Logică computațională
Curs 9

Lector dr. Mihiș Andreea-Diana

Metoda tabelelor semantice în calculul predicatelor

- introdusă de Smullyan
- se bazează pe considerații semantice
- încearcă să construiască modelele unei formule date (FND)
- $\vdash U$ prin respingere, $\neg U$ nu are modele
- ideea:
  - descompunerea formulei inițiale în subformule
  - până la nivel de literali

Clase de formule (1)

- clasa α - formule de tip conjunctiv

$$A \wedge B$$

$$\neg (A \vee B)$$

$$\neg (A \rightarrow B)$$

- clasa β - formule de tip disjunctiv

$$A \vee B$$

$$\neg (A \wedge B)$$

$$A \rightarrow B$$

Clase de formule (2)

- clasa γ - formule cuantificate universal

$$(\forall x)\ A(x)$$

$$\neg\ (\exists x)\ A(x)$$

- clasa δ - formule cuantificate existențial

$$(\exists x)\ A(x)$$

$$\neg\ (\forall x)\ A(x)$$

Reguli de descompunere a formulelor (1)

- regula α

$$\begin{array}{c} A \wedge B \\ | \\ A \\ | \\ B \end{array} \qquad \begin{array}{c} \neg(A \vee B) \\ | \\ \neg A \\ | \\ \neg B \end{array} \qquad \begin{array}{c} \neg(A \to B) \\ | \\ A \\ | \\ \neg B \end{array}$$

- regula β

$$\begin{array}{ccc} & A \vee B & \\ \diagup & & \diagdown \\ A & & B \end{array}$$

$$\begin{array}{ccc} & \neg(A \wedge B) & \\ \diagup & & \diagdown \\ \neg A & & \neg B \end{array}$$

$$\begin{array}{ccc} & A \to B & \\ \diagup & & \diagdown \\ \neg A & & B \end{array}$$

Reguli de descompunere a formulelor (2)

- regula γ

$(\forall x)\, A(x)$

$|$ $c_1, c_2, \ldots, c_n$ – toate constantele existente pe ramură

$A(c_1)$

$|$

$A(c_2)$

$|$

$\vdots$

$|$

$A(c_n)$

$|$

$(\forall x)\, A(x)$

copie formulă

$\neg\, (\exists x)\, A(x)$

$|$

$\neg\, A(c_1)$

$|$

$\neg\, A(c_2)$

$|$

$\vdots$

$|$

$\neg\, A(c_n)$

$|$

$\neg\, (\exists\, x)\, A(x)$ – copie formulă

- regula δ

$(\exists x)\, A(x)$

$|$ a – constantă nou introdusă

$A(a)$

$\neg\, (\forall x)\, A(x)$

$|$

$\neg\, A(a)$

Arborele binar de descompunere a unei formule

Având o formulă *U*, ei i se poate asocia o tabelă semantică, care este de fapt un arbore binar ce conţine în nodurile sale formule şi se construieşte astfel:

- rădăcina arborelui este etichetată cu formula *U*;
- fiecare ramură a arborelui care conţine o formulă va fi extinsă cu subarborele corespunzător regulii de descompunere care se aplică formulei;
- extinderea unei ramuri se *încheie* în două situaţii:
  a) dacă pe ramură apare o formulă şi negaţia sa;
  b) dacă au fost descompuse toate formulele de pe acea ramură sau prin aplicarea regulilor de descompunere nu se mai obțin formule noi pe acea ramură

Tipuri de ramuri

- O *ramură* a tabelei se numeşte *închisă* (simbolizată prin ⊗) dacă ea conţine o formulă şi negaţia ei, în caz contrar *ramura* se numeşte *deschisă* (simbolizată prin⊙).
- O *ramură* a tabelei se numeşte *completă* dacă ea este fie *închisă*, fie *toate formulele* de pe acea ramură au fost *descompuse*.

Tipuri de tabele semantice

- O *tabelă* se numeşte *închisă* dacă toate ramurile sale sunt închise. Dacă o tabelă are cel puţin o ramură deschisă, atunci ea se numeşte *deschisă*.
- O *tabelă* se numeşte *completă* dacă toate ramurile ei sunt complete.

