Logische und funktionale Programmierung

Vorlesung 1: Grundlagen der logischen Programmierung

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

6. Oktober 2017

Einführung

- Weshalb muss ein Informatiker mehrere Programmiersprachen lernen?
- Hier ein paar Gründe:
- Die Vertrautheit mit unterschiedlichen Konzepten von Programmiersprachen ermöglicht es, eigene Ideen bei der Entwicklung von Software besser auszudrücken.
- Das Hintergrundwissen über verschiedene Programmiersprachen ist nötig, um in konkreten Projekten jeweils die am besten geeignete Sprache auszuwählen.

- Wenn man bereits einige konzeptionell verschiedene Programmiersprachen erlernt hat, ist es sehr leicht, sich später weitere Programmiersprachen schnell anzueignen.
- Es ist auch die Aufgabe von Informatikern, neue Programmiersprachen zu entwerfen. Dies kann nur auf der Grundlage der bereits entwickelten Sprachen geschehen.

Imperativ vs. Deklarativ

Imperative Languages

- sequence of instructions, executed after each other

Declarative Languages

- specify *what* should be computed
- compiler determines *how* the computation works

- **Procedural Languages**
  - variables, assignments, control structures
- **Object-Oriented Languages**
  - objects and classes
  - ADT and inheritance
- **Functional Languages**
  - no side-effects
  - recursion
- **Logic Languages**
  - rules to define relations

Imperativ vs. Deklarativ

Imperative Languages

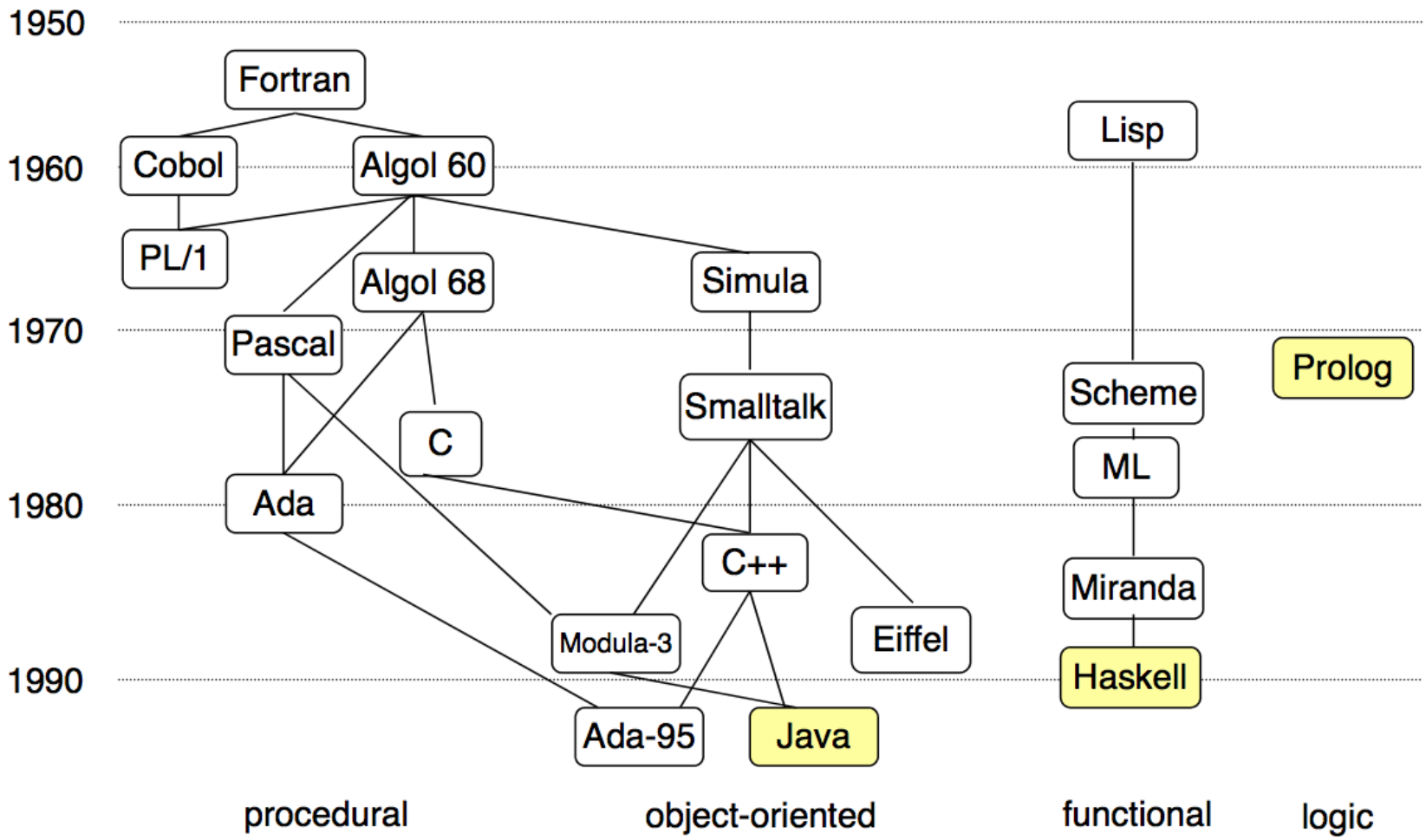
- sequence of instructions, executed after each other

Declarative Languages

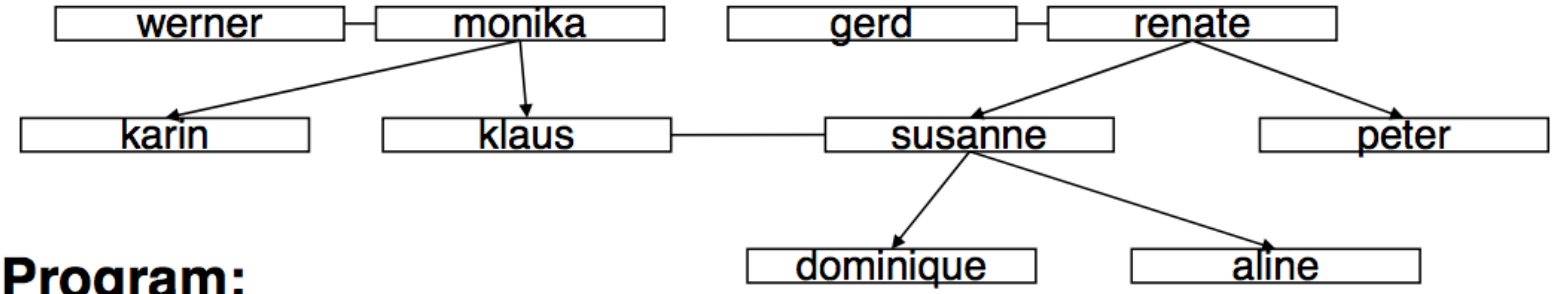
- specify *what* should be computed
- compiler determines *how* the computation works

- **Procedural Languages**
  - variables, assignments, control structures
- **Object-Oriented Languages**
  - objects and classes
  - ADT and inheritance
- **Functional Languages**
  - no side-effects
  - recursion
- **Logic Languages**
  - rules to define relations

Wichtige Programmiersprachen



Fakten und Anfragen



Program:

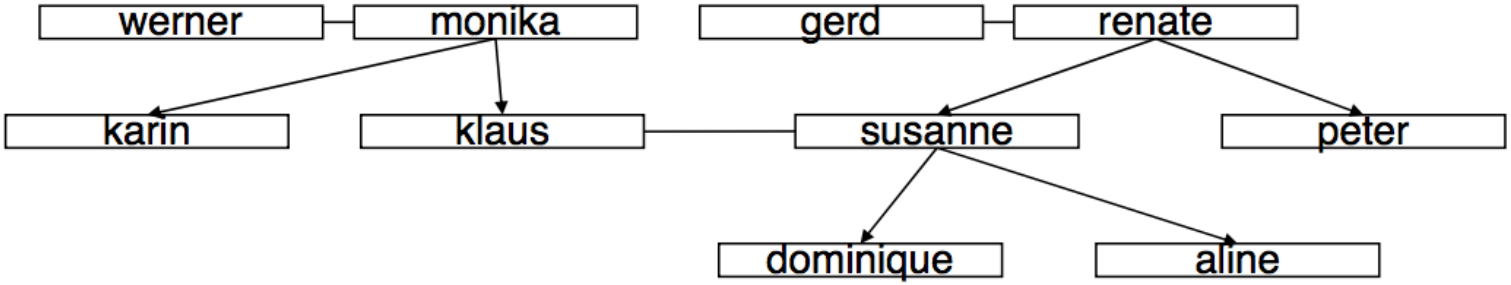
```
female(monika).                    male (werner).
        ...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
fatherOf(F,C)     :- married(F,W), motherOf(W,C).

parent(X, Y)     :- motherOf(X,Y).
parent(X, Y)     :- fatherOf(X,Y).

ancestor(V,X) :- parent(V,X).
ancestor(V,X) :- parent(V,Y), ancestor(Y,X).
```

```
?- ancestor(X, aline).
```

