Curs ID1

Cuprins

Logica propozițională PL (recap.)

Logica propozițională PL

- O propoziție este un enunț care poate fi adevărat (1) sau fals (0).
- Propozițiile sunt notate simbolic (φ, ψ, χ, $\cdots$) și sunt combinate cu ajutorul conectorilor logici ($\neg$, $\rightarrow$, $\vee$, $\wedge$, $\leftrightarrow$).

Exemplu

Fie φ propoziția:

$$(\texttt{stark} \wedge \neg\ \texttt{dead}) \rightarrow (\texttt{sansa} \vee \texttt{arya} \vee \texttt{bran})$$

Cine este $\neg\varphi$? Propoziția $\neg\varphi$ este:

$$\texttt{stark} \wedge \neg\ \texttt{dead} \wedge \neg\texttt{sansa} \wedge \neg\texttt{arya} \wedge \neg\texttt{bran}$$

Limbajul și formulele PL

- Limbajul PL
  - variabile propoziționale: $VP = \{p, q, v, \ldots\}$
  - conectori logici: $\neg$ (unar), $\rightarrow$, $\wedge$, $\vee$, $\leftrightarrow$ (binari)
- Formulele PL

$$
\begin{array}{lcl}
var & ::= & p \mid q \mid v \mid \ldots \\
form & ::= & var \mid (\neg form) \mid form \wedge form \mid form \vee form \\
 & & \mid form \rightarrow form \mid form \leftrightarrow form
\end{array}
$$

Exemplu

- Nu sunt formule: $v_1 \neg \rightarrow (v_2)$, $\neg v_1 v_2$
- Sunt formule: $((v_1 \rightarrow v_2) \rightarrow (\neg v_1))$, $(\neg(v_1 \rightarrow v_2))$

- Notăm cu *Form* mulțimea formulelor.

Limbajul și formulele PL

- Limbajul PL
  - variabile propoziționale: $VP = \{p, q, v, \ldots\}$
  - conectori logici: $\neg$ (unar), $\to$, $\wedge$, $\vee$, $\leftrightarrow$ (binari)

- Formulele PL

$$
\begin{array}{lcl}
var & ::= & p \mid q \mid v \mid \ldots \\
form & ::= & var \mid (\neg form) \mid form \wedge form \mid form \vee form \\
 & & \mid form \to form \mid form \leftrightarrow form
\end{array}
$$

- Conectorii sunt împărțiți în conectori de bază și conectori derivați (în funcție de formalism).

- Legături între conectori:

$$
\begin{array}{rcl}
\varphi \vee \psi & := & \neg\varphi \to \psi \\
\varphi \wedge \psi & := & \neg(\varphi \to \neg\psi) \\
\varphi \leftrightarrow \psi & := & (\varphi \to \psi) \wedge (\psi \to \varphi)
\end{array}
$$

Sintaxa și semantica

Un sistem logic are două componente:

- Sintaxa
  - noțiuni sintactice: demonstrație, teoremă
  - notăm prin $\vdash \varphi$ faptul că φ este teoremă
  - notăm prin $\Gamma \vdash \varphi$ faptul că formula φ este demonstrabilă din mulțimea de formule Γ
- Semantica
  - noțiuni semantice: adevăr, model, tautologie (formulă universal adevărată)
  - notăm prin $\models \varphi$ faptul că φ este tautologie
  - notăm prin $\Gamma \models \varphi$ faptul că formula φ este adevărată atunci când toate formulele din mulțimea Γ sunt adevărate

Exemplu

Formalizați următorul raționament:

`If winter is coming and Ned is not alive then Robb is lord of Winterfell. Winter is coming. Rob is not lord of Winterfell. Then Ned is alive.`

O posibilă formalizare este următoarea:

p = `winter is coming`
q = `Ned is alive`
r = `Robb is lord of Winterfel`

$\{(p \land \neg q) \to r, p, \neg r\} \models q$

Semantica PL

- Mulțimea valorilor de adevăr este $\{0, 1\}$ pe care considerăm următoarele operații:

x	$\neg x$
0	1
1	0

x	y	$x \to y$
0	0	1
0	1	1
1	0	0
1	1	1

$x \vee y := max\{x, y\}$ $\qquad$ $x \wedge y := min\{x, y\}$

Semantica PL

- o funcție $e : VP \to \{0, 1\}$ se numește evaluare (interpretare)
- pentru orice evaluare $e : VP \to \{0, 1\}$ există o unică funcție $e^+ : Form \to \{0, 1\}$ care verifică următoarele proprietăți:

  - $e^+(v) = e(v)$
  - $e^+(\neg\varphi) = \neg e^+(\varphi)$
  - $e^+(\varphi \to \psi) = e^+(\varphi) \to e^+(\psi)$
  - $e^+(\varphi \wedge \psi) = e^+(\varphi) \wedge e^+(\psi)$
  - $e^+(\varphi \vee \psi) = e^+(\varphi) \vee e^+(\psi)$

  oricare ar fi $v \in VP$ și $\varphi, \psi \in Form$.

