Curs 5

2024-2025

Programare Logică şi Funcţională

Cuprins

Prolog: semantica limbajului IMP

Sintaxa BNF a limbajului IMP

$E ::= n \mid x$
$\quad \mid E + E \mid E - E \mid E * E$

$B ::= \texttt{true} \mid \texttt{false}$
$\quad \mid E =< E \mid E >= E \mid E == E$
$\quad \mid \texttt{not}(B) \mid \texttt{and}(B\ ,\ B) \mid \texttt{or}(B\ ,\ B)$

$C ::= \texttt{skip}$
$\quad \mid x = E$
$\quad \mid \texttt{if}(\ B\ ,\ C\ ,\ C\ )$
$\quad \mid \texttt{while}(\ B\ ,\ C\ )$
$\quad \mid \{\ C\ \} \mid C\ ;\ C$

$P ::= \{\ C\ \},\ E$

Decizii de implementare

- {} si ; sunt operatori

```
:- op(100, xf, {}).
:- op(1100, yf, ;).
```

- definim un predicat pentru fiecare categorie sintactică

```
stmt(while(BE,St)) :- bexp(BE),  stmt(St).
```

- `while`, `if`, `and`, etc sunt functori în Prolog
  `while(true,skip)` este un termen compus

- , are semnificatia obisnuită

- pentru valori numerice folosim întregii din Prolog

```
aexp(I) :- integer(I).
```

- pentru identificatori folosim atomii din Prolog

```
aexp(X) :- atom(X).
```

Prolog: corectitudinea sintactică a limbajului IMP

```
aexp(I) :- integer(I).
aexp(X) :- atom(X).
aexp(A1 + A2) :- aexp(A1), aexp(A2).

bexp(true). bexp(false).
bexp(and(BE1,BE2)) :- bexp(BE1), bexp(BE2).
bexp(A1 =< A2) :- aexp(A1), aexp(A2).

stmt(skip).
stmt(X = AE) :- atom(X), aexp(AE).
stmt(St1;St2) :- stmt(St1), stmt(St2).
stmt(if(BE,St1,St2)) :-  bexp(BE), stmt(St1), stmt(St2).

program(St,AE) :- stmt(St), aexp(AE).
```

Exemplu

Exemplu

```
test0 :- program( {x = 10 ; sum = 0;
                   while(0 =< x,
                          {sum = sum + x; x = x-1}
                         )}
                   , sum).
```

?- test0.
true.

Ce înseamnă semantica formală?

Ce defineşte un limbaj de programare?

- Sintaxa – Simboluri de operaţie, cuvinte cheie, descriere (formală) a programelor/expresiilor bine formate
- Practic – Un limbaj e definit de modul cum poate fi folosit
  - Manual de utilizare şi exemple de bune practici
  - Implementare (compilator/interpretor)
  - Instrumente ajutătoare (analizor de sintaxă, depanator)
- Semantica – Ce înseamnă/care e comportamentul unei instrucţiuni?

La ce foloseşte semantica?

- Să înţelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanţii trebuie să ofer
- Ca instrument în proiectarea unui nou limbaj/a unei extensii
  - Înţelegerea componentelor şi a relaţiilor dintre ele
  - Exprimarea (şi motivarea) deciziilor de proiectare
  - Demonstrarea unor proprietăţi generice ale limbajului
- Ca bază pentru demonstrarea corectitudinii programelor

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Axiomatică – descrierea folosind logică a efectelor unei instrucţiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii
- Denotaţională – asocierea unui obiect matematic (denotaţie)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- Operaţională – asocierea unei demonstraţii pentru execuţie
  - $\langle cod, \sigma \rangle \to \langle cod', \sigma' \rangle$
  - modelează un program prin execuţia pe o maşină abstractă
  - utilă pentru implementarea de compilatoare şi interpretoare
- Statică – asocierea unui sistem de tipuri care exclude programe eronate

Semantica small-step

- Semantica operatională descrie cum se execută un program pe o masină abstractă (ideală).
- Semantica operatională **small-step**
  - semantica structurală, a pasilor mici
  - descrie cum o executie a programului avansează în functie de reduceri succesive.

$$\langle cod, \sigma \rangle \to \langle cod', \sigma' \rangle$$

- Semantica operatională **big-step**
  - semantică naturală, într-un pas mare

Starea execuției

- Starea execuției unui program IMP la un moment dat este dată de valorile deținute în acel moment de variabilele declarate în program.

- Formal, starea executiei unui program IMP la un moment dat este o funcție parțială (cu domeniu finit):

$$\sigma : Var \rightharpoonup Int$$

- Notatii:
  - Descrierea funcției prin enumerare: $\sigma = n \mapsto 10, sum \mapsto 0$
  - Funcția vidă $\perp$, nedefinită pentru nicio variabilă
  - Obținerea valorii unei variabile: $\sigma(x)$
  - Suprascrierea valorii unei variabile:

$$\sigma_{x \leftarrow v}(y) = \begin{cases} \sigma(y), \text{ dacă } y \neq x \\ v, \text{ dacă } y = x \end{cases}$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Defineste cel mai mic pas de execuție ca o relație „de tranziție” între configurații:

$$\langle cod\ ,\ \sigma\rangle \to \langle cod\ ,\ \sigma'\rangle$$

- Execuția se obține ca o succesiune de astfel de tranziții:

$$\begin{array}{lcl} \langle x = 0\ ;\ x = x + 1\ ;\ ,\ \bot\rangle & \to & \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle \\ & \to & \langle x = 0 + 1\ ;\ ,\ x \mapsto 0\rangle \\ & \to & \langle x = 1\ ;\ ,\ x \mapsto 0\rangle \\ & \to & \langle \{\}\ ,\ x \mapsto 1\rangle \end{array}$$

- Cum definim această relatie? Prin inductie după elementele din sintaxă.

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\ (0 <= 5 + 7 * x\ ,\ r = 1\ ,\ r = 0)$$

Reguli structurale

- Folosesc la identificarea următorului redex
- Definite recursiv pe structura termenilor

$$\frac{\langle b\ ,\ \sigma\rangle \to \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \to \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

Axiome

- Realizează pasul computațional

$$\langle \texttt{if}\ (\texttt{true}, bl_1, bl_2)\ ,\ \sigma\rangle \to \langle bl_1\ ,\ \sigma\rangle$$

Semantica expresiilor aritmetice

- Semantica unui întreg este o valoare
  - nu poate fi redex, deci nu avem regulă
- Semantica unei variabile

(Id) $\langle x\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle$ *dacă* $i = \sigma(x)$

- Semantica adunării a două expresii aritmetice

(Add) $\langle i_1 + i_2\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle$ *dacă* $i = i_1 + i_2$

$$\frac{\langle a_1\ ,\ \sigma\rangle \to \langle a_1'\ ,\ \sigma\rangle}{\langle a_1 + a_2\ ,\ \sigma\rangle \to \langle a_1' + a_2\ ,\ \sigma\rangle} \qquad \frac{\langle a_2\ ,\ \sigma\rangle \to \langle a_2'\ ,\ \sigma\rangle}{\langle a_1 + a_2\ ,\ \sigma\rangle \to \langle a_1 + a_2'\ ,\ \sigma\rangle}$$

Observatie: ordinea de evaluare a argumentelor este nespecificată.