X = susanne ; X = klaus ;

X = monika ; X = renate; X = werner ; X = gerd

Variablen in Anfragen



Program:

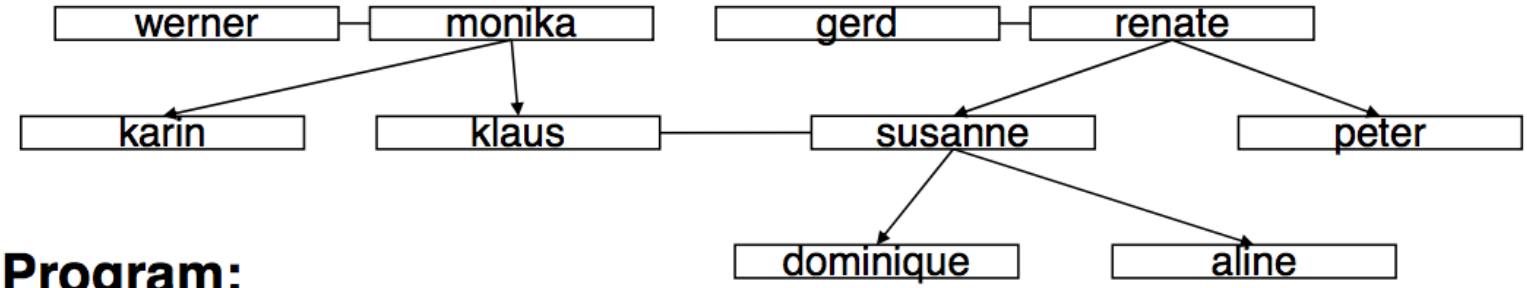
```
female(monika).
female(karin).
female(renate).
female(susanne).
female(aline).

married(werner, monika).
married(gerd, renate).
married(klaus, susanne).

human(X).

male(werner).
male(klaus).
male(gerd).
male(peter).
male(dominique).

motherOf(monika, karin).
motherOf(monika, klaus).
motherOf(renate, susanne).
motherOf(renate, peter).
motherOf(susanne, aline).
motherOf(susanne, dominique).
```

```
?- male(gerd).
true
```

```
?- married(gerd, monika).
false
```

```
?- human(gerd).
true
```

Kombination von Fragen



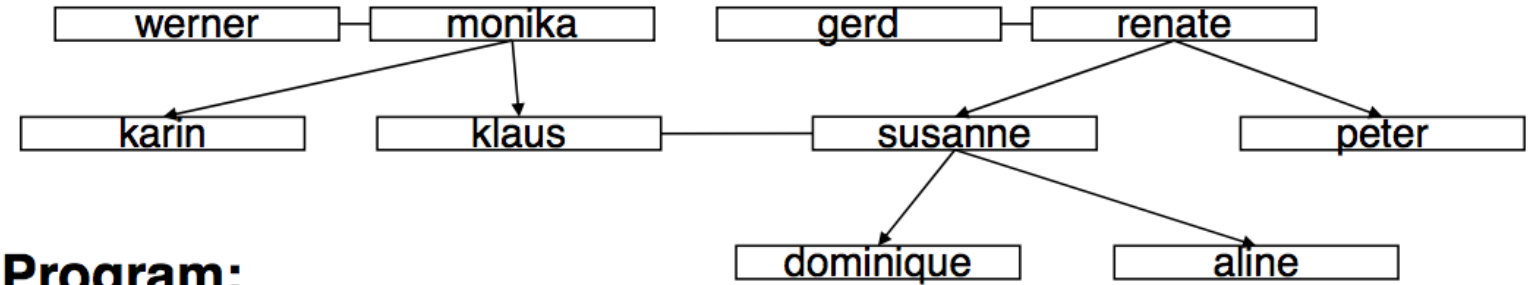
Program:

```
female(monika).
...
female(aline).

married(werner, monika).
...
married(klaus, susanne).

male(werner).
...
male(dominique).

motherOf(monika, karin).
...
motherOf(susanne, dominique).
```

```
?- motherOf(X, susanne).

X = renate
```

```
?- motherOf(renate, Y).

Y = susanne ;

Y = peter
```

Regeln



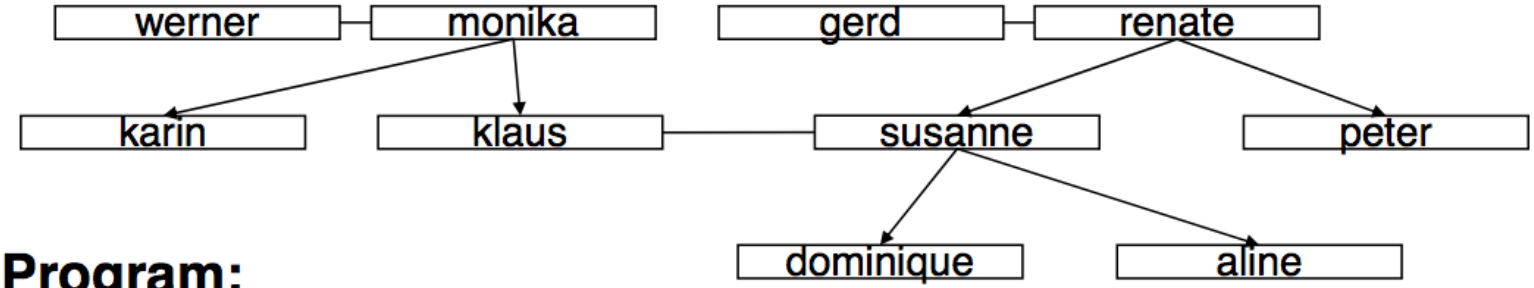
Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
```

```
?- married(gerd,W), motherOf(W,susanne).
W = renate
```

```
?- motherOf(Grandma,Mom), motherOf(Mom,aline).
Grandma = renate
Mom = susanne
```

Mehrere Regeln für ein Prädikat



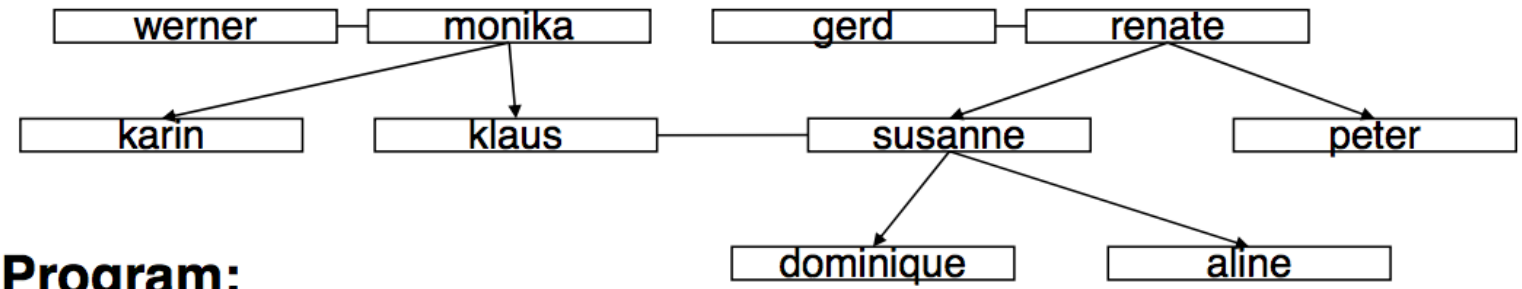
Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).

fatherOf(F,C)     :- married(F,W), motherOf(W,C).
```

```
?- fatherOf(gerd, susanne).
true
```

```
?- fatherOf(gerd, Y).
Y = susanne ;
Y = peter
```

Rekursive Regeln



Program:

```
female(monika).                    male(werner).
...                                ...
female(aline).                     male(dominique).
married(werner, monika).           motherOf(monika, karin).
...                                ...
married(klaus, susanne).           motherOf(susanne, dominique).
fatherOf(F,C)    :- married(F,W), motherOf(W,C).

parent(X, Y)     :- motherOf(X,Y).
parent(X, Y)     :- fatherOf(X,Y).
```

```
?- parent(X, susanne).

X = renate ;
X = gerd
```

Kennzeichen logischer Programme

1

(Rein) logische Programme besitzen keine Kontrollstrukturen zur Steuerung des Programmablaufs. Die Programme sind lediglich Sammlungen von Fakten und Regeln, die von oben nach unten (bzw. von links nach rechts) abgearbeitet werden.

