Programare Logică
Cursurile VI, VII și VIII

Claudia MUREȘAN
cmuresan@fmi.unibuc.ro, c.muresan@yahoo.com

Universitatea din București
Facultatea de Matematică și Informatică
București

2019–2020, Semestrul II

Cuprinsul acestui curs

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor
2. Să exemplificăm noțiunile anterioare într–un program în Prolog
3. Algoritmul de unificare

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor
2. Să exemplificăm noțiunile anterioare într–un program în Prolog
3. Algoritmul de unificare

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor
2. Să exemplificăm noțiunile anterioare într–un program în Prolog
3. Algoritmul de unificare

Ce este un predicat?

- **Predicatele** se mai numesc **propoziții cu variabile**.
- **Exemplu de predicat:** "x este un număr prim" este un predicat cu variabila x; acest predicat nu are o valoare de adevăr; pentru a obține din el un enunț cu valoare de adevăr, trebuie să–i dăm valori variabilei x.
- Variabilei x i se indică un domeniu al valorilor posibile, de exemplu $\mathbb{N}$. Se dorește ca, prin înlocuirea lui x din acest predicat cu o valoare arbitrară din acest domeniu, să se obțină o propoziție adevărată sau falsă.
- Înlocuind în acest predicat pe $x := 2 \in \mathbb{N}$, se obține propoziția adevărată "2 este un număr prim", iar înlocuind $x := 4 \in \mathbb{N}$, se obține propoziția falsă "4 este un număr prim".
- Dacă desemnăm pe $\mathbb{N}$ ca domeniu al valorilor pentru variabila x din predicatul de mai sus, atunci putem să aplicăm un cuantificator acestei variabile, obținând, astfel, tot un enunț cu valoare de adevăr: propoziția $(\forall\, x \in \mathbb{N})\,(x$ este un număr prim$)$ este falsă, în timp ce propoziția $(\exists\, x \in \mathbb{N})\,(x$ este un număr prim$)$ este adevărată. De fapt, în enunțuri cuantificate, vom considera domeniul valorilor variabilelor stabilit de la început, și nu–l vom mai preciza după variabilele cuantificate ("$\in \mathbb{N}$").

În ce fel de structuri algebrice pot lua valori variabilele?

- Așadar, pentru a exprima matematic modul de lucru cu predicate, avem nevoie nu numai de noțiunea de propoziție cu sau fără variabile și de conectori logici, ci și de un domeniu al valorilor pentru variabilele care apar în predicate (i. e. în propozițiile cu variabile).
- Pentru a descrie sistemul formal al calculului cu predicate clasic, vom avea nevoie de noțiunea de **structură de ordinul I**, reprezentând un anumit gen de structuri algebrice. Variabilele vor fi considerate ca luând valori în diverse **structuri de ordinul I**, și **clasa structurilor de ordinul I de un anumit tip** va avea asociată propria ei logică clasică cu predicate (îi vom asocia un limbaj, apoi un sistem logic bazat pe acel limbaj).
- Intuitiv, **structurile de ordinul I** sunt structuri algebrice care posedă o mulțime suport și operații, relații și constante (operații zeroare) pe această mulțime suport, i. e. operații și relații care acționează (numai) asupra elementelor mulțimii suport.
- Când, într–o structură algebrică, există operații sau relații care acționează asupra submulțimilor mulțimii suport, i. e. asupra unor mulțimi de elemente din mulțimea suport, atunci spunem că structura respectivă este o **structură de ordinul II**. În același mod (referindu–ne la mulțimi de mulțimi de elemente ș. a. m. d.) pot fi definite **structurile de ordinul III, IV etc.**.

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor
2. Să exemplificăm noțiunile anterioare într–un program în Prolog
3. Algoritmul de unificare

Structuri algebrice cu operații și relații

Definiție

O *structură de ordinul I* este o structură de forma

$$\mathcal{A} = (A, (f_i)_{i \in I}, (R_j)_{j \in J}, (c_k)_{k \in K}),$$

unde:

- A este o mulțime nevidă, numită *universul structurii* $\mathcal{A}$
- I, J, K sunt mulțimi oarecare de indici (care pot fi și vide)
- pentru fiecare $i \in I$, există $n_i \in \mathbb{N}^*$, a. î. $f_i : A^{n_i} \to A$ (f_i este o operație n_i–ară pe A)
- pentru fiecare $j \in J$, există $m_j \in \mathbb{N}^*$, a. î. $R_j \subseteq A^{m_j}$ (R_j este o relație m_j–ară pe A)
- pentru fiecare $k \in K$, $c_k \in A$ (c_k este o constantă din A)

De obicei, operațiilor și relațiilor din componența lui $\mathcal{A}$ li se atașează indicele $\mathcal{A}$, pentru a le deosebi de simbolurile de operații și relații din limbajul pe care îl vom construi în continuare; astfel, structura de ordinul I de mai sus se notează, de regulă, sub forma: $\mathcal{A} = (A, (f_i^{\mathcal{A}})_{i \in I}, (R_j^{\mathcal{A}})_{j \in J}, (c_k^{\mathcal{A}})_{k \in K})$.

Vom avea clase de structuri de același tip

Definiție

Tipul sau *signatura* structurii de ordinul I $\mathcal{A}$ din definiția anterioară este tripletul de familii de numere naturale: $\tau := ((n_i)_{i\in I}; (m_j)_{j\in J}; (0)_{k\in K})$.
Orice structură de forma lui $\mathcal{A}$ de mai sus se numește *structură de ordinul I de tipul (sau signatura)* τ.

Exemplu

- Orice poset nevid este o structură de ordinul I de forma $\mathcal{P} = (P, \leq)$, de tipul (signatura) $\tau_1 = (\emptyset; 2; \emptyset)$ ($\leq$ este o relație binară). **Nu orice** structură de ordinul I de signatura τ_1 este un poset.
- Orice latice nevidă este o structură de ordinul I de forma $\mathcal{L} = (L, \vee, \wedge, \leq)$, de tipul (signatura) $\tau_2 = (2, 2; 2; \emptyset)$ ($\vee$ și $\wedge$ sunt operații binare (i. e. de aritate 2, i. e. cu câte două argumente), iar $\leq$ este o relație binară). **Nu orice** structură de ordinul I de signatura τ_2 este o latice.
- Orice algebră Boole este o structură de ordinul I de forma $\mathcal{B} = (B, \vee, \wedge, \bar{}, \leq, 0, 1)$, de tipul (signatura) $\tau_3 = (2, 2, 1; 2; 0, 0)$ ($\vee$ și $\wedge$ sunt operații binare, $\bar{}$ este o operație unară, $\leq$ este o relație binară, iar 0 și 1 sunt constante (operații zeroare, i. e. de aritate zero, i. e. fără argumente)). **Nu orice** structură de ordinul I de signatura τ_3 este o algebră Boole.

Limbajul asociat unei signaturi

- Fiecărei signaturi τ a unei structuri de ordinul I (fiecărei clase de structuri de ordinul I de o anumită signatură τ) i se asociază un limbaj, numit **limbaj de ordinul I** și notat, de obicei, cu $\mathcal{L}_\tau$, în care pot fi exprimate proprietățile algebrice ale structurilor de ordinul I de signatura τ.
- Să considerăm o signatură $\tau := ((n_i)_{i\in I}; (m_j)_{j\in J}; (0)_{k\in K})$, pe care o fixăm.
- **Alfabetul limbajului de ordinul I** $\mathcal{L}_\tau$ este format din următoarele **simboluri primitive**:
  1. o mulțime infinită de **variabile**: $Var = \{x, y, z, u, v, \ldots\}$;
  2. **simboluri de operații**: $(f_i)_{i\in I}$; pentru fiecare $i \in I$, numărul natural nenul n_i se numește *ordinul* sau *aritatea* lui f_i;
  3. **simboluri de relații (simboluri de predicate)**: $(R_j)_{j\in J}$; pentru fiecare $j \in J$, numărul natural nenul m_j se numește *ordinul* sau *aritatea* lui R_j;
  4. **simboluri de constante**: $(c_k)_{k\in K}$;
  5. **simbolul de egalitate**: $=$ (un semn egal îngroșat) (*a nu se confunda cu egalul simplu!*);
  6. **conectorii logici primitivi**: $\neg$ (*negația*), $\to$ (*implicația*);
  7. **cuantificatorul universal**: $\forall$ (*oricare ar fi*)
  8. paranteze: $(,), [,]$, precum și virgula.
- Pentru comoditate, vom spune uneori: "operații", "relații"/"predicate" și "constante" în loc de "simboluri de operații", " simboluri de relații/predicate" și " simboluri de constante", respectiv.

Termeni și formule

Definiție

Termenii limbajului $\mathcal{L}_\tau$ se definesc, recursiv, astfel:

1. variabilele și simbolurile de constante sunt termeni;
2. dacă f este un simbol de operație n–ară și $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este un termen;
3. orice termen se obține prin aplicarea regulilor (1) și (2) de un număr finit de ori.

Definiție

Formulele atomice ale limbajului $\mathcal{L}_\tau$ se definesc astfel:

1. dacă t_1 și t_2 sunt termeni, atunci $t_1 = t_2$ este o formulă atomică;
2. dacă R este un simbol de relație m–ară și $t_1, \ldots, t_m$ sunt termeni, atunci $R(t_1, \ldots, t_m)$ este o formulă atomică.

Observație

Majoritatea autorilor consideră virgula ca având semnificație implicită, subînțeleasă, în scrierea termenilor și a formulelor atomice, și nu includ virgula în limbajul $\mathcal{L}_\tau$.
Parantezele care încadrează formule (nu argumentele unei funcții sau relații) au același rol ca în sintaxa logicii propoziționale.

Definiție

Formulele limbajului $\mathcal{L}_\tau$ se definesc, recursiv, astfel:

1. formulele atomice sunt formule;
2. dacă φ este o formulă, atunci $\neg\varphi$ este o formulă;
3. dacă φ și ψ sunt formule, atunci $\varphi \to \psi$ este o formulă;
4. dacă φ este o formulă și x este o variabilă, atunci $\forall x\varphi$ este o formulă;
5. orice formulă se obține prin aplicarea regulilor (1), (2), (3) și (4) de un număr finit de ori.

Notație

Se notează cu $Form(\mathcal{L}_\tau)$ mulțimea formulelor limbajului $\mathcal{L}_\tau$.

Ceilalți conectori logici și celălalt cuantificator

Notație

Introducem abrevierile: pentru orice formule φ, ψ și orice variabilă x:

- **conectorii logici derivați** $\vee$ (*disjuncția*), $\wedge$ (*conjuncția*) și $\leftrightarrow$ (*echivalența*) se definesc astfel:
  $\varphi \vee \psi := \neg \varphi \rightarrow \psi$
  $\varphi \wedge \psi := \neg (\varphi \rightarrow \neg \psi)$
  $\varphi \leftrightarrow \psi := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
- **cuantificatorul existențial** $\exists$ (*există*) se definește astfel:
  $\exists x \varphi := \neg \forall x \neg \varphi$

Observație (convenție privind scrierea conectorilor logici, a cuantificatorilor și a simbolului de egalitate)

- $\neg, \forall, \exists$ vor avea prioritate mai mare;
- $\rightarrow, \vee, \wedge, \leftrightarrow, =$ vor avea prioritate mai mică.

