Le mécanisme de résolution en prolog

I- L'unification

✓ Le fonctionnement de Prolog repose en partie sur le mécanisme d'unification qui permet d'apparier une question avec la tête d'une clause.

✓ L'unification est l'opération élémentaire que réalise Prolog pour rendre deux termes identiques.

✓ Trois cas sont possible pour que l'unification réussie :

1- Deux termes atomiques s'unifient s'ils sont identiques (comme les constantes).

2- Une variable s'unifie avec tout type de terme.

3- Un terme composé s'unifie avec un terme composé de même foncteur et même nombre d'arguments, à condition que ces arguments s'unifient.

✓ En voici un exemple, les deux prédicats suivants sont appariables (l'opérateur d'unification est "=") :

?- p(X, b(Z,a), X) = p(Y, Y, b(V,a)).

X = b(V, a),
Z = V,
Y = b(V, a).

✓ Les atomes logiques P(X,a,Y) et P(c,a,Z), où a et c sont des constantes, sont unifiables {X=c, Y=Z}. Par contre P(X,a,Y) et P(c,b,Z) ne sont pas unifiables car les constantes ne peuvent être remplacées.

✓ Prévoyez les réponses fournies par PROLOG pour ces requêtes :

?- k(s(g),Y)=k(X,t(f)).
Y = t(f),
X = s(g).

?- p(f(Y),W,g(Z)) = p(U,U,V).

W = f(Y),

U = f(Y),

V = g(Z).

?- p(f(Y),W,g(Z)) = p(V,U,V) .

false.

?- p(a,X,f(g(Y))) = p(Z,h(Z,W),f(W)).

X = h(a, g(Y)),

Z = a,

W = g(Y).

II- Principe de résolution

- Différentes stratégies ont été développées pour guider le processus de résolution.
-
- Le mécanisme de résolution de Prolog consiste à parcourir un arbre de gauche à droite et en profondeur d'abord. Chaque appel de prédicat constitue un nœud de cet arbre, et les branches qui en partent correspondent aux possibilités d'appariement avec des têtes de clauses.
-
- L'ensemble des solutions d'un programme prolog peut être calculé par une approche ascendante, dite en chaînage avant: on part des faits, et on applique itérativement toutes les règles pour déduire de nouveaux faits ... jusqu'à ce qu'on ait tout déduit.
-
-
- Considérons par exemple le programme Prolog suivant:

```
parent(ali,said).
parent(said,amina).
parent(amina,marwa).

homme(ali).
homme(said).

pere(X,Y) :- parent(X,Y), homme(X).

grand_pere(X,Y) :- pere(X,Z), parent(Z,Y).
```

✓ A partir de l'ensemble des faits E_0:

E_0 = { parent(ali,said), parent(said,amina), parent(amina,marwa), homme(ali), homme(said) }

✓ Et en appliquant les règles du programme, on obtient l'ensemble E_1:

E_1 = { pere(ali,said), pere(said,amina) }

✓ Et à partir de E_0, E_1 et des règles, on déduit l'ensemble E_2:

E_2 = { grand_pere(ali,amina), grand_pere(said,marwa) }

✓ À partir de E_0, E_1, E_2 et des règles, on ne peut plus rien déduire de nouveau.

✓ L'union de E_0, E_1 et E_2 constitue l'ensemble des conséquences logiques du programme (La dénotation du programme).

✓ D'une façon générale, on ne peut pas calculer l'ensemble des conséquences logiques d'un programme par l'approche ascendante: ce calcul serait trop coûteux, voir infini.

✓ En revanche, on peut démontrer qu'un but composé d'une suite d'atomes logiques est une conséquence logique du programme, en utilisant l'approche descendante, dite en chaînage arrière:

✓ Pour prouver un but composé d'une suite d'atomes logiques :

?- A1, A2, .., An.

Prolog commence par prouver le premier atome logique A1. Pour cela, il cherche une clause dans le programme dont l'atome de tête s'unifie avec cet atome logique. Par exemple la clause :

A1 :- A11,A12,…,A1n.

✓ Puis Prolog remplace l'atome logique A1 dans le but par les atomes logiques du corps de la clause, et on aura un nouveau but à prouver :

A11, A12, …, A1n, A2, …, An.

