

Logik und diskrete Strukturen

(Mathematik
für Informatiker II a)

PETER KOEPKE

Bonn, Sommersemester 2006

Vorläufiges Skript

Vorwort

Die *Mathematik für Informatiker IIa (Logik und diskrete Strukturen)* baut auf der Vorlesung *Mathematik für Informatiker Ia (Lineare Algebra)* auf. Die mathematische Logik formalisiert die Methoden, die in der Vorlesung *Mathematik für Informatiker Ia* informell eingeführt wurden:

- Mathematische Aussagen
- Definitionen
- Sätze
- Schlussfolgerungen
- Beweise

Mathematische Aussagen beziehen sich auf Strukturen, insbesondere eine Schar immer wiederkehrender Grundstrukturen:

- Zahlssysteme, Körper, Vektorräume
- Relationen
- Graphen
- Boolesche Algebren

Wir betrachten insbesondere *diskrete* Strukturen, bei denen kombinatorische und algorithmische Fragen im Vordergrund stehen. Die Methoden der Logik lassen sich in Analogie zu Axiomensystemen und Strukturen auch auf Programmiersprachen und Algorithmen anwenden.

Inhaltsverzeichnis

Vorwort	3
Inhaltsverzeichnis	5
1 Prädikatenlogische Schreibweisen	7
2 Relationen	9
2.1 Äquivalenzrelationen	10
2.2 Ordnungsrelationen	13
2.3 Boolesche Algebren	16
3 Strukturen	21
3.1 Signaturen	21
3.2 Strukturen	22
3.3 Substrukturen	23
3.4 Homomorphismen	24
4 Sprachen	27
4.1 Syntax	27
4.2 Semantik	29
4.3 Allgemeingültige Aussagen	31
5 Aussagenlogische formale Beweise	33
5.1 Formale Beweise mit \wedge , \rightarrow und \leftrightarrow	34
5.2 Beweisregeln für \vee und \neg	35
5.3 Beweisansätze	36
6 Quantorenlogische formale Beweise	39
6.1 All-Aussagen	39
6.2 Existenzaussagen	40
6.3 Mathematische Texte in Naproche	41
6.4 Der Gödelsche Vollständigkeitssatz	41
7 Logisches Programmieren und formale Beweise	45
7.1 Prolog	45
7.2 Ein Beweisprüfer in Prolog	47
7.3 Der Beweisprüfer Naproche	49
8 Kombinatorik	51
8.1 Das Dirichletsche Schubfachprinzip	52
8.2 Zählformeln	53
9 Graphen	57
9.1 Eulersche Graphen	58
9.2 Hamiltonsche Graphen	59
9.3 Bäume	60
9.4 Wurzelbäume (<i>rooted trees</i>)	63
10 Gerichtete Graphen	65

Kapitel 1

Prädikatenlogische Schreibweisen

In der Vorlesung *Mathematik für Informatiker Ia (Lineare Algebra)* wurde besonderer Wert auf *logisches Vorgehen*, d.h. auf vollständige *sprachliche* Formulierungen und Argumentationen gelegt. Es wurde eine „Sprache der Mathematik“ eingeführt, in der in eingeschränkter und kontrollierter Weise mathematische Aussagen gebildet werden können. Der Hauptbegriff der linearen Algebra wurde beispielsweise folgendermaßen erfasst:

... . Dann ist $V = (V, +, \cdot)$ ein **\mathbb{K} -Vektorraum** oder ein **Vektorraum über \mathbb{K}** , wenn die folgenden Axiome gelten:

- a) Für $x, y, z \in V$ gilt $(x + y) + z = x + (y + z)$ (Assoziativgesetz).
- b) Für $x, y \in V$ gilt $x + y = y + x$ (Kommutativgesetz).
- c) Es gibt ein $0 \in V$, so dass für $x \in V$ gilt $x + 0 = x$ (Existenz eines **Nullvektors**).
- d) Für $x \in V$ gibt es ein $-x \in V$, so dass $x + (-x) = 0$ (Existenz additiver Inverser).
- e) Für $\lambda, \mu \in \mathbb{K}$ und $x \in V$ gilt $\lambda(\mu x) = (\lambda\mu)x$ (Assoziativgesetz).
- f) Für $x \in V$ gilt $1x = x$ (Neutralität der 1).
- g) Für $\lambda \in \mathbb{K}$ und $x, y \in M$ gilt $\lambda(x + y) = \lambda x + \lambda y$ (1. Distributivgesetz).
- h) Für $\lambda, \mu \in \mathbb{K}$ und $x \in M$ gilt $(\lambda + \mu)x = \lambda x + \mu x$ (2. Distributivgesetz).

Dieser mathematische Text benutzt nur wenige sprachliche Figuren: „Für ...“, „gibt es ...“ usw. Es bietet sich von daher an, *abkürzende Notationen* für diese Sprachfiguren einzuführen.

In der *Linearen Algebra* hatten wir die Bildung von mathematischen *Aussagen* diskutiert. Die einfachsten Aussagen sind *atomare Aussagen*, die *relationale Aussagen* über *Terme* machen. Wir bilden zunächst wie gewohnt aus Variablen, Konstanten und Funktionssymbolen *Terme*:

$$c, a + b, a^n, \log(x) \text{ usw.}$$

Wenn t_0, t_1, \dots Terme und $=, <, R(\dots)$ Symbole für *Relationen* sind, so lassen sich hieraus *atomare Aussagen*

$$t_0 = t_1, t_0 < t_1, R(t_0, t_1, \dots) \text{ usw.}$$

bilden. Man erhält komplexe Aussagen mit Hilfe folgender *Regeln*:

Aussagenlogische Verknüpfungen:

- Konjunktion: „ A und B “, „es gelten A und B “, oder manchmal auch nur „ A, B “.
- Disjunktion: „ A oder B “, „es gilt A oder B “.
- Negation: „nicht A “, „ A gilt nicht“.

Quantorenlogische Verknüpfungen:

- Existenz: „es gibt ein x mit A “, „es existiert ein x mit A “, „es gibt ein x , so dass A “, „es gibt ein x mit A , so dass B “; „es gibt x_0, \dots, x_{k-1} mit A “.
- Allquantor: „für alle x gilt A “, „für alle x ist A “, „für x gilt A “; „für alle x_0, \dots, x_{k-1} gilt A “.

- Bedingter Allquantor: „für alle x mit A gilt B “, „für alle x mit A ist B “, „für x mit A gilt B “; „für alle x_0, \dots, x_{k-1} mit A gilt B “.
- Eindeutige Existenz: „es gibt genau ein x mit A “.

Wir führen die Abkürzungen $\wedge, \vee, \neg, \exists, \forall$ anstelle von „und“, „oder“, „nicht“, „es existiert“ und „für alle“ ein:

- Konjunktion: $A \wedge B$ für „ A und B “,
- Disjunktion: „ $A \vee B$ für „ A oder B “,
- Negation: „ $\neg A$ für „nicht A “,
- Existenzielle Quantifizierung: $\exists x A$ für „es gibt ein x mit A “,
- Universelle Quantifizierung: $\forall x A$ für „für alle x gilt A “.

Die Vektorraum-Definition kann mit diesen Abkürzungen folgendermaßen geschrieben werden:

- $\forall x, y, z \in V: (x + y) + z = x + (y + z)$ (Assoziativgesetz).
- $\forall x, y \in V: x + y = y + x$ (Kommutativgesetz).
- $\forall x \in V: x + 0 = x$ (Nullvektor).
- $\forall x \in V \exists y \in V: x + y = 0$ (Existenz additiver Inverser).
- $\forall \lambda, \mu \in \mathbb{K} \forall x \in V: \lambda \cdot (\mu \cdot x) = (\lambda \cdot \mu) \cdot x$ (Assoziativgesetz).
- $\forall x \in V: 1 \cdot x = x$ (Neutralität der 1).
- $\forall \lambda \in \mathbb{K} \forall x, y \in M: \lambda \cdot (x + y) = \lambda \cdot x + \lambda \cdot y$ (1. Distributivgesetz).
- $\forall \lambda, \mu \in \mathbb{K} \forall x \in M: (\lambda + \mu) \cdot x = \lambda \cdot x + \mu \cdot x$ (2. Distributivgesetz).

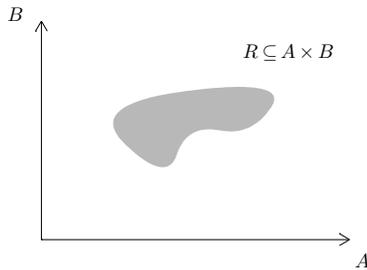
Kapitel 2

Relationen

Definition 2.1. a) R ist eine n -stellige **Relation**, wenn R eine Menge von n -Tupeln ist. Statt $(x_0, \dots, x_{n-1}) \in R$ schreiben wir auch $R(x_0, \dots, x_{n-1})$. Im Fall $n = 2$ schreiben wir statt $R(x, y)$ auch xRy (Infix-Notation).

b) R ist eine Relation auf A und B , wenn $R \subseteq A \times B$. R ist eine 2-stellige Relation auf A , wenn $R \subseteq A \times A$.

Zweistellige Relationen lassen sich 2-dimensional graphisch darstellen; eine Relation kann mit ihrem Graphen identifiziert werden.

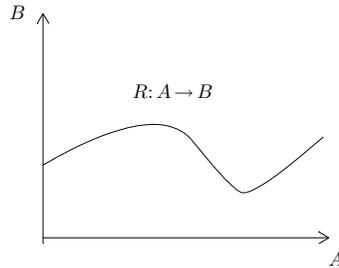


Wir definieren wichtige Eigenschaften von Relationen mit Hilfe prädikatenlogischer Schreibweisen.

Definition 2.2. Sei R eine 2-stellige Relation auf A . Dann definiere

- a) R ist **symmetrisch**, wenn $\forall a, b \in A (aRb \rightarrow bRa)$.
- b) R ist **antisymmetrisch**, wenn $\forall a, b \in A ((aRb \wedge bRa) \rightarrow a = b)$.
- c) R ist **reflexiv**, wenn $\forall a \in A aRa$.
- d) R ist **transitiv**, wenn $\forall a, b, c \in A ((aRb \wedge bRc) \rightarrow aRc)$.
- e) R ist eine **Äquivalenzrelation**, wenn R symmetrisch, reflexiv und transitiv ist.

Funktionen sind Relationen auf $A \times B$, bei denen jedem $a \in A$ genau ein $b \in B$ zugeordnet ist:



- Definition 2.3.**
- a) R ist eine **Funktion** von A nach B , $R: A \rightarrow B$, wenn $\forall a \in A \exists b \in B (aRb \wedge \forall b' \in B (aRb' \rightarrow b = b'))$.
 - b) R ist eine **injektive Funktion** von A nach B , wenn $(R: A \rightarrow B \wedge \forall a, a' \in A \forall b, b' \in B ((aRb \wedge a'Rb' \wedge a \neq a') \rightarrow b \neq b'))$.
 - c) R ist eine **surjektive Funktion** von A auf B , wenn $(R: A \rightarrow B \wedge \forall b \in B \exists a \in A aRb)$.
 - d) R ist eine **Bijektion** zwischen A und B , $R: A \leftrightarrow B$, wenn $(R: A \rightarrow B \wedge R$ ist injektiv $\wedge R$ ist surjektiv).

2.1 Äquivalenzrelationen

Definition 2.4. Sei R eine Äquivalenzrelation auf A . Für $a \in A$ ist

$$[a] = [a]_R = \{b \mid bRa\}$$

die **Äquivalenzklasse** von a bezüglich R

Satz 2.5. Sei R eine Äquivalenzrelation auf A . Dann gilt:

- a) $\forall a, b \in A ([a] = [b] \vee [a] \cap [b] = \emptyset)$.
- b) $A = \bigcup_{a \in A} [a]$.

Beweis. a) Sei $a, b \in A$.

Fall 1: aRb .

Behauptung: $[a] = [b]$.

Beweis: Sei $x \in [a]$. Dann ist xRa ; aRb ; xRb , wegen der Transitivität von R ; $x \in [b]$. Also ist $[a] \subseteq [b]$.

Den Beweis der umgekehrten Inklusion notieren wir noch knapper:

Sei $x \in [b]$.

xRb .

bRa , wegen der Symmetrie von R .

xRa , wegen der Transitivität von R .

$x \in [a]$.

Also gilt für alle $x \in [b]$, dass $x \in [a]$.

$[b] \subseteq [a]$. qed (Behauptung)

Dabei bezeichnet der *Rahmen*

Sei A_1, A_2, \dots, A_m .

ein *Teilargument*, in dem ausgehend von der Annahme A_1 sukzessiv A_2, \dots, A_m gezeigt werden. Nach Abschluss des Teilarguments durch das Wort „Also“ ist die Implikation

$$A_1 \rightarrow A_m$$

bewiesen.

Fall 2: $\neg aRb$:

Behauptung: $[a] \cap [b] = \emptyset$.

Beweis:

Sei $x \in [a] \cap [b]$.

xRa .

xRb .

aRx , wegen der Symmetrie von R .

aRb wegen der Transitivität von R .

Widerspruch zur Fallannahme.

Also $\forall x (x \in [a] \cap [b] \rightarrow \perp)$.

$\forall x (x \notin [a] \cap [b])$.

$[a] \cap [b] = \emptyset$. *qed*(Behauptung)

b) Sei $b \in A$.

bRb , wegen Reflexivität.

$b \in [b]$, nach Definition von $[b]$.

$b \in \bigcup_{a \in A} [a]$, nach Definition der Vereinigung.

Also gilt für alle $b \in A$, dass $b \in \bigcup_{a \in A} [a]$.

$A \subseteq \bigcup_{a \in A} [a]$.

Die Gegenrichtung $A \supseteq \bigcup_{a \in A} [a]$ ist trivial.

$A = \bigcup_{a \in A} [a]$. □

Beispiel 2.6. *Faktorisieren nach Unterräumen.* Es sei V ein \mathbb{K} -Vektorraum und U ein Unterraum von V . Definiere eine Relation $R \subseteq V \times V$ durch

$$xRy \iff x - y \in U.$$

Wir zeigen, dass R eine Äquivalenzrelation auf V ist:

(1) R ist reflexiv.

Beweis: Sei $x \in V$.

$x - x \in U$.

xRx .

Also $\forall x \in V: xRx$.

(2) R ist symmetrisch.

Beweis.

Sei $x, y \in V, xRy$.

$x - y \in U$.

$y - x = -(x - y) \in U$.

yRx .

Also $\forall x, y \in V (xRy \rightarrow yRx)$.

(3) R ist transitiv.

Beweis.

Sei $x, y, z \in V, xRy, yRz$.

$x - y \in U$.

$z - y \in U$.

$x - z = (x - y) + (y - z) \in U$.

xRz .

Also $\forall x, y, z \in V ((xRy \wedge yRz) \rightarrow xRz)$.

Definition 2.7. Sei R eine Äquivalenzrelation auf A . Dann sei

$$A/R = \{[a]_R \mid a \in A\}$$

die Menge der Äquivalenzklassen von R . A/R ist die **Faktorisierung** oder der **Quotient** von A nach R .

Beispiel 2.8. *Quotientenvektorräume.* In dem Beispiel 2.6 sind die Äquivalenzklassen von der Gestalt

$$[x]_R = x + U = \{x + u \mid u \in U\}.$$

Beweis: Sei $v \in [x]_R$.

vRx .

$u = v - x \in U$.

$v = x + u \in x + U$.

Also gilt für alle $v \in [x]_R$, dass $v \in x + U$.

$[x]_R \subseteq x + U$

Umgekehrt:

Sei $v \in x + U$.

Sei $u \in U$ mit $v = x + u$.

$v - x = u \in U$.

vRx .

$v \in [x]_R$.

Also $v \in [x]_R$.

Also gilt für alle $v \in x + U$, dass $v \in [x]_R$.

$[x]_R \supseteq x + U$. *qed*

Geometrisch sind die Äquivalenzklassen *Parallelverschiebungen* $x + U$ des Untervektorraums U um den Vektor x . Nach Satz 2.5 zerfällt der Raum V in zu U parallele Untermengen.

Man kann weiter zeigen, dass die Menge A/R mit folgenden Operationen zu einem \mathbb{K} -Vektorraum $A/R = (A/R, \oplus, \odot)$ erweitert werden kann:

$$\begin{aligned} (x + U) \oplus (y + U) &= (x + y) + U; \\ \lambda \odot (x + U) &= (\lambda \cdot x) + U. \end{aligned}$$

Beispiel 2.9. *Modulare Arithmetik.* Fixiere $n \in \mathbb{N}$, $n \neq 0$. Wir hatten die Relation $\equiv \pmod{n}$ der Kongruenz modulo n auf den ganzen Zahlen \mathbb{Z} definiert:

$$i \equiv j \pmod{n} \text{ gdw. } n \mid (i - j).$$

Diese Relation ist eine Äquivalenzrelation auf \mathbb{Z} ; der Beweis verläuft ähnlich wie bei der Bildung des Quotientenraums bei Vektorräumen. Die Äquivalenzklassen $[i]$ bezüglich dieser Relation sind von der Gestalt

$$[i] = i + n \cdot \mathbb{Z} = \{i + n \cdot k \mid k \in \mathbb{Z}\}.$$

Wir hatten die Äquivalenzklassen durch die *Reste* modulo n repräsentiert:

$$\mathbb{Z}_n = \{0, 1, \dots, n - 1\}.$$

Die natürliche Korrespondenz φ zwischen \mathbb{Z}_n und $\mathbb{Z}/(\text{mod } n)$ ist durch

$$i \mapsto [i] = i + n \cdot \mathbb{Z}$$

gegeben. Die Korrespondenz ist mit den Operationen \oplus_n und \otimes_n der Arithmetik modulo n verträglich:

$$\begin{aligned} \varphi(i \oplus_n j) &= (i \oplus_n j) + n \cdot \mathbb{Z} = (i + j) + n \cdot \mathbb{Z} = (i + n \cdot \mathbb{Z}) \oplus (j + n \cdot \mathbb{Z}) = \varphi(i) \oplus \varphi(j) \\ \varphi(i \otimes_n j) &= (i \otimes_n j) + n \cdot \mathbb{Z} = (i \cdot j) + n \cdot \mathbb{Z} = (i + n \cdot \mathbb{Z}) \otimes (j + n \cdot \mathbb{Z}) = \varphi(i) \otimes \varphi(j). \end{aligned}$$

2.2 Ordnungsrelationen

Definition 2.10. Eine 2-stellige Relation \leq auf X ist eine **partielle Ordnung** wenn \leq transitiv, reflexiv und antisymmetrisch ist. Die Relation \leq ist eine **(totale) Ordnung** oder **lineare Ordnung**, wenn \leq außerdem **linear** ist:

$$\forall x, y \in X (x \leq y \vee x = y \vee y \leq x).$$

Beispiel 2.11. a) Die Inklusion \subseteq auf Mengen ist eine partielle Ordnung. Die Inklusion ist nicht linear, denn es gibt nicht-leere, zueinander disjunkte Mengen:

$$\{0\} \not\subseteq \{1\}, \{0\} \neq \{1\}, \{1\} \not\subseteq \{0\}.$$

Die Antisymmetrie der Inklusion ist das fundamentale Kriterium für die Gleichheit von Mengen (**Extensionalität**), das häufig für den Beweis von Mengengleichheit benutzt wird:

$$\forall x, y ((x \subseteq y \wedge y \subseteq x) \rightarrow x = y).$$

b) Die gewöhnlichen Ordnungen \leq auf den Zahlbereichen \mathbb{N} , \mathbb{Z} , \mathbb{Q} und \mathbb{R} sind totale Ordnungen.

Zu jeder partiellen Ordnung lässt sich eine *strenge* oder *strikte* Variante definieren:

Definition 2.12. Sei \leq eine partielle Ordnung auf X . Definiere dann eine zweistellige Relation $<$ auf X durch:

$$x < y \text{ gdw. } (x \leq y \wedge x \neq y).$$

Diese Relation ist transitiv und irreflexiv:

$$\begin{aligned} \forall x, y, z \in X ((x < y \wedge y < z) \rightarrow x < z), \\ \forall x \in X \neg x < x. \end{aligned}$$

Wenn zusätzlich \leq linear ist, so erfüllt $<$ die folgende Trichotomie:

$$\forall x, y \in X (x < y \vee x = y \vee y < x).$$

Partielle Ordnungen (in nicht-strikter und strikter Form) treten häufig auf. Wir führen einige wichtige Begriffe für partielle Ordnungen ein. Für eine partielle Ordnung \leq auf X definiere den Ausdruck

$$\max(a, X) \leftrightarrow (a \in X \wedge \forall x \in X: x \leq a),$$

der das **Maximum** von X charakterisiert.

Satz 2.13. Eine partielle Ordnung hat höchstens ein Maximum:

$$\forall a, a' ((\max(a, X) \wedge \max(a', X) \rightarrow a = a').$$

Wenn es existiert, bezeichnen wir das Maximum a von X mit

$$a = \max(X).$$

Beweis. Betrachte $a, a' \in X$ mit $\max(a, X)$ und $\max(a', X)$. Dann ist

$$\forall x \in X: x \leq a \text{ und } \forall x \in X: x \leq a'.$$

Speziell ist $a' \leq a$ und $a \leq a'$. Wegen der Antisymmetrie von \leq ist $a = a'$. \square

Definition 2.14. Sei (X, \leq) eine partielle Ordnung. Definiere

a) $\text{os}(a, Y) \leftrightarrow (a \in X \wedge \forall y \in Y: y \leq a)$; $\text{os}(a, Y)$ besagt, dass a eine **obere Schranke** von Y ist;

b) $\text{sup}(a, Y) \leftrightarrow \text{os}(a, Y) \wedge \forall b(\text{os}(b, Y) \rightarrow b \geq a)$; $\text{sup}(a, Y)$ besagt, dass a eine **kleinste obere Schranke** oder ein **Supremum** von Y ist.

Satz 2.15. Wenn $\sup(a, Y)$ und $\sup(a', Y)$, dann ist $a = a'$. Wir können daher $a = \sup(Y)$ anstelle von $\sup(a, Y)$ schreiben.

Beweis. Übung. □

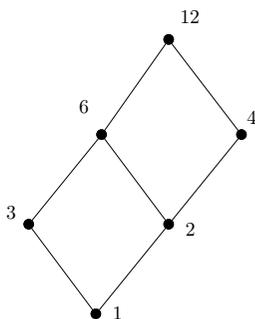
Definition 2.16. Definiere $\text{us}(a, Y) \leftrightarrow \forall x \in Y a \leq x$. Dies besagt, dass a eine **untere Schranke** von Y ist. a ist ein **Infimum** von Y , wenn $\inf(a, Y) \leftrightarrow \text{us}(a, Y) \wedge \forall b(\text{us}(b, Y) \rightarrow b \leq a)$. Wie das Supremum so ist ein Infimum eindeutig definiert, wenn es existiert. Daher schreiben wir auch $a = \inf(Y)$ anstelle von $\inf(a, Y)$

Satz 2.17. Angenommen, $\max(X)$ existiert. Dann ist $\max(X) = \inf(\emptyset)$.