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor
2. Să exemplificăm noțiunile anterioare într–un program în Prolog
3. Algoritmul de unificare

Variabilele care apar într–un termen

Notație (mulțimile din această notație vor fi definite recursiv mai jos)

Pentru orice termen t și orice formulă φ, notăm:

- $V(t) :=$ mulțimea variabilelor termenului t
- $FV(\varphi) :=$ mulțimea variabilelor *libere* ale formulei φ

Definiție

Pentru orice termen t:

- dacă $t = x$, unde x este o variabilă, atunci $V(t) := \{x\}$
- dacă $t = c$, unde c este o constantă, atunci $V(t) := \emptyset$
- dacă $t = f(t_1, \ldots, t_n)$, unde f este un simbol de operație n–ară și $t_1, \ldots, t_n$ sunt termeni, atunci $V(t) := \bigcup_{i=1}^{n} V(t_i)$

Variabilele libere dintr–o formulă

Definiție

Pentru orice formulă φ:

- dacă $\varphi = (t_1 = t_2)$, unde t_1 și t_2 sunt termeni, atunci $FV(\varphi) := V(t_1) \cup V(t_2)$
- dacă $\varphi = R(t_1, \ldots, t_m)$, unde R este un simbol de relație m–ară și $t_1, \ldots, t_m$ sunt termeni, atunci $FV(\varphi) := \bigcup_{i=1}^{m} V(t_i)$
- dacă $\varphi = \neg\psi$, pentru o formulă ψ, atunci $FV(\varphi) := FV(\psi)$
- dacă $\varphi = \psi \to \chi$, pentru două formule ψ, χ, atunci $FV(\varphi) := FV(\psi) \cup FV(\chi)$
- dacă $\varphi = \forall x\psi$, pentru o formulă ψ și o variabilă x, atunci $FV(\varphi) := FV(\psi) \setminus \{x\}$

Remarcă

Este imediat, din definiția anterioară și definiția conectorilor logici derivați și a cuantificatorului existențial, că, pentru orice formule ψ, χ și orice variabilă x:

- $FV(\psi \vee \chi) = FV(\psi \wedge \chi) = FV(\psi \leftrightarrow \chi) = FV(\psi) \cup FV(\chi)$
- $FV(\exists x\psi) = FV(\psi) \setminus \{x\}$

Variabilele legate dintr–o formulă

Definiție

Pentru orice variabilă x și orice formulă φ:

- dacă $x \in FV(\varphi)$, atunci x se numește *variabilă liberă a lui* φ;
- dacă $x \notin FV(\varphi)$, atunci x se numește *variabilă legată a lui* φ;
- dacă $FV(\varphi) = \emptyset$ (i. e. φ nu are variabile libere), atunci φ se numește *enunț*.

Exemplu

- În formula $\exists x(x^2 = x)$, x este variabilă legată. Această formulă este un enunț.
- În formula $\forall y \forall z(z \cdot x \leq y \cdot z)$, x este variabilă liberă.

Notație (cum specificăm că nu avem alte variabile libere)

Fie $n \in \mathbb{N}^*$ și $x_1, \ldots, x_n$ variabile.
Dacă t este un termen cu $V(t) \subseteq \{x_1, \ldots, x_n\}$, atunci vom nota $t(x_1, \ldots, x_n)$.
Dacă φ este o formulă cu $FV(\varphi) \subseteq \{x_1, \ldots, x_n\}$, atunci vom nota $\varphi(x_1, \ldots, x_n)$.

Variabile libere și variabile legate

Observație (diferența dintre tipurile de variabile, intuitiv)

- **Variabilele libere** sunt variabilele care nu intră sub incidența unui cuantificator, variabilele cărora "avem libertatea de a le da valori".
- **Variabilele legate** sunt variabilele care intră sub incidența unui cuantificator, deci sunt destinate parcurgerii unei întregi mulțimi de valori.

Definiție

Fie x o variabilă, $\varphi(x)$ o formulă și t un termen.
Formula obținută din φ prin *substituția lui x cu t* se notează cu $\varphi(t)$ și se definește astfel:

- fiecare $y \in V(t)$ se înlocuiește cu o variabilă $v \notin V(t)$ care nu apare în $\varphi(x)$, în toate aparițiile *legate* ale lui y în $\varphi(x)$;
- apoi se înlocuiește x cu t.

Exemplu

Fie variabilele x, y, z, formula $\varphi(x) := \exists\, y(x < y)$ și termenul $t := y + z$.
Atunci $\varphi(t)$ se obține astfel:

- $\varphi(x) = \exists\, y(x < y)$ se înlocuiește cu $\exists\, v(x < v)$;
- prin urmare, $\varphi(t) = \exists\, v(y + z < v)$.

Axiomele logicii clasice a predicatelor

Axiomele calculului cu predicate clasic: pentru φ, ψ, χ formule arbitrare, t termen arbitrar, n, i numere naturale nenule arbitrare și $x, y, y_1, \ldots, y_n, v_1, \ldots, v_n$ variabile arbitrare:

- axiomele calculului propozițional:
  $(G_1) \quad \varphi \to (\psi \to \varphi)$
  $(G_2) \quad (\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$
  $(G_3) \quad (\neg \varphi \to \neg \psi) \to (\psi \to \varphi)$
- *regula* $(\to \forall)$:
  $(G_4) \quad \forall x(\varphi \to \psi) \to (\varphi \to \forall x\psi)$, dacă $x \notin FV(\varphi)$
- o regulă privind substituțiile:
  $(G_5) \quad \forall x\varphi(x, y_1, \ldots, y_n) \to \varphi(t, y_1, \ldots, y_n)$
- *axiomele egalității*:
  $(G_6) \quad x{=}x$
  (G_7)
  $(x{=}y) \to (t(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_n){=}t(v_1, \ldots, v_{i-1}, y, v_{i+1}, \ldots, v_n))$
  (G_8)
  $(x{=}y) \to (\varphi(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_n){=}\varphi(v_1, \ldots, v_{i-1}, y, v_{i+1}, \ldots, v_n))$

Teoremele formale, i.e. adevărurile sintactice în logica clasică a predicatelor

Notație

Faptul că o formulă φ este *teoremă (formală) (adevăr sintactic) a(l) lui* $\mathcal{L}_\tau$ se notează cu $\vdash \varphi$ și se definește, recursiv, ca mai jos.

Definiție

1. Orice axiomă e teoremă formală a lui $\mathcal{L}_\tau$.
2. Pentru orice formule φ, ψ, $\dfrac{\vdash \psi, \vdash \psi \to \varphi}{\vdash \varphi}$ (regula de deducție *modus ponens* (MP)).
3. Pentru orice formulă φ și orice variabilă x, $\dfrac{\vdash \varphi}{\vdash \forall x \varphi}$ (regula de deducție numită *principiul generalizării* (PG)).
4. Orice teoremă formală se obține prin aplicarea regulilor (1), (2) și (3) de un număr finit de ori.

Deducție sintactică în logica clasică a predicatelor

Notație

Fie Σ o mulțime de formule ale lui $\mathcal{L}_\tau$. Faptul că o formulă φ *se deduce (formal) din ipotezele* Σ (φ *este consecință sintactică a mulțimii de ipoteze* Σ) se notează cu $\Sigma \vdash \varphi$ și se definește, recursiv, ca mai jos.

Definiție

Fie Σ o mulțime de formule ale lui $\mathcal{L}_\tau$.

1. Orice axiomă a lui $\mathcal{L}_\tau$ se deduce formal din Σ.
2. $\Sigma \vdash \varphi$, oricare ar fi $\varphi \in \Sigma$.
3. Pentru orice formule φ, ψ, $\dfrac{\Sigma \vdash \psi, \Sigma \vdash \psi \to \varphi}{\Sigma \vdash \varphi}$ (regula de deducție *modus ponens* (MP)).
4. Pentru orice formulă φ și orice variabilă x, $\dfrac{\Sigma \vdash \varphi}{\Sigma \vdash \forall x \varphi}$ (regula de deducție numită *principiul generalizării* (PG)).
5. Orice consecință sintactică a lui Σ se obține prin aplicarea regulilor (1), (2), (3) și (4) de un număr finit de ori.

Teorema deducției sintactice în logica clasică a predicatelor

Remarcă

Pentru orice formulă φ, are loc echivalența:

$$\emptyset \vdash \varphi \Leftrightarrow \vdash \varphi.$$

Teoremă (Teorema deducției)

Pentru orice mulțime de **formule** Σ, *orice* **enunț** φ *și orice* **formulă** ψ, *are loc echivalența:*

$$\Sigma \vdash \varphi \to \psi \Leftrightarrow \Sigma \cup \{\varphi\} \vdash \psi.$$

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor
2. Să exemplificăm noțiunile anterioare într–un program în Prolog
3. Algoritmul de unificare

O interpretare/evaluare/semantică dă variabilelor valori într–o structură de ordinul I

- Fie $\mathcal{A}$ o structură de ordinul I de signatură τ.
- Fixăm pe $\mathcal{A}$ pentru cele ce urmează.
- A va fi universul structurii $\mathcal{A}$ (mulțimea ei suport).
- Pentru fiecare simbol de operație f, fiecare simbol de relație R și fiecare simbol de constantă c din signatura τ, notăm cu $f^{\mathcal{A}}$, respectiv $R^{\mathcal{A}}$, respectiv $c^{\mathcal{A}}$ operația, respectiv relația, respectiv constanta corespunzătoare din $\mathcal{A}$. Uneori vom nota operațiile $f^{\mathcal{A}}$, relațiile $R^{\mathcal{A}}$ și constantele $c^{\mathcal{A}}$ fără indicele superior $\mathcal{A}$: f, R, respectiv c.

Definiție

O *interpretare* (sau *evaluare*, sau *semantică*) a limbajului $\mathcal{L}_\tau$ în structura $\mathcal{A}$ este o funcție $s : Var \to A$.
Fiecare variabilă $x \in Var$ este "interpretată" prin elementul $s(x) \in A$.

Prelungim o interpretare s la mulțimea tuturor termenilor, obținând o funcție $s : Term(\mathcal{L}_\tau) \to A$.

Definiție (valorile asociate termenilor de o interpretare)

Pentru orice interpretare s și orice termen t, definim recursiv elementul $s(t) \in A$:

- dacă $t = x \in Var$, atunci $s(t) = s(x)$;
- dacă $t = c \in C$, atunci $s(t) = c^{\mathcal{A}}$;
- dacă $t = f(t_1, \ldots, t_n)$, unde $f \in F$ are aritatea $n \in \mathbb{N}^*$, iar $t_1, \ldots, t_n \in Term(\mathcal{L}_\tau)$, atunci $s(t) = f^{\mathcal{A}}(s(t_1), \ldots, s(t_n))$.

Notație

Pentru orice interpretare $s : Var \to A$, orice $x \in Var$ și orice $a \in A$, notăm cu $s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right] : Var \to A$ interpretarea definită prin: oricare ar fi $v \in Var$,

$$s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right](v) = \begin{cases} a, & \text{dacă } v = x, \\ s(v), & \text{dacă } v \neq x. \end{cases}$$

$s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right]$ ia valoarea a în variabila x și aceeași valoare ca s în celelalte variabile.

