Sémantique de la programmation logique

Didier Buchs

Université de Genève

29 avril 2019

Changement de Résolution

- Evaluation 'complète' des expressions
- choix plus aléatoires des valeurs
- Comment procéder ?
  - Etablir une syntaxe pour les clauses de Horn
  - Etablir une sémantique de la résolution générale
  - Construire une implémentation de la résolution générale en Prolog

Symboles fonctionnels

- $Func = (a, b, c, \ldots, z)$
- $Arite$ =nombre de paramètres $\in (0, 1, \ldots n)$

Ensemble de Herbrand, termes

- Tous les termes constructibles sur $f \in \wp(Func)$: $Herb(f)$
- $f_i \in f$, Si n arité de f_i,
  $t_1 \in Herb(f), \ldots, t_n \in Herb(f) \Rightarrow f_i(t_1, t_2, \ldots, t_n) \in Herb(f)$

Noms des prédicats

- $Pred = (aa, bb, cc, \ldots, zz)$

Litéraux

- $x \in \wp(Pred)$ et $f \in \wp(Func)$ tous les litéraux constructibles : Lit(x,f)
- $x_i \in x$, si n arité de x_i,
  $t_1 \in Herb(f), \ldots, t_n \in Herb(f) \Rightarrow x_i(t_1, t_2, \ldots, t_n) \in Lit(x, f)$
- Conjonction de litéraux, l'ensemble $c \in \wp(Lit(x, f))$ décrit une conjonction de litéraux

Sémantique clauses de Horn : syntaxe

Variables

- X un ensemble de variables (implicitement d'arité 0)

Clauses

- Soit X des variables, $p \in \wp(Pred)$, $f \in \wp(Func)$, $Clauses(p, f, X)$ est l'ensemble des clauses constructibles
- $c \in \wp(Lit(p, f \cup X))$, $h \in Lit(x, f \cup X)$ alors $c \Rightarrow h \in Clauses(p, f, X)$

Requêtes

- Une requête est une conjonction de litéraux, $r \in \wp(Lit(p, f \cup X))$

Sémantique clauses de Horn : substitution et unification

Pour un programme prolog constitué de règles : *Axiom* l'évaluation d'un pas de résolution se fait sur $< substitution, requete >$

Definition (Substitution)

$t, t' \in Lit(p, f \cup X)$
soit t et $s : X \rightarrow Lit(p, f \cup X)$
$st = t'$ la substitution remplace les variables par les termes dans t' depuis t

Definition (Unification)

soit $t, t' \in Lit(p, f \cup X)$
il existe $s : X \rightarrow Lit(p, f \cup X)$
$st = st'$ l'unification cherche une substitution qui égalise les termes

Sémantique clauses de Horn : règle de résolution

Pour un programme prolog constitué de règles : *Axiom* l'évaluation d'un pas de résolution se fait sur $< substitution, requete >$

Definition (Sémantique computationelle)

$r, c \in \wp(Lit(p, f \cup X))$ et $h, \rho \in Lit(p, f \cup X)$

Rsol : $\frac{c \Rightarrow h \in Axiom, s'\rho = s'h}{<s, \{\rho\} \cup r> \longrightarrow <s \circ s', s'c \cup s'r>}$

Points de non-déterminismes

- choix des axiomes : $c \Rightarrow h \in Axiom$
- choix du litéral à résoudre : $\{\rho\} \cup r$
- choix de l'unificateur (en fait MGU en Prolog) : $s'\rho = s'h$

Exemple :

MGU ?

