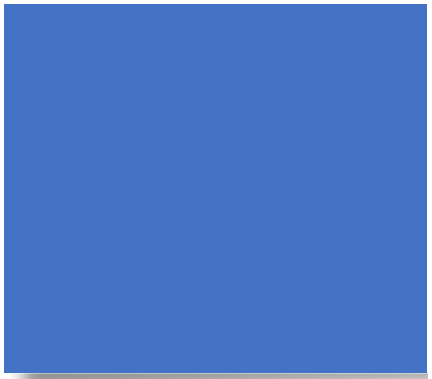
Programmation Logique

Récursivité

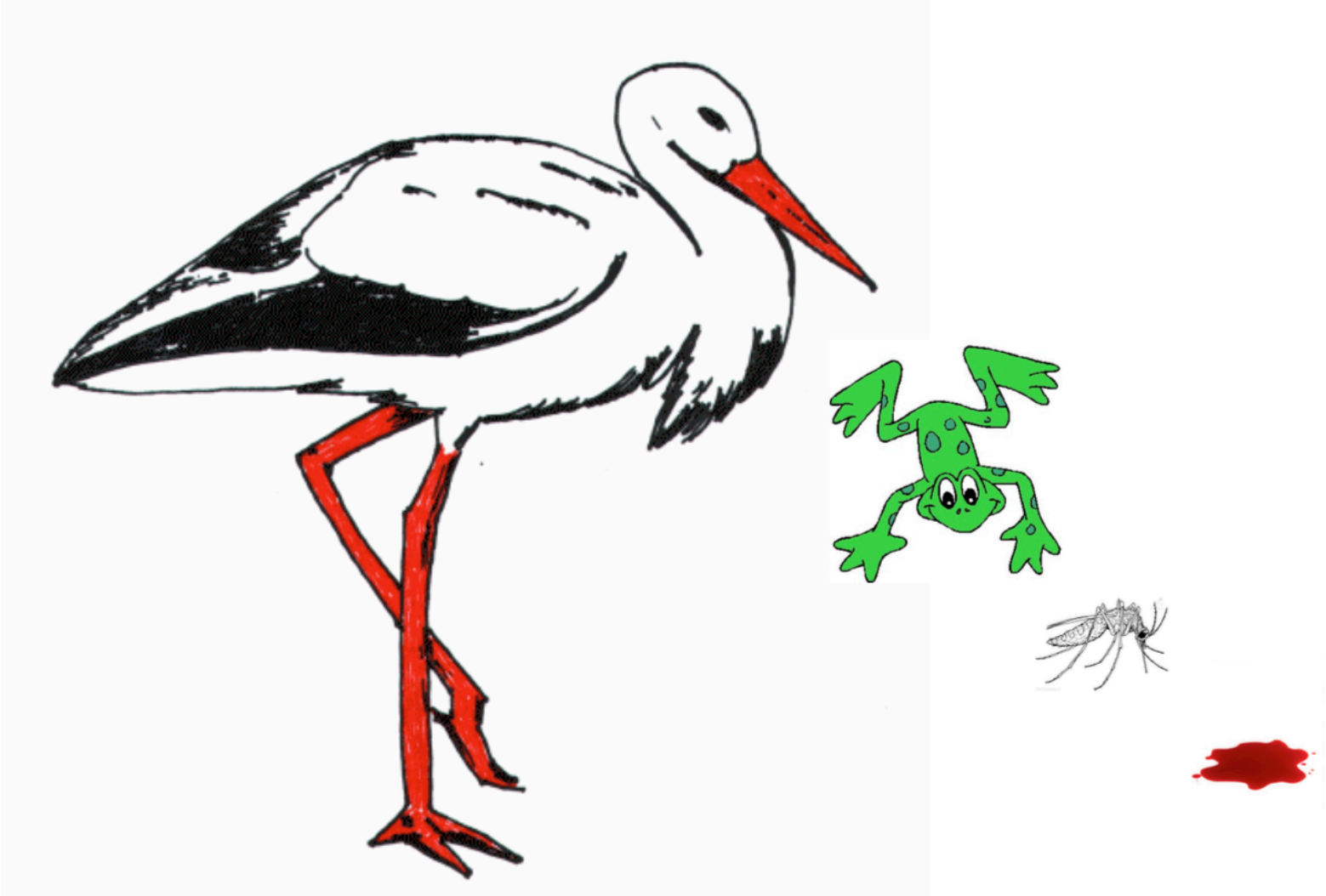


Enseignant: NGUYEN Thi Minh Tuyen

Définition

- Les prédicats Prolog peuvent être définis récursivement.
- Un prédicat est défini récursivement si une ou plusieurs règles dans sa définition se réfèrent à elle-même.

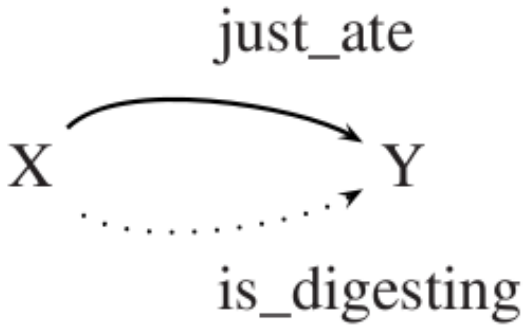
Exemple: Digestion

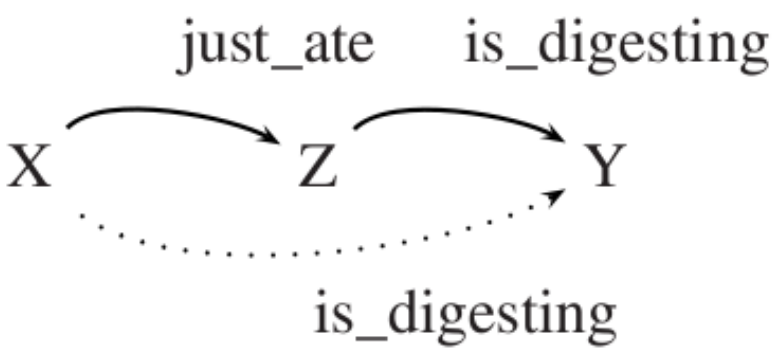


Exemple: Base de connaissance en Prolog


```
is_digesting(X,Y) :- just_ate(X,Y).
is_digesting(X,Y) :-just_ate(X,Z), is_digesting(Z,Y).

just_ate(mosquito,blood(john)).
just_ate(frog,mosquito).
just_ate(stork,frog).
```


just_ate
X
Y
is_digesting


just_ate
is_digesting
X
Z
Y
is_digesting

Requête


```
?- is_digesting(stork,mosquito).
true.
```

Essayer:

```
?- trace.
?- is_digesting(stork,mosquito).
```

Condition d'arrêt

- On considère la règle

  p:-p.

- La requête:

  ?-p.

➔ boucle indéfinie.

Rappel: Ordre des réponses en Prolog

- Parcourt les clauses du haut vers le bas,
- Parcourt les buts de la gauche vers la droite à l'intérieur des clauses,
- Utilise le *backtrack* pour revenir sur ses mauvais choix.
- Ordonne différemment les clauses peut avoir des conséquences importantes concernant l'arbre de déduction

Ordonnancement des règles et ordonnancement des buts [1]

- descendant.pl

```
child(martha,charlotte).
child(charlotte,caroline).
child(caroline,laura).
child(laura,rose).

descend(X,Y) :- child(X,Y).
descend(X,Y) :- child(X,Z),descend(Z,Y).
```

- Requête:

```
?- descend(martha,D).
```

Ordonnancement des règles et ordonnancement des buts [2]

