PLC – Cours 2

Thi-Bich-Hanh Dao

M1 Informatique - Université d'Orléans

Année 2012-2013

Plan

1. Spécifier un prédicat : argument d'entrée et de sortie
2. Sémantique opérationnelle
3. Sémantique déclarative
4. Relation entre les sémantiques

I. Spécifier un prédicat

Exemple : entiers présentés par des termes

- Les entiers naturels sont présentés en notation unaire par les termes :
  `0,  s(0),  s(s(0)),  s(s(s(0))), ...`
- Prédicat `entier` : tester si un terme est un entier *ou* générer des termes-entier, en fonction de l'argument
- Prédicats `plus` et `mult` : différentes utilisations

I. Spécifier un prédicat

Argument d'entrée et de sortie

- Un argument est d'entrée pour un prédicat si, au moment de l'exécution du prédicat, l'argument doit être instancié par un terme clos (sans variable).
- Un argument est de sortie pour un prédicat s'il doit être une variable et au moment où l'exécution du prédicat se termine, l'argument sera instancié.
- + : argument d'entrée
- - : argument de sortie
- ? : argument d'entrée ou de sortie
- Exemple : les spécifications possibles de `entier` sont `entier(+Arg)` et `entier(-Arg)`, donc `entier(?Arg)`

I. Spécifier un prédicat

Relation et relation inverse

- Il est parfois possible d'utiliser un prédicat pour calculer une relation R et sa relation inverse R^{-1}.
- Exemple 1 :

```
?- mult(s(s(zero)),s(s(s(zero))),X).
X = s(s(s(s(s(s(zero)))))) ? ;
no
?- mult(s(s(zero)),X,s(s(s(s(s(s(zero))))))).
X = s(s(s(zero))) ? ;
no
?- mult(X,s(s(zero)),s(s(s(s(s(s(zero))))))).
X = s(s(s(zero))) ? ;
boucle
```

- Exemple 2 : transformer un terme représentant un nombre en nombre. Programme : transform.pl

II. Sémantique opérationnelle

Un exemple

Programme :

```
q(a).
p(X) :- q(X).
p(b).
```

But : `p(X).`

- Les dérivations
- Arbre de recherche

II. Sémantique opérationnelle

L'unification

- Une *substitution* est une fonction partielle qui associe des termes à des variables $\{t_1/v_1, \ldots, t_n/v_n\}$.
  Ex : $\sigma = \{zero/X, succ(T)/Y\}$
- L'application d'une substitution σ à un terme t, notee $t\sigma$, est le terme dans lequel les variables de σ sont remplacées par les termes correspondants.
  Ex : $\texttt{add(X,Y)}\sigma = \texttt{add(zero,succ(T))}$
- Deux termes t et s sont *unifiés* par la substitution σ si $t\sigma$ et $s\sigma$ sont identiques.
- La substitution σ est *l'unificateur le plus général (mgu)* de t et s s'il unifie t et s et tout autre unificateur θ est de la forme $\sigma \circ \sigma'$.

II. Sémantique opérationnelle

Exemples d'unification

```
?- f(X)=f(g(a,Y)).
X=g(a,Y)
?- f(X,X)=f(g(a,Y),g(Z,b)).
X = g(a,b)
Y = b
Z = a
?- X=f(X).
Segmentation fault
```

II. Sémantique opérationnelle
Arbre de recherche

On considère un programme $P = c_1, \ldots, c_k$ et une requête $G = g_1, \ldots, g_l$. Soit $c_i = t_i :- b_i^1, b_i^2, ..., b_i^{n_i}$. (si $n_i = 0$ alors c_i est un fait sinon c'est une règle).
On définit nœuds et arcs de l'arbre de recherche de G pour P par induction sur la hauteur :

- La racine est G.
- Soit $H = h_1, ..., h_j$ un nœud de hauteur n et soit c_s une clause de P dont la tête t_s s'unifie avec h_1 avec mgu σ.
  Alors on crée le nœud $H' = b_s^1\sigma, ..., b_s^{n_s}\sigma, h_2\sigma, ..., h_j\sigma$, de hauteur $n+1$, et on étiquette l'arc de H à H' par σ.

Une feuille est soit un but vide (succes), soit un but dont le premier prédicat ne s'unifie avec aucune tête de clause (echec).

