



Université d'Ottawa • University of Ottawa

Faculté de Génie – EECS

CSI2520 : PARADIGMES DE PROGRAMMATION

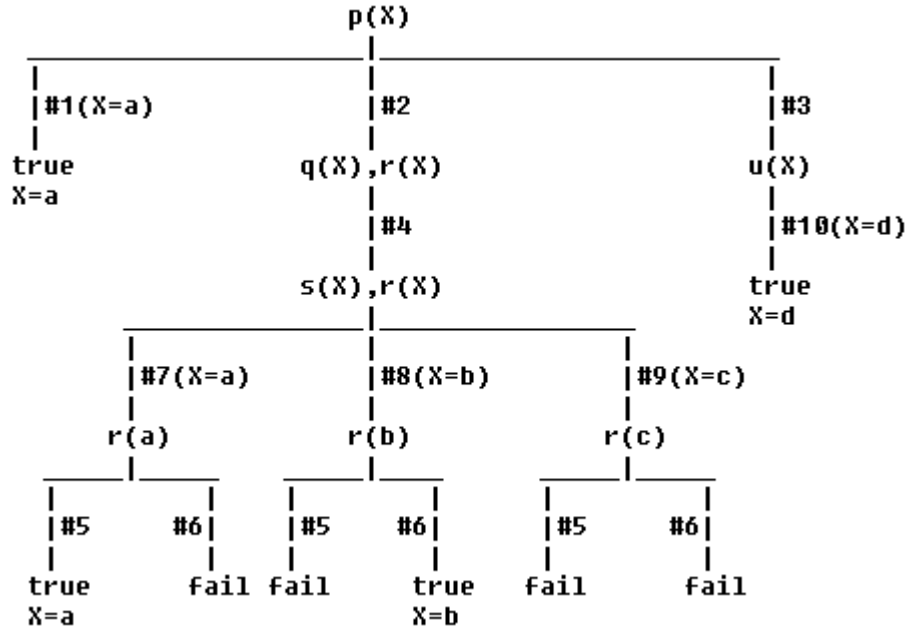
Solution Tutorat 3

Exercice 1

Soit la base de faits et prédicats suivants :

p(a).	/* #1 */
p(X) :- q(X), r(X).	/* #2 */
p(X) :- u(X).	/* #3 */
q(X) :- s(X).	/* #4 */
r(a).	/* #5 */
r(b).	/* #6 */
s(a).	/* #7 */
s(b).	/* #8 */
s(c).	/* #9 */
u(d).	/* #10 */

1. Tracer l'arbre de recherche de P(X)



+

2. Tester les requêtes suivantes:

?- p(X), !.

X=a ;

no

?- r(X), !, s(Y).

X=a Y=a ;

X=a Y=b ;

X=a Y=c ;

no

?- r(X), s(Y), !.

X=a Y=a ;

no

Exercice 2

Soit la base de faits et prédicats suivants :

part(a). part(b). part(c).

red(a). black(b).

color(P, red) :- red(P), !.

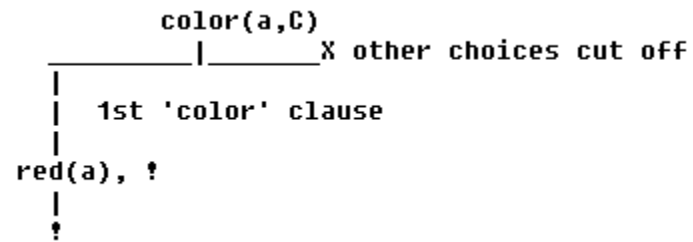
color(P, black) :- black(P), !.

color(P, unknown).

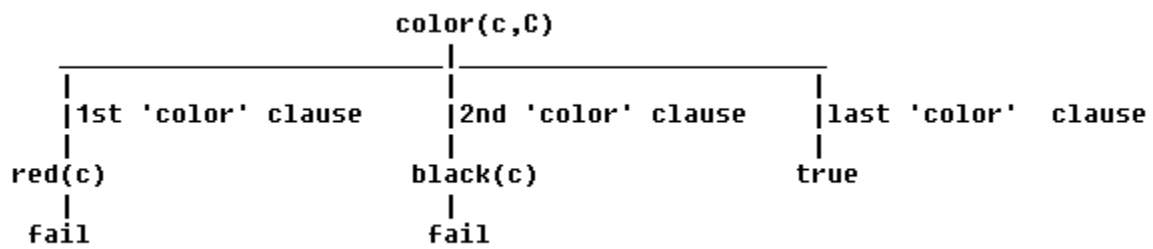
1. Déterminer le résultat des requêtes suivantes en dessinant l'arbre de recherche de chaque requête

?- color(a,C).