Exemplu

Dacă $e(p) = 0$ și $e(q) = 1$ atunci

$$e^+(p \vee (p \to q)) = e^+(p) \vee e^+(p \to q) = e(p) \vee (e(p) \to e(q)) = 1$$

Semantica PL

Considerăm $\Gamma \cup \{\varphi\} \subseteq Form$.

- O evaluare $e : VP \to \{0,1\}$ este model al formulei φ dacă $e^+(\varphi) = 1$. Evaluarea e este model al lui Γ dacă $e^+(\Gamma) = \{1\}$, i.e. $e^+(\gamma) = 1$ oricare $\gamma \in \Gamma$.
- O formulă φ este satisfiabilă dacă are un model. O mulțime Γ de formule este satisfiabilă dacă are un model.
- O formulă φ este tautologie (validă, universal adevarată) dacă $e^+(\varphi) = 1$ pentru orice evaluare $e : VP \to \{0,1\}$.
  Notăm prin $\models \varphi$ faptul că φ este o tautologie.
- O formulă φ este $\Gamma-$tautologie (consecință semantică a lui Γ) dacă orice model al lui Γ este și model pentru φ, i.e. $e^+(\Gamma) = \{1\}$ implică $e^+(\varphi) = 1$ pentru orice evaluare $e : VP \to \{0,1\}$.
  Notăm prin $\Gamma \models \varphi$ faptul că φ este o Γ-tautologie.

Semantica PL

Cum verificăm că o formulă este tautologie: $\models \varphi$?

- Fie $v_1, \ldots, v_n$ variabilele care apar în φ.
- Cele 2^n evaluări posibile $e_1, \ldots, e_{2^n}$ pot fi scrise într-un tabel:

v_1	v_2	$\ldots$	v_n	φ
$e_1(v_1)$	$e_1(v_2)$	$\ldots$	$e_1(v_n)$	$e_1^+(\varphi)$
$e_2(v_1)$	$e_2(v_2)$	$\ldots$	$e_2(v_n)$	$e_2^+(\varphi)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$e_{2^n}(v_1)$	$e_{2^n}(v_2)$	$\ldots$	$e_{2^n}(v_n)$	$e_{2^n}^+(\varphi)$

  Fiecare evaluare corespunde unei linii din tabel!

- $\models \varphi$ dacă și numai dacă $e_1^+(\varphi) = \cdots = e_{2^n}^+(\varphi) = 1$

Verificarea problemei consecinței logice

- În principiu, putem verifica problema consecinței logice construind un tabel de adevăr, cu câte o linie pentru fiecare interpretare posibilă.
- În cazul în care formula conțin n variabile, tabelul de adevăr are 2^n rânduri. Această metodă este atât de costisitoare computațional (timp exponențial).
- Problemă deschisă de un milion de dolari:

  Este posibil să decidem problema consecinței logice în cazul propozițional printr-un algoritm care să funcționeze în timp polinomial?

  Echivalent, este adevărată $P = NP$?
  (Institutul de Matematica Clay – Millennium Prize Problems)
- SAT este problema satisfiabilității în calculul propozițional clasic. SAT-solverele sunt bazate pe metode sintactice.

Clauze propoziționale definite

- O clauză propozițională definită este o formulă care poate avea una din formele:
  1. q (un fapt în Prolog `q.`)
  2. $p_1 \wedge \ldots \wedge p_k \to q$ (o regulă în Prolog `q :- p1,...,pk`)

  unde $q, p_1, \ldots, p_n$ sunt variabile propoziționale

- Numim variabilele propoziționale atomi.

Programare logică – cazul logicii propoziționale

- Un "program logic" este o listă $Cd_1, \ldots, Cd_n$ de clauze definite.
- O întrebare este o listă $q_1, \ldots, q_m$ de atomi.
- Sarcina sistemului este să stabilească:

$$Cd_1, \ldots, Cd_n \vDash q_1 \wedge \ldots \wedge q_m.$$

Clauze propoziționale definite

Exemplu

Cd_1: $\texttt{oslo} \to \texttt{windy}$
Cd_2: $\texttt{oslo} \to \texttt{norway}$
Cd_3: $\texttt{norway} \to \texttt{cold}$
Cd_4: $\texttt{cold} \land \texttt{windy} \to \texttt{winterIsComing}$
Cd_5: $\texttt{oslo}$
q_1: $\texttt{winterIsComing}$

Programul Prolog corespunzător:

```
windy :- oslo.
norway :- oslo.
cold :- norway.
winterIsComing :- windy, cold.
oslo.
```

Intrebare:

```
?- winterIsComing.
```

Logica de ordinul I - sintaxa

Logica de ordinul I

- Sloganul programării logice:

  Un program este o teorie într-o logică formală,
  iar execuția sa este o deducție în teorie.

