Logische Programmierung

2. Logik und Mengenlehre

Elmar Eder

22. März 2021

Logik

- Logische Aussagen
- Verknüpfung von logischen Aussagen
- Wahre und falsche Aussagen
- Objekte und Prädikate
- All-Aussagen und Existenz-Aussagen

Logische Aussagen

Aussagen (z.B. deutsche Sätze), von denen es Sinn macht, zu sagen, dass sie wahr oder falsch sind.

Beispiele (Aussagen)

- Die Erde ist ein Planet. (wahr)
- Dora ist die Mutter von Hans. (je nach Kontext wahr oder falsch)
- $5 < 3$ (falsch)

Beispiele (Keine Aussagen)

- Was ist die Hauptstadt von Österreich?
- Komm her!

Verknüpfung von logischen Aussagen

Logische Aussagen A und B können verknüpft werden zu zusammengesetzten Aussagen

- $\neg A$ („nicht A“)
- $A \wedge B$ („A und B“)
- $A \vee B$ („A oder B“)
- $A \Rightarrow B$ („Wenn A, dann B“)
- $A \Leftrightarrow B$ („A genau dann, wenn B“).

In der formalen Logik schreibt man $\rightarrow$ statt $\Rightarrow$ und $\leftrightarrow$ statt $\Leftrightarrow$.
Statt $A \Rightarrow B$ oder $A \rightarrow B$ schreibt man auch $B \Leftarrow A$ oder $B \leftarrow A$ („B folgt aus A“ oder „B, wenn A“).
In Prolog wird das stilisiert zu $B :- A$.

Wahre und falsche Aussagen

- In einem konkreten Kontext, wir sagen auch bei einer konkreten Interpretation, ist eine Aussage wahr oder falsch.
- Wir sagen auch, sie hat den Wahrheitswert w (für „wahr“) oder f (für „falsch“).
- Der Wahrheitswert einer mit Verknüpfungen zusammengesetzten Aussage ergibt sich aus den Wahrheitswerten der Einzelaussagen durch eine Wahrheitswerttabelle.

Wahrheitswerttabellen

A	$\neg A$
w	f
f	w

A	B	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
w	w	w	w	w	w
w	f	f	w	f	f
f	w	f	w	w	f
f	f	f	f	w	w

Objekte und Prädikate

- Objekte (Individuen, Dinge, Gegenstände)
  z.B. Dora, Hans
- Prädikate machen Aussagen über Objekte
  z.B. „ist Mutter von“:
  Aussage: Dora ist Mutter von Hans.
  Das Prädikat „ist Mutter von“ macht hier eine Aussage über die Objekte „Dora“ und „Hans“.

Schreibweisen

- in der Logik: Mutter(dora,hans) oder $M(d, h)$
- in Prolog: mutter(dora,hans)

All-Aussagen und Existenz-Aussagen

In der Logik möchte man Aussagen machen wie

1. Alle Städte sind Ortschaften.
2. Es gibt Ortschaften, die keine Städte sind.

Das erste ist eine All-Aussage, das zweite eine Existenz-Aussage.

Hierzu verwenden wir Variablen, z.B. x.
Eine Variable steht für ein beliebiges nicht festgelegtes Objekt.
In der Logik schreibt man Variablen klein, in Prolog groß.

Aussageform z.B. „x ist eine Stadt“
(wie Aussage, darf aber Variablen enthalten)

Formel z.B. $\mathrm{Stadt}(x)$
(in der formalen Sprache der Logik formulierte Aussage oder Aussageform)

All-Aussagen und Existenz-Aussagen

Formulierung mit Variable x:

1. Für alle x gilt: „Wenn x eine Stadt ist, dann ist x eine Ortschaft“.
2. Es gibt ein x, sodass x eine Ortschaft ist und x nicht eine Stadt ist.

Als Formeln in der formalen Sprache der Logik:

1. $\forall x\Big(\mathrm{Stadt}(x) \to \mathrm{Ortschaft}(x)\Big)$
2. $\exists x\Big(\mathrm{Ortschaft}(x) \land \neg\mathrm{Stadt}(x)\Big)$

All-Aussagen und Existenz-Aussagen

Allgemein: Sei x eine Variable und F eine Formel.

1. $\forall x F$ bedeutet „Für alle x gilt F“.
2. $\exists x F$ bedeutet „Es gibt ein x, sodass F gilt“.

Das Zeichen $\forall$ heißt All-Quantor.
Das Zeichen $\exists$ heißt Existenz-Quantor.
F ist in $\forall x F$ all-quantifiziert, in $\exists x F$ existenz-quantifiziert.

Programmklauseln in Prolog als All-Aussagen

In Prolog sind alle Programmklauseln als all-quantifiziert über alle enthaltenen Variablen zu verstehen.