Beweis. $\text{us}(a, \emptyset)$ ist äquivalent zu: $\forall x \in \emptyset a \leq x$. Diese Eigenschaft ist immer erfüllt. Daher

$$\inf(a, \emptyset) \leftrightarrow \forall a' a' \leq a \leftrightarrow a = \max(X). \quad \square$$

Beispiel 2.18. (Kleine) endliche Beispiele von partiellen Ordnungen lassen sich durch **Hasse-Diagramme** angeben. Das sind Figuren, in denen die Elemente der Trägermenge durch Kanten verbunden sind. Wenn a und b durch eine Kante verbunden sind und a oberhalb von b liegt, so bedeutet das, dass $a \geq b$ ist. Die Relation \leq besteht aus allen Paaren, die durch einen von unten nach oben laufenden Kantenzug verbunden werden können. Das folgende Bild ist das Hasse-Diagramm der partiellen Ordnung $m|n|12$, d.h. der Teilbarkeitsrelation auf den Teilern von 12.



Wir interessieren uns besonders für Suprema und Infima endlicher Mengen.

Definition 2.19. Sei (X, \leq) eine partielle Ordnung. $a \sqcup b = \sup(\{a, b\})$ ist die **Vereinigung** von a und b . $a \sqcap b = \inf(\{a, b\})$ ist der **Schnitt** von a und b .

Satz 2.20. Wenn \subseteq die partielle Ordnung der **Inklusion** auf Mengen ist, so gilt:

- $\forall a, b: a \sqcup b = a \cup b$.
- $\forall a, b: a \sqcap b = a \cap b$.

Beweis. a) Betrachte Mengen a und b .

Es gilt $a \subseteq a \cup b$ und $b \subseteq a \cup b$. Daher ist $\text{os}(a \cup b, \{a, b\})$. Betrachte ein c mit $\text{os}(c, \{a, b\})$. Dann ist $a \subseteq c$ und $b \subseteq c$. Zusammen ist $a \cup b \subseteq c$. Also ist $\forall c(\text{os}(c, \{a, b\}) \rightarrow a \cup b \subseteq c)$. Damit ist

$$a \cup b = \sup(\{a, b\}) = a \sqcup b.$$

b) lässt sich analog zeigen. □

Definition 2.21. Eine partiell geordnete Menge (X, \leq) ist ein **Verband**, wenn die Suprema und Infima aller endlichen Teilmengen existieren.

Satz 2.22. Sei (X, \leq) ein Verband. Dann besitzt (X, \leq) ein Maximum und ein Minimum, nämlich $\inf(\emptyset)$ und $\sup(\emptyset)$. Wir bezeichnen das Maximum und das Minimum mit \top und \perp (**Top** und **Bottom**).

Folgende Gesetze gelten in Verbänden:

Satz 2.23. Sei X ein Verband. Dann gilt in X :

- a) $\forall x, y: x \leq x \sqcup y$ und $\forall x, y: x \geq x \sqcap y$.
- b) $\forall x: x \sqcup x = x$ und $\forall x: x \sqcap x = x$ (*Idempotenz*).
- c) $\forall x, y: x \sqcup y = y \sqcup x$ und $\forall x, y: x \sqcap y = y \sqcap x$ (*Kommutativität*).
- d) $\forall x, y, z: x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$ und $\forall x, y, z: x \sqcap (y \sqcap z) = (x \sqcap y) \sqcap z$ (*Assoziativität*).
- e) $\forall x, y: x \sqcap (x \sqcup y) = x$ und $\forall x, y: x \sqcup (x \sqcap y) = x$ (*Absorption*).
- f) $\forall x: \perp \sqcup x = x$ und $\forall x: \perp \sqcap x = \perp$ (*Minimum*).
- g) $\forall x: \top \sqcap x = x$ und $\forall x: \top \sqcup x = \top$ (*Maximum*).

Beweis. a) Seien $x, y \in X$. $\text{os}(x \sqcup y, \{x, y\})$ und $x \sqcup y \geq x$.

b) Sei $x \in X$. $x \geq x$ und $\text{os}(x, \{x, x\})$. Sei $y \in X$ mit $\text{os}(y, \{x, x\})$. Dann ist $y \geq x$. Also ist $\forall y \in X (\text{os}(y, \{x, x\}) \rightarrow y \geq x)$. Zusammen ist

$$(\text{os}(x, \{x, x\}) \wedge \forall y \in X (\text{os}(y, \{x, x\}) \rightarrow y \geq x)),$$

d.h. $x = \sup(\{x, x\}) = x \sqcup x$.

d) Seien $x, y, z \in X$. Dann gilt

$$\begin{aligned} x \sqcup (y \sqcup z) &\geq x \\ x \sqcup (y \sqcup z) &\geq y \sqcup z \geq y \\ x \sqcup (y \sqcup z) &\geq x \sqcup y \\ x \sqcup (y \sqcup z) &\geq y \sqcup z \geq z \\ x \sqcup (y \sqcup z) &\geq (x \sqcup y) \sqcup z \end{aligned}$$

Umgekehrt:

$$\begin{aligned} x &\leq x \sqcup y \leq (x \sqcup y) \sqcup z \\ y &\leq x \sqcup y \leq (x \sqcup y) \sqcup z \\ z &\leq (x \sqcup y) \sqcup z \\ y \sqcup z &\leq (x \sqcup y) \sqcup z \\ x \sqcup (y \sqcup z) &\leq (x \sqcup y) \sqcup z \end{aligned}$$

Wegen der Antisymmetrie von \leq ist dann

$$x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z.$$

e) Seien $x, y \in X$. Wir zeigen die Gleichheit durch zwei Ungleichungen:

Beh: $x \sqcap (x \sqcup y) \leq x$. Dies gilt nach a).

Beh: $x \sqcap (x \sqcup y) \geq x$. $x \geq x$ und nach a) $x \sqcup y \geq x$. Also ist $\text{us}(x, \{x, x \sqcup y\})$ und $x \leq \inf(x, x \sqcup y) = x \sqcap (x \sqcup y)$.

Da \leq antisymmetrisch ist, ist $x \sqcap (x \sqcup y) = x$. □

Beim Rechnen mit *zwei* Rechenarten sind Distributivgesetze interessant. Für allgemeine Verbände gilt:

Satz 2.24. Sei X ein Verband.

- a) $\forall x, y, z: (y \geq z \rightarrow x \sqcap y \geq x \sqcap z)$
- b) $\forall x, y, z: x \sqcap (y \sqcup z) \geq (x \sqcap y) \sqcup (x \sqcap z)$

$$c) \forall x, y, z: x \sqcup (y \sqcap z) \leq (x \sqcup y) \sqcap (x \sqcup z).$$

Beweis. a) Betrachte $x, y, z \in X$ mit $y \geq z$. Dann ist $x \sqcap z \leq x$. $x \sqcap z \leq z \leq y$. $us(x \sqcap z, \{x, y\})$. Also $x \sqcap z \leq \inf(\{x, y\}) = x \sqcap y$.

b) Betrachte $x, y, z \in X$. Dann ist $x \geq x \sqcap y$, $x \geq x \sqcap z$ und $x \geq (x \sqcap y) \sqcup (x \sqcap z)$. Weiter ist $y \sqcup z \geq y \geq x \sqcap y$, $y \sqcup z \geq z \geq x \sqcap z$ und $y \sqcup z \geq (x \sqcap y) \sqcup (x \sqcap z)$. Also ist $(x \sqcap y) \sqcup (x \sqcap z)$ untere Schranke von $\{x, y \sqcup z\}$ und $x \sqcap (y \sqcup z) \geq (x \sqcap y) \sqcup (x \sqcap z)$.

c) lässt sich mit einem zum Beweis von b) „dualen“ Argument zeigen, bei dem \sqcap und \sqcup bzw. \leq und \geq vertauscht sind. \square

Definition 2.25. Ein Verband (X, \leq) ist ein **distributiver Verband**, wenn in ihm die **Distributivgesetze** gelten:

$$\begin{aligned} \forall x, y, z: x \sqcap (y \sqcup z) &= (x \sqcap y) \sqcup (x \sqcap z) \\ \forall x, y, z: x \sqcup (y \sqcap z) &= (x \sqcup y) \sqcap (x \sqcup z). \end{aligned}$$

Beispiel 2.26. Der Teilbarkeitsverband ist distributiv (Übung).

2.3 Boolesche Algebren

Wenn wir den Verband der Teilmengen einer Menge Z betrachten, so gibt es dort noch eine weitere wichtige Operation, nämlich die Komplementbildung:

$$A \mapsto -A = \{x \in Z \mid x \notin A\} = Z \setminus A.$$

Das Komplement ist dadurch gekennzeichnet, dass $A \cap (-A) = \emptyset$ und $A \cup (-A) = Z$. Wir können diese Situation abstrakt in beliebigen distributiven Verbänden studieren.

Definition 2.27. Sei (X, \leq) ein distributiver Verband. Elemente $x, x' \in X$ heißen **komplementär**, wenn $x \sqcup x' = \top$ und $x \sqcap x' = \perp$.

Satz 2.28. Sei (X, \leq) ein distributiver Verband und $x, x', x'' \in X$. Wenn x und x' als auch x und x'' komplementär sind, so ist $x' = x''$. Dieses **Komplement** x' von x wird auch mit $-x$ bezeichnet.

Beweis. Betrachte x, x', x'' mit den genannten Eigenschaften.

$$\begin{aligned} x' &= x' \sqcap \top \\ &= x' \sqcap (x \sqcup x'') \\ &= (x' \sqcap x) \sqcup (x' \sqcap x'') \\ &= \perp \sqcup (x' \sqcap x'') \\ &= x' \sqcap x'' \end{aligned}$$

Daraus folgt $x' \leq x''$. Genauso zeigt man die umgekehrte Ungleichung $x'' \leq x'$. Zusammen gilt dann $x' = x''$. \square

Definition 2.29. Eine **Boolesche Algebra** ist ein distributiver Verband (X, \leq) mit der Eigenschaft:

$$\forall x \in X \exists x' \in X (x \text{ und } x' \text{ sind komplementär}).$$

In diesem Verband können wir die Operationen $\sqcup, \sqcap, -$ betrachten. Wir bevorzugen allerdings folgende algebraische Definition von Booleschen Algebren.

Definition 2.30. Eine **Boolesche Algebra** ist eine Struktur $(B, +, \cdot, -, 0, 1)$ mit zwei zwei-stelligen Funktionen $+, \cdot$, einer einstelligen Funktion $-$, und zwei Konstanten 0 und 1, die die folgenden Axiome erfüllt:

- $\forall x, y, z: x + (y + z) = (x + y) + z, \forall x, y, z: x \cdot (y \cdot z) = (x \cdot y) \cdot z;$
- $\forall x, y: x + y = y + x, \forall x, y: x \cdot y = y \cdot x;$

- c) $\forall x: 0 + x = x, \forall x: 1 \cdot x = x;$
 d) $\forall x, y, z: x \cdot (y + z) = (x \cdot y) + (x \cdot z), \forall x, y, z: x + (y \cdot z) = (x + y) \cdot (x + z);$
 e) $\forall x: x + (-x) = 1, \forall x: x \cdot (-x) = 0;$
 f) $-0 = 1, -1 = 0, 0 \neq 1;$
 g) $\forall x: x \cdot 0 = 0, \forall x: x + 1 = 1;$
 h) $\forall x: -(-x) = x;$
 i) $\forall x: x \cdot x = x, \forall x: x + x = x;$
 j) $\forall x, y: -(x \cdot y) = (-x) + (-y), \forall x, y: -(x + y) = (-x) \cdot (-y).$

Die Axiome j) sind die **DE MORGANSCHEN GEsetze**.

Beispiel 2.31. Das einfachste Beispiel einer Booleschen Algebra besteht nur aus den Elementen 0 und 1: $B = \{0, 1\}$. Die Verknüpfungen $+$, \cdot und $-$ müssen auf Grund der Axiome folgendermaßen definiert werden:

$$\begin{array}{|c|c|c|} \hline + & 0 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline \cdot & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline - & \\ \hline 0 & 1 \\ \hline 1 & 0 \\ \hline \end{array}.$$

Diese Algebra ist *isomorph* zur Algebra der *Wahrheitswerte*

$$\begin{array}{|c|c|c|} \hline \text{oder} & \text{W} & \text{F} \\ \hline \text{W} & \text{W} & \text{W} \\ \hline \text{F} & \text{W} & \text{F} \\ \hline \end{array}, \begin{array}{|c|c|c|} \hline \text{und} & \text{W} & \text{F} \\ \hline \text{W} & \text{W} & \text{F} \\ \hline \text{F} & \text{F} & \text{F} \\ \hline \end{array}, \begin{array}{|c|c|} \hline \text{nicht} & \\ \hline \text{W} & \text{F} \\ \hline \text{F} & \text{W} \\ \hline \end{array},$$

die wir im letzten Semester kennen gelernt hatten.

Beispiel 2.32. *Potenzmengen:* Sei $X \neq \emptyset$ eine nicht-leere Menge. Definiere die **Potenzmenge** von X als die Menge aller Teilmengen von X :

$$\text{Pot}(X) = \{Y \mid Y \subseteq X\}.$$

Dann ist $\text{Pot}(X)$ mit den Operationen \cup, \cap und Komplementbildung

$$-Y = X \setminus Y = \{x \in X \mid x \notin Y\}$$

eine Boolesche Algebra.

Satz 2.33. *Sei X eine endliche Menge mit n Elementen, $n \in \mathbb{N}$. Dann besitzt $\text{Pot}(X)$ genau 2^n Elemente.*

Beweis. Durch vollständige Induktion über $n \in \mathbb{N}$.

Induktionsanfang $n = 0$: Dann ist X die *leere Menge*, $X = \emptyset$. Die einzige Teilmenge von \emptyset ist die leere Menge selbst:

$$\text{Pot}(\emptyset) = \{\emptyset\}.$$

Damit hat $\text{Pot}(\emptyset)$ genau $1 = 2^0$ Elemente.

Induktionsschritt: die Behauptung gelte für n . Betrachte eine Menge X mit $n + 1$ Elementen. Wähle ein $a \in X$. Die Menge $X \setminus \{a\}$ hat n Elemente. Dann ist

$$\begin{aligned} \text{Pot}(X) &= \{Y \subseteq X \mid a \in Y\} \cup \{Y \subseteq X \mid a \notin Y\} \\ &= \{Z \cup \{a\} \mid Z \in \text{Pot}(X \setminus \{a\})\} \cup \text{Pot}(X \setminus \{a\}). \end{aligned}$$

Nach Induktionsvoraussetzung hat $\text{Pot}(X \setminus \{a\})$ genau 2^n Elemente. Die beiden „Summanden“ der Vereinigung haben daher jeweils 2^n Elemente. Zusammen hat die Vereinigung und damit die Potenzmenge von X $2^n + 2^n = 2^{n+1}$ Elemente.

Der Satz folgt mit dem Prinzip der vollständigen Induktion. \square

Satz 2.34. *Sei $(B, +, \cdot, -, 0, 1)$ eine Boolesche Algebra. Dann gilt in B :*

- a) $\forall x, y: (x + y = 0 \rightarrow (x = 0 \wedge y = 0)).$

$$b) \forall x, y: (x \cdot (-y) = 0 \leftrightarrow x \cdot y = x).$$

Beweis. a) Betrachte $x, y \in B$ mit $x + y = 0$. Dann ist

$$x = x + 0 = x + (x + y) = (x + x) + y = x + y = 0.$$

Ebenso ist $y = 0$.

b) Betrachte $x, y \in B$. Angenommen $x \cdot (-y) = 0$. Dann

$$\begin{aligned} x \cdot y &= (x \cdot y) + 0 \\ &= (x \cdot y) + (x \cdot (-y)) \\ &= x \cdot (y + (-y)) \\ &= x \cdot 1 \\ &= x. \end{aligned}$$

Andererseits sei $x \cdot y = x$. Dann

$$\begin{aligned} x \cdot (-y) &= (x \cdot y) \cdot (-y) \\ &= x \cdot (y \cdot (-y)) \\ &= x \cdot 0 \\ &= 0. \end{aligned}$$

□

Satz 2.35. Sei $(B, +, \cdot, -, 0, 1)$ eine Boolesche Algebra. Definiere eine 2-stellige Relation \leq auf B durch

$$x \leq y \text{ gdw. } x \cdot (-y) = 0 \text{ (gdw. } x \cdot y = y).$$

Dann ist die Struktur (B, \leq) eine partielle Ordnung.

Beweis. *Transitivität:* Betrachte $x, y, z \in B$ mit $x \leq y$ und $y \leq z$. Dann ist $x \cdot (-y) = 0$ und $y \cdot (-z) = 0$. Hieraus folgt:

$$\begin{aligned} x \cdot (-z) &= (x \cdot 1) \cdot (-z) \\ &= (x \cdot (y + (-y))) \cdot (-z) \\ &= ((x \cdot y) + (x \cdot (-y))) \cdot (-z) \\ &= ((x \cdot y) + 0) \cdot (-z) \\ &= (x \cdot y) \cdot (-z) \\ &= x \cdot (y \cdot (-z)) \\ &= x \cdot 0 \\ &= 0 \end{aligned}$$

Also ist $x \leq z$.

Reflexivität: Betrachte $x \in B$. Dann ist $x \cdot (-x) = 0$ und daher $x \leq x$.

Antisymmetrie: Betrachte $x, y \in B$ mit $x \leq y$ und $y \leq x$. Dann ist $x \cdot (-y) = 0$ und $y \cdot (-x) = 0$. Hieraus folgt:

$$\begin{aligned} x &= x \cdot 1 \\ &= x \cdot (y + (-y)) \\ &= (x \cdot y) + (x \cdot (-y)) \\ &= x \cdot y \\ &= (x \cdot y) + 0 \\ &= (x \cdot y) + ((-x) \cdot y) \\ &= (x + (-x)) \cdot y \\ &= 1 \cdot y \\ &= y. \end{aligned}$$

□

Satz 2.36. Sei $(B, +, \cdot, -, 0, 1)$ eine endliche Boolesche Algebra, d.h. die Trägermenge B ist endlich. Dann ist B isomorph zu einer Potenzmengenalgebra, d.h. es gibt eine Menge A und eine bijektive Abbildung $f: B \leftrightarrow \text{Pot}(A)$, die mit den Algebraoperationen verträglich ist:

- a) $f(0) = \emptyset, f(1) = A$;
- b) $\forall x, y \in B: f(x + y) = f(x) \cup f(y)$;
- c) $\forall x, y \in B: f(x \cdot y) = f(x) \cap f(y)$;
- d) $\forall x \in B: f(-x) = A \setminus f(x)$.

Beweis. Ein Element $a \in B$ ist ein **Atom** in B , wenn

$$a \neq 0 \wedge \forall x \in B: (x \leq a \rightarrow (x = 0 \vee x = a)),$$

wobei \leq die partielle Ordnung aus dem vorangehenden Satz ist.

(1) Sei $x \in B$ und $x \neq 0$. Dann gibt es ein Atom a in B mit $a \leq x$.

Beweis: Angenommen nicht. Definiere dann eine Folge $(x_n | n \in \mathbb{N})$ durch Rekursion: $x_0 = x$. Sei x_n definiert, so dass $x_n \leq x, x_n \neq 0$. Nach der Widerspruchannahme ist x_n kein Atom. Wähle dann $x_{n+1} \leq x_n$, so dass $x_{n+1} \neq x_n$ und $x_{n+1} \neq 0$. Dann ist $(x_n | n \in \mathbb{N})$ eine unendliche absteigende Folge in B :

$$x_0 > x_1 > x_2 > \dots$$

Das ist ein Widerspruch, da B endlich ist. *qed*

Setze $A = \{a \in B | B \text{ ist ein Atom in } B\}$. Definiere $f: B \rightarrow \text{Pot}(A)$ durch

$$f(x) = \{a \in A | a \leq x\}.$$

Wir zeigen die behaupteten Eigenschaften für f .

(2) Seien $a, b \in B$ Atome in B mit $a \cdot b \neq 0$. Dann ist $a = b$.

Beweis: Sei $x = a \cdot b \neq 0$.

$$x \cdot a = (a \cdot b) \cdot a = (b \cdot a) \cdot a = b \cdot (a \cdot a) = b \cdot a = a \cdot b = x$$

und daher ist $x \leq a$. Da a ein Atom ist, ist $x = a$. Ebenso ist $x = b$ und $a = b$.

(3) f ist injektiv.

Beweis: Betrachte $x, y \in B, x \neq y$. Dann ist $x \not\leq y$ oder $y \not\leq x$. Ohne Einschränkung der Allgemeinheit sei $x \not\leq y$. Nach der Definition von \leq ist dann $x \cdot (-y) \neq 0$. Nach (1) wähle ein Atom $a \in A$ mit $a \leq x \cdot (-y)$.

$$\begin{aligned} 0 &= a \cdot (-x \cdot (-y)) \\ &= a \cdot ((-x) + (-(-y))) \\ &= a \cdot ((-x) + y) \\ &= (a \cdot (-x)) + (a \cdot y). \end{aligned}$$

Daraus folgt: $a \cdot (-x) = 0$ und $a \leq x$. Weiter ist $a \cdot y = 0$ und

$$a \cdot (-y) = 0 + (a \cdot (-y)) = (a \cdot y) + (a \cdot (-y)) = a \cdot (y + (-y)) = a \cdot 1 = a \neq 0.$$

Daher ist $a \not\leq y$. Nach Definition von f ist $a \in f(x)$ und $a \notin f(y)$. Damit ist $f(x) \neq f(y)$. *qed*

(4) f ist surjektiv.

Beweis: Betrachte $X \in \text{Pot}(A)$. Sei $X = \{a_0, \dots, a_{n-1}\}$. Definiere

$$x = a_0 + a_1 + \dots + a_{n-1} \in B.$$

Wir zeigen, dass $f(x) = X$. Diese Mengengleichheit weisen wir durch zwei Inklusionen nach.

Betrachte $a \in f(x) = \{a \in A | a \leq x\}$. Dann ist

$$a \leq x = a_0 + a_1 + \dots + a_{n-1}$$

und

$$a = a \cdot x = a \cdot (a_0 + a_1 + \dots + a_{n-1}) = (a \cdot a_0) + \dots + (a \cdot a_{n-1}).$$

Da $a \neq 0$, wähle ein $i < n$ mit $a \cdot a_i \neq 0$. Nach (2) ist dann $a = a_i$ und $a \in X$.
Andererseits sei $a = a_i \in X$. Nach (2) ist dann

$$a \cdot x = a \cdot (a_0 + a_1 + \dots + a_{n-1}) = (a \cdot a_0) + \dots + (a \cdot a_{n-1}) = a$$

und $a \leq x$. Damit ist $a \in f(x)$. □

Aufgabe 2.1. Beweisen Sie die Teile b)-d) des Satzes.

