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

Diskrete Modellierung

Eine Einführung in grundlegende Begriffe und Methoden der Theoretischen Informatik

Skript zur Vorlesung

Prof. Dr. Nicole Schweikardt

Version vom 4. Februar 2010

Inhaltsverzeichnis

1 Einführung ins Thema "Diskrete Modellierung"

1.1 Wozu "Diskrete Modellierung" im Informatik-Studium?

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

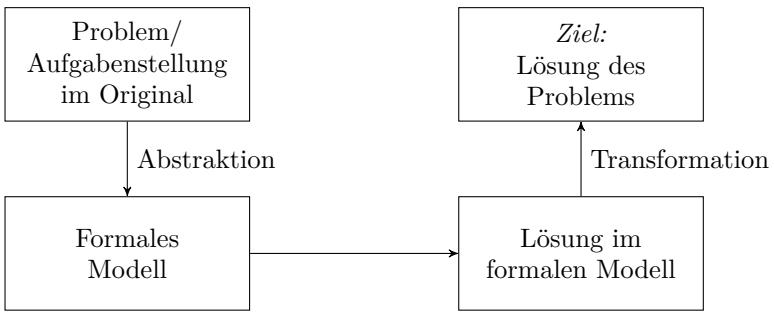
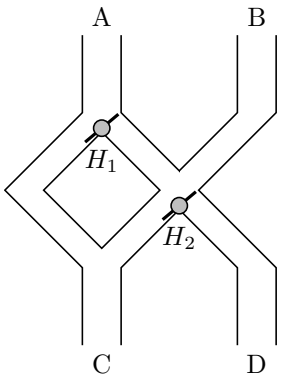


Abbildung 1.1: Generelle Vorgehensweise in der Informatik

In der Veranstaltung "Diskrete Modellierung" werden zunächst die grundlegenden Begriffe, wie z.B. "Modell" und "Modellierung", geklärt. Anschließend werden verschiedene Ausdrucksmittel der Modellierung vorgestellt und anhand von anschaulichen Beispielen verdeutlicht.

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



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

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

Lösungsansätze:

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

Hier wird der 2. Ansatz verfolgt.

Erste Analyse des Problems:

- *relevante Objekte:*
  Spielbahn, Eingänge A und B, Ausgänge C und D, Hebel H_1 und H_2, Murmeln
- *Tätigkeit:*
  Einwerfen von Murmeln an Eingängen A und/oder B
- *Start:*
  Hebel H_1 und H_2 zeigen nach links
- *Ziel:*
  Herausfinden, aus welchem Ausgang die letzte Murmel rollt, wenn nacheinander Murmeln an folgenden Eingängen eingeworfen werden: A, A, B, A, B, B, A
- *Eigenschaften/Beziehungen:*
  - Hebelpositionen: H_1 zeigt entweder nach links oder nach rechts, H_2 zeigt entweder nach links oder nach rechts.
  - Für jeden der beiden Hebel H_1 bzw. H_2 gilt: Wenn er nach links (bzw. rechts) zeigt so rollt die nächste an ihm vorbeirollende Murmel nach links (bzw. nach rechts) weiter.

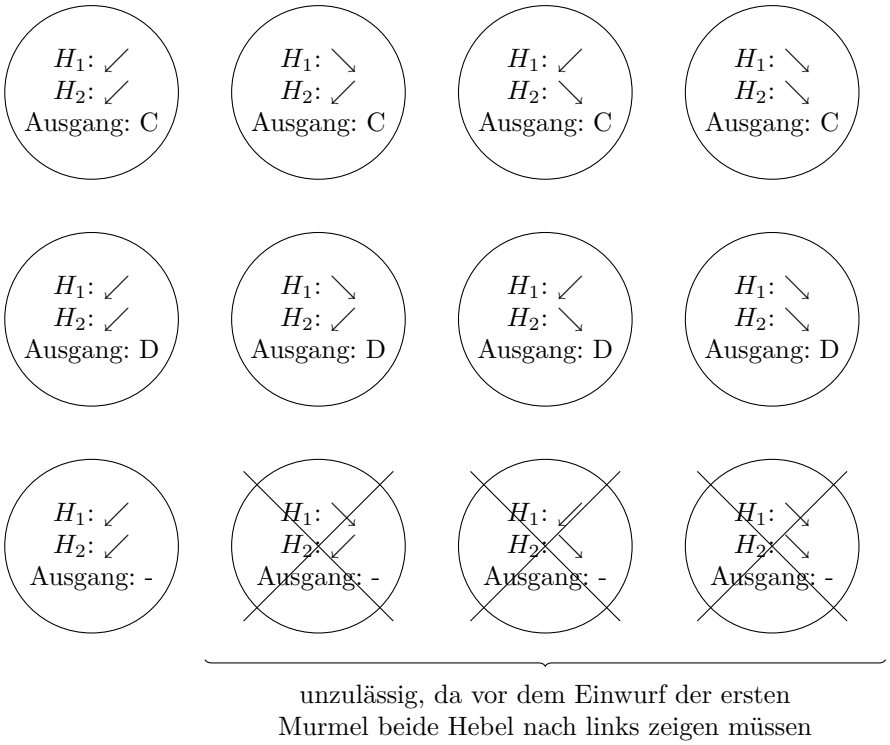
– Jeder der beiden Hebel H_1 bzw. H_2 ändert bei jedem Kontakt mit einer Murmel seine Richtung.
– Eine bei A eingeworfene Murmel rollt zu Hebel H_1.
Eine bei B eingeworfene Murmel rollt direkt zu Hebel H_2, ohne Hebel H_1 zu passieren.
– Zeigt H_1 nach links, so rollt eine bei A eingeworfene Murmel direkt zu Ausgang C.
Zeigt H_1 nach rechts, so rollt eine bei A eingeworfene Murmel zu Hebel H_2.
– Zeigt H_2 nach links, so rollt eine diesen Hebel passierende Murmel zu Ausgang C.
Zeigt H_2 nach rechts, so rollt eine diesen Hebel passierende Murmel zu Ausgang D.

Abstraktionen:

1. Nutze Abkürzungen:

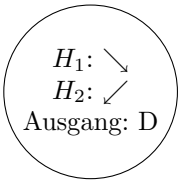
$H_1 : \swarrow$	$\widehat{=}$	Hebel H_1 zeigt nach links
$H_1 : \searrow$	$\widehat{=}$	Hebel H_1 zeigt nach rechts
$H_2 : \swarrow$	$\widehat{=}$	Hebel H_2 zeigt nach links
$H_2 : \searrow$	$\widehat{=}$	Hebel H_2 zeigt nach rechts
Ausgang: C	$\widehat{=}$	die zuvor fallen gelassene Murmel ist an Ausgang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

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



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

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



durch das Tupel (R, L, D).

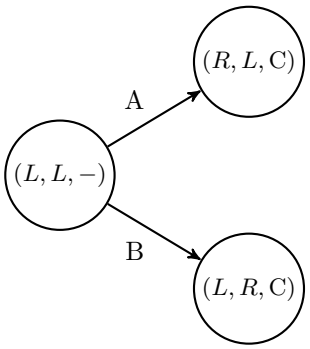
Allgemein wird ein Zustand durch ein Tupel (x, y, z) repräsentiert mit $x \in \{L, R\}$, $y \in \{L, R\}$ und $z \in \{\mathrm{A}, \mathrm{B}, -\}$, für das folgende Bedingung erfüllt ist: falls $z = -$, so ist $x = y = L$.

Übergänge von einem Zustand in einen anderen Zustand:

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

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

Graphische Darstellung:



Insgesamt ergibt sich das in Abbildung 1.2 dargestellte Bild aus Zuständen und Zustandsübergängen.

Lösung des Problems "Murmeln":

An diesem Bild lässt sich unser ursprüngliches Problem "Murmeln" (Frage: Aus welchem Ausgang rollt die letzte Murmel, wenn nacheinander Murmeln an den Eingängen A, A, B, A, B, B, A eingeworfen werden?) leicht lösen, indem man einfach einen Weg vom "Startzustand" sucht, bei dem die Pfeile nacheinander mit A, A, B, A, B, B, A beschriftet sind. In Abbildung 1.2 gibt es genau einen solchen Weg; er endet mit dem Zustand (L, R, C). Die Antwort auf die ursprünglich gestellte Frage lautet also: Wenn nacheinander Murmeln an den Eingängen A, A, B, A, B, B, A eingeworfen werden, so rollt die letzte Murmel durch Ausgang C.

Man beachte, dass man anhand von Abbildung 1.2 auch die folgende Frage beantworten kann:

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

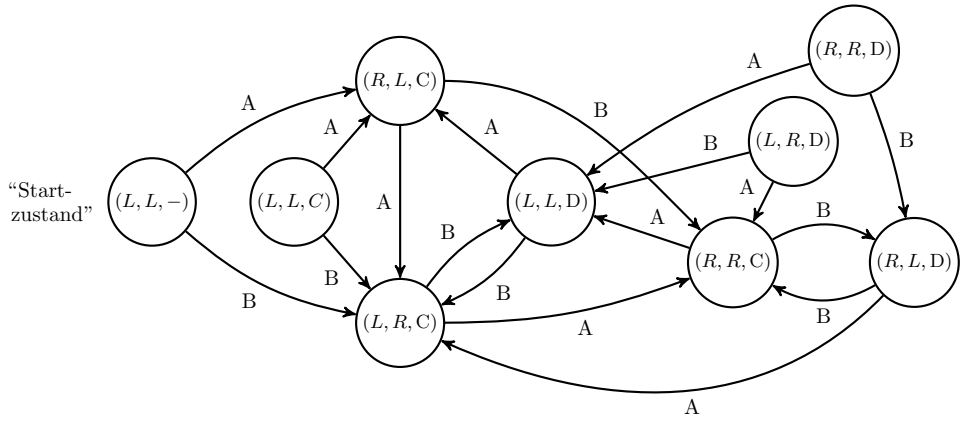


Abbildung 1.2: Übergänge zwischen den Zuständen beim Problem "Murmeln"

Um diese Frage zu beantworten muss man einfach nachprüfen, ob es in Abbildung 1.2 einen Weg vom Startzustand zum Zustand (R, R, D) gibt. Man sieht leicht, dass es in Abbildung 1.2 keinen solchen Weg gibt. Folglich lautet die korrekte Antwort auf obige Frage "nein".

□Ende Beispiel 1.1

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

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

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

Erste Analyse des Problems:

- *relevante Objekte:*
  Mann, Wolf, Ziege, Kohlkopf, Boot, Fluss, Ufer (links und rechts)
- *Eigenschaften/Beziehungen:*
  - Das Boot trägt den Mann und zusätzlich maximal ein weiteres Objekt
  - Der Wolf frisst die Ziege, falls beide unbewacht am gleichen Ufer zurückgelassen werden.

Die Ziege frisst den Kohlkopf, falls beide unbewacht am gleichen Ufer zurückgelassen werden.

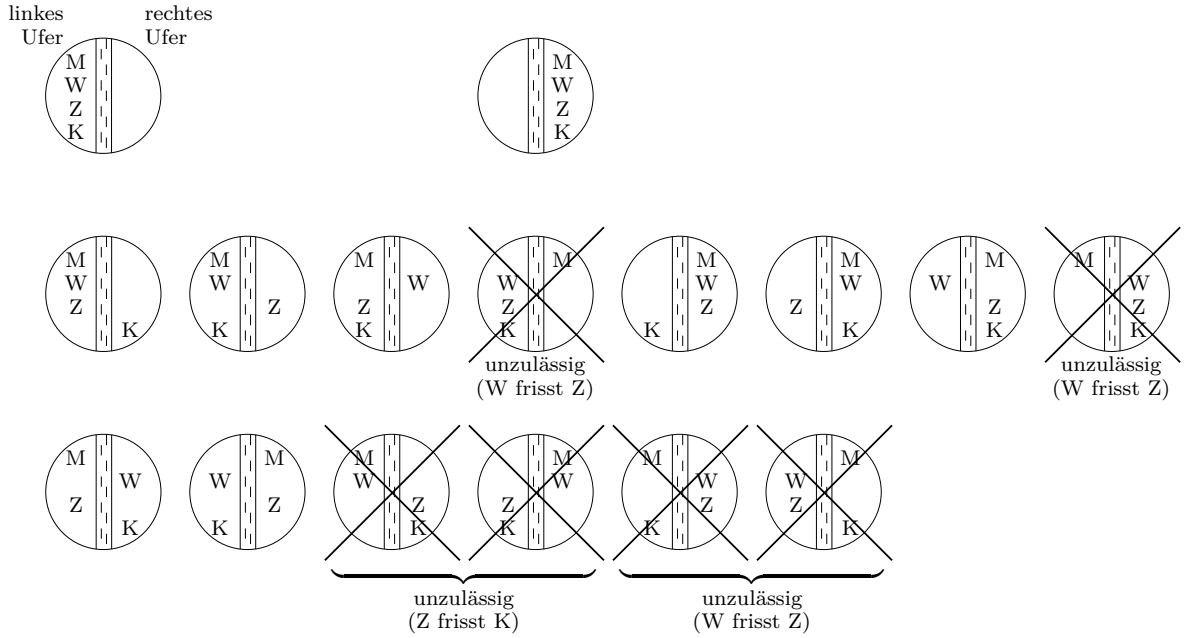
- *Tätigkeit:*
  Überqueren des Flusses
- *Start:*
  Mann, Wolf, Ziege, Kohlkopf (und Boot) am linken Ufer
- *Ziel:*
  Mann, Wolf, Ziege, Kohlkopf (und Boot) am rechten Ufer

Abstraktionen:

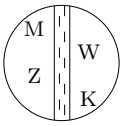
1. Nutze Abkürzungen:

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

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 $\ell \subseteq \{M, Z, W, K\}$ und $r \subseteq \{M, Z, W, K\}$, für das folgende Bedingungen erfüllt sind:

- $\ell \cup r = \{M, Z, W, K\}$
- $\ell \cap r = \emptyset$
- falls $Z, K \in \ell$, so auch $M \in \ell$ (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 $W, Z \in \ell$, so auch $M \in \ell$ (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)

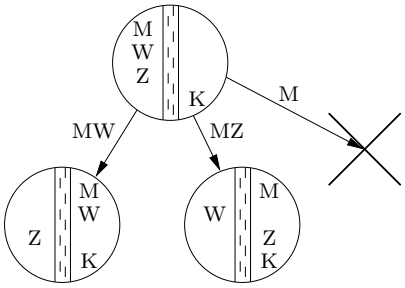
Übergänge von einem Zustand in einen anderen Zustand:

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 1.3 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?) leicht lösen, indem man einfach einen Weg vom "Startzustand" zum "Zielzustand" sucht. In Abbildung 1.3 gibt es zwei verschiedene solche Wege, die jeweils mit 7 Überfahrten auskommen.

□Ende Beispiel 1.2

Diskussion dieser beiden Modellierungsbeispiele:

Abläufe bzw. Folgen von Schritten wurden hier durch ein Zustandsübergangsdiagramm modelliert. Die **Abstraktion** bestand darin, nur die Zustände und deren Übergänge zu betrachten. Die **relevanten Objekte** wurden identifiziert: Beim "Murmelproblem" waren dies die aktuellen Positionen der beiden Hebel (jeweils L oder R) sowie der Ausgang, an dem die letzte Murmel herausgerollt ist (C oder D). Beim "Flussüberquerungsproblem" waren dies die Positionen von M,

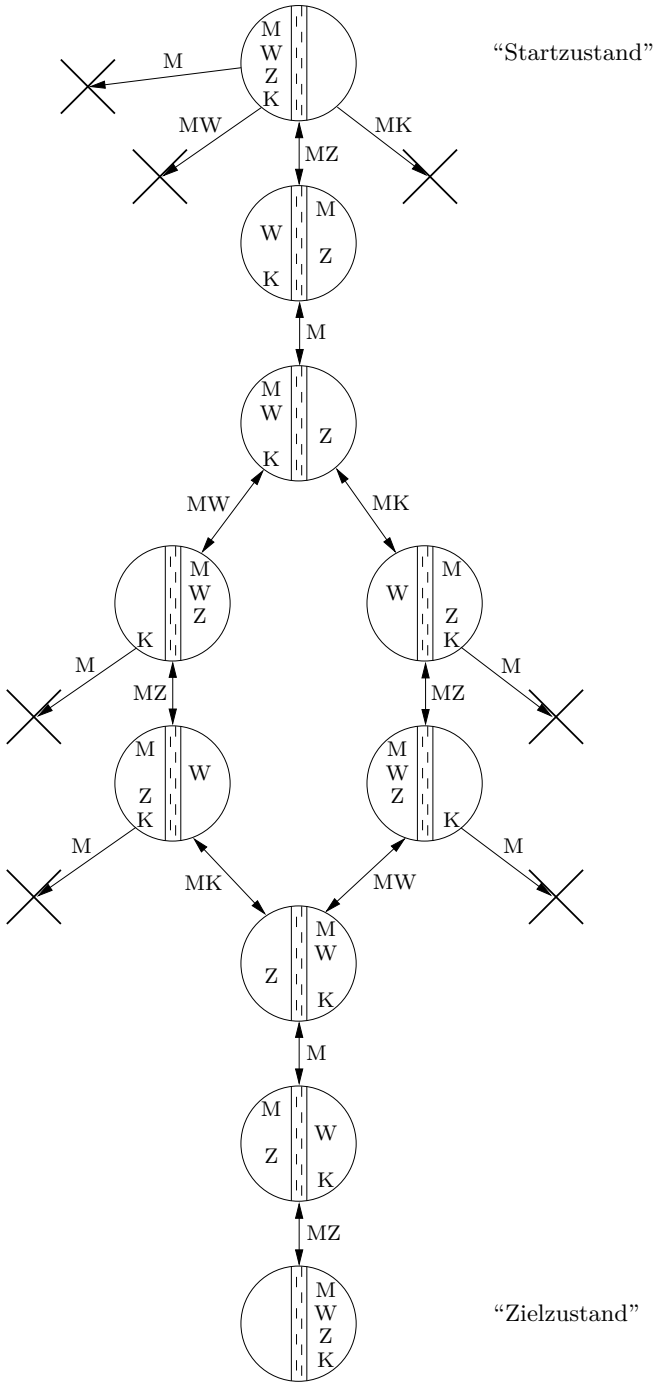


Abbildung 1.3: Übergänge zwischen den Zuständen beim Flussüberquerungsproblem

W, Z, K (jeweils am linken oder am rechten Ufer). Jeden **Zustand** haben wir **repräsentiert** durch ein Tupel ((x, y, z) im "Murmelproblem" bzw. (ℓ, r) im "Flussüberquerungsproblem"). Die möglichen Tupel wurden in **zulässige Zustände** und **unzulässige Zustände** eingeteilt. Übergänge von einem Zustand zu einem anderen Zustand wurden mit den jeweiligen Aktionen beschriftet (dem Eingang, an dem die nächste Murmel eingeworfen wird bzw. mit den Objekten, die als nächstes über den Fluss transportiert werden).

Besonders wichtig ist auch, dass Aspekte, die zur Lösung der Aufgabe irrelevant sind **nicht** modelliert wurden (beim "Murmelproblem" z.B. die genaue Anordnung der Spielbahn, die Gesetze der Schwerkraft und die Kräfte, die mechanisch auf die beiden Hebel wirken; beim "Flussüberquerungsproblem" z.B. Name, Breite, Tiefe des Flusses oder Länge, Geschwindigkeit des Boots etc.).

Die **"Kreative Leistung"**, die hier zur Lösung der beiden Probleme geleistet wurde, war, den Kalkül der Zustandsübergangsdiagramme zu wählen und die Bedeutung der Zustände und Übergänge festlegen. Das konkrete Zustandsübergangsdiagramm aufzustellen und einen Weg mit der entsprechenden Beschriftung (beim "Murmelproblem") bzw. einen Weg vom Start- zum Zielzustand (beim "Flussüberquerungsproblem") zu finden, war dann reine **"Routine-Arbeit"**.

Im Verlauf der Veranstaltung "Diskrete Modellierung" werden wir verschiedene Kalküle kennenlernen, die zur Lösung typischer Informatik-Probleme besonders geeignet sind.

1.2 Ziele der Veranstaltung "Diskrete Modellierung"

Ziel der Veranstaltung "Diskrete Modellierung" ist, einen Überblick über grundlegende Modellierungsmethoden und -kalküle zu geben — insbesondere über Aussagenlogik (Kapitel 3), Graphen und Bäume (Kapitel 4), Markov-Ketten (Kapitel 5), Logik erster Stufe / Prädikatenlogik (Kapitel 6), Transitionssysteme / endliche Automaten (Kapitel 7), Petri-Netze (Kapitel 7), kontextfreie Grammatiken (Kapitel 8) und das Entity-Relationship-Modell (Kapitel 8).

Weitere Ziele sind:

- das Verständnis des konzeptionellen Kerns der Kalküle,
- die Fähigkeit, die Kalküle an typischen Beispielen anzuwenden,
- die Fähigkeit zur präzisen und formalen Ausdrucksweise bei der Analyse von Problemen — dazu gehört auch das Verständnis und der souveräne Umgang mit mathematische Grundlagen und Beweistechniken (Kapitel 2),
- die Erkenntnis des praktischen Wertes präziser Beschreibungen.

1.3 Der Begriff "Diskrete Modellierung"

Einträge in Duden, Deutsches Universalwörterbuch (4. Auflage, 2001):

diskret <Adj.> [1,2: frz. discret < mlat. discretus = abgesondert, zu lat. discernere = absondern, unterscheiden; 3: engl. discrete] (bildungsspr.): diskret
1.a) *so unauffällig behandelt, ausgeführt, dass es von anderen nicht bemerkt wird; vertraulich:* -e Spenden an die Parteien; eine heikle Angelegenheit d. behandeln; **b)** *taktvoll, rücksichtsvoll:* ein -es Verhalten; eine Peinlichkeit d. übergehen; d. schweigen; etw. d. überssehen; **c)** *unaufdringlich;*

zurückhaltend; dezent: ein -es Parfüm, Muster; d. angezogen. **2.** (Technik, Physik, Math.) *durch endliche Intervalle od. Abstände voneinander getrennt:* -e Halbleiter, Bauteile; eine -e (*nicht integrierte*) Schaltung **3.** (Sprachw.) *(von sprachlichen Einheiten) abgrenzbar; abgesondert; unterschieden.*

Modell

Modell, das; -s, -e [ital. modello = Muster, Entwurf, zu lat. modulus, ↑[1]Modul]: **1.a)** *Form, Beschaffenheit, Maßverhältnisse veranschulichende Ausführung eines vorhandenen od. noch zu schaffenden Gegenstandes in bestimmtem (bes. verkleinerndem) Maßstab:* das M. eines Schiffes, Flugzeugs, einer Burg, Fabrik; ein M. entwerfen, bauen; **b)** (Technik, bild. Kunst) *Muster, Entwurf einer Plastik, eines technischen o.ä., durch Guss herzustellenden Gegenstandes, nach dem die Guss- bzw. Gipsform hergestellt wird:* das M. einer Plastik; **c)** (Wissensch.) *innere Beziehungen u. Funktionen von etw., abbildendes bzw. [schematisch] veranschaulichendes [u. vereinfachendes, idealisierendes] Objekt, Gebilde:* ein M. des Atomkerns; **d)** (math. Logik) *Interpretation eines Axiomensystems, nach der alle Axiome des Systems wahre Aussagen sind.* **2.a)** *als Gegenstand der bildnerischen, künstlerischen o.ä. Darstellung od. Gestaltung benutztes Objekt, Lebewesen usw.;* **b)** *Person, die sich [berufsmäßig] als Gegenstand bildnerischer od. fotografischer Darstellung, Gestaltung zur Verfügung stellt:* als M. arbeiten; ***[jmdm.] M. sitzen/stehen** *(jmds. Modell sein)*: sie hat dem Maler für dieses Bild M. gesessen; **c)** [2]*Model* (a); **d)** (verhüll.) *Hostess* (3). **3.a)** *(Gegenstand als) Entwurf, Muster, Vorlage für die serienweise Herstellung von etw.;* **b)** *Typ, Art der Ausführung eines Fabrikats;* **c)** (Rechtspr.) *durch Gesetz urheberrechtlich geschützte Gestaltungsform eines Gebrauchsgegenstandes.* **4.** (Mode) *[Kleidungs]stück, das eine Einzelanfertigung ist [u. ungefähr als Muster, Vorlage od. Anhaltspunkt für die serienweise Herstellung bzw. Konfektion dienen kann]:* ein Pariser M. **5.** (bildungsspr.) **a)** *etw., was (durch den Grad seiner Perfektion, Vorbildlichkeit o.Ä.) für anderes od. für andere Vorbild, Beispiel, Muster sein kann:* etw. nach dem M. von etw. gestalten; **b)** *als Muster gedachter Entwurf:* das M. eines neuen Gesetzes.

Eintrag in Dictionary of computer science, engineering, and technology, CRC Press, 2001:

model

model (1) a representation of reality of an artifact or activity intended to explain the behaviour of some aspects of it. In creating a model, an abstraction technique is used. Thus, the model is typically less complex or complete than the reality modeled and can be regarded as an abstract description. This technique identifies commonalties and, in doing so, loses details.

(2) a mathematical or schematic description of a computer or network system. Modeling usually involves an act of abstraction; i.e., the model only includes the most important properties of the original system.

Eintrag in Duden Informatik – Ein Fachlexikon für Studium und Praxis, 3. Auflage, 2001:

Modell

Modell: Abbild von etwas, oft unter Weglassen von Details, also im Sinne einer vereinfachenden Darstellung. Für die Entwicklung von (Hard- und Software-)Systemen ist die Modellbildung und der sich anschließende Entwurf einer Architektur von zentraler Bedeutung.
(1) Jede wirklichkeitsbezogene Anwendung der Datenverarbeitung basiert auf einem Modell, das einen Teil der Wirklichkeit angenähert widerspiegelt. Da die Wirklichkeit viel zu komplex ist, um sie direkt im Rechner wiedergeben zu können, werden die für die jeweilige Anwendung relevanten Anteile herauskristallisiert, analysiert, zu einer Struktur zusammengefügt und durch ein Modell, also durch entsprechende konkrete Datentypen, Objekte, Attribute, Operationen und deren Beziehungen untereinander dargestellt. Die Darstellung erfolgt meist auf der abstrakteren

Ebene der ↑Klassen und deren hierarchischen Beziehungen (Vererbung), der Abbildungen, die die wechselseitigen Abhängigkeiten beschreiben, und der Architekturen, durch die diese Größen miteinander verknüpft werden. Meist fügt man noch Anforderungen hinzu, die sich auf zu erfüllende Eigenschaften, auf einzuhaltende Rahmenbedingungen usw. beziehen.
Die Erstellung eines Modells zu einer Anwendung bezeichnet man als *Modellierung*. Sie wird heute meist objektorientiert durchgeführt und mithilfe vorhandener Klassenbibliotheken realisiert, in denen entsprechende „Oberklassen“ für viele Anwendungen bereits abgelegt sind. Diese Wiederverwendung erprobter und ausgetesteter Klassen macht eine Stärke des objektorientierten Entwurfs aus. Zur Unterstützung der Modellierung gibt es Schemata und Beschreibungssprachen, z.B. das ↑Entity-Relationship-Modell oder die ↑UML.
Modelle können sehr abstrakt sein (z.B. Darstellungen durch Grammatiken, Graphen, Automaten) und Gesetzmäßigkeiten enthalten, oder sie können sehr nahe an der Wirklichkeit sein und vornehmlich der Veranschaulichung von Sachverhalten dienen. Auf dem Modell setzen dann Verarbeitungsalgorithmen und Simulationen auf. Die Güte eines Modells ergibt sich daraus, inwieweit die Ergebnisse der Berechnungen mit der Wirklichkeit übereinstimmen.
In der Informatik werden meist diskrete Modelle betrachtet. In den Natur- und Ingenieurwissenschaften verwendet man vorwiegend kontinuierliche Modelle, zum Beispiel auf der Basis von Differenzialgleichungen. Mischformen bezeichnet man als hybride Modelle.
(2) Mengen mit Operationen, die eine logische Formel (↑Logik) oder die Gesetze eines ↑Datentyps erfüllen.

Zusammenfassung:

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

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

Modelle sind **absichtlich** nicht originalgetreu. Sie heben bestimmte Eigenschaften hervor und lassen andere weg. Der konkrete Verwendungszweck des Modells bestimmt, welche Eigenschaften

modelliert werden und welcher Kalkül dafür besonders geeignet ist.
Ein Modell beschreibt stets nur einige bestimmte Aspekte des Originals, etwa

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

  Dafür geeignete Kalküle sind z.B. Wertebereiche, Entity-Relationship-Modell, Bäume, Graphen.

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

  Dafür geeignete Kalküle sind z.B. Wertebereiche, Logik, Entity-Relationship-Modell.

- Beziehungen zwischen Teilen des Originals (z.B. "Wolf frisst Ziege, Ziege frisst Kohlkopf" in Beispiel 1.2).

  Dafür geeignete Kalküle sind z.B. Graphen, Logik, Entity-Relationship-Modell.

- Verhalten des Originals unter Operationen (z.B. aufeinanderfolgende Zustände bei wiederholtem Einwurf von Murmeln in Beispiel 1.1).

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

Beispiel 1.3.

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

- Gebäudemodell: zur Vermittlung eines optischen Eindrucks
- Grundriss: zur Einteilung der Räume und des Grundstückes
- Kostenplan: zur Finanzierung

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

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

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

□Ende Beispiel 1.3

1.4 Literaturhinweise zu Kapitel 1

Als vertiefende Lektüre sei Kapitel 1 in [15] empfohlen.

Quellennachweis: Teile dieses Kapitels orientieren sich an Kapitel 1 in [15]; insbesondere Beispiel 1.3 sowie das in Beispiel 1.2 betrachtete "Flussüberquerungsproblem" sind im Wesentlichen aus [15] übernommen. Das "Flussüberquerungsproblem" findet sich bereits in [12]; das "Murmelproblem" aus Beispiel 1.1 ist eine vereinfachte Variante von Aufgabe 2.3 in [12].

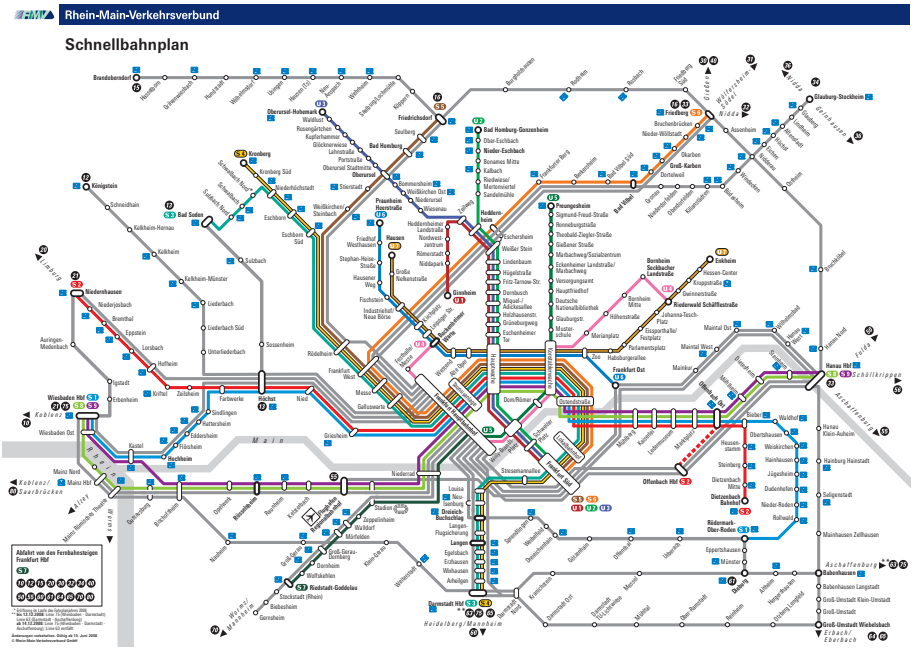


Abbildung 1.4: Schnellbahnplan des Rhein-Main-Verkehrsverbundes (RMV)

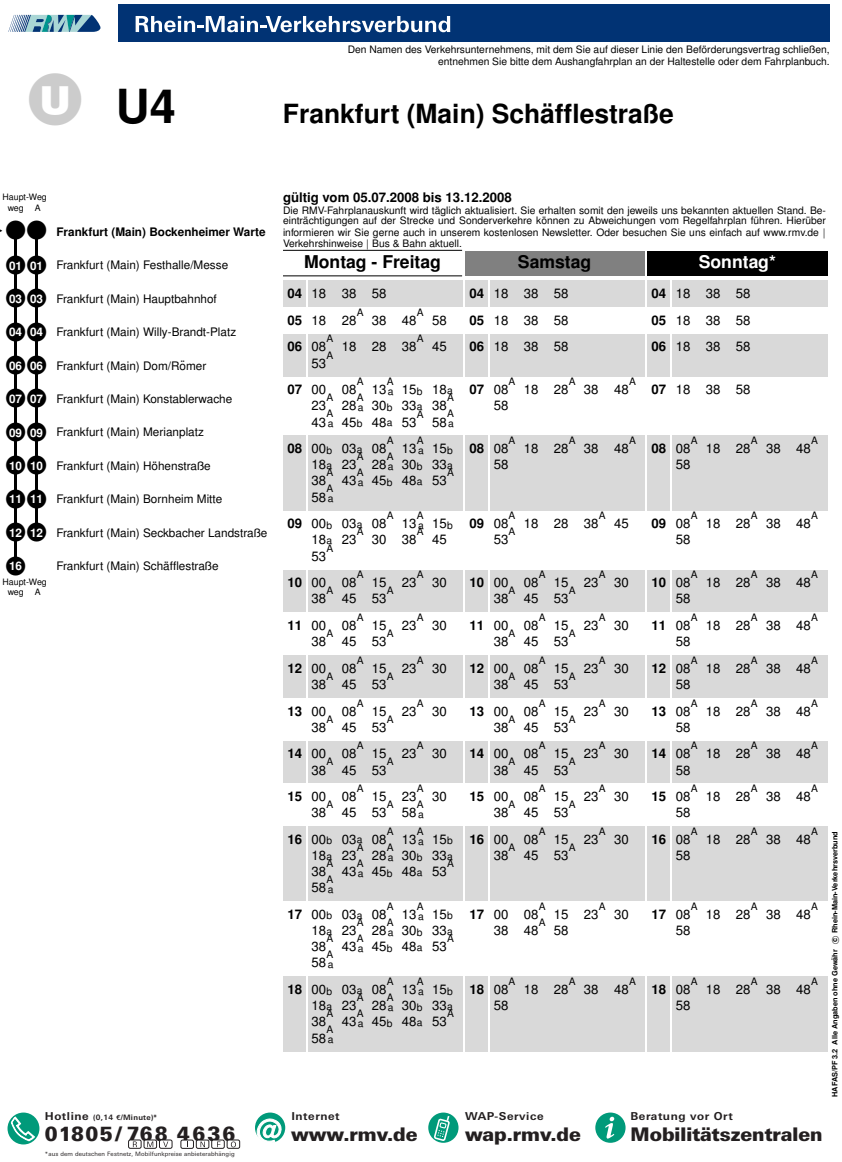
1.5 Übungsaufgaben zu Kapitel 1

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

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

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

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



Rhein-Main-Verkehrsverbund

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

U4

Frankfurt (Main) Schäfflestraße

Haupt-Weg weg A

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

Haupt-Weg weg A

gültig vom 05.07.2008 bis 13.12.2008

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

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

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

2 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 $\Longleftrightarrow$ die erreichte Prozentzahl z ist $\geqslant 50\%$
$\square$	markiert das Ende eines Beweises

Modellierung und Wertebereiche

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

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

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

Die Karten eines Skat-Kartenspiels lassen sich durch folgende Wertebereiche darstellen:

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

□Ende Beispiel 2.1

Wertebereiche sind u.a. wichtig

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

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

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

nutzen. Ziel von Kapitel 2 ist, diese Begriffe zu präzisieren und darüber hinaus auch einige wichtige mathematische Grundlagen und Beweistechniken zu erklären.

2.1 Mengen

2.1.1 Was ist eine Menge?

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

Wir schreiben

$$m \in M$$

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

$$m \notin M$$

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

Notation:

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

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

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

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

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

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

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

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

Somit führen beide Fälle zu einem Widerspruch, obwohl wir wissen, dass einer der beiden Fälle zutreffen müsste.
Fazit: 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 sorgfältig axiomatisch aufbauen (siehe z.B. [6]) — dies sprengt allerdings den Rahmen dieser Vorlesung. Sofern man sich der Problematik bewusst ist, kann man sie im "täglichen Gebrauch", den Informatiker/innen von Mengen machen, vermeiden. Wir arbeiten daher weiter mit einem naiven Mengenbegriff, den wir nach den im Folgenden beschriebenen Grundsätzen verwenden werden.

Beschreibung bzw. Definition von Mengen:

Wir beschreiben bzw. definieren Mengen

- *extensional*, durch Aufzählen der Elemente, z.B.

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

oder

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

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

Extensional lässt sich die Menge M_2 folgendermaßen beschreiben:

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

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

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

Vorsicht:

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

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.

Wichtige grundsätzliche Eigenschaften von Mengen:

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

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

- Mengen können aus atomaren oder aus zusammengesetzten Elementen gebildet werden. Menge kann auch "verschiedenartige" Elemente enthalten.

Beispiel: Die Menge

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

besteht aus 4 Elementen: dem atomaren Wert 1, dem Tupel $(\text{Pik}, 8)$, der Menge $\{\text{rot}, \text{blau}\}$ und dem atomaren Wert 5.

Notationen für bestimmte Zahlenmengen:

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

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

Definition 2.2 (leere Menge).
Die **leere Menge** ist die (eindeutig bestimmte) Menge, die kein(e) Element(e) enthält. Wir bezeichnen sie mit $\emptyset$.

leere Menge
$\emptyset$

Frage 2.3: Gibt es eine "Menge aller Mengen"?

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

2.1.2 Mengenalgebra

In diesem Abschnitt werden einige grundlegende Operationen auf Mengen betrachtet. Nebenbei werden auch einige (sehr einfache) Beispiele von mathematischen Beweisen gegeben.

Definition 2.4 (Gleichheit von Mengen).
Zwei Mengen M und N sind gleich (kurz: $M = N$), falls sie dieselben Elemente enthalten, d.h. falls gilt:

$M = N$

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

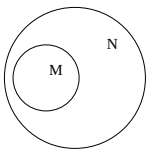
Beachte:
$\emptyset \neq \{\emptyset\}$, denn $\emptyset$ ist die Menge, die keine Elemente enthält, während $\{\emptyset\}$ eine Menge ist, die ein Element (nämlich $\emptyset$) enthält.

Definition 2.5 (Teilmengen). Seien M, N Mengen.

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

Teilmenge
$M \subseteq N$

Skizze:



echte Teilmenge $M \subsetneq N$

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

Obermenge $M \supseteq N$

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

echte Obermenge $M \supsetneq N$

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

Satz 2.6. 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)

$$\begin{aligned} M = N &\overset{\text{Def. 2.4}}{\iff} \text{f.a. } x \in M \text{ gilt } x \in N \text{ und} \\ &\qquad \text{f.a. } x \in N \text{ gilt } x \in M \\ &\iff \text{jedes Element von } M \text{ ist auch ein Element von } N \text{ und} \\ &\qquad \text{jedes Element von } N \text{ ist auch ein Element von } M \\ &\overset{\text{Def. 2.5(a)}}{\iff} M \subseteq N \text{ und } N \subseteq M \\ &\overset{\text{Def. 2.5(c)}}{\iff} M \subseteq N \text{ und } M \supseteq N. \end{aligned}$$

(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 2.7. Seien M und N Mengen.

Durchschnitt $M \cap N$

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

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

Vereinigung $M \cup N$

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

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

Differenz $M \setminus N$

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

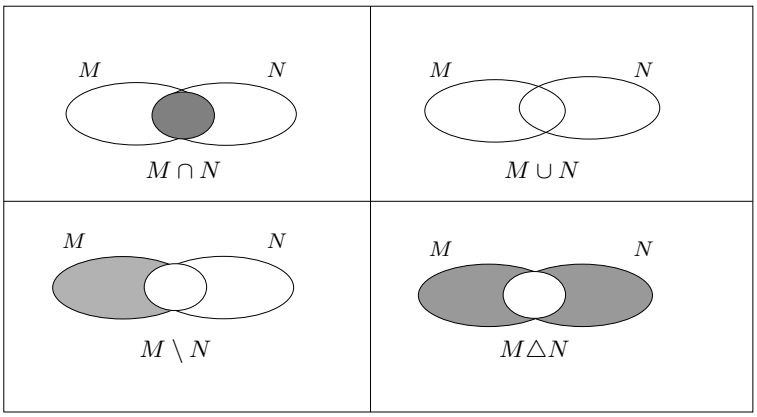
$$M \setminus N := M - N := \{x : x \in M \text{ und } x \notin N\}.$$

symmetrische Differenz $M \triangle N$

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

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

Veranschaulichung durch Venn-Diagramme:



Notation 2.8 (disjunkt).
Zwei Mengen M und N heißen **disjunkt**, falls $M \cap N = \emptyset$, d.h. falls sie keine gemeinsamen Elemente besitzen. Manchmal schreiben wir $M \dot\cup N$

$$M \dot\cup N,$$

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

Rechenregeln für Durchschnitt und Vereinigung:

Satz 2.9. Seien M, N, P Mengen. Dann gelten:

(a) **Idempotenz**: Idempotenz

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

(b) **Kommutativität**: Kommutativität

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

(c) **Assoziativität**: Assoziativität

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

(d) **Absorption**: Absorption

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

(e) **Distributivität**: Distributivität

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

Beweis:

(a)

$$\begin{aligned} M \cap M &\overset{\text{Def. 2.7(a)}}{=} \{x : x \in M \text{ und } x \in M\} \\ &= \{x : x \in M\} \\ &= M. \end{aligned}$$

Analog: $M \cup M = M$.

(b)

$$\begin{aligned} M \cap N &\overset{\text{Def. 2.7(a)}}{=} \{x : x \in M \text{ und } x \in N\} \\ &= \{x : x \in N \text{ und } x \in M\} \\ &\overset{\text{Def. 2.7(a)}}{=} N \cap M. \end{aligned}$$

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

(c)

$$\begin{aligned} M \cap (N \cap P) &\overset{\text{Def. 2.7(a)}}{=} \{x : x \in M \text{ und } x \in N \cap P\} \\ &\overset{\text{Def. 2.7(a)}}{=} \{x : x \in M \text{ und } (x \in N \text{ und } x \in P)\} \\ &= \{x : (x \in M \text{ und } x \in N) \text{ und } x \in P\} \\ &\overset{\text{Def. 2.7(a)}}{=} \{x : x \in M \cap N \text{ und } x \in P\} \\ &\overset{\text{Def. 2.7(a)}}{=} (M \cap N) \cap P. \end{aligned}$$

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

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

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

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

Aus Satz 2.6(a) folgt dann, dass $M \cap (M \cup N) = M$.

Zu Schritt 1:

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

Beweis: Sei $m \in M$ beliebig. Zu zeigen: $m \in M \cap (M \cup N)$. Wegen $m \in M$ gilt auch $m \in M \cup N$ (gemäß Definition 2.7(b)). Wegen $m \in M$ und $m \in M \cup N$ gilt gemäß Definition 2.7(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 2.7(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.

□

2.1.3 Das Komplement einer Menge

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 $\overline{M}$

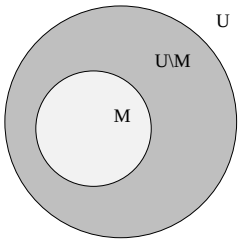
$$\overline{M} := \{x : x \notin M\}$$

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

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

die "Menge aller Mengen", und dass es die nicht geben kann, haben wir bereits bei der Beantwortung von Frage 2.3 gesehen.

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



Rechenregeln für Komplemente:

Satz 2.10.
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

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

(c) **Inversionsregeln**: Inversionsregeln

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

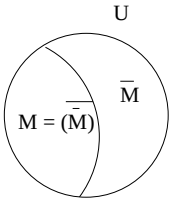
(d) **Identitätsregeln**: Identitätsregeln

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

Beweis: Übung. □

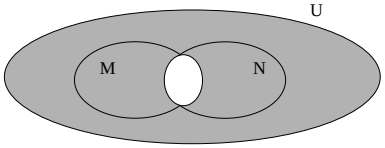
Veranschaulichung durch Venn-Diagramme:

- Doppelte Negation:

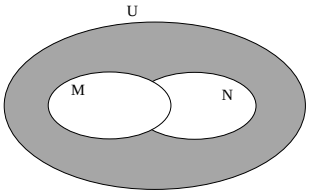


- De Morgansche Regeln:

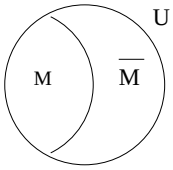
$\overline{M \cap N} = \overline{M} \cup \overline{N}$



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



- Inversionsregel:



2.1.4 Mächtigkeit bzw. Kardinalität einer Menge

Definition 2.11.

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
$|M|$

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

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

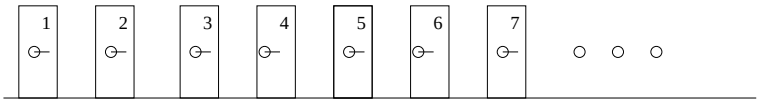
Man beachte, dass "∞" keine natürliche Zahl ist (d.h. $\infty \notin \mathbb{N}$), sondern lediglich eine Abkürzung für das Wort "unendlich".

Beispiel 2.12.

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

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.

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

2.1.5 Die Potenzmenge

Definition 2.13.
Die **Potenzmenge** (engl.: power set) einer Menge M (kurz: $\mathcal{P}(M)$) ist die Menge aller Teilmengen von M. D.h.:

Potenzmenge $\mathcal{P}(M)$

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

Beispiel 2.14.

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

Notation 2.15.
In manchen Büchern wird $\mathcal{P}(M)$ auch mit $\mathrm{Pow}(M)$ (für "power set") oder mit 2^M bezeichnet. Später, in Folgerung 2.39, werden wir nachweisen, dass für jede endliche Menge M gilt:

$\mathrm{Pow}(M)$
2^M
$|\mathcal{P}(M)| = 2^{|M|}$

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

2.2 Kartesische Produkte und Relationen

2.2.1 Paare, Tupel und kartesische Produkte

Definition 2.16 (Paare und Tupel).

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

Paar

k-Tupel

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

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

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

Bemerkung 2.17.

leeres Tupel
()

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

(b) Man beachte den Unterschied zwischen Tupeln und Mengen: z.B.

- $(1, 2) \neq (2, 1)$, aber $\{1, 2\} = \{2, 1\}$.
- $(1, 1, 2) \neq (1, 2)$, aber $\{1, 1, 2\} = \{1, 2\}$.

Definition 2.18.

k-te Potenz
M^k

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

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

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

kartesisches Produkt
Kreuzprodukt
$M \times N$

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

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

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

- $M \times N = \{(a, 1), (a, 2), (a, 3), (b, 1), (b, 2), (b, 3)\}$.
- $M \times \{1\} = \{(a, 1), (b, 1)\}$.
- $M \times \emptyset = \emptyset$.
- $M^2 = \{(a, a), (a, b), (b, a), (b, b)\}$.
- $M^1 = \{(a), (b)\}$.
- $M^0 = \{()\}$.
- $\emptyset^2 = \emptyset$.
- $\emptyset^1 = \emptyset$.
- $\emptyset^0 = \{()\}$.

- In Beispiel 2.1 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".

Notation 2.20.
Ist $k \in \mathbb{N}_{>0}$ und sind $z_1, \ldots, z_k$ Zahlen, so schreiben wir

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

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

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

um das Produkt der Zahlen $z_1, \ldots, z_k$ zu bezeichnen (d.h. die Zahl $z_1 \cdot \cdots \cdot z_k$).
Sind $M_1, \ldots, M_k$ Mengen, so schreiben wir

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

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

Die Mächtigkeit von kartesischen Produkten:

Satz 2.21.

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

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

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

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

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

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

Beweis:

(a) Es gilt:

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

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

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

(b) Analog; Details: Übung.

(c)

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

□

2.2.2 Worte bzw. endliche Folgen

Bemerkung 2.22. Sei A eine Menge.

Wort

- 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.

Alphabet

- A ist dann das **Alphabet**, über dem die Worte gebildet werden, und $a_1 \cdots a_k$ wird "Wort über A" genannt.

leeres Wort (ε)

- Das **leere Tupel** $() \in A^0$ heißt auch **leeres Wort** und wird oft als ε (epsilon) bezeichnet.

Länge

- Die **Länge** eines Wortes $a_1 \cdots a_k$ ist die Zahl

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

  Insbesondere ist $|\varepsilon| = 0$, d.h. das leere Wort hat die Länge 0.

Konkatenation

- Sind $v = a_1 \cdots a_k$ und $w = b_1 \cdots b_\ell$ zwei Worte über A, so ist die **Konkatenation** von v und w das Wort

  $$vw \;:=\; a_1 \cdots a_k b_1 \cdots b_\ell.$$

- Manchmal wird ein Wort $a_1 \cdots a_k$ auch als **Folge** der Länge k aufgefasst.

Definition 2.23 (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.

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

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

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

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

Beispiel 2.24 (Natürliche Sprachen).
Wir betrachten das Alphabet

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

Beispiele für Sprachen über $A_{deutsch}$ sind:

- 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 2.25 (Programmiersprachen).
Wir betrachten das Alphabet

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

Beispiele für Sprachen über Alphabet ASCII sind:

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

2.2.3 Relationen

Relationen sind Teilmengen von kartesischen Produkten. Präzise:

Definition 2.26.

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 2.27. 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 $(24, 12, 2009)$ gehört, nicht aber das Tupel $(30, 2, 2009)$.

Notation 2.28.

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

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

  Beispiel:

  - $m \leqslant n$, für natürliche Zahlen m, n
  - $m \neq n$

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

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

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

  Im Datums-Beispiel gilt: Gültig$(24, 12, 2009)$, aber es gilt **nicht**: Gültig$(30, 2, 2009)$.

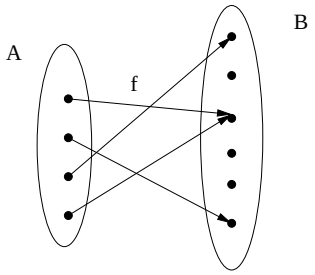
2.3 Funktionen

2.3.1 Totale Funktionen und partielle Funktionen

Definition 2.29. 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 2.30.

(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

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

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

$\mathrm{Abb}(A, B)$

(e) In manchen Büchern wird $\mathrm{Abb}(A, B)$ auch mit $A \to B$ oder mit B^A bezeichnet. Später, in Folgerung 2.39, werden wir sehen, dass

B^A

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

Definition 2.31.
Zwei Funktionen $f\colon A \to B$ und $g\colon A \to B$ sind **gleich** (kurz: $f = g$), falls f.a. $a \in A$ gilt: $f(a) = g(a)$.

Definition 2.32 (Definitionsbereich, Bildbereich, Bild). Sei $f\colon A \to B$.

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

Definitionsbereich
$\mathrm{Def}(f)$

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

Bildbereich

(c) Das **Bild** von f (genauer: das Bild von A unter f) ist die Menge

$\mathrm{Bild}(f)$

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

Definition 2.33 (Restriktionen).

Restriktion
Einschränkung

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 $f|_{A'}(a) := f(a)$.

Definition 2.34.

partielle Funktion

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.

(a) Im Gegensatz zu partiellen Funktionen nennt man Funktionen, wie wir sie in Definition 2.29 definiert haben, auch **totale Funktionen**.

totale Funktion

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

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

2.3.2 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.

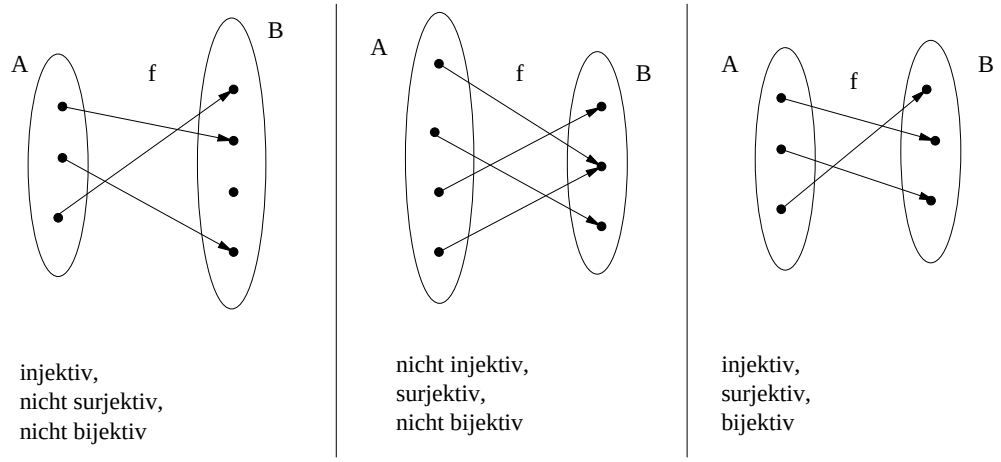
surjektiv

(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 $X \in \mathcal{P}(M)$ (d.h. $X \subseteq M$) durch die so genannte **charakteristische Funktion** $\chi_X\colon M \to \{0,1\}$ mit

charakteristische Funktion

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

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

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

Behauptung: f ist bijektiv.

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

Schritt 1: f ist injektiv:

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

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

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

$$m \in X \iff m \in X'.$$

Somit ist $X = X'$.

Schritt 2: f ist surjektiv:

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

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

Wir wählen

$$X := \{m \in M : h(m) = 1\}.$$

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

(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 2.21(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

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

□

2.3.3 Spezielle Funktionen

Identitätsfunktion id_M

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. $m \in M$.

Definition 2.41 (Multimenge, engl.: bag).

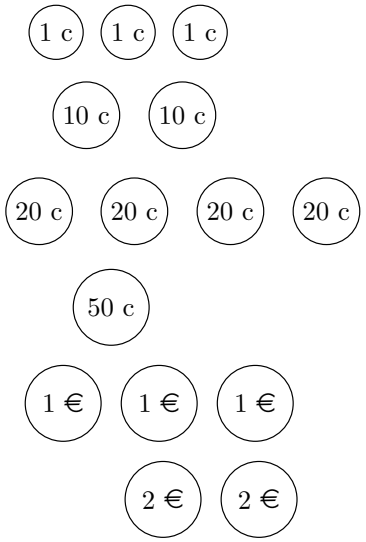
Multimenge

Eine **Multimenge** über einer Menge M ist eine Funktion $f \colon M \to \mathbb{N}$.

Mit solchen "Multimengen" 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
- 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 (die konsistent ist mit Definition 2.29):

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

2.4 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 zunächst nicht gesucht.

Formale Modellierung:

- Menge der **Nationen**:

  $$\text{Nationen} \;:=\; \{\text{D}, \text{F}, \text{Ö}, \text{S}\},$$

  wobei D für Deutschland, F für Frankreich, Ö für Österreich und S für Spanien steht.

- Die **Delegierten** können wir repräsentieren als Paare, die aus einer Nation und einem Element aus $\{1, 2, 3\}$ bestehen, so dass beispielsweise die drei Delegierten aus Deutschland durch die Paare $(\mathrm{D}, 1)$, $(\mathrm{D}, 2)$ und $(\mathrm{D}, 3)$ modelliert werden. Also:

$$\text{Delegierte} \;:=\; \text{Nationen} \times \text{DelegiertenNummer},$$

  wobei $\text{DelegiertenNummer} := \{1, 2, 3\}$.