Observații:

- Procesul de construire a unei tabele semantice este unul *nedeterminist* deoarece regulile de descompunere se pot aplica în orice ordine şi la un moment dat se pot alege mai multe ramuri pentru extindere. Astfel unei formule i se pot asocia mai multe tabele semantice, dar acestea sunt echivalente.
- Pentru a obţine tabele semantice *cât mai simple* (mai puţin ramificate) se recomandă:
  - utilizarea regulilor de tip α înaintea regulilor de tip β care realizează o ramificare;
  - utilizarea regulilor de tip δ (care introduc constante noi) înaintea regulilor de tip γ care utilizează toate constantele de pe ramura respectivă;

Observații (2):

- formulele de pe aceeaşi ramură a unei tabele semantice sunt *legate* între ele prin conectiva logică $\wedge$, iar *ramificarea* corespunde conectivei logice $\vee$.
- tabela semantică asociată unei formule propoziţionale este o reprezentare grafică a *formei* sale *normale disjunctive*. Fiecare ramură reprezintă un *cub* (conjuncţia tuturor literalilor de pe acea ramură), iar arborele este *disjuncţia* tuturor *ramurilor* sale.
- Unei formule *consistente* i se asociază o *tabelă completă deschisă*, iar fiecare *ramură deschisă* a tabelei furnizează cel puţin un *model* pentru formula respectivă.
- O *tabelă semantică închisă* asociată unei formule indică faptul că formula este *inconsistentă*, adică nu există nicio interpretare în care formula să fie adevărată

Teorema de corectitudine şi completitudine a metodei tabelelor semantice

- O formulă U este teoremă (tautologie) dacă şi numai dacă există o tabelă semantică închisă pentru formula $\neg U$.

Teoremă

- $U_1, U_2, ..., U_n \vdash Y$ (echivalent cu $U_1, U_2, ..., U_n \models Y$) dacă şi numai dacă există o tabelă semantică închisă pentru formula $U_1, U_2, ..., U_n \wedge \neg Y$.

Semi-decidabilitatea calculului predicativ

- Pentru cazul logicii predicatelor de ordinul I, arborele poate fi infinit datorită combinării regulilor de tip γ şi δ.

- Dacă arborele asociat negaţiei unei formule predicative este *finit*, atunci *se poate* decide dacă formula respectivă este o tautologie sau nu, dar dacă arborele este *infinit*, *nu* se poate decide nimic asupra validităţii formulei.

Substituții

Definiţie: O *substituţie* este o funcţie definită pe mulţimea variabilelor, *Var* cu valori în mulţimea termenilor, *TERM*. Se notează cu $\theta=[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$, reprezentând o mulţime finită de înlocuiri de variabile cu termeni. $x_1, \ldots, x_k$ sunt variabile distincte, iar $t_1, \ldots, t_k$ sunt termeni, astfel încât $\forall i = 1, \ldots, k$, $t_i \neq x_i$ și x_i nu este *subtermen* al lui t_i.

- $dom(\theta) = \{x_1, \ldots, x_k\}$ se numeşte domeniul substituţiei θ.
- ε – substituția vidă
- $\varphi, \delta, \phi, \eta, \theta, \lambda$

Aplicarea substituției

$\theta=[x_1 \leftarrow t_1,\ldots, x_k \leftarrow t_k]$ asupra formulei U se definește recursiv:

- $\theta(x_i) = t_i$, $x_i \in dom(\theta)$; $\theta(x) = x$, $x \notin dom(\theta)$;
- $\theta(c) = c$, c – constantă;
- $\theta(f(t_1,\ldots, t_n)) = f(\theta(t_1),\ldots, \theta(t_n))$, $f \in \mathscr{F}_n$;
- $\theta(P(t_1,\ldots, t_n)) = P(\theta(t_1),\ldots, \theta(t_n))$, $P \in \mathscr{P}_n$;
- $\theta(\neg U) = \neg\, \theta(U)$;
- $\theta(U \wedge V) = \theta(U) \wedge \theta(V)$;
- $\theta(U \vee V) = \theta(U) \vee \theta(V)$;
- $\theta(U \rightarrow V) = \theta(U) \rightarrow \theta(V)$;
- $\theta(U \leftrightarrow V) = \theta(U) \leftrightarrow \theta(V)$.