Kapitel 3

Strukturen

3.1 Signaturen

Wir haben eine Fülle von *Strukturen* kennengelernt: die *Zahlssysteme* $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ und \mathbb{C} ; *Körper* und *Vektorräume*; *Ordnungsstrukturen*, *Verbände* und *Boolesche Algebren*. Wir wollen den allgemeinen Strukturbegriff, den wir im ersten Semester eingeführt hatten, weiterentwickeln.

Gemeinsames Merkmal von Strukturen ist das Vorhandensein von *Trägermengen* auf denen *Relationen* oder *Funktionen* wirken. Ein \mathbb{K} -Vektorraum V setzt sich aus dem Körper \mathbb{K} mit seiner Trägermenge und den zugehörigen Körperoperationen sowie einer Menge V von Vektoren zusammen. Die Vektoroperationen haben Skalare aus \mathbb{K} oder Vektoren als Argumente:

$$V: \begin{cases} \mathbb{K}: \begin{cases} \cdot_{\mathbb{K}} \\ +_{\mathbb{K}}, \cdot_{\mathbb{K}} \dots \end{cases} \\ V: \begin{cases} +_V \\ +_V, \dots \end{cases} \end{cases}$$

Die Relationen und Funktionen einer Struktur können sich wie die Skalarmultiplikation auf verschiedene Trägermengen beziehen und können verschiedene *Stellenzahlen* haben. Zur Organisation des Systems von Trägermengen und Relationen und Funktionen führen wir allgemein *Signaturen* ein. Eine Signatur liefert *Symbole* zur Referenzierung verschiedener Strukturkomponenten und fixiert die Stellenzahlen von Relationen und Funktionen.

Definition 3.1. Ein 5-Tupel $\sigma = (S, F, R, K, \text{fct})$ ist eine **Signatur**, wenn

- S, F, R, K sind paarweise disjunkte Mengen von **Sorten**, **Funktionssymbolen**, **Relationensymbolen** bzw. **Konstantensymbolen**;
- fct ist eine auf $F \cup R \cup K$ definierte Funktion, die als **Funktionalität** bezeichnet wird;
- für alle $f \in F$ gibt es ein $n \in \mathbb{N}$ mit $\text{fct}(f) \in S^{n+1}$;
- für alle $r \in R$ gibt es ein $n \in \mathbb{N}$ mit $\text{fct}(r) \in S^n$;
- für alle $k \in K$ ist $\text{fct}(k) \in S$.

Bemerkung 3.2. Die Sorten entsprechen den verschiedenen Trägermengen von Strukturen. Die Funktion fct legt die *Typen* der Symbole fest. Wenn $f \in F$ ein Funktionssymbol ist und $\text{fct}(f) = (s_1, \dots, s_n, s_{n+1})$, so bedeutet das, dass f n -stellig ist, seine Argumente aus den mit den Sorten s_1, \dots, s_n bezeichneten Trägermengen bezieht und einen Wert in der mit s_{n+1} bezeichneten Trägermenge liefert.

\mathbb{K} -Vektorräume $V = (V, +, \cdot, 0)$ kann man folgendermaßen durch eine Signatur σ_{VR} erfassen:

Sortenmenge

$$S = \{\text{Vektor}, \text{Skalar}\};$$

als Operationen liegen Körperoperationen $+_{\mathbb{K}}$ und $\cdot_{\mathbb{K}}$, die Vektoraddition $+_V$ und die Skalarmultiplikation \cdot_V vor:

$$F = \{+_{\mathbb{K}}, \cdot_{\mathbb{K}}, +_V, \cdot_V\};$$

Relationen liegen nicht vor:

$$R = \emptyset;$$

es gibt die Körperkonstanten $0_{\mathbb{K}}$ und $1_{\mathbb{K}}$ und den Nullvektor 0_V :

$$K = \{0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_V\}$$

Die Funktionalität der verschiedenen Symbole in $F \cup R \cup K$ ist:

$$\begin{aligned} \text{fct}(+_{\mathbb{K}}) &= (\text{Skalar}, \text{Skalar}, \text{Skalar}) \\ \text{fct}(\cdot_{\mathbb{K}}) &= (\text{Skalar}, \text{Skalar}, \text{Skalar}) \\ \text{fct}(+_V) &= (\text{Vektor}, \text{Vektor}, \text{Vektor}) \\ \text{fct}(\cdot_V) &= (\text{Skalar}, \text{Vektor}, \text{Vektor}) \\ \text{fct}(0_{\mathbb{K}}) &= \text{Skalar} \\ \text{fct}(1_{\mathbb{K}}) &= \text{Skalar} \\ \text{fct}(0_V) &= \text{Vektor}. \end{aligned}$$

Das bedeutet zum Beispiel, dass $+_{\mathbb{K}}$ eine Operation beschreibt, die von \mathbb{K}^2 nach \mathbb{K} geht, dass \cdot_V von $\mathbb{K} \times V$ nach V geht, und dass 0_V ein Element von V ist. Damit ist

$$\sigma_{VR} = (\{\text{Vektor}, \text{Skalar}\}, \{+_{\mathbb{K}}, \cdot_{\mathbb{K}}, +_V, \cdot_V\}, \emptyset, \{0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_V\}, \text{fct}).$$

Die Funktionalitäten von Funktionssymbolen werden auch folgendermaßen geschrieben:

$$\begin{aligned} \text{fct}(+_V) &= (\text{Vektor}, \text{Vektor}) \text{ Vektor} \\ \text{fct}(\cdot_V) &= (\text{Skalar}, \text{Vektor}) \text{ Vektor}. \end{aligned}$$

Derartige Konstrukte finden sich auch in Programmiersprachen. Eine Signatur entspricht der *Klassendeklaration* in einer Programmiersprache wie C++. Wenn man Vektorräume als *Klasse* einführen möchte, würde man unter der Annahme, dass die Datentypen `vektor` und `skalar` vorliegen, etwa schreiben:

```
class Vektorraum
{
public:
vektor Vektoraddition(vektor,vektor);
vektor Skalarmultiplikation(skalar,vektor);
vektor Nullvektor();
};
```

3.2 Strukturen

Eine Signatur kann durch *Strukturen* interpretiert werden. In einer Struktur werden den Symbolen entsprechende Strukturkomponenten zugeordnet.

Definition 3.3. Sei $\sigma = (S, F, R, K, \text{fct})$ eine Signatur. Eine **Struktur** mit Signatur σ oder eine σ -**Struktur** ist ein Tupel

$$\mathfrak{A} = ((A_s)_{s \in S}, (f^A)_{f \in F}, (r^A)_{r \in R}, (k^A)_{k \in K})$$

mit den Eigenschaften:

- für $s \in S$ ist $A_s \neq \emptyset$; jedes A_s ist eine **Trägermenge** der Struktur;
- für $f \in F$ mit $\text{fct}(f) = (s_1, \dots, s_n, s_{n+1})$ ist f^A eine **Funktion**:

$$f^A: A_{s_1} \times \dots \times A_{s_n} \rightarrow A_{s_{n+1}};$$

- für $r \in R$ mit $\text{fct}(r) = (s_1, \dots, s_n)$ ist r^A eine **Relation**:

$$r^A \subseteq A_{s_1} \times \dots \times A_{s_n};$$

d) für $k \in K$ mit $\text{fct}(k) = s$ ist k^A eine **Konstante**:

$$k^A \in A_s.$$

Die Struktur \mathfrak{A} kann als Zuordnung von Symbolen zu (mathematischen) Objekten geeigneten Typs aufgefasst werden:

$$s \mapsto A_s, f \mapsto f^A, r \mapsto r^A, k \mapsto k^A.$$

Im Fall der oben diskutierten Vektorräume kann ein gewöhnlicher Vektorraum als Struktur mit der Signatur

$$\sigma_{\text{VR}} = (\{\text{Vektor}, \text{Skalar}\}, \{+_{\text{Sk}}, \cdot_{\text{Sk}}, +_{\text{Vk}}, \cdot_{\text{Vk}}\}, \emptyset, \{0_{\text{Sk}}, 1_{\text{Sk}}, 0_{\text{Vk}}\}, \text{fct})$$

aufgefasst werden; der 2-dimensionale \mathbb{R} -Vektorraum \mathbb{R}^2 wäre in dieser Formatierung:

$$\mathbb{R}^2 = (\{\mathbb{R}^2, \mathbb{R}\}, \{+_{\mathbb{R}}, \cdot_{\mathbb{R}}, +_{\mathbb{R}^2}, \cdot_{\mathbb{R}^2}\}, \emptyset, \{0, 1, \begin{pmatrix} 0 \\ 0 \end{pmatrix}\}.$$

Formulierungen dieser Art enthalten in der Regel viele redundante Elemente, daher werden sie etwa verkürzt zu dem bekannten

$$\mathbb{R}^2 = (\mathbb{R}^2, +_{\mathbb{R}^2}, \cdot_{\mathbb{R}^2}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}).$$

Da der Skalarkörper \mathbb{R} fixiert ist, werden seine Komponenten unterdrückt; auch die leere Menge der Relationen braucht üblicherweise nicht angegeben zu werden. Weiterhin wird eine Struktur häufig durch (eine) ihre(r) Trägermengen bezeichnet.

Dennoch sollte man sich bewusst sein, dass die üblichen Abkürzungsschreibweisen bei Bedarf zu vollständigen, eindeutigen Formulierungen ausgedehnt werden können.

Aufgabe 3.1. Definieren Sie eine Signatur σ_{arith} der *Arithmetik*, die für verschiedene Zahlssysteme wie $\mathbb{N}, \mathbb{Q}, \mathbb{R}$ mit Addition und Multiplikation adäquat ist. Stellen Sie die Zahlbereiche als Strukturen mit dieser Signatur dar.

Aufgabe 3.2. Definieren Sie Signaturen für *Booleschen Algebren* entsprechend der zwei alternativen Definitionen von Boolescher Algebra und stellen Sie die Potenzmengenalgebra $\mathcal{P}(A)$ als Strukturen zu diesen Signaturen dar.

3.3 Substrukturen

Zu einer gegebenen Signatur σ gibt es eine Fülle von σ -Strukturen. In der Linearen Algebra hatten wir etwa eine große Vielfalt von Vektorräumen kennengelernt. Eine Aufgabe der Theorie ist es, durch Klassifizierungen einen Überblick über die Klasse aller Möglichkeiten zu erlangen. Bei Vektorräumen waren hierzu Begriffe wie *Unterraum* und *lineare Abbildung* eingeführt worden. Wir wollen derartige Definitionen ganz allgemein studieren:

Definition 3.4. Sei $\sigma = (S, F, R, K, \text{fct})$ eine Signatur und seien

$$\mathfrak{A} = ((A_s)_{s \in S}, (f^A)_{f \in F}, (r^A)_{r \in R}, (k^A)_{k \in K})$$

und

$$\mathfrak{B} = ((B_s)_{s \in S}, (f^B)_{f \in F}, (r^B)_{r \in R}, (k^B)_{k \in K})$$

σ -Strukturen. Dann ist \mathfrak{A} **Substruktur** oder **Unterstruktur** von \mathfrak{B} , wenn:

- a) für $s \in S$ ist $A_s \subseteq B_s$;
 b) für $f \in F$ mit $\text{fct}(f) = (s_1, \dots, s_n, s_{n+1})$ und $a_1 \in A_{s_1}, \dots, a_n \in A_{s_n}$ ist

$$f^A(a_1, \dots, a_n) = f^B(a_1, \dots, a_n);$$

- c) für $r \in R$ mit $\text{fct}(r) = (s_1, \dots, s_n)$ und $a_1 \in A_{s_1}, \dots, a_n \in A_{s_n}$ ist

$$r^A(a_1, \dots, a_n) \text{ gdw. } r^B(a_1, \dots, a_n);$$

d) für $k \in K$ mit $\text{fct}(k) = s$ ist

$$k^A = k^B.$$

Man schreibt dann auch $\mathfrak{A} \subseteq \mathfrak{B}$.

Eine Substruktur wird durch *Einschränkung* der Trägermengen gegeben; die übrigen Komponenten der Struktur werden entsprechend eingeschränkt. Ein Untervektorraum U eines \mathbb{K} -Vektorraums V ist eine Substruktur im Sinne dieser Definition:

$$\begin{aligned} U &= (\{U, \mathbb{K}\}, \{+\mathbb{K}, \cdot\mathbb{K}, +U, \cdot U\}, \emptyset, \{0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_U\}) \subseteq \\ V &= (\{V, \mathbb{K}\}, \{+\mathbb{K}, \cdot\mathbb{K}, +V, \cdot V\}, \emptyset, \{0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_V\}). \end{aligned}$$

3.4 Homomorphismen

Im Mittelpunkt der Linearen Algebra steht das Studium der *linearen Abbildungen*. Dieses sind Abbildungen zwischen den Trägermengen der Vektoren, die mit beiden Strukturen verträglich sind. Wir formulieren allgemein:

Definition 3.5. Sei $\sigma = (S, F, R, K, \text{fct})$ eine Signatur und seien

$$\mathfrak{A} = ((A_s)_{s \in S}, (f^A)_{f \in F}, (r^A)_{r \in R}, (k^A)_{k \in K})$$

und

$$\mathfrak{B} = ((B_s)_{s \in S}, (f^B)_{f \in F}, (r^B)_{r \in R}, (k^B)_{k \in K})$$

σ -Strukturen. Sei $s \in S$, so dass für alle $s' \in S \setminus \{s\}$ gilt: $A_{s'} = B_{s'}$. Für eine Abbildung $h: A_s \rightarrow B_s$ und $t \in S$ definiere

$$h_t: A_t \rightarrow B_t, h_t = \begin{cases} h, & \text{für } t = s \\ \text{Id}_{A_t}, & \text{für } t \neq s \end{cases}$$

Dann ist $h: A_s \rightarrow B_s$ ein **s-Homomorphismus** von \mathfrak{A} nach \mathfrak{B} , wenn:

a) für $f \in F$ mit $\text{fct}(f) = (s_1, \dots, s_n, s_{n+1})$ und $a_1 \in A_{s_1}, \dots, a_n \in A_{s_n}$ ist

$$f^B(h_{s_1}(a_1), \dots, h_{s_n}(a_n)) = h_{s_{n+1}}(f^A(a_1, \dots, a_n));$$

b) für $r \in R$ mit $\text{fct}(r) = (s_1, \dots, s_n)$ und $a_1 \in A_{s_1}, \dots, a_n \in A_{s_n}$ ist

$$r^B(h_{s_1}(a_1), \dots, h_{s_n}(a_n)) \text{ gdw. } r^A(a_1, \dots, a_n);$$

c) für $k \in K$ mit $\text{fct}(k) = s_1$ ist

$$k^B = h_{s_1}(k^A).$$

Man schreibt dann auch $h: \mathfrak{A} \rightarrow_s \mathfrak{B}$ oder einfacher $h: \mathfrak{A} \rightarrow \mathfrak{B}$.

Beispiel 3.6. Sei

$$\sigma_{\text{VR}} = (\{\text{Vektor}, \text{Skalar}\}, \{+\text{Sk}, \cdot\text{Sk}, +\text{Vektor}, \cdot\text{Vektor}\}, \emptyset, \{0_{\text{Sk}}, 1_{\text{Sk}}, 0_{\text{Vektor}}\}, \text{fct})$$

die Signatur der Vektorräume und seien

$$U = (\{U, \mathbb{K}\}, \{+\mathbb{K}, \cdot\mathbb{K}, +U, \cdot U\}, \emptyset, \{0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_U\})$$

und

$$V = (\{V, \mathbb{K}\}, \{+\mathbb{K}, \cdot\mathbb{K}, +V, \cdot V\}, \emptyset, \{0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_V\})$$

\mathbb{K} -Vektorräume. Eine Abbildung $h: U \rightarrow V$ ist genau dann ein Vektor-Homomorphismus, wenn h eine lineare Abbildung von \mathbb{K} -Vektorräumen ist. Es gilt $h_{\text{Vektor}} = h$ und $h_{\text{Skalar}} = \text{Id}_{\mathbb{K}}$. Die Homomorphiseigenschaft für das Skalarprodukt eines Skalars $\lambda \in \mathbb{K}$ und eines Vektors $u \in U$ bedeutet:

$$\cdot_V (h_{\text{Skalar}}(\lambda), h_{\text{Vektor}}(u)) = h_{\text{Vektor}}(\cdot_U(\lambda, u))$$

Dies kann äquivalent zur multiplikativen Linearität von h umgeformt werden:

$$\begin{aligned}\cdot_V(\lambda, h(u)) &= h(\cdot_U(\lambda, u)) \\ \lambda \cdot_V h(u) &= h(\lambda \cdot_U u) \\ \lambda \cdot h(u) &= h(\lambda \cdot u).\end{aligned}$$

Wir führen Begriffe zur Beschreibung von Homomorphismen ein:

Definition 3.7. Sei $h: \mathfrak{A} \rightarrow_s \mathfrak{B}$ ein s -Homomorphismus zwischen σ -Strukturen \mathfrak{A} und \mathfrak{B} . Dann ist h eine Abbildung $h: A_s \rightarrow B_s$ zwischen den Trägermengen A_s und B_s . Wir definieren:

- a) h ist ein **Monomorphismus**, wenn h injektiv ist;
- b) h ist ein **Epimorphismus**, wenn h surjektiv ist;
- c) h ist ein **Isomorphismus**, wenn h bijektiv ist;
- d) h ist ein **Endomorphismus**, wenn $A_s = B_s$;
- e) h ist ein **Automorphismus**, wenn h bijektiv und $A_s = B_s$ ist.

e) **Aussagen:** Die Menge Aus^σ der Sprache L^σ wird rekursiv definiert:

- i. jede relationale Aussage ist eine Aussage;
- ii. wenn φ eine Aussage ist, so ist auch $\neg\varphi$ („nicht φ “) eine Aussage;
- iii. wenn φ und ψ Aussagen sind, so sind auch

$$\begin{array}{ll} (\varphi \wedge \psi) & \text{„}\varphi \text{ und } \psi\text{“} \\ (\varphi \vee \psi) & \text{„}\varphi \text{ oder } \psi\text{“} \\ (\varphi \rightarrow \psi) & \text{„}\varphi \text{ impliziert } \psi\text{“} \\ (\varphi \leftrightarrow \psi) & \text{„}\varphi \text{ ist äquivalent zu } \psi\text{“} \end{array}$$

Aussagen;

- iv. wenn φ eine Aussage ist und v_n^s eine Variable, so sind auch

$$\begin{array}{ll} \forall v_n^s \varphi & \text{„für alle } v_n^s \text{ gilt } \varphi\text{“} \\ \exists v_n^s \varphi & \text{„es gibt ein } v_n^s \text{ mit } \varphi\text{“} \end{array}$$

Aussagen.

Beispiel 4.2. Die Sprache der Vektorräume. Wir wenden die Definition an und führen verschiedene abkürzende Schreibweisen ein. Sei

$$\begin{aligned} \sigma_{VR} &= (\{\text{Vektor}, \text{Skalar}\}, \{+_{Sk}, \cdot_{Sk}, +_{Vk}, \cdot_{Vk}\}, \emptyset, \{0_{Sk}, 1_{Sk}, 0_{Vk}\}, \text{fct}) \\ &= (\text{Vektor}, \text{Skalar}; +_{Sk}, \cdot_{Sk}, +_{Vk}, \cdot_{Vk}, 0_{Sk}, 1_{Sk}, 0_{Vk}, \text{fct}) \end{aligned}$$

die Signatur der Vektorräume. Dann hat man:

- *Variablen:* v_n^{Vektor} und v_n^{Skalar} , die üblicherweise durch besondere Buchstaben x, y, z, \dots für Vektor-Variablen und λ, μ, ν, \dots für Skalar-Variablen bezeichnet werden;
- *Terme:* $+_{Sk}(\lambda, \mu)$, $\cdot_{Sk}(\lambda, \mu)$, $+_{Vk}(x, y)$, $\cdot_{Vk}(\lambda, x)$ und komplexere Terme der Gestalt

$$+_{Vk}(\cdot_{Vk}(\lambda, x), \cdot_{Vk}(\mu, y)).$$

Die Terme sind in üblichen arithmetischen Infix-Schreibweisen mit Klammersetzung besser zu verstehen: $\lambda +_{Sk} \mu$, $\lambda \cdot_{Sk} \mu$, ... und

$$(\lambda \cdot_{Vk} x) +_{Vk} (\mu \cdot_{Vk} y).$$

Da die Variablen den Typ der auf sie anwendbaren Funktionen bestimmen, lässt sich ableiten, ob an bestimmten Stellen des Terms $+_{Sk}$ oder \cdot_{Vk} stehen muss. Daher kann man die Indizes Sk oder Vk in der Regel fortlassen:

$$(\lambda \cdot x) + (\mu \cdot y).$$

Mit Konventionen wie „Punktrechnung geht vor Strichrechnung“ kann man weiterhin Klammern um die Skalar-Multiplikationen fortlassen. Außerdem wird der Mal-Punkt \cdot häufig eliminiert:

$$\lambda x + \mu y.$$

- *Relationale Aussagen:* Als relationales Symbol liegt nur das Gleichheitszeichen $=$ vor, daher kommen nur Aussagen der Art

$$+_{Vk}(\cdot_{Vk}(\lambda, x), \cdot_{Vk}(\mu, y)) = 0_{Vk}$$

in Frage, die selbstverständlich vertrauter als

$$\lambda x + \mu y = 0$$

geschrieben werden. Man beachte, dass die Konstante 0 auf der rechten Seite vom Typ Vektor sein muss, da die linke Seite der Gleichung diesen Typ hat. Daher können wir einfach 0 statt 0_{Vk} schreiben.

