Logică computațională

Metoda tablelelor semantice în calculul predicatelor

$P(x)$ este un predicat care spune ceva despre x, de exemplu ("x este un număr par")

- metoda încearcă să construiască modelele unei formule date
- începem cu formula V (sau $\neg V$ dacă dorim să demonstrăm prin respingere)

Clase de formule (1)

clasa α - formule de tip conjunctiv

$A \wedge B$

$\neg(A \vee B)$

$\neg(A \to B)$

clasa β - formule de tip disjunctiv

$A \vee B$

$\neg(A \wedge B)$

$A \to B$

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

$A \wedge B$
|
A
|
B

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

$\neg(A \to B)$
|
A
|
$\neg B$

- regula β

$A \vee B$
A | B

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

$A \to B$
$\neg A$ | B

Reguli de descompunere a formulelor (2)

- regula γ

$(\forall x) A(x)$
|
$c_1, c_2, \ldots, c_m$ – toate constantele existente pe ramură
|
$A(c_1)$
|
$A(c_2)$
|
...
|
$A(c_m)$
|
$(\forall x) A(x)$

$\neg(\exists x) A(x)$
|
$\neg A(c_1)$
|
$\neg A(c_2)$
|
...
|
$\neg A(c_m)$
|
$\neg(\exists x) A(x)$

- regula δ

$(\exists x) A(x)$
|
$A(a)$

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

Arborele binar de descompunere a unei formule

Având o formulă U, îi i se poate asocia o tabelă semantică care este de fapt un arbore construit astfel:

1. Rădăcina este etichetată cu formula U
2. Fiecare ramură care conține o formulă este extinsă cu subarborii ce corespund regulilor de descompunere aplicate formulei.
3. Finalizarea extinderii unei ramuri se întâmplă când:
   a) Dacă pe o ramură apare o formulă și negația sa (contradicție)
   b) Dacă toate formulele au fost descompuse și prin regulile de descompunere nu se mai pot obține formule noi

Tipuri de ramuri

- O ramură se numește închisă ⊗ dacă ea conține o formulă și negația ei, în caz contrar, dacă este completă, ramura se numește deschisă ⊙
- O ramură se numește completă dacă ea este fie închisă fie toate formulele de pe acea ramură au fost descompuse.

Tipuri de tabele semantice

· O tabelă se numește:

a) **închisă** dacă toate ramurile sale sunt închise

b) **deschisă** dacă are cel puțin o ramură deschisă

c) **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. Astfel o formulă are asociate mai multe tabele semantice, care sunt echivalente.

· Recomandări pentru tabele semantice simple:

- utilizarea regulilor α înainte de β
- utilizarea regulilor δ (care introduc constante noi) înainte regulilor γ (care utilizează toate constantele de pe ramura respectivă)

Observații (2):

· Formulele de pe o ramură sunt legate prin $\wedge$, iar ramificarea corespunde conectivei $\vee$

· Tabela semantică a unei formule propoziționale este o reprezentare grafică a FND, unde avem disjuncții între ramuri și fiecare ramură reprezintă un cub.

· Unei formule consistente i se asociază o tabelă completă deschisă, iar fiecare ramură deschisă a tabelei furnizează cel puțin un model pentru formula respectivă, adică o configurație de valori de adevăr care face formula adevărată.

· O tabelă semantică închisă asociată unei formule indică faptul că formula este inconsistentă, adică nu există nicio interpretare în care formula să fie adevărată.

<u>Teorema de corectitudine și completitudine a metodei tabelelor semantice</u>

· formula U este tautologie d.m.d. există o tabelă semantică închisă pentru $\neg U$

· $U_1, U_2, \dots, U_m \vdash Y$ sau $U_1, U_2, \dots, U_m \models Y$ d.m.d există o tabelă semantică închisă pentru formula $U_1 \wedge U_2 \wedge \dots \wedge U_m \wedge \neg Y$

Exemplu

$\stackrel{?}{\models} (\exists x)(A(x) \wedge B(x)) \rightarrow (\exists x) A(x) \wedge (\exists x) B(x)$

$(\exists x)(A(x) \wedge B(x))$ – există un x pentru care $A(x) \wedge B(x)$ este adevărată

$\neg((\exists x)(A(x) \wedge B(x)) \rightarrow (\exists x) A(x) \wedge (\exists x) B(x))$ (1)

| $\mathcal{L}$ (1)

$(\exists x)(A(x) \wedge B(x))$ (2)

|

$\neg((\exists x) A(x) \wedge (\exists x) B(x))$ (3)

| δ (2), a - constantă nouă

$A(a) \wedge B(a)$ (4)

|

$A(a)$

|

$B(a)$

β (3)

