Programmation Logique

Début en Prolog

Enseignant: NGUYEN Thi Minh Tuyen

Plan

1. Introduction
2. Prédicats prédéfinis
3. Clauses et prédicats

Plan

1. Introduction
2. Prédicats prédéfinis
3. Clauses et prédicats

Prolog

- Prolog = " PROgrammation en LOGique"
- Origines :
  - 1972: Création de Prolog par A. Colmerauer et P. Roussel
  - 1980 : reconnaissance de Prolog comme langage de développement en Intelligence Artificielle

Implémentations de Prolog

- Nombreuses implémentations : SWI-Prolog, GNU-Prolog, SICTus-Prolog, Prolog II, Visual-Prolog, Quintus-Prolog, ...
- SWI-Prolog
  - Un interpréteur Prolog gratuit.
  - Développé par l'université d'Amsterdam, libre (GPL), depuis 1987.
  - http://www.swi-prolog.org/
  - Disponible sous Windows, Mac et sous Linux.

Le langage Prolog

- Langage d'expression des connaissances fondé sur le langage des prédicats du premier ordre.
- Programmation déclarative:
  - L'utilisateur définit une base de connaissances
  - L'interpréteur Prolog utilise cette base de connaissances pour répondre à des questions

Premier exemple

```
1. dog(fido).
2. dog(rover).
3. dog(henry).
4. cat(felix).
5. cat(michael).
6. cat(jane).
7. animal(X):-dog(X).
```

```
?-dog(fido).
true.
?-dog(jane).
```
[Is jane a dog? No – a cat]

```
false.
?-animal(fido).
```
[Is fido an animal?]

```
true.
```
[yes - because it is a dog and any dog is an animal]

```
?-dog(X).
```
[Is it possible to find anything, let us call it X, that is a dog?]

```
X = fido;
```
[All 3 possible answers are provided]

```
X = rover;
X = henry
?-animal(felix).
```
[felix is a cat and so does not qualify as an animal, as far as the program is concerned]

```
false.
```

Déduction

ÉTANT DONNÉ QUE

tout X est un animal

ET

fido est un chien

DÉDUIRE

fido doit être un animal si X est un chien

Applications de Prolog

- programmes de traitement d'un texte en "langage naturel",
- systèmes de conseil pour applications juridiques
- applications pour la formation
- génération automatique des histoires
- analyser et mesurer les 'réseaux sociaux'
- un système d'assistance électronique pour les médecins.
- ....

SWI-Prolog – la ligne de commande

- `swipl --help`
- `swipl --version`
- Pour lancer SWI-Prolog: tapez `swipl`
- Pour quitter SWI-Prolog: tapez `halt.`

Plan

Prédicats prédéfinis [1]

- `?-write('message').`
- Un prédicat prédéfini, affiche un message 'message' à l'écran.
- `nl`: Un prédicat prédéfini, saute une ligne
- Exemple:

```
?-write('Hello world!'),nl,write('This is a first example.').
Hello world!
This is a first example.
true.
?-
```

Prédicats prédéfinis [2]

- `?- halt.`
  - Se termine Prolog.
- `?- statistics.`
  - provoque la génération des statistiques du système
- `?- consult('chemin/fichier.pl').`
  - charger un fichier.
- `?- listing.`
  - Lister tous les clauses.

Programmes Prolog

- Le moyen le plus simple de créer un programme Prolog:
  - écrit le programme dans un éditeur de texte (Emacs par exemple)
  - → l'enregistre dans un fichier texte `*.pl`
  - → lancer dans SWI-Prolog
- Pour charger un ficher .pl dans SWI-Prolog:
  - `?-consult('chemin/ficher.pl').`
  - `?-['chemin/fichier.pl’].`
- Exemple:
  - `?- consult('/Users/tuyennguyen1/Prolog/animal.pl').`
  - `?- listing(dog).`

Terme

- En Prolog, toutes les données sont représentées par des termes.
- Type des termes:
  - Nombre
  - Atome
  - Variable
  - Terme composé
  - Liste

Type des termes [1]

- Nombre:
  - Nombres d'entiers: `623, -47, +5, 025`
  - Nombre décimaux: `6.43, -.245, +256`

Type des termes [2]