Semantica expresiilor booleene

- Semantica operatorului de comparatie

(LEQ-FALSE) $\langle i_1 =< i_2 \,,\, \sigma\rangle \rightarrow \langle \texttt{false} \,,\, \sigma\rangle$ *dacă* $i_1 > i_2$

(LEQ-TRUE) $\langle i_1 =< i_2 \,,\, \sigma\rangle \rightarrow \langle \texttt{true} \,,\, \sigma\rangle$ *dacă* $i_1 \leq i_2$

$$\frac{\langle a_1 \,,\, \sigma\rangle \rightarrow \langle a_1' \,,\, \sigma\rangle}{\langle a_1 =< a_2 \,,\, \sigma\rangle \rightarrow \langle a_1' =< a_2 \,,\, \sigma\rangle} \qquad \frac{\langle a_2 \,,\, \sigma\rangle \rightarrow \langle a_2' \,,\, \sigma\rangle}{\langle a_1 =< a_2 \,,\, \sigma\rangle \rightarrow \langle a_1 =< a_2' \,,\, \sigma\rangle}$$

- Semantica negatiei

(!-FALSE) $\langle \texttt{not(true)} \,,\, \sigma\rangle \rightarrow \langle \texttt{false} \,,\, \sigma\rangle$

(!-TRUE) $\langle \texttt{not(false)} \,,\, \sigma\rangle \rightarrow \langle \texttt{true} \,,\, \sigma\rangle$

$$\frac{\langle a \,,\, \sigma\rangle \rightarrow \langle a' \,,\, \sigma\rangle}{\langle \texttt{not}\ (a) \,,\, \sigma\rangle \rightarrow \langle \texttt{not}\ (a') \,,\, \sigma\rangle}$$

Semantica expresiilor booleene

- Semantica operatorului de comparatie
- Semantica negatiei
- Semantica si-ului

(AND-FALSE) $\langle \text{and}\,(\texttt{false}, b_2)\ ,\ \sigma\rangle \to \langle \texttt{false}\ ,\ \sigma\rangle$

(AND-TRUE) $\langle \text{and}\,(\texttt{true}, b_2)\ ,\ \sigma\rangle \to \langle b_2\ ,\ \sigma\rangle$

$$\frac{\langle b_1\ ,\ \sigma\rangle \to \langle b_1'\ ,\ \sigma\rangle}{\langle \text{and}\,(b_1\ ,\ b_2)\ ,\ \sigma\rangle \to \langle \text{and}\,(b_1'\ ,\ b_2)\ ,\ \sigma\rangle}$$

Semantica compunerii si a blocurilor

- Semantica blocurilor

  (Block) $\langle \{\ s\ \}\ ,\ \sigma\rangle \rightarrow \langle s\ ,\ \sigma\rangle$

- Semantica compunerii secventiale

  (Next-stmt) $\langle \mathtt{skip;}\ s_2\ ,\ \sigma\rangle \rightarrow \langle s_2\ ,\ \sigma\rangle$

  $$\frac{\langle s_1\ ,\ \sigma\rangle \rightarrow \langle s_1'\ ,\ \sigma'\rangle}{\langle s_1\ \mathtt{;}\ \ s_2\ ,\ \sigma\rangle \rightarrow\ \rightarrow \langle s_1'\ \mathtt{;}\ \ s_2\ ,\ \sigma'\rangle}$$

- Semantica atribuirii

  (Asgn) $\langle x = i\ ,\ \sigma\rangle \rightarrow \langle \mathtt{skip}\ ,\ \sigma'\rangle$ *dacă* $\sigma' = \sigma_{x \leftarrow i}$

  $$\frac{\langle a\ ,\ \sigma\rangle \rightarrow \langle a'\ ,\ \sigma\rangle}{\langle x = a\ ,\ \sigma\rangle \rightarrow \langle x = a'\ ,\ \sigma\rangle}$$

Semantica lui `if`

- Semantica lui `if`

(If-true) $\langle \texttt{if}\,(\texttt{true}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_1\ ,\ \sigma\rangle$

(If-false) $\langle \texttt{if}\,(\texttt{false}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_2\ ,\ \sigma\rangle$

$$\frac{\langle b\ ,\ \sigma\rangle \rightarrow \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\,(b, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\,(b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

- Semantica lui `while`

(While) $\langle \texttt{while}\,(b, bl)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\,(b, bl\ \texttt{;while}\,(b, bl), \texttt{skip})\ ,\ \sigma\rangle$

- Semantica programelor

(Pgm) $$\frac{\langle a_1\ ,\ \sigma_1\rangle \rightarrow \langle a_2\ ,\ \sigma_2\rangle}{\langle (\texttt{skip}, a_1)\ ,\ \sigma_1\rangle \rightarrow \langle (\texttt{skip}, a_2)\ ,\ \sigma_2\rangle}$$

$$\frac{\langle s_1\ ,\ \sigma_1\rangle \rightarrow \langle s_2\ ,\ \sigma_2\rangle}{\langle (s_1, a)\ ,\ \sigma_1\rangle \rightarrow \langle (s_2, a)\ ,\ \sigma_2\rangle}$$

Semantica small-step a lui IMP

Execuție pas cu pas

$\langle i = 3 \;\texttt{;while}\; (0 \mathrel{<=} i \;\texttt{,}\{\; i = i + -4 \;\}) \;,\; \bot\rangle \xrightarrow{\textsc{Assgn}}$

$\langle \texttt{while}\; (0 \mathrel{<=} i \;\texttt{,}\{\; i = i + -4 \;\}) \;,\; i \mapsto 3\rangle \xrightarrow{\textsc{While}}$

$\langle \texttt{if}\; (0 \mathrel{<=} i \;,\; i = i + -4 \;\texttt{;while}\; (0 \mathrel{<=} i, \{\; i = i + -4 \;\}) \;\texttt{,skip}) \;,\; i \mapsto 3\rangle \xrightarrow{\textsc{Id}}$

$\langle \texttt{if}\; (0 \mathrel{<=} 3 \;,\; i = i + -4 \;\texttt{;while}\; (0 \mathrel{<=} i, \{\; i = i + -4 \;\}) \;\texttt{,skip}) \;,\; i \mapsto 3\rangle \xrightarrow{\textsc{Leq-true}}$

$\langle \texttt{if}\; (\texttt{true},\; i = i + -4 \;\texttt{;while}\; (0 \mathrel{<=} i, \{\; i = i + -4 \;\}) \;\texttt{,skip}) \;,\; i \mapsto 3\rangle \xrightarrow{\textsc{If-true}}$

$\langle i = i + -4 \;\texttt{;while}\; (0 \mathrel{<=} i, \{\; i = i + -4 \;\}) \;,\; i \mapsto 3\rangle \xrightarrow{\textsc{Id}}$

$\ldots$

Prolog: semantica small-step pentru IMP

- Defineste cel mai mic pas de executie ca o relatie de tranzitie între configuratii:

  $\langle cod\ ,\ \sigma \rangle \rightarrow \langle cod'\ ,\ \sigma' \rangle$ `smallstep(Cod,S1,Cod',S2)`
- Executia se obtine ca o succesiune de astfel de tranzitii.
- Starea executiei unui program IMP la un moment dat este o functie partială: $\sigma = n \mapsto 10, sum \mapsto 0$, etc.

