Curs ID3

2018-2019

Programare Logică

Cuprins

Forma normală conjunctivă și forma clauzală

Literali. FNC

- În calculul propozițional un literal este o variabilă sau negația unei variabile.

$$\mathit{literal} := \; p \mid \neg p \qquad \text{unde } p \text{ este variabilă propozițională}$$

- În logica de ordinul I un literal este o formulă atomică sau negația unei formule atomice.

$$\mathit{literal} := \; P(t_1, \ldots, t_n) \mid \neg P(t_1, \ldots, t_n)$$

unde $P \in \mathbf{R}, ari(P) = n,$ și $t_1, \ldots, t_n$ sunt termeni.

- Pentru un literal L vom nota cu L^c literalul complement.

  De exemplu, dacă $L = \neg P(x)$ atunci $L^c = P(x)$ și invers.

  O formulă este în formă normală conjunctivă (FNC) dacă este o conjuncție de disjuncții de literali.

FNC în calculul propozițional

- Pentru orice formulă α există o FNC α^{fc} astfel încât $\alpha \dashv\vdash \alpha^{fc}$.
- Pentru o formulă din calculul propozițional determinăm FNC corespunzătoare prin următoarele transformări:
  1. înlocuirea implicațiilor și echivalențelor

$$\varphi \to \psi \quad \dashv\vdash \quad \neg\varphi \vee \psi$$
$$\varphi \leftrightarrow \psi \quad \dashv\vdash \quad (\neg\varphi \vee \psi) \wedge (\neg\psi \vee \varphi)$$

  2. regulile De Morgan

$$\neg(\varphi \vee \psi) \quad \dashv\vdash \quad \neg\varphi \wedge \neg\psi$$
$$\neg(\varphi \wedge \psi) \quad \dashv\vdash \quad \neg\varphi \vee \neg\psi$$

  3. principiului dublei negații

$$\neg\neg\psi \quad \dashv\vdash \quad \psi$$

  4. distributivitatea

$$\varphi \vee (\psi \wedge \chi) \quad \dashv\vdash \quad (\varphi \vee \psi) \wedge (\varphi \vee \chi)$$
$$(\psi \wedge \chi) \vee \varphi \quad \dashv\vdash \quad (\psi \vee \varphi) \wedge (\chi \vee \varphi)$$

FNC în calculul propozițional

Exemplu

Determinați FNC pentru formula

$(\neg p \to \neg q) \to (p \to q)$

$\dashv\vdash \neg(\neg p \to \neg q) \vee (p \to q)$

$\dashv\vdash \neg(p \vee \neg q) \vee (\neg p \vee q)$

$\dashv\vdash (\neg p \wedge q) \vee (\neg p \vee q)$

$\dashv\vdash (\neg p \vee \neg p \vee q) \wedge (q \vee \neg p \vee q)$

$\dashv\vdash (\neg p \vee q) \wedge (q \vee \neg p)$

FNCP și forma clauzală în logica de ordinul I

- O formulă este formă normală conjunctivă prenex (FNCP) dacă
  - este în formă prenex $Q_1 x_1 \dots Q_n x_n \psi$ ($Q_i \in \{\forall, \exists\}$ oricare i)
  - ψ este FNC

- O formulă este formă clauzală dacă este enunț universal și FNCP:

  $\forall x_1 \dots \forall x_n \psi$ unde ψ este FNC

Exemplu

- $\forall y \exists z \left((P(f(y)) \vee Q(z)) \wedge (\neg Q(z) \vee \neg P(g(z)) \vee Q(y)) \right)$

  este FNCP

- $\forall y \forall z \left((P(f(y)) \vee Q(z)) \wedge (\neg Q(z) \vee \neg P(g(z)) \vee Q(y)) \right)$

  este formă clauzală.

Forma clauzală în logica de ordinul I

- Pentru orice formulă φ din logica de ordinul I există o formă clauzală φ^{fc} astfel încât

  φ este satisfiabilă dacă și numai dacă φ^{fc} este satisfiabilă

- Pentru o formulă φ, forma clauzală φ^{fc} se poate calcula astfel:
  1. se determină forma rectificată
  2. se cuantifică universal variabilele libere
  3. se determină forma prenex
  4. se determină forma Skolem

     în acest moment am obținut o formă Skolem $\forall x_1 \ldots \forall x_n \psi$
  5. se determină o FNC ψ' astfel încât $\psi \dashv\vdash \psi'$
  6. φ^{fc} este $\forall x_1 \ldots \forall x_n \psi'$

Literali, clauze, mulțimi de clauze

Clauze

- O clauză este o disjuncție de literali.
- Dacă $L_1, \ldots, L_n$ sunt literali atunci clauza $L_1 \vee \ldots \vee L_n$ o vom scrie ca mulțimea $\{L_1, \ldots, L_n\}$

  clauză = mulțime de literali

- Clauza $C = \{L_1, \ldots, L_n\}$ este satisfiabilă dacă $L_1 \vee \ldots \vee L_n$ este satisfiabilă.
- O clauză C este trivială dacă conține un literal și complementul lui.
- Când $n = 0$ obținem clauza vidă, care se notează $\square$
- Prin definiție, clauza $\square$ nu este satisfiabilă.

Forma clauzală

- Observăm că o FNC este o conjuncție de clauze.
- Dacă $C_1, \ldots, C_k$ sunt clauze atunci $C_1 \wedge \ldots \wedge C_k$ o vom scrie ca mulțimea $\{C_1, \ldots, C_k\}$

FNC = mulțime de clauze

- O mulțime de clauze $\mathcal{C} = \{C_1, \ldots, C_k\}$ este satisfiabilă dacă $C_1 \wedge \ldots \wedge C_k$ este satisfiabilă
- Când $k = 0$ obținem mulțimea de clauze vidă, pe care o notăm $\{\}$
- Prin definiție, mulțimea de clauze vidă $\{\}$ este satisfiabilă.

