Curs 8

Cuprins

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:

$$\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\} \quad \text{sau} \quad Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k$$

  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 $n > 0$: $Q_1 \wedge \ldots \wedge Q_n \to P$
  - cazul $n = 0$: $\top \to P$ (clauză unitate, fapt)

  Program logic definit = mulțime finită de clauze definite

Clauze definite. Programe logice. Clauze Horn

- clauză:

$$\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\} \quad \text{sau} \quad Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k$$

  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 $n > 0$: $Q_1 \wedge \ldots \wedge Q_n \to P$
  - cazul $n = 0$: $\top \to P$ (clauză unitate, fapt)

  Program logic definit = mulțime finită de clauze definite

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

Clauze definite. Programe logice. Clauze Horn

- clauză:

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

  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 $\square$: $n = k = 0$

Clauze definite. Programe logice. Clauze Horn

- clauză:

  $$\{\neg Q_1, \ldots, \neg Q_n, P_1, \ldots, P_k\} \quad \text{sau} \quad Q_1 \wedge \ldots \wedge Q_n \to P_1 \vee \ldots \vee P_k$$

  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 $n > 0$: $Q_1 \wedge \ldots \wedge Q_n \to P$
  - cazul $n = 0$: $\top \to P$ (clauză unitate, fapt)

  Program logic definit = mulțime finită de clauze definite

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

- 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 $Q_1, \ldots, Q_n$

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

  - 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: $P(t_1, \ldots, t_n)$
  - $Q_1 \land \ldots \land Q_n \to P$
    unde toate Q_i, P sunt formule atomice, $\top$ sau $\perp$

Programare logica

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

- Problema programării logice: reprezentăm cunoștințele ca o mulțime de clauze definite T 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

$$T \models Q_1 \wedge \ldots \wedge Q_n$$

Programare logica

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

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

$$T \models Q_1 \land \ldots \land Q_n$$

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

Programare logica

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

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

$$T \models Q_1 \land \ldots \land Q_n$$

  - Variabilele din T 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:

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

Putem întreba:

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

Cel mai mic model Herbrand

Modele Herbrand

Fie $\mathcal{L}$ un limbaj de ordinul I.

- Presupunem că are cel puțin un simbol de constantă!
- Dacă nu are, adăugăm un simbol de constantă.

Modele Herbrand

Fie $\mathcal{L}$ un limbaj de ordinul I.

- Presupunem că are cel puțin un simbol de constantă!
- Dacă nu are, adăugăm un simbol de constantă.

Universul Herbrand este mulțimea $T_{\mathcal{L}}$ a tututor termenilor lui $\mathcal{L}$ fără variabile.

Modele Herbrand

Fie $\mathcal{L}$ un limbaj de ordinul I.

- Presupunem că are cel puțin un simbol de constantă!
- Dacă nu are, adăugăm un simbol de constantă.

Universul Herbrand este mulțimea $T_{\mathcal{L}}$ a tututor termenilor lui $\mathcal{L}$ fără variabile.

Un model Herbrand este o structură $\mathcal{H} = (T_{\mathcal{L}}, \mathbf{F}^{\mathcal{H}}, \mathbf{P}^{\mathcal{H}}, \mathbf{C}^{\mathcal{H}})$, unde

- pentru orice simbol de constantă c, $c^{\mathcal{H}} = c$
- pentru orice simbol de funcție f de aritate n,
  $f^{\mathcal{H}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$
- pentru orice simbol de relație R de aritate n, $R^{\mathcal{H}}(t_1, \ldots, t_n) \subseteq (T_{\mathcal{L}})^n$

Modele Herbrand

Definim o ordine între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$ este definită astfel:

$$\text{\textit{pentru orice} } R \in \mathbf{R} \text{ \textit{cu} } ari(R) = n \text{ \textit{și pentru orice termeni} } t_1, \ldots, t_n$$
$$\text{dacă } \mathcal{H}_1 \models R(t_1, \ldots, t_n), \text{ atunci } \mathcal{H}_2 \models R(t_1, \ldots, t_n)$$

Modele Herbrand

Definim o ordine între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$ este definită astfel:

pentru orice $R \in \mathbf{R}$ *cu* $ari(R) = n$ *și pentru orice termeni* $t_1, \ldots, t_n$
dacă $\mathcal{H}_1 \models R(t_1, \ldots, t_n)$, atunci $\mathcal{H}_2 \models R(t_1, \ldots, t_n)$

Semantica unui program logic definit T este dată de cel mai mic model Herbrand al lui T!