- choix de l'unificateur (en fait MGU en Prolog) : $s'\rho = s'h$

Exemple :

```
?- node(X,a)=node(node(2,Z),T).

X= node(2,Z)
T= a
```

En réalité :

```
X= node(2,node(3,a))
T= a
X= node(2,node(node(3,a),a))
T= a
...
```

Sémantique globale

- Fermeture transitive de la relation $\longrightarrow$
- Si il y a une solution : forme canonique avec $< S, \emptyset >$
- Evaluation d'une requête : $< \emptyset, r > \longrightarrow^* < S, \emptyset >$

Prolog est un cas particulier : Résolution SLD ! (en fait on remplace les ensembles par des listes ordonnées)

Definition (Sémantique d'évaluation)

$r, c \in \wp(Lit(p, f \cup X))$ et $h, \rho \in Lit(p, f \cup X)$

Rsol : $\dfrac{c \Rightarrow h \in Axiom, s'\rho = s'h, < s \circ s', s'c \cup s'r > \Longrightarrow < s'' >}{< s, \{\rho\} \cup r > \Longrightarrow < s'' >}$

Rbase : $\dfrac{}{< s, \emptyset > \Longrightarrow < s >}$

L' unification en général

Il s'agit de trouver la solution au système d'équations
$U \equiv V \Leftrightarrow \exists s$ une substitution, t.q. $sU = sV$

Une substitution est une application des variables dans les termes :
$s = \{X = node(Z, 3); Y = b\}$

```
X = node(Z,3)
Y = b
```

sU est l'application de la substitution au terme U fournissant un terme instancié
$s(node(node(node(X, Y), 3), node(node(a, b), c))) =$
$node(node(node(node(Z, 3), b), 3), node(node(a, b), c))$

L' unification en général

- Combien de solutions (les plus générales) ?
- Théorie unitaire $= 1$
  Unification de termes libres comme en Prolog
- Théorie finitaire $=$ un nombre fini
  Unification d'ensembles :
  $\{X, Y\} = \{a, b\}$ solution ?
- Théorie infinitaire $=$ une infinité
  Unification de fonctions surjectives !
  Odd(X) $=$true solution ?

Unification, implémentations des différentes variantes

Unitaire et finitaire alors possibilités d'implémentation (sémantique opérationelle)

Definition (Sémantique computationelle)

$r, c \in \wp(Lit(p, f \cup X))$ et $h, \rho \in Lit(p, f \cup X)$
$\forall s' \in subs$

$$\text{Rsol} : \frac{c \Rightarrow h \in Axiom, s'\rho = s'h}{<s, \{\rho\} \cup r> \longrightarrow <s \circ s', s'c \cup s'r>}$$

Implémentation de la règle de résolution :Prolog en Prolog

```
solveSLD([]).

solveSLD([H|T]):-axiom(H,RESTE),solveSLD(RESTE),solveSLD(T).
```

Sémantique clauses de Horn : règle de résolution

```
axiom(a(X,Y),[b(X),c(Y)]).
axiom(b(1),[]).
axiom(b(2),[]).
axiom(c(3),[]).
axiom(c(4),[]).
```

Stratégie 'linéaire'

```
?- solveSLD([a(R,S)]).

R = 1
S = 3 ;

R = 1
S = 4 ;

R = 2
S = 3 ;

R = 2
S = 4 ;

No
```

Résolution 'random'

```
solveRes([]).
solveRes(L):-elem_liste_random(L,H,R),
             bagof(axiom(H,T),H^axiom(H,T),LAXIOMS),
             chooseaxiom(LAXIOMS,H,R).

chooseaxiom([HL|L],H,R):-
             elem_liste_random([HL|L],axiom(HH,RESTE),OTHERS),
             (append(R,RESTE,RR),H=HH,solveRes(RR);
             (OTHERS=[_|_],chooseaxiom(OTHERS,H,R))).
```

Résolution 'random'

```
% choisi aleatoirement un element dans une liste
% elem_liste_random(liste, element choisi, liste privee de l'ele
:- mode(elem_liste_random(+,-,-)).
elem_liste_random([E|L], F, NL) :-
    length([E|L], Length), Lengthi is Length + 1,
    random(1,Lengthi, Ranki),
    (Lengthi = Ranki, Rank = Length,!;Rank = Ranki),   % Hack
    neme_elem(Rank, [E|L], F, NL).

% retourne le neme element d'une liste
:- mode(neme_elem(++,+,-,-)).
neme_elem(1, [E|L], E, L) :- !.
neme_elem(N, [E|L], F, [E|NL]) :-
    M is N-1,
    neme_elem(M, L, F, NL).
```

