5.2 Typen für Prolog

Beispiel:

$$
\begin{array}{lcl}
\mathsf{nat}(X) & \leftarrow & X = 0 \\
\mathsf{nat}(X) & \leftarrow & X = s(Y), \mathsf{nat}(Y) \\
\mathsf{nat\_list}(X) & \leftarrow & X = [\,] \\
\mathsf{nat\_list}(X) & \leftarrow & X = [H|T], \mathsf{nat}(H), \mathsf{nat\_list}(T)
\end{array}
$$

Diskussion

- Ein Typ in Prolog ist eine einfach zu beschreibende Menge von Grundtermen.
- Darüber, was einfach heißt, gehen die Meinungen auseinander :-)
- Eine Möglichkeit sind (nicht-deterministische) endliche Baumautomaten oder normale Hornklauseln:

$$
\begin{array}{lcll}
\mathsf{nat\_list}([H|T]) & \leftarrow & \mathsf{nat}(H), \mathsf{nat\_list}(T) & \text{normal} \\
\mathsf{bin}(node(T,T)) & \leftarrow & \mathsf{bin}(T) & \text{nicht normal} \\
\mathsf{tree}(node(T_1,T_2)) & \leftarrow & \mathsf{tree}(T_1), \mathsf{tree}(T_2) & \text{normal}
\end{array}
$$

Vergleich:

Normale Klausel	Baumautomat
unäres Prädikat	Zustand
Klausel	Transition
Konstruktor im Kopf	Eingabesymbol
Rumpf	Vorbedingung

Allgemeine Form:

$$
\begin{array}{lll}
p(a(X_1, \ldots, X_k)) & \leftarrow & p_1(X_1), \ldots, p_k(X_k) \\
p(X) & \leftarrow & \\
p(b) & \leftarrow &
\end{array}
$$

Eigenschaften:

- Typen sind tatsächlich reguläre Baumsprachen ;-)
- Typen sind unter Durchschnitt abgeschlossen:

$$
\begin{array}{lll}
\langle p, q\rangle(a(X_1, \ldots, X_k)) & \leftarrow & \langle p_1, q_1\rangle(X_1), \ldots, \langle p_k, q_k\rangle(X_k) \qquad \text{falls} \\
p(a(X_1, \ldots, X_k)) & \leftarrow & p_1(X_1), \ldots, p_k(X_k) \qquad \text{und} \\
q(a(X_1, \ldots, X_k)) & \leftarrow & q_1(X_1), \ldots, q_k(X_k)
\end{array}
$$

- Typen sind auch unter Vereinigung abgeschlossen :-)
- Anfragen $p(X)$ und $p(t)$ können in polynomieller Zeit entschieden werden aber:
- Nur mit Tabellierung terminiert die Auswertung !
- Es sei denn, das Programm wäre topdown deterministisch ...

Beispiel: Topdown vs. Bottom-up

$$
\begin{array}{lll}
p(a(X_1, X_2)) & \leftarrow & p_1(X_1), p_2(X_2) \\
p(a(X_1, X_2)) & \leftarrow & p_2(X_1), p_1(X_2) \\
p_1(b) & \leftarrow & \\
p_2(c) & \leftarrow &
\end{array}
$$

... ist bottom-up, aber nicht topdown deterministisch.

Es gibt kein topdown deterministisches Programm für diesen Typ !

$\Longrightarrow$

Topdown deterministische Typen sind unter Durchschnitt, aber nicht unter Vereinigung abgeschlossen !!!

Zu einer Menge T von Bäumen definieren wir die Menge $\Pi(T)$ der Pfade in Bäumen aus T:

$$\Pi(T) = \bigcup\{\Pi(t) \mid t \in T\}$$

$$\begin{array}{lll} \Pi(b) & = & \{b\} \\ \Pi(a(t_1, \ldots, t_k)) & = & \{a_j w \mid w \in \Pi(t_j)\} \qquad (k > 0) \\ & & // \quad \text{für neue unäre Konstruktoren } a_j \end{array}$$

Beispiel

$$\begin{array}{lll} T & = & \{a(b,c), a(c,b)\} \\ \Pi(T) & = & \{a_1 b, a_2 c, a_1 c, a_2 b\} \end{array}$$

Aus einer Menge P von Pfaden lässt sich umgekehrt eine Menge $\Pi^-(P)$ von Bäumen zurück gewinnen:

$$\Pi^-(P) \quad = \quad \{t \mid \Pi(t) \subseteq P\}$$

Beispiel (Forts.):

$$\begin{array}{lll} P & = & \{a_1 b, a_2 c, a_1 c, a_2 b\} \\ \Pi^-(P) & = & \{a(b,b), a(b,c), a(c,b), a(c,c)\} \end{array}$$

Die Menge hat sich offenbar vergrößert !!