Reprezentarea stărilor în Prolog

```
get(S,X,I) :- member(vi(X,I),S).
get(_,_,0).
set(S,X,I,[vi(X,I)|S1]) :- del(S,X,S1).

del([vi(X,_)|S],X,S).
del([H|S],X,[H|S1]) :- del(S,X,S1) .
del([],_,[]).
```

Semantica expresiilor aritmetice

- Semantica unei variabile

  $\langle x \, , \, \sigma \rangle \to \langle i \, , \, \sigma \rangle \quad$ *dacă* $i = \sigma(x)$

Prolog

```
smallstepA(X,S,I,S) :-
  atom(X),
  get(S,X,I).
```

Semantica expresiilor aritmetice

- Semantica adunării a două expresii aritmetice

$$\langle i_1 + i_2 \, , \, \sigma\rangle \rightarrow \langle i \, , \, \sigma\rangle \quad \textit{dacă } i = i_1 + i_2$$

$$\frac{\langle a_1 \, , \, \sigma\rangle \rightarrow \langle a_1' \, , \, \sigma\rangle}{\langle a_1 + a_2 \, , \, \sigma\rangle \rightarrow \langle a_1' + a_2 \, , \, \sigma\rangle} \qquad \frac{\langle a_2 \, , \, \sigma\rangle \rightarrow \langle a_2' \, , \, \sigma\rangle}{\langle a_1 + a_2 \, , \, \sigma\rangle \rightarrow \langle a_1 + a_2' \, , \, \sigma\rangle}$$

Prolog

```
smallstepA(I1 + I2,S,I,S):- integer(I1),integer(I2),
                            I is I1 + I2.

smallstepA(I + AE1,S,I + AE2,S):- integer(I),
                                  smallstepA(AE1,S,AE2,S).

smallstepA(AE1 + AE,S,AE2 + AE,S):-  ...
```

Semantica expresiilor aritmetice

Exemplu

```
?- smallstepA(a + b, [vi(a,1),vi(b,2)],AE, S).
AE = 1+b,
S = [vi(a, 1), vi(b, 2)] .

?- smallstepA(1 + b, [vi(a,1),vi(b,2)],AE, S).
AE = 1+2,
S = [vi(a, 1), vi(b, 2)] .

?- smallstepA(1 + 2, [vi(a,1),vi(b,2)],AE, S).
AE = 3,
S = [vi(a, 1), vi(b, 2)]
```

- Semantica * si - se definesc similar.

Semantica expresiilor booleene

- Semantica operatorului de comparatie

$\langle i_1 =< i_2 \,,\, \sigma \rangle \rightarrow \langle \texttt{false} \,,\, \sigma \rangle \quad \textit{dacă } i_1 > i_2$

$\langle i_1 =< i_2 \,,\, \sigma \rangle \rightarrow \langle \texttt{true} \,,\, \sigma \rangle \quad \textit{dacă } i_1 \leq i_2$

$$\frac{\langle a_1 \,,\, \sigma \rangle \rightarrow \langle a_1' \,,\, \sigma \rangle}{\langle a_1 =< a_2 \,,\, \sigma \rangle \rightarrow \langle a_1' =< a_2 \,,\, \sigma \rangle} \qquad \frac{\langle a_2 \,,\, \sigma \rangle \rightarrow \langle a_2' \,,\, \sigma \rangle}{\langle a_1 =< a_2 \,,\, \sigma \rangle \rightarrow \langle a_1 =< a_2' \,,\, \sigma \rangle}$$

Prolog

```
smallstepB(I1 =< I2,S,true,S):- integer(I1),integer(I2),
                                 (I1 =< I2).
smallstepB(I1 =< I2,S,false,S):- integer(I1),integer(I2),
                                  (I1 > I2).
smallstepB(I =< AE1,S,I =< AE2,S):- ...
smallstepB(AE1 =< AE,S,AE2 =< AE,S):- ...
```

Semantica expresiilor Booleene

- Semantica negatiei

$\langle \texttt{not(true)}\ ,\ \sigma\rangle \to \langle \texttt{false}\ ,\ \sigma\rangle$

$\langle \texttt{not(false)}\ ,\ \sigma\rangle \to \langle \texttt{true}\ ,\ \sigma\rangle$

$$\frac{\langle a\ ,\ \sigma\rangle \to \langle a'\ ,\ \sigma\rangle}{\langle \texttt{not}\ (a)\ ,\ \sigma\rangle \to \langle \texttt{not}\ (a')\ ,\ \sigma\rangle}$$

Prolog

```
smallstepB(not(true),S,false,S) .
smallstepB(not(false),S,true,S) .

smallstepB(not(BE1),S,not(BE2),S) :- ...
```

Semantica compunerii si a blocurilor

- Semantica blocurilor
  $\langle \{\ s\ \}\ ,\ \sigma\rangle \rightarrow\ \rightarrow\langle s\ ,\ \sigma\rangle$
- Semantica compunerii secventiale
  $\langle \{\}\ \ s_2\ ,\ \sigma\rangle \rightarrow \langle s_2\ ,\ \sigma\rangle \qquad \dfrac{\langle s_1\ ,\ \sigma\rangle \rightarrow \langle s_1'\ ,\ \sigma'\rangle}{\langle s_1\ s_2\ ,\ \sigma\rangle \rightarrow \langle s_1'\ s_2\ ,\ \sigma'\rangle}$

Prolog

```
smallstepS({E},S,E,S).

smallstepS((skip;St2),S,St2,S).
smallstepS((St1;St),S1,(St2;St),S2) :- ...
```

Semantica atribuirii

- Semantica atribuirii

  $\langle x = i \;,\; \sigma \rangle \rightarrow \langle \{\} \;,\; \sigma' \rangle \quad dacă\; \sigma' = \sigma[i/x]$

  $$\frac{\langle a \;,\; \sigma \rangle \rightarrow \langle a' \;,\; \sigma \rangle}{\langle x = a \;,\; \sigma \rangle \rightarrow \langle x = a' \;;\; ,\; \sigma \rangle}$$

Prolog

```
smallstepS(X = AE,S,skip,S1) :- integer(AE),set(S,X,AE,S1).

smallstepS(X = AE1,S,X = AE2,S) :- ...
```

Semantica lui `if`

- Semantica lui `if`

$\langle \texttt{if}\ (\texttt{true}, bl_1, bl_2)\ ,\ \sigma \rangle \rightarrow \langle bl_1\ ,\ \sigma \rangle$