- *Aussagen:* Die Vektorraum-Axiome sind eine endliche Menge von Aussagen, die für alle Vektorräume gefordert werden. In der formalen Sprache lautet eines der Assoziativgesetze:

$$\begin{aligned} \forall v_0^{\text{Skalar}} \forall v_1^{\text{Skalar}} \forall v_0^{\text{Vektor}} \quad & \cdot_{\text{Vk}} (v_0^{\text{Skalar}}, \cdot_{\text{Vk}} (v_1^{\text{Skalar}}, v_0^{\text{Vektor}})) \\ & = \cdot_{\text{Vk}} (\cdot_{\text{Sk}} (v_0^{\text{Skalar}}, v_1^{\text{Skalar}}), v_0^{\text{Vektor}}). \end{aligned}$$

Unter der Benutzung obiger Konventionen wird hieraus:

$$\forall \lambda \forall \mu \forall x \lambda \cdot (\mu \cdot x) = (\lambda \cdot \mu) \cdot x.$$

Assoziativgesetze erlauben anschließend das Fortlassen weiterer Klammern, da sie gerade die Unabhängigkeit von Werten von der Klammersetzung zum Inhalt haben. Manchmal deutet man die Typen der Variablen durch zusätzliche mengentheoretische Formulierungen an. Für einen \mathbb{K} -Vektorraum V schreibt man das Assoziativgesetz auch als:

$$\forall \lambda \in \mathbb{K} \forall \mu \in \mathbb{K} \forall x \in V \lambda \cdot (\mu \cdot x) = (\lambda \cdot \mu) \cdot x.$$

Bemerkung 4.3. In der Informatik besteht die ständige Notwendigkeit, adäquate Sprachen zu definieren. In der verbreiteten BACKUS-NAUR-Form (BNF) lässt sich die obige Sprache folgendermaßen beschreiben:

Variablen:	$V ::= v_n^s$	$n \in \mathbb{N}, s \in S$
Terme:	$T ::= V \mid f(V_1, \dots, V_n)$	$f \in F$
Relationale Aussagen:	$RA ::= T_1 = T_2 \mid r(T_1, \dots, T_n)$	$r \in R$
Aussagen:	$A ::= RA \mid \neg A \mid (A_1 \wedge A_2) \mid (A_1 \vee A_2) \mid (A_1 \rightarrow A_2) \mid (A_1 \leftrightarrow A_2)$	

Allerdings werden hier Details wie die Beachtung von Typen bei der Bildung von Termen und Aussagen unterdrückt. Durch die BNF-Definition der formalen Sprache L^σ wird die Ähnlichkeit zu Programmiersprachen betont.

4.2 Semantik

Mit der Sprache L^σ kann man Eigenschaften von σ -Strukturen formulieren. Durch Interpretation der Elemente der Sprache in einer σ -Struktur kann man überprüfen, ob die Struktur die Eigenschaft erfüllt. Die Definition erfolgt rekursiv über den Aufbau der formalen Sprache. Neben den Symbolen aus σ benutzt die formale Sprache auch Variablen für Elemente der Trägermengen. Die Variablen werden interpretiert, indem sie mit Werten aus den Trägermengen belegt werden.

Definition 4.4. Sei $\sigma = (S, F, R, K, \text{fct})$ eine Signatur mit zugehöriger Sprache L^σ . Sei

$$\mathfrak{A} = ((A_s)_{s \in S}, (f^A)_{f \in F}, (r^A)_{r \in R}, (k^A)_{k \in K})$$

eine σ -Struktur. Die Interpretation von L^σ in \mathfrak{A} wird schrittweise definiert:

- Eine **Belegung** in \mathfrak{A} ist eine Funktion

$$\beta: \{v_n^s \mid n \in \mathbb{N}, s \in S\} \rightarrow \bigcup_{s \in S} A_s,$$

so dass für alle $n \in \mathbb{N}$ und $s \in S$ gilt $\beta(v_n^s) \in A_s$. Die Belegung interpretiert also Variablen entsprechend ihrem Typ.

Es ist manchmal wichtig, den Wert einer Belegung β an einer Variablen $v_n^{s'}$ zu einem gegebenen $a \in A_s$ zu modifizieren. Definiere dazu eine modifizierte Belegung

$$\beta \frac{a}{v_n^{s'}}: \{v_n^s \mid n \in \mathbb{N}, s \in S\} \rightarrow \bigcup_{s \in S} A_s$$

durch

$$\beta \frac{a}{v_n^s} (v_n^s) = \begin{cases} \beta(v_n^s), & \text{falls } v_n^s \neq v_n^{s'} \\ a, & \text{falls } v_n^s = v_n^{s'} \end{cases}$$

- Ein σ -**Modell** ist ein geordnetes Paar $\mathfrak{M} = (\mathfrak{A}, \beta)$ aus einer σ -Struktur \mathfrak{A} und einer Belegung β in \mathfrak{A} . Für die weiteren Definitionen sei ein Modell $\mathfrak{M} = (\mathfrak{A}, \beta)$ fixiert.
- Für eine Term $t \in T^\sigma$ der Sprache L^σ definiere die **Interpretation** $\mathfrak{M}(t)$ im Modell \mathfrak{M} durch Rekursion über den Aufbau von t :
 - i. für eine Variable v_n^s ist $\mathfrak{M}(v_n^s) = \beta(v_n^s)$;
 - ii. für ein Konstantensymbol $k \in K$ ist $\mathfrak{M}(k) = k^A$;
 - iii. für ein n -stelliges Funktionssymbol $f \in F$ und Terme $t_1, \dots, t_n \in T^\sigma$ ist

$$\mathfrak{M}(f(t_1, \dots, t_n)) = f^A(\mathfrak{M}(t_1), \dots, \mathfrak{M}(t_n)).$$
- Für eine Aussage $\varphi \in L^\sigma$ definiere, dass \mathfrak{M} ein **Modell von** φ ist, $\mathfrak{M} \models \varphi$, durch Rekursion über den Aufbau von φ :
 - i. für Terme $t_1, t_2 \in T^\sigma$ setze

$$\mathfrak{M} \models t_1 = t_2 \text{ gdw. } \mathfrak{M}(t_1) = \mathfrak{M}(t_2);$$
 - ii. für ein n -stelliges Relationssymbol $r \in R$ und Terme $t_1, \dots, t_n \in T^\sigma$ setze

$$\mathfrak{M} \models r(t_1, \dots, t_n) \text{ gdw. } r^A(\mathfrak{M}(t_1), \dots, \mathfrak{M}(t_n));$$
 - iii. $\mathfrak{M} \models \neg\varphi$ gdw. **nicht** $\mathfrak{M} \models \varphi$;
 - iv. $\mathfrak{M} \models (\varphi \wedge \psi)$ gdw. $\mathfrak{M} \models \varphi$ **und** $\mathfrak{M} \models \psi$;
 - v. $\mathfrak{M} \models (\varphi \vee \psi)$ gdw. $\mathfrak{M} \models \varphi$ **oder** $\mathfrak{M} \models \psi$;
 - vi. $\mathfrak{M} \models (\varphi \rightarrow \psi)$ gdw. $\mathfrak{M} \models \varphi$ **impliziert** $\mathfrak{M} \models \psi$;
 - vii. $\mathfrak{M} \models (\varphi \leftrightarrow \psi)$ gdw. $\mathfrak{M} \models \varphi$ **ist äquivalent zu** $\mathfrak{M} \models \psi$;
 - viii. $\mathfrak{M} \models \forall v_n^s \varphi$ gdw. **für alle** $a \in A_s$ gilt $\mathfrak{M} \frac{a}{v_n^s} = (\mathfrak{M}, \beta \frac{a}{v_n^s}) \models \varphi$;
 - ix. $\mathfrak{M} \models \exists v_n^s \varphi$ gdw. **es existiert ein** $a \in A_s$ mit $\mathfrak{M} \frac{a}{v_n^s} = (\mathfrak{M}, \beta \frac{a}{v_n^s}) \models \varphi$.

Man sagt auch \mathfrak{M} **erfüllt** φ oder φ **gilt in** \mathfrak{M} für $\mathfrak{M} \models \varphi$.

Bemerkung 4.5. Bei den Definitionen wird benutzt, dass Terme und Aussagen der Sprache *eindeutig lesbar* sind, d.h. für jeden Term gibt es genau eine Weise, wie er in konstituierende Teilterme zerfällt; die Interpretation des Terms wird dann aus den Interpretationen dieser Teilterme zusammengesetzt. Entsprechendes gilt für Aussagen. Formale Sprachen sind so zu konstruieren, dass eindeutige Lesbarkeit gegeben ist. Programme und ihre Konstituenten sollen eindeutig lesbar sein, andernfalls gäbe es Probleme mit der Determiniertheit von Übersetzungen und Programmausführungen.

Beispiel 4.6. Die Definition von Term-Interpretationen und Modell-Beziehung stimmen mit dem bisherigen informellen Gebrauch überein. Dies liegt an den naheliegenden Definitionen für aussagenlogische Verknüpfungen und dem natürlichen Zusammenspiel der verschiedenen Definitionen.

Es sei beispielsweise

$$\sigma_{VR} = (\forall k, \text{Sk}; +_{\text{Sk}}, \cdot_{\text{Sk}}, +_{\forall k}, \cdot_{\forall k}, 0_{\text{Sk}}, 1_{\text{Sk}}, 0_{\forall k}, \text{fct})$$

die Signatur der Vektorräume und

$$\mathfrak{V} = (V, \mathbb{K}; +_{\mathbb{K}}, \cdot_{\mathbb{K}}, +_V, \cdot_V, 0_{\mathbb{K}}, 1_{\mathbb{K}}, 0_V)$$

ein Vektorraum mit Skalarkörper \mathbb{K} . Dann ist

$$\mathfrak{M} \models \forall \lambda \forall x \forall y \lambda \cdot (x + y) = \lambda \cdot x + \lambda \cdot y$$

$$\text{gdw. } \mathfrak{M} \models \forall v_0^{\text{Sk}} \forall v_0^{\text{Vk}} \forall v_1^{\text{Vk}} v_0^{\text{Sk}} \cdot v_k (v_0^{\text{Vk}} + v_k v_1^{\text{Vk}}) = v_0^{\text{Sk}} \cdot v_k v_0^{\text{Vk}} + v_k v_0^{\text{Sk}} \cdot v_k v_1^{\text{Vk}}$$

gdw. für alle $a \in \mathbb{K}$ gilt:

$$\mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \models \forall v_0^{\text{Vk}} \forall v_1^{\text{Vk}} v_0^{\text{Sk}} \cdot v_k (v_0^{\text{Vk}} + v_k v_1^{\text{Vk}}) = v_0^{\text{Sk}} \cdot v_k v_0^{\text{Vk}} + v_k v_0^{\text{Sk}} \cdot v_k v_1^{\text{Vk}}$$

gdw. für alle $a \in \mathbb{K}$ gilt: für alle $b \in V$ gilt:

$$\mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \models \forall v_1^{\text{Vk}} v_0^{\text{Sk}} \cdot v_k (v_0^{\text{Vk}} + v_k v_1^{\text{Vk}}) = v_0^{\text{Sk}} \cdot v_k v_0^{\text{Vk}} + v_k v_0^{\text{Sk}} \cdot v_k v_1^{\text{Vk}}$$

gdw. für alle $a \in \mathbb{K}$ gilt: für alle $b \in V$ gilt: für alle $c \in V$ gilt:

$$\mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} \models v_0^{\text{Sk}} \cdot v_k (v_0^{\text{Vk}} + v_k v_1^{\text{Vk}}) = v_0^{\text{Sk}} \cdot v_k v_0^{\text{Vk}} + v_k v_0^{\text{Sk}} \cdot v_k v_1^{\text{Vk}}$$

gdw. für alle $a \in \mathbb{K}$ gilt: für alle $b \in V$ gilt: für alle $c \in V$ gilt:

$$\mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_0^{\text{Sk}} \cdot v_k (v_0^{\text{Vk}} + v_k v_1^{\text{Vk}})) = \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_0^{\text{Sk}} \cdot v_k v_0^{\text{Vk}} + v_k v_0^{\text{Sk}} \cdot v_k v_1^{\text{Vk}})$$

gdw. für alle $a \in \mathbb{K}$ gilt: für alle $b \in V$ gilt: für alle $c \in V$ gilt:

$$\begin{aligned} \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_0^{\text{Sk}}) \cdot v_k (v_0^{\text{Vk}} + v_k v_1^{\text{Vk}}) + v_k \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_1^{\text{Vk}}) &= \\ = \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_0^{\text{Sk}}) \cdot v_k \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_0^{\text{Vk}}) + v_k \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_0^{\text{Sk}}) \cdot v_k \mathfrak{M} \models \frac{a}{v_0^{\text{Sk}}} \frac{b}{v_0^{\text{Vk}}} \frac{c}{v_1^{\text{Vk}}} (v_1^{\text{Vk}}) & \end{aligned}$$

gdw. für alle $a \in \mathbb{K}$ und alle $b, c \in V$ gilt:

$$a \cdot v_k (b + v_k c) = a \cdot v_k b + v_k a \cdot v_k c.$$

4.3 Allgemeingültige Aussagen

Es gibt enge Zusammenhänge zwischen der syntaktischen Form von Aussagen und ihrer Gültigkeit in Modellen. Tatsächlich lassen sich Beweise vollkommen *formal* auf der Ebene der Syntax führen. Wir beginnen mit einfachen Überlegungen:

Definition 4.7. Sei σ eine Signatur und $\varphi \in L^\sigma$ eine Aussage.

- a) φ ist **allgemeingültig** oder eine **Tautologie**, wenn jedes σ -Modell ein Modell von φ ist;
- b) φ ist **erfüllbar**, wenn es ein σ -Modell von φ gibt.

Satz 4.8. Folgende Aussagen sind allgemeingültig für alle Terme $t, t_1, t_2, t_3 \in T^\sigma$ und alle Aussagen $\varphi, \psi, \chi \in L^\sigma$.

- a) $t = t$
- b) $t_1 = t_2 \rightarrow t_2 = t_1$
- c) $t_1 = t_2 \wedge t_2 = t_3 \rightarrow t_1 = t_3$
- d) $\varphi \rightarrow (\psi \rightarrow \varphi \wedge \psi)$
- e) $\varphi \wedge \psi \rightarrow \varphi$ und $\varphi \wedge \psi \rightarrow \psi$
- f) $\varphi \rightarrow \varphi \vee \psi$ und $\psi \rightarrow \varphi \vee \psi$
- g) $(\varphi \rightarrow \chi) \rightarrow ((\psi \rightarrow \chi) \rightarrow ((\varphi \vee \psi) \rightarrow \chi))$
- h) $(\neg \varphi \rightarrow \perp) \rightarrow \varphi$

Beweis. Betrachte ein σ -Modell $\mathfrak{M} = (\mathfrak{A}, \beta)$.

b) Angenommen $\mathfrak{M} \models t_1 = t_2$. Dann ist $\mathfrak{M}(t_1) = \mathfrak{M}(t_2)$. Wegen der Symmetrie der Gleichheit ist $\mathfrak{M}(t_2) = \mathfrak{M}(t_1)$. Also ist $\mathfrak{M} \models t_2 = t_1$. Danach gilt: $\mathfrak{M} \models t_1 = t_2$ impliziert $\mathfrak{M} \models t_2 = t_1$, und $\mathfrak{M} \models (t_1 = t_2 \rightarrow t_2 = t_1)$.

h) Angenommen $\mathfrak{M} \models (\neg \varphi \rightarrow \neg v_0 = v_0)$. Angenommen $\mathfrak{M} \models \neg \varphi$. Dann ist $\mathfrak{M} \models \neg v_0 = v_0$. Also ist $\beta(v_0) \neq \beta(v_0)$, Widerspruch. Danach ist $\mathfrak{M} \models \neg \varphi$ nicht möglich, und wir erhalten $\mathfrak{M} \models \varphi$. Danach gilt: $\mathfrak{M} \models (\neg \varphi \rightarrow \neg v_0 = v_0)$ impliziert $\mathfrak{M} \models \varphi$, und $\mathfrak{M} \models ((\neg \varphi \rightarrow \neg v_0 = v_0) \rightarrow \varphi)$. \square

Man beachte, dass dieser Satz über die Allgemeingültigkeit von Aussagen nur die syntaktische Gestalt der Aussagen betrachtet, unabhängig von der Bedeutung von t oder φ . Die bewiesenen Allgemeingültigkeiten entsprechen auch Schritten in (detaillierten) Beweisen. Man kann z.B. d) lesen als: *wenn φ bewiesen ist und ψ bewiesen ist, dann kann man $(\varphi \wedge \psi)$ beweisen*. In diesem Sinne entsprechen e)-h) den Beweismethoden durch *Konjunktionselimination*, *Disjunktion*, *Fallunterscheidung* bzw. *Widerspruch*. Diese Beweismethoden erscheinen auch in den *formalen Beweisen* des nächsten Kapitels.

Kapitel 5

Aussagenlogische formale Beweise

In ausführlichen Beweisen besitzen viele Beweisschritte einen formalen, rechnerischen Charakter. Der Schritt von den Voraussetzungen φ und ψ zu der Konjunktion $\varphi \wedge \psi$ ist vollkommen unabhängig von der Bedeutung oder dem Wahrheitswert von φ und ψ ; die Schlussfolgerung $\varphi \wedge \psi$ lässt sich durch *syntaktisches Operieren* mit den Zeichenreihen φ und ψ bilden. Dies motiviert die Erwartung, dass ein vollkommen ausführlicher Beweis aus einer Folge von Aussagen besteht, die sich jeweils aus früheren Aussagen mit Hilfe syntaktischer Rechenoperationen ergeben.

Wir wollen dieses für den aussagenlogischen Teil unserer Logik demonstrieren. Wir definieren formale Beweise als *Texte* bestimmter Gestalt. Die Struktur eines korrekten Beweises ist so einfach, dass sie automatisch auf Richtigkeit geprüft werden kann (*proof checking*). Wir benutzen den Proofchecker **Naproche**, um Beweise zu überprüfen.

Wir betrachten ein Beispiel, in dem die Tautologie (= allgemeingültige Aussage) $\varphi \wedge (\varphi \rightarrow \psi) \rightarrow \psi$ (*modus ponens*) bewiesen wird:

Satz. $\varphi \wedge (\varphi \rightarrow \psi) \rightarrow \psi$.

Beweis. Sei $\varphi \wedge (\varphi \rightarrow \psi)$. φ . $\varphi \rightarrow \psi$. ψ . Also $\varphi \wedge (\varphi \rightarrow \psi) \rightarrow \psi$.

Qed.

Dieser Beweis besteht aus einer Abfolge von Aussagen, die durch die Schlüsselwörter (*Keywords*) „Satz“, „Beweis“, „Sei“, „Also“, „Qed“ strukturiert wird. In dem eigentlichen Argument werden Aussagen aus früheren Aussagen nach bestimmten Regeln generiert. Z.B. ergeben sich die Aussagen φ und $\varphi \wedge \psi$ durch *Elimination* der Konjunktion \wedge aus der Aussage $\varphi \wedge (\varphi \rightarrow \psi)$.

Der Beweis enthält weiterhin einen *Rahmen (frame)*

Sei $\varphi \wedge (\varphi \rightarrow \psi)$. φ . $\varphi \rightarrow \psi$. ψ . Also $\varphi \wedge (\varphi \rightarrow \psi) \rightarrow \psi$.

Die erste Aussage $\varphi \wedge (\varphi \rightarrow \psi)$ nach Öffnen des Rahmens durch „Sei“ ist die *Annahme* des Rahmens; die weiteren Aussagen ergeben sich aus dieser Annahme (und eventuellen früheren Aussagen im Beweis) auf Grund der Ableitungsregeln. Nach Schließen des Rahmens durch das Schlüsselwort „Also“ kann aus diesem Rahmen die Implikation

$$\varphi \wedge (\varphi \rightarrow \psi) \rightarrow \psi$$

geschlossen werden. Allgemeiner gilt: aus dem Rahmen

Sei φ_1 . φ_2 φ_r . Also

kann auf

$$\varphi_1 \rightarrow \varphi_r$$

geschlossen werden. Diese Methode der \rightarrow -*Einführung* ist in Beweisen sehr verbreitet.

Darüber hinaus können weitere Regeln zur Gewinnung von Aussagen angewendet werden, mit denen aussagenlogische Verknüpfungen eingeführt oder eliminiert werden. In dem Beispiel wurde aus den Aussagen φ und $\varphi \rightarrow \psi$ auf ψ geschlossen. Diese Regel notieren wir als

$$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

5.1 Formale Beweise mit \wedge , \rightarrow und \leftrightarrow

In einem formalen Beweis wird aus den lokal zur Verfügung stehenden Voraussetzungen mit Hilfe von *Schlussregeln* die nächste Zeile gebildet. Wir diskutieren verschiedene Schlussregeln und entsprechend gebildete Beweise.

Schlussregeln werden allgemein in der Form

$$\frac{\chi_1 \ \chi_2 \ \dots \ \chi_r}{\psi}$$

notiert: aus den Voraussetzungen $\chi_1, \chi_2, \dots, \chi_r$ kann auf ψ geschlossen werden. Dieses Schließen ist im Prinzip rein formal. Wir werden Regeln betrachten, die dem üblichen logischen Schließen entsprechen. Die Regeln führen logische Symbole in Aussagen ein, oder sie eliminieren Symbole. Wir betrachten zunächst Regeln, die die aussagenlogischen Symbole \wedge und \rightarrow betreffen.

\wedge -Regeln:

$$\begin{aligned} \wedge\text{-Einführung: } & \wedge I \frac{\varphi \quad \psi}{(\varphi \wedge \psi)} \\ \wedge\text{-Elimination: } & \wedge E \frac{(\varphi \wedge \psi)}{\varphi} \\ \wedge\text{-Elimination: } & \wedge E \frac{(\varphi \wedge \psi)}{\psi} \end{aligned}$$

\rightarrow -Regeln:

$$\rightarrow\text{-Elimination: } \rightarrow E \frac{\varphi \quad (\varphi \rightarrow \psi)}{\psi}$$

\leftrightarrow -Regel:

Die Aussage $(\varphi \leftrightarrow \psi)$ wird als *Abkürzung* für $((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$ benutzt. Damit wird \leftrightarrow durch die Regeln für \wedge und \rightarrow erfasst.

Wir benutzen diese Regeln in den folgenden Beweisen von Tautologien.

Satz. $\varphi \rightarrow (\psi \rightarrow \varphi)$.

Beweis.

Sei φ .

Sei ψ .

φ .

Also $\psi \rightarrow \varphi$.

Also $\varphi \rightarrow (\psi \rightarrow \varphi)$.

Qed.

Dieser Text wird vom System **Naproche** akzeptiert und es erfolgt die Ausgabe: „The proof is accepted!“ Wir betrachten weitere Beispiele.

Satz. $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$.

Beweis.

Sei φ .

Sei ψ .

$\varphi \wedge \psi$.

Also $\psi \rightarrow (\varphi \wedge \psi)$.

Also $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$.

Qed.

Der Beweisprüfer findet selbstständig die jeweils benutzten Regeln. Er enthält somit eine kleine Komponente *automatischen Beweisens*. Wir wollen das Beispiel noch um die benutzten Regeln ergänzen:

Satz. $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$.

Beweis.

Sei φ . *Einführung einer Annahme*
 Sei ψ . *Einführung einer Annahme*
 $\varphi \wedge \psi$. \wedge -*Einführung*
 Also $\psi \rightarrow (\varphi \wedge \psi)$. \rightarrow -*Einführung*
 Also $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$. \rightarrow -*Einführung*
 Qed.

Man beachte, dass mit der \rightarrow -Einführung auch jeweils eine zuvor eingeführte Annahme abgeschlossen wird. Um die Regelbenutzung noch genauer zu kennzeichnen, wäre anzugeben, auf welche Vorgängerzeilen sich die jeweilige Regel bezieht:

1. Satz. $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$.
2. Beweis.
3. Sei φ . *Einführung einer Annahme*
4. Sei ψ . *Einführung einer Annahme*
5. $\varphi \wedge \psi$. \wedge -*Einführung mit 3,4*
6. Also $\psi \rightarrow (\varphi \wedge \psi)$. \rightarrow -*Einführung mit 4*
7. Also $\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))$. \rightarrow -*Einführung mit 3*
8. Qed.

