Fundamentele limbajelor de programare
CURS 5

Traian Florin Șerbănuță și Andrei Sipoș

Facultatea de Matematică și Informatică, DL Info, Anul II
Semestrul II, 2024/2025

Programare logică

Logică și programare

În anul I, la disciplina „Logică matematică și computațională”, s-au studiat, pe de o parte, logica de ordinul I ca teorie matematică, iar, pe de alta, programarea logică din punct de vedere practic, sub forma limbajului Prolog. Asemănările dintre sintaxa limbajului și logica studiată la cursuri și seminare au fost evidențiate doar la modul intuitiv.

În continuare, vom arăta cum funcționarea acestui limbaj de programare poate fi modelată teoretic, extinzând teoria deja dezvoltată pentru logica de ordinul I. Vom recapitula, dintre lucrurile deja studiate din logică, pe cele care ne vor folosi acum, eventual introducând notații noi.

Logica de ordinul I

Vom fixa patru obiecte $\bot$, $\to$, $\forall$, $=$, diferite două câte două și o mulțime numărabilă (de **variabile logice**)

$$V = \{x_0, x_1, \ldots\},$$

cu $V \cap \{\bot, \to, \forall, =\} = \emptyset$.

Signaturi și structuri

Definim o **signatură de ordinul I** ca fiind un triplet $\sigma = (F, R, r)$ astfel încât $F \cap R = \emptyset$, $(F \cup R) \cap (V \cup \{\bot, \to, \forall, =\}) = \emptyset$ și $r : F \cup R \to \mathbb{N}$.

Dacă $\sigma = (F, R, r)$ este o signatură de ordinul I, atunci numim elementele lui R **simbolurile de relație** ale lui σ, iar elementele lui F **simbolurile de funcție** ale lui σ; pentru orice $s \in F \cup R$, numim $r(s)$ **aritatea** lui s; în particular, acele $s \in F$ pentru care $r(s) = 0$ se numesc **constantele** lui σ. Mai definim și mulțimea

$$S_\sigma := \{\bot, \to, \forall, =\} \cup V \cup F \cup R.$$

Dacă $\sigma = (F, R, r)$ este o signatură de ordinul I, atunci o σ-**structură** va fi o pereche $(A, \{A_s\}_{s \in F \cup R})$, unde $A \neq \emptyset$ (și se va numi **universul**, **mulțimea suport** sau **mulțimea subiacentă** a structurii), pentru orice $s \in F$, $A_s : A^{r(s)} \to A$ și pentru orice $s \in R$, $A_s \subseteq A^{r(s)}$. Structurile vor reprezenta domeniile despre care vor vorbi formulele corespunzătoare signaturilor.

Exemple

Putem obține diverse signaturi dacă punem $\sigma := (F, R, r)$, iar F, R, r sunt, pe rând:

- $F = \{\cdot, 0, 1\}$, $R = \emptyset$, $r(\cdot) = 2$, $r(0) = r(1) = 0$ – atunci σ-structurile vor fi tuplurile $(A, \cdot, 0, 1)$ unde $\cdot : A^2 \to A$, iar 0, $1 \in A$, de exemplu $(\mathbb{Z}, +, 2, 7)$. Observăm că nu impunem în definiția structurii ca ea să respecte anumite legi – acest lucru se va face eventual ulterior, după ce vom defini riguros formulele și satisfacerea lor.
- $F = \emptyset$, $R = \{\leq\}$, $r(\leq) = 2$ – atunci σ-structurile vor fi perechi $(A, \leq)$ unde $\leq$ este o relație binară pe A, de exemplu $(\mathbb{N}, >)$.
- $F = \emptyset$, $R = \{I, A, B\}$, $r(I) = r(A) = r(B) = 1$.

Dacă punem $F = R = \emptyset$, atunci obținem o signatură care, dat fiind că singurele fapte pe care le vom putea exprima peste ea vor folosi semnul $=$, se va numi **signatura egalității**.