- Wir nutzen eine Funktion "spricht", die jedem Delegierten die Menge von **Sprachen** zuordnet, die er beherrscht. Formal:

$$\text{spricht}\colon \text{Delegierte} \to \mathcal{P}(\text{Sprachen}),$$

  wobei

$$\text{Sprachen} \;:=\; \{\text{deutsch}, \text{französisch}, \text{spanisch}, \text{englisch}, \text{italienisch}, \text{chinesisch}, \dots\}.$$

- Die drei Arbeitskreise bezeichnen wir mit AK1, AK2, AK3 und setzen

$$\text{Arbeitskreise} \;:=\; \{\text{AK1}, \text{AK2}, \text{AK3}\}.$$

- Eine konkrete Besetzung der drei Arbeitskreise repräsentieren wir durch eine Funktion

$$\text{AK-Besetzung}\colon \text{Arbeitskreise} \to \mathcal{P}(\text{Delegierte}),$$

  die jedem der 3 Arbeitskreise die Menge der Delegierten zuordnet, die Mitglied des Arbeitskreises sind.

- Die Bedingung, dass jede Nation in jedem Arbeitskreis vertreten ist, lässt sich folgendermaßen formulieren:

$$\text{f.a. } a \in \text{Arbeitskreise ist } \text{Vertretene\_Nationen\_in\_}a \;=\; \text{Nationen},$$

  wobei

$$\text{Vertretene\_Nationen\_in\_}a \;:=\; \Big\{\, n \in \text{Nationen} \;:\; \text{es ex. ein } i \in \text{DelegiertenNummer} \text{ s.d. } (n, i) \in \text{AK-Besetzung}(a) \,\Big\}.$$

- Die Bedingung, dass es für jeden Arbeitskreis eine Sprache gibt, die alle Mitglieder des Arbeitskreises beherrschen, lässt sich folgendermaßen formulieren:

$$\text{f.a. } a \in \text{Arbeitskreise ist } \text{Gemeinsame\_Sprachen\_in\_}a \;\neq\; \emptyset,$$

  wobei

$$\text{Gemeinsame\_Sprachen\_in\_}a \;:=\; \Big\{\, \text{sp} \in \text{Sprachen} \;:\; \text{f.a. } d \in \text{AK-Besetzung}(a) \text{ ist sp} \in \text{spricht}(d) \,\Big\}.$$

□Ende Beispiel 2.43

2.5 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.

2.5.1 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.

2.5.2 Beweistechnik "direkter Beweis"

Bei einem *direkten Beweis* wird die Behauptung eines Satzes "direkt", d.h. ohne "Umwege", bewiesen.

Beispiele für direkte Beweise haben wir bereits kennengelernt, z.B. der Beweis von Satz 2.6, der Beweis von Satz 2.21, der Beweis von Satz 2.38, der Beweis von Folgerung 2.39.

2.5.3 Beweistechnik "Beweis durch Kontraposition"

Man beachte, dass für beliebige Aussagen A und B das Folgende gilt:

	die folgende Aussage ist wahr: "Falls Aussage A gilt, so gilt auch Aussage B"
$\Longleftrightarrow$	Aussage B gilt oder Aussage A gilt nicht
$\Longleftrightarrow$	die folgende Aussage ist wahr: "Falls Aussage B **nicht** gilt, so gilt auch Aussage A **nicht**."

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 nur den Fall, dass n **keine** ungerade Zahl ist (d.h. n ist gerade). Wir müssen zeigen, dass dann auch n^2 **keine** ungerade Zahl ist (d.h. n^2 ist eine gerade Zahl).

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:

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

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

2.5.4 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 $k, \ell \in \mathbb{N}$ s.d. $a = 2 \cdot k$ und $b = 2 \cdot \ell$.
Dann ist $a \cdot b = (2 \cdot k) \cdot (2 \cdot \ell)$. Insbesondere gibt es also ein $m \in \mathbb{N}$, s.d. $a \cdot b = 2 \cdot m$.
Gemäß der Definition von "geraden Zahlen" 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 $n := m$ folgt, dass $m \in M$. ↯ (Widerspruch zu "Fall 1: $m \notin M$").

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

Somit führen beide Fälle zu einem Widerspruch. Daher muss unsere Annahme, dass es eine surjektive Funktion f von $\mathbb{N}$ nach $\mathcal{P}(\mathbb{N})$ gibt, falsch gewesen sein. □

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 $N \in U$.
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: $M \in U$. □

Bemerkung 2.48. Jede Aussage, die durch einen *Beweis durch Kontraposition* bewiesen werden kann, kann auch durch einen *Beweis durch Widerspruch* nachgewiesen werden. Um zu zeigen, dass die Aussage

> "Falls Aussage A gilt, so gilt auch Aussage B"

wahr ist, kann man in einem Beweis durch Widerspruch folgendermaßen vorgehen: Man nimmt an, dass Aussage A gilt und Aussage B nicht gilt und leitet aus dieser Annahme dann einen Widerspruch her.
Übung: Beweisen Sie Satz 2.44 durch einen "Beweis durch Widerspruch".

2.5.5 Beweistechnik "Beweis durch vollständige Induktion"

Um die Grundidee der vollständigen Induktion zu erklären, sei $A(n)$ eine Aussage über die natürliche Zahl n. Das Ziel ist, zu zeigen, dass die Aussage $A(n)$ für jedes $n \in \mathbb{N}$ wahr ist.

Induktionsprinzip

Eine Möglichkeit, dies zu zeigen ist, sich das so genannte **Induktionsprinzip** zu Nutze zu machen: 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.
Induktionsanfang — Diesen Schritt nennt man **Induktionsanfang** bzw. Induktionsbasis.

(2) Danach zeigt man, dass für jede beliebige natürliche Zahl $n \in \mathbb{N}$ gilt:
Falls die Aussage $A(n)$ wahr ist, so ist auch die Aussage $A(n+1)$ wahr.
Induktionsschritt — Diesen Schritt nennt man **Induktionsschritt**.

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

(i) $A(0)$ ist wahr gemäß Schritt (1).

(ii) $A(1)$ ist wahr gemäß (i) und Schritt (2) für $n = 0$,

(iii) $A(2)$ ist wahr gemäß (ii) und Schritt (2) für $n = 1$,

(iv) $A(3)$ ist wahr gemäß (iii) und Schritt (2) für $n = 2$,

(v) $A(4)$ ist wahr gemäß (iv) und Schritt (2) für $n = 3$,

(vi) $A(5)$ ist wahr gemäß (v) und Schritt (2) für $n = 4$,

(vii) usw.

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

Als Beispiel für einen Beweis durch vollständige Induktion betrachten wir den folgenden Satz:

Satz 2.49. F.a. $n \in \mathbb{N}$ gilt:

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

Beweis: Per Induktion nach n.
Die "Aussage $A(n)$", deren Gültigkeit hier f.a. $n \in \mathbb{N}$ bewiesen werden soll, besagt:

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

INDUKTIONSANFANG: $n = 0$

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

Beweis:

Es gilt: $\sum_{i=0}^{0} 2^i \;=\; 2^0 \;=\; 1.$

Außerdem gilt: $2^{0+1} - 1 \;=\; 2^1 - 1 \;=\; 2 - 1 \;=\; 1.$

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

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:

$$\begin{aligned}
\sum_{i=0}^{n+1} 2^i &\overset{\text{Not.2.20}}{=} \left(\sum_{i=0}^{n} 2^i\right) + 2^{n+1} \\
&\overset{\text{Ind.ann.}}{=} (2^{n+1} - 1) + 2^{n+1} \\
&= 2 \cdot 2^{n+1} - 1 \\
&= 2^{(n+1)+1} - 1.
\end{aligned}$$

□

Zwei nützliche Varianten des Induktionsprinzips:

Um zu zeigen, dass eine Aussage $A(n)$ für alle $n \in \mathbb{N}$ mit $n \geqslant n_0$ 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 $n \geqslant n_0$ 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 $n \geqslant n_0$ beliebig.

Induktionsannahme: Für jede natürliche Zahl i mit $n_0 \leqslant i \leqslant n$ ist die Aussage $A(i)$ wahr.

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

Beweis: ...

Beispiel 2.50.
Wir nutzen Variante 1, um die folgende Frage zu beantworten: 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?

Um eine Vermutung darüber zu bekommen, welche der beiden Funktionen die größeren Werte liefert, stellen wir zunächst eine Tabelle auf, die die Funktionswerte für $n = 0$, $n = 1$, $n = 2$, etc. enthält:

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

Anhand dieser Tabelle drängt sich die Vermutung auf, dass f.a. $n \in \mathbb{N}$ mit $n \geqslant 6$ gilt: $f(n) > g(n)$. Die Korrektheit dieser Vermutung weisen wir im Folgenden per Induktion nach n nach.

INDUKTIONSANFANG: $n = 6$

Behauptung: $f(6) > g(6)$

Beweis:
Es gilt: $f(6) = 6^2 - 7 = 29$.
Außerdem gilt: $g(6) = 4 \cdot 6 = 24$.
Also: $f(6) = 29 > 24 = g(6)$.

INDUKTIONSSCHRITT: $n \to n+1$

Sei $n \in \mathbb{N}$ mit $n \geqslant 6$ 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:

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

Insgesamt haben wir damit bewiesen, dass f.a. $n \in \mathbb{N}$ mit $n \geqslant 6$ gilt: $f(n) > g(n)$.

□Ende Beispiel 2.50

Auf ähnliche Weise kann man per Induktion auch Folgendes beweisen:

Satz 2.51.

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

(b) F.a. $n \in \mathbb{N}$ mit $n \geqslant 1$ 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 $n \geqslant 1$ 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 sehr sorgfältig sein muss:

Beispiel 2.52.
Der folgende Satz ist offensichtlich nicht wahr — aber wo steckt der Fehler im Beweis?

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

"Beweis": 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 $n \geqslant 1$ beliebig.

Induktionsannahme: Ist M' eine Menge von Menschen mit $|M'| = n$, so haben alle Menschen in M' die gleiche Größe.

Behauptung: Ist M eine Menge von Menschen mit $|M| = n+1$, so haben alle Menschen in M die gleiche Größe.

Beweis: Sei M eine Menge von Menschen mit $|M| = n+1$. Sei $a_1, a_2, \ldots, a_n, a_{n+1}$ eine Liste aller Menschen in M, d.h. $M = \{a_1, a_2, \ldots, a_n, a_{n+1}\}$. Sei

$$M' := \{a_1, a_2, \ldots, a_n\} \quad \text{und} \quad M'' := \{a_2, \ldots, a_n, a_{n+1}\}.$$

Offensichtlich sind M' und M'' Mengen von Menschen mit $|M'| = n$ 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 2.52

2.6 Rekursive Definitionen von Funktionen und Mengen

2.6.1 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 — Diesen Schritt bezeichnet man als **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 — Diesen Schritt bezeichnet man als **Rekursionsschritt**.

Auch hier sind wieder eine Reihe von Varianten möglich.

Beispiel 2.53.

(a) *Frage:* Wie viele Möglichkeiten gibt es, n Studierende so an n PCs zu verteilen, dass an jedem PC genau ein Studierender sitzt?

Antwort: 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 fak eine Funktion von $\mathbb{N}_{>0}$ nach $\mathbb{N}_{>0}$, d.h. $\mathrm{fak}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$.

Beispielsweise ist

$$\mathrm{fak}(4) \;=\; 4\cdot \mathrm{fak}(3) \;=\; 4\cdot 3\cdot \mathrm{fak}(2) \;=\; 4\cdot 3\cdot 2\cdot \mathrm{fak}(1) \;=\; 4\cdot 3\cdot 2\cdot 1 \;=\; 24.$$

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

$$\mathrm{fak}(n) \;=\; n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 2 \cdot 1 \overset{\text{Not. 2.20}}{=} \prod_{i=1}^{n} i.$$

Notation:
Die Funktion fak wird **Fakultätsfunktion** genannt. Meistens schreibt man $n!$ um die Zahl $\mathrm{fak}(n)$ zu bezeichnen.

Fakultätsfunktion
$n!$

(b) *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: $\mathrm{fib}(n)$, wobei die Funktion $\mathrm{fib}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ rekursiv wie folgt definiert ist:

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

Somit gilt:

n	1	2	3	4	5	6	7	8	9	10	11	12
$\mathrm{fib}(n)$	1	1	2	3	5	8	13	21	34	55	89	144

Die Funktion fib wird auch **Fibonacci-Folge** genannt; sie ist benannt nach italienischen Mathematiker Leonardo Fibonacci (13. Jh.). Die Zahl $\mathrm{fib}(n)$ heißt auch **n-te Fibonacci-Zahl**.

Fibonacci-Folge

Um Aussagen über rekursiv definierte Funktionen zu beweisen, kann man wieder das Induktionsprinzip nutzen. Der folgende Satz gibt dazu ein Beispiel.

Satz 2.54. Sei $\mathrm{fib}\colon \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ die Fibonacci-Folge. Dann gilt f.a. $n \in \mathbb{N}_{>0}$: $\mathrm{fib}(n) \leqslant 2^n$.

Beweis: Per Induktion nach n.

Induktionsanfang: Betrachte $n = 1$ und $n = 2$.

Behauptung: $\mathrm{fib}(1) \leqslant 2^1$ und $\mathrm{fib}(2) \leqslant 2^2$.

Beweis: Es gilt: $\mathrm{fib}(1) \overset{\text{Def.}}{=} 1 \leqslant 2 = 2^1$ und $\mathrm{fib}(2) \overset{\text{Def.}}{=} 1 \leqslant 4 = 2^2$.

INDUKTIONSSCHRITT: $n \to n+1$

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

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

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

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

Bemerkung 2.55. 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 $n \geqslant 3$, 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

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

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

Ein anderer Algorithmus, der für eine Zahl $n \in \mathbb{N}_{>0}$ den Wert $\text{fib}(n)$ berechnet, 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).

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

Mit den in diesem Kapitel bereitgestellten Werkzeugen können wir eine Antwort auf diese Frage finden, und wir können sogar beiweisen, dass die Antwort korrekt ist. Details dazu finden sich in Aufgabe 2.14.

Bemerkung 2.56. Es gibt auch eine "geschlossene Formel", mit der man die n-te Fibonacci-Zahl, 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 finden sich in [22]). □

2.6.2 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:

$$\text{"Wenn } m_1, \ldots, m_k \in M, \text{ dann } m \in M\text{"},$$

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 2.57 (Die Menge PAL).
Betrachte das Alphabet $A := \{a, b\}$. Die Menge $\text{PAL} \subseteq A^*$ sei wie folgt rekursiv definiert:

Basisregeln:

(B1): $\varepsilon \in \text{PAL}$.

(B2): $a \in \text{PAL}$.

(B3): $b \in \text{PAL}$.

Rekursive Regeln:

(R1): Ist $w \in \text{PAL}$, so ist auch $awa \in \text{PAL}$.

(R2): Ist $w \in \text{PAL}$, so ist auch $bwb \in \text{PAL}$.

Beispiele für Worte, die zur Menge PAL gehören:

$$\underbrace{\varepsilon,\ a,\ b}_{\text{durch Basisregeln}} \qquad \underbrace{aa,\ bb}_{\text{durch rek. Regeln mit } w := \varepsilon} \qquad \underbrace{aaa,\ bab}_{\text{durch rek. Regeln mit } w := a} \qquad \underbrace{aba,\ bbb}_{\text{durch rek. Regeln mit } w := b}$$

Es gilt beispielsweise auch: $aababaa \in \text{PAL}$.

Beweis:

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

□

Aber beispielsweise gilt

$$aab \notin \text{PAL},$$

denn aus den Basisregeln und den rekursiven Regeln folgt, dass für jedes Wort $w \in \text{PAL}$ der erste und der letzte Buchstabe von w identisch sind. □Ende Beispiel 2.57

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.
Dieser Schritt heißt **Induktionsanfang**.

(2) Danach betrachtet man nacheinander jede rekursive Regel der Form "Wenn $m_1, \ldots, m_k \in M$, dann $m \in M$" und zeigt folgendes: Wenn die Aussagen $A(m_1), \ldots, A(m_k)$ wahr sind, dann ist auch die Aussage $A(m)$ wahr.
Dieser Schritt heißt **Induktionsschritt**.

Beachte: Man kann leicht sehen, dass folgendes gilt: Wenn man die Schritte (1) und (2) bewiesen hat, so weiß man, dass die Aussage $A(m)$ für alle $m \in M$ wahr ist.

Im Folgenden betrachten wir ein Beispiel dafür, wie das Induktionsprinzip dazu genutzt werden kann, Eigenschaften von rekursiv definierten Mengen nachzuweisen.

Beispiel 2.58 (Palindrome).
Sei $A := \{a, b\}$. Für jedes Wort $w \in A^*$ sei w^R das Wort, das durch "Rückwärtslesen" von w entsteht, d.h.:

- Ist $w = \varepsilon$, so ist $w^R = \varepsilon$.
- Ist $w = w_1 \cdots w_k$ mit $k \in \mathbb{N}_{>0}$ und $w_1, \ldots, w_k \in A$, so ist $w^R := w_k \cdots w_1$.

Beispiel: $aaab^R = baaa$.

Sei PAL die im Beispiel 2.57 rekursiv definierte Teilmenge von A^*.

Behauptung 1: Für jedes Wort $w \in \text{PAL}$ gilt: $w = w^R$.

Beweis: Per Induktion über den Aufbau von PAL.

Induktionsanfang: Betrachte diejenigen Worte, die aufgrund von Basisregeln zur Menge PAL gehören.

Behauptung: $\varepsilon = \varepsilon^R$, $a = a^R$ und $b = b^R$.

Beweis: Gemäß der Definition von w^R gilt offensichtlich, dass $\varepsilon = \varepsilon^R$, $a = a^R$ und $b = b^R$.

Induktionsschritt: Betrachte die rekursiven Regeln.

- (R1): Sei $w \in \text{PAL}$ und sei $v := awa$. Gemäß (R1) ist $v \in \text{PAL}$.

  Induktionsannahme: $w = w^R$.

  Behauptung: $v = v^R$.

  Beweis: $v^R \overset{\text{Def. } v}{=} (awa)^R \overset{\text{Def. } (\cdot)^R}{=} aw^R a \overset{\text{Ind.ann.: } w = w^R}{=} awa \overset{\text{Def. } v}{=} v.$

- (R2): Sei $w \in \mathrm{PAL}$ und sei $v := bwb$. Gemäß (R2) ist $v \in \mathrm{PAL}$.

  Induktionsannahme: $w = w^R$.

  Behauptung: $v = v^R$.

  Beweis: $v^R \overset{\text{Def. } v}{=} (bwb)^R \overset{\text{Def. } (\cdot)^R}{=} bw^Rb \overset{\text{Ind.ann.: } w = w^R}{=} bwb \overset{\text{Def. } v}{=} v.$ $\square_{\text{Beh. 1}}$

Behauptung 2: Für jedes $w \in A^*$ mit $w = w^R$ gilt: $w \in \mathrm{PAL}$.

Beweisansatz: Zeige folgende Aussage per Induktion nach n:

Für alle $n \in \mathbb{N}$ gilt: Ist $w \in A^*$ mit $w = w^R$ und $|w| \leqslant n$, so gilt $w \in \mathrm{PAL}$.

Im Induktionsanfang werden $n = 0$ und $n = 1$ betrachtet; im Induktionsschritt $n \to n+1$ werden alle $n \geqslant 1$ betrachtet.
Details: Übung. $\square_{\text{Beh. 2}}$

Aus Behauptung 1 und Behauptung 2 folgt, dass $\mathrm{PAL} = \{w \in A^* : w = w^R\}$.

$\square_{\text{Ende Beispiel 2.58}}$

Antwort auf die Frage aus Beispiel 2.52:

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

- $M = \{a_1, a_2\}$,
- $M' = \{a_1\}$,
- $M'' = \{a_2\}$.

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

2.7 Literaturhinweise zu Kapitel 2

Als vertiefende Lektüre seien die Kapitel 3, 6 und 7 in [22] empfohlen. Wertvolle Tipps und Tricks zur Formulierung mathematischer Gedanken und Beweise finden sich in [2]. Einen Crashkurs in die diskrete Mathematik für Informatiker/innen gibt das Buch [14]. Eine umfassende Einführung in die Mengenlehre gibt das Lehrbuch [6].

Quellennachweis: Teile der Abschnitte 2.1–2.3 sowie 2.6 orientieren sich an [9]. Das in Abschnitt 2.4 betrachtete Beispiel ist aus [15] entnommen. Teile von Abschnitt 2.5 orientieren sich an [22]. Die folgende Aufgabe 2.10 ist aus [15] entnommen.

2.8 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) $\mathcal{P}(\{\emptyset\})$	(e) $M \times \{a, b\}$	(f) $\{M\} \times \{a, b\}$
(g) $\{P : P \subseteq N \text{ und } \lvert P\rvert = 2\}$	(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) $M \setminus (N \cup P)$

(b) $(M \setminus N) \cup (M \setminus P)$

(c) $(M \cup N) \cap \overline{P}$

(d) $(M \cap \overline{P}) \cup (N \cap \overline{P})$

(e) $M^2 \setminus (N \times P)$

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

(g) $M \times P \times \{a, b\}$

(h) $\{Q : Q \subseteq N, |Q| = 3\}$

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 $N \subsetneq P$, dann $M \subsetneq P$.

(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?

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

(b) Berechnen Sie die folgenden Mengen:

(i) $A \cup C$ (ii) $A \cap E$ (iii) $B \cap D$ (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 $|A| \geqslant 2$ 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 $n \geqslant 1$ gilt: $\displaystyle\sum_{i=1}^{n} (2i - 1) = n^2$.

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

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

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

Aufgabe 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 $n \geqslant 3$, 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

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

(der Einfachheit halber zählen die Zeilen 1, 2 und 5 hier jeweils nur als ein Schritt).
Ein anderer Algorithmus, der für eine Zahl $n \in \mathbb{N}_{>0}$ den Wert $\text{fib}(n)$ berechnet, ist:

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

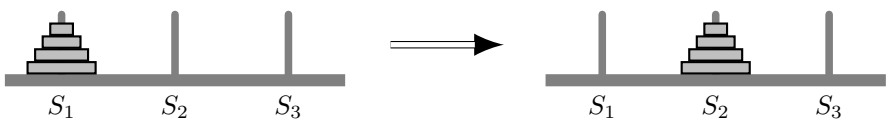
1. Falls $n = 1$ oder $n = 2$, dann gib 1 als Ergebnis zurück.
2. Seien $a_0 := 0$, $a_1 := 1$ und $a_2 := 1$.
3. Wiederhole für alle i von 3 bis n:
4. Ersetze a_0 durch a_1 und a_1 durch a_2.
5. Ersetze a_2 durch $a_0 + a_1$.
6. Gib den Wert a_2 als Ergebnis zurück.

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 $n \geqslant n_0$ 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 $n \geqslant 1$ gilt:

A(n): Seien $i, j \in \{1, 2, 3\}$ mit $i \neq j$, sei $m \in \mathbb{N}$ mit $m \geqslant n$, 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 (nach dem Altgriechischen "Logos": "Vernunft") ist "die Lehre des vernünftigen Schlussfolgerns". Logik ist ein Teilgebiet in den Disziplinen Philosophie, Mathematik, Informatik und Linguistik. Eine zentrale Frage, mit dem sich das Gebiet der Logik beschäftigt ist:

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

In einem gewissen Sinn spielt die Logik in der Informatik eine ähnlich wichtige Rolle wie die Differential- und Integralrechnung in der Physik [20, 10]. Logik wird in der Informatik u.a. genutzt

- zur Repräsentation von statischem Wissen (z.B. im Bereich der künstlichen Intelligenz),
- 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"),
- zur Verifikation von
  - Schaltkreisen (*Ziel*: beweise, dass ein Schaltkreis bzw. Chip "richtig" funktioniert),
  - Programmen (*Ziel*: beweise, dass ein Programm gewisse wünschenswerte Eigenschaften hat),
  - Protokollen (*Ziel*: beweise, dass die Kommunikation zwischen zwei "Agenten", die nach einem gewissen "Protokoll" abläuft, "sicher" ist — etwa gegen Abhören oder Manipulation durch dritte; Anwendungsbeispiel: Internet-Banking).

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

Die **Syntax** legt fest, welche Zeichenketten (Worte) Formeln der Aussagenlogik sind. Die **Semantik** legt fest, welche "Bedeutung" einzelne Formeln haben. Syntax Semantik

Man beachte, dass dies analog zur "Syntax" und "Semantik" von JAVA-Programmen ist: Die Syntax legt fest, welche Zeichenketten JAVA-Programme sind, während die Semantik bestimmt, was das Programm tut.

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 Aussagenvariablen bezeichnen wir mit AVAR. D.h.: Aussagenvariable

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

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

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

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

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

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

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

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

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

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

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

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

- $V_1 \vee V_2 \wedge V_3$ (da die Klammern fehlen),

- $(\neg V_1)$ (da die Klammern "zu viel" sind).

□Ende von Beispiel 3.5

Notation 3.6.

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

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

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

- $(\varphi \wedge \psi)$ **Konjunktion** (bzw. Ver*und*ung) von φ und ψ, — Konjunktion
- $(\varphi \vee \psi)$ **Disjunktion** (bzw. Ver*oder*ung) von φ und ψ, — Disjunktion
- $\neg\varphi$ **Negation** (bzw. Ver*nein*ung) von φ. — Negation

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

Definition 3.7.
Die **Variablenmenge** einer aussagenlogischen Formel φ (kurz: $\mathrm{Var}(\varphi)$) ist die Menge aller Variablen $X \in \mathrm{AVAR}$, die in φ vorkommen. — Variablenmenge $\mathrm{Var}(\varphi)$

Beispiele:

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

Definition 3.8.

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

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

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

Definition 3.9 (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

[1] Die intuitive Bedeutung dabei ist, dass 1 für den Wert "wahr" und 0 für den Wert "falsch" steht.

Rekursionsanfang:

- $[\![\mathbf{0}]\!]^{\mathcal{B}} := 0$.
- $[\![\mathbf{1}]\!]^{\mathcal{B}} := 1$.
- F.a. $X \in \text{AVAR}$ gilt: $[\![X]\!]^{\mathcal{B}} := \mathcal{B}(X)$.

Rekursionsschritt:

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

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

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

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

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

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

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

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

  D.h.: $(\varphi \wedge \psi)$ ist wahr (unter Belegung $\mathcal{B}$) $\iff$ φ ist wahr und ψ ist wahr (unter Belegung $\mathcal{B}$). Zugehörige 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 3.10.
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

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{B}} &\overset{\text{Def. 3.9}}{=} \begin{cases} 1, & \text{falls } [\![\neg V_0]\!]^{\mathcal{B}} = 1 \text{ oder } [\![(V_5 \to V_1)]\!]^{\mathcal{B}} = 1 \\ 0, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 3.9}}{=} \begin{cases} 1, & \text{falls } [\![V_0]\!]^{\mathcal{B}} = 0 \text{ oder } \big([\![V_5]\!]^{\mathcal{B}} = 0 \text{ oder } [\![V_1]\!]^{\mathcal{B}} = 1\big) \\ 0, & \text{sonst} \end{cases} \\
&\overset{\text{Def. 3.9}}{=} \begin{cases} 1, & \text{falls } \mathcal{B}(V_0) = 0 \text{ oder } \mathcal{B}(V_5) = 0 \text{ oder } \mathcal{B}(V_1) = 1 \\ 0, & \text{sonst} \end{cases} \\
&= \quad 1 \qquad (\text{denn gemäß obiger Wahl von } \mathcal{B} \text{ gilt } \mathcal{B}(V_5) = 0).
\end{aligned}$$

Beobachtung 3.11.
Sind $\mathcal{B}$ und $\mathcal{B}'$ zwei Belegungen für eine Formel φ, die auf $\mathrm{Var}(\varphi)$ übereinstimmen (d.h.: f.a. $X \in \mathrm{Var}(\varphi)$ ist $\mathcal{B}(X) = \mathcal{B}'(X)$), so ist $[\![\varphi]\!]^{\mathcal{B}} = [\![\varphi]\!]^{\mathcal{B}'}$.

Koinzidenzlemma

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.

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

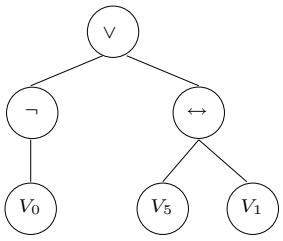
Notation 3.12.

- 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 $[\![\varphi]\!]^{\mathcal{B}} = 1$.

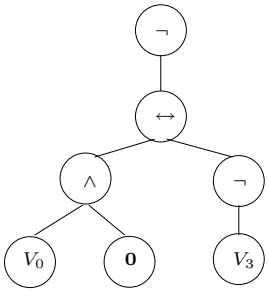
Bemerkung 3.13 (Syntaxbäume zur graphischen Darstellung von Formeln).
Die Struktur einer Formel lässt sich bequem durch einen **Syntaxbaum** (englisch: **parse tree**) darstellen.

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. Zum Beispiel ist

$$\big((V_0 \wedge \mathbf{0}) \rightarrow \neg V_{13}\big)$$

eine Formel in AL, deren ASCII-Repräsentation die folgende Zeichenkette aus AL_{ASCII} ist:

```
( (V0 /\ 0) -> ~V13 ).
```

Wir werden meistens mit der "abstrakten Syntax", d.h. mit der in Definition 3.3 festgelegten Menge AL, arbeiten. Um aber Formeln in Computer-Programme einzugeben, können wir die ASCII-Repräsentation verwenden.

Umgangssprachliche Aussagen lassen sich wie folgt durch aussagenlogische Formeln repräsentieren:

Beispiel 3.16.
Die Zeugenaussage

> "Das Fluchtauto war rot oder grün und hatte weder vorne noch hinten ein Nummernschild."

lässt sich durch die aussagenlogische Formel

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

repräsentieren, die die folgenden atomaren Aussagen nutzt:

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

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 aus Beispiel 3.1 wird durch folgende Formel repräsentiert:

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

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

- φ von $\mathcal{B}$ erfüllt wird, d.h. $[\![\varphi]\!]^{\mathcal{B}} = 1$, und
- $|\{X \in \{A, B, C, D, E\} : \mathcal{B}(X) = 1\}|$ 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 Wahrheitswerte von φ 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 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 viele (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.

A	B	C	D	E	$E \to (C \wedge D)$	$C \to A$	$(B \wedge E) \to \neg D$	$A \to (B \vee C)$	$(B \wedge A) \to \neg E$	φ
0	0	0	0	0	1	1	1	1	1	1
0	0	0	0	1	0	1	1	1	1	0
0	0	0	1	0	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	0
0	0	1	0	0	1	0	1	1	1	0
0	0	1	0	1	0	0	1	1	1	0
0	0	1	1	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1	1	1	0
0	1	0	0	0	1	1	1	1	1	1
0	1	0	0	1	0	1	1	1	1	0
0	1	0	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1	1	1	0
0	1	1	0	1	0	0	1	1	1	0
0	1	1	1	0	1	0	1	1	1	0
0	1	1	1	1	1	0	0	1	1	0
1	0	0	0	0	1	1	1	0	1	0
1	0	0	0	1	0	1	1	0	1	0
1	0	0	1	0	1	1	1	0	1	0
1	0	0	1	1	0	1	1	0	1	0
1	0	1	0	0	1	1	1	1	1	1
1	0	1	0	1	0	1	1	1	1	0
1	0	1	1	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	0	1	1	1	1	1	1
1	1	0	1	1	0	1	0	1	0	0
1	1	1	0	0	1	1	1	1	1	1
1	1	1	0	1	0	1	1	1	0	0
1	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	0	1	0	0

Abbildung 3.1: Wahrheitstafel für die Formel φ aus Beispiel 3.17

Prinzipiell führt diese Vorgehensweise zum Ziel. Leider ist das Verfahren aber recht aufwendig, da die Wahrheitstafel, die man dabei aufstellen muss, sehr groß wird, wie man am Beispiel der Wahrheitstafel für die Formel φ (siehe Abbildung 3.1) sieht. **Erfüllende Belegungen** für φ werden in Abbildung 3.1 durch Zeilen repräsentiert, die grau unterlegt sind.

In der Wahrheitstafel sieht man, dass es **keine** erfüllende Belegung gibt, bei der in den mit A bis E beschrifteten Spalten insgesamt 5 Einsen stehen, und dass es genau **zwei** erfüllende 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 Antwort auf die Frage “Wie viele (und welche) Freunde werden bestenfalls zur Party kommen?” lautet also: Bestenfalls werden 4 der 5 Freunde kommen, und dafür gibt es zwei Möglichkeiten, nämlich

(1) dass alle außer Bernd kommen, und

(2) dass alle außer Eva kommen. □Ende Beispiel 3.19

Angesichts der Wahrheitstafel aus Abbildung 3.1 stellt sich die Frage, wie groß die Wahrheitstafel für eine gegebene Formel φ ist. Die Antwort darauf gibt der folgende Satz.

Satz 3.20.
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:

$$\begin{aligned} &\{\mathcal{B} : \mathcal{B} \text{ ist eine zu } \varphi \text{ passende Belegung mit } \mathrm{Def}(\mathcal{B}) = \mathrm{Var}(\varphi)\} \\ \overset{\text{Def. 3.8}}{=}\ &\{\mathcal{B} : \mathcal{B}\colon\ \mathrm{Var}(\varphi) \to \{0,1\} \text{ ist eine Funktion}\} \\ \overset{\text{Not. 2.30}}{=}\ &\mathrm{Abb}(\mathrm{Var}(\varphi), \{0,1\}). \end{aligned}$$

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 $13,7$ Milliarden Jahre (das sind ungefähr 10^{18} Sekunden) geschätzt.

3.3 Erfüllbarkeit und Allgemeingültigkeit

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 $\mathcal{B}$ mit $[\![\varphi]\!]^{\mathcal{B}} = 1$ gibt. — 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. Beachte:

Falls $\mathcal{B}(X) = 1$, so gilt:

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

Falls $\mathcal{B}(X) = 0$, so gilt:

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

(c) Die Formel $(X \vee \neg X)$ ist **allgemeingültig**, da für jede zur Formel passenden Belegung $\mathcal{B}$ entweder $\mathcal{B}(X) = 1$ oder $\mathcal{B}(X) = 0$ gilt. Somit gilt für alle zur Formel passenden Belegungen $\mathcal{B}$, dass $[\![(X \vee \neg X)]\!]^{\mathcal{B}} = 1$.

Beobachtung 3.23. Für jede aussagenlogische Formel φ gilt:

(a)	φ ist erfüllbar	$\iff$	in der Wahrheitstafel für φ steht in der mit "φ" beschrifteten Spalte mindestens eine 1.
(b)	φ ist unerfüllbar	$\iff$	in der Wahrheitstafel für φ stehen in der mit "φ" beschrifteten Spalte nur Nullen.
(c)	φ ist allgemeingültig	$\iff$	in der Wahrheitstafel für φ stehen in der mit "φ" beschrifteten Spalte nur Einsen.
(d)	φ ist allgemeingültig	$\iff$	$\neg\varphi$ ist unerfüllbar.

aussagenlogisches Erfüllbarkeitsproblem (SAT)

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

AUSSAGENLOGISCHES ERFÜLLBARKEITSPROBLEM (SAT)
Eingabe: Eine aussagenlogische Formel φ.
Frage: Ist φ erfüllbar?

Natürlich kann man dieses Problem dadurch lösen, dass man zur gegebenen Formel φ die Wahrheitstafel aufstellt und testet, ob es in der mit "φ" beschrifteten Spalte mindestens eine 1 gibt. Satz 3.20 und die darauf folgende Bemerkung über die Größe von Wahrheitstafeln besagen allerdings, dass dieses Verfahren recht aufwändig ist.

Ein unter dem Stichwort **SAT-Solving** bekannter Teilbereich der Informatik beschäftigt sich mit der Aufgabe, Verfahren zu entwickeln, die das aussagenlogische Erfüllbarkeitsproblem lösen und dabei wesentlich effizienter sind als das vorgestellte Wahrheitstafel-Verfahren. Ein relativ ernüchterndes Resultat, das Sie in weiterführenden Veranstaltungen der Theoretischen Informatik kennenlernen werden, ist allerdings der folgende Satz:

Satz von Cook (1971) (Stephen A. Cook, * 1939, Professor an der University of Toronto)
Das aussagenlogische Erfüllbarkeitsproblem ist NP-vollständig.

Eine präzise Definition des des Begriffs "NP-vollständig" zu geben, würde den Rahmen dieses Vorlesungsskripts sprengen. 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 für viele Eingabe-Formeln erstaunlich effizient lösen können. Details zum Thema NP-Vollständigkeit und zum Satz von Cook finden sich in den Büchern [26, 29].

3.4 Folgerung und Äquivalenz

Definition 3.24 (semantische Folgerung).
Seien φ und ψ zwei aussagenlogische Formeln. Wir sagen ψ **folgt aus** φ (kurz: $\varphi \models \psi$ bzw. φ **impliziert** ψ), falls für jede zu φ und ψ passende Belegung $\mathcal{B}$ gilt: Falls $[\![\varphi]\!]^{\mathcal{B}} = 1$, so auch $[\![\psi]\!]^{\mathcal{B}} = 1$. Somit gilt:

ψ folgt aus φ
$\varphi \models \psi$

$$\varphi \models \psi \iff \text{Jede Belegung, die zu } \varphi \text{ und } \psi \text{ passt und die } \varphi \text{ erfüllt, erfüllt auch } \psi.$$

Beispiel 3.25.
Sei $\varphi := \big((X \vee Y) \wedge (\neg X \vee Y)\big)$ und $\psi := \big(Y \vee (\neg X \wedge \neg Y)\big)$.
Dann gilt $\varphi \models \psi$, aber es gilt **nicht** $\psi \models \varphi$ (kurz: $\psi \not\models \varphi$), denn:

X	Y	$(X \vee Y)$	$(\neg X \vee Y)$	φ	ψ
0	0	0	1	0	1
0	1	1	1	1	1
1	0	1	0	0	0
1	1	1	1	1	1

Hier repräsentiert jede Zeile eine zu φ und ψ passende Belegung. In jeder Zeile, in der in der mit "φ" beschrifteten Spalte eine 1 steht, steht auch in der mit "ψ" beschrifteten Spalte eine 1. Somit gilt $\varphi \models \psi$.

Andererseits steht in Zeile 1 in der mit "ψ" beschrifteten Spalte eine 1 und in der mit "φ" beschrifteten Spalte eine 0. Für die entsprechende Belegung $\mathcal{B}$ (mit $\mathcal{B}(X) = 0$ und $\mathcal{B}(Y) = 0$) gilt also $[\![\psi]\!]^{\mathcal{B}} = 1$ und $[\![\varphi]\!]^{\mathcal{B}} = 0$. Daher gilt $\psi \not\models \varphi$.

Beobachtung 3.26. Seien φ und ψ beliebige 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 3.27 (logische Äquivalenz).
äquivalent
$\varphi \equiv \psi$
Zwei aussagenlogische Formeln φ und ψ heißen **äquivalent** (kurz: $\varphi \equiv \psi$), wenn für alle zu φ und ψ passenden Belegungen $\mathcal{B}$ gilt: $[\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}}$.

Beispiel 3.28. Sei $\varphi := \big(X \wedge (X \vee Y)\big)$ und $\psi := X$. Dann ist $\varphi \equiv \psi$, denn

X	Y	$(X \vee Y)$	φ	ψ
0	0	0	0	0
0	1	1	0	0
1	0	1	1	1
1	1	1	1	1

Hier ist die mit "φ" beschriftete Spalte identisch zur mit "ψ" beschrifteten Spalte. D.h. für alle zu φ und ψ passenden Belegungen $\mathcal{B}$ gilt $[\![\varphi]\!]^{\mathcal{B}} = [\![\psi]\!]^{\mathcal{B}}$. Somit gilt $\varphi \equiv \psi$.

Beobachtung 3.29. 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 3.30. 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 3.31. Durch schrittweises Anwenden der in Satz 3.30 aufgelisteten Äquivalenzen kann man eine gegebene aussagenlogische Formel in eine zu ihr äquivalente Formel umformen.

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

$$\begin{aligned} (\varphi \leftrightarrow \psi) &\equiv \big((\varphi \to \psi) \wedge (\psi \to \varphi)\big) && \text{(Satz 3.30(l))} \\ &\equiv \big((\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)\big) && \text{(Satz 3.30(k))} \end{aligned}$$

3.5 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 3.32. 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	$(\ X \wedge \neg Y \wedge\ Z)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Wir erhalten dadurch die folgende 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 3.33 (disjunktive Normalform, konjunktive Normalform).

Literal
positives Literal
negatives Literal

(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 $x \in \text{AVAR}$, heißt **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

disjunktive Normalform DNF

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

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

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

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

konjunktive Normalform KNF

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

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

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

disjunktive Klausel

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 3.34.
Für jede aussagenlogische Formel φ gibt es eine Formel ψ_D in DNF und eine Formel ψ_K in KNF, so dass $\varphi \equiv \psi_D$ und $\varphi \equiv \psi_K$.

Das heißt: 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 3.32 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 $\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} \ell_{i,j} \right)$ die Gestalt von ψ'_D.

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

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

Offensichtlich ist ψ_K eine Formel in KNF. Außerdem gilt:

$$\begin{aligned}
\varphi &\equiv \neg\varphi' \\
&\equiv \neg\psi'_D \\
&\equiv \neg\left(\bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} \ell_{i,j} \right) \right) \\
&\overset{\text{Satz 3.30(g)}}{\equiv} \left(\bigwedge_{i=1}^{n} \neg\left(\bigwedge_{j=1}^{m_i} \ell_{i,j} \right) \right) \\
&\overset{\text{Satz 3.30(g)}}{\equiv} \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} \neg\ell_{i,j} \right) \\
&\overset{\text{Def. } \tilde{\ell}_{i,j}}{\equiv} \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} \tilde{\ell}_{i,j} \right) \\
&\overset{\text{Def. } \psi_K}{=} \psi_K.
\end{aligned}$$

□

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

Negationsnormalform
NNF

Definition 3.35 (Negationsnormalform).
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.

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

Basisregeln:

- Für jedes $X \in \textsc{Avar}$ 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 3.36. Jede Formel, die in KNF oder in DNF ist, ist auch in NNF. Aus Satz 3.34 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.

Beobachtung 3.37. 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 3.30(g)) und der Regel für "doppelte Negation" (Satz 3.30(f)):

- Mit den De Morganschen Regeln

$$\neg(\varphi \wedge \psi) \;\equiv\; (\neg\varphi \vee \neg\psi) \qquad \text{bzw.} \qquad \neg(\varphi \vee \psi) \;\equiv\; (\neg\varphi \wedge \neg\psi)$$

  ziehen wir das Negationszeichen nach innen.

- Mit der Regel für "doppelte Negation"

$$\neg\neg\varphi \;\equiv\; \varphi$$

  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 3.30(k)

$$(\varphi \rightarrow \psi) \;\equiv\; (\neg\varphi \vee \psi)$$

  und Satz 3.30(l)

$$(\varphi \leftrightarrow \psi) \;\equiv\; ((\varphi \rightarrow \psi) \vee (\psi \rightarrow \varphi)).$$

- Eventuelle Vorkommen der Symbole **0** bzw. **1** ersetzen wir durch die Formeln

$$(V_0 \wedge \neg V_0) \qquad \text{bzw.} \qquad (V_0 \vee \neg V_0).$$

Beispiel 3.38.
Das Ziel ist, 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 zu bringen, d.h. eine zur gegebenen Formel äquivalente Formel in NNF zu finden.

Lösung: (der Teil einer Formel, der als nächstes ersetzt wird, ist im Folgenden jeweils unterstrichen)

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

Diese Formel ist offensichtlicherweise in Negationsnormalform.

Unter zusätzlicher Verwendung der "Distributivitätsregel" (Satz 3.30(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) Falls φ' nicht in KNF ist, so ersetze eine Teilformel von φ' der Gestalt $\big(\psi_1 \vee (\psi_2 \wedge \psi_3)\big)$ durch die Formel

$$\big((\psi_1 \vee \psi_2) \wedge (\psi_1 \vee \psi_3)\big)$$

oder ersetze eine Teilformel von φ' der Gestalt $\big((\psi_2 \wedge \psi_3) \vee \psi_1\big)$ durch die Formel

$$\big((\psi_2 \vee \psi_1) \wedge (\psi_3 \vee \psi_1)\big).$$

Sei φ'' die resultierende Formel. 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) Falls φ' nicht in DNF ist, so ersetze eine Teilformel von φ' der Gestalt $\big(\psi_1 \wedge (\psi_2 \vee \psi_3)\big)$ durch die Formel

$$\big((\psi_1 \wedge \psi_2) \vee (\psi_1 \wedge \psi_3)\big)$$

oder ersetze eine Teilformel von φ' der Gestalt $\big((\psi_2 \vee \psi_3) \wedge \psi_1\big)$ durch die Formel

$$\big((\psi_2 \wedge \psi_1) \vee (\psi_3 \wedge \psi_1)\big).$$

Sei φ'' die resultierende Formel. 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 einer aussagenlogischen Formel φ nach endlich vielen Schritten an und gibt eine zu φ äquivalente Formel in KNF aus.

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

Beweis: Übung. □

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

Transformation von φ in NNF:

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

Transformation von φ in DNF mittels Algorithmus 3.40:

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

(2) Einmaliges Anwenden von Zeile (ii) des Algorithmus (auf die unterstrichene Teilformel von φ') liefert:

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

Diese Formel ist die DNF-Formel, die von dem Algorithmus ausgegeben wird (die einzelnen konjunktiven Klauseln sind jeweils unterstrichen).

Transformation von φ in KNF mittels Algorithmus 3.39:

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

(2) Einmaliges Anwenden von Zeile (ii) des Algorithmus (auf den unterstrichenen Teil der Formel φ') liefert:

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

Dies ist die KNF-Formel, die von dem Algorithmus ausgegeben wird (die einzelnen disjunktiven Klauseln sind jeweils unterstrichen).

Am Ende von Abschnitt 3.3 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:

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

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

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

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

ist. Offensichtlich gilt:

$$\varphi \text{ ist erfüllbar} \iff \text{für mindestens ein } i \in \{1, \ldots, n\} \text{ ist die Formel } \kappa_i \text{ erfüllbar.}$$

Da κ_i eine Konjunktion von Literalen (d.h. von Variablen und/oder negierten Variablen) ist, gilt:

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

Daher testet der folgende Algorithmus, ob eine gegebene DNF-Formel erfüllbar ist.

Eingabe: Eine aussagenlogische Formel $\varphi = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} \ell_{i,j} \right)$ 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 $\ell_{i,j} = \neg \ell_{i,j'}$ oder $\ell_{i,j'} = \neg \ell_{i,j}$, 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 $2n$ 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. (Details finden sich in Aufgabe 3.15.) □

3.6 Literaturhinweise

Als vertiefende Lektüre seien die Kapitel 1 und 2 in [17] sowie die Einleitung und Kapitel 1 in [27] empfohlen. Einen umfassenden Überblick über die Rolle der Logik in der Informatik gibt [10]. Details zum Thema NP-Vollständigkeit und zum Satz von Cook finden sich in den Büchern [26, 29].

Quellennachweis: Teile dieses Kapitels orientieren sich an [9]. Die folgende Aufgabe 3.3 ist aus [17] entnommen. Beipiel 3.1 ist eine Variante einer Übungsaufgabe aus [17].

3.7 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 3.10.

(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:

- $\varphi_A := \big(X_A \leftrightarrow (\neg X_B \vee \neg X_C)\big)$
- $\varphi_B := \big(X_B \leftrightarrow (X_A \rightarrow X_C)\big)$
- $\varphi_C := \big(X_C \leftrightarrow (\neg X_B \rightarrow X_A)\big)$

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

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

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.

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

(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 3.26 (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

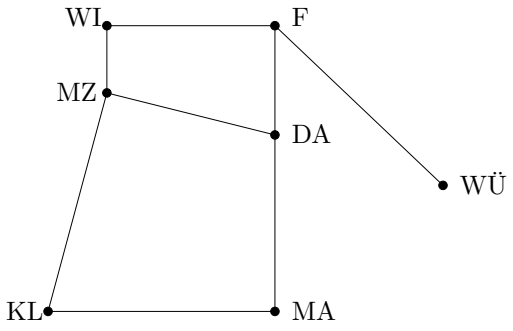
Bei Modellierungsaufgaben geht es oft darum, **Objekte** sowie **Beziehungen** zwischen Objekten zu beschreiben. Graphen und Bäume eignen sich dazu oft besonders gut.
Anschaulich besteht ein Graph aus **Knoten** und **Kanten:**

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

Je nach Aufgabenstellung werden **ungerichtete** Graphen oder **gerichtete** Graphen verwendet. Bäume sind Graphen mit bestimmten Eigenschaften.

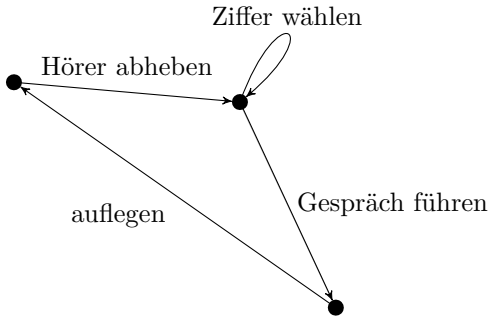
Beispiel 4.1.

(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).

ungerichteter Graph

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

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

Knoten
Kanten

die **Kantenmenge** von G genannt wird. Die Elemente aus V heißen **Knoten** von G (auch: "Ecken"; englisch: **vertices**, singular: vertex); die Elemente aus E heißen **Kanten** von G (englisch: **edges**, singular: edge).

endlich

Ein ungerichteter Graph G heißt **endlich**, falls seine Knotenmenge endlich ist.

Beispiel 4.3. $G = (V, E)$ mit

$$V := \big\{\mathrm{MZ}, \mathrm{WI}, \mathrm{MA}, \mathrm{DA}, \mathrm{KL}, \mathrm{F}, \mathrm{W\ddot{U}}\big\} \text{ und}$$
$$E := \big\{\{\mathrm{MZ}, \mathrm{WI}\}, \{\mathrm{WI}, \mathrm{F}\}, \{\mathrm{F}, \mathrm{DA}\}, \{\mathrm{F}, \mathrm{W\ddot{U}}\}, \{\mathrm{MZ}, \mathrm{DA}\}, \{\mathrm{MZ}, \mathrm{KL}\}, \{\mathrm{KL}, \mathrm{MA}\}, \{\mathrm{DA}, \mathrm{MA}\}\big\}$$

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: In der Literatur wird zumeist die oben genannte Definition von ungerichteten Graphen verwendet. Davon abweichend erlauben einige Bücher in ungerichteten Graphen aber auch "Schleifen" der Form

Notation 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
- Zwei Knoten $v, v' \in V$ heißen **benachbart** (bzw. **adjazent**), falls es eine Kante $e \in E$ gibt, deren Endknoten v und v' sind (d.h. $e = \{v, v'\}$). — benachbart, adjazent

Definition 4.5 (Grad).
Sei $G = (V, E)$ ein endlicher 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. — Grad, $\mathrm{Grad}_G(v)$

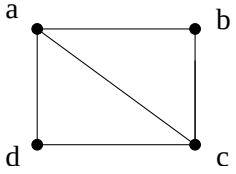
$$\mathrm{Grad}_G(v) \;=\; |\{e \in E : v \in e\}|.$$

Der **Grad von G** ist

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

d.h. $\mathrm{Grad}(G)$ gibt den maximalen Grad eines Knotens von G an.[1]

Beispiel:

Für den Graphen $G =$ 

gilt: $\mathrm{Grad}_G(a) = 3$, $\mathrm{Grad}_G(b) = 2$, $\mathrm{Grad}_G(c) = 3$, $\mathrm{Grad}_G(d) = 2$ und $\mathrm{Grad}(G) = 3$.

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 — gerichteter Graph

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

die **Kantenmenge** von G genannt wird. Die Elemente aus V heißen **Knoten** (bzw. "Ecken"), die Elemente aus E heißen (gerichtete) **Kanten** von G. — Knoten, (gerichtete) Kante
Ein gerichteter Graph G heißt **endlich**, falls seine Knotenmenge endlich ist. — endlich

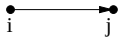
Beispiel 4.7. $G = (V, E)$ mit

$$V \;:=\; \{a, b, c\} \;\text{ und}$$
$$E \;:=\; \big\{(a, b), (b, b), (b, c), (c, a), (a, c)\big\}$$

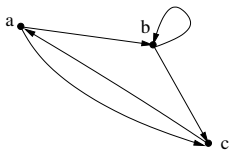
ist ein gerichteter Graph.

In der **graphischen Darstellung** eines gerichteten Graphen werden Knoten werden als Punkte dargestellt. Eine Kante der Form (i, j) wird als Pfeil von Knoten i nach Knoten j dargestellt, also

[1] Ist M eine endliche, nicht-leere Menge von Zahlen, so bezeichnet $\max M$ das größte Element von M.



Der gerichtete Graph aus Beispiel 4.7 lässt sich in graphischer Darstellung also wie folgt darstellen:



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

Ausgangsknoten
Endknoten

- Ist $e = (i, j) \in E$, so heißt i der **Ausgangsknoten** von e und j der **Endknoten** von e, und wir sagen, dass e **von** i **nach** j **verläuft**.

inzident

- Ein Knoten V heißt **inzident** mit einer Kante $e \in E$, falls v der Ausgangs- oder der Endknoten von e ist.

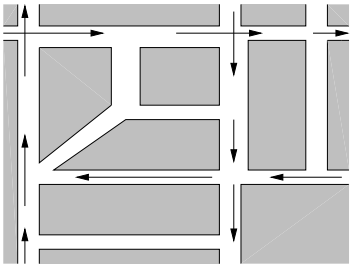
benachbart
adjazent

- Zwei Knoten $v, v' \in V$ heißen **benachbart** (bzw. **adjazent**), falls $(v, v') \in E$ oder $(v', v) \in E$.

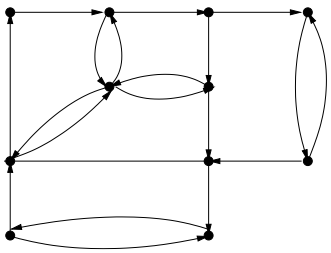
Schleife

- Eine Kante der Form (v, v) wird **Schleife** genannt. D.h.: Eine Schleife ist eine Kante, deren Ausgangs- und Endknoten identisch ist.

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
  (Details dazu finden sich in Kapitel 5.)