Wir führen noch ein paar Beispiele zu \wedge , \rightarrow , \leftrightarrow aus:

Satz. $\varphi \wedge \psi \rightarrow \psi \wedge \varphi$.
 Beweis.
 Sei $\varphi \wedge \psi$.
 φ .
 ψ .
 $\psi \wedge \varphi$.
 Also $\varphi \wedge \psi \rightarrow \psi \wedge \varphi$.
 Qed.

Satz. $\varphi \wedge \varphi \leftrightarrow \varphi$.
 Beweis.
 Sei $\varphi \wedge \varphi$.
 φ .
 Also $\varphi \wedge \varphi \rightarrow \varphi$.
 Sei φ .
 $\varphi \wedge \varphi$.
 Also $\varphi \rightarrow \varphi \wedge \varphi$.
 $\varphi \wedge \varphi \leftrightarrow \varphi$.
 Qed.

5.2 Beweisregeln für \vee und \neg

Wir führen Regeln für die übrigen aussagenlogischen Verknüpfungen ein. Wir benutzen entsprechend der in **Naproche** implementierten Logik die zusätzliche Aussagenkonstante \perp (Bottom) für „falsch“. Die Konstante \perp kann auch mit „Widerspruch“ bezeichnet werden.

\vee -Regeln:

$$\begin{array}{ll} \vee\text{-Einführung:} & \vee I \frac{\varphi}{(\varphi \vee \psi)} \\ \vee\text{-Einführung:} & \vee I \frac{\psi}{(\varphi \vee \psi)} \\ \text{Fallunterscheidung } (\vee\text{-Elimination}): & \vee E \frac{(\varphi \vee \psi) \quad (\varphi \rightarrow \chi) \quad (\psi \rightarrow \chi)}{\chi} \end{array}$$

\neg -Regeln:

$$\perp\text{-Einführung:} \quad \perp I \quad \frac{\varphi \quad \neg\varphi}{\perp}$$

$$\perp\text{-Elimination, Widerspruchsregel: Wid} \quad \frac{(\neg\varphi \rightarrow \perp)}{\varphi}$$

1. Ein formaler Beweis der Tautologie $\neg\varphi \vee \psi \leftrightarrow (\varphi \rightarrow \psi)$ unter Benutzung von Hilfssätzen (Lemmas).

Lemma. $\neg\varphi \rightarrow (\varphi \rightarrow \psi)$.

Beweis.

Sei $\neg\varphi$.

Sei φ .

Widerspruch.

ψ .

Also $\varphi \rightarrow \psi$.

Also $\neg\varphi \rightarrow (\varphi \rightarrow \psi)$.

Qed.

Lemma. $\psi \rightarrow (\varphi \rightarrow \psi)$.

Beweis.

Sei ψ .

Sei φ .

ψ .

Also $\varphi \rightarrow \psi$.

Also $\psi \rightarrow (\varphi \rightarrow \psi)$.

Qed.

Wir verknüpfen diese Lemmas durch Fallunterscheidung:

Lemma. $\neg\varphi \vee \psi \rightarrow (\varphi \rightarrow \psi)$.

Beweis. Sei $\neg\varphi \vee \psi$. $\varphi \rightarrow \psi$. Also $\neg\varphi \vee \psi \rightarrow (\varphi \rightarrow \psi)$. Qed.

Lemma. $(\varphi \rightarrow \psi) \rightarrow \neg\varphi \vee \psi$.

Beweis. Sei $\varphi \rightarrow \psi$.

Angenommen $\neg(\neg\varphi \vee \psi)$.

Angenommen $\neg\varphi$.

$\neg\varphi \vee \psi$. Widerspruch.

Also $\neg\varphi \rightarrow \perp$.

φ .

ψ .

$\neg\varphi \vee \psi$. Widerspruch.

Also $\neg(\neg\varphi \vee \psi) \rightarrow \perp$.

$\neg\varphi \vee \psi$.

Also $(\varphi \rightarrow \psi) \rightarrow \neg\varphi \vee \psi$.

Qed.

Satz. $\neg\varphi \vee \psi \leftrightarrow (\varphi \rightarrow \psi)$. Beweis. Qed.

5.3 Beweisansätze

Die Methoden für das formale Beweisen liefern auch Ansätze für das gewöhnliche Beweisen. Um eine Aussage bestimmter Gestalt zu beweisen, erlauben die formalen Regeln, die die Beweisaufgabe in „kleinere“ Beweisaufgaben zerlegen.

- um eine Konjunktion $\varphi \wedge \psi$ zu beweisen, kann man Beweise von φ und von ψ führen.
- um eine Implikation $\varphi \rightarrow \psi$ zu beweisen, kann man φ annehmen und ψ unter dieser (zusätzlichen) Annahme beweisen.

- um eine Disjunktion $\varphi \vee \psi$ zu beweisen, kann man φ oder ψ beweisen.
- um eine Aussage φ zu beweisen, kann man einen Widerspruchsbeweis führen, bei dem $\neg\varphi$ angenommen wird und daraus ein Widerspruch hergeleitet wird.
- um eine Aussage χ zu beweisen kann man mit einer Fallunterscheidung vorgehen: man zeigt Implikation $\varphi_1 \rightarrow \chi, \dots, \varphi_m \rightarrow \chi$ und eine Disjunktion $\varphi_1 \vee \dots \vee \varphi_m$.
- eine einfache Form der Fallunterscheidung ist die Situation $\varphi_1 = \varphi$ und $\varphi_2 = \neg\varphi$: es genügt, die Implikationen $\varphi \rightarrow \chi$ und $\neg\varphi \rightarrow \chi$ zu zeigen.

Regeln dieser Art liefern einen ersten Anhaltspunkt für das Vorgehen (im aussagenlogischen Fall). Grundsätzlich aber ist Finden von Beweisen von hoher Komplexität (Rechenzeit), es gibt beweisbare Aussagen, deren Beweis nicht durch einfache Regeln gefunden werden kann. Beispielsweise gibt es *unendlich* viele Möglichkeiten, wie bei einem Beweis durch Fallunterscheidung die Fallunterscheidung $\varphi/\neg\varphi$ gewählt werden kann.

Kapitel 6

Quantorenlogische formale Beweise

6.1 All-Aussagen

Wir beginnen mit allgemeinen Überlegungen zum Beweis von Quantoren-Aussagen. Eine All-Aussage $\forall x \varphi$ kann für unendliche Strukturen nicht dadurch bewiesen werden, dass man φ für unendlich viele Elemente einzeln überprüft. Das Vorgehen besteht vielmehr darin, ein „allgemeines“ Element der Struktur zu betrachten und φ hierfür nachzuweisen.

Möchte man z.B. für Vektorräume zeigen, dass

$$\forall x \quad -x = (-1) \cdot x$$

ist, so betrachtet man einen „beliebigen“ Vektor y und weist für diesen nach, dass $-y = (-1) \cdot y$. Dass y beliebig ist, bedeutet, dass die Variable y zu Anfang dieses Argumentes nicht frei in den gerade gültigen Voraussetzungen vorkommt.

...

[Sei $y \in V$.

...

$$-y = (-1) \cdot y.$$

[Also] $y \in V \rightarrow -y = (-1) \cdot y$

$$\forall x (x \in V \rightarrow -x = (-1) \cdot x)$$

$$\forall x \in V \quad -x = (-1) \cdot x.$$

...

Dies ist eine Beweismethode, die wir folgendermaßen in die formalen Beweise aufnehmen: Beim Schließen eines Rahmens wird eine All-Aussage zu den lokalen Voraussetzungen hinzugefügt.

Auf diese Art können in einen Beweis \forall -Aussagen eingeführt werden. Für die *Elimination* des Allquantors gibt es die folgende Regel:

\forall -Regeln:

$$\forall\text{-Einführung: } \forall I \quad \frac{\varphi(y)}{\forall x \varphi(x)}, \quad \text{wenn } y \text{ nicht frei in den lokalen Voraussetzungen vorkommt}$$

$$\forall\text{-Elimination: } \forall E \quad \frac{\forall x \varphi(x)}{\varphi(t)}$$

Das wird in *Naproche* wie im folgenden Beispiel umgesetzt:

Satz. $\forall x \forall y R(x, y) \rightarrow \forall y \forall x R(x, y)$.

Beweis. Sei $\forall x \forall y R(x, y)$.

Sei \top . Sei \top .

$\forall y R(x, y)$.

$R(x, y)$.
 Also $\forall x R(x, y)$.
 Also $\forall y \forall x R(x, y)$.
 Also $\forall x \forall y R(x, y) \rightarrow \forall y \forall x R(x, y)$.
 Qed.

Durch „Sei \top “ wird ein Rahmen geöffnet, in dem etwas über eine gegenwärtig nicht freie Variable gezeigt wird. Nach Schließen des Rahmens hat man eine allgemeine Aussage über diese Variable.

Es gibt noch viele andere Möglichkeiten, All-Aussagen zu beweisen. Z.B. beweist man mit dem Prinzip der *vollständigen Induktion* All-Aussagen über natürliche Zahlen n , indem man den Induktionsanfang für $n = 0$ beweist und zeigt, dass die Eigenschaft an der Stelle $n + 1$ gilt, wenn sie an der Stelle n gilt:

$$R(0) \wedge \forall n (R(n) \rightarrow R(n + 1)) \rightarrow \forall n R(n).$$

6.2 Existenzaussagen

Eine Existenzaussage $\exists x \varphi(x)$ kann man beweisen, indem man $\varphi(t)$ für einen konkreten Term t nachweist. Hierdurch wird ein Existenzquantor in das Argument eingeführt. Dual hierzu gibt es eine Regel zur Elimination von Existenzquantoren.

\exists -Regeln:

$$\begin{array}{l}
 \exists\text{-Einführung} \quad \exists I \quad \frac{\varphi(t)}{\exists x \varphi(x)} \\
 \exists\text{-Elimination:} \quad \exists E \quad \frac{\exists x \varphi(x) \quad (\varphi(y) \rightarrow \psi)}{\psi}
 \end{array}$$

Bei der \exists -Elimination ist als Nebenbedingung zu fordern, dass die Variable y „frei“ gewählt ist: y ist nicht frei in den Voraussetzungen, die zum Zeitpunkt der Einführung von $\varphi(y)$ aktiv sind; außerdem kommt y nicht frei in ψ vor. Dies wird deutlicher in dem formalen Beweis

i	z_i	Kommentar
0:
	⋮	⋮
i :	$\exists x \varphi(x)$	
	⋮	⋮
j :	[Sei $\varphi(y)$ „Wähle y mit der Eigenschaft φ “	
	⋮	⋮
k :	ψ	
$k + 1$:	Also] ψ	\exists -Elimination mit i, j und k
	⋮	⋮

Naprobe-Beispiel zur Umbenennung von Variablen:

Satz. $\exists x R(x) \rightarrow \exists y R(y)$.
 Beweis. Sei $\exists x R(x)$.
 Sei $R(x)$.
 $\exists y R(y)$.
 Also $\exists y R(y)$.
 Also $\exists x R(x) \rightarrow \exists y R(y)$.
 Qed.

Noch ein Beispiel:

Satz. $\exists x \forall y R(x, y) \rightarrow \forall y \exists x R(x, y)$.

Beweis. Sei $\exists x \forall y R(x, y)$.

Sei $\forall y R(x, y)$.

Sei \top .

$R(x, y)$.

$\exists x R(x, y)$.

Also $\forall y \exists x R(x, y)$.

Also $\forall y \exists x R(x, y)$.

Also $\exists x \forall y R(x, y) \rightarrow \forall y \exists x R(x, y)$.

Qed.

6.3 Mathematische Texte in Naproche

Ein **Naproche-Text** ist eine endliche Folge von Zeilen, so dass:

- jede Zeile ist von der Gestalt „ φ .“ oder „Sei φ .“ oder „Also φ .“;
- „Sei“ öffnet ein lokales Argument;
- „Also“ schließt ein lokales Argument und zieht eine logische Konsequenz aus dem lokalen Argument;
- zu jedem „Also“ gehört genau ein vorangehendes „Sei“;
- zu jeder Zeile gehören lokale Voraussetzungen, aus denen die Zeile folgen soll; „ φ .“ oder „Sei φ .“ vergrößert die lokalen Voraussetzungen um φ , „Also ψ .“ entfernt die Aussagen des gerade abgeschlossenen Rahmens aus den lokalen Voraussetzungen und fügt ψ hinzu.

Beispiel:

Text	Lokale Voraussetzungen
Sei $\exists x \forall y R(x, y)$.	$\{\}$
Sei $\forall y R(x, y)$.	$\{\exists x \forall y R(x, y)\}$
Sei \top .	$\{\exists x \forall y R(x, y), \forall y R(x, y)\}$
$R(x, y)$.	$\{\exists x \forall y R(x, y), \forall y R(x, y), \top\}$
$\exists x R(x, y)$.	$\{\exists x \forall y R(x, y), \forall y R(x, y), \top, R(x, y)\}$
Also $\forall y \exists x R(x, y)$.	$\{\exists x \forall y R(x, y), \forall y R(x, y)\}$
Also $\forall y \exists x R(x, y)$.	$\{\exists x \forall y R(x, y)\}$
Also $\exists x \forall y R(x, y) \rightarrow \forall y \exists x R(x, y)$.	$\{\}$

Ein **Naproche-Text** ist ein korrekter Beweis, wenn

- in jeder Beweiszeile der Gestalt „ φ .“ φ mit Hilfe der Ableitungsregeln aus den lokalen Voraussetzungen folgt;
- jede Beweiszeile der Gestalt „Also φ .“ von der Gestalt $\varphi = \psi \rightarrow \chi$ ist, wobei ψ und χ erste und letzte Zeile des gerade geschlossenen Rahmens sind.
- die letzte Zeile ist *bewiesen*, wenn die lokalen Voraussetzungen der letzten Zeile $=\{\}$ sind.

6.4 Der Gödelsche Vollständigkeitsatz

Wir stellen die eingeführten Regeln formalen Beweisens noch einmal zusammen. Wir stellen die eingeführten Regeln formalen Beweisens noch einmal zusammen.

Definition 6.1. Die Regeln des natürlichen Schließens sind:

\wedge -Regeln:

$$\begin{aligned} \wedge\text{-Einführung: } \wedge I & \frac{\varphi \quad \psi}{(\varphi \wedge \psi)} \\ \wedge\text{-Elimination: } \wedge E & \frac{(\varphi \wedge \psi)}{\varphi} \\ \wedge\text{-Elimination: } \wedge E & \frac{(\varphi \wedge \psi)}{\psi} \end{aligned}$$

\rightarrow -Regeln:

$$\rightarrow\text{-Elimination: } \rightarrow E \frac{\varphi \quad (\varphi \rightarrow \psi)}{\psi}$$

Die \rightarrow -Einführung erfolgt über das Schließen von Rahmen.

\vee -Regeln:

$$\begin{aligned} \vee\text{-Einführung: } \vee I & \frac{\varphi}{(\varphi \vee \psi)} \\ \vee\text{-Einführung: } \vee I & \frac{\psi}{(\varphi \vee \psi)} \\ \text{Fallunterscheidung } (\vee\text{-Elimination): } \vee E & \frac{(\varphi \vee \psi) \quad (\varphi \rightarrow \chi) \quad (\psi \rightarrow \chi)}{\chi} \end{aligned}$$

Konstanten-Regeln:

$$\begin{aligned} \top\text{-Einführung: } \top I & \frac{}{\top} \\ \perp\text{-Einführung: } \perp I & \frac{\varphi \quad \neg\varphi}{\perp} \\ \perp\text{-Elimination: } \perp E & \frac{\perp}{\varphi} \end{aligned}$$

\neg -Regel:

$$\text{Widerspruchsregel: } \text{Wid} \frac{(\neg\varphi \rightarrow \perp)}{\varphi}$$

\forall -Regeln:

$$\begin{aligned} \exists\text{-Einführung: } \exists I & \frac{\varphi(y)}{\forall x \varphi(x)} \quad y \text{ nicht frei in lokalen} \\ & \quad \text{Voraussetzungen} \\ \forall\text{-Elimination: } \forall E & \frac{\forall x \varphi(x)}{\varphi(t)} \end{aligned}$$

\exists -Regeln:

$$\begin{aligned} \exists\text{-Einführung } \exists I & \frac{\varphi(t)}{\exists x \varphi(x)} \\ \exists\text{-Elimination: } \exists E & \frac{\exists x \varphi(x) \quad (\varphi(y) \rightarrow \psi)}{\psi} \quad y \text{ nicht frei in lokalen} \\ & \quad \text{Voraussetzungen und } \psi \end{aligned}$$

Definition 6.2. Eine Aussage φ ist **formal beweisbar**, wenn es eine formalen Beweis im Sinne der Definition 6.1 mit den Regeln des natürlichen Schließens gibt.

Wir haben exemplarisch gesehen, dass sich die üblichen mathematischen Argumentationen auf diese Regeln zurückführen lassen. Erstaunlicherweise kann man das auch mathematisch beweisen:

Satz 6.3. (Gödelscher Vollständigkeitssatz). Eine Aussage φ ist genau dann allgemeingültig, wenn sie formal beweisbar ist.

Bemerkung 6.4. Der GÖDELSche Vollständigkeitsatz ist der Hauptsatz der mathematischen Logik. Er verbindet auf bestmögliche Weise Semantik und Syntax formaler Sprachen. Dass jede formal beweisbare Aussage allgemeingültig ist, entspricht der *Korrektheit* der angegebenen formalen Beweismethode. Selbstverständlich nimmt man nur solche Schlussregeln auf, die korrekt sind.

Dass umgekehrt jede allgemeingültige Aussage formal bewiesen werden kann, ist schwieriger zu zeigen und ist das eigentliche GÖDELSche Theorem.

Der Gödelsche Vollständigkeitsatz hat über die mathematische Logik hinaus viele Folgerungen, die auf der Verbindung *natürliche Sprache - formale Sprache - formale Semantik - allgemeine Semantik* beruhen. Wir erwähnen einige Bereiche.

Bemerkung 6.5. Der Vollständigkeitsatz liefert ein *absolutes Korrektheitskriterium* für Beweise. Ein mathematischer Beweis ist genau dann korrekt, wenn er (im Prinzip) in einen formalen Beweis umgeschrieben werden kann. Obwohl man für gewöhnlich informell argumentiert, kann man in Zweifelsfällen Argumente soweit formalisieren und in kleinste Zwischenschritte unterteilen, dass sie rein formal und auch von einem Computer überprüft werden können.

Bemerkung 6.6. Der Erfolg der formalen Methode in der Mathematik regt auch andere Bereiche an, ihre Aussagen und Erkenntnismethoden nach Möglichkeit zu formalisieren. Dies geht einher mit der Erfassung der Welt als Daten, die mit Algorithmen verarbeitet werden.

Bemerkung 6.7. *Automatische Beweisen.* Im Prinzip können alle allgemeingültigen Aussagen φ *automatisch* bewiesen werden: zähle alle Texte auf und überprüfe, ob sie ein formaler Beweis von φ sind. Nach dem GÖDELSchen Vollständigkeitsatz hat φ einen formalen Beweis, der durch dieses Vorgehen schließlich gefunden wird. Allerdings ist dieses Vorgehen aus *Komplexitätsgründen* im Allgemeinen praktisch nicht realisierbar. Für eingeschränkte Bereiche gibt es inzwischen automatische Beweiser.

Bemerkung 6.8. *Logisches Programmieren.* Ein Spezialfall des automatischen Beweisens ist das Logische Programmieren (**Prolog**). Programme bestehen aus Quantorenlogischen Aussagen (beschränkter Komplexität), die Ausführung des Programms besteht in der systematischen Anwendung von Schlussregeln auf das Programm.

Bemerkung 6.9. *Künstliche Intelligenz.* Ein erster Ansatz zur Realisierung von künstlicher Intelligenz bestand in der Formulierung von Umwelttatsachen und Fragen in der formalen Logik. Mit den Methoden des automatischen Beweisens wurde dann nach einem Beweis der Frage oder ihrer Negation gesucht. Diese Ansätze haben sich wegen der erwähnten Komplexitätsprobleme als unrealistisch herausgestellt.

Bemerkung 6.10. *Hoare Logik.* Angeregt von dem Erfolg der Quantorenlogik wurden angepasste Logiken für viele Bereiche entwickelt. Von besonderer Wichtigkeit für die Informatik sind Logiken, mit denen sich die Semantik von Algorithmen beschreiben lässt. Im Gegensatz zu den Strukturen der Quantorenlogik sind Algorithmen dynamisch, jede Anweisung hat einen Zustand vor der Ausführung und einen (veränderten) Zustand nach der Ausführung. Die HOARE-Logik beschreibt solche Übergänge, und sie besitzt Schlussregeln, mit denen man die Korrektheit von Programmen beweisen kann.

Bemerkung 6.11. *Gödelscher Unvollständigkeitsatz.* Der Vollständigkeitsatz darf nicht mit dem Unvollständigkeitsatz verwechselt werden, der mehr öffentliche Aufmerksamkeit findet.

Die mathematische Logik hat trotz der (prinzipiellen) Universalität ihrer Methode auch ihre Grenzen studiert. In genügend starken Theorien lässt sich das bekannte *Lügner-Paradoxon* („alle Kreter sind Lügner“) des EPIMENIDES nachbilden, das der umgangssprachlichen Aussage „dieser Satz ist falsch“ entspricht. Man kann diesem Lügner-Satz keinen Wahrheitswert geben, sein Wahrheitswert ist nicht entscheidbar. Ähnlich kann man einen zahlentheoretischen Satz angeben, der mit den PEANOSchen Axiomen der Zahlentheorie nicht entschieden werden kann. Dies ist ist der Inhalt des (ersten) GÖDELSchen Unvollständigkeitsatzes.

Der Unvollständigkeitsatz steht in enger Beziehung zum *Halteproblem* der Rekursionstheorie und theoretischen Informatik.

Kapitel 7

Logisches Programmieren und formale Beweise

7.1 Prolog

Aussagenlogische Ausdrücke sind Teil jeder Programmiersprache, in der das Vorliegen von logisch verknüpften Bedingungen über den weiteren Verlauf eines Prozesses entscheidet.

Auch Quantorenlogik wird im Rahmen des *logischen Programmierens* als Programmiersprache verwendet. Wir betrachten im folgenden die Sprache **Prolog**. Diese Sprache eignet sich besonders gut zur Sprachverarbeitung und wird vor allem in der Linguistik eingesetzt. Wir betrachten Beispiele aus dem Umfeld des **Prolog**-Programms **Naproche**.