C = red



```
?- color(c,C).
C = unknown
```



Exercice 3

Soit la base de faits et prédicats suivants :

```
s(X,Y):- q(X,Y).
s(0,0).
q(X,Y):- i(X), j(Y).
i(1).
i(2).
j(1).
j(2).
j(3).
```

1. Déterminer le résultat de la requête suivante en dessinant l'arbre de recherche.

```
?- s(X,Y).
```

```
X = 1
Y = 1 ;
```

```
X = 1
Y = 2 ;
```

```
X = 1
Y = 3 ;
```

```
X = 2
Y = 1 ;
```

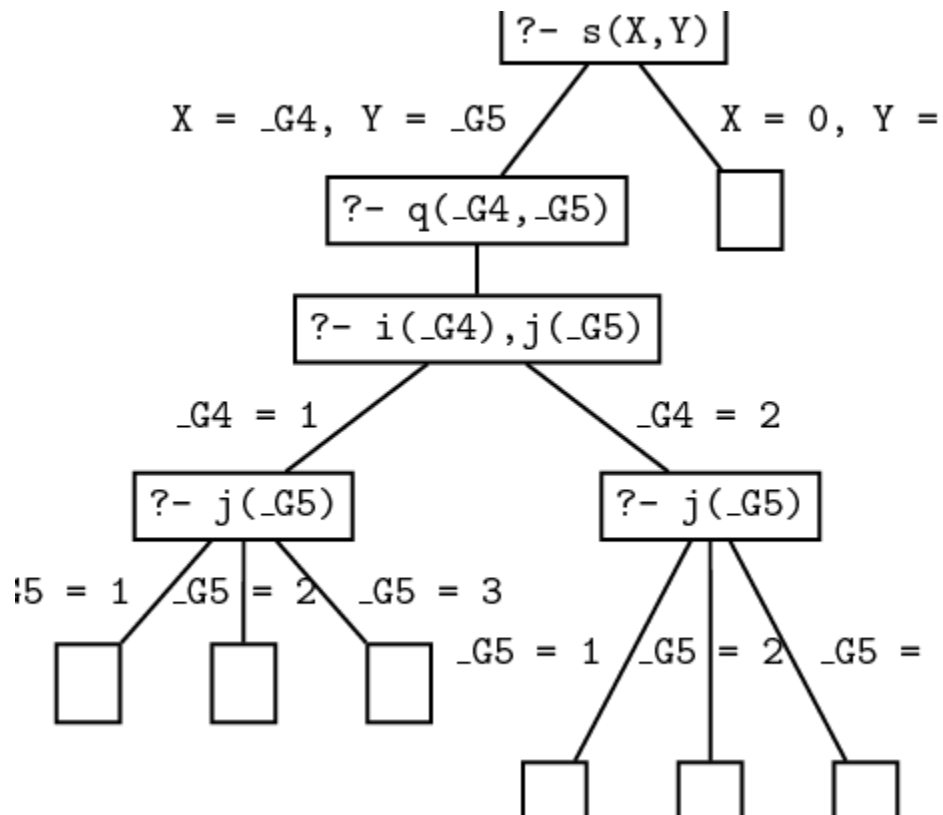
```
X = 2
Y = 2 ;
```

```
X = 2
Y = 3 ;
```

```

X = 0
Y = 0;
no

```



2. Si on définit le prédicat :

```
q(X,Y):- i(X), !, j(Y).
```

Quel sera le résultat de la requête:

```
?- s(X,Y).
```

```

X = 1
Y = 1 ;

```

```

X = 1
Y = 2 ;

```

```

X = 1
Y = 3 ;

```

```

X = 0
Y = 0;
no

```

Exercice 4

Soit la base de faits suivante:

```
p(a).  
p(b).  
q(c).
```

1. Quel est le résultat des requêtes suivantes:

```
?-not p(X), q(X). % fails
```

```
?-q(X), not p(X). % succeeds with X=c
```