Signatura aritmeticii

De asemenea, vom defini **signatura aritmeticii** ca fiind

$$\sigma_{\mathrm{ar}} := (\{\dot{+}, \dot{\times}, \dot{S}, \dot{0}\}, \{\dot{<}\}, r),$$

unde $r(\dot{+}) = r(\dot{\times}) = r(\dot{<}) = 2$, $r(\dot{S}) = 1$, iar $r(\dot{0}) = 0$. Dacă definim funcția $S : \mathbb{N} \to \mathbb{N}$, pentru orice $n \in \mathbb{N}$, prin $S(n) := n + 1$, iar

$$\mathcal{N} := (\mathbb{N}, (N_s)_{s \in F \cup R}),$$

unde $N_{\dot{+}} = +$, $N_{\dot{\times}} = \cdot$, $N_{\dot{S}} = S$, $N_{\dot{0}} = 0$, iar $N_{\dot{<}} = <$, avem că $\mathcal{N}$ este o σ_{ar}-structură.

De remarcat că există și alte σ_{ar}-structuri, de exemplu putem înzestra mulțimea $\{0, 1\}$ cu $\wedge$, $\vee$, $\neg$, 0 și $<$ pentru a obține o altă σ_{ar}-structură.

Termeni

Formulele corespunzătoare unei signaturi $\sigma = (F, R, r)$ vor fi cuvinte peste alfabetul S_σ. Înainte de a defini formulele, va trebui să definim **termenii**. Ei se definesc ca fiind cea mai mică mulțime A de șiruri de simboluri din S_σ astfel încât:

- $V \subseteq A$;
- pentru orice $s \in F$ și orice $t_1, \ldots, t_{r(s)} \in A$, avem $st_1 \ldots t_{r(s)} \in A$ (în particular, dacă $r(s) = 0$, $s \in A$).

Mulțimea termenilor peste σ se va nota cu T_σ.

Inducție și recursie pe termeni

Principiul inducției pe termeni

Fie $B \subseteq T_\sigma$ astfel încât:

- $V \subseteq B$;
- pentru orice $s \in F$ și orice $t_1, \ldots, t_{r(s)} \in B$, avem $st_1 \ldots t_{r(s)} \in B$.

Atunci $B = T_\sigma$.

Principiul recursiei pe termeni

Fie A o mulțime și $G_0 : V \to A$, iar pentru orice $s \in F$, $G_s : A^{r(s)} \to A$. Atunci există și este unică $F : T_\sigma \to A$ astfel încât:

- pentru orice $x \in V$, $F(x) = G_0(x)$;
- pentru orice $s \in F$ și orice $t_1, \ldots, t_{r(s)} \in T_\sigma$, $F(st_1 \ldots t_{r(s)}) = G_s(F(t_1), \ldots, F(t_{r(s)}))$.

Mulțimea variabilelor unui termen

Ca exemplu, putem defini recursiv mulțimea variabilelor unui termen. Practic, definim funcția $Var : T_\sigma \to \mathcal{P}(V)$, prin:

- pentru orice $x \in V$, $Var(x) := \{x\}$;
- pentru orice $s \in F$ și orice $t_1, \ldots, t_{r(s)} \in T_\sigma$, $Var(st_1 \ldots t_{r(s)}) := Var(t_1) \cup \ldots \cup Var(t_{r(s)})$ (în particular, dacă $r(s) = 0$, $Var(s) = \emptyset$).

Notăm cu $\widetilde{T}_\sigma$ mulțimea termenilor fără variabile.

Evaluarea termenilor

Fie $\mathcal{A} = (A, (A_s)_{s \in F \cup R})$ o σ-structură. Atunci pentru orice $v : V \to A$ există și este unică o funcție $(\cdot)^{\mathcal{A}}_v : T_\sigma \to A$ astfel încât

- pentru orice $x \in V$, $x^{\mathcal{A}}_v = v(x)$;
- pentru orice $s \in F$ și orice $t_1, \ldots, t_{r(s)} \in T_\sigma$, $(st_1 \ldots t_{r(s)})^{\mathcal{A}}_v = A_s((t_1)^{\mathcal{A}}_v, \ldots, (t_{r(s)})^{\mathcal{A}}_v)$ (în particular, dacă $r(s) = 0$, $s^{\mathcal{A}}_v = A_s$).

Atenție, din nou: funcția v din definiția de mai sus **nu** are un rol similar cu cel al funcțiilor $e : Q \to 2$ din logica propozițională, cu toate că ar putea părea astfel.