Kennzeichen logischer Programme

2

Das logische Programmieren ist aus dem automatischen Beweisen entstanden und bei der Ausführung eines logischen Programms wird versucht, eine Anfrage zu beweisen. Dabei werden auch Lösungen für Variablen in der Anfrage berechnet. Dies bedeutet, dass bei einem logischen Programm Ein- und Ausgabevariablen nicht festliegen.

Kennzeichen logischer Programme

3

Logische Programme sind besonders gut für Anwendungen der künstlichen Intelligenz geeignet. Beispielsweise lassen sich damit sehr gut Expertensysteme realisieren, wobei die Regeln des Programms aus dem Wissen der Experten gebildet werden. Weitere Hauptanwendungsgebiete sind deduktive Datenbanken und das Rapid Prototyping.

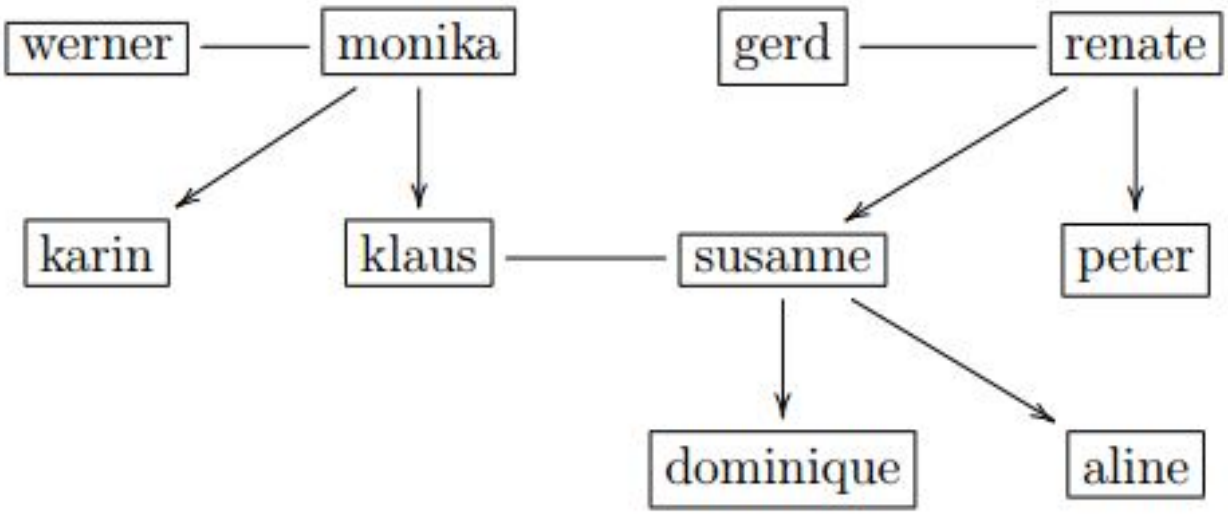

ICH HOFFE, ER HAT DAS BEGRIFFEN ...
VERMUTLICH SCHON ... ICH WERD'S
WOHL NIE MIT SICHERHEIT WISSEN ...

Wiederholung Prädikatenlogik

Signatur

Eine Signatur (Σ, Δ) ist ein Paar mit $\Sigma = \cup_{n \in \mathbb{N}} \Sigma_n$ und $\Delta = \cup_{n \in \mathbb{N}} \Delta_n$. Σ und Δ sind hierbei die Vereinigung von paarweise disjunkten Mengen Σ_n und Δ_n. Jedes $f \in \Sigma_n$ heißt n-stelliges Funktionssymbol und jedes $p \in \Delta_n$ heißt n-stelliges Prädikatensymbol. Die Elemente von Σ_0 werden auch Konstanten genannt. Wir verlangen hierbei stets $\Sigma_0 \neq \emptyset$.

BEISPIEL


werner
monika
gerd
renate
karin
klaus
susanne
peter
dominique
aline

Beispiel Signatur

Signatur (Σ, Δ) mit $\Sigma = \Sigma_0 \cup \Sigma_3$ und $\Delta = \Delta_1 \cup \Delta_2$.

- $\Sigma_0 = \mathbb{N} \cup \{\mathit{monika}, \mathit{karin}, \mathit{renate}, \mathit{susanne}, \mathit{aline}, \mathit{werner}, \mathit{klaus}, \mathit{gerd}, \mathit{peter}, \mathit{dominique}\}$
- $\Sigma_3 = \{\mathit{datum}\}$
- $\Delta_1 = \{\mathit{weiblich}, \mathit{maennlich}, \mathit{mensch}\}$
- $\Delta_2 = \{\mathit{verheiratet}, \mathit{mutterVon}, \mathit{vaterVon}, \mathit{elternteil}, \mathit{vorfahre}, \mathit{geboren}\}$

Terme

Definition

Sei (Σ, Δ) eine Signatur und V eine Menge von Variablen, so dass $V \cap \Sigma = \emptyset$ gilt. Dann bezeichnet $T(\Sigma, V)$ die Menge aller Terme (über Σ und V). Hierbei ist $T(\Sigma, V)$ die kleinste Menge mit

- $V \subseteq T(\Sigma, V)$ *und*
- $f(t_1, \ldots, t_n) \in T(\Sigma, V)$, *falls* $f \in \Sigma_n$ *und* $t_1, \ldots, t_n \in T(\Sigma, V)$ *für ein* $n \in \mathbb{N}$.

$T(\Sigma)$ steht für $T(\Sigma, \emptyset)$, d.h., für die Menge aller variablenfreien Terme (oder Grundterme).
Für einen Term t ist $V(t)$ die Menge aller Variablen in t.

Beispiel Terme

Wenn $V = \{X, Y, Z, Mama, Oma, \ldots\}$ ist, dann erhalten wir z. B. die folgenden Terme in $T(\Sigma, V)$: `monika, 42, datum(15, 10, 1966), X, datum(X, Oma, datum(Y, monika, 101)),...`

Formeln

Die Menge der atomaren Formeln *über* (Σ, Δ) *und* $\mathcal{V}$ *ist definiert als* $\mathcal{A}t(\Sigma, \Delta, \mathcal{V}) = \{p(t_1, \ldots, t_n) \mid p \in \Delta_n$ *für ein* n, $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})\}$.

$\mathcal{F}(\Sigma, \Delta, \mathcal{V})$ *bezeichnet die Menge aller* Formeln *über* (Σ, Δ) *und* $\mathcal{V}$. *Hierbei ist* $\mathcal{F}(\Sigma, \Delta, \mathcal{V})$ *die kleinste Menge mit*

- $\mathcal{A}t(\Sigma, \Delta, \mathcal{V}) \subseteq \mathcal{F}(\Sigma, \Delta, \mathcal{V})$
- *wenn* $\varphi \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$, *dann* $\neg\varphi \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$
- *wenn* $\varphi_1, \varphi_2 \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$, *dann* $(\varphi_1 \wedge \varphi_2), (\varphi_1 \vee \varphi_2), (\varphi_1 \rightarrow \varphi_2), (\varphi_1 \leftrightarrow \varphi_2) \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$
- *wenn* $X \in \mathcal{V}$ *und* $\varphi \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$, *dann* $(\forall X \ \varphi), (\exists X \ \varphi) \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$

Für eine Formel φ *bezeichnet* $\mathcal{V}(\varphi)$ *die Menge aller Variablen in* φ. *Eine Variable* X *ist* frei *in einer Formel* φ *gdw.*

- φ *ist eine atomare Formel und* $X \in \mathcal{V}(\varphi)$ *oder*
- $\varphi = \neg\varphi_1$ *und* X *ist frei in* φ_1 *oder*
- $\varphi = (\varphi_1 \cdot \varphi_2)$ *mit* $\cdot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ *und* X *ist frei in* φ_1 *oder in* φ_2 *oder*
- $\varphi = (Q\,Y\ \varphi_1)$ *mit* $Q \in \{\forall, \exists\}$, X *ist frei in* φ_1 *und* $X \neq Y$.

Eine Formel φ *heißt* geschlossen *gdw. es keine freie Variable in* φ *gibt. Eine Formel heißt* quantorfrei, *wenn sie nicht die Zeichen* $\forall$ *oder* $\exists$ *enthält.*

Bemerkung

$\mathcal{F}(\Sigma, \Delta, \mathcal{V})$ ist also eine formale Sprache, in der mathematische Sachverhalte, aber auch Fakten des täglichen Lebens formuliert werden können. Im Folgenden lassen wir überflüssige Klammern meist weg (d.h., wir schreiben "$\forall X$ $\mathsf{mensch}(X)$" statt "$(\forall X$ $\mathsf{mensch}(X))$").

