Fundamentele limbajelor de programare

Programare Logică. Corectitudinea algoritmului de unificare.

Traian Florin Șerbănuță și Andrei Sipoș

Facultatea de Matematică și Informatică, DL Info

Anul II, Semestrul II, 2024/2025

Secțiunea 1

Programare Logică (de ordinul I)

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 de ordinul I

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:

- **simbolurile de relație** ale lui σ – elementele lui R
- **simbolurile de funcție** ale lui σ – elementele lui F
- $r(s)$ **aritatea** lui s – pentru orice $s \in F \cup R$
- **constantele** lui σ – acele $f \in F$ pentru care $r(f) = 0$

Termeni

Termenii sunt cuvinte peste alfabetul

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

definite prin următoarele două (scheme de) reguli:

- v e termen, pentru orice $v \in V$
- dacă $t_1, \ldots, t_{r(f)}$ sunt termeni,
  atunci $f\ t_1 \ldots t_{r(f)}$ e termen
  pentru orice $f \in F$.
  (în particular, dacă $r(f) = 0$, atunci f este termen)

Mulțimea termenilor peste σ se va nota cu T_σ.

Convenții de scriere

Presupunând că V, F, R nu conțin simbolurile (,), și ,
putem scrie $f(t_1, \ldots, t_{r(f)})$ în loc de $f\ t_1 \cdots t_{r(s)}$ dacă $r(f) > 0$.

Mulțimea variabilelor unui termen

$Var : T_\sigma \to \mathcal{P}(V)$, definită prin:

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

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

Formule atomice

Fie $\sigma = (F, R, r)$ o signatură.

- O formulă atomică **ecuațională** sau doar **ecuație** e un șir de forma $=tu$, cu $t, u \in T_\sigma$
- O formulă atomică **relațională** e un șir de forma $p\, t_1 \cdots t_{r(p)}$ cu $p \in R$ și $t_1, \ldots, t_n \in T_\sigma$
- O **formulă atomică** este fie o ecuație fie o formulă atomică relațională

Mulțimea formulelor atomice peste σ se va nota cu Fa_σ.

Convenții de scriere

- Vom scrie $t = u$ în loc de $=tu$
- Presupunând că V, F, R nu conțin simbolurile (,), și ,
  putem scrie $p(t_1, \ldots, t_{r(p)})$ în loc de $p\, t_1 \cdots t_{r(p)}$ dacă $r(p) > 0$.

Formule

Mulțimea **formulelor** peste σ e definită de următoarele (scheme de) reguli:

- a este formulă, pentru orice formulă atomică $a \in Fa_\sigma$;
- $\bot$ este formulă;
- dacă φ, ψ sunt formula, atunci $\to \varphi\psi$ este formulă;
- dacă φ formulă, atunci $\forall x\varphi \in A$, pentru orice $x \in V$.

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.

Convenții de scriere

Vom scrie $\varphi \to \psi$ în loc de $\to \varphi\psi$ și vom folosi parenteze pentru dezambiguizare.

Conectori logici derivați

Pentru orice $\varphi,\psi \in F_\sigma$

- $\top := \bot \to \bot$
- $\neg\varphi := \varphi \to \bot$
- $\varphi \wedge \psi := \neg(\varphi \to \neg\psi)$
- $\varphi \vee \psi := (\neg\varphi) \to \psi$
- $\varphi \leftrightarrow \psi := (\varphi \to \psi) \wedge (\psi \to \varphi)$
- $\exists x\varphi := \neg\forall x\neg\varphi$, pentru orice $x \in V$ și $\varphi \in F_\sigma$.

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 $p \in R$ și orice $t_1, \ldots, t_{r(p)} \in T_\sigma$, $FV(p\, t_1 \cdots t_{r(p)}) := Var(t_1) \cup \ldots \cup Var(t_{r(p)})$;
- $FV(\bot) := \emptyset$;
- 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.

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.

Secțiunea 2

Unificare

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 este găsirea acelei substituții.

