Formale Techniken der Software-Entwicklung

Matthias Hölzl, Christian Kroiß

2. Juni 2014

Strukturen

Eine Struktur $\mathcal{A}$ zur Signatur $\sigma = (\mathit{Const}, \mathit{Fun}, \mathit{Pred}, |.|)$ (kurz σ-Struktur) besteht aus

- einer nichtleeren Menge A, dem *Träger* (oder der *Grundmenge*) der Struktur
- einem Element $c^{\mathcal{A}} \in A$ für jedes Konstantensymbol $c \in \mathit{Const}$
- einer Funktion $f^{\mathcal{A}} : A^n \to A$ für jedes n-stellige Funktionssymbol $f \in \mathit{Fun}$
- einer Relation $P^{\mathcal{A}} \subseteq A^n$ für jedes n-stellige Prädikatssymbol P

Eine Struktur heißt *endlich* (bzw. *unendlich*), wenn die Trägermenge endlich (bzw. unendlich) ist.

Modelle der Prädikatenlogik

Definition

Ein *Modell* $\mathcal{M}$ einer Sprache $\mathcal{L}$ ist ein Paar $(\mathcal{A}, w)$, bestehend aus einer $\mathcal{L}$-Struktur $\mathcal{A}$ (mit Träger A) und einer *Belegung* $w : Var \to A$.
Wir schreiben $[\![c]\!]_{\mathcal{M}}$ (oder $c^{\mathcal{M}}$, oder $[\![c]\!]$ falls $\mathcal{M}$ klar ist) für $c^{\mathcal{A}}$, entsprechend für Funktions- und Prädikatensymbole.

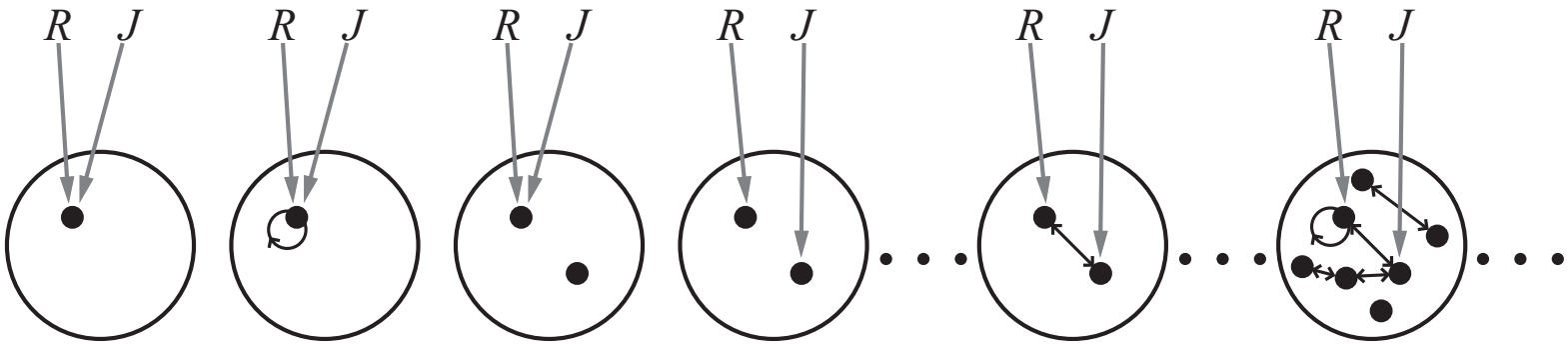
Modelle einer Sprache $\mathcal{L}$ nennt man auch $\mathcal{L}$-Modelle oder Interpretationen von $\mathcal{L}$. Den Träger von $\mathcal{A}$ nennt man auch Träger von $\mathcal{M}$.

Wir schreiben $M[x \mapsto a]$ für das Modell (A, w') mit

$$w'(y) = \begin{cases} w(y) & \text{für } y \neq x \\ a & \text{für } y = x \end{cases}$$

Strukturen

Strukturen für eine Sprache mit zwei Konstantensymbolen J, R und einem binären Prädikatensymbol P. Elemente für die P wahr ist sind durch Pfeile verbunden.



Beispiel: Struktur

Struktur für eine Sprache mit einer Konstante c, einem binären Funktionssymbol f, einem unären Prädikatensymbol P und einem binären Prädikatensymbol Q.

$Const/Fun$	$Const^{\mathcal{A}}/Fun^{\mathcal{A}}$	$Pred$	$Pred^{\mathcal{A}}$
c	2	$P(1)$	wahr
		$P(2)$	falsch
$f(1,1)$	1	$Q(1,1)$	falsch
$f(1,2)$	2	$Q(1,2)$	wahr
$f(2,1)$	2	$Q(2,1)$	falsch
$f(2,2)$	1	$Q(2,2)$	wahr

Semantik von Termen

Durch ein Modell $\mathcal{M}$ wird jedem $\mathcal{L}$-Term ein Element aus A zugeordnet:

$$\begin{aligned}
[\![x]\!]_{\mathcal{M}} &= w(x) \\
[\![c]\!]_{\mathcal{M}} &= c^{\mathcal{A}} \\
[\![f(t_1, \ldots, t_n)]\!]_{\mathcal{M}} &= f^{\mathcal{A}}([\![t_1]\!]_{\mathcal{M}}, \ldots, [\![t_n]\!]_{\mathcal{M}})
\end{aligned}$$

Erfüllungsrelation

Die Semantik von Formeln lässt sich durch die *Erfüllungsrelation* $\mathcal{M} \models \phi$ ($\mathcal{M}$ erfüllt ϕ oder $\mathcal{M}$ ist ein Modell von ϕ) beschreiben:

$$
\begin{aligned}
\mathcal{M} \models R(t_1, \ldots, t_n) &\iff R^{\mathcal{A}}([\![t_1]\!]_{\mathcal{M}}, \ldots, [\![t_n]\!]_{\mathcal{M}}) \\
\mathcal{M} \models t_1 = t_2 &\iff [\![t_1]\!]_{\mathcal{M}} = [\![t_2]\!]_{\mathcal{M}} \\
\mathcal{M} \models \neg\phi &\iff \mathcal{M} \models \phi \text{ ist falsch } (\mathcal{M} \not\models \phi) \\
\mathcal{M} \models \phi \wedge \psi &\iff \mathcal{M} \models \phi \text{ und } \mathcal{M} \models \psi \\
\mathcal{M} \models \phi \vee \psi &\iff \mathcal{M} \models \phi \text{ oder } \mathcal{M} \models \psi \\
\mathcal{M} \models \phi \Rightarrow \psi &\iff \mathcal{M} \not\models \phi \text{ oder } \mathcal{M} \models \psi \\
\mathcal{M} \models \phi \Leftrightarrow \psi &\iff (\mathcal{M} \models \phi \text{ und } \mathcal{M} \models \psi) \\
&\qquad \text{oder } (\mathcal{M} \not\models \phi \text{ und } \mathcal{M} \not\models \psi) \\
\mathcal{M} \models \forall x.\phi &\iff \mathcal{M}[x \mapsto a] \models \phi \text{ für alle } a \\
\mathcal{M} \models \exists x.\phi &\iff \text{es gibt } a \text{ mit } \mathcal{M}[x \mapsto a] \models \phi
\end{aligned}
$$