$\{\}$ este satisfiabilă, dar $\{\square\}$ nu este satisfiabilă

Forma clauzală

- Dacă φ este o formulă în calculul propozițional, atunci
$$\varphi^{fc} = \bigwedge_{i=1}^{k} \bigvee_{j=1}^{n_i} L_{ij} \text{ unde } L_{ij} \text{ sunt literali}$$

- Dacă φ o formulă în logica de ordinul I, atunci
$$\varphi^{fc} = \forall x_1 \ldots \forall x_n \left(\bigwedge_{i=1}^{k} \bigvee_{j=1}^{n_i} L_{ij} \right) \text{ unde } L_{ij} \text{ sunt literali}$$

φ este satisfiabilă dacă și numai dacă

φ^{fc} este satisfiabilă dacă și numai dacă

$\{\{L_{11}, \ldots, L_{1n_1}\}, \ldots, \{L_{k1}, \ldots, L_{kn_k}\}\}$ este satisfiabilă

Forma clauzală

Exemplu

- În calculul propozițional:

  pentru a verifica satisfiabilitatea lui $\varphi := (\neg p \to \neg q) \to (p \to q)$

  determinăm $\varphi^{fc} := (\neg p \vee q) \wedge (q \vee \neg p)$

  și analizăm mulțimea de clauze $\{\{\neg p, q\}, \{q, \neg p\}\}$.

- În logica de ordinul I:

  pentru a verifica satisfiabilitatea formulei
  $\varphi := \forall y \forall z((P(f(y)) \vee Q(z)) \wedge (Q(z) \to (\neg P(g(z)) \vee Q(y))))$

  determinăm
  $\varphi^{fc} := \forall y \forall z\,((P(f(y)) \vee Q(z)) \wedge (\neg Q(z) \vee \neg P(g(z)) \vee Q(y)))$

  și analizăm mulțimea de clauze
  $\{\{P(f(y)), Q(z)\}, \{\neg Q(z), \neg P(g(z)), Q(y)\}\}$

Deducție și satisfiabilitate

Fie $\varphi_1, \ldots, \varphi_n, \varphi$ formule în logica propozițională (enunțuri în calculul cu predicate).

$\{\varphi_1, \ldots, \varphi_n\} \models \varphi$ este echivalent cu

$\models \varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \varphi$ este echivalent cu

$\models \neg\varphi_1 \vee \ldots \neg\varphi_n \vee \varphi$ este echivalent cu

$\varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\varphi$ este satisfiabilă

Pentru a cerceta satisfiabilitatea este suficient să studiem forme clauzale

$$\{\{L_{11}, \ldots, L_{1n_1}\}, \ldots, \{L_{k1}, \ldots, L_{kn_k}\}\}$$

atât în logica propozițională, cât și în calculul cu predicate.

Rezoluție

Rezoluția este o metodă de verificare a satisfiabilității formelor clauzale.

- Rezoluția în calculul propozițional (recap.)
- Rezoluția în logica de ordinul I
  - cazul clauzelor fără variabile
  - cazul general

Rezoluția în calculul propozițional (recap.)

Regula rezoluției

$$Rez \quad \frac{C_1 \cup \{p\}, C_2 \cup \{\neg p\}}{C_1 \cup C_2}$$

unde C_1, C_2 clauze, iar p este variabila propozițională astfel încât $\{p, \neg p\} \cap C_1 = \emptyset$ și $\{p, \neg p\} \cap C_2 = \emptyset$.

Propoziție

Regula *Rez* păstrează satisfiabilitatea. Sunt echivalente:

- mulțimea de clauze $\{C_1 \cup \{p\}, C_2 \cup \{\neg p\}\}$ este satisfiabilă,
- clauza $C_1 \cup C_2$ este satisfiabilă.

Exemplu

$$\frac{\{p, \neg q\}, \{\neg p, q\}}{\{q, \neg q\}}$$

Este mulțimea de clauze $\{\{p, \neg q\}, \{\neg p, q\}\}$ satisfiabilă?

Derivare prin rezoluție

Fie $\mathcal{C}$ o mulțime de clauze. O derivare prin rezoluție din $\mathcal{C}$ este o secvență finită de clauze astfel încât fiecare clauză este din $\mathcal{C}$ sau rezultă din clauzele anterioare prin rezoluție (este rezolvent).

Exemplu

Fie $\mathcal{C} = \{\{\neg q, \neg p\}, \{q\}, \{p\}\}$ o mulțime de clauze. O derivare prin rezoluție pentru $\square$ din $\mathcal{C}$ este

$$
\begin{array}{lll}
C_1 = & \{\neg q, \neg p\} & \\
C_2 = & \{q\} & \\
C_3 = & \{\neg p\} & (Rez, C_1, C_2) \\
C_4 = & \{p\} & \\
C_5 = & \square & (Rez, C_3, C_4)
\end{array}
$$

Teorema de completitudine

$\models \varphi$ dacă și numai dacă există o derivare prin rezoluție a lui $\square$ din $(\neg\varphi)^{fc}$.

Procedura Davis-Putnam DPP (informal)

Intrare: o mulțime $\mathcal{C}$ de clauze

Se repetă următorii pași:

- se elimină clauzele triviale
- se alege o variabilă p
- se adaugă la mulțimea de clauze toți rezolvenții obținuti prin aplicarea *Rez* pe variabila p
- se șterg toate clauzele care conțin p sau $\neg p$

Ieșire: dacă la un pas s-a obținut $\square$, mulțimea $\mathcal{C}$ nu este satisfiabilă; altfel $\mathcal{C}$ este satisfiabilă.