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',

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

Pentru o variabilă x și un termen t definim substituția singleton $[x \mapsto t]$ prin

$$[x \mapsto t](y) = \begin{cases} t, & \text{dacă } y = x, \\ y, & \text{altfel.} \end{cases}$$

Spunem că o substituție Θ e **mai generală** decât altă substituție Θ' dacă există o substituție Δ astfel încât $\Theta' = \widetilde{\Delta} \circ \Theta$

- Notăm cu $\Theta \succ \Theta'$ faptul că Θ e mai generală decât Θ'

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 $\widetilde{\theta}(s) = \widetilde{\theta}(t)$

Notăm cu $\mathcal{U}(\mathcal{E})$ mulțimea unificatorilor lui $\mathcal{E}$

Un unificator θ pentru $\mathcal{E}$ se numește **cel mai general unificator** (**cgu**; **most general unifier**, **mgu**) dacă este mai general (ca substituție) decât orice alt unificator θ' pentru $\mathcal{E}$.

Formal Θ e mgu pentru $\mathcal{E}$ ddacă

- $\Theta \in \mathcal{U}(\mathcal{E})$ și
- pentru orice $\Theta' \in \mathcal{U}(\mathcal{E})$, $\Theta \succ \Theta'$

Propoziție (definiție alternativă)

Θ e mgu pentru $\mathcal{E}$ ddacă

$$\mathcal{U}(\mathcal{E}) = \{\Theta' : V \to T_\sigma \mid \Theta \succ \Theta'\}$$

Algoritm de unificare (ușor formalizat)

Algoritmul pornește cu configurația inițială $(\Theta, \mathcal{R}) = (1_V, \mathcal{E})$ și constă în aplicarea nedeterministă a regulilor de mai jos[1]:

SCOATE $(\Theta, \mathcal{R} \cup \{t = t\}) \Longrightarrow (\Theta, \mathcal{R})$

DESCOMPUNE $(\Theta, \mathcal{R} \cup \{f\ t_1 \cdots t_n = f\ t'_1 \cdots t'_n\})$
$\Longrightarrow (\Theta, \mathcal{R} \cup \{t_1 = t'_1, \ldots, t_n = t'_n\}$

REZOLVĂ $(\Theta, \mathcal{R} \cup \{x = t\})$ **sau** $(\Theta, \mathcal{R} \cup \{t = x\})$
$\Longrightarrow ([x \mapsto t] \circ \Theta, \{[x \mapsto t](t_1) = [x \mapsto t](t_2) \mid t_1 = t_2 \in \mathcal{R}\}$
dacă $x \notin Var(t)$.

EȘEC (conflict) există în $\mathcal{R}$ o ecuație de forma
$f\ s_1 \cdots s_n = g\ t_1 \cdots t_m$ cu $f \neq g$.

EȘEC (ciclu) există în $\mathcal{R}$ o ecuație de forma
$x = t$ sau $t = x$ cu $t \neq x$ și $x \in Var(t)$.

[1] Condiție suplimentară pentru orice regulă $(\Theta, \mathcal{R} \cup \{e\}) \Longrightarrow (\Theta', \mathcal{R}')$: $e \notin \mathcal{R}$.

Accesibilitate

Definiție: Configurația $(\Theta', \mathcal{R}')$ e **accesibilă** din configurația $(\Theta, \mathcal{R})$ dacă există $n \in \mathbb{N}$ și o secvență $(\Theta_i, \mathcal{R}_i)_{i \in \overline{1.n}}$, numită **derivare** astfel încât:

- $(\Theta_0, \mathcal{R}_0) = (\Theta, \mathcal{R})$
- $(\Theta_n, \mathcal{R}_n) = (\Theta', \mathcal{R}')$
- $(\Theta_i, \mathcal{R}_i) \Longrightarrow (\Theta_{i+1}, \mathcal{R}_{i+1})$ pentru orice $0 \leq i < n$.

