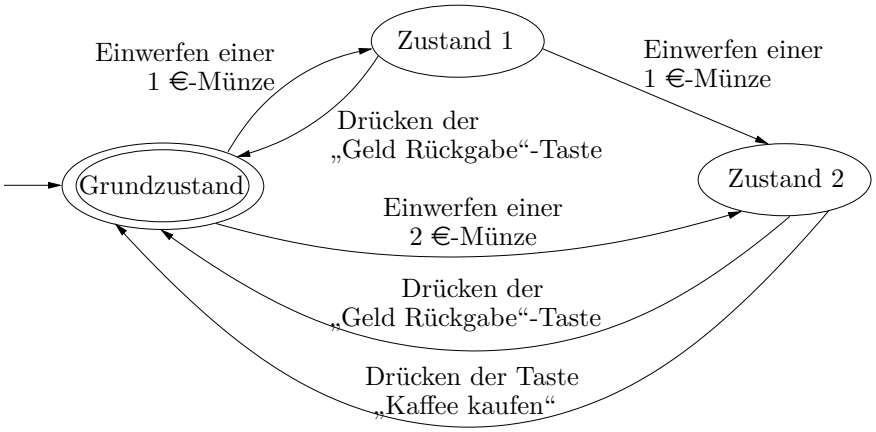
Dieser endliche Automat „akzeptiert“ genau diejenigen Folgen von einzelnen Flussüberquerungen, die vom Startzustand in den akzeptierenden Zustand führen.

Beispiel 7.2. Betrachte einen einfachen Getränkeautomat, der folgende Bedienoptionen hat:

- Einwerfen einer 1 €-Münze
- Einwerfen einer 2 €-Münze
- Taste „Geld Rückgabe“ drücken
- Taste „Kaffee kaufen“ drücken

und bei dem man ein einziges Getränk kaufen kann, das 2 € kostet.

Dieser Getränkeautomat kann durch folgenden endlichen Automaten modelliert werden:



Dieser endliche Automat „akzeptiert“ genau diejenigen Folgen von Bedienoperationen, die vom Grundzustand aus wieder in den Grundzustand führen.

7.1.1 Deterministische endliche Automaten

(engl.: deterministic finite automaton, kurz: DFA)

Definition 7.3 (DFA)**.** Ein **deterministischer endlicher Automat** deterministischer endlicher Automat

$$A \;=\; (\Sigma,\, Q,\, \delta,\, q_0,\, F)$$

besteht aus:

- einer endlichen Menge Σ, dem so genannten **Eingabealphabet** Eingabealphabet
- einer endlichen Mengen Q, der so genannten **Zustandsmenge** (die Elemente aus Q werden **Zustände** genannt) Zustandsmenge Zustände
- einer partiellen Funktion δ von $Q \times \Sigma$ nach Q, der so genannten (Zustands-)**Übergangsfunktion** (oder **Überführungsfunktion**) Übergangsfunktion Überführungsfunktion
- einem Zustand $q_0 \in Q$, dem sogenannten **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“). Endzustände akzeptierenden Zustände

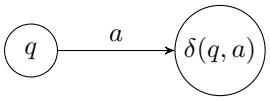
Graphische Darstellung endlicher Automaten:

Endliche Automaten lassen sich anschaulich durch beschriftete Graphen darstellen (vgl. Beispiel 7.1 und 7.2):

- Für jeden Zustand $q \in Q$ gibt es einen durch  dargestellten Knoten.
- Der Startzustand q_0 wird durch einen in ihn hinein führenden 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 dass $(q, a) \in \mathrm{Def}(\delta)$ liegt, so gibt es in der graphischen Darstellung von A einen mit dem Symbol a beschrifteten Pfeil von Knoten  zu Knoten  , d.h.:



Beispiel 7.4. Die graphische Darstellung aus Beispiel 7.2 repräsentiert den DFA $A = (\Sigma, Q, \delta, q_0, F)$ mit

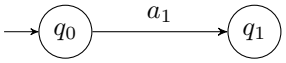
- Σ = {Einwerfen einer 1 €-Münze, Einwerfen einer 2 €-Münze, Drücken der „Geld Rückgabe“-Taste, Drücken der Taste „Kaffee kaufen“}
- Q = {Grundzustand, Zustand 1, Zustand 2}
- q_0 = Grundzustand
- F = {Grundzustand}
- δ ist die partielle Funktion von $Q \times \Sigma$ nach Q mit

$$\begin{aligned}
\delta(\text{Grundzustand}, \text{Einwerfen einer 1 €-Münze}) &= \text{Zustand 1} \\
\delta(\text{Grundzustand}, \text{Einwerfen einer 2 €-Münze}) &= \text{Zustand 2} \\
\delta(\text{Zustand 1}, \text{Einwerfen einer 1 €-Münze}) &= \text{Zustand 2} \\
\delta(\text{Zustand 1}, \text{Drücken der „Geld Rückgabe“-Taste}) &= \text{Grundzustand} \\
\delta(\text{Zustand 2}, \text{Drücken der „Geld Rückgabe“-Taste}) &= \text{Grundzustand} \\
\delta(\text{Zustand 2}, \text{Drücken der Taste „Kaffee kaufen“}) &= \text{Grundzustand}
\end{aligned}$$

Die von einem DFA akzeptierte Sprache:

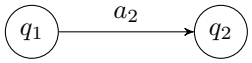
Ein DFA $A = (\Sigma, Q, \delta, q_0, F)$ erhält als Eingabe ein Wort $w \in \Sigma^*$, das eine Folge von „Aktionen“ oder „Bedienoperationen“ repräsentiert. Ist das Eingabewort w von der Form $a_1 \cdots a_n$ mit $n \geq 0$ und $a_1 \in \Sigma, \ldots, a_n \in \Sigma$, so geschieht bei der „Verarbeitung“ von w durch A folgendes: A wird im „Startzustand“ q_0 gestartet. Durch Lesen der ersten Buchstabens von w, also a_1, geht der Automat über in den Zustand $q_1 := \delta(q_0, a_1)$. 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.



Dies ist allerdings nur möglich, wenn $(q_0, a_1) \in \mathrm{Def}(\delta)$ liegt – d.h. wenn es in der graphischen Darstellung von A eine mit a_1 beschriftete Kante gibt, die aus Zustand q_0 herausführt. Falls es keine solche Kante gibt, d.h. falls $(q_0, a_1) \notin \mathrm{Def}(\delta)$, so „stürzt A ab“, und die Verarbeitung des Worts w ist beendet. Ansonsten ist A nach Lesen des ersten Symbols von w im Zustand $q_1 := \delta(q_0, a_1)$. Durch Lesen des zweiten Symbols von w, also a_2, geht A nun in den Zustand $q_2 := \delta(q_1, a_2)$ über – bzw. „stürzt ab“, falls $(q_1, a_2) \notin \mathrm{Def}(\delta)$. In der graphischen Darstellung wird

durch die mit a_2 beschriftete Kante verlassen (falls eine solche Kante existiert); und $q_2 := \delta(q_1, q_2)$ ist der Endknoten dieser Kante, d.h.



Auf diese Wiese wird nach und nach das gesamte Eingabewort $w = a_1 \cdots a_n$ abgearbeitet und – ausgehend vom Startzustand q_0 – werden 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 (falls der Automat nicht zwischendurch abstürzt, d.h. falls es überhaupt einen mit $a_1, \ldots, a_n$ beschrifteten in

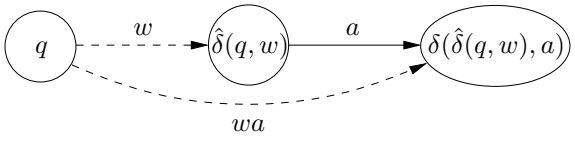
startenden Weg gibt), ist **der von A bei Eingabe w erreichte Zustand**, kurz: $q_n = \widehat{\delta}(q_0, w)$. (Im Fall, dass A bei Eingabe von w zwischendurch abstürzt, sagen wir: $\widehat{\delta}(q_0, w)$ ist undefiniert).

Präzise Definition von $\widehat{\delta}$:

Definition 7.5. Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein DFA. Die partielle Funktion $\widehat{\delta}$ von $Q \times \Sigma^*$ nach 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:
  Falls $(q, w) \in \mathrm{Def}(\widehat{\delta})$ und $(\widehat{\delta}(q, w), a) \in \mathrm{Def}(\delta)$, so ist $\widehat{\delta}(q, wa) := \delta(\widehat{\delta}(q, w), a)$.

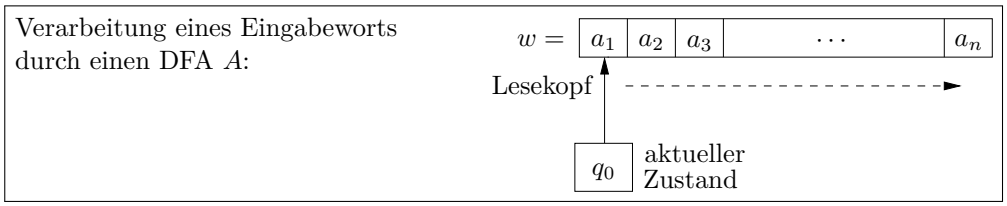
  Graphische Darstellung:



Insgesamt gilt: Falls $(q_0, w) \in \mathrm{Def}(\hat{\delta})$, so ist $\hat{\delta}(q_0, w)$ der Zustand, der durch Verarbeiten des Worts w erreicht wird; falls $(q_0, w) \notin \mathrm{Def}(\hat{\delta})$, so stürzt der Automat beim Verarbeiten des Worts w ab.

Das Eingabewort w wird vom DFA A **akzeptiert**, falls er bei Eingabe von w nicht abstürzt und der durch Verarbeiten des Worts w erreichte Zustand zur Menge F der akzeptierenden Zustände gehört. 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.



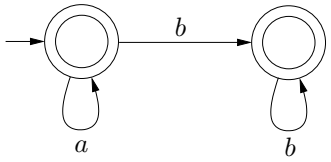
Definition 7.6 (Die von einem DFA A akzeptierte Sprache $L(A)$)**.**
Die von einem DFA $A = (\Sigma, Q, \delta, q_0, F)$ akzeptierte Sprache $L(A)$ ist

$$L(A) \;:=\; \{w \in \Sigma^* : \hat{\delta}(q_0, w) \in F\}.$$

D.h. ein Wort $w \in \Sigma^*$ gehört genau dann zur Sprache $L(A)$, wenn es vom DFA A akzeptiert wird.

Beispiel 7.7. Der Einfachheit halber betrachten wir das Eingabealphabet $\Sigma := \{a, b\}$.