II. Sémantique opérationnelle
Exemples d'arbre de recherche

Avec le programme de transformation entre terme et entier, construire l'arbre de recherche pour les requêtes suivantes :

```
?- tr(X,1)
?- tr(s(0),X)
?- trans(s(s(0)),X).
?- trans(X,2).
?- trans2(s(s(0)),X).
?- trans2(X,2).
```


II. Sémantique opérationnelle
Exploration d'arbre de recherche

- L'exécution d'une requête G pour un programme P a comme résultat l'ensemble de feuilles succes de l'arbre de recherche correspondant.
  Plus précisément, pour chacun de ces feuilles, le résultat est l'ensemble des instantiations des variables de G qui se trouvent sur le chemin qui mène de la racine à la feuille en question.
- Exploration en profondeur, l'ordre des clauses est important.
  ```
  p(X) :- p(X). p(a)
  p(a). p(X) :- p(X).
  ```


II. Sémantique opérationnelle
L'ordre de prédicats est important

- BDD parentale et deux prédicats :
  ```
  ascendant(A,P) :-              ascendant2(A,P) :-
      parent(A,P).                   parent(A,P).
  ascendant(A,P) :-              ascendant2(A,P) :-
      parent(X,P),                   ascendant2(A,X),
      ascendant(A,X).                parent(X,P).
  ```
- Quelles sont les réponses pour les requêtes suivantes ?
  ```
  ?- ascendant(A, aude).
  ?- ascendant2(A, aude).
  ```

III. Sémantique déclarative

Logique du 1er ordre : la syntaxe (1)

Soit V un ensemble dénombrable de symboles de variables, par exemple $V = \{x, y, z, \ldots, x_1, \ldots\}$.
Un *alphabet* F, P consiste de deux ensembles de symboles : fonction et prédicat. Les éléments de F et P ont chacun une arité prédéfinie. Les éléments de F d'arité 0 sont les constantes, les éléments de P d'arité 0 sont les constantes de prédicat.
Par exemple, un alphabet pour les entiers en notation unaire pourrait être $F_{int} = \{zero/0, succ/1\}$, $P_{int} = \{pair/1, plus/3\}$

III. Sémantique déclarative

Logique du 1er ordre : la syntaxe (2)

L'ensemble de *termes* sur un alphabet F, P donné est défini inductivement par :

- Tout élément de V est un terme.
- Tout élément de F d'arité 0 est un terme.
- Si $t_1, \ldots, t_n$ sont des termes et $f \in F$ d'arité n, alors $f(t_1, \ldots, t_n)$ est un terme.

Exemples de termes sur F_{int}, P_{int} :
$x,\ zero,\ succ(succ(zero)),\ succ(succ(z))$

III. Sémantique déclarative

Logique du 1er ordre : la syntaxe (3)

L'ensemble de *formules* sur un alphabet F, P donné est défini inductivement par :

- Si $p \in P$ d'arité n et $t_1, \ldots, t_n$ sont des termes, alors $p(t_1, \ldots, t_n)$ est une formule (un atome).
- Si G et H sont des formules, alors $\neg G$, $G \wedge H$, $G \vee H$, $G \rightarrow H$ sont des formules.
- Si $x \in V$ et G est une formule, alors $\forall x G$ et $\exists x G$ sont des formules.

Exemples de formules sur F_{int}, P_{int} :
$pair(x), pair(zero) \wedge \neg pair(succ(zero))$,
$\forall x (plus(x, y, z) \rightarrow plus(succ(x), y, succ(z)))$

III. Sémantique déclarative

Les clauses

Un *littéral* est un atome ou la négation d'un atome. Une *clause* est une formule de la forme :

$$\forall x_1 \ldots \forall x_n (L_1 \vee \cdots \vee L_k)$$

où les L_i sont des littéraux et les x_j sont toutes et seules les variables de la formule.
La clause $\forall x_1 \ldots \forall x_n (A_1 \vee \cdots \vee A_i \vee \neg B_1 \vee \cdots \vee \neg B_j)$ où $A_1, \ldots, A_i, B_1, \ldots, B_j$ sont des atomes, est notée

$$A_1, \ldots, A_i \ \leftarrow \ B_1, \ldots, B_j$$

et se lit : si B_1 et ... et B_j alors A_1 ou ... ou A_i.
Si dans la clause précédente, $i \leq 1$ alors c'est une clause de Horn.

III. Sémantique déclarative

Clauses de Horn

- Clause de Horn : $A \leftarrow B_1, \ldots, B_k$
  correspondent à une clause de programme (ou règle)
  `A :- B1,...,Bk.`
- Clause de Horn : $A \leftarrow$
  correspondent à une clause unitaire (ou fait, ou assertion) :
  `A.`
- Clause de Horn :
  $\leftarrow B_1, \ldots, B_k$
  correspondent à un but :
  `?- B1,...,Bk.`

Un programme logique (défini) est un ensemble fini de règles et de faits.

III. Sémantique déclarative

Interprétation

Une interprétation $\mathcal{I}$ d'un langage du premier ordre sur un alphabet F, P consiste en :

- Un ensemble D (le domaine de l'interprétation).
- Pour tout symbole de fonction f d'arité n, $\mathcal{I}(f) : D^n \rightarrow D$.
- Pour tout symbole de prédicat p d'arité n, $\mathcal{I}(p) \subseteq D^n$.