Endlichkeitssatz

Sei Φ eine (möglicherweise unendliche) Menge von Formeln.

Wir schreiben $\Phi \vdash_{\text{seq}} \phi$, wenn es eine Ableitung im Sequenzenkalkül gibt, deren Antezedens nur Formeln aus Φ enthält und deren Sukzedens ϕ enthält. Wir schreiben $\Phi \models \bot$ wenn Φ unerfüllbar ist.

Theorem (Endlichkeitssatz)

Wenn gilt $\Phi \vdash_{\text{seq}} \phi$, *so gibt es eine endliche Teilmenge* $\Phi_0 \subseteq \Phi$ *für die gilt* $\Phi_0 \vdash_{\text{seq}} \phi$.

Das ist klar, da Herleitungen endliche Bäume sind, und in jeder Sequenz nur endlich viele Terme vorkommen.

Korrektheit und Vollständigkeit

Sei Φ eine (möglicherweise unendliche) Menge von Formeln. Dann bedeuten

- Korrektheit eines Beweissystems

$$\Phi \left|\frac{}{\text{seq}}\right. \phi \quad \text{impliziert} \quad \Phi \models \phi$$

- Vollständigkeit eines Beweissystems

$$\Phi \models \phi \quad \text{impliziert} \quad \Phi \left|\frac{}{\text{seq}}\right. \phi$$

Gödelscher Vollständigkeitssatz

Theorem (Gödelscher Vollständigkeitssatz)

Sei $\Phi \subseteq \mathcal{L}$ eine Menge von Formeln und $\phi \in \mathcal{L}$ eine Formel. Dann gilt

$$\Phi \vdash_{\text{seq}} \phi \quad \textit{genau dann, wenn} \quad \Phi \models \phi$$

Wir zeigen in dieser Vorlesung die Vollständigkeit des Resolutionskalküls. Für den Sequenzenkalkül findet man den Beweis z.B. im Buch „Basic Proof Theory“ von Troelstra und Schwichtenberg.

Aus dem Vollständigkeitssatz und dem Endlichkeitssatz folgt sofort

Theorem (Endlichkeitssatz (für das Folgern))

Wenn $\Phi \models \phi$ gilt, so gibt es eine endliche Teilmenge Φ_0 für die $\Phi_0 \models \phi$ gilt.

Satz von Löwenheim-Skolem

Theorem (Satz von Löwenheim-Skolem)

Eine abzählbar konsistente Theorie hat immer auch ein abzählbares Modell.

Der Beweis dieses Satzes ergibt sich aus der im Beweis des Vollständigkeitssatzes verwendeten Konstruktion: man konstruiert dabei ein Termmodell, d.h., ein Modell dessen Elemente nur aus einer Menge von Konstanten und der Anwendung von Funktionssymbolen auf diese Konstanten bestehen. Da die Menge der dabei eingeführten Konstantensymbole abzählbar ist, ergibt sich die Aussage.

Aufzählbarkeit, Entscheidbarkeit

Im Gegensatz zur Aussagenlogik ist die Prädikatenlogik nicht entscheidbar. Es gilt:

- Die Sätze einer axiomatisierbaren Theorie T sind effektiv aufzählbar, d.h., es gibt einen Algorithmus der alle Sätze von T der Reihe nach erzeugt
- Eine vollständige axiomatisierbare Theorie T ist entscheidbar, d.h., es gibt einen Algorithmus, der für jede Formel ϕ aus $\mathcal{L}(T)$ in endlicher Zeit bestimmen kann ob $\phi \in T$ oder $\phi \notin T$ gilt
- Es gibt unentscheidbare axiomatisierbare Theorien T, d.h., für manche Theorien gibt es keinen Algorithmus der zu einer vorgegebenen Formel ϕ in endlicher Zeit bestimmen kann ob $\phi \in T$ oder $\phi \notin T$ gilt

Aussagenlogische Resolution

Falls L_i und M_j komplementär sind (d.h. $L_i \equiv \neg M_j$ oder $\neg L_i \equiv M_j$) gilt

$$\frac{L_1, \ldots, L_m \qquad M_1, \ldots, M_n}{L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_m, M_1 \ldots, M_{j-1}, M_{j+1}, M_n}$$

Diese Form der Ableitung heißt Resolution.
$L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_m, M_1 \ldots, M_{j-1}, M_{j+1}, \ldots, M_n$ heißt *Resolvente* von $L_1, \ldots, L_m$ und $M_1, \ldots, M_n$.

Falls $L_1, \ldots, L_m$ und $M_1, \ldots, M_n$ keine komplementären Literale haben, so kann die Resolutionsregel nicht angewendet werden.

Wir wollen im Folgenden die aussagenlogische Resolution auf die Prädikatenlogik erweitern.

Prädikatenlogische Resolution

Welche Probleme treten auf, wenn wir Resolution für prädikatenlogische Formeln definieren wollen?

- Quantoren: Resolution arbeitet auf Klauseln; wie gehen wir mit Quantoren um?
- Wann sind zwei prädikatenlogische Atome komplementär? Sind $P(x)$ und $\neg P(y)$ komplementär? Sind $P(x) und \neg P(a)$ komplementär? Sind $P(a) und \neg P(b)$ komplementär?

