Conception et implémentation

IFT-17586 Intelligence artificielle I - A2002

Séance 6

- Autres particularités de Prolog

Prédicats particuliers

- Coupure : !/0
  - Pour obtenir une seule solution
  - Pour réduire l'arbre de résolution
- Négation : not/1
  - Pour nier un But
- Échec : fail/0
  - Pour faire échouer un But

Prédicats particuliers

- ! : stopper la récursion
  - exemple :

a(1). a(2). a(3). b(2). b(3). b(4). c(A) :- a(A), b(A). d(A) :- !, a(A), b(A). e(A) :- a(A), !, b(A). f(A) :- a(A), b(A), !.	| ?- c(X). X = 2 ; X = 3 | ?- d(X). X = 2 ; X = 3	| ?- e(X). no | ?- f(X). X = 2

Prédicats particuliers

- ! : stopper la récursion
  - exemple incorrect :

%version incorrecte max(M,N,M):-M >= N,!. max(M,N,N).	| ?- max(1,2,M). M = 2 | ?- max(2,1,M). M = 2 | ?- max(2,1,1). yes

La coupure : une seule réponse

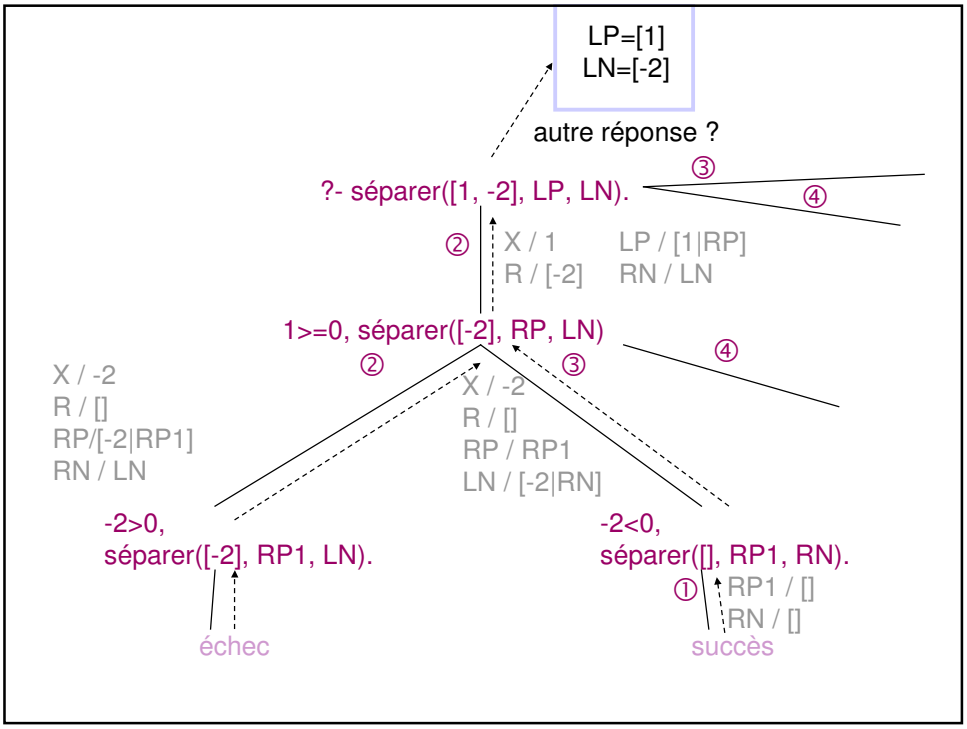
```
parent(pierre, sébastien).
parent(nathalie, sébastien).
parent(marie, nathalie).
parent(robert, nathalie).

?-parent(X, sébastien).
X=pierre;
X=nathalie;
no
?-
```

```
?-parent(X, sébastien), !.
X=pierre
?-
```


La coupure : réduire la recherche

Séparer une liste de nombre en une liste de positifs et une liste de négatifs, les 0 sont supprimés

