Curs 4

Cuprins

1. Logica de ordinul I - sintaxa (recap.)

2. Substituții și unificare

Logica de ordinul I - sintaxa (recap.)

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

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.

Logica de ordinul I

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 $\mathcal{L}$.

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

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

Logica de ordinul I

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.

Logica de ordinul I

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

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 $ari(s) = 1$, $ari(+) = ari(<) = 2$.

Logica de ordinul I

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:

Logica de ordinul I

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

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

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, $s(0)$, $s(s(0))$, $s(x)$, $s(s(x))$, ...,
$+(0,0)$, $+(s(s(0)), +(0, s(0)))$, $+(x, s(0))$, $+(x, s(x))$, ...,

Exemple de formule atomice:

$<(0,0)$, $<(x,0)$, $<(s(s(x)), s(0))$, ...

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, $s(0)$, $s(s(0))$, $s(x)$, $s(s(x))$, …,
$+(0,0)$, $+(s(s(0)), +(0, s(0)))$, $+(x, s(0))$, $+(x, s(x))$, …,

Exemple de formule atomice:

$<(0,0)$, $<(x,0)$, $<(s(s(x)), s(0))$, …

Exemple de formule:

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

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:

$$\begin{array}{l} father(jon, ken). \\ father(ken, liz). \\ father(X, Y) \rightarrow ancestor(X, Y) \\ dauther(X, Y) \rightarrow ancestor(Y, X) \\ ancestor(X, Y) \wedge ancestor(Y, Z) \rightarrow ancestor(X, Z) \end{array}$$

Putem întreba:

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

Logica clauzelor definite

Exemplu

Fie următoarele clauze definite:

$$\begin{array}{l} father(jon, ken). \\ father(ken, liz). \\ father(X, Y) \to ancestor(X, Y) \\ dauther(X, Y) \to ancestor(Y, X) \\ ancestor(X, Y) \land ancestor(Y, Z) \to ancestor(X, Z) \end{array}$$

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.

Substituții

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

Exemplu

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

Substituții

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

Exemplu

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

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 $\phi = \{x/y,\ y/g(a)\}$
- $\phi(f(x)) = f(y)$
- $\phi(f(x)) \neq f(g(a))$

Substituții

Două substituții σ_1 și σ_2 se pot compune

$$\sigma_1 ; \sigma_2$$

(aplicăm întâi σ_1, apoi σ_2).

Substituții

Două substituții σ_1 și σ_2 se pot compune

$$\sigma_1 ; \sigma_2$$

(aplicăm întâi σ_1, apoi σ_2).

Exemplu

- $t = P(u, v, x, y, z)$

Substituții

Două substituții σ_1 și σ_2 se pot compune

$$\sigma_1 ; \sigma_2$$

(aplicăm întâi σ_1, apoi σ_2).

Exemplu

- $t = P(u, v, x, y, z)$
- $\tau = \{x/f(y),\ y/f(a),\ z/u\}$
- $\mu = \{y/g(a),\ u/z,\ v/f(f(a))\}$

Substituții

Două substituții σ_1 și σ_2 se pot compune

$$\sigma_1 ; \sigma_2$$

(aplicăm întâi σ_1, apoi σ_2).

Exemplu

- $t = P(u, v, x, y, z)$
- $\tau = \{x/f(y),\ y/f(a),\ z/u\}$
- $\mu = \{y/g(a),\ u/z,\ v/f(f(a))\}$
- $(\tau;\mu)(t) = \mu(\tau(t)) = \mu(P(u, v, f(y), f(a), u)) =$
  $= P(z, f(f(a)), f(g(a)), f(a), z)$

Substituții

Două substituții σ_1 și σ_2 se pot compune

$$\sigma_1 ; \sigma_2$$

(aplicăm întâi σ_1, apoi σ_2).

Exemplu

- $t = P(u, v, x, y, z)$
- $\tau = \{x/f(y),\ y/f(a),\ z/u\}$
- $\mu = \{y/g(a),\ u/z,\ v/f(f(a))\}$
- $(\tau;\mu)(t) = \mu(\tau(t)) = \mu(P(u, v, f(y), f(a), u)) =$
  $= P(z, f(f(a)), f(g(a)), f(a), z)$
- $(\mu;\tau)(t) = \tau(\mu(t)) = \tau(P(z, f(f(a)), x, g(a), z))$
  $= P(u, f(f(a)), f(y), g(a), u)$

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.