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

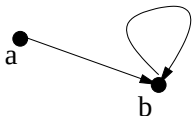
- Der **Ausgangsgrad** von v in G (engl.: out-degree), kurz: Aus-Grad$_G(v)$, ist die Anzahl der Kanten, die v als Ausgangsknoten haben. D.h.: Ausgangsgrad Aus-Grad$_G(v)$

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

- 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.: Eingangsgrad Ein-Grad$_G(v)$

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

Beispiel:

Für den Graphen $G =$ 

gilt:

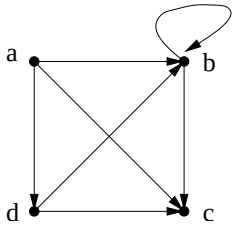
$\text{Ein-Grad}_G(a) = 0$
$\text{Ein-Grad}_G(b) = 2$
$\text{Aus-Grad}_G(a) = 1$
$\text{Aus-Grad}_G(b) = 1$

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

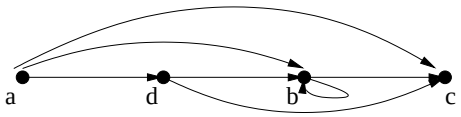
- **abstrakt**, durch Angabe der Knotenmenge V und der Kantenmenge E.
  Beispiel: $G_1 = (V_1, E_1)$ mit

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

- **graphisch** (bzw. **anschaulich**): Der obige Beispiel-Graph G_1 kann graphisch dargestellt werden durch



oder, äquivalent dazu, durch

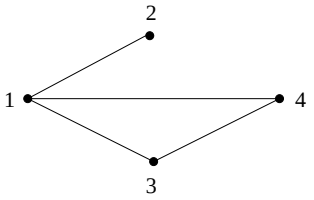


Adjazenzliste

- durch 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

$G_2 \quad := \quad$ 

durch die Adjazenzliste

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

Adjazenzmatrix

- durch Angabe einer **Adjazenzmatrix**, d.h. einer Tabelle, deren Zeilen und Spalten mit Knoten beschriftet sind, und die in der mit Knoten i beschrifteten Zeile und der mit Knoten j beschrifteten Spalte
  - den Eintrag 1 hat, falls es eine Kante von Knoten i nach Knoten j gibt, und
  - den Eintrag 0 hat, falls es keine Kante von i nach j gibt.

  Beispielsweise sieht die Adjazenzmatrix des gerichteten Graphen G_1 wie folgt aus:

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

Die Adjazenzmatrix des ungerichteten Graphen G_2 ist:

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

4.1.2 Wege in Graphen

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

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

$$(v_0, \ldots, v_\ell) \in V^{\ell+1},$$

für ein $\ell \in \mathbb{N}$, so dass für alle $i \in \mathbb{N}$ mit $0 \leqslant i < \ell$ gilt:

- falls G ein gerichteter Graph ist, so ist $(v_i, v_{i+1}) \in E$,
- falls G ein ungerichteter Graph ist, so ist $\{v_i, v_{i+1}\} \in E$.

Das Tupel $(v_0, \ldots, v_\ell)$ wird dann **ein Weg von** v_0 **nach** v_ℓ genannt. ℓ ist die **Länge des Weges**. D.h.: Die **Länge** des Weges gibt gerade an, wie viele **Kanten** auf dem Weg durchlaufen werden. Weglänge

Beachte:
Gemäß dieser Definition ist für jedes $v \in V$ das Tupel (v) ein Weg der Länge 0 von v nach v.

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

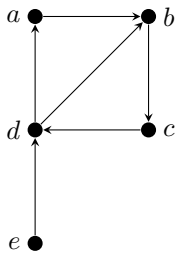
(c) Ein Weg $(v_0, \ldots, v_\ell)$ heißt **Kreis**, wenn $\ell \geqslant 1$ und $v_\ell = v_0$ ist. Kreis

(d) Ein Kreis $(v_0, \ldots, v_\ell)$ heißt **einfach**, wenn keine Kante mehrfach durchlaufen wird und — abgesehen vom Start- und Endknoten — kein Knoten mehrfach besucht wird. D.h.: einfacher Kreis

- In einem **gerichteten** Graphen G sind **einfache** Kreise genau die Wege der Form $(v_0, \ldots, v_\ell)$, für die gilt: $\ell \geqslant 1$ und $v_\ell = v_0$ und $|\{v_0, \ldots, v_{\ell-1}\}| = \ell$.
- In einem **ungerichteten** Graphen G sind **einfache** Kreise genau die Wege der Form $(v_0, \ldots, v_\ell)$, für die gilt: $\ell \geqslant 3$ und $v_\ell = v_0$ und $|\{v_0, \ldots, v_{\ell-1}\}| = \ell$.

Beispiel 4.13.

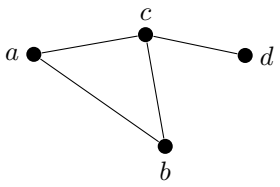
(a) Für den Graphen



gilt:

- (e, d, b, c, d) ist ein Weg der Länge 4, aber kein einfacher Weg.
- (d, b, c, d) ist ein einfacher Kreis.
- (e, d, a, b) ist ein einfacher Weg.
- (b, d, a) ist kein Weg.
- (a, b, c, d, b, c, d, a) ist ein Kreis, aber kein einfacher Kreis.

(b) Für den Graphen



gilt:

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

Definition 4.14 (azyklischer Graph, DAG).

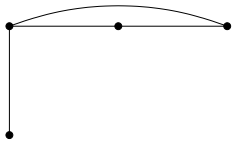
azyklisch (a) Ein Graph heißt **azyklisch**, falls er keinen einfachen Kreis enthält.

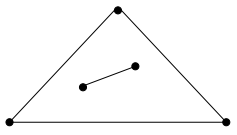
DAG (b) Gerichtete azyklische Graphen werden im Englischen **directed acyclic graph**, kurz: DAG, genannt.

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.

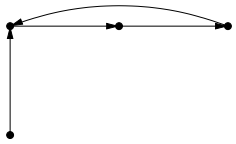
Beispiel:

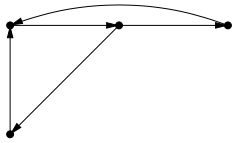
Der Graph  ist zusammenhängend.

Der Graph  ist nicht zusammenhängend.

stark 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.

Beispiel:

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

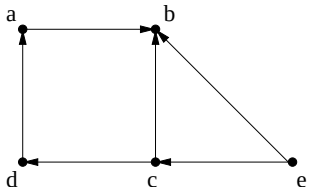
Der Graph  ist stark zusammenhängend.

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

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

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

Beispiel: Der Graph G

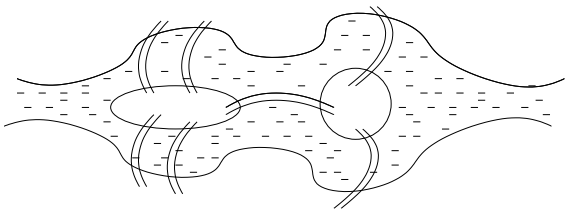


hat einen Hamilton-Weg, nämlich (e, c, d, a, b), aber keinen Hamilton-Kreis (da Aus-Grad$_G(b) = 0$ ist).

Ein Anwendungsbeispiel:
Beim Problem des **Handlungsreisenden** (engl.: **Travelling Salesman Problem**, kurz: **TSP**) 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 schwieriges Problem: Man kann zeigen, dass es (genau wie das auf Seite 74 betrachtete aussagenlogische Erfüllbarkeitsproblem) NP-vollständig ist. Handlungsreisendenproblem

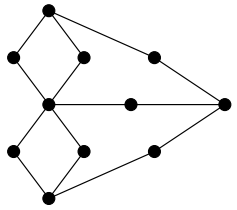
Im Gegensatz zu Hamilton-Wegen (bei denen es darum geht, einen Weg zu finden, der jeden **Knoten** des Graphen genau einmal besucht), geht es bei den im Folgenden betrachteten **Euler-Wegen** darum, einen Weg zu finden, der jede **Kante** des Graphen genau einmal besucht.

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 auf die in der folgenden Skizze dargestellten Art miteinander verbanden.



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

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

Satz 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 $K = (v_0, \ldots, v_\ell)$ ein Euler-Kreis. Insbesondere gilt: $v_0 = v_\ell$.

Schritt 1: Jeder Knoten $v \in \{v_0, \ldots, v_{\ell-1}\}$ hat geraden Grad, denn:
Sei $v \in \{v_0, \ldots, v_{\ell-1}\}$ beliebig. Zu jedem $i \in \{0, \ldots, \ell-1\}$ mit $v = v_i$ gibt es im Euler-Kreis K zwei verschiedene Kanten, nämlich

- $\{v_{i-1}, v_i\}$ und $\{v_i, v_{i+1}\}$, falls $i \neq 0$, bzw.
- $\{v_0, v_1\}$ und $\{v_{\ell-1}, v_0\}$, falls $i = 0$ (beachte: $v_0 = v_\ell$).

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

Schritt 2: $\{v_0, \ldots, v_{\ell-1}\} = V$, denn:
Laut Voraussetzung ist G zusammenhängend. Für beliebige Knoten $v, w \in V$ gilt daher: es gibt in G einen Weg von v nach w. Da K ein Euler-Kreis ist, enthält K sämtliche Kanten, die auf dem Weg von v nach w vorkommen. Insbesondere gilt also f.a. $v, w \in V$, dass $v, w \in \{v_0, \ldots, v_{\ell-1}\}$.

Zusammenfassung: Aus den Schritten 1 und 2 folgt direkt, dass jeder Knoten von G geraden Grad hat.

"$\Longleftarrow$": Sei G ein zusammenhängender ungerichteter Graph, in dem jeder Knoten geraden Grad hat. Es sei

$$W \;=\; (v_0, \ldots, v_\ell)$$

ein Weg **maximaler Länge** in G, der **keine Kante(n) mehrfach** enthält. Da wir W nicht mehr verlängern können, liegen alle mit v_ℓ inzidenten Kanten auf W. Da laut unserer Voraussetzung die Anzahl dieser Kanten gerade ist, folgt $v_\ell = v_0$.

Zu zeigen: W ist ein Euler-Kreis.

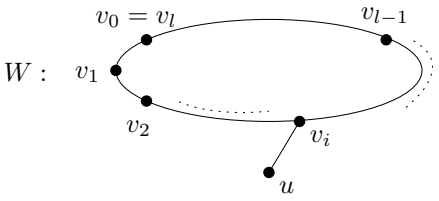
Angenommen, W ist **kein** Euler-Kreis.

Dann gibt es in G eine Kante e', die nicht auf W liegt. Da G zusammenhängend ist, gibt es einen Weg, der von einem Endknoten von e' zu einem zu W gehörenden Knoten führt. Sei e die erste Kante auf diesem Weg, die einen Endpunkt in W hat. Sei v_i der zu e inzidente Knoten aus W und sei $u \in V$ der andere zu e inzidente Knoten, d.h. $e = \{u, v_i\}$. Dann ist der Weg

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

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

Skizze:



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

(b) Die Richtung "$\Longrightarrow$" folgt analog zu (a): Sei $K = (v_0, \ldots, v_\ell)$ ein Euler-Weg, der kein Euler-Kreis ist. Man sieht leicht, dass die beiden Endknoten von K ungeraden Grad haben, und dass alle anderen Knoten geraden Grad haben.

Zum Beweis der Richtung "$\Longleftarrow$" kann man (a) verwenden: Seien x und y die beiden Knoten von G, die ungeraden Grad haben. Wir betrachten den Graphen $G' := (V', E')$ mit $V' := V \cup \{z\}$ und $E' := E \cup \{\{x, z\},\ \{y, z\}\}$, wobei z ein "neuer" Knoten ist, der nicht zu V gehört.

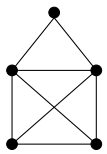
Offensichtlich hat jeder Knoten in G' geraden Grad. Außerdem ist G' zusammenhängend (da G zusammenhängend ist). Aus (a) folgt, dass G' einen Euler-Kreis besitzt. Wegen $\text{Grad}_{G'}(z) = 2$ wird z auf diesem Kreis genau einmal besucht. Durch Entfernen der Kanten $\{x, z\}$ und $\{z, y\}$ erhält man einen Euler-Weg in G, der die beiden Knoten x und y als Anfangs- und Endpunkt hat. □

Beispiel 4.20 (Lösung des Königsberger Brückenproblems).
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.

Beweis: 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.
Unter Verwendung von Satz 4.19 kann man auch die folgende Frage leicht lösen.

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.

Definition 4.22 (Teilgraph).
Seien $G = (V, E)$ und $G' = (V', E')$ zwei (gerichtete oder ungerichtete) Graphen.

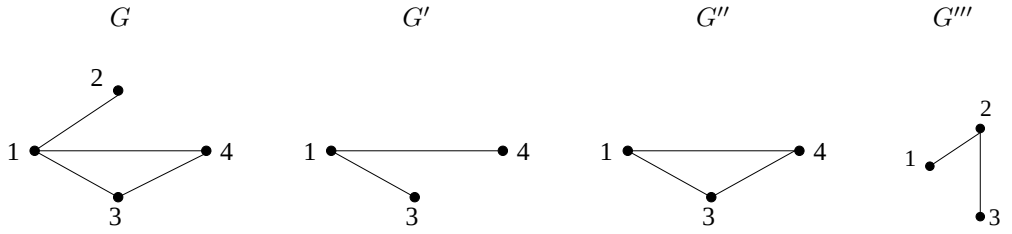
Teilgraph

(a) G' heißt **Teilgraph von** G, falls $V' \subseteq V$ und $E' \subseteq E$.

(b) Sei $W \subseteq V$. Der von W induzierte Teilgraph von G ist der Graph $G_{|W}$ mit Knotenmenge W und Kantenmenge $E_{|W} := \{e \in E : \text{alle mit } e \text{ inzidenten Knoten liegen in } W\}$.

(c) $G' = (V', E')$ heißt **induzierter Teilgraph von** G, falls $V' \subseteq V$ und $G' = G_{|V'}$. induzierter Teilgraph

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.

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

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

Beispielsweise sind die beiden Graphen

nicht gleich, da sie unterschiedliche Knotenmengen besitzen. Intuitiv sind die beiden Graphen aber *"prinzipiell gleich"* (Fachbegriff: **isomorph**, kurz: $G \cong G'$), da der zweite Graph aus dem ersten durch Umbenennung der Knoten entsteht. Der Begriff der Isomorphie wird durch die folgende Definition präzisiert:

Definition 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:

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

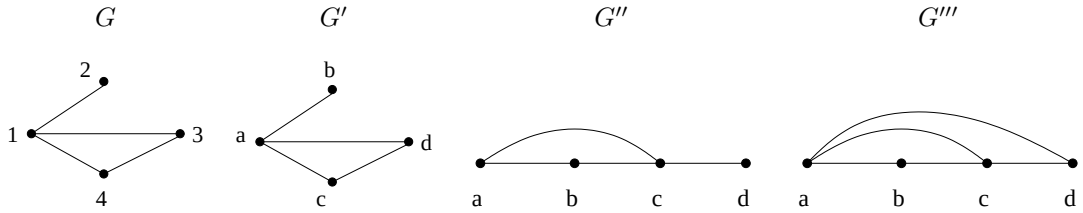
- falls G und G' ungerichtet sind:

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

Isomorphismus

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

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:

Knotenmarkierung

(a) Eine **Knotenmarkierung** eines (gerichteten oder ungerichteten) Graphen $G = (V, E)$ ist eine Abbildung

$$m\colon V \to W,$$

wobei W 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.

Kantenmarkierung

(b) Eine **Kantenmarkierung** von G ist eine Abbildung

$$m\colon E \to W,$$

wobei W 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.

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: Sei $G = (V, E)$ der Graph mit $V = \{a, b, c\}$ und $E = \{\{a, b\}, \{b, c\}, \{c, a\}\}$. Sei $m : E \to \mathbb{N}$ mit $m(\{a, b\}) = 1$, $m(\{b, c\}) = 1$ und $m(\{c, a\}) = 2$. Dann ist (G, m) ein Multigraph, der graphisch wie folgt dargestellt werden kann:

4.1.5 Zuordnungsprobleme

Wir betrachten zunächst zwei typische Beispiele von Zuordnungsproblemen.

Beispiel 4.29. (a) In einem Tennisverein sollen die Vereinsmitglieder für ein Turnier zu Doppelpaarungen zusammengestellt werden. Dabei möchte man jeweils nur befreundete Personen als "Doppel" zusammen spielen lassen.

Um diese Aufgabe zu lösen, modellieren wir die Situation durch den ungerichteten Graphen $G_T := (V_T, E_T)$ mit

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

Das **Ziel** ist, eine größtmögliche Anzahl von Doppelpaarungen zu finden. D.h., wir wollen eine möglichst große Menge $E' \subseteq E_T$ finden, so dass kein Vereinsmitglied Endpunkt von mehr als einer Kante aus E' ist.

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

Auch hier modellieren wir die Situation durch einen ungerichteten Graphen $G_F := (V_F, E_F)$ mit

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

Das **Ziel** ist, einen Flugplan aufzustellen, so dass jeder Pilot das ihm zugeteilte Flugzeug fliegen kann. D.h.: Wir wollen eine möglichst große Menge $E' \subseteq E_F$ finden, 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 **Matching** (bzw. **Paarung** oder **Menge unabhängiger Kanten**) genannt:

Definition 4.30.

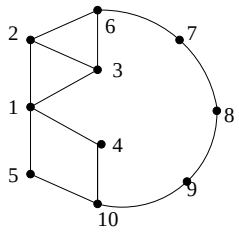
Matching
Paarung

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

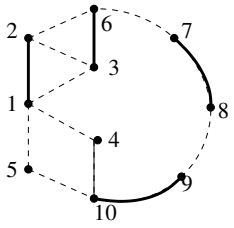
Ziel in Beispiel 4.29 (a) und (b) ist es, ein Matching maximaler Größe zu finden, d.h. ein Matching, das so viele Kanten wie möglich enthält.

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

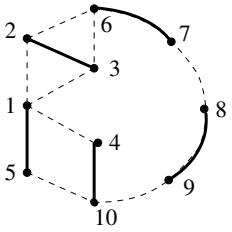
G_T =



sind z.B. die folgenden beiden Kantenmengen Matchings:



und



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

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:

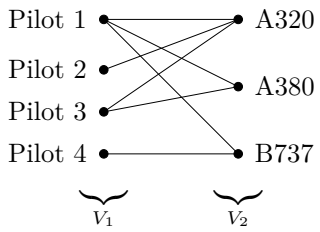
Definition 4.32 (bipartiter Graph).

bipartit

Ein ungerichteter Graph $G = (V, E)$ heißt **bipartit**, wenn seine Knotenmenge V so in zwei disjunkte Teilmengen V_1 und V_2 zerlegt werden kann (d.h. $V = V_1 \dot\cup V_2$), dass jede Kante aus E einen Endknoten in V_1 und einen Endknoten in V_2 hat.

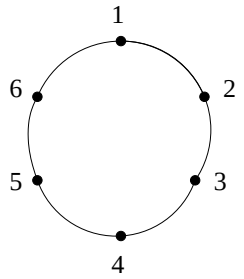
Beispiel 4.33.

(a) Der Graph

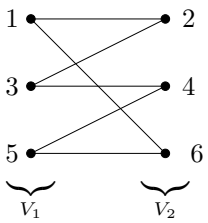


ist bipartit mit $V_1 = \{\text{Pilot 1}, \text{Pilot 2}, \text{Pilot 3}, \text{Pilot 4}\}$ und $V_2 = \{\text{A320}, \text{A380}, \text{B737}\}$.

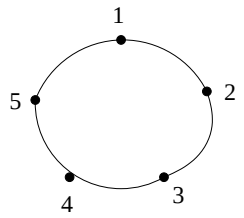
(b) Der Graph



ist bipartit mit $V_1 = \{1, 3, 5\}$ und $V_2 = \{2, 4, 6\}$. Der Graph lässt sich auch wie folgt graphisch darstellen:



(c) Der Graph



ist **nicht** bipartit.

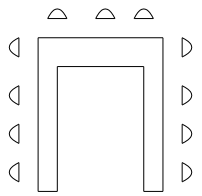
Beweis: Durch Widerspruch. Angenommen, er ist doch bipartit. Dann seien V_1 und V_2 die beiden disjunkten Teilmengen der Knotenmenge, so dass jede Kante des Graphen einen Endknoten in V_1 und einen Endknoten in V_2 hat. Wir können ohne Beschränkung der Allgemeinheit annehmen, dass $1 \in V_1$ ist (falls nicht, vertauschen wir einfach V_1 und V_2). Dann muss aber gelten: $2 \in V_2$, $3 \in V_1$, $4 \in V_2$ und $5 \in V_1$, also $V_1 = \{1, 3, 5\}$ und $V_2 = \{2, 4\}$. Im Graphen gibt es aber auch eine Kante zwischen 1 und 5, und beide Knoten gehören zu V_1. Dies ist ein Widerspruch zu der Annahme, dass jede Kante einen Endpunkt in V_1 und einen Endpunkt in V_2 hat. □

Allgemein 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:

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

Wir betrachten ein weiteres typisches Beispiel für ein Zuordnungsproblem:

Beispiel 4.34 (Sitzordnung bei einer Familienfeier).
Die Gäste einer Familienfeier sollen so an einer hufeisenförmigen Tafel



platziert werden, dass niemand neben jemanden sitzt, den er nicht leiden kann.

Lösungsansatz:

Schritt 1: Stelle den **Konfliktgraphen** $G = (V, E)$ auf, wobei

$$\begin{aligned} V &:= \{x : \text{Person } x \text{ soll zur Feier kommen}\} \text{ und} \\ E &:= \left\{ \{x, y\} : \begin{array}{l} \text{Person } x \text{ kann Person } y \text{ nicht leiden oder} \\ \text{Person } y \text{ kann Person } x \text{ nicht leiden} \end{array} \right\} \end{aligned}$$

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

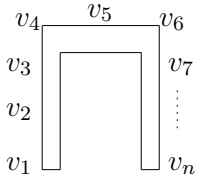
Schritt 2: Bilde das Komplement des Konfliktgraphen, d.h. betrachte den Graphen $\tilde{G} = (\tilde{V}, \tilde{E})$ mit

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

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 ist, die Gäste an **mehrere** Tische zu verteilen. Dies kann wie folgt modelliert werden:

Beispiel 4.35 (Sitzordnung bei einer Familienfeier, Teil 2).
Die Gäste einer Familienfeier sollen so an mehreren Tischen platziert werden, dass Personen, die sich nicht leiden können, an verschiedenen Tischen sitzen. Dabei sollen so wenig Tische wie möglich verwendet werden.

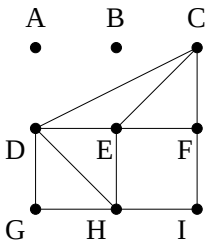
Diese Aufgabe kann folgendermaßen modelliert werden: Die verfügbaren Tische werden mit den Zahlen $1, 2, 3, \ldots$ durchnummeriert. Die geladenen Gäste und die herrschenden Konflikte zwischen Gästen werden durch den in Beispiel 4.34 betrachteten Konfliktgraphen $G = (V, E)$ repräsentiert. Die Zuordnung, wer an welchem Tisch sitzen soll, wird durch eine Knotenmarkierung $m\colon V \to \mathbb{N}_{>0}$ repräsentiert, wobei $m(x) = i$ bedeutet, dass Person x am Tisch i sitzen soll.

Das **Ziel** ist, eine **konfliktfreie Knotenmarkierung** $m\colon V \to \mathbb{N}_{>0}$ zu finden. Dabei soll $|\text{Bild}(m)|$ möglichst klein sein — dies entspricht dem Ziel, die Gäste an möglichst wenige Tische zu verteilen.

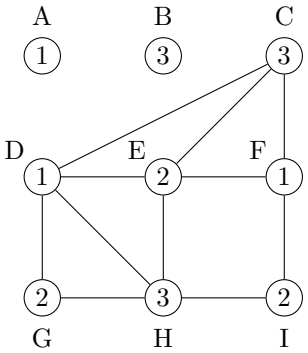
Definition 4.36 (konfliktfreie Knotenmarkierung).
Sei $G = (V, E)$ ein ungerichteter Graph. Eine Funktion $m\colon V \to \mathbb{N}$ heißt **konfliktfreie Knotenmarkierung** (bzw. konfliktfreie **Färbung**), wenn für jede Kante $\{x, y\} \in E$ gilt: $m(x) \neq m(y)$.

konfliktfreie Knotenmarkierung

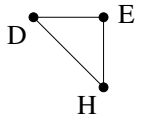
Beispiel 4.37. Um Beispiel 4.35 fortzuführen, betrachten wir eine Familienfeier mit Gästen A, B, C, D, E, F, G, H, I und folgendem Konfliktgraphen:



Die folgende Graphik gibt eine konfliktfreie Knotenmarkierung $m\colon V \to \mathbb{N}$ an, wobei für jeden Knoten $v \in V$ der Wert $m(v)$ in den Kreis geschrieben ist, der den Knoten v repräsentiert.



Für die hier gegebene Markierung m gilt $|\text{Bild}(m)| = 3$, die Gäste werden also an 3 Tische verteilt. Dies ist optimal, da der Konfliktgraph ein Dreieck, z.B.



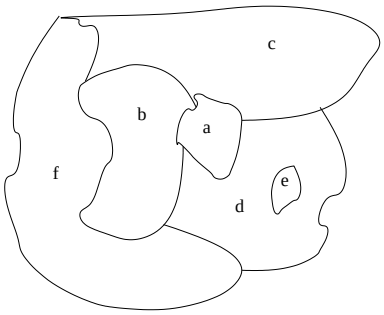
als Teilgraph enthält – deshalb muss für jede konfliktfreie Knotenmarkierung m' gelten: $|\text{Bild}(m')| \geqslant 3$.

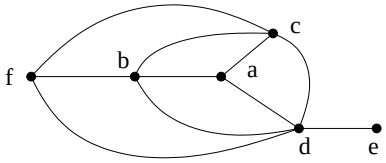
Bemerkung 4.38 (4-Farben-Problem).
Ein sehr bekannter Vertreter dieser Art von Markierungs- oder Färbungsaufgaben ist das so genannte **4-Farben-Problem**. Dabei handelt es sich um die Frage, wie viele verschiedene Farben nötig sind, um jede Landkarte so einzufärben, dass zwei Staaten, die ein Stück gemeinsamer Grenze haben, durch unterschiedliche Farben dargestellt werden. 1976 wurde bewiesen, dass vier Farben ausreichen. Der Beweis basiert auf einer Fallunterscheidung mit mehr als 1000 Fällen, die mit Hilfe eines Computerprogramms analysiert wurden.

4-Farben-Problem

Das Problem, eine Landkarte einzufärben, kann durch einen ungerichteten Graphen modelliert werden, dessen Knoten gerade die Staaten repräsentieren, und bei dem es eine Kante zwischen zwei Staaten gibt, falls diese eine gemeinsame Grenze besitzen. Ziel ist, eine konfliktfreie Knotenmarkierung m zu finden, bei der $|\text{Bild}(m)|$ so klein wie möglich ist.

Beispiel:
Wir betrachten eine kleine Landkarte und den zugehörigen Konfliktgraphen:





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

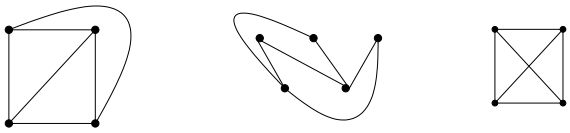
Da bei den vier Knoten a, b, c, d paarweise jeder zu jedem benachbart ist, muss eine konfliktfreie Färbung diesen vier Knoten vier verschiedene Farben zuordnen — für a, b, c, d etwa *rot*, *gelb*, *grün*, *blau*. Da f außerdem mit b, c, d benachbart ist, muss f dann wieder *rot* gefärbt sein; e kann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

Die aus Landkarten entstehenden Konfliktgraphen haben eine besondere Eigenschaft: Sie sind **planar**.

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

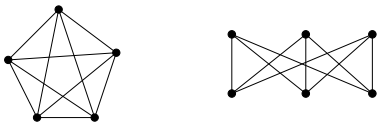
planar

Beispiele für planare Graphen sind:



(Der dritte Graph ist planar, da er wie der erste Graph kreuzungsfrei in die Ebene gezeichnet werden kann.)

Beispiele für nicht-planare Graphen sind:



Bemerkung 4.40 (chromatische Zahl).
Die Anzahl verschiedener "Farben" bzw. "Markierungen", die nötig sind, um einen Graphen $G = (V, E)$ konfliktfrei zu färben (bzw. zu markieren), nennt man die **chromatische Zahl**, kurz: $\chi(G)$. Die präzise Definition ist[2]

chromatische Zahl $\chi(G)$

$$\chi(G) \;:=\; \min\big\{\, |\text{Bild}(m)| \;:\; m\colon V \to \mathbb{N} \text{ ist eine konfliktfreie Knotenmarkierung für } G \,\big\}.$$

Weitere Beispiele für Zuordnungsprobleme, die durch Graphen modelliert werden können, finden sich in den Aufgaben 4.8, 4.9 und 4.10.

4.2 Bäume

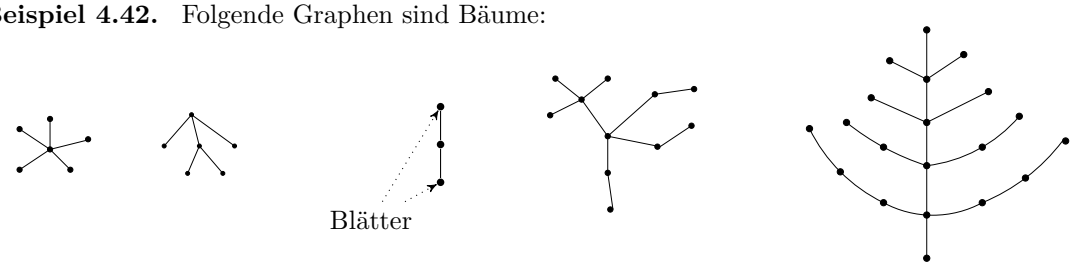
Eine für die Informatik besonders wichtige Art von Graphen sind die so ganannten **Bäume**. Wir betrachten im Folgenden zunächst ungerichtete Bäume und danach gerichtete Bäume.

4.2.1 Ungerichtete Bäume

Definition 4.41 (ungerichteter Baum).
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.

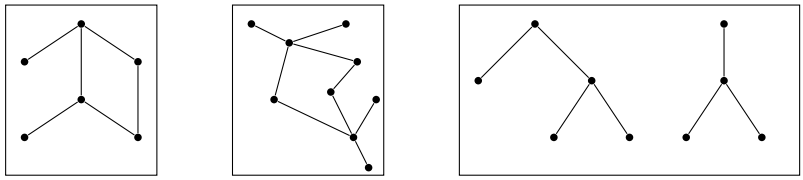
ungerichteter Baum
Blätter

Beispiel 4.42. Folgende Graphen sind Bäume:



Folgende Graphen sind keine Bäume:

[2] Ist M eine endliche, nicht-leere Menge von Zahlen, so bezeichnet $\min M$ das kleinste Element von M.

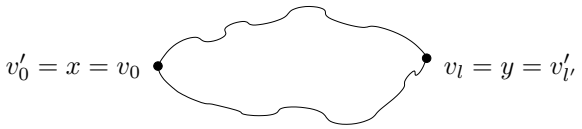


Beobachtung 4.43. Ist $B = (V, E)$ ein ungerichteter Baum, so gilt für alle Knoten $x, y \in V$:

Es gibt in B genau einen einfachen Weg von x nach y.

Denn: B ist ein ungerichteter Baum, d.h. B ist zusammenhängend und enthält keinen einfachen Kreis. Da B zusammenhängend ist, gibt es mindestens einen einfachen Weg von x nach y. Angenommen, $(v_0, \ldots, v_\ell)$ und $(v'_0, \ldots, v'_{\ell'})$ sind zwei verschiedene einfache Wege von x nach y. Insbesondere gilt dann $v_0 = x = v'_0$ und $v_\ell = y = v'_{\ell'}$.

Skizze:



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

Der folgende Satz besagt, dass die Anzahl der Kanten eines ungerichteten Baums durch die Anzahl der Knoten genau festgelegt ist.

Satz 4.44 (Anzahl der Kanten eines Baums).

Baum: $|E| = |V| - 1$

Für jeden ungerichteten Baum $B = (V, E)$, dessen Knotenmenge endlich und nicht-leer ist, 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 $E = \emptyset$. Für diesen Graphen gilt:

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

Induktionsschritt: $n \to n + 1$

Sei $n \in \mathbb{N}$ mit $n \geqslant 1$ beliebig.

Induktionsannahme: Für jeden ungerichteten Baum $B' = (V', E')$ mit $0 \neq |V'| \leqslant n$ 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 $|V| = n + 1 \geqslant 1 + 1 = 2$ 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 $V_x \cap V_y = \emptyset$. 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) $V = V_x \,\dot\cup\, V_y$, $V_x \neq \emptyset$ und $V_y \neq \emptyset$.

(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 $0 \neq |V_x| \leqslant n$ und $0 \neq |V_y| \leqslant n$. 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:

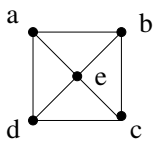
$$|E| \;\overset{(2)}{=}\; |E_x| + |E_y| + 1 \;\overset{(5)}{=}\; |V_x| - 1 + |V_y| - 1 + 1 \;\overset{(1)}{=}\; |V| - 1.$$

□

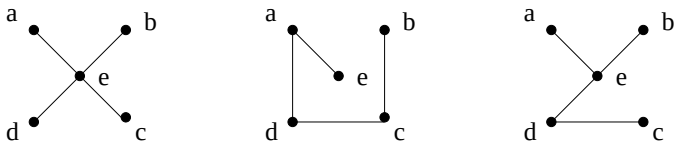
Bäume finden sich als Teilgraphen von zusammenhängenden Graphen. Besonders wichtig für die Informatik sind die so genannten **Spannbäume**.

Definition 4.45 (Spannbaum).
Sei $G = (V, E)$ ein ungerichteter Graph. Ein Graph $G' = (V', E')$ heißt **Spannbaum von** G, falls G' ein ungerichteter Baum mit $V' = V$ und $E' \subseteq E$ ist. Spannbaum

Beispiel 4.46. Der Graph



hat u.a. folgende Spannbäume:



Jeder zusammenhängende Baum besitzt einen Spannbaum. Präzise gilt:

Satz 4.47. Sei $G = (V, E)$ ein ungerichteter Graph, dessen Knotenmenge endlich ist. Dann gilt:

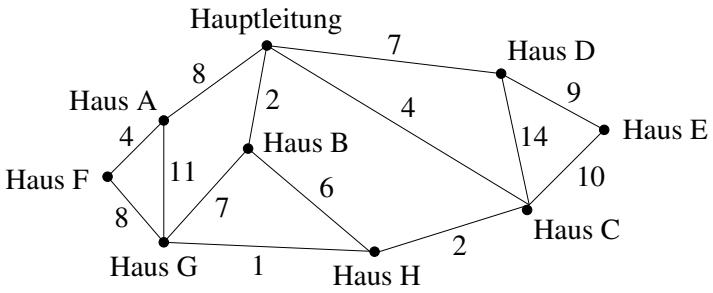
Es gibt (mindestens) einen Spannbaum von $G \iff G$ ist zusammenhängend.

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

Geht man von einem zusammenhängenden Graphen zu einem seiner Spannbäume über, so verkleinert man gemäß Satz 4.44 die Kantenmenge von $|E|$ auf $|V| - 1$ Kanten, ohne dabei den Zusammenhang des Graphen aufzugeben. Mit dem Begriff des Spannbaums wird also ein "bezüglich der Kantenzahl kostengünstigerer Zusammenhang" modelliert.

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. Dazu betrachten wir das folgende Beispiel.

Beispiel 4.48 (Kabelfernsehen). Eine Firma will Leitungen zum Empfang von Kabelfernsehen in einem neuen Wohngebiet verlegen. Der folgende Graph skizziert das Wohngebiet:



Knoten entsprechen dabei einzelnen Häusern bzw. der Hauptleitung, die aus einem bereits verkabelten Gebiet heranführt. Eine Kante zwischen zwei Knoten zeigt an, dass es prinzipiell möglich ist, eine direkte Leitung zwischen den beiden Häusern zu verlegen. Der Wert, mit dem die Kante markiert ist, beschreibt, wie teuer (in 1000 €) es ist, diese Leitung zu verlegen.

Ziel ist, Leitungen so zu verlegen, dass

(1) jedes Haus ans Kabelfernsehen angeschlossen ist und

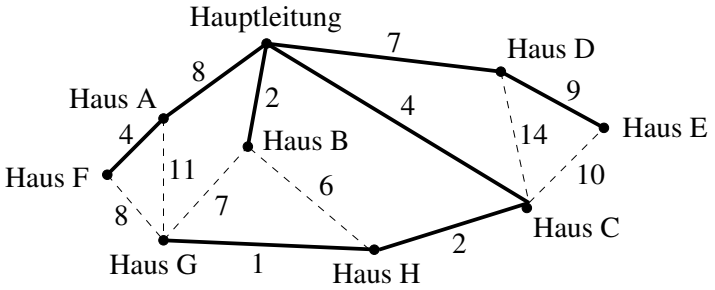
(2) die Kosten für das Verlegen der Leitungen so gering wie möglich sind.

minimaler Spannbaum

Es wird also ein Spannbaum gesucht, bei dem die Summe seiner Kantenmarkierungen so klein wie möglich ist. Ein solcher Spannbaum wird **minimaler Spannbaum** (engl.: **minimum spanning**

tree) genannt.

Die im Folgenden **fett** gezeichneten Kanten geben die Kanten eines minimalen Spannbaums an:



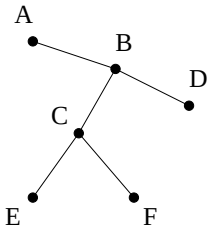
Verlegt die Firma genau diese Leitungen, so hat sie das neue Wohngebiet mit den geringstmöglichen Kosten ans Kabelfernsehen angeschlossen.

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

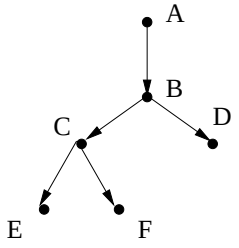
4.2.2 Gerichtete Bäume

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

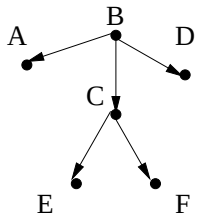
Beispiel 4.49. Ungerichteter Baum:



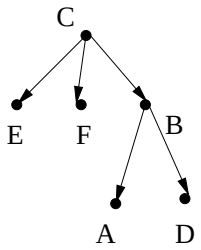
- Zugehöriger gerichteter Baum mit Wurzel A:

$G_A :=$ 

- Zugehöriger gerichteter Baum mit Wurzel B:

$G_B :=$ 

- Zugehöriger gerichteter Baum mit Wurzel C:

$G_C :=$ 

Die präzise Definition des Begriffs "gerichteter Baum" ist wie folgt:

Definition 4.50 (gerichteter Baum).

gerichteter Baum

Ein gerichteter Graph $G = (V, E)$ heißt **gerichteter Baum**, falls er folgende Eigenschaften hat:

(1) G besitzt genau einen Knoten $w \in V$ mit Ein-Grad$_G(w) = 0$.
Wurzel — 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$_G(v) \leqslant 1$.

Definition 4.51 (Blätter, innere Knoten, Höhe).

(a) Sei $B = (V, E)$ ein gerichteter Baum. Diejenigen Knoten, deren Aus-Grad 0 ist, heißen **Blätter**.
Blätter

Beispiel: In Beispiel 4.49 hat G_A die Blätter D, E, F. G_B hat die Blätter A, D, E, F und G_C die Blätter A, D, E, F.

(b) Diejenigen Knoten eines gerichteten Baums, die weder Wurzel noch Blätter sind, heißen **innere Knoten**.
innere Knoten

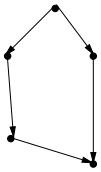
(c) Sei $B = (V, E)$ ein gerichteter Baum. Die **Höhe** (bzw. **Tiefe**, engl.: height, depth) von B ist die Länge eines längsten Weges in B.
Höhe
Tiefe

Beispiel: In Beispiel 4.49 hat G_A die Höhe 3, G_B die Höhe 2 und G_C die Höhe 2.

Beobachtung 4.52.

(a) Jeder gerichtete Baum ist ein gerichteter azyklischer Graph (kurz: DAG, vgl. Definition 4.14). Aber es gibt gerichtete azyklische 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 4.44, 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 4.53 (gerichtete Bäume, rekursive Definition).
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 $B := (V, \emptyset)$ ein gerichteter Baum.

Skizze: $B :=$ ●

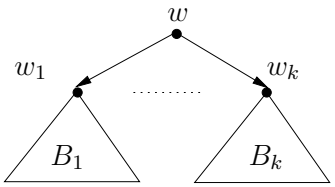
Der (eindeutig bestimmte) Knoten in V heißt **Wurzel** von B.
Die **Höhe** von B ist 0.

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. $V_i \cap V_j = \emptyset$ f.a. $i, j \in \{1, \ldots, k\}$ mit $i \neq j$), sind $w_1 \in V_1, \ldots, w_k \in V_k$ die Wurzeln von $B_1, \ldots, B_k$, und ist w ein Element, das nicht in $V_1 \cup \cdots \cup V_k$ liegt, dann ist der Graph $B = (V, E)$ mit

$$V := \{w\} \cup V_1 \cup \cdots \cup V_k \qquad \text{und} \qquad E := E_1 \cup \cdots \cup E_k \cup \{(w, w_i) : i \in \{1, \ldots, k\}\}$$

ein gerichteter Baum.

Skizze:

$B :=$ 

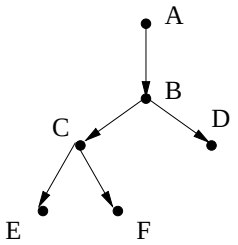
Der Knoten w heißt **Wurzel** von B.
Die **Höhe** von B ist $1 + \max\{h_1, \ldots, h_k\}$, wobei $h_1, \ldots, h_k \in \mathbb{N}$ die Höhen der gerichteten Bäume $B_1, \ldots, B_k$ sind.

Notation 4.54 (Kinder eines Knotens).
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 zwei Kinder hat. Mit solchen Bäumen kann man z.B. Binär-Codierung oder Kaskaden von JA-NEIN-Entscheidungen beschreiben.

Definition 4.55 (Binärbaum, vollständiger Binärbaum).

Binärbaum

(a) Ein gerichteter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt: Aus-Grad$_B(v) \leqslant 2$.

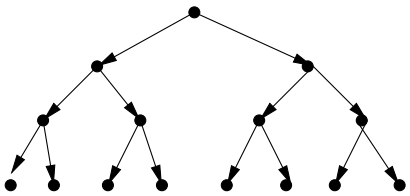
vollständiger Binärbaum

(b) Ein Binärbaum $B = (V, E)$ heißt **vollständiger Binärbaum**, falls gilt:

(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 4.56. 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:

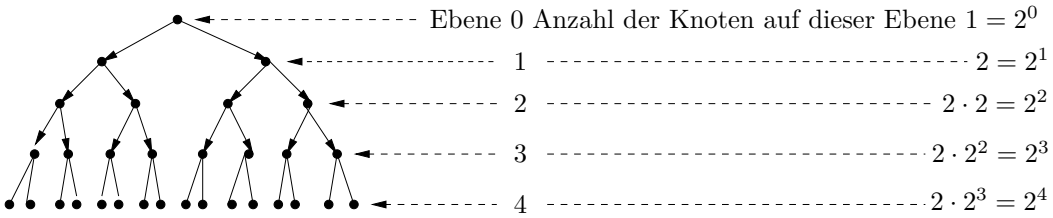
Satz 4.57. Sei $h \in \mathbb{N}$.

(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) *Skizze:*



Anhand dieser Skizze sieht man leicht, dass ein vollständiger Binärbaum der Höhe h genau 2^h Blätter und

$$2^0 + 2^1 + 2^2 + \cdots + 2^h \overset{\text{Satz 2.49}}{=} 2^{h+1} - 1$$

Knoten besitzt.

Den formalen Beweis führen wir 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.

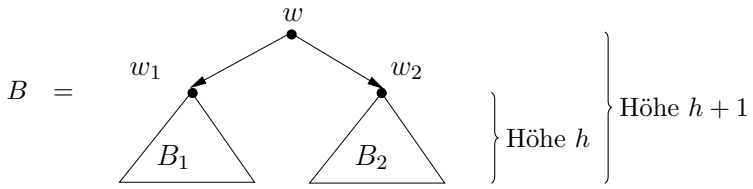
Induktionsschritt: $h \to h + 1$:
Sei $h \in \mathbb{N}$ beliebig.

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 $h + 1 \geqslant 1$ hat w genau 2 Kinder. Seien $w_1 \in V$ und $w_2 \in V$ diese beiden Kinder von w. Für $i \in \{1, 2\}$ sei V_i die Menge aller Knoten aus V, zu denen von w_i aus ein Weg führt; und sei $B_i := (V_i, E_i)$ der induzierte Teilgraph von B mit Knotenmenge V_i.

Skizze:



Offensichtlich ist sowohl B_1 als auch B_2 ein vollständiger Binärbaum der Höhe h. Gemäß Induktionsannahme hat jeder der beiden Bäume B_1 und B_2 genau 2^h Blätter und genau $2^{h+1} - 1$ Knoten.

Der Baum B hat daher genau $2^h + 2^h = 2^{h+1}$ Blätter und genau $1 + (2^{h+1} - 1) + (2^{h+1} - 1) = 2 \cdot 2^{h+1} - 1 = 2^{h+2} - 1$ Knoten.

(b) Analog. Details: Übung.

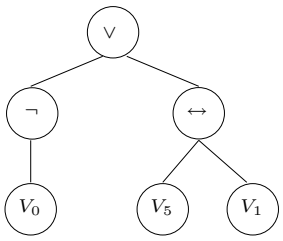
□

4.2.3 Modellierungsbeispiele

Gerichtete Bäume mit Knoten- oder Kantenmarkierungen können auf vielfältige Arten zur Modellierung genutzt werden.

Beispiel 4.58. In Kapitel 3 (Seite 68) haben wir Bäume bereits genutzt, um die Struktur einer aussagenlogischen Formel übersichtlich darzustellen. Der entsprechende Baum heißt **Syntaxbaum** der Formel.

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



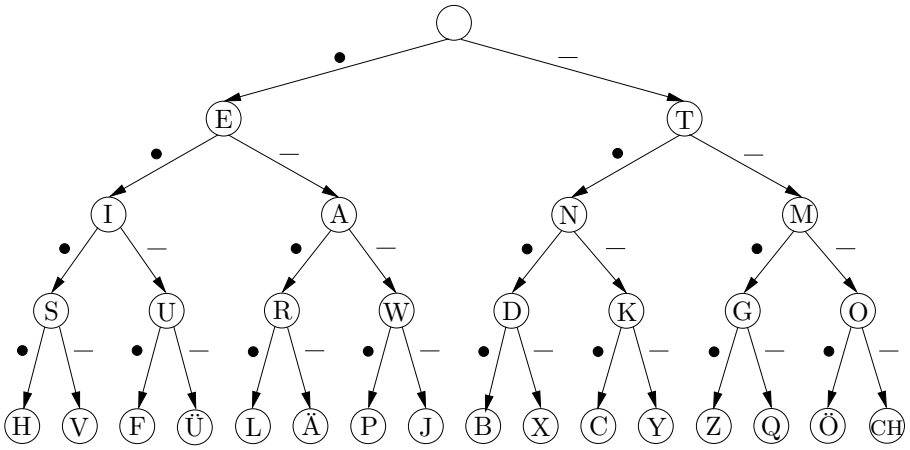
Auf ähnliche Art werden markierte Bäume genutzt, um die Struktur vieler anderer Objekte (an Stelle von aussagenlogischen Formeln) zu beschreiben — z.B. für arithmetische Terme, zur Darstellung von Klassen- und Objekthierarchien, zur Beschreibung der Struktur von Computerprogrammen oder umgangssprachlichen Texten oder auch zur Beschreibung der hierarchischen Organisationsstruktur einer Firma.

Entscheidungsbäume

Beispiel 4.59. Folgen von Entscheidungen können in vielen Zusammenhängen durch gerichtete markierte Bäume modelliert werden. Solche Bäume heißen **Entscheidungsbäume**. Durch einen solchen Entscheidungsbaum erhält man beispielsweise eine kompakte Darstellung des **Morse-Codes**.

Im Morse-Code wird jeder Buchstabe durch eine Folge von kurzen und langen Signalen repräsentiert. Ein "kurzes Signal" wird im folgenden Baum als Kantenmarkierung "•" dargestellt; ein "langes Signal" wird als "—" dargestellt. Insgesamt wird der Morsecode durch folgenden Entscheidungsbaum repräsentiert:

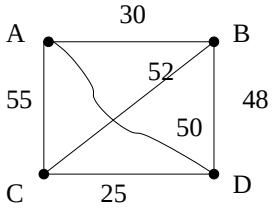
M =



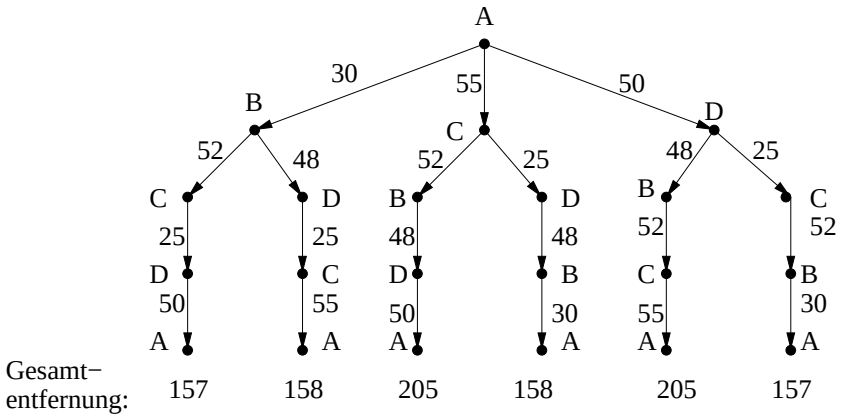
Eine eingehende Meldung aus kurzen und langen Signalen wird entschlüsselt, indem man an der Wurzel des Baums M beginnt und bei einem kurzen Signal nach links, bei einem langen nach rechts weitergeht. Eine längere Pause zeigt an, dass ein Buchstabe vollständig übermittelt ist.

In jedem Entscheidungsbaum modellieren die Knoten einen Zwischenstand bei der Entscheidungsfindung. Sie können entsprechend markiert sein, z.B. mit dem 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. Beim Morse-Code ist das jeweils ein kurzes oder ein langes Signal, das als Kantenmarkierung angegeben wird.

Beispiel 4.60. Markierte Bäume können auch genutzt werden, um den Lösungsraum kombinatorischer Probleme darzustellen. Als Beispiel betrachten wir einen Handlungsreisender, der einen möglichst kurzen Rundweg finden soll, auf dem er jede der Städte A, B, C, D besucht. Die Entfernungen (in km) zwischen den Städten sind als Kantenmarkierungen des folgenden Graphen gegeben:



Der folgende Baum repräsentiert alle möglichen in Stadt A startenden Rundwege:



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 4.61. 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. Ein Beispiel dazu findet sich in Aufgabe 4.12.

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

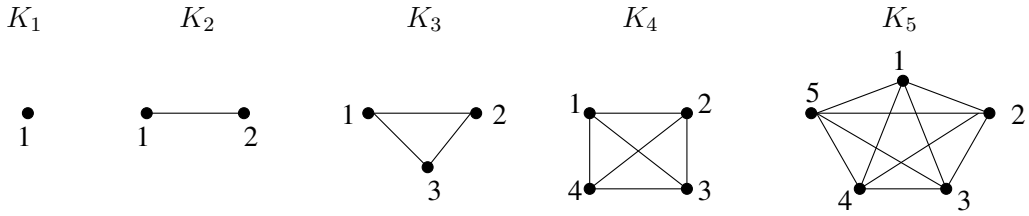
In diesem Abschnitt werden einige spezielle Arten von Graphen vorgestellt, die eine wichtige Rolle in der Informatik spielen.

4.3.1 Spezielle ungerichtete Graphen

Definition 4.62 (Der vollständige Graph K_n).
Sei $n \in \mathbb{N}_{>0}$. Der **vollständige ungerichtete Graph** K_n hat Knotenmenge $\{1, \ldots, n\}$ und Kantenmenge $\big\{\{i, j\} : i, j \in \{1, \ldots, n\},\, i \neq j\big\}$.

K_n
vollständiger ungerichteter Graph

Beispiele:



Beobachtung 4.63. Der Graph K_n hat n Knoten und $\dfrac{n \cdot (n-1)}{2}$ Kanten.

Definition 4.64 (Der vollständige bipartite Graph $K_{m,n}$).
Seien $m, n \in \mathbb{N}_{>0}$. Der **vollständige ungerichtete bipartite Graph** $K_{m,n}$ hat Knotenmenge

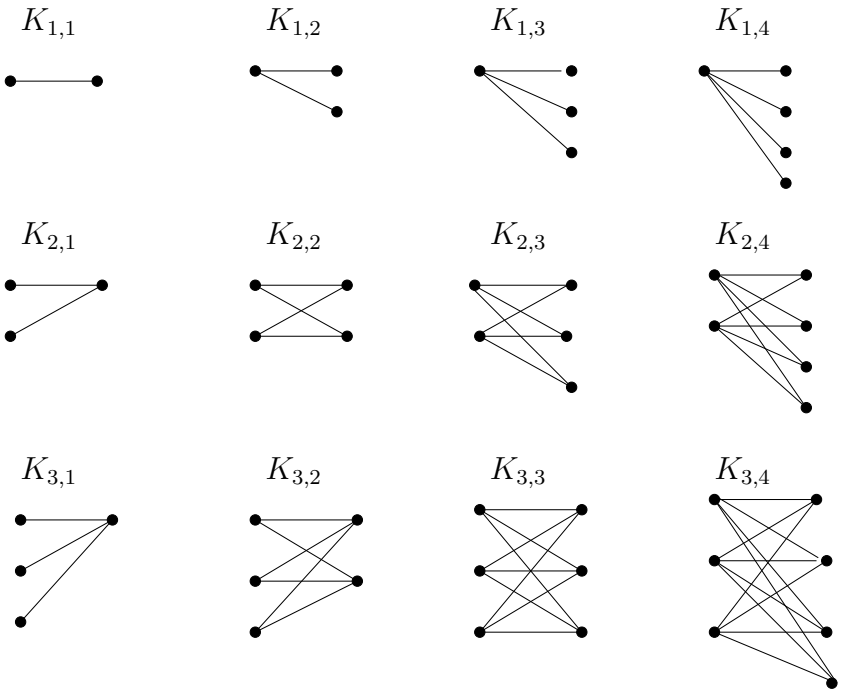
$K_{m,n}$
vollständiger ungerichteter bipartiter Graph

$$\big\{\, (1,i) : i \in \{1,\dots,m\} \,\big\} \;\cup\; \big\{\, (2,j) : j \in \{1,\dots,n\} \,\big\}$$

und Kantenmenge

$$\big\{\, \{\, (1,i),\, (2,j) \,\} \;:\; i \in \{1,\dots,m\},\; j \in \{1,\dots,n\} \,\big\}.$$

Beispiele:



Beobachtung 4.65. Der Graph $K_{m,n}$ hat $m + n$ Knoten und $m \cdot n$ Kanten.

Notation 4.66. 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 2.26(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 V als gerichteten Graph mit Knotenmenge V und Kantenmenge R auffassen. Gerichtete Graphen mit Knotenmenge V sind also dasselbe wie 2-stellige Relationen über einer Menge V.

