PROGRAMMIERUNG

ÜBUNG 9: LOGIKPROGRAMMIERUNG MIT PROLOG⁻

Eric Kunze

eric.kunze@mailbox.tu-dresden.de

INHALT

Logikprogrammierung und Prolog⁻

EINFÜHRUNG IN PROLOG

- Französisch: programmation en logique
  (deutsch: Programmieren in Logik)
- **Online-Editor & Interpreter:** `swish.swi-prolog.org`
- Prolog-Programme bestehen aus **Fakten** und **Regeln**.

- Statements werden mit `.` abgeschlossen.
- Variablen beginnen mit Großbuchstaben.

- **UND**-Operator. `,`
- **ODER**-Operator. `;`

Aufgabe 1

AUFGABE 1 – TEIL (A)

Ziel: binäre Relation `sublist` mit $(\ell_1, \ell_2) \in \texttt{sublist} \Leftrightarrow \ell_1 \subseteq \ell_2$

$\rightarrow \ell_1$ soll *Teilliste* von ℓ_2 sein

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

AUFGABE 1 – TEIL (A)

Ziel: binäre Relation `sublist` mit $(\ell_1, \ell_2) \in \texttt{sublist} \Leftrightarrow \ell_1 \subseteq \ell_2$

$\rightarrow \ell_1$ soll *Teilliste* von ℓ_2 sein

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

```
6  sublist(Xs    , [Y|Ys]) :- nat(Y), sublist(Xs, Ys).
7  sublist(Xs    , Ys )    :- prefix(Xs, Ys).
8
9  prefix([]     , Ys )    :- listnat(Ys).
10 prefix([X|Xs], [X|Ys]) :- nat(X), prefix(Xs, Ys).
```

AUFGABE 1 – TEIL (B)

Belegung 1:

```
            ?-  sublist ([<4>|Xs], [<5>, <4>, <3>]).
            ?-  nat(<5>), sublist ([<4>|Xs], [<4>, <3>]).  % 6
            ?-* nat(0), sublist ([<4>|Xs], [<4>, <3>]).    % 2
            ?-  sublist ([<4>|Xs], [<4>, <3>]).            % 1
            ?-  prefix ([<4>|Xs], [<4>, <3>]).             % 7
            ?-  nat(<4>), prefix(Xs , [<3>]).              % 10
            ?-* nat(0), prefix(Xs , [<3>]).                % 2
            ?-  prefix(Xs , [<3>]).                        % 1
{Xs = []}   ?-  listnat ([<3>]).                           % 9
            ?-  nat(<3>), listnat ([]).                    % 5
            ?-* nat(0), listnat ([]).                      % 2
            ?-  listnat ([]).                              % 1
            ?-  .                                          % 4
```

Somit also `Xs = []`.

AUFGABE 1 – TEIL (B)

Belegung 2:

```
                  ?-  sublist ([<4>|Xs], [<5>, <4>, <3>]).
                  ?-  nat(<5>), sublist ([<4>|Xs], [<4>, <3>]).
                                                          % 6
                  ?-* nat(0), sublist ([<4>|Xs], [<4>, <3>]).
                                                          % 2
                  ?-  sublist ([<4>|Xs], [<4>, <3>]).     % 1
                  ?-  prefix ([<4>|Xs], [<4>, <3>]).      % 7
                  ?-  nat(<4>), prefix(Xs , [<3>]).       % 10
                  ?-* nat(0), prefix(Xs , [<3>]).         % 2
                  ?-  prefix(Xs , [<3>]).                 % 1
{Xs=[<3>|Xs1]}    ?-  nat(<3>), prefix(Xs1 , []).         % 10
                  ?-* nat(0), prefix(Xs1 , []).           % 2
                  ?-  prefix(Xs1 , []).                   % 1
{Xs1 = []}        ?-  listnat ([]).                       % 9
                  ?-  .                                   % 4
```

Somit also `Xs = [<3>|Xs1] = [<3>].`

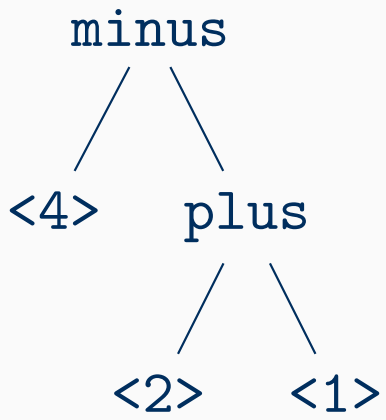
Aufgabe 2

AUFGABE 2

Wir wollen einen binären Termbaum auswerten.

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

Beispiel:



AUFGABE 2

Wir wollen einen binären Termbaum auswerten.

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

```
5 eval( X          , X ) :-  nat(X).
6 eval( plus (L,R), X ) :-  eval(L, LE), eval(R, RE), sum(LE, RE,  X).
7 eval( minus(L,R), X ) :-  eval(L, LE), eval(R, RE), sum(RE,  X, LE).
```

Aufgabe 3

AUFGABE 3 – TEIL (A)

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

```
                          ?-  subt(s(X,Y), s(s(a,b), s(b,a))).
{X = s(a,b), Y=s(b,a)}    ?-  .                                 % 1

                          ?-  subt(s(X,Y), s(s(a,b), s(b,a))).
                          ?-  subt(s(X,Y), s(b,a)).           % 2
{X = b, Y=a}              ?-  .                                 % 1

                          ?-  subt(s(X,Y), s(s(a,b), s(b,a))).
                          ?-  subt(s(X,Y), s(a,b)).           % 3
{X = a, Y=b }             ?-  .                                 % 1
```

AUFGABE 3 – TEIL (B)

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

```
                    ?-  subt(s(a,a), X).
{X = s(a,a)}        ?-  .                       % 1
                                                ⇒ X = s(a,a))
                    ?-  subt(s(a,a), X).
{X = s(_ , X1)}     ?-  subt(s(a,a), X1).       % 2
{X1 = s(a,a)}       ?-  .                       % 1
                                                ⇒ X = s(a,s(a,a))

                    ?-  subt(s(a,a), X).
{X = s(X2, _)}      ?-  subt(s(a,a), X2).       % 3
{X2 = s(a,a) }      ?-  .                       % 1
                                                ⇒ X = s(s(a,a),c)
```