Curs recapitulativ

Cuprins

1. Logica propozițională (recap.)
   - Deducția naturală
   - Clauze propoziționale definite
2. Logica de ordinul I (recap.)
3. Algoritmul de unificare
4. Forme prenex și Skolem. Modele Herbrand
5. Formă clauzală. Rezoluție
   - Rezoluția în logica propozițională (recap.)
   - Rezoluția în logica de ordinul I
6. Logica Horn
7. Introducere în semantică (implementare în Prolog)

Logica propozițională (recap.)

Semantica logicii propoziționale

- Mulțimea valorilor de adevăr este $\{0, 1\}$ pe care considerăm următoarele operații:

x	$\neg x$
0	1
1	0

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

$$x \vee y := max\{x, y\} \qquad x \wedge y := min\{x, y\}$$

Semantica logicii propoziționale

- o funcție $e : Var \to \{0,1\}$ se numește evaluare (interpretare)
- pentru orice evaluare $e : Var \to \{0,1\}$ există o unică funcție $e^{+} : Form \to \{0,1\}$ care verifică următoarele proprietăți:
  - $e^{+}(v) = e(v)$
  - $e^{+}(\neg\varphi) = \neg e^{+}(\varphi)$
  - $e^{+}(\varphi \to \psi) = e^{+}(\varphi) \to e^{+}(\psi)$
  - $e^{+}(\varphi \wedge \psi) = e^{+}(\varphi) \wedge e^{+}(\psi)$
  - $e^{+}(\varphi \vee \psi) = e^{+}(\varphi) \vee e^{+}(\psi)$

  oricare ar fi $v \in Var$ și $\varphi, \psi \in Form$.

Semantica logicii propoziționale

Cum verificăm că o formulă este tautologie: $\models \varphi$?

- Fie $v_1, \ldots, v_n$ variabilele care apar în φ.
- Cele 2^n evaluări posibile $e_1, \ldots, e_{2^n}$ pot fi scrise într-un tabel:

v_1	v_2	$\ldots$	v_n	φ
$e_1(v_1)$	$e_1(v_2)$	$\ldots$	$e_1(v_n)$	$e_1^+(\varphi)$
$e_2(v_1)$	$e_2(v_2)$	$\ldots$	$e_2(v_n)$	$e_2^+(\varphi)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e_{2^n}(v_1)$	$e_{2^n}(v_2)$	$\ldots$	$e_{2^n}(v_n)$	$e_{2^n}^+(\varphi)$

  Fiecare evaluare corespunde unei linii din tabel!

- $\models \varphi$ dacă și numai dacă $e_1^+(\varphi) = \cdots = e_{2^n}^+(\varphi) = 1$

Sistemul Hilbert

- Oricare ar fi $\varphi, \psi, \chi \in$ *Form* următoarele formule sunt axiome:

  (A1) $\varphi \to (\psi \to \varphi)$
  (A2) $(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$
  (A3) $(\neg\psi \to \neg\varphi) \to (\varphi \to \psi)$.

- Regula de deducție este modus ponens: $\dfrac{\varphi, \ \varphi \to \psi}{\psi}\,\text{MP}$

- O demonstrație din ipotezele Γ (sau Γ-demonstrație) pentru φ este o secvență de formule $\gamma_1, \ldots, \gamma_n$ astfel încât $\gamma_n = \varphi$ și, pentru fiecare $i \in \{1, \ldots, n\}$, una din următoarele condiții este satisfăcută:
  - γ_i este axiomă,
  - $\gamma_i \in \Gamma$
  - γ_i se obține din formulele anterioare prin MP:
    există $j, k < i$ astfel încât $\gamma_j = \gamma_k \to \gamma_i$

- O formulă φ este Γ-teoremă dacă are o Γ-demonstrație.
  Notăm prin $\Gamma \vdash \varphi$ faptul că φ este o Γ-teoremă

Sistemul Hilbert

Teorema deducției TD (Herbrand, 1930)

Fie $\Gamma \cup \{\varphi\} \subseteq Form$. Atunci

$$\Gamma \vdash \varphi \rightarrow \psi \text{ dacă și numai dacă } \Gamma \cup \{\varphi\} \vdash \psi$$

Sistemul Hilbert

Exercițiu

Fie φ și ψ formule în logica propozițională. Să se arate sintactic că

$$\vdash \varphi \rightarrow (\neg\varphi \rightarrow \psi).$$

Soluție

Avem următoarea demonstrație:

(1)	$\{\varphi, \neg\varphi\}$	$\vdash \neg\varphi \rightarrow (\neg\psi \rightarrow \neg\varphi)$	(A1)
(2)	$\{\varphi, \neg\varphi\}$	$\vdash \neg\varphi$	(ipoteză)
(3)	$\{\varphi, \neg\varphi\}$	$\vdash \neg\psi \rightarrow \neg\varphi$	(1), (2), MP
(4)	$\{\varphi, \neg\varphi\}$	$\vdash (\neg\psi \rightarrow \neg\varphi) \rightarrow (\varphi \rightarrow \psi)$	(A3)
(5)	$\{\varphi, \neg\varphi\}$	$\vdash \varphi \rightarrow \psi$	(3), (4), MP
(6)	$\{\varphi, \neg\varphi\}$	$\vdash \varphi$	(ipoteză)
(7)	$\{\varphi, \neg\varphi\}$	$\vdash \psi$	(5), (6), MP
(8)	$\{\varphi\}$	$\vdash \neg\varphi \rightarrow \psi$	(7) Teorema Deducției
(9)		$\vdash \varphi \rightarrow (\neg\varphi \rightarrow \psi)$	(8) Teorema Deducției

Deducția naturală

Deducția naturală

- Numim secvent o expresie de forma
$$\varphi_1, \ldots, \varphi_n \vdash \psi$$
- Formulele $\varphi_1, \ldots, \varphi_n$ se numesc premise, iar ψ se numește concluzie.
- Un secvent este valid dacă există o demonstrație folosind regulile de deducție.
- O teoremă este o formulă ψ astfel încât $\vdash \psi$
(adică ψ poate fi demonstrată din mulțimea vidă de ipoteze).
- Pentru fiecare conector logic vom avea reguli de introducere și reguli de eliminare.

Regulile deducției naturale

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}\ (\wedge i)$$

$$\frac{\boxed{\begin{array}{c}\varphi\\ \vdots\\ \psi\end{array}}}{\varphi \to \psi}\ (\to i)$$

$$\frac{\varphi}{\varphi \vee \psi}\ (\vee i_1) \qquad \frac{\psi}{\varphi \vee \psi}\ (\vee i_2)$$

$$\frac{\varphi}{\neg\neg\varphi}\ (\neg\neg i) \qquad \frac{\boxed{\begin{array}{c}\varphi\\ \vdots\\ \bot\end{array}}}{\neg\varphi}\ (\neg i)$$

$$\frac{\varphi \wedge \psi}{\varphi}\ (\wedge e_1) \qquad \frac{\varphi \wedge \psi}{\psi}\ (\wedge e_2)$$

$$\frac{\varphi \quad \varphi \to \psi}{\psi}\ (\to e)$$

$$\frac{\varphi \vee \psi \quad \boxed{\begin{array}{c}\varphi\\ \vdots\\ \chi\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \vdots\\ \chi\end{array}}}{\chi}\ (\vee e)$$

$$\frac{\neg\neg\varphi}{\varphi}\ (\neg\neg e) \qquad \frac{\varphi \quad \neg\varphi}{\bot}\ (\neg e) \qquad \frac{\bot}{\varphi}\ (\bot e)$$

Deducție naturală

Exercițiu

Demonstrați că următorul secvent este valid:

$$p \land q \to r \vdash p \to (q \to r)$$

Soluție

1	$p \land q \to r$	*premiza*
2	p	*ipoteza*
3	q	*ipoteza*
4	$p \land q$	$(\land i)$,2,3
5	r	$(\to e)$,1,4
6	$q \to r$	$(\to i)$,3–5
7	$p \to (q \to r)$	$(\to i)$,2–6

Deducție naturală

Exercițiu

Demonstrați că următorul secvent este valid:

$$p \to q, p \to \neg q \vdash \neg p$$

Soluție

1	$p \to q$	*premiza*
2	$p \to \neg q$	*premiza*
3	p	*ipoteza*
4	q	$(\to e)$,1,3
5	$\neg q$	$(\to e)$,2,3
6	$\perp$	$(\neg e)$,4,5
7	$\neg p$	$(\neg i)$,3–6

Deducție naturală

Exercițiu

Echivalența logică este definită prin $\varphi \leftrightarrow \psi = (\varphi \to \psi) \wedge (\psi \to \varphi)$. Găsiți reguli de introducere și eliminare pentru $\leftrightarrow$.

