BAZE DE CUNOŞTINŢE

Sinteza 2-1:

Fundamentele programelor logice

Nicolae Ţăndăreanu

Facultatea de Matematică-Informatică,

Universitatea din Craiova,

str.A.I.Cuza 13, 1100-Craiova, Romania

e-mail: ntand@oltenia.ro

1 Obiective

Obiectivele acestui capitol sunt următoarele:

- înţelegerea corectă a conceptelor fundamentale din programarea logică: termen, formulă, clauză, program logic
- formarea deprinderilor de calcul la evaluarea termenilor şi a formulelor în raport cu o structură dată

În vederea atingerii acestor obiective cursantul trebuie să înţeleagă modul în care:

- se construiesc elementele mulţimilor

$$TERM(B, \mathcal{S}_V), ATOM(B, \mathcal{S}_V), FORM(B, \mathcal{S}_V)$$

- se calculează valorile funcţiilor $eval_\Sigma$ şi $truth\_val_\Sigma$

2 Termeni, formule şi programe logice

Programarea logică se ocupă cu studiul programelor logice. Ea este strâns legată de logica matematică. Putem spune că programarea logică se ocupă cu *reprezentarea cunoştinţelor prin formule logice şi studiul metodelor de demonstrare a consecinţelor logice ale acestora.*

Pentru a defini o reprezentare şi procesare logică a cunoştinţelor este necesar să definim :

- un limbaj formal prin intermediul căruia construim formule corecte; aceasta presupune să definim *sintaxa limbajului*

- o procedură pentru calculul valorii de adevăr a unei formule, adică *semantica limbajului*

Pentru a defini sintaxa unui limbaj de ordinul I este necesar să definim :

1) Un triplet $B = (\mathcal{S}_C, \mathcal{S}_F, \mathcal{S}_P)$ numit *baza* limbajului, unde $\mathcal{S}_C, \mathcal{S}_F$ şi $\mathcal{S}_P$ sunt mulţimi finite de simboluri, disjuncte două cîte două, ale căror elemente se numesc *simboluri de constante*, *simboluri de funcţii*, respectiv *simboluri de predicate*. Fiecare element din $\mathcal{S}_F \cup \mathcal{S}_P$ are ataşat un număr natural numit *aritatea* acestuia.

2) O mulţime $\mathcal{S}_V$ finită, ale cărei elemente se numesc *simboluri de variabile*

3) O mulţime $\mathcal{S}_O$ de *simboluri de operaţii*

De cele mai multe ori aritatea unui element din mulţimea $\mathcal{S}_F \cup \mathcal{S}_P$ se notează ca exponent al simbolului respectiv, inclus între paranteze rotunde. Astfel, atunci când scriem expresia $f^{(2)} \in \mathcal{S}_F$ înţelegem că f este un simbol de funcţie de aritate 2. De asemenea, scrierea $p^{(3)} \in \mathcal{S}_P$ ne arată că avem un simbol p de predicat de aritate 3. Aritatea unui simbol este utilizată în construcţia termenilor şi a formulelor după cum se

va arăta în cele ce urmează. Deseori utilizăm următoarea convenţie de notare a constantelor, variabilelor şi a simbolurilor de funcţii şi predicate:

- constantele vor fi notate cu primele litere ale alfabetului, eventual utilizând indici:
  $a, b, c, \ldots, a_1, b_3, \ldots$
- variabilele vor fi notate cu $x, y, z, x_1, x_2, y_1, \ldots$
- simbolurile de funcţii se vor nota cu $f, g, h, f_1, g_3, h_2, \ldots$
- simbolurile de predicate vor fi notate cu $p, q, r, p_1, p_2, q_1, q_2, \ldots$

Unii autori consideră că elementele mulţimii $\mathcal{S}_C$ sunt simboluri de funcţii de aritate zero. Justificarea acestei convenţii va fi evidentă în paragraful care urmează, unde se defineşte evaluarea unui termen şi a unei formule.

Există două concepte de bază utilizate în programarea logică, care vor fi prezentate în cele ce urmează: *termenul* şi *formula*.

Definiţia 2.1 *Mulţimea $TERM(B, \mathcal{S}_V)$ a tuturor termenilor peste baza B şi mulţimea $\mathcal{S}_V$ este cea mai mică mulţime care satisface următoarele proprietăţi :*

- $\mathcal{S}_C \subseteq TERM(B, \mathcal{S}_V)$
- $\mathcal{S}_V \subseteq TERM(B, \mathcal{S}_V)$

- *Dacă* $f \in \mathcal{S}_F$ *este un simbol de functie de aritate* n *şi* $t_1, ..., t_n \in TERM(B, \mathcal{S}_V)$ *atunci* $f(t_1, ..., t_n) \in TERM(B, \mathcal{S}_V)$

Deseori vom spune despre un element din $TERM(B, \mathcal{S}_V)$ că este un termen peste $B \cup \mathcal{S}_V$.

Analizând definiţia 2.1 putem spune că termenii peste o bază B şi o mulţime de variabile $\mathcal{S}_V$ se obţin astfel:

- orice constantă este un termen
- orice variabilă este un termen
- dacă $f \in \mathcal{S}_F$ şi are aritatea n, iar $t_1, \ldots, t_n$ sunt termeni peste $B \cup \mathcal{S}_V$ atunci $f(t_1, \ldots, t_n)$ este de asemenea un termen peste $B \cup \mathcal{S}_V$
- pentru orice termen t peste $B \cup \mathcal{S}_V$ există un număr natural k, există un simbol $g \in \mathcal{S}_F$ de aritate k şi există k termeni $t_1, \ldots, t_k \in TERM(B, \mathcal{S}_V)$ astfel încât $t = g(t_1, \ldots, t_k)$

Un element din $TERM(B, \mathcal{S}_V)$ care nu conţine variabile se numeşte *termen ground*, iar mulţimea acestora o notăm cu TG_B. Observăm că în construcţia unui termen nu apar elementele lui $\mathcal{S}_P$. Într-adevăr, în definiţia 2.1 nu se utilizează simbolurile de predicate. Pentru acest motiv în exemplul care urmează nu se precizează această mulţime.