(a) Sei A_1 ein DFA mit folgender graphischer 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, $aabbba$. Insgesamt gilt:

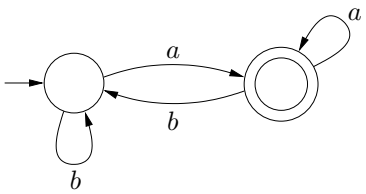
$$L(A_1) \;=\; \{a^n b^m : n \in \mathbb{N},\, m \in \mathbb{N}\}$$

(**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

$$L(A_2) \;=\; \{w \in \{a, b\}^* : \text{der letzte Buchstabe von } w \text{ ist ein } a\}$$

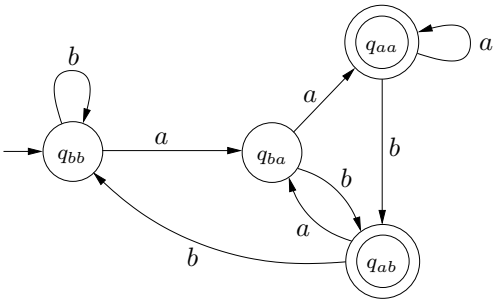
ist



(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



Bemerkung 7.8.

- Die in Definition 7.3 eingeführten DFAs $A = (\Sigma, Q, \delta, q_0, F)$ heißen **deterministisch**, weil es zu jedem Paar $(q, a) \in Q \times \Sigma$ höchstens einen „Nachfolgezustand“ $\delta(q, a)$ gibt (da δ eine partielle Funktion von $Q \times \Sigma$ nach Q ist). Beim Verarbeiten eines Eingabeworts ist daher zu jedem Zeitpunkt klar, ob A „abstürzt“ oder nicht – und falls nicht, welchen eindeutig festgelegten Nachfolgezustand A annimmt. deterministisch

- Ein DFA A heißt **vollständig**, wenn die Übergangsfunktion δ eine totale Funktion $\delta\colon Q \times \Sigma \to Q$ ist. vollständig

  Beispiel: Die DFAs A_2 und A_3 aus Beispiel 7.7 sind vollständig; der DFA A_1 nicht, und auch die DFAs aus Beispiel 7.1 und 7.2 sind nicht vollständig.

  Für die graphische Darstellung eines DFAs gilt: Der DFA ist genau dann vollständig, wenn für jeden Zustand q gilt: Für jedes Symbol $a \in \Sigma$ gibt es genau eine aus  herausführende Kante, die mit a beschriftet ist.

Beachte: In manchen Büchern weicht die Definition von DFAs von Definition 7.3 ab, indem gefordert wird, dass DFAs grundsätzlich vollständig sein müssen.

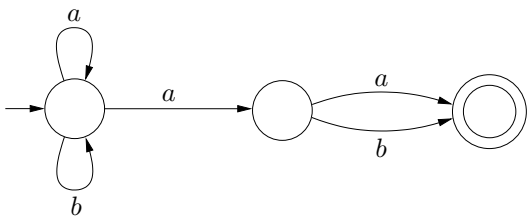
7.1.2 Nicht-deterministische endliche Automaten
(engl.: non-deterministic finite automaton, kurz: NFA)

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 nennt man nicht-deterministisch. Nicht-deterministische Modelle sind häufig einfacher aufzustellen und leichter zu verstehen als deterministische.

Beispiel 7.9. In Beispiel 7.7(c) haben wir einen (recht komplizierten) DFA A_3 kennengelernt mit

$$L(A_3) \;=\; \{w \in \{a,b\}^* : \text{der } \textbf{vorletzte} \text{ Buchstabe von } w \text{ ist ein } a\}$$

Dieselbe Sprache wird auch vom deutlich einfacheren **nicht-deterministischen endlichen Automaten** (kurz: NFA) A_4 mit der folgenden graphischen Darstellung akzeptiert: Generell gilt:



Ein Eingabewort w 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.

Präzise Definition von NFAs:

nicht-det. endlicher Automat

Definition 7.10. Ein **nicht-deterministischer endlicher Automat** $A = (\Sigma, Q, \delta, q_0, F)$ besteht aus:

Eingabealphabet
- einer endlichen Menge Σ, dem so genannten **Eingabealphabet**,

Zustandsmenge
- einer endlichen Menge Q, der so genannten **Zustandsmenge**,

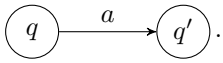
Übergangsfunktion
- einer Funktion $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$, der so genannten **Übergangsfunktion**, die jedem Zustand $q \in Q$ und jedem Symbol $a \in \Sigma$ eine **Menge $\delta(q,a)$ von möglichen Nachfolgerzuständen** zuordnet (beachte: möglicherweise ist $\delta(q,a) = \varnothing$ – dann „stürzt“ der Automat ab, wenn er im Zustand q ist und das Symbol a liest),

Startzustand
- einem Zustand $q_0 \in Q$, dem so genannten **Startzustand**,

Endzustände akzeptierenden Zustände
- einer Menge $F \subseteq Q$, der so genannten Menge der **Endzustände** bzw. **akzeptierenden Zustände**.

Graphische Darstellung von NFAs:

Wie bei 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  zu Knoten  , d.h.



.

Die von einem NFA A akzeptierte Sprache $L(A)$:

Definition 7.11. 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 ganau 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.

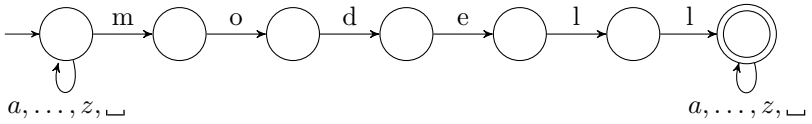
(b) Die von A akzeptierte Sprache $L(A)$ ist

$$L(A) \;:=\; \{w \in \Sigma^* : A \text{ akzeptiert } w\}.$$

Ein Anwendungsbeispiel: Stichwort-Suche in Texten

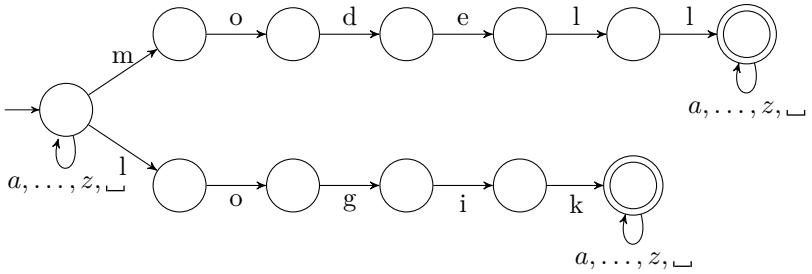
Gegeben:	Ein Stichwort, z.B. „modell“
Eingabe:	Ein Text, der aus den Buchstaben $a, \ldots, z, \text{␣}$ besteht
Frage:	Kommt das Stichwort „modell“ irgendwo im Eingabetext vor? – Der Eingabetext soll genau dann akzeptiert werden, wenn er das Stichwort „modell“ enthält.

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



Variante: Kommt mindestens eins der Stichworte „modell“ bzw. „logik“ im Eingabetext vor?

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



7.1.3 NFAs vs. DFAs

Frage:	Können NFAs wirklich „mehr“ als DFAs?
Antwort:	Nein:

Satz 7.12. *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 DFA können genau dieselben Sprachen akzeptieren.

Beweis: In der Vorlesung GL-2. □

Bemerkung 7.13. Typische Fragen bzgl. DFAs bzw. NFAs:

(a) Gegeben sei ein DFA oder NFA A. Wie kann man herausfinden, ob $L(A) \neq \varnothing$ ist, d.h. ob es (mindestens) ein Eingabewort gibt, das von A akzeptiert wird (↗ vgl. das Flussüberquerungsproblem aus Beispiel 1.1).

Antwort: Teste, ob es in der graphischen Darstellung von A einen Weg gibt, der vom Startzustand zu einem akzeptierenden Zustand führt.

(b) Wie schwierig ist es, zu einem gegebenen NFA einen DFA zu finden, der dieselbe Sprache akzeptiert?

Antwort(en): In der Vorlesung GL-2.

regulär

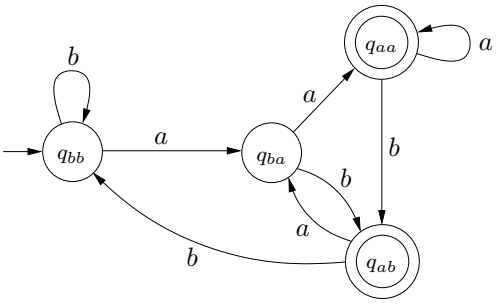
Definition 7.14 (reguläre Sprachen)**.** Sei Σ ein endliches Alphabet. Eine Sprache $L \subseteq \Sigma^*$ heißt **regulär**, falls es einen NFA $A = (\Sigma, Q, \delta, q_0, F)$ mit $L(A) = L$ gibt.
(d.h. für jedes Eingabewort $w \in \Sigma^*$ gilt: A akzeptiert $w \iff w \in L$)

Klar: Um zu zeigen, dass eine Sprache $L \subseteq \Sigma^*$ regulär ist, reicht es, einen NFA oder einen DFA A mit $L(A) = L$ zu finden.

Frage: Wie kann man nachweisen, dass eine bestimmte Sprache $L \subseteq \Sigma^*$ **nicht** regulär ist?

Ein nützliches Werkzeug dazu ist der folgende Satz 7.16, der unter dem Namen „Pumping-Lemma“ bekannt ist. Bevor wie den Satz präzise angeben, betrachten wir zunächst ein Beispiel:

Beispiel 7.15. Sei A_3 der endliche Automat aus Beispiel 7.7(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 $i \geq 0$ das Eingabewort $ba(ba)^i a$.

Der folgende Satz 7.16 beruht auf demselben Prinzip sowie der Tatsache, dass in jedem Graph auf z Knoten gilt: Jeder Weg der Länge $\geq z$ enthält einen Kreis (d.h. mindestens ein Knoten wird auf dem Weg mehr als 1-mal besucht).

Satz 7.16 (Pumping-Lemma). *Sei Σ ein endliches Alphabet. Für jede* ***reguläre*** *Sprache $L \subseteq \Sigma^*$ gibt es eine Zahl $z \in \mathbb{N}$, so dass für jedes Wort $x \in L$ der Länge $|x| \geq z$ gilt:*
Es gibt eine Zerlegung von x in Worte $u, v, w \in \Sigma^*$, so dass die folgenden Bedingungen erfüllt sind:

(1) $x = uvw$

(2) $|uv| \leq z$

(3) $|v| \geq 1$

(4) für jedes $i \in \mathbb{N}$ gilt: $uv^iw \in L$.
(d.h.: $uw \in L$, $uvw \in L$, $uvvw \in L$, $uvvvw \in L$, ...)

