CENTRE UNIVERSITAIRE DE MILA
INSTITUT DES SCIENCES ET DE LA TECHNOLOGIE

Module: Intelligence Artificielle **Enseignante:** M. BOUZAHZAH

LA COUPURE

I- Définition

- La coupure est aussi appelée "cut", est notée ! . C'est un prédicat sans signification logique (la coupure n'est ni vraie ni fausse). Elle est utilisée pour "couper" des branches de l'arbre de recherche.

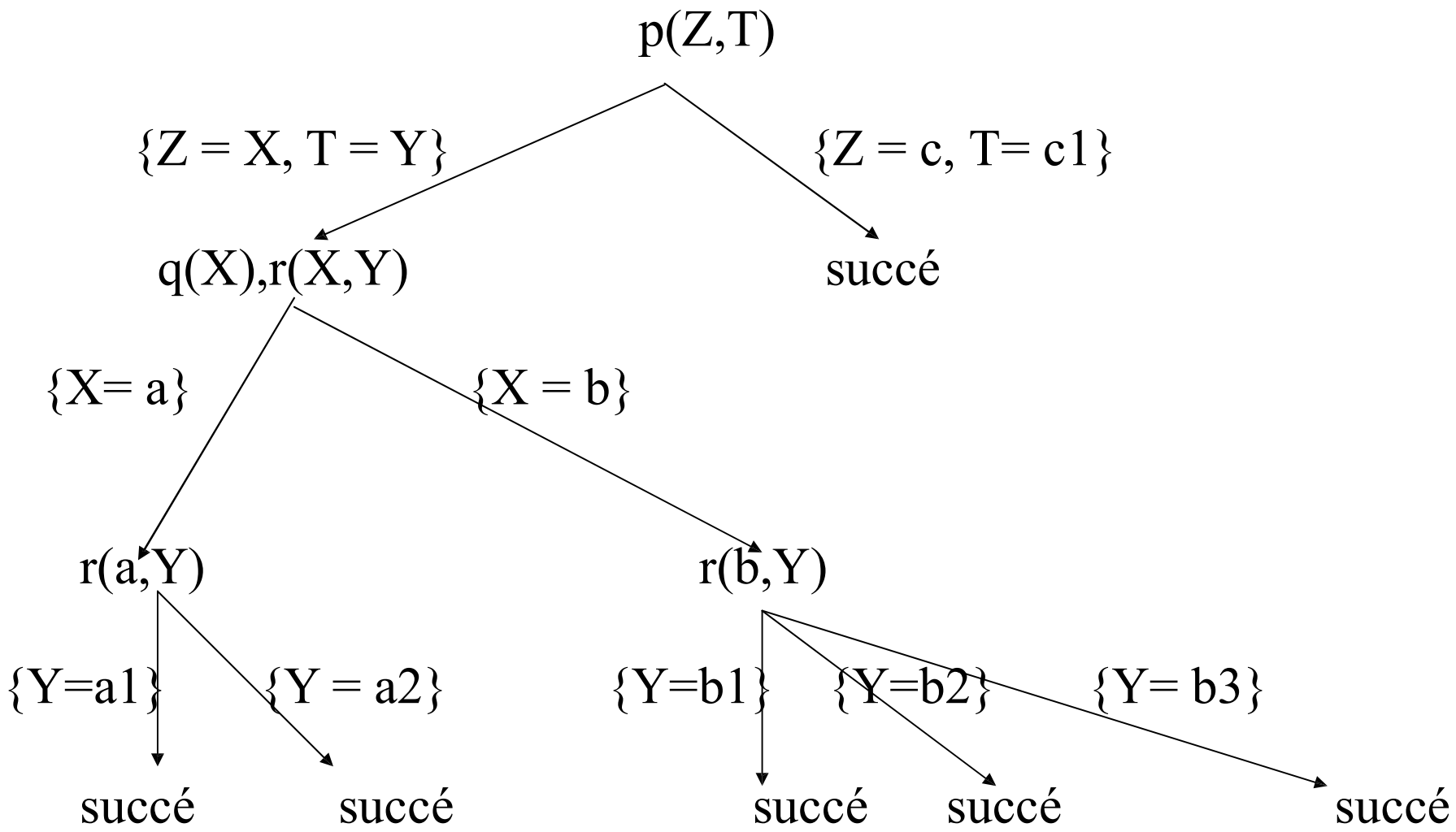
Exemple :