- Atome: constantes qui ne sont pas de valeurs numériques.
  - Toute séquence d'une ou plusieurs lettres (majuscules ou minuscules), chiffres et _, commençant par une lettre minuscule
    - Exemple: `john`, `today_is_Tuesday`, `fred_jones`, `a32_BCD`
    - MAIS PAS: `Today`, `today-is-Tuesday`, `32abc`
  - Toute séquence de caractères entre ' et ', y compris les espaces et les lettres majuscules
    - Exemple: `'Today is Tuesday'`, `'today-is-Tuesday'`, `'32abc'`
  - Toute séquence d'un ou de plusieurs caractères spéciaux d'une liste comprenant les + - * / > <= & # @
    - Exemple: `+++`, `>=`, `>`, `+-`

Type des termes [3]

- Variables
  - Chaînes de caractères commençant par une majuscule
  - Chaînes de caractères commençant par _
  - La variable « indéterminée »/ variable anonyme: _
  - Exemple: `X, Author, Person_A, _123A`
  - MAIS PAS: `45_ABC, Person-A, author`
- Termes composés
  - `foncteur(t1,t2, ... ,tn)` avec n≥1
  - Le nombre d'arguments d'un terme composé: arité
  - Exemple: `likes(paul,prolog), read(X), dog(henry), cat(X), >(3,2), person('john smith',32,doctor,london)`

Type des termes [4]

- Liste:
  - []
  - [dog,cat,y,mypred(A,b,c),[p,q,R],z]
  - [[john,28],[mary,56,teacher],robert,parent(victoria,albert),[a,b,[c,d,e],f],29]
  - [[portsmouth,edinburgh,london,dover],[portsmouth,london,edinburgh],[glasgow]]

Exercice 1

1. Créez un fichier animals.pl, puis entrez le programme suivant:

```
dog(fido).
cat(mary). dog(rover).
dog(tom). cat(harry).
dog(henry).
cat(bill). cat(steve).
```

2. Chargez le program. Vérifiez que la base de données est correcte en utilisant la commande `listing`.
3. Entrez certaines buts pour tester.

Exercice 2

1. Écrivez un programme et mettez des faits qui indiquent que:
    1. Un lion, un tigre et une vache sont des animaux enregistrés dans la base de données.
    2. Deux d'entre eux (lion et tigre) sont des carnivores.
2. Entrez les buts pour tester si:
    a. Il y a un animal tel qu'un tigre est dans la base de données.
    b. Une vache et un tigre sont dans la base de données (une conjonction de deux objectifs).
    c. Un lion est un animal et aussi un carnivore.
    d. Une vache est un animal et aussi un carnivore.

Exercice 3

Que affiche à l'écran pour les buts suivants. Donnez votre explication pour chaque résultat:

- `?-write(hello).`
- `?-write(Hello).`
- `?-write('Hello!').`
- `?-write('Hello!'),nl.`
- `?-100=100.`
- `?-100=1000/10.`
- `?-100 is 1000/10.`
- `?-1000 is 100*10.`
- `?-2 is (5+7)/6.`
- `?-74 is (5+7)*6.`

Plan

1. Introduction
2. Prédicats prédéfinis
3. Clauses et prédicats

Clause

- Un programme Prolog consiste à une suite des clauses.
- Une clause peut être exécutée sur plusieurs lignes ou plusieurs sur la même ligne.
- Une clause se termine par un point '.', suivi d'au moins un caractère "espace blanc".
- Deux types des clauses:
  - faits: `tête.`
  - règles: `tête:-`t_1`,`t_2`,...,` t_k`.` (Avec k≥1)

Exemple

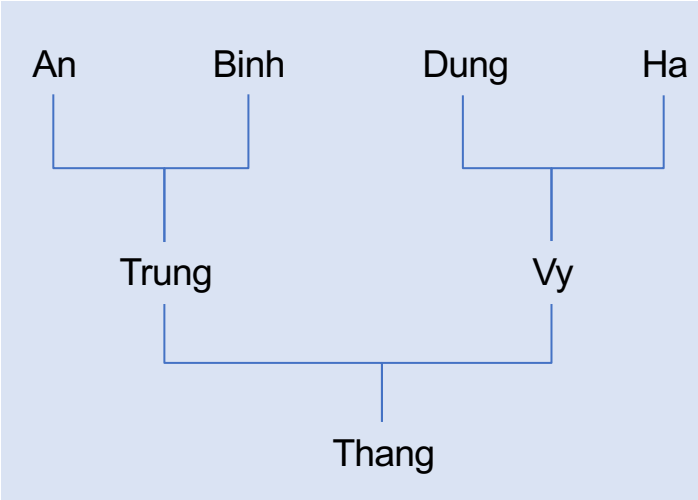

```
christmas.
likes(john,mary).
likes(X,prolog).
dog(fido).
large_animal(X):-animal(X),large(X).
grandparent(X,Y):-father(X,Z),parent(Z,Y).
go:-write('hello world'),nl.
```