Procedura Davis-Putnam DPP

Exemplu

Este $\mathcal{C}_0 = \{\{p, \neg r\}, \{q, p\}, \{q, \neg p, r\}, \{q, \neg r\}\}$ *satisfiabilă?*

Alegem variabila r și selectăm $\mathcal{C}_0^r := \{\{q, \neg p, r\}\}$,
$\mathcal{C}_0^{\neg r} := \{\{p, \neg r\}, \{q, \neg r\}\}$.
Mulțimea rezolvenților posibili este $\mathcal{R}_0 := \{\{q, \neg p, p\}, \{q, \neg p\}\}$;
Se observă că $p, \neg p \in \{q, \neg p, p\}$ deci $\mathcal{R}_0 := \{\{q, \neg p\}\}$

Se elimină clauzele în care apare r și se adaugă noii rezolvenți
$\mathcal{C}_1 := \{\{q, p\}, \{q, \neg p\}\}$

Alegem variabila q și selectăm $\mathcal{C}_1^q := \{\{q, p\}, \{q, \neg p\}\}$, $\mathcal{C}_1^{\neg q} := \emptyset$.
Mulțimea rezolvenților posibili este vidă $\mathcal{R}_1 := \emptyset$.

Se elimină clauzele în care apare q și se adaugă noii rezolvenți
$\mathcal{C}_2 := \{\}$ mulțimea de clauze vidă

Deoarece $\{\}$ este satisfiabilă, rezultă că $\mathcal{C}_0$ este satisfiabilă.

Atenție! La fiecare pas se alege pentru prelucrare o singură variabilă.

Rezoluția în logica de ordinul I

Clauze închise

- Fie C o clauză. Spunem că C' este o instață a lui C dacă există o substituție $\theta : V \to Trm_{\mathcal{L}}$ astfel încât $C' = \theta(C)$.

  Spunem că C' este o instanță închisă a lui C dacă există o substituție $\theta : V \to T_{\mathcal{L}}$ such that $C' = \theta(C)$ (C' se obține din C înlocuind variabilele cu termeni din universul Herbrand)

- Fie $\mathcal{C}$ o mulțime de clauze. Definim

$$\mathcal{H}(\mathcal{C}) := \{\theta(C) \mid C \in \mathcal{C}, \theta : V \to T_{\mathcal{L}}\}$$

  $\mathcal{H}(\mathcal{C})$ este mulțimea instanțelor închise ale clauzelor din $\mathcal{C}$.

Teoremă

O mulțime de clauze $\mathcal{C}$ este satisfiabilă dacă și numai dacă $\mathcal{H}(\mathcal{C})$ este satisfiabilă. O mulțime de clauze $\mathcal{C}$ este nesatisfiabilă dacă și numai dacă există o submulțime finită a lui $\mathcal{H}(\mathcal{C})$ care este nesatisfiabilă.

Clauze închise

Exemplu

Cercetați satisfiabilitatea mulțimii de clauze

$$\mathcal{C} = \{\{\neg P(x), Q(x)\}, \{P(y)\}, \{\neg Q(z)\}\}$$

Dacă c este constantă atunci
$\{\{\neg P(c), Q(c)\}, \{P(c)\}, \{\neg Q(c)\}\} \subseteq \mathcal{H}(\mathcal{C})$.
Considerăm toate valorile de adevăr pentru $P(c)$ și $Q(c)$:

$P(c)$	$Q(c)$	$(\neg P(c) \vee Q(c)) \wedge P(c) \wedge \neg Q(c)$
0	0	0
0	1	0
1	0	0
1	1	0

$\{\{\neg P(c), Q(c)\}, \{P(c)\}, \{\neg Q(c)\}\}$ este nesatisfiabilă, deci
$\mathcal{C}$ este nesatisfiabilă.

Putem gândi formulele atomice închise ca variabile propoziționale.

Rezoluția pe clauze închise

$$Rez \ \frac{C_1 \cup \{L\}, C_2 \cup \{\neg L\}}{C_1 \cup C_2}$$

unde C_1, C_2 clauze închise, iar L este o formulă atomică închisă astfel încât $\{L, \neg L\} \cap C_1 = \emptyset$ și $\{L, \neg L\} \cap C_2 = \emptyset$.

Propoziție

Regula *Rez* păstrează satisfiabilitatea. Sunt echivalente:

- mulțimea de clauze $\{C_1 \cup \{L\}, C_2 \cup \{\neg L\}\}$ este satisfiabilă,
- clauza $C_1 \cup C_2$ este satisfiabilă.

Teoremă

Fie φ o formulă arbitrară în logica de ordinul I. Atunci $\vDash \varphi$ dacă și numai dacă există o derivare pentru $\square$ din $\mathcal{H}(\mathcal{C})$ folosind *Rez*, unde $\mathcal{C}$ este mulțimea de clauze asociată lui $(\neg\varphi)^{fc}$.

Rezoluția pe clauze închise

Exemplu

Fie f, g simboluri de funcții unare, P, Q simboluri de predicate unare. Cercetați satisfiabilitatea formulei:

$$\varphi = \forall x((\neg P(x) \vee Q(f(x))) \wedge P(g(x)) \wedge \neg Q(x))$$

Determinăm forma clauzală:

$$\mathcal{C} = \{\{\neg P(x), Q(f(x))\}, \{P(g(x))\}, \{\neg Q(x)\}\}$$

Pentru c o constantă obținem următoarea derivare:

$$\begin{array}{lll} C_1 = & \{\neg P(g(c)), Q(f(g(c)))\} & \\ C_2 = & \{P(g(c))\} & \\ C_3 = & \{Q(f(g(c)))\} & Rez, C_1, C_2 \\ C_4 = & \{\neg Q(f(g(c)))\} & \\ C_5 = & \square & Rez, C_3, C_4 \end{array}$$

Rezoluția pe clauze arbitrare

Observații:

- Unificarea literalilor revine la unificarea argumentelor

  Dacă $\sigma : V \to Trm_{\mathcal{L}}$ o substituție, atunci sunt echivalente
  - $\sigma(P(t_1, \ldots, t_n)) = \sigma(P(t'_1, \ldots, t'_n))$
  - $\sigma(\neg P(t_1, \ldots, t_n)) = \sigma(\neg P(t'_1, \ldots, t'_n))$
  - $\sigma(t_1) = \sigma(t'_1), \ldots, \sigma(t_n) = \sigma(t'_n)$