Exemplul 2.1 *Considerăm baza B formată din mulţimile $\mathcal{S}_C = \{a, b\}$, $\mathcal{S}_F = \{f^{(2)}, g^{(1)}\}$ şi mulţimea de variabile $\mathcal{S}_V = \{x, y\}$. Următoarele stringuri sunt construcţii corecte de termeni peste $B \cup \mathcal{S}_V$:*

- *a, b, x, y sunt termeni; a şi b sunt termeni ground*
- *$f(x, g(a))$, $g(f(b, b))$, $f(y, x)$, $g(f(x, x))$ sunt termeni*
- *$f(g(f(a, b)), a)$, $g(g(b))$, $f(f(a, b), f(a, a))$ sunt termeni ground*

iar următoarele construcţii nu ne dau termeni corect constituiţi peste $B \cup \mathcal{S}_V$:

$$f(x, y, x),\ g(f(a, b), a),\ f(x, z),\ h(x, g(a)),\ f(c, g(x))$$

pentru motive cum sunt: nu se respectă aritatea unui simbol de funcţie aşa cum este ea precizată în definirea mulţimii $\mathcal{S}_F$, se utilizează simboluri de variabile care nu sunt în mulţimea $\mathcal{S}_V$, se utilizează simboluri de funcţii care nu sunt în $\mathcal{S}_F$ sau se utilizează constante care nu apar în $\mathcal{S}_C$.

Vom considera mulţimea $\mathcal{S}_O = \{\neg, \vee, \wedge, \rightarrow, \longleftrightarrow\}$, ale carei elemente vor reprezenta respectiv *negaţia, disjuncţia, conjuncţia, implicaţia* şi *echivalenţa*. Elementele acestei mulţimi se utilizează în construcţia formulelor. Mai întâi definim conceptul de formula atomică sau atom.

Definiţia 2.2 *O* **formulă atomică** *peste $B \cup \mathcal{S}_V$ sau* **atom** *peste $B \cup \mathcal{S}_V$ este o expresie de forma $p(t_1, \ldots, t_n)$ unde $p^{(n)} \in \mathcal{S}_P$ şi $t_1, ..., t_n \in$*

$TERM(B, \mathcal{S}_V)$. *Mulţimea tuturor atomilor peste* $B \cup \mathcal{S}_V$ *o notăm cu* $ATOM(B, \mathcal{S}_V)$.

În afara elementelor din $\mathcal{S}_O$, la definirea conceptului de formulă se mai utilizează semnele $\forall$ şi $\exists$. Aceste simboluri desemnează cuantificatorul universal, respectiv existenţial.

Definiţia 2.3 *Mulţimea* $FORM(B, \mathcal{S}_V)$ *a tuturor formulelor peste* $B \cup \mathcal{S}_V$ *este cea mai mică mulţime care satisface condiţiile:*

- $ATOM(B, \mathcal{S}_V) \subseteq FORM(B, \mathcal{S}_V)$
- *dacă* $\varphi, \psi \in FORM(B, \mathcal{S}_V)$ *atunci* $(\varphi \vee \psi) \in FORM(B, \mathcal{S}_V)$ *şi* $(\neg\varphi) \in FORM(B, \mathcal{S}_V)$
- *dacă* $\varphi \in FORM(B, \mathcal{S}_V)$ *şi* $x \in \mathcal{S}_V$ *atunci*

$$(\exists x \varphi) \in FORM(B, \mathcal{S}_V)$$

Definiţia 2.4 *Un* **literal** *peste* $B \cup \mathcal{S}_V$ *este un atom peste* $B \cup \mathcal{S}_V$ *sau negaţia unui atom peste* $B \cup \mathcal{S}_V$. *Un atom se mai numeşte* **literal pozitiv**, *iar negaţia unui atom se numeşte* **literal negativ**.

Vom utiliza în continuare următoarele notaţii:

$$\varphi \wedge \psi = \neg((\neg\varphi) \vee (\neg\psi))$$

$$\varphi \rightarrow \psi = ((\neg\varphi) \vee \psi)$$

$$\varphi \longleftrightarrow \psi = (\varphi \to \psi) \wedge (\psi \to \varphi)$$

$$\forall x\varphi = \neg(\exists x(\neg\varphi))$$

Să aplicăm definiţia 2.3 pentru formulele φ şi ψ din $FORM(B, \mathcal{S}_V)$. Constatăm că $\neg\varphi$ şi $\neg\psi$ sunt de asemenea formule, deci $\varphi \wedge \psi$, $\varphi \to \psi$, $\varphi \longleftrightarrow \psi$ şi $\forall x\varphi$ sunt formule din $FORM(B, \mathcal{S}_V)$. În acest fel, formulele enumerate mai sus devin notaţii prescurtate ale altor formule.

Facem observaţia că vom utiliza convenţiile obişnuite cu privire la prioritatea unei operaţii pentru a evita utilizarea parantezelor inutile. Ordinea descrescătoare a priorităţilor simbolurilor de operaţii este următoarea:

negaţia

conjuncţia

disjuncţia

implicaţia

Astfel, în loc de $((\neg x) \to y)$ vom scrie $\neg x \to y$ deoarece negaţia este mai puternică decît implicaţia. De asemenea, scrierea $\neg p \vee q$ este echivalentă cu $(\neg p) \vee q$, iar scrierea $p \longrightarrow p \vee q$ este echivalentă cu $p \longrightarrow (p \vee q)$. Aceste convenţii se vor utiliza numai atunci când nu apare nici un pericol de confuzii. Deseori în scrierea formulelor cu cuantificatori vom utiliza parantezele pentru a înlătura orice confuzie posibilă.