Prédicats

- Prédicat: a un nom, zéro ou n paramètres.
  - nom: atome
  - chaque paramètre : un terme
  - Un prédicat avec un nom **Pred** et arité **N**, dénoté **Pred/N**.
- Prédicats sont définis par une collection des clauses. Chaque clause est un fait ou une règle.

Trois sortes de connaissances

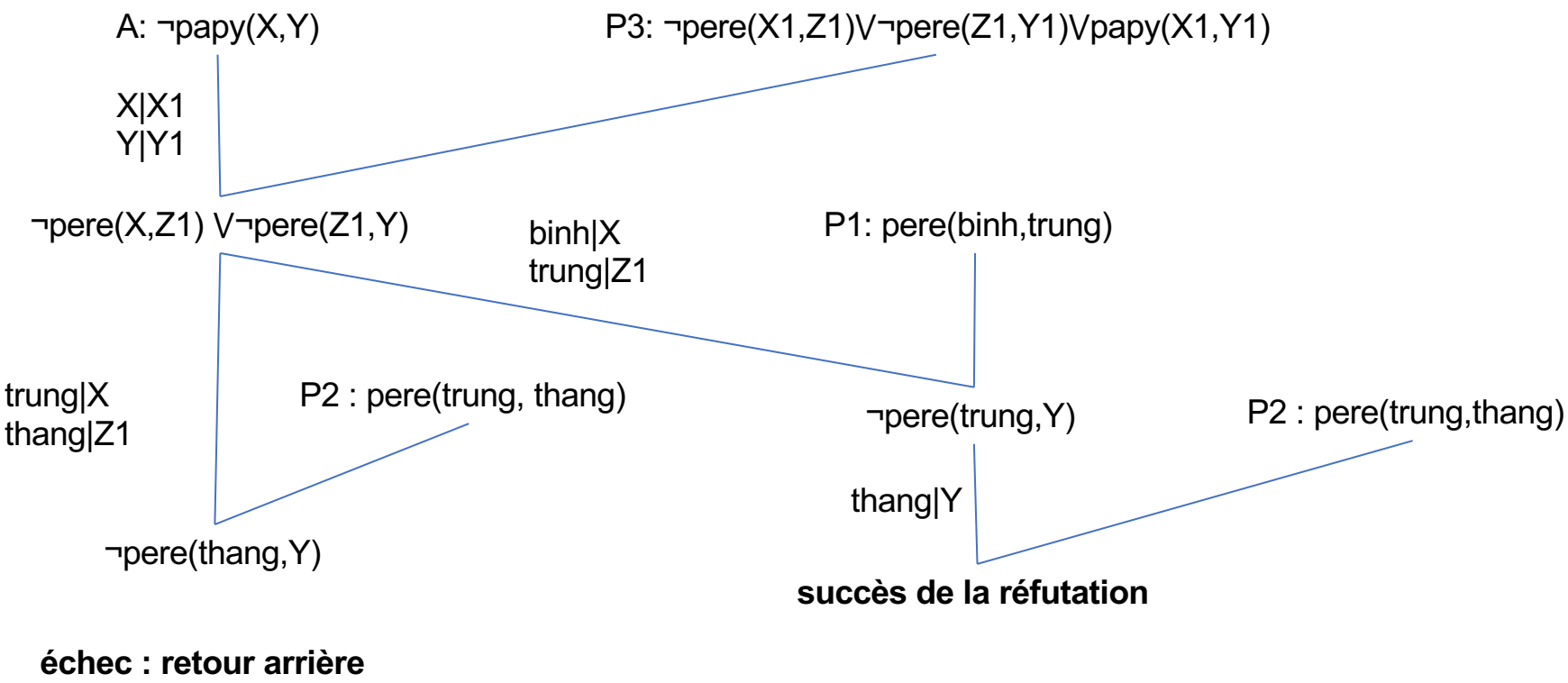
- Faits : **P(...).** avec **P** un prédicat
  - pere(binh, trung)**.**
  - pere(trung, thang)**.**
  - Clause de Horn réduite à un littéral positif
- Règles : **P(...) :- Q(...), ..., R(...).**
  - papy(X,Y) :- pere(X,Z), pere(Z,Y)**.**
  - Clause de Horn complète
- Requêtes : **S(...), ..., T(...).**
  - pere(binh,X), mere(an,X)**.**
  - Clause de Horn sans littéral positif.