- Redenumirea variabilelor în clauze păstrează validitatea

Exemplu

Deoarece $\forall x(\varphi \wedge \psi) \dashv\vdash (\forall x\varphi) \wedge (\forall x\psi)$ obținem

$\forall x((P_1(x) \vee P_2(x)) \wedge (Q_1(x) \vee Q_2(x)))$

$\dashv\vdash (\forall x(P_1(x) \vee P_2(x))) \wedge (\forall x(Q_1(x) \vee Q_2(x)))$
$\dashv\vdash (\forall x(P_1(x) \vee P_2(x))) \wedge (\forall y(Q_1(y) \vee Q_2(y)))$

$\dashv\vdash \forall x \forall y(P_1(x) \vee P_2(x)) \wedge (Q_1(y) \vee Q_2(y))$

Rezoluția pe clauze arbitrare

Regula rezoluției pentru clauze arbitrare

$$Rez\ \frac{C_1, C_2}{(\sigma C_1 \setminus \sigma Lit_1) \cup (\sigma C_2 \setminus \sigma Lit_2)}$$

dacă următoarele condiții sunt satisfăcute:

1. C_1, C_2 clauze care nu au variabile comune,
2. $Lit_1 \subseteq C_1$ și $Lit_2 \subseteq C_2$ sunt mulțimi de literali,
3. σ este un unificator pentru Lit_1 și Lit_2^c, adică σ unifică toți literalii din Lit_1 și Lit_2^c.

O clauză C se numește rezolvent pentru C_1 și C_2 dacă există o redenumire de variabile $\theta : V \to V$ astfel încât C_1 și θC_2 nu au variabile comune și C se obține din C_1 și θC_2 prin Rez.

Rezoluția în logica de ordinul I

Exemplu

Găsiți un rezolvent pentru clauzele:

$$\begin{aligned} C_1 &= \{ P(f(x), g(y)),\ Q(x, y)\} \text{ și} \\ C_2 &= \{\neg P(f(f(a)), g(y)),\ Q(f(a), g(y))\} \end{aligned}$$

- redenumim variabilele pentru a satisface condițiile din *Rez*
  $\theta C_2 = \{\neg P(f(f(a)), g(z)),\ Q(f(a), g(z))\}$ unde $\theta = \{y \leftarrow z\}$
- determinăm Lit_1 și Lit_2
  $$Lit_1 = \{ P(f(x), g(y))\} \text{ și } Lit_2 = \{\neg P(f(f(a)), g(z))\}$$
- găsim un unificator σ care este unificator pentru
  $$Lit_1 = \{ P(f(x), g(y))\} \text{ și } Lit_2^c = \{P(f(f(a)), g(z))\}$$
  $$\sigma = \{x \leftarrow f(a), y \leftarrow z\}$$
- Rezolventul este $C = (\sigma C_1 \setminus \sigma Lit_1) \cup (\sigma(\theta C_2) \setminus \sigma Lit_2)$
  $$C = \{Q(f(a), z), Q(f(a), g(z))\}$$

Rezoluția în logica de ordinul I

- Fie $\mathcal{C}$ o mulțime de clauze. O derivare prin rezoluție din mulțimea $\mathcal{C}$ pentru o clauză C este o secvență $C_1, \ldots, C_n$ astfel încât $C_n = C$ și, pentru fiecare $i \in \{1, \ldots, n\}$, $C_i \in \mathcal{C}$ sau C_i este un rezolvent pentru două cauze C_j, C_k cu $j, k < i$.

Teoremă

O mulțime de clauze $\mathcal{C}$ este nesatisfiabilă dacă și numai dacă există o derivare a clauzei vide $\square$ din $\mathcal{C}$ prin *Rez*.

Rezoluția este corectă și completă în calculul cu predicate,
dar nu este procedură de decizie.

Rezoluția în logica de ordinul I

Exemplu

Găsiți o derivare a $\square$ din $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$ unde:

$C_1 = \{ \neg P(x, y),\ P(y, x) \}$
$C_2 = \{ \neg P(x, y),\ \neg P(y, z),\ P(x, z) \}$
$C_3 = \{ P(x, f(x)) \}$
$C_4 = \{ \neg P(x, x) \}$

$C'_3 = \{ P(x_1, f(x_1)) \}$	redenumire în C_3
$C_5 = \{ P(f(x), x) \}$	$Rez,\ \sigma = \{x_1 \leftarrow x, y \leftarrow f(x)\}, C_1, C'_3$
$C''_3 = \{ P(x_2, f(x_2)) \}$	redenumire în C_3
$C_6 = \{ \neg P(f(x), z),\ P(x, z) \}$	$Rez,\ \sigma = \{x_2 \leftarrow x, y \leftarrow f(x)\}, C_2, C''_3$
$C'_5 = \{ P(f(x_3), x_3) \}$	redenumire în C_5
$C_7 = \{ P(x, x) \}$	$Rez,\ \sigma = \{x_3 \leftarrow x, z \leftarrow x\}, C_6, C'_5$
$C'_4 = \{ \neg P(x_4, x_4) \}$	redenumire în C_4
$C_5 = \square$	$Rez,\ \sigma = \{x_4 \leftarrow x\}, C_7, C'_4$

Exemplu: "Did curiosity kill the cat?"

Exemplu

Cercetați dacă din ipotezele (i1)-(i6) se deduce (c), unde

(i1) Jack owns a dog.

(i2) Anyone who owns a dog is a lover of animals.

(i3) Lovers of animals do not kill animals.

(i4) Either Jack killed Tuna or curiosity killed Tuna.

(i5) Tuna is a cat.

(i6) All cats are animals.

(c) Curiosity killed Tuna.

Vom formaliza ipotezele și concluzia în logica de ordinul I si vom face demonstrația folosind rezoluția.

Exemplu: "Did curiosity kill the cat?"

Exemplu

Formalizăm ipotezele și determinăm forma clauzală:

(i1) Jack owns a dog.