Formule

Fie $\sigma = (F, R, r)$ o signatură. Numim **formulă atomică** peste σ un șir de forma $= tu$, cu t, $u \in T_\sigma$ (numită formulă atomică **ecuațională** sau doar **ecuație**) sau un șir de forma $st_1 \dots t_n$ cu $s \in R$, $n = r(s)$ și $t_1, \dots, t_n \in T_\sigma$ (numită formulă atomică **relațională**). Mulțimea formulelor atomice peste σ se va nota cu Fa_σ. Mulțimea **formulelor** peste σ se definește ca fiind cea mai mică mulțime A de șiruri de simboluri din S_σ astfel încât:

- formulele atomice aparțin lui A;
- $\bot \in A$;
- dacă φ, $\psi \in A$, atunci $\to \varphi\psi \in A$;
- dacă $\varphi \in A$ și $x \in V$, atunci $\forall x\varphi \in A$.

Mulțimea formulelor peste σ se va nota cu F_σ.

De asemenea, vom defini **formulele relaționale** în același mod, cu excepția că acceptăm în cadrul lor doar formule atomice relaționale.

Notație infixată și conectori derivați

Vom folosi aceeași metodă pe care am folosit-o la logica propozițională pentru a putea nota formulele infixat – pentru orice $\varphi, \psi \in F_\sigma$, $\varphi \to \psi$ în loc de $\to \varphi\psi$, dar și, pentru orice $t, u \in T_\sigma$, $t = u$ în loc de $= tu$.

De asemenea, vom nota, ca mai înainte, pentru orice $\varphi, \psi \in F_\sigma$,

$$\top := \bot \to \bot, \quad \neg\varphi := \varphi \to \bot, \quad \varphi \wedge \psi := \neg(\varphi \to \neg\psi),$$

$$\varphi \vee \psi := (\neg\varphi) \to \psi, \quad \varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi).$$

dar și, pentru orice $x \in V$ și $\varphi \in F_\sigma$,

$$\exists x\varphi := \neg\forall x\neg\varphi.$$

Inducție pe formule

Principiul inducției pe formule

Fie $B \subseteq F_\sigma$ astfel încât:

- formulele atomice aparțin lui B;
- $\bot \in B$;
- dacă $\varphi, \psi \in B$, atunci $\varphi \to \psi \in B$;
- dacă $\varphi \in B$ și $x \in V$, atunci $\forall x \varphi \in B$.

Atunci $B = F_\sigma$.

Recursie pe formule

Principiul recursiei pe formule

Fie A o mulțime și $G_0 : Fa_\sigma \to A$, $G_\perp \in A$, $G_\to : A^2 \to A$, $G_\forall : V \times A \to A$. Atunci există și este unică $F : F_\sigma \to A$ astfel încât:

- pentru orice $\varphi \in Fa_\sigma$, $F(\varphi) = G_0(\varphi)$;
- $F(\perp) = G_\perp$;
- pentru orice $\varphi, \psi \in F_\sigma$, $F(\varphi \to \psi) = G_\to(F(\varphi), F(\psi))$;
- pentru orice $\varphi \in F_\sigma$ și $x \in V$, $F(\forall x \varphi) = G_\forall(x, F(\varphi))$.

Mulțimea variabilelor libere ale unei formule

Ca exemplu, putem defini recursiv mulțimea variabilelor **libere** ale unei formule. Definim funcția $FV : F_\sigma \to \mathcal{P}(V)$, prin:

- pentru orice t, $u \in T_\sigma$, $FV(t = u) := Var(t) \cup Var(u)$;
- pentru orice $s \in R$ și orice $t_1, \ldots, t_{r(s)} \in T_\sigma$, $FV(st_1 \ldots t_{r(s)}) := Var(t_1) \cup \ldots \cup Var(t_{r(s)})$;
- $FV(\bot) := \emptyset$;
- pentru orice φ, $\psi \in F_\sigma$, $FV(\varphi \to \psi) := FV(\varphi) \cup FV(\psi)$;
- pentru orice $\varphi \in F_\sigma$ și $x \in V$, $FV(\forall x\varphi) := FV(\varphi) \setminus \{x\}$.

Dacă $\varphi \in F_\sigma$ cu $FV(\varphi) = \emptyset$, atunci φ se numește **enunț**.
Mulțimea enunțurilor peste σ se notează cu E_σ.