Definiție (valorile booleene asociate formulelor de o interpretare)

Pentru orice interpretare s și orice formulă φ, *valoarea de adevăr a lui φ în interpretarea s* este un element din algebra Boole standard $\mathcal{L}_2 = \{0, 1\}$, notat cu $s(\varphi)$, definit, recursiv, astfel:

- dacă $\varphi = (t_1 = t_2)$, pentru doi termeni t_1, t_2, atunci
$$s(\varphi) = \begin{cases} 1, & \text{dacă } s(t_1) = s(t_2), \\ 0, & \text{dacă } s(t_1) \neq s(t_2) \end{cases}$$
- dacă $\varphi = R(t_1, \ldots, t_m)$, unde $R \in \mathcal{R}$ are aritatea $m \in \mathbb{N}^*$, iar $t_1, \ldots, t_m$ sunt termeni, atunci $s(\varphi) = \begin{cases} 1, & \text{dacă } (s(t_1), \ldots, s(t_m)) \in R^{\mathcal{A}}, \\ 0, & \text{dacă } (s(t_1), \ldots, s(t_m)) \notin R^{\mathcal{A}} \end{cases}$
- dacă $\varphi = \neg\, \psi$, pentru o formulă ψ, atunci $s(\varphi) = \overline{s(\psi)}$;
- dacă $\varphi = \psi \to \chi$, pentru două formule ψ, χ, atunci $s(\varphi) = s(\psi) \to s(\chi)$;
- dacă $\varphi = \forall\, x\psi$, unde $x \in \mathit{Var}$, iar ψ este o formulă, atunci $s(\varphi) = \bigwedge_{a \in A} s[{}^{x}_{a}](\psi)$.

Remarcă

Este imediat că, pentru orice interpretare $s : Var \to A$, orice formule ψ, χ și orice $x \in Var$, au loc egalitățile:

- $s(\psi \vee \chi) = s(\psi) \vee s(\chi)$;
- $s(\psi \wedge \chi) = s(\psi) \wedge s(\chi)$;
- $s(\psi \leftrightarrow \chi) = s(\psi) \leftrightarrow s(\chi)$;
- $s(\exists x\psi) = \bigvee_{a \in A} s[{}^{x}_{a}](\psi)$.

Lemă (dacă două interpretări coincid pe variabilele dintr–un termen, atunci ele coincid în acel termen)

Fie $s_1, s_2 : Var \to A$ două interpretări. Atunci, pentru orice termen t, are loc implicația: $s_1 \mid_{V(t)} = s_2 \mid_{V(t)} \Rightarrow s_1(t) = s_2(t)$.

Propoziție (dacă două interpretări coincid pe variabilele libere dintr–o formulă, atunci ele coincid în acea formulă)

Fie $s_1, s_2 : Var \to A$ două interpretări. Atunci, pentru orice formulă φ, are loc implicația: $s_1 \mid_{FV(\varphi)} = s_2 \mid_{FV(\varphi)} \Rightarrow s_1(\varphi) = s_2(\varphi)$.

În mod trivial, oricare două interpretări $s_1, s_2 : Var \to A$ coincid pe $\emptyset$: $(\forall x)(x \in \emptyset \Rightarrow s_1(x) = s_2(x))$ e adevărată, pentru că $x \in \emptyset$ e falsă pentru orice x, așadar $x \in \emptyset \Rightarrow s_1(x) = s_2(x)$ e adevărată pentru orice x. Prin urmare:

Corolar (cum enunțurile nu au variabile libere (i.e. $FV(\varphi) = \emptyset$ pt. orice enunț φ), rezultă că, într–un enunț, toate interpretările au aceeași valoare)

Dacă φ este un enunț, atunci $s(\varphi)$ nu depinde de interpretarea $s : Var \to A$.

Notație (această valoare nu depinde decât de structura algebrică $\mathcal{A}$)

Corolarul anterior ne permite să notăm, pentru orice enunț φ, cu $||\varphi||_{\mathcal{A}} = s(\varphi)$ valoarea oricărei interpretări $s : Var \to A$ în enunțul φ.

Definiție

Pentru orice enunț φ, notăm:

$$\mathcal{A} \models \varphi \text{ ddacă } ||\varphi||_{\mathcal{A}} = 1.$$

În acest caz, spunem că $\mathcal{A}$ *satisface* φ sau φ *este adevărat în* $\mathcal{A}$ sau $\mathcal{A}$ *este model pentru* φ.
Pentru orice mulțime Γ de enunțuri, spunem că $\mathcal{A}$ *satisface* Γ sau că $\mathcal{A}$ *este model pentru* Γ ddacă $\mathcal{A} \models \varphi$, pentru orice $\varphi \in \Gamma$. Notăm acest lucru cu $\mathcal{A} \models \Gamma$.

Recapitulare Logica Clasică a Predicatelor

Remarcă

Este imediat, din definiția mulțimii variabilelor libere ale unei formule, că, dacă $n \in \mathbb{N}^*$, $x_1, \ldots, x_n \in V$ și $\varphi(x_1, \ldots, x_n)$ este o formulă, atunci $\forall x_1 \ldots \forall x_n \varphi(x_1, \ldots, x_n)$ este un enunț.

Definiție

Pentru orice $n \in \mathbb{N}^*$, orice variabile $x_1, \ldots, x_n$ și orice formulă $\varphi(x_1, \ldots, x_n)$, notăm:

$$\mathcal{A} \vDash \varphi(x_1, \ldots, x_n) \text{ ddacă } \mathcal{A} \vDash \forall x_1 \ldots \forall x_n \varphi(x_1, \ldots, x_n).$$

În acest caz, spunem că $\mathcal{A}$ *satisface* $\varphi(x_1, \ldots, x_n)$ sau $\varphi(x_1, \ldots, x_n)$ *este adevărată în* $\mathcal{A}$ sau $\mathcal{A}$ *este model pentru* $\varphi(x_1, \ldots, x_n)$.
Pentru orice mulțime Σ de formule, spunem că $\mathcal{A}$ *satisface* Σ sau că $\mathcal{A}$ *este model pentru* Σ ddacă $\mathcal{A}$ este model pentru fiecare formulă din Σ. Notăm acest lucru cu $\mathcal{A} \vDash \Sigma$.

Remarcă

$\mathcal{A} \vDash \emptyset$.

Tautologiile, i.e. adevărurile semantice ale logicii clasice a predicatelor

- Renunțăm la fixarea structurii $\mathcal{A}$ (**nu** și la fixarea signaturii τ).

Definiție

Dacă φ este un enunț, atunci spunem că φ este *universal adevărat (adevăr semantic, tautologie)* ddacă $\mathcal{A} \models \varphi$, oricare ar fi structura de ordinul I $\mathcal{A}$ de signatură τ. Notăm acest lucru cu $\models \varphi$.

Definiție

Dacă $n \in \mathbb{N}^*$, $x_1, \ldots, x_n \in V$ și $\varphi(x_1, \ldots, x_n)$ este o formulă, atunci spunem că $\varphi(x_1, \ldots, x_n)$ este *universal adevărată (adevăr semantic, tautologie)* ddacă enunțul $\forall x_1 \ldots \forall x_n \varphi(x_1, \ldots, x_n)$ este universal adevărat. Notăm acest lucru cu $\models \varphi(x_1, \ldots, x_n)$.

Deducția semantică în logica clasică a predicatelor

Definiție

Pentru orice mulțime Σ de formule și orice formulă φ, spunem că φ *se deduce semantic din ipotezele* Σ sau că φ *este consecință semantică a mulțimii de ipoteze* Σ ddacă φ este adevărată în orice model $\mathcal{A}$ al lui Σ, i. e., pentru orice structură de ordinul I $\mathcal{A}$ de signatură τ, are loc implicația: $\mathcal{A} \models \Sigma \Rightarrow \mathcal{A} \models \varphi$. Notăm acest lucru prin: $\Sigma \models \varphi$.

Remarcă

Pentru orice formulă φ, are loc echivalența:

$$\emptyset \models \varphi \Leftrightarrow \models \varphi.$$

Teoremă (Teorema deducției semantice)

Pentru orice mulțime de **formule** Σ, *orice* **enunț** φ *și orice* **formulă** ψ, *are loc echivalența:*

$$\Sigma \models \varphi \to \psi \Leftrightarrow \Sigma \cup \{\varphi\} \models \psi.$$

Teorema de completitudine pentru logica clasică a predicatelor

Teoremă (Teorema de completitudine tare (Teorema de completitudine extinsă))

Pentru orice formulă φ și orice mulțime de formule Σ, are loc echivalența:

$$\Sigma \vdash \varphi \Leftrightarrow \Sigma \models \varphi.$$

În cazul particular în care $\Sigma = \emptyset$, din **Teorema de completitudine extinsă** obținem:

Corolar (Teorema de completitudine)

Pentru orice formulă φ, are loc echivalența:

$$\vdash \varphi \Leftrightarrow \models \varphi.$$

1. Recapitulare Logica Clasică a Predicatelor
   - Calculul clasic cu predicate
   - Structuri de ordinul I și limbaje asociate signaturilor lor
   - Variabile și substituții
   - Sintaxa calculului cu predicate clasic
   - Semantica logicii clasice a predicatelor

2. Să exemplificăm noțiunile anterioare într–un program în Prolog

3. Algoritmul de unificare

Vom vedea că un program în Prolog definește o întreagă clasă de structuri de ordinul I

$femeie(ana).$ $femeie(carmen).$ $femeie(elena).$ $femeie(eva).$
$femeie(maria).$ $femeie(sara).$ $femeie(valentina).$
$barbat(adam).$ $barbat(constantin).$ $barbat(eugen).$ $barbat(george).$
$barbat(ion).$ $barbat(iosif).$ $barbat(tudor).$ $barbat(victor).$
$parinte(sara, adam).$ $parinte(sara, eva).$
$parinte(constantin, iosif).$ $parinte(constantin, elena).$
$parinte(valentina, ion).$ $parinte(valentina, sara).$
$parinte(victor, constantin).$ $parinte(victor, maria).$
$parinte(ana, victor).$ $parinte(ana, valentina).$
$parinte(carmen, victor).$ $parinte(carmen, valentina).$
$parinte(george, victor).$ $parinte(george, valentina).$
$parinte(eugen, tudor).$ $parinte(eugen, ana).$
$frati(X, Y)$:- $X \backslash= Y$, $parinte(X, P)$, $parinte(Y, P)$.
$tata(X, T)$:- $parinte(X, T)$, $barbat(T)$.
$mama(X, M)$:- $parinte(X, M)$, $femeie(M)$.
$unchi(X, U)$:- $parinte(X, P)$, $frati(P, U)$, $barbat(U)$.
$matusa(X, M)$:- $parinte(X, P)$, $frati(P, M)$, $femeie(M)$.
$bunic(X, B)$:- $parinte(X, P)$, $parinte(P, B)$.
$stramos(X, X, 0)$. % al treilea argument e numarul de generatii
$stramos(X, S, G)$:- $parinte(X, P)$, $stramos(P, S, H)$, G is $H + 1$.