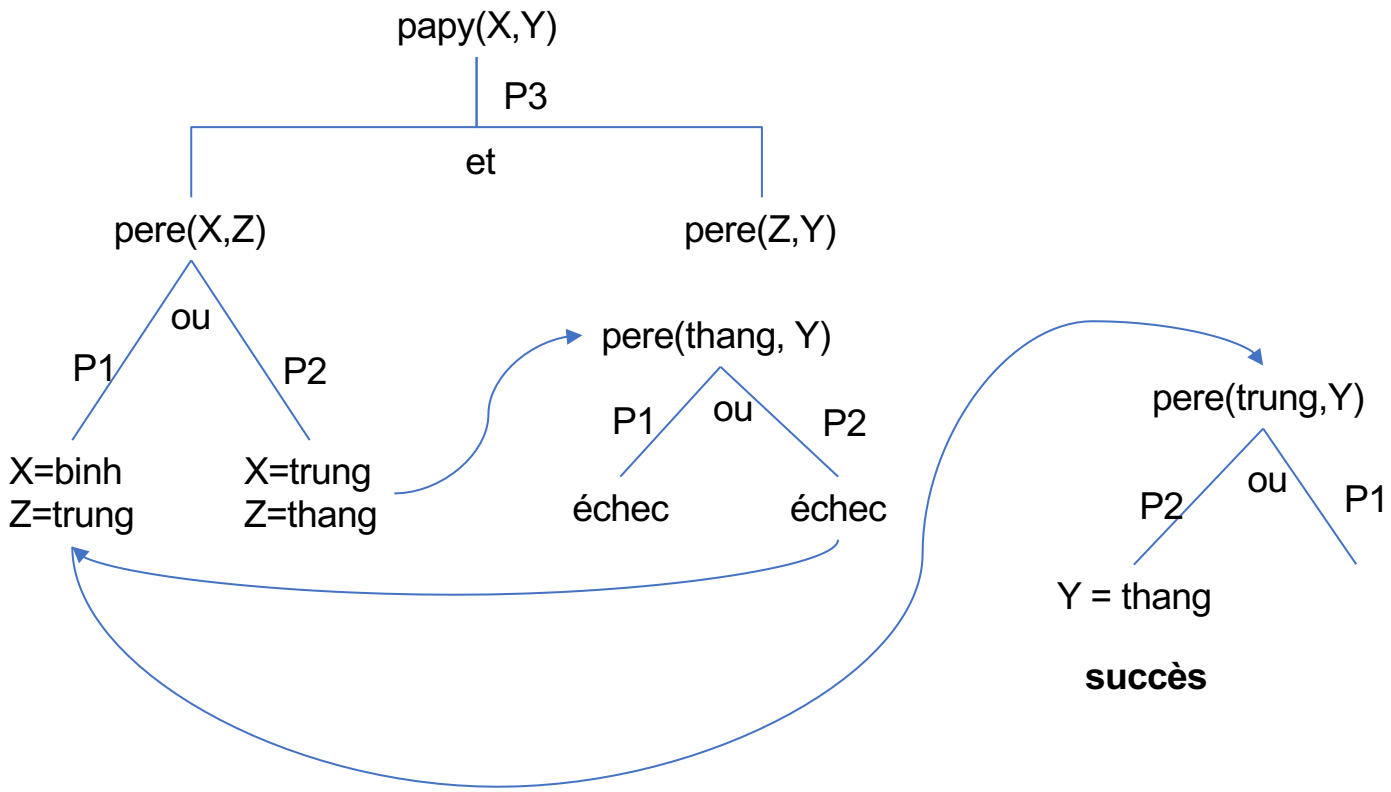
Réfutation par résolution

- Programme **P**
  - P1 : pere(binh, trung).
  - P2 : pere(trung, thang).
  - P3 : papy(X,Y) :- pere(X,Z), pere(Z,Y).
- Appel du programme **P**
  - A : papy(X,Y).
- Réponse : X=binh, Y=thang

Graphe de résolution



Interprétation procédurale: arbre et-ou



Mon deuxième programme

cours2_pere.pl

```
pere(binh, trung).
pere(trung, thang).
papy(X,Y) :- pere(X,Z), pere(Z,Y).
```

```
?-['~/Prolog/cours2_pere.pl'].
true.
```

?- pere(binh,trung).

true.