$\langle \texttt{if}\ (\texttt{false}, bl_1, bl_2)\ ,\ \sigma \rangle \rightarrow \langle bl_2\ ,\ \sigma \rangle$

$$\frac{\langle b\ ,\ \sigma \rangle \rightarrow \langle b'\ ,\ \sigma \rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma \rangle \rightarrow \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma \rangle}$$

Prolog

```
smallstepS(if(true,St1,_),S,St1,S).
smallstepS(if(false,_,St2),S,St2,S).

smallstepS(if(BE1,St1,St2),S,if(BE2,St1,St2),S) :- ...
```

Semantica lui `while`

- Semantica lui `while`

  $\langle \texttt{while}\ (b, bl)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b, bl\ ;\texttt{while}\ (b, bl), \texttt{skip})\ ,\ \sigma\rangle$

Prolog

```
smallstepS(while(BE,St),S,if(BE,(St;while(BE,St)),skip),S).
```

- Semantica programelor

$$\frac{\langle a_1 \ , \ \sigma_1 \rangle \to \langle a_2 \ , \ \sigma_2 \rangle}{\langle (\texttt{skip}, a_1) \ , \ \sigma_1 \rangle \to \langle (\texttt{skip}, a_2) \ , \ \sigma_2 \rangle}$$

$$\frac{\langle s_1 \ , \ \sigma_1 \rangle \to \langle s_2 \ , \ \sigma_2 \rangle}{\langle (s_1, a) \ , \ \sigma_1 \rangle \to \langle (s_2, a) \ , \ \sigma_2 \rangle}$$

Prolog

```
smallstepP(skip,AE1,S1,skip,AE2,S2) :-
                                smallstepA(AE1,S1,AE2,S2) .
smallstepP(St1,AE,S1,St2,AE,S2) :-
                                smallstepS(St1,S1,St2,S2).
```

Executia programelor

Prolog

```
run(skip,I,_,I):- integer(I).
run(St1,AE1,S1,I) :- smallstepP(St1,AE1,S1,St2,AE2,S2),
                     run(St2,AE2,S2,I).

run_program(Name) :- defpg(Name,{P},E), run(P,E, [],I),
                     write(I).
```

Exemplu

```
defpg(pg2, {x = 10 ; sum = 0; while(0 =< x, {
                                  sum = sum + x;
                                  x = x - 1})},sum)
```

?- run_program(pg2).
55
true

Executia programelor: `trace`

Putem defini o functie care ne permite să urmărim executia unui program în implementarea noastră?

Prolog

```
mytrace(skip,I,_) :- integer(I).
mytrace(St1,AE1,S1) :-  smallstepP(St1,AE1,S1,St2,AE2,S2),
                        write(St2),nl,
                        write(AE2),nl,
                        write(S2),nl,
                        mytrace(St2,AE2,S2).

trace_program(Name) :- defpg(Name,{P},E),
                       mytrace(P,E,[]).
```

Executia programelor: `trace_program`

Exemplu

```
?- trace_program(pg2).
. . .
[vi(x,-1),vi(sum,55)]
if(0=<x,(sum=sum+x;x=x-1;while(0=<x,sum=sum+x;x=x-1)),skip)
sum
[vi(x,-1),vi(sum,55)]
if(0=< -1,(sum=sum+x;x=x-1;while(0=<x,sum=sum+x;x=x-1)),skip)
sum
[vi(x,-1),vi(sum,55)]
if(false,(sum=sum+x;x=x-1;while(0=<x,sum=sum+x;x=x-1)),skip)
sum
[vi(x,-1),vi(sum,55)]
skip
sum
[vi(x,-1),vi(sum,55)]
skip
55
[vi(x,-1),vi(sum,55)]
true .
```

Semantica programelor Prolog

Puncte fixe. Teorema Knaster-Tarski

Mulţimi parţial ordonate

- O mulţime parţial ordonată (mpo) este o pereche $(M, \leq)$ unde $\leq \subseteq M \times M$ este o relaţie de ordine.

  O submulţime de perechi $R \subseteq M \times M$ este relaţie
  - reflexivă dacă $(x, x) \in R$ oricare $x \in M$,
  - antisimetrică dacă $(x, y) \in R$ şi $(y, x) \in R$ implică $x = y$,
  - tranzitivă dacă $(x, y) \in R$ şi $(y, z) \in R$ implică $(x, z) \in R$.

  O relaţie de ordine este o relaţie reflexivă, antisimetrică şi tranzitivă.

- O mpo $(L, \leq)$ se numeşte lanţ dacă este total ordonată, adică $x \leq y$ sau $y \leq x$ pentru orice $x,y \in L$. Vom considera lanţuri numărabile:

$$x_1 \leq x_2 \leq x_3 \leq \ldots$$

Mulţimi parţial ordonate complete

O mpo $(C, \leq)$ este completă (cpo) dacă:

- C are prim element $\perp$ ($\perp \leq x$ oricare $x \in C$),
- $\bigvee_n x_n$ există în C pentru orice lanţ $x_1 \leq x_2 \leq x_3 \leq \ldots$

Exemplu

Fie X o mulţime şi $\mathcal{P}(X)$ mulţimea submulţimilor lui X.

$(\mathcal{P}(X), \subseteq)$ este o cpo:

- $\subseteq$ este o relaţie de ordine
- $\emptyset$ este prim element ($\emptyset \subseteq Q$ pentru orice $Q \in \mathcal{P}(X)$)
- pentru orice şir (numărabil) de submulţimi ale lui X $Q_1 \subseteq Q_2 \subseteq \ldots$ evident $\bigcup_n Q_n \in \mathcal{P}(X)$

Funcţie monotonă

- Fie $(A, \leq_A)$ şi $(B, \leq_B)$ mulţimi parţial ordonate.
  O funcţie $f : A \to B$ este monotonă (crescătoare)

  dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcţii $f_i : \mathcal{P}(\{1,2,3\}) \to \mathcal{P}(\{1,2,3\})$ cu $i \in \{1,2,3\}$

- $f_1(Y) = Y \cup \{1\}$ este monotonă.
- $f_2(Y) = \begin{cases} \{1\} & \text{dacă } 1 \in Y \\ \emptyset & \text{altfel} \end{cases}$ este monotonă.
- $f_3(Y) = \begin{cases} \emptyset & \text{dacă } 1 \in Y \\ \{1\} & \text{altfel} \end{cases}$ nu este monotonă.
  De exemplu, $\emptyset \subseteq \{1\}$, dar $f_3(\emptyset) = \{1\}$, $f_3(\{1\}) = \emptyset$ şi $f_3(\emptyset) \not\subseteq f_3(\{1\})$.

Funcţie continuă

- Fie $(A, \leq_A)$ şi $(B, \leq_B)$ mulţimi parţial ordonate complete.
  O funcţie $f : A \to B$ este continuă dacă
  $$f(\bigvee_n a_n) = \bigvee_n f(a_n) \text{ pentru orice lanţ } \{a_n\}_n \text{ din } A.$$

- Observăm că orice funcţie continuă este crescătoare.