Antwort:

- Pränex-Normalform
- Skolemisierung
- Unifikation

Prädikatenlogische Resolution

- Pränex-Normalform: Alle Quantoren werden an den Anfang der Formel gezogen. Aus $(\forall x.P(x)) \wedge (\exists y.Q(y))$ wird $\forall x.\exists y.P(x) \wedge Q(y)$ oder $\exists y.\forall x.P(x) \wedge Q(y)$
- Umwandlung des quantorenfreien Teils der Formel in CNF
- Skolemisierung: Existenzquantoren werden durch neue Konstanten oder Funktionssymbole ersetzt. Aus $\exists y.\forall x.P(x) \wedge Q(y)$ wird, $\forall x.P(x) \wedge Q(c^s)$, aus $\forall x.\exists y.P(x) \wedge Q(y)$ wird $\forall x.P(x) \wedge Q(f^s(x))$
- Allquantoren am Anfang der Formel werden weggelassen. Aus $\forall x.P(x) \wedge Q(c^s)$ wird $P(x) \wedge Q(c^s)$
- Unifikation stellt fest wann $P(t_1, \ldots, t_n)$ und $P(t'_1, \ldots, t'_n)$ komplementär sind und ermöglicht die Berechnung der Resolvente zweier Klauseln

Normalisierung von Formeln

Genau wie in der Aussagenlogik gelten folgende semantischen Äquivalenzen:

$$\phi \Leftrightarrow \psi \cong \phi \Rightarrow \psi \wedge \psi \Rightarrow \phi$$
$$\phi \Rightarrow \psi \cong \neg\phi \vee \psi$$
$$\phi \vee \psi \cong \psi \vee \phi \qquad \phi \wedge \psi \cong \psi \wedge \phi$$
$$(\phi \vee \psi) \vee \xi \cong \phi \vee (\psi \vee \xi) \qquad (\phi \wedge \psi) \wedge \xi \cong \phi \wedge (\psi \wedge \xi)$$
$$\phi \vee (\psi \wedge \xi) \cong (\phi \vee \psi) \wedge (\phi \vee \xi) \qquad \phi \wedge (\psi \vee \xi) \cong (\phi \wedge \psi) \vee (\phi \wedge \xi)$$
$$\phi \vee \bot \cong \phi \qquad \phi \wedge \top \cong \phi$$
$$\phi \vee \top \cong \top \qquad \phi \wedge \bot \cong \bot$$
$$\phi \vee \neg\phi \cong \top \qquad \phi \wedge \neg\phi \cong \bot$$
$$\neg\neg\phi \cong \phi$$
$$\neg(\phi \vee \psi) \cong \neg\phi \wedge \neg\psi \qquad \neg(\phi \wedge \psi) \cong \neg\phi \vee \neg\psi$$

Damit lassen sich quantorenfreie Formeln in CNF (und DNF) umwandeln.

Normalisierung von Formeln

Zusätzlich gelten folgende Äquivalenzen für quantisierte Formeln, falls $x \notin \mathrm{fv}(\xi)$, $y \notin \mathrm{fv}(\phi)$, $Q, Q_i \in \{\forall, \exists\}$:

$$(Q\,x.\phi) \vee \xi \cong Q\,x.\phi \vee \xi \qquad (Q\,x.\phi) \wedge \xi \cong Q\,x.\phi \wedge \xi$$

$$\neg\forall x.\phi \cong \exists x.\neg\phi \qquad \neg\exists x.\phi \cong \forall x.\neg\phi$$

$$\forall x.\phi \wedge \forall x.\psi \cong \forall x.\phi \wedge \psi \qquad \exists x.\phi \vee \exists x.\psi \cong \exists x.\phi \vee \psi$$

$$(Q_1\,x.\phi) \wedge (Q_2\,y.\xi) \cong Q_1\,x.\,Q_2\,y.\phi \wedge \xi$$

$$(Q_1\,x.\phi) \vee (Q_2\,y.\xi) \cong Q_1\,x.\,Q_2\,y.\phi \vee \xi$$

Beachten Sie:

$$\forall x.\phi \vee \forall x.\psi \ncong \forall x.\phi \vee \psi$$

$$\exists x.\phi \wedge \exists x.\psi \ncong \exists x.\phi \wedge \psi$$

Pränex-Normalform

Definition

Eine prädikatenlogische Formel ϕ ist genau dann in *Pränex-Normalform*, wenn sie die Form

$$Q_1\, x_1 \ldots . Q_n\, x_n . \mu$$

hat, wobei $Q_i \in \{\forall, \exists\}$ und μ eine quantorenfreie Formel ist.
$Q_1\, x_1 \ldots . Q_n\, x_n$ heißt der *Präfix*, μ die *Matrix* der Formel.

Zu jeder Formel gibt es (mindestens) eine Pränex-Normalform.

Umwandlung in Pränex-Normalform

Durch den im Folgenden skizzierten Algorithmus kann man eine beliebige Formel in Pränex-Form umwandeln:

1. Eliminiere $\Leftrightarrow$ und $\Rightarrow$ durch
$$\phi \Leftrightarrow \psi \cong \phi \Rightarrow \psi \wedge \psi \Rightarrow \phi$$
$$\phi \Rightarrow \psi \cong \neg\phi \vee \psi$$
2. Verwende die Umformungen
$$\neg\neg\phi \cong \phi$$
$$\neg(\phi \vee \psi) \cong \neg\phi \wedge \neg\psi \qquad \neg(\phi \wedge \psi) \cong \neg\phi \vee \neg\psi$$
$$\neg\forall x.\phi \cong \exists x.\neg\phi \qquad \neg\exists x.\phi \cong \forall x.\neg\phi$$
um Negationen unmittelbar vor Atome zu schieben
3. Benenne gegebenenfalls gebundene Variablen um
4. Verwende die Quantorenregeln um alle Quantoren nach links zu schieben

CNF und Skolemisierung

Nachdem eine Formel in Pränex-Form gebracht wurde kann ihre Matrix, genau wie in der Aussagenlogik, in CNF umgewandelt werden. Als weiter Vorbereitung für die Anwendung der Resolution werden dann die Existenzquantoren durch sog. *Skolem-Konstanten* bzw. *Skolem-Funktionen* ersetzt.

Ein Verfahren zur Skolemisierung wird auf der nächsten Folie beschrieben.

Wenn eine Formel ϕ in Pränex-Form ist, ihr Präfix nur aus Allquantoren besteht, und ihre Matrix in CNF ist, dann sagt man ϕ sei in *Skolem-Normalform* oder *Standardform*. Üblicherweise lässt man den Präfix dann weg.

Skolemisierung

Sei ϕ in der Pränex-Form

$$Q_1\, x_1 \ldots . \, Q_n\, x_n . \mu$$

Sind alle Q_i Allquantoren, so ist ϕ bereits skolemisiert. Andernfalls sei Q_r der erste Existenzquantor.