Exemplul 2.2 *Să adăugăm la elementele considerate în exemplul 2.1 mulţimea* $\mathcal{S}_P = \{p^{(2)}, q^{(1)}\}$. *Conform definiţiei 2.2 următoarele elemente sunt atomi sau literali pozitivi:*

$$q(f(a,b)),\ p(f(x,y),g(x)),\ p(g(b),g(x))$$

în timp ce

$$\neg q(f(a,b)),\ \neg p(f(x,y),g(x)),\ \neg p(g(b),g(x))$$

sunt literali negativi. Expresiile

$$\forall x p(x,g(a)),\ p(x,b) \longrightarrow (\exists y q(y)),\ \neg q(f(a,b)) \vee \neg p(f(x,y),g(x))$$

sunt formule corecte în $FORM(B,\mathcal{S}_V)$, *iar următoarele expresii nu sunt formule corecte:*

$$\exists x f(x,y),\ p(\forall x,a),\ q(p(x,y))$$

deaorece cuantificatorul existenţial nu se aplică unui termen, expresia $\forall x$ *din primul argument al celei de a doua expresii nu este un termen, iar la a treia expresie simbolul de predicat* q *se aplică unui atom şi nu unui termen.*

Variabilele care apar lângă cuantificatorul universal sau existenţial se numesc *variabile legate*. Celelate variabile se numesc *variabile libere*. Astfel, în expresia $\exists x(p(f(x,b),y) \longrightarrow q(g(x)))$, variabila x este variabilă legată, iar y este variabilă liberă. Cuantificatorii se pot aplica numai variabilelor libere. În acelaşi timp, vom accepta scrieri de forma $p(x,g(y)) \wedge$

$\exists x q(x)$. Aceasta este o formulă corectă deoarece $p(x, g(y))$ şi $\exists x q(x)$ sunt formule corecte. Facem precizarea că în formula $p(x, g(y)) \wedge \exists x q(x)$ variabilele x şi y din subformula $p(x, g(y))$ sunt variabile libere şi mai mult, acestea sunt singurele variabile libere care apar. Vom conveni ca în scrierea cuantificatorilor din formule să utilizăm parantezele pentru a preciza subformula căreia i se aplică cuantificatorul. Astfel, formula $\exists x(q(x) \wedge p(x, y))$ precizează că se aplică cuantificatorul existenţial formulei $q(x) \wedge p(x, y)$. În acelaşi timp, scrierea de forma $\exists x q(x) \wedge p(x, y)$ va reprezenta notaţia fără paranteze a formulei $\exists x(q(x)) \wedge p(x, y)$.

Definiţia 2.5 *O formulă care nu are variabile libere se numeşte formulă ground.*

Se cuvine să facem următoarea precizare foarte importantă: elementele mulţimii $FORM(B, \mathcal{S}_V)$ sunt considerate în acest moment drept entităţi sintactice, adică sunt privite ca stringuri. Drept urmare, formulele $p(x, y) \vee q(g(x))$ şi $q(g(x)) \vee p(x, y)$ sunt considerate formule distincte. Acest aspect este de natură sintactică, dar din punct de vedere semantic vom vedea că cele două formule sunt echivalente, adică au acelaşi înţeles.

Ori de câte ori am fixat o bază B, o mulţime $\mathcal{S}_V$ de variabile, o mulţime $\mathcal{S}_O$ de operaţii şi o *mulţime de reguli* prin care construim termeni şi formule, spunem că am definit un limbaj de ordinul I. Din punct de vedere sintactic, limbajul definit în acest caz este format din toate formulele care se pot construi aplicând regulile limbajului.

Definiţia 2.6 *O clauză este o formulă de forma:*

$$\forall x_1 \ldots \forall x_k (A_1 \vee \ldots \vee A_m \vee \neg B_1 \vee \ldots \vee \neg B_n)$$

unde $\{A_1, \ldots, A_m, B_1, \ldots, B_n\} \subseteq ATOM(B, \mathcal{S}_V)$, *iar* $x_1, \ldots, x_k$ *sunt toate variabilele care apar în formula* $A_1 \vee \ldots \vee A_m \vee \neg B_1 \vee \ldots \vee \neg B_n$.

Pentru o clauză vom utiliza următoarea notaţie consacrată :

$$A_1 \vee \ldots \vee A_m \leftarrow B_1, ..., B_n$$

şi în această notaţie se subânţelege că toate variabilele care apar sunt cuantificate universal. Vom considera că în această scriere $A_1, \ldots, A_m$ sunt atomi, iar $B_1, \ldots, B_n$ sunt literali.

În general se utilizează următoarea terminologie:

- expresia $B_1, \ldots, B_n$ formează *corpul clauzei*;
- expresia $A_1 \vee \ldots \vee A_m$ este *capul clauzei*;
- B_i este un *subscop*, $i \in \{1, \ldots, n\}$

Definiţia 2.7 *O clauză* $A_1 \vee \ldots \vee A_m \leftarrow B_1, ..., B_n$ *se numeste :*

- **clauză Horn** *dacă* $m \leq 1$ *şi orice subscop al ei este un atom*
- **clauză Horn generală** *dacă* $m \leq 1$
- **clauză disjunctivă pozitivă** *dacă* $m > 1$ *şi orice subscop este atom*

- **clauză disjunctivă** *dacă* $m > 1$

Pentru o clauză $A_1 \vee \ldots \vee A_m \leftarrow B_1, ..., B_n$ vom utiliza de asemenea terminologia următoare : clauza se numeşte **regulă** dacă $m \geq 1$ şi $n \geq 1$, **fapt** dacă $m \geq 1$ şi $n = 0$, **scop** dacă $m = 0$ şi $n \geq 1$.