Das folgende Programm definiert Terme, die aus den Variablen x, y und den einstelligen Funktionssymbolen f, g gebildet werden können:

```
variable(x).
variable(y).
funktionssymbol(f).
funktionssymbol(g).
term(T) :- variable(T).
term([F,T]) :- funktionssymbol(F), term(T).
```

Wir wollen dieses Programm aus Sicht der Quantorenlogik diskutieren:

- x, y, f, g sind Konstanten (Kleinschreibung), über die bestimmte atomare Aussagen gemacht werden; es seien c_x, c_y, c_f, c_g entsprechende Konstantensymbole der Quantorenlogik.
- **variable**, **funktionssymbol**, **term** sind Relationssymbole (Kleinschreibung).
- Die ersten vier Programmzeilen sind atomare Aussagen, wie wir sie bisher durch $R(c)$ bezeichnet haben.
- Die fünfte Zeile ist eine implikative Allaussage, die äquivalent ist zu

$$\forall t (\text{Variable}(t) \rightarrow \text{Term}(t))$$

Variable wie F, T werden durch Großschreibung gekennzeichnet. Die Implikation \leftarrow wird durch $:-$ notiert. Quantoren werden nicht explizit angegeben, alle freien Variablen werden als universell quantifiziert verstanden.

- Die sechste Zeile ist ebenfalls eine Implikation, mit zwei Voraussetzungen, die durch Konjunktion ($,$) verbunden sind:

$$\forall f \forall t (\text{Funktionssymbol}(f) \wedge \text{Term}(t) \rightarrow \text{Term}([f, t]))$$

- die Zeilen eines **Prolog**-Programms werden als **Klauseln** bezeichnet;
- In **Prolog** sind Listenoperationen fest implementiert. Listen werden explizit als $[f, x]$ oder $[F, T]$ angegeben. Es stehen noch weitere Listenoperationen zur Verfügung.

Damit entspricht das Programm folgender Menge Φ von Aussagen der Quantorenlogik:

$\text{Variable}(c_x)$

```

Variable( $c_y$ )
Funktionssymbol( $c_f$ )
Funktionssymbol( $c_g$ )
 $\forall t(\text{Variable}(t) \rightarrow \text{Term}(t))$ 
 $\forall f \forall t(\text{Funktionssymbol}(f) \wedge \text{Term}(t) \rightarrow \text{Term}([f, t]))$ 

```

Eine geschachtelte Liste ist ein „Term“, wenn dies aus der Aussagenmenge Φ folgt.

Nach dem Laden des Prolog-Programms kann man „Fragen“ stellen, etwa ob $[f, [g, x]]$ ein Term ist. Das Prolog-System sucht dann systematisch nach einem Beweis für die Term-Eigenschaft und gibt entsprechend Yes oder No aus. Das Suchen geschieht durch das Anpassen der Eingabe an Zeilen des Programms. Hierdurch werden neue Fragen generiert:

```

Welcome to SWI-Prolog (Multi-threaded, Version 5.4.4)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

```

```
For help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- % /tmp/procomp11301sER.pl compiled 0.01 sec, 1,052 bytes
```

```
Yes
```

```
?- term([f, [g, x]]).
```

```
Yes
```

```
?- trace.
```

```
Yes
```

```

[trace] ?- term([f, [g, x]]).
Call: (7) term([f, [g, x]]) ?
Call: (8) variable([f, [g, x]]) ?
Fail: (8) variable([f, [g, x]]) ?
Redo: (7) term([f, [g, x]]) ?
Call: (8) funktionssymbol(f) ?
Exit: (8) funktionssymbol(f) ?
Call: (8) term([g, x]) ?
Call: (9) variable([g, x]) ?
Fail: (9) variable([g, x]) ?
Redo: (8) term([g, x]) ?
Call: (9) funktionssymbol(g) ?
Exit: (9) funktionssymbol(g) ?
Call: (9) term(x) ?
Call: (10) variable(x) ?
Exit: (10) variable(x) ?
Exit: (9) term(x) ?
Exit: (8) term([g, x]) ?
Exit: (7) term([f, [g, x]]) ?

```

```
Yes
```

```
[debug] ?-
```

In der `trace` sieht man, wie die Anfangsfrage `term([f, [g, x]])?` mit den Klauseln des Programms verglichen wird, und dass eventuell neue Fragen wie `term([g, x])?` generiert werden, die dann ebenso behandelt werden. Eine Frage wird (rekursiv) positiv beantwortet, wenn es eine Klausel gibt, so dass alle dadurch entstehenden Unterfragen positiv beantwortet werden können. Das Durchsuchen geschieht entlang der Reihenfolge der Klauseln im Programm.

7.2 Ein Beweisprüfer in Prolog

Der Beweisprüfer *Naproche* ist in Prolog geschrieben. Wir beschreiben hier einen sehr kleinen Beweisprüfer für eine aussagenlogische Sprache, der das Grundprinzip des Prüfalgorithmus andeutet. Außerdem wird die „sei“-„also“-Konstruktion noch einmal illustriert. Das Prädikat `beweis(Text,Hyp)` ist wahr, wenn `Text` unter Voraussetzung der Hypothesen `Hyp` ein korrekter Beweistext ist. In der Klausel

```
beweis([Phi|Rest],Hyp) :-
    member(Psi,Hyp), member([Psi,impliziert,Phi],Hyp),
    beweis(Rest,[Phi|Hyp]).
```

wird eine mögliche Bedingung für die Korrektheit eines Textes `[Phi|Rest]` definiert, wobei `Phi` das erste Element und `Rest` die Liste der übrigen Elemente des Textes ist. Die Klausel entspricht der *modus ponens* Regel. Mit Hilfe des Prolog-Prädikats `member` wird abgefragt, ob Aussagen `Psi` und `[Psi,then,Phi]` unter den Hypothesen vorkommen. In diesem Fall ist `Phi` gerechtfertigt und kann für den Rest des Beweises zu den Hypothesen hinzugefügt werden: `[Phi|Hyp]`. `[Phi|Rest]` ist also korrekt, wenn `Phi` durch *modus ponens* gerechtfertigt ist, und wenn der Rest `Rest` unter den erweiterten Hypothesen `[Phi|Hyp]` korrekt ist.

Man beachte, dass bei dieser Schlussregel nach der *Existenz* eines geeigneten `Phi` gefragt wird. In quantorenlogischer Schreibweise:

$$\forall \varphi \forall R \forall H (\exists \psi (\psi \in H \wedge (\psi \rightarrow \varphi) \in H \wedge \text{Beweis}(R, H \hat{\ } \varphi)) \rightarrow \text{Beweis}(\varphi \hat{\ } R, H)).$$

```
beweis([],Hyp).
beweis([[sei,Phi]|Rest],Hyp) :- beweis(Rest,[sei,Phi|Hyp]).
beweis([Phi|Rest],Hyp) :- member(Phi,Hyp), beweis(Rest,[Phi|Hyp]).
beweis([Phi|Rest],Hyp) :-
    member(Psi,Hyp), member([Psi,impliziert,Phi],Hyp),
    beweis(Rest,[Phi|Hyp]).
beweis([Phi|Rest],Hyp) :-
    member([Phi,impliziert,falsum],impliziert,falsum,Hyp),
    beweis(Rest,[Phi|Hyp]).
beweis([[also,[Psi,impliziert,Phi]]|Rest],[Phi|Hyp]) :-
    rpop(Hyp,Psi,H1),beweis(Rest,[Psi,impliziert,Phi]|H1)).

rpop([sei,Psi|H],[Psi,H1]).
rpop([Psi|H1],X,Y) :- rpop(H1,X,Y).
```

Prolog-Programme können durch Text nach `%`-Zeichen kommentiert werden. Wir kommentieren das Programm für `beweis`:

```
beweis([],Hyp).
%% Der leere Text ist immer ein Beweis.
beweis([[sei,Phi]|Rest],Hyp) :- beweis(Rest,[sei,Phi|Hyp]).
%% Eine sei-Annahme kann immer gemacht werden; die mit 'sei'
%% eingeführte Annahme wird den Hypothesen hinzugefügt.
beweis([Phi|Rest],Hyp) :- member(Phi,Hyp), beweis(Rest,[Phi|Hyp]).
%% Ein in den Hypothesen vorkommende Aussage kann wiederholt werden.
beweis([Phi|Rest],Hyp) :-
    member(Psi,Hyp), member([Psi,then,Phi],Hyp),
    beweis(Rest,[Phi|Hyp]).
%% Siehe oben.
beweis([Phi|Rest],Hyp) :-
    member([Phi,impliziert,falsum],impliziert,falsum,Hyp),
    beweis(Rest,[Phi|Hyp]).
%% Diese Regel entspricht der Elimination einer doppelten
```

```

%% Negation: von ((Phi -> falsum) -> falsum)
%% kann auf Phi geschlossen werden
beweis([[also],[Psi,impliziert,Phi]]|Rest],[Phi|Hyp]) :-
    rpop(Hyp,Psi,H1), beweis(Rest,[[Psi,impliziert,Phi]|H1]).
%% ‘‘also’’ signalisiert das Schliessen eines lokalen Arguments.
%% Danach ist die durch das lokale Argument gezeigte Implikation
%% bewiesen und kann als neue lokale Voraussetzung benutzt werden.
%% Das Praedikat rpop ‘‘popt’’ den Stack der Hypothesen bis
%% zum vorangehenden ‘‘sei’’ und gibt dann die dazugehoerige
%% Annahme und die damaligen lokalen Voraussetzungen aus.

```

```

rpop([sei,Psi|H1],Psi,H1).
rpop([Psi|H1],X,Y) :- rpop(H1,X,Y).

```

Als Beispiel demonstrieren wir einen Beweis der Tautologie $\varphi \rightarrow \varphi$:

```

Welcome to SWI-Prolog (Multi-threaded, Version 5.4.4)
Copyright (c) 1990-2003 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

```

For help, use ?- help(Topic). or ?- apropos(Word).

Yes

```

?- beweis([
|   [sei,phi],
|   phi,
|   [also,phi,impliziert,phi]],
|   []).

```

No

```

?- beweis([
|   [sei,phi],
|   phi,
|   [also,[phi,impliziert,phi]]],
|   []).

```

Yes

```

?- trace.

```

Yes

```

[trace] ?- beweis([
|   [sei,phi],
|   phi,
|   [also,[phi,impliziert,phi]]],
|   []).
Call: (7) beweis([[sei, phi], phi, [also, [phi, impliziert, phi]]], []) ?
Call: (8) beweis([phi, [also, [phi, impliziert, phi]]], [sei, phi]) ?
Call: (9) lists:member(phi, [sei, phi]) ?
Exit: (9) lists:member(phi, [sei, phi]) ?
Call: (9) beweis([[also, [phi, impliziert, phi]]], [phi, sei, phi]) ?
Call: (10) lists:member([also, [phi, impliziert, phi]], [phi, sei, phi]) ?
Fail: (10) lists:member([also, [phi, impliziert, phi]], [phi, sei, phi]) ?
Redo: (9) beweis([[also, [phi, impliziert, phi]]], [phi, sei, phi]) ?
Call: (10) lists:member(L285, [phi, sei, phi]) ?
Exit: (10) lists:member(phi, [phi, sei, phi]) ?
Call: (10) lists:member([phi, impliziert, [also, [phi, impliziert[...]]]], [phi, sei, phi]) ?
Fail: (10) lists:member([phi, impliziert, [also, [phi, impliziert[...]]]], [phi, sei, phi]) ?
Redo: (10) lists:member(L285, [phi, sei, phi]) ?
Exit: (10) lists:member(sey, [phi, sei, phi]) ?
Call: (10) lists:member([sei, impliziert, [also, [phi, impliziert[...]]]], [phi, sei, phi]) ?
Fail: (10) lists:member([sei, impliziert, [also, [phi, impliziert[...]]]], [phi, sei, phi]) ?
Redo: (10) lists:member(L285, [phi, sei, phi]) ?
Exit: (10) lists:member(phi, [phi, sei, phi]) ?
Call: (10) lists:member([phi, impliziert, [also, [phi, impliziert[...]]]], [phi, sei, phi]) ?
Fail: (10) lists:member([phi, impliziert, [also, [phi, impliziert[...]]]], [phi, sei, phi]) ?

```

```

Redo: (10) lists:member(_L285, [phi, sei, phi]) ?
Fail: (10) lists:member(_L285, [phi, sei, phi]) ?
Redo: (9) beweis([[also, [phi, impliziert, phi]]], [phi, sei, phi]) ?
Call: (10) lists:member([[also, [phi, impliziert, phi]], impliziert, falsum], impliziert,
falsum), [phi, sei, phi]) ?
Fail: (10) lists:member([[also, [phi, impliziert, phi]], impliziert, falsum], impliziert,
falsum), [phi, sei, phi]) ?
Redo: (9) beweis([[also, [phi, impliziert, phi]]], [phi, sei, phi]) ?
Call: (10) rpop([sei, phi], phi, _L287) ?
Exit: (10) rpop([sei, phi], phi, []) ?
Call: (10) beweis([], [[phi, impliziert, phi]]) ?
Exit: (10) beweis([], [[phi, impliziert, phi]]) ?
Exit: (9) beweis([[also, [phi, impliziert, phi]]], [phi, sei, phi]) ?
Exit: (8) beweis([phi, also, [phi, impliziert, phi]], [sei, phi]) ?
Exit: (7) beweis([[sei, phi], phi, [also, [phi, impliziert, phi]]], []) ?

Yes
[debug] ?-

```

7.3 Der Beweisprüfer Naproche

Ein formaler Beweistext ist ein korrekter Beweis, wenn er bestimmte syntaktische Kriterien erfüllt. Diese Kriterien können mit symbolverarbeitenden Programmen automatisch überprüft werden.

Das Programm `Naproche.pl` überprüft aussagenlogische Beweise im Sinne der in der Vorlesung eingeführten formalen Sprache und Beweisregeln. Als aussagenlogische Variablen stehen alle kleinen griechischen Buchstaben zur Verfügung. Der korrekte Beweis aus der Vorlesung

```

Satz.  $\exists x R(x) \rightarrow \exists y R(y)$ .
Beweis. Sei  $\exists x R(x)$ .
Sei  $R(x)$ .
 $\exists y R(y)$ .
Also  $\exists y R(y)$ .
Also  $\exists x R(x) \rightarrow \exists y R(y)$ .
Qed.

```

kann in genau dieser Form eingegeben werden; das Programm `Naproche` bestätigt die Korrektheit durch die Ausgabe `The proof is accepted`. Inkorrekte Texte führen zur Ausgabe `The proof is NOT accepted`.

Für die Benutzung des Programms wird der mathematische WYSIWYG-Texteditor `TeXMACS` sowie ein Prolog-Interpreter/Compiler benötigt.

Die momentane Version von `Naproche` ist ein Prototyp, für dessen Richtigkeit keinerlei Gewähr übernommen wird. Sie kann über die Homepage der Vorlesung heruntergeladen werden. Sie läuft mit `TeXMACS` Version 1.0.6 (www.texmacs.org) und dem bekannten SWI-Prolog Version 5.4.4 (www.swi-prolog.org) unter SuSe-Linux 9.1.

Kapitel 8

Kombinatorik

Die (endliche) Kombinatorik untersucht die Frage: wieviel Elemente enthält eine endliche Menge. Die Anzahl der Elemente wird durch den Grundbegriff der *Kardinalität* gegeben.

Definition 8.1. Die **Mächtigkeit** oder **Kardinalität** einer endlichen Menge S ist die Anzahl der in S enthaltenen Elemente und wird mit $|S|$ bezeichnet. $|S| = m$ genau dann, wenn sich S aufzählen als

$$S = \{s_1, \dots, s_m\}$$

mit paarweise verschiedenen Elementen s_1, \dots, s_m aufzählen lässt.

Insbesondere ist $|\emptyset| = 0$ und $|\{s\}| = 1$. Wir werden im folgenden verschiedene arithmetische Gesetzmäßigkeiten der $|\cdot|$ -Funktion studieren.

Satz 8.2. Seien A und B endliche Mengen. Dann gilt:

- a) Wenn A und B zueinander disjunkt sind, d.h. $A \cap B = \emptyset$, dann ist $|A \cup B| = |A| + |B|$.
- b) Seien A_1, \dots, A_m endliche Mengen sind, die paarweise disjunkt sind, d.h. für $i < j \leq m$ ist $A_i \cap A_j = \emptyset$. Dann ist

$$|A_1 \cup \dots \cup A_m| = |A_1| + \dots + |A_m|.$$

- c) $|A \cup B| = |A| + |B| - |A \cap B|$ (Einschluss-Ausschluss-Prinzip).

- d) $|A \times B| = |A| \cdot |B|$.

- e) Seien A_1, \dots, A_m endliche Mengen. Dann ist

$$|A_1 \times \dots \times A_m| = |A_1| \cdot \dots \cdot |A_m|.$$

Beweis. a) Seien $A = \{a_1, \dots, a_m\}$ und $B = \{b_1, \dots, b_n\}$ mit jeweils paarweise verschiedenen Elementen a_1, \dots, a_m und b_1, \dots, b_n . Da A und B disjunkt sind, sind die Elemente $a_1, \dots, a_m, b_1, \dots, b_n$ paarweise verschieden.

$$A \cup B = \{a_1, \dots, a_m, b_1, \dots, b_n\}$$

und daher

$$|A \cup B| = m + n = |A| + |B|.$$

- b) Durch vollständige Induktion über $m \geq 1$.

Induktionsanfang: $m = 1$. $|A_1| = |A_1|$ gilt trivial.

Induktionsschritt: Die Behauptung gelte für m . Betrachte paarweise disjunkte endliche Mengen A_1, \dots, A_m, A_{m+1} . Dann sind $A_1 \cup \dots \cup A_m$ und A_{m+1} disjunkt:

$$(A_1 \cup \dots \cup A_m) \cap A_{m+1} = \emptyset.$$

Nach a) und der Induktionsannahme gilt dann

$$\begin{aligned} |A_1 \cup \dots \cup A_{m+1}| &= |(A_1 \cup \dots \cup A_m) \cup A_{m+1}| \\ &= |(A_1 \cup \dots \cup A_m)| + |A_{m+1}| \\ &= (|A_1| + \dots + |A_m|) + |A_{m+1}| \\ &= |A_1| + \dots + |A_{m+1}|. \end{aligned}$$

c) Die Mengen $A \setminus B$, $A \cap B$ und $B \setminus A$ sind paarweise disjunkt und es gelten die Gleichungen:

$$\begin{aligned} A &= (A \setminus B) \cup (A \cap B) \\ B &= (B \setminus A) \cup (A \cap B) \\ A \cup B &= (A \setminus B) \cup (A \cap B) \cup (B \setminus A) \end{aligned}$$

Nach a) und b) folgt daraus:

$$\begin{aligned} |A| &= |A \setminus B| + |A \cap B| \\ |B| &= |B \setminus A| + |A \cap B| \\ |A \cup B| &= |A \setminus B| + |A \cap B| + |B \setminus A| \\ &= (|A| - |A \cap B|) + |A \cap B| + (|B| - |A \cap B|) \\ &= |A| + |B| - |A \cap B|. \end{aligned}$$

d) Seien $A = \{a_1, \dots, a_m\}$ und $B = \{b_1, \dots, b_n\}$ mit jeweils paarweise verschiedenen Elementen a_1, \dots, a_m und b_1, \dots, b_n . Dann ist

$$\begin{aligned} A \times B &= \{(a_i, b_j) | i=1, \dots, m \text{ und } j=1, \dots, n\} \\ &= \bigcup_{i=1, \dots, m} \{(a_i, b_j) | j=1, \dots, n\} \end{aligned}$$

Die Mengen auf der rechten Seite sind paarweise disjunkt und haben alle die Kardinalität n . Nach b) gilt dann:

$$\begin{aligned} |A \times B| &= \left| \bigcup_{i=1, \dots, m} \{(a_i, b_j) | j=1, \dots, n\} \right| \\ &= \sum_{i=1, \dots, m} |\{(a_i, b_j) | j=1, \dots, n\}| \\ &= \sum_{i=1, \dots, m} n \\ &= m \cdot n \end{aligned}$$

e) lässt sich aus d) durch vollständige Induktion ähnlich b) zeigen. □

8.1 Das Dirichletsche Schubfachprinzip

Wenn $n + 1$ Gegenstände in n Schubfächer gelegt werden, so gibt es in mindestens einem der Schubfächer mindestens 2 Gegenstände:

Satz 8.3. Seien A und B endliche Mengen mit $|A| > |B|$ und $f: A \rightarrow B$ eine Funktion. Dann gibt es $a, a' \in A$, $a \neq a'$ mit $f(a) = f(a')$. Das bedeutet, dass die Abbildung $f: A \rightarrow B$ nicht injektiv ist.

Beispiel 8.4. Unter 13 Personen haben mindestens 2 im gleichen Monat Geburtstag.

Das Prinzip lässt sich mit Berechnungen von Kardinalitäten endlicher Mengen kombinieren.

Beispiel 8.5. Wieviele verschiedene Familiennamen müssen in einem Telefonbuch vorkommen, damit es auf jeden Fall zwei Anschlüsse gibt, deren Namen dieselben ersten und zweiten Buchstaben haben? Wir definieren eine Abbildung f von den Anschlüssen in das Paar bestehend aus dem ersten und zweiten Buchstaben des zugehörigen Namens:

$$a_1 a_2 \dots a_m \mapsto (a_1, a_2).$$

Die Bilder der Abbildung liegen in der Menge $\mathcal{A} \times \mathcal{A}$ aller Buchstabenpaare, wobei $\mathcal{A} = \{a, \dots, z\}$ das Standardalphabet mit 26 Buchstaben ist. Nach vorangehenden Sätzen über Kardinalitäten ist $|\mathcal{A} \times \mathcal{A}| = 26 \cdot 26 = 676$. Wenn das Telefonbuch mindestens $676 + 1 = 677$ Einträge enthält, so gibt es zwei Anschlüsse, deren Namen dieselben ersten und zweiten Buchstaben haben.

Beispiel 8.6. Aus den Zahlen $1, 2, \dots, 8$ werden 5 Zahlen ausgewählt. Dann ist die Summe von zweien der ausgewählten Zahlen gleich 9. Sei A die Menge der 5 ausgewählten Zahlen, B die Menge

$$\{\{1, 8\}, \{2, 7\}, \{3, 6\}, \{4, 5\}\}$$

der Zweiermengen, deren Summe 9 ist. Da $|B| = 4$ ist, gibt es zwei verschiedene Elemente in A , die Elemente desselben Elementes von B sind. Die Summe dieser zwei verschiedenen Elemente ist dann 9.

8.2 Zählformeln

Oft müssen Folgen von Elementen aus gegebenen Mengen betrachtet werden. Man muss unterscheiden, ob in diesen Folgen Wiederholungen erlaubt sind und dasselbe Element mehrfach vorkommen darf, und ob die Reihenfolge der Elemente berücksichtigt wird.