Soluție

Observăm că $\leftrightarrow$ este o combinație între $\to$ și $\wedge$. Regulile pentru $\leftrightarrow$ se obțin combinând regulile pentru $\to$ și $\wedge$.

Introducerea $(\leftrightarrow i)$: pentru a introduce $\varphi \leftrightarrow \psi$ trebuie să introducem $\varphi \to \psi$ și $\psi \to \varphi$, apoi să introducem $\wedge$.

$$\dfrac{\boxed{\begin{array}{c}\varphi\\ \vdots\\ \psi\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \vdots\\ \varphi\end{array}}}{\varphi \leftrightarrow \psi}\ (\leftrightarrow i)$$

Soluție (cont.)

Eliminarea $(\leftrightarrow i)$: pentru a elimina $\varphi \leftrightarrow \psi$ trebuie să eliminăm $\wedge$ apoi să eliminăm o $\rightarrow$; vom avea două variante:

$$\frac{\varphi \leftrightarrow \psi \quad \psi}{\varphi}\ (\leftrightarrow e_1) \qquad \frac{\varphi \leftrightarrow \psi \quad \varphi}{\psi}\ (\leftrightarrow e_2)$$

Clauze propoziționale definite

Clauze propoziționale definite

O clauză definită este o formulă care poate avea una din formele:

1. q (clauză unitate)
2. $p_1 \land \ldots \land p_k \rightarrow q$

unde $q, p_1, \ldots, p_n$ sunt variabile propoziționale.

Sistem de deducție pentru clauze definite propoziționale

Pentru o mulțime S de clauze definite propoziționale, avem

- Axiome (premise): orice clauză din S
- Reguli de deducție:

$$\frac{P \quad P \rightarrow Q}{Q}\ (MP) \qquad\qquad \frac{P \quad Q}{P \land Q}\ (andI)$$

Mulțimi parțial ordonate

- O mulțime parțial ordonată (mpo) este o pereche $(M, \leq)$ unde $\leq \subseteq M \times M$ este o relație de ordine.
  - relație de ordine: reflexivă, antisimetrică, tranzitivă

- O mpo $(L, \leq)$ se numește lanț dacă este total ordonată, adică $x \leq y$ sau $y \leq x$ pentru orice $x, y \in L$. Vom considera lanțuri numărabile, i.e.

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

- 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 pentru orice lanț $x_1 \leq x_2 \leq x_3 \leq \ldots$

Funcții monotone și continue

- Fie $(A, \leq_A)$ și $(B, \leq_B)$ mulțimi parțial ordonate.
  O funcție $f : A \to B$ este monotonă (crescătoare)
  $$\text{dacă } a_1 \leq_A a_2 \text{ implică } f(a_1) \leq_B f(a_2) \text{ oricare } a_1,\ a_2 \in A.$$
- Fie $(A, \leq_A)$ și $(B, \leq_B)$ mulțimi parțial ordonate complete.
  O funcție $f : A \to B$ este continuă dacă
  $$f(\bigvee_n a_n) = \bigvee_n f(a_n) \text{ pentru orice lanț } \{a_n\}_n \text{ din } A.$$
- Observăm că orice funcție continuă este crescătoare.

Teorema de punct fix

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

Teorema Knaster-Tarski pentru CPO

Fie $(C, \leq)$ o mulțime parțial ordonată completă și $\mathbf{F} : C \to C$ o funcție continuă. Atunci

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

este cel mai mic punct fix al funcției $\mathbf{F}$.

Puncte fixe

Exercițiu

Care sunt punctele fixe ale următoarei funcții? Dar cel mai punct fix?

$$f_1 : \mathcal{P}(\{1,2,3\}) \to \mathcal{P}(\{1,2,3\}), \quad f_1(Y) = Y \cup \{1\}$$

Soluție

Se observă că punctele fixe ale lui f_1 sunt submulțimile Y ale lui $\{1,2,3\}$ care îl conțin pe 1 (dacă $1 \notin Y$, atunci $f_1(Y) = Y \cup \{1\}$ și evident $Y \neq Y \cup \{1\}$).

Deci punctele fixe ale lui f_1 sunt $\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}$.

Evident, cel mai mic punct fix este $\{1\}$.

Puncte fixe

Exercițiu

Care sunt punctele fixe ale următoarei funcții? Dar cel mai punct fix?

$$f_2 : \mathcal{P}(\{1,2,3\}) \to \mathcal{P}(\{1,2,3\}), \quad f_2(Y) = \begin{cases} \{1\} & \text{dacă } 1 \in Y \\ \emptyset & \text{altfel} \end{cases}$$

Soluție

Se observă că singurele puncte fixe ale lui f_2 sunt $\emptyset$ și $\{1\}$. Evident $\emptyset$ este cel mai mic punct fix.

Clauze definite și funcții monotone

Fie A mulțimea atomilor $p_1, p_2, \ldots$ care apar în S.

Fie $Baza = \{p_i \mid p_i \in S\}$ mulțimea atomilor care apar în clauzele unitate din S.

Definim funcția $f_S : \mathcal{P}(A) \to \mathcal{P}(A)$ prin

$$\boxed{\begin{aligned} f_S(Y) = Y \cup Baza & \\ & \cup \{a \in A \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } S, \\ & \qquad s_1 \in Y, \ldots, s_n \in Y\} \end{aligned}}$$

Clauze definite și funcții monotone

Exercițiu

Arătați că funcția f_S este monotonă.

Soluție

Fie $Y_1, Y_2 \subseteq A$ astfel încât $Y_1 \subseteq Y_2$. Trebuie să arătăm că $f_S(Y_1) \subseteq f_S(Y_2)$. Fie următoarele mulțimi:

$$\begin{aligned} Z_1 &= \{a \in A \mid (s_1 \wedge \ldots \wedge s_n \rightarrow a) \text{ este în } S, s_1 \in Y_1, \ldots, s_n \in Y_1\}, \\ Z_2 &= \{a \in A \mid (s_1 \wedge \ldots \wedge s_n \rightarrow a) \text{ este în } S, s_1 \in Y_2, \ldots, s_n \in Y_2\}. \end{aligned}$$

Deci $f_S(Y_1) = Y_1 \cup Baza \cup Z_1$ și $f_S(Y_2) = Y_2 \cup Baza \cup Z_2$.
Cum $Y_1 \subseteq Y_2$, rămâne să arătăm doar că $Z_1 \subseteq Z_2$.
Fie $a \in Z_1$. Atunci există $s_1 \wedge \ldots \wedge s_n \rightarrow a \in S$ și $s_1, \ldots, s_n \in Y_1$.
Deci $s_1, \ldots, s_n \in Y_2$, de unde rezultă că $a \in Z_2$.

Clauze definite și funcții monotone

Pentru funcția continuă $f_S : \mathcal{P}(A) \to \mathcal{P}(A)$

$$\boxed{\begin{aligned} f_S(Y) = Y \cup Baza \\ & \cup \{a \in A \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } S, \\ & \qquad s_1 \in Y, \ldots, s_n \in Y\} \end{aligned}}$$

aplicând Teorema Knaster-Tarski pentru CPO, obținem că

$$\bigcup_n f_S^n(\emptyset)$$

este cel mai mic punct fix al lui f_S.

Cel mai mic punct fix

Exercițiu

Calculați cel mai mic punct fix pentru functia f_{S_1} unde

$$S_1 = \{x_1 \land x_2 \to x_3,\ x_4 \land x_2 \to x_5,\ x_2,\ x_6,\ x_6 \to x_1\}$$

Soluție

Observăm că $A = \{x_1, x_2, \ldots, x_6\}$ și $Baza = \{x_2, x_6\}$.
Cum f_S este continuă, aplicăm Teorema Knaster-Tarski pentru a calcula cel mai mic punct fix:

$$f_{S_1}(\emptyset) = Baza = \{x_2, x_6\}$$
$$f_{S_1}(\{x_2, x_6\}) = \{x_2, x_6, x_1\}$$
$$f_{S_1}(\{x_2, x_6, x_1\}) = \{x_2, x_6, x_1, x_3\}$$
$$f_{S_1}(\{x_2, x_6, x_1, x_3\}) = \{x_2, x_6, x_1, x_3\}$$

În concluzie, cel mai mic punct fix căutam este $\{x_2, x_6, x_1, x_3\}$.