✓ Prolog recommence alors ce processus, avec toutes les clauses qui s'unifient avec les atomes logiques du but, jusqu'à ce qu'il n'y ait plus rien à prouver. A ce moment, Prolog affiche yes or no ou bien les valeurs de variables.

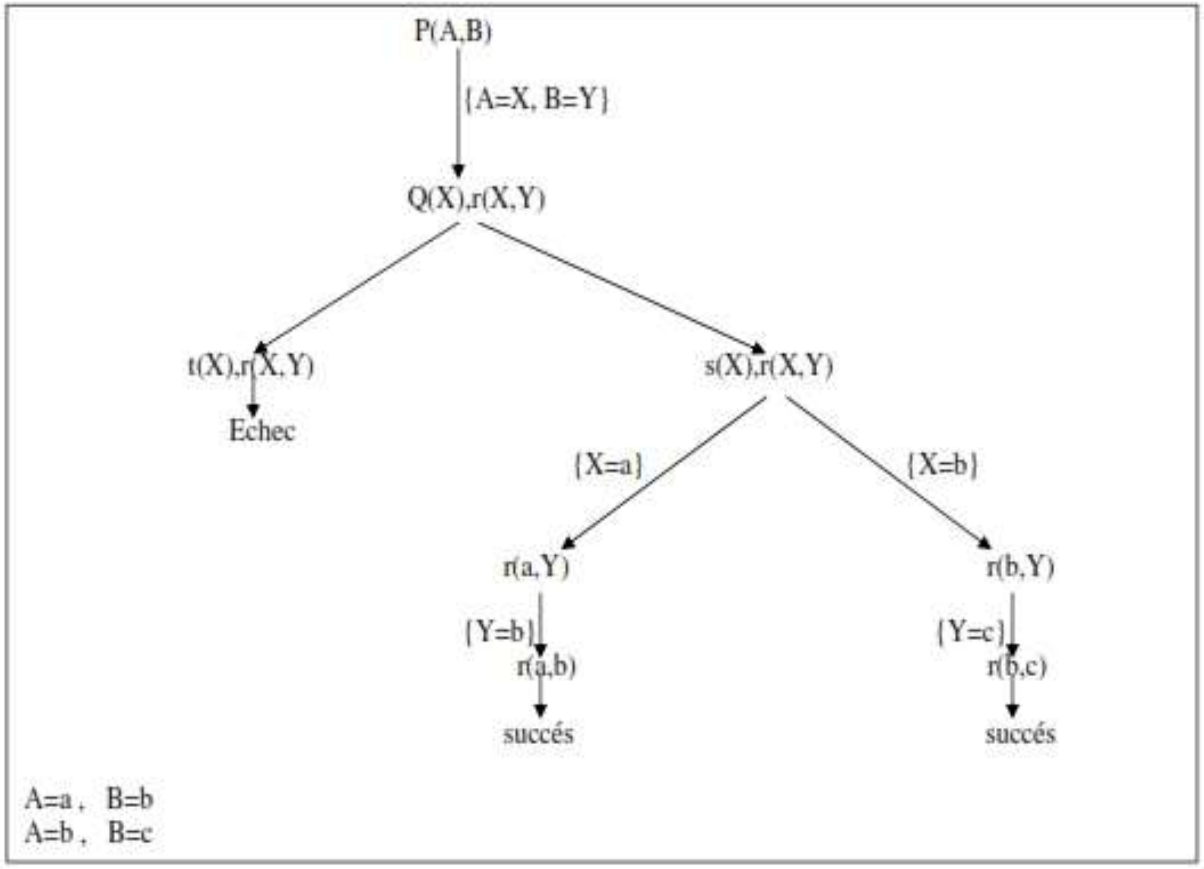
✓ Exemple : Soit le programme prolog suivant :

s(a).
s(b).
r(a,b).
r(b,c).
r(c,b).

p(X,Y) :- q(X),r(X,Y).
q(X) :- t(X).
q(X) :- s(X).

✓ Soit la requête suivante : ?- p(A,B).

✓ L'arbre de recherche du but p(A,B) est le suivant :


P(A,B)
{A=X, B=Y}
Q(X),r(X,Y)
t(X),r(X,Y)
Echec
s(X),r(X,Y)
{X=a}
{X=b}
r(a,Y)
r(b,Y)
{Y=b}
{Y=c}
r(a,b)
r(b,c)
succés
succés
A=a , B=b
A=b , B=c

Retrouvez La dénotation de ce programme (l'ensemble des solutions) en utilisant l'approche ascendante.