Beispiel 8.7. Aus der Menge $A = \{a, b, c\}$ sollen 2 Elemente ausgewählt werden:

- **Stichproben:** Wiederholungen erlaubt, Reihenfolge relevant: Dann gibt es $9 = 3 \cdot 3$ *Stichproben*:

$$(a, a), (a, b), (a, c), (b, a), (b, b), (b, c), (c, a), (c, b), (c, c).$$

- **Auswahlen:** Wiederholungen erlaubt, Reihenfolge irrelevant: Dann gibt es 6 *Auswahlen*:

$$(a, a), (a, b), (a, c), (b, b), (b, c), (c, c).$$

- **Permutationen:** Wiederholungen nicht erlaubt, Reihenfolge relevant: Dann gibt es 6 *Permutationen*:

$$(a, b), (a, c), (b, a), (b, c), (c, a), (c, b).$$

- **Kombinationen:** Wiederholungen nicht erlaubt, Reihenfolge irrelevant: Dann gibt es 3 *Kombinationen*:

$$(a, b), (a, c), (b, c).$$

Definition 8.8. Sei $A = \{a_1, \dots, a_n\}$ eine endliche Menge und $k \in \mathbb{N}$ eine natürliche Zahl.

a) s ist eine **k -Stichprobe** aus A , wenn s eine Folge der Länge k mit Einträgen in A ist: $s \in A^k$.

b) s ist eine **k -Auswahl** aus A , wenn s eine Funktion $s: A \rightarrow \mathbb{N}$ ist mit

$$s(a_1) + \dots + s(a_n) = k$$

c) s ist eine **k -Permutation** aus A , wenn $s = (s_1, \dots, s_k)$ eine injektive Folge der Länge k mit Einträgen in A ist, d.h. $\forall i < j \leq k \ s_i \neq s_j$.

d) s ist eine **k -Kombination** aus A , wenn s eine Teilmenge von A von der Kardinalität k ist.

Derartige Auswahlen kommen in der Wahrscheinlichkeitstheorie vor. Ein Ereignis besteht in einer Auswahl von Elementen aus einer Menge, zur Bestimmung von Wahrscheinlichkeiten ist es erforderlich, die Anzahl der möglichen Auswahlen zu bestimmen.

Satz 8.9. Sei $A = \{a_1, \dots, a_n\}$ eine endliche Menge der Kardinalität $n \geq 1$ und $k \in \mathbb{N}$ eine natürliche Zahl. Dann gilt

a) $|\{s \mid s \text{ ist eine } k\text{-Stichprobe aus } A\}| = n^k$.

b) $|\{s \mid s \text{ ist eine } k\text{-Auswahl aus } A\}| = \frac{(n+k-1)!}{k!(n-1)!}$.

$$c) |\{s \mid s \text{ ist eine } k\text{-Permutation aus } A\}| = \frac{n!}{(n-k)!}.$$

$$d) |\{s \mid s \text{ ist eine } k\text{-Kombination aus } A\}| = \frac{n!}{k!(n-k)!}.$$

Beweis. a) Durch Induktion über $k \in \mathbb{N}$.

Induktionsanfang: $k = 0$. Es gibt genau eine 0-Stichprobe aus A , nämlich die eindeutig bestimmte Folge der Länge 0. Ist die Anzahl der 0-Stichproben $= 1 = n^0$, und die Behauptung gilt für $k = 0$.

Induktionsschritt: Betrachte $k \in \mathbb{N}$, und die Behauptung gelte für k . Dann ist

$$\{s \mid s \text{ ist } k+1\text{-Stichp. aus } A\} = \bigcup_{a \in A} \{s \mid s \text{ ist } k+1\text{-Stichpr. aus } A, s(k) = a\}.$$

Die rechte Seite ist eine disjunkte Vereinigung von Mengen. Die Menge

$$\{s \mid s \text{ ist } k+1\text{-Stichpr. aus } A, s(k) = a\}$$

ist genauso groß wie die Menge

$$\{s \mid s \text{ ist } k\text{-Stichpr. aus } A\}.$$

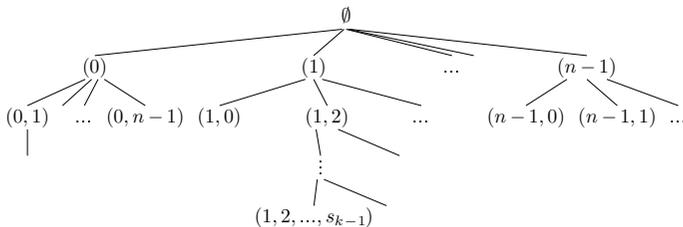
Nach der Induktionsvoraussetzung hat diese Menge die Kardinalität n^k . Zusammen gilt:

$$\begin{aligned} |\{s \mid s \text{ ist } k+1\text{-Stichp. aus } A\}| &= \sum_{a \in A} |\{s \mid s \text{ ist } k\text{-Stichpr. aus } A\}| \\ &= \sum_{a \in A} n^k = n \cdot n^k = n^{k+1}. \end{aligned}$$

Also gilt die Behauptung für $k+1$.

Auch die anderen Behauptungen des Satzes lassen sich durch Induktion zeigen, wir wollen aber anschauliche Argumente benutzen.

c) Wir visualisieren die Menge der k -Permutationen (s_1, \dots, s_k) aus A ; zur Vereinfachung nehmen wir an, dass $A = \{0, 1, \dots, n-1\}$:



Dieser Baum verzweigt bei der *Wurzel* \emptyset n -fach, auf der nächsten Ebene $n-1$ -fach, und allgemein auf der l -ten Ebene $n-l$ -fach. Die k -Permutation stehen auf der k -ten Ebene. Die Anzahl der Punkte auf der k -ten Ebene ergibt sich durch die Verzweigungsmöglichkeiten auf den Ebenen zuvor. Wir multiplizieren die Verzweigungsgrade auf den Ebenen $0, 1, \dots, (k-1)$:

$$\begin{aligned} n \cdot (n-1) \cdots (n-(k-1)) &= \frac{n \cdot (n-1) \cdots (n-(k-1)) \cdot (n-k) \cdot (n-k-1) \cdots 1}{(n-k) \cdot (n-k-1) \cdots 1} \\ &= \frac{n!}{(n-k)!}. \end{aligned}$$

d) Sei X die Anzahl der k -Kombinationen aus A , d.h. die Anzahl der Teilmengen von A mit k Elementen. Diese Zahl steht in Beziehung zur gerade bestimmten Anzahl der k -Permutationen: Eine k -Permutation kann als eine k -Permutation einer k -elementigen Teilmenge von A aufgefasst werden:

$$\{s \mid s \text{ ist } k\text{-Permutation von } A\} = \bigcup_{\substack{B \subseteq A \\ |B|=k}} \{s \mid s \text{ ist } k\text{-Permutation von } B\}.$$

Die rechte Seite ist eine disjunkte Vereinigung, die Anzahl der Summanden ist die gesuchte Anzahl der k -Kombinationen aus A . Es gilt:

$$\begin{aligned} |\{s \mid s \text{ ist } k\text{-Permutation von } A\}| &= \sum_{\substack{B \subseteq A \\ |B|=k}} |\{s \mid s \text{ ist } k\text{-Permutation von } B\}| \\ \frac{n!}{(n-k)!} &= \sum_{\substack{B \subseteq A \\ |B|=k}} \frac{k!}{(k-k)!} \\ &= X \cdot \frac{k!}{(k-k)!} = X \cdot k! \\ X &= \frac{n!}{k! \cdot (n-k)!} \end{aligned}$$

b) Eine k -Auswahl aus A ist eine Funktion $s: A \rightarrow \mathbb{N}$ mit

$$s(a_1) + \dots + s(a_n) = k.$$

Wir können s darstellen als eine geordnete Folge

$$\underbrace{(a_0, \dots, a_0)}_{s(a_0)}, \underbrace{(a_1, \dots, a_1)}_{s(a_1)}, \dots, \underbrace{(a_{n-1}, \dots, a_{n-1})}_{s(a_{n-1})}.$$

Diese Folge kann auch dargestellt werden durch Angabe der „Trenner“ zwischen den Blöcken mit den Elementen von A , die wir als $|$ schreiben können:

$$\underbrace{(a_1, \dots, a_1)}_{s(a_1)}, |, \underbrace{(a_2, \dots, a_2)}_{s(a_2)}, |, \dots, |, \underbrace{(a_n, \dots, a_n)}_{s(a_n)}.$$

Die Folge hat die Länge $k + (n - 1)$ und ist durch die Positionen der Trenner eindeutig bestimmt:

$$(\dots, |, \dots, |, \dots).$$

Diese Positionsfolge durch die Menge der Trennerpositionen angegeben werden. Diese Menge ist eine $n - 1$ -elementige Teilmenge einer Menge mit $k + (n - 1)$ Elementen. Daher hat die Menge der k -Auswahlen aus A genausoviele Elemente wie die Menge der $n - 1$ -Kombinationen aus einer Menge mit $k + (n - 1)$ Elementen. Diese Anzahl beträgt nach d):

$$\frac{(k + (n - 1))!}{(n - 1)! \cdot (k + (n - 1) - (n - 1))!} = \frac{(n + k - 1)!}{k!(n - 1)!}. \quad \square$$

Beispiel 8.10. Das Geburtstagsparadoxon: Betrachte eine Menge von 20 Personen. Die Folge der Geburtstage (Tag und Monat) kann als 20-Stichprobe aus $\{1, \dots, 365\}$ aufgefasst werden. Es gibt 365^{20} solcher Folgen.

Die Menge der 20-Permutationen aus $\{1, \dots, 365\}$ besteht aus allen Konfigurationen, in denen keine zwei Personen den gleichen Geburtstag haben. Es gibt

$$\frac{365!}{345!}$$

solcher Konfigurationen. Das Verhältnis von Konfigurationen mit paarweise verschiedenen Geburtstagen zu den Gesamtmöglichkeiten ist

$$\frac{365!}{345!} / 365^{20} = \frac{365!}{345! \cdot 365^{20}}.$$

Wir bestimmen diese Zahl numerisch:

maxima] Float((365!)/((345!)*(365^20)))

(D18) 0.58856159741595

(C19)

Also gibt es bei mehr als 40% der Konfigurationen gemeinsame Geburtstage. Man kann diese Zahl als *Wahrscheinlichkeit* für das Auftreten gemeinsamer Geburtstage auffassen. Bei 50 Personen kann man fast sicher sein, gemeinsame Geburtstage vorzufinden:

(C19) `Float((365!)/((315!)*(365^50)))`

(D24) 0.02962642037145

(C25)

Kapitel 9

Graphen

Definition 9.1. Ein **schlichter Graph** ist eine Struktur $G = (E, K)$, wobei K eine zweistellige Relation auf E ist, die irreflexiv und symmetrisch ist:

$$\forall x \in E \neg xKx \text{ und } \forall x, y \in E (xKy \rightarrow yKx).$$

Die Elemente der Trägermenge E heißen **Ecken** von G . Die Mengen $\{a, b\}$ mit aKb heißen **Kanten** von G ; wir schreiben auch ab statt aKb . Gilt aKb , so sind a und b in G **benachbart** oder **adjazent**, und die Kante ab ist **inzident** zu a (und zu b). Der **Grad** einer Ecke a ist

$$\delta(a) = |\{b \in E \mid aKb\}|.$$

Definition 9.2. Sei $G = (E, K)$ ein schlichter Graph mit endlicher Trägermenge $E = \{e_1, \dots, e_m\}$. Die **Adjazenzmatrix** von G ist eine Matrix $A = (a_{ij})_{1 \leq i, j \leq m}$ mit Einträgen $a_{ij} \in \{\mathbf{W}, \mathbf{F}\}$ aus der Menge der Wahrheitswerte und

$$a_{ij} = \mathbf{W} \text{ gdw. } e_i K e_j.$$

Die Adjazenzmatrix von G ist symmetrisch und alle Diagonaleinträge sind $= \mathbf{F}$.

Definition 9.3. Sei $G = (E, K)$ ein schlichter Graph.

- a) Eine Folge (a_1, \dots, a_l) von Ecken ist ein **Pfad der Länge $l - 1$** in G , wenn

$$\forall i < l \ a_i K a_{i+1}.$$

- b) Eine Pfad (a_1, \dots, a_l) heißt **kantendisjunkt**, wenn die Kanten in dem Pfad paarweise verschieden sind, d.h.

$$\forall 1 \leq i < j < l \ a_i a_{i+1} \neq a_j a_{j+1}.$$

- c) Eine Pfad (a_1, \dots, a_l) heißt **knotendisjunkt**, wenn die Knoten in dem Pfad paarweise verschieden sind, d.h.

$$\forall 1 \leq i < j < l \ a_i \neq a_j.$$

- d) Ein knotendisjunkter Pfad ist ein **Weg**.

- e) Ein Pfad (a_1, \dots, a_l) ist ein **Zyklus** oder **Kreis** in G , wenn $l \geq 3$, a_1, \dots, a_{l-1} paarweise verschieden sind und $a_l = a_1$.

- f) Der Graph G ist **azyklisch**, wenn er keinen Zyklus enthält.

- g) Der Graph G ist **zusammenhängend**, wenn es für alle $a, b \in E$ einen Weg $(a, a_2, \dots, a_{l-1}, b)$ in G gibt; wir sagen, dass $(a, a_2, \dots, a_{l-1}, b)$ ein **Weg von a nach b** ist.

Lemma 9.4. Sei $G = (E, K)$ ein schlichter Graph. Für $a, b \in E$ sind die folgenden Aussagen äquivalent:

- a) Es gibt einen Pfad von a nach b .
 b) Es gibt einen Weg von a nach b .

Beweis. $a) \rightarrow b)$ Sei $(a, a_2, \dots, a_{l-1}, b)$ ein Pfad von a nach b . Existieren $i < j$ mit $a_i = a_j$, so ist mit $(a, a_2, \dots, a_{i-1}, a_i, \dots, a_j, a_{j+1}, \dots, a_{l-1}, b)$ auch $(a, a_2, \dots, a_{i-1}, a_j, a_{j+1}, \dots, a_{l-1}, b)$ ein Pfad von a nach b . Induktiv folgt, dass man alle mehrfach vorkommenden Knoten aus dem Pfad entfernen kann.

$b) \rightarrow a)$ ist klar, da jeder Weg ein Pfad ist. \square

Betrachte eine Graphen $G = (E, K)$. Definiere eine Relation \sim auf E , $a \sim b$ gdw. es einen Weg von a nach b gibt. Die Relation \sim ist eine *Äquivalenzrelation* auf E :

- a) Betrachte $a \in E$. (a) ist ein (trivialer) Weg von a nach a . Also ist $a \sim a$. Damit ist \sim reflexiv.
- b) Betrachte $a, b \in E$ mit $a \sim b$. Wähle einen Weg $(a, a_2, \dots, a_{l-1}, b)$ von a nach b . Dann ist $(b, a_{l-1}, \dots, a_2, a)$ ein Weg von b nach a . Also ist $b \sim a$. Damit ist \sim symmetrisch.
- c) Betrachte $a, b, c \in K$ mit $a \sim b$ und $b \sim c$. Wähle Wege $(a, a_2, \dots, a_{l-1}, b)$ von a nach b und $(b, b_2, \dots, b_{k-1}, c)$ von b nach c . Verkette diese Wege zum Weg

$$(a, a_2, \dots, a_{l-1}, b, b_2, \dots, b_{k-1}, c)$$

von a nach c . Also ist $a \sim c$. Damit ist \sim transitiv.

Definition 9.5. Sei $G = (E, K)$ ein Graph und definiere die Zusammenhangsrelation \sim wie eben. Die Äquivalenzklassen von E nach \sim sind die **Zusammenhangskomponenten** von G . Die Anzahl der Zusammenhangskomponenten von G ist die **Konnektivitätszahl** $c(G)$ von G .

Beispiel 9.6. Beispiel eines Graphen und seiner Zusammenhangskomponenten.

Definition 9.7. Sei $G = (E, K)$ ein Graph.

- a) G ist ein **Eulerscher Graph**, wenn es einen Pfad $(a_1, a_2, \dots, a_{l-1}, a_l)$ in G gibt, so dass $a_1 = a_l$ und in dem jede Kante (e, f) von G genau einmal vorkommt.
- b) G ist ein **Hamiltonscher Graph**, wenn einen Zyklus (a_1, a_2, \dots, a_l) in G gibt, in dem jede Ecke von G genau einmal vorkommt:

$$a_1 = a_l, \forall a \in E \exists i \leq l \ a = a_i \text{ und } \forall i, j \leq l (i \neq j \rightarrow a_i \neq a_j).$$

9.1 Eulersche Graphen

Satz 9.8. Sei $G = (E, K)$ ein zusammenhängender endlicher Graph mit $|E| \geq 2$. Dann ist G Eulersch genau dann, wenn jede Ecke $a \in E$ geraden Grad $\delta(a)$ hat.

Beweis. Sei G Eulersch. Betrachte eine Ecke $a \in E$. Wähle einen Pfad $(a_1, a_2, \dots, a_{l-1}, a_l)$ in G , so dass $a_1 = a_l$ und in dem jede Kante (e, f) von G genau einmal vorkommt. Wir können weiter annehmen, dass $a = a_1$. Die Menge der zu a inzidenten Kanten in G ist dann:

$$\{a a_2, a_{l-1} a\} \cup \bigcup_{2 < i < l-1, a_i = a} \{a_{i-1} a_i, a_i a_{i+1}\}.$$

Da in dem Pfad jede Kante genau einmal auftritt, ist dies eine disjunkte Vereinigung von jeweils zwei-elementigen Mengen. Ihre Kardinalität ist der Grad von a , diese ist eine gerade Zahl ≥ 2 .

Für die Umkehrung zeigen wir zunächst einen Hilfssatz:

(1) Sei $G' = (E', K')$ ein endlicher Graph, in dem jede Ecke e einen geraden Grad $\delta_{G'}(e)$ hat. Sei $a \in E'$ mit $\delta_{G'}(a) \geq 2$. Dann gibt es einen *geschlossenen* Pfad $(a_1, a_2, \dots, a_{l-1}, a_l)$ in G' mit $l \geq 4$ und $a_1 = a_l = a$, in dem jede Kante ef von G' höchstens einmal auftritt.

Beweis: Wähle einen Pfad $w = (a_1, a_2, \dots, a_{l-1}, a_l)$ von maximaler Länge, so dass $a_1 = a$ und so dass jede Kante ef von K' höchstens einmal in w auftritt. Angenommen, $a_l \neq a$. Dann kommt a_l in den Kanten von w ungerade oft vor. Da der Grad von a_l gerade ist, gibt es eine Kante $a_l a_{l+1}$, die nicht in w vorkommt. Dann tritt in dem Pfad $(a_1, a_2, \dots, a_l, a_l, a_{l+1})$ jede Kante höchstens einmal auf, im Widerspruch zur Maximalität von w . *qed*(1)

Wähle ein $a \in E$ und einen *geschlossenen* Pfad $w = (a, a_2, \dots, a_{l-1}, a)$ maximaler Länge, in dem jede Kante ef von G höchstens einmal auftritt. Sei K_0 die Menge der in w vorkommenden Kanten von G . Angenommen $K_0 \neq K$. Dann setze

$$G' = (E, K \setminus K_0).$$

Wähle eine Kante $ef \in K \setminus K_0$. Da G zusammenhängend ist, wähle einen Weg p in G von a nach e .

Fall 1: Der Weg p verläuft vollständig in w . Dann ist der Endpunkt $e \in w$. Da $ef \in K \setminus K_0$, ist $\delta_{G'}(e) > 0$.

Fall 2: Es gibt eine Kante in p , die nicht in w liegt. Sei $bc \in K \setminus K_0$ die erste solche Kante entlang des Pfades. Dann ist $b \in w$ und ähnlich wie oben ist $\delta_{G'}(b) > 0$.

In jedem Fall können wir also $b \in w$ wählen mit $\delta_{G'}(b) > 0$ hat. Anwendung von (1) auf G' liefert einen geschlossenen Pfad $w' = (b, b_2, \dots, b_{k-1}, b)$ in G' , in dem jede Kante von G' höchstens einmal vorkommt. Wenn $b = a_i$ so ist

$$(a_1, \dots, a_{i-1}, a_i = b, b_2, \dots, b_{k-1}, b = a_i, a_{i+1}, \dots, a_{l-1}, a_l)$$

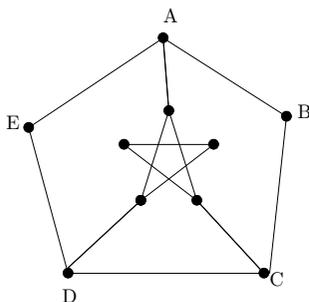
ein längerer Pfad in G , in dem jede Ecke höchstens einmal vorkommt. Dies ist ein Widerspruch zur Maximalität von w . \square

Aus dem vorangehenden Beweis kann man einen *Algorithmus* zur Konstruktion einer Eulertour in einem endlichen zusammenhängenden G extrahieren: man finde zunächst einen geschlossenen Pfad, in dem jede Kante höchstens einmal vorkommt. Dann suche man einen Punkt auf dem Pfad, von dem eine Kante abgeht, die nicht in dem Pfad liegt. Wenn es keinen solchen Punkt gibt, ist man fertig. Ansonsten bilde einen Pfad in G' wie im Beweis und verketten diesen mit dem ursprünglichen Pfad. Durch Iteration dieses Verfahrens wird der Graph ausgeschöpft und man erhält einen Eulerschen Pfad.

Die Frage nach Eulerschen Pfaden in Graphen ist also recht gut algorithmisch beherrschbar. Ein vorgelegter Graph G wird auf Zusammenhang überprüft: ausgehend von einer beliebig gewählten Ecke a wird sukzessiv die Zusammenhangskomponente von a erzeugt. Wenn diese den gesamten Graphen ausschöpft, ist G zusammenhängend. Der Graph G ist weiterhin Eulersch, wenn jede Ecke geraden Grad besitzt.

9.2 Hamiltonsche Graphen

Hamiltonsche Graphen lassen sich algorithmisch nicht einfach beherrschen, es gibt noch viele ungelöste Probleme in diesem Zusammenhang. Die Entscheidung, ob ein vorgelegter Graph Hamiltonsch ist, kann bisher nicht durch einen allgemeinen Algorithmus getroffen werden. Wir verdeutlichen das am Beispiel eines Graphen, bei dem die Eigenschaft nicht-Hamiltonsch zu sein durch ein etwas verwickeltes Argument gezeigt werden kann:



Definition 9.9. Ein **gewichteter Graph** ist eine Struktur $G = (E, \mathbb{R}, K, w)$, wobei (E, K) ein *schlichter Graph* ist und $w: E \times E \rightarrow \mathbb{R}$, so dass

$$\forall x, y \in E \ w(x, y) = w(y, x).$$

Wenn $xy \in K$ so ist $w(x, y) = w(xy)$ das **Gewicht** der Kante xy . Wenn (a_1, \dots, a_l) ein Weg in G ist, so ist das **Gewicht** von G definiert als

$$w(a_1, \dots, a_l) = \sum_{i=1}^{l-1} w(a_i, a_{i+1}).$$

Gewichte können als Aufwand für das Zurücklegen einer Kante betrachtet werden, etwa als Zeit oder Entfernung. Das *Problem des Handlungsreisenden* ist folgende Aufgabe:

Zu einem Hamiltonschen Graphen G finde man einen Hamiltonschen Pfad mit minimalem Gewicht.