Beweis: Da L regulär ist, gibt es einen NFA $A = (\Sigma, Q, \delta, q_0, F)$ mit $L(A) = L$. Sei $z := |Q|$ die Anzahl der Zustände von A.

Sei nun $x \in \Sigma^*$ ein beliebiges Wort der Länge $|x| \geq z$, das in L liegt, d.h. das von A akzeptiert wird. Sei $q_0, q_1, \ldots, q_{|x|}$ die Folge von Zuständen, die A beim Verarbeiten von x durchläuft. Da $|x| \geq z = |Q|$ ist, können die Zustände $q_0, q_1, \ldots, q_z$ nicht alle verschieden sein. Daher gibt es ein $k \geq 0$ und ein $\ell \geq 1$, so dass $q_k = q_{k+\ell}$ und $k + \ell \leq z$. Wir wählen folgende Zerlegung von x in Worte $u, v, w \in \Sigma^*$:

- u besteht aus den ersten k Buchstaben von x.
- v besteht aus den nächsten ℓ Buchstaben von x.
- w besteht aus den restlichen Buchstaben von x.

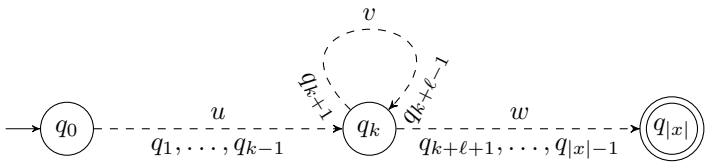
Offensichtlich gilt:

(1) $x = uvw$

(2) $|uv| = k + \ell \leq z$

(3) $|v| = \ell \geq 1$

Skizze:



Daher gilt für jedes $i \geq 0$: A akzeptiert das Eingabewort uv^iw, d.h. $uv^iw \in L$. □

Unter Verwendung des Pumping-Lemmas kann man nachweisen, dass gewisse Sprachen **nicht** regulär sind:

Beispiel 7.17. Sei $\Sigma := \{a, b\}$. Die Sprache $L := \{a^nb^n : n \in \mathbb{N}\}$ ist **nicht** regulär.
(Zum Vergleich: Gemäß Lemma 7.7(a) ist die Sprache $L_1 := \{a^nb^m : n \in \mathbb{N}, m \in \mathbb{N}\}$ regulär.)

Beweis: Durch Widerspruch.
Angenommen, L ist regulär. Dann sei $z \in \mathbb{N}$ gemäß dem Pumping-Lemma (Satz 7.16) gewählt. Betrachte das Wort $x := a^zb^z$. Klar: $x \in L$ und $|x| \geq z$. Gemäß dem Pumping-Lemma gibt es eine Zerlegung von x in Worte $u, v, w \in \{a, b\}^*$, so dass

(1) $x = uvw$

(2) $|uv| \leq z$

(3) $|v| \geq 1$

(4) f.a. $i \in \mathbb{N}$ gilt: $uv^i w \in L$.

Wegen $x = a^z b^z = uvw$ und $|uv| \leq z$ und $|v| \geq 1$ gibt es eine Zahl $\ell \in \mathbb{N}$ mit $\ell \geq 1$, so dass $v = a^\ell$. Wegen (4) gilt insbesondere für $i = 0$, dass $uw \in L$. Wegen $x = uvw = a^z b^z$ und $v = a^\ell$ gilt

$$uw \quad = \quad a^{z-\ell} b^z \quad \overset{\ell \geq 1}{\notin} \quad \{a^n b^n : n \in \mathbb{N}\} \quad = \quad L.$$

Widerspruch! □

Beispiel 7.18. 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 $z \in \mathbb{N}$ gemäß dem Pumping-Lemma (Satz 7.16) gewählt. Da es unendlich viele Primzahlen gibt, gibt es auch eine Primzahl p mit $p \geq z + 2$. Sei p solch eine Primzahl. Betrachte nun das Wort $x := a^p$. Klar: $x \in L$ und $|x| \geq z$. Gemäß dem Pumping-Lemma gibt es also eine Zerlegung von x in Worte $u, v, w \in \{a\}^*$, so dass

(1) $x = uvw$

(2) $|uv| \leq z$

(3) $|v| \geq 1$

(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

$$\left|uv^i w\right| \quad = \quad |uw| + i \cdot |v| \quad = \quad |uw| + |uw| \cdot |v| \quad = \quad |uw| \cdot (1 + |v|).$$

Wegen

$$|uw| \quad \geq \quad |w| \quad = \quad |x| - |uv| \quad \overset{|uv| \leq z}{\geq} \quad p - z \quad \overset{p \geq z+2}{\geq} \quad 2 \qquad \text{und} \qquad 1 + |v| \quad \overset{|v| \geq 1}{\geq} \quad 2$$

ist $\left|uv^i w\right|$ daher keine Primzahl, d.h. $uv^i w \notin L$. Widerspruch! □

Auf ähnliche Weise kann man auch zeigen, dass keine der folgenden Sprachen regulär ist:

- $\{a^n b^m : n, m \in \mathbb{N} \text{ mit } n \leq m\}$
- $\{ww : w \in \{a, b\}^*\}$
- $\{w : w \in \{a, b\}^*, w = w^R\}$ („Palindrome“)
- $\{w \in \{a\}^* : |w| \text{ ist eine Quadratzahl, d.h. ex. } n \in \mathbb{N} \text{ s.d. } |w| = n^2\}$

7.2 Reguläre Ausdrücke

Reguläre Ausdrücke beschreiben Mengen von Worten, die nach bestimmten Regeln bzw. „Mustern“ aufgebaut sind.

Beispiel 7.19. Die Menge aller Wörter über dem Alphabet $\{a, b\}$, deren vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a|b)^* a(a|b)$$

Definition 7.20 (Reguläre Ausdrücke – Syntax)**.** Sei Σ ein endliches Alphabet. **Die Menge aller regulären Ausdrücke über Σ** ist rekursiv wie folgt definiert:

Menge aller regulären Ausdrücke über Σ

Basisregeln:

- $\varnothing$ 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“).
  - $(R|S)$ ein regulärer Ausdruck über Σ („Vereinigung“).

Definition 7.21 (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:

- $L(\varnothing) \;:=\; \varnothing$.
- $L(\varepsilon) \;:=\; \{\varepsilon\}$.
- Für jedes $a \in \Sigma$ gilt: $L(a) \;:=\; \{a\}$.
- Ist R ein regulärer Ausdruck über Σ, so ist

$$L(R^*) \;:=\; \{\varepsilon\} \;\cup\; \{w_1 \cdots w_k : k \in \mathbb{N}_{>0},\, w_1 \in L(R), \ldots, w_k \in L(R)\}.$$

- Sind R und S reguläre Ausdrücke über Σ, so ist
  - $L\big((R \cdot S)\big) \;:=\; \{wu : w \in L(R),\, u \in L(S)\}$.
  - $L\big((R|S)\big) \;:=\; L(R) \cup L(S)$.

Notation 7.22. Zur vereinfachten Schreibweise und 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 $(R_1|R_2|R_3|R_4)$ statt $\Big(\big((R_1|R_2)|R_3\big)|R_4\Big)$ und $(R_1R_2R_3R_4)$ statt $\Big(\big((R_1R_2)R_3\big)R_4\Big)$.

- „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 benutzt werden.

Beispiel 7.23.

(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 7.24.

(a) Wir wollen einen regulären Ausdruck angeben, der die Sprache all jener Wörter über dem Alphabet $\Sigma = \{0,1,\ldots,9,/\}$ definiert, die Telefonnummern der Form

$$\mathit{Vorwahl}/\mathit{Nummer}$$

kodieren, wobei *Vorwahl* und *Nummer* nicht-leere Ziffernfolgen sind, *Vorwahl* mit einer Null beginnt und *Nummer nicht* mit einer Null beginnt. Wörter dieser Sprache sind z.B. 069/7980 und 06131/3923378, aber *nicht* die Wörter 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:

$$\big(\,\varepsilon \mid 069/\,\big)\;\; 798\;\; \big(\,\varepsilon \mid \text{-}\,\big)\;\; \big(\,0 \;\mid\; (1|\cdots|9)(0|1|\cdots|9)^*\,\big)$$

R definiert eine Sprache, die beispielsweise die Wörter 069/798-0 und 7980 enthält, aber nicht das Wort 069/798-028362.

Frage: Welche Arten von Sprachen können durch reguläre Ausdrücke beschrieben werden?

Antwort: Genau dieselben Sprachen, die durch (deterministische oder nicht-deterministische) endliche Automaten akzeptiert werden können. – Diese Sprachen werden **reguläre Sprachen** genannt.

Details: In der Veranstaltung GL-2.

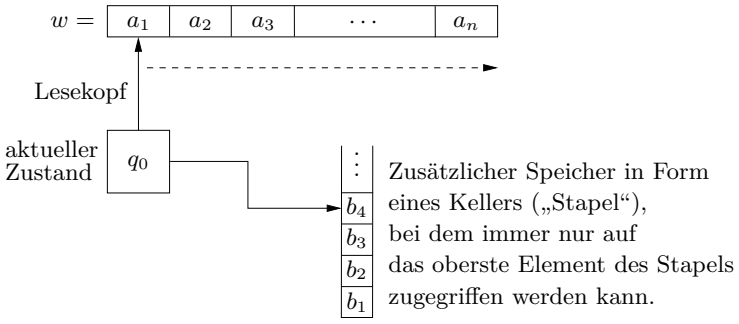
Ausblick:

Abgesehen von DFAs, NFAs und regulären Ausdrücken kann man die regulären Sprachen auch durch bestimmte Grammatiken erzeugen, so genannte reguläre Grammatiken – das sind kontextfreie Grammatiken, die von einer besonders einfachen Form sind. Generell gilt: Für jede reguläre Sprache L gibt es eine kontextfreie Grammatik, die die Sprache L erzeugt. Aber es gibt kontextfreie Grammatiken, die nicht-reguläre Sprachen erzeugen.

Analog zu DFAs und NFAs gibt es auch ein erweitertes Automatenmodell, das genau diejenigen Sprachen akzeptiert, die von kontextfreien Grammatiken erzeugt werden: so genannte **Kellerautomaten**.

