CSU34011 Symbolic Programming

Second of Two Assessed Assignments

Submit to Blackboard by Nov 24 (Wed)

Problem 1 Exercise 6.6 in Learn Prolog Now describes a street with

($*$) three neighbouring houses that all have a different colour, namely red, blue, and green. People of different nationalities live in the different houses and they all have a different pet.

Leaving out all the other constraints mentioned in that exercise, write a DCG that outputs strings

```
[h(Col1,Nat1,Pet1), h(Col2,Nat2,Pet2), h(Col3,Nat3,Pet3)]
```

satisyfing ($*$), where the nationalities are

`english, spanish, japanese`

and the pets are

`jaguar, snail, zebra`.

Use the predicate `nbd/2` for the 3 houses so that for example,

```
?- nbd([h(red,english,snail), h(blue,japanese,jaguar),
       h(green,spanish,Z)], []).
Z = zebra ;
false.
```

[20 marks]

Problem 2 For an integer $n \geq 0$, the nth *Fibonacci number* F_n is defined as follows

$$\begin{aligned} F_0 &:= 0 \\ F_1 &:= 1 \\ F_{n+2} &:= F_n + F_{n+1} \end{aligned}$$

giving $F_2 = 1$, $F_3 = 2$, $F_4 = 3$, $F_5 = 5$, etc. Define a DCG that generates for every $k \geq 1$, lists $[F_0, F_1, \ldots, F_k]$ so that, for example,

```
?- fib(L,[]).
L = [0,1] ;
L = [0,1,1] ;
L = [0,1,1,2] ;
L = [0,1,1,2,3] ;
L = [0,1,1,2,3,5] ;
...
```

[20 marks]

Problem 3 For each integer $n > 0$, let

$$L_n \;:=\; \{s \in \{0,1\}^+ \mid s \text{ ends in a string from } 1(0+1)^{n-1}\}$$

be the set of bit-strings whose n-th to the last bit is 1. That is, L_n is described by the regular expression

$$(0+1)^*1(0+1)^{n-1}.$$

(a) Define predicates `tran/4` and `final/2` so that for every integer $n > 0$,

(i) `tran(`n`,Q1,X,Q2)` picks out transitions between states `Q1` to `Q2` labeled by `X`, and

(ii) `final(`n`,Q)` picks out final states `Q`

for a finite automaton with initial state `q0` accepting L_n according to

```
accept(N,String) :- steps(N,q0,String,Q), final(N,Q).
steps(_,Q,[],Q).
steps(N,Q1,[H|T],Q2) :- tran(N,Q1,H,Q), steps(N,Q,T,Q2).
```

For example,

```
?- accept(3,L).
L = [1, 0, 0] ;
L = [1, 0, 1] ;
L = [1, 1, 0] ;
L = [1, 1, 1] ;
L = [0, 1, 0, 0] ;
L = [0, 1, 0, 1] ;
...
```

[20 marks]

(b) Define a DCG for the 3-ary predicate `s/3` such that `s(`n, s`,[])` is true exactly if s encodes a string in L_n. For example,

```
?- s(3,[A,1,Z],[]).
A = 1, Z = 0;
A = 1, Z = 1;
false.
```

[20 marks]

(c) Define predicates `ith/3` and `initial/3` such that if ai is the i^{th} string returned by the query `s(`n`,X,[])` then `ith(`i, n`,Z)` is true exactly if `Z =` ai

and

`initial(`i, n`,Z)` is true exactly if `Z = [`$ai, \ldots, a2, a1$`]`.

For example, assuming

```
?- s(3,X,[]).
X = [1,0,0] ;
X = [1,0,1] ;
X = [1,1,0] ;
X = [1,1,1] ;
X = [0,1,0,0] ;
...
```

then

```
?- ith(5,3,A).
A = [0,1,0,0].
```

and

```
?- initial(5,3,L).
L = [[0,1,0,0], [1,1,1], [1,1,0], [1,0,1], [1,0,0]].
```

[20 marks]