Conceptul de clauză stă la baza noţiunii de program logic, care se defineşte astfel:

Definiţia 2.8 *Un program logic este o mulţime finită de reguli şi fapte.*

Observăm că un program logic nu poate să conţină scopuri.

După tipul clauzelor pe care le conţine, un program logic poate fi:

- **program Horn** dacă conţine numai clauze Horn
- **program Horn general** dacă conţine numai clauze Horn şi/sau clauze Horn generale
- **program disjunctiv pozitiv** dacă conţine numai clauze Horn şi/sau clauze disjunctive pozitive
- **program disjunctiv** dacă conţine clauze Horn şi/sau clauze Horn generale şi/sau clauze disjunctive

Uneori în literatura de specialitate se consideră că un **program normal** este un program Horn sau Horn general.

De cele mai multe ori vom considera un program logic P fără să specificăm explicit baza acestuia. Evident, există mai multe baze B şi mai multe mulţimi $\mathcal{S}_V$ astfel încât $P \subseteq FORM(B, \mathcal{S}_V)$. Cea mai mică bază cu această proprietate va fi considerată şi numită *baza programului* P. Completări cu privire la acest concept se găsesc în exerciţiile de la sfârşitul acestui capitol.

3 Evaluarea termenilor şi formulelor

O formulă oarecare este privită nu numai sub aspectul sintactic ci şi din punct de vedere semantic. Aspectul semantic este legat de valoarea de adevăr a formulei. Pentru a prezenta acest aspect avem nevoie de conceptul de structură, care este prezentat în definiţia care urmează.

Definiţia 3.1 *O structură Σ peste baza B este o pereche $\Sigma = (D, k)$ unde:*

- *D este o multime nevidă numită* **domeniul structurii**
- *pentru fiecare $f \in \mathcal{S}_F$,*
  - *$k(f)$ este o funcţie, $k(f) : D^n \longrightarrow D$, unde n este aritatea lui f;*
- *pentru fiecare $p \in \mathcal{S}_P$,*

– $k(p)$ *este o funcţie,* $k(p) : D^n \longrightarrow \{true, false\}$*, unde* n *este aritatea lui* p

Mulţimea tuturor structurilor peste B *o notăm cu* $STRUCT_B$*.*

Atunci când definim o structură, domeniul ei D este o mulţime efectiv definită, precizată. Prin intermediul unei structuri simbolurile de funcţii devin funcţii, iar simbolurile de predicate devin predicate. Dacă considerăm că orice element $a \in \mathcal{S}_C$ este un simbol de aritate zero atunci constatăm că $k(a)$ trebuie să fie o funcţie de aritate zero, $k(a) : D^0 \longrightarrow D$. Deoarece D^0 conţine un singur element, simbolul vid, observăm că prin intermediul unei structuri un simbol de constantă este transformat într-un element din domeniul structurii.

Definiţia 3.2 *O funcţie* $F : \mathcal{S}_V \longrightarrow D$ *se numeşte* **asignare de variabile** *sau mai simplu,* **asignare**. *Mulţimea tuturor asignărilor de variabile din* $\mathcal{S}_V$ *la mulţimea* D *se notează cu* $D^{\mathcal{S}_V}$*.*

De îndată ce avem definită o structură $\Sigma \in STRUCT_B$ şi o asignare F putem **evalua termeni** şi putem **evalua formule**. Rezultatul evaluării unui termen este un element din D, iar rezultatul evaluării unei formule este **true** sau **false**. Formalizarea conceptului de evaluare este prezentată în cele două definiţii care urmează.

Definiţia 3.3 *Fie* B *o bază,* $\Sigma = (D, k) \in STRUCT_B$ *şi* $\mathcal{S}_V$ *o mulţime*

de variabile. Definim funcţia

$$eval_\Sigma : TERM(B, \mathcal{S}_V) \times D^{\mathcal{S}_V} \longrightarrow D$$

astfel:

- $eval_\Sigma(x, F) = F(x)$ *pentru orice* $x \in \mathcal{S}_V$ *şi orice* $F \in D^{\mathcal{S}_V}$
- $eval_\Sigma(h(t_1, ..., t_n), F) = k(h)(eval_\Sigma(t_1, F), \ldots, eval_\Sigma(t_n, F))$ *pentru orice* $F \in D^{\mathcal{S}_V}$

Definiţia dată mai sus precizează explicit modul în care se evaluează o variabilă şi un termen. În mod implicit definiţia arată cum se evaluează o constantă. Într-adevăr, dacă $a \in \mathcal{S}_C$ atunci am convenit să considerăm pe a ca simbol de funcţie de aritate zero. Aceasta înseamnă că aplicând definiţia dată avem:

$$eval_\Sigma(a, F) = k(a)$$

Pe mulţimea $\{true, false\}$ definim operaţia unară $\neg true = false$ şi $\neg false = true$. Considerăm de asemenea operaţia binară $\vee$ definită în **Tabelul 1**. Elementele *true* şi *false* se numesc *valori de adevăr*.

Cu ajutorul acestor operaţii putem defini de asemenea următoarele operaţii pe mulţimea $\{true, false\}$:

$$x \wedge y = \neg(\neg x \vee \neg y)$$

$$x \rightarrow y = ((\neg x) \vee y)$$

$\vee$	$true$	$false$
$true$	$true$	$true$
$false$	$true$	$false$

Tabelul 1: Operaţia $\vee$

$$x \longleftrightarrow y = (x \to y) \wedge (y \to x)$$

Definiţia 3.4 *Fie B o bază, $\Sigma = (D, k) \in STRUCT_B$ şi $\mathcal{S}_V$ o mulţime de variabile. Definim funcţia*

$$truth\_val_\Sigma : FORM(B, \mathcal{S}_V) \times D^{\mathcal{S}_V} \longrightarrow \{true, false\}$$

astfel:

(1) $truth\_val_\Sigma(p(t_1, ..., t_n), F) =$

$$k(p)(eval_\Sigma(t_1, F), \ldots, eval_\Sigma(t_n, F))$$

pentru fiecare atom $p(t_1, \ldots, t_n) \in ATOM(B, \mathcal{S}_V)$

(2) fie $\varphi, \psi \in FORM(B, \mathcal{S}_V)$*;*

a) $truth\_val_\Sigma(\neg\varphi, F) = \neg(truth\_val_\Sigma(\varphi, F))$

b) $truth\_val_\Sigma(\varphi \vee \psi, F) = truth\_val_\Sigma(\varphi, F) \vee$

$$truth\_val_\Sigma(\psi, F)$$

c) $truth\_val_\Sigma(\exists x\varphi, F) = true$ *dacă şi numai dacă există* $c \in D$ *astfel încît*

$$truth\_val_\Sigma(\varphi, F\{x|c\}) = true$$

unde $F\{x|c\}$ *este asignarea obţinută din* F *în următorul mod:*

$$F\{x|c\}(y) = \begin{cases} F(y) & \text{dacă } y \neq x \\ c & \text{dacă } y = x \end{cases}$$

Definiţia 3.5 *O formulă* φ *cu proprietatea* $truth\_val_\Sigma(\varphi, F) = true$ *se numeşte* **formulă adevărată** *în structura* Σ *şi asignarea* F*. Dacă* $truth\_val_\Sigma(\varphi, F) = false$ *atunci* φ *se numeşte* **formulă falsă**.

În relaţiile *a*) şi *b*) din definiţia 3.4 apar semnele $\neg$ şi $\vee$. Evident, semnele din stânga sunt din $\mathcal{S}_O$, în timp ce semnele din dreapta reprezintă operaţiile din mulţimea $\{true, false\}$. Astfel, prin intermediul funcţiei $truth\_val_\Sigma$ orice formulă primeşte una din valorile de adevăr *true* sau *false*.

Referitor la valoarea de adevăr a unei formule în care apare cuantificatorul universal, putem demonstra următoarea proprietate:

Propoziţia 3.1 *Pentru orice* $\varphi \in FORM(B, \mathcal{S}_V)$ *şi orice asignare* $F : \mathcal{S}_V \longrightarrow D$ *avem*

$$truth\_val_\Sigma(\forall x\varphi, F) = true$$

dacă şi numai dacă pentru orice $c \in D$ *avem* $truth\_val_\Sigma(\varphi, F\{x|c\}) = true$

Proof. Deoarece formula $\forall x\varphi$ reprezintă notaţia formulei $\neg\exists x\neg\varphi$, rezultă că

$$truth\_val_\Sigma(\forall x\varphi, F) = truth\_val_\Sigma(\neg\exists x\neg\varphi, F) =$$

$$\neg truth\_val_\Sigma(\exists x\neg\varphi, F)$$

Pe de altă parte avem următorul lanţ de relaţii echivalente:

$$\neg truth\_val_\Sigma(\exists x\neg\varphi, F) = true$$

$$truth\_val_\Sigma(\exists x\neg\varphi, F) = false$$

pentru orice $c \in D$ avem $truth\_val_\Sigma(\neg\varphi, F\{x|c\}) = false$

pentru orice $c \in D$ avem $truth\_val_\Sigma(\varphi, F\{x|c\}) = true$

■

Să considerăm baza $B = (\{a\}, \{f^{(1)}\}, \{p^{(2)}, q^{(2)}\})$, $\mathcal{S}_V = \{x, y, z\}$ şi o structură $\Sigma = (D, k)$ peste baza B. Fie F o asignare oarecare de variabile. Expresiile $\alpha = p(x, y) \vee q(f(z), x)$ şi $\beta = q(f(z), x) \vee p(x, y)$ sunt formule peste $B \cup \mathcal{S}_V$. Evaluând valorile de adevăr ale acestor formule obţinem:

$$truth\_val_\Sigma(\alpha, F) =$$

$$truth\_val_\Sigma(p(x, y), F) \vee truth\_val_\Sigma(q(f(z), x), F) =$$

$$truth\_val_\Sigma(q(f(z), x), F) \vee truth\_val_\Sigma(p(x, y), F) =$$

$$truth\_val_\Sigma(\beta, F)$$

Aşadar formulele α şi β sunt distincte din punct de vedere sintactic, dar au aceleaşi valori de adevăr în raport cu orice structură Σ şi orice asignare de variabile.

Definiţia 3.6 *Fie B o bază şi $\mathcal{S}_V$ o mulţime de variabile. Pe mulţimea $FORM(B, \mathcal{S}_V)$ definim relaţia $\approx$ astfel:*

$$\alpha \approx \beta \text{ dacă } truth\_val_\Sigma(\alpha, F) = truth\_val_\Sigma(\beta, F)$$

pentru orice $\Sigma \in STRUCT_B$ şi orice asignare de variabile F.

Propoziţia 3.2 *Relaţia $\approx$ este reflexivă, simetrică şi tranzitivă, deci este o relaţie de echiva- lenţă pe mulţimea $FORM(B, \mathcal{S}_V)$.*

Demonstraţia acestei proprietăţi este evidentă pe baza proprietăţilor relaţiei de egalitate.

De cele mai multe ori, atunci când $\alpha \approx \beta$ se spune că α şi β sunt formule *logic echivalente*.

4 Entităţi Herbrand

Există două entităţi Herbrand şi ele joacă un rol deosebit de important în programarea logică: univers Herbrand şi bază Herbrand. Aceste entităţi sunt prezentate în cele ce urmează.