Beispiel

```
grossvater(G,K):- kind(K,X), kind(X,G), maennlich(G).
```

bedeutet:

$$\forall G\,\forall K\,\forall X\left(\mathrm{grossvater}(G,K) \leftarrow \mathrm{kind}(K,X) \wedge \mathrm{kind}(X,G) \wedge \mathrm{maennlich}(\mathrm{G})\right)$$

Für alle G, alle K und alle X gilt:
„G ist Großvater von K, wenn
K Kind von X und
X Kind von G und
G männlich ist“.

Anfragen in Prolog als Existenz-Aussagen

In Prolog kann man Anfragen als Existenz-Aussagen betrachten.

Beispiel

Die Anfrage

```
?- kind(K,P).
```

gelingt genau dann, wenn Prolog beweisen kann, dass aus dem Programm die Existenzaussage

$$\exists K \exists P \, \mathrm{kind}(K, P)$$

logisch folgt.

Prolog beweist eine solche Existenzaussage immer konstruktiv, d.h. in unserem Beispiel, es findet tatsächlich Werte (Instantiierungen) für die Variablen K und P, für die $\mathrm{kind}(K, P)$ logisch aus dem Programm folgt. Prolog gibt diese Instantiierungen als Antworten aus.

Mengen und ihre Elemente

Definition

Eine Menge ist eine Zusammenfassung von wohlunterscheidbaren Dingen, ihren Elementen.

Beispiel

Beispiel: Die Menge $\{2, 3, 7\}$ hat die Elemente 2, 3 und 7.

Definition

$x \in A$ bedeutet: x ist Element der Menge A.
$x \notin A$ bedeutet: x ist nicht Element der Menge A.

Beispiele

$2 \in \{2, 3, 7\}$ und $4 \notin \{2, 3, 7\}$.

Extensionalität, leere Menge

Extensionalität

Zwei Mengen sind genau dann gleich, wenn sie die gleichen Elmente haben.

Für zwei Mengen A und B gilt also

$$A = B \iff \forall x(x \in A \Leftrightarrow x \in B).$$

Beispiele

$\{2, 3, 2\} = \{3, 2\}$ und $\{a, b\} = \{b, a\}$.

Insbesondere gibt es nur eine leere Menge $\{\}$,
üblicherweise geschrieben als $\emptyset$.

Zahlenmengen

In der Mathematik häufig verwendete Mengen von Zahlen sind

- Die Menge $\mathbb{N}$ der natürlichen Zahlen $0, 1, 2, 3, \ldots$.
  (In der Mathematik wird die 0 meist nicht dazu gerechnet, wohl aber in der Mengenlehre, in der Logik und in der Informatik.)
- Die Menge $\mathbb{Z}$ der ganzen Zahlen (englisch integers) $0, 1, -1, 2, -2, \ldots$
  Testprädikat in Prolog: `integer/1`
- Die Menge $\mathbb{Q}$ der rationalen Zahlen (Bruchzahlen), z.B. $-7/4$
- Die Menge $\mathbb{R}$ der reellen Zahlen, z.B. $-7/4$ sowie π und e
  (im Computer dargestellt als Fließkommazahlen, englisch floating point numbers, z.B. 3.5e-3). Testprädikat in Prolog: `float/1`
- Die Menge $\mathbb{C}$ der komplexen Zahlen, z.B. i, $2 + 3i$ und $-4 + \frac{5}{3}i$.
  Dabei ist $i = \sqrt{-1}$.

Komprehension

Definition (Komprehension)

Für eine Variable x und eine Aussageform oder Formel F über x ist $\{x \mid F\}$ definiert als die Menge aller x, für die F gilt.

Beispiel

$\{x \mid x \text{ ist Primzahl}\}$ ist die Menge der Primzahlen.

Aus dem Prinzip der Extensionalität folgt, dass eine Menge eindeutig durch ihre Elemente bestimmt ist. Also ist $\{x \mid F\}$ eindeutig bestimmt.

Komprehension

Oft geht man von einer Grundmenge oder Definitionsmenge D aus.

Definition (Komprehension)

$\{x \in D \mid F\}$ ist definiert als die Menge aller $x \in D$, für die F gilt.

Beispiele

Grundmenge sei die Menge $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ der natürlichen Zahlen. Dann gilt

- $\{x \in \mathbb{N} \mid x \text{ ist gerade}\} = \{0,2,4,6,\dots\}$
- $\{x \in \mathbb{N} \mid x \geq 3 \wedge x \leq 7\} = \{3, 4, 5, 6, 7\}$

Manchmal schreibt man links vom $\mid$ einen zusammengesetzten Term. Beispiel: $\{x + 1 \mid x \in \{2, 6, 9\}\}$ ist die Menge aller $x + 1$ mit $x \in \{2, 6, 9\}$, also die Menge $\{3, 7, 10\}$.

Teilmenge, Operationen auf Mengen

Definition

Seien A und B Mengen.