În programul în Prolog de mai sus:

- X, P, T, M, U, B, S, G, H sunt **variabile**;
- $+$ este **operație**;
- *femeie*, *barbat*, *frati*, *parinte*, *tata*, *mama*, *unchi*, *matusa*, *bunic*, *stramos* sunt **predicate**;
- $H + 1$ este un **termen**;
- *ana*, *carmen*, *elena*, *eva*, *maria*, *sara*, *valentina*, *adam*, *constantin*, *eugen*, *george*, *ion*, *iosif* , *tudor*, *victor*, 0 și 1 sunt **constante** Prolog;
- de exemplu, $femeie(eva)$, $frati(victor, george)$, $parinte(victor, ion)$, $parinte(X, M)$, $stramos(X, X, 0)$, $stramos(X, S, G)$ sunt **formule atomice**.

REGULILE și ultimul FAPT din programul anterior corespund **formulelor**:

(atenție: toate variabilele cuantificate UNIVERSAL)

- $(\forall X)(\forall Y)(\forall P)[(\neg(X = Y) \wedge parinte(X, P) \wedge parinte(Y, P)) \rightarrow frati(X, Y)]$
- $(\forall X)(\forall T)[(parinte(X, T) \wedge barbat(T)) \rightarrow tata(X, T)]$
- $(\forall X)(\forall M)[(parinte(X, M) \wedge femeie(M)) \rightarrow mama(X, M)]$
- $(\forall X)(\forall U)(\forall P)[(parinte(X, P) \wedge frati(P, U) \wedge barbat(U)) \rightarrow unchi(X, U)]$
- $(\forall X)(\forall M)(\forall P)[(parinte(X, P) \wedge frati(P, M) \wedge femeie(M)) \rightarrow matusa(X, M)]$
- $(\forall X)(\forall B)[(parinte(X, P) \wedge parinte(P, B)) \rightarrow bunic(X, B)]$

- $(\forall X)\,(stramos(X, X, 0))$
- $(\forall X)\,(\forall S)\,(\forall G)\,[(parinte(X, P) \wedge stramos(P, S, H) \wedge (G = H + 1)) \rightarrow stramos(X, S, G)]$

INTEROGĂRILE:

?- *parinte*(*C*, *victor*). % Care sunt copiii lui Victor?
?- *parinte*(*F*, *victor*), *femeie*(*F*). % Care sunt fiicele lui Victor?
?- *stramos*(*carmen*, *S*, 3), *barbat*(*S*). /* Care sunt strabunicii (cunoscuti, adica din baza de cunostinte de mai sus, ai) lui Carmen? */
?- *tata*(*X*, *T*), *stramos*(*T*, *elena*, *G*). /* Pentru cine este Elena stramos din partea tatalui, cine este tatal acelui urmas si cate generatii sunt intre tata si Elena? */

corespund **formulelor**:

(atenție: toate variabilele cuantificate EXISTENȚIAL)

- $(\exists C)\,(parinte(C, victor))$
- $(\exists F)\,(parinte(F, victor) \wedge femeie(F))$
- $(\exists S)\,(stramos(carmen, S, 3) \wedge barbat(S))$
- $(\exists X)\,(\exists T)\,(\exists G)\,(tata(X, T) \wedge stramos(T, elena, G))$

Care este **signatura** limbajului de ordinul I definit în programul Prolog de mai sus?

Pentru început, să considerăm programul anterior, mai puțin definiția predicatului **stramos**:

$femeie(ana).$ $femeie(carmen).$ $femeie(elena).$ $femeie(eva).$
$femeie(maria).$ $femeie(sara).$ $femeie(valentina).$
$barbat(adam).$ $barbat(constantin).$ $barbat(eugen).$ $barbat(george).$
$barbat(ion).$ $barbat(iosif).$ $barbat(tudor).$ $barbat(victor).$
$parinte(sara, adam).$ $parinte(sara, eva).$
$parinte(constantin, iosif).$ $parinte(constantin, elena).$
$parinte(valentina, ion).$ $parinte(valentina, sara).$
$parinte(victor, constantin).$ $parinte(victor, maria).$
$parinte(ana, victor).$ $parinte(ana, valentina).$
$parinte(carmen, victor).$ $parinte(carmen, valentina).$
$parinte(george, victor).$ $parinte(george, valentina).$
$parinte(eugen, tudor).$ $parinte(eugen, ana).$
$frati(X, Y) :- X \backslash= Y, parinte(X, P), parinte(Y, P).$
$tata(X, T) :- parinte(X, T), barbat(T).$
$mama(X, M) :- parinte(X, M), femeie(M).$
$unchi(X, U) :- parinte(X, P), frati(P, U), barbat(U).$
$matusa(X, M) :- parinte(X, P), frati(P, M), femeie(M).$
$bunic(X, B) :- parinte(X, P), parinte(P, B).$

Signatura corespunzătoare programului de mai sus este
$\tau = (\underbrace{\emptyset}_{\text{aritățile operațiilor}} ; \underbrace{2,2,2,2,2,2,2,1,1}_{\text{aritățile relațiilor}} ; \underbrace{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}_{\text{aritățile constantelor}})$.
Amintesc că *aritățile* sunt numerele de argumente.
Structura de ordinul I de această signatură definită în programul de mai sus este:
$\mathcal{A} = (A; \emptyset;$ *frati*, *parinte*, *tata*, *mama*, *unchi*, *matusa*, *bunic*, *femeie*, *barbat*; *ana*, *carmen*, *elena*, *eva*, *maria*, *sara*, *valentina*, *adam*, *constantin*, *eugen*, *george*, *ion*, *iosif* , *tudor*, *victor*), cu:

- mulțimea suport (i.e. mulțimea elementelor) $A = \{$*ana*, *carmen*, *elena*, *eva*, *maria*, *sara*, *valentina*, *adam*, *constantin*, *eugen*, *george*, *ion*, *iosif* , *tudor*, *victor*$\}$;
- nicio operație;
- 7 predicate (i.e. relații) binare (i.e. de aritate 2, i.e. cu 2 argumente): *frati*, *parinte*, *tata*, *mama*, *unchi*, *matusa*, *bunic* $\subseteq A^2$;
- 2 predicate (i.e. relații) unare (i.e. de aritate 1, i.e. cu 1 argument): *femeie*, *barbat* $\subseteq A$;
- 15 constante (amintesc că acestea sunt operațiile zeroare, operațiile de aritate 0, i.e. operațiile fără argumente): *ana*, *carmen*, *elena*, *eva*, *maria*, *sara*, *valentina*, *adam*, *constantin*, *eugen*, *george*, *ion*, *iosif* , *tudor*, *victor* $\in A$.

Primele fapte din acest program definesc relațiile unare *femeie* și *barbat*:

- $femeie = \{ana, carmen, elena, eva, maria, sara, valentina\}$;
- $barbat = \{adam, constantin, eugen, george, ion, iosif, tudor, victor\}$.

Pentru orice $x \in A$, faptul că $x \in femeie$ se mai scrie: $femeie(x)$. La fel pentru relația unară *barbat*.

Următoarele fapte definesc relația binară *parinte*:

- $parinte = \{(sara, adam), (sara, eva), (constantin, iosif), (constantin, elena), (valentina, ion), (valentina, sara), (victor, constantin), (victor, maria), (ana, victor), (ana, valentina), (carmen, victor), (carmen, valentina), (george, victor), (george, valentina), (eugen, tudor), (eugen, ana)\}$.

Pentru orice $x, y \in A$, faptul că $(x, y) \in parinte$ se mai scrie: $parinte(x, y)$. La fel pentru următoarele relații binare.

Regulile următoare definesc relațiile binare *frati*, *tata*, *mama*, *unchi* și *matusa*:

- $frati = \{(x, y) \in A^2 \mid x \neq y, (\exists\, p \in A)\,((x, p), (y, p) \in parinte)\}$;
- $tata = \{(x, t) \in A^2 \mid (x, t) \in parinte, t \in barbat\}$;
- $mama = \{(x, m) \in A^2 \mid (x, m) \in parinte, m \in femeie\}$;
- $unchi = \{(x, u) \in A^2 \mid (\exists\, p \in A)\,((x, p) \in parinte, (p, u) \in frati, u \in barbat)\}$;
- $matusa = \{(x, m) \in A^2 \mid (\exists\, p \in A)\,((x, p) \in parinte, (p, m) \in frati, m \in femeie)\}$;
- $bunic = \{(x, b) \in A^2 \mid (\exists\, p \in A)\,((x, p), (p, b) \in parinte)\}$.

Și acum să reanalizăm programul inițial, care include și predicatul ternar (relația ternară, i.e. de aritate 3, i.e. cu 3 argumente) $stramos \subseteq A \times A \times \mathbb{N}$, definită prin ultimul fapt și ultima regulă, care dau următoarea definiție recursivă:

- $\{(x, x, 0) \mid x \in A\} \subseteq stramos$ și:
- pentru orice $x, s, p \in A$ și orice $h \in \mathbb{N}$, dacă $(x, p) \in parinte$ și $(p, s, h) \in stramos$, atunci $(x, s, h + 1) \in stramos$.

Pentru orice $x, s \in A$ și orice $g \in \mathbb{N}$, faptul că $(x, s, g) \in stramos$ se mai scrie $stramos(x, s, g)$.

Dar *stramos* nu este o relație ternară pe A, ci are primele două argumente din A, iar al treilea din $\mathbb{N}$!
În plus, 0 și 1 sunt constante din $\mathbb{N}$, nu din A, iar $+ : \mathbb{N}^2 \to \mathbb{N}$, deci $+$ este o operație binară pe $\mathbb{N}$, nu pe A.

Așadar cadrul de lucru din secțiunea anterioară a acestui curs trebuie extins!
Putem face acest lucru înlocuind mulțimea suport a structurii $\mathcal{A}$ cu $A \cup \mathbb{N}$ și permițând operațiilor să fie parțial definite.

Definiție (mnemonic din cursul de logică matematică)

Dacă M și N sunt mulțimi, atunci o *funcție parțială* (*relație binară funcțională*) de la M la N este o relație binară între M și N $\varphi \subseteq M \times N$ cu proprietatea că:

pentru orice $x \in M$, există **cel mult un** $y \in N$ cu proprietatea că $(x, y) \in \varphi$.

Definiție (continuare)

Dacă, pentru un $x \in M$, există un $y \in N$ cu $(x, y) \in \varphi$, atunci, ca în cazul *funcțiilor* (i.e. al *relațiilor binare funcționale totale*), se notează $\varphi(x) = y$ și acest $y \in N$ se numește *valoarea funcției parțiale* φ *în punctul* x.