Kellerautomaten

Schematische Darstellung der Verarbeitung eines Eingabeworts durch einen Kellerautomaten:



Details: In der Vorlesung GL-2. Dort werden auch allgemeinere Arten von Grammatiken betrachtet, z.B. so genannte **kontext-sensitive Grammatiken**.

kontext-sensitive Grammatiken

7.3 Petri-Netze

Petri-Netze:

- eingeführt von C. A. Petri, 1962
- formaler Kalkül zur Modellierung von Abläufen, an denen mehrere Prozesse beteiligt sind
- modelliert werden die Interaktionen zwischen Prozessen und die Effekte, die sich daraus ergeben, dass Operationen prinzipiell gleichzeitig ausgeführt werden können (Stichwort: „Nebenläufigkeit“)

Typische Anwendungsbeispiele für Petri-Netze:

zur Modellierung von

- realen oder abstrakten Automaten und Maschinen
- kommunizierenden Prozessen (z.B. in Rechnern)
- Verhalten von Software- oder Hardware-Komponenten
- Geschäftsabläufe
- Spiele (Spielregeln)
- biologische Prozesse (Bioinformatik)

Der Kalkül der Petri-Netze basiert auf bipartiten gerichteten Graphen:

- 2 Sorten von Knoten, die
  - Bedingungen oder Zustände (sog. „Stellen“) bzw.
  - Aktivitäten (sog. „Transitionen“)

  repräsentieren.
- Kanten verbinden „Aktivitäten“ mit ihren „Vorbedingungen“ und ihren „Nachbedingungen“.
- Knotenmarkierungen repräsentieren den veränderlichen „Zustand“ des Systems.

Petri-Netz

Definition 7.25 (Petri-Netz)**.** Ein **Petri-Netz** $P = (S, T, F)$ besteht aus

Stellen

- einer endlichen Menge S, den sog. **Stellen** von P,

Transitionen

- einer endlichen Menge T, den sog. **Transitionen** von P,
- einer Relation $F \subseteq (S \times T) \cup (T \times S)$, den sog. Kanten von P.

Die Mengen S und T sind disjunkt, d.h. $S \cap T = \varnothing$.

Ein Petri-Netz P bildet einen bipartiten gerichteten Graphen mit Knotenmenge $S \cup T$ und Kantenmenge F.

Graphische Darstellung:

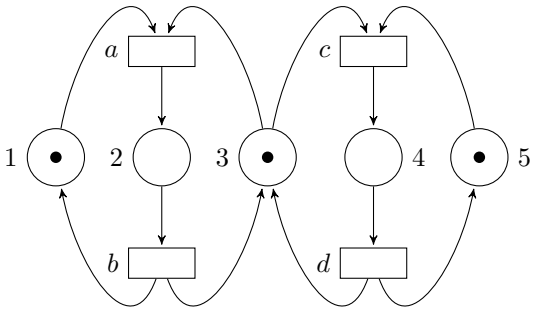
- „Stellen“, d.h. Knoten in S, werden durch Kreise dargestellt.
- „Transitionen“, d.h. Knoten in T, werden durch Rechtecke dargestellt.

Markierungsfunktion
Markierung

Definition 7.26. Der „Zustand“ eines Petri-Netzes $P = (S, T, F)$ wird durch eine **Markierungsfunktion** (kurz: **Markierung**) $M \colon S \to \mathbb{N}$, die jeder Stelle $s \in S$ eine Anzahl $M(s)$ von sog. **Marken** zuordnet, repräsentiert.

Marken

Beispiel 7.27. Graphische Darstellung eines Petri-Netzes $P = (S, T, F)$ und einer Markierung M:



die einzelnen „Marken“, die M einer Stelle $s \in S$ zuordnet, werden durch Punkte • in dem die Stelle s repräsentierenden Knoten dargestellt.

- $S = \{1, 2, 3, 4, 5\}$
- $T = \{a, b, c, d\}$
- $F = \{(1, a), (3, a), (a, 2), (2, b), (b, 1), (b, 3), (3, c), (5, c), (c, 4), (4, d), (d, 3), (d, 5)\}$
- $M\colon S \to \mathbb{N}$ mit $M(1) = M(3) = M(5) = 1$ und $M(2) = M(4) = 0$.

Definition 7.28. Sei $P = (S, T, F)$ ein Petri-Netz und sei $t \in T$ eine Transition von P. Wir setzen

- $\text{Vorbereich}(t) := \{s \in S : (s, t) \in F\}$
  $= $ „Menge aller Stellen, von denen aus eine Kante in t hineinführt“
- $\text{Nachbereich}(t) := \{s \in S : (t, s) \in F\}$
  $= $ „Menge aller Stellen, in die eine von t ausgehende Kante hinführt.“

Petri-Netze verändern ihren „Zustand“, indem Transitionen „schalten“ und dadurch die „Markierung“ des Petri-Netzes ändern. Dies wird folgendermaßen präzisiert:

Schaltregel: Sei $P = (S, T, F)$ ein Petri-Netz und sei $M\colon S \to \mathbb{N}$ eine Markierung. Das „Schalten einer Transition $t \in T$“ überführt die Markierung M in eine Markierung $M'\colon S \to \mathbb{N}$. Die Transition ***t* kann schalten**, wenn gilt:

> F.a. Stellen $s \in \text{Vorbereich}(t)$ ist $M(s) \geq 1$, d.h.: Jede Stelle s in $\text{Vorbereich}(t)$ hat mindestens eine Marke.

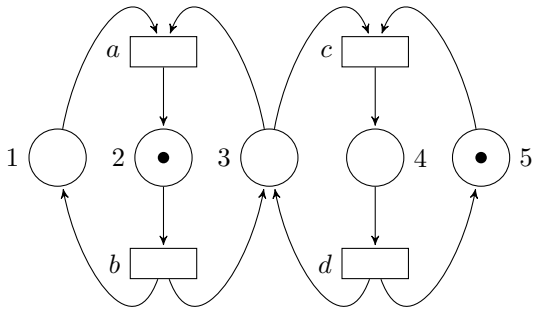
Wenn die Transition t schaltet, so gilt für die sog. **Nachfolgemarkierung *M'***: f.a. $s \in S$ Nachfolgemarkierung

$$M'(s) := \begin{cases} M(s) - 1, & \text{falls } s \in \text{Vorbereich}(t) \setminus \text{Nachbereich}(t) \\ M(s) + 1, & \text{falls } s \in \text{Nachbereich}(t) \setminus \text{Vorbereich}(t) \\ M(s), & \text{sonst.} \end{cases}$$

D.h.: Das „Schalten von Transition t“ bewirkt, dass in jeder Stelle in $\text{Vorbereich}(t)$ eine Marke entfernt wird und dass in jeder Stelle in $\text{Nachbereich}(t)$ eine Marke hinzugefügt wird. Wenn mehrere Transitionen schalten können, so wird eine davon „nicht-deterministisch ausgewählt“.

In jedem Schritt schaltet genau eine Transition. Durch schrittweises Schalten von Transitionen wird der Ablauf von Prozessen modelliert.

Beispiel 7.29. Seien $P = (S, T, F)$ und $M\colon S \to \mathbb{N}$ das Petri-Netz und die Markierung aus Beispiel 7.27. Bei der angegebenen Markierung M können die Transitionen a und c schalten. Wenn wir a schalten lassen, ergibt sich als Nachfolgemarkierung die Markierung $M'\colon S \to \mathbb{N}$ mit $M(1) = 0$, $M'(3) = 0$, $M'(2) = 1$, $M'(4) = 0$, $M'(5) = 1$, d.h.:



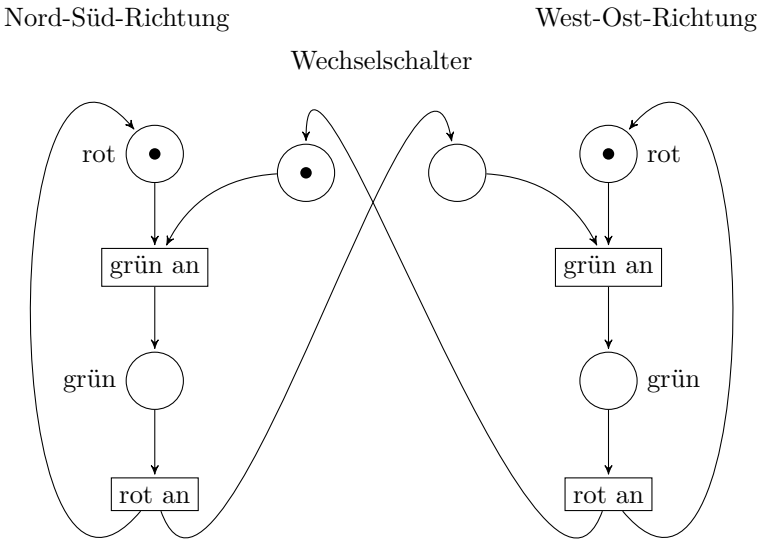
Als nächstes kann Transition c **nicht** schalten, da die Stelle 3 keine Marke trägt. Die einzige Transition, die jetzt schalten kann, ist Transition b, deren Schalten bewirkt, dass die Marke bei 2 verschwindet und stattdessen Marken bei 1 und 3 erzeugt werden. Nach dem Schalten von b ist das System also wieder in seinem „ursprünglichen Zustand“, d.h. es trägt die Markierung M.

Insgesamt gilt: Das Petri-Netz P aus Beispiel 7.27 (zusammen mit der Startmarkierung M) modelliert zwei zyklisch ablaufende Prozesse. Die Stelle 3 „synchronisiert“ die beiden Prozesse, so dass sich nie gleichzeitig in beiden Stellen 2 und 4 eine Marke befinden kann. Auf diese Weise könnte man z.B. beschreiben, wie Autos eine 1-spurige Brücke von 2 Seiten überqueren, so dass sich immer nur 1 Auto auf der Brücke befindet.

Beispiel 7.30. Modellierung einer Ampel an einer Kreuzung

- 2 sich zyklisch wiederholende Prozesse:
  - „grün“ in Nord-Süd-Richtung
  - „grün“ in West-Ost-Richtung
- die beiden Prozesse sollen sich immer abwechseln

Petri-Netz inklusive Anfangs-Markierung:

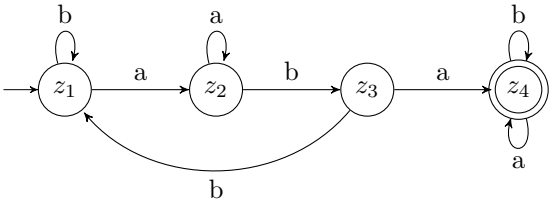


Die beiden Stellen „Wechselschalter“ koppelt die Prozesse, so dass abwechselnd in West-Ost- bzw. in Nord-Süd-Richtung die Ampel „grün“ ist.

7.4 Übungsaufgaben zu Kapitel 7

Aufgabe 7.1.

(a) 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 Wörter werden von A akzeptiert, welche nicht?

- *bbaabba*
- *abbaaababbba*
- *aabbaab*

Begründen Sie Ihre Antworten.

(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.

(b) Geben Sie die graphische Darstellung eines nicht-deterministischen endlichen Automaten an, der genau diejenigen Wörter über dem Alphabet $\{a, b\}$ akzeptiert, deren drittletzter Buchstabe ein a ist.

