CENTRE UNIVERSITAIRE DE MILA
INSTITUT DES SCIENCES ET DE LA TECHNOLOGIE

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

LA NEGATION

I- Définition

- La négation, en Prolog, peut être vue comme un prédicat qui agit sur des prédicats.

- not(p) réussit si p échoue, et échoue sinon. C'est ce qu'on appelle **la négation par l'échec**. Le même effet peut être obtenu par l'association du **cut** avec le prédicat prédéfini **fail** qui a la propriété de toujours échouer :

```
q(X,Y):- p(X,Y), !, fail.

q(X,Y).
```

q(X,Y) se comporte exactement comme not(p(X,Y)). La négation logique pure n'existe pas en Prolog, on se repose sur la négation par l'échec. not(X) ne veut pas dire que X est faux, il veut dire que X ne peut pas être prouvé.

- En négation par l'échec, un prédicat est considéré comme faux si, en un temps fini, on échoue à montrer qu'il est vrai (par l'algorithme de résolution de Prolog). Cela est appelé l'hypothèse du monde clos : on considère que tout ce qui doit être connu est inclus dans la base de données. En d'autres termes, tant qu'un fait n'est pas connu comme étant vrai, il est considéré comme faux.

Exemple1 :

```
petit(0).  petit(1).  petit(2).
zero(0).
?- petit(X), not(zero(X)).
X = 1 ? ;
X = 2 ? ;
no
?- not(zero(X)), petit(X).

No
```

Exemple2 :

```
homme( ali).
femme( amina).

femme( X ) :- not(homme(X) ).

?- femme( adam ).
yes
```

Exemple3 :

```
big(bear).
small(X) :-not(big(X)).

?- small (cat).
Yes
```

?-small (X).
No

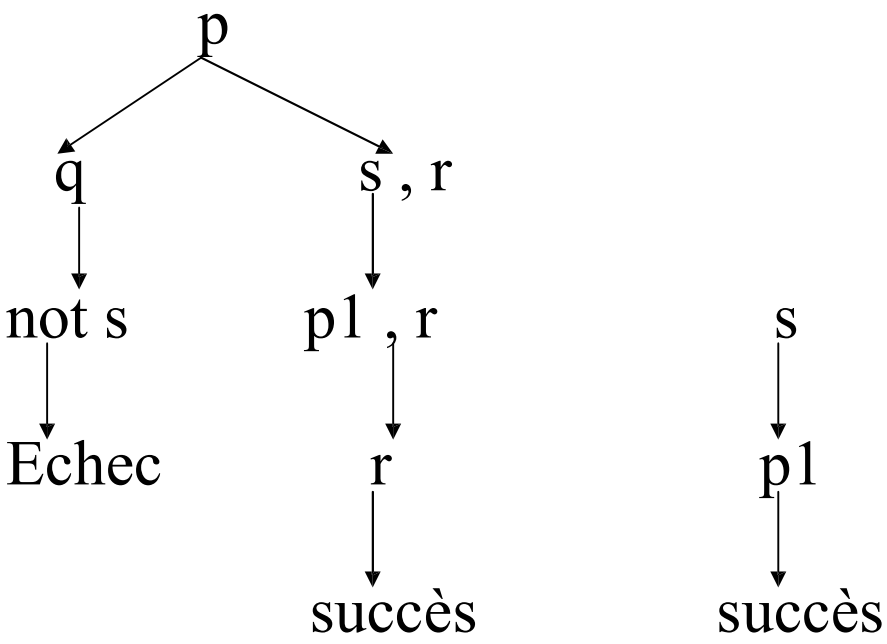
- Attention, la négation en Prolog peut fonctionner étrangement.

- La sémantique de la négation par l'échec n'est obtenue que si toutes les variables du but « sous le not » sont instanciées au moment de l'appel pour sa résolution. On parle alors de négation sure

Exemple4:

```
p:- q.
p:- s , r .
r.
q:-not(s).
s:-p1.
p1.
```

L’arbre du but ?-p.



?-p.
yes

Exemple5 :

```
student(ali).
married(mohamed).
unmarried_student(X) :- not(married(X)), student(X).
```

Si on veut un étudiant non marrié, on demande:

```
?- unmarried_student(X).
no
```

```
?- unmarried_student(ali).
yes
```

On peut régler le problème en redéfinissant le prédicat:

```
unmarried_student(X) :- student(X), not(married(X)).
```

?- unmarried_student(X).
X=ali

Exemple :

etudiant(amine).
etudiant(khalil).
mineur(saber).

etudiantmajeur(X) :- not(mineur(X)), etudiant(X).

?- etudiantmajeur(khalil).

yes

?- etudiantmajeur(X).

No