Programe logice și cel mai mic punct fix

Teoremă

Fie X este cel mai mic punct fix al funcției f_S. Atunci $q \in X$ ddacă $S \models q$.

Intuiție: Cel mai mic punct fix al funcției f_S este mulțimea tuturor atomilor care sunt consecințe logice ale programului.

Avem o metodă de decizie *(decision procedure)* pentru a verifica $S \vdash q$. Metoda constă în:

- calcularea celui mai mic punct fix X al funcției f_S
- dacă $q \in X$ atunci returnăm **true**, altfel returnăm **false**

Această metodă se termină.

Logica de ordinul I (recap.)

Logica de ordinul I - sintaxa

Limbaj de ordinul I $\mathcal{L}$

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

Termenii lui $\mathcal{L}$, notați $Trm_{\mathcal{L}}$, sunt definiți inductiv astfel:

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

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

- dacă $R \in \mathbf{R}$, $ar(R) = n$ și $t_1, \ldots, t_n$ 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 și ψ sunt formule, atunci $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ sunt formule
- dacă φ este o formulă și 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 mulțime nevidă
- $\mathbf{F}^{\mathcal{A}} = \{f^{\mathcal{A}} \mid f \in \mathbf{F}\}$ este o mulțime de operații 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 mulțime de relații 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 funcție $I : V \to A$.

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 $\mathcal{A}$ și o $\mathcal{A}$-interpretare I astfel încât $\mathcal{A}, I \models \varphi$.

Validitate și satisfiabilitate

Propoziție

Dacă φ este o formulă atunci

φ este validă dacă și numai dacă $\neg\varphi$ nu este satisfiabilă.

Algoritmul de unificare

Unificare

- O subtituție σ este o funcție (parțială) de la variabile la termeni,
$$\sigma : V \to Trm_{\mathcal{L}}$$
- Doi termeni t_1 și t_2 se unifică dacă există o substituție θ astfel încât
$$\theta(t_1) = \theta(t_2).$$
- În acest caz, θ se numește unificatorul termenilor t_1 și t_2.
- Un unificator ν pentru t_1 și t_2 este un cel mai general unificator (cgu,mgu) dacă pentru orice alt unificator ν' pentru t_1 și t_2, există o substituție μ astfel încât
$$\nu' = \nu; \mu.$$

Algoritmul de unificare

- Pentru o mulțime finită de termeni $\{t_1, \ldots, t_n\}$, $n \geq 2$, algoritmul de unificare stabilește dacă există un cgu.

- Algoritmul lucrează cu două liste:
  - Lista soluție: S
  - Lista de rezolvat: R

- Inițial:
  - Lista soluție: $S = \emptyset$
  - Lista de rezolvat: $R = \{t_1 \doteq t_2, \ldots, t_{n-1} \doteq t_n\}$

- $\doteq$ este un simbol nou care ne ajută sa formăm perechi de termeni (ecuații).

Algoritmul de unificare

Algoritmul constă în aplicarea regulilor de mai jos:

- SCOATE
  - orice ecuație de forma $t \doteq t$ din R este eliminată.
- DESCOMPUNE
  - orice ecuație de forma $f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n)$ din R este înlocuită cu ecuațiile $t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$.
- REZOLVĂ
  - orice ecuație de forma $x \doteq t$ sau $t \doteq x$ din R, unde variabila x nu apare în termenul t, este mutată sub forma $x \doteq t$ în S.
    În toate celelalte ecuații (din R și S), x este înlocuit cu t.

Algoritmul de unificare

Algoritmul se termină normal dacă $R = \emptyset$. În acest caz, S dă cgu.

Algoritmul este oprit cu concluzia inexistenței unui cgu dacă:

1. În R există o ecuație de forma
$$f(t_1, \ldots, t_n) \doteq g(t'_1, \ldots, t'_k) \text{ cu } f \neq g.$$
2. În R există o ecuație de forma $x \doteq t$ sau $t \doteq x$ și variabila x apare în termenul t.

Algoritmul de unificare - schemă

	Lista soluție S	Lista de rezolvat R
Inițial	$\emptyset$	$t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$
SCOATE	S	$R', \ t \doteq t$
	S	R'
DESCOMPUNE	S	$R', \ f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n)$
	S	$R', \ t_1 \doteq t'_1, \ldots t_n \doteq t'_n$
REZOLVĂ	S	R', $x \doteq t$ sau $t \doteq x$, x nu apare în t
	$x \doteq t$, $S[x/t]$	$R'[x/t]$
Final	S	$\emptyset$

$S[x/t]$: în toate ecuațiile din S, x este înlocuit cu t

Exemplu

Exemplu

- Ecuațiile $\{g(y) \doteq x,\ f(x, h(x), y) \doteq f(g(z), w, z)\}$ au gcu?

S	R	
$\emptyset$	$g(y) \doteq x,\ f(x, h(x), y) \doteq f(g(z), w, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(g(y)), y) \doteq f(g(z), w, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq g(z),\ h(g(y)) \doteq w,\ y \doteq z$	REZOLVĂ
$w \doteq h(g(y)),$ $x \doteq g(y)$	$g(y) \doteq g(z),\ y \doteq z$	REZOLVĂ
$y \doteq z, x \doteq g(z),$ $w \doteq h(g(z))$	$g(z) \doteq g(z)$	SCOATE
$y \doteq z, x \doteq g(z),$ $w \doteq h(g(z))$	$\emptyset$	

- $\nu = \{y/z,\ x/g(z),\ w/h(g(z))\}$ este cgu.

Exemplu

Exemplu

- Ecuațiile $\{g(y) \doteq x,\ f(x, h(y), y) \doteq f(g(z), b, z)\}$ au gcu?

S	R	
$\emptyset$	$g(y) \doteq x,\ f(x, h(y), y) \doteq f(g(z), b, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(y), y) \doteq f(g(z), b, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq g(z),\ h(y) \doteq b,\ y \doteq z$	- EȘEC -

- h și b sunt simboluri de operații diferite!
- Nu există unificator pentru ecuațiile din U.

Exemplu

Exemplu

- Ecuațiile $\{g(y) \doteq x,\ f(x, h(x), y) \doteq f(y, w, z)\}$ au gcu?

S	R	
$\emptyset$	$g(y) \doteq x,\ f(x, h(x), y) \doteq f(y, w, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(g(y)), y) \doteq f(y, w, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq y,\ h(g(y)) \doteq w,\ y \doteq z$	- EȘEC -

- În ecuația $g(y) \doteq y$, variabila y apare în termenul $g(y)$.
- Nu există unificator pentru ecuațiile din U.

Forme prenex și Skolem. Modele Herbrand

Variabile libere. Variabile legate. Enunțuri

Fie φ o formulă și $Var(\varphi)$ mulțimea variabilelor care apar în φ.

- Variabilele libere ale unei formule φ sunt variabilele care nu sunt cuantificate.

- Mulțimea $FV(\varphi)$ a variabilelor libere ale unei formule φ poate fi definită prin inducție după formule:

$$
\begin{array}{lcll}
FV(\varphi) & = & Var(\varphi), & \text{dacă } \varphi \text{ este formulă atomică} \\
FV(\neg\varphi) & = & FV(\varphi) & \\
FV(\varphi \circ \psi) & = & FV(\varphi) \cup FV(\psi), & \text{dacă } \circ \in \{\rightarrow, \vee, \wedge\} \\
FV(\forall x \varphi) & = & FV(\varphi) - \{x\} & \\
FV(\exists x \varphi) & = & FV(\varphi) - \{x\} &
\end{array}
$$

- O variabilă $v \in Var(\varphi)$ care nu este liberă se numește legată în φ.

- Un enunț este o formulă fără variabile libere.

- Pentru orice structură $\mathcal{A}$ și orice enunț φ, o $\mathcal{A}$-interpretare I nu joacă niciun rol în a determina dacă $\mathcal{A}, I \vDash \varphi$.

Enunțuri

Fie φ o formulă și $FV(\varphi) = \{x_1, \ldots, x_n\}$.

Propozitie

Pentru orice structură $\mathcal{A}$ avem

$$\mathcal{A} \vDash \varphi \text{ dacă și numai dacă } \mathcal{A} \vDash \forall x_1 \cdots \forall x_n \, \varphi.$$

A verifica validitatea unei formule revine la
a verifica validitatea enunțului asociat.