Von besonderem Interesse sind 2-stellige Relationen, die eine oder mehrere der folgenden Eigenschaften besitzen:

Definition 4.67.
Sei E eine 2-stellige Relation über einer Menge V (d.h. $G = (V, E)$ ist ein gerichteter Graph).

reflexiv

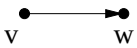
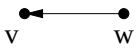
(a) E heißt **reflexiv**, falls für alle $v \in V$ gilt:

$$(v, v) \in E. \qquad \text{(Skizze: } v \text{)}$$

symmetrisch

(b) E heißt **symmetrisch**, falls f.a. $v, w \in V$ gilt:

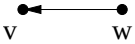
$$\text{Wenn } (v, w) \in E, \text{ dann auch } (w, v) \in E.$$

(d.h. zu jeder Kante  gibt es auch eine "Rückwärtskante" )

antisymmetrisch

(c) E heißt **antisymmetrisch**, falls f.a. $v, w \in V$ gilt:

$$\text{Wenn } (v, w) \in E \text{ und } (w, v) \in E, \text{ dann } v = w.$$

(d.h: Ist $v \neq w$, so gibt es in E allenfalls eine der beiden Kanten  und )

konnex

(d) E heißt **konnex**, falls f.a. $v, w \in V$ gilt:

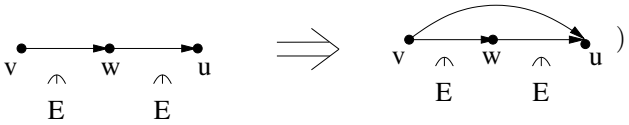
$$(v, w) \in E \text{ oder } (w, v) \in E.$$

(d.h. mindestens eine der beiden Kanten  und  liegt in E)

transitiv

(e) E heißt **transitiv**, falls f.a. $v, w, u \in V$ gilt:

$$\text{Ist } (v, w) \in E \text{ und } (w, u) \in E, \text{ so auch } (v, u) \in E.$$

(Skizze: )

Äquivalenzrelationen

Definition 4.68 (Äquivalenzrelation).
Eine **Äquivalenzrelation** ist eine 2-stellige Relation, die **reflexiv**, **transitiv** und **symmetrisch** ist. Äquivalenzrelation

Beispiel 4.69. Beispiele für Äquivalenzrelationen:

(a) **Gleichheit:** Für jede Menge M ist

$$E := \{ (m, m) : m \in M \}$$

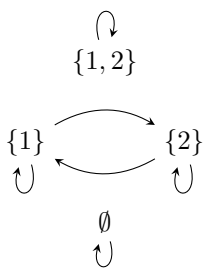
eine Äquivalenzrelation. Die Aussage "$(x, y) \in E$" entspricht gerade der Aussage "$x = y$".

(b) **Gleichmächtigkeit:** Für jede endliche Menge M ist

$$E := \{ (A, B) : A \subseteq M,\ B \subseteq M,\ |A| = |B| \}$$

eine Äquivalenzrelation über der Potenzmenge $\mathcal{P}(M)$.

Skizze für $M = \{1, 2\}$:



(c) **Logische Äquivalenz:** Die Relation

$$E := \{ (\varphi, \psi) : \varphi, \psi \in \mathrm{AL},\ \varphi \equiv \psi \}$$

ist eine Äquivalenzrelation über der Menge AL aller aussagenlogischen Formeln.

Bemerkung 4.70 (Äquivalenzklassen). Sei E eine Äquivalenzrelation über einer Menge V. Für jedes $v \in V$ bezeichnet

$$[v]_E := \{ v' \in V : (v, v') \in E \}$$

die **Äquivalenzklasse** von v bezüglich E. D.h.: Die Äquivalenzklasse $[v]_E$ besteht aus allen Elementen von V, die laut E "äquivalent" zu v sind. $[v]_E$ Äquivalenzklasse

Eine Menge $W \subseteq V$ heißt *Äquivalenzklasse* (bzgl. E), falls es ein $v \in V$ mit $W = [v]_E$ gibt. Das Element v wird dann ein *Vertreter* seiner Äquivalenzklasse W genannt.

Man sieht leicht, dass für alle $v, w \in V$ gilt: Entweder $[v]_E = [w]_E$ oder $[v]_E \cap [w]_E = \emptyset$. Falls V endlich und nicht leer ist, folgt daraus, dass es eine Zahl $k \in \mathbb{N}_{>0}$ und Äquivalenzklassen $W_1, \ldots, W_k$ geben muss, so dass $V = W_1 \dot\cup \cdots \dot\cup W_k$ ist. Die Zahl k wird auch **Index** von E genannt. D.h.: Der Index einer Äquivalenzrelation gibt an, wie viele verschiedene Äquivalenzklassen es gibt. Index

Beispielsweise hat die Gleichmächtigkeits-Relation aus Beispiel 4.69 (b) den Index $|M| + 1$.

Ordnungsrelationen

Definition 4.71 (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

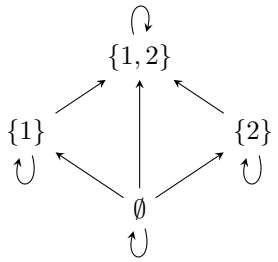
(c) E heißt **lineare Ordnung** oder **totale Ordnung**, falls E **reflexiv, transitiv, antisymmetrisch und konnex** ist.

Beispiel 4.72.

(a) $\leqslant$ ist eine **lineare Ordnung** auf $\mathbb{N}$ (und $\mathbb{Z}$, $\mathbb{Q}$ und $\mathbb{R}$). Ebenso ist $\geqslant$ eine lineare Ordnung auf $\mathbb{N}$ (und $\mathbb{Z}$, $\mathbb{Q}$ und $\mathbb{R}$).

(b) Für jede Menge M sind $\subseteq$ und $\supseteq$ **partielle Ordnungen** auf der Potenzmenge $\mathcal{P}(M)$ (aber keine linearen Ordnungen).

Skizze für "$\subseteq$" bei $M = \{1, 2\}$:



(c) Die Folgerungsrelation für aussagenlogische Formeln (siehe Definition 3.24)

$$E \;:=\; \{\, (\varphi, \psi) \,:\, \varphi, \psi \in \mathrm{AL},\, \varphi \models \psi \,\}$$

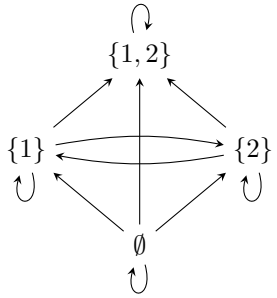
ist eine **Präordnung** auf AL (aber keine partielle Ordnung).

(d) Für jede endliche Menge M ist

$$E \;:=\; \{\, (A, B) \,:\, A, B \subseteq M,\, |A| \leqslant |B| \,\}$$

eine **Präordnung** auf $\mathcal{P}(M)$ (aber keiner partielle Ordnung).

Skizze für $M = \{1, 2\}$:



Die reflexive und transitive Hülle einer Relation

Definition 4.73 (reflexive und transitive Hülle).
Sei $G = (V, E)$ ein gerichteter Graph. Die **reflexive und transitive Hülle** (bzw. der **reflexive und transitive Abschluss** von E auf V ist die rekursiv wie folgt definierte Relation $E^* \subseteq V \times V$: reflexive und transitive Hülle

Basisregeln:

- F.a. $v \in V$ ist $(v, v) \in E^*$.
- F.a. $(v, w) \in E$ ist $(v, w) \in E^*$

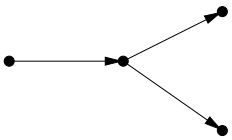
Rekursive Regel:

- Sind $(v, w) \in E^*$ und $(w, u) \in E^*$, so ist auch $(v, u) \in E^*$.

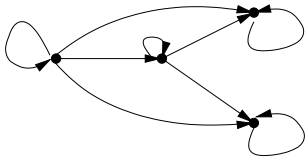
Das heißt: Der reflexive und transitive Abschluss von E auf V ist die kleinste Obermenge von E, die reflexiv und transitiv ist.

Beispiel:

$G = (V, E)$:=



$G^* = (V, E^*)$:



Beobachtung 4.74. 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 und transitive Hülle von E auf V ist.

(b) Es gibt in G einen Weg von v nach w.

Beweis: Übung. □

4.4 Literaturhinweise

Als vertiefende Lektüre sei Kapitel 5 in [15], Kapitel 11 in [22], Teile der Kapitel 0–4 und 8 in [5], sowie Teile der Kapitel 7–10 und 13 in [19] empfohlen. Eine umfassende Einführung in die Graphentheorie gibt das Lehrbuch [5].

Quellennachweis: Viele der in diesem Kapitel angegebenen Modellierungsbeispiele sowie die folgenden Aufgaben 4.1, 4.3, 4.4 und 4.12 sind dem Buch [15] entnommen.

4.5 Übungsaufgaben zu Kapitel 4

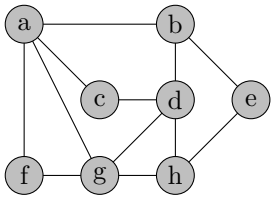
Aufgabe 4.1. Betrachten Sie den ungerichteten Graphen G auf der rechten Seite.

(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?

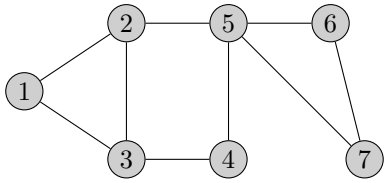
(c) Geben Sie einen Hamilton-Kreis in G an.

(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.

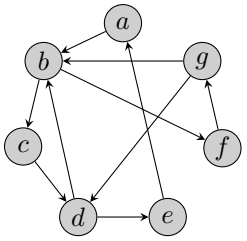


Aufgabe 4.2. Es seien die folgenden beiden Graphen G_1 und G_2 gegeben:

G_1:



G_2:



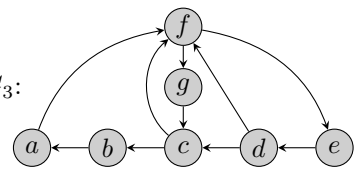
(a) Geben Sie für jeden der beiden Graphen G_1 und G_2 die Knotenmenge und die Kantenmenge an. Repräsentieren Sie außerdem jeden der beiden Graphen durch eine Adjazenzmatrix und eine Adjazenzliste.

(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.

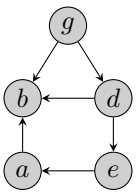
(c) Ist G_1 zusammenhängend? Ist G_2 stark zusammenhängend? Ist G_2 azyklisch?

(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.

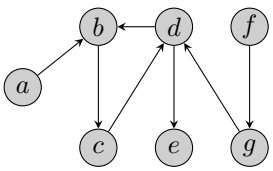
G_3:



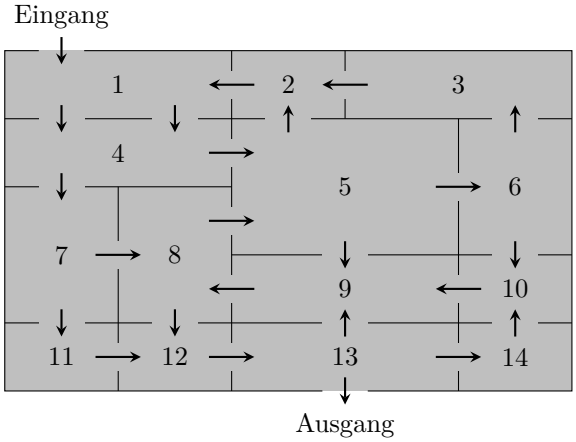
G_4:



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.

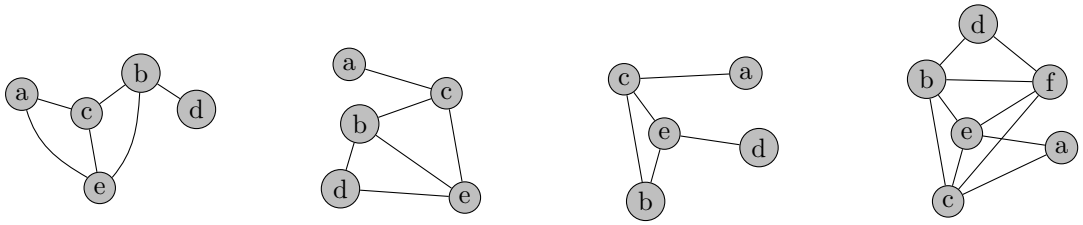
Dabei können auf einer Leitung Daten in beide Richtungen gesendet werden. Ein solches Netzwerk lässt sich leicht als ungerichteter Graph darstellen: ein Knoten repräsentiert einen Rechner, und eine Kante repräsentiert eine Leitung.

(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 $E_1 = \{\{1, i\} : 2 \leqslant i \leqslant n\}$
- $G_2 = (V_2, E_2)$ mit $V_2 = V_1$ und $E_2 = \{\{i, i+1\} : 1 \leqslant i < n\}$
- $G_3 = (V_3, E_3)$ mit $V_3 = V_1$ und $E_3 = E_2 \cup \{\{n, 1\}\}$

Aufgabe 4.5. 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

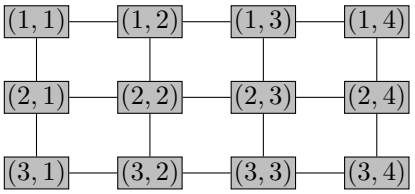
(b) Überprüfen Sie, welche der Graphen isomorph zueinander sind. Falls zwei Graphen G_i und G_j isomorph sind, so geben Sie einen Isomorphismus von G_i nach G_j an. Falls hingegen G_i und G_j nicht isomorph sind, so begründen Sie dies.

(c) Welche der Graphen kann man nachzeichnen, ohne den Stift abzusetzen oder eine Kante doppelt zu ziehen?

Aufgabe 4.6. Für $m, n \in \mathbb{N}_{>0}$ sei das $m \times n$-*Gitter* der Graph $G_{m\times n} = (V_{m\times n}, E_{m\times n})$ mit

$$\begin{aligned} V_{m\times n} &:= \big\{\, (i,j) \,:\, 1 \leqslant i \leqslant m,\, 1 \leqslant j \leqslant n \,\big\}, \\ E_{m\times n} &:= \big\{\, \{(i,j),(i,j+1)\} \,:\, 1 \leqslant i \leqslant m,\, 1 \leqslant j < n \,\big\} \cup \\ &\quad\ \big\{\, \{(i,j),(i+1,j)\} \,:\, 1 \leqslant i < m,\, 1 \leqslant j \leqslant n \,\big\}. \end{aligned}$$

Das 3×4-Gitter $G_{3\times 4}$ sieht z.B. wie folgt aus:



(a) Überprüfen Sie, ob $G_{3\times 4}$ bipartit ist. Falls $G_{3\times 4}$ bipartit ist, so geben Sie zwei disjunkte Knotenmengen $V_1, V_2 \subseteq V_{3\times 4}$ mit $V_1 \cup V_2 = V_{3\times 4}$ an, so dass jede Kante aus $E_{3\times 4}$ einen Knoten aus V_1 und einen Knoten aus V_2 miteinander verbindet. Falls $G_{3\times 4}$ nicht bipartit ist, so begründen Sie dies.

(b) Geben Sie ein Matching maximaler Größe in $G_{3\times 4}$ an.

(c) Geben Sie einen Hamilton-Kreis in $G_{3\times 4}$ an.

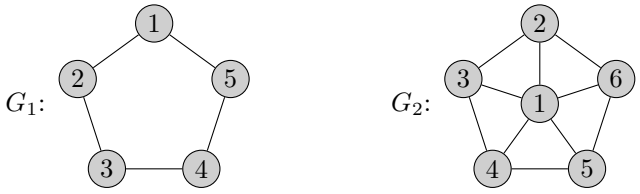
(d) Für welche $m, n \in \mathbb{N}_{>0}$ besitzt $G_{m\times n}$ einen Hamilton-Kreis, für welche nicht?

Hinweis: Stellen Sie sich vor, dass die Knoten des Gitters so mit den Farben rot oder blau eingefärbt sind, dass benachbarte Knoten unterschiedliche Farben besitzen. Jeder Weg durch das Gitter besucht daher immer abwechselnd einen blauen und einen roten Knoten.

Aufgabe 4.7. Zwei Personen A und B spielen ein Spiel auf einem zusammenhängenden ungerichteten Graphen $G = (V, E)$. Die Spieler wählen abwechselnd Knoten $v_1, v_2, v_3, \ldots$ aus V, so dass $v_1, v_2, v_3, \ldots$ verschiedene Knoten sind und jeweils gilt: $\{v_i, v_{i+1}\} \in E$. Den ersten Knoten wählt A. Der letzte Spieler, der einen Knoten wählen kann, gewinnt.

Ein Spieler hat eine *Gewinnstrategie* in dem Spiel genau dann, wenn der Spieler das Spiel, unabhängig davon wie der andere Spieler spielt, gewinnen kann.

(a) Geben Sie für jeden der beiden folgenden Graphen G_1 und G_2 ein Matching maximaler Größe an und entscheiden Sie, welcher der beiden Spieler in dem Spiel auf dem entsprechenden Graph eine Gewinnstrategie hat.



(b) Beweisen Sie, dass die beiden folgenden Aussagen äquivalent sind:

(i) G besitzt ein 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 4.8. 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 4.9. 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 4.10. 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 4.11. 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 4.12. 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 4.13. 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 Markov-Ketten als Grundlage der Funktionsweise von Suchmaschinen im Internet

Ziel dieses Kapitels ist, einen kurzen Überblick über die Arbeitsweise von Suchmaschinen für das Internet zu geben. Wir betrachten hierbei eine Suchmaschine, die als Eingabe ein Stichwort oder eine Liste von Stichworten erhält, und die als Ausgabe eine Liste von Links auf Webseiten geben soll, deren Inhalt relevante Informationen zu den eingegebenen Stichworten enthält. Diese Liste soll so sortiert sein, dass die informativsten Links am weitesten oben stehen.

Die Herausforderungen, die sich beim Bau einer Suchmaschine stellen, sind vielfältig. Zum einen ist die Anzahl der Webseiten *sehr* groß: Bereits im Jahr 2005 gab es mehr als 8 Milliarden Webseiten. Niemand kennt den genauen Inhalt des gesamten Internets, und das Internet verändert sich ständig: Täglich kommen neue Webseiten hinzu, viele Webseiten werden täglich aktualisiert, und andere nach einiger Zeit auch wieder gelöscht.

Eine Suchmaschine muss daher eine enorm große Menge von Daten verarbeiten, die in kurzen Zeitabständen immer wieder aktualisiert werden. Trotzdem müssen Suchanfragen, die an eine Suchmaschine geschickt werden, in "Echtzeit" beantwortet werden. Um die Ergebnisse nach ihrer Relevanz für die jeweiligen Suchbegriffe sortieren zu können, benötigt man auch ein sinnvolles Maß dafür, welche Webseiten als besonders "informativ" bewertet werden sollen.

5.1 Die Architektur von Suchmaschinen

Die Herausforderung besteht darin, Anfragen für einen sich rasant ändernden Suchraum gigantischer Größe ohne merkliche Reaktionszeit zu beantworten. Um dies zu gewährleisten, nutzen Suchmaschinen u.a. die folgenden Komponenten:

(1) **Web-Crawler:** Computerprogramme, die **Crawler** genannt werden, durchforsten das Internet, um neue oder veränderte Webseiten zu identifizieren. Die von den Crawlern gefundenen Informationen über Webseiten und deren Inhalt werden aufbereitet und gespeichert.

(2) **Indexierung:** Die Informationen werden in einer Datenstruktur gespeichert, mit deren Hilfe bei Eingabe eines Suchworts in "Echtzeit" alle Webseiten ermittelt werden können, die das Suchwort enthalten.

(3) **Bewertung der Webseiten:** Die ausgewählten Webseiten werden im Hinblick auf ihren Informationsgehalt (hinsichtlich möglicher Suchworte sowie hinsichtlich ihrer generellen Bedeutung im Internet) bewertet.

Zu jeder vom Crawler gefundenen Webseite wird die URL (d.h. die Adresse) sowie der Inhalt der Webseite gespeichert.

Der Inhalt der Webseite wird analysiert und es werden Informationen darüber gespeichert, welches Wort mit welcher Häufigkeit und an welchen Positionen (etwa: im Titel, als Überschrift, im Fließtext, mit welcher Schriftgröße etc.) in der Webseite vorkommt. Diese Informationen werden im so genannten **Index** gespeichert. Index

Außerdem werden die Links, die auf Webseiten angegeben sind, analysiert. Enthält Webseite i einen Link auf eine Webseite j, so wird der Text, mit dem der Link beschriftet ist, im zu j gehörenden Index-Eintrag abgelegt. Diese Linkbeschriftungen geben wertvolle Hinweise darüber, welche Informationen die Webseite j enthält.

Aus dem Index wird der so genannte **invertierte Index** generiert. Dies ist eine Datenstruktur, die zu jedem möglichen Suchwort eine Liste aller Webseiten angibt, die dieses Suchwort enthalten. Dabei werden jeweils auch Zusatzinformationen gespeichert, die die Wichtigkeit des Suchworts innerhalb der Webseite beschreiben, z.B. die Häufigkeit des Stichworts, seine Position und Schriftgröße innerhalb der Webseite sowie das Vorkommen des Stichworts in Beschriftungen von Links *auf* die Webseite. invertierter Index

Die **Link-Struktur** des Internets kann man durch einen gerichteten Graphen modellieren, bei dem jede Webseite (d.h. jede URL) durch einen Knoten repräsentiert wird, und bei dem es eine Kante von Knoten i zu Knoten j gibt, wenn die Webseite i einen Link auf Webseite j enthält. Dieser Graph wird **Link-Index** oder **Web-Graph** genannt. Der Web-Graph wird üblicherweise als Adjazenzliste gespeichert. Link-Index Web-Graph

Bearbeitung von Such-Anfragen:

Bei Eingabe einer Liste von Such-Stichworten soll die Suchmaschine die hinsichtlich dieser Stichworte informativsten Webseiten finden und diese sortiert nach ihrer Relevanz anzeigen. Dabei werden folgende Kriterien berücksichtigt:

(1) die Häufigkeit und Positionierung der Suchbegriffe auf der jeweiligen Webseite sowie in der Beschriftung von Links, die auf diese Webseite verweisen, und

(2) die grundlegende Bedeutung einer Webseite.

Für (1) können Methoden aus dem Bereich **Information Retrieval** verwendet werden; Details dazu finden sich z.B. in Kapitel 6 von [21].

Für (2) wird die Link-Struktur des Internets, d.h. der Web-Graph berücksichtigt. Als Rechtfertigung für die Güte dieses Ansatzes, geht man von der folgenden Annahme aus: Wenn eine Webseite i einen Link auf eine Webseite j enthält, dann

- gibt es eine inhaltliche Beziehung zwischen beiden Webseiten, und
- der Autor der Webseite i hält die Informationen auf Webseite j für wertvoll.

Es gibt verschiedene Verfahren, die Maße für die grundlegende Bedeutung einer Webseite liefern, beispielsweise das von *Google* genutzte **Page-Rank** Verfahren von Brin und Page [3] oder die **HITS** (Hypertext Induced Topic Search) Methode von Kleinberg [16]. Beide Ansätze versuchen, die in der Link-Struktur manifestierte "relative Wertschätzung" zwischen einzelnen Webseiten in eine "grundlegende Bedeutung" der Webseiten umzurechnen. Details zu den beiden Verfahren finden sich in dem Buch [18].

Bei der Bearbeitung einer Suchanfrage, bei der eine Liste s von Such-Stichworten eingegeben wird, wird dann unter Verwendung von (1) und (2) jeder Webseite i ein Wert $Score(i, s)$ zugeordnet, der als Maß für die Relevanz der Webseite i hinsichtlich der Suchanfrage s dient. Als Trefferliste gibt die Suchmaschine dann eine Liste aller Webseiten aus, deren Score über einer

bestimmten Schranke liegt und sortiert die Liste so, dass die Webseiten mit dem höchsten Score am weitesten oben stehen. Wie der Wert $Score(i, s)$ gewählt wird, ist Betriebsgeheimnis der einzelnen Betreiber von Suchmaschinen.

Im Rest dieses Kapitels werden wir uns anhand des Page-Rank Verfahrens etwas genauer ansehen, wie die "grundlegende Bedeutung" einer Webseite modelliert und berechnet werden kann.

5.2 Der Page-Rank einer Webseite

Der Page-Rank liefert ein Maß für die "grundlegende Bedeutung" einer Webseite, das allein also aus der Link-Struktur des Internets bestimmt wird, ohne dass der textuelle Inhalt einer Webseite dabei berücksichtigt wird.

Wir schreiben im Folgenden $G = (V, E)$, um den Web-Graphen zu bezeichnen. Der Einfachheit halber nehmen wir an, dass die Webseiten mit den Zahlen $1, \ldots, n$ durchnummeriert sind (wobei $n = |V|$ ist), und dass $V = \{1, 2, \ldots, n\}$ ist.

Jeder Knoten von G repräsentiert eine Webseite, und jede Kante $(i, j) \in E$ modelliert einen Link von Webseite i auf Webseite 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 der Webseite i auf andere Webseiten verweisen. Für eine Webseite $j \in V$ schreiben wir $\text{Vor}_G(j)$, um die Menge aller Webseiten zu bezeichnen, die einen Link auf j enthalten, d.h.

$$\text{Vor}_G(j) \;=\; \{i \in V : (i, j) \in E\}.$$

Die Elemente in $\text{Vor}_G(j)$ werden **Vorgänger** von j genannt.

PR_i
Page-Rank

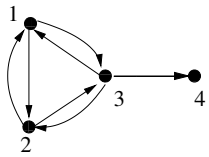
Die "grundlegende Bedeutung" einer Webseite i wird im Folgenden durch eine Zahl PR_i modelliert, dem so genannten **Page-Rank** von i. Der Wert PR_i soll die Qualität (im Sinne von "Renommee" oder "Ansehen") von Webseite i widerspiegeln; die Zahl PR_i soll umso größer sein, je höher das Renommee der Webseite i ist. Das Renommee (und damit der Wert PR_j) einer Webseite j wird als hoch bewertet, wenn viele Webseiten i mit hohem Page-Rank PR_i einen Link auf die Seite j enthalten. Die Werte PR_i, die allen Webseiten $i \in V$ zugeordnet werden, werden daher so gewählt, dass folgendes gilt:

> Eine Webseite i mit a_i ausgehenden Links "vererbt" ihren Page-Rank an jede Webseite j mit $(i, j) \in E$ um den Anteil $\frac{\text{PR}_i}{a_i}$.

Mit dieser Sichtweise müsste also für alle $j \in V$ mit $\text{Vor}_G(j) \neq \emptyset$ gelten:

$$\text{PR}_j \;=\; \sum_{i \in \text{Vor}_G(j)} \frac{\text{PR}_i}{a_i}. \tag{5.1}$$

Ein Problem stellen hierbei Knoten dar, deren Ausgangsgrad 0 ist, da solche Knoten ihren Page-Rank nicht an andere Knoten weitervererben und daher zu Werten PR_i führen können, die kein sinvolles Maß für die Bedeutung einer Webseite liefern. Als Beispiel betrachte man den folgenden Graphen $G = (V, E)$:

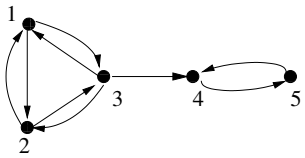


Die einzigen Werte $PR_1, PR_2, PR_3, PR_4 \in \mathbb{R}$, die die Gleichung (5.1) erfüllen, sind $PR_1 = PR_2 = PR_3 = PR_4 = 0$. Diese Werte spiegeln aber nicht die intuitive "grundlegende Bedeutung" wider, die man den Webseiten 1, 2, 3 und 4 zuordnen würde.

Im Folgenden werden **Knoten vom Ausgangsgrad 0** auch **Senken** genannt. Zur Bestimmung des Page-Ranks betrachtet man in der Regel nur **Graphen ohne Senken**, d.h. gerichtete Graphen, bei denen jeder Knoten einen Ausgangsgrad $\geqslant 1$ hat. Natürlich gibt es keine Garantie, dass der Web-Graph keine Senken besitzt. Die Autoren von [3, 23] schlagen zwei Möglichkeiten vor, den Web-Graphen in einen Graphen ohne Senken zu transformieren: Die eine Möglichkeit ist, von jeder Senke Kanten zu *allen* Knoten hinzuzufügen. Die andere Möglichkeit ist, alle Senken zu löschen und dies rekursiv so lange zu tun, bis ein Graph übrig bleibt, der keine Senke besitzt. Wir nehmen im Folgenden an, dass eine dieser beiden Transformationen durchgeführt wurde und dass der Web-Graph durch einen endlichen gerichteten Graphen $G = (V, E)$ repräsentiert wird, der keine Senke besitzt.

Senke

Ein weiteres Problem stellen Knotenmengen dar, die unter sich zwar verbunden sind, die aber keine Kante zu einem anderen Knoten des Graphen G enthalten. Als einfaches Beispiel betrachten wir den folgenden Graphen $G = (V, E)$:



Man kann sich leicht davon überzeugen, dass Werte $PR_1, PR_2, PR_3, PR_4, PR_5 \in \mathbb{R}$ genau dann die Gleichung (5.1) erfüllen, wenn $PR_1 = PR_2 = PR_3 = 0$ und $PR_4 = PR_5$ ist. Ähnlich wie im vorherigen Beispiel spiegeln diese Werte nicht die intuitive "grundlegende Bedeutung" wider, die man den Webseiten 1–5 zuordnen würde. D.h. die durch die Gleichung (5.1) gegebenen Werte $PR_1, \ldots, PR_5$ liefern kein sinnvolles Maß, um die grundlegende Bedeutung der einzelnen Webseiten zu bewerten.

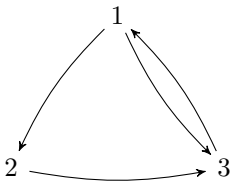
Um dieses Problem zu vermeiden, wird die Vererbung von PR_i auf die Nachfolgeseiten j mit $(i, j) \in E$ meistens um einen **Dämpfungsfaktor** d mit $0 \leqslant d \leqslant 1$ abgeschwächt.
Dies wird in der folgenden Definition präzisiert.

Definition 5.1 (Page-Rank-Eigenschaft).
Sei d eine reelle Zahl mit $0 \leqslant d \leqslant 1$. Die Zahl d wird im Folgenden **Dämpfungsfaktor** genannt. Dämpfungsfaktor
Sei $G = (V, E)$ ein gerichteter Graph, der keine Senke besitzt, und sei $n := |V| \in \mathbb{N}_{>0}$ und $V = \{1, \ldots, n\}$.
Für alle $i, j \in V$ sei $a_i := \text{Aus-Grad}_G(i)$ und $\text{Vor}_G(j) := \{i \in V : (i, j) \in E\}$.
Ein Tupel $PR = (PR_1, \ldots, PR_n) \in \mathbb{R}^n$ hat die **Page-Rank-Eigenschaft bezüglich** d, wenn für alle $j \in V$ gilt: Page-Rank-Eigenschaft bezüglich d

$$PR_j \quad = \quad \frac{1-d}{n} \quad + \; d \cdot \sum_{i \in \text{Vor}_G(j)} \frac{PR_i}{a_i}. \tag{5.2}$$

Beachte: Für den Dämpfungsfaktor $d = 1$ erhält man gerade die Gleichung (5.1).
Für den Dämpfungsfaktor $d = 0$ ist $\mathrm{PR}_1 = \mathrm{PR}_2 = \ldots = \mathrm{PR}_n = \frac{1}{n}$.
In [3] wird empfohlen, den Wert $d = 0.85 = \frac{17}{20}$ zu wählen.

Beispiel 5.2. Zur Veranschaulichung der Page-Rank-Eigenschaft betrachten wir den Dämpfungsfaktor $d := \frac{1}{2}$ und den folgenden Graphen $G = (V, E)$:



Wir suchen ein Tupel $\mathrm{PR} = (\mathrm{PR}_1, \mathrm{PR}_2, \mathrm{PR}_3)$ von reellen Zahlen, das die Page-Rank-Eigenschaft bzgl. $d = \frac{1}{2}$ hat, d.h. es gilt:

(1) $\mathrm{PR}_1 = \frac{1}{2\cdot 3} + \frac{1}{2} \cdot \frac{\mathrm{PR}_3}{1}$

(2) $\mathrm{PR}_2 = \frac{1}{2\cdot 3} + \frac{1}{2} \cdot \frac{\mathrm{PR}_1}{2}$

(3) $\mathrm{PR}_3 = \frac{1}{2\cdot 3} + \frac{1}{2} \cdot \left(\frac{\mathrm{PR}_1}{2} + \frac{\mathrm{PR}_2}{1}\right)$.

Die Werte PR_1, PR_2 und PR_3 können wir daher finden, indem wir das Lineare Gleichungssystem lösen, das aus den folgenden drei Gleichungen besteht:

(1) $1 \cdot \mathrm{PR}_1 - \frac{1}{2} \cdot \mathrm{PR}_3 = \frac{1}{6}$

(2) $-\frac{1}{4} \cdot \mathrm{PR}_1 + 1 \cdot \mathrm{PR}_2 = \frac{1}{6}$

(3) $-\frac{1}{4} \cdot \mathrm{PR}_1 - \frac{1}{2} \cdot \mathrm{PR}_2 + 1 \cdot \mathrm{PR}_3 = \frac{1}{6}$

Die Auflösung dieses linearen Gleichungssystems (z.B. mittels **Gauß-Elimination**) liefert die Werte

$$\mathrm{PR}_1 = \tfrac{14}{39}, \quad \mathrm{PR}_2 = \tfrac{10}{39}, \quad \mathrm{PR}_3 = \tfrac{15}{39}.$$

□Ende Beispiel 5.2

Auf die gleiche Art wie in diesem Beispiel erhält man auch für den Web-Graphen und einen geeigneten Dämpfungsfaktor d ein entsprechendes lineares Gleichungssystem. Um den Page-Rank der einzelnen Webseiten zu berechnen, müssen wir "nur" dieses lineare Gleichungssystem lösen. Dabei stellen sich folgende Probleme:

(1) Zunächst ist völlig unklar, ob dieses lineare Gleichungssystem überhaupt eine Lösung besitzt, und falls ja, ob die Lösung eindeutig ist. Anhand von Definition 5.1 ist nämlich prinzipiell auch denkbar, dass es gar kein Tupel gibt, das die Page-Rank-Eigenschaft bzgl. d hat, oder dass es mehrere verschiedene Tupel gibt, die die Page-Rank-Eigenschaft bzgl. d besitzen.

(2) Das lineare Gleichungssystem hat n Unbekannte, wobei n die Anzahl der Webseiten im Internet ist — und diese Zahl ist enorm groß. Um den Page-Rank aller Webseiten zu bestimmen, benötigen daher ein extrem effizientes Verfahren zum Lösen dieses linearen Gleichungssystems.

In den folgenden beiden Abschnitten werden wir sehen, dass die Theorie der **Markov-Ketten** uns hilft, diese Probleme zu lösen. Dazu ist die im folgenden Abschnitt dargestellte Sichtweise auf den Page-Rank sehr hilfreich.

5.3 Der Zufalls-Surfer

Wir nehmen an, dass der Webgraph durch einen gerichteten Graphen $G = (V, E)$ mit Knotenmenge $V = \{1, \ldots, n\}$ repräsentiert wird, der keine Senke besitzt. Des Weiteren sei d eine beliebige reelle Zahl mit $0 \leqslant d \leqslant 1$.

Wir betrachten einen **Zufalls-Surfer** (englisch: **random surfer**), der auf einer beliebigen Webseite beginnt und beliebige Links verfolgt, ohne dabei auf Inhalte zu achten. Wenn der Zufalls-Surfer auf einer Webseite i ist, so wählt er Zufalls-Surfer

- mit Wahrscheinlichkeit d einen Link, der von Seite i ausgeht. Hierbei wird dann jeder der $a_i = \text{Aus-Grad}_G(i)$ ausgehenden Links mit derselben Wahrscheinlichkeit $\frac{d}{a_i}$ ausgewählt.
- mit Wahrscheinlichkeit $(1 - d)$ eine **beliebige** Webseite im Web-Graphen. Hierbei wird dann jede der n Webseiten mit derselben Wahrscheinlichkeit $\frac{1-d}{n}$ ausgewählt.

Für alle $i, j \in V$ gibt daher

$$p_{i,j} \;:=\; \begin{cases} \dfrac{1-d}{n} + \dfrac{d}{a_i} & \text{, falls } (i,j) \in E \\[2ex] \dfrac{1-d}{n} & \text{, falls } (i,j) \notin E \end{cases} \tag{5.3}$$

die Wahrscheinlichkeit an, mit der der Zufalls-Surfer in einem Schritt von Seite i zu Seite j wechselt. Diese Wahrscheinlichkeiten, mit denen sich der Zufalls-Surfer von Knoten zu Knoten bewegt, lassen sich kompakt durch die folgende Matrix darstellen.

Definition 5.3 (Die Page-Rank-Matrix $P(G, d)$).
Sei $d \in \mathbb{R}$ mit $0 \leqslant d \leqslant 1$, sei $n \in \mathbb{N}_{>0}$ und sei $G = (V, E)$ mit $V = \{1, \ldots, n\}$ ein gerichteter Graph ohne Senke. Für jedes $i \in V$ sei $a_i := \text{Aus-Grad}_G(i)$. Die **Page-Rank-Matrix** ist die $n \times n$-Matrix Page-Rank-Matrix

$$P(G,d) \;:=\; \begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix},$$

wobei für alle $i, j \in V$ der Eintrag in Zeile i und Spalte j der in Gleichung (5.3) festgelegte Wert $p_{i,j}$ ist. Wir schreiben auch kurz $(p_{i,j})_{i,j=1,\ldots,n}$, um die Matrix $P(G, d)$ zu bezeichnen.

Beispiel 5.4. Für den Wert $d = \frac{1}{2}$ und den Graphen G aus Beispiel 5.2 ist beispielsweise $p_{1,1} = \frac{1}{6}$, $p_{1,2} = \frac{1}{6} + \frac{1}{4} = \frac{5}{12}$, $p_{2,3} = \frac{1}{6} + \frac{1}{2} = \frac{2}{3}$ und insgesamt

$$P(G,d) \;=\; \begin{pmatrix} \frac{1}{6} & \frac{5}{12} & \frac{5}{12} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix}.$$

Um den Zusammenhang zwischen dem Zufalls-Surfer, der Page-Rank-Matrix und Tupeln mit der Page-Rank-Eigenschaft beschreiben zu können, benötigen wir folgende Notation für das Rechnen mit Matrizen.

Definition 5.5 (Vektor-Matrix-Produkt).
Sei $n \in \mathbb{N}_{>0}$, und für alle $i, j \in \{1, \ldots, n\}$ sei $p_{i,j}$ eine reelle Zahl. Sei $P := (p_{i,j})_{i,j=1,\ldots,n}$ die $n \times n$-Matrix, die in Zeile i und Spalte j den Eintrag $p_{i,j}$ hat (für alle $i, j \in \{1, \ldots, n\}$). Ist $X = (X_1, \ldots, X_n)$ ein Tupel aus n reellen Zahlen, so ist das **Vektor-Matrix-Produkt**

Vektor-Matrix-Produkt
$X \cdot P$

$$X \cdot P$$

das Tupel $Y = (Y_1, \ldots, Y_n) \in \mathbb{R}^n$, bei dem für jedes $j \in \{1, \ldots, n\}$ gilt:

$$Y_j \;:=\; \sum_{i=1}^{n} X_i \cdot p_{i,j}.$$

Beispiel 5.6. Sei $P := P(G, d)$ die Matrix aus Beispiel 5.4 und sei $X := (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. Dann gilt:

$$X \cdot P \quad = \quad \left(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}\right) \cdot \begin{pmatrix} \frac{1}{6} & \frac{5}{12} & \frac{5}{12} \\ \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{6} & \frac{1}{6} \end{pmatrix} \quad = \quad \left(\tfrac{1}{3}, \tfrac{1}{4}, \tfrac{5}{12}\right).$$

Der folgende Satz beschreibt den genauen Zusammenhang zwischen Zufalls-Surfer, Page-Rank-Matrix und Tupeln mit der Page-Rank-Eigenschaft.

Satz 5.7. Sei $d \in \mathbb{R}$ mit $0 \leqslant d < 1$, sei $n \in \mathbb{N}_{>0}$ und sei $G = (V, E)$ ein gerichteter Graph mit $V = \{1, \ldots, n\}$, der keine Senke besitzt. Dann gilt:

(a) Ist $\mathrm{PR} = (\mathrm{PR}_1, \ldots, \mathrm{PR}_n) \in \mathbb{R}^n$ ein Tupel, das die Page-Rank-Eigenschaft bzgl. d besitzt, so ist $\sum_{i=1}^{n} \mathrm{PR}_i = 1$.

(b) Für jedes Tupel $X = (X_1, \ldots, X_n) \in \mathbb{R}^n$ mit $\sum_{i=1}^{n} X_i = 1$ gilt:

$$X \text{ besitzt die Page-Rank-Eigenschaft bzgl. } d \quad \Longleftrightarrow \quad X \cdot P(G, d) \;=\; X.$$

Beweis:

(a) *Übung.*

(b) Sei $X = (X_1, \ldots, X_n) \in \mathbb{R}^n$ mit $\sum_{i=1}^{n} X_i = 1$. Sei $Y = (Y_1, \ldots, Y_n)$ so dass $X \cdot P(G, d) = Y$.

Dann gilt gemäß Definition 5.5 und Definition 5.3 für jedes $j \in \{1, \ldots, n\}$, dass

$$\begin{aligned} Y_j &= \sum_{i=1}^{n} X_i \cdot p_{i,j} \overset{\text{Gl. (5.3)}}{=} \sum_{i=1}^{n} X_i \cdot \frac{1-d}{n} + \sum_{i \in \text{Vor}_G(j)} X_i \cdot \frac{d}{a_i} \\ &= \frac{1-d}{n} \cdot \sum_{i=1}^{n} X_i + d \cdot \sum_{i \in \text{Vor}_G(j)} \frac{X_i}{a_i} \\ &\overset{\sum_{i=1}^{n} X_i = 1}{=} \frac{1-d}{n} + d \cdot \sum_{i \in \text{Vor}_G(j)} \frac{X_i}{a_i} , \end{aligned}$$

d.h. es gilt

$$Y_j = \frac{1-d}{n} + d \cdot \sum_{i \in \text{Vor}_G(j)} \frac{X_i}{a_i} . \tag{5.4}$$

Aus Definition 5.1 zusammen mit Gleichung (5.4) folgt:

$$\begin{aligned} & X \text{ besitzt die Page-Rank-Eigenschaft bzgl. } d \\ \iff & \text{f.a. } j \in \{1, \ldots, n\} \text{ gilt: } X_j = Y_j \\ \iff & X \cdot P(G, d) = X. \end{aligned}$$

□

Beachte: Für Satz 5.7 (a) ist wichtig, dass $d \neq 1$ ist und dass G keine Senke besitzt.

Notation 5.8 (Eigenvektor).
Ein Vektor $X = (X_1, \ldots, X_n)$ heißt **linker Eigenvektor zum Eigenwert** 1 der $n \times n$-Matrix P, falls gilt: $X \cdot P = X$ und $X \neq (0, \ldots, 0)$. Eigenvektor

Satz 5.7 besagt also, dass ein Tupel $\text{PR} = (\text{PR}_1, \ldots, \text{PR}_n) \in \mathbb{R}^n$ genau dann die Page-Rank-Eigenschaft bzgl. d besitzt, wenn es ein linker Eigenvektor zum Eigenwert 1 der Matrix $P(G, d)$ ist, für den $\sum_{i=1}^{n} \text{PR}_i = 1$ ist.

Diese Sichtweise auf den Page-Rank sowie die im folgenden Abschnitt vorgestellte Theorie der Markov-Ketten helfen uns, um die beiden am Ende von Abschnitt 5.2 gestellten Probleme zu lösen.

5.4 Markov-Ketten

Markov-Ketten sind nach dem russischen Mathematiker Andrei A. Markov (1856–1922) benannt. In der Literatur werden unterschiedliche Schreibweisen des Namens verwendet, z.B. Markov, Markow oder Markoff.

Definition 5.9 (Markov-Kette).
Eine **(homogene) Markov-Kette** mit **Übergangsmatrix** P wird durch eine $n \times n$-Matrix Markov-Kette, Übergangsmatrix

$$P = \big(p_{i,j}\big)_{i,j=1\ldots,n}$$

mit $n \in \mathbb{N}_{>0}$ beschrieben, für die gilt:

(1) $p_{i,j} \geqslant 0$ für alle $i, j \in \{1, \ldots, n\}$, und

(2) für jede Zeile $i \in \{1, \ldots, n\}$ gilt: $\sum_{j=1}^{n} p_{i,j} = 1$.

stochastische Matrix

Eine Matrix P, die die Eigenschaften (1) und (2) besitzt, wird auch **stochastische Matrix** genannt.

Der **zu P gehörende Graph** ist der gerichtete Graph mit Knotenmenge $V = \{1, \ldots, n\}$, so dass für alle $i, j \in \{1, \ldots n\}$ gilt: Es gibt in G genau dann eine Kante von i nach j, wenn $p_{i,j} > 0$ ist. Den Eintrag $p_{i,j}$ in Zeile i und Spalte j von P kann man als Wahrscheinlichkeit dafür auffassen, dass ein Zufalls-Surfer im Graphen G in einem Schritt von Knoten i zu Knoten j springt.

Beispiel 5.10. Sei $G = (V, E)$ ein beliebiger gerichteter Graph mit Knotenmenge $V = \{1, \ldots, n\}$ (für $n := |V| \in \mathbb{N}_{>0}$), der keine Senke besitzt. Seit d eine reelle Zahl mit $0 \leqslant d < 1$ und sei $P := P(G, d)$ die zugehörige Page-Rank-Matrix.

Gemäß der Definition von $P(G, d)$ ist $p_{i,j} > 0$ für alle $i, j \in \{1, \ldots, n\}$ (dazu beachte man, dass $0 \leqslant d < 1$ ist). Außerdem gilt für jede Zeile $i \in \{1, \ldots, n\}$, dass

$$\sum_{j=1}^{n} p_{i,j} \quad = \quad \sum_{j=1}^{n} \frac{1-d}{n} \quad + \sum_{j\,:\,(i,j)\in E} \frac{d}{a_i} \quad \overset{G \text{ ohne Senke}}{=} \quad (1-d) \;+\; a_i \cdot \frac{d}{a_i} \quad = \quad 1.$$

Somit ist P eine stochastische Matrix, die eine Markov-Kette beschreibt. Für jedes $i, j \in \{1, \ldots, n\}$ gibt der Wert $p_{i,j}$ die Wahrscheinlichkeit dafür an, dass der Zufalls-Surfer in einem Schritt von Webseite i zu Webseite j springt.

Da $p_{i,j} > 0$ ist, ist der zu P gehörende Graph der **vollständige gerichtete Graph** auf n Knoten, d.h. der Graph mit Knotenmenge $V = \{1, \ldots, n\}$ und Kantenmenge $V \times V$. Diesen Graphen bezeichnen wir im Folgenden mit $\vec{K}_n$.

$\vec{K}_n$

Die Theorie der Markov-Ketten und der stochastischen Matrizen wurde in der Literatur gut untersucht (siehe [13, 8]). Insbesondere ist folgendes bekannt (vgl. [7]):

Satz 5.11. Sei $n \in \mathbb{N}_{>0}$ und sei $P = (p_{i,j})_{i,j=1,\ldots,n}$ eine stochastische Matrix, bei der für alle $i, j \in \{1, \ldots, n\}$ gilt: $p_{i,j} > 0$. Dann gibt es genau ein Tupel $X = (X_1, \ldots, X_n) \in \mathbb{R}^n$ mit $\sum_{i=1}^{n} X_i = 1$, das ein linker Eigenvektor zum Eigenwert 1 von P ist. Dieses Tupel hat die Eigenschaft, dass für jedes $i \in \{1, \ldots, n\}$ der Wert $X_i > 0$ ist.

Einen Beweis dieses Satzes zu geben, würde den Rahmen dieses Vorlesungsskripts sprengen. Man beachte, dass sich aus der Kombination von Satz 5.11, Beispiel 5.10 und Satz 5.7 die Lösung des am Ende von Abschnitt 5.2 genannten Problems (1) ergibt.

Folgerung 5.12 (Lösung von Problem (1) auf Seite 138).
Ist $G = (V, E)$ ein gerichteter Graph mit $V = \{1, \ldots, n\}$ (für $n \in \mathbb{N}_{>0}$), der keine Senke besitzt, und ist $d \in \mathbb{R}$ ein Dämpfungsfaktor mit $0 \leqslant d < 1$, so gibt es genau ein Tupel $\mathrm{PR} = (\mathrm{PR}_1, \ldots, \mathrm{PR}_n) \in \mathbb{R}^n$, das die Page-Rank-Eigenschaft bezüglich d besitzt. Für dieses Tupel gilt: $\mathrm{PR}_i > 0$ für alle $i \in \{1, \ldots, n\}$ und $\sum_{i=1}^{n} \mathrm{PR}_i = 1$.

5.5 Die effiziente Berechnung des Page-Rank

Um zu sehen, dass die Theorie der Markov-Ketten uns auch eine Lösung für Problem (2) auf Seite 138 liefert, schauen wir uns die Bewegungen des Zufalls-Surfers auf dem Web-Graphen etwas genauer an.

Für unsere Betrachtungen ist folgendermaßen definierte Begriff einer **Verteilung** sehr nützlich.

Definition 5.13. Sei $n \in \mathbb{N}_{>0}$.
Eine **Verteilung** auf $V = \{1, \ldots, n\}$ ist ein Tupel $X = (X_1, \ldots, X_n) \in \mathbb{R}^n$, für das gilt: Verteilung

(1) für alle $i \in \{1, \ldots, n\}$ ist $X_i \geqslant 0$ und

(2) $\sum_{i=1}^{n} X_i = 1$.

Ist G ein gerichteter Graph mit Knotenmenge $V = \{1, \ldots, n\}$ und ist $X = (X_1, \ldots, X_n)$ eine Verteilung auf V, so fassen wir für jedes $i \in V$ die Zahl X_i als Wahrscheinlichkeit dafür auf, dass ein Zufalls-Surfer in G sich auf Knoten i befindet.

Beobachtung 5.14. Sei $n \in \mathbb{N}_{>0}$ und sei $P = (p_{i,j})_{i,j=1,\ldots,n}$ eine stochastische Matrix. Ist $X = (X_1, \ldots, X_n)$ eine Verteilung auf $V := \{1, \ldots, n\}$, so gibt das Tupel $Y = (Y_1, \ldots, Y_n)$ mit

$$X \cdot P \;=\; Y$$

folgendes an: Wenn wir in dem zu P gehörenden Graphen für jedes $i \in V$ den Zufalls-Surfer mit Wahrscheinlichkeit X_i auf Knoten i beginnen lassen, so gibt für jedes $j \in V$ die Zahl

$$Y_j \;=\; \sum_{i=1}^{n} X_i \cdot p_{i,j}$$

die Wahrscheinlichkeit dafür an, dass der Zufalls-Surfer sich nach einem Schritt auf Knoten j befindet.

Rekursiv können so wir für jedes $k \in \mathbb{N}$ eine Verteilung $X^{(k)} = (X_1^{(k)}, \ldots, X_n^{(k)})$ angeben, so dass für jedes $j \in V$ der Wert $X_j^{(k)}$ die Wahrscheinlichkeit dafür angibt, dass der Zufalls-Surfer sich nach k Schritten auf Knoten j befindet. Dazu wählen wir

$$X^{(0)} \;:=\; X \qquad \text{und} \qquad X^{(k+1)} \;:=\; X^{(k)} \cdot P\,, \qquad \text{f.a. } k \in \mathbb{N}.$$

Unter Verwendung des in der folgenden Definition gegebenen Produkts von Matrizen erhalten wir per Induktion nach k, dass für alle $k \in \mathbb{N}_{>0}$ gilt:

$$X^{(k)} \;=\; X \cdot P^k.$$

Definition 5.15 (Matrix-Produkt). Sei $n \in \mathbb{N}_{>0}$, und für alle $i, j \in \{1, \ldots, n\}$ sei $a_{i,j} \in \mathbb{R}$ und $b_{i,j} \in \mathbb{R}$. Wir betrachten die beiden $n \times n$-Matrizen

$$A \;:=\; (a_{i,j})_{i,j=1,\ldots,n} \qquad \text{und} \qquad B \;:=\; (b_{i,j})_{i,j=1,\ldots,n}.$$

(a) Das **Produkt**

$$A \cdot B$$

ist die $n\times n$-Matrix $C = (c_{i,j})_{i,j=1,\ldots,n}$, die für alle $i, j \in \{1, \ldots, n\}$ in Zeile i und Spalte j den Eintrag

$$c_{i,j} \;:=\; \sum_{\ell=1}^{n} a_{i,\ell} \cdot b_{\ell,j}$$

hat.

(b) Für jede Zahl $k \in \mathbb{N}_{>0}$ ist die $n\times n$-Matrix A^k folgendermaßen rekursiv definiert:

$$A^1 := A \qquad \text{und} \qquad A^{k+1} := A \cdot A^k \quad (\text{für alle } k \in \mathbb{N}).$$

Für alle $i, j \in \{1, \ldots, n\}$ schreiben wir $(A^k)_{i,j}$, um den Eintrag in Zeile i und Spalte j der Matrix A^k zu bezeichnen.

Anhand dieser Definition sieht man leicht, dass folgendes gilt:

Beobachtung 5.16. Ist $n \in \mathbb{N}_{>0}$ und ist $P = (p_{i,j})_{i,j=1,\ldots,n}$ eine stochastische Matrix, die eine Markov-Kette beschreibt, so können wir für jedes $k \in \mathbb{N}_{>0}$ den Eintrag $(P^k)_{i,j}$ in Zeile i und Spalte j der Matrix P^k als die Wahrscheinlichkeit dafür auffassen, dass der Zufalls-Surfer auf dem zu P gehörenden Graphen innerhalb von genau k Schritten von Knoten i zu Knoten j gelangt.

Zur effizienten Berechnung des Page-Ranks machen wir uns zu nutze, dass die durch die Page-Rank-Matrix $P(G, d)$ (für $0 \leqslant d < 1$) beschriebene Markov-Kette die folgende Eigenschaft hat:

Definition 5.17 (Ergodische Markov-Ketten).
Sei $n \in \mathbb{N}_{>0}$ und sei $P = (p_{i,j})_{i,j=1,\ldots,n}$ eine stochastische Matrix. Die durch P beschriebene Markov-Kette heißt **ergodisch**, wenn für alle $i, i' \in \{1, \ldots, n\}$ und alle $j \in \{1, \ldots, n\}$ gilt: Die Grenzwerte

ergodisch

$$\lim_{k\to\infty} (P^k)_{i,j} \qquad \text{und} \qquad \lim_{k\to\infty} (P^k)_{i',j}$$

existieren und es gilt

$$\lim_{k\to\infty} (P^k)_{i,j} \quad = \quad \lim_{k\to\infty} (P^k)_{i',j} \quad > \quad 0.$$

Bemerkung 5.18 (eine Charakterisierung ergodischer Markov-Ketten).
Es ist bekannt (für einen Beweis sei auf [13] verwiesen), dass eine durch eine stochastische Matrix $P = (p_{i,j})_{i,j=1,\ldots,n}$ gegebene Markov-Kette genau dann ergodisch ist, wenn sie **irreduzibel** und **aperiodisch** ist. Dabei heißt P

irreduzibel
aperiodisch

- **irreduzibel**, falls der zu P gehörende Graph stark zusammenhängend ist;
- **aperiodisch**, falls für jeden Knoten i im zu P gehörenden Graphen gilt: Der größte gemeinsame Teiler der Längen aller Wege von i nach i ist 1.

Falls $P = P(G, d)$ die Page-Rank-Matrix für einen Dämpfungsfaktor d mit $0 \leqslant d < 1$ und einen gerichteten Graphen G ist, der keine Senke besitzt, so wissen wir aus Beispiel 5.10, dass der zu P gehörende Graph der vollständige gerichtete Graph $\vec{K}_n$ ist. Dieser ist offensichtlicherweise irreduzibel und aperiodisch. Daher beschreibt die Page-Rank-Matrix $P(G, d)$ eine ergodische Markov-Kette.