Ist $r = 1$, dann ersetzen wir x_r durch eine neue Konstante c_r, löschen Q_r aus dem Präfix, und wenden das Verfahren rekursiv auf die neue Formel an.

Ist $r > 1$, dann ersetzen wir x_r durch eine neue Funktion $f(x_1, \ldots, x_{r-1})$, löschen Q_r aus dem Präfix und skolemisieren die so erhaltene Formel.

Man sieht leicht, dass die Matrix der skolemisierten Formel in konjunktiver Normalform ist wenn μ in CNF ist.

Beispiel

Bei der Skolemisierung der Formel

$$\exists x.\forall y.\forall z.\exists u.\forall v.\exists w.P(x, y, z, u, v, w)$$

steht x an der ersten Position im Präfix, daher ersetzen wir x durch eine Konstante c_x. Die Variablen u und w stehen hinter den allquantifizierten Variablen y und z, bzw. y, z und v, daher ersetzen wir sie durch Funktionen $f_u(y, z)$ und $f_w(y, z, v)$. Insgesamt erhalten wir

$$\forall y.\forall z.\forall v.P(c_x, y, z, f_u(y, z), v, f_w(y, z, v))$$

als Skolem-Normalform.

Beispiel

Die Formel

$$\forall x.\exists y.\exists z.(\neg P(x,y) \wedge Q(x,z)) \vee R(x,y,z)$$

ist bereits in Pränex-Form. Wir wandeln die Matrix in CNF um und erhalten

$$\forall x.\exists y.\exists z.(\neg P(x,y) \vee R(x,y,z)) \wedge (Q(x,z) \vee R(x,y,z))$$

Die existenzquantifizierten Variablen y und z folgen auf x, daher werden sie durch Skolem-Funktionen $f_y(x)$ und $f_z(x)$ ersetzt. Wir erhalten

$$\forall x.(\neg P(x, f_y(x)) \vee R(x, f_y(x), f_z(x))) \wedge (Q(x, f_z(x)) \vee R(x, f_y(x), f_z(x)))$$

oder, nach Streichen des Präfixes

$$(\neg P(x, f_y(x)) \vee R(x, f_y(x), f_z(x))) \wedge (Q(x, f_z(x)) \vee R(x, f_y(x), f_z(x)))$$

Erinnerung: Literale, Klauseln

Definition

Ein (prädikatenlogisches) Literal ist eine Primformel oder die Negation einer Primformel. Eine Klausel ist eine endliche Disjunktion von Literalen. Eine Klausel in der keine Variablen vorkommen nennt man *Grundklausel*; eine Klausel die genau ein Literal enthält nennt man *Einheitsklausel* (*Unit-Clause*); eine Klausel die gar kein Literal enthält nennt man die *leere Klausel*. Die leere Klausel ist äquivalent zu $\bot$.

Wir fassen eine Klausel oft als Menge der darin enthaltenen Literale auf und schreiben $\{\phi_1, \ldots, \phi_n\}$ für $\phi_1 \vee \cdots \vee \phi_n$. Die leere Klausel schreibt man oft als $\square$.

Mengen von Klauseln, CNF

Eine Menge von Klauseln fasst man als *Konjunktion* der Klauseln auf. Z.B. schreibt man

$$(\neg P(x, f_y(x)) \vee R(x, f_y(x), f_z(x))) \wedge (Q(x, f_z(x)) \vee R(x, f_y(x), f_z(x)))$$

als

$$\{\{\neg P(x, f_y(x)), R(x, f_y(x), f_z(x))\}, \{Q(x, f_z(x)), R(x, f_y(x), f_z(x))\}\}$$

Skolem-Normalform erhält Inkonsistenz

Proposition

Sei ϕ eine Formel und S die nach dem oben beschriebenen Verfahren erhaltene Skolem-Normalform zu ϕ. S ist genau dann inkonsistent, wenn ϕ inkonsistent ist.

Der wesentliche Schritt im Beweis ist zu zeigen, dass das Einführen von Skolem-Konstanten bzw. Skolem-Funktionen keinen Einfluss auf die Inkonsistenz einer Formel hat. Sei also $\phi = Q_1\, x_1 \ldots . Q_n\, x_n . \mu$ in Pränex-Form und sei Q_r der erste Existenzquantor. Sei ϕ' die bei der Skolemisierung eingeführte Funktion

$$Q_2\, x_2 \ldots . Q_n\, x_n . \mu[x_1 \mapsto c_1]$$

falls $r = 1$ bzw.

$$Q_1\, x_1 \ldots . Q_{r-1}\, x_{r-1} . Q_{r+1}\, x_{r+1} \ldots . Q_n\, x_n . \mu[x_r \mapsto f_r(x_1, \ldots, x_{r-1}]$$

falls $r > 1$.

Beweis

Sei ϕ inkonsistent. Um zu einem Widerspruch zu kommen nehmen wir an, ϕ' sei konsistent. Dann gibt es ein Modell $\mathcal{M}$, das ϕ' erfüllt, d.h., für alle $x_1, \ldots, x_{r-1}$ gibt es mindestens ein Element (nämlich $f_r(x_1, \ldots, x_{r-1})^{\mathcal{M}}$), so dass ϕ' in $\mathcal{M}$ wahr ist. Für diesen Wert ist aber, nach Definition von $\models$, auch ϕ in $\mathcal{M}$ erfüllt, im Widerspruch zur Inkonsistenz von ϕ. Damit ist also auch ϕ' inkonsistent.

Sei andererseits ϕ' inkonsistent. Wir nehmen an, ϕ wäre konsistent. Dann gibt es ein Modell $\mathcal{M}$ in dem es für alle Werte $a_1, \ldots, a_{r-1}$ von $x_1, \ldots, x_{r-1}$ einen Wert a_r für x_r gibt, so dass ϕ in $\mathcal{M}[\vec{x_r} \mapsto \vec{a_r}]$ erfüllt ist. Dann kann man aber $f^{\mathcal{M}}$ definieren durch $f^{\mathcal{M}}(a_1, \ldots, a_{r-1}) = a_r$ und erhält somit ein Modell von ϕ', im Widerspruch zur Inkonsistenz von ϕ'. Also ist mit ϕ' auch ϕ inkonsistent.

Vorsicht!