BEISPIEL

$$
\begin{aligned}
\text{weiblich}(\text{monika}) &\in \mathcal{A}t(\Sigma, \Delta, \mathcal{V}) \\
\text{mutterVon}(X, \text{susanne}) &\in \mathcal{A}t(\Sigma, \Delta, \mathcal{V}) \\
\text{geboren}(\text{monika}, \text{datum}(15, 10, 1966)) &\in \mathcal{A}t(\Sigma, \Delta, \mathcal{V}) \\
\forall F\ (\text{verheiratet}(\text{gerd}, F) \wedge \text{mutterVon}(F, K)) &\in \mathcal{F}(\Sigma, \Delta, \mathcal{V}) \\
\text{verheiratet}(\text{gerd}, F) \wedge \neg(\forall F\ \text{mutterVon}(F, K)) &\in \mathcal{F}(\Sigma, \Delta, \mathcal{V}).
\end{aligned}
$$

In der 2. Zeile ist die Variabel X frei. Für die 4. Zeile gilt $\mathcal{V}(\phi) = \{F, K\}$ aber nur die Variable K ist frei. In der 5. Formel sind sowohl F als auch K frei.

Beispiel: Formeln

$$\forall X \qquad \text{mensch}(X)$$

$$\forall V, F, K \quad \text{verheiratet}(V, F) \wedge \text{mutterVon}(F, K) \rightarrow \text{vaterVon}(V, K)$$

$$\forall X, Y \qquad \text{mutterVon}(X, Y) \rightarrow \text{elternteil}(X, Y)$$
$$\forall X, Y \qquad \text{vaterVon}(X, Y) \rightarrow \text{elternteil}(X, Y)$$

$$\forall V, X \qquad \text{elternteil}(V, X) \rightarrow \text{vorfahre}(V, X)$$
$$\forall V, Y, X \quad \text{elternteil}(V, Y) \wedge \text{vorfahre}(Y, X) \rightarrow \text{vorfahre}(V, X)$$

Substitutionen

Definition

Eine Abbildung $\sigma : V \to T(\sigma, V)$ *heißt Substitution gdw.* $\sigma(X) \neq X$ *nur für endlich viele* $X \in V$ *gilt.* $DOM(\sigma) = \{X \in V \mid \sigma(X) \neq X\}$ *heißt der Domain von* σ*. Da* $DOM(\sigma)$ *endlich ist, ist eine Substitution* σ *als die endliche Menge von Paaren* $\{X/\sigma(X) \mid X \in DOM(\sigma)\}$ *darstellbar. Eine Substitution* σ *ist eine Grundsubstitution auf* $DOM(\sigma)$ *gdw.* $\mathcal{V}(\sigma(X)) = \emptyset$ *für alle* $X \in DOM(\sigma)$*. Eine Substitution* σ *ist eine Variablenumbenennung gdw.* σ *injektiv ist und* $\sigma(X) \in \mathcal{V}$ *für alle* $X \in \mathcal{V}$ *gilt.*

SUBSTITUTIONEN

Substitutionen werden homomorph zu Abbildungen $\sigma : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{T}(\Sigma, \mathcal{V})$ *erweitert, d.h.* $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$. *Analog kann man Substitutionen auch auf Formeln erweitern:*

(1) $\sigma(p(t_1, \ldots, t_n)) = p(\sigma(t_1), \ldots, \sigma(t_n))$

(2) $\sigma(\neg\varphi) = \neg\sigma(\varphi)$

(3) $\sigma(\varphi_1 \cdot \varphi_2) = \sigma(\varphi_1) \cdot \sigma(\varphi_2)$ *für* $\cdot \in \{\wedge, \vee, \to, \leftrightarrow\}$

(4) $\sigma(Q\,X\,\varphi) = Q\,X\,\sigma(\varphi)$, *für* $Q \in \{\forall, \exists\}$, *falls* $X \notin \mathcal{V}(\sigma(DOM(\sigma))) \cup DOM(\sigma)$

(5) $\sigma(Q\,X\,\varphi) = Q\,X'\,\sigma(\delta(\varphi))$, *für* $Q \in \{\forall, \exists\}$, *falls* $X \in \mathcal{V}(\sigma(DOM(\sigma))) \cup DOM(\sigma)$. *Hierbei ist* X' *eine neue Variable mit* $X' \notin \mathcal{V}(\sigma(DOM(\sigma))) \cup DOM(\sigma) \cup \mathcal{V}(\varphi)$ *und* $\delta = \{X/X'\}$.

Substitutionen

Die Bedingung in Fall (4) bedeutet, dass die Anwendung von σ auf die Formel $\forall X\ \varphi$ nur dann direkt möglich ist, wenn die durch den Quantor gebundene Variable X nicht ersetzt wird und wenn dadurch keine andere Variable in φ durch einen Term ersetzt wird, der X enthält. Anderenfalls muss (im Fall (5)) zunächst X in eine neue Variable X' umbenannt werden und anschließend wird die Substitution σ angewendet. Solch eine Umbenennung von quantifizierten Variablen bezeichnet man als gebundene Umbenennung.

Eine Instanz $\sigma(t)$ eines Terms t (bzw. eine Instanz $\sigma(\varphi)$ einer quantorfreien Formel φ) bezeichnet man als Grundinstanz, *falls $\mathcal{V}(\sigma(t)) = \varnothing$ (bzw. $\mathcal{V}(\sigma(\varphi)) = \varnothing$) ist.*

Beispiel

Ein Beispiel für eine Substitution wäre
$\Sigma = \{X/datum(X, Y, Z), Y/monika, Z/datum(Z, Z, Z)\}$.

$$\begin{aligned} \sigma(\mathsf{datum}(X, Y, Z)) &= \mathsf{datum}(\mathsf{datum}(X, Y, Z), \mathsf{monika}, \mathsf{datum}(Z, Z, Z)) \\ \sigma(\forall Y\ \mathsf{verheiratet}(X, Y)) &= \forall Y'\ \mathsf{verheiratet}(\mathsf{datum}(X, Y, Z), Y') \end{aligned}$$

Wir schreiben auch statt "$\sigma(\mathsf{datum}(X, Y, Z))$*" oft*

$$\mathsf{datum}(X, Y, Z)\ [X/\mathsf{datum}(X, Y, Z),\ Y/\mathsf{monika},\ Z/\mathsf{datum}(Z, Z, Z)].$$

Semantik der Prädikatenlogik

Für eine Signatur (Σ, Δ) ist eine Interpretation ein Tripel $I = (\mathcal{A}, \alpha, \beta)$. Die Menge A heit der Träger der Interpretation, wobei wir $\mathcal{A} \neq \emptyset$ verlangen. Weiter ist α eine Abbildung, die jedem Funktionssymbol $f \in \Sigma_n$ eine Funktion $\alpha_f : \mathcal{A}_n \to \mathcal{A}$ und jedem Prädikatensymbol $p \in \Delta_n$ eine Menge (bzw. Relation) $\alpha_p \subseteq A_n$ zuordnet. Die Funktion α_f bzw. die Relation α_p heißen die Deutung des Funktionssymbols f bzw. des Prädikatssymbols p unter der Interpretation I. Die Abbildung $\beta : \mathcal{V} \to \mathcal{A}$ heißt Variablenbelegung für die Interpretation I.

Semantik der Prädikatenlogik

Zu jeder Interpretation I erhält man eine Funktion $I : \mathcal{T}(\Sigma, \mathcal{V}) \to \mathcal{A}$ wie folgt:

$$\begin{aligned} I(X) &= \beta(X) \text{ für alle } X \in \mathcal{V} \\ I(f(t_1, \ldots, t_n)) &= \alpha_f(I(t_1), \ldots, I(t_n)) \text{ für alle } f \in \Sigma_n \text{ und } t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V}) \end{aligned}$$

Man nennt $I(t)$ die Interpretation des Terms t *unter der Interpretation I.*

Für $X \in \mathcal{V}$ und $\mathbf{a} \in \mathcal{A}$ ist $\beta[\![X/\mathbf{a}]\!]$ die Variablenbelegung mit $\beta[\![X/\mathbf{a}]\!](X) = \mathbf{a}$ und $\beta[\![X/\mathbf{a}]\!](Y) = \beta(Y)$ für alle $Y \in \mathcal{V}$ mit $Y \neq X$. $I[\![X/\mathbf{a}]\!]$ bezeichnet die Interpretation $(\mathcal{A}, \alpha, \beta[\![X/\mathbf{a}]\!])$.

