Examen Prolog — ESSI2

(Documents autorisés: *tous*)

Michel Rueher

Mai 1998

1. (3 points) Ecrire en `prolog` le prédicat défini par la clause:

   $(gp(X,Z) \vee \neg p(Y,Z) \vee \neg p(X,Y)) \wedge (gp(X,Z) \vee \neg p(Y,Z) \vee \neg m(X,Y))$

2. (3 points) Expliquer pourquoi la coupure ci-dessous —appelée *red cut*— est dangereuse.

```
min(X,Y,X) :-  Y  ≥ X, !.
min(X,Y,Y).
```

3. (3 points) Expliquer quel est l'avantage du prédicat `len1/2` par rapport au prédicat `len2/2` ci-dessous.

```
len1(L,N) :- len1(L,0,N).
len1([],N,N).
len1([H|T],N0,N) :- N1 is N0 +1,
                    len1(T,N1,N).
```

```
len2([],0).
len2([X|L],N) :- len2(L,N0),
                 N is N0 + 1.
```

4. (3 points) Expliquer pourquoi le prédicat `tri/2` ci-dessous ne réalise pas un tri sur place.

```
tri([],[]).
tri([H|L],T):-
        p(L,H,P1,P2),
        tri(P1,T1),
        tri(P2,T2),
        append(T1,[H|T2],T).
% append/3: prédicat de concaténation de la bibliothèque Sicstus
            (fonctionne comme  conc/3)
```

```
p([],_,[],[]).
p([E1|L],H,[E1|P1],P2):- E1 @=< H,
                         p(L,H,P1,P2).
p([E1|L],H,P1,[E1|P2]):- E1 @> H,
                         p(L,H,P1,P2).
```

5. (3 points) Problème du zèbre. Indiquer comment il faut compléter le prédicat `relation/5` ci-dessous si on remplace `C=[_,bleu,_,_,_]` par `C=[_,_,_,_,_]` .

```
relation(C,N,F,B,A):-
          C=[_,bleu,_,_,_], N =[norvegien,_,_,_,_],
          B= [_,_,lait,_,_], A= [_,_,_,_,_],
          meme_position(rouge,anglais,C,N),
```

```
meme_position(vert,cafe,C,B),
meme_position(jaune,kool,C,F),
voisin_droit(vert,blanc,C),
meme_position(espagnol,chien,N,A),
meme_position(ukrainien,the,N,B),
meme_position(japonais,craven,N,F),
meme_position(old_gold,escargot,F,A),
meme_position(gitane,vin,F,B),
voisin(chesterfield,renard,F,A),
voisin(kool,cheval,F,A).
```

5. (5 points) Soit une expression booléenne construite à l'aide:

   - de variables booléennes instanciées à `true` ou `false` (représentées par les constantes 1 et 0);
   - des opérateurs binaires `et` et `ou` (représentés par les caractères * et +);
   - de l'opérateur unaire de négation (représenté par le caractère -).

   (a) Donner la `DCG` qui permet d'analyser ce type d'expression booléenne;

   (b) Donner le programme `prolog` généré à partir de la `DCG` par l'interprète `Sicstus`;

   (c) Donner la `DCG` qui permet d'analyser et d'évaluer simultanément ce type d'expression booléenne;

   Exemple:

```
> eval_bool(Z,''0*1+-0'',[]).
> Z=0
> eval_bool(Z,''1+-1'',[])
> Z=1
```