Spre evaluarea formulelor

Fie $\mathcal{A} = (A, (A_s)_{s \in F \cup R})$ o σ-structură. Pentru orice $v : V \to A$, $x \in V$, $a \in A$, definim $v_{x \leftarrow a} : V \to A$, pentru orice $y \in V$, prin

$$v_{x \leftarrow a}(y) := \begin{cases} v(y), & \text{dacă } y \neq x, \\ a, & \text{dacă } y = x. \end{cases}$$

Observăm că pentru orice variabile x, y cu $x \neq y$, orice $v : V \to A$ și orice a, $b \in A$, avem că

$$(v_{y \leftarrow b})_{x \leftarrow a} = (v_{x \leftarrow a})_{y \leftarrow b}.$$

În acest caz, notăm valoarea lor comună cu $v_{x \leftarrow a, y \leftarrow b}$. Așadar, pentru orice $z \in V$,

$$v_{x \leftarrow a, y \leftarrow b}(z) = \begin{cases} v(z) & \text{dacă } z \neq x \text{ și } z \neq y, \\ a & \text{dacă } z = x, \\ b & \text{dacă } z = y. \end{cases}$$

Evaluarea formulelor

Avem că există și este unică o funcție $\|\cdot\|^{\mathcal{A}} : F_\sigma \to \{0,1\}^{A^V}$ astfel încât, pentru orice $v : V \to A$, avem:

- pentru orice t, $u \in T_\sigma$, $\|t = u\|_v^{\mathcal{A}} = 1 \Leftrightarrow t_v^{\mathcal{A}} = u_v^{\mathcal{A}}$;
- pentru orice $s \in R$ și orice $t_1, \ldots, t_{r(s)} \in T_\sigma$, $\|st_1 \ldots t_{r(s)}\|_v^{\mathcal{A}} = 1 \Leftrightarrow ((t_1)_v^{\mathcal{A}}, \ldots, (t_{r(s)})_v^{\mathcal{A}}) \in A_s$;
- $\|\bot\|_v^{\mathcal{A}} = \bot = 0$;
- pentru orice φ, $\psi \in F_\sigma$, $\|\varphi \to \psi\|_v^{\mathcal{A}} = \|\varphi\|_v^{\mathcal{A}} \to \|\psi\|_v^{\mathcal{A}}$;
- pentru orice $\varphi \in F_\sigma$ și $x \in V$, $\|\forall x\varphi\|_v^{\mathcal{A}} = 1 \Leftrightarrow$ pentru orice $a \in A$, $\|\varphi\|_{v_{x \leftarrow a}}^{\mathcal{A}} = 1$.

Dacă $\varphi \in F_\sigma$ este astfel încât pentru orice $\mathcal{A}$ și v, $\|\varphi\|_v^{\mathcal{A}} = 1$, spunem că φ este **validă**.

Evaluarea conectorilor derivați

Fie $\mathcal{A} = (A, (A_s)_{s \in F \cup R})$ o σ-structură. Este acum imediat că pentru orice $v : V \to A$, avem:

- $\|\top\|_v^{\mathcal{A}} = \mathsf{T} = 1$;
- pentru orice $\varphi, \psi \in F_\sigma$, $\|\varphi \wedge \psi\|_v^{\mathcal{A}} = \|\varphi\|_v^{\mathcal{A}} \wedge \|\psi\|_v^{\mathcal{A}}$;
- pentru orice $\varphi, \psi \in F_\sigma$, $\|\varphi \vee \psi\|_v^{\mathcal{A}} = \|\varphi\|_v^{\mathcal{A}} \vee \|\psi\|_v^{\mathcal{A}}$;
- pentru orice $\varphi, \psi \in F_\sigma$, $\|\varphi \leftrightarrow \psi\|_v^{\mathcal{A}} = \|\varphi\|_v^{\mathcal{A}} \leftrightarrow \|\psi\|_v^{\mathcal{A}}$;
- pentru orice $\varphi \in F_\sigma$ și $x \in V$,
  $\|\exists x \varphi\|_v^{\mathcal{A}} = 1 \Leftrightarrow$ există $a \in A$ cu $\|\varphi\|_{v_{x \leftarrow a}}^{\mathcal{A}} = 1$.

