Programmation en logique et par contraintes
Cours 1

Thi-Bich-Hanh Dao

M1 Informatique - Université d'Orléans

Année 2012-2013

Le module PLC

Organisation

- Cours : 11 séances de 1h30 le mardi de 10h15 à 11h45
- TD : 10 séances de 2h le vendredi de 15h45 à 17h45, séances de TD avec une pratique éventuelle sur ordinateur

Les intervenants

- Cours : Thi-Bich-Hanh Dao
- TD : Abdel-Ali Ed-Dbali

Evaluation

Contrôle continu

- Un ou deux contrôles sur machine en TD
- Un contrôle écrit au milieu du semestre
- L'absence à un contrôle continu : 0

Contrôle terminal

- Contrôle écrit à la fin du semestre
- L'absence au contrôle terminal : ABI

Note finale

- Note finale = moyenne pondérée des CC et du CT
- Attention : note éliminatoire si inférieur à 7

Ressources

- L'art de Prolog, *Leon Sterling, Ehud Shapiro*
- Programmation logique par contraintes, *François Fages*
- Programming with Constraints : An Introduction, *Kim Marriott, Peter J. Stuckey*

Contenu du module

1. Programmation logique en Prolog
2. Sémantiques opérationnelle et déclarative, leur relation
3. Listes en Prolog
4. La coupure, la négation
5. Prédicats du second-ordre
6. Programmation non déterministe
7. Programmation logique avec contraintes
8. Contraintes des domaines finis

Le module au sein du Master INIS

M1

- Modélisation&vérification : vision et outils complémentaire pour modéliser et résoudre des problèmes
- Compilation : outil expressif pour étudier des grammaires
- Intelligence artificielle : outil de base pour l'implantation des algorithmes

M2

Les connaissances en PLC sont nécessaires ou utiles pour

- Programmation par contraintes
- Extraction de connaissances dans les BD
- Fouille de données et de textes

Plan du cours 1

1. Introduction
2. Syntaxe de Prolog
3. Entiers en Prolog
4. Termes et prédicats

I. Introduction

Des langages de programmation

Popularité des langages 9/2012 (Source TIOBE)

- C : 19,29%
- Java : 16,26%
- C++ : 9,14%
- Ada : 0,7%
- Prolog : 0,33%

Popularité par catégories :

- Langages orienté-objet : 57,1%
- Langages procéduraux : 38,1%
- Langages fonctionnels : 3,2%
- Langages logiques : 1,7%

I. Introduction

Prolog : un concept différent de programmation

- langage relationnel, logique
- déclaratif
- adapté aux problèmes décisionnels, d'Intelligence artificielle
- nouvelle façon de programmer
- excellent support des contraintes

I. Introduction

Comparaison

	impératif	fonctionnel	logique
programme	suite d'instructions	suite de définitions de fonction	suite de définitions de prédicats
exécution	modification de la mémoire	réécriture	recherche de preuves
résultat	état final de la mémoire	forme normale	valeur des variables du but
modèle de calcul	machine de Turing	λ-calcul	calcul des prédicats

I. Introduction

La programmation logique en deux mots

- programmer avec des relations
- programme = description de relations (prédicats) + requête
- pas d'affectation, pas d'itération
- une seule SDD : l'arbre
- récursivité

I. Introduction

Que peut-on faire avec la PLC ?

- Gestion de production : séquencement d'une ligne d'assemblage de voitures
- Distribution et logistique : localisation d'entrepôt
- Finance : gestion de portefeuille d'actions
- Diagnostic : localisation de défauts dans des circuits numériques
- Conception des circuits VLSI, vérification formelle des circuits logiques
- Planification : planification de production, de livraison, de ligne d'assemblage
- Gestion de ressources : rotation d'équipes de personnel
- Traitement de langues : construction d'analyseur efficace
- Système expert

I. Introduction

Bref historique

- Unification : J.A. Robinson, 1965
- Prolog : A. Colmerauer, 1972
- CLP(R) : J. Jaffar, J-L. Lassez, 1987
- CHIP : P. Van Hentenryck, 1989
- Différents langages : Prolog IV, Gnu-Prolog, Sicstus Prolog, Swi-Prolog, Yap, ...