Invariant

Teoremă

Pentru orice configurație $(\Theta, \mathcal{R})$ accesibilă dintr-o configurație $(1_V, \mathcal{E})$:

$$\mathcal{U}(\mathcal{E}) = \{\Theta' \in \mathcal{U}(\mathcal{R}) \mid \Theta \succ \Theta'\}$$

Demonstrație (schiță)

Demonstrăm concluzia teoremei împreună cu următoarele afirmații

- $\forall x \in V.\forall y \in Var(\Theta(x)).\Theta(y) = y$
- $\forall x \in Var(\mathcal{R}).\Theta(x) = x$
  (unde $Var(\mathcal{R}) = \bigcup_{(t_1=t_2)\in\mathcal{R}}(Var(t_1) \cup Var(t_2))$)

prin inducție după lungimea unei derivări.

Lemă ajutătoare

Fie Θ unificator pentru $x = t$, unde x variabilă.
Atunci $\Theta = \widetilde{\Theta} \circ [x \mapsto t]$.

Variant

Pentru un termen t definim $n_F(t)$, numărul de simboluri din t recursiv prin:

- $n_F(x) = 0$ pentru orice variabilă x
- $n_F(f\ t_1 \cdots t_{r(f)}) = 1 + n_F(t_1) + \ldots + n_F(t_{r(s)})$

n_F se extinde natural la ecuații și mulțimi de ecuații.

- n_F numără doar simbolurile de funcție (deci nu numără $=$ ca simbol)

Dată fiind o configurație $(\Theta, \mathcal{R})$ definim

$$variant((\Theta, \mathcal{R})) := (|Var(\mathcal{R})|, n_F(\mathcal{R}), |\mathcal{R}|).$$

Pe $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ notăm *ordinea lexicografică* cu $<$

- $(n_1, n_2, n_3) < (m_1, m_2, m_3)$ ddacă
  $n_1 < m_1 \vee (n_1 = m_1 \wedge n_2 < m_2) \vee (n_1 = m_1 \wedge n_2 = m_2 \wedge n_3 < m_3)$
- $<$ este o *bună ordine*, i.e., nu există secvențe infinite descrescătoare:

$$\nexists (a_n)_{n \in \mathbb{N}} \text{ cu } a_n > a_{n+1} \text{ pentru orice } n \in \mathbb{N}$$

Variant

Teoremă

Dacă $(\Theta, \mathcal{R}) \Longrightarrow (\Theta', \mathcal{R}')$ atunci $\mathit{variant}((\Theta, \mathcal{R})) > \mathit{variant}((\Theta', \mathcal{R}'))$

Demonstrație

Discuție după pasul din algoritm folosit:

SCOATE primele două componente nu cresc și ultima sigur scade cu 1

DESCOMPUNE prima componentă rămâne la fel și a doua scade cu 1

REZOLVĂ Prima componentă scade, pentru că:

- variabila care este rezolvată apărea în $\mathcal{R}$ dar nu va mai apare în $\mathcal{R}'$ (e substituită peste tot cu un termen în care nu apare)
- nu apar alte variabile noi

Terminare

Teoremă

Algoritmul de unificare nu admite derivări infinite.

Demonstrație

Presupunem există o derivare infinită $(\Theta_n, \mathcal{R}_n)_{n\in\mathbb{N}}$
Considerăm șirul $(a_n)_{n\in\mathbb{N}}$ definit prin $a_n := \mathit{variant}((\Theta_n, \mathcal{R}_n))$
Deoarece $(\Theta_n, \mathcal{R}_n) \Longrightarrow (\Theta_{n+1}, \mathcal{R}_{n+1})$ pentru orice $n \in \mathbb{N}$,
reiese că $a_n > a_{n+1}$ pentru orice $n \in \mathbb{N}$ (din teorema de variant).
Contradicție cu faptul că $<$ e bună ordine pe $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$.