- Programarea logică folosește un fragment din logica de ordinul I (calculul cu predicate) ca limbaj de reprezentare.

- În această reprezentare, programele sunt teorii logice – mulțimi de formule din calculul cu predicate.

- Reamintim că problema constă în căutarea unei derivări a unei întrebări (formule) dintr-un program (teorie).

Limbaje de ordinul I

Un limbaj $\mathcal{L}$ de ordinul I este format din:

- o mulțime numărabilă de variabile $V = \{x_n \mid n \in \mathbb{N}\}$
- conectorii $\neg, \rightarrow, \wedge, \vee$
- paranteze
- cuantificatorul universal $\forall$ și cuantificatorul existențial $\exists$
- o mulțime $\mathbf{R}$ de simboluri de relații
- o mulțime $\mathbf{F}$ de simboluri de funcții
- o mulțime $\mathbf{C}$ de simboluri de constante
- o funcție aritate $ar : \mathbf{F} \cup \mathbf{R} \rightarrow \mathbb{N}^*$

Logica de ordinul I

- $\mathcal{L}$ este unic determinat de $\tau = (\mathbf{R}, \mathbf{F}, \mathbf{C}, ari)$
- τ se numește signatura (vocabularul, alfabetul) lui $\mathcal{L}$

Exemplu

Un limbaj $\mathcal{L}$ de ordinul I în care:

- $\mathbf{R} = \{P, R\}$
- $\mathbf{F} = \{f\}$
- $\mathbf{C} = \{c\}$
- $ari(P) = 1$, $ari(R) = 2$, $ari(f) = 2$

Sintaxa Prolog

Atenție!

- În sintaxa Prolog
  - termenii compuși sunt predicate: `father(eddard, jon_snow)`
  - operatorii sunt funcții: `+`, `*`, `mod`
- Sintaxa Prolog nu face diferență între simboluri de funcții și simboluri de predicate!
- Dar este important când ne uităm la teoria corespunzătoare programului în logică să facem acestă distincție.

Termenii lui $\mathcal{L}$ sunt definiți inductiv astfel:

- orice variabilă este un termen;
- orice simbol de constantă este un termen;
- dacă $f \in \mathbf{F}$, $ar(f) = n$ și $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este termen.

Notăm cu $Trm_{\mathcal{L}}$ mulțimea termenilor lui $\mathcal{L}$.

Exemplu

$$c, \quad x_1, \quad f(x_1, c), \quad f(f(x_2, x_2), c)$$

Formulele atomice ale lui $\mathcal{L}$ sunt definite astfel:

- dacă $R \in \mathbf{R}$, $ar(R) = n$ și $t_1, \ldots, t_n$ sunt termeni, atunci $R(t_1, \ldots, t_n)$ este formulă atomică.

Exemplu

$$P(f(x_1, c)), \quad R(c, x_3)$$

Logica de ordinul I

Formulele lui $\mathcal{L}$ sunt definite astfel:

- orice formulă atomică este o formulă
- dacă φ este o formulă, atunci $\neg\varphi$ este o formulă
- dacă φ și ψ sunt formule, atunci $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ sunt formule
- dacă φ este o formulă și x este o variabilă, atunci $\forall x\, \varphi$, $\exists x\, \varphi$ sunt formule

Exemplu

$$P(f(x_1, c)), \quad P(x_1) \vee P(c), \quad \forall x_1\, P(x_1), \quad \forall x_2\, R(x_2, x_1)$$

Exemplu

Fie limbajul $\mathcal{L}_1$ cu $\mathbf{R} = \{<\}$, $\mathbf{F} = \{s, +\}$, $\mathbf{C} = \{0\}$ și $ari(s) = 1$, $ari(+) = ari(<) = 2$.

Exemple de termeni:

0, x, $s(0)$, $s(s(0))$, $s(x)$, $s(s(x))$, $\ldots$,
$+(0,0)$, $+(s(s(0)), +(0, s(0)))$, $+(x, s(0))$, $+(x, s(x))$, $\ldots$,

Exemple de formule atomice:

$<(0,0)$, $<(x,0)$, $<(s(s(x)), s(0))$, $\ldots$

Exemple de formule:

$\forall x \forall y < (x, +(x,y))$
$\forall x < (x, s(x))$

Logica de ordinul I - sintaxa

Limbaj de ordinul I $\mathcal{L}$

- unic determinat de $\tau = (\mathbf{R}, \mathbf{F}, \mathbf{C}, ari)$

Termenii lui $\mathcal{L}$, notați $Trm_{\mathcal{L}}$, sunt definiți inductiv astfel:

- orice variabilă este un termen;
- orice simbol de constantă este un termen;
- dacă $f \in \mathbf{F}$, $ar(f) = n$ și $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este termen.