Semantik der Prädikatenlogik

Eine Interpretation $I = (\mathcal{A}, \alpha, \beta)$ erfüllt *eine Formel* $\varphi \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$, *geschrieben "*$I \models \varphi$*", gdw.*

	φ	$= p(t_1, \ldots, t_n)$	*und* $(I(t_1), \ldots, I(t_n)) \in \alpha_p$
oder	φ	$= \neg\varphi_1$	*und* $I \not\models \varphi_1$
oder	φ	$= \varphi_1 \wedge \varphi_2$	*und* $I \models \varphi_1$ *und* $I \models \varphi_2$
oder	φ	$= \varphi_1 \vee \varphi_2$	*und* $I \models \varphi_1$ *oder* $I \models \varphi_2$
oder	φ	$= \varphi_1 \rightarrow \varphi_2$	*und falls* $I \models \varphi_1$, *dann auch* $I \models \varphi_2$
oder	φ	$= \varphi_1 \leftrightarrow \varphi_2$	*und* $I \models \varphi_1$ *gdw.* $I \models \varphi_2$
oder	φ	$= \forall X\ \varphi_1$	*und* $I[X/\mathbf{a}] \models \varphi_1$ *für alle* $\mathbf{a} \in \mathcal{A}$
oder	φ	$= \exists X\ \varphi_1$	*und* $I[X/\mathbf{a}] \models \varphi_1$ *für ein* $\mathbf{a} \in \mathcal{A}$

Semantik der Prädikatenlogik

- Eine Interpretation I heißt Modell von ϕ gdw. $I \models \phi$. I ist Modell einer Menge von Formeln Φ gdw. $I \models \phi$ für alle $\phi \in \Phi$ gilt. Zwei Formeln ϕ_1 und ϕ_2 heißen äquivalent gdw. für alle Interpretationen I gilt: $I \models \phi_1$ gdw. $I \models \phi_2$.
- Eine Formel bzw. eine Formelmenge heißt erfüllbar, wenn sie ein Modell besitzt, und unerfüllbar, wenn sie kein Modell besitzt. Sie heißt allgemeingültig, wenn jede Interpretation ein Modell ist.
- Eine Interpretation ohne Variablenbelegung $S = (\mathcal{A}, \alpha)$ wird als Struktur bezeichnet.
- Eine Struktur besitzt also eine Trägermenge, wobei den Funktionssymbolen aus Σ Funktionen auf dieser Trägermenge zugeordnet werden und den Prädikatssymbolen aus Δ Relationen auf der Trägermenge zugeordnet werden.

Semantik der Prädikatenlogik

Sofern wir nur geschlossene Formeln betrachten, so kann man daher Erfüllbarkeit *und* Modell *auch schon mit Hilfe von Strukturen definieren. Eine Struktur S erfüllt dann eine geschlossene Formel φ (d.h., "$S \models \varphi$" bzw. S ist Modell von φ) gdw. $I \models \varphi$ für eine Interpretation der Form $I = (\mathcal{A}, \alpha, \beta)$ gilt. Die Variablenbelegung β spielt hierbei keine Rolle, da φ ja keine freien Variablen enthält. Ebenso kann man bei Grundtermen t auch bereits $S(t)$ definieren (die Interpretation des Terms t unter der Struktur S).*

Interpretation $I = (\mathcal{A}, \alpha, \beta)$

Beispiel

$$
\begin{aligned}
\mathcal{A} &= \mathbb{N} \\
\alpha_n &= n \text{ für alle } n \in \mathbb{N} \\
\alpha_{\mathsf{monika}} &= 0 \\
\alpha_{\mathsf{karin}} &= 1 \\
\alpha_{\mathsf{renate}} &= 2 \\
&\vdots \\
\alpha_{\mathsf{datum}}(n_1, n_2, n_3) &= n_1 + n_2 + n_3 \text{ für alle } n_1, n_2, n_3 \in \mathbb{N} \\
\alpha_{\mathsf{weiblich}} &= \{n \mid n \text{ ist gerade}\} \\
\alpha_{\mathsf{maennlich}} &= \{n \mid n \text{ ist ungerade}\} \\
\alpha_{\mathsf{mensch}} &= \mathbb{N} \\
\alpha_{\mathsf{verheiratet}} &= \{(n, m) \mid n > m\} \\
&\vdots \\
\beta(X) &= 0 \\
\beta(Y) &= 1 \\
\beta(Z) &= 2 \\
&\vdots
\end{aligned}
$$

Dann ist $I(\mathsf{datum}(1, X, \mathsf{karin})) = \alpha_{\mathsf{datum}}(\alpha_1, \beta(X), \alpha_{\mathsf{karin}}) = 1 + 0 + 1 = 2$. *Somit gilt z.B.*

Interpretation $I = (\mathcal{A}, \alpha, \beta)$

Beispiel

Ebenso gilt

$$I \models \forall X\ \mathsf{weiblich}(\mathsf{datum}(X, X, \mathsf{monika})),$$

denn für alle $\mathbf{a} \in \mathcal{A}$ *ist* $I[\![X/\mathbf{a}]\!](\mathsf{datum}(X, X, \mathsf{monika})) = \mathbf{a} + \mathbf{a} + 0$ *eine gerade Zahl. Da die Formel* $\forall X\ \mathsf{weiblich}(\mathsf{datum}(X, X, \mathsf{monika}))$ *geschlossen ist, ist auch schon die Struktur* $S = (\mathcal{A}, \alpha)$ *ein Modell der Formel, d.h.*

$$S \models \forall X\ \mathsf{weiblich}(\mathsf{datum}(X, X, \mathsf{monika})).$$

Die obigen Formeln sind somit alle erfüllbar, aber sie sind nicht allgemeingültig, da sie nicht von jeder Interpretation erfüllt werden. Beispiele für allgemeingültige Formeln sind $\varphi \vee \neg\varphi$ *für alle Formeln* φ*. Beispiele für unerfüllbare Formeln sind* $\varphi \wedge \neg\varphi$ *für alle Formeln* φ*.*

Der Zusammenhang zwischen dem syntaktischen Begriff *Substitution* und dem semantischen Begriff *Variablenbelegung* wird durch folgendes Lemma beschrieben.

Substitutionslemma

Sei $I = (\mathcal{A}, \alpha, \beta)$ eine Interpretation für eine Signatur (Σ, Δ), sei $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$ eine Substitution. Dann gilt:

1. $I(\sigma(t)) = I[\![X_1/t_1, \ldots, X_n/t_n]\!](t)$ für alle $t \in \mathcal{T}(\Sigma, \mathcal{V})$.
2. $I \models \sigma(\phi)$ gdw. $I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!] \models \phi$ für alle $\pi \in \mathcal{F}(\Sigma, \Delta, \mathcal{V})$.

Beweis

Teil 1

Der Beweis wird durch strukturelle Induktion über den Aufbau des Terms t geführt. Im Induktionsanfang ist t entweder eine Variable oder eine Konstante (d.h., ein Funktionssymbol aus Σ_0).

Falls t eine Variable X_i ist, so gilt

$$I(\sigma(X_i)) = I(t_i) = I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](X_i).$$

Ist t eine Variable Y, die verschieden von allen $X_1, \ldots, X_n$ ist, so erhält man

$$I(\sigma(Y)) = I(Y) = \beta(Y) = I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](Y).$$

Hat t die Gestalt $f(s_1, \ldots, s_k)$ (wobei $k = 0$ möglich ist), so gilt

$$I(\sigma(t)) = I(\sigma(f(s_1, \ldots, s_k))) = \alpha_f(I(\sigma(s_1)), \ldots, I(\sigma(s_k))).$$

Aus der Induktionshypothese folgt $I(\sigma(s_i)) = I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](s_i)$. Man erhält daher

$$\begin{aligned}
& \alpha_f(I(\sigma(s_1)), \ldots, I(\sigma(s_k))) \\
= \; & \alpha_f(I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](s_1), \ldots, I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](s_k)) \\
= \; & I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](f(s_1, \ldots, s_k)) \\
= \; & I[\![X_1/I(t_1), \ldots, X_n/I(t_n)]\!](t).
\end{aligned}$$

Beweis

Teil 2

Wir verwenden eine strukturelle Induktion über den Aufbau der Formel φ.