Satz:

Sei T eine reguläre Menge von Bäumen. Dann gilt:

- $\Pi(T)$ ist regulär :-)
- $T \subseteq \Pi^-(\Pi(T))$:-)
- $T = \Pi^-(\Pi(T))$ genau dann wenn T topdown deterministisch ist :-)
- $\Pi^-(\Pi(T))$ ist die kleinste Obermenge von T, die topdown deterministisch ist. :-)

Folgerung:

Sind wir an topdown deterministischen Typen interessiert, reicht es, die Menge der Pfade in Termen zu ermitteln !!!

Beispiel (Forts.):

$$
\begin{array}{lcl}
\mathsf{add}(X,Y,Z) & \leftarrow & X = 0, \mathsf{nat}(Y), Y = Z \\
\mathsf{add}(X,Y,Z) & \leftarrow & \mathsf{nat}(X), X = s(X'), Z = s(Z'), \mathsf{add}(X',Y,Z') \\
\mathsf{mult}(X,Y,Z) & \leftarrow & X = 0, \mathsf{nat}(Y), Z = 0 \\
\mathsf{mult}(X,Y,Z) & \leftarrow & \mathsf{nat}(X), X = s(X'), \mathsf{mult}(X',Y,Z'), \mathsf{add}(Z',Y,Z)
\end{array}
$$

Frage:

Welche der Überprüfungen sind erforderlich?

Idee:

- Approximiere die Semantik der Prädikate durch topdown-deterministische reguläre Baumsprachen !
- Alternativ: Approximiere die Menge der Pfade in der Semantik der Prädikate durch reguläre Wortsprachen !

Vereinfachung:

- Alle Prädikate sind unär.
- Dazu führen wir einfach für jede Stelligkeit k einen neuen Konstruktor: $()$ ein.
- Dann ersetzen wir:

$$p(t_1, \ldots, t_k) \qquad \text{durch:} \quad p(()(t_1, \ldots, t_k))$$

Semantik:

Sei $\mathcal{C}$ eine Menge von Klauseln.

Die Menge $[\![p]\!]_{\mathcal{C}}$ ist die Menge der Grundtermen s, für die $p(s)$ beweisbar ist :-)

$[\![p]\!]_{\mathcal{C}}$ (p Prädikat) is damit die kleinste Kollektion von Mengen, für die:

$$p(\sigma(t)) \in [\![p]\!]_{\mathcal{C}} \quad \text{wann immer} \quad \forall\, i.\ p_i(\sigma(t_i)) \in [\![p_i]\!]_{\mathcal{C}}$$

für eine Klausel $p(t) \leftarrow p_1(t_1), \ldots, p_n(t_n) \in \mathcal{C}$ und eine Grundsubstitution σ.

Approximation der Pfade:

Jede Klausel

$$p(t) \;\leftarrow\; p_1(t_1), \ldots, p_r(t_r)$$

approximieren wir durch die Menge der Klauseln:

$$p(w) \;\leftarrow\; \bigwedge\{p_1(w_1) \mid w_1 \in \Pi(t_1)\}, \ldots, \bigwedge\{p_r(w_r) \mid w_r \in \Pi(t_r)\}$$

$(w \in \Pi(t))$.

Beispiel:

$$\begin{array}{lll} \mathsf{add}(()(0, Y, Y)) & \leftarrow & \mathsf{nat}(Y) \\ \mathsf{add}(()(s(X), Y, s(Z))) & \leftarrow & \mathsf{add}(()(X, Y, Z)) \end{array}$$

liefert:

$$
\begin{array}{lcl}
\mathsf{add}(()_1\, 0) & \leftarrow & \mathsf{nat}(Y) \\
\mathsf{add}(()_2\, Y) & \leftarrow & \mathsf{nat}(Y) \\
\mathsf{add}(()_3\, Y) & \leftarrow & \mathsf{nat}(Y) \\[1ex]
\mathsf{add}(()_1\, s\, X) & \leftarrow & \mathsf{add}(()_1\, X), \mathsf{add}(()_2\, Y), \\
 & & \mathsf{add}(()_3\, Z) \\
\mathsf{add}(()_2\, Y) & \leftarrow & \mathsf{add}(()_1\, X), \mathsf{add}(()_2\, Y), \\
 & & \mathsf{add}(()_3\, Z) \\
\mathsf{add}(()_3\, s\, Z) & \leftarrow & \mathsf{add}(()_1\, X), \mathsf{add}(()_2\, Y), \\
 & & \mathsf{add}(()_3\, Z)
\end{array}
$$