Notație

Dacă M și N sunt mulțimi, atunci faptul că φ este o funcție parțială de la M la N se notează:

$$\varphi : M \dashrightarrow N.$$

Exemplu (să analizăm această soluție pe un tip cunoscut de algebră)

Dacă $\mathcal{V}$ este un spațiu vectorial, format din:

- un grup abelian $(G, \circ, \bar{}, e)$, cu $\circ : G^2 \to G$ (operație binară pe G: legea de compoziție), $\bar{} : G \to G$ (operație unară pe G: inversarea) și $e \in G$ (constantă din G, i.e. operație zeroară pe G: elementul neutru),
- un corp $(K, +, \cdot, -, ^{-1}, 0, 1)$, cu $+ : K^2 \to K$ și $\cdot : K^2 \to K$ (operații binare pe K), $- : K \to K$ și $^{-1} : K \to K$ (operații unare pe K), iar $0, 1 \in K$ (constante din K, i.e. operații zeroare pe K),
- și o lege de compoziție între scalarii din K și vectorii din G: $* : K \times G \to G$.

O structură algebrică având mulțimea suport $K \times V$ ar trebui să aibă orice operație f de aritate $n \in \mathbb{N}$ de forma $f : (K \times V)^n \to K \times V$, iar, în cazul unei

Exemplu (continuare)

structuri cu mulțimea suport $K \cup V$, o operație g de aritate n ar trebui să fie o funcție $g : (K \cup V)^n \to K \cup V$.
Așadar, putem privi spațiul vectorial $\mathcal{V}$ ca pe o structură algebrică având mulțimea suport $K \cup V$ dacă permitem operațiilor sale să fie funcții parțiale.
O altă posibilitate este să privim spațiul vectorial $\mathcal{V}$ ca pe o structură algebrică având două mulțimi suport: K și G, mai precis familia de mulțimi (K, G), cu operațiile total definite, dar astfel încât, pentru orice $m, p \in \mathbb{N}$, o operație h de aritate $m + p$ (i.e. cu $m + p$ argumente) a lui $\mathcal{V}$ să poată fi de forma:
$h : K^m \times G^p \to K$ sau $h : K^m \times G^p \to G$.
O astfel de structură algebrică se numește *algebră bisortată*, i.e. *cu două sorturi*, cu două mulțimi suport, două tipuri de elemente, în cazul acesta: SCALARI din K și VECTORI din G.

La fel ca în exemplul anterior, o *algebră multisortată*, i.e. *cu mai multe sorturi*, cu mai multe mulțimi suport, mai multe tipuri de elemente, și care, pe lângă operații, e înzestrată și cu, relații, va fi o algebră de forma:
$\mathcal{M} = ((M_s)_{s \in S}; (f_i)_{i \in I}; (\rho_j)_{j \in J}; (c_k)_{k \in K})$, unde S e *mulțimea de sorturi*, elementele structurii algebrice $\mathcal{M}$ sunt elementele mulțimilor din familia de mulțimi $(M_s)_{s \in S}$, $(f_i)_{i \in I}$ este familia de operații, $(\rho_j)_{j \in J}$ este familia de relații, iar $(c_k)_{k \in K}$ este familia de constante ale structurii algebrice $\mathcal{M}$:

Pentru orice mulțime **nevidă** S, o *algebră S–sortată* înzestrată și cu operații, și cu, relații este o structură algebrică $\mathcal{M} = ((M_s)_{s \in S}; (f_i)_{i \in I}; (\rho_j)_{j \in J}; (c_k)_{k \in K})$, unde:

- elementele lui S se numesc *sorturi* (intuitiv, *sorturile* sunt indici corespunzători TIPURILOR DE ELEMENTE, în acest caz TIPURILOR DE DATE cu care lucrează programul de mai sus);
- $(M_s)_{s \in S}$ este *mulțimea S–sortată* a **elementelor** lui $\mathcal{M}$, adică M este o familie de mulțimi indexată de S și, pentru fiecare $s \in S$, elementele lui M_s se numesc *elementele lui $\mathcal{M}$ de sort s*;
- I, J, K sunt mulțimi, nu neapărat nevide;
- pentru fiecare $i \in I$, există $n_i \in \mathbb{N}^*$ și sorturile $s_{i,1}, \ldots, s_{i,n_i}, s_i \in S$ astfel încât $f_i : M_{s_{i,1}} \times \ldots \times M_{s_{i,n_i}} \to M_{s_i}$: f_i este o operație de aritate n_i a lui $\mathcal{M}$;
- pentru fiecare $j \in J$, există $m_j \in \mathbb{N}^*$ și sorturile $t_{j,1}, \ldots, t_{j,m_j} \in S$ astfel încât $\rho_j \subseteq M_{t_{j,1}} \times \ldots \times M_{t_{j,m_j}} \to M_{t_j}$: ρ_j este o relație de aritate m_j a lui $\mathcal{M}$;
- pentru fiecare $k \in K$, există sortul $u_k \in S$ astfel încât $c_k \in M_{u_k}$: c_k este o constantă a lui $\mathcal{M}$.

Putem defini structura de ordinul I corespunzătoare programului de mai sus ca pe o algebră bisortată, cu mulțimea suport $(A, \mathbb{N})$.

Desigur, operația binară $+$ pe $\mathbb{N}$ nu este definită în programul de mai sus, ci este predefinită în Prolog, împreună cu alte operații aritmetice și relații, de exemplu $<$, $=<$ etc..

Să mai observăm că relațiile (predicatele) pot fi privite ca operații având *sortul rezultatului* BOOLEAN: adăugăm o a treia mulțime suport, conținând valorile booleene: $\{\text{false}, \text{true}\}$ sau $\mathcal{L}_2$=$\{0,1\}$ ($0 = \text{false}$, $1 = \text{true}$), astfel că structura algebrică asociată programului de mai sus devine o *algebră trisortată*, cu mulțimea suport $(A_s)_{s\in\overline{1,3}} = (A, \mathbb{N}, \{\text{false}, \text{true}\})$, și orice relație din această algebră devine o operație cu valori în $\{\text{false}, \text{true}\}$: pentru orice $n \in \mathbb{N}^*$ și orice relație n–ară $\rho \subseteq A_{s_1} \times \ldots \times A_{s_n}$, unde $s_1, \ldots, s_n$ sunt SORTURI, în acest caz aparținând mulțimii de sorturi $\overline{1,3}$, în Prolog această relație este implementată ca o operație

$$\rho' : A_{s_1} \times \ldots \times A_{s_n} \to \{\text{false}, \text{true}\},$$

definită astfel: pentru orice $x_1 \in A_{s_1}, \ldots, x_n \in A_{s_n}$,

$$\rho'(x_1, \ldots, x_n) = \begin{cases} \text{true}, & \text{dacă } (x_1, \ldots, x_n) \in \rho, \\ \text{false}, & \text{altfel.} \end{cases}$$

De acum încolo, operația ρ' va fi notată tot ρ. De exemplu, pentru programul de mai sus:
$\mathit{frati}(\mathit{ana}, \mathit{carmen}) = \text{true}$,
$\mathit{frati}(\mathit{ana}, \mathit{eva}) = \text{false}$,
pentru orice $X \in A$, $\mathit{stramos}(X, X, 0) = \text{true}$, iar $\mathit{stramos}(X, X, 1) = \text{false}$.
Iar, cum **constantele** sunt *operații zeroare (nulare)*, i.e. operații de aritate 0, operații fără argumente, nu e nevoie să facem distincție între constante și operații.

Ce este o *algebră multisortată* înzestrată numai cu OPERAŢII? Aceasta este noțiunea propriu–zisă de algebră multisortată.

Definiție

Pentru orice mulțime **nevidă** S, o *algebră S–sortată* este o structură algebrică $\mathcal{M} = ((M_s)_{s \in S}; (f_i)_{i \in I})$, unde:

- elementele lui S se numesc *sorturi* (repet că *sorturile* sunt indici corespunzători TIPURILOR DE ELEMENTE, TIPURILOR DE DATE);
- mulțimea suport, i.e. mulțimea *elementelor* algebrei $\mathcal{M}$ este *mulțimea S–sortată*, adică familia de mulțimi indexată de S $(M_s)_{s \in S}$; pentru fiecare $s \in S$, elementele lui M_s se numesc *elementele lui $\mathcal{M}$ de sort s*;
- $(f_i)_{i \in I}$ este familia *operațiilor* lui $\mathcal{M}$; pentru fiecare $i \in I$, există $n_i \in \mathbb{N}$ și sorturile $s_{i,1}, \ldots, s_{i,n_i}, s_i \in S$ astfel încât $f_i : M_{s_{i,1}} \times \ldots \times M_{s_{i,n_i}} \to M_{s_i}$: f_i este o operație de aritate n_i a lui $\mathcal{M}$; dacă $n_i = 0$, atunci această declarație a lui f_i devine: $f_i \in M_{s_i}$ (a se revedea discuția dintr–un curs anterior privind operațiile zeroare).

Așadar ce structură algebrică multisortată corespunde programului anterior în Prolog? Cum am procedat și mai sus, să scriem operațiile (acestea incluzând relațiile, predicatele) descrescător după arități.

(Algebra trisortată definită în programul în Prolog de mai sus)

$\mathcal{A} = ((A, \mathbb{N}, \{\text{false}, \text{true}\}); \mathit{stramos}, +, \mathit{frati}, \mathit{parinte}, \mathit{tata}, \mathit{mama}, \mathit{unchi}, \mathit{matusa}, \mathit{bunic}, \mathit{femeie}, \mathit{barbat}, \mathit{ana}, \mathit{carmen}, \mathit{elena}, \mathit{eva}, \mathit{maria}, \mathit{sara}, \mathit{valentina}, \mathit{adam}, \mathit{constantin}, \mathit{eugen}, \mathit{george}, \mathit{ion}, \mathit{iosif}, \mathit{tudor}, \mathit{victor}, 0, 1, \text{false}, \text{true})$, unde:

- $A = \{\mathit{ana}, \mathit{carmen}, \mathit{elena}, \mathit{eva}, \mathit{maria}, \mathit{sara}, \mathit{valentina}, \mathit{adam}, \mathit{constantin}, \mathit{eugen}, \mathit{george}, \mathit{ion}, \mathit{iosif}, \mathit{tudor}, \mathit{victor}\}$;
- $\mathit{stramos} \subseteq A \times A \times \mathbb{N}$ (relație ternară, predicat ternar), așadar, ca operație ternară: $\mathit{stramos} : A \times A \times \mathbb{N} \to \{\text{false}, \text{true}\}$;
- $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ (operație binară);
- $\mathit{frati}, \mathit{parinte}, \mathit{tata}, \mathit{mama}, \mathit{unchi}, \mathit{matusa}, \mathit{bunic} \subseteq A \times A$ (relații binare, predicate binare), așadar, ca operații binare: $\mathit{frati}, \mathit{parinte}, \mathit{tata}, \mathit{mama}, \mathit{unchi}, \mathit{matusa}, \mathit{bunic} : A \times A \to \{\text{false}, \text{true}\}$;
- $\mathit{femeie}, \mathit{barbat} \subseteq A$ (relații unare, predicate unare), așadar, ca operații unare: $\mathit{femeie}, \mathit{barbat} : A \to \{\text{false}, \text{true}\}$;
- $\mathit{ana}, \mathit{carmen}, \mathit{elena}, \mathit{eva}, \mathit{maria}, \mathit{sara}, \mathit{valentina}, \mathit{adam}, \mathit{constantin}, \mathit{eugen}, \mathit{george}, \mathit{ion}, \mathit{iosif}, \mathit{tudor}, \mathit{victor} \in A$ (constante);
- $0, 1 \in \mathbb{N}$ (constante);
- $\text{false}, \text{true} \in \{\text{false}, \text{true}\}$ (constante);

operațiile nonnulare (i.e. cu argumente) ale algebrei $\mathcal{A}$ sunt definite astfel:

operația binară $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ este predefinită în Prolog;

$$femeie(x) = \begin{cases} \text{true}, & x \in \{ana, carmen, elena, eva, maria, sara, valentina\}, \\ \text{false}, & x \in A \setminus \{ana, carmen, elena, eva, maria, sara, valentina\}; \end{cases}$$

$$barbat(x) = \begin{cases} \text{true}, & x \in \{adam, constantin, eugen, george, ion, iosif, tudor, victor\}, \\ \text{false}, & x \in A \setminus \{adam, constantin, eugen, george, ion, iosif, tudor, victor\}; \end{cases}$$

pentru orice $u \in A \times A$, $parinte(u) =$

$$\begin{cases} \text{true}, & u \in \{(sara, adam), (sara, eva), (constantin, iosif), (constantin, elena), \\ & (valentina, ion), (valentina, sara), (victor, constantin), (victor, maria), \\ & (ana, victor), (ana, valentina), (carmen, victor), (carmen, valentina), \\ & (george, victor), (george, valentina), (eugen, tudor), (eugen, ana)\}, \\ \text{false}, & \text{altfel}; \end{cases}$$

pentru orice $(x, y) \in A^2$, (eliminând, conform convenției folosite uzual, o pereche de paranteze din scrierea $frati((x, y))$, și la fel mai jos)

$$frati(x, y) = \begin{cases} \text{true}, & x \neq y \text{ și } (\exists\, p \in A)\,(parinte(x, p) = parinte(y, p) = \text{true}), \\ \text{false}, & \text{altfel}; \end{cases}$$

altfel scris, cu notațiile obișnuite $\wedge$ pentru conjuncția logică și $\neg$ pentru negația logică, $frati(x, y) = \neg(x = y) \wedge \bigwedge_{p \in A} (parinte(x, p) \wedge parinte(y, p))$;

în același mod pot fi scrise definițiile următoarelor predicate: pentru orice

$(x, y) \in A^2$, $tata(x, y) = parinte(x, y) \wedge barbat(y)$,
$mama(x, y) = parinte(x, y) \wedge femeie(y)$,
$unchi(x, y) = \bigwedge_{p \in A} (parinte(x, p) \wedge frati(p, y) \wedge barbat(y))$,
$matusa(x, y) = \bigwedge_{p \in A} (parinte(x, p) \wedge frati(p, y) \wedge femeie(y))$ și
$bunic(x, y) = \bigwedge_{p \in A} (parinte(x, p) \wedge parinte(p, y))$;
iar, dacă analizăm definiția predicatului *stramos*, observăm că poate fi scrisă în modul următor: pentru orice $x, y \in A$, $stramos(x, y, 0) = \begin{cases} \text{true}, & x = y, \\ \text{false}, & x \neq y, \end{cases}$ și,
pentru orice $g \in \mathbb{N}^*$, $stramos(x, y, g) = \bigwedge(parinte(x, p) \wedge stramos(p, y, g - 1))$.

Dacă adoptăm prima soluție, a operațiilor definite parțial, atunci structura algebrică definită de programul de mai sus devine o algebră monosortată, i.e. cu o unică mulțime suport (nu familie de mulțimi ca suport):

(algebră cu unica mulțime de elemente $A \cup \mathbb{N} \cup \{\text{false}, \text{true}\}$)

$\mathcal{A}' = (A' = A \cup \mathbb{N} \cup \{\text{false}, \text{true}\}; stramos, +, frati, parinte, tata, mama, unchi, matusa, bunic, femeie, barbat, ana, carmen, elena, eva, maria, sara, valentina, adam, constantin, eugen, george, ion, iosif, tudor, victor, 0, 1, \text{false}, \text{true})$, unde:

- $A = \{ana, carmen, elena, eva, maria, sara, valentina, adam, constantin, eugen, george, ion, iosif, tudor, victor\}$, așadar mulțimea elementelor lui $\mathcal{A}'$ este $A' = \{ana, carmen, elena, eva, maria, sara, valentina, adam, constantin, eugen, george, ion, iosif, tudor, victor, \text{false}, \text{true}\} \cup \mathbb{N}$;
- $stramos \subseteq (A')^3 = A' \times A' \times A'$ (relație ternară, predicat ternar), așadar, ca operație ternară parțială: $stramos : (A')^3 \dashrightarrow A'$;
- $+ : A' \times A' \dashrightarrow A'$ (operație binară parțială);
- $frati, parinte, tata, mama, unchi, matusa, bunic \subseteq A' \times A'$ (relații binare, predicate binare), așadar, ca operații binare parțiale: $frati, parinte, tata, mama, unchi, matusa, bunic : A' \times A' \dashrightarrow A'$;
- $femeie, barbat \subseteq A'$ (relații unare, predicate unare), așadar, ca operații unare parțiale: $femeie, barbat : A' \dashrightarrow A'$;
- $ana, carmen, elena, eva, maria, sara, valentina, adam, constantin, eugen, george, ion, iosif, tudor, victor, 0, 1, \text{false}, \text{true} \in A'$ (constante);

operațiile nonnulare (i.e. cu argumente, de aritate nonzero) ale algebrei $\mathcal{A}'$ sunt definite ca în cazul algebrei trisortate $\mathcal{A}$, și rămân nedefinite în celelalte elemente ale domeniului lor ($(A')^2$, $A' \times A'$ sau A', după cum sunt ternare, binare, respectiv unare).

Cum răspunde Prologul la următoarea interogare?

?- $stramos(carmen, S, 3)$, $barbat(S)$.

Prologul trebuie să găsească valorile variabilei S pentru care $stramos(carmen, S, 3) \wedge barbat(S) = \text{true}$, unde $\wedge$ este operația booleană de conjuncție pe mulțimea $\{\text{false}, \text{true}\}$ a valorilor booleene predefinite în Prolog.
Respectând notația pentru variabile din Prolog și fără a intra în detalii (pentru acestea a se vedea al doilea seminar), Prologul execută următorii pași:
$stramos(carmen, S, 3)$ nu unifică, cu $femeie(X)$, $barbat(X)$, $frati(X, Y)$, $parinte(X, Y)$, $tata(X, Y)$, $mama(X, Y)$, $unchi(X, Y)$, $matusa(X, Y)$ sau $bunic(X, Y)$, pentru că predicatul *stramos* nu coincide cu niciunul dintre predicatele *femeie*, *barbat*, *frati*, *parinte*, *tata*, *mama*, *unchi*, *matusa*, *bunic*;
$stramos(carmen, S, 3)$ nu unifică, cu $stramos(X, X, 0)$, pentru că, constantele 0 și 3 nu coincid;
$stramos(carmen, S, 3)$ unifică, cu $stramos(X, S, G)$ pentru $X = carmen$, așadar următorul scop compus de satisfăcut este:

?- $parinte(carmen, P)$, $stramos(P, S, H)$.

$parinte(carmen, P)$ unifică numai cu $parinte(carmen, victor)$ și $parinte(carmen, valentina)$, pentru $P = victor$, respectiv $P = valentina$.
Aceste două clauze apar ca fapte în programul Prolog de mai sus, așadar nu se intră în alte recursii după aceste unificări, ci se continuă cu satisfacerea scopurilor

stramos(*victor*, S, H), respectiv *stramos*(*valentina*, S, H) (desigur, pe rând). Acestea unifică, cu *stramos*(*victor*, *victor*, 0), respectiv *stramos*(*valentina*, *valentina*, 0), dar:

- *barbat*(*valentina*) nu unifică, cu niciunul dintre faptele care definesc predicatul *barbat* și, desigur, cu niciun alt fapt sau membru stâng de regulă,
- în timp ce *barbat*(*victor*) e satisfăcut, pentru că este unul dintre faptele din acest program, însă $0 + 1 = 1 \neq 3$, așadar *stramos*(*carmen*, S, 3) nu unifică, cu *stramos*(*carmen*, *victor*, 1).

Se continuă cu satisfacerea scopurilor *stramos*(*victor*, S, H) și *stramos*(*valentina*, S, H). Acestea au fost deja unificate cu clauza din faptul *stramos*(X, X, 0); acum sunt unificate cu membrul stâng al regulii din definiția predicatului *stramos*.

Și recursia continuă în acest mod. Sunt satisfăcute *parinte*(*victor*, *constantin*), *parinte*(*victor*, *maria*), *parinte*(*valentina*, *ion*) și *parinte*(*valentina*, *sara*), precum și *stramos*(*constantin*, *constantin*, 0), *stramos*(*maria*, *maria*, 0), *stramos*(*ion*, *ion*, 0) și *stramos*(*sara*, *sara*, 0), dar nu și *barbat*(*maria*) sau *barbat*(*sara*), iar $0 + 1 = 1$ și $1 + 1 = 2 \neq 3$.

parinte(*maria*, P) și *parinte*(*ion*, P) nu sunt satisfăcute, pentru că acestea nu unifică, cu niciun fapt sau membru stâng al unei reguli.

Dar sunt satisfăcute *parinte*(*constantin*, *iosif*), *parinte*(*constantin*, *elena*), *parinte*(*sara*, *adam*) și *parinte*(*sara*, *eva*), precum și *stramos*(*iosif*, *iosif*, 0), *stramos*(*elena*, *elena*, 0), *stramos*(*adam*, *adam*, 0) și *stramos*(*eva*, *eva*, 0).

barbat(*elena*) și *barbat*(*eva*) nu sunt satisfăcute, dar *barbat*(*iosif*) și *barbat*(*adam*) sunt satisfăcute, iar $0 + 1 = 1$, $1 + 1 = 2$ și $2 + 1 = 3$.
Așadar răspunsurile (dacă le cerem pe amândouă, cu ";" /"Next") Prologului la interogarea:

?- *stramos*(*carmen*, S, 3), *barbat*(S).

vor fi:

S = *iosif* și S = *adam*.

Cum sunt efectuate unificările de mai sus?
Prologul aplică **algoritmul de unificare**.
Amintesc că și **constantele** sunt **operații**, anume *operații fără argumente*, *operații cu 0 argumente*, *operații de aritate 0*, numite și *operații zeroare* sau *nulare*, așadar nu trebuie tratate ca un caz separat în cele ce urmează.
Și, referitor la terminologia din secțiunea recapitulativă care deschide acest curs, pentru că **predicatele**, i.e. **relațiile**, sunt considerate OPERAŢII AVÂND REZULTATUL BOOLEAN, **formulele atomice** vor fi considerate tot **termeni**, având OPERAŢIA DOMINANTĂ DE REZULTAT BOOLEAN.
În esență, pentru orice $n, k \in \mathbb{N}$, orice operații (incluzând aici predicatele, relațiile) f și g de arități n, respectiv k și orice termeni (care pot avea operația dominantă de rezultat boolean, i.e. predicat, relație) $t_1, \ldots, t_n, u_1, \ldots, u_k$:

(în esență, algoritmul de unificare funcționează astfel)

termenii $f(t_1, \ldots, t_n)$ și $g(u_1, \ldots, u_k)$ unifică ddacă: $\begin{cases} n = k, \\ t_1 \text{ și } u_1 \text{ unifică}, \\ \vdots \\ t_n \text{ și } u_n \text{ unifică}, \end{cases}$

și această recursie continuă până la unificări de termeni cu termeni care sunt:

- fie variabile, iar acestea unifică, cu orice termen care nu le conține,
- fie constante, iar acestea sunt operații fără argumente, așadar, conform regulii de mai sus, nu unifică decât cu ele însele.