Die Skolem-Normalform ϕ_S einer Formel ϕ ist *nicht* äquivalent zu ϕ. Sei, zum Beispiel

$$\phi = \exists x.P(x)$$

und $\mathcal{A}$ die folgende Struktur

Träger	$\{1, 2\}$
Konstanten	$c^{\mathcal{A}} = 1$
Prädikate	$P(1)^{\mathcal{A}} =$ falsch, $P(2)^{\mathcal{A}} =$ wahr

Dann gilt $\mathcal{A} \models \phi$ aber $\mathcal{A} \not\models \phi_S$

Herbrand-Universum

Sei S eine Menge von Klauseln. Nach Definition ist S inkonsistent, wenn jedes Modell von S (d.h., jede Struktur, die die Konstanten-, Funktions- und Relationssymbole aus S interpretiert, zusammen mit einer Variablenbelegung) mindestens eine Klausel aus S als falsch interpretiert.

Oft wäre es hilfreich, wenn man diese Eigenschaft auf eine kleinere Menge von Modellen einschränken könnte. Das wird durch das *Herbrand-Universum* ermöglicht.

Herbrand-Universum

Definition

Sei S eine Menge von Klauseln und H_0 die Menge aller Konstanten, die in S vorkommen. Falls in S keine Konstante vorkommt, so sei $H_0 = \{a\}$ für ein beliebiges a. Wir definieren H_{i+1} rekursiv als Vereinigung von H_i und allen Termen, die sich aus den Funktionssymbolen in S und Termen aus H_i bilden lassen, d.h., allen Termen

$$f(t_1, \ldots, t_n)$$

für ein n-stelliges Funktionssymbol f und $t_j \in H_i$. Jedes H_i heißt *Konstantenmenge der i-ten Stufe* und

$$H = \bigcup_{i=1}^{\infty} H_i$$

heißt das *Herbrand-Universum* von S.

Beispiel

Sei $S = \{P(a), \neg P(x) \vee P(f(x))\}$. Dann sind

$$
\begin{aligned}
H_o &= \{a\} \\
H_1 &= \{a, f(a)\} \\
H_2 &= \{a, f(a), f(f(a))\} \\
&\vdots \\
H &= \{a, f(a), f(f(a)), f(f(f(a))), \dots\}
\end{aligned}
$$

Herbrand-Basis

Definition

Sei S eine Menge von Klauseln und H ihr Herbrand-Universum. Die Menge aller Terme der Form $P(t_1, \ldots, t_n)$ mit $t_j \in H$ (wobei die Stelligkeit von P gleich n ist) heißt die *Atommenge* oder *Herbrand-Basis* von S.
Sei C eine Klausel aus S. Jede Klausel, die man aus C durch Ersetzung aller Variablen durch Terme aus H erhält nennt man eine *Grundinstanz* von C.

Grundinstanzen sind also alle Klauseln der Form $C[\mathrm{fv}(C) \mapsto \vec{h}]$ mit $h_i \in H$.

Hinweis: Wir beschränken uns im Moment auf Logik ohne Gleichheit, da das die folgenden Definitionen und Beweise vereinfacht.

H-Strukturen und H-Modelle

Definition

Seien S eine Menge von Klauseln, H das Herbrand-Universum von S, $\mathcal{A}$ eine Struktur zur Signatur von S mit Träger H und $\mathcal{M}$ ein Modell über $\mathcal{A}$. $\mathcal{A}$ heißt H-Struktur von S und $\mathcal{M}$ H-Modell von S wenn gilt:

- $c^{\mathcal{M}} = c$ für alle Konstanten
- $f(h_1, \ldots, h_n)^{\mathcal{M}} = f(h_1, \ldots, h_n)$ für alle $h_i \in H$

H-Strukturen und H-Modell interpretiert also alle Konstanten und Funktionssymbole syntaktisch. Die Interpretation von Prädikaten und die möglichen Variablenbelegungen sind nicht festgelegt.

Sei $\{A_1, A_2, \ldots\}$ die Herbrand-Basis von S. Eine H-Struktur kann man angeben, indem man für jedes Atom A_i angibt, ob A_i oder $\neg A_i$ wahr ist. Schreibt man m_i für einen dieser beiden Terme, so kann man jede H-Struktur angeben als $\{m_1, m_2, \ldots\}$.

Beispiel

Sei $S = \{P(x) \vee Q(x), R(f(y))\}$. Das Herbrand-Universum H von S ist

$$H = \{a, f(a), f(f(a)), \dots\}$$

Die Herbrand-Basis (Atommenge) von S ist

$$B = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\}$$

H-Strukturen sind z.B.

$$\mathcal{A}_1 = \{P(a), Q(a), R(a), P(f(a)), Q(f(a)), R(f(a)), \dots\}$$
$$\mathcal{A}_2 = \{\neg P(a), \neg Q(a), \neg R(a), \neg P(f(a)), \neg Q(f(a)), \neg R(f(a)), \dots\}$$
$$\mathcal{A}_3 = \{P(a), \neg Q(a), R(a), \neg P(f(a)), \neg Q(f(a)), R(f(a)), \dots\}$$

Strukturen und H-Strukturen

Seien S eine Menge von Klauseln und $\mathcal{A}$ eine beliebige Struktur zu S (d.h., zur Signatur von S). Zum Beispiel, seien $S = \{P(x), Q(x, f(y, a))\}$, $A = \{1, 2\}$ und interpretiere $\mathcal{A}$ Terme und Atome folgendermaßen:

$Const/Fun$	$Const^{\mathcal{A}}/Fun^{\mathcal{A}}$	$Pred$	$Pred^{\mathcal{A}}$
a	2	$P(1)$	wahr
		$P(2)$	falsch
$f(1,1)$	1	$Q(1,1)$	falsch
$f(1,2)$	2	$Q(1,2)$	wahr
$f(2,1)$	2	$Q(2,1)$	falsch
$f(2,2)$	1	$Q(2,2)$	wahr

Es gibt immer eine H-Struktur $\mathcal{A}^*$, die $\mathcal{A}$ entspricht, d.h., so dass für jede Klausel C gilt $\mathcal{A} \models C$ genau dann, wenn $\mathcal{A}^* \models C$.