Exemple : F_{int}, P_{int}, une interprétation pourrait être :
$D = Nat$, $\mathcal{I}(zero) = 0$, $\mathcal{I}(succ) : n \mapsto n+1$, $\mathcal{I}(pair) = \{n | n \text{ est pair }\}$,
$\mathcal{I}(plus) = \{(x, y, z) \in Nat^3 | x + y = z\}$
mais aussi être
$D = \{a\}$, $\mathcal{I}(succ) : x \mapsto x$, $\mathcal{I}(pair) = \mathcal{I}(plus) = \emptyset$

III. Sémantique déclarative

Interprétation des termes

Soit $\mathcal{I}$ une interprétation de F, P de domaine D. Une affectation est une fonction $\rho : V \rightarrow D$.
La valeur $\mathcal{I}_\rho(t)$ des termes t dans l'interprétation $\mathcal{I}$ et l'affectation ρ est définie comme suit :

- $\mathcal{I}_\rho(x) = \rho(x)$.
- $\mathcal{I}_\rho(c) = \mathcal{I}(c)$, pour tout constante c.
- $\mathcal{I}_\rho(f(t_1, ..., t_n)) = \mathcal{I}(f)(\mathcal{I}\rho(t_1, ..., \mathcal{I}_\rho(t_n))))$, pour tout $f \in F$ d'arité n.

III. Sémantique déclarative

Interprétation des formules

La valeur $\mathcal{I}_\rho$ des formules dans l'interprétation $\mathcal{I}$ et l'affectation ρ est définie comme suit :

- $\mathcal{I}_\rho(p(t_1, ..., t_n)) = \begin{cases} vrai & \text{si } (\mathcal{I}_\rho(t_1), ..., \mathcal{I}_\rho(t_n)) \in \mathcal{I}(p) \\ faux & \text{sinon} \end{cases}$
- $\mathcal{I}_\rho(\neg G)$, $\mathcal{I}_\rho(G \wedge H)$, $\mathcal{I}_\rho(G \vee H)$, $\mathcal{I}_\rho(G \rightarrow H)$ sont définis en fonction de $\mathcal{I}_\rho(G)$ et $\mathcal{I}_\rho(H)$ avec des tables de vérités usuelles.
- $\mathcal{I}_\rho(\forall x G) = vrai$ si et seulement si pour tout $d \in D$, $\mathcal{I}_{\rho[x \leftarrow d]}(G) = vrai$
- $\mathcal{I}_\rho(\exists x G) = vrai$ si et seulement si il existe $d \in D$, $\mathcal{I}_{\rho[x \leftarrow d]}(G) = vrai$

III. Sémantique déclarative

Conséquence logique

- Un *modèle* d'une formule G est une interprétation $\mathcal{I}$ telle que pour tout ρ, $\mathcal{I}_\rho(G) = vrai$.
- Un modèle d'un ensemble de formules est une interprétation qui est modèle de chaque formule de l'ensemble.
- Une formule G est *conséquence logique* d'un ensemble S si tout modèle de S est aussi modèle de G.
- La *résolution* permet de démontrer toutes les conséquences logiques d'un ensemble de clauses (modèles de Herbrand, théorème de Herbrand).

III. Sémantique déclarative

Sémantique déclarative d'un programme logique

Soit P un programme logique. La sémantique déclarative de P est l'ensemble des atomes clos qui sont conséquences logiques de P.
Exemple : la sémantique déclarative du programme
$pair(zero)$
$pair(succ(succ(X))) \leftarrow pair(X)$
est l'ensemble d'atomes

$$\left\{ \begin{array}{l} pair(zero), \\ pair(s(s(zero))), \\ pair(s(s(s(s(zero))))), \\ \ldots \end{array} \right\}$$

IV. Relation entre les sémantiques

Sémantique opérationnelle

Soit P un programme. L'ensemble succes de P est l'ensemble des atomes clos A tels que l'arbre de recherche de racine A possède au moins une branche succès. Cet ensemble est la sémantique opérationnelle de P.
Exemple : l'ensemble succès du programme

```
pair(zero).
pair(s(s(X))) :- pair(X).
```

est l'ensemble d'atomes

$$\left\{ \begin{array}{l} pair(zero), \\ pair(s(s(zero))), \\ pair(s(s(s(s(zero))))), \\ \ldots \end{array} \right\}$$

IV. Relation entre les sémantiques

Correction et complétude

Soit P un programme, on note
$decl(P)$ sa sémantique déclarative
$op(P)$ sa sémantique opérationnelle.

Théorème de correction et complétude

Pour tout programme logique P, $decl(P) = op(P)$.

IV. Relation entre les sémantiques

Problèmes liés à la complétude

- Soit $ef(P)$ l'ensemble des atomes clos sur lesquels l'interpréteur Prolog termine avec succès.
- Pour des raisons pratiques, $ef(P) \subset op(P)$.
- Exemple : pour le programme P
  $p(X) \leftarrow p(X)$
  $p(a)$
  on a $decl(P) = op(P) = \{p(a)\}$, mais $ef(P) = \emptyset$.
- Ceci peut être du à :
  - La règle de sélection de clauses.
  - Prédicats primitifs "extra-logique" qui "coupent" des parties de l'arbre de recherche.