Compunerea substituțiilor

$\theta_1=[x_1\leftarrow t_1,\ldots, x_k\leftarrow t_k]$ și $\theta_2=[y_1\leftarrow s_1,\ldots, y_k\leftarrow s_k]$

$$\theta = \theta_1 \circ \theta_2 = [x_i \leftarrow \theta_2(t_i) \mid x_i \in dom(\theta_1), x_i \neq \theta_2(t_i)] \cup [y_i \leftarrow s_j \mid y_i \in dom(\theta_2) \setminus dom(\theta_1)]$$

- Obs.: Nu întotdeauna compunerea unor substituții este o substituție.

Proprietăți ale operației de compunere

- Element neutru: ε – substituția vidă:

$$\varepsilon\theta = \theta\varepsilon = \theta, \forall\theta - \text{substituție}$$

- Asociativitatea: $\theta_1(\theta_2\theta_3) = (\theta_1\theta_2)\theta_3 = \theta_1\theta_2\theta_3$

- În general compunerea nu este comutativă

Unificatori

- O substituţie θ se numeşte *unificator* al termenilor t_1 şi t_2 dacă $\theta(t_1) = \theta(t_2)$. Termenul $\theta(t_1)$ se numeşte *instanţa comună* a termenilor unificaţi.
- Un *unificator al mulţimii* de formule $\{U_1, U_2, \ldots, U_n\}$ este o substituţie θ cu proprietatea: $\theta(U_1) = \ldots = \theta(U_n)$.

- *Cel mai general unificator* ***(mgu)*** este un unificator μ cu proprietatea că orice alt unificator θ se obţine din compunerea lui μ cu o altă substituţie λ: $\theta = \mu\lambda$.

Algoritm pentru determinarea celui mai general unificator a doi literali (1)

Date de intrare: $l_1 = P_1(t_{1_1}, t_{1_2}, \ldots, t_{1_n})$ şi $l_2 = P_2(t_{2_1}, t_{2_2}, \ldots, t_{2_k})$ doi literali

Date de ieşire: $mgu(l_1, l_2)$ sau “l_1, l_2 nu sunt unificabili”

dacă $(P_1 \neq P_2)$ // simbolurile predicative sunt diferite

atunci *scrie “l_1, l_2 nu sunt unificabili”*; STOP;

sf_dacă

dacă$(n \neq k)$ // aritate diferită pentru acelaşi simbol predicativ

atunci *scrie “l_1, l_2 nu sunt unificabili”*; STOP;

sf_dacă

$\theta \leftarrow \varepsilon$; // iniţializare cu substituţia vidă

Algoritm pentru determinarea celui mai general unificator a doi literali (2)

cât_timp ($\theta(l_1) \neq \theta(l_2)$)

Din $\theta(l_1), \theta(l_2)$ se determină cele mai din stânga simbol de funcţie, constantă sau variabilă diferite şi notăm cu t_1 şi t_2 termenii lor corespunzători.

dacă (*niciunul dintre* t_1 *şi* t_2 *nu este variabilă* sau *unul este subtermenul celuilalt*)

atunci *scrie "l_1, l_2 nu sunt unificabili"*; STOP;

sf_dacă

dacă (*t_1 este variabilă*)

atunci $\lambda = [t_1 \leftarrow t_2]$;

altfel $\lambda = [t_2 \leftarrow t_1]$;

sf_dacă

$\theta \leftarrow \theta\, \lambda$;

dacă (*θ nu este substituție*)

atunci *scrie "l_1, l_2 nu sunt unificabili"*; STOP;

sf_dacă

sf_cât_timp

scrie "*l_1 şi l_2 sunt unificabili,mgu(l_1, l_2)=* " θ

Sf_algoritm