$\exists x\,(D(x) \wedge O(jack, x))$ $\qquad \mathbf{R} = \{D, O\}, \mathbf{C} = \{jack\}$

$D(dog) \wedge O(jack, dog)$ $\qquad$ Skolemizare

$\mathbf{R} = \{D, O\}, \mathbf{C} = \{jack, dog\}$

Exemplu: "Did curiosity kill the cat?"

Exemplu

(i2) Anyone who owns a dog is a lover_of_animals.

$\forall x\,(\exists y\,(D(y) \wedge O(x,y))) \rightarrow L(x))$ $\quad\quad$ $\mathbf{R} = \{D, O, L\}, \mathbf{C} = \{jack, dog\}$

$\forall x\,((\neg(\exists y\,(D(y) \wedge O(x,y))) \vee L(x))$

$\forall x\,((\forall y\,(\neg D(y) \vee \neg O(x,y))) \vee L(x))$

$\forall x\,\forall y\,(\neg D(y) \vee \neg O(x,y)) \vee L(x))$

$\neg D(y) \vee \neg O(x,y)) \vee L(x)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

(i3) Lovers_of_animals do not kill animals.

$\forall x\,(L(x) \rightarrow (\forall y\,A(y) \rightarrow \neg K(x,y)))$
$\mathbf{R} = \{D, O, L, A, K\}, \mathbf{C} = \{jack, dog\}$

$\forall x\,(\neg L(x) \vee (\forall y\,\neg A(y) \vee \neg K(x,y)))$

$\forall x\,\forall y(\neg L(x) \vee \neg A(y) \vee \neg K(x,y))$

$\neg L(x) \vee \neg A(y) \vee \neg K(x,y)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

(i4) Either Jack killed Tuna or curiosity killed Tuna.

$K(jack, tuna) \vee K(curiosity, tuna)$

$$\mathbf{R} = \{D, O, L, A, K\}, \mathbf{C} = \{jack, dog, tuna, curiosity\}$$

(i5) Tuna is a cat.

$C(tuna)$

$$\mathbf{R} = \{D, O, L, A, K, C\}, \mathbf{C} = \{jack, dog, tuna, curiosity\}$$

(i6) All cats are animals.

$\forall x\, C(x) \rightarrow A(x)$
$\neg C(x) \vee A(x)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

(i1) $D(dog) \wedge O(jack, dog)$

(i2) $\neg D(y) \vee \neg O(x, y) \vee L(x)$

(i3) $\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$

(i4) $K(jack, tuna) \vee K(curiosity, tuna)$

(i5) $C(tuna)$

(i6) $\neg C(x) \vee A(x)$

(c) $K(curiosity, tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

(i1) $D(dog)$

(i1) $O(jack, dog)$

(i2) $\neg D(y) \vee \neg O(x, y) \vee L(x)$

(i3) $\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$

(i4) $K(jack, tuna) \vee K(curiosity, tuna)$

(i5) $C(tuna)$

(i6) $\neg C(x) \vee A(x)$

($\neg$c) $\neg K(curiosity, tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$

$K(jack, tuna) \vee K(curiosity, tuna)$

$C(tuna)$

$\neg C(x) \vee A(x)$

$\neg K(curiosity, tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$

$K(jack, tuna)$

$C(tuna)$

$\neg C(x) \vee A(x)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$

$K(jack, tuna)$

$C(tuna)$

$\neg C(x) \vee A(x)\ \{x \leftarrow tuna\}$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$

$K(jack, tuna)$

$A(tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg A(y) \vee \neg K(x, y)$ $\{y \leftarrow tuna\}$

$K(jack, tuna)$

$A(tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg K(x, tuna)$

$K(jack, tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(x) \vee \neg K(x, tuna)$ $\{x \leftarrow jack\}$

$K(jack, tuna)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)$

$\neg L(jack)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(x, y) \vee L(x)\ \{x \leftarrow jack\}$

$\neg L(jack)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(jack, y)$

Exemplu: "Did curiosity kill the cat?"

Exemplu

$D(dog)$

$O(jack, dog)$

$\neg D(y) \vee \neg O(jack, y)$ $\{y \leftarrow dog\}$

"Curiosity killed the cat!"

Exemplu

$O(jack, dog)$

$\neg O(jack, dog)$

□

În concluzie, am arătat că

$$\{(\text{i1}), (\text{i2}), (\text{i3}), (\text{i4}), (\text{i5}), (\text{i6})\} \models (\text{c})$$

Clauze Horn

Clauze în logica de ordinul I

$$\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\}$$

unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

- formula corespunzătoare este

$$\forall x_1 \ldots \forall x_m (\neg Q_1 \vee \ldots \vee \neg Q_n \vee P_1 \vee \ldots \vee P_k)$$

  unde $x_1, \ldots, x_m$ sunt toate variabilele care apar în clauză

- echivalent, putem scrie

$$\forall x_1 \ldots \forall x_m (Q_1 \wedge \ldots \wedge Q_n \rightarrow P_1 \vee \ldots \vee P_k)$$

- cuantificarea universală a clauzelor este implicită

$$Q_1 \wedge \ldots \wedge Q_n \rightarrow P_1 \vee \ldots \vee P_k$$

Clauze definite. Programe logice. Clauze Horn

- clauză:

  $\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\}$ sau $Q_1 \wedge \ldots \wedge Q_n \rightarrow P_1 \vee \ldots \vee P_k$

  unde $n, k \geq 0$ și $Q_1, \ldots, Q_n, P_1, \ldots, P_k$ sunt formule atomice.