- descendant.pl

```
child(martha,charlotte).
child(charlotte,caroline).
child(caroline,laura).
child(laura,rose).

descend(X,Y) :- child(X,Z),descend(Z,Y).
descend(X,Y) :- child(X,Y).
```

- Requête:

```
?- descend(martha,D).
```

Ordonnancement des règles et ordonnancement des buts [3]

- descendant.pl

```
child(martha,charlotte).
child(charlotte,caroline).
child(caroline,laura).
child(laura,rose).

descend(X,Y) :- descend(Z,Y), child(X,Z).
descend(X,Y) :- child(X,Y).
```

- Requête:

```
?- descend(martha,D).
```

Règles

- On place en premier la(les) clause(s) permettant de sortir de la récursivité;
- Dans une clause avec récursivité, les buts induisant la récursivité doivent être placé le plus vers la droite possible.

Boucle avec un nombre fixe de fois

- Plusieurs langages de programmation fournissent des boucles `for` qui permettent à un ensemble d'instructions d'être éxécuté dans un nombre fixe de fois.
- Prolog ne supporte pas ce type de boucle.
- Prolog peut l'imiter ce type de boucle en utilisant la récursivité.

Exemple 1

- Définissez un prédicat `loop/1` récursive qui permet d'afficher les entiers de N → 1, avec N>0.
- Sortie:

```
?- loop(1).
The value is: 1

?- loop(5).
The value is: 5
The value is: 4
The value is: 3
The value is: 2
The value is: 1
```

Comparison arithmétique:
> < =< >= =:= =\=
Affectation: is
Calcul arithmétique: + - * /

Exemple 2

- Définissez récursivement un prédicat `output_values(First,Last)` qui permet d'afficher les entiers de First → Last.
- Sortie:

```
?- output_values(2, 2).
true .

?- output_values(3,2).
false.

?- output_values(2, 4).
2
3
true .
```

Comparison arithmétique:
> < =< >= =:= =\=
Affectation: is
Calcul arithmétique: + - * /

Exemple 3

- Définissez un prédicat `sumto(N,S)` pour trouver la somme S des entiers de 1 à N (disons pour N = 100).
- Sortie:

```
?- sumto(5,S).
S = 15.

?- sumto(1,S).
S = 1.

?- sumto(10,S).
S = 55.

?- sumto(3,S).
S = 6.
```

Exemple 4

- Définissez un prédicat `writesquares(N)` pour afficher les carrés des N premiers nombres entiers, un par ligne.
- Sortie:

```
?- writesquares(1).
1^2 = 1
true.

?- writesquares(2).
1^2 = 1
2^2 = 4
true .
```

```
?- writesquares(4).
1^2 = 1
2^2 = 4
3^2 = 9
4^2 = 16
true .
```

Boucler jusqu'à ce qu'une condition soit satisfaite

- Plusieurs langages de programmation fournissent des boucles `'until'` qui permettent à un ensemble d'instructions d'être éxécuté de manière répétée jusqu'à ce qu'une condition donnée soit remplie.
- Ce type de boucle n’est pas disponible directement dans Prolog
- Il y a certaine manières pour imiter cela
  - Récursivité
  - Utilisation du prédicat 'repeat'

Prédicat 'repeat'


```
get_answer(Ans):-
     write('Enter answer to question'),nl,
     repeat,write('answer yes or no: '),
     read(Ans), valid(Ans),
     write('Answer is '),write(Ans),nl.

valid(yes). valid(no).
```

`read(Ans)` invitera l'utilisateur à entrer un terme

Backtracking avec le prédicat `fail`

- Le prédicat `fail` échoue toujours, quelque soit l'évaluation "standard" de gauche à droite ou en retour arrière.
- **Avantage**: chercher dans la base de données toutes les clauses ayant une propriété spécifiée

Recherche dans la base de données Prolog


```
dog(fido).
dog(fred).
dog(jonathan).
alldogs:-dog(X),write(X),
                    write(' is a dog'),nl,fail.



?-alldogs.
```