Falls $\varphi = p(s_1, \ldots, s_m)$ ist, so gilt

$$\begin{aligned} I \models \sigma(\varphi) \quad & \text{gdw.} \quad I \models p(\sigma(s_1), \ldots, \sigma(s_m)) \\ & \text{gdw.} \quad (I(\sigma(s_1)), \ldots, I(\sigma(s_m))) \in \alpha_p \\ & \text{gdw.} \quad (I[\![X_1/I(s_1), \ldots, X_n/I(s_m)]\!](s_1), \ldots, \\ & \qquad I[\![X_1/I(s_1), \ldots, X_n/I(s_m)]\!](s_m)) \in \alpha_p \end{aligned}$$

nach Teil (a). Also erhält man

$$\begin{aligned} & (I[\![X_1/I(s_1), \ldots, X_n/I(s_m)]\!](s_1), \ldots, I[\![X_1/I(s_1), \ldots, X_n/I(s_m)]\!](s_m)) \in \alpha_p \\ \text{gdw.} \quad & I[\![X_1/I(s_1), \ldots, X_n/I(s_m)]\!] \models p(s_1, \ldots, s_m) \\ \text{gdw.} \quad & I[\![X_1/I(s_1), \ldots, X_n/I(s_m)]\!] \models \varphi. \end{aligned}$$

Die Fälle $\varphi = \neg\varphi_1$ oder $\varphi = \varphi_1 \cdot \varphi_2$ mit $\cdot \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$ sind sehr einfach. Betrachten wir noch den Fall $\varphi = \forall X\ \varphi_1$ (der Fall $\varphi = \exists X\ \varphi_1$ ist analog). O.B.d.A. betrachten wir nur den Fall $X \notin \mathcal{V}(\sigma(DOM(\sigma))) \cup DOM(\sigma)$. (Ansonsten wird die Formel zuerst in $\forall X'\ \varphi_1[X/X']$ überführt, die offensichtlich äquivalent zu φ ist.) Es gilt $\sigma(\varphi) = \forall X\ \sigma(\varphi_1)$ und

$$I \models \sigma(\varphi) \quad \text{gdw.} \quad I[\![X/\mathbf{a}]\!] \models \sigma(\varphi_1) \text{ für alle } \mathbf{a} \in \mathcal{A}.$$

Beweis

Teil 2

Sei $\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$ (wobei nach Voraussetzung X keine der Variablen X_i ist). Nach der Induktionshypothese ergibt sich

$$\begin{aligned} & I[X/\mathbf{a}] \models \sigma(\varphi_1) \text{ für alle } \mathbf{a} \in \mathcal{A} \\ \text{gdw.} \quad & I[X/\mathbf{a}][X_1/I(t_1), \ldots, X_n/I(t_n)] \models \varphi_1 \text{ für alle } \mathbf{a} \in \mathcal{A}. \end{aligned}$$

Da X verschieden von allen $X_1, \ldots, X_n$ ist, gilt $I[X/\mathbf{a}][X_1/I(t_1), \ldots, X_n/I(t_n)] = I[X_1/I(t_1), \ldots, X_n/I(t_n)][X/\mathbf{a}]$ und wir erhalten somit

$$\begin{aligned} & I[X_1/I(t_1), \ldots, X_n/I(t_n)][X/\mathbf{a}] \models \varphi_1 \text{ für alle } \mathbf{a} \in \mathcal{A} \\ \text{gdw.} \quad & I[X_1/I(t_1), \ldots, X_n/I(t_n)] \models \forall X\, \varphi_1 \\ \text{gdw.} \quad & I[X_1/I(t_1), \ldots, X_n/I(t_n)] \models \varphi. \end{aligned}$$

□

BEISPIEL

Wir betrachten die Interpretation von Seite 35, die Substitution $\sigma = \{X/\mathit{datum}(1, X, \mathsf{karin})\}$ und den Term $t = \mathsf{datum}(X, Y, Z)$. Dann gilt

$$\begin{aligned} I(\sigma(t)) &= I(\mathsf{datum}(\mathsf{datum}(1, X, \mathsf{karin}), Y, Z)) \\ &= \alpha_1 + \beta(X) + \alpha_{\mathsf{karin}} + \beta(Y) + \beta(Z) \\ &= 1 + 0 + 1 + 1 + 2 \\ &= 5. \end{aligned}$$

Beispiel

Ebenso gilt

$$\begin{aligned} I[X/I(\mathsf{datum}(1,X,\mathsf{karin}))](t) &= I[\![X/2]\!](\mathsf{datum}(X,Y,Z)) \\ &= 2+\beta(Y)+\beta(Z) \\ &= 2+1+2 \\ &= 5. \end{aligned}$$

Folgerbarkeit

Nun definieren wir, wann aus einer Formelmenge eine andere Formel folgt. Dies ist genau die Frage, die bei der Ausführung von Logikprogrammen untersucht wird.

Definition

Aus einer Formelmenge Φ folgt die Formel ϕ (abgekürzt $\Phi \models \phi$) gdw. für alle Interpretationen I mit $I \models \Phi$ gilt $I \models \phi$. Sofern Φ und ϕ keine freien Variablen enthalten, ist dies gleichbedeutend mit der Forderung, dass für alle Strukturen S mit $S \models \Phi$ gilt $S \models \phi$. Anstelle von $\emptyset \models \phi$ schreibt man meist $\models \phi$ (d.h., die Formel ϕ ist allgemeingültig).

Beispiel

Sei Φ die Formelmenge, die dem Verwandschaftslogikprogramm entspricht. Wenn man nun eine Anfrage wie

```
? - maennlich (gerd).
```

stellt, bedeutet das, dass man versucht $\Phi \models$`maennlich(gerd)` zu beweisen. Dies gilt natürlich im obigen Beispiel, da die Formel `maennlich(gerd)` in Φ enthalten ist. Ebenso gilt auch $\Phi \models$`mensch(gerd)`, da Φ die Formel $\forall X$mensch(X) enthält.

Beispiel

Die Anfrage
`?- mutterVon(X,susanne).`
bedeutet, dass man untersuchen will, ob
$\Phi \models \exists X \text{mutterVon}(X, \text{susanne})$ gilt. Wie erwähnt sind Variablen X im Logikprogramm allquantifiziert und in Anfragen existenzquantifiziert. Da Φ die Formel `mutterVon(renate, susanne)` enthält, gilt $\Phi \models \exists X \text{mutterVon}(X, \text{susanne})$. Hierbei muss die Variable X jeweils so belegt werden, wie die Deutung von `renate`.

SCHLUSSFOLGERUNG

Zur Ausführung von Logikprogrammen muss man also für eine Formelmenge Φ (das Logikprogramm) und eine Formel Φ (die Anfrage) herausfinden, ob $\Phi \models \phi$ gilt. Im Folgenden werden wir zeigen, wie diese Frage automatisch untersucht werden kann.


ICH HOFFE, ER HAT DAS BEGRIFFEN ...
VERMUTLICH SCHON ... ICH WERD'S
WOHL NIE MIT SICHERHEIT WISSEN ...

Resolutionskalkül

- Die Idee des Resolutionskalküls ist, das Folgerbarkeitsproblem $\Phi \models \phi$ auf ein Unerfüllbarkeitsproblem zu reduzieren und anschließend dieses Unerfüllbarkeitsproblem durch Resolution zu untersuchen.
- Das folgende Lemma zeigt, wie jedes Folgerbarkeitsproblem in ein Unerfüllbarkeitsproblem überführt werden kann.

Lemma 10: Überführung von Folgerbarkeits- in Unerfüllbarkeitsprobleme

Seien $\phi_1, \ldots, \phi_k, \phi$ Formeln einer Signatur Σ über ein Vokabular V. Dann gilt $\{\phi_1, \ldots, \phi_k\} \models \phi$ gdw. die Formel $\phi_1 \wedge \cdots \wedge \phi_k \wedge \neg\phi$ unerfüllbar ist.

Beweis

Es gilt	$\{\varphi_1, \ldots, \varphi_k\} \models \varphi$
gdw.	für alle Interpretationen I mit $I \models \{\varphi_1, \ldots, \varphi_k\}$ gilt $I \models \varphi$
gdw.	es gibt keine Interpretation I mit $I \models \{\varphi_1, \ldots, \varphi_k\}$ und $I \models \neg\varphi$
gdw.	$\varphi_1 \wedge \ldots \wedge \varphi_k \wedge \neg\varphi$ ist unerfüllbar

Beispiel

Um zu zeigen, dass in dem Logikprogramm mit dem Faktum
`mutterVon(renate,susanne).`
die Anfrage
`?- mutterVon(X,susanne).`
beweisbar ist, muss man zeigen, dass

$$\{\texttt{mutterVon}(\texttt{renate}, \texttt{susanne})\} \models \exists \texttt{X} \texttt{mutterVon}(\texttt{X}, \texttt{susanne})$$

gilt. Stattdessen kann man also jetzt die Ünerfullbarkeit der folgenden Formel nachweisen:

$$\texttt{mutterVon}(\texttt{renate}, \texttt{susanne}) \wedge \neg \exists \texttt{X} \texttt{mutterVon}(\texttt{X}, \texttt{susanne}).$$

- Das Problem, die Ünerfullbarkeit einer Menge von Formeln nachzuweisen (und damit auch das Problem, die Folgerbarkeit zu untersuchen) ist im allgemeinen unentscheidbar.
- Es existiert also kein automatisches Verfahren, dass stets terminiert und zu jeder Formel die Unerfüllbarkeit überprüfen kann.
- Allerdings ist die Unerfüllbarkeit (und damit auch die Folgerbarkeit) semi - entscheidbar.
- Es existiert also ein Algorithmus, der für jede unerfüllbare Formel die Unerfüllbarkeit in endlicher Zeit herausfindet, bei erfüllbaren Formeln jedoch eventuell nicht terminiert.
- Für das Folgerbarkeitsproblem bedeutet das, dass man $\Phi \models \phi$ mit dem Algorithmus stets in endlicher Zeit beweisen kann, falls es gilt.