Formulele atomice ale lui $\mathcal{L}$ sunt definite astfel:

- dacă $R \in \mathbf{R}$, $ar(R) = n$ și $t_1, \ldots, t_n$ sunt termeni, atunci $R(t_1, \ldots, t_n)$ este formulă atomică.

Formulele lui $\mathcal{L}$ sunt definite astfel:

- orice formulă atomică este o formulă
- dacă φ este o formulă, atunci $\neg\varphi$ este o formulă
- dacă φ și ψ sunt formule, atunci $\varphi \vee \psi$, $\varphi \wedge \psi$, $\varphi \rightarrow \psi$ sunt formule
- dacă φ este o formulă și x este o variabilă, atunci $\forall x\, \varphi$, $\exists x\, \varphi$ sunt formule

Semantica

- Un predicat este formalizarea unei relații care are pentru noi o valoare de adevăr.
- De exemplu când scriem `father(jon,ken)` înțelegem: "`ken` *este tatăl lui* `jon` " (sau invers).

Cum definim ceea ce este adevărat în logica de ordinul I?

Pentru a stabili dacă o formulă este adevărată, avem nevoie de o interpretare într-o structură!

Logica de ordinul I - semantica

Structură

Definiție

O structură este de forma $\mathcal{A} = (A, \mathbf{F}^{\mathcal{A}}, \mathbf{R}^{\mathcal{A}}, \mathbf{C}^{\mathcal{A}})$, unde

- A este o mulțime nevidă
- $\mathbf{F}^{\mathcal{A}} = \{f^{\mathcal{A}} \mid f \in \mathbf{F}\}$ este o mulțime de operații pe A; dacă f are aritatea n, atunci $f^{\mathcal{A}} : A^n \to A$.
- $\mathbf{R}^{\mathcal{A}} = \{R^{\mathcal{A}} \mid R \in \mathbf{R}\}$ este o mulțime de relații pe A; dacă R are aritatea n, atunci $R^{\mathcal{A}} \subseteq A^n$.
- $\mathbf{C}^{\mathcal{A}} = \{c^{\mathcal{A}} \in A \mid c \in \mathbf{C}\}$.

- A se numește universul structurii $\mathcal{A}$.
- $f^{\mathcal{A}}$ (respectiv $R^{\mathcal{A}}$, $c^{\mathcal{A}}$) se numește interpretarea lui f (respectiv R, c) in $\mathcal{A}$.

Structură

Exemplu

$\mathcal{L}_1$: $\mathbf{R} = \{<\}$, $\mathbf{F} = \{s, +\}$, $\mathbf{C} = \{0\}$ cu $ari(s) = 1$, $ari(+) = ari(<) = 2$.

$\mathcal{N} = (\mathbb{N}, s^{\mathcal{N}}, +^{\mathcal{N}}, <^{\mathcal{N}}, 0^{\mathcal{N}})$ unde

- $s^{\mathcal{N}} : \mathbb{N} \to \mathbb{N}, \quad s^{\mathcal{N}}(n) := n + 1,$
- $+^{\mathcal{N}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \quad +^{\mathcal{N}}(n, m) := n + m,$
- $<^{\mathcal{N}} \subseteq \mathbb{N} \times \mathbb{N}, \quad <^{\mathcal{N}} = \{(n, m) \mid n < m\},$
- $0^{\mathcal{N}} := 0$

Modelarea unei lumi

Presupunem că putem descrie o lume prin:

- o mulțime de obiecte
- funcții
- relații

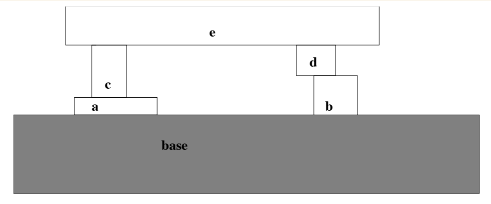
unde

- funcțiile duc obiecte în obiecte
- relațiile cu n argumente descriu proprietățile a n obiecte

Modelarea unei lumi

Exemplu

Să considerăm o lume în care avem cutii:



- Putem descrie lumea folosind obiecte

$$O = \{base, a, b, c, d, e\}.$$

- Putem descrie ce obiect se află deasupra altui obiect folosind un predicat binar *on*:

$$on = \{(e, c), (c, a), (e, d), (d, b), (a, base), (b, base)\}$$

Sursa exemplului: `https://www.inf.ed.ac.uk/teaching/courses/lp/`

Structură

Exemplu

Lumea în care avem cutii.