Exemplu

Fie următoarele funcţii $f_i : \mathcal{P}(\{1,2,3\}) \to \mathcal{P}(\{1,2,3\})$ cu $i \in \{1,2,3\}$

- $f_1(Y) = Y \cup \{1\}$ este continuă.

- $f_3(Y) = \begin{cases} \emptyset & \text{dacă } 1 \in Y \\ \{1\} & \text{altfel} \end{cases}$ nu este continuă.
  De exemplu, consideram lantul $\emptyset \subseteq \{1\}$.
  Avem $\emptyset \cup \{1\} = \{1\}$ şi $f_3(\{1\}) = \emptyset$.
  Dar $f_3(\emptyset) = \{1\}$, $f_3(\{1\}) = \emptyset$ şi $f_3(\emptyset) \cup f_3(\{1\}) = \{1\}$.

Teorema de punct fix

- Un element $a \in C$ este punct fix al unei funcţii $f : C \to C$ dacă $f(a) = a$.

Teorema Knaster-Tarski pentru CPO

Fie $(C, \leq)$ o mulţime parţial ordonată completă şi $\mathbf{F} : C \to C$ o funcţie continuă. Atunci

$$a = \bigvee_n \mathbf{F}^n(\perp)$$

este cel mai mic punct fix al funcţiei **F**.

- Observăm că în ipotezele ultimei teoreme secvenţa

$$\mathbf{F}^0(\perp) = \perp \leq \mathbf{F}(\perp) \leq \mathbf{F}^2(\perp) \leq \cdots \leq \mathbf{F}^n(\perp) \leq \cdots$$

este un lanţ, deci $\bigvee_n \mathbf{F}^n(\perp)$ există.

Teorema Knaster-Tarski pentru CPO

Demonstraţie

Fie $(C, \leq)$ o mulţime parţial ordonată completă şi $\mathbf{F} : C \to C$ o funcţie continuă.

- Arătăm că $a = \bigvee_n \mathbf{F}^n(\bot)$ este punct fix, i.e. $\mathbf{F}(a) = a$

$$
\begin{aligned}
\mathbf{F}(a) &= \mathbf{F}(\textstyle\bigvee_n \mathbf{F}^n(\bot)) \\
&= \textstyle\bigvee_n \mathbf{F}(\mathbf{F}^n(\bot)) \text{ din continuitate} \\
&= \textstyle\bigvee_n \mathbf{F}^{n+1}(\bot) \\
&= \textstyle\bigvee_n \mathbf{F}^n(\bot) = a
\end{aligned}
$$

Teorema Knaster-Tarski pentru CPO

Demonstraţie (cont.)

- Arătăm că a este cel mai mic punct fix.

Fie b un alt punct fix, i.e. $\mathbf{F}(b) = b$.

Demonstrăm prin inducţie după $n \geq 1$ că $\mathbf{F}^n(\bot) \leq b$.

Pentru $n = 0$, $\mathbf{F}^0(\bot) = \bot \leq b$ deoarece $\bot$ este prim element.

Dacă $\mathbf{F}^n(\bot) \leq b$, atunci $\mathbf{F}^{n+1}(\bot) \leq \mathbf{F}(b)$, deoarece $\mathbf{F}$ este crescătoare.
Deoarece $\mathbf{F}(b) = b$ rezultă $\mathbf{F}^{n+1}(\bot) \leq b$.

Ştim $\mathbf{F}^n(\bot) \leq b$ oricare $n \geq 1$, deci $a = \bigvee_n \mathbf{F}^n(\bot) \leq b$.

Am arătat că a este cel mai mic punct fix al funcţiei $\mathbf{F}$. $\square$

Programe Prolog propoziţionale

Fie KB o multime de clauze definite propoziţionale şi fie At mulţimea variabilelor propoziţionale (atomilor) $p_1, p_2, \ldots$ care apar în KB.

Fie $AtFact = \{p_i \mid p_i \in KB\}$ mulţimea faptelor din KB.

Exemplu

$$
\begin{array}{rcl}
\texttt{oslo} & \rightarrow & \texttt{windy} \\
\texttt{oslo} & \rightarrow & \texttt{norway} \\
\texttt{norway} & \rightarrow & \texttt{cold} \\
\texttt{cold} \land \texttt{windy} & \rightarrow & \texttt{winterIsComing} \\
 & & \texttt{oslo}
\end{array}
$$

$At = \{oslo, windy, norway, cold, winterIsComing\}$

$AtFact = \{oslo\}$

Programelor Prolog propoziţionale

Fie KB o multime de clauze definite propoziţionale şi fie At mulţimea atomilor $p_1, p_2, \ldots$ care apar în KB.

Fie $AtFact = \{p_i \mid p_i \in KB\}$ mulţimea atomilor care apar în faptele din KB.

Definim funcţia $f_{KB} : \mathcal{P}(At) \to \mathcal{P}(At)$ prin

$$\begin{aligned} f_{KB}(Y) = Y \cup AtFact & \\ & \cup \{a \in At \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } KB, \\ & \qquad s_1 \in Y, \ldots, s_n \in Y\} \end{aligned}$$

- Funcţia f_{KB} este continuă.

Semantica programelor Prolog propoziţionale

Pentru funcţia continuă $f_{KB} : \mathcal{P}(At) \to \mathcal{P}(At)$

$$\boxed{\begin{aligned} f_{KB}(Y) = Y \cup AtFact \\ \cup \{a \in At \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } KB, \\ s_1 \in Y, \ldots, s_n \in Y\} \end{aligned}}$$

aplicând Teorema Knaster-Tarski pentru CPO, obţinem că

$$\bigcup_n f_{KB}^n(\emptyset)$$

este cel mai mic punct fix al lui f_{KB}.

Semantica programelor Prolog propoziţionale

- Analizaţi ce se întamplă când considerăm succesiv

$$\emptyset, \quad f_{KB}(\emptyset), \quad f_{KB}(f_{KB}(\emptyset)), \quad f_{KB}(f_{KB}(f_{KB}(\emptyset))), \ldots$$

  La fiecare aplicare a lui f_{KB}, rezultatul fie se măreşte, fie rămâne neschimbat.

- Să presupunem că în $\mathcal{S}$ avem k atomi. Atunci după $k+1$ aplicări ale lui f_{KB}, trebuie să existe un punct în şirul de mulţimi obţinute de unde o nouă aplicare a lui f_{KB} nu mai schimbă rezultatul (punct fix):

$$f_{KB}(X) = X$$

- Dacă aplicăm f_{KB} succesiv până găsim un X cu proprietatea $f_{KB}(X) = X$, atunci găsim cel mai mic punct fix al lui f_{KB}.

Cel mai mic punct fix

Exemplu

$$\begin{array}{rcl} cold & \to & wet \\ wet \land cold & \to & scotland \end{array}$$

$$\boxed{\begin{array}{l} f_{KB}(Y) = Y \cup AtFact \\ \cup\{a \in At \mid (s_1 \land \ldots \land s_n \to a) \text{ este în } KB, \\ s_1 \in Y, \ldots, s_n \in Y\} \end{array}}$$

Se observă că $f_{KB}(\emptyset) = \emptyset$, deci $\emptyset$ este cel mai mic punct fix.