Evaluarea enunțurilor

Așadar, dacă $\mathcal{A} = (A, (A_s)_{s \in F \cup R})$ este o σ-structură și φ este enunț, pentru orice v_1, $v_2 : V \to A$, avem $\|\varphi\|^{\mathcal{A}}_{v_1} = \|\varphi\|^{\mathcal{A}}_{v_2}$, deci este echivalent faptul că pentru orice $v : V \to A$, $\|\varphi\|^{\mathcal{A}}_{v} = 1$ cu faptul că există $v : V \to A$ cu $\|\varphi\|^{\mathcal{A}}_{v} = 1$. Numim această stare de fapt cu

$$\mathcal{A} \models \varphi$$

și spunem că $\mathcal{A}$ **satisface** φ sau că $\mathcal{A}$ este **model** pentru φ.

Iată, deci, că structurile de ordinul I reprezintă analogul funcțiilor de evaluare întâlnite în logica propozițională. De aici provine și numele acelei ramuri a logicii care studiază structurile de ordinul I în conjuncție cu enunțurile satisfăcute de ele: **teoria modelelor**.

Semnul $\models$ în logica de ordinul I

Vom defini următoarele concepte, precum și noi semnificații ale semnului $\models$, prin analogie cu logica propozițională:

- Dacă $\varphi \in E_\sigma$ și φ este valid, vom scrie $\models \varphi$.
- Spunem că $\varphi \in E_\sigma$ este **satisfiabil** dacă există $\mathcal{A}$ cu $\mathcal{A} \models \varphi$.
- Spunem că un enunț φ este **nesatisfiabil** dacă φ nu este satisfiabil.
- Fie φ, $\psi \in E_\sigma$. Spunem că din φ **se deduce semantic** ψ și scriem $\varphi \models \psi$ dacă pentru orice $\mathcal{A}$ cu $\mathcal{A} \models \varphi$ avem $\mathcal{A} \models \psi$.

Clar, $\bot$ este enunț, iar pentru orice σ-structură $\mathcal{A}$, avem $\mathcal{A} \not\models \bot$, i.e. $\bot$ este nesatisfiabil.

Mulțimi de enunțuri

Complet analog celor din logica propozițională, vom introduce noțiuni de satisfiabilitate pentru mulțimi de formule, precum și semnificații corespunzătoare ale semnului $\models$.

Fie $\Gamma \subseteq E_\sigma$. Pentru orice σ-structură $\mathcal{A}$, spunem că $\mathcal{A}$ **satisface** Γ sau că $\mathcal{A}$ este **model** pentru Γ, și scriem $\mathcal{A} \models \Gamma$, dacă pentru orice $\varphi \in \Gamma$, $\mathcal{A} \models \varphi$.

Spunem că Γ este **satisfiabilă** dacă există o σ-structură $\mathcal{A}$ cu $\mathcal{A} \models \Gamma$; că este **nesatisfiabilă** dacă nu este satisfiabilă.

Proprietăți

Următoarele proprietăți se demonstrează perfect analog celor din logica propozițională.

Proprietăți

Fie $\Gamma \subseteq E_\sigma$, $\Delta \subseteq \Gamma$ și $\mathcal{A}$ o σ-structură. Avem următoarele:

- Dacă $\mathcal{A} \models \Gamma$, atunci $\mathcal{A} \models \Delta$.
- Dacă Δ este nesatisfiabilă, atunci Γ este nesatisfiabilă.
- Avem că $\mathcal{A} \models \Gamma$ dacă și numai dacă pentru orice $\Sigma \subseteq \Gamma$ finită, $\mathcal{A} \models \Sigma$.

Deducție semantică din mulțimi

Fie $\Gamma \subseteq E_\sigma$ și $\varphi \in E_\sigma$. Spunem că din Γ **se deduce semantic** φ, și scriem $\Gamma \models \varphi$, dacă pentru orice σ-structură $\mathcal{A}$ cu $\mathcal{A} \models \Gamma$ avem $\mathcal{A} \models \varphi$. Această noțiune are următoarele proprietăți analoage celor din logica propozițională și demonstrabile similar.