- A ist eine Teilmenge von B, in Zeichen $A \subseteq B$, wenn jedes Element von A auch Element von B ist (also wenn $\forall x(x \in A \Rightarrow x \in B)$ gilt).
- $A \cap B := \{x \mid x \in A \wedge x \in B\}$ (Durchschnitt von A und B)
- $A \cup B := \{x \mid x \in A \vee x \in B\}$ (Vereinigung von A und B)
- $A \setminus B := \{x \mid x \in A \wedge x \notin B\}$ (Mengendifferenz von A und B)
- $\mathcal{P}(A) = \{X \mid X \subseteq A\}$ (Potenzmenge von A)
- Die Mächtigkeit $|A|$ einer endlichen Menge A ist die Anzahl ihrer Elemente.

Teilmenge, Operationen auf Mengen

Beispiele

Seien a, b, c, d, e fünf verschiedene Dinge.

- $\{a, c, d\} \subseteq \{a, b, c, d, e\}$
- $\{a, c, d\} \cap \{b, c, d, e\} = \{c, d\}$
- $\{a, c, d\} \cup \{b, c, d, e\} = \{a, b, c, d, e\}$
- $\{a, c, d\} \setminus \{b, c, d, e\} = \{a\}$
- $\mathcal{P}(\{a, c, d\}) = \{\emptyset, \{a\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}, \{c, d\}, \{a, c, d\}\}$
- $|\{a, b, c, d, e\}| = 5$
- $|\{a, b, c\}| = 3$
- $|\mathcal{P}(\{a, c, d\})| = 8$

Eigenschaften der Teilmengenrelation $\subseteq$

Für Mengen A, B und C gilt

$$A = B \iff A \subseteq B \wedge B \subseteq A$$

Die Teilmengenrelation $\subseteq$ ist eine partielle Ordnung auf der Klasse aller Mengen, d.h. für alle Mengen A, B und C gilt

- $A \subseteq A$ (Reflexivität)
- Wenn $A \subseteq B$ und $B \subseteq A$, dann $A = B$. (Antisymmetrie)
- Wenn $A \subseteq B$ und $B \subseteq C$, dann $A \subseteq C$. (Transitivität)

Eigenschaften der Mengenoperationen

- $A \cap A = A$ (Idempotenz von $\cap$)
- $A \cup A = A$ (Idempotenz von $\cup$)
- $A \cap B = B \cap A$ (Kommutativität von $\cap$)
- $A \cup B = B \cup A$ (Kommutativität von $\cup$)
- $(A \cap B) \cap C = A \cap (B \cap C)$ (Assoziativität von $\cap$)
- $(A \cup B) \cup C = A \cup (B \cup C)$ (Assoziativität von $\cup$)
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (Distributivität von $\cap$ über $\cup$)
- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ (Distributivität von $\cup$ über $\cap$)
- $A \cap (A \cup B) = A$ (Absorptionsgesetz)
- $A \cup (A \cap B) = A$ (Absorptionsgesetz)

Wegen der Assoziativitätsgesetze lässt man in Ausdrücken wie $(A \cap B) \cap C$ die Klammern üblicherweise weg.

Durchschnitt und Vereinigung einer Menge von Mengen

Man definiert nicht nur Durchschnitt und Vereinigung von zwei Mengen, sondern auch Durchschnitt und Vereinigung einer beliebigen (im Falle des Durchschnitts nichtleeren) Menge von Mengen:

Definition

Sei A eine Menge von Mengen.

- $\bigcap A := \{x \mid \forall X \in A\colon x \in X\}$, wenn $A \neq \emptyset$.
- $\bigcup A := \{x \mid \exists X \in A\colon x \in X\}$.

Dabei ist „$\forall X \in A\colon x \in X$“ zu lesen als „für alle $X \in A$ ist $x \in X$“.
Dies ist äquivalent zu $\forall X(X \in A \Rightarrow x \in X)$.
„$\exists X \in A\colon x \in X$“ ist zu lesen als „Es gibt ein $X \in A$ mit $x \in X$“.
Dies ist äquivalent zu $\exists X(X \in A \wedge x \in X)$.

Durchschnitt und Vereinigung einer Menge von Mengen

Beispiele

- $\bigcap\{\{a,b,c,d\},\{b,c,e\},\{c,f\}\} = \{c\}$
- $\bigcup\{\{a,b,c,d\},\{b,c,e\},\{c,f\}\} = \{a,b,c,d,e,f\}$
- $\bigcup \emptyset = \emptyset$

Für Mengen X, Y und Z gilt:

- $\bigcap\{X\} = X$
- $\bigcup\{X\} = X$
- $\bigcap\{X,Y\} = X \cap Y$
- $\bigcup\{X,Y\} = X \cup Y$
- $\bigcap\{X,Y,Z\} = X \cap Y \cap Z$
- $\bigcup\{X,Y,Z\} = X \cup Y \cup Z$