- Falls aber $\Phi \models \phi$ nicht gilt, so terminiert der Algorithmus evtl. nicht.
- Da der Resolutionskalkül ein korrektes und vollständiges Verfahren zur Überprüfung der Unerfüllbarkeit einer Formel ist, handelt es sich damit um ein solches Semi-Entscheidungsverfahren.
- Falls die Formel unerfüllbar ist, so kann man dies durch eine Automatisierung des Resolutionskalküls auch nachweisen.
- Falls sie aber erfüllbar ist, so terminiert die Automatisierung des Resolutionskalküls eventuell nicht.

- Wir führen das Resolutionsprinzip zur Überprüfung der Unerfüllbarkeit einer Formel ϕ in vier Schritten ein.
- Die Grundidee hierbei ist, das Resolutionsprinzip für die Prädikatenlogik auf das Resolutionsprinzip der Aussagenlogik zu reduzieren.
- In der Aussagenlogik ist die Korrektheit und Vollständigkeit der Resolution leichter nachweisbar. (In der Aussagenlogik ist die Resolution außerdem ein Entscheidungsverfahren für die Unerfüllbarkeit.)

Schritt 1

- Zunächst zeigen wir, dass man ϕ zur Überprüfung der Unerfüllbarkeit in Skolem - Normalform überführen kann.
- Formeln in Skolem - Normalform sind geschlossen, sie besitzen keine Existenzquantoren mehr und sie enthalten Allquantoren nur ganz außen.
- Solche Formeln haben also die Gestalt $\forall X_1, \ldots, X_n \psi$, wobei ψ quantorfrei ist und keine Variablen außer $X_1, \ldots, X_n$ enthält.

Schritt 2

- Dann zeigen wir, dass man bei Formeln in Skolem - Normalform zur Untersuchung der Unerfüllbarkeit nicht alle Interpretationen betrachten muss, sondern sich auf sogenannte Herbrand-Interpretationen beschränken kann.
- Dies sind Interpretationen, bei denen Grundterme als *sich selbst* gedeutet werden.
- Hiermit ergibt sich bereits ein erstes automatisierbares Verfahren zur Untersuchung der Unerfüllbarkeit, das allerdings noch recht ineffizient ist.

Schritt 3

- Um ein effizienteres Verfahren zu erhalten, erweitern wir schließlich die aussagenlogische Resolution auf die Prädikatenlogik, indem wir hierzu das Verfahren der Unifikation benutzen.

Schritt 4

- Um die Effizienz weiter zu erhöhen, zeigen wir wie sich die Resolution weiter einschränken lässt.

Skolem - Normalform

- Unser Ziel besteht zunächst darin, Formeln in eine Normalform der Gestalt $\forall X_1, \ldots, X_n \psi$ zu überführen, wobei ψ quantorfrei ist und keine Variablen außer $X_1, \ldots, X_n$ enthält.
- Hierzu geht man in zwei Schritten vor.
- Zunächst überführt man die Formel in Pränex - Normalform.

Pränex-Normalform

- Eine Formel ϕ ist in Pränex - Normalform gdw. sie die Gestalt $Q_1 X_1 \ldots Q_n X_n \psi$ mit $Q_i \in \{\forall, \exists\}$ hat, wobei ψ quantorfrei ist.
- Der folgende Satz zeigt, dass (und wie) man jede Formel in eine äquivalente Formel in Pränex - Normalform überführen kann.

Satz 11: Überführung in Pränex-Normalform

Zu jeder Formel ϕ lässt sich automatisch eine Formel ϕ' in Pränex - Normalform konstruieren, so dass ϕ und ϕ' äquivalent sind.

Beweis

- Zunächst werden alle Teilformeln $\phi_1 \leftrightarrow \phi_2$ in ϕ durch $(\phi_1 \rightarrow \phi_2) \wedge (\phi_2 \rightarrow \phi_1)$ ersetzt.
- Anschließend werden alle Teilformeln $\phi_1 \rightarrow \phi_2$ in ϕ durch $\neg\phi_1 \vee \phi_2$ ersetzt.
- Die Überführung der verbleibenden Formel geschieht mit folgendem Algorithmus PRAENEX, dessen Terminierung und Korrektheit offensichtlich ist.
- Der Algorithmus bekommt als Eingabe eine beliebige Formel ϕ ohne die Junktoren $\leftrightarrow$ und $\rightarrow$ und liefert als Ausgabe eine dazu äquivalente Formel in Pränex - Normalform.

PRAENEX ALGORITHMUS

- Falls φ quantorfrei ist, so gib φ zurück.
- Falls $\varphi = \neg\varphi_1$ ist, so berechne $PRAENEX(\varphi_1) = Q_1 X_1 \ \ldots Q_n X_n \ \psi$. Liefere $\overline{Q_1} X_1 \ \ldots \overline{Q_n} X_n \ \neg\psi$ zurück, wobei $\overline{\forall} = \exists$ und $\overline{\exists} = \forall$.
- Falls $\varphi = \varphi_1 \cdot \varphi_2$ mit $\cdot \in \{\wedge, \vee\}$, so berechne $PRAENEX(\varphi_1) = Q_1 X_1 \ \ldots Q_n X_n \ \psi_1$ und $PRAENEX(\varphi_2) = R_1 Y_1 \ \ldots R_m Y_m \ \psi_2$. Durch Umbenennung gebundener Variablen erreichen wir, dass $X_1, \ldots, X_n$ nicht in $R_1 Y_1 \ \ldots R_m Y_m \ \psi_2$ auftreten und dass $Y_1, \ldots, Y_m$ nicht in $Q_1 X_1 \ \ldots Q_n X_n \ \psi_1$ auftreten. Gib nun die folgende Formel zurück:

$$Q_1 X_1 \ \ldots Q_n X_n \ R_1 Y_1 \ \ldots R_m Y_m \ \ (\psi_1 \cdot \psi_2)$$

- Falls $\varphi = Q X \ \varphi_1$ mit $Q \in \{\forall, \exists\}$ ist, so berechne die Formel $PRAENEX(\varphi_1) = Q_1 X_1 \ \ldots Q_n X_n \ \psi$. Durch Umbenennung gebundener Variablen erreichen wir, dass $X_1, \ldots, X_n$ verschieden von X sind. Dann gib $Q X \ Q_1 X_1 \ \ldots Q_n X_n \ \ \psi$ zurück.

Beispiel

- Wir betrachten die Überführung der folgenden Formel:

$$\neg\exists \mathtt{X}(\mathtt{verheiratet}(\mathtt{X},\mathtt{Y}) \vee \neg\exists \mathtt{Y}\mathtt{mutterVon}(\mathtt{X},\mathtt{Y}))$$

- Zunächst berechnen wir $\mathtt{PRAENEX}(\neg\exists \mathtt{Y}\mathtt{mutterVon}(\mathtt{X},\mathtt{Y})) = \forall \mathtt{Y}\neg\mathtt{mutterVon}(\mathtt{X},\mathtt{Y})$.
- Damit die gebundene Variable Y in dieser Teilformel von der freien Variablen Y in der Teilformel $\mathtt{verheiratet}(\mathtt{X},\mathtt{Y})$ verschieden ist, benennen wir erstere in Z um und erhalten somit $\forall \mathtt{Z}\neg\mathtt{mutterVon}(\mathtt{X},\mathtt{Z})$.
- Nun berechnen wir $\mathtt{PRAENEX}(\mathtt{verheiratet}(\mathtt{X},\mathtt{Y}) \vee \neg\exists \mathtt{Y}\mathtt{mutterVon}(\mathtt{X},\mathtt{Y}))$, was die folgende Formel ergibt:

$$\forall \mathtt{Z}(\mathtt{verheiratet}(\mathtt{X},\mathtt{Y}) \vee \neg\mathtt{mutterVon}(\mathtt{X},\mathtt{Z}))$$

Beispiel

- Schließlich ergibt sich
  $\texttt{PRAENEX}(\neg\exists \texttt{X}(\texttt{verheiratet}(\texttt{X},\texttt{Y}) \vee \neg\exists \texttt{Y}\texttt{mutterVon}(\texttt{X},\texttt{Y}))) =$
  $\forall \texttt{X}\exists \texttt{Z}\neg(\texttt{verheiratet}(\texttt{X},\texttt{Y}) \vee \neg\texttt{mutterVon}(\texttt{X},\texttt{Z}))$.

Skolem-Normalform

Eine Formel ϕ ist in Skolem - Normalform gdw. sie geschlossen ist und sie die Gestalt $\forall X_1, \ldots, X_n \psi$ hat, wobei ψ quantorfrei ist.

