Logikprogrammieren Hausaufgaben

Blatt 3

Morten Seemann 6945442, Tore Wiedenmann 6488837

Aufgabe 1: Welche Eigenschaften (symmetrisch, reflexiv, transitiv, funktional in einem der Argumente) haben die folgenden Relationen?

A ist das Geburtsdatum von B: Ist weder reflexiv noch symmetrisch oder transitiv da A stets ein Datum und B nie ein Datum ist und Daten nie ein Geburtsdatum besitzen. Es ist von Typ 1:m da eine Person nur ein Geburtsdatum hat

A ist im Turnier gegen B angetreten ist nicht reflexiv da man nie gegen sich selbst antritt, sofern es sich nicht um ein "round-robin" Turnier handelt auch nicht transitiv aber symmetrisch. Keine Funktionale Abhängigkeit, da es in einem Turnier viele verschieden Teilnehmer geben kann.

A ist eine Teilmenge von B Ist reflexiv da jede Menge eine Teilmenge von sich selbst ist, nicht symmetrisch da A echte Teilmenge sein kann und transitiv. Keine funktionale Abhängigkeit, da man auch Teilmenge einer Teilmenger einer anderen Menge sein kann.

A und B spielen im gleichen Film mit reflexiv da man in allen Filmen mitspielt in denen man selbst mitspielt.
symmetrisch da es sich um den gleichen Film handelt.
Nicht transitiv da es sich beim $A \sim B$ Film um einen anderen Film als den $B \sim C$ Film handeln kann.
Keine funktionale Abhängigkeit da es viele Schauspieler in einem Film gibt

A ist kongruent zu B Ist reflexiv da alles kongruent zu sich selbst ist. Symmetrisch da kongruent sein symmetrisch ist. transitiv da kongruent sein transitiv ist. Sowohl vom Typ 1:m als auch vom Typ m:1 , da eine Fläche nur zu sich selbst kongruent sein kann. Sobald eine andere Fläche zur ursprünglichen kongruent ist sind die beiden Flächen bereits gleich.

Aufgabe 2: Definieren Sie für das in der Datei "dateiverzeichnis.pl" gegebene Dateiverzeichnis aus Aufgabenblatt 3 die folgenden Prädikate:

1. Ein Prädikat, das überprüft, ob ein Zugriffspfad zwischen zwei Verzeichnissen existiert

```
% Entweder Start ist Oberverzeichniss von Ziel
ist_erreichbar(S,SID,Z,ZID) :- parent_dir(S,SID,Z,ZID),

% oder Start ist Oberverzeichniss eines Oberverzeichniss von Ziel.
ist_erreichbar(S,SID,Z,ZID) :- parent_dir(S,SID,X,XID),
                               ist_erreichbar(X,XID,Z,ZID).

%dabei ermittelt parent_dir ob es sich um ein direktes oberverzeichniss
   handelt.
parent_dir(S,SID,Z,ZID) :- directory(ZID,Z,SID,_,_),
                           directory(SID,S,_,_,_).
```

2. Ein Prädikat, das überprüft, ob eine Datei vom Wurzelverzeichnis aus nicht mehr erreichbar ist.

```
von_root_erreichbar(F,FID) :- file(FID,DID,F,_,_,_),
                              directory(DID,DIR,_,_,_),
                              ist_erreichbar(root,1,DIR,DID),
                              !.

nicht_von_root_erreichbar(F,FID) :- not(von_root_erreichbar(F,FID)).
```

3. Ein Prädikat, das zu einem gegebenen Verzeichnis alle seine Unterverzeichnisse, d.h. auch die nur indirekt erreichbaren, ermittelt.

```
alle_unterverzeichnisse(DIR,DID,L) :- findall(XID,
                                        (directory(XID,X,_,_,_),
                                         ist_erreichbar(DIR,DID,X,XID))
                                       ,L).
%Ergebniss ist Liste der IDs der Unterverzeichnisse.
```

4. Ein Prädikat, das die Gesamtgröße aller Dateien in einem Teilbaum des Verzeichnisbaums berechnet

```
%Erstellt eine Liste von filesizes eines directories(benutz praedikate vom
    letzen Blatt)
von_sizes_in(LISTE,DID) :- findall(SIZE,
                                   (file(_,DID,FILE,SIZE,_,_),
                                    liegt_in_dir(FILE,_,DID)),
                                   LISTE).

%Berechnet die Groesse eines directories falls es keine subdirectories
   besitzt
teilbaumsize(RID,SIZE) :- directory(RID,ROOT,_,_,_),
                          not(parent_dir(ROOT,RID,_,_)),
                          von_sizes_in(L,RID),
                          sumlist(L,SIZE).

%Wenn subdirectories vorhanden ermittelt es deren groesse und gibt sie aus
teilbaumsize(RID,SIZE) :- directory(RID,ROOT,_,_,_),
                          alle_unterverzeichnisse(ROOT,RID,L),
                          member(SDID,L),
                          von_sizes_in(FSL,SDID),
                          sumlist(FSL,SIZE).

%Das gesuchte praedikat erstellt eine Liste aller groessen aller nicht
   mehr unterteielten directories und summiert diese.
verzeichnissgroesse(VID,VGR) :- findall(SIZE,teilbaumsize(VID,SIZE),L),
                                sumlist(L,VGR).
```

Aufgabe 3: 1. Ermitteln Sie die höchsten Punkte in dem Skigebiet, von denen aus alle Pisten bergab fuhren, bzw. die niedrigsten, von denen aus man nicht weiter ins Tal abfahren kann:

Die höchsten Punkte im Skigebiet sind bgrootmoos und bgzirben.
Sie haben untereinander keine Verbindungsstrecke. Von ihnen aus erreicht man stets den niedrigsten Talpunkt tlgondel.

2. Definieren Sie ein Prädikat "ist-erreichbar(Start,Ziel)", das berechnet, ob zwischen zwei Knotenpunkten Start und Ziel eine durchgängig zu befahrende, bergabwärts führende Verbindung besteht.

```
ist_erreichbar(S,Z) :- strecke(_,S,Z,_,_).
ist_erreichbar(S,Z) :- strecke(_,S,X,_,_),ist_erreichbar(X,Z),!.
```

Ist ein Terminierungssicheres, transitives Prädikat. Es ist nicht reflexiv oder symmetrisch da die Streckeneinträge stets gerichtet mit länge größer 0 sind.

3. Definieren Sie ein Prädikat ort/1, das alle Orte im Skigebiet ermittelt, die entweder Start- oder Zielpunkt einer Strecke sind (oder auch beides). Achten Sie dabei darauf, dass Ihr Prädikat keine Doppelergebnisse produziert.

```
%Jede Strecke endet irgendwann in tlgondel also ist alles was irgendwann
   in tlgondel endet ein Ort
moeglicher_ort(O) :- ist_erreichbar(O,tlgondel).
%Eine Liste aller Orte
alle_moegl_orte(L) :- findall(O,moeglicher_ort(O),L).
%sort bereinigt duplikate
ohne_duplikate(X):- alle_moegl_orte(L),sort(L,X).
%Man ist also genau dann ein ort wenn man in der Liste ohne duplikate
   steht.
ort(O) :- member(O,L),ohne_duplikate(L).
```