- clauză program definită: $k = 1$
  - cazul $n > 0$: $Q_1 \wedge \ldots \wedge Q_n \rightarrow P$
  - cazul $n = 0$: $\top \rightarrow P$ (clauză unitate, fapt)

  Program logic definit = mulțime finită de clauze definite

- scop definit (țintă, întrebare): k=0
  - $Q_1 \wedge \ldots \wedge Q_n \rightarrow \bot$

- clauza vidă $\square$: $n = k = 0$

  Clauza Horn = clauză program definită sau clauză scop ($k \leq 1$)

Clauze Horn țintă

- scop definit (țintă, întrebare): $Q_1 \wedge \ldots \wedge Q_n \rightarrow \bot$
  - fie $x_1, \ldots, x_m$ toate variabilele care apar în $Q_1, \ldots, Q_n$

$$\forall x_1 \ldots \forall x_m (\neg Q_1 \vee \ldots \vee \neg Q_n) \leftrightarrows \neg \exists x_1 \ldots \exists x_m (Q_1 \wedge \ldots \wedge Q_n)$$

  - clauza țintă o vom scrie $Q_1, \ldots, Q_n$

Negația unei "întrebări" în PROLOG este clauză Horn țintă.

Programare logica

- Logica clauzelor definite/Logica Horn: un fragment al logicii de ordinul I în care singurele formule admise sunt clauze Horn
  - formule atomice: $P(t_1, \ldots, t_n)$
  - $Q_1 \wedge \ldots \wedge Q_n \rightarrow P$
    unde toate Q_i, P sunt formule atomice, $\top$ sau $\bot$

- Problema programării logice: reprezentăm cunoștințele ca o mulțime de clauze definite KB și suntem interesați să aflăm răspunsul la o întrebare de forma $Q_1 \wedge \ldots \wedge Q_n$, unde toate Q_i sunt formule atomice

$$KB \models Q_1 \wedge \ldots \wedge Q_n$$

  - Variabilele din KB sunt cuantificate universal.
  - Variabilele din $Q_1, \ldots, Q_n$ sunt cuantificate existențial.

Limbajul PROLOG are la bază logica clauzelor Horn.

Logica clauzelor definite

Exemplu

Fie următoarele clauze definite:

$father(jon, ken).$
$father(ken, liz).$
$father(X, Y) \rightarrow ancestor(X, Y)$
$dauther(X, Y) \rightarrow ancestor(Y, X)$
$ancestor(X, Y) \wedge ancestor(Y, Z) \rightarrow ancestor(X, Z)$

Putem întreba:

- $ancestor(jon, liz)$
- dacă există Q astfel încât $ancestor(Q, ken)$
  (adică $\exists Q\, ancestor(Q, ken)$)

Sistem de deducție pentru logica Horn

Sistem de deducție *backchain*

Sistem de deducție pentru clauze Horn

Pentru un program logic definit KB avem

- Axiome: orice clauză din KB
- Regula de deducție: regula *backchain*

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n) \quad (Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \rightarrow P)}{\theta(Q)}$$

unde $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \rightarrow P \in KB$, iar θ este unificator pentru Q și P.

Sistem de deducție

Exemplu

KB conține următoarele clauze definite:

$father(jon, ken)$.
$father(ken, liz)$.
$father(X, Y) \rightarrow ancestor(X, Y)$
$daugther(X, Y) \rightarrow ancestor(Y, X)$
$ancestor(X, Y) \wedge ancestor(Y, Z) \rightarrow ancestor(X, Z)$

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n) \quad (Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \rightarrow P)}{\theta(Q)}$$

unde $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \rightarrow P \in KB$, iar θ este unificator pentru Q și P

Sistem de deducție

Pentru o țintă Q, trebuie să găsim o clauză din KB

$$Q_1 \wedge \ldots \wedge Q_n \rightarrow P,$$

și un unificator θ pentru Q și P. În continuare vom verifica $\theta(Q_1), \ldots, \theta(Q_n)$.

Exemplu

Pentru ținta

$$ancestor(ken, Z),$$

putem folosi o clauză

$$father(Y, X) \rightarrow ancestor(Y, X)$$

cu unificatorul

$$\{Y/ken, X/Z\}$$

pentru a obține o nouă țintă

$$father(ken, Z).$$

Sistem de deducție

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \ldots \quad \theta(Q_n) \quad (Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \rightarrow P)}{\theta(Q)}$$

unde $Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \rightarrow P \in KB$, iar θ este unificator pentru Q și P.

Exemplu

$$\frac{\dfrac{father(ken, liz)}{father(ken, Z)} \quad (father(Y, X) \rightarrow ancestor(Y, X))}{ancestor(ken, Z)}$$

Puncte de decizie în programarea logica

Având doar această regulă, care sunt punctele de decizie în căutare?

- Ce clauză să alegem.
  - Pot fi mai multe clauze a căror parte dreaptă se potrivește cu o țintă.
  - Aceasta este o alegere de tip **SAU**: este suficient ca oricare din variante să reușească.
- Ordinea în care rezolvăm noile ținte.
  - Aceasta este o alegere de tip **ȘI:** trebuie arătate toate țintele noi.
  - Ordinea în care le rezolvăm poate afecta găsirea unei derivări, depinzând de strategia de căutare folosită.

Strategia de căutare din Prolog

- Regula *backchain* conduce la un sistem de deducție complet:

  Pentru o mulțime de clauze KB și o țintă Q,
  dacă $KB \models Q$,
  atunci există o derivare a lui Q folosind regula *backchain*.

- Strategia de căutare din Prolog este de tip *depth-first*,
  - de sus în jos
    - pentru alegerile de tip **SAU**
    - alege clauzele în ordinea în care apar în program
  - de la stânga la dreapta
    - pentru alegerile de tip **ȘI**
    - alege noile ținte în ordinea în care apar în clauza aleasă

Sistemul de inferență backchain

Notăm cu $KB \vdash_b Q$ dacă există o derivare a lui Q din KB folosind sistemul de inferență *backchain*.

Teoremă

Sistemul de inferență backchain *este corect și complet pentru formule atomice fără variabile Q.*

$$KB \models Q \quad \text{dacă și numai dacă} \quad KB \vdash_b Q$$

Sistemul de inferență *backchain* este corect și complet și pentru formule atomice cu variabile Q:

$$KB \models \exists x Q(x) \text{ dacă și numai dacă } KB \vdash_b \theta(Q)$$
$$\text{pentru o substituție } \theta.$$

Corectitudine

Propoziție (Corectitudine)

Dacă $KB \vdash_b Q$, atunci $KB \models Q$.