De aici deducem că niciun atom nu este consecinţă logică a formulelor de mai sus.

Cel mai mic punct fix

Exemplu

$$\begin{array}{rcl} cold & & \\ cold & \rightarrow & wet \\ windy & \rightarrow & dry \\ wet \wedge cold & \rightarrow & scotland \end{array}$$

$$\boxed{\begin{array}{l} f_{KB}(Y) = Y \cup AtFact \\ \cup\{a \in At \mid (s_1 \wedge \ldots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \ldots, s_n \in Y\} \end{array}}$$

$$\begin{array}{rcl} f_{KB}(\emptyset) & = & \{\ cold\ \} \\ f_{KB}(\{\ cold\ \}) & = & \{\ cold, wet\ \} \\ f_{KB}(\{\ cold, wet\ \}) & = & \{\ cold, wet, scotland\ \} \\ f_{KB}(\{\ cold, wet, scotland\ \}) & = & \{\ cold, wet, scotland\ \} \end{array}$$

Deci cel mai mic punct fix este $\{\ cold, wet, scotland\ \}$.

Semantica programelor Prolog propoziţionale

Funcţia $f_{KB} : \mathcal{P}(At) \to \mathcal{P}(At)$ este definită prin

$$
\begin{aligned}
f_{KB}(Y) = {} & Y \cup AtFact \\
& \cup \{a \in At \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } KB, s_1 \in Y, \ldots, s_n \in Y\}
\end{aligned}
$$

unde At este mulţimea atomilor din $\mathcal{S}$ şi $Baza = \{p_i \mid p_i \in KB\}$ este mulţimea atomilor care apar în faptele din $\mathcal{S}$.

Teoremă

Fie FP_{KB} este cel mai mic punct fix al funcţiei f_{KB}. Atunci

$$q \in FP_{KB} \quad \text{dacă şi numai dacă} \quad KB \models q.$$

- Semantica programului KB este FP_{KB}, cel mai mic punct fix al funcţiei f_{KB}.
- FP_{KB} conţine toate consecinţele logice ale bazei de cunoştinţe KB.

Limbajul PROLOG / Logica Horn

- Logica Horn: un fragment al logicii de ordinul I în care singurele formule admise sunt clauze Horn
  - formule atomice: $P(t_1, \ldots, t_n)$
  - $Q_1 \wedge \ldots \wedge Q_n \rightarrow P$
    unde toate Q_i, P sunt formule atomice, $\top$ sau $\bot$
- Problema programării logice: reprezentăm cunoştinţele ca o mulţime de clauze definite KB şi suntem interesaţi să aflăm răspunsul la o întrebare de forma $Q_1 \wedge \ldots \wedge Q_n$, unde toate Q_i sunt formule atomice

$$KB \models Q_1 \wedge \ldots \wedge Q_n$$

  - Variabilele din KB sunt cuantificate universal.
  - Variabilele din $Q_1, \ldots, Q_n$ sunt cuantificate existenţial.
- Semantica de punct fix a unui program PROLOG este definită folosind modele Herbrand (caz particular de modele ale logicii de ordinul I).

Modele în logica de ordinul I

Logica de ordinul I - sintaxa

Limbaj de ordinul I $\mathcal{L}$

- unic determinat de $\tau = (\mathbf{R}, \mathbf{F}, \mathbf{C}, ari)$

Termenii lui $\mathcal{L}$, notaţi $Trm_{\mathcal{L}}$, sunt definiţi inductiv astfel:

- orice variabilă este un termen;
- orice simbol de constantă este un termen;
- dacă $f \in \mathbf{F}$, $ar(f) = n$ şi $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este termen.

Formulele atomice ale lui $\mathcal{L}$ sunt definite astfel:

- dacă $R \in \mathbf{R}$, $ar(R) = n$ şi $t_1, \ldots, t_n$ sunt termeni, atunci $R(t_1, \ldots, t_n)$ este formulă atomică.

Formulele lui $\mathcal{L}$ sunt definite astfel:

- orice formulă atomică este o formulă
- dacă φ este o formulă, atunci $\neg\varphi$ este o formulă
- dacă φ şi ψ sunt formule, atunci $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ sunt formule
- dacă φ este o formulă şi x este o variabilă, atunci $\forall x\, \varphi$, $\exists x\, \varphi$ sunt formule

Logica de ordinul I - semantică

O structură este de forma $\mathcal{A} = (A, \mathbf{F}^{\mathcal{A}}, \mathbf{R}^{\mathcal{A}}, \mathbf{C}^{\mathcal{A}})$, unde

- A este o mulţime nevidă
- $\mathbf{F}^{\mathcal{A}} = \{f^{\mathcal{A}} \mid f \in \mathbf{F}\}$ este o mulţime de operaţii pe A; dacă f are aritatea n, atunci $f^{\mathcal{A}} : A^n \to A$.
- $\mathbf{R}^{\mathcal{A}} = \{R^{\mathcal{A}} \mid R \in \mathbf{R}\}$ este o mulţime de relaţii pe A; dacă R are aritatea n, atunci $R^{\mathcal{A}} \subseteq A^n$.
- $\mathbf{C}^{\mathcal{A}} = \{c^{\mathcal{A}} \in A \mid c \in \mathbf{C}\}$.

O interpretare a variabilelor lui $\mathcal{L}$ în $\mathcal{A}$ ($\mathcal{A}$-interpretare) este o funcţie $I : V \to A$.

Inductiv, definim interpretarea termenului t în $\mathcal{A}$ sub I notat $t_I^{\mathcal{A}}$.

Inductiv, definim când o formulă este adevărată în $\mathcal{A}$ în interpretarea I notat $\mathcal{A}, I \models \varphi$. În acest caz spunem că $(\mathcal{A}, I)$ este model pentru φ.

O formulă φ este adevărată într-o structură $\mathcal{A}$, notat $\mathcal{A} \models \varphi$, dacă este adevărată în $\mathcal{A}$ sub orice interpretare. Spunem că $\mathcal{A}$ este model al lui φ.

O formulă φ este adevărată în logica de ordinul I, notat $\models \varphi$, dacă este adevărată în orice structură. O formulă φ este validă dacă $\models \varphi$.

O formulă φ este satisfiabilă dacă există o structură $\mathcal{A}$ şi o $\mathcal{A}$-interpretare I astfel încât $\mathcal{A}, I \models \varphi$.

Interpretare

Fie $\mathcal{L}$ un limbaj de ordinul I şi $\mathcal{A}$ o ($\mathcal{L}$-)structură.