Exemple 2


```
person(john,smith,45,london,doctor).
person(martin,williams,33,birmingham,teacher).
person(henry,smith,26,manchester,plumber).
person(jane,wilson,62,london,teacher).
person(mary,smith,29,glasgow,surveyor).

allteachers:- person(Forename,Surname,_,_,teacher),
      write(Forename),write(' '),write(Surname),nl, fail.



?-allteachers.
```

```
somepeople:-person(Forename,Surname,_,_,_),
            write(Forename),write(' '),write(Surname),nl,
            Surname=williams.



?-somepeople.
```

Trouver plusieurs solutions


```
find_all_routes(Town1,Town2):-
                    findroute(Town1,Town2,Route),
                    write('Possible route: ‘),
                    write(Route),nl,fail.



?-find_all_routes(_,_).
```

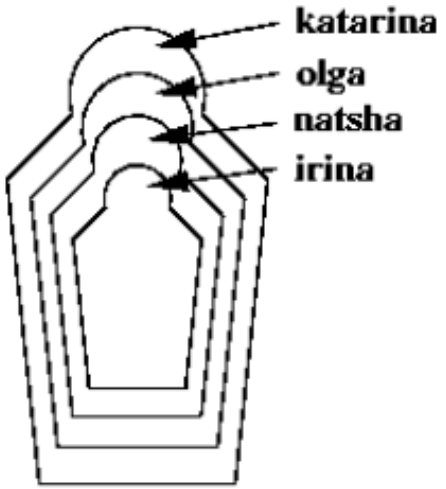
Exercice 1: Nombres entiers

- Définition récursive de successeur
- Ajouter des prédicats suivants
  - `Add`
  - `Mult`
  - `Plus_grand_que`
  - `Moins_que`
  - `Egale_a`

```
numeral(0).
numeral(succ(X)):-numeral(X).
```

Exercice 2: Matriochkas(pouppées russes)[1]

- Série de poupées de tailles décroissantes placées les unes à l'intérieur des autres.



Exercice 2: Matriochkas(pouppées russes)[2]

1. Écrivez la base de faits *inclus(X,Y)* pour exprimer l'inclusion immédiate d'une poupée dans une autre.
2. Définissez récursivement le prédicat *intérieur(X,Y)* : la poupée X est incluse dans Y.
3. Donnez des requêtes pour vérifier vos résultats.

Exercice 3: Réseau ferré

- La base de faits suivante explicite les villes qu'il est possible de relier via un train direct.

```
trainDirect(paris,nancy).
trainDirect(nancy,metz).
trainDirect(metz,strasbourg).
trainDirect(paris,lyon).
trainDirect(lyon,marseille).
trainDirect(marseille,nice).
trainDirect(paris,lehavre).
```

Construire un prédicat récursif `allerDe(X,Y)`, qui est vrai si l'on peut aller de X à Y.

Exercice 4: Voyager

- Étant donné une base de faits suivante:

```
byCar(auckland,hamilton).
byCar(hamilton,raglan).
byCar(valmont,saarbruecken).
byCar(valmont,metz).
byTrain(metz,frankfurt).
byTrain(saarbruecken,frankfurt).
byTrain(metz,paris).
byTrain(saarbruecken,paris).
byPlane(frankfurt,bangkok).
byPlane(frankfurt,singapore).
byPlane(paris,losAngeles).
byPlane(bangkok,auckland).
byPlane(losAngeles,auckland).
```

1. Définir un prédicat `travel(X,Y)` qui est vrai si l'on peut aller de X à Y.
2. Définir un prédicat `travel/3` qui est vrai si l'on peut aller de X à Y en affichant toutes les villes traversées
3. Définir un prédicat `travel/3` qui est vrai si l'on peut aller de X à Y en affichant toutes les villes traversées et tous les moyens de transport utilisés

Question?