Proprietăți

Fie $\Gamma \subseteq E_\sigma$, $\Delta \subseteq \Gamma$ și $\varphi, \psi \in E_\sigma$. Avem următoarele:

- Dacă $\Delta \models \varphi$, atunci $\Gamma \models \varphi$.
- Mulțimea Γ este nesatisfiabilă dacă și numai dacă $\Gamma \models \bot$.
- Avem $\Gamma \models \varphi$ dacă și numai dacă $\Gamma \cup \{\neg\varphi\}$ este nesatisfiabilă.

Substituția termenilor

Vom numi **substituție** o funcție $\theta : V \to T_\sigma$. Folosind Principiul de recursie pe termeni, ea se extinde natural în mod unic la o funcție $\widetilde{\theta} : T_\sigma \to T_\sigma$.

Se observă, folosind unicitatea, că, pentru orice două substituții θ, θ',

$$\widetilde{\widetilde{\theta'} \circ \theta} = \widetilde{\theta'} \circ \widetilde{\theta}.$$

Clauze

Dacă avem o formulă φ, $n \in \mathbb{N}$ și $k_1 < \ldots < k_n$ cu $FV(\varphi) = \{x_{k_1}, \ldots, x_{k_n}\}$, notăm cu $\forall\varphi$ enunțul $\forall x_{k_1} \ldots \forall x_{k_n} \varphi$.

Numim **literal** o formulă de forma φ sau $\neg\varphi$, unde φ este o formulă atomică relațională (literalul este, respectiv, **pozitiv** sau **negativ**). Numim **clauză** o formulă de forma $\forall(L_1 \vee \ldots \vee L_m)$, unde $L_1, \ldots, L_m$ sunt literali. Numim **clauză definită** o clauză unde apare exact un literal pozitiv, anume pe prima poziție. Dacă $A_0, \ldots A_m$ sunt formule atomice relaționale, atunci clauza

$$\forall(A_0 \vee \neg A_1 \vee \ldots \vee \neg A_m)$$

se scrie și sub forma (cunoscută din limbajul Prolog)

$$A_0 \leftarrow A_1, \ldots, A_m.$$

Un **program** va fi o mulțime finită de clauze definite. Un **scop definit** este o clauză în care apar doar literali negativi.

Structuri Herbrand

De acum încolo, vom face presupunerea că există măcar o constantă în signatură, așadar $\widetilde{T}_\sigma \neq \emptyset$. Notăm cu B_σ (numită **baza Herbrand**) mulțimea tuturor σ-formulelor atomice relaționale fără variabile.

Spunem că o σ-structură este **Herbrand** atunci când universul ei este $\widetilde{T}_\sigma$, iar simbolurile de funcție sunt interpretate „de ele însele". Observăm că o σ-structură Herbrand este complet determinată de mulțimea J a acelor formule din B_σ adevărate în ea. Pentru orice submulțime J a lui B_σ și orice enunț φ, spunem că $J \models_H \varphi$ atunci când structura Herbrand asociată lui J satisface φ.

Dacă $\mathcal{A}$ este o σ-structură, vom nota $J_\mathcal{A} := \{\varphi \in B_\sigma \mid \mathcal{A} \models \varphi\}$.

Proprietăți

Teoremă

Fie $\mathcal{A}$ o σ-structură și φ o clauză definită (sau un scop definit) cu $\mathcal{A} \models \varphi$. Atunci $J_{\mathcal{A}} \models_H \varphi$.

Demonstrație

Presupunem φ clauză definită. Avem că există m, $n \in \mathbb{N}$, formule atomice relaționale A_0, $A_1, \ldots, A_m$ și variabile $x_1, \ldots, x_n$ cu

$$\varphi = \forall x_1 \ldots \forall x_n (A_0 \vee \neg A_1 \vee \ldots \vee \neg A_m).$$

