Prolog. Gekwoteerde zitting.

Patrick De Causmaecker

29/3/2023, 9:00 - 12:00

Er zijn drie vragen. Het symbool (*) duidt een (deel van een) vraag aan die je als moeilijker kan beschouwen en eventueel uitstellen voor het geval er op het einde tijd over is. Je kan de documentatie bij SWI-Prolog gebruiken en de slides. Andere resources op het internet zijn niet toegelaten. E-mail en andere communicatie-toepassingen worden uitgeschakeld.

Als je klaar bent verwittig je de toezichter. Die zal je toelaten om je e-mail op te starten. Zend dan je antwoorden naar `patrick.decausmaecker@kuleuven.be`

1 Collatz (1/3)

De rij van Collatz, ook well de rij $3n+1$, is een rij die voor elk natuurlijkgetal de opeenvolgende getransformeerden berekent volgens de functie

$$C(n) = \begin{cases} 3n+1 & \text{if } n \text{ is odd} \\ n \; div \; 2 & \text{if } n \text{ is even} \end{cases} \tag{1}$$

Het vermoeden van Collatz stelt dat elke rij op een bepaald ogenblik de waarde 1 bereikt, waarna een oneindige lus $1, 4, 2, 1, 4, 2, 1, \ldots$ ontstaat. Gevraagd is

1. Een predicaat (0.2/3)

   $collatz(+N : integer, -M : integer)$

   dat slaagt als M het aantal stappen in de Collatz rij is waarna, startend bij N, voor het eerst de waarde 1 wordt bereikt.

   ```
   ?- collatz(1,M).
   M = 0.
   ?- collatz(1271298765,M).
   M = 196.
   ```

2. Een predicaat (0.4/3)

   $collatz(-N : integer, +M : integer)$

   dat slaagt zoals $collatz(+N : integer, -M : integer)$, maar toelaat dat M gegeven wordt en dat dus slaagt voor alle getallen N waarvoor het aantal stappen naar 1 gelijk is aan M.

   ```
   ?- collatz(N,3).
   N = 8 ;
   false.
   ?- collatz(N,4).
   N = 16 ;
   false.
   ?- collatz(N,5).
   N = 32 ;
   N = 5 ;
   ```

```
false.
?- collatz(N,6).
N = 64 ;
N = 10 ;
false.
?- collatz(N,7).
N = 128 ;
N = 21 ;
N = 20 ;
N = 3 ;
false.
?- time(findall(N,collatz(N,17),L)),sort(L,LS),length(LS,LL),print((LS,LL)),nl,fail.
% 770 inferences, 0.000 CPU in 0.000 seconds (93% CPU, 2692308 Lips)
[14,15,88,90,92,93,544,552,554,560,564,565,600,602,604,605,3072,3328,
3392,3408,3412,3413,3616,3624,3626,3632,3636,3637,20480,21504,21760,
21824,21840,21844,21845,131072],36
false.
?- time((findall(N,collatz(N,25),L),sort(L,LS))),length(LS,LL),print((LS,LL)),nl,fail.
% 5,023 inferences, 0.001 CPU in 0.002 seconds (93% CPU, 3423995 Lips)
[98,99,100,101,102,576,592,596,597,608,616,618,625,642,643,648,650,652,653,662,663,713,715,
...
5570560,5586944,5591040,5592064,5592320,5592384,5592400,5592404,5592405,33554432],227
false
```

3. Een memoïzerende versie van $collatz(+N,-M)$ (0.4/3). Verwacht wordt een versnelling gelijkaardig aan die hieronder.

```
?- member(N,[1,11,111,1111,2222,222,22,2,1,11,111,1111]),time(collatz(N,M)).
% 2 inferences, 0.000 CPU in 0.000 seconds (47% CPU, 285714 Lips)
N = 1,
M = 0 ;
% 85 inferences, 0.000 CPU in 0.000 seconds (60% CPU, 1049383 Lips)
N = 11,
M = 14 ;
% 368 inferences, 0.001 CPU in 0.001 seconds (74% CPU, 603279 Lips)
N = 111,
M = 69 ;
% 104 inferences, 0.000 CPU in 0.000 seconds (70% CPU, 622754 Lips)
N = 1111,
M = 31 ;
% 8 inferences, 0.000 CPU in 0.000 seconds (71% CPU, 210526 Lips)
N = 2222,
M = 32 ;
% 8 inferences, 0.000 CPU in 0.000 seconds (72% CPU, 210526 Lips)
N = 222,
M = 70 ;
% 2 inferences, 0.000 CPU in 0.000 seconds (59% CPU, 86957 Lips)
N = 22,
M = 15 ;
% 2 inferences, 0.000 CPU in 0.000 seconds (60% CPU, 83333 Lips)
N = 2,
M = 1 ;
% 1 inferences, 0.000 CPU in 0.000 seconds (63% CPU, 32258 Lips)
N = 1,
M = 0 ;
% 2 inferences, 0.000 CPU in 0.000 seconds (64% CPU, 68966 Lips)
```

```
N = 11,
M = 14 ;
% 2 inferences, 0.000 CPU in 0.000 seconds (60% CPU, 90909 Lips)
N = 111,
M = 69 ;
% 2 inferences, 0.000 CPU in 0.000 seconds (63% CPU, 62500 Lips)
N = 1111,
M = 31.
```