Soit le programme Prolog suivant:

```
p(X,Y) :- q(X), r(X,Y).
p(c,c1).

q(a).
q(b).
```

```
r(a,a1).
r(a,a2).
r(b,b1).
r(b,b2).
r(b,b3).
```

L'arbre de recherche construit par Prolog pour le but p(Z,T) est:



Z=a, T=a1 ; Z=b, T=b1 ; Z=c, T=c1
Z=a, T=a2 ; Z=b, T=b2 ;
Z=b, T=b3 ;

- En fonction de l'endroit où l'on place une coupure dans la définition de p, certaines branches de l'arbre de recherche seront supprimées :

  - Si on définit p par :

    p(X,Y) :- q(X), r(X,Y), !.
    p(c,c1).

    Prolog donne la première solution puis coupe toutes les branches en attente (b112, b12 et b2).

Z=a,
T=a1.

- Si on définit p par :

```
p(X,Y) :- q(X), !, r(X,Y).
p(c,c1).
```

Prolog donne les solutions 1 et 2 puis coupe les branches en attente (b12 et b2).

- Si on définit p par :

```
p(X,Y) :- !, q(X), r(X,Y).
p(c,c1).
```

Prolog donne les solutions 1, 2, 3, 4 et 5 puis coupe la branche en attente (b2).

Exemple

p(1).
p(2).
p(3).

?- p(X).
X = 1 ;
X = 2 ;
X=3 ;
No

?- p(X),P(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 = 3, Y = 1;
X = 3, Y = 2;
X = 3, Y = 3;
No

?- p(X), !, P(Y).
X = 1, Y = 1;
X = 1, Y = 2;
X = 1, Y =3;
No

p(1).
p(2):-!.
p(3).

?- p(X).
X = 1 ;
X = 2 ;
No

?- p(X),P(Y).
X = 1, Y = 1;
X = 1, Y = 2;
X = 2, Y = 1;
X = 2, Y = 2;
No

?- p(X), !, P(Y).
X = 1, Y = 1;
X = 1, Y = 2;
No

II- L'utilité de la coupure

- Par défaut Prolog tente de trouver toutes les solutions possibles en utilisant le retour_arrière (back tracking). Dans certains cas ce retour-arrière peut être nuisible:

    1. si le retour-arrière va **répéter une solution.**
    2. si le retour-arrière **n'a aucune chance de trouver une solution**.
    3. si on veut avoir seulement la **première solution**.

- La **coupure** permet de signaler qu'un retour-arrière n'est pas désiré
- On peut définir un "if-then-else" des langages impératifs en utilisant le cut. Pour cela, on peut définir une expression de la forme :

```
S :- P,  !, Q.   /* si P s'unifie, on essaie d'unifier Q  */
S :- R                    /* sinon, on unifie R */
```

- **<u>Remarque</u>** :

rel(...) :- <*G*>, !, <*D*>.

Si le système trouve une preuve pour <*G*> alors un échec de prouver <*D*> sera un échec pour cette règle et **aucune autre règle pour rel(...) ne sera essayée**.