Aufgabe 7.2. Von einem Computervirus ist bekannt, dass in den vom ihm befallenen Dateien mindestens eine der folgenden Bitfolgen auftritt: 101 bzw. 111.

(a) Modellieren Sie potenziell befallene Dateien durch einen regulären Ausdruck. Der Ausdruck soll also die Sprache aller Wörter beschreiben, in denen 101 oder 111 als Teilwort vorkommt.

(b) Geben Sie die graphische Darstellung eines nicht-deterministischen endlichen Automaten an, der potenziell befallene Dateien erkennt. Der Automat soll also genau diejenigen Wörter akzeptieren, in denen 101 oder 111 als Teilwort vorkommt.

Aufgabe 7.3.

1. Geben Sie einen regulären Ausdruck an, der die Sprache der Wörter über dem Alphabet $\Sigma = \{0, 1, \ldots, 9\}$ definiert, die natürliche Zahlen ohne führende Nullen kodieren. Wörter aus der Sprache sind z.B. 42, 0, 1, aber nicht 0042 oder das leere Wort.

2. Sei R der folgende reguläre Ausdruck:

$$\Big(0 \,|\, \big((1|\cdots|9)(0|1|\cdots|9)^*\big)\Big)\ \Big(\varepsilon \,|\, \big(,(0|1|\cdots|9)(0|1|\cdots|9)\big)\Big)\ €$$

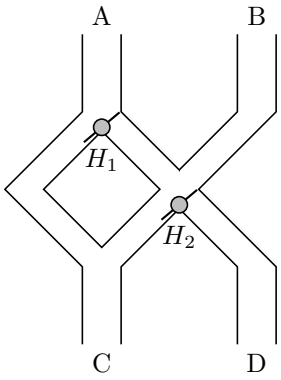
(i) Welche der folgenden Wörter liegen in der von R definierten Sprache $L(R)$, welche nicht?

1,99€	,69€	1,9€
01,99€	1€	1,09

Geben Sie jeweils eine kurze Begründung für Ihre Antwort.

(ii) Beschreiben Sie umgangssprachlich, welche Sprache $L(R)$ von R definiert wird.

Aufgabe 7.4. 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.



Ziel dieser Aufgabe ist, dieses Spiel als deterministischen endlichen Automaten zu modellieren, der als Eingabe ein Wort über dem Alphabet $\{a, b\}$ erhält. Diese Eingaben entsprechen dem Fallenlassen von Murmeln bei A oder B. Zum Beispiel entspricht die Eingabe *aba* dem Fallenlassen von 3 Murmeln, von denen die erste und dritte Murmel bei A und die zweite Murmel bei B fallengelassen wird. In diesem Fall würden die ersten beiden Murmeln an der Öffnung C und die letzte Murmel an der Öffnung D herausfallen. Der Automat soll genau dann akzeptieren, wenn die letzte Murmel durch die Öffnung D rollt. Im obigen Beispiel soll der Automat also die Eingabe *aba* akzeptieren. Der Startzustand soll dem Zustand in der Abbildung entsprechen, d.h. im Startzustand fällt die nächste Murmel, die auf H_1 oder H_2 trifft, nach links.

1. Geben Sie die graphische Darstellung des Automaten an.

   Hinweis: Verwenden Sie als Zustandsmenge die Menge der Tripel $(h_1, h_2, \ddot{o}) \in \{\mathrm{L}, \mathrm{R}\}^2 \times \{\mathrm{C}, \mathrm{D}\}$, wobei h_i angibt, ob die nächste Murmel, die an Hebel H_i ankommt, nach links ($h_i = \mathrm{L}$) oder rechts ($h_i = \mathrm{R}$) fällt, und $\ddot{o}$ angibt, ob die zuvor fallengelassene Murmel bei C ($\ddot{o} = \mathrm{C}$) oder D ($\ddot{o} = \mathrm{D}$) herausgerollt ist.

2. Welche der folgenden Eingaben wird akzeptiert, welche nicht?

   abbab *aababba* *baba*

8 Eine Fallstudie

In diesem Kapitel steht ein konkretes Anwendungsbeispiel im Vordergrund. Seine Strukturen, Eigenschaften etc. werden mit verschiedenen Kalkülen modelliert (die unterschiedlichen Kalküle werden eingesetzt, um unterschiedliche Aspekte des Anwendungsbeispiels zu beschreiben).

8.1 Aufgabenstellung: Autowerkstatt

Ziel: Modelliere die Auftragsabwicklung in einer Autowerkstatt.

Genauer:

- Datenbank entwerfen
- Abläufe analysieren und verbessern

8.2 Datenbank-Entwurf: Autowerkstatt

Kurzbeschreibung der „Informationsstruktur“:

(1) **Kunde** : hat einen Namen, besitzt Kraftfahrzeug(e) (kurz: KFZ), erteilt Aufträge

(2) **Auftrag**: hat ein Eingangsdatum, betrifft ein KFZ, wird von Mechaniker(n) bearbeitet, benötigt Ersatzteile bestimmter Arten und Mengen (i.S.v. „Anzahlen“)

(3) **KFZ**: hat Fahrgestellnummer und Baujahr, ist entweder ein PKW oder ein Motorrad; zu PKWs interessiert ihre Farbe, zu Motorrädern der Tuningsatz.

(4) **Typ**: jedes KFZ hat einen Typ; jeder Mechaniker ist für einige Typen ausgebildet; Ersatzteile sind für bestimmte Typen verwendbar.

„Informationsstruktur“ als ER-Modell

Zentrale Entity-Typen:

Kunde, Auftrag, KFZ, KFZ-Typ

Relationen-Typen:

- besitzt (Kunde besitzt KFZ)
- erteilt (Kunde erteilt Auftrag)
- betrifft (Auftrag betrifft KFZ)
- hat Typ (KFZ hat KFZ-Typ)

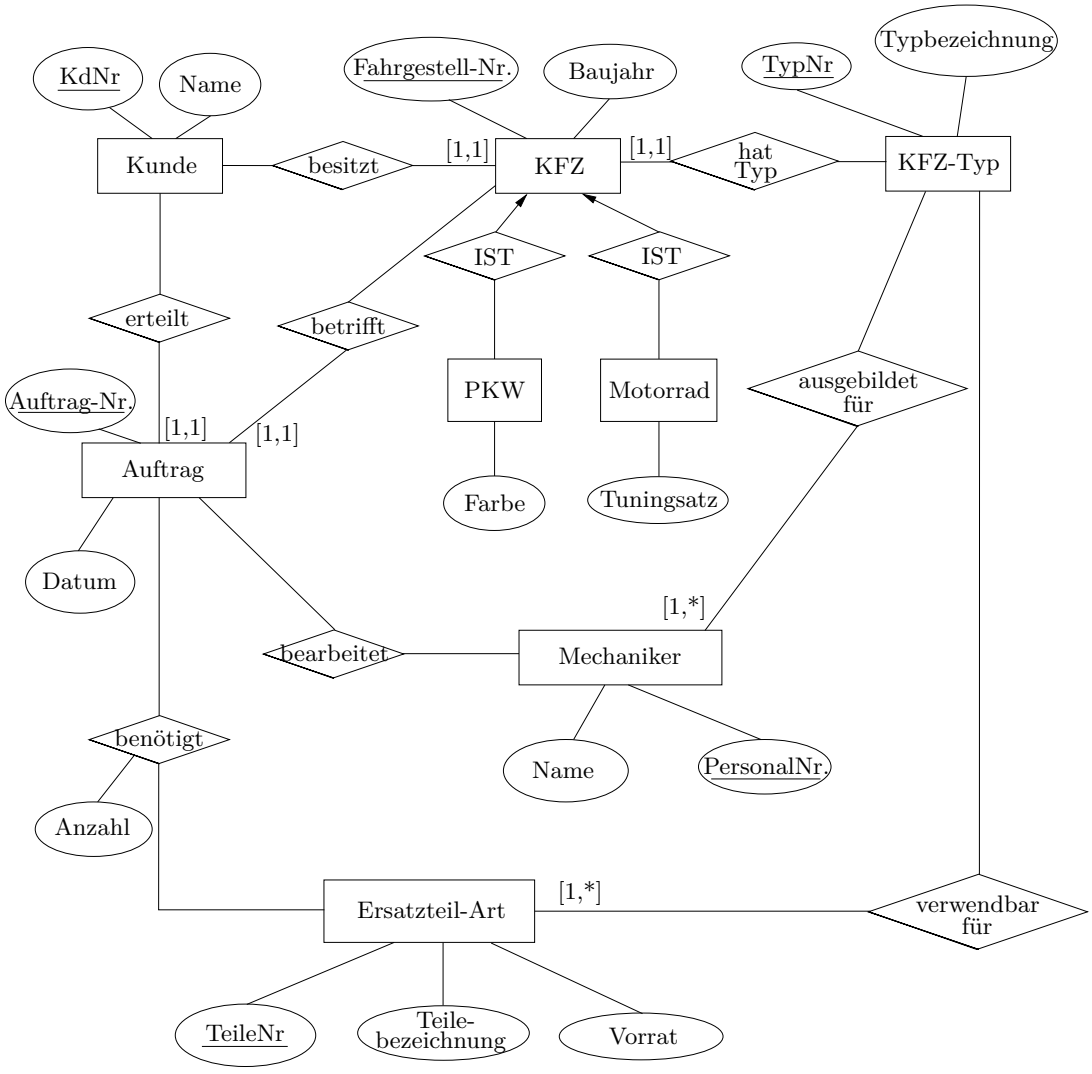
- ausgebildet für (Mechaniker ist ausgebildet für KFZ-Typen)
  (⇒ Entity-Typ „Mechaniker“ einführen)

auch nötig: Relationen zur Modellierung davon

- welche Ersatzteile ein Auftrag benötigt
- welche Ersatzteile für welchen KFZ-Typ geeignet sind
- welcher Mechaniker welchen Auftrag bearbeitet.

auch noch nötig: Unterscheidung von KFZ in PKW und Motorräder.

ER-Modell: Autowerkstatt



Beachte: Durch Angabe von Kardinalitäten haben wir einige Entscheidungen getroffen:

- Jedes KFZ hat genau einen KFZ-Typ.
- Jedes KFZ hat genau einen Besitzer.
- Jeder Auftrag betrifft genau ein KFZ.
- Jeder Mechaniker ist für mindestens einen KFZ-Typ ausgebildet.
- Jede Ersatzteil-Art ist für mindestens einen KFZ-Typ verwendbar.

8.3 Abläufe bei der Auftragserteilung

Eine Untersuchung der Geschäftsabläufe in der Autowerkstatt ergibt, dass jeder Auftrag folgende Stationen durchläuft:

(1) Der Auftrag wird erteilt.