E_0= { s(a), s(b), r(a,b), r(b,c), r(c,b) }
E_1= { p(a,b), p(b,c)}
E_2= {q(a),q(b)}
E={E_0+E_1+E_2}

✓Considérons le programme prolog suivant :

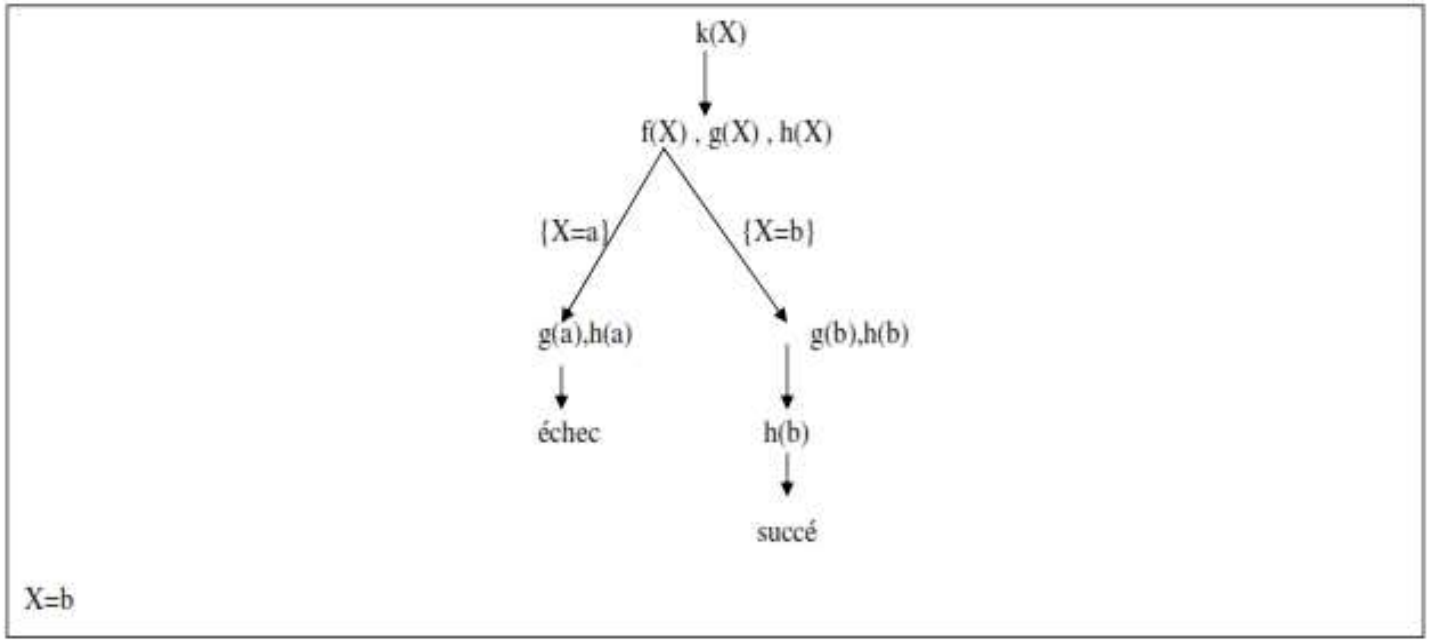
```
f(a).
f(b).
g(b).
h(b).
k(X) :- f(X), g(X), h(X).
```

Et soit la requête suivante :

```
?- k(X).
```

La seule réponse de prolog est : X=b.

Comment prolog a-t-il trouvé cette réponse ? Dessinez l'arbre de résolution (l'arbre de recherche).



E={f(a),f(b),g(b),h(b).k(b)}

Exemple

- Soit le programme
- p(a).
- p(b).
- p(c).
- q(c).
- q(d).
- r(a).
- p(X)←– q(X).
- q(X)←– r(X).
- Soit le but ?-p(X).