Substituții și formule echivalente

- Substituțiile înlocuiesc variabilele libere cu termeni.
- O substituție aplicată unui termen întoarce un alt termen.

- Fie φ o formulă și $t_1, \ldots, t_n$ termeni care nu conțin variabile din φ. Notăm $\varphi[x_1/t_1, \ldots, x_n/t_n]$ formula obținută din φ substituind toate aparițiile libere ale lui $x_1, \ldots, x_n$ cu $t_1, \ldots, t_n$.

$$\varphi[x_1/t_1, \ldots, x_n/t_n] = \{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\}\varphi$$

- Notăm prin $\varphi \mathrel{\models\!\!\!\dashv} \psi$ faptul că $\models \varphi \leftrightarrow \psi$, adică φ și ψ au aceleași modele.

Forma rectificată

- O formulă φ este în formă rectificată dacă:
  1. nici o variabilă nu apare și liberă și legată
  2. cuantificatori distincți leagă variabile distincte
- Pentru orice formulă φ există o formulă φ^r în formă rectificată astfel încât $\varphi \mathrel{\vDash\!\!\dashv} \varphi^r$.
- Intuitiv, forma rectificată a unei formule se obține prin redenumirea variabilelor astfel încât să nu apară conflicte.

În continuare vom presupune că
toate formulele sunt în formă rectificată.

Forma prenex

O formulă prenex este o formulă de forma

$$Q_1 x_1\, Q_2 x_2 \ldots Q_n x_n\, \varphi$$

unde $Q_i \in \{\forall, \exists\}$ pentru orice $i \in \{1, \ldots, n\}$, $x_1, \ldots, x_n$ sunt variabile distincte și φ nu conține cuantificatori.

Cum calculăm forma prenex?

- Se înlocuiesc $\rightarrow$ și $\leftrightarrow$:

$$\begin{aligned} \varphi \rightarrow \psi &\quad \dashv\vdash \quad \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi &\quad \dashv\vdash \quad (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi) \end{aligned}$$

- Se aplică următoarele echivalențe:

$$\begin{aligned} \neg\exists x \neg\varphi &\quad \dashv\vdash \quad \forall x \varphi & \qquad \forall x \varphi \wedge \forall x \psi &\quad \dashv\vdash \quad \forall x (\varphi \wedge \psi) \\ \neg\forall x \neg\varphi &\quad \dashv\vdash \quad \exists x \varphi & \qquad \exists x \varphi \vee \exists x \psi &\quad \dashv\vdash \quad \exists x (\varphi \vee \psi) \end{aligned}$$

$$\begin{aligned} \neg\exists x \varphi &\quad \dashv\vdash \quad \forall x \neg\varphi & \qquad \forall x \forall y \varphi &\quad \dashv\vdash \quad \forall y \forall x \varphi \\ \neg\forall x \varphi &\quad \dashv\vdash \quad \exists x \neg\varphi & \qquad \exists x \exists y \varphi &\quad \dashv\vdash \quad \exists y \exists x \varphi \end{aligned}$$

$$\begin{aligned} \forall x \varphi \vee \psi &\quad \dashv\vdash \quad \forall x (\varphi \vee \psi) \text{ dacă } x \notin FV(\psi) \\ \forall x \varphi \wedge \psi &\quad \dashv\vdash \quad \forall x (\varphi \wedge \psi) \text{ dacă } x \notin FV(\psi) \\ \exists x \varphi \vee \psi &\quad \dashv\vdash \quad \exists x (\varphi \vee \psi) \text{ dacă } x \notin FV(\psi) \\ \exists x \varphi \wedge \psi &\quad \dashv\vdash \quad \exists x (\varphi \wedge \psi) \text{ dacă } x \notin FV(\psi) \end{aligned}$$

Forma prenex

Exercițiu

Considerăm un limbaj de ordinul I cu $\mathbf{R} = \{P, R, Q\}$ cu $ari(P) = 1$ și $ari(R) = ari(Q) = 2$.
Găsiți forma echivalentă prenex pentru următoarea formulă:

$$\forall x \exists y (R(x,y) \to R(y,x)) \to \exists x R(x,x)$$

Soluție

$$\begin{array}{lll}
 & \forall x \exists y (R(x,y) \to R(y,x)) \to \exists x R(x,x) & \\
\mathrel{\Dashv\vDash} & \forall x \exists y (R(x,y) \to R(y,x)) \to \exists z R(z,z) & \text{(redenumim variabile)} \\
\mathrel{\Dashv\vDash} & \neg \forall x \exists y (\neg R(x,y) \lor R(y,x)) \lor \exists z R(z,z) & \\
\mathrel{\Dashv\vDash} & \exists x \forall y (R(x,y) \land \neg R(y,x)) \lor \exists z R(z,z) & \\
\mathrel{\Dashv\vDash} & \exists z (\exists x \forall y (R(x,y) \land \neg R(y,x)) \lor R(z,z)) & \\
\mathrel{\Dashv\vDash} & \exists z \exists x (\forall y (R(x,y) \land \neg R(y,x)) \lor R(z,z)) & \\
\mathrel{\Dashv\vDash} & \exists z \exists x \forall y ((R(x,y) \land \neg R(y,x)) \lor R(z,z)) &
\end{array}$$

Forma Skolem

Fie φ enunț în formă prenex. Definim φ^{sk} și $\mathcal{L}^{sk}(\varphi)$ astfel:

- dacă φ este liberă de cuantificatori, atunci $\varphi^{sk} = \varphi$ și $\mathcal{L}^{sk}(\varphi) = \mathcal{L}$,
- dacă φ este universală, atunci $\varphi^{sk} = \varphi$ și $\mathcal{L}^{sk}(\varphi) = \mathcal{L}$,
- dacă $\varphi = \exists x \psi$ atunci introducem un nou simbol de constantă c și considerăm $\varphi^1 = \psi[x/c]$, $\mathcal{L}^1 = \mathcal{L} \cup \{c\}$.
- dacă $\varphi = \forall x_1 \dots \forall x_k \exists x \psi$ atunci introducem un nou simbol de funcție f de aritate k și considerăm $\mathcal{L}^1 = \mathcal{L} \cup \{f\}$,

$$\varphi^1 = \forall x_1 \dots \forall x_k \psi[x/f(x_1 \dots x_k)]$$

În ambele cazuri, φ^1 are cu un cuantificator existențial mai puțin decât φ. Dacă φ^1 este liberă de cuantificatori sau universală, atunci $\varphi^{sk} = \varphi^1$. Dacă φ^1 nu este universală, atunci formăm $\varphi^2, \varphi^3, \dots$, până ajungem la o formulă universală și aceasta este φ^{sk}.

Definiție

φ^{sk} este o formă Skolem a lui φ.

Forma Skolem

Exercițiu

Considerăm un limbaj de ordinul I cu $\mathbf{C} = \{b\}$ și $\mathbf{R} = \{P, R, Q\}$ cu $ari(P) = 1$ și $ari(R) = ari(Q) = 2$.

Găsiți forma Skolem pentru următoarea formulă în formă prenex

$$\varphi = \forall x \exists y \forall z \exists w (R(x, y) \land (R(y, z) \to (R(z, w) \land R(w, w))))$$

Soluție

$\varphi_1 = \forall x \forall z \exists w (R(x, f(x)) \land (R(f(x), z) \to (R(z, w) \land R(w, w))))$
$(y \mapsto f(x))$

$\varphi_2 = \forall x \forall z (R(x, f(x)) \land (R(f(x), z) \to (R(z, g(x, z)) \land R(g(x, z), g(x, z)))))$
$(w \mapsto g(x, z))$

$\varphi^{sk} = \varphi_2$

Model Herbrand

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

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

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

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

- pentru orice simbol de constantă c, $c^{\mathcal{H}} = c$
- pentru orice simbol de funcție f de aritate n,
  $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$

Atenție! Într-o structură Herbrand nu fixăm o definiție pentru relații: pentru orice simbol de relație R de aritate n, $R^{\mathcal{H}}(t_1, \ldots, t_n) \subseteq (T_{\mathcal{L}})^n$

O interpretare Herbrand este o interpretare $H : V \to T_{\mathcal{L}}$

O structură Herbrand $\mathcal{H}$ este model al unei formule φ dacă $\mathcal{H} \vDash \varphi$. În acest caz spunem că $\mathcal{H}$ este model Herbrand al lui φ.