- Limbajul $\mathcal{L}$
  - $\mathbf{R} = \{on\}$
  - $\mathbf{F} = \emptyset$
  - $\mathbf{C} = \emptyset$
  - $ari(on) = 2$

- O structură $\mathcal{A}$:
  - $A = \{base, a, b, c, d, e\}$
  - $\mathbf{F}^{\mathcal{A}} = \emptyset$.
  - $\mathbf{C}^{\mathcal{A}} = \emptyset$.
  - $\mathbf{R}^{\mathcal{A}} = \{on^{\mathcal{A}}\}$, unde
    $on^{\mathcal{A}} = \{(e, c), (c, a), (e, d), (d, b), (a, base), (b, base)\} \subseteq A^2$.

Interpretare

Fie $\mathcal{L}$ un limbaj de ordinul I și $\mathcal{A}$ o ($\mathcal{L}$-)structură.

Definiție

O interpretare a variabilelor lui $\mathcal{L}$ în $\mathcal{A}$ este o funcție

$$I : V \to A.$$

Definiție

Inductiv, definim interpretarea termenului t în $\mathcal{A}$ sub I ($t_I^{\mathcal{A}}$) prin:

- dacă $t = x_i \in V$, atunci $t_I^{\mathcal{A}} := I(x_i)$
- dacă $t = c \in \mathbf{C}$, atunci $t_I^{\mathcal{A}} := c^{\mathcal{A}}$
- dacă $t = f(t_1, \ldots, t_n)$, atunci $t_I^{\mathcal{A}} := f^{\mathcal{A}}((t_1)_I^{\mathcal{A}}, \ldots, (t_n)_I^{\mathcal{A}})$

Interpretare

Definim inductiv faptul că o formulă este adevărată în $\mathcal{A}$ sub interpretarea I astfel:

- $\mathcal{A}, I \vDash P(t_1, \ldots, t_n)$ dacă $P^{\mathcal{A}}((t_1)_I^{\mathcal{A}}, \ldots, (t_n)_I^{\mathcal{A}})$
- $\mathcal{A}, I \vDash \neg\varphi$ dacă $\mathcal{A}, I \nvDash \varphi$
- $\mathcal{A}, I \vDash \varphi \vee \psi$ dacă $\mathcal{A}, I \vDash \varphi$ sau $\mathcal{A}, I \vDash \psi$
- $\mathcal{A}, I \vDash \varphi \wedge \psi$ dacă $\mathcal{A}, I \vDash \varphi$ și $\mathcal{A}, I \vDash \psi$
- $\mathcal{A}, I \vDash \varphi \rightarrow \psi$ dacă $\mathcal{A}, I \nvDash \varphi$ sau $\mathcal{A}, I \vDash \psi$
- $\mathcal{A}, I \vDash \forall x\, \varphi$ dacă pentru orice $a \in A$ avem $\mathcal{A}, I_{x_i \leftarrow a} \vDash \varphi$
- $\mathcal{A}, I \vDash \exists x\, \varphi$ dacă există $a \in A$ astfel încât $\mathcal{A}, I_{x_i \leftarrow a} \vDash \varphi$

unde pentru orice $a \in A$, $I_{x \leftarrow a}(y) = \begin{cases} I(y) & \text{dacă } y \neq x \\ a & \text{dacă } y = x \end{cases}$

Interpretare

- O formulă φ este adevărată într-o structură $\mathcal{A}$, notat $\mathcal{A} \vDash \varphi$, dacă este adevărată în $\mathcal{A}$ sub orice interpretare.

  Spunem că $\mathcal{A}$ este model al lui φ.

- O formulă φ este adevărată în logica de ordinul I, notat $\vDash \varphi$, dacă este adevărată în orice structură.

Model

Exemplu

Fie limbajul $\mathcal{L}$ cu $\mathbf{F} = \{s\}$, $\mathbf{R} = \{P\}$, $\mathbf{C} = \{0\}$ cu $ari(s) = ari(P) = 1$.

Fie structura $\mathcal{N} = (\mathbb{N}, s^{\mathcal{N}}, P^{\mathcal{N}}, 0^{\mathcal{N}})$ unde $0^{\mathcal{N}} := 1$ și

- $s^{\mathcal{N}} : \mathbb{N} \to \mathbb{N}$, $s^{\mathcal{N}}(n) := n^2$
- $P^{\mathcal{N}} \subset \mathbb{N}$, $P^{\mathcal{N}} = \{n \mid n \text{ este impar }\}$

Demonstrați că $\mathcal{N} \vDash \forall x\,(P(x) \to P(s(x)))$.