Demonstrație [schiță]

- Presupunem că toate clauzele din KB sunt adevărate.
- Ne uităm, inductiv, la cazurile care pot să apară în derivarea lui Q.

□

Completitudine

Teoremă (Completitudine)

Dacă $KB \models Q$, *atunci* $KB \vdash_b Q$.

Trebuie să arătăm că

pentru orice structură și orice interpretare,
dacă orice clauză din KB este adevărată, atunci și Q este adevărată,

$\Downarrow$

există o derivare a lui Q din KB.

Demonstrația este mai simplă deoarece

este suficient să ne uităm la modelul Herbrand!

Rezoluție SLD

Regula *backchain* și rezoluția SLD

- Regula *backchain* este implementată în programarea logică prin rezoluția SLD (Selected, Linear, Definite).
- Prolog are la bază rezoluția SLD.

Rezoluția SLD

Fie KB o mulțime de clauze definite.

$$\text{SLD}\ \boxed{\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}}$$

unde

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din KB (în care toate variabilele au fost redenumite) și
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este unificator pentru Q_i și Q

Rezoluția SLD

Exemplu

```
father(eddard,sansa).
father(eddard,jonSnow).

stark(eddard).
stark(catelyn).

stark(X) :- father(Y,X),
stark(Y).
```

```
?- stark(jonSnow)
```

$$\text{SLD}\ \frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}$$

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din KB
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este unificator pentru Q_i și Q.

Rezoluția SLD

Exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$

$stark(eddard)$
$stark(catelyn)$

$stark(X) \vee \neg father(Y, X) \vee \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{\neg father(Y, jonSnow) \vee \neg stark(Y)}$$

$$\theta(X) = jonSnow$$

SLD
$$\frac{\neg Q_1 \vee \cdots \vee \neg Q_i \vee \cdots \vee \neg Q_n}{\theta(\neg Q_1 \vee \cdots \vee \neg P_1 \vee \cdots \vee \neg P_m \vee \cdots \vee \neg Q_n)}$$

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din KB
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este unificator pentru Q_i și Q.

Rezoluția SLD

Exemplu

$father(eddard, sansa)$
$father(eddard, jonSnow)$
$stark(eddard)$
$stark(catelyn)$
$stark(X) \vee \neg father(Y, X) \vee \neg stark(Y)$

$$\frac{\neg stark(jonSnow)}{\neg father(Y, jonSnow) \vee \neg stark(Y)}$$

$$\frac{\neg father(Y, jonSnow) \vee \neg stark(Y)}{\neg stark(eddard)}$$

$$\frac{\neg stark(eddard)}{\square}$$

Rezoluția SLD

Fie KB o mulțime de clauze definite și $Q_1 \wedge \ldots \wedge Q_m$ o întrebare, unde Q_i sunt formule atomice.

- O derivare din KB prin rezoluție SLD este o secvență

$$G_0 := \neg Q_1 \vee \ldots \vee \neg Q_m, \quad G_1, \quad \ldots, \quad G_k, \ldots$$

  în care G_{i+1} se obține din G_i prin regula SLD.
- Dacă există un k cu $G_k = \square$ (clauza vidă), atunci derivarea se numește SLD-respingere.

Rezoluția SLD

Teoremă (Completitudinea SLD-rezoluției)

Sunt echivalente:

- *există o SLD-respingere a lui* $Q_1 \wedge \ldots \wedge Q_m$ *din* KB,
- $KB \vdash_b Q_1 \wedge \ldots \wedge Q_m$,
- $KB \models Q_1 \wedge \cdots \wedge Q_m$.

Demonstrație

Rezultă din completitudinea sistemului de deducție backchain și din faptul că:

există o SLD-respingere a lui $Q_1 \wedge \ldots \wedge Q_m$ din KB

ddacă

$KB \vdash_b Q_1 \wedge \ldots \wedge Q_m$

□

Rezoluția SLD - arbori de căutare

Arbori SLD

- Presupunem că avem o mulțime de clauze definite KB și o țintă $G_0 = \neg Q_1 \vee \ldots \vee \neg Q_m$
- Construim un arbore de căutare (arbore SLD) astfel:
  - Fiecare nod al arborelui este o țintă (posibil vidă)
  - Rădăcina este G_0
  - Dacă arborele are un nod G_i, iar G_{i+1} se obține din G_i folosind regula SLD folosind o clauză $C_i \in KB$, atunci nodul G_i are copilul G_{i+1}. Muchia dintre G_i și G_{i+1} este etichetată cu C_i.
- Dacă un arbore SLD cu rădăcina G_0 are o frunză $\square$ (clauza vidă), atunci există o SLD-respingere a lui G_0 din KB.

Rezoluția SLD

Exemplu

- Fie KB următoarea mulțime de clauze definite:
  1. $grandfather(X, Z) :- father(X, Y), parent(Y, Z)$
  2. $parent(X, Y) :- father(X, Y)$
  3. $parent(X, Y) :- mother(X, Y)$
  4. $father(ken, diana)$
  5. $mother(diana, brian)$

- Găsiți o respingere din KB pentru

$$:- grandfather(ken, Y)$$

Rezoluția SLD

Exemplu

- Fie KB următoarea mulțime de clauze definite:
  1. $grandfather(X, Z) \vee \neg father(X, Y) \vee \neg parent(Y, Z)$
  2. $parent(X, Y) \vee \neg father(X, Y)$
  3. $parent(X, Y) \vee \neg mother(X, Y)$
  4. $father(ken, diana)$
  5. $mother(diana, brian)$