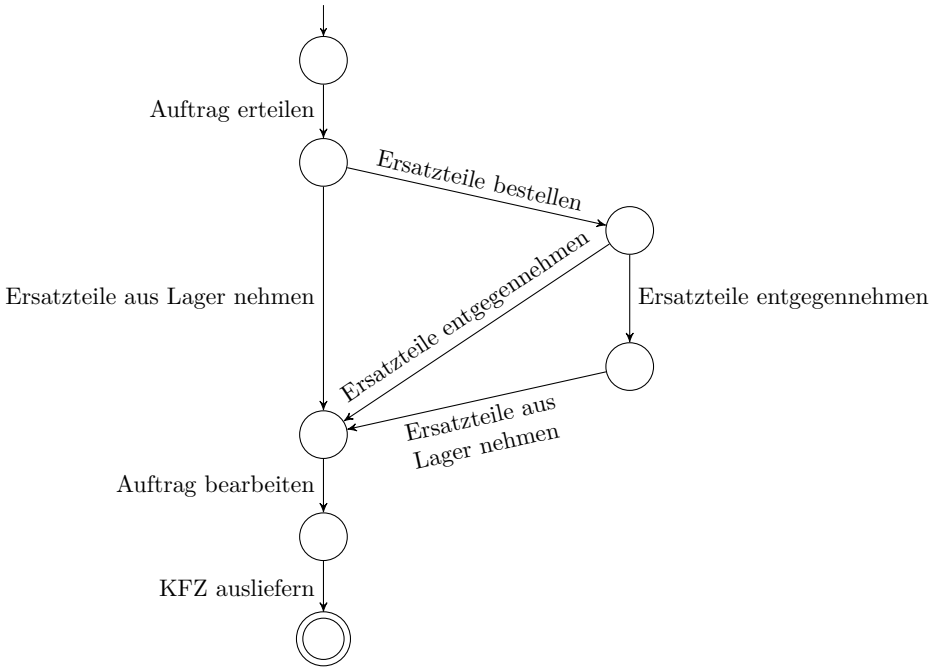
(2) Fehlende Ersatzteile werden bestellt und nach dem Eintreffen entgegengenommen.

(3) Vorhandene Ersatzteile werden aus dem Lager genommen.

(4) Der Auftrag wird von einem Mechaniker bearbeitet.

(5) Das KFZ wird dem Kunden ausgeliefert.

Modellierung dieser Abläufe als Transitionssystem (endlicher Automat)



Einschränkungen dieses Modells:

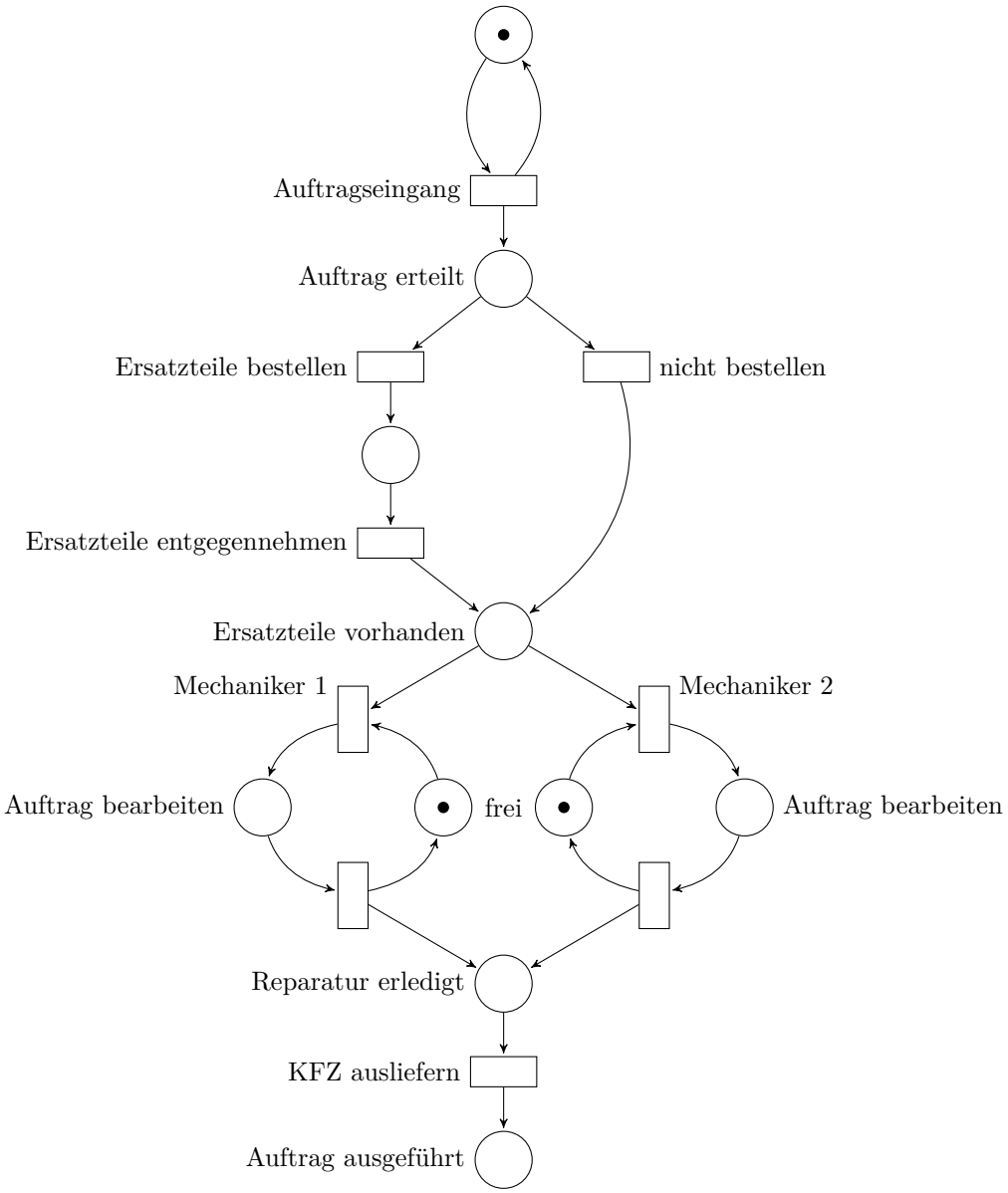
- Die Abläufe in der Werkstatt werden uns aus der Sicht eines einzelnen Auftrags beschrieben.

- Das Modell spricht nur über die „Ersatzteile insgesamt“, aber nicht über ihre Art und Anzahl.
- Aktionsfolgen, bei denen mehrere Aufträge von mehreren Mechanikern bearbeitet werden, können durch dieses Modell nicht beschrieben werden.

8.3.1 Modellierung der Auftragsbearbeitung durch ein Petri-Netz

Ziel: Modelliere, wie mehrere Aufträge nebenläufig von 2 miteinander um Aufträge konkurrierenden Mechanikern bearbeitet werden.

Petri-Netz:



Kapitel 9: Markov-Ketten als Grundlage der Funktionsweise von Suchmaschinen im Internet

Situation:

Eingabe: eine Anfrage, bestehend aus einem oder mehreren Stichworten

Ziel: Bestimme diejenigen Websiten, deren Inhalt hinsichtlich der Anfrage-Stichworte am "informativsten" ist.

Probleme:

- Es gibt sehr viele Websiten (in 2005: mehr als 8 Milliarden Stück)
- Ständig kommen neue hinzu
- Viele Websiten werden (fast) täglich aktualisiert; viele werden nach einiger Zeit auch wieder gelöscht.
- Suchanfragen müssen in Echtzeit beantwortet werden
- Welches sind die "informativsten" Websiten?

9.1 Die Architektur von Suchmaschinen

Herausforderung:

Für einen sich rasant ändernden Suchraum gigantischer Größe müssen Anfragen ohne merkliche Reaktionszeit beantwortet werden.

Um dies zu gewährleisten, nützen Suchmaschinen folgende Komponenten:

- Der Crawler durchforstet das Internet, um neue oder veränderte Webseiten zu bestimmen.
- Abspeichern der wichtigsten Informationen: Die vom Crawler gefundene Information muss aufbereitet und gespeichert werden
- Indizierung: Die Datenstruktur der gespeicherten Informationen muss in Echtzeit alle (relevanten) Webseiten bestimmen, die die Anfrage-Stichworte enthalten.

- Bewertung der Webseiten: Die ausgewählten Webseiten müssen im Hinblick auf ihren Informationsgehalt bewertet werden.

Details zum Thema "Indizierung":

- Es werden alle Worte einer jeden Webseite erfasst.
- Fundamentale Datenstruktur: der invertierte Index, der zu jedem Stichwort s alle Webseiten bestimmt, die s enthalten.
  Oft liefert der invertierte Index noch Zusatzinformationen, die die Wichtigkeit des Stichworts innerhalb der Webseite beschreiben (etwa: Häufigkeit des Stichworts, seine Schriftgröße, das Vorkommen des Stichworts in Beschriftungen von Links auf die Webseite).
  Insgesamt: Der invertierte Index ist Text-basiert.
- Eine weitere wichtige Datenstruktur: der Link-Index, mit dem die Graph-Struktur des Internets modelliert wird.

Ansatz: Jede Webseite wird durch einen Knoten repräsentiert.
Eine Kante von Knoten i zu Knoten j entspricht einem Link von Webseite i auf Webseite j.

Der Link-Index (oder: Webgraph) wird üblicherweise als Adjazenzliste gespeichert

Details zum Thema "Bewertung der Webseiten":

- Ziel: Bestimme die für die gegebenen Anfrage-Stichworte informativsten Webseiten, d.h. sortiere die Webseiten, die die Anfrage-Stichworte enthalten, nach ihrer "Relevanz"

Dabei werden u.a. folgende Kriterien berücksichtigt:

1) Häufigkeit und Positionierung der Suchbegriffe auf der jeweiligen Webseite sowie in der Beschriftung von Links, die auf diese Webseite verweisen.
2) Die grundlegende Bedeutung einer Webseite. Bei Google wird dies durch den so genannten Page-Rank der Webseite modelliert.

Ein etwas anderer Ansatz zur Bewertung der grundlegenden Bedeutung einer Webseite ist der "Hubs and Authorities"-Ansatz von Kleinberg.

Beide Ansätze betrachten allein den Webgraphen (d.h. die Link-Struktur des Internets, nicht den textuellen Inhalt der Webseiten) und gehen von der folgenden Annahme aus:

Wenn eine Webseite i einen Link auf 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.

Beide Ansätze versuchen, die "relative Wertschätzung" zwischen Webseiten in eine "absolute Relevanz" der Webseiten umzurechnen.

Im Rest von Kapitel 9 werden wir uns etwas genauer ansehen, wie die "absolute Relevanz" einer Webseite modelliert und berechnet werden kann.