Configurații finale

O configurație $(\Theta, \mathcal{R})$ se numește **finală** dacă
orice ecuație din $\mathcal{R}$ este "de eșec", i.e., de forma:

- $x = t$ sau $t = x$ unde $t \neq x$ și $x \in Var(t)$
- $f\ t_1 \cdots t_{r(f)} = g\ t'_1 \cdots t'_{r(g)}$ cu $f \neq g$

Teoremă

Dacă $(\Theta, \mathcal{R})$ e finală atunci $\mathcal{U}(\mathcal{R}) = \begin{cases} {T_\sigma}^V & \text{dacă } \mathcal{R} = \emptyset \\ \emptyset & \text{altfel} \end{cases}$

Demonstrație

Dacă $\mathcal{R} = \emptyset$ afirmația e evidentă din definiția unificatorilor[a].
Altfel, prin reducere la absurd pentru $\widetilde{\Theta}(t_1) = \widetilde{\Theta}(t_2)$ unde $(t_1 = t_2) \in \mathcal{R}$

- $\widetilde{\Theta}(x) = \widetilde{\Theta}(t)$ contradicție cu $n_F(\widetilde{\Theta}(x)) < n_F(\widetilde{\Theta}(t))$
- $\widetilde{\Theta}(f\ t_1 \cdots t_{r(f)}) = \widetilde{\Theta}(g\ t'_1 \cdots t'_{r(g)})$ contradicție cu $f \neq g$

[a] ${T_\sigma}^V$ este multimea funcțiilor de la V la T_σ, adică a tuturor substituțiilor.

Corectitudine parțială

Fie $(\Theta, \mathcal{R})$ configurație finală accesibilă din $(1_V, \mathcal{E})$.

- Dacă $\mathcal{R} = \emptyset$ atunci Θ este mgu pentru $\mathcal{E}$.
- Dacă $\mathcal{R} \neq \emptyset$ atunci $\mathcal{U}(\mathcal{E}) = \emptyset$.

Demonstrație

Folosind teorema despre invariant avem că

$$\mathcal{U}(\mathcal{E}) = \{\Theta' \in \mathcal{U}(\mathcal{R}) \mid \Theta \succ \Theta'\}$$

Folosim teorema despre configurații finale:

- dacă $\mathcal{R} = \emptyset$, atunci $\mathcal{U}(\mathcal{R}) = {T_\sigma}^V$, deci
  $\mathcal{U}(\mathcal{E}) = \{\Theta' : V \to T_\sigma \mid \Theta \succ \Theta'\}$
  de unde Θ mgu pentru $\mathcal{E}$.
- dacă $\mathcal{R} \neq \emptyset$, atunci $\mathcal{U}(\mathcal{R}) = \emptyset$, deci $\mathcal{U}(\mathcal{E}) = \emptyset$

Progres

Teoremă

Pentru orice configurație ne-finală $(\Theta, \mathcal{R})$ există o configurație $(\Theta', \mathcal{R}')$ cu

$$(\Theta, \mathcal{R}) \Longrightarrow (\Theta', \mathcal{R}')$$

Demonstrație

Există cel puțin o ecuație $(t_1 = t_2) \in \mathcal{R}$ care nu e "de eșec".
Analiză după forma ei. Avem trei cazuri:

$x = x$ Aplicăm regula ELIMINĂ

$x = t$ sau $t = x$ cu $t \neq x$ Avem că $x \notin Var(t)$ (nu e de eșec).
Aplicăm regula REZOLVĂ

$f\ t_1 \cdots t_{r(f)} = g\ t'_1 \cdots t'_{r(g)}$ Avem că $f = g$ (nu e de eșec).
Aplicăm regula DESCOMPUNE

Corectitudine totală

Teoremă

Algoritmul de unificare se termină mereu și produce rezultatul corect.

Demonstrație

- Terminare: din teorema de terminare.
- Admite configurații finale: din teorema de progres (și terminare).
- Produce rezultatul corect: din teorema de corectitudine parțială.