Tupel

- Ein Paar ist eine Zusammenfassung von zwei Dingen.
  Beispiel: (hans,anna).
- Allgemein: Ein *n*-Tupel ist eine Zusammenfassung von *n* Dingen.
  Beispiel: $(2, 3, 7)$.
- Im Gegensatz zu Mengen kommt es bei Tupeln auf die Reihenfolge an und darauf, ob ein Ding mehrfach oder nur einfach darin vorkommt:
  $(2, 3, 2) \neq (3, 2)$ und $(\texttt{hans}, \texttt{anna}) \neq (\texttt{anna}, \texttt{hans})$.
- Wörter für *n*-Tupel
  2-Tupel: Paar
  3-Tupel: Tripel
  4-Tupel: Quadrupel
  5-Tupel: Quintupel

Extensionalität für Tupel

Extensionalität für Tupel

Zwei Tupel $(x_1, \ldots, x_m)$ und $(y_1, \ldots, y_n)$ sind genau dann gleich, wenn $m = n$ ist und für alle $i = 1, \ldots, m$ gilt $x_i = y_i$.

Kartesisches Produkt

Definition

Wenn A und B Mengen sind, dann ist das kartesische Produkt $A \times B$ von A und B die Menge aller Paare (x, y) mit $x \in A$ und $y \in B$.

Allgemein:

Definition

- Wenn $A_1, \ldots, A_n$ Mengen sind, dann ist das kartesische Produkt $A_1 \times \cdots \times A_n$ von $A_1, \ldots, A_n$ die Menge der n-Tupel $(x_1, \ldots, x_n)$ mit $x_1 \in A_1 \wedge \cdots \wedge x_n \in A_n$.
- $A^n := \underbrace{A \times \cdots \times A}_{n \text{ mal}}$

Relation, Prädikat

Definition (Relation)

Eine n-stellige Relation R ist eine Teilmenge eines kartesischen Produkts $A_1 \times \cdots \times A_n$.

Definition (Prädikat)

In der Logik nennt man dann R ein n-stelliges Prädikat und statt $(x_1, \ldots, x_n) \in R$ schreibt man $R(x_1, \ldots, x_n)$.

Beispiel

- $A_1 = A_2 =$ Menge der Personen in einer Familie.
- $R = \texttt{kind} = \{(x_1, x_2) \mid x_1 \text{ ist Kind von } x_2\}$.
- `kind(hans,anna)` bedeutet: Hans ist ein Kind von Anna.

Funktion

Der Begriff der Funktion

Eine Funktion oder Abbildung $f : A \to B$ von einer Menge A in eine Menge B ist eine Zuordnung, die jedem Element x von A ein und nur ein Element y von B zuordnet.

- A heißt Definitionsbereich von F.
- B heißt Zielbereich von F.
- x heißt Argument.
- y heißt Funktionswert und wird mit $f(x)$ bezeichnet.

Funktion

Beispiel

- Durch
$$f(x) := x + 1$$
wird eine Funktion $f : \mathbb{N} \to \mathbb{N}$ (die Nachfolgerfunktion) auf der Menge $\mathbb{N}$ der natürlichen Zahlen definiert.
- Sie ordnet jeder natürlichen Zahl ihren Nachfolger zu.
- Zum Beispiel ordnet sie der Zahl 4 die Zahl 5 zu. Es ist $f(4) = 5$.
- Definitions- und Zielbereich ist die Menge $\mathbb{N}$.

Graph einer Funktion

Der Graph einer Funktion

Der Graph Graph(f) einer Funktion $f : A \to B$ ist die Menge

$$\{(x, f(x)) \mid x \in A\}$$

Der Graph G von f ist eine Relation $G \subseteq A \times B$, also eine Menge von Paaren von jeweils einem Element von A und einem Element von B.

Beispiel

Der Graph der Nachfolgerfunktion ist die unendliche Menge $\{(0, 1), (1, 2), (2, 3), (3, 4), \dots\}$.

Mathematische Definition des Begriffs der Funktion

Mathematischen definieren wir die Funktion als Tripel (G, A, B):

Definition

Eine Funktion oder Abbildung $f : A \to B$ von einer Menge A in eine Menge B ist ein Tripel (G, A, B), wobei gilt:

- $G \subseteq A \times B$.
- Für jedes $x \in A$ gibt es ein und nur ein $y \in B$ mit $(x, y) \in G$.

Es ist dann:

- G der Graph von f.
- A der Definitionsbereich von f.
- B der Zielbereich von f.
- $f(x)$ das einzige Element y in B mit $(x, y) \in G$.