Fie $t_1, \ldots, t_n \in \widetilde{T}_\sigma$. Notăm, pentru orice i, $A'_i := A_i[x_1 := t_1] \ldots [x_n := t_n]$ și $\varphi' := A'_0 \vee \neg A'_1 \vee \ldots \vee \neg A'_m$. Vrem $J_{\mathcal{A}} \models_H \varphi'$. Presupunem că, pentru orice $i \geq 1$, $J_{\mathcal{A}} \models_H A'_i$. Arătăm că $J_{\mathcal{A}} \models_H A'_0$. Deci, pentru orice $i \geq 1$, cum $A'_i \in B_\sigma$, $A'_i \in J_{\mathcal{A}}$, deci $\mathcal{A} \models A'_i$. Cum $\mathcal{A} \models \varphi$, avem $\mathcal{A} \models \varphi'$, iar, cum, pentru orice $i \geq 1$, $\mathcal{A} \models A'_i$, avem $\mathcal{A} \models A'_0$. Cum $A'_0 \in B_\sigma$, $A'_0 \in J_{\mathcal{A}}$, deci $J_{\mathcal{A}} \models_H A'_0$.

Proprietăți

Teoremă

Fie P un program. Atunci $K_P := \{J \subseteq B_\sigma \mid J \models_H P\}$ este o mulțime Moore pe B_σ.

Demonstrație

Faptul că $B_\sigma \models_H P$ rămâne ca exercițiu. Fie acum $K \subseteq K_P$ cu $K \neq \emptyset$. Vrem $\bigcap K \models_H P$. Fie $\varphi \in P$. Fie $t_1, \ldots, t_n \in \widetilde{T}_\sigma$. Folosim aceleași notații ca în demonstrația precedentă. Presupunem că, pentru orice $i \geq 1$, $\bigcap K \models_H A'_i$. Arătăm că $\bigcap K \models_H A'_0$. Deci, pentru orice $i \geq 1$, cum $A'_i \in B_\sigma$, $A'_i \in \bigcap K$, deci, pentru orice $J \in K$, $A'_i \in J$, adică $J \models_H A'_i$. Fie $J \in K$. Avem $J \models_H \varphi$, deci $J \models_H \varphi'$, iar, cum, pentru orice $i \geq 1$, $J \models_H A'_i$, avem $J \models_H A'_0$, deci, cum $A'_0 \in B_\sigma$, $A'_0 \in J$. Așadar, $A'_0 \in \bigcap K$, deci $\bigcap K \models_H A'_0$.

Pentru orice program P, notăm $M_P := \bigcap K_P \in K_P$.

Proprietăți

Teoremă

Pentru orice program P, $M_P = \{\varphi \in B_\sigma \mid P \models \varphi\}$.

Demonstrație

Pentru „$\supseteq$”, fie $\varphi \in B_\sigma$ cu $P \models \varphi$. Fie $J \in K_P$. Vrem $\varphi \in J$. Cum $J \models_H P$, avem $J \models_H \varphi$, iar, cum $\varphi \in B_\sigma$, $\varphi \in J$.

Pentru „$\subseteq$”, fie $\varphi \in M_P$. Fie $\mathcal{A}$ cu $\mathcal{A} \models P$. Vrem $\mathcal{A} \models \varphi$. Cum P este o mulțime de clauze definite, dintr-o teoremă anterioară avem $J_\mathcal{A} \models_H P$, deci $J_\mathcal{A} \in K_P$. Rezultă $M_P \subseteq J_\mathcal{A}$, deci $\varphi \in J_\mathcal{A}$, adică $J_\mathcal{A} \models_H \varphi$. Rezultă $\mathcal{A} \models \varphi$.

Motivația unificării

Să zicem că avem un program Prolog care conține regula

$$p(f(x), y) \leftarrow p(x, y)$$

și vrem să interogăm $p(z, f(t))$. Pentru a găsi o soluție a acestei interogări folosind regula de mai sus, trebuie să substituim $x \mapsto x$, $z \mapsto f(x)$, $y \mapsto f(t)$, $t \mapsto t$, interogarea devenind $p(f(x), f(t))$, care este redusă, conform regulii, la $p(x, f(t))$, care devine practic o nouă interogare.

Observăm că o parte esențială a procesului de rulare a unui program Prolog (pe care îl vom studia data viitoare, când vom vedea exact care este legătura dintre regulile/interogările afișate mai sus și teoria prezentată astăzi) este găsirea acelei substituții. Acest procedeu se va numi **unificare**.