Fie $I : V \to \mathbb{N}$ o interpretare. Observăm că
$\mathcal{N}, I \vDash P(x)$ dacă $P^{\mathcal{N}}(I(x))$, adică $\mathcal{N}, I \vDash P(x)$ dacă $I(x)$ este impar.

$\mathcal{N}, I \vDash \forall x\,(P(x) \to P(s(x)))$ dacă
$\mathcal{N}, I_{x\leftarrow n} \vDash P(x) \to P(s(x))$ oricare $n \in N$
$\mathcal{N}, I_{x\leftarrow n} \nvDash P(x)$ sau $\mathcal{N}, I_{x\leftarrow n} \vDash P(s(x))$ oricare $n \in N$
$I_{x\leftarrow n}(x)$ nu este impar sau $I_{x\leftarrow n}(s(x))$ este impar oricare $n \in \mathbb{N}$
n este par sau n^2 este impar oricare $n \in \mathbb{N}$

ceea ce este întodeauna adevărat.

Logica de ordinul I - semantică

O structură este de forma $\mathcal{A} = (A, \mathbf{F}^{\mathcal{A}}, \mathbf{R}^{\mathcal{A}}, \mathbf{C}^{\mathcal{A}})$, unde

- A este o mulțime nevidă
- $\mathbf{F}^{\mathcal{A}} = \{f^{\mathcal{A}} \mid f \in \mathbf{F}\}$ este o mulțime de operații pe A; dacă f are aritatea n, atunci $f^{\mathcal{A}} : A^n \to A$.
- $\mathbf{R}^{\mathcal{A}} = \{R^{\mathcal{A}} \mid R \in \mathbf{R}\}$ este o mulțime de relații pe A; dacă R are aritatea n, atunci $R^{\mathcal{A}} \subseteq A^n$.
- $\mathbf{C}^{\mathcal{A}} = \{c^{\mathcal{A}} \in A \mid c \in \mathbf{C}\}$.

O interpretare a variabilelor lui $\mathcal{L}$ în $\mathcal{A}$ ($\mathcal{A}$-interpretare) este o funcție $I : V \to A$.

Inductiv, definim interpretarea termenului t în $\mathcal{A}$ sub I notat $t_I^{\mathcal{A}}$.

Inductiv, definim când o formulă este adevărată în $\mathcal{A}$ în interpretarea I notat $\mathcal{A}, I \vDash \varphi$. În acest caz spunem că $(\mathcal{A}, I)$ este model pentru φ.

O formulă φ este adevărată într-o structură $\mathcal{A}$, notat $\mathcal{A} \vDash \varphi$, dacă este adevărată în $\mathcal{A}$ sub orice interpretare. Spunem că $\mathcal{A}$ este model al lui φ.

O formulă φ este adevărată în logica de ordinul I, notat $\vDash \varphi$, dacă este adevărată în orice structură. O formulă φ este validă dacă $\vDash \varphi$.

O formulă φ este satisfiabilă dacă există o structură $\mathcal{A}$ și o $\mathcal{A}$-interpretare I astfel încât $\mathcal{A}, I \vDash \varphi$.

Consecință logică

Definiție

O formulă φ este o consecință logică a formulelor $\varphi_1, \ldots, \varphi_n$, notat

$$\varphi_1, \ldots, \varphi_n \vDash \varphi,$$

dacă pentru orice structură $\mathcal{A}$

$$\text{dacă } \mathcal{A} \vDash \varphi_1 \text{ și } \ldots \text{ și } \mathcal{A} \vDash \varphi_n, \text{ atunci } \mathcal{A} \vDash \varphi$$

Problemă semidecidabilă!

Nu există algoritm care să decidă mereu dacă o formula este sau nu consecință logică a altei formule în logica de ordinul I!

Formule echivalente

- Fie φ și ψ două formule. Notăm prin

$$\varphi \mathrel{\text{⫤}\!\!\models} \psi$$

faptul că $\models \varphi \leftrightarrow \psi$, adică φ și ψ au aceleași modele.

Exemplu

Dacă P este un simbol de relație de aritate 1 și x și y sunt variabile distincte, atunci

$$\forall x\, P(x) \mathrel{\text{⫤}\!\!\models} \forall y\, P(y) \qquad \text{și} \qquad P(x) \mathrel{\text{⫤}\!\!\models} P(y)$$

Validitate și satisfiabilitate

Propoziție

Dacă φ este o formulă atunci

φ este validă   dacă și numai dacă $\neg\varphi$ nu este satisfiabilă.

Demonstrație

Exercițiu!