2 Hamiltionian Cycle (1/3)

Een Hamiltonian Cycle in een gerichte grafe is een opeenvolging van vertices waarbij elke vertex in de grafe verbonden is met zijn voorganger, waar alle vertices in juist een keer in voorkomen en waarbij de laatste vertex verbonden is met de eerste. In de voorbeelden hieronder is de grafe voorgesteld door een lijst van gerichte bogen $b(v_1, v_2)$. Gevraagd is

1. Een predicaat (0.5/3)

   $ham(+Grafe : list, -Pad : list)$

   dat slaagt als Pad een lijst is van vertices die een hamiltonian pad in Grafe voorstelt dat begint bij de kleinste vertex in Grafe volgens de canonieke sortering van termen in Prolog. Bijvoorbeeld

   ```
   ?- ham([b(1,2),b(2,3),b(3,4),b(4,1),b(1,3),b(3,2),b(2,4)],P).
   P = [1, 2, 3, 4] ;
   P = [1, 3, 2, 4] ;
   ```

2. (*) Een predicaat (0.5/3)

   $ham2(+Grafe : list, -Pad : list)$

   dat slaagt zoals *ham* maar dat gebruik maakt van $freeze/2$ om de beperkingen van de hamiltonian cycle op te leggen alvorens de variabelen die de vertices voorstellen geünificeerd worden met werkelijke vertices in een permutatie predicaat. Op het kleine voorbeeld:

   ```
   ?- time(ham([b(1,2),b(2,3),b(3,4),b(4,1),b(1,3),b(3,2),b(2,4)],P)).
   % 179 inferences, 0.000 CPU in 0.000 seconds (78% CPU, 2712121 Lips)
   P = [1, 2, 3, 4] ;
   % 145 inferences, 0.000 CPU in 0.000 seconds (61% CPU, 1907895 Lips)
   P = [1, 3, 2, 4] ;
   % 62 inferences, 0.000 CPU in 0.000 seconds (75% CPU, 1291667 Lips)
   ?- time(ham2([b(1,2),b(2,3),b(3,4),b(4,1),b(1,3),b(3,2),b(2,4)],P)).
   % 197 inferences, 0.000 CPU in 0.000 seconds (76% CPU, 2010204 Lips)
   P = [1, 2, 3, 4] ;
   % 132 inferences, 0.000 CPU in 0.000 seconds (47% CPU, 1128205 Lips)
   P = [1, 3, 2, 4] ;
   % 71 inferences, 0.000 CPU in 0.000 seconds (78% CPU, 1420000 Lips)
   ```

3 Matrices(1/3)

Een matrix bestaat uit een aantal rijen en kolommen. De transpositie van een matrix ontstaat door van de rijen kolommen te maken en omgekeerd. Gevraagd is

1. Drie predicaten (0.5/3), $matrix(+L * K : integer * integer, -M : list)$ dat slaagt als M geunificeerd kan worden met een matrix van L rijen en K kolommen, $transpose\_matrix(+M : list, -MT : list)$ en $transpose\_matrix(-M : list, +MT : list)$, die slagen als MT de getransponeeerde matrix is van M. Bijvoorbeeld

```
?- matrix(3*5,M).
M = [[_, _, _, _, _], [_, _, _, _, _], [_, _, _, _, _]].
?- transpose_matrix([[1,2],[3,4],[5,2],[6,4]],C).
C = [[1, 3, 5, 6], [2, 4, 2, 4]].
?- transpose_matrix(C,[[1,2],[3,4],[5,2],[6,4]]).
C = [[1, 3, 5, 6], [2, 4, 2, 4]].
?- matrix(3*5,M),transpose_matrix(M,C).
M = [[_A, _B, _C, _D, _E], [_F, _G, _H, _I, _J], [_K, _L, _M, _N, _O]],
C = [[_A, _F, _K], [_B, _G, _L], [_C, _H, _M], [_D, _I, _N], [_E, _J, _O]].
```

2. Een predicaat (0.2/3), $show(+X, +M : list)$ dat een matrix M op het scherm afdrukt, met X in geval een variabele niet werd geunificeerd. Bijvoorbeeld

```
?- matrix(3*4,M),show(x,M).
xxxx
xxxx
xxxx
M = [[_, _, _, _], [_, _, _, _], [_, _, _, _]] ;
```

3. (*) Een predicaat (0.3/3). $solve\_binary(M)$ dat slaagt als de matrix M enkel waarden 0 en 1 bevat en voldoet aan.

   - op elke rij en in elke kolom bevinden zich evenveel waarden 0 als waarden 1
   - in elke rij en in elke kolom komen nooit meer dan twee gelijke opeenvolgende waarden voor

   In het voorbeeld gebruiken we predicaten $binary\_x/1$ die slagen voor een gedeeltelijk ingevulde matrix.

```
?- binary_0(M),show(x,M),solve_binary(M),show(x,M).
11xx
1x1x
xx0x
x0x1

1100
1010
0101
0011

M = [[1, 1, 0, 0], [1, 0, 1, 0], [0, 1, 0, 1], [0, 0, 1, 1]];
?- binary_1(M),show(x,M),solve_binary(M),show(x,M).
11xxxx
1x1xx1
x11xxx
1xxxx1
xxxxx1
xxxxxx

110100
101001
011010
100101
001011
010110

M = [[1, 1, 0, 1, 0, 0], [1, 0, 1, 0, 0, 1], [0, 1, 1, 0, 1, 0], [1, 0, 0, 1, 0|...], ...
```