Beobachtung 5.19 (Eigenschaften ergodischer Markov-Ketten).
Ist P eine stochastische Matrix, die eine ergodische Markov-Kette beschreibt, so gilt offensichtlicherweise folgendes:

(1) Die Matrix

$$P' \;:=\; \Big(\lim_{k\to\infty} \big(P^k\big)_{i,j}\Big)_{i,j=1,\ldots,n} \tag{5.5}$$

ist wohldefiniert (da die Grenzwerte existieren), und

(2) alle Zeilen von P' sind identisch.

Wir schreiben $p' := (p'_1, \ldots, p'_n)$, um die erste Zeile von P' zu bezeichnen.
Die Matrix P' sieht daher folgendermaßen aus:

$$P' \quad = \quad \begin{pmatrix} p' \\ p' \\ \vdots \\ p' \end{pmatrix} \quad = \quad \begin{pmatrix} p'_1 \;\ldots\; p'_n \\ p'_1 \;\ldots\; p'_n \\ \vdots \\ p'_1 \;\ldots\; p'_n \end{pmatrix}.$$

Wegen Gleichung (5.5) gilt $P' \cdot P = P'$, und daher gilt insbesondere für die Verteilung p', dass

$$p' \cdot P \;=\; p',$$

d.h. p' ist ein linker Eigenvektor zum Eigenwert 1 der Matrix P.

Notation: Eine Verteilung Y mit $Y \cdot P = Y$ wird auch **stationäre Verteilung** für P genannt. stationäre Verteilung

Für jede beliebige Verteilung $X = (X_1, \ldots, X_n)$ gilt:

$$X \cdot P' \;=\; p', \tag{5.6}$$

denn für jedes $j \in V$ ist der j-te Eintrag im Tupel $X \cdot P'$ gerade die Zahl $\sum_{i=1}^n X_i \cdot p'_j \;=\; p'_j \cdot \sum_{i=1}^n X_i \;=\; p'_j$. Daher gilt:

(a) $p' = (p'_1, \ldots, p'_n)$ ist die **einzige** stationäre Verteilung, die P besitzt, und

(b) wenn der Zufalls-Surfer im zu P gehörenden Graphen seinen Startknoten gemäß einer beliebigen Anfangsverteilung $X = (X_1, \ldots, X_n)$ wählt und hinreichend viele Schritte macht, so ist für jedes $j \in V$ die Wahrscheinlichkeit, bei Knoten j zu landen beliebig nah bei p'_j. Die Wahl des Anfangsknotens ist für einen Zufalls-Surfer, der hinreichend lange surft, also ohne Belang.

Aufgrund der Gleichungen (5.5) und (5.6) erhalten wir:

$$p' \;\overset{(5.6)}{=}\; X \cdot P' \;\overset{(5.5)}{=}\; X \cdot \lim_{k\to\infty} P^k \;=\; \lim_{k\to\infty} \big(X \cdot P^k\big) \;=\; \lim_{k\to\infty} X^{(k)},$$

wobei $X^{(0)} := X$ und $X^{(k+1)} := X^{(k)} \cdot P$, f.a. $k \in \mathbb{N}$.

Um eine Näherung für das Tupel p' zu berechnen, können wir daher wie folgt vorgehen: Wir starten mit einer beliebigen Verteilung $X^{(0)} = X$ (etwa der **Gleichverteilung** $X = (\frac{1}{n}, \cdots, \frac{1}{n})$) und berechnen nacheinander für $k = 1, 2, 3$ usw. das Tupel $X^{(k+1)} := X^{(k)} \cdot P$. Dieser Prozess

wird beendet, sobald das Tupel $X^{(k+1)}$ sich nicht mehr viel vom Tupel $X^{(k)}$ unterscheidet, d.h. sobald für jedes $j \in \{1, \ldots, n\}$ die Zahl $|X_j^{(k+1)} - X_j^{(k)}|$ kleiner als eine vorher festgelegte Schranke ε ist (wobei $X_j^{(k+1)}$ und $X_j^{(k)}$ der Eintrag in der j-ten Komponente von $X^{(k+1)}$ bzw. $X^{(k)}$ ist). □Beobachtung 5.19

Folgerung 5.20 (Lösung von Problem (2) auf Seite 138).
Sei $P := P(G, d)$ die Page-Rank-Matrix für einen Dämpfungsfaktor d mit $0 \leqslant d < 1$ und einen gerichteten Graphen $G = (V, E)$ ohne Senke. Von Bemerkung 5.18 wissen wir, dass P ergodisch ist. Aus Beobachtung 5.19 folgt daher, dass die stationäre Verteilung p' von P das (eindeutig festgelegte) Tupel ist, das die Page-Rank-Eigenschaft bzgl. d besitzt. Das am Ende von Beobachtung 5.19 beschriebene Vorgehen liefert ein effizientes Verfahren, um eine Näherung für das Tupel p' zu berechnen.

Aus der Theorie der Markov-Ketten und den speziellen Eigenschaften der Page-Rank-Matrix $P(G, d)$ (für d mit $0 \leqslant d < 1$) ergibt sich, dass aufgrund des hohen Zusammenhangs des Web-Graphen die Folge der Tupel $X^{(k)}$ für $k = 0, 1, 2, 3$ usw. sehr schnell gegen die stationäre Verteilung p' konvergiert. Details dazu finden sich in [18, 13].

Für eine schnelle Berechnung des Vektor-Matrix-Produkts $X^{(k+1)} := X^{(k)} \cdot P(G, d)$ wird ausgenutzt, dass $P(G, d)$ viele identische Einträge der Form $\frac{1-d}{n}$ hat. Außerdem ist die Berechnung des Vektor-Matrix-Produkts sehr gut parallelisierbar. Details hierzu finden sich in [18].

Derzeit werden mehrere Tausend PCs eingesetzt, die mehrere Stunden zur Berechnung des Page-Ranks benötigen — was in Anbetracht der Tatsache, dass es mehrere Milliarden Webseiten gibt, erstaunlich gering ist.

5.6 Literaturhinweise

Zur vertiefenden Lektüre seien Kapitel 2 von [25] sowie das Buch [13] empfohlen, das eine Algorithmen-orientierte Einführung in die Theorie der Markov-Ketten gibt. Einen Überblick über die Architektur von Suchmaschinen gibt der Artikel [1]; Details zum Page-Rank und zum HITS Verfahren finden sich in dem Buch [18] sowie in den Originalarbeiten [3, 23, 16, 7]. Als Einführung ins Thema *Information Retrieval* sei das Buch [21] empfohlen. Das Buch [8] ist ein "Klassiker", der eine umfassende Einführung in die Wahrscheinlichkeitstheorie (und insbesondere auch ins Thema Markov-Ketten) gibt.

Viele Informationen und Literaturhinweise zum Thema *Suchmaschinen* finden sich auf der Webseite von Martin Sauerhoffs Vorlesung *Internet Algorithmen* an der TU Dortmund; siehe `http://ls2-www.cs.uni-dortmund.de/lehre/winter200910/IntAlg/`. Ein kurzer und allgemein verständlicher Überblick über das Page-Rank Verfahren wird in dem Spiegel-Online Artikel *Wie Google mit Milliarden Unbekannten rechnet* von Holger Dambeck gegeben; siehe `http://www.spiegel.de/wissenschaft/mensch/0,1518,646448,00.html`.

Quellennachweis: Teile dieses Kapitels orientieren sich an [25].

6 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.

6.1 Motivation zur Logik erster Stufe

Grenzen der Aussagenlogik:

Beispiel 6.1 (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 6.2 (Arithmetische Aussagen). Die Aussagenlogik kann helfen, um Sätze der Art

> "Wenn eine Zahl gerade ist, dann ist sie nicht ungerade."

zu formalisieren. Für viele andere Aussagen ist die Aussagenlogik aber eher ungeeignet, zum Beispiel:

> "Es gibt eine Zahl, die nicht Summe zweier Primzahlen ist."

Ein Überblick über die Logik erster Stufe:

Die Logik erster Stufe ist ein Formalismus, mit dem man die in den beiden obigen Beispielen genannten Aussagen bequem beschreiben kann. Genau wie die Aussagenlogik besitzt die Logik erster Stufe:

- eine **Syntax**, die 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 6.1) oder den natürlichen Zahlen und deren Addition und Multiplikation (Beispiel 6.2)) 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.

6.2 Strukturen

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

- 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.

Signatur
Vokabular
Symbolmenge
Stelligkeit

Definition 6.3. Eine **Signatur** (bzw. ein **Vokabular** bzw. eine **Symbolmenge**; englisch: signature, vocabulary) ist eine Menge σ von Relationssymbolen, Funktionssymbolen und/oder Konstantensymbolen. Jedes Relationssymbol $\dot{R} \in \sigma$ und jedes Funktionssymbol $\dot{f} \in \sigma$ hat eine **Stelligkeit** (bzw. Arität, engl. arity)

$$\mathrm{ar}(\dot{R}) \in \mathbb{N}_{>0} \quad \text{bzw.} \quad \mathrm{ar}(\dot{f}) \in \mathbb{N}_{>0}.$$

Notation 6.4.

- In diesem Kapitel bezeichnet der griechische Buchstabe σ (in Worten: sigma) immer eine Signatur.
- Wir kennzeichnen Symbole aus σ immer mit einem Punkt, wie in $\dot{R}$ bzw. $\dot{f}$.
- Für Relationssymbole verwenden wir meistens Großbuchstaben wie $\dot{R}, \dot{P}, \dot{E}, \dot{R}_1, \dot{R}_2, \ldots$,
  für Funktionssymbole verwenden wir meistens Kleinbuchstaben wie $\dot{f}, \dot{g}, \dot{h}, \ldots$,
  für Konstantensymbole verwenden wir meistens Kleinbuchstaben wie $\dot{c}, \dot{d}, \ldots$.

- Gelegentlich verwenden wir als Relations- und Funktionssymbole auch Zeichen wie

$$\begin{array}{ll} \dot{\leqslant} & \text{(2-stelliges Relationssymbol)}, \\ \dot{+}, \dot{\times} & \text{(2-stellige Funktionssymbole)}, \\ \dot{0}, \dot{1} & \text{(Konstantensymbole)}. \end{array}$$

Definition 6.5. Eine **Struktur über der Signatur** σ (kurz: **σ-Struktur**) ist ein Paar

Struktur
σ-Struktur

$$\mathfrak{A} \;=\; (A, \alpha),$$

bestehend aus:

- einer nicht-leeren Menge A, dem so genannten **Universum** (bzw. **Träger**, engl.: universe, domain) von $\mathfrak{A}$, und

Universum
Träger

- 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 6.6.

- 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

$$\dot{S}^{\mathfrak{A}} \;\text{ an Stelle von }\; \alpha(\dot{S}).$$

  An Stelle von $\mathfrak{A} = (A, \alpha)$ schreiben wir oft auch $\mathfrak{A} = (A, (\dot{S}^{\mathfrak{A}})_{\dot{S} \in \sigma})$.
  Falls $\sigma = \{\dot{R}_1, \ldots, \dot{R}_k, \dot{f}_1, \ldots, \dot{f}_l, \dot{c}_1, \ldots, \dot{c}_m\}$ ist, schreiben wir auch

$$\mathfrak{A} \;=\; (A, \dot{R}_1^{\mathfrak{A}}, \ldots, \dot{R}_k^{\mathfrak{A}}, \dot{f}_1^{\mathfrak{A}}, \ldots, \dot{f}_l^{\mathfrak{A}}, \dot{c}_1^{\mathfrak{A}}, \ldots, \dot{c}_m^{\mathfrak{A}}).$$

Beispiel 6.7 (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 6.8 (Graphen und Bäume). Sei $\sigma_{\mathrm{Graph}} := \{\dot{E}\}$, wobei $\dot{E}$ ein 2-stelliges Relationssymbol ist. Jeder gerichtete Graph bzw. gerichtete Baum (V, E) lässt sich als σ_{Graph}-Struktur $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ mit Universum $A := V$ und Relation $\dot{E}^{\mathfrak{A}} := E$ auffassen.

Beispiel 6.9 (Ordnungen). Sei $\sigma_{\text{Ord}} := \{\dot{\leqslant}\}$, wobei $\dot{\leqslant}$ ein 2-stelliges Relationssymbol ist. Jeder Präordnung, partiellen Ordnung oder linearen Ordnung $\leqslant$ auf einer Menge A entspricht eine σ_{Ord}-Struktur

$$\mathfrak{A} \;=\; (A, \dot{\leqslant}^{\mathfrak{A}})$$

mit $\dot{\leqslant}^{\mathfrak{A}} := \leqslant$.

Frage: Wann sind zwei σ-Strukturen $\mathfrak{A}$ und $\mathfrak{B}$ "prinzipiell gleich" (Fachbegriff: isomorph)?

Antwort: Falls $\mathfrak{B}$ aus $\mathfrak{A}$ entsteht, indem man die Elemente des Universums von $\mathfrak{A}$ umbenennt.

Analog zum Begriff der Isomorphie von Graphen (Definition 4.25) wird dies durch folgende Definition präzisiert:

isomorph

Definition 6.10. 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:

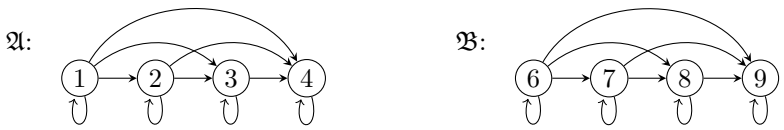
- für jedes Relationssymbol $\dot{R} \in \sigma$, für $r := \text{ar}(\dot{R})$ und für alle r-Tupel $(a_1, \ldots, a_r) \in A^r$ gilt:
$$(a_1, \ldots, a_r) \in \dot{R}^{\mathfrak{A}} \iff \big(\pi(a_1), \ldots, \pi(a_r)\big) \in \dot{R}^{\mathfrak{B}}.$$
- für jedes Konstantensymbol $\dot{c} \in \sigma$ gilt:
$$\pi(\dot{c}^{\mathfrak{A}}) \;=\; \dot{c}^{\mathfrak{B}}.$$
- für jedes Funktionssymbol $\dot{f} \in \sigma$, für $r := \text{ar}(\dot{f})$ und für alle r-Tupel $(a_1, \ldots, a_r) \in A^r$ gilt:
$$\pi\big(\dot{f}^{\mathfrak{A}}(a_1, \ldots, a_r)\big) \;=\; \dot{f}^{\mathfrak{B}}\big(\pi(a_1), \ldots, \pi(a_r)\big).$$

Isomorphismus

Eine solche Abbildung π wird **Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$** genannt.

Beispiel 6.11.

(a) Ist $A = \{1, 2, 3, 4\}, B = \{6, 7, 8, 9\}$, und sind $\dot{\leqslant}^{\mathfrak{A}}$ und $\dot{\leqslant}^{\mathfrak{B}}$ die natürlichen linearen Ordnungen auf A und B, so sind die beiden σ_{Ord}-Strukturen $\mathfrak{A} = (A, \dot{\leqslant}^{\mathfrak{A}})$ und $\mathfrak{B} = (B, \dot{\leqslant}^{\mathfrak{B}})$ isomorph.
Skizze:



Allgemein gilt: Sind A und B endliche Mengen mit $|A| = |B|$ und sind $\dot{\leqslant}^{\mathfrak{A}}$ und $\dot{\leqslant}^{\mathfrak{B}}$ lineare Ordnungen auf den Universen A und B, so ist $\mathfrak{A} \cong \mathfrak{B}$, und die Abbildung π, die das (bzgl. $\dot{\leqslant}^{\mathfrak{A}}$) kleinste Element in A auf das (bzgl. $\dot{\leqslant}^{\mathfrak{B}}$) kleinste Element von $\mathfrak{B}$ abbildet und, allgemein, für jedes $i \in \{1, \ldots, |A|\}$ das i-kleinste Element in A (bzgl. $\dot{\leqslant}^{\mathfrak{A}}$) auf das i-kleinste Element in B (bzgl. $\dot{\leqslant}^{\mathfrak{B}}$) abbildet, ein Isomorphismus von $\mathfrak{A}$ nach $\mathfrak{B}$.

(b) Sind $\dot{\leqslant}^{\mathcal{N}}$ und $\dot{\leqslant}^{\mathcal{Z}}$ die natürlichen linearen Ordnungen auf $\mathbb{N}$ und $\mathbb{Z}$, so sind die σ_{Ord}-Strukturen $\mathcal{N} := (\mathbb{N}, \dot{\leqslant}^{\mathcal{N}})$ und $\mathcal{Z} := (\mathbb{Z}, \dot{\leqslant}^{\mathcal{Z}})$ **nicht isomorph** (kurz: $\mathcal{N} \ncong \mathcal{Z}$).

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.

6.3 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 6.12 (Variablen und Alphabet der Logik erster Stufe).

Individuenvariable, Variable

(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. $\text{VAR} = \{v_i : i \in \mathbb{N}\}$.

Alphabet A_σ

(b) Sei σ eine Signatur. Das **Alphabet** A_σ der **Logik erster Stufe über** σ besteht aus:

- 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 6.13 (Terme der Logik erster Stufe).

σ-Terme

Sei σ eine Signatur. Die Menge T_σ der **σ-Terme** ist die folgendermaßen rekursiv definierte Teilmenge von A_σ^*:

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 6.14. Sei $\sigma = \{\dot{f}, \dot{c}\}$ die Signatur aus Beispiel 6.11(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 6.15 (Formeln der Logik erster Stufe).
Sei σ eine Signatur. Die Menge $\mathrm{FO}[\sigma]$ aller **Formeln der Logik erster Stufe über der Signatur** σ (kurz: **FO**$[\sigma]$**-Formeln**; FO steht für die englische Bezeichnung der Logik erster Stufe: first-order logic) ist die folgendermaßen rekursiv definierte Teilmenge von A_σ^*:

FO$[\sigma]$-Formeln

Basisregeln:

- Für alle σ-Terme t_1 und t_2 in T_σ gilt:
$$t_1 \doteq t_2 \ \in \ \mathrm{FO}[\sigma].$$
- Für jedes Relationssymbol $\dot{R} \in \sigma$, für $r := \mathrm{ar}(\dot{R})$ und für alle σ-Terme $t_1, \ldots, t_r$ in T_σ gilt:
$$\dot{R}(t_1, \ldots, t_r) \ \in \ \mathrm{FO}[\sigma].$$

Bemerkung: $\mathrm{FO}[\sigma]$-Formeln der Form $t_1 \doteq t_2$ oder $\dot{R}(t_1, \ldots, t_r)$ heißen **atomare σ-Formeln**.

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 $x \in \mathrm{VAR}$, so ist auch
  - $\exists x\, \varphi \ \in \ \mathrm{FO}[\sigma]$
  - $\forall x\, \varphi \ \in \ \mathrm{FO}[\sigma]$.

Beispiel 6.16.

(a) Sei $\sigma = \{\dot{f}, \dot{c}\}$ die Signatur aus Beispiel 6.11(c), die aus einem 2-stelligen Funktionssymbol $\dot{f}$ und einem Konstantensymbol $\dot{c}$ besteht.

Folgende Worte aus A_σ^* sind $\mathrm{FO}[\sigma]$-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** $\mathrm{FO}[\sigma]$-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 $\mathrm{FO}[\sigma_{\mathrm{Graph}}]$-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 4.67). 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 6.17.

- 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 $\dot{\leqslant} \in \sigma_{\mathrm{Ord}}$ 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 $\dot{\leqslant}(x, y)$ schreiben wir $x \dot{\leqslant} y$.

Wir wissen nun, welche Zeichenketten (über dem Alphabet A_σ) **FO**$[\sigma]$**-Formeln** genannt werden.

6.4 Semantik der Logik erster Stufe

Bevor wir die Semantik der Logik erster Stufe formal definieren, betrachten wir zunächst einige Beispiele, um ein intuitives Verständnis der Semantik der Logik erster Stufe zu erlangen.

6.4.1 Beispiele zur Semantik der Logik erster Stufe

Beispiel 6.18 (gerichtete Graphen).
Sei $\sigma_{\mathrm{Graph}} = \{\dot{E}\}$, wobei $\dot{E}$ ein 2-stelliges Relationssymbol ist.

(a) Die FO$[\sigma_{\mathrm{Graph}}]$-Formel

$$\varphi \quad := \quad \forall x \forall y \big(\dot{E}(x, y) \to \dot{E}(y, x) \big)$$

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:

$$\varphi(x, y) \quad := \quad \exists z_1 \exists z_2 \Big(\big(\dot{E}(x, z_1) \wedge \dot{E}(z_1, z_2) \big) \wedge \dot{E}(z_2, y) \Big).$$

(c) Die FO[σ_{Graph}]-Formel

$$\forall x \forall y \exists z_1 \exists z_2 \Big(\big(\dot{E}(x, z_1) \wedge \dot{E}(z_1, z_2) \big) \wedge \dot{E}(z_2, y) \Big)$$

sagt in einem Graphen $\mathfrak{A}$ aus, dass es zwischen je 2 Knoten einen Weg der Länge 3 gibt.

Beispiel 6.19 (Verwandtschaftsbeziehungen). Um Verwandtschaftsbeziehungen zu modellieren, können wir die Signatur σ benutzen, die aus den folgenden Symbolen besteht:

- 1-stellige Funktionssymbole $\dot{Vater}$, $\dot{Mutter}$

  (Bedeutung: $x \doteq \dot{Vater}(y)$ besagt "x ist der Vater von y".)

- 2-stellige Relationssymbole $\dot{Geschwister}$, $\dot{Vorfahr}$

  (Bedeutung: $\dot{Geschwister}(x, y)$ besagt, dass x und y Geschwister sind; $\dot{Vorfahr}(x, y)$ besagt, dass x ein Vorfahr von y ist.)

Generelles Wissen über Verwandtschaftsbeziehungen lässt sich durch Formeln der Logik erster Stufe repräsentieren, beispielsweise:

- "Personen mit gleichem Vater und gleicher Mutter sind Geschwister":

$$\forall x \forall y \Big(\big(\dot{Vater}(x) \doteq \dot{Vater}(y) \wedge \dot{Mutter}(x) \doteq \dot{Mutter}(y) \big) \rightarrow \dot{Geschwister}(x, y) \Big).$$

- "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:

$$\psi(x) \;:=\; \exists y_1 \exists y_2 \Big(\Big(\big(x \doteq \dot{Vater}(y_1) \wedge x \doteq \dot{Vater}(y_2) \big) \wedge \neg y_1 \doteq y_2 \Big) \wedge \forall z \big(x \doteq \dot{Vater}(z) \rightarrow (z \doteq y_1 \vee z \doteq y_2) \big) \Big).$$

6.4.2 Formale Definition der Semantik der Logik erster Stufe

Um die formale Definition der Semantik der Logik erster Stufe angeben zu können, benötigen wir noch folgende Begriffe:

Notation 6.20.

Teilformel

(a) 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

(b) Ist φ eine Formel und x eine Variable, so heißt jedes Vorkommen von x in einer Teilformel der Form $\exists x\, \psi$ oder $\forall x\, \psi$ **gebunden**. Jedes andere Vorkommen von x in φ heißt **frei**.
Beispiel:

$$\varphi \;:=\; \big(\dot{f}(v_0, \dot{c}) \doteq v_3 \wedge \exists v_0 \dot{f}(v_0, v_1) \doteq \dot{c}\big)$$

freies Vorkommen von v_0 in φ — freies Vorkommen von v_3 in φ — gebundene Vorkommen von v_0 in φ — freies Vorkommen von v_1 in φ

frei(φ)
freien Variablen

(c) Die Menge frei(φ) aller **freien Variablen** einer FO[σ]-Formel φ besteht aus allen Variablen, die mindestens einmal frei in φ vorkommen.
Beispiele:

- $\text{frei}\big(\dot{f}(v_0, \dot{c}) \doteq v_3\big) = \{v_0, v_3\}$
- $\text{frei}\big(\exists v_0\, \dot{f}(v_0, v_1) \doteq \dot{c}\big) = \{v_1\}$
- $\text{frei}\big(\dot{f}(v_0, \dot{c}) \doteq v_3 \wedge \exists v_0\, \dot{f}(v_0, v_1) \doteq \dot{c}\big) = \{v_0, v_3, v_1\}$

Satz

(d) Eine FO[σ]-Formel φ heißt **Satz** (genauer: FO[σ]-Satz), falls sie keine freien Variablen besitzt, d.h. falls $\text{frei}(\varphi) = \emptyset$.

Definition 6.21 (Belegung und Interpretation).

Belegung

(a) Eine **Belegung** in einer σ-Struktur $\mathfrak{A} = (A, \alpha)$ ist eine partielle Funktion β von VAR nach A (d.h. β ordnet jeder Variablen $x \in \text{Def}(\beta)$ ein Element $\beta(x)$ aus dem Universum von $\mathfrak{A}$ zu).

passend zu t

(b) Eine Belegung β ist eine **Belegung für einen σ-Term t** (bzw. **passend zu t**), wenn Def(β) alle in t vorkommenden Variablen enthält.

passend zu φ

(c) Eine Belegung β ist eine **Belegung für eine FO[σ]-Formel φ** (bzw. **passend zu φ**), wenn $\text{frei}(\varphi) \subseteq \text{Def}(\beta)$.

σ-Interpretation

(d) Eine **σ-Interpretation** ist ein Paar

$$\mathcal{I} \;=\; (\mathfrak{A}, \beta)$$

bestehend aus einer σ-Struktur $\mathfrak{A}$ und einer Belegung β in $\mathfrak{A}$.

$\mathcal{I} = (\mathfrak{A}, \beta)$ ist eine **Interpretation für einen σ-Term t** (bzw. **passend zu t**), wenn β passend zu t ist.

Interpretation für eine FO[σ]-Formel

$\mathcal{I} = (\mathfrak{A}, \beta)$ ist eine **Interpretation für eine FO[σ]-Formel φ** (bzw. **passend zu φ**), wenn β passend zu φ ist.

Definition 6.22 (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 $x \in \textsc{Var}$ 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 6.23. 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 6.11(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 6.24.

(a) Ist β eine Belegung in einer σ-Struktur $\mathfrak{A}$, ist $x \in \textsc{Var}$ und ist $a \in A$, so sei

$$\beta\tfrac{a}{x}$$

die Belegung mit $\mathrm{Def}(\beta\frac{a}{x}) := \mathrm{Def}(\beta) \cup \{x\}$, die für alle $y \in \mathrm{Def}(\beta\frac{a}{x})$ definiert ist durch

$$\beta\tfrac{a}{x}(y) := \begin{cases} a, & \text{falls } y = x \\ \beta(y) & \text{sonst.} \end{cases}$$

(b) Ist $\mathcal{I} = (\mathfrak{A}, \beta)$ eine σ-Interpretation, ist $x \in \textsc{Var}$ und ist $a \in A$, so sei

$$\mathcal{I}\tfrac{a}{x} := (\mathfrak{A}, \beta\tfrac{a}{x}).$$

Wir können nun (endlich) die formale Semantik der Logik erster Stufe festlegen.

Definition 6.25 (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:

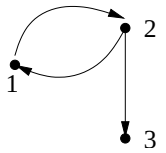
- Für alle σ-Terme t_1 und t_2 in T_σ gilt:
$$[\![t_1 \doteq t_2]\!]^{\mathcal{I}} \;:=\; \begin{cases} 1, & \text{falls } [\![t_1]\!]^{\mathcal{I}} = [\![t_2]\!]^{\mathcal{I}} \\ 0, & \text{sonst.} \end{cases}$$
- Für jedes Relationssymbol $\dot{R} \in \sigma$, für $r := \mathrm{ar}(\dot{R})$ und für alle σ-Terme $t_1, \ldots, t_r$ in T_σ gilt:
$$[\![\dot{R}(t_1, \ldots, t_r)]\!]^{\mathcal{I}} \;:=\; \begin{cases} 1, & \text{falls } ([\![t_1]\!]^{\mathcal{I}}, \ldots, [\![t_r]\!]^{\mathcal{I}}) \in \dot{R}^{\mathfrak{A}} \\ 0, & \text{sonst.} \end{cases}$$

Rekursionsschritt:

- Die Semantik der Junktoren $\neg$, $\wedge$, $\vee$, $\rightarrow, \leftrightarrow$ ist wie in der Aussagenlogik definiert – 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 $x \in \mathrm{VAR}$, so ist
  - $[\![\exists x\, \varphi]\!]^{\mathcal{I}} \;:=\; \begin{cases} 1, & \text{falls es (mindestens) ein } a \in A \text{ gibt, so dass } [\![\varphi]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\ 0, & \text{sonst.} \end{cases}$
  - $[\![\forall x\, \varphi]\!]^{\mathcal{I}} \;:=\; \begin{cases} 1, & \text{falls für alle } a \in A \text{ gilt: } [\![\varphi]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\ 0, & \text{sonst.} \end{cases}$

Beispiel 6.26. Sei $\sigma_{\mathrm{Graph}} = \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol $\dot{E}$ besteht. Betrachte die FO[σ_{Graph}]-Formel
$$\varphi \;:=\; \forall x \forall y \big(\dot{E}(x, y) \rightarrow \dot{E}(y, x)\big).$$
Für jede zu φ passende σ_{Graph}-Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ gilt:
$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{I}} \;=\; 1 \;&\Longleftrightarrow\; \text{für alle } a \in A \text{ gilt: } [\![\forall y \big(\dot{E}(x, y) \rightarrow \dot{E}(y, x)\big)]\!]^{\mathcal{I}\frac{a}{x}} = 1 \\
&\Longleftrightarrow\; \text{für alle } a \in A \text{ gilt:} \\
&\qquad\quad \text{für alle } b \in A \text{ gilt: } [\![\big(\dot{E}(x, y) \rightarrow \dot{E}(y, x)\big)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1 \\
&\Longleftrightarrow\; \text{für alle } a \in A \text{ und alle } b \in A \text{ gilt:} \\
&\qquad\quad \text{falls } [\![\dot{E}(x, y)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1, \text{ so auch } [\![\dot{E}(y, x)]\!]^{\mathcal{I}\frac{a}{x}\frac{b}{y}} = 1 \\
&\Longleftrightarrow\; \text{für alle } a \in A \text{ und alle } b \in A \text{ gilt:} \\
&\qquad\quad \text{falls } (a, b) \in \dot{E}^{\mathfrak{A}}, \text{ so auch } (b, a) \in \dot{E}^{\mathfrak{A}} \\
&\Longleftrightarrow\; \dot{E}^{\mathfrak{A}} \text{ ist symmetrisch, (vgl. Def. 4.67).}
\end{aligned}$$

Sei nun $\mathfrak{A}$ die σ_{Graph}-Struktur, die den gerichteten Graphen



repräsentiert, d.h. $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ mit $A = \{1, 2, 3\}$ und $\dot{E}^{\mathfrak{A}} = \{(1, 2), (2, 1), (2, 3)\}$. Sei β die Belegung mit leerem Definitionsbereich und sei $\mathcal{I} := (\mathfrak{A}, \beta)$. Dann gilt: Da in unserem konkreten Graphen $\mathfrak{A}$ für $a = 2$ und $b = 3$ gilt: $(a, b) \in \dot{E}^{\mathfrak{A}}$, aber $(b, a) \notin \dot{E}^{\mathfrak{A}}$, ist $\dot{E}^{\mathfrak{A}}$ nicht antisymmetrisch, und daher ist hier $[\![\varphi]\!]^{\mathcal{I}} = 0$.

Notation 6.27. Sei σ eine Signatur und sei φ eine FO[σ]-Formel.

(a) Sei $\mathcal{I} = (\mathfrak{A}, \beta)$ eine zu φ passende σ-Interpretation.

Wir sagen "$\mathcal{I}$ **erfüllt** φ" (bzw. "$\mathcal{I}$ **ist ein Modell von** φ", kurz: $\mathcal{I} \models \varphi$), falls $[\![\varphi]\!]^{\mathcal{I}} = 1$. $\mathcal{I} \models \varphi$

Wir sagen "$\mathcal{I}$ **erfüllt** φ **nicht**" (bzw. "$\mathcal{I}$ **ist kein Modell von** φ", kurz: $\mathcal{I} \not\models \varphi$), falls $[\![\varphi]\!]^{\mathcal{I}} = 0$. $\mathcal{I} \not\models \varphi$

(b) Ist φ ein **Satz** (d.h. φ hat keine freien Variablen), so hängt die Tatsache, ob φ von einer Interpretation $\mathcal{I} = (\mathfrak{A}, \beta)$ erfüllt wird, nur von der Struktur $\mathfrak{A}$ und nicht von der Belegung β ab. An Stelle von "$\mathcal{I} \models \varphi$" schreiben wir dann kurz "$\mathfrak{A} \models \varphi$" und sagen "die σ-Struktur $\mathfrak{A}$ erfüllt den Satz φ."

6.5 Erfüllbarkeit, Allgemeingültigkeit, Folgerung und Äquivalenz

Definition 6.28. Sei σ eine Signatur und sei φ eine FO[σ]-Formel.

(a) φ heißt **erfüllbar**, wenn es (mindestens) eine zu φ passende σ-Interpretation $\mathcal{I}$ gibt, die φ erfüllt. erfüllbar

(b) φ heißt **unerfüllbar**, wenn φ nicht erfüllbar ist. unerfüllbar

(c) φ heißt **allgemeingültig**, wenn jede zu φ passende σ-Interpretation φ erfüllt. allgemeingültig

Beispiel 6.29. 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:

$$\begin{aligned}
[\![\varphi]\!]^{\mathcal{I}} = 1 &\iff \text{Für jedes } b \in B \text{ gilt: } [\![\dot{E}(x,y)]\!]^{\mathcal{I}\frac{b}{y}} = 1 \\
&\iff \text{Für jedes } b \in B \text{ gilt: } \big(\beta\frac{b}{y}(x), \beta\frac{b}{y}(y)\big) \in \dot{E}^{\mathfrak{B}} \\
&\iff \text{Für jedes } b \in B \text{ gilt: } (1, b) \in \dot{E}^{\mathfrak{B}}.
\end{aligned}$$

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 6.30. 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. □

ψ folgt aus φ

Definition 6.31 (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:

$$\text{Falls} \quad \underbrace{\mathcal{I} \models \varphi}_{\text{d.h. } [\![\varphi]\!]^{\mathcal{I}} = 1} \quad , \quad \text{so auch} \quad \underbrace{\mathcal{I} \models \psi}_{\text{d.h. } [\![\psi]\!]^{\mathcal{I}} = 1}.$$

äquivalent

Definition 6.32 (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:

$$\mathcal{I} \models \varphi \iff \mathcal{I} \models \psi.$$

Beobachtung 6.33. 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. □

6.6 Grenzen der Logik erster Stufe

In Beispiel 6.18 und 6.19 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 6.34. Sei $\sigma_{\text{Graph}} := \{\dot{E}\}$ die Signatur, die aus einem 2-stelligen Relationssymbol $\dot{E}$ besteht. Es gilt:

(a) Es gibt keinen FO[σ_{Graph}]-Satz φ, so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ gilt:

$$\mathfrak{A} \text{ erfüllt } \varphi \iff \mathfrak{A} \text{ ist azyklisch (vgl. Definition 4.14).}$$

(b) Es gibt keinen FO[σ_{Graph}]-Satz φ', so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ gilt:

$$\mathfrak{A} \text{ erfüllt } \varphi' \iff \mathfrak{A} \text{ ist stark zusammenhängend (vgl. Definition 4.15).}$$

(c) Es gibt keine FO[σ_{Graph}]-Formel ψ mit freien Variablen x und y, so dass für jeden gerichteten Graphen $\mathfrak{A} = (A, \dot{E}^{\mathfrak{A}})$ und jede zu ψ passende Belegung β in $\mathfrak{A}$ gilt:

$$(\mathfrak{A}, \beta) \text{ erfüllt } \psi \iff \text{es gibt in } \mathfrak{A} \text{ einen Weg von Knoten } \beta(x) \text{ zu Knoten } \beta(y).$$

Einen Beweis dieses Satzes können Sie in der Vorlesung "Logik in der Informatik" kennenlernen.

6.7 Ein Anwendungsbereich der Logik erster Stufe: Datenbanken

Relationale Datenbanken bestehen aus **Tabellen**, die sich als Relationen auffassen lassen. Datenbanken lassen sich daher als **Strukturen** über einer passenden Signatur auffassen. Die in der Praxis gebräuchlichste Datenbankanfragesprache ist **SQL**. Der "Kern" von SQL basiert auf der Logik erster Stufe, die in der Datenbankterminologie oft auch "relationaler Kalkül" (engl.: "relational calculus") bezeichnet wird.

Zur Illustration von Anfragen verwenden wir eine kleine Datenbank mit Kinodaten, bestehend aus:

- einer Tabelle *Orte*, die Informationen über Kinos (Kino, Adresse, Telefonnummer) enthält,
- einer Tabelle *Filme*, die Informationen über Filme enthält (Titel, Regie, Schauspieler).
- eine Tabelle *Programm*, die Informationen zum aktuellen Kinoprogramm enthält (Kino, Titel, Zeit).

Orte-Tabelle:

Kino	Adresse	Telefon
Babylon	Dresdner Str. 2	61609693
Casablanca	Friedenstr. 12	6775752
Cinestar Cubix Alexanderplatz	Rathausstr. 1	2576110
Die Kurbel	Giesebrechtstr. 4	88915998
Filmpalast Berlin	Kurfürstendamm 225	8838551
International	Karl-Marx-Allee 33	24756011
Kino in der Kulturbrauerei	Schönhauser Allee 36	44354422
Moviemento	Kottbusser Damm 22	6924785

Filme-Tabelle:

Titel	Regie	Schauspieler
Capote	Bennet Miller	Philip Seymour Hoffman
Capote	Bennet Miller	Catherine Keener
Das Leben der Anderen	F. Henkel von Donnersmarck	Martina Gedeck
Das Leben der Anderen	F. Henkel von Donnersmarck	Ulrich Tukur
Der ewige Gärtner	Fernando Meirelles	Ralph Fiennes
Der ewige Gärtner	Fernando Meirelles	Rachel Weisz
Good Night and Good Luck	George Clooney	David Strathairn
Good Night and Good Luck	George Clooney	Patricia Clarkson
Knallhart	Detlev Buck	Jenny Elvers
Knallhart	Detlev Buck	Jan Henrik Stahlberg
Raumpatrouille Orion – Rücksturz ins Kino	Michael Braun	Dietmar Schönherr
Raumpatrouille Orion – Rücksturz ins Kino	Theo Mezger	Dietmar Schönherr
Raumpatrouille Orion – Rücksturz ins Kino	Michael Braun	Eva Pflug
Raumpatrouille Orion – Rücksturz ins Kino	Theo Mezger	Eva Pflug
Raumpatrouille Orion – Rücksturz ins Kino	Michael Braun	Wolfgang Völz
Raumpatrouille Orion – Rücksturz ins Kino	Theo Mezger	Wolfgang Völz
Requiem	Hans-Christian Schmid	Sandra Hüller
Sommer vorm Balkon	Andreas Dresen	Nadja Uhl
Sommer vorm Balkon	Andreas Dresen	Inka Friedrich
Sommer vorm Balkon	Andreas Dresen	Andreas Schmidt
Syriana	Stephen Gaghan	George Clooney
Syriana	Stephen Gaghan	Matt Damon
V wie Vendetta	James McTeigue	Natalie Portman
Walk the Line	James Mangold	Joaquin Phoenix
Walk the Line	James Mangold	Reese Witherspoon

Programm-Tabelle:

Kino	Titel	Zeit
Babylon	Capote	17:00
Babylon	Capote	19:30
Kino in der Kulturbrauerei	Capote	17:30
Kino in der Kulturbrauerei	Capote	20:15
International	Das Leben der Anderen	14:30
International	Das Leben der Anderen	17:30
International	Das Leben der Anderen	20:30
Filmpalast Berlin	Good Night and Good Luck	15:30
Filmpalast Berlin	Good Night and Good Luck	17:45
Filmpalast Berlin	Good Night and Good Luck	20:00
Kino in der Kulturbrauerei	Good Night and Good Luck	18:00
Kino in der Kulturbrauerei	Good Night and Good Luck	20:00
Kino in der Kulturbrauerei	Good Night and Good Luck	22:45
Babylon	Sommer vorm Balkon	21:45
Kino in der Kulturbrauerei	Sommer vorm Balkon	21:45
Filmmuseum Potsdam	Raumpatrouille Orion – Rücksturz ins Kino	22:00

Für eine geeignete Signatur σ_{Kino} können wir diese Datenbank durch eine σ_{Kino}-Struktur $\mathfrak{A}_{\text{Kino}}$ folgendermaßen modellieren: Die Signatur σ_{Kino} besteht aus:

- einem 3-stelligen Relationssymbol $\dot{Orte}$
- einem 3-stelligen Relationssymbol $\dot{Filme}$
- einem 3-stelligen Relationssymbol $\dot{Programm}$
- Konstantensymbolen '$\dot{c}$', die allen potentiellen Einträgen c der Datenbank entsprechen, also '$\dot{Babylon}$', '$\dot{Casablanca}$', ..., '$\dot{Capote}$', '$\dot{Das}$ $\dot{Leben}$ $\dot{der}$ $\dot{Anderen}$', ... usw., aber auch z.B. '$\dot{Stephen}$ $\dot{Spielberg}$' oder '$\dot{Lola}$ $\dot{rennt}$'. D.h.: Für jedes Wort c über dem ASCII-Alphabet gibt es ein Konstantensymbol '$\dot{c}$'.

Die σ_{Kino}-Struktur $\mathfrak{A}_{\text{Kino}}$ hat als Universum die Menge aller Worte über dem ASCII-Alphabet, d.h.

$$A_{\text{Kino}} := \text{ASCII}^*,$$

die 3-stelligen Relationen

$$\begin{aligned}
\dot{Orte}^{\mathfrak{A}_{\text{Kino}}} := \{&(\text{Babylon}, \text{Dresdner Str. } 2, 61609693),\\
&(\text{Casablanca}, \text{Friedenstr. } 12, 6775752),\\
&\ldots,\\
&(\text{Moviemento}, \text{Kottbusser Damm } 22, 6924785)\},\\
\dot{Filme}^{\mathfrak{A}_{\text{Kino}}} := \{&(\text{Capote}, \text{Bennet Miller}, \text{Philip Seymour Hoffman}),\\
&(\text{Capote}, \text{Bennet Miller}, \text{Catherine Keener}),\\
&\ldots,\\
&(\text{Walk the Line}, \text{James Mangold}, \text{Reese Witherspoon})\},\\
\dot{Programm}^{\mathfrak{A}_{\text{Kino}}} := \{&(\text{Babylon}, \text{Capote}, 17{:}00),\\
&(\text{Babylon}, \text{Capote}, 19{:}30),\\
&(\text{Kino in der Kulturbrauerei}, \text{Capote}, 17{:}30),\\
&\ldots\}
\end{aligned}$$

sowie für jedes in σ_{Kino} vorkommende Konstantensymbol '$\dot{c}$' die Konstante '$\dot{c}$'$^{\mathfrak{A}_{\text{Kino}}} := c$.
Zum Beispiel:

$$\begin{aligned}
&'\dot{Babylon}'^{\mathfrak{A}_{\text{Kino}}} = \text{Babylon},\\
&'\dot{Capote}'^{\mathfrak{A}_{\text{Kino}}} = \text{Capote},\\
&'\dot{George}\ \dot{Clooney}'^{\mathfrak{A}_{\text{Kino}}} = \text{George Clooney}.
\end{aligned}$$

Anfragen an die Kinodatenbank lassen sich auf unterschiedliche Art formulieren:

Beispiel 6.35. Eine Anfrage an unsere Kinodatenbank:

> "Gib die Titel aller Filme aus, die um 17:30 Uhr laufen."

In der Datenbankanfragesprache SQL lässt sich dies folgendermaßen formulieren:

```
SELECT  Titel
FROM    Programm
WHERE   Zeit = '17:30'
```

Dieselbe Anfrage lässt sich auch durch die folgende Formel der Logik erster Stufe beschreiben:

$$\varphi_{\text{Filme um 17:30 Uhr}}(x_{\mathrm{T}}) \;:=\; \exists x_{\mathrm{K}}\, \dot{Programm}(x_{\mathrm{K}}, x_{\mathrm{T}}, \dot{'17:30'}).$$

Notation 6.36. Sei σ eine Signatur und seien $x_1, \ldots, x_n$ Variablen.

- Die Notation $\varphi(x_1, \ldots, x_n)$ deutet an, dass φ eine FO[σ]-Formel mit $\mathrm{frei}(\varphi) = \{x_1, \ldots, x_n\}$ ist, d.h. dass $x_1, \ldots, x_n$ diejenigen Variablen sind, die in φ frei vorkommen.
- Ist $\varphi(x_1, \ldots, x_n)$ eine FO[σ]-Formel, ist $\mathfrak{A}$ eine σ-Struktur und sind $a_1, \ldots, a_n \in A$ Elemente im Universum von $\mathfrak{A}$, so schreiben wir

  $$\mathfrak{A} \;\models\; \varphi[a_1, \ldots, a_n],$$

  um auszudrücken, dass für die Belegung $\beta\colon \{x_1, \ldots, x_n\} \to A$ mit $\beta(x_1) = a_1, \ldots, \beta(x_n) = a_n$ gilt:

  $$(\mathfrak{A}, \beta) \;\models\; \varphi.$$

Definition 6.37. Sei σ eine Signatur, $\varphi(x_1, \ldots, x_n)$ eine FO[σ]-Formel und $\mathfrak{A}$ eine σ-Signatur. Die von φ in $\mathfrak{A}$ definierte n-stellige Relation ist

$$\varphi(\mathfrak{A}) \;:=\; \{(a_1, \ldots, a_n) \in A^n : \mathfrak{A} \models \varphi[a_1, \ldots, a_n]\}.$$

Beispiel 6.38. Die FO[σ_{Kino}]-Formel $\varphi_{\text{Filme um 17:30}}(x_{\mathrm{T}})$ aus Beispiel 6.35 definiert in unserer Kinodatenbank $\mathfrak{A}_{\text{Kino}}$ die 1-stellige Relation:

$$\varphi_{\text{Filme um 17:30}}(\mathfrak{A}_{\text{Kino}}) \;=\; \{\ (\text{Capote}),\ (\text{Das Leben der Anderen})\ \}.$$

Darstellung als Tabelle:

Filme um 17:30 Uhr:

Titel
Capote
Das Leben der Anderen

Beispiel 6.39. 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:

$$\varphi_{\text{Kinos mit George Clooney}}(x_{\mathrm{K}}, x_{\mathrm{A}}) \;:=\; \exists x_{\mathrm{Tel}} \exists x_{\mathrm{T}} \exists x_{\mathrm{Z}} \Big(\big(\dot{Orte}(x_{\mathrm{K}}, x_{\mathrm{A}}, x_{\mathrm{Tel}}) \wedge \dot{Programm}(x_{\mathrm{K}}, x_{\mathrm{T}}, x_{\mathrm{Z}}) \big) \wedge \big(\exists x_{\mathrm{R}}\, \dot{Filme}(x_{\mathrm{T}}, x_{\mathrm{R}}, \text{'George } \dot{\text{Clooney'}}) \vee \exists x_{\mathrm{S}}\, \dot{Filme}(x_{\mathrm{T}}, \text{'George } \dot{\text{Clooney'}}, x_{\mathrm{S}}) \big) \Big)$$

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.

6.8 Literaturhinweise

[27] Kapitel 2.1

[17] Kapitel 4.A

[15] Kapitel 4.2

Vorsicht: Jedes dieser Bücher verwendet unterschiedliche Notationen, die wiederum etwas von den in der Vorlesungen eingeführten Notationen abweichen.

6.9 Übungsaufgaben zu Kapitel 6

Aufgabe 6.1. 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 6.2. 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 $i \geqslant 4$.

(a) Berechnen Sie $[\![t_1]\!]^{\mathcal{I}}$ und $[\![t_2]\!]^{\mathcal{I}}$ für

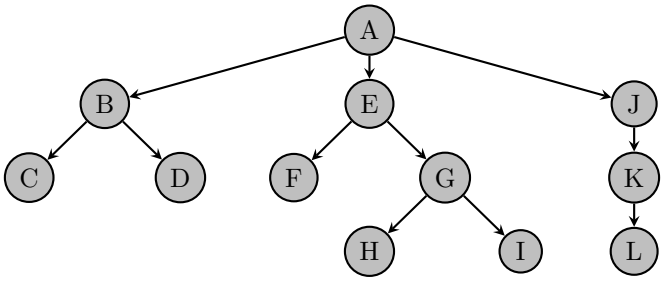
- $t_1 \ :=\ \dot{f}(\dot{f}(v_1, v_5), \dot{c})$
- $t_2 \ :=\ \dot{f}\big(\dot{f}(\dot{c}, \dot{f}(v_2, \dot{c})), v_1\big)$

(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 6.3. 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 $\{\dot{Elternknoten}\}$ modelliert werden kann. Geben Sie die entsprechende Struktur für den folgenden Baum an:



(b) Geben Sie je eine Formel $\varphi(x)$ der Logik erster Stufe an, die ausdrückt, dass der Knoten x

(i) ein Blatt ist,

(ii) die Wurzel ist,

(iii) genau zwei Kinder hat.

Aufgabe 6.4. 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 6.5 (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 6.6 (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 6.7 (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:

$$\begin{aligned}
\varphi_1(x_{\text{K}}) &= \exists x_{\text{Z}}\, \dot{Programm}(x_{\text{K}}, \text{`}\dot{Capote}\text{'}, x_{\text{Z}}) \\
\varphi_2(x_{\text{S}}) &= \exists x_{\text{T}}\big(\, \exists x_{\text{R}}\, \dot{Filme}(x_{\text{T}}, x_{\text{R}}, x_{\text{S}}) \;\wedge\; \exists x_{\text{K}} \exists x_{\text{Z}}\, \dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \,\big) \\
\varphi_3(x_{\text{T}}) &= \exists x_{\text{K}} \exists x_{\text{Z}} \Big(\dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \;\wedge \\
&\qquad\qquad \forall y_{\text{K}} \forall y_{\text{Z}} \big(\dot{Programm}(y_{\text{K}}, x_{\text{T}}, y_{\text{Z}}) \rightarrow y_{\text{Z}} \doteq x_{\text{Z}} \big) \Big) \\
\varphi_4(x_{\text{T}}, x_{\text{K}}, x_{\text{A}}) &= \Big(\exists x_{\text{Z}}\, \exists x_{\text{Tel}} \big(\dot{Programm}(x_{\text{K}}, x_{\text{T}}, x_{\text{Z}}) \wedge \dot{Orte}(x_{\text{K}}, x_{\text{A}}, x_{\text{Tel}}) \big) \;\wedge \\
&\qquad\qquad \exists x_{\text{S}}\, \dot{Filme}(x_{\text{T}}, \text{`}\dot{George}\;\dot{Clooney}\text{'}, x_{\text{S}}) \Big)
\end{aligned}$$

(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.

7 Endliche Automaten zur 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 (z.B. einen Geldautomaten bei der Bank) oder auf die Benutzungsoberflächen von Software-Systemen,
- Schaltfolgen von Ampelanlagen,
- Abläufe von Geschäftsprozessen in Firmen oder
- die Steuerung von Produktionsanlagen.

Solche Abläufe werden modelliert, indem man die Zustände angibt, die ein System annehmen kann, und beschreibt, unter welchen Bedingungen es aus einem Zustand in einen anderen übergehen kann (vgl. das Murmel-Problem aus Beispiel 1.1).

In diesem Kapitel werden die **endlichen Automaten** (bzw. **Transitionssysteme**) als ein grundlegender Kalkül vorgestellt, die sich gut zur Modellierung sequentieller Abläufe eignet und u.a. auch zur Spezifikation von realen oder abstrakten Maschinen genutzt werden kann.
Endliche Automaten

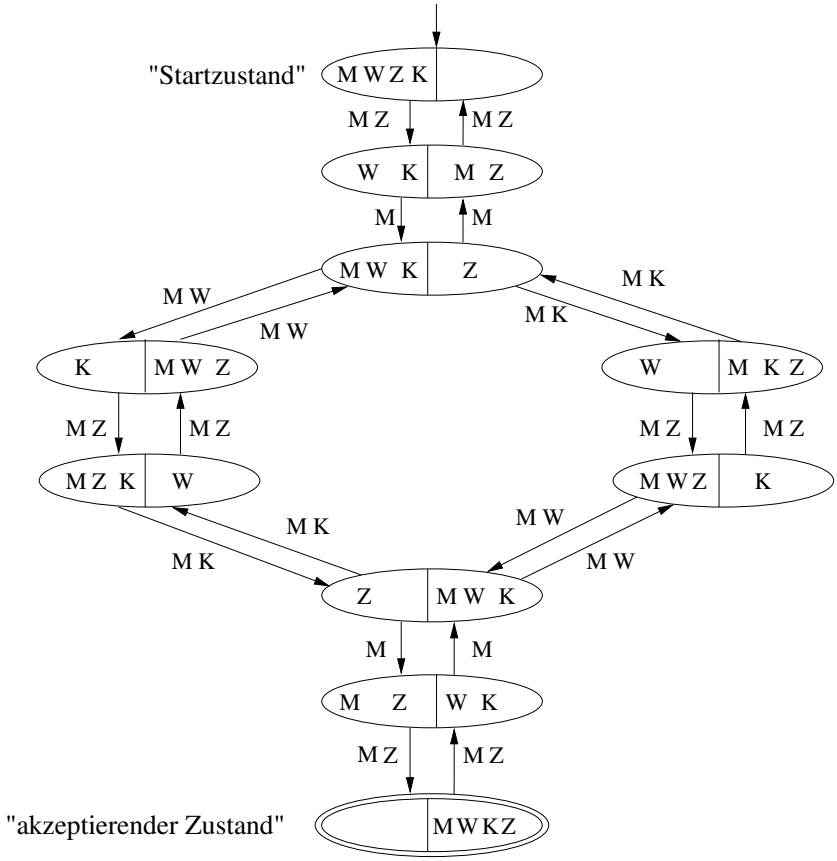
- reagieren auf äußere Ereignisse,
- ändern ggf. ihren „inneren Zustand“ und
- produzieren ggf. eine Ausgabe.

Sie werden z.B. eingesetzt, um

- das Verhalten realer Maschinen zu spezifizieren
  (z.B. ein Geldautomat oder ein Getränkeautomat),
- das Verhalten von Software-Komponenten zu beschreiben
  (z.B. 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.
  (Bei der „Flussüberquerung“ aus Beispiel 1.2 sind das genau diejenigen Folgen von „Flussüberquerungsschritten“, mit denen man vom „Startzustand“ $(\{M, W, Z, K\}, \emptyset)$ zum „Zielzustand“ $(\emptyset, \{M, W, Z, K\})$ gelangen kann).

Vor der formalen Definitionen endlicher Automaten betrachten wir zunächst zwei einführende Beispiele:

Beispiel 7.1.
Graphische Darstellung eines endlichen Automaten zum Flussüberquerungs-Problem aus Beispiel 1.2:



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 den in Abbildung 7.1 dargestellten endlichen Automaten modelliert werden. Dieser endliche Automat „akzeptiert“ genau diejenigen Folgen von Bedienoperationen, die vom Grundzustand aus wieder in den Grundzustand führen.