Definiţie

O interpretare a variabilelor lui $\mathcal{L}$ în $\mathcal{A}$ este o funcţie

$$I : V \to A.$$

Definiţie

Inductiv, definim interpretarea termenului t în $\mathcal{A}$ sub I ($t_I^{\mathcal{A}}$) prin:

- dacă $t = x_i \in V$, atunci $t_I^{\mathcal{A}} := I(x_i)$
- dacă $t = c \in \mathbf{C}$, atunci $t_I^{\mathcal{A}} := c^{\mathcal{A}}$
- dacă $t = f(t_1, \ldots, t_n)$, atunci $t_I^{\mathcal{A}} := f^{\mathcal{A}}((t_1)_I^{\mathcal{A}}, \ldots, (t_n)_I^{\mathcal{A}})$

Interpretare

Definim inductiv faptul că o formulă este adevărată în $\mathcal{A}$ sub interpretarea I astfel:

- $\mathcal{A}, I \models P(t_1, \ldots, t_n)$ dacă $P^{\mathcal{A}}((t_1)_I^{\mathcal{A}}, \ldots, (t_n)_I^{\mathcal{A}})$
- $\mathcal{A}, I \models \neg\varphi$ dacă $\mathcal{A}, I \not\models \varphi$
- $\mathcal{A}, I \models \varphi \vee \psi$ dacă $\mathcal{A}, I \models \varphi$ sau $\mathcal{A}, I \models \psi$
- $\mathcal{A}, I \models \varphi \wedge \psi$ dacă $\mathcal{A}, I \models \varphi$ şi $\mathcal{A}, I \models \psi$
- $\mathcal{A}, I \models \varphi \rightarrow \psi$ dacă $\mathcal{A}, I \not\models \varphi$ sau $\mathcal{A}, I \models \psi$
- $\mathcal{A}, I \models \forall x\, \varphi$ dacă pentru orice $a \in A$ avem $\mathcal{A}, I_{x \leftarrow a} \models \varphi$
- $\mathcal{A}, I \models \exists x\, \varphi$ dacă există $a \in A$ astfel încât $\mathcal{A}, I_{x \leftarrow a} \models \varphi$

unde pentru orice $a \in A$, $I_{x \leftarrow a}(y) = \begin{cases} I(y) & \text{dacă } y \neq x \\ a & \text{dacă } y = x \end{cases}$

Model în logica de ordinul I

Exemplu

Fie limbajul $\mathcal{L}$ cu $\mathbf{F} = \{s\}$, $\mathbf{R} = \{P\}$, $\mathbf{C} = \{0\}$ cu $ari(s) = ari(P) = 1$.

Fie structura $\mathcal{N} = (\mathbb{N}, s^{\mathcal{N}}, P^{\mathcal{N}}, 0^{\mathcal{N}})$ unde $0^{\mathcal{N}} := 1$ şi

- $s^{\mathcal{N}} : \mathbb{N} \to \mathbb{N}$, $s^{\mathcal{N}}(n) := n^2$
- $P^{\mathcal{N}} \subset \mathbb{N}$, $P^{\mathcal{N}} = \{n \mid n \text{ este impar }\}$

Demonstraţi că $\mathcal{N} \models \forall x\,(P(x) \to P(s(x)))$.

Fie $I : V \to \mathbb{N}$ o interpretare. Observăm că
$\mathcal{N}, I \models P(x)$ dacă $P^{\mathcal{N}}(I(x))$, adică $\mathcal{N}, I \models P(x)$ dacă $I(x)$ este impar.

$\mathcal{N}, I \models \forall x\,(P(x) \to P(s(x)))$ dacă
$\mathcal{N}, I_{x \leftarrow n} \models P(x) \to P(s(x))$ oricare $n \in N$
$\mathcal{N}, I_{x \leftarrow n} \not\models P(x)$ sau $\mathcal{N}, I_{x \leftarrow n} \models P(s(x))$ oricare $n \in N$
$I_{x \leftarrow n}(x)$ nu este impar sau $I_{x \leftarrow n}(s(x))$ este impar oricare $n \in \mathbb{N}$
n este par sau n^2 este impar oricare $n \in \mathbb{N}$

ceea ce este întodeauna adevărat.

Modele Herbrand

Modele Herbrand

Fie $\mathcal{L}$ un limbaj de ordinul I.

- Presupunem că are cel puţin un simbol de constantă!
- Dacă nu are, adăugăm un simbol de constantă.

Universul Herbrand este mulţimea $T_{\mathcal{L}}$ a tututor termenilor lui $\mathcal{L}$ fără variabile.

Un model Herbrand este o structură $\mathcal{H} = (T_{\mathcal{L}}, \mathbf{F}^{\mathcal{H}}, \mathbf{P}^{\mathcal{H}}, \mathbf{C}^{\mathcal{H}})$, unde

- pentru orice simbol de constantă c, $c^{\mathcal{H}} = c$
- pentru orice simbol de funcţie f de aritate n,
  $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
- pentru orice simbol de relaţie R de aritate n, $R^{\mathcal{H}}(t_1, \ldots, t_n) \subseteq (T_{\mathcal{L}})^n$

Pentru a defini un model Herbrand concret trebuie
sa definim interpretarea relaţiilor.

Model Herbrand

Exemplu

Fie $\mathcal{L}$ un limbaj de ordinul I cu un simbol de funcţie f de aritate 1, un simbol de constantă a şi un simbol de relaţie R de aritate 2.

O structură Herbrand $\mathcal{H} = (T_{\mathcal{L}}, \mathbf{F}^{\mathcal{H}}, \mathbf{R}^{\mathcal{H}}, \mathbf{C}^{\mathcal{H}})$ unde

- $T_{\mathcal{L}} = \{a, f(a), f(f(a)), \ldots\}$
- $a^{\mathcal{H}} = a \in T_{\mathcal{L}}$
- $f_{\mathcal{H}}^{T}(t) = f(t)$
- $R^{\mathcal{H}} = \{(t, t) \mid t \in T_{\mathcal{L}}\} = \{(a, a), (f(a), f(a)), (f(f(a)), f(f(a))), \ldots\}$

$\mathcal{H} \models \forall x\, R(x, x)$.

Model Herbrand

Exemplu

Fie $\mathcal{L}$ un limbaj de ordinul I cu un simbol de funcţie f de aritate 1, un simbol de constantă a şi un simbol de relaţie R de aritate 2.

O structură Herbrand $\mathcal{H} = (T_{\mathcal{L}}, \mathbf{F}^{\mathcal{H}}, \mathbf{R}^{\mathcal{H}}, \mathbf{C}^{\mathcal{H}})$ unde

- $T_{\mathcal{L}} = \{a, f(a), f(f(a)), \ldots\}$
- $a^{\mathcal{H}} = a \in T_{\mathcal{L}}$
- $f_{\mathcal{H}}^{T}(t) = f(t)$
- $R^{\mathcal{H}} = \{(t, f(t)) \mid t \in T_{\mathcal{L}}\} =$
  $\{(a, f(a)), (f(a), f(f(a))), (f(f(a)), f(f(f(a)))), \ldots\}$

$\mathcal{H} \not\models \forall x\, R(x, x)$.

