

Les principales caractéristiques des langages de la Programmation Logique

1. Caractéristiques du langage.

- Logique : le langage est un sous-ensemble de la logique et une exécution est une preuve.
- Déclaratif : le “que faire” plutôt que le “comment faire”.
- Relationnel : un programme logique décrit un état du “monde” en termes de données et de prédicats (relations) entre ces données.
- Indéterministe : le résultat est l’ensemble des données qui vérifient une question dans une description donnée du “monde”.
- Symbolique : les données manipulées sont principalement des symboles.
- Haut niveau : aucune gestion de la mémoire et masquage du caractère impératif de la machine.
- Les variables logiques, ne pas confondre avec les variables classiques qui représentent des emplacements mémoire.

N.B. La variable logique n’est pas un emplacement mémoire , et la substitution n’est pas une affectation comme en Pascal.

Tant qu'on n'affecte pas de valeur à une variable, on parle de « variable libre ». Une fois qu'une valeur est affectée, on parle de « variable liée ». On affecte une valeur une variable par ce que l'on appelle le principe d'« unification ».

- En Prolog, il n'y a pas de structures itératives telles que **for, while, do ... while**, qui sont propres aux langages impératifs.
- Par conséquent, en Prolog, toutes les fonctionnalités sont programmées de manière récursive
- Une règle ressemble à une déclaration de procédure classique $C \leftarrow A, B$
C est nom de la procédure A B noms de procédures à appeler

Langage procédural : où tout est défini, impérativement (C, Pascal).

Langage déclaratif : où on exprime ce qu’on veut (langage fonctionnel Lisp, langage logique).

2. comparatif entre programmation logique et impérative.

Programmation = **Logique** + contrôle

Programmation logique	Programmation impérative.
Logique	impérative
Formule	procédure
Ensemble (conjonction) de formules	programme
Question	appel de procédure
Preuve	exécution
Substitution (unification)	passage de paramètres

Contrôle : orienter et modifier le déroulement de la preuve.

Exemple : ?- pere(ali, X) procédure pere qui calcule les enfants de ali

3. Interprétation procédurale des clauses de Horn dans Prolog :

3.1. Faits :

Clause	Procédure sans corps
--------	----------------------

3.2. Règles :

clause	Déclaration de procédure
Tête de clause	Nom de procédure
Corps de clause	Corps de procédure
Termes des prédicats (arguments)	Paramètres formels
Condition (prémisses)	Appel de procédure

p :- p1, p2. Définition de P
 ?- p appel de p

3.3. Buts :

Clause	Procédure sans nom
Corps clause	Appel de procédure

3.4. Clause vide

énoncé d'arrêt.

4. Une étape de résolution :

1. sélectionner un appel quelconque.
2. sélectionner une procédure
3. unifier les paramètres effectifs et les paramètres formels
4. remplacer l'appel par le corps de la procédure
5. appliquer l'unification au nouveau but.

En prolog, la résolution d'un but peut se décomposer en la résolution de sous buts et ainsi de suite.

Prolog essaye les clauses susceptibles de résoudre un but dans l'ordre de leur apparition, selon le procédé de retour arrière.

Prolog examine la première clause et évalue les buts qu'elle contient:

* Prolog évalue le premier but. Ce but est susceptible de retourner :

- 1 solution
- plusieurs solutions
- aucune solution

Lorsqu'un but est susceptible de retourner plusieurs solutions, on dit qu'il y a un **point de choix**.

Le plus souvent en Prolog, on a plusieurs points de choix qui constituent ce qu'on appelle un **arbre décisionnel**.

Lorsqu'un but retourne au moins une solution, Prolog garde la première solution avant d'évaluer les buts suivants.

Lorsqu'un but échoue (pas de solution), Prolog effectue un « **retour arrière** » (ou backtracking) sur le but précédent (et évalue la solution suivante, si elle existe).

S'il n'y a pas de but précédent, la clause échoue.

Lorsque Prolog réussit à examiner tous les (sous) buts contenus dans le corps d'une clause, alors le prédicat retourne une solution.

Lorsque Prolog a fini d'évaluer une clause (qu'elle réussisse ou qu'elle échoue), Prolog examine la clause suivante, à la recherche d'autres solutions

Il existe un **OU inclusif entre les clauses** d'un prédicat Prolog (disjonction) ce qui constitue un paquet de clauses.

Par ex. le prédicat *grandpere* dispose d'un paquet de 2 clauses.

Il existe un **ET entre les buts** d'une clause.

Par ex., une question peut être composée :

?- pere(X,jamila), pere(X,kader) « trouver X tel que X est le père de jamila *et* kader »

De ce fait, le contrôle de cette exécution est similaire à un appel de procédures récursif ce qui nécessite l'utilisation d'une pile à plusieurs niveaux,

Pour le backtracking comment procède et que doit utiliser prolog ?

Les mécanismes de retour arrière provoquent un dépilement, afin de gérer les recherches de solutions en profondeurs et les retours !

4.1. Interprétation de l'unification.

L'unification est :

- ✓ un mécanisme de transmission des paramètres
- ✓ moyen de construction des valeurs pour les variables effectives d'un but.

L'ordinateur utilise la méthode syntaxique pour la résolution des buts. L'interprétation échappe à l'ordinateur, elle est perçue par l'humain.

4.2. Méthode syntaxique :

C'est un procédé qui consiste à dériver de nouvelles formules à partir de P

Une démonstration syntaxique est une suite de formules

Chaque formule est soit :

- ✓ Une formule de P
- ✓ Un axiome de P ou la méthode syntaxique
- ✓ Une inférence de formules déjà établies grâce à des règles d'inférences de la méthode syntaxique.

4.3. Exemple : soit le programme P

F1 : p1
 F2. p2
 F3. p3
 F4. p4
 R1 p5 :- p1,p2.
 R2 p6 :- p5,p3.
 Q1 ?- p6,p4.

Démonstration syntaxique :

?- p6,p4 l'appel de p6 est activé
 ?- p5,p3,p4 S1 p6 remplacé par p5,p3
 ?-,p1,p2, p3,p4 S2 p5 remplacé par p1,p2
 ?-,p2 p3;p4 S3 p1 résolu unifié par substitution vide
 ?- p3,p4 S4 p2 résolu
 ?- p4 S5
 ?-

S1S2S3S4S5 donne une valeur pour chaque variable effectif de p6,p4,

P6 est une conséquence logique du programme P

La démonstration de p4 nécessite 1 étape car c'est un fait