Herkansing Prolog
24 augustus 2007

De eindbeoordeling is "het aantal punten + 10" gedeeld door 10.

Vraag:	1	2a	2b	3a	3b	3c	3d	4	5a	5b
Punten:	10	10	10	10	10	5	5	20	5	5

Vraag 1: termen en lijsten

Geef voor elk van de volgende goals aan of ze slagen of falen. Geef bij slagende goals aan wat de resulterende variabele-binding is, geef bij falende goals een korte uitleg.

1. `?- [X,Y,Z] = [a,b,c].`
2. `?- [kat] = [X|Y].`
3. `?- [X,Y | Z] = [aap, noot , mies].`
4. `?- [[a, Y] | Z] = [[X, 2], [b, 3]]`
5. `?- [a | T] = [a, b].`
6. `?- f(1,2)=X(Y,Z).`
7. `?- Y = 12, X is 2 + Y.`
8. `?- X is 12, X =:= 6 + 6.`
9. `?- f(A,B) == f(X,Y).`
10. `?- X = 12, X = 4 + 8.`

Vraag 2: Programma executie

Beschouw het onderstaande programma:

```
cons(a).
cons(b).
term(X) :- cons(X).
term(g(X)) :- term(X).
term(f(X,Y)) :-term(X), term(Y).
```

(a) Geef de afleidingsboom voor de eerste vier antwoorden van de query `?:- term(f(X,Y)).` Geef daarbij duidelijk aan welke clauses gebruikt worden, en welke variabele bindingen gedaan worden, en de volgorde waarin dit gebeurt.

(b) Geef twee voorbeelden van veranderingen in bovenstaande code waardoor de procedurele betekenis van het programma verandert, maar de declaratieve betekenis gelijk blijft.

Opgave 3: Prolog executie en cuts

Beschouw de volgende code:

```
atoms(A,T) :- atomic(T), !, A=T.
atoms(A,T) :- T =.. L, member(T1,L), atoms(A,T1).
```

(a.) Leg uit wat de functie van deze code is. Beschrijf het gedrag van de code en de resulterende uitvoer voor de aanroep atoms(X, f(a,b,g(c)).

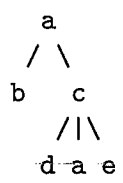
(b.) Wat verandert er aan de werking en de uitvoer van het bovenstaande programma als de cut in de eerste clause verwijderd wordt?

(c.) Wat verandert er aan de werking en de uitvoer van het bovenstaande programma als naast de cut in de eerste clause ook een cut na de aanroep van member in de tweede clause wordt toegevoegd?

(d.) Geef aan van de twee cuts uit opgave b en c of ze zogeheten red of green cuts zijn, en motiveer je antwoord.

Vraag 4: data-structuren en recursie

Ontwerp een data-structuur voor bomen. In zulke bomen heeft elke knoop een waarde, en kunnen knopen een willekeurig aantal kinderen hebben. Schrijf vervolgens een procedure om de waarden van alle knopen in een gegeven boom te verzamelen in een lijst. Als een waarde bij meerdere knopen voorkomt mag hij maar één keer in de berekende lijst voorkomen. Voorbeeld: voor de boom



zou de berekende lijst `[a,b,c,d,e]` kunnen zijn (of deze elementen in een andere volgorde, want de volgorde van de berekende lijst maakt niet uit).

Vraag 5: Zoektechnieken

De kracht van Prolog is goed te zien aan de eenvoud van de implementatie van depth-first search:

```
solve(Node,[Node]) :-
  goal(Node).
solve(Node,[Node|Path]) :-
  step(Node,Next),
  solve(Next,Path).
```

(a) Eén van de nadelen van depth-first search is dat het verstrikt kan raken in een loop in de zoek-ruimte. Verander bovenstaande code zodat er een versie van depth-first search ontstaat die niet verstrikt raakt in loops in de zoekruimte.

(b) Een ander nadeel van depth-first search is dat het verloren kan raken in oneindige paden in de zoekruimte. Een nogal brute maar eenvoudige oplossing hiervoor is om een versie van depth-first search te maken die maar tot aan een op te geven maximale diepte zoekt. Geef de code voor zo'n aangepaste versie van depth-first search.