Darstellung einer Funktion durch eine Tabelle

Beispiel

Sei A die (endliche) Menge der Wochentage

$$\{\text{montag}, \text{dienstag}, \text{mittwoch}, \text{donnerstag}, \text{freitag}, \text{samstag}, \text{sonntag}\}$$

Wir wollen eine Funktion $f : A \to A$ definieren, die jedem Wochentag den darauffolgenden Wochentag zuordnet, also

$$\begin{aligned} f(\text{montag}) &= \text{dienstag} \\ f(\text{dienstag}) &= \text{mittwoch} \\ f(\text{mittwoch}) &= \text{donnerstag} \\ f(\text{donnerstag}) &= \text{freitag} \\ f(\text{freitag}) &= \text{samstag} \\ f(\text{samstag}) &= \text{sonntag} \\ f(\text{sonntag}) &= \text{montag} \end{aligned}$$

Darstellung einer Funktion durch eine Tabelle

Eine Funktion $f : A \to B$ auf einer endlichen Menge A kann dargestellt werden als Tabelle.

Beispiel (Fortsetzung)

t	$f(t)$
montag	dienstag
dienstag	mittwoch
mittwoch	donnerstag
donnerstag	freitag
freitag	samstag
samstag	sonntag
sonntag	montag

Darstellung einer Funktion in Prolog

- In Prolog kann man eine Funktion $f : A \to B$ nicht direkt als Funktion definieren. Stattdessen muss man ihren Graphen als zweistelliges Prädikat p definieren.
- Es gilt: $p(x, y) \iff f(x) = y$.
- Anstatt anzugeben, wie man den Funktionswert $y = f(x)$ berechnet, gibt man an, unter welchen Bedingungen $p(x, y)$ gilt.

Darstellung einer Funktion in Prolog

Beispiel (Fortsetzung und Anfang von `wochentag.pl`)

Wir definieren den Graphen der Funktion f als zweistelliges Prädikat `folgetag/2`, wobei für zwei Wochentage x und y gilt:

$$\texttt{folgetag}(x, y) \iff f(x) = y.$$

Die Definition des Prädikats `folgetag/2` in Prolog lautet dann

```
folgetag(montag,dienstag).
folgetag(dienstag,mittwoch).
folgetag(mittwoch,donnerstag).
folgetag(donnerstag,freitag).
folgetag(freitag,samstag).
folgetag(samstag,sonntag).
folgetag(sonntag,montag).
```

Berechnung des Funktionswerts in Prolog

Um $y = f(x)$ zu berechnen, gibt man in Prolog x vor und stellt eine Anfrage
?- $p(x, \mathtt{Y})$.

Beispiel (Fortsetzung)

Anfrage: Was ist der Folgetag des Mittwochs?

$$f(\mathtt{mittwoch}) = \mathtt{donnerstag}$$

In Prolog:

```
?- folgetag(mittwoch,Y).
Y = donnerstag
```

Eingabe- und Ausgabeargumente

- Sei $f : A \to B$ eine Funktion.
- Sei p der Graph von f.
- Dann ist $p \subseteq A \times B$ eine zweistellige Relation,
  also ein zweistelliges Prädikat.
- In $p(x, y)$ ist x das Eingabeargument und y das Ausgabeargument,
  d.h. zu einem gegebenen Wert x (Eingabewert)
  soll ein Wert $y = f(x)$ (Ausgabewert) berechnet werden.
- In der Dokumentation von Prologprogrammen und -systemen (nicht in den Klauseln!) werden Eingabeargumente überlicherweise mit einem vorangestellten +, Ausgabeargumente mit – gekennzeichnet.

Eingabe- und Ausgabeargumente

Beispiel (Kommentarzeile zu `wochentag.pl`)

```
% folgetag(+T1,-T2)  Auf Wochentag T1 folgt Wochentag T2.
folgetag(montag,dienstag).
folgetag(dienstag,mittwoch).
folgetag(mittwoch,donnerstag).
folgetag(donnerstag,freitag).
folgetag(freitag,samstag).
folgetag(samstag,sonntag).
folgetag(sonntag,montag).
```

In der Kommentarzeile ist

- T1 das Eingabeargument
- T2 das Ausgabeargument.

Komposition zweier Funktionen

Definition

- Seien A, B und C Mengen.
- Sei $f : A \to B$.
- Sei $g : B \to C$.
- Dann ist die Komposition (Zusammensetzung) von g und f die Funktion $g \circ f : A \to C$, die definiert ist durch

$$(g \circ f)(x) := g(f(x)) \qquad \text{für } x \in A.$$

Komposition zweier Funktionen

Beispiel

Für ein Kalenderprogramm ist es nützlich, jedem Wochentag t eine Kennzahl $k(t)$ zwischen 0 und 6 zuzuordnen:

$$\begin{aligned} k(\text{montag}) &= 0 \\ k(\text{dienstag}) &= 1 \\ k(\text{mittwoch}) &= 2 \\ k(\text{donnerstag}) &= 3 \\ k(\text{freitag}) &= 4 \\ k(\text{samstag}) &= 5 \\ k(\text{sonntag}) &= 6 \end{aligned}$$

Dann ist k eine Funktion von der Menge der Wochentage in die Menge $\{0, 1, 2, 3, 4, 5, 6\}$.