9.2 Page-Rank, Zufalls-Surfer und Markov-Ketten

Sei $G = (V, E)$ mit $V = \{1, 2, \ldots, n\}$ und $E \subseteq V \times V$ der Webgraph.
D.h.: Knoten von G repräsentieren Websiten, und jede Kante $(i, j) \in E$ modelliert einen Link von Website i auf Website j.

Für jeden Knoten $i \in V$ sei

$$a_i := \text{Aus-Grad}_G(i)$$

der Ausgangsgrad von i in G
D.h. a_i ist die Anzahl der Hyperlinks, die von Website i auf andere Websiten verweisen.

Die "grundlegende Bedeutung" einer Website i wird im Folgenden durch eine Zahl PR_i (dem so genannten Page-Rank von i) modelliert.
Der Wert PR_i soll die Qualität von Website i widerspiegeln (je höher, desto besser) und wird durch einen "Peer-Review" bestimmt:

- Der Page-Rank PR_j von Webseite j ist hoch, wenn viele Webseiten i mit hohem Page-Rank PR_i auf die Seite j verweisen.
- Eine Webseite i mit a_i ausgehenden Links "vererbt" ihren Page-Rank dabei anteilig an alle Webseiten j mit $(i,j) \in E$ im den Anteil $\frac{PR_i}{a_i}$ weiter.

In dieser Sichtweise müsste also f.a. $j \in V$ gelten:

$$PR_j = \sum_{\substack{i \in V: \\ (i,j) \in E}} \frac{PR_i}{a_i} . \qquad (*)$$

Aus praktischen Gründen (mehr dazu: später) wird die Vererbung von PR_i auf die "Nachfolgeseiten" j mit $(i,j) \in E$ meistens um einen "Dämpfungsfaktor" d (mit $0 \leq d \leq 1$) abgeschwächt und die Werte werden so normiert, dass gilt: $PR_1, \ldots, PR_n \geq 0$ und $\sum_{i \in V} PR_i = 1$:

Definition 9.1 (Page-Rank)

Sei d eine reelle Zahl mit $0 \leq d \leq 1$;
d wird im Folgenden auch Dämpfungsfaktor genannt.

Sei $G = (V, E)$ mit $V = \{1, \dots, n\}$ und $E \subseteq V \times V$ der Webgraph und sei f.a. $i \in V$ $a_i := \text{Aus-Grad}_G(i)$.

Ein Tupel $PR = (PR_1, \dots, PR_n)$ von reellen Zahlen > 0 hat die Page-Rank-Eigenschaft (bzgl. Dämpfungsfaktor d), wenn gilt:

(a) $\sum_{i \in V} PR_i = 1$ und

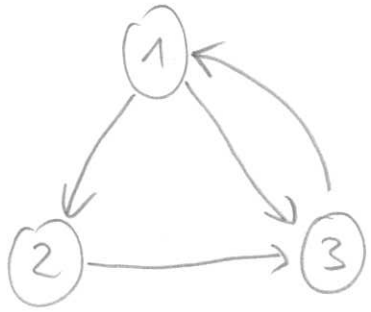
(b) f.a. $j \in V$ gilt:

$$PR_j = \frac{1-d}{n} + d \cdot \sum_{\substack{i \in V: \\ (i,j) \in E}} \frac{PR_i}{a_i} \qquad (**)$$

(Beachte: Für $d = 1$ erhält man gerade die Gleichung $(*)$;
für $d = 0$ ist $PR_1 = PR_2 = \dots = PR_n = \frac{1}{n}$.)

Beispiel 9.2

Zur Veranschaulichung der Page-Rank-Eigenschaft betrachten wir den Dämpfungsfaktor $d := \frac{1}{2}$ und den folgenden (sehr kleinen) Webgraphen, der aus nur drei Dokumenten besteht:



Wir suchen ein Tupel $PR = (PR_1, PR_2, PR_3)$ von reellen Zahlen ≥ 0, das die Page-Rank-Eigenschaft bzgl $d = \frac{1}{2}$ hat, d.h. es gilt:

0) $PR_1 + PR_2 + PR_3 = 1$

1) $PR_1 = \frac{1}{2 \cdot 3} + \frac{1}{2} \cdot PR_3$

2) $PR_2 = \frac{1}{6} + \frac{1}{2 \cdot 2} PR_1$

3) $PR_3 = \frac{1}{6} + \frac{1}{2 \cdot 2} \cdot PR_1 + \frac{1}{2} \cdot PR_2$

Eine Lösung dieses linearen Gleichungssystems (z.B. mittels Gauß-Elimination) liefert:

$PR_1 = \frac{14}{39}$, $PR_2 = \frac{10}{39}$, $PR_3 = \frac{15}{39}$.

Eine etwas andere Schreibweise der Bedingung (b) in Definition 9.1 zeigt, dass die Tupel $PR = (PR_1, \dots, PR_n)$, die die Page-Rank-Eigenschaft (bzgl d) besitzen, gerade die Eigenvektoren zum Eigenwert 1 der folgenden Matrix

$$P = \left(p_{i,j}\right)_{i,j=1,\dots,n} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix}$$

sind: F.a. $i,j \in \{1,\dots,n\}$ sei

$$p_{i,j} := \begin{cases} \frac{1-d}{n} + \frac{d}{a_i} & , \text{falls } (i,j) \in E \\ \frac{1-d}{n} & , \text{falls } (i,j) \notin E \end{cases}$$

Beobachtung 9.3: Für jedes Tupel $X = (x_1, \dots, x_n)$ von reellen Zahlen mit $\sum_{i=1}^{n} x_i = 1$ gilt:

$$X \cdot P = \left(\frac{1-d}{n} + d \cdot \sum_{\substack{i \in V: \\ (i,j) \in E}} \frac{x_i}{a_i} \right)_{j=1,\dots,n}$$

Beweis:

$$x \cdot P = (x_1, \dots, x_n) \cdot \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix}.$$

Die j-te Komponente von $x \cdot P$ ist also:

$$\sum_{i=1}^{n} x_i \cdot p_{i,j} \underset{\text{Def } p_{i,j}}{=} \sum_{i=1}^{n} x_i \cdot \frac{1-d}{n} + \sum_{\substack{i \in V: \\ (i,j) \in E}} x_i \cdot \frac{d}{a_i}$$

$$= \frac{1-d}{n} \cdot \sum_{i=1}^{n} x_i + d \cdot \sum_{\substack{i \in V: \\ (i,j) \in E}} \frac{x_i}{a_i}$$

$$\underset{\sum_{i=1}^{n} x_i = 1}{=} \frac{1-d}{n} + d \cdot \sum_{\substack{i \in V \\ (i,j) \in E}} \frac{x_i}{a_i}$$

□

Folgerung 9.4:

Ein Tupel $PR = (PR_1, \dots, PR_n)$ von reellen Zahlen ≥ 0 hat genau dann die Page-Rank-Eigenschaft (bzgl d), wenn gilt:

(a) $\sum_{i=1}^{n} PR_i = 1$ und

(b) $PR \cdot P = PR$, (zum Eigenwert 1)

d.h. PR ist ein Eigenvektor der Matrix P.

(bzw. eine Lösung des linearen Gleichungssystems $PR \cdot \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - P\right) = 0$

Um den Page-Rank der einzelnen Websiten zu berechnen, müssen wir also "nur" einen Eigenvektor zum Eigenwert 1 der Matrix P bestimmen

(z.B. durch Lösen des linearen Gleichungssystems

$$PR \cdot \left(\begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix} - P \right) = 0 \quad).$$

Probleme:

- Das ist sehr aufwändig, da der Webgraph sehr groß ist.
- Ist "der" Eigenvektor (zum Eigenwert 1) überhaupt eindeutig bestimmt? Muss es einen solchen geben? Kann es evtl. verschiedene solche Eigenvektoren von P geben?

Um diese Probleme zu bewältigen, hilft uns die Theorie der Markov-Ketten sowie die folgende Sichtweise auf die Matrix P:

Der Zufalls-Surfer (Random-Surfer):

Ein Zufalls-Surfer beginnt auf einer beliebigen Websete und verfolgt beliebige Links, ohne dabei auf Inhalte zu achten.

Der Eintrag $P_{i,j}$ in der Matrix P gibt dabei die Wahrscheinlichkeit an, mit der der Zufalls-Surfer in einem Schritt von Seite i zu Seite j wechselt. Die Wahl

$$P_{i,j} = \begin{cases} \frac{1-d}{n} + \frac{d}{a_i}, & \text{falls } (i,j) \in E \\ \frac{1-d}{n}, & \text{falls } (i,j) \notin E \end{cases}$$

bedeutet dabei: Wenn der Zufalls-Surfer auf Webseite i ist, so wählt er

- mit Wahrscheinlichkeit d einen Link, der von Seite i ausgeht — und jeder der a_i ausgehenden Links wird mit derselben Wahrscheinlichkeit, d/a_i, ausgewählt
- mit Wahrscheinlichkeit (1-d) eine beliebige Webseite im Webgraphen — und jede der n Websiten wird mit derselben Wahrscheinlichkeit, $\frac{1-d}{n}$, ausgewählt.

Beachte: Die Werte $p_{i,j}$ können wir tatsächlich als Wahrscheinlichkeiten interpretieren, da gilt: $p_{i,j} \geq 0$ (f.a. $i,j \in \{1,\dots,n\}$) und

f.a. $i \in \{1,\dots,n\}$ gilt: $\sum_{j=1}^{n} p_{i,j} = 1$

(denn: $\sum_{j=1}^{n} p_{i,j} = \sum_{j=1}^{n} \frac{1-d}{n} + \sum_{\substack{j \in V: \\ (i,j) \in E}} \frac{d}{a_i} = (1-d) + a_i \cdot \frac{d}{a_i} = 1$).

Mit dieser Sichtweise auf die Matrix P gilt für den "modifizierten Webgraphen" $G' = (V, E')$ mit $V = \{1,\dots,n\}$ und $E' = V \times V$, dass (G', P) eine Markov-kette im folgenden Sinn ist:

(Schreibweisen: Markov, Markow, Markoff)

Definition 9.5 (Markov-Kette)

Eine Markov-Kette $(\hat{G}, \hat{P})$ besteht aus einem gerichteten Graphen $\hat{G} = (\hat{V}, \hat{E})$ mit $\hat{V} = \{1, \dots, n\}$ (für ein geeignetes $n \in \mathbb{N}$) und einer Matrix

$$\hat{P} = (\hat{p}_{i,j})_{i,j=1\dots n} = \begin{pmatrix} \hat{p}_{1,1} & \cdots & \hat{p}_{1,n} \\ \vdots & & \\ \hat{p}_{n,1} & \cdots & \hat{p}_{n,n} \end{pmatrix}$$

so dass gilt:

(1) f.a. $i,j \in \{1,\dots,n\}$ mit $(i,j) \notin \hat{E}$ ist $\hat{p}_{i,j} = 0$

und

(2) für jede Zeile $i \in \{1,\dots,n\}$ gilt

$$\sum_{j=1}^{n} p_{i,j} = 1$$

Die Matrix $\hat{P}$ heißt Übergangsmatrix.