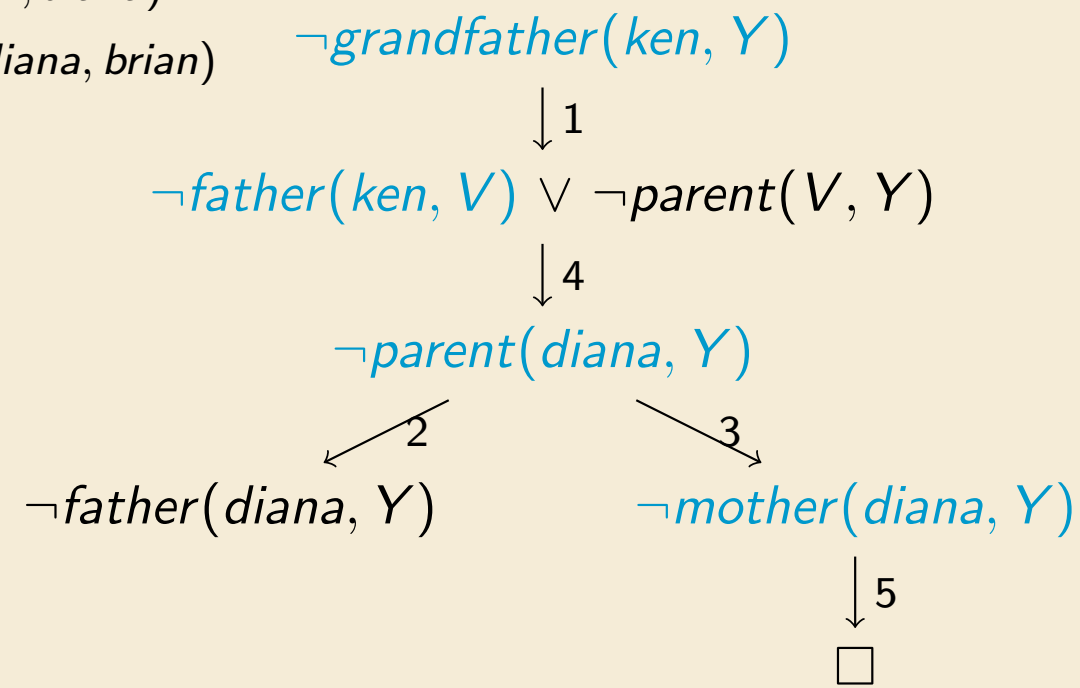
- Găsiți o respingere din KB pentru

$$\neg grandfather(ken, Y)$$

Rezoluția SLD

Exemplu

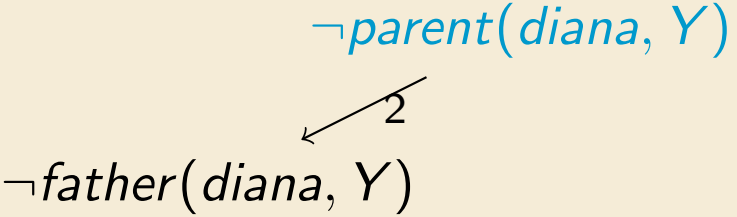
1. $grandfather(X, Z) \vee \neg father(X, Y) \vee \neg parent(Y, Z)$
2. $parent(X, Y) \vee \neg father(X, Y)$
3. $parent(X, Y) \vee \neg mother(X, Y)$
4. $father(ken, diana)$
5. $mother(diana, brian)$



Rezoluția SLD

Exemplu

2 $parent(X, Y) \vee \neg father(X, Y)$



Aplicarea SLD:

- redenumesc variabilele: $parent(X, Y_2) \vee \neg father(X, Y_2)$
- determin unificatorul: $\theta = X/diana, Y_2/Y$
- aplic regula: $\dfrac{\neg parent(diana, Y)}{\neg father(diana, Y)}$

Limbajul Prolog

- Am arătat că sistemul de inferență din spatele Prolog-ului este complet.
  - Dacă o întrebare este consecință logică a unei mulțimi de clauze, atunci există o derivare a întrebării.
- Totuși, strategia de căutate din Prolog este incompletă!
  - Chiar dacă o întrebare este consecință logică a unei mulțimi de clauze, Prolog nu găsește mereu o derivare a întrebării.

Limbajul Prolog

Exemplu

```
warmerClimate :- albedoDecrease.
warmerClimate :- carbonIncrease.
iceMelts :- warmerClimate.
albedoDecrease :- iceMelts.
carbonIncrease.

?- iceMelts.
! Out of local stack
```

Limbajul Prolog

Exemplu

```
warmerClimate :- albedoDecrease.
warmerClimate :- carbonIncrease.
iceMelts :- warmerClimate.
albedoDecrease :- iceMelts.
carbonIncrease.

?- iceMelts.
! Out of local stack
```

Exemplu (cont.)

Există o derivare a lui *iceMelts* în sistemul de deducție din clauzele:

$$\begin{aligned} \mathit{albedoDecrease} &\rightarrow \mathit{warmerClimate} \\ \mathit{carbonIncrease} &\rightarrow \mathit{warmerClimate} \\ \mathit{warmerClimate} &\rightarrow \mathit{iceMelts} \\ \mathit{iceMelts} &\rightarrow \mathit{albedoDecrease} \\ \top &\rightarrow \mathit{carbonIncrease} \end{aligned}$$

$$\dfrac{\dfrac{\mathit{carbonInc}. \qquad \mathit{carbonInc}. \rightarrow \mathit{warmerClim}.}{\mathit{warmerClim}.} \qquad \mathit{warmerClim}. \rightarrow \mathit{iceMelts}}{\mathit{iceMelts}}$$

Bibliografie

- M. Ben-Ari, **Mathematical Logic for Computer Science**, Springer, 2012.
- P. Blackburn, J. Bos, K. Striegnitz, **Learn Prolog now**, College Publications, 2006.
- M. Huth, M. Ryan, **Logic in Computer Science: Modelling and Reasoning about Systems**, Cambridge University Press New York, 2004.
- J.W. Lloyd, **Foundations of Logic Programming**, Springer, 1987.
- Logic Programming, The University of Edinburgh, https://www.inf.ed.ac.uk/teaching/courses/lp/

Succes la examen!