Logica clauzelor definite

Alegem un fragment al logicii de ordinul I astfel:

- Renunțăm la cuantificatori (dar păstrăm variabilele)
- Renunțăm la $\neg, \vee$ (dar păstrăm $\wedge, \rightarrow$)
- Singurele formule admise sunt de forma:
  - $P(t_1, \ldots, t_n)$, adică formule atomice
  - $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \alpha$,
    unde $\alpha_1, \ldots, \alpha_n, \alpha$ sunt formule atomice.

Astfel de formule se numesc clauze definite (sau clauze Horn).

Acest fragment al logicii de ordinul I se numește logica clauzelor definite (sau logica clauzelor Horn).

Programare logica

- Presupunem că putem reprezenta cunoștințele ca o mulțime de clauze definite Δ și suntem interesați să aflăm răspunsul la o întrebare de forma $\alpha_1 \wedge \ldots \wedge \alpha_n$, unde toate α_i sunt formule atomice.
- Adică vrem să aflăm dacă

$$\Delta \models \alpha_1 \wedge \ldots \wedge \alpha_n$$

- Variabilele din Δ sunt considerate ca fiind cuantificate universal!
- Variabilele din $\alpha_1, \ldots, \alpha_n$ sunt considerate ca fiind cuantificate existențial!

Logica clauzelor definite

Exemplu

Fie următoarele clauze definite:

$$father(jon, ken).$$
$$father(ken, liz).$$
$$father(X, Y) \to ancestor(X, Y)$$
$$father(X, Y) \land ancestor(Y, Z) \to ancestor(X, Z)$$

Putem întreba:

- $ancestor(jon, liz)$
- $ancestor(Q, ken)$ adică $\exists Q\, ancestor(Q, ken)$

Răspunsul la întrebare este dat prin unificare!

Substituții și unificare

Substituții

Definiție

O subtituție σ este o funcție (parțială) de la variabile la termeni, adică

$$\sigma : V \to Trm_{\mathcal{L}}$$

Exemplu

În notația uzuală, $\sigma = \{x/a,\ y/g(w), z/b\}$.

Substituții

- Substituțiile sunt o modalitate de a înlocui variabilele cu alți termeni.
- Substituțiile se aplică simultan pe toate variabilele.

Exemplu

- substituția $\sigma = \{x/a,\ y/g(w), z/b\}$
- $\sigma(P(x, g(x), y)) = P(a, g(a), g(w))$
- substituția $\phi = \{x/y,\ y/g(a)\}$
- $\phi(f(x)) = f(y)$
- $\phi(f(x)) \neq f(g(a))$

Unificare

- Doi termeni t_1 și t_2 se unifică dacă există o substituție ν astfel încât
$$\nu(t_1) = \nu(t_2).$$
- În acest caz, ν se numește unificatorul termenilor t_1 și t_2.
- În programarea logică, unificatorii sunt ingredientele de bază în execuția unui program.

Exemplu

- $t = x + (y \star y) = +(x, \star(y, y))$
- $t' = x + (y \star x) = +(x, \star(y, x))$
- $\nu = \{x/y, y/y\}$
  - $\nu(t) = y + (y \star y)$
  - $\nu(t') = y + (y \star y)$
  - ν este unificator

Unificare

- În programarea logică, unificatorii sunt ingredientele de bază în execuția unui program.

Exemplu

```
father(jon,ken).
father(ken,liz).

ancestor(X,Y):- father(X,Y).
ancestor(X,Z):- father(X,Y), ancestor(Y,Z).

?- ancestor(Q,ken).
Q = jon
```

- Atunci când întrebarea conține variabile, Prolog încearcă să găsească o substituție care face ca predicatul să fie adevărat.

Algoritmul de unificare

- Pentru o mulțime finită de termeni $\{u_1, \ldots, u_n\}$, $n \geq 2$, algoritmul de unificare stabilește dacă există un unificator.

- Există algoritmi mai eficienți, dar îl alegem pe acesta pentru simplitatea sa.

- Algoritmul lucrează cu două liste:
  - Lista soluție: S
  - Lista de rezolvat: R

- Inițial:
  - Lista soluție: $S = \emptyset$
  - Lista de rezolvat: $R = \{u_1 \doteq u_2, \ldots, u_{n-1} \doteq u_n\}$

- $\doteq$ este un simbol nou care ne ajută sa formăm perechi de termeni (ecuații).

Algoritmul de unificare

Algoritmul constă în aplicarea regulilor de mai jos:

- SCOATE
  - orice ecuație de forma $t \doteq t$ din R este eliminată.
- DESCOMPUNE
  - orice ecuație de forma $f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n)$ din R este înlocuită cu ecuațiile $t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$.
- REZOLVĂ
  - orice ecuație de forma $x \doteq t$ sau $t \doteq x$ din R, unde variabila x nu apare în termenul t, este mutată sub forma $x \doteq t$ în S. În toate celelalte ecuații (din R și S), x este înlocuit cu t.

