Übungsblatt 7

Abgabe der Lösungen: Tutorium in der Woche 06.01-10.12

Aufgabe 1 Listen umdrehen (Präsenzaufgabe)

Programmieren Sie in Prolog ein Prädikat `reverse/2`, das eine Liste umdreht, d.h. es soll gelten

$$\texttt{reverse}([a_1, \ldots, a_n], l) \iff l = [a_n, \ldots, a_1].$$

Es soll hierbei natürlich kein „eingebautes“ reverse-Prädikat verwendet werden, sondern Sie sollen die Funktion selbst programmieren. Sie können aber das Prädikat `append/3` aus der Vorlesung verwenden, das zwei Listen aneinanderhängt, d.h.

$$\texttt{append}([a_1, \ldots, a_n], [b_1, \ldots, b_k], l) \iff l = [a_1, \ldots, a_n, b_1, \ldots, b_k].$$

Programmieren Sie nun *ohne Verwendung von* `reverse` (und selbstverständlich auch ohne `reverse` noch einmal nachzuimplementieren, etwa per gesonderten Klauseln für $\texttt{reverse2}(k, [])$) eine Version `reverse2` von `reverse` mit *Akkumulator*, d.h. es soll gelten

$$\texttt{reverse2}([a_1, \ldots, a_n], [b_1, \ldots, b_k], l) \iff l = [a_n, \ldots, a_1, b_1, \ldots, b_k]$$

(man beachte die umgedrehte Reihenfolge der a_i). Wie erhält man eine Implementierung von `reverse` aus `reverse2`?

Aufgabe 2 SLD-Resolution (Präsenzaufgabe)

Geben Sie die zwei kürzesten SLD-Refutationen gemäß der Prolog-Regel für die Anfrage

$$\leftarrow \texttt{append([0|X],[1|Y],[0,1|Z])}$$

an (mit `append` wie in der Vorlesung).

Aufgabe 3 SLD-Resolution (5 Punkte)

Geben Sie die zwei kürzesten SLD-Refutationen gemäß der Prolog-Regel für die Anfrage

$$\leftarrow \texttt{opa(daniel,X)}$$

in Bezug auf das Prolog-Program

```
1  vater(daniel,berthold).
2  vater(berthold,claus).
3  vater(daniel,tobias).
4  vater(carsten,frank).
5  vater(tobias,carsten).
6
7  opa(X,Y):-vater(X,Z),vater(Z,Y).
8  onkel(X,Y):-vater(Z,X),opa(Z,Y).
```

an.

Aufgabe 4 Binäre Bäume (7 Punkte)

Wir repräsentieren binäre Bäume als Terme, definiert durch die folgende Grammatik:

$$\gamma_1, \gamma_2 ::= X \mid bin(\gamma_1, \gamma_2) \qquad (X \in V)$$

wobei V ein Vorrat von Variablennamen $X, Y, \ldots$ ist. Beispiel: $bin(X, bin(X, Y))$.
Wir definieren die *Tiefe* eines Baums rekursiv wie folgt:

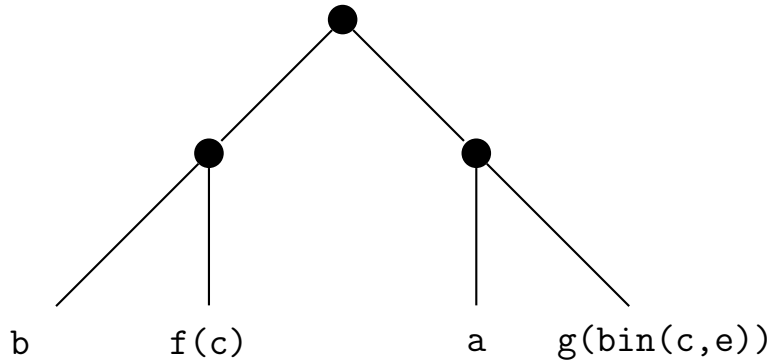
- Die Tiefe einer Variablen ist 0.
- Die Tiefe eines Baums $bin(a, b)$ ist 1 plus das Maximum der Tiefen von a und b.

Somit ist zum Beispiel $2 = 1 + \max\{0, 1 + \max\{0, 0\}\}$ die Tiefe von $bin(X, bin(X, Y))$.

1. Definieren Sie (erneut) rekursiv den Substitutionsoperator $b[X_1 \mapsto b_1, \ldots, X_n \mapsto b_n]$ für binäre Bäume $b, b_1, \ldots, b_n$.
2. Seien a und b zwei binäre Bäume. Beweisen Sie per Induktion über die Struktur von b, dass die Tiefe von $b[X_1 \mapsto a, \ldots, X_n \mapsto a]$ die Summe der Tiefen von a und b ist, wobei $Vars(b) = \{X_1, \ldots, X_n\}$ (notwendigerweise $n > 0$).

Binäre Bäume in Prolog (8 Punkte)

Binäre Bäume werden in Prolog einfach als Terme mit einer speziellen binären Operation—wir nehmen beispielsweise `bin/2`—präsentiert. Somit entspricht beispielsweise der folgende Baum dem Term `bin(bin(b,f(c)),bin(a,g(bin(c,e))))`.



Implementieren Sie folgende Operationen über binären Bäumen in Prolog.

1. `mirror/2` — Spiegelung des Baums von links nach rechts.
2. `bin_to_list/3` und `bin_to_list/2` — Übersetzung des Baums (erstes Argument) in eine Liste (zweites Argument), die alle Blätter des Baums enthält. Das dritte Argument der ersten Funktion ist eine Liste, die an das Ergebnis am Ende angehängt werden soll (Akkumulator). Das zweite Prädikat darf das erste Prädikat verwenden (aber nicht umgekehrt).

3. `rem_first/3` — Löschen des ersten Vorkommens eines gegebenen Elements aus dem Baum.

4. `rem_all/3` — Löschen aller Vorkommen eines gegebenen Elements aus dem Baum; wenn dadurch keine Elemente übrig bleiben, einfach dieses Element als Ergebnis ausgeben.

Sie können bei der Programmierung Negation verwenden, etwa in der Form `\+(X=bin(_,_))` – die Beispielformel drückt aus, dass die beiden Seiten nicht unifizierbar sind, `X` also nicht von der Form `bin(E,D)` ist. Die obigen Spezifikationen unterstellen eine *Reihenfolge* der Blätter des Baums; gemeint ist hier die offensichtliche Reihenfolge, d.h. von links nach rechts; in der Abbildung oben beispielsweise ist die Reihenfolge $b, f(c), a, g(bin(c, e))$.

Beispiel des beabsichtigten Programmablaufs:

```
?- mirror(bin(bin(a,b),c), X).
X = bin(c, bin(b, a)) ;
false.

?- bin_to_list(bin(bin(a,b),c), X, [d]).
X = [a, b, c, d] .
false.

?- bin_to_list(bin(bin(a,b),c), X).
X = [a, b, c] ;
false.

?- rem_first(bin(bin(a,b),a), a, X).
X = bin(b, a) ;
false.

?- rem_all(bin(bin(a,b),a), a, X).
X = b ;
false.
?

?- rem_all(bin(bin(a,a),a), a, X).
X = a ;
X = a ;
X = a ;
X = a.
```