Résolution ‘random’

```
?- solveRes([a(R,S)]).

R = 1
S = 3 ;

R = 2
S = 3 ;

R = 1
S = 4 ;

R = 2
S = 4 ;

No
```

Creation aléatoire de chemin dans un arbre binaire

```
axiom(path(leaf),[]).
axiom(path(l(P)),[path(P)]).
axiom(path(r(P)),[path(P)]).

axiom(length(leaf,0),[]).
axiom(length(l(P),s(R)),[length(P,R)]).
axiom(length(r(P),s(R)),[length(P,R)]).
```

Creation aléatoire de chemin dans un arbre binaire

```
?- solveRes([path(S)]).

S = r(l(r(leaf))) ;

S = r(l(r(r(leaf)))) ;

S = r(l(r(r(r(leaf))))) ;

S = r(l(r(r(r(l(l(r(r(r(...)))))))))) ;
```

Creation aléatoire de chemins bornés dans un arbre binaire

```
?- solveRes([path(S),length(S,s(s(0)))]).

S = r(r(leaf)) ;

S = r(l(leaf)) ;

S = l(r(leaf)) ;

S = l(l(leaf)) ;
No
```

Résumé

- Différentes méthodes de formulation d'une sémantique
- Caractérisation des correspondances entre sémantiques
- Comment décrire la sémantique en présence de variables, fonctions (notion de contexte).
- Création d'un interprète à partir des règles déclaratives
- Analyse de programmes par exécution symbolique
- Sémantique de Prolog

Discussion sur l'opérationalité des règles d'inférences

- Problématique similaire à la différence entre clauses de Horn et programme Prolog
- est la source d'inefficacité voire d'incomplétude
- Se base sur le fait que :
  - La conjonction de litéraux est commutative
  - Le flux des variables n'est pas obligatoirement 'linéaire', mais induit implicitement des équations 'cachées'
  - éventuellement l'unification n'est pas unitaire

Exemple de la multiplication Egyptienne

cf.

http://rigolmath.free.fr/xegypte/multipli.htm

m * n = ?
Etapes :

$$
\begin{array}{rcl}
facteurs & garde\ facteur(o/n) & somme \\
1 * m = m & O & m \\
2 * m = 2 * m = k_1 & O & k_1 \\
4 * m = 2 * k_1 = k_2 & N & \\
\ldots & & \\
2^l * m = 2 * k_{l-1} = k_l & O & k_l \\
2^{l+1} * m = 2 * k_l = k_{l+1} & (2^{l+1} > n) & S = m + k_1 + \ldots + k_l
\end{array}
$$

Exemple de la multiplication Egyptienne

Règles :

$$\frac{mult(m, n, 2 * l, f, k + k, s + k, r), f + l \leq n, l \leq n}{mult(m, n, l, f + l, k, s, r)}$$

$$\frac{mult(m, n, 2 * l, f, k + k, s, r), f + l > n, l \leq n}{mult(m, n, l, f, k, s, r)}$$

$$\frac{l > n}{mult(m, n, l, 0, k, s, s)}$$

requête : mult(m,n,1,n,m,0,r)

Implémentation

```
mult(_m,_n,_l,_fact,_k,_s,_r):-_fact=<_n,
           _l=<_n,_f is _fact-_l,
           _ll is _l+_l,_km is _k+_k,
           _ss is _s + _k, mult(_m,_n,_ll,_f,_km,_ss,_r),
           print('level : '),print(_l), print(' fact : '),
           print(_fact),print(' k : '),print(_k),nl.
mult(_m,_n,_l,_fact,_k,_s,_r):-_fact+_l>_n,
           _l=<_n,_ll is _l+_l, _km is _k+_k,
           mult(_m,_n,_ll,_fact,_km,_s,_r).
mult(_m,_n,_l,0,_k,_s,_s):- _l>_n.

mult(_m,_n,_r):- mult(_m,_n,1,_n,_m,0,_r).
```