Strukturen und H-Strukturen

Die Herbrand-Basis zu $S = \{P(x), Q(x, f(y, a))\}$ ist

$$B = \{P(a), Q(a, a), P(f(a, a)), Q(a, f(a, a)), Q(f(a, a), a), \\ Q(f(a, a), f(a, a)), \dots\}$$

Wir ordnen jedem Element aus B den Wahrheitswert zu, den es in $\mathcal{A}$ bekommen würde:

$$\begin{aligned} P(a)^{\mathcal{A}^*} &= P(a)^{\mathcal{A}} = P^{\mathcal{A}}(2) = \text{falsch} \\ Q(a, a)^{\mathcal{A}^*} &= Q(a, a)^{\mathcal{A}} = Q^{\mathcal{A}}(2, 2) = \text{wahr} \\ P(f(a, a))^{\mathcal{A}^*} &= P(f(a, a))^{\mathcal{A}} = P^{\mathcal{A}}(f^{\mathcal{A}}(2, 2)) = P^{\mathcal{A}}(1) = \text{wahr} \\ Q(a, f(a, a))^{\mathcal{A}^*} &= Q(a, f(a, a))^{\mathcal{A}} = Q^{\mathcal{A}}(2, f^{\mathcal{A}}(2, 2)) = Q^{\mathcal{A}}(2, 1) = \text{falsch} \end{aligned}$$

Die H-Struktur zu $\mathcal{A}$ ist also

$$\mathcal{A}^* = \{\neg P(a), Q(a, a), P(f(a, a)), \neg Q(a, f(a, a)), \dots\}$$

Strukturen und H-Strukturen

Falls die Menge von Klauseln S keine Konstante enthält, wird a aus H_0 auf ein beliebiges Element von A abgebildet. Dann gibt es, falls A mehr als ein Element enthält, mehrere H-Strukturen zu $\mathcal{A}$.

Strukturen und H-Strukturen

Seien $S = \{P(x), Q(x, f(y, z))\}$, $A = \{1, 2\}$ und interpretiere $\mathcal{A}$ Terme und Atome folgendermaßen:

Fun	$Fun^{\mathcal{A}}$	$Pred$	$Pred^{\mathcal{A}}$
		$P(1)$	wahr
		$P(2)$	falsch
$f(1,1)$	1	$Q(1,1)$	falsch
$f(1,2)$	2	$Q(1,2)$	wahr
$f(2,1)$	2	$Q(2,1)$	falsch
$f(2,2)$	1	$Q(2,2)$	wahr

Strukturen und H-Strukturen

Die Herbrand-Basis zu $S = \{P(x), Q(x, f(y, z))\}$ ist wieder

$$B = \{P(a), Q(a, a), P(f(a, a)), Q(a, f(a, a)), Q(f(a, a), a), \\ Q(f(a, a), f(a, a)), \dots\}$$

In diesem Fall gibt es zwei H-Strukturen zu S, die der Wahl $a = 1$ oder $a = 2$ entsprechen:

$$\mathcal{A}_1^* = \{P(a), \neg Q(a, a), P(f(a, a)), \neg Q(a, f(a, a)), \dots\}$$
$$\mathcal{A}_2^* = \{\neg P(a), Q(a, a), P(f(a, a)), \neg Q(a, f(a, a)), \dots\}$$

Zugeordnete H-Strukturen

Definition

Seien S eine Menge von Klauseln und $\mathcal{A}$ eine Struktur zu S mit Träger $\mathcal{A}$. Eine zu $\mathcal{A}$ zugeordnete (oder $\mathcal{A}$ entsprechende) H-Struktur $\mathcal{A}^*$ ist eine Struktur mit Träger $\mathcal{A}$, die folgende Eigenschaft hat:
Für alle $n \in \mathbb{N}$, jedes n-stellige Prädikatenzeichen P und alle $h_1, \ldots, h_n$ aus dem Herbrand-Universum H und gilt

$$P(h_1, \ldots, h_n)^{\mathcal{A}} = P(h_1, \ldots, h_n)^{\mathcal{A}^*}$$

Falls in S keine Konstantensymbole vorkommen, so setzen wir $a^{\mathcal{A}} = c$ für ein beliebiges Element $c \in A$.

Offensichtlich gibt es zu jeder Menge von Klauseln S und jeder S-Struktur $\mathcal{A}$ eine zugeordnete H-Struktur.

Zugeordnete H-Strukturen

Lemma

Seien S eine Menge von Klauseln und $\mathcal{A}$ eine S-Struktur. Wenn gilt $\mathcal{A} \models S$, dann gilt auch $\mathcal{A}^* \models S$ für jede zugeordnete H-Struktur $\mathcal{A}^*$.

Satz

Eine Menge von Klauseln S ist genau dann unerfüllbar, wenn S für jede H-Struktur von S falsch ist.

Beweis: $\Rightarrow$ ist klar, denn wenn S unerfüllbar ist, dann ist S in jeder Struktur falsch, also auch in jeder H-Struktur

$\Leftarrow$: Sei S falsch in allen H-Strukturen. Wir nehmen an, S sei erfüllbar, habe also ein Modell $\mathcal{M}$ über einer Struktur $\mathcal{A}$. Dann wäre nach dem Lemma auch $\mathcal{A}^*$ ein Modell von S, d.h., S wäre wahr in einer H-Struktur. Die Annahme S sei erfüllbar ist also falsch, S ist unerfüllbar.

Satz von Herbrand

Mittels der zugeordneten Herbrand-Strukturen kann man folgende Aussage beweisen:

Satz (Herbrand)

Eine Menge von Klauseln S ist genau dann unerfüllbar, wenn es eine endliche, unerfüllbare Menge S' gibt, die aus Grundinstanzen von S besteht.

Der Satz von Herbrand ist die Rechtfertigung für automatisierte Beweisverfahren wie den DPLL-Algorithmus und die Grundlage für den Korrektheitsbeweis des Resolutionskalküls.

Zurück zur Resolution

Literale L und M heißen komplementär, wenn $L = \neg M$ oder $M = \neg L$ ist.

In der Aussagenlogik haben wir gesehen, dass die Resolvente zweier Klauseln L und M, die komplementäre Literale L_i und M_j enthalten, eine logische Konsequenz aus L und M ist.

Ein Resolutionsbeweis von einer Klausel C aus einer Menge von Klauseln S ist eine endliche Folge $C_1, \ldots, C_n = C$ von Klauseln, so dass jedes C_i in S enthalten oder eine Resolvente von C_j, C_k mit $j, k < i$ ist.

Wir haben mit der Skolem-Normalform eine Möglichkeit, prädikatenlogische Formeln in Klauseln umzuwandeln, müssen aber die Auswahl der komplementären Literale noch verallgemeinern.