Komposition zweier Funktionen

Beispiel (Fortsetzung)

- Daher ist $k \circ f$ eine Funktion von der Menge der Wochentage in die Menge $\{0, 1, 2, 3, 4, 5, 6\}$.
- $(k \circ f)(t)$ ist die Kennzahl des auf den Wochentag t folgenden Wochentags.
- Beispiel:
  $(k \circ f)(\text{mittwoch}) = k(f(\text{mittwoch})) = k(\text{donnerstag}) = 3$

Komposition zweier Funktionen in Prolog

Zur Berechnung des Funktionswertes $z = (k \circ f)(x) = k(f(x))$ in Prolog ist folgendes zu beachten:

- Funktionen müssen als Prädikate für ihre Graphen dargestellt werden, hier z.B. $p/2$ für f und $q/2$ für k.
- Prädikate kann man nicht schachteln.
- Stattdessen muss man für das Zwischenergebnis $f(x)$ eine neue Variable, z.B. y einführen.
- Dann ist $y = f(x) \wedge z = k(y)$.
- Mit den Prädikaten p und q ausgedrückt: $p(x, y) \wedge q(y, z)$.
- Anfrage: `?-` $p(x, \mathtt{Y}),\ q(\mathtt{Y}, \mathtt{Z})$.
- Definition der Komposition $k \circ f$ als Prädikat r.
- $r(\mathtt{X}, \mathtt{Z})$`:-` $p(\mathtt{X}, \mathtt{Y}),\ q(\mathtt{Y}, \mathtt{Z})$.

Komposition zweier Funktionen in Prolog

Beispiel (Fortsetzung wochentag.pl)

```
% kennzahl(+T,-Z)   Z ist die Kennzahl des Wochentags T.
kennzahl(montag,0).
kennzahl(dienstag,1).
kennzahl(mittwoch,2).
kennzahl(donnerstag,3).
kennzahl(freitag,4).
kennzahl(samstag,5).
kennzahl(sonntag,6).

% ?- folgetag(mittwoch,Y), kennzahl(Y,Z).

% kennzahl_folgetag(+X,-Z)      Z ist die Kennzahl des
% auf den Wochentag X folgenden Wochentags.
kennzahl_folgetag(X,Z) :- folgetag(X,Y), kennzahl(Y,Z).
```

Injektive, surjektive und bijektive Funktionen

Definition

Eine Funktion $f : A \to B$ heißt

- injektiv, wenn für alle $x, y \in A$ gilt: $f(x) = f(y) \Rightarrow x = y$.
- surjektiv, wenn für alle $y \in B$ ein $x \in A$ existiert mit $f(x) = y$.
- bijektiv, wenn sie injektiv und surjektiv ist.

Bei einer injektiven Funktion f gilt $f(x) \neq f(y)$, wenn $x \neq y$.

Beispiele (Sei $A =$ Menge der Wochentage, $B = \{0, 1, 2, 3, 4, 5, 6\}$.)

- Obige Funktionen $f : A \to A$ und $k : A \to B$ sind bijektiv.
- k als Funktion von A nach $\mathbb{N}$ wäre injektiv, aber nicht surjektiv.
- Die Nachfolgerfunktion auf $\mathbb{N}$ ist injektiv, aber nicht surjektiv.
- $f : \mathbb{R} \to \mathbb{R}$ mit $f(x) := x^2$ ist weder injektiv noch surjektiv.
- $f : \{1, \ldots, 30\} \to A$ mit "$f(x) :=$ Wochentag des x-ten April 2020" ist surjektiv, aber nicht injektiv.

Umkehrfunktion

Definition

Die Umkehrfunktion einer bijektiven Funktion $f : A \to B$ ist die Funktion $f^{-1} : B \to A$, die definiert ist durch $f(f^{-1}(y)) := y$ für alle $y \in B$.

Für die Umkehrfunktion gilt

- $f^{-1}(f(x)) = x$ für alle $x \in A$.
- f^{-1} ist bijektiv.
- $(f^{-1})^{-1} = f$.
- $(y, x) \in \text{Graph}(f^{-1}) \iff (x, y) \in \text{Graph}(f)$

Umkehrfunktion

Beispiel

- Für ein Kalenderprogramm möchte man zu einem Datum (Jahr, Monat, Tag) den Wochentag berechnen.
- Dazu berechnet man zunächst die Anzahl der seit einem bestimmten Stichtag (mit Kennzahl 0) vergangenen Tage.
- Der Rest dieser Zahl modulo (d.h. bei Division durch) 7 ergibt die Kennzahl z des gesuchten Wochentags.
- Der gesuchte Wochentag ist dann $k^{-1}(z)$.