Exemplu

$stramos(carmen, S, 3)$:

- nu unifică, cu $bunic(X, Y)$, pentru că $stramos \neq bunic$;
- nu unifică, cu $stramos(X, X, 0)$, pentru că $3 \neq 0$;
- unifică, cu $stramos(X, X, Y)$ pentru $carmen = X = S$ și $Y = 3$;
- nu unifică, cu $stramos(X, X, X)$ pentru că, dacă ar avea loc această unificare, atunci am avea unificările $carmen = X$, $S = X$ și $3 = X$, deci $X = carmen = S = 3$, așadar constantele $carmen$ și 3 ar trebui să unifice, dar $carmen \neq 3$.

Să adaptăm cadrul de lucru din prima secțiune a acestui curs la cadrul de lucru pentru programarea în Prolog: vom considera o **structură de ordinul I** $\mathcal{A}$, fie multisortată, fie cu operațiile parțial definite, și în care relațiile (predicatele) sunt considerate operații având rezultatul **true** sau **false**, iar constantele sunt operațiile zeroare, astfel că $\mathcal{A}$ este înzestrată doar cu o familie $(f_i)_{i \in I}$ de operații.

Notație

Pentru fiecare $i \in I$, să notăm cu $dom(f_i)$ domeniul funcției f_i, adică:

1. dacă, pentru o mulțime nevidă S, $\mathcal{A}$ este o algebră S–sortată cu mulțimea suport $A = (A_s)_{s \in S}$ (mulțime S–sortată, adică familie de mulțimi indexată de S), atunci, pentru fiecare $i \in I$, există $n_i \in \mathbb{N}$ și $s_{i,1}, \ldots, s_{i,n_i}, s_i \in S$ astfel încat $f_i : A_{s_{i,1}} \times \ldots \times A_{s_{i,n_i}} \to A_{s_i}$; atunci $\underline{\underline{dom(f_i) = A_{s_{i,1}} \times \ldots \times A_{s_{i,n_i}}}}$;
2. dacă $\mathcal{A}$ este o algebră monosortată cu mulțimea suport A (monosortată, i.e. o unică mulțime) și cu operațiile parțial definite, atunci, pentru fiecare $i \in I$, există $n_i \in \mathbb{N}$ astfel încat $f_i : A^{n_i} \dashrightarrow A$; atunci $\underline{\underline{dom(f_i)}}$ este mulțimea elementelor produsului cartezian A^{n_i} în care funcția parțială f_i este definită.

Notăm cu $dom(\mathcal{A})$ mulțimea elementelor lui $\mathcal{A}$, indiferent de sort/tip de date, i.e.:

1. în cazul S–sortat de mai sus, $dom(\mathcal{A}) = \bigcup_{s \in S} A_s$;
2. în cazul monosortat de mai sus, $dom(\mathcal{A}) = A$.

Observație (putem pune tipurile de date laolaltă, la fel ca în Prolog, unde toate sunt înglobate în tipul TERMEN)

Cazul multisortat poate fi înglobat în cazul monosortat cu operații parțiale, astfel: pentru orice $i \in I$, dacă f_i are aritatea $n_i \in \mathbb{N}^*$, atunci considerăm

$$f_i : dom(\mathcal{A})^{n_i} \multimap dom(\mathcal{A}),$$

f_i definită numai pe elementele lui $dom(f_i)$.

În cele ce urmează, vom lucra cu algebre monosortate înzestrate cu operații parțiale.

Definiție

Tipul sau *signatura* structurii algebrice $\mathcal{A}$ este $\tau = (n_i)_{i \in I} \subseteq \mathbb{N}$, unde, pentru fiecare $i \in I$, n_i este aritatea lui f_i.

Pentru fiecare $i \in I$, vom considera un **simbol de operație** n_i–ară φ_i. Elementele mulțimii $\{\varphi_i \mid i \in I\}$ le considerăm **două câte două distincte**.

Definiție

O *structură algebrică de signatură* τ este o algebră $\mathcal{M} = (M; (\varphi_i^{\mathcal{M}})_{i \in I})$, unde, pentru fiecare $i \in I$, $\varphi_i^{\mathcal{M}} : M^{n_i} \multimap M$.

Exemplu

Astfel, algebra $\mathcal{A}$ de mai sus este algebra de signatură τ $(dom(\mathcal{A}); (\varphi_i^{\mathcal{A}})_{i \in I})$, unde, pentru fiecare $i \in I$, $\varphi_i^{\mathcal{A}} = f_i$.

Să considerăm și o mulțime infinită *Var* de **variabile**, ale cărei elemente le considerăm **două câte două distincte**.
De asemenea, considerăm mulțimea de variabile *disjunctă* de cea a simbolurilor de operații: $Var \cap \{\varphi_i \mid i \in I\} = \emptyset$.
Deocamdată, vom considera doar **alfabetul** format din **variabile**, **simboluri de operații**, virgula și parantezele rotunde: $Var \cup \{\varphi_i \mid i \in I\} \cup \{,\} \cup \{(,)\}$.

Definiție și notație (termenii peste alfabetul de mai sus)

Considerăm următoarea mulțime de cuvinte finite și nevide peste alfabetul de mai sus: $Term \subseteq (Var \cup \{\varphi_i \mid i \in I\} \cup \{,\} \cup \{(,)\})^+ = \{a_1 \dots a_n \mid n \in \mathbb{N}^*, a_1, \dots, a_n \in Var \cup \{\varphi_i \mid i \in I\} \cup \{,\} \cup \{(,)\}\}$, ale cărei elemente le vom numi *termeni*, definită recursiv în modul următor:

1. $Var \subseteq Term$ (variabilele sunt termeni de lungime 1, i.e. constând dintr–o singură literă: variabila respectivă);
2. pentru orice $i \in I$, dacă $n_i = 0$ (adică φ_i este un *simbol de operație zeroară*, un *simbol de constantă*, atunci $\varphi_i \in Term$ (simbolurile de constante sunt termeni, tot de lungime 1, constând dintr–o singură literă, anume acel simbol de constantă);
3. pentru orice $i \in I$, dacă $n_i \in \mathbb{N}^*$ (adică φ_i este un simbol de operație cu $n_i \neq 0$ argumente), atunci, pentru orice $t_1, \dots, t_{n_i} \in Term$, rezultă că $\varphi_i(t_1, \dots, t_{n_i}) \in Term$ (orice simbol de operație cu argumente aplicată unor termeni are drept rezultat un termen); termenii formați în acest al treilea mod se numesc *termeni compuși*.

Observație

Cum variabilele și simbolurile de operații sunt două câte două distincte, rezultă că orice termen are o unică scriere ca:

- variabilă,
- simbol de constantă sau
- termen compus, de forma $\varphi(t_1, \ldots, t_n)$, cu simbolul de operație φ, $n \in \mathbb{N}^*$ și termenii $t_1, \ldots, t_n$ unic determinați.

Definiție și notație

Pentru orice termen $t \in Term$, notăm cu $V(t)$ *mulțimea variabilelor care apar în* t, definită, recursiv, astfel:

- oricare ar fi $v \in Var$, $V(v) = \{v\}$;
- pentru orice $i \in I$ cu $n_i = 0$, $V(\varphi_i) = \emptyset$ (i.e. simbolurile de constante nu au variabile);
- pentru orice $i \in I$ cu $n_i \in \mathbb{N}^*$ și orice termeni $t_1, \ldots, t_{n_i} \in Term$, $V(\varphi_i(t_1, \ldots, t_{n_i})) = V(t_1) \cup \ldots \cup V(t_{n_i})$.

Definiție (substituțiile dau variabilelor valori în mulțimea termenilor)

O *substituție* este o funcție $\sigma : Var \to Term$.

Notație

Dacă $n \in \mathbb{N}^*$, $\{v_1, \ldots, v_n\} \subseteq Var$ este o mulțime finită de variabile, iar $\{t_1, \ldots, t_n\} \subseteq Term$ este o mulțime de termeni, atunci se notează cu $\{v_1/t_1, \ldots, v_n/t_n\}$ următoarea substituție:

$$\{v_1/t_1, \ldots, v_n/t_n\} : Var \to Term, \text{ pentru orice } v \in Var,$$

$$\{v_1/t_1, \ldots, v_n/t_n\}(v) = \begin{cases} t_i, & (\exists\, i \in \overline{1,n})\,(v = v_i), \\ v, & v \in Var \setminus \{v_1, \ldots, v_n\}. \end{cases}$$

Definiție și notație

Fie $\sigma : Var \to Term$ o substituție. Următoarea extindere a lui σ la întreaga mulțime a termenilor se numește tot *substituție* (și, de obicei, se notează tot cu σ): $\widetilde{\sigma} : Term \to Term$, definită, recursiv, astfel:

- pentru orice $v \in Var$, $\widetilde{\sigma}(v) = \sigma(v)$ (adică $\widetilde{\sigma} \mid_{Var} = \sigma$: pe variabile, $\widetilde{\sigma}$ coincide cu σ);
- pentru orice $i \in I$ astfel încât $n_i = 0$, $\widetilde{\sigma}(\varphi_i) = \varphi_i$ (adică, pe simbolurile de constante, $\widetilde{\sigma}$ este funcția identică);
- pentru orice $i \in I$ astfel încât $n_i \in \mathbb{N}^*$ și orice termeni $t_1, \ldots, t_{n_i} \in Term$ (astfel încât $\widetilde{\sigma}$ a fost definită în $t_1, \ldots, t_{n_i}$), $\widetilde{\sigma}(\varphi_i(t_1, \ldots, t_{n_i})) = \varphi_i(\widetilde{\sigma}(t_1), \ldots, \widetilde{\sigma}(t_n))$.

Observație

Extinderea $\widetilde{\sigma}$ din definiția anterioară este:

- **complet definită**, pentru că toți termenii se scriu ca mai sus, i.e. sunt obținuți prin acea recursie;
- **corect definită**, pentru că orice termen are o unică scriere ca mai sus.