Modele Herbrand

Definim o ordine între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$ este definită astfel:

pentru orice $R \in \mathbf{R}$ *cu* $ari(R) = n$ *și pentru orice termeni* $t_1, \ldots, t_n$
dacă $\mathcal{H}_1 \models R(t_1, \ldots, t_n)$, atunci $\mathcal{H}_2 \models R(t_1, \ldots, t_n)$

Semantica unui program logic definit T este dată de cel mai mic model Herbrand al lui T!

- De ce există? Este unic?

Modele Herbrand

Definim o ordine între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$ este definită astfel:

pentru orice $R \in \mathbf{R}$ *cu* $ari(R) = n$ *și pentru orice termeni* $t_1, \ldots, t_n$
dacă $\mathcal{H}_1 \models R(t_1, \ldots, t_n)$, atunci $\mathcal{H}_2 \models R(t_1, \ldots, t_n)$

Semantica unui program logic definit T este dată de
cel mai mic model Herbrand al lui T!

- De ce există? Este unic?
- Definim $\mathcal{LH}_T := \bigcap\{\mathcal{H} \mid \mathcal{H}$ model Herbrand pentru $T\}$

Modele Herbrand

Definim o ordine între modelele Herbrand:

$\mathcal{H}_1 \leq \mathcal{H}_2$ este definită astfel:

pentru orice $R \in \mathbf{R}$ *cu* $ari(R) = n$ *și pentru orice termeni* $t_1, \ldots, t_n$
dacă $\mathcal{H}_1 \models R(t_1, \ldots, t_n)$, atunci $\mathcal{H}_2 \models R(t_1, \ldots, t_n)$

Semantica unui program logic definit T este dată de
cel mai mic model Herbrand al lui T!

- De ce există? Este unic?
- Definim $\mathcal{LH}_T := \bigcap\{\mathcal{H} \mid \mathcal{H} \text{ model Herbrand pentru } T\}$
- $\mathcal{LH}_T \models T$.
  Exercițiu: De ce?

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Demonstrație

$T \models Q$
ddacă $T \cup \{\neg Q\}$ nesatisfiabilă

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Demonstrație

$T \models Q$
ddacă $T \cup \{\neg Q\}$ nesatisfiabilă
ddacă $T \cup \{\neg Q\}$ nu are niciun model Herbrand

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Demonstrație

$T \models Q$
ddacă $T \cup \{\neg Q\}$ nesatisfiabilă
ddacă $T \cup \{\neg Q\}$ nu are niciun model Herbrand
ddacă $\neg Q$ este falsă în toate modelele Herbrand ale lui T

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Demonstrație

$T \models Q$
ddacă $T \cup \{\neg Q\}$ nesatisfiabilă
ddacă $T \cup \{\neg Q\}$ nu are niciun model Herbrand
ddacă $\neg Q$ este falsă în toate modelele Herbrand ale lui T
ddacă Q este adevărată în toate modelele Herbrand ale lui T

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Demonstrație

$T \models Q$
ddacă $T \cup \{\neg Q\}$ nesatisfiabilă
ddacă $T \cup \{\neg Q\}$ nu are niciun model Herbrand
ddacă $\neg Q$ este falsă în toate modelele Herbrand ale lui T
ddacă Q este adevărată în toate modelele Herbrand ale lui T
ddacă Q este adevărată în $\mathcal{LH}_T$

Cel mai mic model Herbrand

Fie T un program logic definit.

Propoziție

Pentru orice formulă atomică Q,

$$T \models Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q.$$

Demonstrație

$T \models Q$
ddacă $T \cup \{\neg Q\}$ nesatisfiabilă
ddacă $T \cup \{\neg Q\}$ nu are niciun model Herbrand
ddacă $\neg Q$ este falsă în toate modelele Herbrand ale lui T
ddacă Q este adevărată în toate modelele Herbrand ale lui T
ddacă Q este adevărată în $\mathcal{LH}_T$

Vom caracteriza cel mai mic model Herbrand $\mathcal{LH}_T$ printr-o construcție de punct fix.

Cel mai mic model Herbrand

- O instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.

Cel mai mic model Herbrand

- O instanță de bază a unei clauze $Q_1(x_1) \land \ldots \land Q_n(x_n) \to P(y)$ este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.

