SEIII - Logikprogrammierung

Übungsblatt 06

Nico Hahn 6990715 Hieu Nguyen 6632126

Aufgabe 1

```
% Aufgabe 1.1
% zins(+Anlagebetrag,+Zinsfaktor,+Anlagedauer,?Endguthaben)

zins(Anlagebetrag,_,0,Anlagebetrag).
zins(Anlagebetrag,Zinsfaktor,Anlagedauer,Endguthaben) :-
        Anlage is ((Zinsfaktor +1) * Anlagebetrag),
        Dauer is Anlagedauer - 1,
        zins(Anlage,Zinsfaktor,Dauer,Endguthaben).

% test:
% ?- zins(1000,0.05,2,X).
% X = 1102.5

% Aufgabe 1.2
% zinsOhneRek(+Anlagebetrag,+Zinsfaktor,+Anlagedauer,?Endguthaben)
zinsOhneRek(Anlagebetrag,_,0,Anlagebetrag).
zinsOhneRek(Anlagebetrag,Zinsfaktor,Anlagedauer,Endguthaben) :-
        Endguthaben is (Zinsfaktor * Anlagebetrag * Anlagedauer) +
Anlagebetrag.

% test:
% ?- zinsOhneRek(1000,0.05,2,X).
% X = 1100.0.



% Aufgabe 1.3
% Aufgabe 1.1 ist bereits Endrekursiv gelöst.

% Aufgabe 1.4
% zuwachsZins(+Anlagebetrag,+Bonuszins,+Basiszinz,+Anlagedauer,?Endguthaben)

zuwachsZins(Anlagebetrag,_,_,0,Anlagebetrag).
zuwachsZins(Anlagebetrag,Bonuszins,Basiszinz,Anlagedauer,Endguthaben) :-
        Bonus is (Bonuszins / 2),
        Zins is Basiszinz + Bonus,
        Anlage is (1+ Zins) * Anlagebetrag,
        Dauer is Anlagedauer - 1,
        zuwachsZins(Anlage, Bonus, Zins, Dauer, Endguthaben).
```

```
%test:
% ?- zuwachsZins(1000,0.04,0.01,2,X).
% X = 1071.2.
% ?- zuwachsZins(1000,0.04,0.01,2,1071.2).
% true.

% Aufgabe 1.5
%
% Das Modell mit dem festen Zinssatz eignet sich für 1 - 4 Jahre.
% Ab dem 4. Jahr eignet sich der variable Zinssatz mehr, da man dort mehr
Zinsen / Jahr macht.
```

Aufgabe 2

```
% 2.1
% pi_aufstieg(+Iterationen, ?Resultat)
pi_aufstieg(0,0).
pi_aufstieg(N,Res) :-
  ResN is 4.0 * (-1.0)^(N+1) / (2*N-1),
  NeuN is N - 1,
  pi_aufstieg(NeuN,NeuRes),
  Res is ResN + NeuRes.

% pi_abstieg(+Iterationen, ?Resultat)
pi_abstieg(N,Res) :- pi_rek(N,0,Res).
pi_rek(0,Acc,Res) :- Res is Acc.
pi_rek(N,Acc,Res) :-
  NeuAcc is Acc + 4.0 * (-1.0)^(N+1) / (2*N-1),
  NeuN is N - 1,
  pi_rek(NeuN,NeuAcc,Res).

% Es liegt nur bei der pi_abstieg Endrekursion vor, denn die
Zwischenergebnisse
% werden bei pi_abstieg berechnet bevor es in die naechste Rekursionschritt
% geht. Am Ende der Rekursion liegt das Endergebnis vor.

% 2.2
% Bzgl. der Verstaendlichkeit ist die pi_aufstieg Version verstaendlicher,
% denn man besser sehen kann, dass das Ergebnis eines Problem von dem
Ergebnis
% des kleineren Problems abhaengt. Das ist schwerer bei pi_abstieg zu sehen.
%
% Bgzl. des Berechnungsverhalten ist die pi_abstieg Version effizienter. Denn
% nach dem Abschluss einer Rekursionschritt nichts mehr berechnet werden
muss,
% kann ein Kompiler so optimieren, dass beim Aufruf einer rekursiven Funktion
% das Stack fuer die aktuelle Funktion freigegeben wird.

% 2.3
```

```
% pi_alt(+Iterationen, ?Resultat)
pi_alt(N,Res) :- pi_alt_rek(N,1,Result), Res is Result * 2.
pi_alt_rek(0,Acc,Res) :- Res is Acc.
pi_alt_rek(N,Acc,Res) :-
  NeuAcc is Acc * ((2*N)/(2*N-1)) * ((2*N)/(2*N+1)),
  NeuN is N - 1,
  pi_alt_rek(NeuN,NeuAcc,Res).
```

Aufgabe 4

```
% Aufgabe 4.1
% over(n, k, Result)
over(_,0,1).
over(K,K,1).
over(N,K,Result) :-
        N1 is (N - 1),
        K1 is (K - 1),
        over(N1,K1,SumA),
        over(N1,K,SumB),
        Result is SumA + SumB.
```