Beispiel

Seien

$$C_1 = P(x) \vee Q(x)$$
$$C_2 = \neg P(f(x)) \vee R(x)$$

C_1 und C_2 haben keine komplementären Literale. Wenn wir aber auf C_1 die Substitution $[x \mapsto a]$ und auf C_2 die Substitution $[x \mapsto f(a)]$ anwenden, erhalten wir

$$C_1' = P(f(a)) \vee Q(f(a))$$
$$C_2' = \neg P(f(a)) \vee R(a)$$

und können mit einem Resolutionsschritt die Resolvente

$$Q(f(a)) \wedge R(a)$$

erhalten.

Unifikatoren

Definition (Komposition von Substitutionen)

Seien σ und θ Substitutionen. Die Komposition von θ und σ, $\theta \circ \sigma$ schreibt man auch $\sigma\theta$.

Bei der Komposition von Substitutionen wird also die links stehende Substitution zuerst ausgeführt. Das ist sinnvoll, weil die a Substitution von rechts angewendet wird: $(\phi\sigma)\theta = \phi(\sigma\theta) = \phi(\theta \circ \sigma)$.

Definition

Sei $E = \{E_1, \ldots E_k\}$ eine Menge von Ausdrücken (Termen oder Formeln). Eine Substitution σ heißt *Unifikator* von E wenn $E_1\sigma \equiv E_2\sigma \equiv \ldots \equiv E_k\sigma$ ist. E heißt *unifizierbar* wenn es einen Unifikator von E gibt. Ein Unifikator σ heißt *allgemeinster Unifikator* (*most general Unifyer*, *MGU*) von E, wenn sich jeder andere Unifikator θ in der Form $\theta = \sigma\eta$ schreiben lässt.

Unifikation

Wie soll sich ein Unifikationsalgorithmus für Ausdrücke verhalten?

- $f(t_1, \ldots, t_m)$ und $g(t'_1, \ldots, t'_n)$ sollen genau dann unifizierbar sein, wenn $f = g$ $m = n$ und t_i mit t'_i unifizierbar ist
- Genauso für Prädikate
- Variablen sind implizit allquantifiziert, x und y sollen also unifizierbar sein, wenn diese Umbenennung konsistent möglich ist
- Eine Variable steht für einen beliebigen anderen Term, also soll eine Variable x mit jedem Term t unifizierbar sein
- Außer t enthält x: $f(x)$ und x sind nicht (endlich) unifizierbar

Unifikationsalgorithmus

function UNIFY(x, y, σ)
    if $\sigma = \bot$ **then return** $\bot$
    else if $x = y$ **then return** σ
    else if Variable?(x) **then return** Unify-Var(x, y, σ)
    else if Variable?(y) **then return** Unify-Var(y, x, σ)
    else if Compound?(x) $\wedge$ Compound?(y) **then**
        return Unify-Compound(x, y, σ)
    else if List?(x) $\wedge$ List?(y) **then**
        return Unify($x.rest, y.rest,$ Unify($x.first, y.first, \sigma$))
    else
        return $\bot$
    end if
end function

Unifikationsalgorithmus

function UNIFY-VAR(var, x, σ)
    if $\langle var, val \rangle \in \sigma$ **then return** UNIFY(val, x, σ)
    else if $\langle x, val \rangle \in \sigma$ **then return** UNIFY(var, val, σ)
    else if OCCUR-CHECK(var, x) **then return** $\bot$
    else
        return σ with$[var \leftarrow x]$
    end if
end function

function UNIFY-COMPOUND(x, y, σ)
    if $x.op = y.op$ **then**
        return Unify(x.args, y.args)
    else
        return $\bot$
    end if
end function

Unifikationsalgorithmus

Proposition

Sind zwei Ausdrücke x und y unifizierbar, so gibt es einen allgemeinsten Unifikator σ.
Der Unifikationsalgorithmus $\text{Unify}(x, y, \epsilon)$ *terminiert und gibt einen allgemeinsten Unifikator für x und y zurück, falls ein Unifikator existiert. Falls kein Unifikator existiert gibt der Algorithmus $\perp$ zurück.*

Binäre Resolution

Seien $L = \{L_1, \ldots, L_m\}$ und $M = \{M_1, \ldots, M_n\}$ so, dass L und M keine gemeinsamen Variablen haben. Falls L_i und M_j mit MGU σ unifizierbar sind und $L_i\sigma$ und $M_j\sigma$ komplementär sind (d.h. $L_i\sigma \equiv \neg M_j\sigma$ oder $\neg L_i\sigma \equiv M_j\sigma$) gilt

$$\frac{L_1, \ldots, L_m \qquad M_1, \ldots, M_n}{(L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_m, M_1 \ldots, M_{j-1}, M_{j+1}, M_n)\sigma}$$

Diese Form der Ableitung heißt binäre Resolution.
$(L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_m, M_1 \ldots, M_{j-1}, M_{j+1}, \ldots, M_n)\sigma$ heißt *binäre Resolvente* von $L_1, \ldots, L_m$ und $M_1, \ldots, M_n$.

Binäre Resolution

Binäre Resolution für Prädikatenlogik ist keine vollständige Ableitungsregel. Man muss binäre Resolution entweder durch Resolution von Mengen unifizierbarer komplementärer Literale erweitern oder sog. Faktorisierung einführen.

Definition

Sei $L = \{L_1, \ldots L_m\}$ eine Menge von Literalen. Sind zwei oder mehrere L_i mit MGU σ unifizierbar, so heißt $L\sigma$ ein *Faktor* von L.

Resolution

Seien L und M Mengen von Literalen. N ist eine Resolvente von L und M, wenn die Resolvente einer Anwendung binärer Resolution auf

1. L und M,
2. L und einen Faktor von M,
3. einen Faktor von L und M oder
4. einen Faktor von L und einen Faktor von M

ist. Die Ableitung einer solchen Resolvente nennt man *(prädikatenlogische) Resolution*. Wir schreiben dann $L, M \left|\frac{}{\text{res}}\right. M$.

Vollständigkeit der Resolution

Satz

Prädikatenlogische Resolution ist vollständig und korrekt, d.h., es gilt

$$S \left|\frac{}{\text{res}}\right. M \quad \textit{genau dann, wenn} \quad S \models M$$

Der Beweis erfolgt durch den Satz von Herbrand bzw. die im Satz von Herbrand verwendete Konstruktion.