Umkehrfunktion in Prolog

- Sei $f : A \to B$ eine bijektive Funktion.
- Das zweistellige Prädikat $p/2$ sei der Graph von f.
- In $p(x, y)$ ist x das Eingabeargument und y das Ausgabeargument.
- In Prolog lässt sich dann oft (nicht immer) die Umkehrfunktion f^{-1} dadurch realisieren, dass man in $p(x, y)$ das y als Eingabeargument und das x als Ausgabeargument von p betrachtet.
- Ein Argument eines Prädikats, das Ein- oder Ausgabeargument sein kann, wird in der Dokumentation üblicherweise mit einem vorangestellten ? gekennzeichnet.

Umkehrfunktion in Prolog

Beispiel (`wochentag.pl`: Kommentarzeile zu `kennzahl/2` geändert)

```
% kennzahl(?T,?Z)   Z ist die Kennzahl des Wochentags T.
kennzahl(montag,0).
kennzahl(dienstag,1).
kennzahl(mittwoch,2).
kennzahl(donnerstag,3).
kennzahl(freitag,4).
kennzahl(samstag,5).
kennzahl(sonntag,6).
```

Anfrage

```
?- kennzahl(T,2).
T = mittwoch
```

liefert für `T` den Wochentag $k^{-1}(2) = \text{mittwoch}.$

Mehrstellige Funktionen

Definition

- Eine *n*-stellige Funktion ist eine Funktion $f : A_1 \times \cdots \times A_n \to B$ von einem kartesischen Produkt $A_1 \times \cdots \times A_n$ in eine Menge B.
- Statt $f((x_1, \ldots, x_n))$ schreibt man einfach $f(x_1, \ldots, x_n)$.
- Als Graph einer n-stelligen Funktion $f : A_1 \times \cdots \times A_n \to B$ bezeichnet man dann üblicherweise die Menge

$$\{(x_1, \ldots, x_n, f(x_1, \ldots, x_n)) \mid x_1 \in A_1 \wedge \cdots \wedge x_n \in A_n\}.$$

Dies ist eine Teilmenge von $A_1 \times \cdots \times A_n \times B$, also eine $(n+1)$-stellige Relation oder ein $(n+1)$-stelliges Prädikat.

Darstellung einer Funktion in Prolog

- In Prolog kann man eine n-stellige Funktion f nicht direkt als Funktion definieren. Stattdessen muss man ihren Graphen als $(n+1)$-stelliges Prädikat p definieren.
- Es gilt: $p(x_1, \ldots, x_n, y) \iff f(x_1, \ldots, x_n) = y$.
- Anstatt anzugeben, wie man den Funktionswert $y = f(x_1, \ldots, x_n)$ berechnet, gibt man an, unter welchen Bedingungen $p(x_1, \ldots, x_n, y)$ gilt.
- Um $y = f(x_1, \ldots, x_n)$ zu berechnen, gibt man in Prolog $x_1, \ldots, x_n$ vor und stellt eine Anfrage
  ?- $p(x_1, \ldots, x_n, \mathtt{Y})$.

Berechnung des Funktionswerts in Prolog

Beispiel (Fortsetzung `wochentag.pl`)

```
% tag(?Abstand,?T1,?T2)   Abstand ist naechster, voriger
% oder uebernaechster und Wochentag T2 ist gegenueber
% Wochentag T1 der naechste, vorige bzw. uebernaechste
tag(naechster,T1,T2) :- folgetag(T1,T2).
tag(voriger,T1,T2) :- folgetag(T2,T1).
tag(uebernaechster,T1,T2) :- folgetag(T1,T), folgetag(T,T2).
```

Die zweite Regel verwendet die Umkehrfunktion f^{-1}, die dritte Regel die Komposition $f \circ f$.

Anfrage: Was ist der übernächste Tag zum Mittwoch?

```
?- tag(uebernaechster,mittwoch,T).
T = freitag
```

Eingabe- und Ausgabeargumente

- Für $p = \text{Graph}(f)$ sind in $p(x_1, \ldots, x_n, y)$ die $x_1, \ldots, x_n$ die Eingabeargumente und y das Ausgabeargument.
- Ein Prologprädikat kann auch mehrere Eingabe- und mehrere Ausgabeargumente haben.
- In Prolog muss im Gegensatz zu Funktionen das Ausgabeargument nicht eindeutig bestimmt sein. Beispiel: `tag/3` im Aufrufmodus `-+-` (zweites Argument als Eingabeargument; erstes und drittes Argument als Ausgabeargumente):

  ```
  ?- tag(Abstand,mittwoch,T).
  ```

  liefert drei verschiedene Antworten.

Funktionen als Prädikate

Gegenüberstellung: Funktion versus Prädikat (Graph der Funktion).
In Prolog muss man Prädikate verwenden.