Exemple :

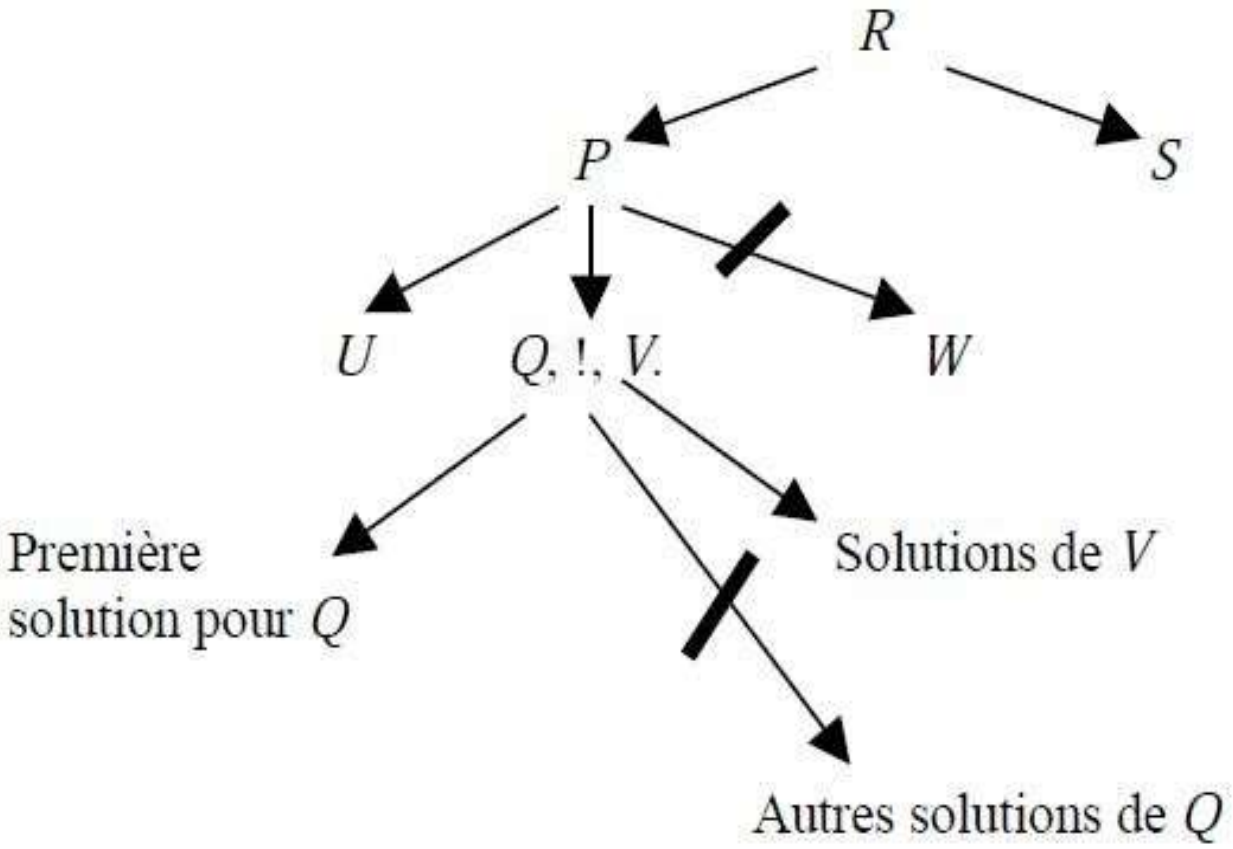
```
couleur(bleu).

couleur(orange).
couleur(chocolat).
```

```
aliment(pomme).
aliment(orange).
aliment(chocolat).
```

```
deuxsens1(X) :- couleur(X), aliment(X).

deuxsens2(X) :- couleur(X), !, aliment(X).

deuxsens3(X) :- couleur(X), aliment(X), !.

 ?- deuxsens1(orange).
yes

?- deuxsens2(orange).
yes

?- deuxsens3(orange).
yes
```

```
?- deuxsens1(X).
   X = orange ? ;
   X = chocolat ? ;
   no

?- deuxsens2(X).
no

?- deuxsens3(X).
X = orange ? ;
no
```

R :- P.
R :- S.
P :- U.
P :- Q, !, V.
P :- W.



```
min(X, Y, X) :-  X <= Y, !.
min(X, Y, Y) :-  X > Y.
```

```
min(X, Y, X) :-  X <= Y, !.
min(X, Y, Y).
```

Attention min(3, 5, 5).