Hoe werkt het ook alweer in Prolog?

5 maart 2012

1 Basisprincipes

Een Prolog-programma bestaat uit feiten en regels. De regels kunnen worden gebruikt om uit bestaande feiten nieuwe feiten af te leiden.

Prolog is geïnstalleerd op de Linuxomgeving in de computerzalen. Het kan worden gestart met het commando `swipl`. Let op: Prolog is niet geïnstalleerd op Silver (de ssh-server van Liacs). Wel kan je vanaf Silver door-sshën naar een studenten-pc, bijvoorbeeld met `ssh pc306b`. Een bestand laden in Prolog kan met het commando `consult('bestand.pl').` of korter met `[bestand].`. Vergeet de punt niet. Prolog afsluiten kan met het commando `halt.`.

Feiten in Prolog bestaan uit n-plaatsige predicaten. Binnen een predicaat kunnen constantes staan, lijsten, of andere predicaten. Predicaten en constantes beginnen met een kleine letter. Constantes binnen quotes kunnen met een hoofdletter beginnen. Voorbeelden:

```
/* 0-plaatsig */
zonnig.
/* 1-plaatsig */
student(jan).
/* 3-plaatsig */
cijfer(jan,'Kunstmatige Intelligentie',7).
/* 2-plaatsig, 1-plaatsig predicaat als 2e argument */
tevreden(jan,minimaal(7.5)).
/* 2-plaatsig, lijst als 2e argument */
cijfers(jan,[8,7,6.5,8]).
```

Regels bestaan uit een conclusie gevolgd door 1 of meerdere voorwaarden. De voorwaarden worden gescheiden door een komma, de conclusie en de voorwaarden zijn gescheiden door `:-`. Voorbeelden:

```
/* Als de zon schijnt (voorwaarde) is het mooi weer (conclusie) */
mooiweer :- zonnig.
/* Als Jan student is heeft hij college */
college(jan) :- student(jan).
/* Als iemand student is heeft hij college (variabele X) */
college(X) :- student(X).
```

Predicaten in Prolog hebben vaak een recursieve definitie. Recursie kan worden gebruik om door een lijst heen te lopen. Hiervoor wordt de lijstoperator | gebruikt, deze verdeelt een lijst in losse elementen voor de | (meestal 1 los element) en de rest van de lijst na de |. Een lijst `[1,2,3,4]` kan worden gebruikt als `[H|T]`, dan geldt `H=1` en `T=[2,3,4]`. Voorbeelden:

A is een voorouder van B als A de vader is van B (basisgeval) of als X de vader is van B en A is een voorouder van X. Hier: Henk (A) is een voorouder van Irene (B) als Jan (X) de vader is van Irene en Henk is een voorouder van Jan.

```
vader(henk,klaas).
vader(klaas,jan).
vader(jan,irene).

voorouder(A,B) :- vader(A,B).

voorouder(A,B) :-
vader(X,B),
voorouder(A,X).
```

De som van een lijst is het eerste getal + de som van de rest van de lijst. De som van de lege lijst is 0.

```
som(0,[]).

som(X,[H|T]):-
som(Y,T),
X is H+Y.
```

Variabelen kunnen op elke positie een waarde krijgen, ook in de conclusie van een regel. Hier kan je gebruik van maken om regels korter en overzichtelijker op te schrijven. Voorbeelden:

```
studie(jan,informatica).
studie(marie,wiskunde).
studie(klaas,wiskunde).
studie(klaas,informatica).
studie(anna,informatica).
studie(anna,wiskunde).
```

Het predicaat dubbel(X) wordt gebruikt om te kijken wie twee studies volgt (hier: Klaas en Anna). In het onderstaande voorbeeld wordt twee keer het feit studie(Student,Studierichting) gematcht waarbij de student gelijk moet zijn (A=B).

```
dubbel1(X) :-
```

```
        studie(A,S1),
        studie(B,S2),
        S1='informatica',
        S2='wiskunde',
        A=B,
        X=A.
```