Cel mai mic model Herbrand

- Definim o ordine între modelele Herbrand:

  $\mathcal{H}_1 \leq \mathcal{H}_2$ dacă şi numai dacă

  pentru orice $R \in \mathbf{R}$ cu $ari(R) = n$ şi pentru orice termeni $t_1, \ldots, t_n$
  dacă $\mathcal{H}_1 \models R(t_1, \ldots, t_n)$, atunci $\mathcal{H}_2 \models R(t_1, \ldots, t_n)$

- Pentru KB, un program Prolog (o mulţime de clauze definite), definim intersecţia modelelor Herbrand ale programului

$$\mathcal{LH}_{KB} := \bigcap \{\mathcal{H} \mid \mathcal{H} \models KB\}$$

  Se observă că $\mathcal{LH}_{KB} \models KB$.

Semantica unui program logic definit KB este dată de cel mai mic model Herbrand al lui KB. Vom caracteriza cel mai mic model Herbrand $\mathcal{LH}_{KB}$ printr-o construcţie de punct fix.

Baza Herbrand

- O formulă fără variabile se numeşte închisă.
- Baza Herbrand $B_{\mathcal{L}}$ este mulţimea formulelor atomice închise.
- O instanţă închisă a unei clauze $Q_1 \wedge \ldots \wedge Q_n \to P$ este rezultatul obţinut prin înlocuirea variabilelor cu termeni fără variabile.

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un simbol de constantă 0, un simbol de funcţie unară s şi un simbol de relaţie unară par. Notăm $s(x)$ cu sx.
- $T_{\mathcal{L}} = \{0, s0, ss0, \ldots\}$
- Fie KB mulţimea clauzelor:

$$par(0) \qquad\qquad par(x) \to par(ssx)$$

- Instanţe de bază:
  - $par(0) \to par(ss0)$
  - $par(s0) \to par(sss0)$
- $B_{\mathcal{L}} = \{par(0), par(s0), par(ss0), par(sss0) \ldots\}$

Baza Herbrand

- O formulă fără variabile se numeşte închisă.
- Baza Herbrand $B_{\mathcal{L}}$ este mulţimea formulelor atomice închise.
- O instanţă închisă a unei clauze $Q_1 \wedge \ldots \wedge Q_n \to P$ este rezultatul obţinut prin înlocuirea variabilelor cu termeni fără variabile.
- Pentru o mulţime de clauze definite KB, dacă $P \in B_{\mathcal{L}}$ şi $X \subseteq B_{\mathcal{L}}$ spunem că

  $$oneStep_{KB}(P, X) \text{ este adevărat}$$

  dacă există $Q_1, \ldots, Q_n \in X$ astfel încât $Q_1 \wedge \ldots \wedge Q_n \to P$ este o instanţă de închisă a unei clauze din KB.
- Pentru o mulţime de clauze definite KB, definim

  $$f_{KB} : \mathcal{P}(B_{\mathcal{L}}) \to \mathcal{P}(B_{\mathcal{L}})$$

  $$f_{KB}(X) = \{P \in B_{\mathcal{L}} \mid oneStep_{KB}(P, X)\}$$

- f_{KB} este continuă (exerciţiu).

Baza Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un simbol de constantă 0, un simbol de funcţie unară s şi un simbol de relaţie unară par. Notăm $s(x)$ cu sx.
- $T_{\mathcal{L}} = \{0, s0, ss0, \ldots\}$
- Fie KB mulţimea clauzelor:

$$par(0) \qquad\qquad par(x) \to par(ssx)$$

- Instanţe de bază:
  - $par(0) \to par(ss0)$
  - $par(s0) \to par(sss0)$
- $B_{\mathcal{L}} = \{par(0), par(s0), par(ss0), par(sss0) \ldots\}$
- $oneStep_{KB}(P, \{\})$ selecteaza faptele din KB
  $f_{KB}(\{\}) = \{par(0)\}$
- $f_{KB}(\{par(0)\}) = \{par(0), par(ss0)\}$
- $f_{KB}(\{par(s(0))\}) = \{par(0), par(sss0)\}$
- $f_{KB}(\{par(ss0)\}) = \{par(0), par(ssss0)\}$

Cel mai mic model Herbrand

Fie KB un program logic definit.

- Din teorema Knaster-Tarski, f_{KB} are un cel mai mic punct fix FP_{KB}.
- FP_{KB} este reuniunea tuturor mulţimilor

$$f_{KB}(\{\}),\ f_{KB}(f_{KB}(\{\})),\ f_{KB}(f_{KB}(f_{KB}(\{\}))),\ldots$$

Caracterizarea $\mathcal{LH}_{KB}$ ca punct fix.

Pentru orice $R \in \mathbf{R}$ cu $ari(R) = n$ şi pentru orice $t_1, \ldots, t_n$ termeni, avem

$$(t_1, \ldots, t_n) \in R^{\mathcal{LH}_{KB}} \text{ ddacă } R(t_1, \ldots, t_n) \in FP_{KB}$$

Relaţiile care definesc cel mai mic model Herbrand al unui program Prolog sunt caracterizate folosind teorema de punct fix Knaster-Tarski.

Cel mai mic model Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un simbol de constantă 0, un simbol de funcţie unară s şi un simbol de relaţie unară par. Notăm $s(x)$ cu sx.
- $T_{\mathcal{L}} = \{0, s0, ss0, \ldots\}$
- Fie KB mulţimea clauzelor:

$$par(0) \qquad\qquad par(x) \to par(ssx)$$

- Instanţe de bază:
  - $par(0) \to par(ss0)$
  - $par(s0) \to par(sss0)$
- $B_{\mathcal{L}} = \{par(0), par(s0), par(ss0), par(sss0) \ldots\}$
- $FP_{KB} = \bigcup_n f^n_{KB}(\emptyset) = f_{KB}(\{\}) \cup f_{KB}(f_{KB}(\{\})) \cup f_{KB}(f_{KB}(f_{KB}(\{\}))) \cup \ldots$
  $= \{par(0), par(ss0), par(ssss0), \ldots\}$

Observăm ca relaţia $par^{\mathcal{LH}_{KB}}$ conţine numai reprezentările numerelor pare.

Cel mai mic model Herbrand

Fie KB un program logic definit.

Teoremă

Pentru orice formulă atomică Q,

$$KB \models Q \quad \text{dacă şi numai dacă} \quad \mathcal{LH}_{KB} \models Q.$$

- Semantica programului KB este $\mathcal{LH}_{KB}$, cel mai mic model Herbrand.
- Relaţiile modelului $\mathcal{LH}_{KB}$ sunt definite folosind teorema de punct fix Knaster-Tarski.

Cel mai mic model Herbrand

Fie KB un program logic definit.

Teoremă

Pentru orice formulă atomică Q,

$$KB \models Q \quad \text{dacă şi numai dacă} \quad \mathcal{LH}_{KB} \models Q.$$

Demonstraţie

Q este adevărată în $\mathcal{LH}_{KB}$
ddacă Q este adevărată în toate modelele Herbrand ale lui KB
ddacă $\neg Q$ este falsă în toate modelele Herbrand ale lui KB
ddacă $KB \cup \{\neg Q\}$ nu are niciun model Herbrand (1)
ddacă $KB \cup \{\neg Q\}$ nesatisfiabilă (2)
ddacă $KB \models Q$

(1) $\Leftrightarrow$ (2) rezultă din Teorema lui Herbrand