Programmierung

Aufgabe 1 (AGS 13.12)

(a) Programmieren Sie in Prolog$^-$ eine binäre Relation `sublist`, die für jedes Paar (l_1, l_2) von Listen über natürlichen Zahlen wahr ist, wenn l_1 eine Teilliste von l_2 ist. Zum Beispiel gilt: `sublist([<2>, <3>],[<1>, <2>, <3>])` wobei, wie üblich, `<2>` die natürliche Zahl `s(s(0))` bezeichnet.

Hinweis: Nutzen Sie die beiden folgenden Prädikate.

```
1  nat(0).
2  nat(s(X)) :- nat(X).
3
4  listnat([]).
5  listnat([X|XS]) :- nat(X), listnat(XS).
```

(b) Bestimmen Sie durch SLD-Resultion für das Goal

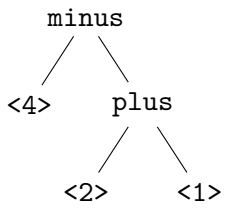
```
?- sublist([<4>|XS], [<5>, <4>, <3>]).
```

zwei Belegungen der Variablen `XS`.

Aufgabe 2 (AGS 13.13 a)

Ein *binärer Termbaum* ist ein Binärbaum über den zweistelligen Symbolen `plus` und `minus` und den natürlichen Zahlen (in der selben Form wie in Aufgabe 1) als nullstellige Symbole. In der Abbildung unten rechts ist ein Beispiel für einen binären Termbaum schematisch dargestellt.

Programmieren Sie in Prolog$^-$ eine binäre Relation `eval`, die genau die Paare `(T, X)` enthält, sodass `T` ein binärer Termbaum und `X` das natürlichzahlige Ergebnis der Auswertung dieses Terms ist. Die Subtraktion soll nur definiert sein, wenn der Minuend größer oder gleich dem Subtrahenden ist. Beispielweise soll der Term aus der rechts gezeigten Abbildung das Ergebnis `<1>` haben.



Hinweis: Nutzen Sie dafür die Prädikate `nat` aus Aufgabe 1 und `sum`.

```
3  sum(0, Y, Y) :- nat(Y).
4  sum(s(X), Y, s(S)) :- sum(X, Y, S).
```

Aufgabe 3 (AGS 13.5)

Das Prädikat `subt` sei wiefolgt gegeben (intuitiv beschreibt es die Teilbaumrelation):

```
1  subt(X, X).
2  subt(S1, s(_, T2)) :- subt(S1, T2).
3  subt(S1, s(T1, _)) :- subt(S1, T1).
```

Hinweis: Jedes Vorkommen von „_“ in einem Kopfliteral verhält sich wie eine neue Variable, die *nicht* auf der rechten Seite vorkommt. D.h. bei der Unifikation in einem Resolutionsschritt können diese Vorkommen ignoriert werden.

(a) Bestimmen Sie durch SLD-Refutation alle Belegungen von `X` und `Y` für das Goal

```
?- subt(s(X, Y), s(s(a, b), s(b, a))).
```

(b) Bestimmen Sie durch SLD-Refutation drei verschiedene Lösungen für das Goal

```
?- subt(s(a, a), X).
```

Zusatzaufgabe 1 (AGS 13.7)

Gegeben sei das dreistelliges Prädikat `append`, welches die Verkettung von Listen ausdrückt.

```
1 append([], Ys, Ys).
2 append([X|Xs], Ys, [X|Zs]) :- append(Xs, Ys, Zs).
```

(a) Geben Sie unter Nutzung von `append` ein Prologprogramm für das zweistellige Prädikat `reverse` an, welches erfüllt ist, falls das erste Argument eine Liste `Xs` ist und das zweite Argument `Ys` die Elemente von `Xs` in umgekehrter Reihenfolge enthält. Beispielsweise soll gelten: `reverse([1,2,3], [3,2,1]).`

(b) Eine Liste wird *Palindrom* genannt, falls die Folgen ihrer Elemente von vorn nach hinten und von hinten nach vorn gleich sind. Die Liste `[1,2,3,2,1]` ist zum Beispiel ein Palindrom. Bilden Sie diese Eigenschaft als einstelliges Prädikat `palindrom` in Prolog ab. Sie dürfen die Hilfsfunktionen `append` und `reverse` benutzen.

(c) Gegeben sei das folgende Prologprogramm:

```
1 s(alice, bob).
2 s(bob, charly).
3 s(alice, dory).
4 s(eve, alice).
5
6 t(X, Y) :- s(X, Y).
7 t(X, Z) :- s(X, Y), t(Y, Z).
```

Geben Sie alle SLD-Refutationen für `?- t(alice, X).` an. Geben Sie dabei die Belegungen für alle Variablen an. Geben Sie die Menge der möglichen Belegungen für `X` an.

Zusatzaufgabe 2 (AGS 13.13 a)

Wir betrachten Binärbäume in Prolog$^-$ als Terme der Form `nil` oder `tree(X, L, R)`, wobei `X` für ein Symbol steht und `L` und `R` Binärbäume sind. Das Funktionssymbol `nil` steht für den leeren Baum.

Gegeben seien die zwei Bäume

`<t1>` = `tree(a, tree(b, nil, nil), tree(v, nil, nil))`

und `<t2>` = `tree(c, nil, tree(d, nil, nil))`

und das folgende Prolog$^-$-Programm:

```
1 istree(nil).
2 istree(tree(_, L, R)) :- istree(L), istree(R).
3
4 insert(nil, _, nil).
5 insert(tree(v, _, _), T, T) :- istree(T).
6 insert(tree(X, L, R), T, tree(X, LT, RT))
7   :- insert(L, T, LT), insert(R, T, RT).
```

Bestimmen Sie durch SLD-Refutation für das Goal `?- insert(<t1>, <t2>, X).` eine Belegung der Variablen `X`.

Hinweise: Jedes Vorkommen von „`_`“ in einem Kopfliteral verhält sich wie eine neue Variable, die *nicht* auf der rechten Seite vorkommt. D.h. bei der Unifikation in einem Resolutionsschritt können diese Vorkommen ignoriert werden.
Beachten Sie, dass in der Zeile 5 das Symbol `v` ein spezifischer nullstelliger Konstruktor ist. Sie dürfen die oben genannten Bäume weiterhin mit `<t1>` und `<t2>` abkürzen. Mehrere Resolutionsschritte unter Anwendung der selben Zeile können Sie mit `?-*` zusammenfassen.