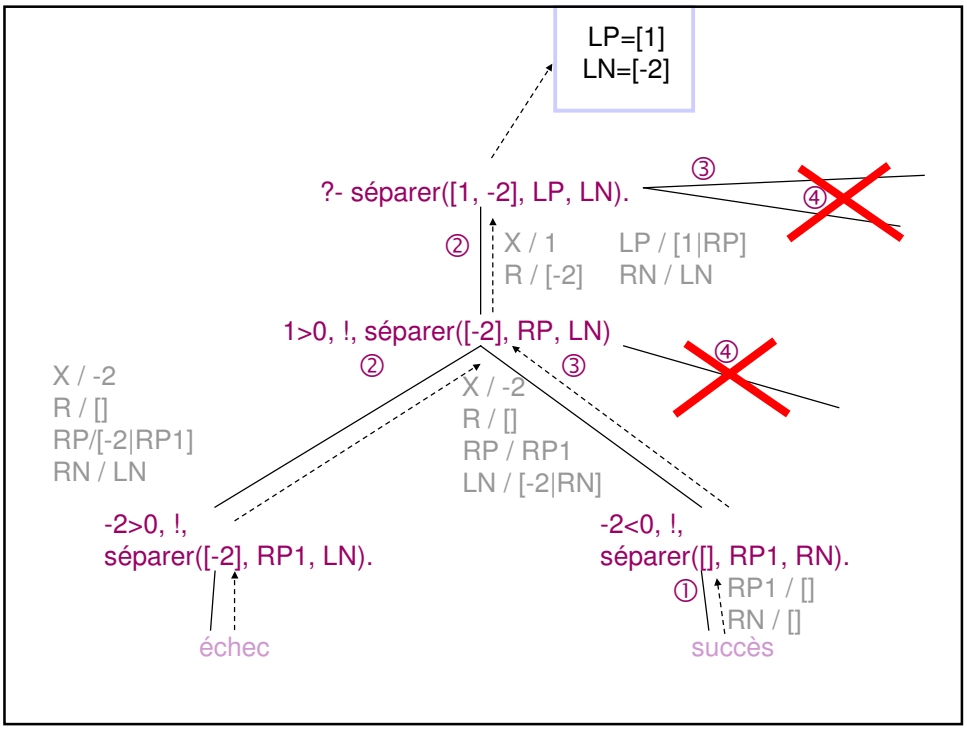
```
séparer([ ], [ ], [ ]).
séparer([X|R], [X|RP], RN) :- X>0, séparer(R, RP, RN).
séparer([X|R], RP, [X|RN]) :- X<0, séparer(R, RP, RN).
séparer([X|R], RP, RN) :- X=:=0, séparer(R, RP, RN).
```



La coupure : réduire la recherche

Comment ? ... En rendant chaque clause exclusive.

```
séparer([ ], [ ], [ ]).
séparer([X|R], [X|RP], RN) :- X>0, !, séparer(R, RP, RN).
séparer([X|R], RP, [X|RN]) :- X<0, !, séparer(R, RP, RN).
séparer([X|R], RP, RN) :- X=:=0, !, séparer(R, RP, RN).
```



Exemple (1)

« Marie aime les animaux, mais pas les serpents »
Comment traduire ces connaissances en Prolog ?

Si X est un animal alors Marie aime X
Implémentation de cette première règle :
aime(marie, X) :- animal(X).

Si X est un serpent alors (Marie aime X) est faux
Comment implémenter cette nouvelle règle ?

Implémentation de la négation

Le prédicat not/1 utilise la coupure et l'échec.

non(But) :
si But est vrai alors non(But) est faux
si But est faux alors non(But) est vrai

```
non( But ) :- But, !, fail.
non( But ).
```


Exemple (2)

2 possibilités :

- Négation : on modifie la première règle

```
aime(marie, X) :- animal(X), not serpent(X).
```

- Coupure et échec : on ajoute une nouvelle règle

```
aime(marie, X) :- serpent(X), !, fail.
```

Remarque :
cette règle devra être placée avant l'autre, pourquoi ?

Prédicats particuliers

- forall/2 : générer + tester
  - exemple :

a(1). a(2). a(3).	| ?- forall(**a(X), write(X)**). 123X = _ | ?- forall(**a(X), (write(*), write(X))**). *1*2*3X = _ | ?- forall(**a(X), (X == 1, write(*), write(X))**). *1no

Prédicats particuliers

- (... -> ... ; ...) : si ... alors ... sinon ...
  - exemple :

a(1).	| ?- a(1) -> write('a'). ayes | ?- a(2) -> write('a'). no | ?- a(1) -> write('a') ; write(b). ayes | ?- a(2) -> write('a') ; write(b). byes

Exercices : programmer les prédicats suivants en Prolog en utilisant la coupure

- enlever_tous(X,L,R) : enlever toutes les occurrences de X dans L (résultat = R)
- intersection(L1,L2,I) : faire l'intersection des listes L1 et L2 (résultat = I)
- union(L1,L2,U) : faire l'intersection des listes L1 et L2 (résultat = U)
- remplacer_tous(L,X,Y,R) : remplacer toutes les occurrences de X par Y dans L (résultat = R)
- séparer(L,X,LAvant,LApres) : séparer la liste L en deux sous-listes (LAvant = sous-liste de L précédant la première occurrence de X et