Teorema lui Herbrand

Teorema lui Herbrand

Fie $n \geq 0$ și $\varphi = \forall x_k \ldots \forall x_1 \psi$ un enunț în forma Skolem.
Atunci φ are un model dacă și numai dacă are un model Herbrand.

Teorema lui Herbrand reduce problema satisfiabilității la găsirea unui model Herbrand.

Universul Herbrand al unei formule

Fie φ un enunț în forma Skolem, adică $\varphi = \forall x_1 \ldots \forall x_n \psi$.

Definim $T(\varphi)$, universul Herbrand al formulei φ, astfel:

- dacă c este o constantă care apare în φ atunci $c \in T(\varphi)$,
- dacă φ nu conține nicio constantă atunci alegem o constantă arbitrară c și considerăm că $c \in T(\varphi)$,
- dacă f este un simbol de funcție care apare în φ cu $ari(f) = n$ și $t_1, \ldots, t_n \in T(\varphi)$ atunci $f(t_1, \ldots, t_n) \in T(\varphi)$.

Intuitiv, $T(\varphi)$ este mulțimea termenilor care se pot construi folosind simbolurile de funcții care apar în φ.

Definim extensia Herbrand a lui φ astfel

$$\mathcal{H}(\varphi) = \{\psi[x_1/t_1, \ldots, x_n/t_n] \mid t_1, \ldots, t_n \in T(\varphi)\}$$

Extensia Herbrand a unei formule

Fie φ un enunț în forma Skolem, adică $\varphi = \forall x_1 \ldots \forall x_n \psi$.

Teoremă

Sunt echivalente:

- φ este satisfiabilă,
- φ are un model Herbrand $\mathcal{H}$ cu proprietatea că $\mathbf{R}^{\mathcal{H}} \subseteq T(\varphi)^n$ pentru orice relație $R \in \mathbf{R}$ cu $ari(R) = n$ care apare în φ,
- mulțimea de formule $\mathcal{H}(\varphi)$ este satisfiabilă.

Extensia Herbrand a unei formule

Exercițiu

Considerăm un limbaj de ordinul I cu $\mathbf{F} = \{f, g\}$ cu $ari(f) = 2$ și $ari(g) = 1$, $\mathbf{C} = \{b, c\}$ și $\mathbf{R} = \{P, Q\}$ cu $ari(P) = 3, ari(Q) = 2$.
Descrieți termenii din universul Herbrand și formulele din expansiunea Herbrand a următoarei formule:

$$\varphi := \forall x \forall y\, P(c, f(x, b), g(y))$$

Soluție

Universul Herbrand
$T(\varphi) = \{b, c, g(b), g(c), g(g(b)), g(g(c)), \ldots,$
$f(b, c), f(b, g(b)), f(b, g(c)), f(g(c), b), f(g(c), g(c)), \ldots\}$

Expansiunea Herbrand
$\mathcal{H}(\varphi) = \{P(c, f(b, b), g(b)), P(c, f(b, b), g(c)),$
$P(c, f(c, b), g(b)), P(c, f(g(b), b), g(g(g(b)))), \ldots\}$

Logica de ordinul I

- Cercetarea validității poate fi redusă la cercetarea satisfiabilității.
- Cercetarea satisfiabilității unei formule poate fi redusă la cercetarea satisfiabilității unui enunț în forma Skolem.
- Teorema lui Herbrand reduce verificarea satisfiabilitătii unui enunț în forma Skolem la verificarea satisfiabilității în universul Herbrand.
- În situații particulare Teorema lui Herbrand ne dă o procedură de decizie a satisfiabilității, dar acest fapt nu este adevărat în general:

  dacă limbajul $\mathcal{L}$ conține cel putin o constantă și cel puțin un simbol de funcție f cu $ari(f) \geq 1$ atunci universul Herbrand $T_{\mathcal{L}}$ este infinit.

Logica de ordinul I

Problema validității

- nu este decidabilă.
- este semi-decidabilă.

Problema satisfiabilității

- nu este decidabilă.
- nu este semi-decidabilă.

Formă clauzală. Rezoluție

Literali. FNC

- În logica propozițională un literal este o variabilă sau negația unei variabile.

$$literal := \; p \mid \neg p \qquad \text{unde } p \text{ este variabilă propozițională}$$

- În logica de ordinul I un literal este o formulă atomică sau negația unei formule atomice.

$$literal := \; P(t_1, \ldots, t_n) \mid \neg P(t_1, \ldots, t_n)$$

unde $P \in \mathbf{R}, ari(P) = n$, și $t_1, \ldots, t_n$ sunt termeni.

- Pentru un literal L vom nota cu L^c literalul complement.

O formulă este în formă normală conjunctivă (FNC) dacă este o conjuncție de disjuncții de literali.

Forma clauzală în logica propozițională

- Pentru orice formulă α există o FNC α^{fc} astfel încât $\alpha \mathrel{\dashv\vDash} \alpha^{fc}$.
- Pentru o formulă din logica propozițională determinăm FNC corespunzătoare prin următoarele transformări:

1. înlocuirea implicațiilor și echivalențelor

$$\begin{aligned} \varphi \to \psi &\quad \mathrel{\dashv\vDash} \quad \neg\varphi \vee \psi \\ \varphi \leftrightarrow \psi &\quad \mathrel{\dashv\vDash} \quad (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi) \end{aligned}$$

2. regulile De Morgan

$$\begin{aligned} \neg(\varphi \vee \psi) &\quad \mathrel{\dashv\vDash} \quad \neg\varphi \wedge \neg\psi \\ \neg(\varphi \wedge \psi) &\quad \mathrel{\dashv\vDash} \quad \neg\varphi \vee \neg\psi \end{aligned}$$

3. principiului dublei negații

$$\neg\neg\psi \quad \mathrel{\dashv\vDash} \quad \psi$$

4. distributivitatea

$$\begin{aligned} \varphi \vee (\psi \wedge \chi) &\quad \mathrel{\dashv\vDash} \quad (\varphi \vee \psi) \wedge (\varphi \vee \chi) \\ (\psi \wedge \chi) \vee \varphi &\quad \mathrel{\dashv\vDash} \quad (\psi \vee \varphi) \wedge (\chi \vee \varphi) \end{aligned}$$

Forma clauzală în logica de ordinul I

- O formulă este formă normală conjunctivă prenex (FNCP) dacă
  - este în formă prenex $Q_1 x_1 \ldots Q_n x_n \psi$ ($Q_i \in \{\forall, \exists\}$ oricare i)
  - ψ este FNC
- O formulă este formă clauzală dacă este enunț universal și FNCP:

$$\forall x_1 \ldots \forall x_n \psi \text{ unde } \psi \text{ este FNC}$$

- Pentru orice formulă φ din logica de ordinul I există o formă clauzală φ^{fc} astfel încât

$$\varphi \text{ este satisfiabilă dacă și numai dacă } \varphi^{fc} \text{ este satisfiabilă}$$

- Pentru o formulă φ, forma clauzală φ^{fc} se poate calcula astfel:
  1. se determină forma rectificată
  2. se cuantifică universal variabilele libere
  3. se determină forma prenex
  4. se determină forma Skolem

     în acest moment am obținut o formă Skolem $\forall x_1 \ldots \forall x_n \psi$
  5. se determină o FNC ψ' astfel încât $\psi \vDash\!\!\!\dashv \psi'$
  6. φ^{fc} este $\forall x_1 \ldots \forall x_n \psi'$

Clauze

- O clauză este o disjuncție de literali.
- Dacă $L_1, \ldots, L_n$ sunt literali atunci clauza $L_1 \vee \ldots \vee L_n$ o vom scrie ca mulțimea $\{L_1, \ldots, L_n\}$

  clauză = mulțime de literali
- Clauza $C = \{L_1, \ldots, L_n\}$ este satisfiabilă dacă $L_1 \vee \ldots \vee L_n$ este satisfiabilă.
- O clauză C este trivială dacă conține un literal și complementul lui.
- Când $n = 0$ obținem clauza vidă, care se notează $\square$
- Prin definiție, clauza $\square$ nu este satisfiabilă.