Definiţia 4.1 *Considerăm o bază* $B = (\mathcal{S}_C, \mathcal{S}_F, \mathcal{S}_P)$. *Mulţimea* TG_B *a termenilor ground peste* B *se numeşte* **universul Herbrand** *al lui* B *şi se notează cu* UH_B. *Mulţimea*

$$\{p(x_1, \ldots, x_n) \mid p^{(n)} \in \mathcal{S}_P; n \geq 1; x_1, \ldots, x_n \in UH_B\}$$

se notează cu BH_B *şi se numeşte* **baza Herbrand** *ataşată lui* B. *Dacă* B *este baza unui program logic* P *atunci* UH_B *se mai notează cu* UH_P *şi se numeşte* **universul Herbrand al programului** P. *În acest caz, mulţimea* BH_B *se notează cu* BH_P *şi se numeşte* **baza Herbrand a programului** P.

Exemplul 4.1 *Considerăm următorul program logic:*

$$\begin{cases} p(a, b) \leftarrow \\ q(b) \leftarrow \\ p(f(x), x) \leftarrow q(x) \end{cases}$$

Baza programului P *va fi* $B = (\mathcal{S}_C, \mathcal{S}_F, \mathcal{S}_P)$, *unde* $\mathcal{S}_C = \{a, b\}$, $\mathcal{S}_F = \{f^{(1)}\}$, $\mathcal{S}_P = \{p^{(2)}, q^{(1)}\}$. *Universul Herbrand al programului* P *este mulţimea*

$$UH_P = \{a, b, f(a), f(b), f(f(a)), f(f(b)), \ldots\}$$

Baza Herbrand este mulţimea $BH_P = \{q(x) \mid x \in UH_P\} \cup \{p(x, y) \mid x, y \in UH_P\}$.

Remarcă 4.1 *Orice element din* $\mathcal{S}_C$ *va fi considerat în cele ce urmează simbol de funcţie de aritate zero.*

5 Exemple şi aplicaţii

În logica matematică calculul propoziţional este legat de conceptul de **limbaj de ordinul zero**. Pentru a defini un asemenea limbaj din punct de vedere sintactic este necesar să precizăm:

- o mulţime de simboluri numite litere propoziţionale

- o mulţime de simboluri de operaţii

- regulile de formare corectă a formulelor

De exemplu, să considerăm mulţimea literelor propoziţionale $A = \{p, q, r\}$, mulţimea de simboluri de operaţii $O = \{\neg, \wedge\}$ şi precizăm următoarele reguli de formare a formulelor:

- orice literă propoziţională este o formulă
- dacă φ şi ψ sunt formule atunci $\neg\varphi$, $\varphi \wedge \psi$ sunt formule
- orice formulă se obţine aplicînd de un număr finit de ori regulile anterioare.

Astfel, $p, p \wedge q, \neg r \wedge \neg p, \ldots$ sunt formule corecte din punct de vedere sintactic, iar $\wedge p, s \wedge p, \ldots$ sunt formule incorecte.

Dacă dorim să introducem şi alte simboluri de operaţii, acestea se pot introduce astfel:

$$x \vee y = \neg(\neg x \wedge \neg y)$$

$$x \rightarrow y = \neg x \vee y$$

$$x \longleftrightarrow y = (x \rightarrow y) \wedge (y \rightarrow x)$$

Observăm că un limbaj de ordin zero nu are variabile şi nici cuantificatori. Lipsa cuantificatorilor se justifică prin lipsa simbolurilor de predicate.

Valoarea de adevăr a unei formule se obţine cu ajutorul unei funcţii $v : A \longrightarrow \{true, false\}$, care se extinde la funcţia $v : L \longrightarrow \{true, false\}$, unde L este limbajul tuturor formulelor corecte, astfel:

$$v(x \wedge y) = v(x) \wedge v(y)$$

$$v(\neg x) = \neg v(x)$$

Astfel, dacă luăm $v(p) = true, v(q) = false, v(r) = false$ atunci $v(p \wedge q) = v(p) \wedge v(q) = true \wedge false = false$, $v(\neg q \wedge \neg r \wedge p) = v(\neg q) \wedge v(\neg r) \wedge v(p) = \neg v(q) \wedge \neg v(r) \neg v(p) = \neg false \wedge \neg false \wedge true = true \wedge true \wedge true = true$.

Să considerăm acum un exemplu de limbaj de ordinul 1. Considerăm baza $B = (\mathcal{S}_C, \mathcal{S}_F, \mathcal{S}_P)$, unde $\mathcal{S}_C = \{a\}, \mathcal{S}_F = \{f\}, \mathcal{S}_P = \{p, q\}$, f este un simbol unar, p şi q sunt simboluri binare. Universul Herbrand UH_B este multimea

$$\{a, f(a), f(f(a)), \ldots\}$$

Notăm $f^n(a) = f(f(f \ldots (f(a)) \ldots))$, unde simbolul f apare de n ori.

Pentru a lua un exemplu de structură vom considera $D = N = \{0, 1, 2, \ldots\}$, $k(f) : N \longrightarrow N$ este functia $k(f)(n) = n + 3$, $k(p) : N \times N \longrightarrow \{true, false\}$ este definită prin :

$$k(p)(n,m) = \begin{cases} true & dacă\ n|m \\ \\ false & altfel \end{cases}$$

iar $k(q) : N \times N \longrightarrow \{true, false\}$ o definim prin:

$$k(q)(n,m) = \begin{cases} true & dacă\ n < m \\ \\ false & altfel \end{cases}$$

Mai este necesar să definim $k(a)$ şi luăm $k(a) = 1$.

Pentru a obţine termeni şi formule corecte este necesar să precizăm mulţimea $\mathcal{S}_V$ a variabilelor: vom considera $\mathcal{S}_V = \{x, y, z\}$. Putem considera următoarele exemple de termeni şi formule:

- termeni: $x, f(x), f(y), f(z), f(f(x)), f(f(f(y))), a, f(a), f(f(a))$
- formule atomice ground : $p(a, f(a)), q(f^2(a), f^5(a))$ etc.
- formule : $\forall x p(x, f(a))$, $\forall x \forall y q(x, y)$, ş.a.m.d.