Skolem-Normalform

- Zur Überführung einer Formel in Skolem - Normalform wird die Formel zunächst in Pränex - Normalform überführt.
- Die verbleibende Überführung dient dazu, die freien Variablen und Existenzquantoren zu beseitigen.
- Anders als bei der Pränex - Normalform existiert nicht zu jeder Formel eine äquivalente Formel in Skolem - Normalform, denn offensichtlich gibt es z.B. keine Formel in Skolem - Normalform, die zu `weiblich(X)` oder zu $\exists$`Xweiblich(X)` äquivalent ist.
- Es gibt aber zu jeder Formel eine dazu erfüllbarkeitsäquivalente Formel in Skolem - Normalform.

Skolem-Normalform

- Die Idee der Überführung besteht darin, zuerst die Pränex - Normalform zu bilden, dann alle freien Variablen durch Existenzquantoren zu binden, und schließlich alle existenzquantifizierten Variablen mit Hilfe neuer Funktionssymbole zu beseitigen.

Satz 12: Überführung in Skolem - Normalform

Zu jeder Formel ϕ lässt sich automatisch eine Formel ϕ' in Skolem - Normalform konstruieren, so dass ϕ genau dann erfüllbar ist, wenn ϕ' erfüllbar ist.

Beweis

- Zunächst wird die Formel ϕ mit dem Verfahren aus Satz 11 in Pränex - Normalform überführt.
- Seien $X_1, \ldots, X_n$ die freien Variablen in der entstehenden Formel ϕ_1, die zu ϕ äquivalent ist.
- Dann wird ϕ_1 weiter überführt in die geschlossene Formel ϕ_2 der Gestalt $\exists X_1, \ldots, X_n \phi_1$, die zu ϕ_1 erfüllbarkeitsäquivalent ist:
- Aus $I \models \phi_1$ folgt $I[\![X_1/\beta(X_1), \ldots, X_n/\beta(X_n)]\!] \models \phi_1$ und damit offensichtlich auch $I \models \exists X_1, \ldots, X_n \phi_1$.
- Umgekehrt folgt aus $I \models \exists X_1, \ldots, X_n \phi_1$, dass es $a_1, \ldots, a_n$ gibt, so dass $I[\![X_1/a_1, \ldots, X_n/a_n]\!] \models \phi_1$ gilt.

Beweis

Die Formel φ_2 ist damit geschlossen, in Pränex–Normalform und zu φ erfüllbarkeitsäquivalent. Nun eliminieren wir die Existenzquantoren schrittweise von außen nach innen. Falls φ_2 die Formel $\forall X_1, \ldots, X_n \exists Y\ \psi$ ist, so ersetzen wir φ_2 durch die folgende Formel:

$$\forall X_1, \ldots, X_n\ \psi[Y/f(X_1, \ldots, X_n)]$$

Hierbei werden also alle Vorkommen von Y durch $f(X_1, \ldots, X_n)$ ersetzt, wobei f ein neues n-stelliges Funktionssymbol ist. Dieses Vorgehen wird solange wiederholt, bis keine Existenzquantoren mehr vorhanden sind. Die entstehende Formel ist erfüllbarkeitsäquivalent zu φ_2 und damit auch zu φ. Es gilt nämlich:

Beweis

$I \models \forall X_1, \ldots, X_n \; \psi[Y/f(X_1, \ldots, X_n)]$

$\curvearrowright$ $I[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!] \models \psi[Y/f(X_1, \ldots, X_n)]$ für alle $\mathbf{a_1}, \ldots, \mathbf{a_n} \in \mathcal{A}$

$\curvearrowright$ $I[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!][\![Y/I[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!](f(X_1, \ldots, X_n))]\!] \models \psi$
für alle $\mathbf{a_1}, \ldots, \mathbf{a_n} \in \mathcal{A}$, wegen Substitutionslemma

$\curvearrowright$ $I[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!] \models \exists Y \; \psi$ für alle $\mathbf{a_1}, \ldots, \mathbf{a_n} \in \mathcal{A}$

$\curvearrowright$ $I \models \forall X_1, \ldots, X_n \; \exists Y \; \psi$ für alle $\mathbf{a_1}, \ldots, \mathbf{a_n} \in \mathcal{A}$

Beweis

Aus $I \models \forall X_1, \ldots, X_n \exists Y\ \psi$ mit $I = (\mathcal{A}, \alpha, \beta)$ folgt, dass es für alle $\mathbf{a_1}, \ldots, \mathbf{a_n} \in \mathcal{A}$ ein $\mathbf{b} \in \mathcal{A}$ gibt, so dass $I[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}, Y/\mathbf{b}]\!] \models \psi$. Sei F die Funktion von $\mathcal{A}^n \to \mathcal{A}$, die jedem Tupel $(\mathbf{a_1}, \ldots, \mathbf{a_n})$ jeweils das entsprechende $\mathbf{b}$ zuordnet. Sei $I' = (\mathcal{A}, \alpha', \beta)$, wobei sich α' von α nur in der Deutung des neuen Funktionssymbols f unterscheidet. Hier definieren wir $\alpha'_f = F$. Dann erhalten wir $I' \models \forall X_1, \ldots, X_n\ \psi[Y/f(X_1, \ldots, X_n)]$. Der Grund ist, dass für alle $\mathbf{a_1}, \ldots, \mathbf{a_n} \in \mathcal{A}$ gilt:

Beweis

$$
\begin{array}{ll}
 & I[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}, Y/F(\mathbf{a_1}, \ldots, \mathbf{a_n})]\!] \models \psi \\
\text{gdw.} & I'[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}, Y/F(\mathbf{a_1}, \ldots, \mathbf{a_n})]\!] \models \psi \text{ da } f \text{ nicht in } \psi \text{ auftritt} \\
\text{gdw.} & I'[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!][\![Y/I'[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!](f(X_1, \ldots, X_n))]\!] \models \psi \\
\text{gdw.} & I'[\![X_1/\mathbf{a_1}, \ldots, X_n/\mathbf{a_n}]\!] \models \psi[Y/f(X_1, \ldots, X_n)]
\end{array}
$$

Beweis

- Da diese Aussage für alle $a_1, ..., a_n \in \mathcal{A}$ gilt, erhält man also $I' \models \forall X_1, ..., X_n \psi[Y/f(X_1, ..., X_n)]$.

Beispiel

- Wir betrachten die Überführung der Formel

$$\neg\exists \mathtt{X}(\mathtt{verheiratet}(\mathtt{X},\mathtt{Y}) \vee \neg\exists \mathtt{Y}\mathtt{mutterVon}(\mathtt{X},\mathtt{Y}))$$

  in Skolem - Normalform.
- Zuerst wird die Formel in Pränex - Normalform transformiert, was zu

$$\forall \mathtt{X}\exists \mathtt{Z}\neg(\mathtt{verheiratet}(\mathtt{X},\mathtt{Y}) \vee \neg\mathtt{mutterVon}(\mathtt{X},\mathtt{Z}))$$

  führt.
- Anschließend wird die freie Variable Y existenzquantifiziert, was

$$\exists \mathtt{Y}\forall \mathtt{X}\exists \mathtt{Z}\neg(\mathtt{verheiratet}(\mathtt{X},\mathtt{Y}) \vee \neg\mathtt{mutterVon}(\mathtt{X},\mathtt{Z}))$$

  ergibt.

Beispiel

- Um die Existenzquantoren zu beseitigen, wird nun zuerst die äußerste existenzquantifizierte Variable Y durch eine neue Konstante a ersetzt.
- Hierbei ist a nullstellig, weil es vor dem Existenzquantor von Y keine Allquantoren mehr gibt.
- Dies ergibt

$$\forall \mathtt{X} \exists \mathtt{Z} \neg(\mathtt{verheiratet}(\mathtt{X}, \mathtt{a}) \vee \neg \mathtt{mutterVon}(\mathtt{X}, \mathtt{Z})).$$

- Schließlich wird Z durch $f(X)$ ersetzt, wobei f ein neues einstelliges Funktionssymbol ist.
- Dies führt zu

$$\forall \mathtt{X} \neg(\mathtt{verheiratet}(\mathtt{X}, \mathtt{a}) \vee \neg \mathtt{mutterVon}(\mathtt{X}, \mathtt{f}(\mathtt{X}))).$$

Herbrand-Strukturen reloaded

- Das Problem bei der Untersuchung der Unerfüllbarkeit einer Formel ist, dass dazu alle Interpretationen bzw. Strukturen betrachtet werden mussen und der Träger kann hierbei eine vollkommen beliebige Menge sein.
- In diesem Abschnitt zeigen wir, dass man sich bei Formeln in Skolem - Normalform auf solche Strukturen beschränken kann, bei denen der Träger gerade aus den Grundtermen besteht und bei denen Funktionssymbole als *sich selbst* gedeutet werden.
- Solche Strukturen bezeichnet man als Herbrand - Strukturen (nach dem Logiker *Jacques Herbrand*).