Inleiding Logica 2013

Praktica Logisch Programmeren

21, 28 oktober 2013

1 Stamboom

Op deze wiki pagina vind je een stukje stamboom van een willekeurige familie.

Opgave 1 Vertaal die stamboom, beginnend bij Juliana en Bernhard, in een databank van prolog feiten en regels.

Gebruik feiten voor predicaten `man/1`, `vrouw/1` en `ouder/2`, waarbij je `ouder(X,Y)` interpreteert als `X` is een ouder van `Y`. Je mag je beperken tot voornamen.

Geef nu regels voor de familierelaties `gehuwd/2`, `vader/2`, `moeder/2`, `kind/2`, `zoon/2`, `dochter/2`, `broer/2`, `zuster/2`, `oom/2`, `tante/2`, `neef/2`, `nicht/2`, `grootouder/2`, `grootvader/2`, `grootmoeder/2`, `kleinkind/2`, `kleinzoon/2`, `kleindochter/2`.

Voorzie de definities telkens van een commentaarregel waarin je duidelijk maakt welke interpretatie de predicaten hebben. Bijvoorbeeld (prolog commentaar staat achter het percentteken):

```
% kind/2. kind(X,Y) betekent 'X is een kind van Y'

kind(X,Y) :- ...
```

Let op: je wil niet dat iemand broer (zus, ...) van zichzelf is. Het ingebouwde predicaat `'\='/2` is handig in dit verband: `X\=Y` betekent dat `X` niet unificeerbaar is met `Y`. Zorg wel dat je `'\='/2` op de goede plaats in je regels aanroept.

Opgave 2 Zet de volgende vragen om in prolog queries.

Q1 Wie zijn de tantes van Willem-Alexander?

Q2 Is prinses Alexia een zus van prinses Mabel?

Q3 Wie zijn de kleinkinderen van prins Bernhard?

Q4 Noem de schoondochters van prinses Beatrix.

2 Latijnse vierkanten

Een Latijns vierkant is een $n \times n$ matrix gevuld met n verschillende symbolen waarbij elk symbool precies één keer voorkomt in elke rij en in elke kolom. De populaire 9×9 sudoku is een speciaal geval van een Latijns vierkant, waarbij de 9×9 matrix wordt opgedeeld in 3×3 kwadranten; elk kwadrant bevat de 9 gebruikte symbolen.

Hieronder links een gedeeltelijk ingevuld 3×3 vierkant. De gebruikte symbolen zijn de getallen 1,2,3. Er is precies één manier om het aan te vullen tot een Latijns vierkant.

1		
	2	
		3

$\rightsquigarrow$

1	3	2
3	2	1
2	1	3

Opdracht Schrijf een programma waarmee je voor een symboolset $\{1, 2, 3\}$, alle mogelijke 3×3 Latijnse vierkanten genereert. Hier is het stappenplan.

1. Declareer de te gebruiken symbolen als feiten voor een predicaat `symbool/1`.
2. Definieer een predicaat `verschillend/3` voor legitieme combinaties van drie symbolen: combinaties waarbij elk symbool verschillend is. Je definitie voor `verschillend/3` kan uitsluitend feiten gebruiken. Of je kan een regel geven, waarbij je het ingebouwde `'\='/2` gebruikt. `T1\=T2` slaagt, als termen `T1` en `T2` *niet* unificeerbaar zijn.
3. Geef tenslotte een regel voor `latijns_vierkant/9` waarbij je `verschillend/3` gebruikt. De bedoelde interpretatie voor de argumenten van `latijns_vierkant/9` is zoals hieronder.

A	B	C
D	E	F
G	H	I

$\rightsquigarrow$ `latijns_vierkant(A,B,C,D,E,F,G,H,I)`

Visualisatie Hieronder nog twee predicaten om je vierkanten als een 3 bij 3 matrix op het scherm af te drukken. (Het ingebouwde `format/2` komt later aan de orde. Met `~d` druk je getallen af.)

```
toon_lat :-
    latijns_vierkant(A,B,C,D,E,F,G,H,I),
    format("~d~d~d\n~d~d~d\n~d~d~d",
        [A,B,C,D,E,F,G,H,I]).

toon_lat(A,B,C,D,E,F,G,H,I) :-
    latijns_vierkant(A,B,C,D,E,F,G,H,I),
    format("~d~d~d\n~d~d~d\n~d~d~d",
        [A,B,C,D,E,F,G,H,I]).
```