Stratégie de Prolog (1/2)

- Pour résoudre un but, Prolog construit l'arbre de recherche du but et le parcours en profondeur d'abord
  - nœud feuille de succès : c'est une solution, Prolog l'affiche et peut chercher d'autres solutions en remontant au dernier *point de choix*
  - nœud feuille d'échec : remontée dans l'arbre jusqu'à un point de choix possédant des branches non explorées
- Prolog ne trouve pas toujours de solution
  - il faut des faits et règles correspondant au but
  - il faut pouvoir unifier
- *Exemples* :
  - `p(X,X)` et `p(titi,toto)` *ne peuvent être unifiés*
  - `q(X,Y)` et `q(p(T),Y)` *peuvent être unifiés*

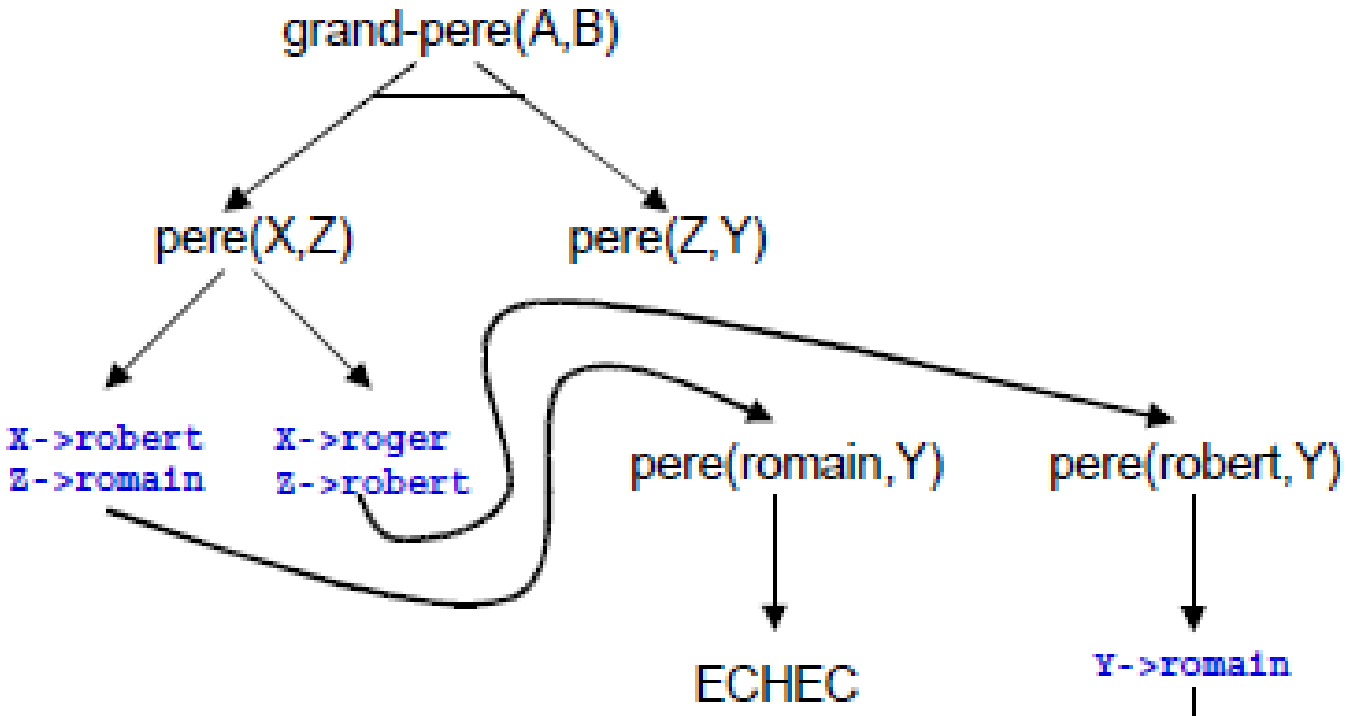
Stratégie de Prolog (2/2)

- Si l 'unification échoue : situation d'échec sur la règle considérée pour démontrer la formule.

- Si l'unification réussit : substitution des variables présentes dans la queue de la clause par les valeurs correspondantes des variables de l 'unificateur.

- A la fin, l'ensemble des couples valeur-variable des variables présentes dans la question initiale forme la ***solution*** affichée par Prolog.

Graphe ET/OU




```
pere(robert,romain).
pere(roger,robert).

grand_pere(X,Y) :- pere(X,Z),pere(Z,Y).
```

A = X = roger
B = Y = romain