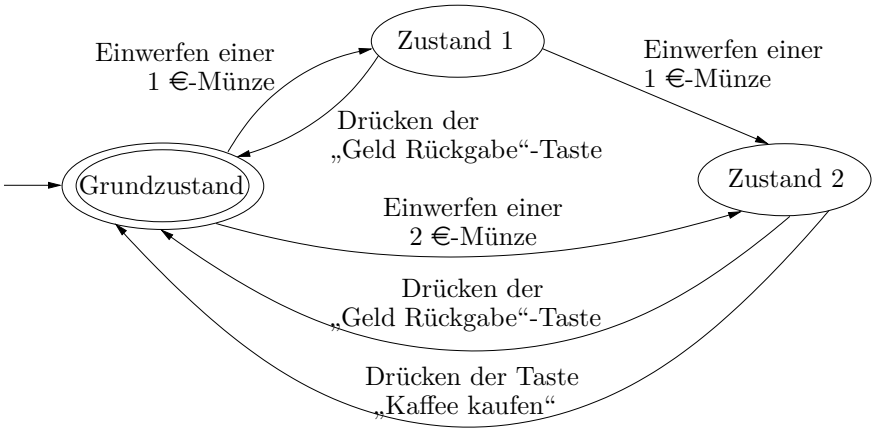


Abbildung 7.1: Graphische Darstellung eines endlichen Automaten, der einen Getränkeautomaten modelliert

7.1 Deterministische endliche Automaten

Definition 7.3 (DFA). Ein **deterministischer endlicher Automat**[1] (kurz: **DFA**) DFA

$$A \;=\; (\Sigma,\, Q,\, \delta,\, q_0,\, F)$$

besteht aus:

- einer endlichen Menge Σ, dem so genannten **Eingabealphabet**, Eingabealphabet
- einer endlichen Mengen Q, der so genannten **Zustandsmenge** (die Elemente aus Q werden **Zustände** genannt), Zustandsmenge
- einer partiellen Funktion δ von $Q \times \Sigma$ nach Q, der so genannten (Zustands-)**Übergangsfunktion** (oder **Überführungsfunktion**), Übergangsfunktion
- einem Zustand $q_0 \in Q$, dem so genannten **Startzustand**, Startzustand
- einer Menge $F \subseteq Q$, der so genannten Menge der **Endzustände** bzw. **akzeptierenden Zustände** (der Buchstabe F steht für „final states“, also „Endzustände“). Endzustände

Graphische Darstellung endlicher Automaten:

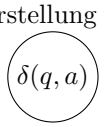
Endliche Automaten lassen sich anschaulich durch beschriftete Graphen darstellen (vgl. Beispiel 7.1 und 7.2):

- 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.:

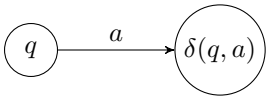
[1] engl.: deterministic finite automaton, kurz: DFA

- 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

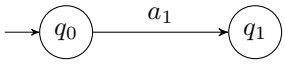
- Σ = {Einwerfen einer 1 €-Münze, Einwerfen einer 2 €-Münze, Drücken der „Geld Rückgabe“-Taste, Drücken der Taste „Kaffee kaufen“}
- Q = {Grundzustand, Zustand 1, Zustand 2}
- q_0 = Grundzustand
- F = {Grundzustand}
- δ ist die partielle Funktion von $Q \times \Sigma$ nach Q mit

$$\begin{aligned}
\delta(\text{Grundzustand}, \text{Einwerfen einer 1 €-Münze}) &= \text{Zustand 1},\\
\delta(\text{Grundzustand}, \text{Einwerfen einer 2 €-Münze}) &= \text{Zustand 2},\\
\delta(\text{Zustand 1}, \text{Einwerfen einer 1 €-Münze}) &= \text{Zustand 2},\\
\delta(\text{Zustand 1}, \text{Drücken der „Geld Rückgabe“-Taste}) &= \text{Grundzustand},\\
\delta(\text{Zustand 2}, \text{Drücken der „Geld Rückgabe“-Taste}) &= \text{Grundzustand},\\
\delta(\text{Zustand 2}, \text{Drücken der Taste „Kaffee kaufen“}) &= \text{Grundzustand}.
\end{aligned}$$

Die Verarbeitung eines Eingabeworts durch einen DFA:

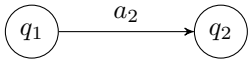
Ein DFA $A = (\Sigma, Q, \delta, q_0, F)$ erhält als Eingabe ein Wort $w \in \Sigma^*$, das eine Folge von „Aktionen“ oder „Bedienoperationen“ repräsentiert. Bei Eingabe eines Worts w wird A im Startzustand q_0 gestartet. Falls w das leere Wort ist, d.h. $w = \varepsilon$, so passiert nichts weiter. Falls w ein Wort von der Form $a_1 \cdots a_n$ mit $n \in \mathbb{N}_{>0}$ und $a_1, \ldots, a_n \in \Sigma$, so geschieht bei der Verarbeitung von w durch A folgendes: Durch Lesen der ersten Buchstabens von w, also a_1, geht der Automat in den Zustand $q_1 := \delta(q_0, a_1)$ über. In der graphischen Darstellung von A wird der Zustand

durch die mit a_1 beschriftete Kante verlassen, und q_1 ist der Endknoten dieser Kante, d.h.



Dies ist allerdings nur möglich, wenn (q_0, a_1) im Definitionsbereich von δ liegt (also $(q_0, a_1) \in \mathrm{Def}(\delta)$) — 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 der Automat landet in Zustand q_2, wobei $q_2 := \delta(q_1, q_2)$ der Endknoten dieser Kante ist.

Auf diese Weise wird nach und nach das gesamte Eingabewort $w = a_1 \cdots a_n$ abgearbeitet. Ausgehend vom Startzustand q_0 werden dabei nacheinander Zustände $q_1, \ldots, q_n$ erreicht. In der graphischen Darstellung von A entspricht dies gerade dem Durchlaufen eines Weges der Länge n, der im Knoten

startet und dessen Kanten mit den Buchstaben $a_1, \ldots, a_n$ beschriftet sind. Der Knoten

der am Ende dieses Weges erreicht wird (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" und schreiben kurz $\widehat{\delta}(q_0, w) = \bot$.

Definition 7.5 (Die erweiterte Übergangsfunktion $\widehat{\delta}$ eines DFA).
Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein DFA. Sei $\bot$ ein Symbol, das nicht in der Zustandsmenge Q liegt, und sei $Q_\bot := Q \cup \{\bot\}$. Die Funktion

$$\widehat{\delta} \ : \ Q_\bot \times \Sigma^* \ \to \ Q_\bot$$

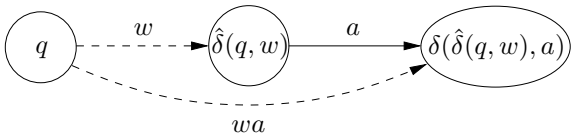
ist rekursiv wie folgt definiert:

- F.a. $w \in \Sigma^*$ ist $\widehat{\delta}(\bot, w) \ := \ \bot$.
- F.a. $q \in Q$ ist $\widehat{\delta}(q, \varepsilon) \ := \ q$.

- F.a. $q \in Q$, $w \in \Sigma^*$ und $a \in \Sigma$ gilt für $q' := \widehat{\delta}(q, w)$:

$$\widehat{\delta}(q, wa) \;:=\; \begin{cases} \bot & \text{falls } (q', a) \notin \mathrm{Def}(\delta), \\ \delta(q', a) & \text{falls } (q', a) \in \mathrm{Def}(\delta). \end{cases}$$

  Graphische Darstellung:



Insgesamt gilt: Falls $\widehat{\delta}(q_0, w) \neq \bot$, so ist $\hat{\delta}(q_0, w)$ der Zustand, der durch Verarbeiten des Worts w erreicht wird. Falls $\widehat{\delta}(q_0, w) = \bot$, so stürzt der Automat beim Verarbeiten des Worts w ab.

Die von einem DFA akzeptierte Sprache:

Das Eingabewort w wird vom DFA A **akzeptiert**, falls

$$\widehat{\delta}(q_0, w) \in F,$$

d.h., A stürzt bei Eingabe von w nicht ab, und der durch Verarbeiten des Worts w erreichte Zustand gehört zur Menge F der akzeptierenden Zustände.

In der graphischen Darstellung von A heißt das für ein Eingabewort $w = a_1 \cdots a_n$, dass es einen in

startenden Weg der Länge n gibt, dessen Kanten mit den Symbolen $a_1, \ldots, a_n$ beschriftet sind, und der in einem akzeptierenden Zustand  endet.

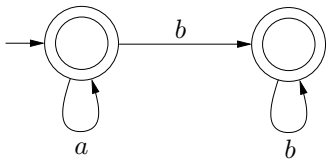
Definition 7.6 (Die von einem DFA A akzeptierte Sprache $L(A)$).
Die von einem DFA $A = (\Sigma, Q, \delta, q_0, F)$ akzeptierte Sprache $L(A)$ ist

$$L(A) \;:=\; \{\, w \in \Sigma^* \;:\; \hat{\delta}(q_0, w) \in F \,\}.$$

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:

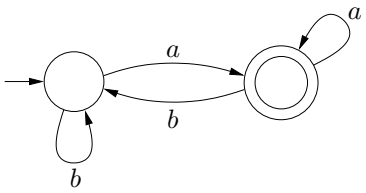
$$L(A_1) \;=\; \{a^n b^m : n \in \mathbb{N},\, m \in \mathbb{N}\}$$

(**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

$$L(A_2) \;=\; \{w \in \{a,b\}^* : \text{der letzte Buchstabe von } w \text{ ist ein } a\}$$

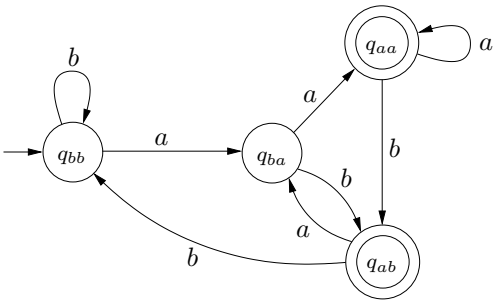
ist



(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

  Beispiele: Die DFAs A_2 und A_3 aus Beispiel 7.7 sind vollständig; der DFA A_1 nicht. Die DFAs aus Beispiel 7.1 und 7.2 sind nicht vollständig.

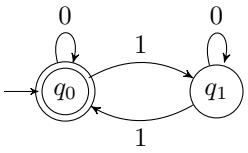
Für die graphische Darstellung eines DFAs gilt: Der DFA ist genau dann vollständig, wenn für jeden Zustand q gilt: Für jedes Symbol $a \in \Sigma$ gibt es genau eine aus  herausführende Kante, die mit a beschriftet ist.

Beachte: In manchen Büchern weicht die Definition von DFAs von Definition 7.3 ab, indem gefordert wird, dass DFAs grundsätzlich vollständig sein müssen.

Ein Anwendungsbeispiel: Paritätscheck durch einen DFA

Bei der Speicherung von Daten auf einem Speichermedium eines Computers werden Informationen durch Worte über dem Alphabet $\{0,1\}$ kodiert. Um eventuelle Fehler beim Übertragen der Daten erkennen zu können, wird der Kodierung oft ein so genanntes **Paritätsbit** angehängt, das bewirkt, dass die Summe der Einsen im resultierenden Wort gerade ist.

Für ein beliebiges Wort $w \in \{0,1\}^*$ sagen wir "w besteht den Paritätscheck", falls die Anzahl der Einsen in w gerade ist. Zur effizienten Durchführung eines Paritätschecks für ein gegebenes Wort $w \in \{0,1\}^*$ kann man den folgenden DFA A benutzen:



Für diesen Automaten gilt: $L(A) = \{w \in \{0,1\}^* : w \text{ besteht den Paritätscheck}\}$.

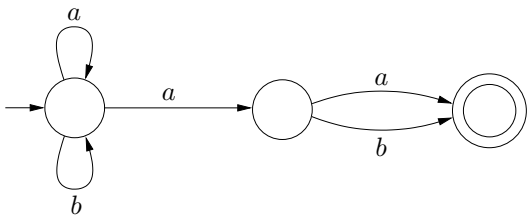
7.2 Nichtdeterministische endliche Automaten

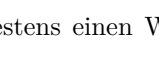
Für manche Modellierungsaufgaben ist die Forderung, dass es für jeden Zustand q und jedes Eingabesymbol a höchstens einen Nachfolgezustand $\delta(q,a)$ gibt, zu restriktiv, da man in manchen Zuständen für den Übergang mit einem Symbol a mehrere Möglichkeiten angeben will, ohne festzulegen, welche davon gewählt wird. Solche Entscheidungsfreiheiten in der Modellierung von Abläufen bezeichnet man als **nichtdeterministisch**. Nichtdeterministische Modelle sind oft einfacher aufzustellen und leichter zu verstehen als deterministische.

Beispiel 7.9. In Beispiel 7.7(c) haben wir einen (recht komplizierten) DFA A_3 mit

$$L(A_3) \;=\; \{w \in \{a,b\}^* : \text{der } \textbf{vorletzte} \text{ Buchstabe von } w \text{ ist ein } a\}$$

kennengelernt. Dieselbe Sprache wird auch von dem deutlich einfacheren **nichtdeterministischen endlichen Automaten** (kurz: NFA) A_4 mit der folgenden graphischen Darstellung akzeptiert:



Ein Eingabewort $w \in \{a,b\}^*$ wird von dem NFA A_4 genau dann akzeptiert, wenn es in der graphischen Darstellung mindestens einen Weg gibt, der im Startzustand  beginnt, dessen Kanten mit w beschriftet sind, und der im akzeptierenden Zustand  endet.

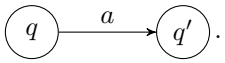
Die präzise Definition von NFAs wird in der folgenden Definition gegeben.

Definition 7.10 (NFA). Ein **nichtdeterministischer endlicher Automat**[2] (kurz: **NFA**) $A = (\Sigma, Q, \delta, q_0, F)$ besteht aus: NFA

- einer endlichen Menge Σ, dem so genannten **Eingabealphabet**, Eingabealphabet
- einer endlichen Menge Q, der so genannten **Zustandsmenge**, Zustandsmenge
- einer Funktion[3] $\delta\colon Q \times \Sigma \to \mathcal{P}(Q)$, der so genannten **Übergangsfunktion**, die jedem Zustand $q \in Q$ und jedem Symbol $a \in \Sigma$ eine **Menge $\delta(q,a)$ von möglichen Nachfolgezuständen** zuordnet (beachte: möglicherweise ist $\delta(q,a) = \emptyset$ — dann „stürzt" der Automat ab, wenn er im Zustand q ist und das Symbol a liest), Übergangsfunktion
- einem Zustand $q_0 \in Q$, dem so genannten **Startzustand**, Startzustand
- einer Menge $F \subseteq Q$, der so genannten Menge der **Endzustände** bzw. **akzeptierenden Zustände**. Endzustände

Graphische Darstellung von NFAs:

Die graphische Darstellung von NFAs erfolgt analog der graphischen Darstellung von DFAs. Ist $q \in Q$ ein Zustand und ist $a \in \Sigma$ ein Eingabesymbol, so gibt es **für jeden Zustand $q' \in \delta(q,a)$** in der graphischen Darstellung des NFAs einen mit dem Symbol a beschrifteten Pfeil von Knoten  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.

[2] engl.: non-deterministic finite automaton, kurz: NFA
[3] Zur Erinnerung: $\mathcal{P}(Q)$ bezeichnet die **Potenzmenge** von Q.

(b) Die von A akzeptierte Sprache $L(A)$ ist

$$L(A) \;:=\; \{w \in \Sigma^* : A \text{ akzeptiert } w\}.$$

Ähnlich wie bei DFAs können wir auch für NFAs eine erweiterte Überführungsfunktion $\widehat{\delta}$ definieren. Für einen Zustand q und ein Eingabewort w gibt $\widehat{\delta}(q, w)$ die **Menge** aller Zustände an, die durch Verarbeiten des Worts w erreicht werden können, wenn der Automat im Zustand q beginnt. Dies wird durch die folgende Definition präzisiert.

Definition 7.12 (Die erweiterte Übergangsfunktion $\widehat{\delta}$ eines NFA).
Sei $A := (\Sigma, Q, \delta, q_0, F)$ ein NFA. Die Funktion

$$\widehat{\delta} \;:\; Q \times \Sigma^* \;\to\; \mathcal{P}(Q)$$

ist rekursiv wie folgt definiert:

- F.a. $q \in Q$ ist $\widehat{\delta}(q, \varepsilon) \;:=\; \{q\}$.
- F.a. $q \in Q$, $w \in \Sigma^*$ und $a \in \Sigma$ ist $\widehat{\delta}(q, wa) \;:=\; \bigcup_{q' \in \widehat{\delta}(q,w)} \delta(q', a)$.

Man beachte, dass ein Wort w genau dann von einem NFA $A := (\Sigma, Q, \delta, q_0, F)$ akzeptiert wird, wenn gilt:

$$\widehat{\delta}(q_0, w) \cap F \;\neq\; \emptyset.$$

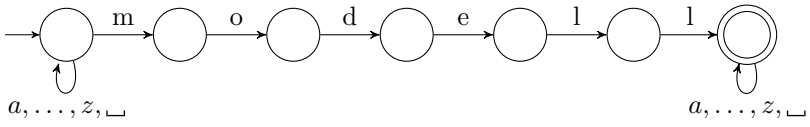
Somit ist

$$L(A) \;=\; \{\, w \in \Sigma^* \;:\; \widehat{\delta}(q_0, w) \cap F \;\neq\; \emptyset \,\}.$$

Ein Anwendungsbeispiel: Stichwort-Suche in Texten

Gegeben:	Ein Stichwort, z.B. „modell“
Eingabe:	Ein Text, der aus den Buchstaben “a” bis “z” sowie dem Leerzeichen “$\llcorner\!\lrcorner$” besteht
Frage:	Kommt das Stichwort „modell“ irgendwo im Eingabetext vor? Der Eingabetext soll genau dann akzeptiert werden, wenn er das Stichwort „modell“ enthält.

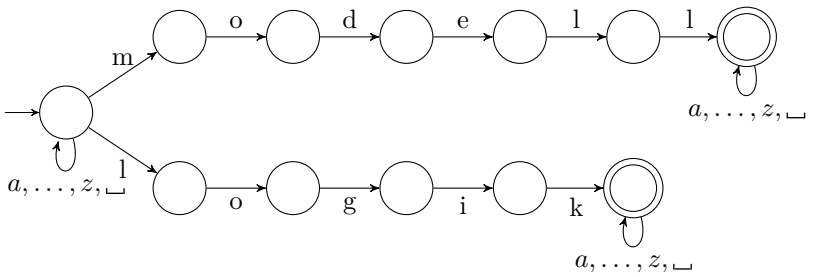
Graphische Darstellung eines NFAs, der dies bewerkstelligt:



Auf ähnliche Art können auch Varianten dieser Stichwortsuche behandelt werden, zum Beispiel die Frage

Kommt mindestens eins der Stichworte „modell“ bzw. „logik“ im Eingabetext vor?

Graphische Darstellung eines NFAs, der dies bewerkstelligt:



7.3 Äquivalenz von NFAs und DFAs

Nachdem wir mit NFAs und DFAs zwei Sorten von endlichen Automaten kennen gelernt haben, drängt sich natürlich die Frage auf, ob NFAs wirklich "mehr" können als DFAs. Der folgende Satz beantwortet diese Frage mit "nein".

Satz 7.13. Für jeden NFA $A = (\Sigma, Q, \delta, q_0, F)$ gibt es einen DFA $A' = (\Sigma, Q', \delta', q_0', F')$ mit $L(A') = L(A)$. D.h.: NFAs und DFAs können genau dieselben Sprachen akzeptieren.

Beweis: Sei $A = (\Sigma, Q, \delta, q_0, F)$ der gegebene NFA. Wir betrachten den DFA $A' = (\Sigma, Q', \delta', q_0', F')$ mit

- Eingabealphabet Σ,
- Zustandsmenge $Q' := \mathcal{P}(Q)$,
- Startzustand $q_0' := \{q_0\}$,
- Endzustandsmenge $F' := \{X \in Q' \,:\, X \cap F \neq \emptyset\}$,
- Übergangsfunktion $\delta' : Q' \times \Sigma \to Q'$, wobei für alle $X \in Q' = \mathcal{P}(Q)$ und alle $a \in \Sigma$ gilt:

$$\delta'(X, a) \;:=\; \bigcup_{q \in X} \delta(q, a).$$

Per Induktion nach n kann man leicht nachweisen (Details: **Übung!**), dass für alle $n \in \mathbb{N}$ und jedes $w \in \Sigma^*$ mit $|w| = n$ gilt:

$$\widehat{\delta'}(q_0', w) \;=\; \widehat{\delta}(q_0, w).$$

Daraus folgt, dass für jedes Eingabewort $w \in \Sigma^*$ gilt:

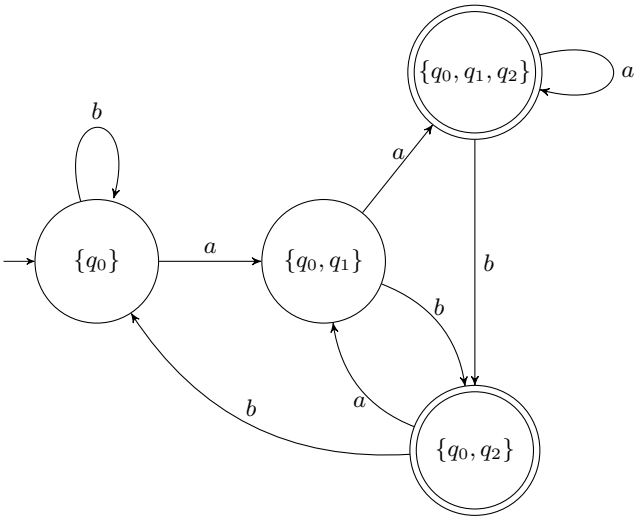
$$\begin{aligned} w \in L(A') \quad &\iff \quad \widehat{\delta'}(q_0', w) \in F' \\ &\iff \quad \widehat{\delta'}(q_0', w) \cap F \neq \emptyset \\ &\iff \quad \widehat{\delta}(q_0, w) \cap F \neq \emptyset \\ &\iff \quad w \in L(A). \end{aligned}$$

Insbesondere ist daher $L(A') = L(A)$. □

Die im obigen Beweis durchgeführte Konstruktion eines DFAs A' wird auch **Potenzmengenkonstruktion** (engl: **subset construction**) genannt.

Potenzmengenkonstruktion

Beispiel 7.14. Wir führen die Potenzmengenkonstruktion an dem NFA A_3 aus Beispiel 7.9 durch, wobei wir die Zustände von A_3, von links nach rechts gelesen, mit q_0, q_1 und q_2 bezeichnen. Im folgenden ist die graphische Darstellung des aus der Potenzmengen-Konstruktion resultierenden DFAs A_3' angegeben, wobei nur solche Zustände aus $\mathcal{P}(\{q_0, q_1, q_2\})$ berücksichtigt werden, die vom Startzustand $q_0' := \{q_0\}$ aus erreicht werden können:



7.4 Das Pumping-Lemma für reguläre Sprachen

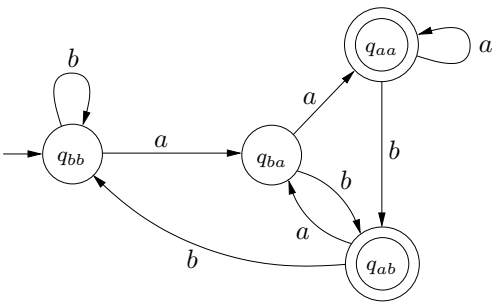
Definition 7.15 (reguläre Sprachen). Sei Σ ein endliches Alphabet.
reguläre Sprache
Eine Sprache $L \subseteq \Sigma^*$ heißt **regulär**, falls es einen NFA $A = (\Sigma, Q, \delta, q_0, F)$ mit $L(A) = L$ gibt.

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 7.17, der unter dem Namen **Pumping-Lemma** bekannt ist. Bevor wie den Satz präzise angeben, betrachten wir zunächst ein Beispiel:

Beispiel 7.16. 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 $i \geqslant 0$ das Eingabewort $ba(ba)^i a$.

Der folgende Satz 7.17 beruht auf demselben Prinzip sowie der Tatsache, dass in jedem Graph auf z Knoten gilt: Jeder Weg der Länge $\geqslant z$ enthält einen Kreis (d.h. mindestens ein Knoten wird auf dem Weg mehr als 1-mal besucht).

Pumping-Lemma

Satz 7.17 (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 $|x| \geqslant z$ 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) $|uv| \leqslant z$

(3) $|v| \geqslant 1$

(4) für jedes $i \in \mathbb{N}$ gilt: $uv^i w \in L$.
(d.h.: $uw \in L$, $uvw \in L$, $uvvw \in L$, $uvvvw \in L$, ...)

Beweis: Da L regulär ist, gibt es einen 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 $|x| \geqslant z$, 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 $|x| \geqslant z = |Q|$ ist, können die Zustände $q_0, q_1, \ldots, q_z$ nicht alle verschieden sein. Daher gibt es ein $k \geqslant 0$ und ein $\ell \geqslant 1$, so dass $q_k = q_{k+\ell}$ und $k + \ell \leqslant z$. 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.

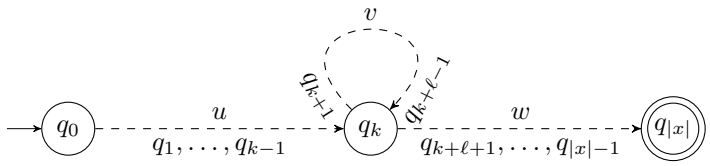
Offensichtlich gilt:

(1) $x = uvw$,

(2) $|uv| = k + \ell \leqslant z$,

(3) $|v| = \ell \geqslant 1$.

Skizze:



Daher gilt für jedes $i \geqslant 0$: A akzeptiert das Eingabewort uv^iw, d.h. $uv^iw \in L$. □

Unter Verwendung des Pumping-Lemmas kann man nachweisen, dass gewisse Sprachen **nicht** regulär sind:

Beispiel 7.18. Sei $\Sigma := \{a, b\}$. Die Sprache $L := \{a^n b^n : n \in \mathbb{N}\}$ ist **nicht** regulär.
(Zum Vergleich: Gemäß Lemma 7.7(a) ist die Sprache $L_1 := \{a^n b^m : n \in \mathbb{N},\, m \in \mathbb{N}\}$ regulär.)

Beweis: Durch Widerspruch.
Angenommen, L ist regulär. Dann sei $z \in \mathbb{N}$ gemäß dem Pumping-Lemma (Satz 7.17) gewählt. Betrachte das Wort $x := a^z b^z$. Klar: $x \in L$ und $|x| \geqslant z$. Gemäß dem Pumping-Lemma gibt es eine Zerlegung von x in Worte $u, v, w \in \{a, b\}^*$, so dass

(1) $x = uvw$

(2) $|uv| \leqslant z$

(3) $|v| \geqslant 1$

(4) f.a. $i \in \mathbb{N}$ gilt: $uv^iw \in L$.

Wegen $x = a^z b^z = uvw$ und $|uv| \leqslant z$ und $|v| \geqslant 1$ gibt es eine Zahl $\ell \in \mathbb{N}_{>0}$, so dass $v = a^\ell$. Wegen (4) gilt insbesondere für $i = 0$, dass $uw \in L$. Wegen $x = uvw = a^z b^z$ und $v = a^\ell$ mit $\ell \geqslant 1$ gilt

$$uw \;=\; a^{z-\ell} b^z \;\overset{\ell \geqslant 1}{\notin}\; \{a^n b^n : n \in \mathbb{N}\} \;=\; L.$$

Widerspruch! □

Beispiel 7.19. 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 7.17) gewählt. Da es unendlich viele Primzahlen gibt, gibt es auch eine Primzahl p mit $p \geqslant z + 2$. Sei p solch eine Primzahl. Betrachte nun das Wort $x := a^p$. Klar: $x \in L$ und $|x| \geqslant z$. Gemäß dem Pumping-Lemma gibt es also eine Zerlegung von x in Worte $u, v, w \in \{a\}^*$, so dass

(1) $x = uvw$,

(2) $|uv| \leqslant z$,

(3) $|v| \geqslant 1$,

(4) f.a. $i \in \mathbb{N}$ gilt: $uv^iw \in L$.

Wegen (4) gilt insbesondere für $i = |uw|$, dass $uv^i w \in L$. Es gilt dann

$$\left|uv^i w\right| \;=\; |uw| + i \cdot |v| \;=\; |uw| + |uw| \cdot |v| \;=\; |uw| \cdot (1 + |v|).$$

Wegen

$$|uw| \;\geqslant\; |w| \;=\; |x| - |uv| \overset{|uv| \leqslant z}{\geqslant} p - z \overset{p \geqslant z+2}{\geqslant} 2 \qquad \text{und} \qquad 1 + |v| \overset{|v| \geqslant 1}{\geqslant} 2$$

ist $\left|uv^i w\right|$ daher keine Primzahl, d.h. $uv^i w \notin L$. Widerspruch! □

Auf ähnliche Weise kann man auch zeigen, dass keine der folgenden Sprachen regulär ist:

- $\{a^n b^m : n, m \in \mathbb{N} \text{ mit } n \leqslant m\}$
- $\{ww : w \in \{a, b\}^*\}$
- $\{w : w \in \{a, b\}^*,\, w = w^R\}$ („Palindrome“)
- $\{w \in \{a\}^* : |w| \text{ ist eine Quadratzahl, d.h. ex. } n \in \mathbb{N} \text{ s.d. } |w| = n^2\}$.

7.5 Reguläre Ausdrücke

Reguläre Ausdrücke beschreiben Mengen von Worten, die nach bestimmten Regeln bzw. „Mustern“ aufgebaut sind.

Beispiel 7.20. Die Menge aller Wörter über dem Alphabet $\{a, b\}$, deren vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:

$$(a \mid b)^*\, a\, (a \mid b)$$

Definition 7.21 (Reguläre Ausdrücke: Syntax). Sei Σ ein endliches Alphabet.
Die Menge aller **regulären Ausdrücke über Σ** ist rekursiv wie folgt definiert:

reguläre Ausdrücke über Σ

Basisregeln:

- $\emptyset$ ist ein regulärer Ausdruck über Σ („leere Menge“).
- ε ist ein regulärer Ausdruck über Σ („leeres Wort“).
- Für jedes $a \in \Sigma$ gilt: a ist ein regulärer Ausdruck über Σ.

Rekursive Regeln:

- Ist R ein regulärer Ausdruck über Σ, so ist auch R^* ein regulärer Ausdruck über Σ („Kleene-Stern“).
- Sind R und S reguläre Ausdrücke über Σ, so ist auch
  - $(R \cdot S)$ ein regulärer Ausdruck über Σ („Konkatenation“).
  - $(R \mid S)$ ein regulärer Ausdruck über Σ („Vereinigung“).

Definition 7.22 (Reguläre Ausdrücke: Semantik). Sei Σ ein endliches Alphabet.
Jeder reguläre Ausdruck R über Σ **beschreibt** (oder definiert) eine Sprache $L(R) \subseteq \Sigma^*$, die induktiv wie folgt definiert ist:

- $L(\emptyset) := \emptyset$.
- $L(\varepsilon) := \{\varepsilon\}$.
- Für jedes $a \in \Sigma$ gilt: $L(a) := \{a\}$.
- Ist R ein regulärer Ausdruck über Σ, so ist
$$L(R^*) := \{\varepsilon\} \cup \{w_1 \cdots w_k : k \in \mathbb{N}_{>0},\, w_1 \in L(R), \ldots, w_k \in L(R)\}.$$
- Sind R und S reguläre Ausdrücke über Σ, so ist
  - $L\big((R \cdot S)\big) := \{wu : w \in L(R),\, u \in L(S)\}$.
  - $L\big((R \mid S)\big) := L(R) \cup L(S)$.

Notation 7.23. Zur vereinfachten Schreibweise und besseren Lesbarkeit von regulären Ausdrücken vereinbaren wir folgende Konventionen:

- Den „Punkt“ bei der Konkatenation $(R \cdot S)$ darf man weglassen.
- Bei Ketten gleichartiger Operatoren darf man Klammern weglassen: z.B. schreiben wir kurz $(R_1 \mid R_2 \mid R_3 \mid R_4)$ statt $\Big(\big((R_1 \mid R_2) \mid R_3\big) \mid R_4\Big)$ und $(R_1R_2R_3R_4)$ statt $\Big(\big((R_1R_2)R_3\big)R_4\Big)$.
- „Präzedenzregeln“: (1): $*$ bindet stärker als $\cdot$
  (2): $\cdot$ bindet stärker als $\mid$
- Äußere Klammern, die einen regulären Ausdruck umschließen, dürfen weggelassen werden.
- Zur besseren Lesbarkeit dürfen zusätzliche Klammern benutzt werden.

Beispiel 7.24.

(a) $a \mid bc^*$ ist eine verkürzte Schreibweise für den regulären Ausdruck $(a \mid (b \cdot c^*))$.
Die von diesem regulären Ausdruck beschriebene Sprache ist
$$L(a \mid bc^*) = \{a\} \cup \{w \in \{a,b,c\}^* : \text{der erste Buchstabe von } w \text{ ist ein } b \text{ und alle weiteren Buchstaben von } w \text{ sind } c\text{'s}\}.$$

(b) $L((a \mid b)^*) = \{a,b\}^*$.

(c) Die Menge aller Worte über $\{a,b,c\}$, in denen abb als Teilwort vorkommt, wird durch den folgenden regulären Ausdruck beschrieben:
$$(a \mid b \mid c)^* abb\, (a \mid b \mid c)^*.$$

(d) Die Menge aller Worte über $\{a,b,c\}$, deren letzter oder vorletzter Buchstabe ein a ist, wird durch den folgenden regulären Ausdruck beschrieben:
$$(a \mid b \mid c)^* a\, (\varepsilon \mid a \mid b \mid c).$$

Beispiel 7.25.

(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

$$\textit{Vorwahl}/\textit{Nummer}$$

kodieren, wobei *Vorwahl* und *Nummer* nicht-leere Ziffernfolgen sind, *Vorwahl* mit einer Null beginnt und *Nummer nicht* mit einer Null beginnt. 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:

$$\big(\,\varepsilon\,|\,069/\,\big)\;\;798\;\;\big(\,\varepsilon\,|\,\text{-}\,\big)\;\;\big(\;0\;\;|\;\;(1\,|\,\cdots\,|\,9)(0\,|\,1\,|\,\cdots\,|\,9)^*\;\big)$$

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 nichtdeterministische) endliche Automaten akzeptiert werden können. Diese Sprachen werden **reguläre Sprachen** genannt.

Details: In der Veranstaltung "GL-2: Formale Sprachen und Berechenbarkeit".

7.6 Ausblick

Reguläre Grammatiken:

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 (vgl. Kapitel 8), die von einer besonders einfachen Form sind.

Das Leerheitsproblem:

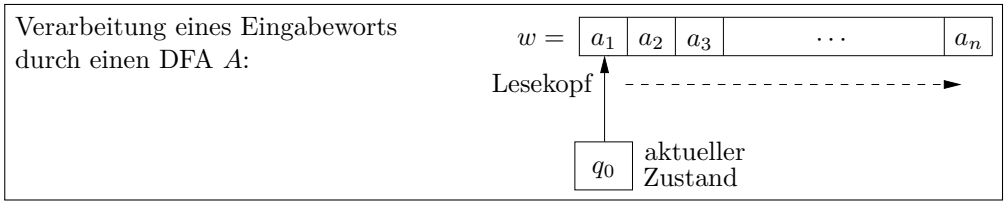
Eine typische Fragestellung bzgl. DFAs oder NFAs ist das so genannte **Leerheitsproblem**:

> Gegeben sei ein DFA oder NFA A. Wie kann man herausfinden, ob $L(A) \neq \emptyset$ ist, d.h. ob es (mindestens) ein Eingabewort gibt, das von A akzeptiert wird?

(Man erinnere sich z.B. an das Flussüberquerungsproblem aus Beispiel 1.2.)
Durch Betrachtung der graphischen Darstellung von A kann diese Frage leicht beantwortet werden:

> Teste, ob es in der graphischen Darstellung von A einen Weg gibt, der vom Startzustand zu einem akzeptierenden Zustand führt.

Eine andere Sichtweise auf DFAs und NFAs:



7.7 Literaturhinweise

[15] Kapitel 7.1

[12] Kapitel 2 und 3

[28] Kapitel 4.1 und 4.2

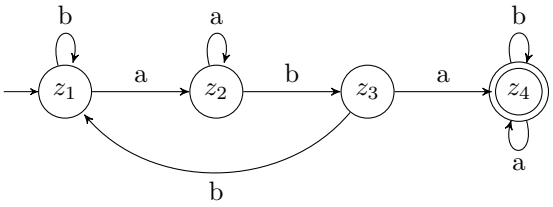
[29] Kapitel 4.1

[26] Kapitel 1.2

7.8 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 nichtdeterministischen 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 nichtdeterministischen 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.

8 Kontextfreie Grammatiken zur 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 6.2 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 6.7 gesehen – auch relationale Datenbanken, die z.B. Informationen über Kinofilme und das aktuelle Kinoprogramm enthalten. In Kapitel 8 werden wir nun einen weiteren Kalkül kennenlernen, mit dem man strukturelle Eigenschaften von Systemen beschreiben kann: **kontextfreie Grammatiken**.

Kontextfreie Grammatiken (kurz: KFGs) eignen sich besonders gut zur Modellierung von beliebig tief geschachtelten baumartigen Strukturen. KFGs können gleichzeitig

- hierarchische Baumstrukturen und
- Sprachen und deren textuelle Notation

spezifizieren. KFGs werden z.B. angewendet zur Definition von:

- Programmen einer Programmiersprache und deren Struktur, z.B. Java, C, Pascal (KFGs spielen z.B. beim „Compilerbau“ eine wichtige Rolle).
- Datenaustauschformaten, d.h. Sprachen als Schnittstelle zwischen Software-Werkzeugen, z.B. HTML, XML.
- Bäumen zur Repräsentation strukturierter Daten, z.B. XML.
- Strukturen von Protokollen beim Austausch von Nachrichten zwischen Prozessen oder Geräten.

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: Formale Sprachen und Berechenbarkeit” behandelt.

8.1 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 8.1 (KFG). Eine kontextfreie Grammatik $G = (T, N, S, P)$ besteht aus

- einer endlichen Menge T, der so genannten Menge der **Terminalsymbole** (die Elemente aus T werden auch **Terminale** genannt). — Terminalsymbole, Terminale
- 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. $T \cap N = \emptyset$. Die Menge $V := T \cup N$ heißt **Vokabular**; die Elemente in V nennt man auch **Symbole**. — Nichtterminalsymbole, Nichtterminale; Vokabular, Symbole
- einem Symbol $S \in N$, dem so genannten **Startsymbol**. — 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. $A ::= x$). — Produktionen

Beispiel 8.2. 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).

8.2 Bedeutung der Produktionen: Semantik von KFGs

Jede Produktion einer KFG, etwa die Produktion $A \to x$, kann man auffassen als:

- eine *Strukturregel*, die besagt „Ein A besteht aus x“ oder als
- eine *Ersetzungsregel*, die besagt „A kann man durch x ersetzen.“

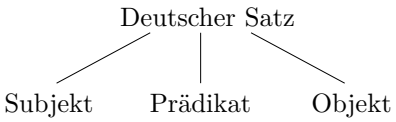
Beispielsweise kann man die Produktion

$$\text{DeutscherSatz} \to \text{Subjekt Prädikat Objekt}$$

verstehen als Aussage, die besagt:

„Ein Deutscher Satz ist aufgebaut aus Subjekt Prädikat Objekt."

Graphische Darstellung:



Das Grundkonzept für die Anwendung von Produktionen einer KFG ist die „Ableitung":

Definition 8.3 (Ableitung). Sei $G = (T, N, S, P)$ 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

  $$uAv \Longrightarrow_G uxv \quad (\text{bzw. kurz: } uAv \Longrightarrow uxv)$$

  Ableitungsschritt

  und sagen, dass uAv in einem **Ableitungsschritt** zu uxv umgeformt werden kann.

  Ableitung

- Eine **Ableitung** ist eine endliche Folge von hintereinander angewendeten Ableitungsschritten. Für Worte $w \in V^*$ und $w' \in V^*$ schreiben wir

  $$w \Longrightarrow_G^* w' \quad (\text{bzw. kurz: } w \Longrightarrow^* w'),$$

  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 8.4. Sei G_K die „Klammer-Grammatik" aus Beispiel 8.2.
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

$$\text{Klammerung} \implies^* (()())$$

Definition 8.5 (Die Sprache $L(G)$ einer KFG G).
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.:

Sprache einer KFG, $L(G)$

$$L(G) := \{w \in T^* : S \implies_G^* w\}.$$

Man beachte, dass $L(G) \subseteq T^*$ ist. Daher kommen in Worten aus $L(G)$ keine Nichtterminale vor!

Beispiel 8.6. Die KFG G_K aus Beispiel 8.2 definiert die Sprache $L(G_\text{K})$, die aus allen „wohlgeformten Klammerausdrücken“ besteht, die von einem Klammerpaar umschlossen werden. Beispielsweise gehören folgende Worte zu $L(G_\text{K})$:

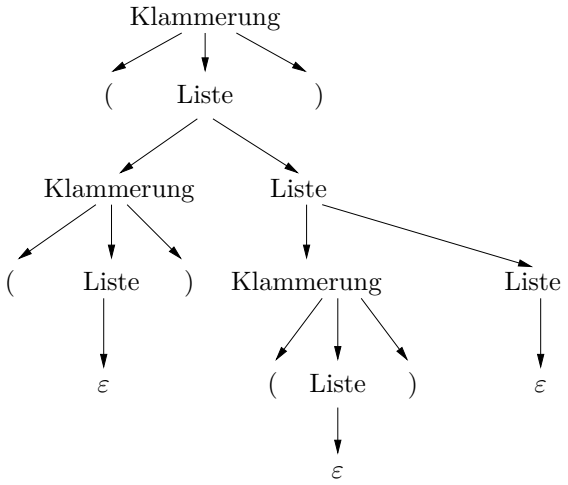
$$() , \quad (()) , \quad (()()) , \quad (((())())) ,$$

aber die Worte $)($, $(((\,$, $(()()\,$, $(())()\,$, (Liste) gehören **nicht** zu $L(G_\text{K})$.

Beispiel 8.7. Sei G_K die „Klammer-Grammatik“ aus Beispiel 8.2. Die Ableitung

$$\begin{aligned}
\text{Klammerung} &\implies (\text{Liste}) \\
&\implies (\text{Klammerung Liste}) \\
&\implies ((\text{Liste}) \text{ Liste}) \\
&\implies (() \text{ Liste}) \\
&\implies (() \text{ Klammerung Liste}) \\
&\implies (() \text{ Klammerung}) \\
&\implies (()(\text{Liste})) \\
&\implies (()())
\end{aligned}$$

wird durch den folgenden **Ableitungsbaum** dargestellt:



Ableitungsbaum

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. $i \in \{1, \ldots, |x|\}$).
- Die Anwendung einer Produktion der Form $A \to \varepsilon$ mit $A \in N$ wird im Ableitungsbaum repräsentiert durch einen Knoten, der mit dem Symbol A markiert ist und der genau ein Kind hat, das mit ε markiert ist.

Beachte: Ein Ableitungsbaum kann mehrere Ableitungen repräsentieren. Beispielsweise repräsentiert der Ableitungsbaum aus Beispiel 8.7 auch die Ableitung

$$\begin{aligned}
\text{Klammerung} &\Longrightarrow (\text{Liste}) \\
&\Longrightarrow (\text{Klammerung Liste}) \\
&\Longrightarrow (\text{Klammerung Klammerung Liste}) \\
&\Longrightarrow ((\text{Liste}) \text{ Klammerung Liste}) \\
&\Longrightarrow ((\text{Liste})(\text{Liste}) \text{ Liste}) \\
&\Longrightarrow (()(\text{Liste}) \text{ Liste}) \\
&\Longrightarrow (()() \text{ Liste}) \\
&\Longrightarrow (()()) \ ,
\end{aligned}$$

in der gegenüber der ursprünglichen Ableitung aus Beispiel 8.7 einige Ableitungsschritte vertauscht sind. Im Ableitungsbaum wird von der konkreten Reihenfolge, in der die einzelnen Ableitungsschritte vorkommen, abstrahiert.

8.3 Beispiele

Im Folgenden betrachten wir einige weitere Beispiele für kontextfreie Grammatiken.

Beispiel 8.8 (Aussagenlogik). Wir konstruieren eine KFG

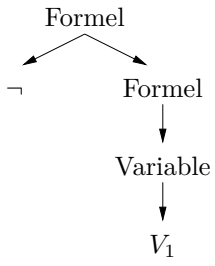
$$G_{\text{AL}} \ = \ (T, N, S, P),$$

deren Sprache $L(G_{\text{AL}})$ gerade die Menge aller aussagenlogischen Formeln ist, in denen nur Variablen aus $\{V_0, V_1, V_2\}$ vorkommen:

- Terminalsymbole $T := \{V_0, V_1, V_2, \mathbf{0}, \mathbf{1}, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$
- Nichtterminalsymbole $N := \{\text{Formel}, \text{Variable}, \text{Junktor}\}$
- Startsymbol $S := \text{Formel}$
- Produktionen

$$\begin{aligned} P := \{&\text{Formel} \to \mathbf{0},\ \text{Formel} \to \mathbf{1},\ \text{Formel} \to \text{Variable},\\ &\text{Formel} \to \neg\text{Formel},\\ &\text{Formel} \to (\text{Formel Junktor Formel}),\\ &\text{Variable} \to V_0,\ \text{Variable} \to V_1,\ \text{Variable} \to V_2,\\ &\text{Junktor} \to \wedge,\ \text{Junktor} \to \vee,\ \text{Junktor} \to \rightarrow,\ \text{Junktor} \to \leftrightarrow\}. \end{aligned}$$

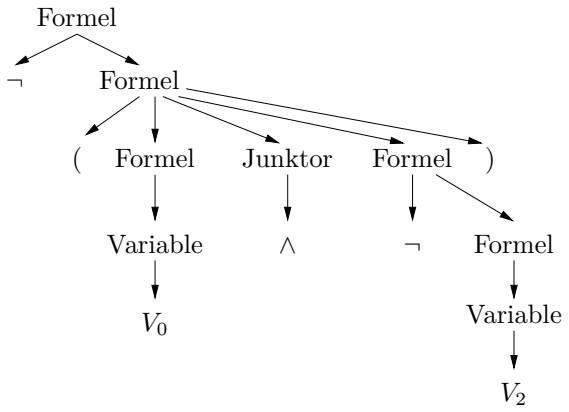
Beispiele für Ableitungsbäume:



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 $L(G_{\text{AL}})$ ist die Formel $\neg V_1$.



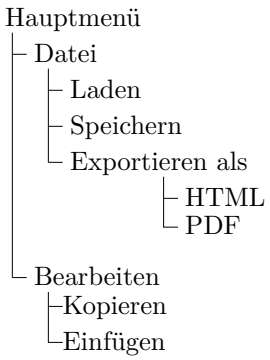
Dieser Ableitungsbaum repräsentiert die Ableitung

$$\begin{aligned} \text{Formel} &\Longrightarrow \neg\text{Formel}\\ &\Longrightarrow \neg(\text{Formel Junktor Formel})\\ &\Longrightarrow \neg(\text{Variable Junktor Formel})\\ &\Longrightarrow \neg(V_0 \text{ Junktor Formel})\\ &\Longrightarrow \neg(V_0 \wedge \text{Formel})\\ &\Longrightarrow \neg(V_0 \wedge \neg\text{Formel})\\ &\Longrightarrow \neg(V_0 \wedge \neg\text{Variable})\\ &\Longrightarrow \neg(V_0 \wedge \neg V_2). \end{aligned}$$

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 8.9 (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:



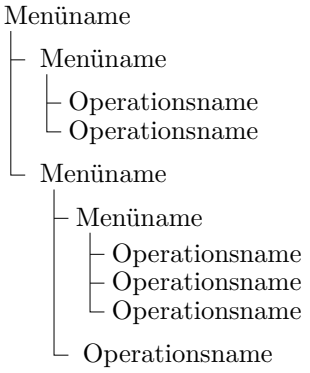
Zur Spezifizierung der Grundstruktur solcher Menüs kann man folgende Grammatik $G_{\text{Menü}}$ verwenden:

$$G_{\text{Menü}} \;=\; (T, N, S, P)$$

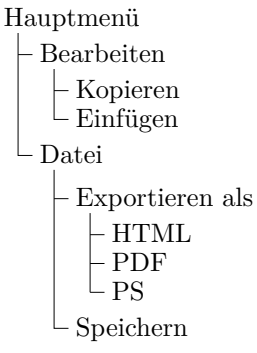
mit

- $T \;:=\; \{\text{Menüname}, \text{Operationsname}\}$
- $N \;:=\; \{\text{Menü}, \text{Eintragsfolge}, \text{Eintrag}\}$
- $S \;:=\; \text{Menü}$
- $P \;:=\; \{\; \text{Menü} \rightarrow \text{Menüname Eintragsfolge},$
  $\text{Eintragsfolge} \rightarrow \text{Eintrag},$
  $\text{Eintragsfolge} \rightarrow \text{Eintrag Eintragsfolge},$
  $\text{Eintrag} \rightarrow \text{Operationsname},$
  $\text{Eintrag} \rightarrow \text{Menü} \;\}$

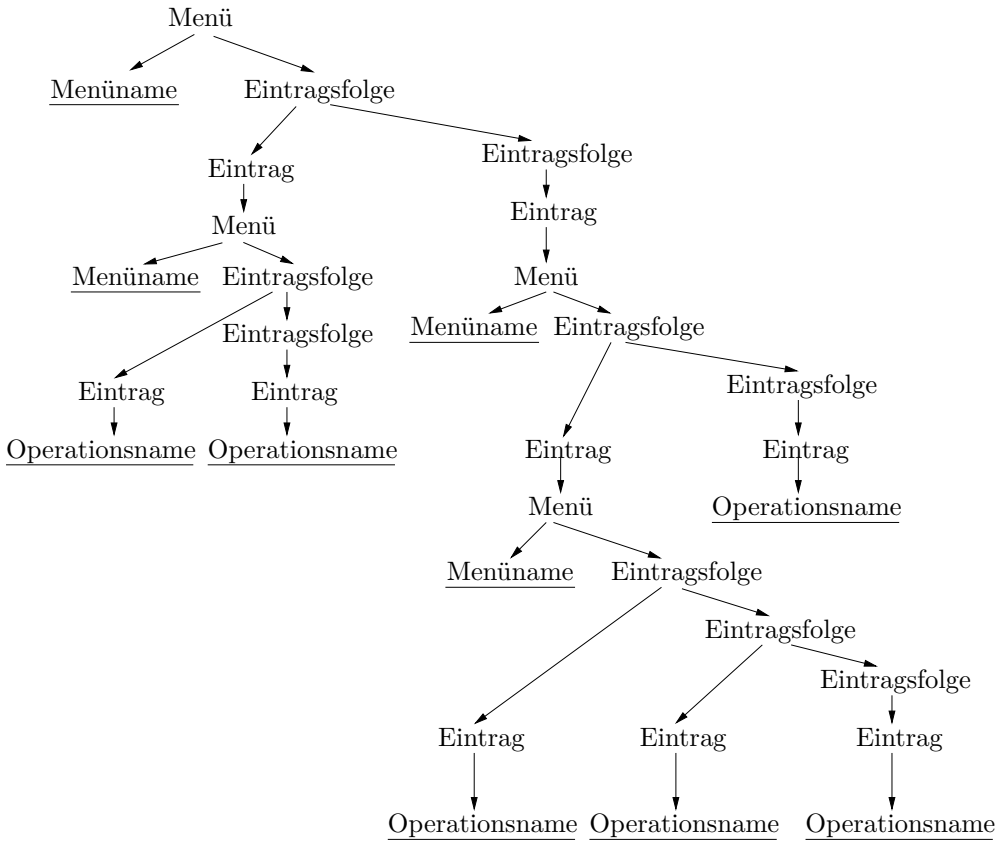
Jeder Ableitungsbaum repräsentiert die Struktur eines Menüs. Ein Menü der Struktur



Beispiel:



wird z.B. von folgendem Ableitungsbaum repräsentiert:



Beispiel 8.10 (HTML-Tabellen).
HTML (HyperText Markup Language) ist ein Format zur Beschreibung von verzweigten Dokumenten im Internet. Ein Bestandteil, der oft im Quell-Code von Internet-Seiten vorkommt, sind Tabellen. Z.B. wird der Eintrag