Falls zusätzlich gilt

(3) f.a. $i,j \in \{1,\dots,n\}$ ist $\hat{p}_{i,j} \geq 0$,

so heißt $\hat{P}$ auch stochastische Matrix.

Der Eintrag $\hat{p}_{i,j}$ kann dann als Wahrscheinlichkeit

aufgefasst werden, mit der ein
"Zufalls-Surfer im Graphen $\hat{G}$" in einem
Schritt von Knoten i zu Knoten j springt.

Definition 9.6 (stationäre Verteilung)

Sei $(\hat{G}, \hat{P})$ eine Markov-Kette mit $\hat{G} = (\hat{V}, \hat{E})$ und $\hat{V} = \{1, \dots, n\}$.

(a) Eine Verteilung auf $\hat{V}$ ist ein Vektor $x = (x_1, \dots, x_n)$ von reellen Zahlen ≥ 0 mit $\sum_{i=1}^{n} x_i = 1$.

(x_i kann dabei als Wahrscheinlichkeit dafür aufgefasst werden, dass der "Zufalls-Surfer auf $\hat{G}$" sich zu Beginn auf Knoten i befindet).

(b) Eine Verteilung x auf $\hat{V}$ heißt Stationäre Verteilung, falls gilt: $x \cdot P = x$
(d.h. x ist ein Eigenvektor zum Eigenwert 1 von P)

Beachte: Gemäß Folgerung 9.4 gilt für den "modifizierten Webgraphen G'" und die Matrix P folgendes:

Ein Tupel $PR = (PR_1, \ldots, PR_n)$ hat genau dann die Page-Rank-Eigenschaft (bzgl d), wenn PR eine stationäre Verteilung (für die Markov-Kette (G', P)) ist.

Die Theorie der Markov-Ketten wurde in der Literatur gut untersucht. Insbesondere ist ein nützliches Kriterium bekannt, das garantiert, dass eine stationäre Verteilung existiert, eindeutig bestimmt ist und sehr effizient näherungsweise berechnet werden kann. Dies ist bei so genannten ergodischen Markov-Ketten (siehe Definition 9.7) der Fall.

Notation:

Ist $\hat{P}$ eine $n \times n$-Matrix und $k \geq 1$ eine natürliche Zahl, so ist

$\hat{P}^k := \underbrace{\hat{P} \cdot \ldots \cdot \hat{P}}_{k\text{-mal}}$ die $n \times n$-Matrix, die aus dem Matrix-Produkt von k Kopien von $\hat{P}$ entsteht.

Wenn $\hat{P}$ eine stochastische Matrix ist, so können wir den Eintrag $(\hat{P}^k)_{i,j}$ in Zeile i und Spalte j der Matrix $\hat{P}^k$ auffassen als die Wahrscheinlichkeit, dass der "Zufalls-Surfer auf $\hat{G}$" innerhalb von genau k Schritten von Knoten i zu Knoten j gelangt.

Definition 9.7 (Ergodische Markov-ketten)

Eine Markov-Kette $(\hat{G}, \hat{P})$ mit $\hat{G} = (\hat{V}, \hat{E})$ und $\hat{V} = \{1, \ldots, n\}$ heißt <u>ergodisch</u>, wenn f.a. $i_1, i_2 \in \{1, \ldots, n\}$ und alle $j \in \{1, \ldots, n\}$ gilt:
Die Grenzwerte $\lim_{k \to \infty} (\hat{P}^k)_{i_1,j}$ und $\lim_{k \to \infty} (\hat{P}^k)_{i_2,j}$ existieren und es gilt: $\lim_{k \to \infty} (\hat{P}^k)_{i_1,j} = \lim_{k \to \infty} (\hat{P}^k)_{i_2,j} > 0$.

Beachte:

Ist $(\hat{G}, \hat{P})$ ergodisch, so gilt:

1) Die Matrix

$$\hat{P}^\infty := \lim_{k\to\infty}\left(\hat{P}^k\right) := \left(\lim_{k\to\infty}\left(\hat{P}^k\right)_{ij}\right)_{i,j=1\dots n}$$

ist wohldefiniert (da die Grenzwerte existieren),

und

2) alle Zeilen in $\hat{P}^\infty$ sind identisch.

Notation: $\pi^\infty := (\pi_1^\infty, \dots, \pi_n^\infty)$ sei die 1. Zeile von $\hat{P}^\infty$.

Somit ist $\hat{P}^\infty = \begin{pmatrix} \pi^\infty \\ \vdots \\ \pi^\infty \end{pmatrix} = \begin{pmatrix} \pi_1^\infty & \cdots & \pi_n^\infty \\ \pi_1^\infty & \cdots & \pi_n^\infty \\ \vdots & & \vdots \\ \pi_1^\infty & \cdots & \pi_n^\infty \end{pmatrix}$, und

für jede Verteilung $X = (X_1, \dots, X_n)$ gilt: $X \cdot \hat{P}^\infty = \pi^\infty$.

Daher gilt: Wenn der Zufalls-Surfer auf $\hat{G}$ seinen Startknoten gemäß einer beliebigen Anfangsverteilung $X = (X_1, \dots, X_n)$ wählt und hinreichend viele zufällige Schritte macht, so ist die Wahrscheinlichkeit, bei Knoten $j \in \hat{V}$ zu landen, beliebig nah bei π_j^∞.

Die Wahl des Anfangsknotens ist für einen hinreichend langen "Random Walk" also ohne Belang, und die "Grenzverteilung" π^∞ eines ergodischen Markov-Kette "vergisst" den Startpunkt eines "Random Walk".
Daher gilt folgender Satz:

Satz 9.8

Jede ergodische Markov-Kette $(\vec{G}, \hat{P})$ besitzt eine eindeutig bestimmte stationäre Verteilung χ, und es gilt $\chi = \pi^\infty$.

Beobachtung 9.9:

Ist G' der "modifizierte Webgraph" und P die Übergangsmatrix für Dämpfungsfaktor d $(0 \leq d \leq 1)$, so gilt:

(a) Falls $d < 1$, so ist die Markov-Kette (G', P) ergodisch (hier ohne Beweis), und gemäß Satz 9.8 ist π^∞ das eindeutig bestimmte Tupel, das die Page-Rank-Eigenschaft bzgl d hat.

(b) Falls $d = 1$ ist, so ist (G', P) nicht unbedingt ergodisch, denn der Zufalls-Surfer wird z.B. auf Webseiten, aus denen keine Hyperlinks herausführen gefangen.

(Das ist der auf Seite 9.7 erwähnte "praktische Grund" wegen dem der Dämpfungsfaktor d eingeführt wurde).

9.3 Die effiziente Berechnung des Page-Rank

Wir wählen einen Dämpfungsfaktor $d < 1$ und nützen Beobachtung 9.9 (a).

Als Anfangsverteilung für den Zufalls-Surfer wählen wir die Gleichverteilung $x^{(0)} := (\underbrace{\frac{1}{n}, \dots, \frac{1}{n}}_{n\text{-mal}})$

Für $k = 1, 2, 3, \dots$ berechnen wir nach und nach die Vektoren $x^{(k)} := x^{(k-1)} \cdot P$.

Somit gilt f.a. $k \in \mathbb{N}$: $x^{(k)} = x^{(0)} \cdot P^k$.

Da (G', P) ergodisch ist, wissen wir, dass

$PR := \pi^\infty$ das eindeutig bestimmte Tupel ist, das die Page-Rank-Eigenschaft bzgl. d hat.

Außerdem gilt: $\pi^\infty = \lim_{k\to\infty} \left(x^{(0)} \cdot P^k\right) = \lim_{k\to\infty} \left(x^{(k)}\right)$

Für näherungsweisen Berechnung von π^∞. beginnen wir mit $x^{(0)} = \left(\frac{1}{n}, \dots, \frac{1}{n}\right)$ und berechnen nacheinander für $k = 1, 2, 3, \dots$ den Vektor $x^{(k)} := x^{(k-1)} \cdot P$.

Aufgrund des hohen Zusammenhangs des Webgraphen konvergert die Folge $(x^{(0)}, x^{(1)}, x^{(2)}, \dots)$ sehr schnell gegen π^∞.

Für eine schnelle Berechnung des Vektor-Matrix-Produkts $x^{(k)} := x^{(k-1)} \cdot P$ wird ausgenützt, dass P viele "identische" Einträge der Form $\frac{1-d}{n}$ hat. Außerdem ist die Berechnung des Vektor-Matrix-Produkts sehr gut parallelisierbar.

Derzeit werden mehrere Tausend PCs eingesetzt, die mehrere Stunden zur Berechnung des Page-Ranks benötigen (... was in Anbetracht der Tatsache, dass es mehrere Milliarden Websiten gibt, erstaunlich gering ist).

Mehr zum Thema "Suchmaschinen" und "Markov-Ketten" können Sie in der Veranstaltung "Internet Algorithmen" bei Prof. Georg Schnitger lernen!

Literaturverzeichnis

[1] U. Kastens und H. Kleine Büning. *Modellierung. Grundlagen und formale Methoden*
Hanser, 2005.

[2] C. Meinel und M. Mundhenk. *Mathematische Grundlagen der Informatik. Mathematisches Denken und Beweisen.*
Teubner, 2002.

[3] A. Beutelspacher. *„Das ist o.B.d.A. trivial!“ Tipps und Tricks zur Formulierung mathematischer Gedanken.*
Vieweg Studium.

[4] M. Kreuzer und S. Kühlig. *Logik für Informatiker.*
Pearson Studium, 2006.

[5] U. Schöning. *Logik für Informatiker.*
Springer, 2000.

[6] R. Diestel. *Graphentheorie.*
Springer, 2006 (3. Auflage).

[7] L. Lovasz, J. Pelikan und K. Vesztergombi. *Discrete Mathematics. Elementary and Beyond*
Springer, 2003.

[8] J. E. Hopcroft, R. Motwani und J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.*
Addison-Wesley, 2001.

[9] U. Schöning. *Theoretische Informatik – kurzgefasst.*
Springer, 2001 (4. Auflage).

[10] I. Wegener. *Kompendium Theoretische Informatik – eine Ideensammlung.*
Teubner, 1996.

[11] I. Wegener. *Theoretische Informatik.*
Teubner, 1993.

[12] W. Reisig. *Petrinetze.*
Springer, 1982.

[13] G. Schnitger. *Skript zur Vorlesung „Internet Algorithmen“.*
Goethe-Universität Frankfurt am Main, 2009.