Diskussion:

- Jedes Literal enthält maximal ein Variablen-Vorkommen.
- Die Literale $q_j(w_j Y)$ mit einer Variable Y, die nicht im Kopf vorkommt, stellen einen Test dar:

  Gibt es ein w mit $w_j w \in [\![q_j]\!]_{C^\sharp}$ für alle solchen j, können wir die Literale streichen.

  Gibt es kein solches w, können wir die Klausel streichen ...

... im Beispiel:

Die Literale:

$$\mathsf{add}(()_1\, X), \mathsf{add}(()_2\, Y), \mathsf{add}(()_3\, Z)$$

sind sämtlich erfüllbar :-)

Wir folgern:

$$
\begin{array}{lcl}
\mathsf{add}(()_1\, 0) & \leftarrow & \\
\mathsf{add}(()_2\, Y) & \leftarrow & \mathsf{nat}(Y) \\
\mathsf{add}(()_3\, Y) & \leftarrow & \mathsf{nat}(Y) \\
\\
\mathsf{add}(()_1\, s\, X) & \leftarrow & \mathsf{add}(()_1\, X) \\
\mathsf{add}(()_2\, Y) & \leftarrow & \mathsf{add}(()_2\, Y) \\
\mathsf{add}(()_3\, s\, Z) & \leftarrow & \mathsf{add}(()_3\, Z)
\end{array}
$$

Wir folgern:

$$
\begin{array}{lcl}
\mathsf{add}(()_1\, 0) & \leftarrow & \\
\mathsf{add}(()_2\, Y) & \leftarrow & \mathsf{nat}(Y) \\
\mathsf{add}(()_3\, Y) & \leftarrow & \mathsf{nat}(Y) \\
\\
\mathsf{add}(()_1\, s\, X) & \leftarrow & \mathsf{add}(()_1\, X) \\
\\
\\
\mathsf{add}(()_3\, s\, Z) & \leftarrow & \mathsf{add}(()_3\, Z)
\end{array}
$$

Wir vergewissern uns:

Satz

Sei $\quad \mathcal{C} \quad$ eine Menge von Klauseln.

Sei $\quad \mathcal{C}^\sharp \quad$ die zugehörige Menge von Klauseln für die Pfade.

Dann gilt f ür alle Prädikate p:

$$\Pi([\![p]\!]_{\mathcal{C}}) \;\subseteq\; [\![p]\!]_{\mathcal{C}^\sharp}$$

Beweis:

Induktion über die einzelnen Approximationen der jeweiligen Fixpunkte :-)

Eine Menge von Klauseln mit unären Prädikaten und Konstruktoren heißt Alternating Pushdown System (APS).

Theorem

- Jedes APS ist äquivalent zu einem einfachen APS der Form:

$$\begin{array}{lll} p(a\,X) & \leftarrow & p_1(X), \ldots, p_r(X) \\ p(X) & \leftarrow & \\ p(b) & \leftarrow & \end{array}$$

- Jedes APS ist äquivalent zu einem normalen APS der Form:

$$\begin{array}{lll} p(a\,X) & \leftarrow & p_1(X) \\ p(X) & \leftarrow & \\ p(b) & \leftarrow & \end{array}$$

Schritt 1: Beseitigung komplizierter Köpfe

Für $w = a^{(1)} \ldots a^{(m)} \quad (m > 1)$ ersetzen wir

$$p(w\,X) \quad \leftarrow \quad rhs \qquad \text{durch:}$$

$$\begin{array}{lcl}
p(a^{(1)}\,X) & \leftarrow & p_2(X) \\
p_2(a^{(2)}\,X) & \leftarrow & p_3(X) \\
 & \ldots & \\
p_{m-1}(a^{(m-1)}\,X) & \leftarrow & p_m(X) \\
p_m(a^{(m)}\,X) & \leftarrow & rhs \\
 & & // \quad p_j \text{ alle neu}
\end{array}$$