?- pere(trung,binh).

false.

?- pere(X,trung).

X = binh.

?- pere(X,Y).

X = binh,

Y = trung

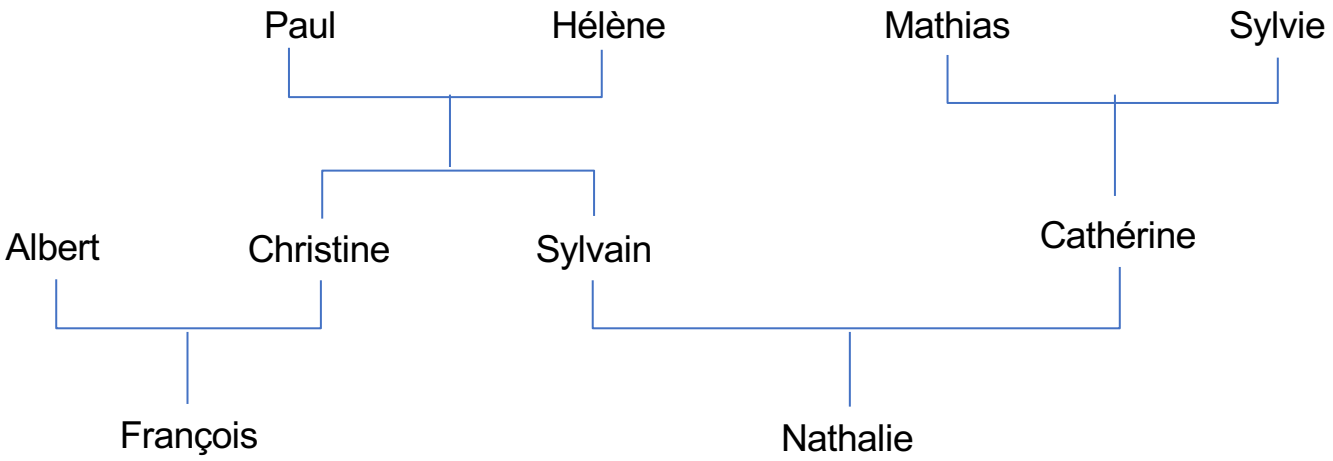
X = trung,

Y = thang.

Ordre des réponses

- Prolog parcourt le paquet de clauses de haut en bas, et
- Pour chaque clause, Prolog parcourt de gauche à droite

Exercice: Un arbre généalogique



- Construisez la base de connaisance pour l'arbre généalogique en utilisant le prédicat `pere_de(Personne, Enfant)`, `mere_de(Personne, Enfant)`.
- Définissez les règles pour représenter les relations suivantes: `parent_de`, `enfant_de`, `grand_mere_de`, `grand_pere_de`, `grand_parent_de`, `femme_de`, `mari_de`

Simplifier les buts

- Au lieu de saisir plusieurs fois dans le système :

```
?-dog(X),large(X),write(X),write(' is a large dog'),nl.
```

- Nous pouvons définir un prédicat tel que `go/0` ou `start/0`:

```
go:-dog(X),large(X),write(X), write(' is a large dog'),nl.
```

- Cela nous permet de raccourcir notre buts:

```
?-go.
```

Récursion

- C'est une technique importante pour définir les prédicats.
- Comme une définition récursive.
- Deux formes:
  - Récursion directe: Le prédicat `pred1` est défini en termes de lui-même ;
  - Récursion indirecte: Le prédicat `pred1` est défini à l'aide de `pred2`, qui est défini.
  - Récursion directe est le plus commune.
- Exemple:
  - `likes(john,X):-likes(X,Y),dog(Y).`

Prédicats et fonctions

- Le terme "prédicat" est liée à son utilisation en mathématique.
  - un prédicat est considéré comme une relation entre plusieurs valeurs
  - Par exemple: `likes(henry,mary)` ou `X=Y` qui retourne soit vrai soit faux.
- Contrairement, une fonction, tel que 6+4, racine carrée de 64 ou les trois premiers caractères de 'hello world' qui retourne un nombre, une chaîne de caractères ou une autre valeur, ainsi que vrai et faux.

