Programmier-Paradigmen

Tutorium – Gruppe 2 & 8

Henning Dieterichs

Unifikation & Prolog

Begriffe

- **Substitution**
  - Abbildung σ: $Vars \ \rightarrow Term$
  - Z.B. $[\ X_1 \Rightarrow f(X_2, t)\ ]$
  - Lässt sich rekursiv nach $Term \ \rightarrow Term$ fortsetzen
- **Unifikator von Gleichung** $\tau_1 = \tau_2$
  - Substitution σ, sodass $\sigma(\tau_1) = \ \sigma(\tau_2)$
  - Lässt sich auf Mengen fortsetzen
- **Allgemeinster Unifikator (mgu)**
  - Unifikator σ, sodass $\forall \text{Unifikator } \gamma\text{: } \exists \text{Substitution } \delta\text{: } \gamma = \delta \circ \sigma$

Aufgabe 1: Unifikation

1. Gegeben ist folgendes Term-Gleichungssystem (in Prolog-Notation):

$$\begin{aligned} \mathtt{X}_1 &= \mathtt{X}_2 \\ \mathtt{X}_2 &= \mathtt{X}_3 \end{aligned}$$

   Es seien außerdem folgende Substitutionen gegeben:

$$\begin{aligned} \sigma_1 &= [\mathtt{X}_1 \Rightarrow \mathtt{X}_2, \mathtt{X}_2 \Rightarrow \mathtt{X}_3] \\ \sigma_2 &= [\mathtt{X}_2 \Rightarrow \mathtt{X}_3] \circ [\mathtt{X}_1 \Rightarrow \mathtt{X}_2] \\ \sigma_3 &= [\mathtt{X}_1 \Rightarrow a, \mathtt{X}_2 \Rightarrow a, \mathtt{X}_3 \Rightarrow a] \end{aligned}$$

   Welche der Substitutionen ist

   - ein Unifikator für das gegebene Gleichungssystem?
   - ein allgemeinster Unfikator für das gegebene Gleichungssystem?

Aufgabe 1 : Unifikation

2. Geben Sie für das folgende Gleichungssystem einen allgemeinsten Unifikator an:

$$\mathtt{a}(\mathtt{t}_1, \mathtt{a}(\mathtt{X}_3, \mathtt{X}_4)) = \mathtt{a}(\mathtt{X}_1, \mathtt{X}_2)$$
$$\mathtt{X}_3 = \mathtt{t}_2$$
$$\mathtt{X}_4 = \mathtt{X}_1$$

Rechnen Sie den Unifikator vollständig aus, d.h. geben Sie ihn in der Form

$$[\mathtt{X}_1 \Rightarrow \ldots, \mathtt{X}_2 \Rightarrow \ldots, \mathtt{X}_3 \Rightarrow \ldots, \mathtt{X}_4 \Rightarrow \ldots]$$

an.

Unifikationsalgorithmus nach Robinson

Unifikationsalgorithmus: $\mathtt{unify}(C) =$

$\mathtt{if}\ C\ \mathtt{==}\ \emptyset\ \mathtt{then}\ []$
$\mathtt{else\ let}\ \{\theta_l = \theta_r\} \cup C' = C\ \mathtt{in}$
$\quad \mathtt{if}\ \theta_l\ \mathtt{==}\ \theta_r\ \mathtt{then\ unify}(C')$
$\quad \mathtt{else\ if}\ \theta_l\ \mathtt{==}\ \mathrm{Y}\ \mathtt{and}\ \mathrm{Y} \notin FV(\theta_r)\ \mathtt{then\ unify}([\mathrm{Y} \Rightarrow \theta_r]\, C') \circ [\mathrm{Y} \Rightarrow \theta_r]$
$\quad \mathtt{else\ if}\ \theta_r\ \mathtt{==}\ \mathrm{Y}\ \mathtt{and}\ \mathrm{Y} \notin FV(\theta_l)\ \mathtt{then\ unify}([\mathrm{Y} \Rightarrow \theta_l]\, C') \circ [\mathrm{Y} \Rightarrow \theta_l]$
$\quad \mathtt{else\ if}\ \theta_l\ \mathtt{==}\ \mathtt{f}(\theta_l^1, \ldots, \theta_l^n)\ \mathtt{and}\ \theta_r\ \mathtt{==}\ \mathtt{f}(\theta_r^1, \ldots, \theta_r^n)$
$\qquad \mathtt{then\ unify}(C' \cup \{\theta_l^1 = \theta_r^1, \ldots, \theta_l^n = \theta_r^n\})$
$\quad \mathtt{else}\ \mathbf{fail}$