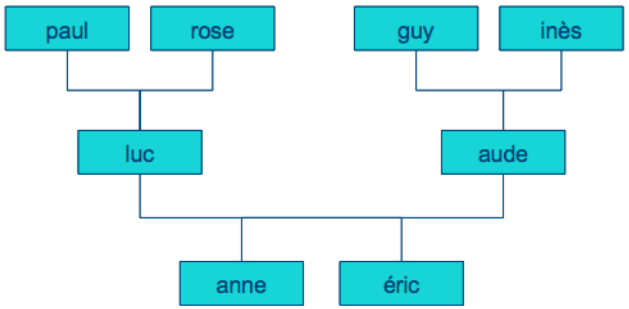
I. Introduction

Langages disponibles gratuitement

- Gnu-Prolog, Daniel Diaz
  http ://www.gprolog.org
- Yap (Yet Another Prolog), Université de Porto
  http ://www.ncc.up.pt/~vsc/Yap/
- Swi-Prolog, Université d'Amsterdam
  http ://www.swi-prolog.org

I. Introduction

Un premier programme : BDD parentale



- Définir la relation père, mère, enfant
- Des relations plus générales : un parent, les parents, grand père, un descendant
- Requêtes au programme

II. Syntaxe

Faits et règles

- Faits (assertions) : le type d'énoncé le plus simple, moyen d'affirmer qu'une relation a lieu entre des objets.
  `mere(rose,luc).`
- Faits universels : les variables sont aussi utiles dans les faits
  `plus(0,X,X).`
- Règle
  `un_parent(P,E) :- pere(P,E).`
  `un_parent(P,E) :- mere(P,E).`
  plusieurs définitions pour le même prédicat, d'une façon implicite, `P,E` sont quantifiés universellement
- Exemple : BDD parentale, relation ascendant, descendant

II. Syntaxe

Requêtes

- Requête (question) : moyen de retrouver de l'information dans un programme.
  `mere(rose,luc).`
  Réponse `yes` ou `no` à cette question
- Les prédicats qui composent une question sont appelés des buts.
- Une question simple consiste en un seul but.
- Les questions simples ont le même syntaxe que les faits, ils sont différenciés par le contexte.
- Questions existentielles : d'une façon implicite, les variables dans les questions ont quantifiées existentiellement
  `mere(X,luc),pere(Y,luc).`

III. Les entiers

Quelques symboles de fonctions

- + l'addition
- - la soustraction
- * la multiplication
- / la division
- ** la puissance
- // la division entière
- `mod` le reste de la division entière

III. Les entiers

Prédicat `is`

```
?- X=3+4.
X=3+4
?- X is 3+4.
X=7
?- 7 is 3+4.
yes
?- 3+4 is 3+4.
no
?- X is Y+1.
error
?- Y = 2, X is Y + Y.
X = 4
Y = 2
```


III. Les entiers

Quelques prédicats

- `X < Y`, `X > Y` , `X =< Y`, `X >= Y`
- `X =:= Y` les valeurs de `X` et `Y` sont égaux
- `X =\= Y` les valeurs de `X` et `Y` sont différents

Les deux arguments doivent être clos au moment d'évaluation.

```
?- X is 3+4, X =:= 7.
X = 7
?- X =:= 7, X is 3+4.
erreur d'instanciation
```

Attention = est différent de =:=

IV. Termes et prédicats

- Les entités, les objets dont on parle : les termes, construits à l'aide des symboles de fonction.
- Les relations entre objets : les prédicats, construits à l'aide des symboles de prédicat.
- Exemple :
  Toto est un cheval. Toto est un parent de Titi. Toto aime les pommes. Les chevaux et les moutons sont des animaux. Chaque animal a un parent.
  - Les objets (symboles de fonction) :
    `toto/0, titi/0, pommes/0`
  - Les relations entre objets (symboles de prédicats) :
    `cheval/1, mouton/1, animal/1, parent/2, aime/2`
  - Définition des prédicats : ...

IV. Termes et prédicats

Un autre exemple

Un arbre binaire est soit une feuille, soit un nœud dont les deux fils sont des arbres binaires.
La hauteur d'une feuille est 0, la hauteur d'un nœud est le max des hauteurs de ses fils plus 1.
La taille d'une feuille est 1, la taille d'un nœud est la somme des tailles de ses fils plus 1.

IV. Termes et prédicats

Un autre exemple

Un arbre binaire est soit une feuille, soit un nœud dont les deux fils sont des arbres binaires.
La hauteur d'une feuille est 0, la hauteur d'un nœud est le max des hauteurs de ses fils plus 1.
La taille d'une feuille est 1, la taille d'un nœud est la somme des tailles de ses fils plus 1.

- Symboles de fonction : `feuille/0, noeud/2`
- Présentation d'arbres binaires : `feuille`,
  `noeud(feuille,feuille)`,
  `noeud(feuille,noeud(feuille,feuille))`, ...
- Symboles de prédicats : `hauteur/2, taille/2`

IV. Termes et prédicats

Il est nécessaire de distinguer entre *prédicats* et *termes*

- Terme : pour désigner un objet, un élément
  `feuille, noeud(feuille,feuille),`
  `noeud(X,noeud(feuille,feuille)), ...`
- Prédicat : pour représenter une relation entre des éléments
  `hauteur(feuille,0)`
- Il n'y a pas de différence syntaxique entre prédicats et termes : les termes aussi peuvent avoir des arguments
  `hauteur(noeud(X,Y),H)`