Forma clauzală

- Observăm că o FNC este o conjuncție de clauze.
- Dacă $C_1, \ldots, C_k$ sunt clauze atunci $C_1 \wedge \ldots \wedge C_k$ o vom scrie ca mulțimea $\{C_1, \ldots, C_k\}$

  FNC = mulțime de clauze

- O mulțime de clauze $\mathcal{C} = \{C_1, \ldots, C_k\}$ este satisfiabilă dacă $C_1 \wedge \ldots \wedge C_k$ este satisfiabilă
- Când $k = 0$ obținem mulțimea de clauze vidă, pe care o notăm $\{\}$
- Prin definiție, mulțimea de clauze vidă $\{\}$ este satisfiabilă.

  $\{\}$ este satisfiabilă, dar $\{\square\}$ nu este satisfiabilă

Forma clauzală

- Dacă φ este o formulă în calculul propozițional, atunci
$$\varphi^{fc} = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{n_i} L_{ij} \text{ unde } L_{ij} \text{ sunt literali}$$

- Dacă φ o formulă în logica de ordinul I, atunci
$$\varphi^{fc} = \forall x_1 \ldots \forall x_n \left(\bigwedge_{i=1}^{k} \bigvee_{j=1}^{n_i} L_{ij} \right) \text{ unde } L_{ij} \text{ sunt literali}$$

φ este satisfiabilă dacă și numai dacă

φ^{fc} este satisfiabilă dacă și numai dacă

$\{\{L_{11}, \ldots, L_{1n_1}\}, \ldots, \{L_{k1}, \ldots, L_{kn_k}\}\}$ este satisfiabilă

Rezoluția în logica propozițională (recap.)

Regula rezoluției

$$Rez \ \frac{C_1 \cup \{p\}, C_2 \cup \{\neg p\}}{C_1 \cup C_2}$$

unde C_1, C_2 clauze, iar p este variabila propozițională astfel încât $\{p, \neg p\} \cap C_1 = \emptyset$ și $\{p, \neg p\} \cap C_2 = \emptyset$.

Fie $\mathcal{C}$ o mulțime de clauze. O derivare prin rezoluție din $\mathcal{C}$ este o secvență finită de clauze astfel încât fiecare clauză este din $\mathcal{C}$ sau rezultă din clauzele anterioare prin rezoluție (este rezolvent).

Procedura Davis-Putnam DPP (informal)

Intrare: o mulțime $\mathcal{C}$ de clauze

Se repetă următorii pași:

- se elimină clauzele triviale
- se alege o variabilă p
- se adaugă la mulțimea de clauze toți rezolvenții obținuti prin aplicarea *Rez* pe variabila p
- se șterg toate clauzele care conțin p sau $\neg p$

Ieșire: dacă la un pas s-a obținut $\square$, mulțimea $\mathcal{C}$ nu este satisfiabilă; altfel $\mathcal{C}$ este satisfiabilă.

Procedura Davis-Putnam DPP

Exercițiu

Folosind algoritmul Davis-Putnam, cercetați dacă următoarea mulțime de clauze din calculul propozițional este satisfiabilă:

$$\mathcal{C} = \{\{v_0\}, \{\neg v_0, v_1\}, \{\neg v_1, v_2, v_3\}, \{\neg v_3, v_4\}, \{\neg v_4\}, \{\neg v_2\}\}$$

Soluție

Pasul 1.
Alegem variabila v_0 și selectăm $\mathcal{C}_0^{v_0} := \{\{v_0\}\}$, $\mathcal{C}_0^{\neg v_0} = \{\{\neg v_0, v_1\}\}$.
Mulțimea rezolvenților posibili este $\mathcal{R}_0 := \{\{v_1\}\}$.
Se elimină clauzele în care apare v_0, adăugăm rezolvenții și obținem:

$$\mathcal{C}_1 := \{\{\neg v_1, v_2, v_3\}, \{\neg v_3, v_4\}, \{\neg v_4\}, \{\neg v_2\}, \{v_1\}\}$$

Procedura Davis-Putnam DPP

Soluție (cont.)

Pasul 2.
Alegem variabila v_1 și selectăm $\mathcal{C}_1^{v_1} := \{\{v_1\}\}$ și $\mathcal{C}_1^{\neg v_1} := \{\{\neg v_1, v_2, v_3\}\}$.
Mulțimea rezolvenților posibili este $\mathcal{R}_1 := \{\{v_2, v_3\}\}$.
Se elimină clauzele în care apare v_1, adăugăm rezolvenții și obținem:
$\mathcal{C}_2 := \{\{\neg v_3, v_4\}, \{\neg v_4\}, \{\neg v_2\}, \{v_2, v_3\}\}$.

Pasul 3.
Alegem variabila v_2 și selectăm $\mathcal{C}_2^{v_2} := \{\{v_2, v_3\}\}$, $\mathcal{C}_2^{\neg v_2} := \{\{\neg v_2\}\}$.
Mulțimea rezolvenților posibili este $\mathcal{R}_2 := \{\{v_3\}\}$.
Se elimină clauzele în care apare v_2, adăugăm rezolvenții și obținem:
$\mathcal{C}_3 := \{\{\neg v_3, v_4\}, \{\neg v_4\}, \{v_3\}\}$.

Procedura Davis-Putnam DPP

Soluție (cont.)

Pasul 4.
Alegem variabila v_3 și selectăm $\mathcal{C}_3^{v_3} := \{\{v_3\}\}$, $\mathcal{C}_3^{\neg v_3} := \{\{\neg v_3, v_4\}\}$.
Mulțimea rezolvenților posibili este $\mathcal{R}_3 := \{\{v_4\}\}$.
Se elimină clauzele în care apare v_3, adăugăm rezolvenții și obținem:
$\mathcal{C}_4 := \{\{\neg v_4\}, \{v_4\}\}$.

Pasul 5.
Alegem variabila v_4 și selectăm $\mathcal{C}_4^{v_4} := \{\{v_4\}\}$, $\mathcal{C}_4^{\neg v_4} := \{\{\neg v_4\}\}$.
Mulțimea rezolvenților posibili este $\mathcal{R}_4 := \{\square\}$.
Se elimină clauzele în care apare v_4, adăugăm rezolvenții și obținem:
$\mathcal{C}_5 := \{\square\}$.
Deoarece $\mathcal{C}_5 = \{\square\}$, obținem că mulțimea de clauze $\mathcal{C}$ nu este satisfiabilă.

Rezoluția în logica de ordinul I

Clauze închise

- Fie C o clauză. Spunem că C' este o instață a lui C dacă există o substituție $\theta : V \to Trm_{\mathcal{L}}$ astfel încât $C' = \theta(C)$.

  Spunem că C' este o instanță închisă a lui C dacă există o substituție $\theta : V \to T_{\mathcal{L}}$ such that $C' = \theta(C)$ (C' se obține din C înlocuind variabilele cu termeni din universul Herbrand)

- Fie $\mathcal{C}$ o mulțime de clauze. Definim

$$\mathcal{H}(\mathcal{C}) := \{\theta(C) \mid C \in \mathcal{C}, \theta : V \to T_{\mathcal{L}}\}$$

  $\mathcal{H}(\mathcal{C})$ este mulțimea instanțelor închise ale clauzelor din $\mathcal{C}$.

Rezoluția pe clauze închise

$$Rez \ \frac{C_1 \cup \{L\}, C_2 \cup \{\neg L\}}{C_1 \cup C_2}$$

unde C_1, C_2 clauze închise, iar L este o formulă atomică închisă astfel încât $\{L, \neg L\} \cap C_1 = \emptyset$ și $\{L, \neg L\} \cap C_2 = \emptyset$.

Teoremă

Fie φ o formulă arbitrară în logica de ordinul I. Atunci $\vDash \varphi$ dacă și numai dacă există o derivare pentru $\square$ din $\mathcal{H}(\mathcal{C})$ folosind *Rez*, unde $\mathcal{C}$ este mulțimea de clauze asociată lui $(\neg\varphi)^{fc}$.

Exercițiu

Considerăm următoarea mulțime de clauze în logica de ordinul I:

$$\mathcal{C} = \{\{\neg P(f(a)), Q(y)\}, \{P(y)\}, \{\neg Q(b)\}\}$$

Arătați că $\mathcal{C}$ nu este satisfiabilă prin următoarele metode:

1) Găsiți o submulțime finită nesatisfiabilă lui $\mathcal{H}(\mathcal{C})$.
2) Găsiți o derivare pentru $\square$ folosind rezoluția pe clauze închise.