Tag	Zeit	Raum
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> NM 10 </td>
  </tr>
</table>
```

erzeugt.

Das Symbol `<table>` steht hier für den Anfang einer Tabelle, `</table>` steht für das Ende einer Tabelle. Die Symbole `<tr>` und `</tr>` stehen für den Anfang bzw. das Ende einer Zeile der Tabelle. Die Symbole `<td>` und `</td>` stehen für den Anfang bzw. das Ende eines Eintrags in einer Zelle der Tabelle. Als Einträge in einzelnen Zellen kann z.B. Text stehen oder eine weitere Tabelle.

Im Folgenden konstruieren wir eine Grammatik

$$G_{\text{HTML-Tabellen}} \;=\; (T, N, S, P),$$

so dass die von $G_{\text{HTML-Tabellen}}$ erzeugte Sprache aus (möglicherweise geschachtelten) HTML-Tabellen besteht:

- $T \;:=\; \big\{$ `<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>`,
  $\mathrm{a}, \ldots, \mathrm{z},\ \mathrm{A}, \ldots, \mathrm{Z},\ 0, 1, \ldots, 9,\ :,\ \text{-},\ \text{␣},\ \ddot{\mathrm{a}},\ \ddot{\mathrm{o}},\ \ddot{\mathrm{u}},\ \ss,\ \ddot{\mathrm{A}},\ \ddot{\mathrm{O}},\ \ddot{\mathrm{U}}\big\}$
- $N \;:=\; \big\{$ Tabelle, Zeile, Eintrag, Text, Zeilen, Einträge $\big\}$
- $S \;:=$ Tabelle
- $P \;:=\; \big\{$ Tabelle $\to$ `<table>`Zeilen`</table>`,
  Zeilen $\to$ Zeile,
  Zeilen $\to$ Zeile Zeilen,
  Zeile $\to$ `<tr>`Einträge`</tr>`,
  Einträge $\to$ Eintrag,
  Einträge $\to$ Eintrag Einträge,
  Eintrag $\to$ `<td>`Text`</td>`,
  Eintrag $\to$ Tabelle,
  Text $\to$ a, $\ldots$, Text $\to$ z, Text $\to$ A, $\ldots$, Text $\to$ Ü,
  Text $\to$ a Text, $\ldots$, Text $\to$ z Text, Text $\to$ A Text, $\ldots$, Text $\to$ Ü Text $\big\}$.

Die oben angegebene Beispiel-HTML-Tabelle wird z.B. durch eine Ableitung erzeugt, die durch den Ableitungsbaum in Abbildung 8.1 repräsentiert wird.

8.4 Ausblick

Typische Fragestellungen bzgl. kontextfreien Grammatiken:

(a) Welche Sprachen können prinzipiell durch KFGs erzeugt werden, welche nicht?

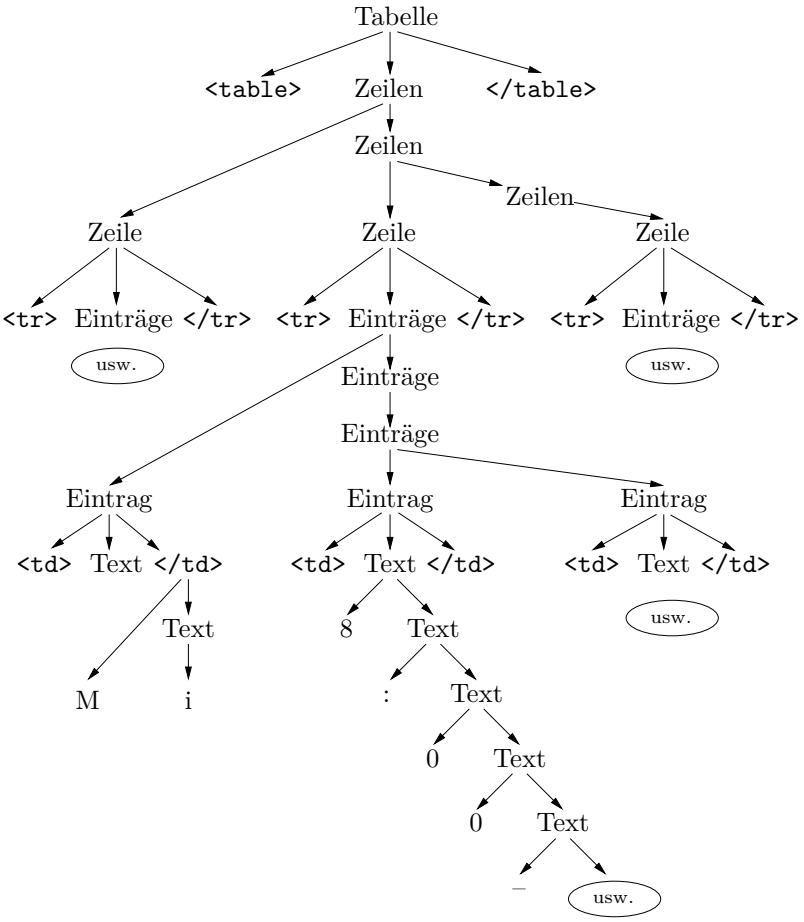


Abbildung 8.1: Ableitungsbaum für eine Beispiel-HTML-Tabelle

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 $\underbrace{aa\cdots a}_{n \text{ mal}}\underbrace{bb\cdots b}_{n \text{ mal}}$.

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: Formale Sprachen und Berechenbarkeit". Dort wird in Analogie zu Satz 7.17 auch ein "Pumping-Lemma für KFGs" gezeigt.

(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**.

Einen Algorithmus zum Lösen des Wortproblems für KFGs werden Sie in der Vorlesung "GL-2: Formale Sprachen und Berechenbarkeit" kennenlernen: den so genannten CYK-Algorithmus,

der nach seinen Erfindern Cocke, Younger und Kasami benannt ist.

Reguläre Sprachen vs. kontextfreie Grammatiken.

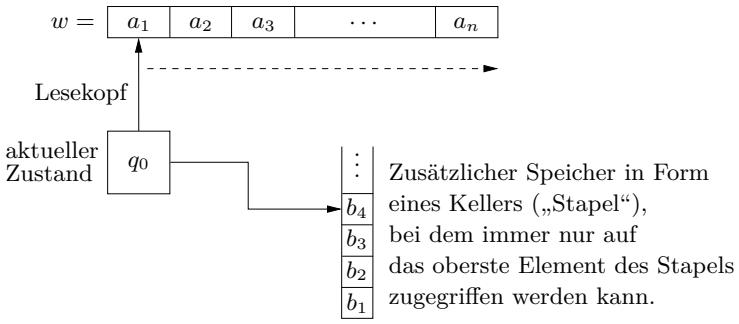
Generell gilt: Für jede reguläre Sprache L gibt es eine kontextfreie Grammatik, die die Sprache L erzeugt. Aber es gibt kontextfreie Grammatiken, die nicht-reguläre Sprachen erzeugen.

Beispiel: Die Sprache $L_1 = \{a^n b^n : n \in \mathbb{N}\}$ wird von der oben angegebenen KFG G_1 erzeugt. Andererseits wissen wir aus Beispiel 7.18, dass die Sprache L_1 nicht regulär ist.

Kellerautomaten

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**.

Schematische Darstellung der Verarbeitung eines Eingabeworts durch einen Kellerautomaten:



Details: In der Vorlesung “GL-2: Formale Sprachen und Berechenbarkeit”. Dort werden auch allgemeinere Arten von Grammatiken betrachtet, z.B. so genannte **kontextsensitive Grammatiken**.

kontextsensitive Grammatiken

8.5 Literaturhinweise

[15] Kapitel 6.1 und 6.2

[28] Kapitel 6.1

[29] Kapitel 6.1

[26] Kapitel 1.3

8.6 Übungsaufgaben zu Kapitel 8

Aufgabe 8.1. 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 8.2. 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.

9 Ausblick auf weitere Modellierungstechniken

In diesem Kapitel werden die Grundzüge von zwei weiteren Kalkülen vorgestellt, die sich zur Beschreibung von Abläufen bzw. zur Modellierung von Datenbanken eignen: zum einen die **Petri-Netze** in Abschnitt 9.1, und zum anderen das **Entity-Relationship-Modell** in Abschnitt 9.2. Das Kapitel schließt in Abschnitt 9.3 mit einer **Fallstudie** ab, in der das Zusammenspiel von verschiedenen Modellierungskalkülen zur Beschreibung eines konkreten, etwas umfangreicheren Problems beleuchtet wird.

9.1 Petri-Netze zur Modellierung von Abläufen

Petri-Netze liefern einen formalen Kalkül, der besonders gut geeignet ist, um Abläufe zu modellieren, an denen mehrere Prozesse beteiligt sind. Hierbei werden die Interaktionen zwischen Prozessen, sowie die Effekte modelliert, die sich daraus ergeben, dass Operationen prinzipiell gleichzeitig ausgeführt werden können (Stichwort: „Nebenläufigkeit“). Daher eignen sich Petri-Netze besonders gut dazu, nebenläufige Prozesse zu beschreiben, bei denen Ereignisse gleichzeitig an mehreren Stellen des Systems Zustandsänderungen bewirken können. Petri-Netze wurden 1962 von C. A. Petri eingeführt.

Petri-Netze eigenen sich beispielsweise gut zur Modellierung von

- realen oder abstrakten Automaten und Maschinen,
- kommunizierenden Prozessen (z.B. in Rechnern),
- Verhalten von Software- oder Hardware-Komponenten,
- Geschäftsabläufen in einer Firma,
- Spielen (bzw. Spielregeln),
- biologischen Prozessen (Bioinformatik).

Der Kalkül der Petri-Netze basiert auf bipartiten gerichteten Graphen:
Es gibt zwei Sorten von Knoten:

- zum einen, Knoten, die „Bedingungen“ (so genannte **Stellen**) repräsentieren, und
- zum anderen, Knoten, die „Aktivitäten“ (so genannte **Transitionen**) repräsentieren.

Kanten verbinden „Aktivitäten“ mit ihren „Vorbedingungen“ und ihren „Nachbedingungen“. Knotenmarkierungen repräsentieren den veränderlichen „Zustand“ des Systems.

Definition 9.1 (Petri-Netz). Ein **Petri-Netz** $P = (S, T, F)$ besteht aus

Petri-Netz

- einer endlichen Menge S, den so genannten **Stellen** von P, Stellen
- einer endlichen Menge T, den so genannten **Transitionen** von P, Transitionen
- einer Relation $F \subseteq (S \times T) \cup (T \times S)$, den so genannten **Kanten** von P.

Die Mengen S und T sind disjunkt, d.h. $S \cap T = \emptyset$.

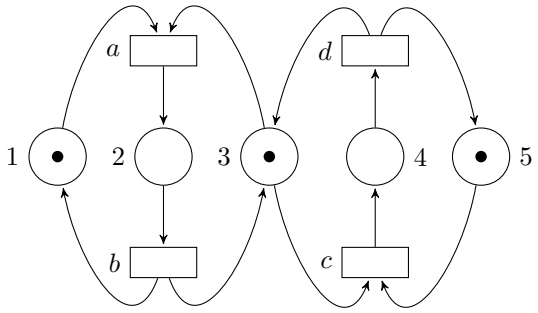
Ein Petri-Netz P bildet einen bipartiten gerichteten Graphen mit Knotenmenge $S \cup T$ und Kantenmenge F.

In der graphischen Darstellung werden **Stellen**, d.h. Knoten in S, durch Kreise repräsentiert; **Transitionen**, d.h. Knoten in T, werden durch Rechtecke dargestellt.

Definition 9.2 (Markierung).
Der „aktuelle Zustand“ eines Petri-Netzes $P = (S, T, F)$ wird durch eine **Markierungsfunktion** (kurz: **Markierung**) $M\colon S \to \mathbb{N}$ repräsentiert, die jeder Stelle $s \in S$ eine Anzahl $M(s)$ von so genannten **Marken** zuordnet.

Markierung
Marken

Beispiel 9.3. Die folgende Skizze gibt die graphische Darstellung eines Petri-Netzes $P = (S, T, F)$ und einer Markierung M an.



Die einzelnen “Marken”, die M einer Stelle $s \in S$ zuordnet, werden durch Punkte “•” in dem die Stelle s repräsentierenden Knoten dargestellt.
Die obige Skizze repräsentiert das Petri-Netz $P = (S, T, F)$ und die Markierung $M : S \to \mathbb{N}$ mit

- $S = \{1, 2, 3, 4, 5\}$,
- $T = \{a, b, c, d\}$,
- $F = \{(1, a), (3, a), (a, 2), (2, b), (b, 1), (b, 3), (3, c), (5, c), (c, 4), (4, d), (d, 3), (d, 5)\}$,
- $M\colon S \to \mathbb{N}$ mit $M(1) = M(3) = M(5) = 1$ und $M(2) = M(4) = 0$.

Ein Petri-Netz P zusammen mit einer Markierung M kann man sich als eine “Momentaufnahme”, d.h. als die Beschreibung eines “aktuellen Zustands” eines Systems vorstellen. Um Änderungen am Zustand des Systems beschreiben zu können, sind die folgenden Begriffe nützlich.

Definition 9.4 (Vorbereich und Nachbereich).
Sei $P = (S, T, F)$ ein Petri-Netz und sei $t \in T$ eine Transition von P. Wir setzen

Vorbereich(t)

- Vorbereich(t) $:= \{s \in S : (s, t) \in F\}$,

Nachbereich(t)

- Nachbereich(t) $:= \{s \in S : (t, s) \in F\}$.

Somit ist Vorbereich(t) die Menge aller Stellen, von denen aus eine Kante in t hineinführt, während Nachbereich(t) die Menge aller Stellen, ist 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:

Definition 9.5 (Schaltregel).
Sei $P = (S, T, F)$ ein Petri-Netz und sei $M\colon S \to \mathbb{N}$ eine Markierung. Das so genannte **Schalten einer Transition** $t \in T$ überführt die Markierung M in eine Markierung $M'\colon S \to \mathbb{N}$. Die Transition $\boldsymbol{t}$ **kann schalten**, wenn gilt:

F.a. Stellen $s \in$ Vorbereich(t) ist $M(s) \geqslant 1$.
D.h.: Jede Stelle s in Vorbereich(t) hat mindestens eine Marke.

Nachfolgemarkierung

Wenn die Transition t schaltet, so gilt für die so genannte **Nachfolgemarkierung** $\boldsymbol{M'}$ folgendes: f.a. $s \in S$ ist

$$M'(s) \;=\; \begin{cases} M(s) - 1, & \text{falls } s \in \text{Vorbereich}(t) \setminus \text{Nachbereich}(t) \\ M(s) + 1, & \text{falls } s \in \text{Nachbereich}(t) \setminus \text{Vorbereich}(t) \\ M(s), & \text{sonst.} \end{cases}$$

D.h.: Das **Schalten von Transition** $\boldsymbol{t}$ bewirkt, dass in jeder Stelle in Vorbereich(t) eine Marke entfernt wird und dass in jeder Stelle in Nachbereich(t) eine Marke hinzugefügt wird. Wenn mehrere Transitionen schalten können, so wird eine davon „nichtdeterministisch“ ausgewählt.

In jedem Schritt schaltet genau eine Transition. Durch schrittweises Schalten von Transitionen wird der Ablauf von Prozessen modelliert.

Beispiel 9.6. Seien $P = (S, T, F)$ und $M\colon S \to \mathbb{N}$ das Petri-Netz und die Markierung aus Beispiel 9.3. 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 $M'(1) = 0$, $M'(3) = 0$, $M'(2) = 1$, $M'(4) = 0$, $M'(5) = 1$. Die graphische Darstellung von P und M' ist in Abbildung 9.1 angegeben.

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 9.3, zusammen mit der Startmarkierung M, modelliert zwei zyklisch ablaufende Prozesse. Die Stelle 3 „synchronisiert“ die beiden Prozesse, so dass sich nie gleichzeitig in den 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.

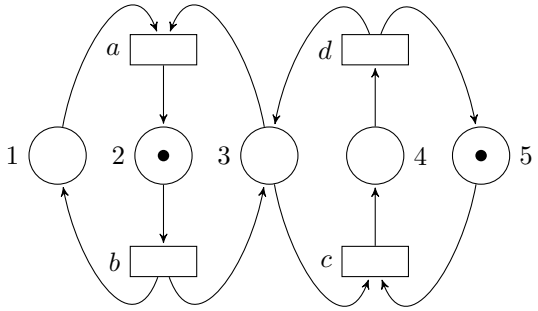


Abbildung 9.1: Skizze zu Beispiel 9.6

Beispiel 9.7. Ziel ist, die Schaltung einer Ampelanlage an einer Kreuzung zu modellieren. Dabei gibt es zwei sich zyklisch wiederholende Prozesse:

- „Grün“-Phase in Nord-Süd-Richtung,
- „Grün“-Phase in West-Ost-Richtung.

Die beiden Prozesse sollen sich immer abwechseln. Die graphische Darstellung eines Petri-Netzes, inklusive Anfangs-Markierung, das eine solche Ampelanlage modelliert, findet sich in Abbildung 9.2.

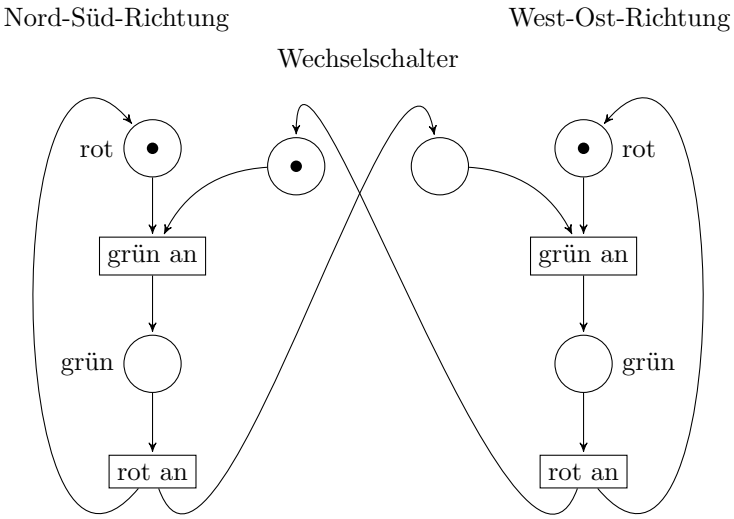


Abbildung 9.2: Petri-Netz zur Modellierung einer Ampelanlage

Die beiden Stellen namens „Wechselschalter“ koppeln die Prozesse, so dass abwechselnd in West-Ost- bzw. in Nord-Süd-Richtung die Ampel „grün“ ist.

9.2 Das Entity-Relationship-Modell zur Modellierung von Datenbanken

ER-Modell

Das Entity-Relationship-Modell (kurz: **ER-Modell**) geht zurück auf einen grundlegenden Artikel [4] von P. P. Chen aus dem Jahr 1976.

Das ER-Modell wird heute als Standard-Werkzeug 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

- **Entity**: „zu modellierende Informationseinheit“ (deutsch: „Objekt“, „Ding“, „Entität“)

Relationship

- **Relationship**: zur Modellierung von Beziehungen zwischen Entities (deutsch: „Beziehung“, „Relation“)

Attribut

- **Attribut**: Eigenschaft eines Entitys oder einer Beziehung.

Genauer:

Entity

- **Entity**: ein 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-Typ
Entity-Menge

- **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.

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 9.8. Abbildung 9.3 zeigt eine graphische Darstellung für eine Modellierung im ER-Modell. In diesem Beispiel geht es darum, Vorlesungen, Dozenten und für die Vorlesungen empfohlene Bücher darzustellen.

Die graphische Darstellung eines ER-Modells geschieht nach folgenden Konventionen:

- **Entity-Typen** werden als Rechtecke dargestellt (hier: Dozent/in, Vorlesung, Buch).
- **Eigenschaften von Entities**, so genannte **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 jedem 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

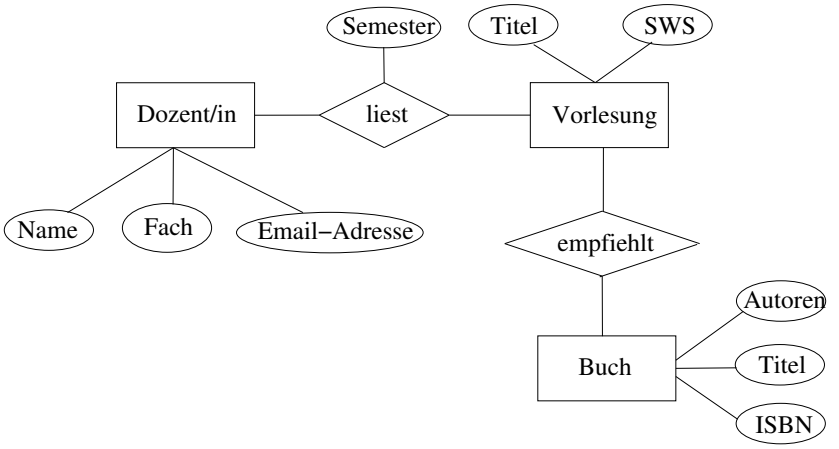
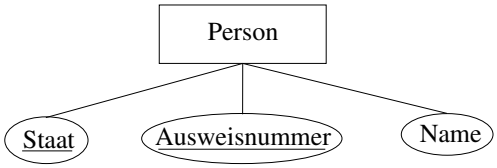
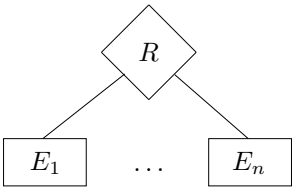


Abbildung 9.3: Ein ER-Modell, das Vorlesungen, Dozenten und für die Vorlesungen empfohlene Bücher darstellt



- Typen von Relationships, so genannte **Relationen-Typen**, werden durch Rauten dargestellt, die mit den betreffenden Entity-Typen durch Striche verbunden sind (z.B. ist in Beispiel 9.8 „liest“ ein Relationen-Typ, der angibt, welche/r Dozent/in welche Vorlesung liest). Relationen-Typen

  Allgemein gilt: Ein Relationen-Typ modelliert Beziehungen zwischen den Entities der betroffenen Entity-Typen. Ein ***n*-stelliger Relationen-Typ *R*** (für $n \geqslant 2$) verknüpft Entities aus n Entity-Typen $E_1, \ldots, E_n$. Er wird graphisch repräsentiert durch n-stelliger Relationen-Typ



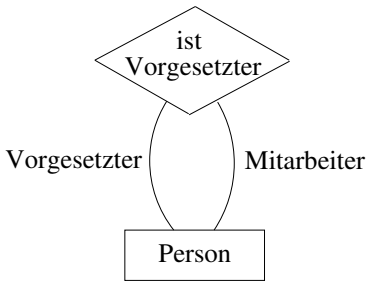
  Eine **konkrete Ausprägung des Relationen-Typs** R ist eine Menge von n-Tupeln $(e_1, \ldots, e_n)$, wobei für jedes $i \in \{1, \ldots, n\}$ gilt: e_i ist ein Entity des Entity-Typs E_i.

- Auch Relationen-Typen können Attribute haben (z.B. hat der Relationen-Typ „liest“ in Beispiel 9.8 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 dann, 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 9.9. Man beachte die Auswirkung von Modellierungsentscheidungen beim Entwickeln eines ER-Modells:

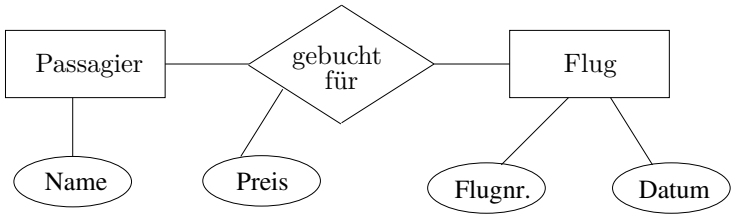


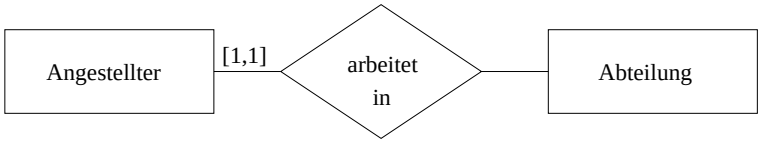
Abbildung 9.4: ER-Modell für ein Reisebüro

Nutzt ein Reisebüro das ER-Modell aus Abbildung 9.4, 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 9.5 benutzen.

Kardinalität

Ein weiterer Bestandteil von ER-Modellen: die Kardinalität von Relationen-Typen:

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 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:



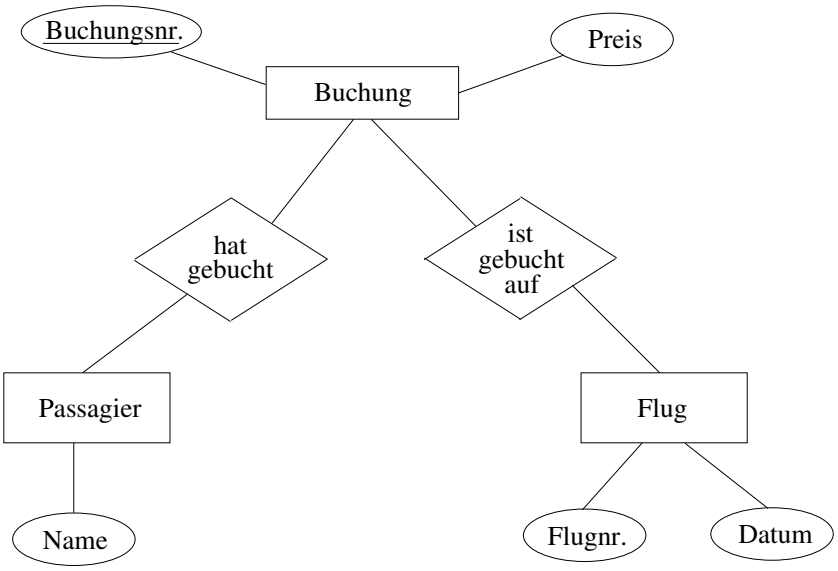
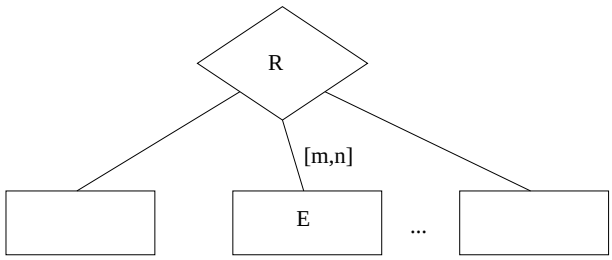


Abbildung 9.5: ER-Modell für ein Reisebüro mit „Mehrfachbuchungen“

Allgemein besagt ein Relationen-Typ der Form

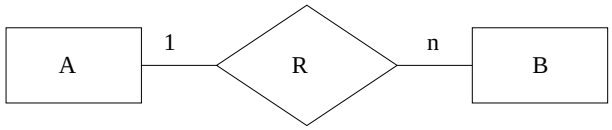


dass für jede konkrete Ausprägung dieses Typs gelten muss: Jedes Entity e der konkreten Ausprägung des Entity-Typs E kommt in mindestens m und höchstens n Tupeln vor.

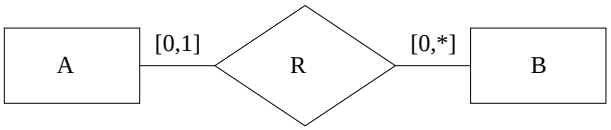
Spezialfälle für $[m, n]$:

- $[1, 1]$ bedeutet: „in genau einem Tupel“,
- $[0, 1]$ bedeutet: „in höchstens einem Tupel“,
- $[0, *]$ bedeutet: „in beliebig vielen Tupeln“.
  Die Angabe $[0, *]$ wird oft einfach weggelassen.

Kurznotation für 2-stellige Relationen-Typen:



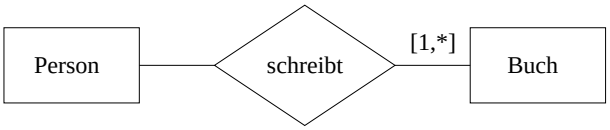
bedeutet



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."

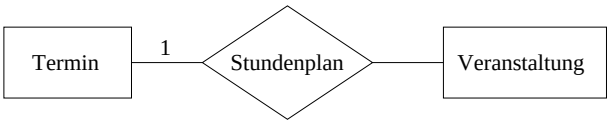
Beispiel 9.10.

(a)



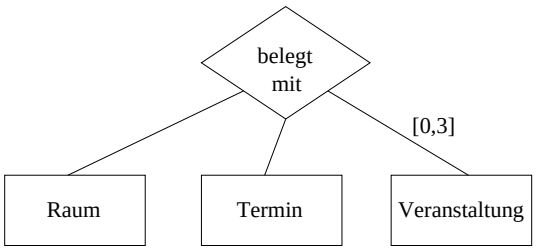
bedeutet: „Jedes Buch wird von mindestens einer Person geschrieben."

(b)



bedeutet: „Jeder Termin im Stundenplan ist mit höchstens einer Veranstaltung belegt."

(c)



bedeutet: „Jede Veranstaltung wird höchstens dreimal (pro Woche) angeboten."

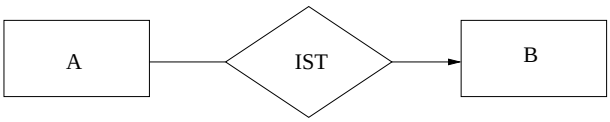
IST-Beziehung

Ein weiterer Bestandteil von ER-Modellen: Die IST-Beziehung[1]

Der spezielle Relationen-Typ IST definiert eine Spezialisierungs-Hierarchie.

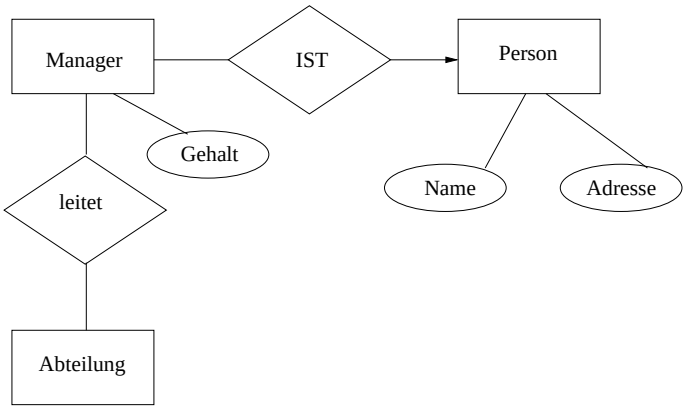
Graphische Darstellung:

[1] englisch: "is-a"



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 9.11. Ein umfangreiches ER-Modell, das einige Aspekte einer Fluggesellschaft modelliert, ist in Abbildung 9.6 dargestellt. In diesem ER-Modell wird u.a. folgendes 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, indem jedem Flugzeug genau ein Flugzeugtyp zuordnet wird. 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 (d.h. 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.

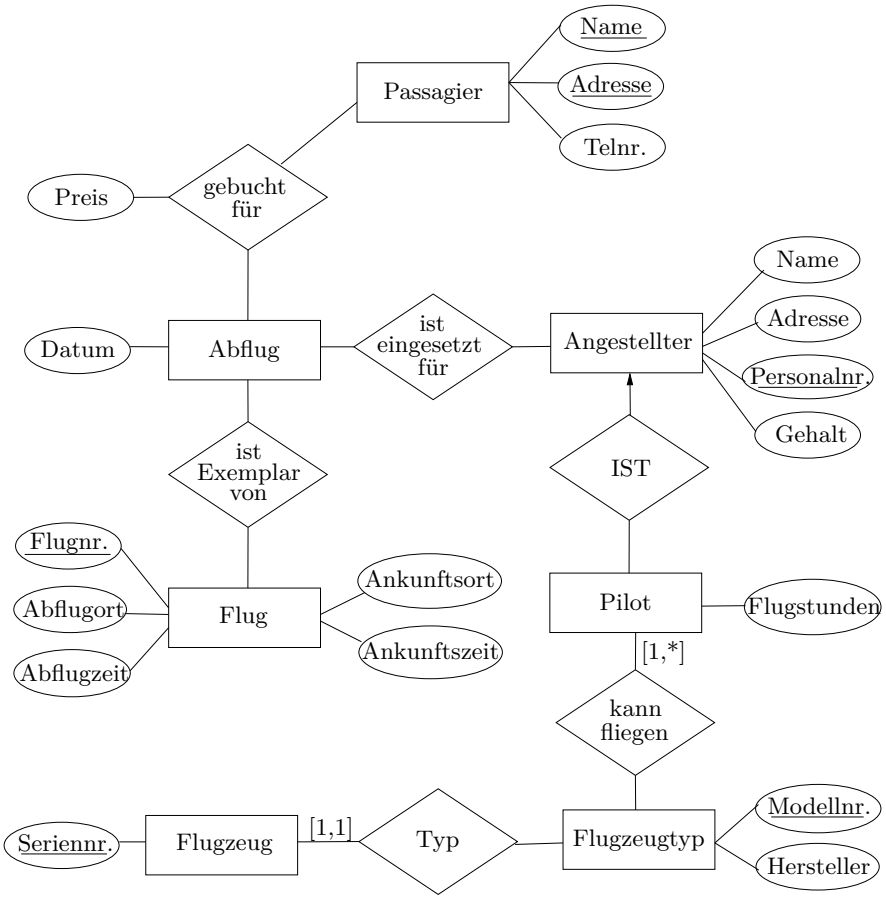


Abbildung 9.6: ER-Modell, das einige Aspekte einer Fluggesellschaft modelliert

9.3 Eine Fallstudie

In diesem Abschnitt 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.

Als konkretes Anwendungsbeispiel betrachten wir hier eine **Autowerkstatt**. Unser Ziel ist, die Auftragsabwicklung der Autowerkstatt zu modellieren. Dabei soll zum einen eine geeignete Datenbank entworfen werden; zum anderen sollen Abläufe in der Autowerkstatt analysiert und verbessert werden.

9.3.1 Datenbank-Entwurf: Autowerkstatt

Es sollen Daten zu Kunden, Aufträgen, Kraftfahrzeugen (kurz: KFZ) und KFZ-Typen gespeichert werden:

- Jeder **Kunde** hat einen Namen, besitzt Kraftfahrzeug(e) (kurz: KFZ) und erteilt ggf. einen oder mehrere Aufträge.

- Jeder **Auftrag** hat ein Eingangsdatum, betrifft ein KFZ und wird von einem oder mehreren Mechaniker(n) bearbeitet und benötigt Ersatzteile bestimmter Arten und Mengen (im Sinne von „Anzahlen“)
- Jedes **KFZ** hat eine Fahrgestellnummer und ein Baujahr und ist entweder ein PKW oder ein Motorrad. Bei PKWs sollen Informationen zur Farbe, bei Motorrädern Informationen zum Tuningsatz gespeichert werden.
- Jedes KFZ hat einen **Typ**. Jeder Mechaniker ist für einen oder mehrere Typen ausgebildet; Ersatzteile sind für bestimmte Typen verwendbar.

Zum Entwurf eines ER-Modells zum Verwalten dieser Daten, wählen wir folgende Entity- und Relationen-Typen.

Zentrale Entity-Typen:

- Kunde
- Auftrag
- KFZ
- KFZ-Typ.

Zentrale 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)

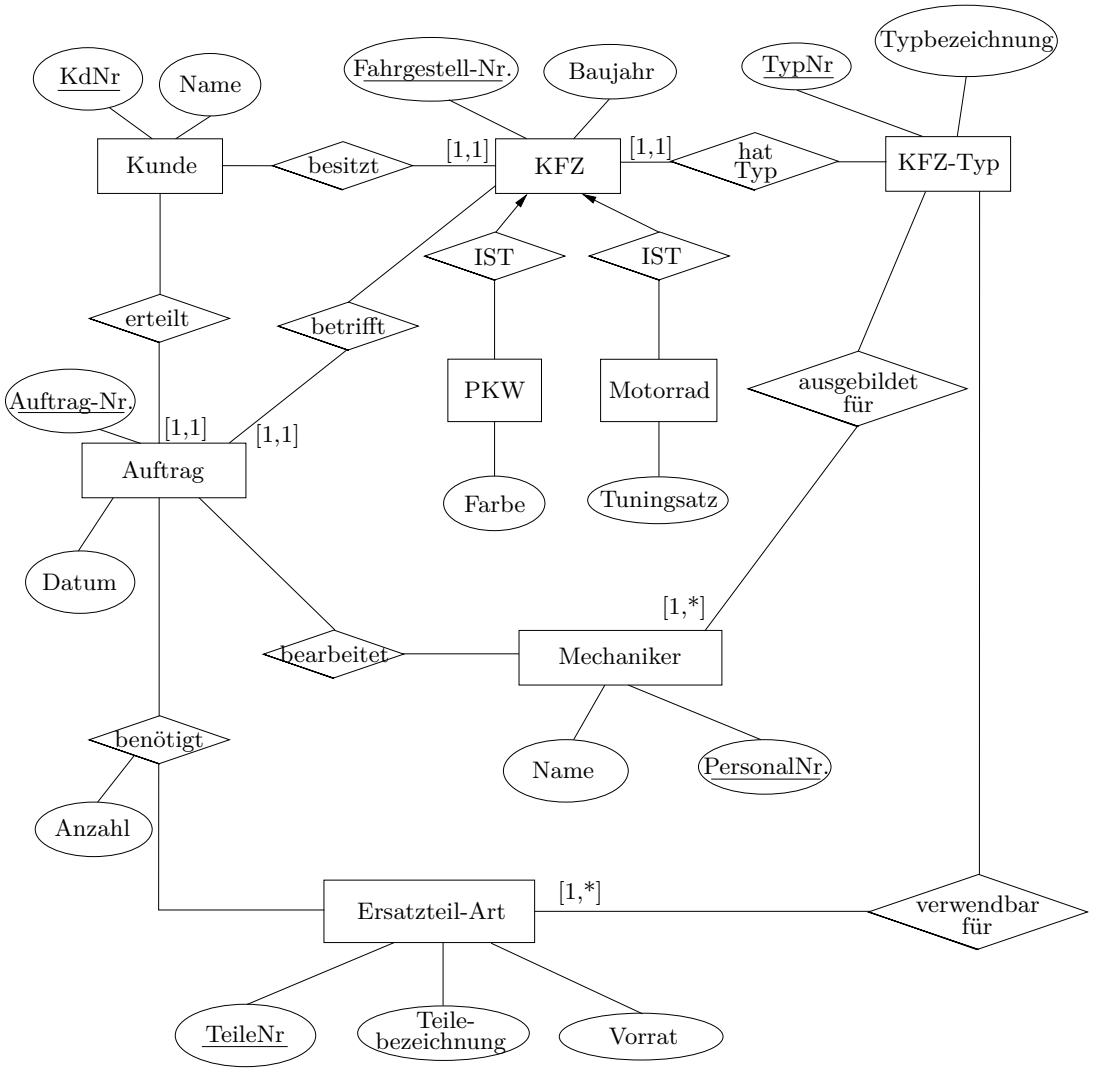
Für den letzten Relationen-Typ ist es sinnvoll einen weiteren *Entity-Typ* einzuführen:

- Mechaniker.

Des Weiteren benötigen wir noch *Relationen-Typen* zur Modellierung davon,

- welche Ersatzteile ein Auftrag benötigt,
- welche Ersatzteile für welchen KFZ-Typ geeignet sind,
- welcher Mechaniker welchen Auftrag bearbeitet.

Außerdem müssen wir eine Möglichkeit bereitstellen, KFZ in PKW und Motorräder einzuteilen. Dies wird durch das folgende ER-Modell gewährleistet:



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 Auftrag wird von genau einem Kunden erteilt.
- Jeder Mechaniker ist für mindestens einen KFZ-Typ ausgebildet.
- Jede Ersatzteil-Art ist für mindestens einen KFZ-Typ verwendbar.

9.3.2 Abläufe bei der Auftragserteilung

Eine Untersuchung der Geschäftsabläufe in der Autowerkstatt ergibt, dass jeder Auftrag folgende Stationen durchläuft:

(1) Der Auftrag wird erteilt.

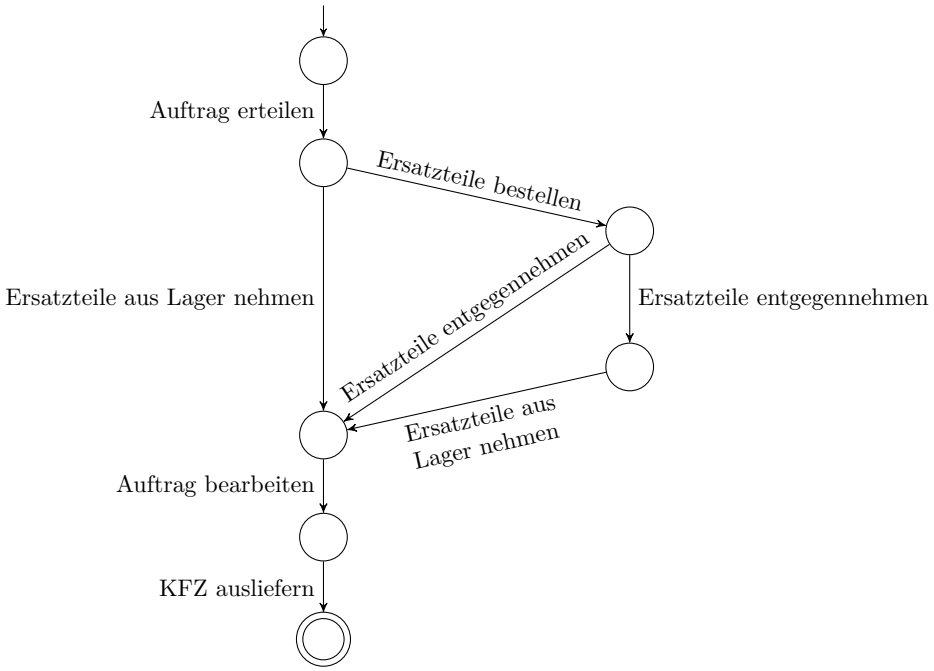
(2) Fehlende Ersatzteile werden bestellt und nach dem Eintreffen entgegengenommen.

(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 (d.h. als endlicher Automat):



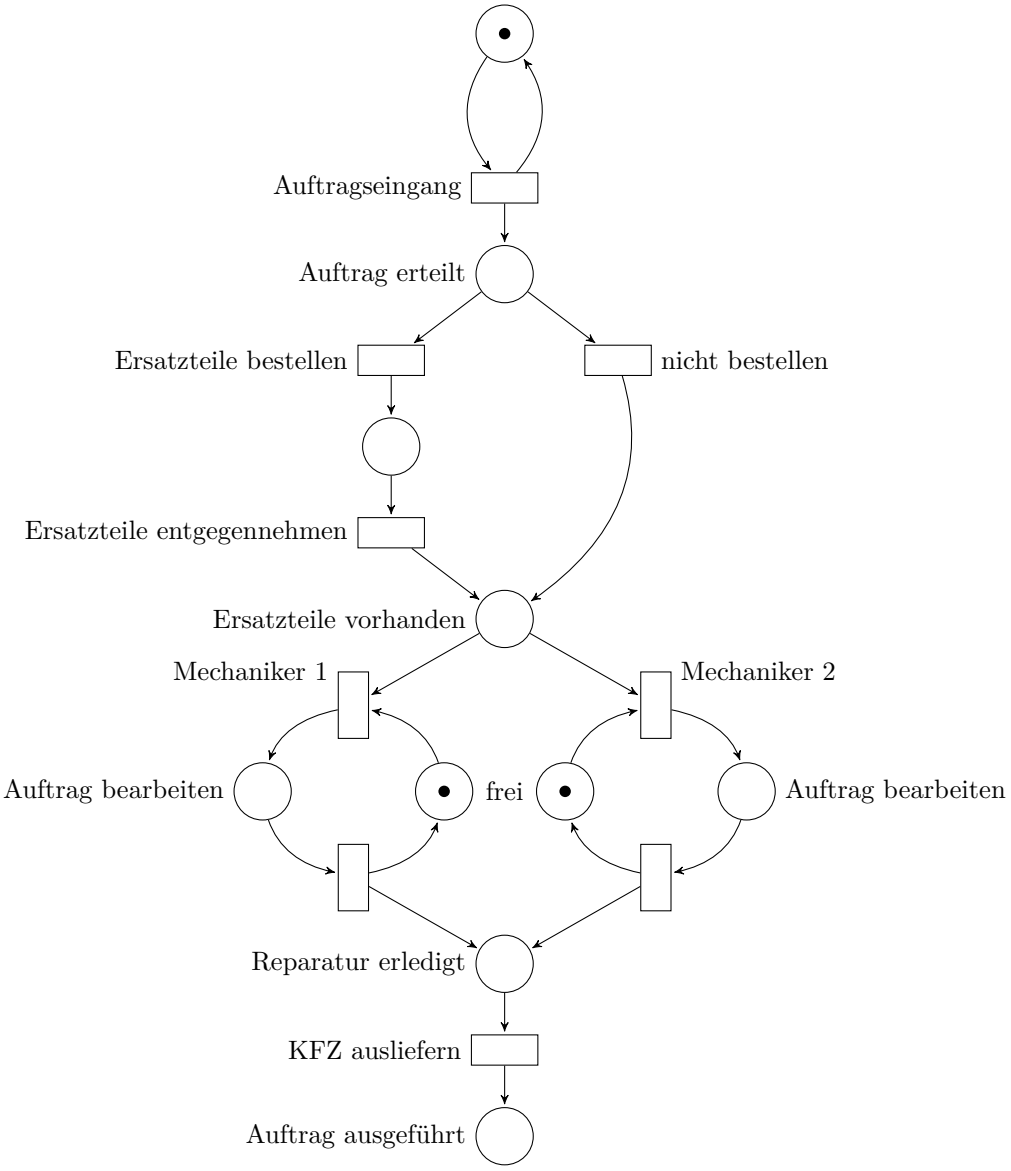
Einschränkungen dieses Modells:

- 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.

Modellierung der Auftragsbearbeitung durch ein Petri-Netz:

Ziel: Modelliere, wie mehrere Aufträge nebenläufig von zwei miteinander um Aufträge konkurrierenden Mechanikern bearbeitet werden.

Petri-Netz:

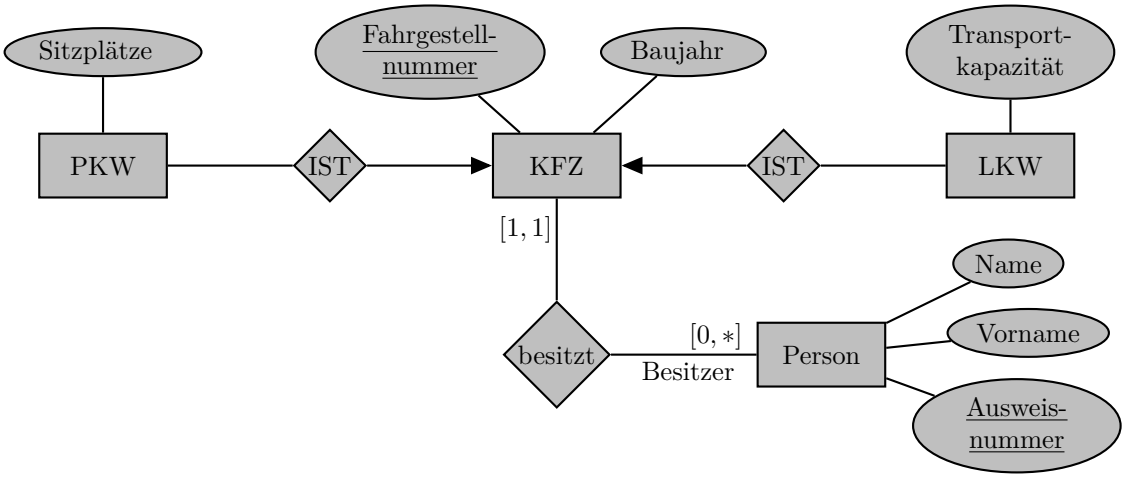


9.4 Literaturhinweise

Dieses Kapitel orientiert sich an den Kapiteln 6.3, 7.2 und 8.1 von [15]. Einen umfassenden Überblick zum Thema Petri-Netze gibt das Buch [24]. Eine Einführung in Methoden zum Datenbankentwurf und insbesondere ins Entity-Relationship-Modell wird in [11] gegeben.

9.5 Übungsaufgaben zu Kapitel 9

Aufgabe 9.1. 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 9.2. 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!

10 Beispielklausuren

Am Ende des Semesters findet die Modulabschlussprüfung Diskrete Modellierung in Form einer Klausur statt. Die schriftliche Prüfung dauert 120 Minuten.

Details zum Ablauf der Klausur:

- Grundsätzlich gelten die in der Ordnung Ihres Studiengangs festgelegten Regelungen. Dieses hier sind nur ergänzende Hinweise.
- Alle Klausurteilnehmer/innen müssen sich zu Beginn der Klausur durch (1) den Studierendenausweis und (2) die „Goethe-Card“ oder einen Lichtbildausweis ausweisen.
- Außer einem dokumentenechten Schreibstift sind keine weiteren Hilfsmittel zugelassen. (Insbesondere: Kein Vorlesungsskript, keine mitgebrachten Notizen, kein von Ihnen mitgebrachtes Papier, kein Taschenrechner, kein Handy. Bitte beachten Sie, dass ein während der Klausur eingeschaltetes Handy als Betrugsversuch gewertet wird.)
- Schreibpapier wird von uns bereitgestellt.
- Die Sitzordnung wird von uns festgelegt und kurz vor Beginn der Klausur bekanntgegeben.

Checkliste - zur Klausur müssen Sie mitbringen:

- einen dokumentenechten Schreibstift
- einen gültigen Lichtbildausweis (z.B. Ihre Goethe-Card oder Ihren Personalausweis)
- Ihren Studierendenausweis (... es sei denn, Sie sind als „Schülerstudent“ für die Veranstaltung angemeldet)

Durch die in den Übungen gesammelten Punkte kann ein Bonus für die Klausur erworben werden. Zur Benotung werden neben dem Klausurergebnis Bonuspunkte mit einem Maximalgewicht von 10% eingehen. Die Klausur ist bestanden, wenn mit dem Bonus mindestens 50% der in der Klausur erzielbaren Punkte erreicht werden.

Nachfolgend sind die Klausuren aus dem Wintersemester 2008/2009 und Sommersemester 2009 angehängt.

Goethe-Universität Frankfurt am Main
Institut für Informatik
Theorie komplexer Systeme
Prof. Dr. Nicole Schweikardt

3. März 2009

Diskrete Modellierung (WS 08/09)
Klausur (Modulabschlussprüfung)

Name: _____ Vorname: _____

Matrikelnummer: _____ Studiengang: _____

⇓ BITTE GENAU LESEN ⇓

- Außer einem dokumentenechten Schreibstift sind zu dieser Klausur keine weiteren Hilfsmittel erlaubt. Bitte beachten Sie, dass das Mitbringen nicht zugelassener Hilfsmittel eine Täuschung darstellt und zwangsläufig zum Nichtbestehen der Klausur führt.

  Insbesondere müssen Sie Ihre Handys vor Beginn der Klausur ausschalten.
- Bitte legen Sie Ihre Goethe-Card bzw. Ihren Studierendenausweis und einen gültigen Lichtbildausweis deutlich sichtbar an Ihren Platz, damit wir im Laufe der Klausur die Identität überprüfen können.
- Zur Bearbeitung der Aufgaben stehen Ihnen 120 Minuten zur Verfügung.
- Überprüfen Sie, ob Ihr Exemplar der Klausur aus insgesamt 16 durchnummerierten Blättern besteht.
- Bitte schreiben Sie Ihre Lösungen direkt an die dafür vorgesehene Stelle. Gegebenenfalls können Sie auch die Rückseiten und die beigefügten Zusatzblätter benutzen. Weitere Blätter sind auf Nachfrage erhältlich.
- Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies verlangt.
- Jedes Blatt der abgegebenen Lösung muss mit Namen, Vornamen und Matrikelnummer gekennzeichnet sein; andernfalls werden diese Blätter nicht gewertet.
- Schreiben Sie **nicht** mit Bleistift – mit Bleistift angefertigte Lösungen werden nicht gewertet.
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung.
- In der Klausur können maximal 100 Punkte erreicht werden. Die in den Übungsaufgaben im WS 08/09 erworbenen Bonuspunkte werden zu der in der Klausur erreichten Punktzahl hinzuaddiert. Werden insgesamt $z \geq 50$ Punkte erreicht, so ist die Prüfung bestanden. Die Noten verteilen sich wie folgt:

Note	z	Note	z	Note	z	Note
1:			$z \geq 90$	1,0	$90 > z \geq 85$	1,3
2:	$85 > z \geq 80$	1,7	$80 > z \geq 76$	2,0	$76 > z \geq 72$	2,3
3:	$72 > z \geq 67$	2,7	$67 > z \geq 63$	3,0	$63 > z \geq 59$	3,3
4:	$59 > z \geq 54$	3,7	$54 > z \geq 50$	4,0		

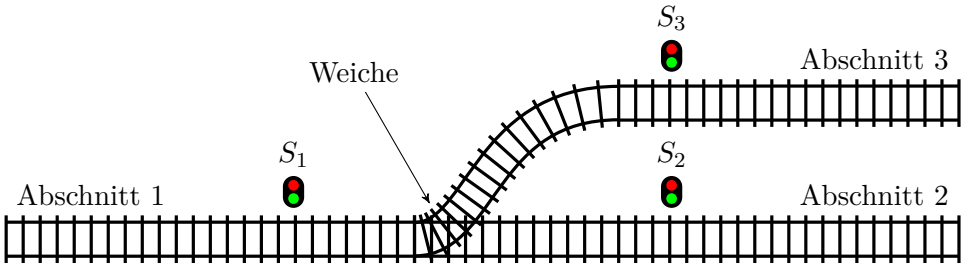
- Die Ergebnisse der Klausur und Termine zur Klausureinsicht werden spätestens am 10.4.2009 auf der zur Vorlesung gehörigen Internetseite (`www.informatik.uni-frankfurt.de/~tkshp/lehre/WS0809/DM/`) bekanntgegeben.

Aufgabe	1	2	3	4	5	6	Klausur	Bonus	Gesamt
maximale Punkte	21	21	12	11	15	20	100	10	110
erreichte Punkte									

Note:

Aufgabe 1: **(21 Punkte)**

(a) Es sei die folgende Weiche einer Bahnanlage gegeben:



Die drei Signale S_1, S_2 und S_3 geben an, ob ein Zug vom entsprechenden Gleisabschnitt über die Weiche fahren darf: Leuchtet S_i grün, dann darf ein Zug vom Gleisabschnitt i aus über die Weiche fahren.
Die Weiche ist entweder auf „geradeaus fahren“ oder „abbiegen“ eingestellt. Wenn sie auf „geradeaus fahren“ eingestellt ist, dann fahren Züge, die vom Gleisabschnitt 1 aus über die Weiche fahren, auf den Gleisabschnitt 2 und umgekehrt. Ansonsten fahren Züge, die vom Gleisabschnitt 1 aus über die Weiche fahren, auf den Gleisabschnitt 3 und umgekehrt.

Mit Hilfe der folgenden atomaren Aussagen lassen sich nun einfache Anforderungen an die Weichenanlage formulieren:

- X_1: S_1 leuchtet grün.
- X_2: S_2 leuchtet grün.
- X_3: S_3 leuchtet grün.
- X_W: Die Weiche ist auf „geradeaus fahren“ eingestellt.

Beispielsweise besagt die aussagenlogische Formel $(\neg X_\mathrm{W} \wedge (X_1 \wedge \neg X_3))$, dass die Weiche auf „abbiegen“ eingestellt ist, S_1 grün leuchtet und S_3 rot leuchtet.

Geben Sie aussagenlogische Formeln an, die nur die Variablen X_1, X_2, X_3 und X_W benutzen und Folgendes aussagen:

- Wenn S_2 grün leuchtet, dann ist die Weiche auf „geradeaus fahren“ eingestellt, und wenn S_3 grün leuchtet, dann ist die Weiche auf „abbiegen“ eingestellt. (2 Pkte)

- Höchstens eines der drei Signale S_1, S_2, S_3 leuchtet grün. D.h. wenn S_1 grün leuchtet, dann leuchten S_2 und S_3 nicht grün und analog für die anderen Signale. Beachten Sie, dass auch der Fall eintreten kann, bei dem keines der drei Signale grün leuchtet. (3 Pkte)

(b) Geben Sie für jede der folgenden aussagenlogischen Formeln an, ob sie erfüllbar und/oder allgemeingültig ist. (Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.) (7 Pkte)

Geben Sie außerdem folgendes für jede Formel an: Falls die Formel erfüllbar ist, geben Sie eine zur Formel passende Belegung an, die die Formel erfüllt. Falls die Formel nicht allgemeingültig ist, geben Sie eine zur Formel passende Belegung an, die die Formel *nicht* erfüllt.

- $\varphi \;=\; \Big(\; \big((X_1 \vee X_2) \leftrightarrow X_3\big) \;\wedge\; (X_1 \wedge \neg X_3) \;\Big)$

erfüllbar: ☐ ja ☐ nein
allgemeingültig: ☐ ja ☐ nein

Falls φ erfüllbar ist, geben Sie hier eine zu φ passende Belegung an, die φ erfüllt:

Falls φ nicht allgemeingültig ist, geben Sie hier eine zu φ passende Belegung an, die φ nicht erfüllt:

- $\psi \;=\; \Big(\; \big(X_1 \;\wedge\; (X_1 \rightarrow (X_2 \vee X_3))\big) \;\rightarrow\; X_3 \;\Big)$

erfüllbar: ☐ ja ☐ nein
allgemeingültig: ☐ ja ☐ nein

Falls ψ erfüllbar ist, geben Sie hier eine zu ψ passende Belegung an, die ψ erfüllt:

Falls ψ nicht allgemeingültig ist, geben Sie hier eine zu ψ passende Belegung an, die ψ nicht erfüllt:

(c) Welche der folgenden Formeln sind in Negationsnormalform (NNF), disjunktiver Normalform (DNF) und/oder konjunktiver Normalform (KNF)? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen halben Punkt, für jedes **falsche Kreuz** wird **ein halber Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

	$\big((\neg X_1 \wedge (X_2 \vee \neg X_3)) \vee \neg X_2\big)$		$\big((X_1 \vee \neg X_2) \wedge ((\neg X_1 \vee X_3) \vee X_4)\big)$	
in NNF?	☐ ja	☐ nein	☐ ja	☐ nein
in DNF?	☐ ja	☐ nein	☐ ja	☐ nein
in KNF?	☐ ja	☐ nein	☐ ja	☐ nein

(d) Geben Sie eine zur Formel (6 Pkte)

$$\varphi \;:=\; \big((X_2 \to X_1) \vee (X_2 \wedge \neg X_3)\big)$$

äquivalente aussagenlogische Formel in konjunktiver Normalform an. Geben Sie auch Ihren Lösungsweg an.

Aufgabe 2: **(21 Punkte)**

(a) Sei $G = (V, E)$ der gerichtete Graph mit der folgenden Adjazenzliste:

Knoten	Nachfolger
a	(b, e)
b	(d)
c	(a, d)
d	(a, c, e)
e	(b, c)

(i) Geben Sie die graphische Darstellung von G an. Geben Sie außerdem die Kantenmenge E von G an und repräsentieren Sie G durch eine Adjazenzmatrix.

graphische Darstellung: (2 Pkte)

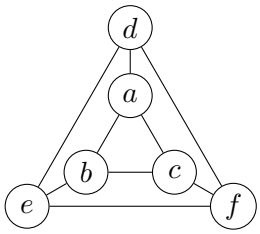
$E =$ (1 Pkt)

Adjazenzmatrix: (1 Pkt)

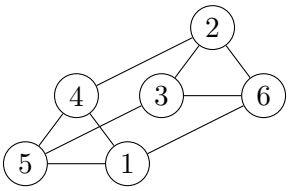
(ii) Wie groß ist der Eingangsgrad des Knotens a in G? (1 Pkt)

(iii) Geben Sie einen Kreis in G an, der durch den Knoten a verläuft und *nicht* einfach ist: (1 Pkt)

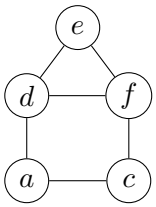
(b) Betrachten Sie die folgenden ungerichteten Graphen G_1, G_2, G_3, G_4:



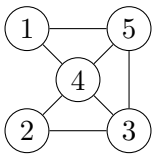
G_1



G_2



G_3



G_4

(i) Welche der folgenden Aussagen sind wahr, welche falsch? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

- G_4 ist ein Teilgraph von G_2. ☐ wahr ☐ falsch
- G_3 ist ein induzierter Teilgraph von G_1. ☐ wahr ☐ falsch
- G_3 und G_4 sind isomorph. ☐ wahr ☐ falsch

(ii) Geben Sie einen Isomorphismus von G_1 nach G_2 an. (2 Pkte)

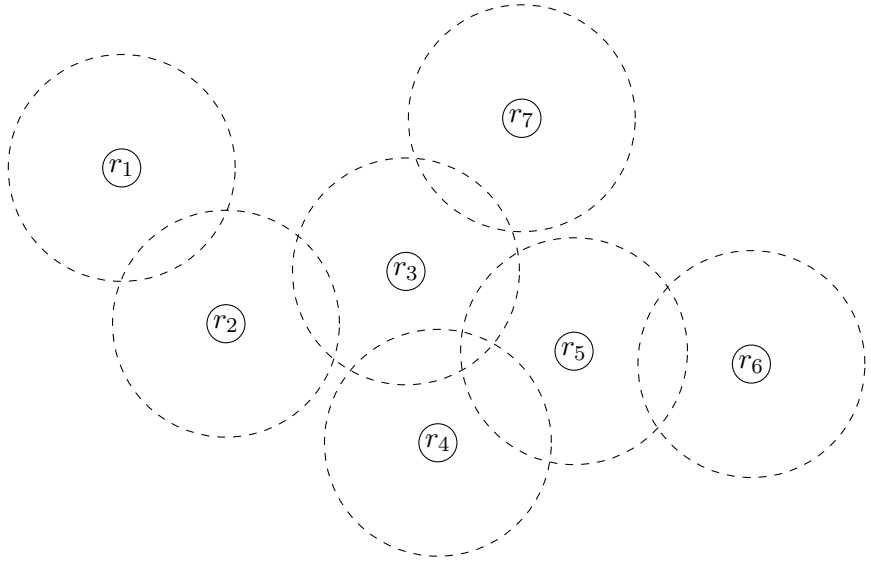
(c) Welche der folgenden Aussagen über Bäume sind wahr, welche falsch? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

- Sind x und y Knoten in einem ungerichteten Baum B, so gibt es in B genau einen einfachen Weg von x nach y. ☐ wahr ☐ falsch
- Es gibt einen Baum B und zwei Knoten x und y in B, so dass es in B mindestens zwei verschiedene einfache Wege von x nach y gibt. ☐ wahr ☐ falsch
- Ist $B = (V, E)$ ein Baum, so gilt $|E| = |V| + 1$. ☐ wahr ☐ falsch