Exercice 5

Soit la base de faits et prédicats suivants :

```
sad(X) :- \+ happy(X).  
happy(X) :- beautiful(X), rich(X).  
rich(bill).  
beautiful(michael).  
rich(michael).  
beautiful(cinderella).
```

1. Quel est le résultat des requêtes suivantes:

```
?- sad(bill).
```

Yes

```
?- sad(cinderella).
```

Yes

```
?- sad(michael).
```

No

```
?- sad(jim).
```

Yes

```
?- sad(Someone).
```

No

Exercice 6

Comment peut-on représenter "Colbert n'aime pas les ours (indépendamment de ce qu'il aime d'autre)." ?

1. Une solution: Ajouter "not(ours(X))" à chaque règle qui décrit ce que Colbert aime.

Par exemple:

```
aime(colbert, X) :- animal(X), not(ours(X)).
aime(colbert, X) :- jouet(X), not(ours(X)).
aime(colbert, X) :- vitEnArctique(X), not(ours(X)).
...
```

2. Essayons d'utiliser l'échec (*fail*) comme alternative.

Premier essai:

```
ours(yogi).
animal(yogi).
aime(colbert, X) :- ours(X), fail.
aime(colbert, X) :- animal(X).
```

Quel est le résultat de la requête:

```
?- aime(colbert, yogi).
Yes
```

Vous remarquez qu'on doit ajouter une coupe (*cut*) pour empêcher les autres règles d'être essayées après que la première règle arrive à un échec.

Deuxième essai:

```
ours(yogi).
chat(tom).
animal(yogi).
animal(tom).
aime(colbert, X) :- ours(X), !, fail.
aime(colbert, X) :- animal(X).
```

3. Quel est le résultat des requêtes suivantes:

```
?- aime(colbert, yogi).
No
?- aime(colbert, tom).
Yes
?- aime(colbert, X).
No
```

4. Quel est l'inconvénient de cette solution ?

Cette solution marche seulement lorsque X est instancié.

Exercice 7

1. Qu'affichera la requête:

```
?- boucle(10).
```

pour le programme suivant et expliquer le fonctionnement:

```
entier(0).  
entier(X):- entier(Y), X is Y+1.  
boucle(N) :- repeat, entier(X), writeln(X), X>=N, !.
```

```
?- boucle(10).
```

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
true.
```

Exercice 8

Soit le programme Prolog suivant:

```
q(a).  
q(b).  
s(a).  
s(e).  
s(f).  
s(g).  
k(e).  
k(f).  
u(e).  
u(g).  
p(X,X,c):-q(X).  
p(X,Y,Z):-r(X,Y,Z),s(Z).  
r(X,Y,Z):-t(X,Z), u(Y).  
t(X,Z):-k(X),k(Z).
```

a)

Dessiner l'arbre de résolution complet généré par la requête:

```
?-p(X,Y,Z).
```


en montrant bien les solutions qui seront trouvées.

b)

Ajouter une et une seule coupe (!) au programme ci-haut de façon à obtenir les solutions suivantes:

```
t(X,Z):-k(X),!,k(Z).
?-p(X,Y,Z).
X = Y, Y = a,
Z = c ;
X = Y, Y = b,
Z = c ;
X = Y, Y = Z, Z = e ;
X = Z, Z = e,
Y = g.
```

c)

Ajouter une et une seule coupe (!) au programme ci-haut de façon à obtenir seulement la solution suivante:

```
p(X,X,c):-q(X),!.
?-p(X,Y,Z).
X = Y, Y = a,
Z = c.
```

Exercice 9

Écrire un programme Prolog interactif où l'utilisateur entre un nombre entier positif et est alors affiché la valeur 2 à la puissance de ce nombre. Puis un nouveau nombre est demandé. Si un nombre négatif est entré alors le programme s'arrête.

```
start:-repeat,    % loop forever
read(X),          % read number
power2(X,Y),      % our evaluation
write(Y), nl,     % output result
Y = end, !.       % stopping condition
```

```
power2(X,Y):- X<0, Y=end,!.
power2(X,Y):- pow(2,X,Y).
pow(X, 1, X).
pow(X, Y, Z):- Y > 1, Y1 is Y-1,pow(X, Y1, Z1), Z is X * Z1.
```