Beweisen durch Resolution: Snark

Terme:

$t \in \mathcal{T}$	$t' \in$ Snark
x	`?x, x`
c	`c`
$f(t_1, \ldots, t_n)$	`(f` $t'_1 \ldots t'_n$`)`

Formeln:

$\xi \in \mathcal{L}$	$\xi' \in$ Snark
$P(t_1, \ldots, t_n)$	`(p` $t'_1 \ldots t'_n$`)`
$t_1 = t_2$	`(=` t'_1 t'_2`)`
$\neg\phi$	`(not` ϕ'`)`
$\phi \wedge \psi$	`(and` ϕ' ψ'`)`
$\phi \vee \psi$	`(or` ϕ' ψ'`)`
$\phi \Rightarrow \psi$	`(implies` ϕ' ψ'`)`
$\phi \Leftrightarrow \psi$	`(iff` ϕ' ψ'`)`
$\forall x.\phi$	`(forall (?x)` ϕ'`), (forall (x)` ϕ'`)`
$\exists x.\phi$	`(exists (?x)` ϕ'`), (exists (x)` ϕ'`)`

Beispiel

$\xi \in \mathcal{L}$	$\xi' \in$ Snark
$M(\text{Plato})$	`(m plato)`
$M(x)$	`(m ?x)`

$\forall x.M(x) \Rightarrow S(x)$	`(forall (?x)`
	`  (implies (m ?x) (s ?x)))`
	oder
	`(forall (x)`
	`  (implies (m x) (s x)))`

$\forall x.M(x) \Rightarrow M(\text{mutter}(x))$	`(forall (x)`
	`  (implies (m x)`
	`  (m (mutter x))))`

Variablen können immer in der Form $?x$ geschrieben werden. Bei quantifizierten Variablen kann das führende Fragezeichen auch weggelassen werden.

Beweis mit Snark

```
(assert '(and (/= betty carol)
              (/= betty joe)
              (/= carol joe)))

(assert '(parent betty carol)
        :name 'betty-is-parent-of-carol)

(assert '(= (father carol) joe)
        :name 'joe-is-father-of-carol)

(assert '(parent (father ?c) ?c)
        :name 'father-is-parent)

(assert '(parent (mother ?c) ?c)
        :name 'mother-is-parent)
```

Beweis mit Snark

```
(assert '(implies (exists (?c) (= ?m (mother ?c)))
                  (is-female ?m))
        :name 'mother-is-female)

(assert '(iff (is-male ?x) (not (is-female ?x))))

(assert '(iff (parent ?p ?c)
              (or (= ?p (father ?c)) (= ?p (mother ?c)))))

(assert '(/= (father ?c) (mother ?c))
        :name 'father-is-not-mother)
```

Resolutionsbeweis mit Snark

```
SNARK-USER> (prove '(is-male joe))
; Running SNARK from /Users/tc/Prog/Lisp/Hacking/Iliad/Libraries/Snark/ in
(Refutation
(Row JOE-IS-FATHER-OF-CAROL
   (= (FATHER CAROL) JOE)
   ASSERTION)
(Row FATHER-IS-MALE
   (OR (NOT (= ?X (FATHER ?Y))) (IS-MALE ?X))
   ASSERTION)
(Row 16
   (NOT (IS-MALE JOE))
   NEGATED_CONJECTURE)
(Row 20
   (IS-MALE JOE)
   (RESOLVE FATHER-IS-MALE JOE-IS-FATHER-OF-CAROL))
(Row 21
   FALSE
   (REWRITE 16 20)))
:PROOF-FOUND
SNARK-USER>
```

Resolutionsbeweis mit Snark

```
SNARK-USER> (prove '(is-male joe))
; Running SNARK from /Users/tc/Prog/Lisp/Hacking/Iliad/Libraries/Snark/ in
(Refutation
(Row JOE-IS-FATHER-OF-CAROL
   (= (FATHER CAROL) JOE)
   ASSERTION)
(Row FATHER-IS-MALE
   (OR (NOT (= ?X (FATHER ?Y))) (IS-MALE ?X))
   ASSERTION)
(Row 16
   (NOT (IS-MALE JOE))
   NEGATED_CONJECTURE)
(Row 20
   (IS-MALE JOE)
   (RESOLVE FATHER-IS-MALE JOE-IS-FATHER-OF-CAROL))
(Row 21
   FALSE
   (REWRITE 16 20)))
:PROOF-FOUND
SNARK-USER>
```

Ausdrucksmächtigkeit der Prädikatenlogik

Viele Sachverhalte können in der Prädikatenlogik ausgedrückt werden:

- Mathematische Theorien, z.B. Gruppentheorie, Mengenlehre, ...
- Chemische Reaktionen
- Abläufe (durch Formalisierung von Situationen, Fluenten, etc.)

Aber: Manche Sachverhalte lassen sich nicht beschreiben, z.B. die transitive Hülle: Wir haben das Prädikat E für Eltern definiert. Das Prädikat V soll die Vorfahren repräsentieren, $V(x, y)$ genau dann, wenn y ein Vorfahre von x ist.

- $P(x, y) \Rightarrow V(x, y)$
- $(\exists z. P(x, z) \wedge V(z, y)) \Rightarrow V(x, y)$

Ist V eine korrekte Formalisierung des Begriffs „Vorfahre“?

Transitiver Abschluss

Frage: Ist V eine korrekte Formalisierung des Begriffs „Vorfahre“?

Antwort: Nein. V erzwingt zwar die Transitivität, stellt aber nicht sicher, dass das Modell von V minimal ist. Z.B. kann man V in einer Struktur $\mathcal{A}$ durch A^2 interpretieren. Was wir suchen ist die minimale Relation, für die die geforderte Abschlusseigenschaft gilt; *nur* die Elemente, die durch die Axiome gefordert sind sollen in V enthalten sein.

Frage: Wie können wir die Minimalität von V in der Prädikatenlogik ausdrücken?

Antwort: Gar nicht. Die Minimalität von V ist eine Eigenschaft, die über andere Pädikate quantisieren muss (oder eine ähnliche Konstruktion vornimmt). Das ist in der Prädikatenlogik nicht ausdrückbar:

$$\left[\forall Q : P(x,y) \Rightarrow Q(x,y) \wedge ((\exists z.P(x,z) \wedge Q(z,y)) \Rightarrow Q(x,y))\right] \Rightarrow V(x,y) \Rightarrow Q(x,y)$$