Fundamente de informatică

Logica predicatelor

Marius Minea
marius@cs.upt.ro

http://www.cs.upt.ro/~marius/curs/fi

14 noiembrie 2012

Ce vrem să exprimăm

Proprietăți mai complexe decât în logica propozițională:

$member(x, A) \rightarrow member(x, union(A, B))$ mulțimi

$leq(x, y) \rightarrow leq(f(x), f(y))$ funcție monotonă

$apel(nrX, nrY) \wedge eq(rețea(nrX), rețea(nrY)) \wedge prepay(nrX)$
$\rightarrow eq(cost(nrX, nrY), 0.11)$

$apel(nrX, nrY) \wedge fix(nrX) \wedge fix(nrY) \rightarrow eq(cost(nrX, nrY), 0.04)$

Avem:

- variabile (x, y, nrX, nrY)
- funcții (*union*, f, *rețea*, *cost*)
- predicate (*member*, *leq*, *apel*, *prepay*, *fix*)
  - (egalitatea e un predicat considerată uneori separat)

Un *predicat* = o afirmație relativ la una sau mai multe variabile, care, dând valori variabilelor, poate lua valoarea adevărat sau fals.

Limbajul logicii predicatelor (logica de ordinul I)

Simboluri: parantezele ( )

conectorii $\neg$ și $\rightarrow$

cuantificatorul $\forall$ (universal)

o mulțime de identificatori $v_0, v_1, \cdots$ pentru *variabile*

o mulțime (posibil vidă) de simboluri pentru *constante*

pt. orice $n \geq 1$ o mulțime de simboluri de *funcții* n-are

pt. orice $n \geq 1$ o mulțime de simboluri de *predicate* n-are

Limbajele de ordinul I cu egalitate:

conțin și $=$ ca simbol special pe lângă cele de mai sus.

Termeni și formule

Termenii unui limbaj de ordinul I (definiți structural recursiv)

orice simbol de variabilă v_n

orice simbol de constantă c

$f(t_1, \cdots, t_n)$

dacă f e un simbol de funcție n-ară și $t_1, \cdots, t_n$ sunt termeni

Termeni și formule

Termenii unui limbaj de ordinul I (definiți structural recursiv)

orice simbol de variabilă v_n

orice simbol de constantă c

$f(t_1, \cdots, t_n)$

dacă f e un simbol de funcție n-ară și $t_1, \cdots, t_n$ sunt termeni

Formule bine formate (well-formed formulas):

$P(t_1, \cdots, t_n)$ cu P predicat n-ar; $t_1, \cdots, t_n$ termeni

$t_1 = t_2$ cu t_1, t_2 termeni (în limbaje cu egalitate)

$\neg\alpha$ unde α este o formulă

$\alpha \to \beta$ unde α, β sunt formule

$\forall v\, \varphi$ unde v e o variabilă și φ e o formulă

Notăm: $\exists x\varphi \stackrel{def}{=} \neg\forall x(\neg\varphi)$

Substituții și unificare

O *substituție* e o funcție care asociază unor variabile niște termeni:

$$\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$$

De exemplu $f(x, g(y, z), a, t)\{x \mapsto g(y), y \mapsto f(b), t \mapsto u\}$
$= f(g(y), g(f(b), z), a, u)$

Unificare = găsirea unei substituții care face doi termeni egali

Exemplu: $f(x, g(y))\{x \mapsto a\} = f(a, g(y) = f(a, z)\{z \mapsto g(y)\}$

Problemă: pot fi unificați doi termeni ?

Motivație:

$\forall x.\forall y.P(x, g(y))$ și $\forall z.\neg P(z, a)$. Se contrazic ?

Dar $\forall x.\forall y.P(x, g(y))$ și $\forall z.\neg P(a, z)$?

$\forall x.\forall y.P(g(x), y) \rightarrow A(x, y)$ și $\forall z.P(z, a) \rightarrow B(x, y)$

Există u, v pentru care putem deduce $A(u, v)$ și $B(u, v)$?

Reguli de unificare

O variabilă x poate fi unificată cu orice termen t
dacă x *nu apare* în t (nu: x cu $f(g(y), h(x, z))$
(pentru că altfel, substituția ar duce la un termen infinit)

Două constante pot fi unificate doar dacă sunt identice

Doi termeni funcționali pot fi unificați doar dacă au funcții identice, și termenii argument corespunzători pot fi unificați

$\Rightarrow$ cu aceste reguli, scriem un algoritm recursiv de unificare

Union-Find

Structură de date / algoritm pentru mulțimi cu clase de echivalență.

Operații:
find(element): găsește reprezentantul clasei de echivalență
union(elem1, elem2): declară elementele ca fiind echivalente
(și rămân aăa mai departe)

Implementare: pădure de arbori cu legături de la fiu la părinte
find: returnează rădăcina (chiar nodul, dacă e singur)
union: leagă rădăcina unuia de rădăcina celuilalt