(d) Es seien Radiostationen $r_1, \ldots, r_7$ gegeben. Jeder Radiostation soll eine von drei Frequenzen f_1, f_2, f_3 zugeordnet werden. Radiostationen, die zu dicht beieinander liegen, dürfen allerdings nicht die gleichen Frequenzen zugewiesen bekommen.

Das folgende Diagramm stellt die Lage der einzelnen Radiostationen dar.



Um jede Station ist ein gestrichelter Kreis eingezeichnet. Schneiden sich die Kreise von zwei Radiostationen r_i und r_j, so liegen r_i und r_j zu dicht beieinander und dürfen nicht die gleiche Frequenz zugeordnet bekommen. Wir sagen auch, dass r_i und r_j in Konflikt zueinander stehen. Zum Beispiel stehen r_1 und r_2 in Konflikt zueinander, r_1 und r_3 aber nicht. r_1 und r_2 dürfen also nicht die gleiche Frequenz zugeordnet bekommen, wohingegen r_1 und r_3 auf der gleichen Frequenz senden dürfen.

(i) Geben Sie den Konfliktgraph G an (in graphischer Darstellung). (2 Pkte)

(ii) Weisen Sie jeder der Radiostationen $r_1, \ldots, r_7$ genau eine der drei Frequenzen f_1, f_2, f_3 zu, so dass Radiostationen, die zueinander in Konflikt stehen, nicht der gleichen Frequenz zugeordnet sind. (3 Pkte)

D.h.: Sei V die Menge der Knoten des Konfliktgraphen G aus (i). Geben Sie eine konfliktfreie Knotenmarkierung $m\colon V \to \{1, 2, 3\}$ an.

(iii) Wie viele Frequenzen werden für die Radiostationen mindestens benötigt? Begründen Sie Ihre Antwort. (2 Pkte)

Aufgabe 3: **(12 Punkte)**

(a) Sei $\sigma := \{\dot{S}, \dot{G}\}$ eine Signatur mit 2-stelligen Relationssymbolen $\dot{S}, \dot{G}$. Sei $\mathfrak{A} = (A, \dot{S}^{\mathfrak{A}}, \dot{G}^{\mathfrak{A}})$ die σ-Struktur, in der

- A die Menge der Spieler eines Turniers ist,
- $\dot{S}^{\mathfrak{A}}$ alle Tupel $(x, y) \in A \times A$ enthält, so dass x gegen y gespielt hat und
- $\dot{G}^{\mathfrak{A}}$ alle Tupel $(x, y) \in A \times A$ enthält, so dass x gegen y gewonnen hat.

(i) Geben Sie eine Formel φ der Logik erster Stufe über der Signatur σ an, die in $\mathfrak{A}$ aussagt, dass es mindestens zwei *verschiedene* Spieler gibt, die noch nicht gegeneinander gespielt haben. (2 Pkte)

$\varphi :=$

(ii) Geben Sie eine Formel ψ der Logik erster Stufe über der Signatur σ an, die in $\mathfrak{A}$ aussagt, dass es keinen Spieler gibt, der in allen Spielen, an denen er teilgenommen hat, gewonnen hat. (2 Pkte)

$\psi :=$

(iii) Beschreiben Sie umgangssprachlich, was die folgende Formel in $\mathfrak{A}$ aussagt: (2 Pkte)

$$\forall x \forall y \Big(\dot{S}(x, y) \rightarrow (\dot{G}(x, y) \leftrightarrow \neg \dot{G}(y, x)) \Big)$$

(b) Sei $\sigma := \{\dot{E}\}$ eine Signatur mit einem 2-stelligen Relationssymbol $\dot{E}$. Geben Sie für die Formel (6 Pkte)

$$\varphi(x,y) \quad := \quad \Big(\dot{E}(x,y) \wedge \exists z(\dot{E}(y,z) \wedge \dot{E}(z,x)) \Big)$$

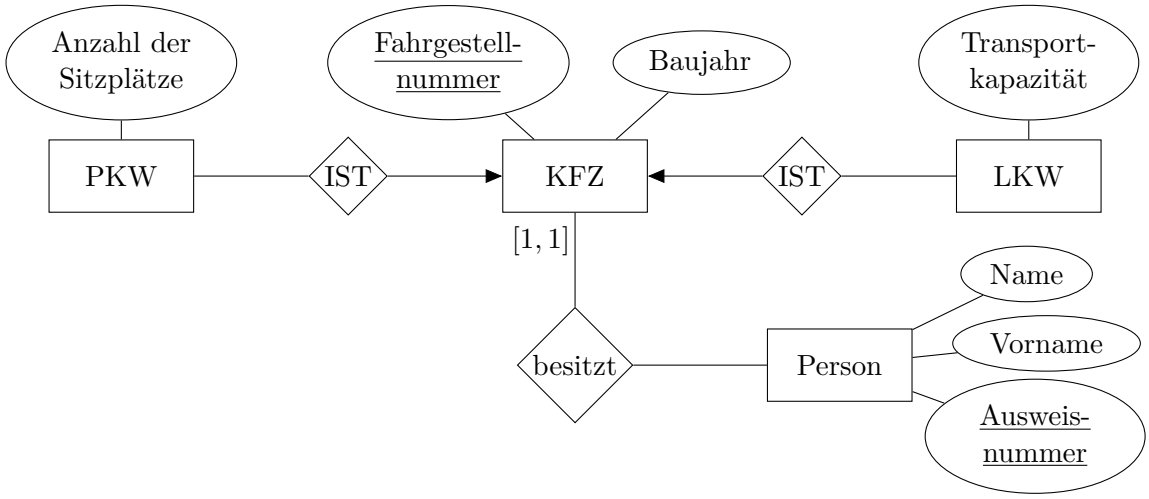
zwei σ-Interpretationen $\mathcal{I}_1, \mathcal{I}_2$ an, so dass $\mathcal{I}_1$ die Formel φ erfüllt und $\mathcal{I}_2$ die Formel φ *nicht* erfüllt.

$\mathcal{I}_1 :=$

$\mathcal{I}_2 :=$

Aufgabe 4: **(11 Punkte)**

(a) Es sei folgendes Entity-Relationship-Modell gegeben: (4 Pkte)



Welche der folgenden Aussagen sind im Sinne des oben angegebenen Entity-Relationship-Modells wahr, welche nicht?

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

(i) Es ist möglich, dass es ein KFZ ohne Besitzer gibt. ☐ wahr ☐ falsch

(ii) Jedes KFZ hat genau einen Besitzer. ☐ wahr ☐ falsch

(iii) Es ist möglich, dass es ein PKW (z.B. mit Baujahr 2002) und einen LKW (z.B. mit Baujahr 1997) gibt, die dieselbe Fahrgestellnummer besitzen. ☐ wahr ☐ falsch

(iv) Es ist möglich, dass es zwei Personen mit demselben (Nach)namen und demselben Vornamen gibt. ☐ wahr ☐ falsch

(b) Sei $G = (T, N, S, P)$ die kontextfreie Grammatik mit

- der Menge der Terminalsymbole $T = \{a, b, c, d\}$
- der Menge der Nichtterminalsymbole $N = \{S, X, Y\}$
- dem Startsymbol S
- der Menge der Produktionen $P = \{ S \to XcYd,\ X \to aY,\ Y \to aY,\ Y \to bY,\ Y \to \varepsilon \}$

(i) Geben Sie für jedes der folgenden Wörter an, ob es zu der von G erzeugten Sprache $L(G)$ gehört. Geben Sie außerdem für jedes Wort, das zur Sprache $L(G)$ gehört, einen Ableitungsbaum an. (4 Pkte)

- *abcbad*

gehört zu $L(G)$? ☐ ja ☐ nein

Falls das Wort zu $L(G)$ gehört, geben Sie hier einen Ableitungsbaum an:

- *bacabad*

gehört zu $L(G)$? ☐ ja ☐ nein

Falls das Wort zu $L(G)$ gehört, geben Sie hier einen Ableitungsbaum an:

(ii) Geben Sie eine (mathematische oder umgangssprachliche) Beschreibung der Sprache $L(G)$ an, die von der oben angegebenen Grammatik G erzeugt wird. (3 Pkte)

Aufgabe 5: **(15 Punkte)**

Die Sprache SFR über dem Alphabet $\Sigma := \{a, b, \varnothing, \sim, |, \cdot, (,)\}$ sei wie folgt rekursiv definiert:

Basisregeln:
(B1) Das Symbol $\varnothing$ ist in SFR.
(B2) Die Symbole a und b sind in SFR.

Rekursive Regeln:
(R1) Ist $w \in$ SFR, so ist auch $\sim w$ in SFR.
(R2) Sind w_1 und w_2 in SFR, so sind auch $(w_1 \,|\, w_2)$ und $(w_1 \cdot w_2)$ in SFR.

(a) Welche der folgenden Wörter gehören zur Sprache SFR, welche nicht? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

Wort	... liegt in SFR?	
$((\sim\varnothing \cdot a) \cdot (\sim\varnothing \cdot b))$	☐ ja	☐ nein
$\sim((\sim\varnothing \cdot a) \cdot b) \,|\, b$	☐ ja	☐ nein
$(\sim(\sim\varnothing \cdot a) \mid ((b \cdot a) \cdot \varnothing))$	☐ ja	☐ nein

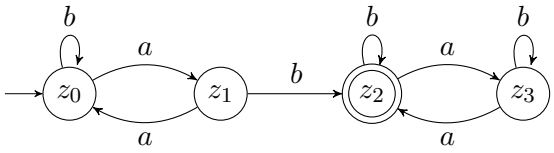
(b) Für jedes Wort $w \in \Sigma^*$ bezeichne $s(w)$ die Anzahl der Vorkommen der Symbole $\varnothing, a, b$ in w und $o(w)$ die Anzahl der Symbole $\sim$, $|$ und $\cdot$ in w. Zum Beispiel gilt für $w = ((a \cdot \sim b) \,|\, \sim a)$, dass $s(w) = 3$ und $o(w) = 4$. (8 Pkte)

Beweisen Sie durch Induktion, dass für alle Wörter $w \in$ SFR gilt: $s(w) \leq o(w) + 1$.

(c) Geben Sie eine kontextfreie Grammatik G an, die genau die Sprache SFR erzeugt. (4 Pkte)

Aufgabe 6: **(20 Punkte)**

(a) Sei A der folgende deterministische endliche Automat: (3 Pkte)



Welche der folgenden Wörter werden von A akzeptiert, welche nicht?

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

Wort	... wird akzeptiert?	
abbabb	☐ ja	☐ nein
babaab	☐ ja	☐ nein
aababaa	☐ ja	☐ nein

(b) Es sei R der folgende reguläre Ausdruck über dem Alphabet $\Sigma := \{0, 1, \ldots, 9, .\}$:

$$(\varepsilon \mid (1 \mid \cdots \mid 9)(0 \mid 1 \mid \cdots \mid 9)^*) \; . \; (0 \mid 1 \mid \cdots \mid 9)(0 \mid 1 \mid \cdots \mid 9)^*$$

(i) Welche der folgenden Wörter liegen in der von R definierten Sprache $L(R)$, welche nicht? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

Wort	... liegt in $L(R)$?	
.075	☐ ja	☐ nein
12	☐ ja	☐ nein
012.56637	☐ ja	☐ nein

(ii) Geben Sie eine (mathematische oder umgangssprachliche) Beschreibung der Sprache $L(R)$ an, die von R definiert wird. (3 Pkte)

(c) Geben Sie die graphische Darstellung eines nicht-deterministischen endlichen Automaten an, der genau diejenigen Wörter über dem Alphabet $\{a, b\}$ akzeptiert, die die Teilwörter ab und ba enthalten, so dass ba mindestens einmal hinter einem Vorkommen von ab steht. (4 Pkte)

D.h. der Automat soll genau die Wörter $w \in \{a, b\}^*$ akzeptieren, für die es Wörter $x, y, z \in \{a, b\}^*$ gibt, so dass $w = xabybaz$.

(d) Betrachten Sie das Alphabet $\Sigma := \{a, b\}$ und zeigen Sie, dass die Sprache (7 Pkte)

$$L := \{w \in \{a, b\}^* : w \text{ enthält genau so viele } a\text{s wie } b\text{s}\}$$

nicht regulär ist.

Hinweis: Verwenden Sie das Pumping-Lemma aus der Vorlesung:

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 vier Bedingungen erfüllt sind:

(i) $x = uvw$
(ii) $|uv| \leq z$
(iii) $|v| \geq 1$
(iv) für jedes $i \in \mathbb{N}$ gilt: $uv^iw \in L$.
(d.h.: $uw \in L$, $uvw \in L$, $uvvw \in L$, $uvvvw \in L$, ...)

Goethe-Universität Frankfurt am Main
Institut für Informatik
Theorie komplexer Systeme
Prof. Dr. Nicole Schweikardt

29. September 2009

Diskrete Modellierung (SS 09)
Klausur (Modulabschlussprüfung)

Name: _____ Vorname: _____

Matrikelnummer: _____ Studiengang: _____

⇓ **BITTE GENAU LESEN** ⇓

- Außer einem dokumentenechten Schreibstift sind zu dieser Klausur keine weiteren Hilfsmittel erlaubt. Bitte beachten Sie, dass das Mitbringen nicht zugelassener Hilfsmittel eine Täuschung darstellt und zwangsläufig zum Nichtbestehen der Klausur führt.

  Insbesondere müssen Sie Ihre Handys vor Beginn der Klausur ausschalten.
- Bitte legen Sie Ihre Goethe-Card bzw. Ihren Studierendenausweis und einen gültigen Lichtbildausweis deutlich sichtbar an Ihren Platz, damit wir im Laufe der Klausur die Identität überprüfen können.
- Zur Bearbeitung der Aufgaben stehen Ihnen 120 Minuten zur Verfügung.
- Überprüfen Sie, ob Ihr Exemplar der Klausur alle von 2 bis 23 durchnummerierten Seiten enthält.
- Bitte schreiben Sie Ihre Lösungen direkt an die dafür vorgesehene Stelle. Gegebenenfalls können Sie auch die beigefügten Zusatzblätter benutzen. Weitere Blätter sind auf Nachfrage erhältlich.
- Begründungen sind nur dann notwendig, wenn die Aufgabenformulierung dies verlangt.
- Jedes Blatt der abgegebenen Lösung muss mit Namen, Vornamen und Matrikelnummer gekennzeichnet sein; andernfalls werden diese Blätter nicht gewertet.
- Schreiben Sie **nicht** mit Bleistift – mit Bleistift angefertigte Lösungen werden nicht gewertet.
- Werden zu einer Aufgabe zwei oder mehr Lösungen angegeben, so gilt die Aufgabe als nicht gelöst. Entscheiden Sie sich also immer für **eine** Lösung.
- In der Klausur können maximal 100 Punkte erreicht werden. Die in den Übungsaufgaben im WS 08/09 erworbenen Bonuspunkte werden zu der in der Klausur erreichten Punktzahl hinzuaddiert. Werden insgesamt $z \geq 50$ Punkte erreicht, so ist die Prüfung bestanden. Die Noten verteilen sich wie folgt:

Note	z	Note	z	Note	z	Note
1:			$z \geq 90$	1,0	$90 > z \geq 85$	1,3
2:	$85 > z \geq 80$	1,7	$80 > z \geq 76$	2,0	$76 > z \geq 72$	2,3
3:	$72 > z \geq 67$	2,7	$67 > z \geq 63$	3,0	$63 > z \geq 59$	3,3
4:	$59 > z \geq 54$	3,7	$54 > z \geq 50$	4,0		

- Die Ergebnisse der Klausur und Termine zur Klausureinsicht werden spätestens am 13.10.2009 auf der zur Vorlesung gehörigen Internetseite (`www.informatik.uni-frankfurt.de/~tkshp/lehre/WS0809/DM/`) bekanntgegeben.

Aufgabe	1	2	3	4	5	6	Klausur	Bonus	Gesamt
maximale Punkte	22	23	12	15	8	20	100	10	110
erreichte Punkte									

Note:

Aufgabe 1: **(22 Punkte)**

(a) Betrachten Sie die folgende Aussage: (2 Pkte)

> Wenn mein Wecker nicht kaputt geht, komme ich morgen pünktlich zur Vorlesung, falls die Tram nach Fahrplan fährt.

Formalisieren Sie die Aussage durch eine aussagenlogische Formel, in der Sie die atomaren Aussagen K (der Wecker geht kaputt), P (ich komme pünktlich zur Vorlesung) und F (die Tram fährt nach Fahrplan) verwenden.

(b) Bekanntlich sagt nicht jedermann die Wahrheit. Markus, Otto und Paul geben folgende Auskunft: (4 Pkte)

1) Markus sagt: „Otto lügt."
2) Otto sagt: „Paul lügt."
3) Paul sagt: „Otto und Markus lügen."

Dieser Sachverhalt wird mit Hilfe der atomaren Aussagen M (Markus sagt die Wahrheit), O (Otto sagt die Wahrheit) und P (Paul sagt die Wahrheit) durch die drei Formeln

$$\varphi_1 := (M \leftrightarrow \neg O), \quad \varphi_2 := (O \leftrightarrow \neg P) \text{ und } \varphi_3 := (P \leftrightarrow (\neg O \wedge \neg M))$$

formalisiert.

Aber wer sagt die Wahrheit und wer lügt? Gibt es eine Belegung der aussagenlogischen Variablen M, P und O, derart dass das System aus den Aussagen 1) - 3) widerspruchsfrei ist? Geben Sie Ihren Lösungsweg an.

(c) Geben Sie für jede der folgenden aussagenlogischen Formeln an, ob sie erfüllbar und/oder allgemeingültig ist. (Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.) (7 Pkte)

Geben Sie außerdem folgendes für jede Formel an: Falls die Formel erfüllbar ist, geben Sie eine zur Formel passende Belegung an, die die Formel erfüllt. Falls die Formel nicht allgemeingültig ist, geben Sie eine zur Formel passende Belegung an, die die Formel *nicht* erfüllt.

- $\varphi \;=\; \Big(\; ((X_1 \vee X_2) \leftrightarrow \neg X_3) \;\wedge\; (X_1 \wedge \neg X_3) \;\Big)$

erfüllbar: ☐ ja ☐ nein
allgemeingültig: ☐ ja ☐ nein

Falls φ erfüllbar ist, geben Sie hier eine zu φ passende Belegung an, die φ erfüllt:

Falls φ nicht allgemeingültig ist, geben Sie hier eine zu φ passende Belegung an, die φ nicht erfüllt:

- $\psi \;=\; \Big(\; \big((X_1 \;\wedge\; X_2) \leftrightarrow X_3\big) \;\vee\; (X_1 \vee X_3) \;\Big)$

erfüllbar: ☐ ja ☐ nein
allgemeingültig: ☐ ja ☐ nein

Falls ψ erfüllbar ist, geben Sie hier eine zu ψ passende Belegung an, die ψ erfüllt:

Falls ψ nicht allgemeingültig ist, geben Sie hier eine zu ψ passende Belegung an, die ψ nicht erfüllt:

(d) Geben Sie eine präzise Definition für die logische Äquivalenz zweier aussagenlogischer Formeln an. (2 Pkte)

Definition: (logische Äquivalenz)

(e) Seien φ, ψ und χ beliebige aussagenlogische Formeln. Gelten folgende Äquivalenzen ? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

$\psi \equiv (\psi \vee \varphi)$	☐ ja	☐ nein
$(\varphi \vee (\psi \wedge \chi)) \equiv ((\varphi \wedge \psi) \vee (\varphi \wedge \chi))$	☐ ja	☐ nein
$(\psi \leftrightarrow \varphi) \equiv ((\neg\varphi \vee \psi) \wedge (\psi \rightarrow \varphi))$	☐ ja	☐ nein

(f) (4 Pkte)

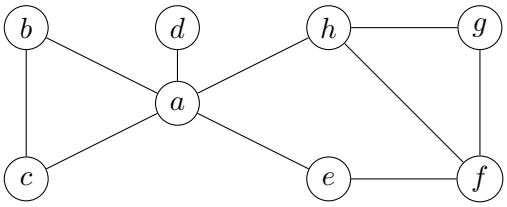
Geben Sie eine zur Formel

$$\varphi := ((\neg X_1 \rightarrow \mathbf{0}) \rightarrow ((X_1 \vee \neg X_2) \wedge X_3))$$

äquivalente Formel in Negationsnormalform an. Geben Sie auch Ihren Lösungsweg an.

Aufgabe 2: **(23 Punkte)**

(a) Sei $G = (V, E)$ der folgende Graph:



(i) Geben Sie die Knotenmenge V und die Kantenmenge E von G an. Repräsentieren Sie G außerdem durch eine Adjazenzliste.

$V =$ (1 Pkt)

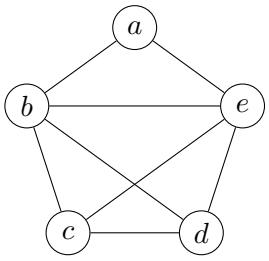
$E =$ (1 Pkt)

Adjazenzliste: (1 Pkt)

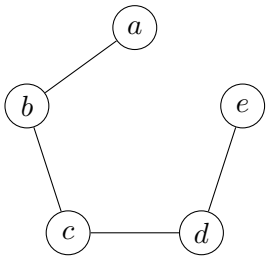
(ii) Geben Sie einen einfachen Weg der Länge 3 vom Knoten d zum Knoten g an: (1 Pkt)

(iii) Wie viele verschiedene einfache Kreise enthalten den Knoten a als Start- und Endknoten? (1 Pkt)

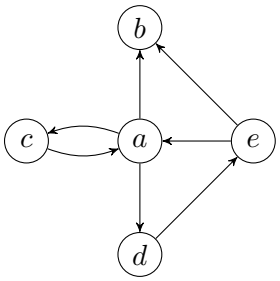
(b) Betrachten Sie die folgenden ungerichteten Graphen G_1, G_2 und den gerichteten Graphen G_3:



G_1



G_2



G_3

Welche der folgenden Aussagen sind wahr, welche falsch? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

- G_1 besitzt einen Eulerkreis. ☐ wahr ☐ falsch
- G_2 ist ein Spannbaum von G_1. ☐ wahr ☐ falsch
- G_3 ist stark zusammenhängend. ☐ wahr ☐ falsch

(c) Welche der folgenden Aussagen über Binärbäume sind wahr, welche falsch? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

- In jedem vollständigen Binärbaum ist die Anzahl der Kanten genau doppelt so groß wie die Anzahl der Knoten. ☐ wahr ☐ falsch
- Es existiert ein vollständiger Binärbaum B, der einen Knoten v enthält mit Ein-Grad$_B(v)$ = Aus-Grad$_B(v) = 0$. ☐ wahr ☐ falsch
- In jedem Binärbaum gibt es einen einfachen Weg von der Wurzel zu einem Blatt. ☐ wahr ☐ falsch

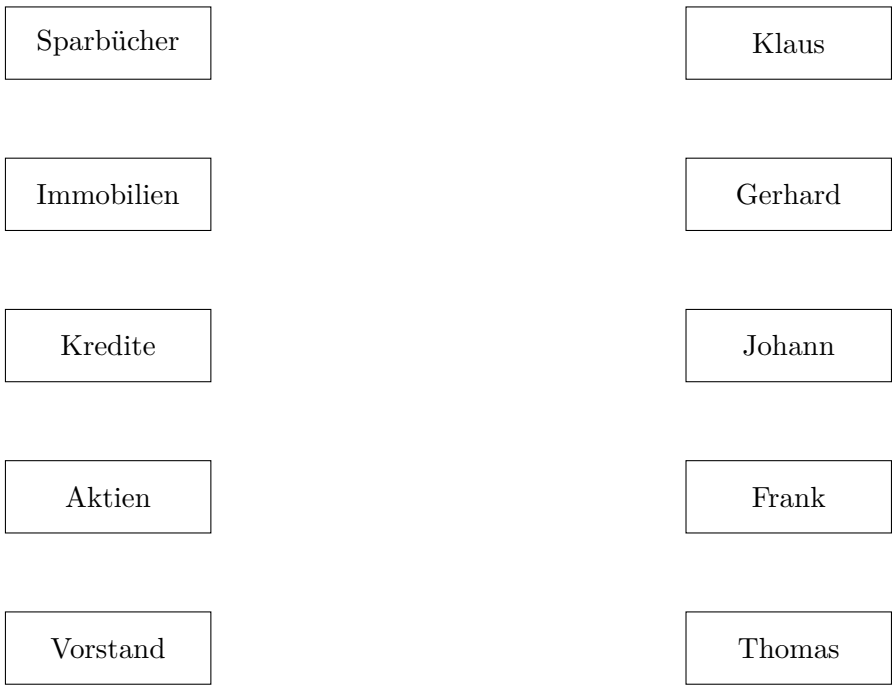
(d) Die "Hyper Real Finance" Bank hat die Finanzkrise überstanden. Sie schreibt Stellenangebote aus: In den vier Abteilungen *Sparbücher, Immobilien, Kredite* und *Aktien* ist jeweils eine Position zu besetzen. Außerdem ist ein Job im *Vorstand* frei. Passend dazu treffen fünf Bewerbungen ein. Die Bewerber sind allerdings recht eigenwillig, jeder von ihnen hat starke Abneigungen gegen bestimmte Abteilungen, in denen er auf keinen Fall arbeiten will. Im Einzelnen ergeben sich folgende Zu- und Abneigungen:

- Alle Bewerber können sich vorstellen, in den Vorstand zu gehen.
- Andererseits will keiner außer Klaus mit den Sparbüchern zu tun haben.
- Klaus wiederum möchte weder in der Kredit- noch in der Aktienabteilung arbeiten.
- Gerhard möchte sich nicht mit den Immobilien beschäftigen.
- Auch Johann mag keine Immobilien und zusätzlich möchte er nicht in die Aktienabteiung gehen.
- Für Frank kommt weder die Aktien- noch die Kredit- oder die Immobilienabteilung in Frage.
- Für Thomas gibt es bis auf die Sparbücher und die Immobilien keine Einschränkungen.

Es ist klar, dass jede Stelle nur von höchstens einem Bewerber besetzt werden kann und jeder Bewerber nur höchstens eine Stelle erhalten kann.

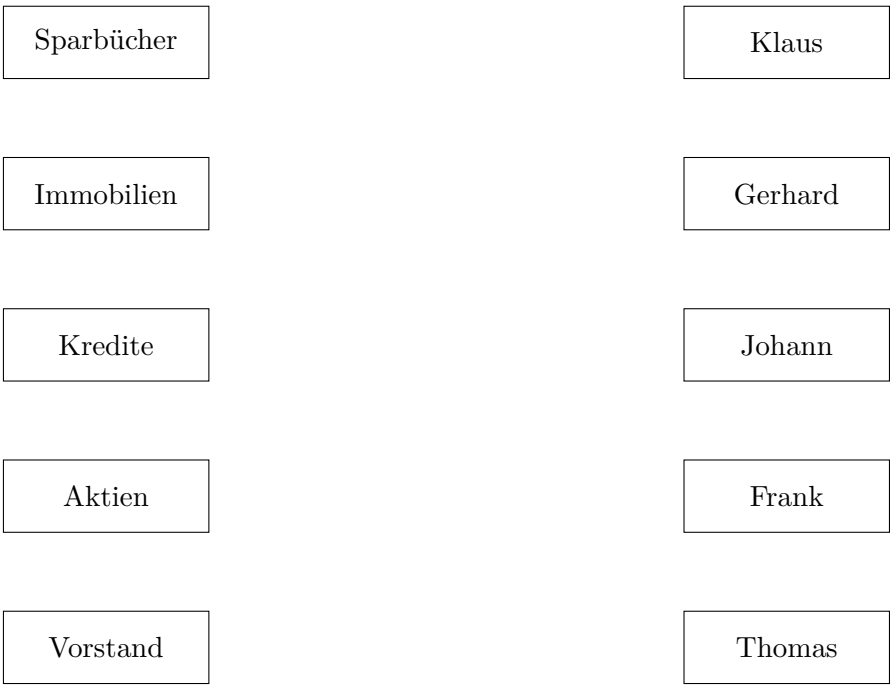
(i) Stellen Sie den Konfliktgraphen auf, in dem eine Kante zwischen Stelle A und Bewerber B dafür steht, dass B nicht auf der Stelle A arbeiten will. (2 Pkte)

Konfliktgraph:



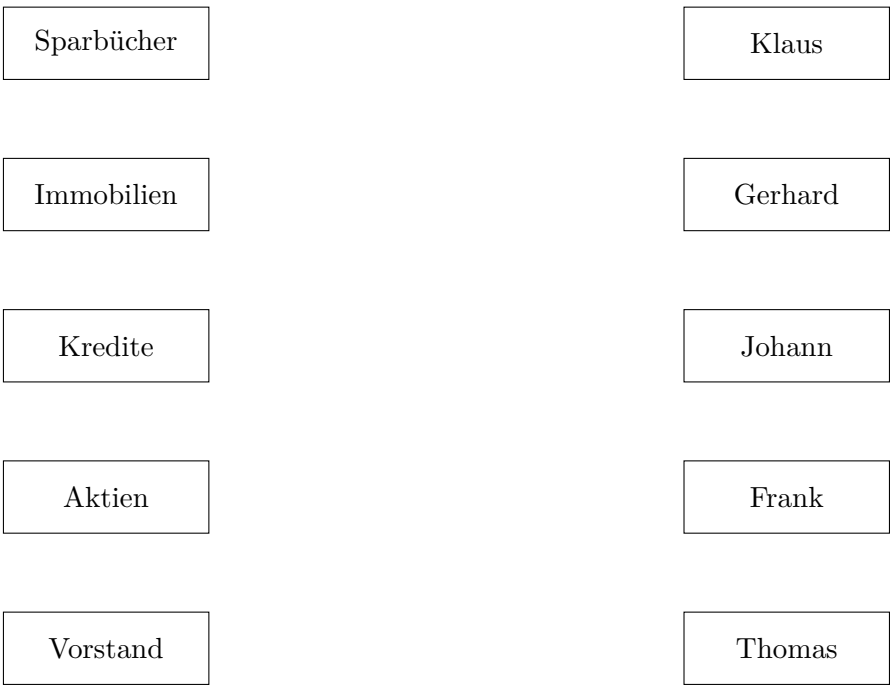
(ii) Bilden Sie auf der Grundlage Ihres Konfliktgraphen einen "Zufriedenheitsgraphen", in dem eine Kante zwischen Stelle A und Bewerber B dafür steht, dass B mit der Stelle A zufrieden wäre. (2 Pkte)

"Zufriedenheitsgraph":



(iii) Geben sie ein Matching maximaler Größe in Ihrem "Zufriedenheitsgraphen" an (2 Pkte)

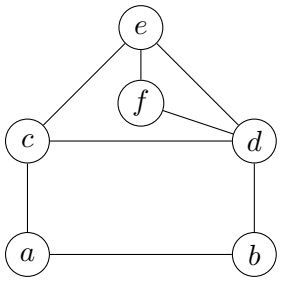
Matching maximaler Größe im "Zufriedenheitsgraph":



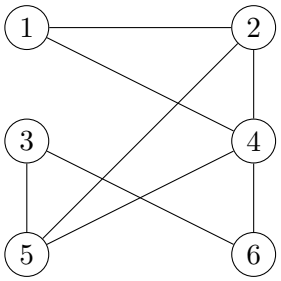
(iv) Geben Sie eine Möglichkeit an, wie die Bank die Bewerber auf die Stellen verteilen kann, so dass möglichst viele Stellen besetzt werden und kein Bewerber eine Stelle erhält, auf der er nicht arbeiten will. (1 Pkt)

(v) Begründen Sie, warum die von Ihnen gefundene Zuordnung bestmöglich ist, das heißt, warum es keine Zuordnung von den Bewerbern auf die Stellen gibt, die mehr Stellen besetzt. (2 Pkte)

(e) Betrachten Sie die folgenden ungerichteten Graphen G_4 und G_5:



G_4



G_5

(i) Geben Sie einen Isomorphismus von G_4 nach G_5 an. (2 Pkte)

(ii) Ist der Graph G_5 planar? Begründen Sie Ihre Antwort. (1 Pkt)

Aufgabe 3: **(12 Punkte)**

(a) Sei $\sigma = \{\dot{B}\}$ eine Signatur mit einem zweistelligen Relationssymbol $\dot{B}$, wobei $\dot{B}(x, y)$ besagt, dass das Bauteil y Bestandteil des Bauteils x ist. (4 Pkte)

(i) Geben Sie eine Formel φ der Logik erster Stufe über die Signatur σ an, die aussagt, dass mindestens zwei Bauteile existieren, welche nicht Bestandteil eines anderen Bauteils sind.

$\varphi =$

(ii) Beschreiben Sie umgangssprachlich, was die folgende Formel

$$\forall x \forall y \forall z \Big(((\dot{B}(x, y) \wedge \dot{B}(y, z)) \rightarrow \dot{B}(x, z)) \wedge (\dot{B}(x, y) \rightarrow \neg \dot{B}(y, x)) \Big)$$

aussagt.

(b) Sei φ eine FO[σ]-Formel. Wann heißt eine Belegung β *passend* zu φ? Geben Sie eine präzise Definition an. (2 Pkte)

Definition: (passende Belegung)

(c) Sei $\sigma = \{\dot{E}\}$ eine Signatur mit einem zweistelligen Relationssymbol $\dot{E}$. Geben Sie für die Formel (4 Pkte)

$$\varphi(x) := \forall y \exists z \big((\dot{E}(y, x) \rightarrow \dot{E}(x, z)) \vee x \doteq y \big)$$

eine Struktur $\mathfrak{A}$ und zwei Interpretationen $\mathcal{I}_1 = (\mathfrak{A}, \beta_1)$ und $\mathcal{I}_2 = (\mathfrak{A}, \beta_2)$ an, so dass $\mathcal{I}_1$ die Formel φ erfüllt und $\mathcal{I}_2$ nicht.

(d) Sei $\sigma = \{\dot{E}\}$ eine Signatur mit einem zweistelligen Relationssymbol $\dot{E}$. Strukturen über dieser Signatur kann man als gerichtete Graphen auffassen. Wir betrachten zwei beliebige σ-Strukturen $\mathfrak{A}$ und $\mathfrak{B}$, welche zueinander isomorph sind. (2 Pkte)

Kann es einen Satz φ der Logik erster Stufe über der Signatur σ geben, so dass $\mathfrak{A}$ den Satz erfüllt, aber $\mathfrak{B}$ nicht? Begründen Sie Ihre Antwort.

Aufgabe 4: **(15 Punkte)**

Die Sprache UPNZ (Umgekehrte Polnische Notation auf Ziffern) sei über dem Alphabet $\Sigma := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, *\}$ wie folgt rekursiv definiert:

Basisregel: (B) Jede Ziffer, also jedes Zeichen aus $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ist in UPNZ

Rekursive Regeln: (R1) Sind w_1 und w_2 in UPNZ, so ist auch $w_1 w_2 +$ in UPNZ

(R2) Sind w_1 und w_2 in UPNZ, so ist auch $w_1 w_2 *$ in UPNZ

(a) Welche der folgenden Wörter gehören zur Sprache UPNZ, welche nicht? (3 Pkte)

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

Wort	... liegt in UPNZ?	
123	☐ ja	☐ nein
$69 + 38 * +$	☐ ja	☐ nein
$9 + 3$	☐ ja	☐ nein

(b) Geben Sie eine kontextfreie Grammatik G an, die genau die Sprache UPNZ erzeugt. (4 Pkte)

(c) Zur Erinnerung wird die bereits zu Beginn der Aufgabe 4 angegebene Definition der Sprache UPNZ wiederholt:

Die Sprache UPNZ (Umgekehrte Polnische Notation auf Ziffern) sei über dem Alphabet $\Sigma := \{0,1,2,3,4,5,6,7,8,9,+,*\}$ wie folgt rekursiv definiert:

Basisregel: (B) Jede Ziffer, also jedes Zeichen aus $\{0,1,2,3,4,5,6,7,8,9\}$ ist in UPNZ

Rekurs. Regeln: (R1) Sind w_1 und w_2 in UPNZ, so ist auch $w_1 w_2 +$ in UPNZ

(R2) Sind w_1 und w_2 in UPNZ, so ist auch $w_1 w_2 *$ in UPNZ

Wir definieren zwei Funktionen f und g, die jedem Wort $w \in$ UPNZ jeweils einen Wert zuweisen.
Sei z eine Ziffer, also ein Zeichen aus $\{0,1,2,3,4,5,6,7,8,9\}$ und seien w_1, w_2 Worte aus UPNZ. Die Funktionen $f : \text{UPNZ} \to \mathbb{N}$ und $g : \text{UPNZ} \to \{0,1\}$ sind induktiv entsprechend der Definition von UPNZ wie folgt definiert:

$$f(z) = z \qquad g(z) = \begin{cases} 0, & \text{falls } z \text{ gerade ist} \\ 1, & \text{sonst} \end{cases}$$

$$f(w_1 w_2 +) = f(w_1) + f(w_2) \qquad g(w_1 w_2 +) = \begin{cases} 0, & \text{falls } g(w_1) = g(w_2) \\ 1, & \text{sonst} \end{cases}$$

$$f(w_1 w_2 *) = f(w_1) \cdot f(w_2) \qquad g(w_1 w_2 *) = \begin{cases} 0, & \text{falls } g(w_1) = 0 \text{ oder } g(w_2) = 0 \\ 1, & \text{sonst} \end{cases}$$

Nach diesen Definitionen ist zum Beispiel $f(32 + 47 + *) = 55$ und $g(32 + 47 + *) = 1$.

(i) Berechnen Sie die folgenden Funktionswerte (2 Pkte)

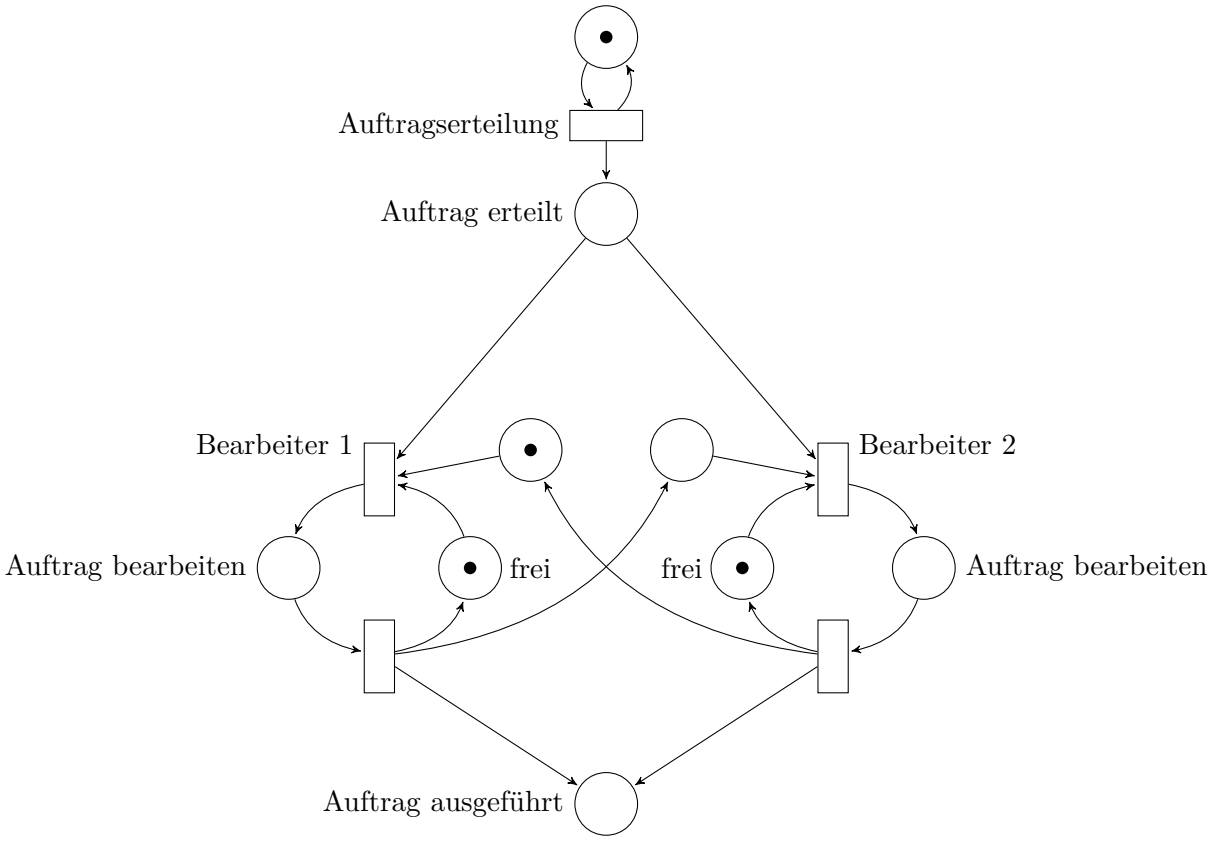
$f(24 * 91 * +) =$ $\square$ $\qquad$ $g(24 * 91 * +) =$ $\square$

(ii) Beweisen Sie den folgenden Zusammenhang durch vollständige Induktion: (6 Pkte)

Für jedes Wort $w \in \text{UPNZ}$ gilt: $f(w)$ ist gerade $\iff$ $g(w) = 0$

Aufgabe 5: **(8 Punkte)**

(a) Die Bearbeitung von Aufträgen durch zwei Bearbeiter wurde durch folgendes Petri-Netz modelliert. (4 Pkte)



Welche der folgenden Aussagen sind im Sinne des modellierten Petri-Netzes wahr, welche nicht?

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

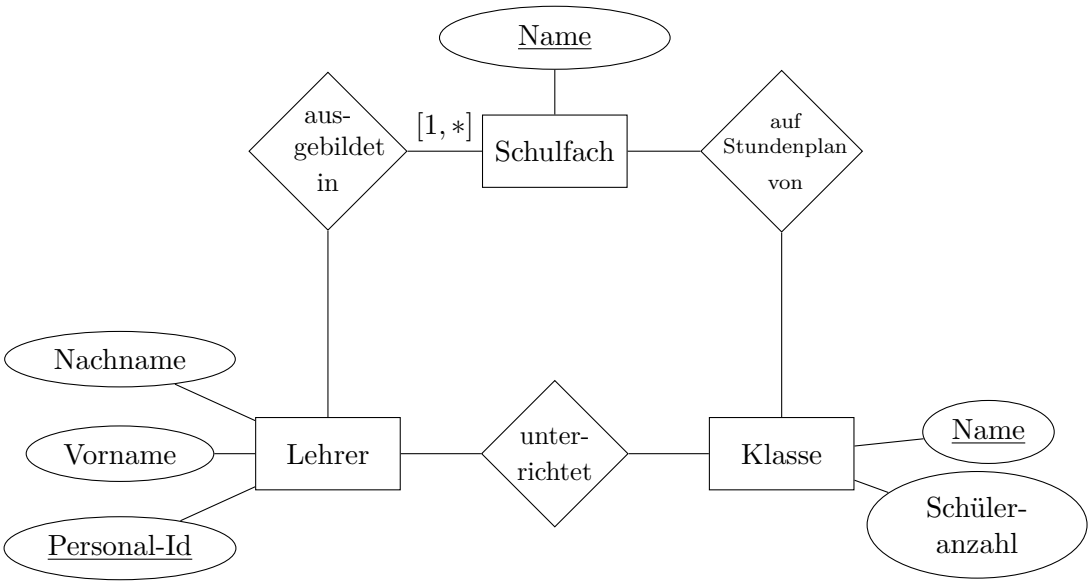
(i) Es können sich beliebig viele erteilte Aufträge für die Bearbeiter anhäufen. ☐ wahr ☐ falsch

(ii) Ein neuer Auftrag kann nur erteilt werden, wenn der letzte ausgeführt wurde. ☐ wahr ☐ falsch

(iii) Es können zwei Aufträge gleichzeitig bearbeitet werden. ☐ wahr ☐ falsch

(iv) Es ist möglich, dass der Bearbeiter 1 bereits 42 Aufträge mehr als der Bearbeiter 2 bearbeitet hat. ☐ wahr ☐ falsch

(b) Es sei folgendes Entity-Relationship-Modell gegeben: (4 Pkte)



Welche der folgenden Aussagen sind im Sinne des oben angegebenen Entity-Relationship-Modells wahr, welche nicht?

Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

(i) In jeder Klasse wird jedes Schulfach unterrichtet. ☐ wahr ☐ falsch

(ii) Für jedes Schulfach ist mindestens ein Lehrer ausgebildet. ☐ wahr ☐ falsch

(iii) In jeder Klasse müssen alle Schüler unterschiedliche Namen haben. ☐ wahr ☐ falsch

(iv) Es ist möglich, dass es zwei Lehrer mit demselben Nachnamen und demselben Vornamen gibt. ☐ wahr ☐ falsch

Aufgabe 6: **(20 Punkte)**

(a) R sei als folgender regulärer Ausdruck über dem Alphabet $\Sigma := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :\}$ gegeben:

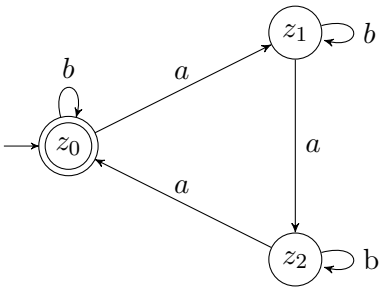
$$((1 (0 \mid 1 \mid 2)) \mid (0 \mid 1 \mid \cdots \mid 9)) \; : \; (0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5) (0 \mid 1 \mid \cdots \mid 9)$$

(i) Welche der folgenden Wörter liegen in der von R definierten Sprache $L(R)$, welche nicht? (3 Pkte)
Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

Wort	... liegt in $L(R)$?	
9 : 40	☐ ja	☐ nein
125	☐ ja	☐ nein
13 : 28	☐ ja	☐ nein

(ii) Geben Sie eine (mathematische oder umgangssprachliche) Beschreibung der Sprache $L(R)$ an, die von R definiert wird. (2 Pkte)

(b) Sei A der folgende deterministische endliche Automat:



(i) Welche der folgenden Wörter werden von A akzeptiert, welche nicht? (3 Pkte)
Kreuzen Sie alle richtigen Antworten an. Für jedes korrekte Kreuz bekommen Sie einen Punkt, für jedes **falsche Kreuz** wird **ein Punkt abgezogen**. Die Gesamtpunktzahl ist aber mindestens 0.

Wort	... wird akzeptiert?	
aababbb	☐ ja	☐ nein
bbabbabb	☐ ja	☐ nein
ababaaaba	☐ ja	☐ nein

(ii) Geben Sie eine (mathematische oder umgangssprachliche) Beschreibung der Sprache $L(A)$ an, die vom Automaten A akzeptiert wird. (2 Pkte)

(c) Geben Sie die graphische Darstellung eines nicht-deterministischen endlichen Automaten an, der genau diejenigen Wörter über dem Alphabet $\{a, b\}$ akzeptiert, die das Teilwort bb zweimal enthalten, getrennt durch mindestens ein weiteres Zeichen. (3 Pkte)

D.h. der Automat soll genau die Wörter $w \in \{a, b\}^*$ akzeptieren, für die es Wörter $x, y, z \in \{a, b\}^*$ mit $y \neq \varepsilon$ gibt, so dass $w = xbbybbz$.

(d) Betrachten Sie das Alphabet $\Sigma := \{a, b\}$ und zeigen Sie, dass die Sprache (7 Pkte)

$$L := \{w \in \{a, b\}^* : w \text{ enthält genau doppelt so viele } a\text{'s wie } b\text{'s}\}$$

nicht regulär ist.

Hinweis: Verwenden Sie das Pumping-Lemma aus der Vorlesung:

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 vier Bedingungen erfüllt sind:

(i) $x = uvw$
(ii) $|uv| \leq z$
(iii) $|v| \geq 1$
(iv) für jedes $i \in \mathbb{N}$ gilt: $uv^iw \in L$.
(d.h.: $uw \in L$, $uvw \in L$, $uvvw \in L$, $uvvvw \in L$, ...)

Literaturverzeichnis

[1] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Searching the web. *ACM Transactions on Internet Technology*, 1(1):2–43, 2001.

[2] Albrecht Beutelspacher. *"Das ist o.B.d.A. trivial!". Tipps und Tricks zur Formulierung mathematischer Gedanken.* Vieweg Studium, Braunschweig, 2002.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[4] P. P. Chen. The Entity-Relationship-Model — Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[5] Reinhard Diestel. *Graphentheorie.* Springer-Verlag, Berlin, 2006.

[6] Heinz-Dieter Ebbinghaus. *Einführung in die Mengenlehre.* Spektrum Akademischer Verlag, Heidelberg Berlin, 2003.

[7] Ayman Farahat, Thomas LoFaro, Joel C. Miller, Gregory Rae, and Lesley A. Ward. Authority rankings from HITS, PageRank, and SALSA: Existence, uniqueness, and effect of initialization. *SIAM Journal on Scientific Computing*, 27(4):1181–1201, 2006.

[8] William Feller. *An Introduction to Probability Theory and Its Applications: Volume I.* Wiley, 3rd edition, 1968. ISBN: 978-0-471-25708-0.

[9] Martin Grohe. *Theoretische Informatik I.* Skript zur Vorlesung am Institut für Informatik, Humboldt-Universität zu Berlin, 2007.

[10] J. Y. Halpern, R. Harper, N. Immerman, P. G. Kolaitis, M. Y. Vardi, and V. Vianu. On the unusual effectiveness of logic in computer science. *The Bulletin of Symbolic Logic*, 7(2):213–236, June 2001.

[11] Andreas Heuer and Gunter Saake. *Datenbanken: Konzepte und Sprachen.* MITP-Verlag, 2. Auflage edition, 2000.

[12] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[13] Olle Häggström. *Finite Markov chains and algorithmic applications.* Number 52 in London Mathematical Society Student Texts. Cambridge University Press, 2002. ISBN-10: 0521890012.

[14] Stasys Jukna. *Crashkurs Mathematik für Informatiker.* Leitfäden der Informatik. Teubner Verlag, 2008. ISBN 978-3-8351-0216-3.

[15] Uwe Kastens and Hans Kleine Büning. *Modellierung. Grundlagen und formale Methoden.* Carl Hanser Verlag, München, 2005.

[16] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[17] Martin Kreuzer and Stefan Kühling. *Logik für Informatiker*. Pearson Studium, München, 2006.

[18] Amy N. Langville and Carl D. Meyer. *Google's Pagerank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[19] L. Lovász, J. Pelikán, and K. Vesztergombi. *Discrete Mathematics*. Springer Science+Business Media, LLC, New York, 2003.

[20] Zohar Manna and Richard Waldinger. *The logical basis for computer programming*. Addison-Wesley, 1985.

[21] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[22] Christoph Meinel and Martin Mundhenk. *Mathematische Grundlagen der Informatik. Mathematisches Denken und Beweisen - Eine Einführung*. B.G. Teubner, Stuttgart, 2000.

[23] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66 (previous number: SIDL-WP-1999-0120), Stanford InfoLab, November 1999. The article is available from http://ilpubs.stanford.edu:8090/422/.

[24] Wolfgang Reisig. *Petrinetze*. Springer-Verlag, Berlin, 1982.

[25] Georg Schnitger. *Internet Algorithmen*. Skript zur Vorlesung am Institut für Informatik, Goethe-Universität Frankfurt am Main, 2009.

[26] Uwe Schöning. *Theoretische Informatik – kurzgefasst*. Spektrum Akademischer Verlag, Heidelberg, 2001.

[27] Uwe Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, Heidelberg, 2005.

[28] Ingo Wegener. *Kompendium Theoretische Informatik – eine Ideensammlung*. B.G. Teubner, Stuttgart, 1996.

[29] Ingo Wegener. *Theoretische Informatik*. B.G. Teubner, Stuttgart, 1999.