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.
- Un unificator ν pentru t_1 și t_2 este un cel mai general unificator (cgu,mgu) dacă pentru orice alt unificator ν' pentru t_1 și t_2, există o substituție μ astfel încât
$$\nu' = \nu; \mu.$$

Unificator

Exemplu

- $t = x + (y \star y) = +(x, \star(y, y))$
- $t' = x + (y \star x) = +(x, \star(y, x))$

Unificator

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 cgu

Unificator

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 cgu
- $\nu' = \{x/0, y/0\}$
  - $\nu'(t) = 0 + (0 \star 0)$
  - $\nu'(t') = 0 + (0 \star 0)$

Unificator

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 cgu
- $\nu' = \{x/0, y/0\}$
  - $\nu'(t) = 0 + (0 \star 0)$
  - $\nu'(t') = 0 + (0 \star 0)$
  - $\nu' = \nu; \{y/0\}$

Unificator

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 cgu
- $\nu' = \{x/0, y/0\}$
  - $\nu'(t) = 0 + (0 \star 0)$
  - $\nu'(t') = 0 + (0 \star 0)$
  - $\nu' = \nu; \{y/0\}$
  - ν' este unificator, dar nu este gcu

Algoritmul de unificare

- Pentru o mulțime finită de termeni $\{u_1, \ldots, u_n\}$, $n \geq 2$, algoritmul de unificare stabilește dacă există un cgu.
- Există algoritmi mai eficienți, dar îl alegem pe acesta pentru simplitatea sa.

Algoritmul de unificare

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

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

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

Algoritmul de unificare

- Pentru o mulțime finită de termeni $\{u_1, \ldots, u_n\}$, $n \geq 2$, algoritmul de unificare stabilește dacă există un cgu.
- 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: $R = \{u_1 \doteq u_2, \ldots, u_{n-1} \doteq u_n\}$

Algoritmul de unificare

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

Algoritmul de unificare

Algoritmul constă în aplicarea regulilor de mai jos:

- SCOATE
  - orice ecuație de forma $t \doteq t$ din R este eliminată.

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 $t_1 \doteq t'_1, \ldots, t_n \doteq t'_n$.

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 cgu.

Algoritmul de unificare

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

Algoritmul este oprit cu concluzia inexistenței unui cgu dacă:

1. În R există o ecuație de forma
$$f(t_1, \dots, t_n) \doteq g(t'_1, \dots, t'_k) \text{ cu } f \neq g.$$
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?

Exemplu

Exemplu

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

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

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

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
$x \doteq g(y)$	$f(g(y), h(g(y)), y) \doteq f(g(z), w, z)$	

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

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
$x \doteq g(y)$	$g(y) \doteq g(z),\ h(g(y)) \doteq w,\ y \doteq z$	

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

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

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

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 cgu.

Exemplu

Exemplu

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

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 -

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?

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 -

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.

Terminarea algoritmului

Propoziție

Algoritmul de unificare se termină.

Terminarea algoritmului

Propoziție

Algoritmul de unificare se termină.

Demonstrație

- Notăm cu
  - N_1: numărul variabilelor care apar în R
  - N_2: numărul aparițiilor simbolurilor care apar în R
- Este suficient să arătăm că perechea (N_1, N_2) descrește strict în ordine lexicografică la execuția unui pas al algoritmului:

  dacă la execuția unui pas (N_1, N_2) se schimbă în (N_1', N_2'), atunci
  $$(N_1, N_2) \geq_{lex} (N_1', N_2')$$

Demonstrație (cont.)

Fiecare regulă a algoritmului modifică N_1 și N_2 astfel:

	N_1	N_2
SCOATE	$\geq$	$>$
DESCOMPUNE	$=$	$>$
REZOLVĂ	$>$	

- N_1: numărul variabilelor care apar în R
- N_2: numărul aparițiilor simbolurilor care apar în R

□

Corectitudinea algoritmului

Lema 1

Mulțimea unificatorilor pentru ecuațiile din $R \cup S$ nu se modifică prin aplicarea celor trei reguli ale algoritmului de unificare.

Corectitudinea algoritmului

Lema 1

Mulțimea unificatorilor pentru ecuațiile din $R \cup S$ nu se modifică prin aplicarea celor trei reguli ale algoritmului de unificare.

Demonstrație

Analizăm fiecare regulă:

- SCOATE: evident

Corectitudinea algoritmului

Lema 1

Mulțimea unificatorilor pentru ecuațiile din $R \cup S$ nu se modifică prin aplicarea celor trei reguli ale algoritmului de unificare.