Unificatori

Dacă $\mathcal{E}$ este o mulțime de ecuații, numim **unificator** pentru ea o substituție θ astfel încât, pentru orice $(s = t) \in \mathcal{E}$, avem $\widetilde{\theta}(s) = \widetilde{\theta}(t)$; el se numește **cel mai general unificator** (**cgu**; **most general unifier**, **mgu**) dacă, pentru orice unificator θ' pentru $\mathcal{E}$, avem că există o substituție ω cu $\theta' = \widetilde{\omega} \circ \theta$.

Propoziție

Fie $n \in \mathbb{N}$, $z_1, \ldots, z_n \in V$ (diferite două câte două) și $t_1, \ldots, t_n \in T_\sigma$ astfel încât, pentru orice i, j, $z_i \notin Var(t_j)$. Atunci substituția θ care duce, pentru orice i, z_i în t_i este cgu pentru $\{z_i = t_i \mid i \in \{1, \ldots, n\}\}$.

Demonstrație

Faptul că este unificator este evident. Fie acum θ' un unificator. Vrem ω cu $\theta' = \widetilde{\omega} \circ \theta$. Luăm $\omega := \theta'$. Avem, într-adevăr, că, pentru orice i, $(\widetilde{\theta'} \circ \theta)(z_i) = \widetilde{\theta'}(t_i) = \widetilde{\theta'}(z_i) = \theta'(z_i)$ și, pentru orice altă variabilă y, $(\widetilde{\theta'} \circ \theta)(y) = \widetilde{\theta'}(y) = \theta'(y)$.

Algoritm de unificare

Vom descrie acum un algoritm de unificare. El are ca intrare o mulțime de ecuații $\mathcal{E}$, iar la ieșire, un cgu dat de o mulțime de ecuații ca în propoziția precedentă, sau „eșec", dacă un cgu nu există. Algoritmul trece printr-un ciclu care, la fiecare pas, verifică dacă există vreo ecuație în mulțime căruia i se poate aplica vreuna dintre următoarele operații – în caz contrar, el se oprește:

- o ecuație de forma $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ este înlocuită de ecuațiile $s_1 = t_1, \ldots, s_n = t_n$;
- o ecuație de forma $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)$, unde $f \neq g$, produce „eșec";
- o ecuație de forma $x = x$, cu $x \in V$, este eliminată;
- o ecuație de forma $t = x$ cu $x \in V$ și $t \notin V$ este înlocuită cu $x = t$;
- o ecuație de forma $x = t$ cu $x \in V$, $t \neq x$ și $x \in Var(t)$ produce „eșec";
- o ecuație de forma $x = t$ cu $x \in V$, $t \neq x$, $x \notin Var(t)$, dar cu x apărând și în alte ecuații din mulțime, face ca toate celelalte sale apariții să fie substituite cu t.

Corectitudine

Teoremă

Algoritmul de unificare prezentat se termină mereu și produce rezultatul corect.

Demonstrație

Pentru terminare, observăm că doar ultimul pas crește numărul de simboluri, dar acela poate fi aplicat doar o dată pentru fiecare variabilă.

Pentru corectitudine, observăm întâi că mulțimile care produc „eșec" nu au unificator și că, atunci când algoritmul se termină, mulțimea are exact forma din propoziția precedentă. Apoi, trebuie să arătăm că la fiecare pas, mulțimea rezultantă are aceiași unificatori ca cea precedentă. Tratăm ultimul caz doar. Notând cu ψ substituția $x \mapsto t$, avem că ecuațiile $s = u$ care nu sunt $x = t$ sunt înlocuite de $\widetilde{\psi}(s) = \widetilde{\psi}(u)$.

Demonstrație (cont.)

Fie θ un unificator pentru mulțimea inițială, în particular $\theta(x) = \widetilde{\theta}(t)$. Fie $s = u$ o ecuație din ea care nu este $x = t$. Vrem $\widetilde{\theta}(\widetilde{\psi}(s)) = \widetilde{\theta}(\widetilde{\psi}(u))$. Se arată ușor că $\widetilde{\theta} \circ \psi = \theta$, de unde rezultă concluzia.

Fie acum θ un unificator pentru mulțimea rezultantă, în particular (din nou) $\theta(x) = \widetilde{\theta}(t)$. Se arată, din nou, ușor, că $\widetilde{\theta} \circ \psi = \theta$, iar concluzia rezultă analog, scriind egalitățile invers.