$\neg(\exists x) A(x)$ (5) | γ (5), a - constantă existentă

$\neg A(a)$

|

$\neg(\exists x) A(x)$ (5') ⊗

$\neg(\exists x) B(x)$ (6) | γ (6), a - constantă existentă

$\neg B(a)$

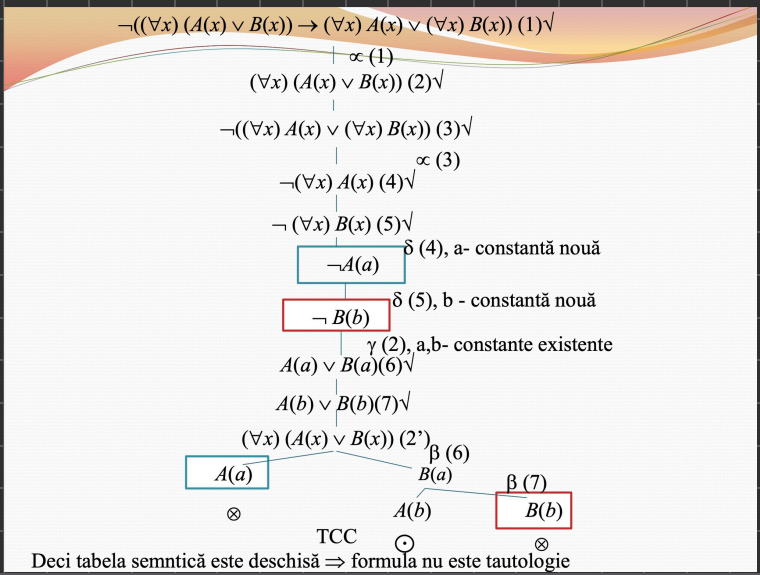
|

$\neg(\exists x) B(x)$ (6') ⊗

Tabela semantică a formulei negate este închisă $\stackrel{TCC}{\Rightarrow}$ formula este tautologie

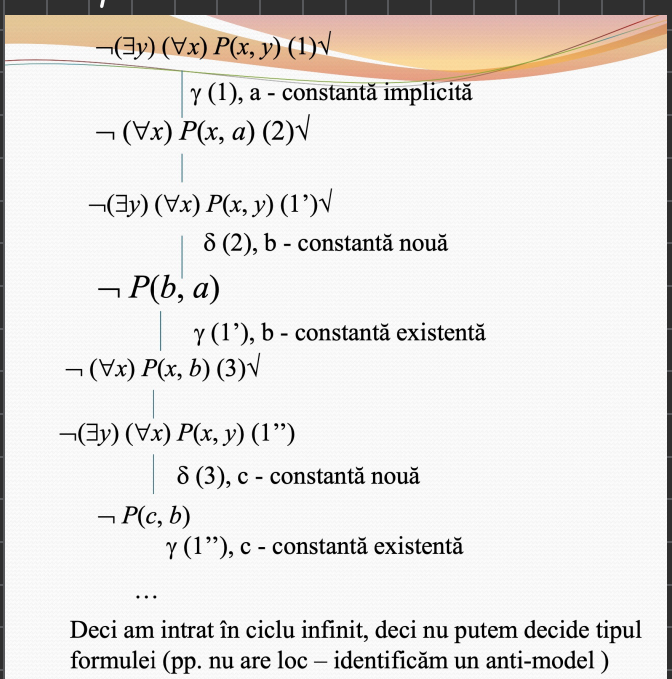
Exemple

? $\models (\forall x)(A(x) \vee B(x)) \rightarrow (\forall x) A(x) \vee (\forall x) B(x)$



Deci tabela semntică este deschisă ⇒ formula nu este tautologie

? $\models (\exists y)(\forall x) P(x, y)$



Deci am intrat în ciclu infinit, deci nu putem decide tipul formulei (pp. nu are loc – identificăm un anti-model)

Semi-decidabilitatea calculului predicativ

- Pentru predicatele de ordin I (funcții logice care exprimă proprietăți, ex: P(x): "x este roșu") arborele poate fi infinit datorită regulilor:
  - γ (care generează mai multe ramificații pentru formula $\forall$ cu toate constantele deja existente pe ramură)
  - δ (care generează ramuri noi pentru formula $\exists$ cu constante noi)
- Dacă arborele asociat negației unei formule predicative este finit atunci se poate decide dacă formula este tautologie sau nu, dar dacă este infinit, nu se poate decide nimic.