Demonstrație

Analizăm fiecare regulă:

- SCOATE: evident
- DESCOMPUNE: Trebuie să arătăm că

$$\begin{array}{ccc} \nu \text{ unificator pt.} & \Leftrightarrow & \nu \text{ unificator pt.} \\ f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n) & & t_i \doteq t'_i, \text{ or. } i = 1, \ldots, n. \end{array}$$

Corectitudinea algoritmului

Lema 1

Mulțimea unificatorilor pentru ecuațiile din $R \cup S$ nu se modifică prin aplicarea celor trei reguli ale algoritmului de unificare.

Demonstrație

Analizăm fiecare regulă:

- SCOATE: evident
- DESCOMPUNE: Trebuie să arătăm că

$$\begin{array}{ccc} \nu \text{ unificator pt.} & \Leftrightarrow & \nu \text{ unificator pt.} \\ f(t_1,\ldots,t_n) \doteq f(t'_1,\ldots,t'_n) & & t_i \doteq t'_i, \text{ or. } i=1,\ldots,n. \end{array}$$

$$\begin{aligned} & \nu \text{ unif. pt. } f(t_1,\ldots,t_n) \doteq f(t'_1,\ldots,t'_n) \\ & \Leftrightarrow \nu(f(t_1,\ldots,t_n)) = \nu(f(t'_1,\ldots,t'_n)) \\ & \Leftrightarrow f(\nu(t_1),\ldots,\nu(t_n)) = f(\nu(t'_1),\ldots,\nu(t'_n)) \\ & \Leftrightarrow \nu(t_i) = \nu(t'_i), \text{or. } i=1,\ldots,n \\ & \Leftrightarrow \nu \text{ unificator pt. } t_i \doteq t'_i, \text{or. } i=1,\ldots,n \end{aligned}$$

Demonstrație (cont.)

- REZOLVĂ:
  - Se observă că orice unificator ν pentru ecuațiile din $R \cup S$, atât înainte cât și după aplicarea regulii REZOLVĂ, trebuie să satisfacă:
    $$\nu(x) = \nu(t).$$
  - Dacă μ este unificator pentru $x \doteq t$ observăm că:
    $$(x \leftarrow t); \mu = \mu$$
    unde $(x \leftarrow t)(x) = t$ și $(x \leftarrow t)(y) = y$ pentru orice $y \neq x \in V$.
    - $((x \leftarrow t); \mu)(x) = \mu(t) = \mu(x)$
    - $((x \leftarrow t); \mu)(y) = \mu(y)$, pentru orice $y \neq x$
  - Deci,

    μ este un unificator pentru ecuațiile din $R \cup S$ înainte de REZOLVĂ

    $\Leftrightarrow$

    μ este un unificator pentru ecuațiile din $R \cup S$ după REZOLVĂ $\square$

Corectitudinea algoritmului

- Pres. că algoritmul de unificare se termină cu $R = \emptyset$.
- Fie $x_i \doteq t_i$, $i = 1, \ldots, k$, ecuațiile din S.
- Variabilele care apar în partea stângă a ecuațiilor din S sunt distincte două câte două și nu mai apar în termenii $t1$, ..., t_k.
- Definim substituția:

$$\nu(x_i) = t_i \text{ pentru orice } i = 1, \ldots, k.$$

- Observăm că $\nu(t_i) = t_i = \nu(x_i)$ oricare $i = 1, \ldots, k$, deci ν este un unificator pentru $R \cup S$.

Corectitudinea algoritmului

Lema 2

ν definit mai sus cf. algoritmului de unificare este cgu pentru $R \cup S$.

Corectitudinea algoritmului

Lema 2

ν definit mai sus cf. algoritmului de unificare este cgu pentru $R \cup S$.

Demonstrație

La ultimul pas $R = \emptyset$ și $\nu(x_i) = t_i$ oricare $i = 1, \ldots, k$

- Fie μ un alt unificator pentru S. Avem
  - $\mu(\nu(x_i)) = \mu(t_i) = \mu(x_i)$, or. $i = 1, \ldots, k$,
  - $\mu(\nu(y)) = \mu(y)$, or. $y \neq x$.

  Deci $\nu; \mu = \mu$. În concluzie, ν este cgu deoarece oricare alt

  unificator se poate scrie ca o compunere a lui ν cu o substituție. □

- Din Lema 1 rezultă că ν este unificator pentru problema inițială $\{u_1 \doteq u_2, \ldots, u_{n-1} \doteq u_n\}$, deci

$$\nu(u_1) = \cdots = \nu(u_n).$$

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

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

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

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

- 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).
```

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.
```

Pe săptămâna viitoare!