$\mathrm{Y} \in FV(\theta)$ occur check, verhindert zyklische Substitutionen

Aufgabe 1 : Unifikation

$$\begin{aligned}
& \mathit{unify}(\{\mathtt{a}(\mathtt{t}_1, \mathtt{a}(\mathtt{X}_3, \mathtt{X}_4)) = \mathtt{a}(\mathtt{X}_1, \mathtt{X}_2),\ \mathtt{X}_3 = \mathtt{t}_2,\ \underline{\mathtt{X}_4 = \mathtt{X}_1}\}) \\
= \ & \mathit{unify}(\{\mathtt{a}(\mathtt{t}_1, \mathtt{a}(\mathtt{X}_3, \mathtt{X}_1)) = \mathtt{a}(\mathtt{X}_1, \mathtt{X}_2),\ \underline{\mathtt{X}_3 = \mathtt{t}_2}\}) \circ [\mathtt{X}_4 \Rightarrow \mathtt{X}_1] \\
= \ & \mathit{unify}(\{\underline{\mathtt{a}(\mathtt{t}_1, \mathtt{a}(\mathtt{t}_2, \mathtt{X}_1)) = \mathtt{a}(\mathtt{X}_1, \mathtt{X}_2)}\}) \circ [\mathtt{X}_3 \Rightarrow \mathtt{t}_2] \circ [\mathtt{X}_4 \Rightarrow \mathtt{X}_1] \\
= \ & \mathit{unify}(\{\underline{\mathtt{t}_1 = \mathtt{X}_1}, \mathtt{a}(\mathtt{t}_2, \mathtt{X}_1) = \mathtt{X}_2\}) \circ [\mathtt{X}_3 \Rightarrow \mathtt{t}_2] \circ [\mathtt{X}_4 \Rightarrow \mathtt{X}_1] \\
= \ & \mathit{unify}(\{\underline{\mathtt{a}(\mathtt{t}_2, \mathtt{t}_1) = \mathtt{X}_2}\}) \circ [\mathtt{X}_1 \Rightarrow \mathtt{t}_1] \circ [\mathtt{X}_3 \Rightarrow \mathtt{t}_2] \circ [\mathtt{X}_4 \Rightarrow \mathtt{X}_1] \\
= \ & [\mathtt{X}_2 \Rightarrow \mathtt{a}(\mathtt{t}_2, \mathtt{t}_1)] \circ [\mathtt{X}_1 \Rightarrow \mathtt{t}_1] \circ [\mathtt{X}_3 \Rightarrow \mathtt{t}_2] \circ [\mathtt{X}_4 \Rightarrow \mathtt{X}_1] \\
= \ & [\mathtt{X}_2 \Rightarrow \mathtt{a}(\mathtt{t}_2, \mathtt{t}_1)] \circ [\mathtt{X}_1 \Rightarrow \mathtt{t}_1] \circ [\mathtt{X}_3 \Rightarrow \mathtt{t}_2, \mathtt{X}_4 \Rightarrow \mathtt{X}_1] \\
= \ & [\mathtt{X}_2 \Rightarrow \mathtt{a}(\mathtt{t}_2, \mathtt{t}_1)] \circ [\mathtt{X}_1 \Rightarrow \mathtt{t}_1, \mathtt{X}_3 \Rightarrow \mathtt{t}_2, \mathtt{X}_4 \Rightarrow \mathtt{t}_1] \\
= \ & [\mathtt{X}_1 \Rightarrow \mathtt{t}_1, \mathtt{X}_2 \Rightarrow \mathtt{a}(\mathtt{t}_2, \mathtt{t}_1), \mathtt{X}_3 \Rightarrow \mathtt{t}_2, \mathtt{X}_4 \Rightarrow \mathtt{t}_1]
\end{aligned}$$

Aufgabe 1 : Unifikation

3. Bestimmen Sie einen allgemeinsten Unifikator für die Gleichung:

$$\mathtt{a}([1,2,3],[3,4],\mathtt{L}) = \mathtt{a}([\mathtt{X}|\mathtt{Xs}],[\mathtt{Y}|\mathtt{Ys}],\mathtt{L2})$$

Verwenden Sie Prolog-Notation: $\mathtt{X}, \mathtt{Xs}, \mathtt{Y}, \mathtt{Ys}, \mathtt{L}, \mathtt{L2}$ sind Variablen, `[_|_]`, `[_,_,_]` etc. Listen.