Rezultatul evaluării se obţine definind o asignare de variabile

$$F : \{x, y, z\} \longrightarrow N$$

Vom lua $F(x) = 3, F(y) = 8, F(z) = 9$. Am obţinut astfel o structura $\Sigma = (N, k)$ şi o asignare F. Rezultatul evaluării unui termen va fi un număr natural. De exemplu :

$$eval_\Sigma(x, F) = 3$$

$$eval_\Sigma(f(y), F) = k(f)(eval_\Sigma(y, F)) = k(f)(F(y)) = k(f)(8) = 11$$

Rezultatul evaluării unei formule este **true** sau **false**. De exemplu :

- $truth\_val_\Sigma(p(a, a), F) =$

  $$k(p)(eval_\Sigma(a, F), eval_\Sigma(a, F)) = k(p)(1, 1) = true$$

  deoarece a este considerat simbol de funcţie de aritate zero şi prin urmare $eval_\Sigma(a, F) = k(a) = 1$

- $truth\_val_\Sigma(q(f(z), a), F) = k(q)(eval_\Sigma(f(z), F), eval_\Sigma(a, F)) =$

  $$k(q)(k(f)(eval_\Sigma(z, F)), k(a)) = k(q)(k(f)(F(z)), k(a)) =$$

  $$k(q)(k(f)(9), 1) = k(q)(12, 1) = false$$

- $truth\_val_\Sigma(\exists x p(x, f(y)), F) = true$ dacă şi numai dacă există $c \in N$ astfel încît

  $$truth\_val_\Sigma(p(x, f(y)), F\{x|c\}) = true$$

  Observăm că:

  $$truth\_val_\Sigma(p(x, f(y)), F\{x|c\}) =$$

$$k(p)(eval_\Sigma(x, F\{x|c\}), k(f)(eval_\Sigma(y, F\{x|c\}))) =$$

$$k(p)(c, k(f)(8)) = k(p)(c, 11)$$

Deoarece $k(p)(1, 11) = true$ rezultă că

$$truth\_val_\Sigma(\exists x p(x, f(y)), F) = true$$

- $truth\_val_\Sigma(\forall x p(x, f(y)), F) = false$ deoarece

$$truth\_val_\Sigma(\forall x p(x, f(y)), F) = truth\_val_\Sigma(\neg\exists x \neg p(x, f(y)), F)$$

şi $truth\_val_\Sigma(\exists x \neg p(x, f(y)), F) = true.$

Într-adevăr,

$$truth\_val_\Sigma(\exists x \neg p(x, f(y)), F) = true$$

dacă şi numai dacă există $c \in N$ astfel încât

$$truth\_val_\Sigma(\neg p(x, f(y)), F\{x|c\}) = true$$

Dar

$$truth\_val_\Sigma(\neg p(x, f(y)), F\{x|c\}) =$$

$$\neg truth\_val_\Sigma(p(x, f(y)), F\{x|c\}) =$$

$$\neg k(p)(eval_\Sigma(x, F\{x|c\}), eval_\Sigma(f(y), F\{x|c\})) =$$

$$\neg k(p)(c, k(f)(F(y))) = \neg k(p)(c, k(f)(8)) =$$

$$\neg k(p)(c, 11)$$

Aşadar $truth\_val_{\Sigma}(\exists x \neg p(x, f(y)), F) = true$ dacă şi numai dacă există $c \in N$ astfel încât $k(p)(c, 11) = false$. Dar $k(p)(c, 11) = false$ dacă şi numai dacă c nu divide pe 11. În concluzie,

$$truth\_val_{\Sigma}(\exists x \neg p(x, f(y)), F) = true$$

dacă şi numai dacă există un număr natural c care nu divide pe 11, ceea ce este adevărat.

6 TEME

Tema 1

Exerciţiul nr. 1

Fie $\mathcal{S}_C = \{a, b\}$, $\mathcal{S}_F = \{f^{(1)}, g^{(2)}\}$, $\mathcal{S}_P = \{p^{(2)}\}$. *Fie* $B = (\mathcal{S}_C, \mathcal{S}_F, \mathcal{S}_P)$. *Verificaţi că:*

a) $f(f(a)), g(f(a), g(b, b))$ *sunt termeni ground peste* B

b) $g(f(x), g(a, b))$ *este un termen peste* B

c) $\neg p(f(a), g(a, a))$ *şi* $\exists x p(x, g(x, y))$ *sunt formule peste* B

Exerciţiul nr. 2

Fie B *o bază. Arătaţi că* TG_B *este cea mai mică mulţime care satisface proprietăţile:*

- $\mathcal{S}_C \subseteq TG_B$

- *dacă* $f^{(n)} \in \mathcal{S}_F$ *şi* $t_1, \ldots, t_n \in TG_B$ *atunci* $f(t_1, \ldots, t_n) \in TG_B$.

Exerciţiul nr. 3

Considerăm un program logic P *şi notăm cu* $\mathcal{S}_V$ *mulţimea variabilelor cuantificate universal în* P. *Fie*

$$B_1 = (\mathcal{S}_{C1}, \mathcal{S}_{F1}, \mathcal{S}_{P1}), B_2 = (\mathcal{S}_{C2}, \mathcal{S}_{F2}, \mathcal{S}_{P2})$$

două baze astfel încât să avem

$$P \subseteq FORM(B_1, \mathcal{S}_V), P \subseteq FORM(B_2, \mathcal{S}_V)$$

- *Demonstraţi că* $B = (\mathcal{S}_{C1} \cap \mathcal{S}_{C2}, \mathcal{S}_{F1} \cap \mathcal{S}_{F2}, \mathcal{S}_{P1} \cap \mathcal{S}_{P2})$ *satisface proprietatea* $P \subseteq FORM(B, \mathcal{S}_V)$, *deci* B *este o bază a lui* P.