Bijvoorbeeld

```
?- toon_lat(1,_,_,_,2,_,_,_,3).
132
321
213
```

3 Reizen

Hieronder als feiten voor een predicaat `c/2` een aantal directe verbindingen van het Thalys netwerk, en enkele IC aansluitingen. De aanhalingstekens zorgen ervoor dat de argumenten als constanten worden geïnterpreteerd: zonder die aanhalingstekens zou `Amsterdam` staan voor een variabele.

```
c('Amsterdam','Schiphol').
c('Schiphol','Rotterdam').
```

```
c('Rotterdam','Antwerpen').
c('Antwerpen','Brussel').
c('Brussel','Parijs').
c('Brussel','Gent').
c('Brussel','Zaventem').
c('Brussel','Londen').
c('Brussel','Luik').
c('Parijs','Bergen').
c('Bergen','Charleroi').
c('Charleroi','Namen').
c('Namen','Luik').
c('Gent','Brugge').
c('Brugge','Oostende').
c('Luik','Aken').
c('Aken','Keulen').
c('Utrecht','Rotterdam').
c('Den Haag','Rotterdam').
```

3.1

Definieer een predicaat `reis/3`: `reis(Start,Eind,Stops)` slaagt als je van `Start` naar `Eind` kan reizen met `Stops` tussenstops.

Je definitie heeft een grensgeval (directe verbinding, geen tussenstops) en een recursief geval. Bouw het aantal tussenstops op als een getal in successornotatie: schrijf `0` voor zero, en `s(N)` voor de opvolger van `N`.

HINT Je kan inspiratie opdoen met de definitie voor de `descendant/2` relatie.

3.2

Hieronder de definitie voor optellen voor getallen in successor notatie.

```
som(0,A,A).
som(s(A),B,s(C)) :- som(A,B,C).
```

Definieer een predicaat `reis_via/4`: `reis_via(Via,Start,Eind,Stops)` slaagt als je van `Start` naar `Eind` kan komen via tussenstation `Via` waarbij `Stops` het totale aantal tussenstops van de reis is.

Geef je definitie voor de relatie `reis_via/4` in termen van `reis/3` van de vorige opgave. Gebruik `som/3` om de tussenstops voor de deelreizen op te tellen.

3.3

In het geval van alternatieve reismogelijkheden laat het predicaat `reis/3` je toe een keuze te maken op grond van het aantal tussenstops. Prettiger zou zijn als je ook de trajecten met elkaar zou kunnen vergelijken.

Breid `reis/3` uit tot een definitie voor `reis/4` met een extra argument voor het gevolgde traject. `reis(Start,Eind,Traject,Stops)` slaagt als `Traject` een traject is om van `Start` naar `Eind` te komen met `Stops` tussenstops.

Bouw het traject op met behulp van een predicaat `ga/3`: de term `ga(A,B,einde)` staat voor de laatste stap van een traject; een term `ga(A,B,Traject)`, waarbij `Traject` zelf weer een `ga/3` term is, staat voor een langer traject.

Hieronder een voorbeeldaanroep.

```
?- reis('Utrecht','Luik',Traject,Stops).
Traject = ga('Utrecht', 'Rotterdam', ga('Rotterdam', 'Antwerpen',
             ga('Antwerpen', 'Brussel', ga('Brussel', 'Luik', einde)))),
Stops = s(s(s(0))) .
```

3.4

Voeg ook aan `reis_via/4` een extra argument toe voor het gevolgde traject.

In de eerdere opgave heb je `som/3` gebruikt om de tussenstops voor de deelreizen op te tellen. Definieer nu een predicaat `som_traject/3` waarmee je de deeltrajecten combineert tot het totale traject.

HINT. De `s/1` termen voor getallen in successornotatie en de `ga/3` termen voor trajecten hebben dezelfde recursieve opbouw.

Hieronder enkele voorbeeldaanroepen.

```
?- reis('Rotterdam','Brussel',T,S),reis('Brussel','Aken',T1,S1),
     reis_via('Brussel','Rotterdam','Aken',T2,S2).

T = ga('Rotterdam', 'Antwerpen', ga('Antwerpen', 'Brussel', einde)),
S = s(0),
T1 = ga('Brussel', 'Luik', ga('Luik', 'Aken', einde)),
S1 = s(0),
T2 = ga('Rotterdam', 'Antwerpen', ga('Antwerpen', 'Brussel',
       ga('Brussel', 'Luik', ga('Luik', 'Aken', einde)))),
S2 = s(s(s(0)))
```

□