Dit kan korter door als tweede argument van studie(Y,Z) constantes te gebruiken in plaats van variabelen.

```
dubbel2(X) :-
        studie(A,informatica),
        studie(B,wiskunde),
        A=B,
        X=A.
```

Nog korter door twee keer dezelfde variabele A te gebruiken. Na de eerste regel studie(A,informatica) wordt A gelijk gesteld aan de eerste constante in de eerste vermelding van studie(_,informatica). Dit is 'jan'. In de tweede regel wordt dan gezocht naar studie(jan,wiskunde). Dit wordt niet gevonden, dus wordt de eerste regel opnieuw gedaan. Nu wordt A gelijk aan 'klaas'. In de tweede regel wordt nu gezocht naar studie(klaas,wiskunde). Dit wordt wel gevonden, dus klaas is een goed antwoord.

```
dubbel3(X) :-
        studie(A,informatica),
        studie(A,wiskunde),
        X=A.
```

Nog korter: gebruik de variabele voor het matchen van studie() ook in de conclusie.

```
dubbel4(X) :-
        studie(X,informatica),
        studie(X,wiskunde).
```

Het gemiddelde cijfer is het totaal van alle cijfers delen door het aantal cijfers. Het totaal van alle cijfers in een lijst is het cijfer vooraan de lijst + het totaal van de rest van de lijst (recursieve definitie). Het totaal van een lijst met 1 cijfer is dat cijfer (basisgeval). Het aantal cijfers in een lijst is 1 voor het eerste element + het aantal cijfers in de rest van de lijst.

```
/* Lijst met cijfers */
resultaten(anna,[res('PM',8),res('FI1',7),res('Logica',8)]).

gemiddelde_f(Naam,Gem) :-
        resultaten(Naam,L),
        vakkenlijst_f(Totaal,Aantal,L),
        Gem is Totaal/Aantal.
```

Basisgeval: het totaal van een lijst met 1 element is het cijfer in dat element. Het aantal is 1.

```
vakkenlijst_f(Totaal,Aantal,L) :-
L=[res(_,Cijfer)],
Aantal=1,
Totaal=Cijfer.

/* Recursiegeval. */
vakkenlijst_f(Totaal,Aantal,L) :-
        L=[H|T],
        vakkenlijst_f(Totaal1,Aantal1,T),
        H=res(_,Cijfer),
        Totaal is Totaal1+Cijfer,
        Aantal is Aantal1+1.

/* Kortere versie */
gemiddelde(Naam,Gem) :-
        resultaten(Naam,L),
        vakkenlijst(Totaal,Aantal,L),
        Gem is Totaal/Aantal.
```

Basisgeval, korter opgeschreven. De laatste aanroep in de recursie, als de rest van de lijst nog maar 1 element bevat, is
vakkenlijst(Totaal1,Aantal1,res('Logica',8)). De eerste twee variabelen liggen dus nog niet vast, en de derde is compleet bepaald. Bij het matchen van het basisgeval worden eerst de bekende variabelen ingevuld. In dit geval is dat res(_,Cijfer), waarbij Cijfer wordt vastgelegd als 8 (de _ wordt altijd buiten beschouwing gelaten). Nu de waarde van Cijfer bekend is kan de eerste variabele van het basisgeval ook worden toegekend. De tweede variabele (Aantal1) wordt gebonden aan de constante 1 in het basisgeval. Nu kunnen in het recursiegeval de twee variabelen worden gebruikt in de optelling.

```
vakkenlijst(Cijfer,1,[res(_,Cijfer)]).
```

Recursiegeval, korter opgeschreven. De lijstvariabele L bestaat uit het voorste resultaat (res(_,Cijfer)) en de rest van de lijst T. Dit kan al in de conclusie worden aangegeven zodat de lijst L niet meer in de voorwaarden hoeft te worden geanalyseerd.

```
vakkenlijst(Totaal,Aantal,[res(_,Cijfer)|T]) :-
        vakkenlijst(Totaal1,Aantal1,T),
        Totaal is Totaal1+Cijfer,
        Aantal is Aantal1+1.
```