Das Problem ist in dieser Form allgemein nicht mit vertretbarem Rechenaufwand lösbar. In seiner allgemeinen Form ist die Lösung des Problems in polynomieller Zeit äquivalent zu $P = NP$. Wegen der grundsätzlichen Bedeutung des Problems - nicht nur für Handlungsreisende - sind eine Reihe von effizienten Näherungsalgorithmen entwickelt worden, die allgemein oder auf Spezialfälle zutreffen. Ein primitiver *suboptimaler* Algorithmus ist es, an jeder Stelle zum nächsten Nachbarn weiterzulaufen, d.h. die Kante von geringstem Gewicht zu benutzen. Dieser Algorithmus wird aber im allgemeinen keinen Hamiltonschen Weg finden, oder einen Hamiltonschen Weg mit suboptimalem Gewicht.

9.3 Bäume

Definition 9.10. Ein Graph $G = (E, K)$ ist ein **Baum**, wenn G zusammenhängend und azyklisch ist.

Satz 9.11. Sei $G = (E, K)$ ein endlicher Graph mit n Ecken und m Kanten. Dann sind folgende Aussagen äquivalent:

- G ist ein Baum.
- Zwischen zwei beliebigen Ecken von G existiert **genau ein** Weg.
- G ist zusammenhängend, und wenn man eine beliebige Kante aus G entfernt, so ist G nicht mehr zusammenhängend.
- G ist zusammenhängend und $m = n - 1$.
- G ist azyklisch, und durch Hinzufügen einer neuen Kante erhält G einen Zyklus.

Beweis. $a) \rightarrow b)$ Sei G Baum. Seien a und b Ecken von G . Da G zusammenhängend ist, gibt es einen Weg w von a nach b . Angenommen $w = (a, a_2, \dots, a_{l-1}, b)$ und $w' = (a, a'_2, \dots, a'_{k-1}, b)$ sind zwei verschiedene Wege von a nach b . Da $w \neq w'$ können wir ein kleinstes $i + 1$ wählen mit $a_{i+1} \neq a'_{i+1}$. Wähle dann minimale Indizes $j, k > i$ mit $a_j = a'_k$. Dann ist

$$(a_i, \dots, a_j = a'_k, a'_{k-1}, \dots, a'_i = a_i)$$

ein Zyklus in G , im Widerspruch zur Azyklizität von G .

$b) \rightarrow c)$ Es gelte b). Betrachte eine Kante ab in G . Nach b) ist dies der einzige Weg von a nach b . Nach Entfernen von ab gibt es keinen Weg mehr von a nach b und der Graph wird unzusammenhängend.

c) \rightarrow d) Wir beweisen die Implikation durch Induktion über die Mächtigkeit $|E|$ von G . Betrachte einen Graphen $G = (E, K)$, der c) erfüllt, und die Implikation gelte für alle kleineren Graphen als G . Wähle eine Kante $ab \in E$.

(1) Der Graph $G' = (E, K \setminus \{ab\})$ hat genau zwei Zusammenhangskomponenten: die durch a bzw. b bestimmt werden.

Beweis: Angenommen, die Ecken a und b lassen sich in G' durch einen Weg p verbinden. Betrachte beliebige Ecken $e, f \in E$. Da G zusammenhängend ist, gibt es in G einen Weg w von e nach f . Falls die Kante ab in w vorkommt, ersetze diese jeweils durch den Weg p . Wir erhalten einen Pfad von e nach f in G' . Also ist G' zusammenhängend, im Widerspruch zu c).

Betrachte nun eine beliebige Ecke $e \in E$. Da G zusammenhängend ist, gibt es in G einen Weg w von e nach a .

Fall 1: w enthält die Kante ab nicht. Dann ist w ein Weg in G' von e nach a . Also liegt e in der Zusammenhangskomponente von a in G' .

Fall 2: w enthält die Kante ab . Sei $w' = (e, \dots, f)$ maximales Anfangsstück von w , das die Kante ab nicht enthält. Dann ist $f = a$ oder $f = b$, w' ist ein Weg in G' , und e ist in der Zusammenhangskomponente von a oder von b in G' . *qed(1)*

Sei G_a die Zusammenhangskomponente von a und G_b die Zusammenhangskomponente von b bezüglich G' .

(2) G_a und G_b erfüllen c).

Beweis: Als Zusammenhangskomponenten sind G_a und G_b zusammenhängend. Betrachte eine beliebige Kante ef in G_a . Die Ecken e und f sind dann nicht mehr in $K \setminus \{ef\}$ durch einen Weg verbindbar. Dies impliziert, dass e und f nach Fortlassen von ef auch nicht in G_a verbindbar sind. *qed(2)*

Nach Induktionsvoraussetzung gilt d) für G_a und G_b : Wenn n_a bzw. m_a die Anzahl der Ecken und Kanten von G_a ist, so gilt $m_a = n_a - 1$. Entsprechend ist $m_b = n_b - 1$. Da die Trägermenge des Graphen G die disjunkte Vereinigung der Trägermengen der Graphen G_a bzw. G_b ist, ist

$$n = n_a + n_b.$$

Wir hatten schon oben überlegt, dass die einzige gewöhnliche Verbindung von G_a und G_b ist entlang der Kante ab . Es gibt keine weiteren Kanten, die G_a und G_b verbinden. Für die Anzahlen der Kanten gilt:

$$m = m_a + m_b + 1.$$

Nach Induktionsvoraussetzung ist $m_a = n_a - 1$ und $m_b = n_b - 1$. Damit ist

$$m = m_a + m_b + 1 = n_a - 1 + n_b - 1 + 1 = (n_a + n_b) - 1 = n - 1.$$

Also gilt d) auch für G .

d) \rightarrow e) Wir beweisen die Implikation durch Induktion über die Eckenzahl $n \in \mathbb{N} \setminus \{0\}$.

Induktionsanfang: Sei $n = 1$. Dann ist G der triviale Graph \bullet mit einer Ecke. G erfüllt die Eigenschaft e) aus trivialen Gründen.

Induktionsschritt: Angenommen, die Implikation gilt für n . Betrachte einen zusammenhängenden Graphen $G = (E, K)$ mit $n + 1$ Ecken und n Kanten.

(3) Es gibt eine Ecke $a \in E$ mit $\delta_G(a) = 1$.

Beweis. Da G zusammenhängend ist und $|E| \geq 2$, ist $\forall a \in E \delta_G(a) \geq 1$. Angenommen es gilt $\forall a \in E \delta_G(a) \geq 2$. Zu jeder Ecken gehören dann zwei Kanten, von denen jede wiederum zu zwei Ecken adjazent ist. Damit ist

$$(n + 1) \cdot \frac{2}{2} = n + 1$$

eine untere Schranke für die Anzahl der Kanten, im Widerspruch zur Voraussetzung. *qed(3)*

Wähle ein $a \in E$ mit $\delta_G(a) = 1$. Wähle das eindeutig bestimmte $b \in E$ mit $ab \in K$. Wir entfernen die Ecke a und die Kante ab aus G . Der Graph

$$G' = (E \setminus \{a\}, K \setminus \{ab\})$$

ist zusammenhängend mit Eckenzahl n und Kantenzahl $n - 1$. Nach Induktionsvoraussetzung erfüllt G' die Eigenschaft e): G' ist azyklisch und durch Hinzufügen einer neuen Kante erhält G' einen Zyklus.

(4) G ist azyklisch.

Beweis: Angenommen, $z = (a_1, \dots, a_l = a_1)$ ist ein Zyklus in G . Jede Ecke in einem Zyklus hat mindestens Grad 2. Da $\delta_G(a) = 1$ ist, kommt a nicht in z vor. Also ist z ein Zyklus in G' , im Widerspruch zur Azyklichkeit von G' . *qed*(4)

(5) Seien $e, f \in E$, $e \neq f$ und $ef \notin K$. Dann enthält der Graph $G^+ = (E, K \cup \{ef\})$ einen Zyklus.

Beweis: Wenn $e, f \in E \setminus \{a\}$, so ist ef eine neue Kante für G' . G' enthält nach Hinzufügen von ef einen Zyklus, der auch Zyklus in G^+ ist. Andernfalls können wir annehmen, dass $e = a$. Dann ist $f \neq b$ und $f, b \in E \setminus \{a\}$. Da G' zusammenhängend ist, gibt es einen Weg $(f = a_1, a_2, \dots, a_l = b)$ von minimaler Länge, der f und b verbindet. Dann ist $(e = a, f = a_1, a_2, \dots, a_l = b, a)$ ein Zyklus in dem Graphen G^+ . *qed*(5)

e) \rightarrow a) Angenommen, G erfüllt e). Für a) genügt es zu zeigen, dass G zusammenhängend ist. Betrachte Ecken $a, b \in E$. Falls $ab \in K$, so gibt es trivialerweise einen Weg von a nach b . Falls $ab \notin K$, füge ab als neue Kante zu G hinzu. Dann ist der erweiterte Graph zyklisch. Wähle einen Zyklus $z = (a_1, a_2, \dots, a_l)$ im erweiterten Graphen. Da der ursprüngliche Graph azyklisch war, muss die neue Kante ab in z vorkommen. Da z ein Zyklus ist, kommt ab genau einmal in z vor. Ohne Einschränkung ist $a = a_1 = a_l$ und $b = a_2$. Dann ist $(b = a_2, \dots, a_l = a)$ ein Weg von b nach a im Graphen G . Also ist G zusammenhängend. \square

Jeder Graph enthält Teilbäume, z.B. die einpunktigen Teilgraphen. Von besonderem Interesse sind maximale Teilbäume:

Definition 9.12. Sei $T = (E, K')$ ein Teilgraph von $G = (E, K)$, d.h. $K' \subseteq K$. Dann ist T ein **Spannbaum** von G , wenn T ein Baum ist. Ein Spannbaum ist also ein Teilbaum, der alle Ecken von G umfasst.

Wenn G einen Spannbaum enthält, so ist G ein zusammenhängender Graph. Zu einem gegebenen zusammenhängenden Graphen gibt es umgekehrt (in der Regel viele) Spannbäume. Ein Problem, das auf den ersten Blick dem Problem des Handlungsreisenden ähnelt, ist die Bestimmung eines bezüglich einer Wichtung minimalen Spannbaums:

Definition 9.13. Sei $G = (E, \mathbb{R}, K, w)$ ein endlicher gewichteter zusammenhängender Graph. Ein **minimaler Spannbaum** (minimal spanning tree, MST) ist ein Spannbaum $T = (E, K')$ von G , dessen Gesamtgewicht

$$\sum_{ab \in K'} w(a, b) \in \mathbb{R}$$

minimal ist.

Probleme dieser Art treten ebenfalls bei der Routenplanung, aber z.B. auch bei der Verdrahtung von Schaltkreisen (Chips) auf. Der folgende Algorithmus ermittelt einen minimalen Spannbaum T zu einem gegebenen endlichen gewichteten Graphen $G = (E, \mathbb{R}, K, w)$:

```

begin
  T :=  $\emptyset$ 
  E' := E
  while E'  $\neq \emptyset$  do
    begin
      e' := eine Kante in E' von kleinstem Gewicht
      T := T  $\cup \{e'\}$ 
      E' := Menge der in E-T enthaltenen Kanten, die genau einen      in T
      liegenden Endpunkt haben
    end
  end
end

```

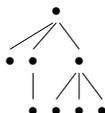
Der Beweis, dass dieser Algorithmus einen *minimalen* Spannbaum liefert, ist nicht trivial. Einfacher ist es zu überlegen, dass der Algorithmus einen Spannbaum liefert. Der Spannbaum-Algorithmus ist übrigens auch ein *sub-optimaler* Algorithmus für den das *traveling-salesman*-Problem. Man kann einen Spannbaum so durchlaufen, dass jede Kante höchstens zweimal durchlaufen wird und jede Ecke des Baumes erreicht wird. Zu der eingehenden Analyse der Algorithmen würde noch eine Komplexitätsbetrachtung gehören. Es ist klar, dass die Komplexität des obigen Algorithmus proportional zu der Anzahl der $n - 1$ Schleifendurchläufe ist, wobei $n = |E|$ ist. Ein Schleifendurchlauf verlangt die Bestimmung der Menge E' von Kanten mit bestimmten Eigenschaften. Die Anzahl der Kanten ist die Anzahl $\frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$ aller 2-Kombinationen aus der Menge E der Ecken. Wenn wir bei der Bildung von E' von dieser Maximalzahl ausgehen, ist die Anzahl der Rechenschritte in der Schleife etwa proportional zu n^2 . Insgesamt erhalten wir eine obere Schranke der Rechenschritte, die proportional zu n^3 ist. Diese Abschätzung lässt sich noch verbessern.

9.4 Wurzelbäume (rooted trees)

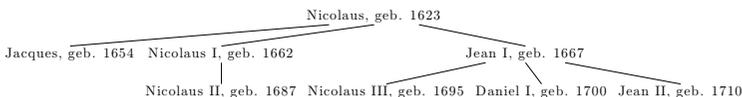
In einem Baum kann man eine Ecke als *Wurzel* auszeichnen. Alle anderen Ecken lassen sich dann auf eindeutigen zyklusfreien Wegen von der Wurzel aus erreichen. Die Wurzel wird als *Konstante* formalisiert.

Definition 9.14. Eine Graph $T = (E, K, w)$ ist ein **Wurzelbaum**, wenn $T = (E, K)$ ein Baum ist und $w \in E$.

Wir stellen Wurzelbäume als nach unten wachsende Baum-artige Diagramme dar:



Ein Beispiel dafür sind *Stammbäume*, wie dieser der Mathematiker-Familie BERNOULLI:



Hierdurch werden weitere Begriffe nahegelegt: wenn w die Wurzel von T ist und a und b sind adjazente Ecken in T , so dass a näher als b von der Wurzel w liegt, so ist b ein **Kind** von a . Diejenigen Ecken von T , die keine Kinder haben, heißen **Blätter** des Baumes. Durch die Länge der kürzesten Wege zur Wurzel wird auf Wurzelbäumen eine Art *Abstand* zur Wurzel bestimmt, die man auch als zeitliche Abfolgen interpretieren kann.

Von besonderer Bedeutung sind *binäre Bäume*, in denen jede Ecke, die kein Blatt ist, genau zwei Kinder hat:



Kapitel 10

Gerichtete Graphen

Wenn man einen schlichten Graphen als System von gewöhnlichen Straßen auffassen kann, so kann man ein System von Einbahnstraßen durch einen *gerichteten Graphen* modellieren:

Definition 10.1. Ein (schlichter) gerichteter Graph (Digraph) ist eine Struktur $G = (E, K)$ wobei K eine irreflexive 2-stellige Relation auf der Menge E ist. Die Elemente von E heißen wie bisher Ecken. Ein geordnetes Paar (a, b) ist eine Kante von G , wenn $(a, b) \in K$; wir schreiben auch ab statt (a, b) . Die Kanten eines gerichteten Graphen haben eine Richtung: ab geht von a nach b . Es ist möglich, dass es eine Kante von a nach b und eine andere von b nach a gibt.

Wege und Zyklen auf gerichteten Graphen respektieren die auf den Kanten vorgegebenen Richtungen:

Definition 10.2. Sei $G = (E, K)$ ein schlichter gerichteter Graph.

a) Eine Folge (a_0, \dots, a_{l-1}) von Ecken ist ein (gerichteter) Weg der Länge l in G , wenn

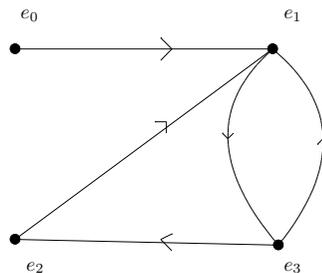
$$\forall i < l - 1 \ a_i K a_{i+1}.$$

b) Ein Weg (a_0, \dots, a_{l-1}) ist ein (gerichteter) Zyklus oder Kreis in G , wenn $l \geq 3$, a_0, \dots, a_{l-2} paarweise verschieden sind und $a_0 = a_{l-1}$.

c) Der Graph G ist azyklisch, wenn er keinen gerichteten Zyklus enthält.

d) Der Graph G ist zusammenhängend, wenn es für alle $a, b \in E$ einen gerichteten Weg $(a, a_1, \dots, a_{l-2}, b)$ in G gibt. Wir sagen dann, dass $(a, a_1, \dots, a_{l-2}, b)$ ein Weg von a nach b ist.

Ein gerichteter Graph wird zweidimensional dargestellt durch eine Konfiguration von Ecken, zwischen die *gerichtete* Kanten eingezeichnet sind. Die Richtung der Kanten wird durch einen Pfeil markiert.



Gerichtete Graphen haben Adjazenzmatrizen, die im Allgemeinen nicht symmetrisch sind. Im obigen Beispiel lautet die Adjazenzmatrix

$$M = \begin{pmatrix} F & W & F & F \\ F & F & F & W \\ F & W & F & F \\ F & W & W & F \end{pmatrix}.$$

Eine Adjazenzmatrix ist ein Beispiel für eine Boolesche Matrix.

Definition 10.3. Seien $n, m \in \mathbb{N}$. Eine Boolesche $n \times m$ -Matrix ist eine Folge $M = (a_{ij} \mid i = 0, \dots, n-1, j = 0, \dots, m-1)$ von Elementen $a_{ij} \in \{F, W\}$, die wir suggestiv schreiben können als

$$\begin{pmatrix} a_{00} & a_{01} & \dots & a_{0m-1} \\ a_{10} & a_{11} & \dots & a_{1m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \dots & a_{n-1,m-1} \end{pmatrix}.$$

Die Menge der Booleschen $n \times m$ -Matrizen wird mit $\text{BM}(n \times m)$ bezeichnet.

Boolesche Matrizen können auf der Basis der logischen Operationen **oder** und **und** ähnlich wie Matrizen über Körpern addiert und multipliziert werden.

Definition 10.4. a) Für Boolesche Matrizen $(a_{ij}), (b_{ij}) \in \text{BM}(n \times m)$ definiere die Summe

$$(a_{ij}) + (b_{ij}) = (c_{ij}) \in \text{BM}(n \times m)$$

durch

$$c_{ij} = a_{ij} \text{ oder } b_{ij}.$$

b) Für Boolesche Matrizen $A = (a_{i,k}) \in \text{BM}(r \times n)$ und $B = (b_{k,j}) \in \text{BM}(n \times m)$ definiere das Produkt

$$A \cdot B = \left(\bigvee_{k=0}^{n-1} (a_{i,k} \text{ und } b_{k,j}) \right)_{i=0, \dots, r-1, j=0, \dots, m-1} \in \text{BM}(r \times m),$$

wobei

$$\bigvee_{k=0}^{n-1} (a_{i,k} \text{ und } b_{k,j}) = (a_{i,0} \text{ und } b_{0,j}) \text{ oder } \dots \text{ oder } (a_{i,n-1} \text{ und } b_{n-1,j}). \quad (10.1)$$

Für die oben betrachtete Adjazenzmatrix gilt

$$M^2 = M \cdot M = \begin{pmatrix} F & W & F & F \\ F & F & F & W \\ F & W & F & F \\ F & W & W & F \end{pmatrix} \cdot \begin{pmatrix} F & W & F & F \\ F & F & F & W \\ F & W & F & F \\ F & W & W & F \end{pmatrix} = \begin{pmatrix} F & F & F & W \\ F & W & W & F \\ F & F & F & W \\ F & W & F & W \end{pmatrix}.$$

Der Matrixeintrag in Zeile i und Spalte j beschreibt, ob man von der Ecke e_i zur Ecke e_j mit einem gerichteten Weg der Länge 2 gelangen kann: dass ist genau dann möglich, wenn es eine Zwischenecke e_k gibt, sodass man von e_i zu e_k und von e_k zu e_j jeweils mit Wegen (=Kanten) der Länge 1 gelangen kann. Dieses entspricht der Gleichung 9.1.

Mit Hilfe vollständiger Induktion kann man für $n \in \mathbb{N}$ zeigen: der Eintrag in Zeile i und Spalte j der Matrix M^n ist genau dann $=W$, wenn es von der Ecke e_i zur Ecke e_j einen gerichteten Weg der Länge n gibt. Die Frage der Erreichbarkeit der Ecke e_j von e_i mit Wegen beliebiger Länge kann folgendermaßen beschrieben werden:

Definition 10.5. Es sei $M \in \text{BM}(n \times n)$ die Adjazenzmatrix eines gerichteten Graphen $G = (E, K)$ mit n Ecken. Definiere die Erreichbarkeitsmatrix von G als

$$M^* = M + M^2 + \dots + M^n.$$

Die Berechnung von M^* wird durch die Matrixoperationen für großes n sehr komplex. Der folgende **Algorithmus von Warshall** ist ein effizienterer Algorithmus zur Bestimmung von $W = M^*$:

```

begin
  W:=M
  for k=1 to n do
    for i=1 to n do
      for j=1 to n do
        W(i,j):=(W(i,j) oder (W(i,k) und W(k,j)));
      end
    end
  end
end

```

Die Matrix $W = M$ wird in n Durchläufen modifiziert. In den Durchläufen werden sukzessiv die Zeilen $i = 1, \dots, n$ betrachtet. Die Berechnung der i -ten Zeile im k -ten Durchlauf hängt davon ab, ob der Wert $W(i, k) = F$ oder $= W$ ist:

Fall 1: $W(i, k) = F$. Dann wird die i -te Zeile nicht verändert.

Fall 2: $W(i, k) = W$. Dann entsteht die i -te Zeile als punktweises **oder** von i -ter und k -ter Zeile.

Wir führen diesen Algorithmus für die obige Adjazenzmatrix $W_0 = M$ aus. Die Matrizen W nach den n Durchläufen bezeichnen wir jeweils als W_1, \dots, W_n .

$$\begin{aligned}
 W_0 &= \begin{pmatrix} F & W & F & F \\ F & F & F & W \\ F & W & F & F \\ F & W & W & F \end{pmatrix} \\
 W_1 &= \begin{pmatrix} F & W & F & F \\ F & F & F & W \\ F & W & F & F \\ F & W & W & F \end{pmatrix} \\
 W_2 &= \begin{pmatrix} F & W & F & W \\ F & F & F & W \\ F & W & W & W \\ F & W & W & W \end{pmatrix} \\
 W_3 &= \begin{pmatrix} F & W & F & W \\ F & F & F & W \\ F & W & W & W \\ F & W & W & W \end{pmatrix} \\
 W_4 &= \begin{pmatrix} F & W & W & W \\ F & W & W & W \\ F & W & W & W \\ F & W & W & W \end{pmatrix}
 \end{aligned}$$

Das bedeutet, dass von jeder Ecke e_i die Ecke e_0 nicht und die Ecken e_1, e_2, e_3 in der Tat durch einen gerichteten Weg positiver Länge erreicht werden können.