Aufgabe 2: Tester, Generatoren

In der Vorlesung wurden Prädikate `odd(X)` und `even(X)` vorgestellt. Diese *testen* ob `X` gerade ist, bzw. ob `X` ungerade ist. So wird die Anfrage `?even(4)` erfüllt, `?odd(4)` hingegen nicht.

```
even(0).
even(X)  :- X>0, X1 is X-1, odd(X1).

odd(1).
odd(X)   :- X>1, X1 is X-1, even(X1).
```

Listing 1: Tester

Die folgende Variante ist geeignet zum *Generieren* aller geraden bzw. aller ungeraden Zahlen. Beispielsweise gibt die Anfrage `?even(X)` bei wiederholter Neuerfüllung alle geraden Zahlen aus.

```
even(0).
even(X)  :-  odd(Y), X is Y+1, X>0.

odd(1).
odd(X)   :- even(Y), X is Y+1, X>1.
```

Listing 2: Generatoren

1. Warum ist keine der beiden Varianten sowohl als Tester als auch als Generator anwendbar?
2. Was passiert, wenn bei den Generatoren die Teilziele `X>0` und `X>1` weggelassen werden?

Aufgabe 3: Prolog, freie Variablen

```
del₁([],_,[]).
del₁([X|T1],X,L2)      :- !, del₁(T1,X,L2).
del₁([Y|T1],X,[Y|T2]) :- del₁(T1,X,T2).

del₂([],_,[]).
del₂([X|T1],X,L2)      :- del₂(T1,X,L2).
del₂([Y|T1],X,[Y|T2]) :- del₂(T1,X,T2), not(X=Y).

del₃([X|L],X,L).
del₃([Y|T1],X,[Y|T2]) :- del₃(T1,X,T2).
```

Anfrage: $\texttt{del}_i$`([1,2,1],X,L).`

a) `X=1,L=[2]` und
`X=2,L=[1,1]`

b) `X=1,L=[2]`

c) `X=1,L=[2,1]` und
`X=2,L=[1,1]` und
`X=1,L=[1,2]`

d) nichts (Endlosschleife, Stacküberlauf o. Ä.)

.. gibt aus ..	a)	b)	c)	d)
`del`$_1$	☐	☒	☐	☐
`del`$_2$	☒	☐	☐	☐
`del`$_3$	☐	☐	☒	☐

Aufgabe 3: Prolog, freie Variablen

```
del1([],_,[]).
del1([X|T1],X,L2)      :- !, del1(T1,X,L2).
del1([Y|T1],X,[Y|T2]) :- del1(T1,X,T2).

del2([],_,[]).
del2([X|T1],X,L2)      :- del2(T1,X,L2).
del2([Y|T1],X,[Y|T2]) :- del2(T1,X,T2), not(X=Y).

del3([X|L],X,L).
del3([Y|T1],X,[Y|T2]) :- del3(T1,X,T2).
```

2. Folgende Anfragen versuchen, die Rückwärtsausführung des Prädikats `del` ausnutzen. Welche Anfragen sind mit welcher Variante von $\texttt{del}_i$ erfüllbar? Warum ist das so?

   a) `deli(L, 2, [1, 3]).`
   b) `deli([1, 2, 3], X, [1, 3]).`
   c) `deli([1, 2, 3, 2], X, [1, 3]).`
   d) `deli([1, 2, 3, 2], X, [1, 2, 3]).`
   e) `deli([1|L], 1, X).`

Beispiellösung:

erfüllt	a)	b)	c)	d)	e
$\texttt{del}_1$	☐	☐	☐	☐	☒
$\texttt{del}_2$	☐	☒	☒	☐	☒
$\texttt{del}_3$	☒	☒	☐	☒	☒

Aufgabe 3: Prolog, freie Variablen

In Prolog können λ-Ausdrücke als Terme dargestellt werden. Wir betrachten nur Terme der Form `42` (Integer-Konstante), `x` (Variable), `abs(x,T)` (λ-Abstraktion) und `app(T,U)` (Applikation).

3. Geben Sie ein Prolog-Prädikat `fv(T,F)` an, das zu einem Lambda-Ausdruck `T` die Liste `F` der in `T` frei vorkommenden Variablen berechnet. [6 Punkte]

Beispiel: für den Lambda-Ausdruck $(\lambda$x. λy. x z 17) u

```
?fv(app(abs(x, abs(y, app(app(x, z), 17))), u), F).
F = [z,u];
no.
```

Hinweis: Sie können der Einfachheit halber annehmen, dass λ-gebundene Variablen nicht anderswo im Term frei verwendet werden.
Verwenden Sie Listenprädikate wie `append`, $\texttt{del}_i$, ... sowie Testprädikate **`integer`**`(X)` und **`atom`**`(X)`.