- *Definiţi relaţia* $B_1 \sqsubseteq B_2$ *dacă* $\mathcal{S}_{C1} \subseteq \mathcal{S}_{C2}$, $\mathcal{S}_{F1} \subseteq \mathcal{S}_{F2}$, $\mathcal{S}_{P1} \subseteq \mathcal{S}_{P2}$. *Arătaţi că* $\sqsubseteq$ *este o relaţie de ordine parţială.*

- *Demonstraţi că există cea mai mică bază* B *în raport cu* $\sqsubseteq$ *astfel încât să avem relaţia* $P \subseteq FORM(B, \mathcal{S}_V)$.

Tema 2

Exerciţiul nr. 4

Se consideră următorul program logic P:

$$\begin{cases} p(a) \leftarrow \\ p(b) \leftarrow \\ q(f(x)) \leftarrow p(x) \end{cases}$$

a) Aflaţi tipul lui P în clasificarea programelor logice

b) Calculaţi baza B a lui P

c) Calculaţi universul Herbrand şi baza Herbrand a lui P

Exerciţiul nr. 5

Daţi exemple de două programe logice diferite care să aibă aceeaşi bază (în consecinţă acelaşi univers Herbrand şi aceeaşi bază Herbrand).

Exerciţiul nr. 6

Consideră m următorul program logic P:

$$\begin{cases} zbor(x,y) \leftarrow zbor\_direct(x,y) \\ zbor(x,y) \leftarrow zbor\_direct(x,z), zbor(z,y) \\ zbor\_direct(a,b) \leftarrow \\ zbor\_direct(b,c) \leftarrow \end{cases}$$

a) Calculaţi baza B a programului P

b) Calculaţi universul Herbrand şi baza Herbrand ataşată lui P

c) Adăugaţi următoarea clauză lui P:

$$zbor\_indirect(x, y) \leftarrow zbor(x, y), \neg zbor\_direct(x, y)$$

Notaţi cu P_1 programul obţinut. Ce fel de program logic este P_1 ?

d) Adăugaţi la P clauza de mai jos şi notaţi cu P_2 programul obţinut:

$$zbor\_indirect(x, y) \vee zbor\_direct(x, y) \leftarrow zbor(x, y)$$

Ce tip de program logic s-a obţinut?

Exerciţiul nr. 7

Demonstraţi următoarele relaţii:

- $truth\_val_\Sigma(\varphi \wedge \psi, F) = truth\_val_\Sigma(\varphi, F) \wedge truth\_val_\Sigma(\psi, F)$
- $truth\_val_\Sigma(\varphi \rightarrow \psi, F) = truth\_val_\Sigma(\varphi, F) \rightarrow truth\_val_\Sigma(\psi, F)$
- $truth\_val_\Sigma(\varphi \leftrightarrow \psi, F) = truth\_val_\Sigma(\varphi, F) \leftrightarrow truth\_val_\Sigma(\psi, F)$

Tema 3

Exerciţiul nr. 8

Fie $\mathcal{S}_C = \{a\}$, $\mathcal{S}_F = \{f^{(1)}\}$, $\mathcal{S}_P = \{p^{(2)}, q^{(2)}\}$. Considerăm structura

$\Sigma = (N, k)$, *unde* N *este mulţimea numerelor naturale, iar* k *este definit astfel:*

$k(f) : N \longrightarrow N$, $k(f)(n) = n + 3$;

$k(p) : N \longrightarrow \{true, false\}$, *unde*

$$k(p)(n, m) = \begin{cases} true \text{ dacă } n < 2m \\ \\ false \text{ altfel} \end{cases}$$

$k(q) : N \longrightarrow \{true, false\}$, *unde*

$$k(q)(n, m) = \begin{cases} true \text{ dacă } n^2 + m^2 < 100 \\ \\ false \text{ altfel} \end{cases}$$

$k(a) = 2$

Consideră $\mathcal{S}_V = \{x, y, z\}$ *şi asignarea* $F(x) = 2, F(y) = 5, F(z) = 6$. *Calculaţi* $truth\_val_\Sigma$ *pentru* F *definit mai sus şi formulele* $p(a, a)$, $\exists x p(x, f(y))$, $\exists x p(x, f(z))$, $\exists x p(x, f(x))$, $\forall x q(x, y)$, $\exists x q(f(a), x)$.

Exerciţiul nr. 9

Arătaţi că următoarele formule sunt logic echivalente:

$$\forall x \forall y [p(X, Y) \leftarrow \exists z (r(x, y, z) \wedge q(x, y, z))]$$

$$\forall x \forall y \forall z [p(X, Y) \leftarrow (r(x, y, z) \wedge q(x, y, z)]$$

Tema 4

Exerciţiul nr. 10

Consideră $p^{(1)} \in \mathcal{S}_P$. *Arătaţi că:*

1) $\forall x p(x) \approx \forall y p(y)$

2) $\forall x p(y) \approx p(y)$

3) $\neg \forall x p(x) \approx \exists x \neg p(x)$

4) $\neg \exists x p(x) \approx \forall x \neg p(x)$

5) $p(x) \approx \neg\neg p(x)$

Exerciţiul nr. 11

Pentru orice $\alpha, \beta \in FORM(B, \mathcal{S}_V)$ *arătaţi că*

1) $\neg(\alpha \vee \beta) \approx \neg\alpha \wedge \neg\beta$

2) $\neg(\alpha \wedge \beta) \approx \neg\alpha \vee \neg\beta$

3) $\neg\neg\alpha \approx \alpha$

Exerciţiul nr. 12

Pentru fiecare din formulele de mai jos găsiţi o clauză logic echivalentă:

1) $\forall x(p(x) \vee \neg\exists y(q(x, y) \wedge r(x)))$

2) $\forall x(\neg p(x) \vee (q(x) \rightarrow r(x)))$