Definiție (să dăm variabilelor valori într–o algebră de signatură τ, apoi să calculăm valorile tuturor termenilor pe baza celor date variabilelor – reținem această definiție pentru mai târziu)

Pentru orice algebră $\mathcal{M} = (M; (\varphi_i^{\mathcal{M}})_{i \in I})$ de signatură τ, o funcție $s : Var \to M$ va fi numită *interpretare*.
Următoarea prelungire a lui s la mulțimea $Term$ a tuturor termenilor se numește tot *interpretare* și se notează, de obicei, tot cu s: $\widetilde{s} : Term \to M$, definită, recursiv, astfel:

- pentru orice $v \in Var$, $\widetilde{s}(v) = s(v)$ (i.e. $\widetilde{s}\,|_{Var} = s$);
- pentru orice $i \in I$ cu $n_i = 0$, $\widetilde{s}(\varphi_i) = \varphi_i^{\mathcal{M}}$ (valoarea lui $\widetilde{s}$ într–un simbol de constantă este constanta din $\mathcal{M}$ corespunzătoare acelui simbol de constantă);
- pentru orice $i \in I$ cu $n_i \in \mathbb{N}^*$ și orice $t_1, \ldots, t_{n_i} \in Term$ astfel încât ($\widetilde{s}$ a fost definită în termenii $t_1, \ldots, t_{n_i}$ și) $(\widetilde{s}(t_1), \ldots, \widetilde{s}(t_{n_i})) \in dom(\varphi_i^{\mathcal{M}})$, $\widetilde{s}(\varphi_i(t_1, \ldots, t_{n_i})) = \varphi_i^{\mathcal{M}}(\widetilde{s}(t_1), \ldots, \widetilde{s}(t_{n_i}))$.

Algoritm pentru rezolvarea unei probleme de unificare

Definiție (ce înseamnă a unifica mai mulți termeni)

Fie $k \in \mathbb{N} \setminus \{0, 1\}$ și $t_1, \ldots, t_k \in \mathit{Term}$. Spunem că termenii $t_1, \ldots, t_k$ *unifică* ddacă există o substituție $\sigma : \mathit{Term} \to \mathit{Term}$, numită *unificator pentru* $t_1, \ldots, t_k$, cu proprietatea că $\sigma(t_1) = \ldots = \sigma(t_k)$.

Fixăm un număr natural $k \geq 2$ și k termeni $t_1, \ldots, t_k \in \mathit{Term}$.

Cerința de a determina dacă termenii $t_1, \ldots, t_k$ unifică și, în caz afirmativ, a determina și un unificator pentru acești termeni este o *problemă de unificare*.

Observație

În cazul în care $t_1, \ldots, t_k$ unifică, următorul algoritm obține, într–un număr finit de pași, *un cel mai general unificator pentru* $t_1, \ldots, t_k$, adică un unificator $\mu : \mathit{Term} \to \mathit{Term}$ cu proprietatea că orice unificator $\sigma : \mathit{Term} \to \mathit{Term}$ pentru $t_1, \ldots, t_k$ este de forma $\sigma = \lambda \circ \mu$ pentru o substituție $\lambda : \mathit{Term} \to \mathit{Term}$.
În cazul în care $t_1, \ldots, t_k$ nu unifică, algoritmul următor se termină într–un număr finit de pași cu EŞEC, i.e. fără a găsi un unificator.

(Algoritmul de unificare)

Vom reține două liste (mulțimi) de *ecuații* (*egalități*, *probleme de unificare între câte doi termeni*):

o *listă soluție* S și o *listă de rezolvat* R.

Inițial:

- $S = \emptyset$;
- $R = \{t_1 = t_2, t_2 = t_3, \ldots, t_{k-1} = t_k\}$.

Se aplică, pe rând, următorii pași, în orice ordine posibilă:

- SCOATERE (ELIMINARE, ŞTERGERE): orice ecuație de forma $t = t$ (i.e. între un termen și el însuși) este eliminată din lista R;
- DESCOMPUNERE: orice ecuație de forma $\varphi_i(u_1, \ldots, u_{n_i}) = \varphi_i(w_1, \ldots, w_{n_i})$, unde $i \in I$ astfel încât $n_i > 0$ și $u_1, \ldots, u_{n_i}, w_1, \ldots, w_{n_i} \in Term$ din lista R este înlocuită cu următoarele n_i ecuații: $u_1 = w_1, \ldots, u_{n_i} = w_{n_i}$;
- REZOLVARE: orice ecuație din R de forma $v = t$ sau $t = v$, unde $v \in Var$ și $t \in Term$, este scoasă din R și introdusă în lista soluție S sub forma $v = t$ (dacă și $t \in Var$, atunci nu contează cum orientăm ecuația: putem alege pe oricare dintre variabile ca membru stâng), apoi toate aparițiile lui v în termenii care apar în ecuațiile din lista de rezolvat R și din lista soluție S se înlocuiesc cu t, adică toți acești termeni u se înlocuiesc cu $\{v/t\}(u)$.

Algoritmul se ÎNCHEIE în oricare dintre cazurile următoare:

1. dacă, după execuția unui pas de SCOATERE sau REZOLVARE, obținem $R = \emptyset$, i.e. lista de rezolvat devine vidă, iar lista soluție curentă este $S = \{v_1 = u_1, \ldots, v_n = u_n\}$, atunci **un (cel mai general) unificator** pentru $t_1, \ldots, t_k$ este substituția $\{v_1/u_1, \ldots, v_n/u_n\}$: IEŞIRE CU SUCCES;
2. dacă, în cursul execuției pașilor de mai sus, apare în lista de rezolvat R:
   - fie o ecuație de forma $v = t$, cu $t \in Term$ și $v \in V(t)$ (i.e. cu variabila v apărând în t),
   - fie o ecuație de forma $\varphi_i(u_1, \ldots, u_{n_i}) = \varphi_j(w_1, \ldots, w_{n_j})$, cu $i, j \in I$, $n_i, n_j \in \mathbb{N}$ (nu neapărat nenule), $u_1, \ldots, u_{n_i}, w_1, \ldots, w_{n_j} \in Term$ și $i \neq j$, așadar cu simbolurile de operații dominante ale termenilor din cei doi membri diferite: $\varphi_i \neq \varphi_j$,

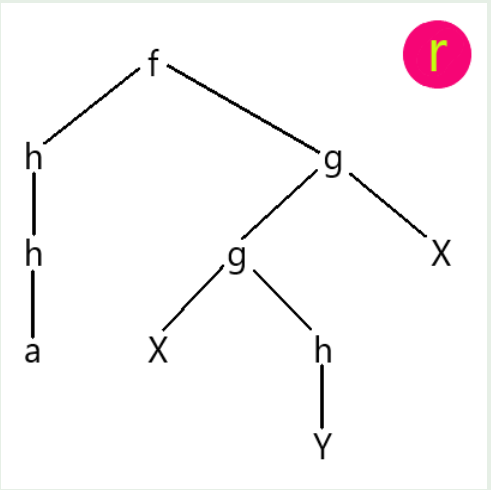
   atunci se IESE CU EŞEC: nu s–a găsit niciun unificator, așadar **termenii $t_1, \ldots, t_k$ nu unifică**.

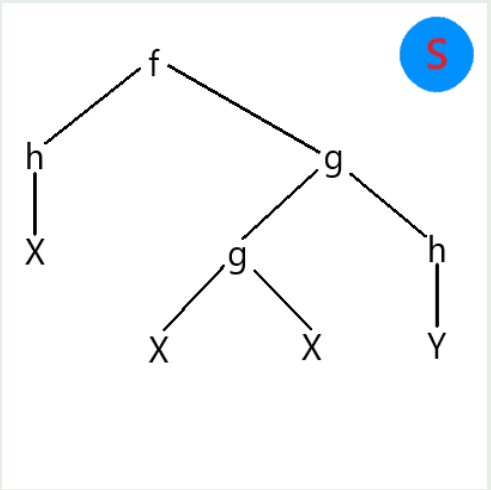
Exercițiu (temă pentru seminar)

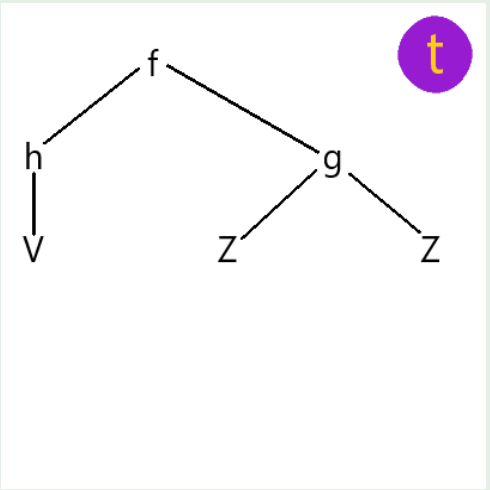
Să considerăm două simboluri de operații binare distincte f și g, unul de operație unară h, două simboluri de constante diferite a și b și patru variabile $V, X, Y, Z \in Var$, două câte două distincte.
Considerăm următorii termeni formați cu simbolurile de operații și variabilele de mai sus: $r, s, t, u, w \in Term$,

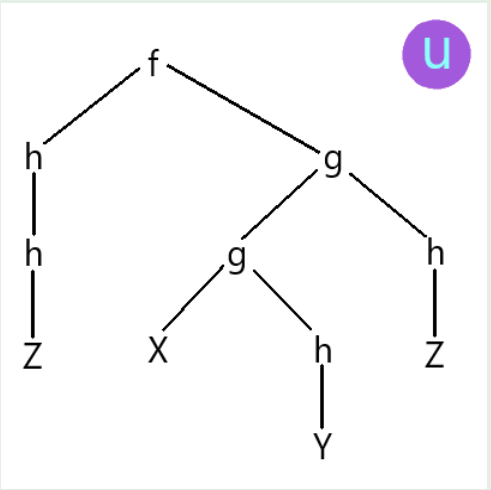
Exercițiu (continuare: termenii de unificat, dați prin expresiile lor și prin arborii asociați acestor expresii)

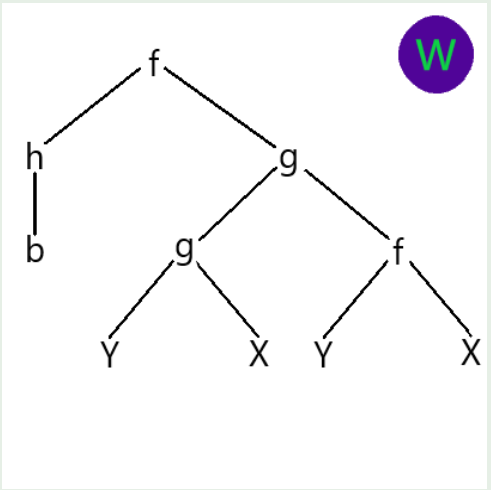
$r = f(h(h(a)), g(g(X, h(Y)), X))$,
$s = f(h(X), g(g(X, X), h(Y)))$,
$t = f(h(V), g(Z, Z))$,
$u = f(h(h(Z)), g(g(X, h(b)), h(Z)))$,
$w = f(h(b), g(g(V, Z), f(V, Z)))$.
Să se unifice acești termeni doi câte doi, adică să se rezolve, pe rând, problemele de unificare:
$r = s$, $r = t$,
$r = u$, $r = w$,
$s = t$, $s = u$,
$s = w$, $t = u$,
$t = w$, $u = w$.











Pentru perechile $\{p, q\}$ de termeni $p, q \in \{r, s, t, u, w\}$ care unifică, să se unifice și reuniunea tuturor acestor perechi.
Pentru **rezolvare**, a se vedea SEMINARUL.