Schritt 1 (Forts.): Beseitigung komplizierter Köpfe

Für $w = a^{(1)} \ldots a^{(m)} b \quad (m > 0)$ ersetzen wir

$$p(w) \quad \leftarrow \quad rhs \qquad \text{durch:}$$

$$
\begin{array}{lcl}
p(a^{(1)}\, X) & \leftarrow & p_2(X) \\
p_2(a^{(2)}\, X) & \leftarrow & p_3(X) \\
 & \ldots & \\
p_{m-1}(a^{(m-1)}\, X) & \leftarrow & p_m(X) \\
p_m(a^{(m)}\, X) & \leftarrow & p_{m+1}(X) \\
p_{m+1}(b) & \leftarrow & rhs \\
 & & // \quad p_j \text{ alle neu}
\end{array}
$$

Beispiel:

$$\begin{array}{lcl} \mathsf{add}(()_1\, s\, X) & \leftarrow & \mathsf{add}(()_1\, X) \\ & & \text{ersetzen wir durch:} \\ \mathsf{add}(()_1\, X) & \leftarrow & \mathsf{add}_1(X) \\ \mathsf{add}_1(s\, X) & \leftarrow & \mathsf{add}(()_1\, X) \end{array}$$

Schritt 2: Splitting

Wir trennen unabängige Anteile der Voraussetzungen ab:

$$\begin{array}{lll} head & \leftarrow & rest,\; p_1(w_1\, X), \ldots, p_m(w_m\, X) \\ & & \qquad\qquad (X \text{ kommt nicht in } head, \; rest \text{ vor}) \end{array}$$

wird ersetzt durch:

$$\begin{array}{lll} head & \leftarrow & rest, q(()) \\ q(()) & \leftarrow & p_1(w_1\, X), \ldots, p_m(w_m\, X) \end{array}$$

für ein neues Prädikat q.

Schritt 3: Normalisierung

Wir fügen abgeleitete einfachere Klauseln hinzu:

$$\begin{array}{lll} \textit{head} & \leftarrow & p(a\,w), \textit{rest} \\ p(a\,X) & \leftarrow & p_1(X), \ldots, p_r(X) \end{array}$$

impliziert:

$$\begin{array}{lll} \textit{head} & \leftarrow & p_1(w), \ldots, p_r(w), \textit{rest} \end{array}$$

$$\begin{array}{lll} p(X) & \leftarrow & p_1(X), \ldots, p_m(X) \\ p_i(a\,X) & \leftarrow & p_{i1}(X), \ldots, p_{ir_i}(X) \end{array}$$

impliziert:

$$\begin{array}{lll} p(a\,X) & \leftarrow & p_{11}(X), \ldots, p_{mr_m}(X) \end{array}$$

Schritt 3 (Forts.): Normalisierung

$$\begin{array}{lll} head & \leftarrow & p(w), rest \\ p(X) & \leftarrow & \\ & & \text{impliziert:} \\ head & \leftarrow & rest \end{array}$$

$$\begin{array}{lll} head & \leftarrow & p(b), rest \\ p(b) & \leftarrow & \\ & & \text{impliziert:} \\ head & \leftarrow & rest \end{array}$$

$$\begin{array}{lll} p(()) & \leftarrow & p_1(X), \ldots, p_m(X) \\ p_i(a\,X) & \leftarrow & p_{i1}(X), \ldots, p_{ir_i}(X) \\ & & \text{impliziert:} \\ p(()) & \leftarrow & p_{11}(X), \ldots, p_{mr_m}(X) \end{array}$$

Beispiel:

$$\begin{array}{lll}
\mathsf{add}(()_1\, X) & \leftarrow & \mathsf{add}_0(X) \\
\mathsf{add}_0(0) & \leftarrow & \\
\mathsf{add}(()_1\, X) & \leftarrow & \mathsf{add}_1(X) \\
\mathsf{add}_1(s\, X) & \leftarrow & \mathsf{add}(()_1\, X)
\end{array}$$

... liefert an neuen Klauseln:

$$\begin{array}{lcl} \mathsf{add}_1(s\,X) & \leftarrow & \mathsf{add}_1(X) \\ \mathsf{add}_1(s\,X) & \leftarrow & \mathsf{add}_0(X) \end{array}$$