Rezoluția pe clauze închise

Soluție

1) $\mathcal{H}(C) = \{\{\neg Q(b)\}, \{\neg P(f(a)), Q(a)\}, \{\neg P(f(a)), Q(b)\}, \{P(a)\}, \{P(b)\}, \{P(f(a))\}, \cdots\}$

   O submulțime nesatisfiabilă este
   $\{\{\neg P(f(a)), Q(b)\}, \{P(f(a))\}, \{\neg Q(b)\}\}$

2) Derivare pentru $\square$:
   1. $\{\neg P(f(a)), Q(b)\}$
   2. $\{P(f(a))\}$
   3. $\{Q(b)\}$
   4. $\{\neg Q(b)\}$
   5. $\square$

Rezoluția pe clauze arbitrare

Regula rezoluției pentru clauze arbitrare

$$Rez \ \frac{C_1, C_2}{(\sigma C_1 \setminus \sigma Lit_1) \cup (\sigma C_2 \setminus \sigma Lit_2)}$$

dacă următoarele condiții sunt satisfăcute:

1. C_1, C_2 clauze care nu au variabile comune,
2. $Lit_1 \subseteq C_1$ și $Lit_2 \subseteq C_2$ sunt mulțimi de literali,
3. σ este un cgu pentru Lit_1 și Lit_2^c, adică
   σ unifică toți literalii din Lit_1 și Lit_2^c.

   O clauză C se numește rezolvent pentru C_1 și C_2 dacă există o redenumire de variabile $\theta : V \to V$ astfel încât C_1 și θC_2 nu au variabile comune și C se obține din C_1 și θC_2 prin Rez.

Rezoluția în logica de ordinul I

Exercițiu

Găsiți o derivare prin rezoluție a $\square$ pentru următoarea mulțime de clauze:

$C_1 = \{\neg P(x),\ R(x, f(x))\}$
$C_2 = \{\neg R(a, x), Q(x)\}$
$C_3 = \{P(a)\}$
$C_4 = \{\neg Q(f(x))\}$

unde P, Q, R sunt simboluri de relații, f e simbol de funție, a este o constantă, x, y sunt variabile.

Soluție

$C_5 = \{R(a, f(a))\}$ din $Rez, C_1, C_3, \theta = \{x \leftarrow a\}$
$C_4' = \{\neg Q(f(z))\}$ redenumire
$C_6 = \{\neg R(a, f(z))\}$ din $Rez, C_4', C_2, \theta = \{y \leftarrow f(z)\}$
$\square$ din $Rez, C_6, C_5, \theta = \{z \leftarrow a\}$

Logica Horn

Clauze definite. Programe logice. Clauze Horn

- clauză:

$$\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\} \quad \text{sau} \quad Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k$$

  unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

- clauză program definită: $k = 1$
  - cazul $n > 0$: $\quad Q_1 \wedge \ldots \wedge Q_n \to P$
  - cazul $n = 0$: $\quad \top \to P$ (clauză unitate, fapt)

  Program logic definit = mulțime finită de clauze definite

- scop definit (țintă, întrebare): k=0
  - $Q_1 \wedge \ldots \wedge Q_n \to \bot$

- clauza vidă $\square$: $n = k = 0$

Clauza Horn = clauză program definită sau clauză scop ($k \leq 1$)

Programare logica

- Logica clauzelor definite/Logica Horn: un fragment al logicii de ordinul I în care singurele formule admise sunt clauze Horn
  - formule atomice: $P(t_1, \dots, t_n)$
  - $Q_1 \wedge \dots \wedge Q_n \to P$
    unde toate Q_i, P sunt formule atomice, $\top$ sau $\bot$

- Problema programării logice: reprezentăm cunoștințele ca o mulțime de clauze definite KB și suntem interesați să aflăm răspunsul la o întrebare de forma $Q_1 \wedge \dots \wedge Q_n$, unde toate Q_i sunt formule atomice

$$KB \vDash Q_1 \wedge \dots \wedge Q_n$$

  - Variabilele din KB sunt cuantificate universal.
  - Variabilele din $Q_1, \dots, Q_n$ sunt cuantificate existențial.

Limbajul PROLOG are la bază logica clauzelor Horn.

Modele Herbrand

Definim o ordine între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$ este definită astfel:

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

Semantica unui program logic definit KB este dată de
cel mai mic model Herbrand al lui KB!

- Definim $\mathcal{LH}_{KB} := \bigcap\{\mathcal{H} \mid \mathcal{H}$ model Herbrand pentru $KB\}$
- $\mathcal{LH}_{KB} \vDash KB$.
- Vom caracteriza cel mai mic model Herbrand $\mathcal{LH}_{KB}$ printr-o construcție de punct fix.

Cel mai mic model Herbrand

- O instanță de bază a unei clauze $Q_1(x_1) \land \ldots \land Q_n(x_n) \to P(y)$ este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.

- Pentru o mulțime de clauze definite KB, o formulă atomică P și o mulțime de formule atomice X,

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

  dacă există o instanță de bază a unei clauze $Q_1(x_1) \land \ldots \land Q_n(x_n) \to P(y)$ din KB astfel încât P este instanța lui $P(y)$ și instanța lui $Q_i(x_i)$ este în X, pentru orice $i = 1, \ldots, n$.

- Baza Herbrand $B_{\mathcal{L}}$ este mulțimea formulelor atomice fără variabile.

- Pentru o mulțime de clauze definite KB, definim

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

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

Cel mai mic model Herbrand

Fie KB un program logic definit.

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

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

Propoziție (caracterizarea $\mathcal{LH}_{KB}$)

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

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

Sistem de deducție *backchain*

Sistem de deducție pentru clauze Horn

Pentru un program logic definit KB avem

- Axiome: orice clauză din KB
- Regula de deducție: regula *backchain*

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n) \quad (Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \to P)}{\theta(Q)}$$

unde $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \to P \in KB$, iar θ este cgu pentru Q și P.

Rezoluția SLD

Fie T o mulțime de clauze definite.

$$\text{SLD}\quad \boxed{\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

unde

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din KB (în care toate variabilele au fost redenumite) și
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este c.g.u pentru Q_i și Q

Rezoluția SLD

Fie KB o mulțime de clauze definite și $Q_1 \land \ldots \land Q_m$ o întrebare, unde Q_i sunt formule atomice.

- O derivare din KB prin rezoluție SLD este o secvență

  $$G_0 := \neg Q_1 \lor \ldots \lor \neg Q_m, \quad G_1, \quad \ldots, \quad G_k, \ldots$$

  în care G_{i+1} se obține din G_i prin regula SLD.
- Dacă există un k cu $G_k = \square$ (clauza vidă), atunci derivarea se numește SLD-respingere.

Rezoluția SLD

Exercițiu

Găsiți o SLD-respingere pentru următorul program Prolog și ținta:

```
1. p(X) :- q(X,f(Y)), r(a).        ?- p(X), q(Y,Z).
2. p(X) :- r(X).
3. q(X,Y) :- p(Y).
4. r(X) :- q(X,Y).
5. r(f(b)).
```

Soluție

$G_0 = \neg p(X) \vee \neg q(Y,Z)$	
$G_1 = \neg r(X_1) \vee \neg q(Y,Z)$	(2 cu $\theta(X) = X_1$)
$G_2 = \neg q(Y,Z)$	(5 cu $\theta(X_1) = f(b)$)
$G_3 = \neg p(Z_1)$	(3 cu $\theta(X) = Y_1$ și $\theta(Y) = Z_1$)
$G_4 = \neg r(X)$	(2 cu $\theta(Z_1) = X$)
$G_5 = \square$	(5 cu $\theta(X) = f(b)$)

Rezoluția SLD - arbori de căutare

Arbori SLD

- Presupunem că avem o mulțime de clauze definite KB și o țintă $G_0 = \neg Q_1 \vee \ldots \vee \neg Q_m$
- Construim un arbore de căutare (arbore SLD) astfel:
  - Fiecare nod al arborelui este o țintă (posibil vidă)
  - Rădăcina este G_0
  - Dacă arborele are un nod G_i, iar G_{i+1} se obține din G_i folosind regula SLD folosind o clauză $C_i \in KB$, atunci nodul G_i are copilul G_{i+1}. Muchia dintre G_i și G_{i+1} este etichetată cu C_i.
- Dacă un arbore SLD cu rădăcina G_0 are o frunză $\square$ (clauza vidă), atunci există o SLD-respingere a lui G_0 din KB.