Algoritmul de unificare

Algoritmul se termină normal dacă $R = \emptyset$. În acest caz, S conține un unificator.

Algoritmul este oprit cu concluzia inexistenței unui unificator dacă:

1. În R există o ecuație de forma
$$f(t_1, \ldots, t_n) \doteq g(t'_1, \ldots, t'_k) \text{ cu } f \neq g.$$
2. În R există o ecuație de forma $x \doteq t$ sau $t \doteq x$ și variabila x apare în termenul t.

Algoritmul de unificare - schemă

	Lista soluție S	Lista de rezolvat R
Inițial	$\emptyset$	$t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$
SCOATE	S	R', $t \doteq t$
	S	R'
DESCOMPUNE	S	R', $f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n)$
	S	R', $t_1 \doteq t'_1, \ldots t_n \doteq t'_n$
REZOLVĂ	S	R', $x \doteq t$ sau $t \doteq x$, x nu apare în t
	$x \doteq t$, $S[x/t]$	$R'[x/t]$
Final	S	$\emptyset$

$S[x/t]$: în toate ecuațiile din S, x este înlocuit cu t

Exemplu

Exemplu

- Ecuațiile $\{g(y) \doteq x,\ f(x,h(x),y) \doteq f(g(z),w,z)\}$ au gcu?

S	R	
$\emptyset$	$g(y) \doteq x,\ f(x,h(x),y) \doteq f(g(z),w,z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y),h(g(y)),y) \doteq f(g(z),w,z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq g(z),\ h(g(y)) \doteq w,\ y \doteq z$	REZOLVĂ
$w \doteq h(g(y)),$ $x \doteq g(y)$	$g(y) \doteq g(z),\ y \doteq z$	REZOLVĂ
$y \doteq z, x \doteq g(z),$ $w \doteq h(g(z))$	$g(z) \doteq g(z)$	SCOATE
$y \doteq z, x \doteq g(z),$ $w \doteq h(g(z))$	$\emptyset$	

- $\nu = \{y/z,\ x/g(z),\ w/h(g(z))\}$ este unificator.

Exemplu

Exemplu

- Ecuațiile $\{g(y) \doteq x,\ f(x, h(y), y) \doteq f(g(z), b, z)\}$ au gcu?

S	R	
$\emptyset$	$g(y) \doteq x,\ f(x, h(y), y) \doteq f(g(z), b, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(y), y) \doteq f(g(z), b, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq g(z),\ h(y) \doteq b,\ y \doteq z$	- EȘEC -

- h și b sunt simboluri de operații diferite!
- Nu există unificator pentru ecuațiile din U.

Exemplu

Exemplu

- Ecuațiile $\{g(y) \doteq x,\ f(x, h(x), y) \doteq f(y, w, z)\}$ au gcu?

S	R	
$\emptyset$	$g(y) \doteq x,\ f(x, h(x), y) \doteq f(y, w, z)$	REZOLVĂ
$x \doteq g(y)$	$f(g(y), h(g(y)), y) \doteq f(y, w, z)$	DESCOMPUNE
$x \doteq g(y)$	$g(y) \doteq y,\ h(g(y)) \doteq w,\ y \doteq z$	- EȘEC -

- În ecuația $g(y) \doteq y$, variabila y apare în termenul $g(y)$.
- Nu există unificator pentru ecuațiile din U.

Complexitatea algoritmului

- Problema de unificare

  $R = \{x_1 \doteq f(x_0, x_0), x_2 \doteq f(x_1, x_1), \ldots, x_n \doteq f(x_{n-1}, x_{n-1})\}$

  are unificator $S = \{x_1 \leftarrow f(x_0, x_0), x_2 \leftarrow f(f(x_0, x_0), f(x_0, x_0)), \ldots\}$.

- La pasul Elimină, pentru a verifica că o variabilă x_i nu apare în membrul drept al ecuației (occur check) facem 2^i comparații.

- Algoritmul de unificare prezentat anterior este exponențial. Complexitatea poate fi îmbunătățită printr-o reprezentare eficientă a termenilor.

  K. Knight, Unification: A Multidisciplinary Survey, ACM Computing Surveys, Vol. 21, No. 1, 1989.

Unificare în Prolog

- Ce se întâmplă dacă încercăm să unificăm X cu ceva care conține X? Exemplu: `?- X = f(X).`
- Conform teoriei, acești termeni nu se pot unifica.
- Totuși, multe implementări ale Prolog-ului sar peste această verificare din motive de eficiență.

```
?- X = f(X).
X = f(X).
```

- Putem folosi `unify_with_occurs_check/2`

```
?- unify_with_occurs_check(X,f(X)).
false.
```