Variables [1]

- Peuvent être utilisées dans l'en-tête ou le corps d'une clause et dans les buts entrés à l'invite du système.
- Leur interprétation dépend de l'endroit où ils sont utilisés.
- Variables dans les requêtes
  - Peuvent être interprétées comme "trouver les valeurs des variables qui permettent d'atteindre l'objectif".
  - Par exemple, l'objectif:

    ```
    ?-large_animal(A).
    ```

  trouve une valeur de `A` telle que `large_animal (A)` soit satisfaite.

Variables [2]

- Binding Variables
  - Toutes les variables utilisées dans une clause sont dites non liées (unbound): elles n'ont pas de valeurs.
  - Lorsque le système Prolog évalue un but, certaines variables peuvent recevoir des valeurs : Binding Variables.
  - Une variable qui a été liée peut redevenir non liée et éventuellement liée à une valeur différente par le processus de backtracking.

Portée lexicale des variables

- Dans une clause:

```
parent (X, Y): - pere (X, Y)
```

  - les variables X et Y n'ont complètement aucun lien avec d'autres variables du même nom utilisées dans un autre endroit.
  - Toutes les occurrences des variables X et Y dans la clause peuvent être remplacées de manière cohérente par toute autre variable.
  - Par exemple:

```
parent(First_person,Second_person):-
                                pere(First_person,Second_person)
```

Variables universellement quantifiées

- Si une variable apparaît dans l'en-tête d'une règle ou d'un fait, cela signifie que la règle ou le fait s'applique à toutes les valeurs possibles de la variable.
- Par exemple:

```
large_animal(X):-dog(X),large(X)
```

  - "pour toutes les valeurs de X, X est un grand animal si X est un chien et X est grand".
  - La variable X est dite universellement quantifiée.

Variables Existentiellement Quantifiées

- Exemple:

```
person(frances,wilson,female,28,architect).
person(fred,jones,male,62,doctor).
person(paul,smith,male,45,plumber).
person(martin,williams,male,23,chemist).
person(mary,jones,female,24,programmer).
person(martin,johnson,male,47,solicitor).
man(A):-person(A,B,male,C,D).

?- man(paul).
true.
```

Variables Existentiellement Quantifiées

- Si une variable, disons Y, apparaît dans le corps d'une clause mais pas dans sa tête, elle est interprétée comme signifiant «il y a (ou il existe) au moins une valeur de Y».
- On dit que ces variables sont quantifiées existentiellement.

Variable anonyme

- Le caractère _ désigne une variable spéciale appelée variable anonyme.
  - L'tilisateur n'a pas besoin de savoir la valeur de la variable.

```
?-person(paul,Surname,Sex,Age,Occupation).
Surname=smith,
Sex=male,
Age=45,
Occupation=plumber
?- person(paul,_,_,_,_).
true.
? - personne (paul, Surname, _, _, _).
Surname=Smith.
true.
```

Exercice 3

1. **Tapez le programme suivant dans un fichier et chargez-le dans Prolog.**

```
/* Animals Database */
  animal(mammal,tiger,carnivore,stripes).
  animal(mammal,hyena,carnivore,ugly).
  animal(mammal,lion,carnivore,mane).
  animal(mammal,zebra,herbivore,stripes).
  animal(bird,eagle,carnivore,large).
  animal(bird,sparrow,scavenger,small).
  animal(reptile,snake,carnivore,long).
  animal(reptile,lizard,scavenger,small).
```

2. **Définissez et testez des buts pour trouver**
   a. tous les mammifères,
   b. tous les carnivores qui sont des mammifères,
   c. tous les mammifères avec des rayures,
   d. s'il y a un reptile qui a une crinière.

Exercice 4

1. Tapez le programme suivant dans un fichier et chargez-le dans Prolog.

```
/* Dating Agency Database */
  person(bill,male).
  person(george,male).
  person(alfred,male).
  person(carol,female).
  person(margaret,female).
  person(jane,female).
```

2. Ajoutez dans le programme: une règle qui définit un **couple** de prédicats avec deux arguments, le premier étant le nom d'un homme et le second, le nom d'une femme.
3. Chargez votre programme dans Prolog et testez-le.

Question?