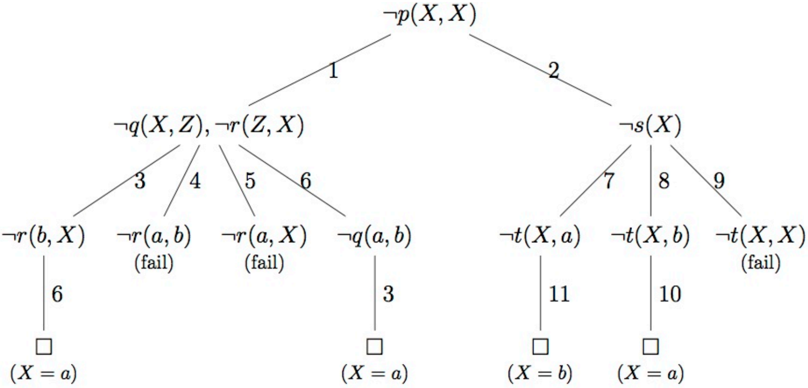
Exercițiu

Desenați arborele SLD pentru programul Prolog de mai jos și ținta
`?- p(X,X).`

```
1. p(X,Y) :- q(X,Z), r(Z,Y).
2. p(X,X) :- s(X).
3. q(X,b).
4. q(b,a).
5. q(X,a) :- r(a,X).
6. r(b,a).
7. s(X) :- t(X,a).
8. s(X) :- t(X,b).
9. s(X) :- t(X,X).
10. t(a,b).
11. t(b,a).
```

Rezoluția SLD - arbori de căutare

Soluție



Introducere în semantică

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

Vom defini un limbaj şi semantica lui operaţională în PROLOG!

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește 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:

$$\begin{aligned}
\langle \texttt{int}\ x = 0\ ;\ x = x + 1\ ;\ ,\ \bot\rangle &\to \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle\\
&\to \langle x = 0 + 1\ ;\ ,\ x \mapsto 0\rangle\\
&\to \langle x = 1\ ;\ ,\ x \mapsto 0\rangle\\
&\to \langle \{\}\ ,\ x \mapsto 1\rangle
\end{aligned}$$

Limbajul IMP

Vom implementa un limbaj care conține:

- Expresii
  - Aritmetice `x + 3`
  - Booleene `x >= 7`
- Instrucțiuni
  - De atribuire `x = 5`
  - Condiționale `if(x >= 7, x =5, x = 0)`
  - De ciclare `while(x >= 7,x = x - 1)`
- Compunerea instruțiunilor `x=7;while(x>=0,x=x-1)`
- Blocuri de instrucțiuni `{x=7;while(x>=0,x=x-1)}`

Limbajul IMP

Exemplu

Un program în limbajul IMP

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

- Semantica
  după execuția programului, se evaluează `sum`

Sintaxa BNF a limbajului IMP

$$
\begin{aligned}
E ::= {} & n \mid x \\
& \mid E + E \mid E - E \mid E * E
\end{aligned}
$$

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

$$
\begin{aligned}
St ::= {} & \texttt{skip} \\
& \mid x \texttt{=} E \\
& \mid \texttt{if(}\ B\ ,\ St\ ,\ St\ \texttt{)} \\
& \mid \texttt{while(}\ B\ ,\ St\ \texttt{)} \\
& \mid \texttt{\{}\ St\ \texttt{\}} \mid St\ ;\ St
\end{aligned}
$$

$$
P ::= \texttt{\{}\ St\ \texttt{\}},\ E
$$

Implementarea limbajului IMP în Prolog

- {} și ; 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 semnificația obișnuită

- pentru valori numerice folosim întregii din Prolog

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

- pentru identificatori folosim atomii din Prolog

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

Expresiile aritmetice

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

Prolog

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

Expresiile aritmetice

Exemplu

```
?- aexp(1000).
true.
?- aexp(id).
true.
?- aexp(id + 1000).
true.
?- aexp(2 + 1000).
true.
?- aexp(x * y).
true.
?- aexp(- x).
false.
```

Expresiile booleene

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

Prolog

```
bexp(true). bexp(false).
bexp(and(BE1,BE2)) :- bexp(BE1), bexp(BE2).
bexp(or(BE1,BE2)) :-  bexp(BE1), bexp(BE2).
bexp(not(BE)) :-  bexp(BE).

bexp(A1 =< A2) :- aexp(A1), aexp(A2).
bexp(A1 >= A2) :- aexp(A1), aexp(A2).
bexp(A1 == A2) :- aexp(A1), aexp(A2).
```

Expresiile booleene

Exemplu

```
?- bexp(true).
true.
?- bexp(id).
false.
?- bexp(not(1 =< 2)).
true.
?- bexp(or(1 =< 2,true)).
true.
?- bexp(or(a =< b,true)).
true.
?- bexp(not(a)).
false.
?- bexp(!(a)).
false.
```

Instrucțiunile

St ::= `skip`
| x `=` E `;`
| `if(` B `)` St `else` St
| `while(` B `)` St
| `{` St `}` | St `;` St

Prolog

```
stmt(skip).
stmt(X = AE) :- atom(X), aexp(AE).
stmt(St1;St2) :- stmt(St1), stmt(St2).
stmt((St1;St2)) :- stmt(St1), stmt(St2).
stmt({St}) :- stmt(St).
stmt(if(BE,St1,St2)) :-  bexp(BE), stmt(St1), stmt(St2).
stmt(while(BE,St)) :- bexp(BE), stmt(St).
```

Instrucțiunile

Exemplu

```
?- stmt(id = 5).
true.
?- stmt(id = a).
true.
?- stmt(3 = 6).
false.
?- stmt(if(true, x=2;y=3, x=1;y=0)).
true.
?- stmt(while(x =< 0,skip)).
true.
?- stmt(while(x =< 0,)).
false.
?- stmt(while(x =< 0,skip)).
true .
```

Programele

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

Prolog

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

Exemplu

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

?- test0.
true.

Semantica small-step

- Definește cel mai mic pas de execuție ca o relație de tranziție între configurații:

  $\langle cod \;,\; \sigma\rangle \rightarrow \langle cod' \;,\; \sigma'\rangle$ `smallstep(Cod,S1,Cod',S2)`
- Execuția se obține ca o succesiune de astfel de tranziții.
- Starea executiei unui program IMP la un moment dat este o funcție parțială: $\sigma = n \mapsto 10, sum \mapsto 0$, 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

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

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

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):-
                                 smallstepA(AE1,S,AE2,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 $*$ și − se definesc similar.

Semantica expresiilor booleene

- Semantica operatorului de comparație

$\langle i_1 \texttt{ =< } i_2\ ,\ \sigma\rangle \to \langle \texttt{false}\ ,\ \sigma\rangle$ *dacă* $i_1 > i_2$

$\langle i_1 \texttt{ =< } i_2\ ,\ \sigma\rangle \to \langle \texttt{true}\ ,\ \sigma\rangle$ *dacă* $i_1 \leq i_2$

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

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):- integer(I),
                                smallstepA(AE1,S,AE2,S).
smallstepB(AE1 =< AE,S,AE2 =< AE,S):-
                                smallstepA(AE1,S,AE2,S).
```

Semantica expresiilor Booleene

- Semantica negației

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

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

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

Prolog

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

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

Semantica compunerii și a blocurilor

- Semantica blocurilor

  $\langle \{\ s\ \}\ ,\ \sigma\rangle \to \langle s\ ,\ \sigma\rangle$
- Semantica compunerii secvențiale

  $\langle \{\}\ \ s_2\ ,\ \sigma\rangle \to \langle s_2\ ,\ \sigma\rangle \qquad \dfrac{\langle s_1\ ,\ \sigma\rangle \to \langle s_1'\ ,\ \sigma'\rangle}{\langle s_1\ s_2\ ,\ \sigma\rangle \to \langle s_1'\ s_2\ ,\ \sigma'\rangle}$

Prolog

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

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

Semantica atribuirii

- Semantica atribuirii

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

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

Prolog

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

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

Semantica lui `if`

- Semantica lui `if`

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

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

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

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) :-
                                    smallstepB(BE1,S,BE2,S) .
```

Semantica lui `while`

- Semantica lui `while`

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

Prolog

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

Semantica programelor

- Semantica programelor

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

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

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).
```

Execuția 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

Execuția programelor: trace

Putem defini o funcție care ne permite să urmărim execuția 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,[]).
```

Execuția 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 .
```

Succes la examen!