SEIII - Logikprogrammierung

Übungsblatt 04

Nico Hahn 6990715 Hieu Nguyen 6632126

Aufgabe 1

- A ist Nachbarland von B - symmetrisch und vllt. reflexiv: B ist auch Nachbarland von A wenn A Nachbarland von B ist. Man kann auch sagen dass die Relation reflexiv ist - A ist auch ein Nachbarland von A.
- A und B sind (nach deutschem Recht) miteinander verheiratet - symmetrisch: A kann nicht sich selbst heiraten - also nicht reflexiv. A und B sind verheiratet bedeutet dann sind B und A auch.
- A und B sind Geschwister - symmetrisch,s ggf. reflexiv: Geschwister-Relation zwischen A und B ist klar symmetrisch. Sie ist dann reflexiv, wenn die Definition von Geschwister ist, dass sie die gleichen Eltern haben. Dann sind A und C auch Geschwister falls B und C Geschwister sind.
- A ist größer oder gleich B - reflexiv und transitiv: A ist selbsverständlich größer oder gleich A (reflexiv). Falls B größer oder gleich C ist, dann ist A größer oder gleich C.
- A und B haben ein gemeinsames Hobby - symmetrisch und reflexiv: A und B haben ein gemeinsames Hobby dann B und A haben auch ein gemeinsames Hobby. A hat selbstverständlich mit A ein gemeinsames Hobby. Falls B und C ein gemeinsames Hobby bedeutet es nicht, dass dieses Hobby gleich das Hobby zwischen A und B. Daher muss A und C auch nicht ein gemeinsames Hobby haben.
- A und B sind Häuser in der gleichen Straße - symmetrisch, reflexiv und transitiv: A ist mit A zusammen in der gleichen Straße - das ist klar (reflexiv). A und B sind in der gleichen Straße dann B und A auch (symmetrisch). A und B sind in der gleichen Straße und B und C sind in der gleichen Straße; da es nur eine Straße sein kann, sind A und C auch Häuser in der gleichen Straße (transitiv).

Aufgabe 2

```
?- consult('dateiverzeichnis.pl').

% 2.1 Zugriffpfad zwischen zwei Verzeichnissen
% zugriffpfad(?DirId1, ?DirId2)
con_sym(DId1, DId2) :- directory(DId1,_,DId2,_,_).
```

```
con_sym(DId1, DId2) :- directory(DId2,_,DId1,_,_).
zugriffpfad(DId1, DId2) :- con_sym(DId1,DId2).
zugriffpfad(DId1, DId2) :- con_sym(DId1,DId3), zugriffpfad(DId3, DId2).

% 2.2 Ueberpreuft, ob eine Datei vom Wurzelverzeichnis aus nicht mehr erreichbar ist
% nicht_zugreifbar(?FId)
wurzel_dir(DId) :- directory(DId,_,1,_,_).
wurzel_dir(DId) :- directory(DId,_,PId,_,_), wurzel_dir(PId).
wurzel_datei(FId) :- file(FId,DId,_,_,_,_), wurzel_dir(DId).
nicht_zugreifbar(FId) :- \+ wurzel_datei(FId).

% 2.3 Direkte und indirekte Unterverzeichnisse ermitteln
% uv(+PId,-UnterDId)
uv(PId,UnterDId) :- directory(UnterDId,_,PId,_,_).
uv(PId,UnterDId) :- directory(UnterDId,_,UnterDId2,_,_),
                    uv(PId,UnterDId2).

% 2.4 Gesamtgröße aller Datein
% sumsize(+DId, -Size)
filesizes(DId, Size) :- file(_,DId,_,Size,_,_);
                        (uv(DId,UnterDId),file(_,UnterDId,_,Size,_,_)).
sumsize(DId, Size) :- findall(FSize,filesizes(DId,FSize),SizeList),
                      sumlist(SizeList, Size).
```

Aufgabe 3

```
?- consult('skigebiet.pl').
% Aufgabe 3.1
is_highest(P) :- strecke(_,P,_,_,_), \+ strecke(_,_,P,_,_).
% Hoechste Punkte treten niemals als ZeilPunkt auf
% -> Hoechste Punkte sind bgrootmoos, bgpanorama und bgzirben
is_lowest(P) :- strecke(_,_,P,_,_), \+ strecke(_,P,_,_,_).
% Niedrigste Punkte treten niemals als StartPunkt auf
% -> Niedrigste Punkte sind tlzollberg und tlgondel

% Aufgabe 3.2
% ist_erreichbar(?Start,?Ziel)
ist_erreichbar_piste(Start,Ziel,PNr) :-
  strecke(_,Start,Ziel,PNr,_).
ist_erreichbar_piste(Start,Ziel,PNr) :-
  strecke(_,Start,ZielX,PNr,_),
  ist_erreichbar_piste(ZielX,Ziel,PNr).
ist_erreichbar(Start,Ziel) :- ist_erreichbar_piste(Start,Ziel,_).

% Es ist nicht terminierungssicher. Wäre ein Strecke falsch definiert,
```

```
% sodass ein Kreis entsteht, % dann haben wir eine endlose Schleife.
% Eigenschaft: transitiv - falls es einen Pfad von A zu B und von B zu C
% dann gibt es auch einen Pfad von A zu C.

% Aufgabe 3.3
% keine_verbindung(?OrtA,?OrtB)
% pit_con_sym prueft ob zwei Orte mit einander verbunden sind
pit_con_sym(OrtA, OrtB) :- ist_erreichbar(OrtA, OrtB).
pit_con_sym(OrtA, OrtB) :- ist_erreichbar(OrtB, OrtA).
keine_verbindung(OrtA, OrtB) :-
  (strecke(_,OrtA,_,_,_); strecke(_,_,OrtA,_,_)),
  (strecke(_,OrtB,_,_,_); strecke(_,_,OrtB,_,_)),
  OrtA \= OrtB,
  \+ pit_con_sym(OrtA, OrtB).

% Aufgabe 3.4
% treffpunkt(+OrtA, +OrtB, -Treffpunkt)
treffpunkt(OrtA, OrtB, X) :- OrtA = OrtB, X = OrtA.
treffpunkt(OrtA, OrtB, X) :- ist_erreichbar(OrtA, OrtB), X = OrtB.
treffpunkt(OrtA, OrtB, X) :- ist_erreichbar(OrtB, OrtA), X = OrtA.
treffpunkt(OrtA, OrtB, X) :- ist_erreichbar(OrtA, X),
                             ist_erreichbar(OrtB, X).

% Aufgabe 3.5
:- dynamic(gesperrt/1).
% ist_erreichbar_sperre(?Start,?Ziel)
hilf_ist_erreichbar_sperre(StrNr,Start,Ziel,PNr) :-
  strecke(StrNr,Start,Ziel,PNr,_),
  \+ gesperrt(StrNr).
hilf_ist_erreichbar_sperre(StrNr,Start,Ziel,PNr) :-
  strecke(StrNr,Start,ZielMittel,PNr,_),
  \+ gesperrt(StrNr),
  hilf_ist_erreichbar_sperre(_,ZielMittel,Ziel,PNr).
ist_erreichbar_sperre(Start,Ziel) :- hilf_ist_erreichbar_sperre(_,Start,Ziel,_).
```