Funktionen (prozedurale und funktionale Programmierung	Prädikate (logische Programmierung
Funktion f	Prädikat p
$f(x_1, \ldots, x_m) = y$	$p(X_1, \ldots, X_m, Y_1, \ldots, Y_n)$
$x_1, \ldots, x_m$ Argumente	$X_1, \ldots, X_m$ Eingabeargumente
y Funktionswert	$Y_1, \ldots, Y_n$ Ausgabeargumente
Geschachtelter Funktionsaufruf $f(g(x))$	Zwei Aufrufe $q(X, Y), p(Y, Z)$
Umkehrfunktion nicht implementierbar	Aufrufmodus z.B. −+ statt +−

Weitere Unterschiede zwischen prozeduraler und logischer Programmierung

Prozedurale Programmierung	Logische Programmierung
Variablenwerte können überschrieben werden.	Instantiierte Variablen können ihren Wert nicht mehr ändern. Variablen sind single assignment.
Programmschleifen durch Rekursion oder Iteration.	Programmschleifen durch Rekursion. Der Compiler wandelt Tail Recursion (Endrekursion) in Iteration um.

Wahrheitswertfunktionen

- Eine n-stellige Wahrheitswertfunktion ist eine n-stellige Funktion $f : \mathbb{B}^n \to \mathbb{B}$ auf der Menge $\mathbb{B} = \{w, f\}$ der Wahrheitswerte (auch Boole'sche Werte genannt).
- Zu jeder aussagenlogischen Verknüpfung gibt es eine entsprechende Wahrheitswertfunktion.
- Diese wird mit dem gleichen Zeichen wie die Verknüpfung bezeichnet.

Wahrheitswerttabellen für Wahrheitswertfunktionen

a	$\neg a$
w	f
f	w

a	b	$a \wedge b$	$a \vee b$	$a \rightarrow b$	$a \leftrightarrow b$
w	w	w	w	w	w
w	f	f	w	f	f
f	w	f	w	w	f
f	f	f	f	w	w

Darstellung in Prolog: wahrheitswertfunktionen.pl

```
% nicht(A,B)   B = nicht A
nicht(w,f).
nicht(f,w).

% und(A,B,C)   C = A und B
und(w,w,w).
und(w,f,f).
und(f,w,f).
und(f,f,f).

% oder(A,B,C)   C = A oder B
oder(w,w,w).
oder(w,f,w).
oder(f,w,w).
oder(f,f,f).
```

wahrheitswertfunktionen.pl (Fortsetzung)

```
% folgt(A,B,C)   C = A folgt B
folgt(w,w,w).
folgt(w,f,f).
folgt(f,w,w).
folgt(f,f,w).

% gdw(A,B,C)   C = A gdw B   (genau dann, wenn)
gdw(w,w,w).
gdw(w,f,f).
gdw(f,w,f).
gdw(f,f,w).
```

Eigenschaften der Wahrheitswertfunktionen

Die Menge $\mathbb{B}$ mit den Wahrheitswertfunktionen $\neg$, $\wedge$ und $\vee$ bildet eine Boolesche Algebra:

- $a \wedge a = a$ (Idempotenz von $\wedge$)
- $a \vee a = a$ (Idempotenz von $\vee$)
- $a \wedge b = b \wedge a$ (Kommutativität von $\wedge$)
- $a \vee b = b \vee a$ (Kommutativität von $\vee$)
- $(a \wedge b) \wedge c = a \wedge (b \wedge c)$ (Assoziativität von $\wedge$)
- $(a \vee b) \vee c = a \vee (b \vee c)$ (Assoziativität von $\vee$)
- $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ (Distributivität von $\wedge$ über $\vee$)
- $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ (Distributivität von $\vee$ über $\wedge$)
- $a \wedge (a \vee b) = a$ (Absorptionsgesetz)
- $a \vee (a \wedge b) = a$ (Absorptionsgesetz)

Eigenschaften der Wahrheitswertfunktionen

- $\neg\neg a = a$ (doppelte Negation)
- $\neg(a \wedge b) = \neg a \vee \neg b$ (De Morgansches Gesetz)
- $\neg(a \vee b) = \neg a \wedge \neg b$ (De Morgansches Gesetz)
- $a \wedge w = a$
- $a \vee f = a$
- $a \wedge f = f$
- $a \vee w = w$
- $a \wedge \neg a = f$
- $a \vee \neg a = w$
- $\neg f = w$
- $\neg w = f$

Eigenschaften mit Prolog testen

Beispiel (Distributivität von $\wedge$ über $\vee$)

Das Distributivgesetz

$$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$$

soll mit Prolog überprüft werden.

Dazu darf es keine Wahrheitswerte a, b und c geben, für die $\neg(a \wedge (b \vee c)) = (a \wedge b) \vee (a \wedge c)$ ist:

```
?- oder(B,C,BoderC), und(A,BoderC,AundBoderC),
   und(A,B,AundB), und(A,C,AundC),
   oder(AundB,AundC,AundBoderAundC),
   nicht(AundBoderC,AundBoderAundC).
no
```