- Pentru o mulțime de clauze definite T, o formulă atomică P și o mulțime de formule atomice X,

$$\mathit{oneStep}_T(P, X) \text{ este adevărat}$$

Cel mai mic model Herbrand

- O instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.

- Pentru o mulțime de clauze definite T, o formulă atomică P și o mulțime de formule atomice X,

  $$oneStep_T(P, X) \text{ este adevărat}$$

  dacă există o instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ din T astfel încât P este instanța lui $P(y)$ și instanța lui $Q_i(x_i)$ este în X, pentru orice $i = 1, \ldots, n$.

Cel mai mic model Herbrand

- O instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.

- Pentru o mulțime de clauze definite T, o formulă atomică P și o mulțime de formule atomice X,

  $$oneStep_T(P, X) \text{ este adevărat}$$

  dacă există o instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ din T astfel încât P este instanța lui $P(y)$ și instanța lui $Q_i(x_i)$ este în X, pentru orice $i = 1, \ldots, n$.

- Baza Herbrand $B_{\mathcal{L}}$ este mulțimea formulelor atomice fără variabile.

Cel mai mic model Herbrand

- O instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ este rezultatul obținut prin înlocuirea variabilelor cu termeni fără variabile.

- Pentru o mulțime de clauze definite T, o formulă atomică P și o mulțime de formule atomice X,

  $$oneStep_T(P, X) \text{ este adevărat}$$

  dacă există o instanță de bază a unei clauze $Q_1(x_1) \wedge \ldots \wedge Q_n(x_n) \to P(y)$ din T astfel încât P este instanța lui $P(y)$ și instanța lui $Q_i(x_i)$ este în X, pentru orice $i = 1, \ldots, n$.

- Baza Herbrand $B_{\mathcal{L}}$ este mulțimea formulelor atomice fără variabile.

- Pentru o mulțime de clauze definite T, definim

  $$f_T : \mathcal{P}(B_{\mathcal{L}}) \to \mathcal{P}(B_{\mathcal{L}})$$

  $$f_T(X) = \{P \in B_{\mathcal{L}} \mid oneStep_T(P, X)\}$$

Cel mai mic model Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un sb. de constantă 0, un sb. de funcție unară s și un sb. de relație unar par

Cel mai mic model Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un sb. de constantă 0, un sb. de funcție unară s și un sb. de relație unar par
- $T_{\mathcal{L}} = \{0, s(0), s(s(0)), \ldots\}$

Cel mai mic model Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un sb. de constantă 0, un sb. de funcție unară s și un sb. de relație unar par
- $T_{\mathcal{L}} = \{0, s(0), s(s(0)), \ldots\}$
- Fie T mulțimea clauzelor:

$$par(0) \qquad\qquad par(x) \to par(s(s(x)))$$

Cel mai mic model Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un sb. de constantă 0, un sb. de funcție unară s și un sb. de relație unar par
- $T_{\mathcal{L}} = \{0, s(0), s(s(0)), \ldots\}$
- Fie T mulțimea clauzelor:

$$par(0) \qquad\qquad par(x) \to par(s(s(x)))$$

- Instanțe de bază:
  - $par(0) \to par(s(s(0)))$
  - $par(s(0)) \to par(s(s(s(0))))$

Cel mai mic model Herbrand

Exemplu

- Fie $\mathcal{L}$ un limbaj cu un sb. de constantă 0, un sb. de funcție unară s și un sb. de relație unar par
- $T_{\mathcal{L}} = \{0, s(0), s(s(0)), \ldots\}$
- Fie T mulțimea clauzelor:

$$par(0) \qquad\qquad par(x) \to par(s(s(x)))$$

- Instanțe de bază:
  - $par(0) \to par(s(s(0)))$
  - $par(s(0)) \to par(s(s(s(0))))$

- $f_T(\{\}) = \{par(0)\}$
- $f_T(\{par(0)\}) = \{par(0), par(s(s(0)))\}$
- $f_T(\{par(s(0))\}) = \{par(0), par(s(s(s(0))))\}$
- $f_T(\{par(s(s(0)))\}) = \{par(0), par(s(s(s(s(0)))))\}$

Cel mai mic model Herbrand

Fie T un program logic definit.

- Exercițiu: f_T este continuă
- Din teorema Knaster-Tarski, f_T are un cel mai mic punct fix FP_T.
- FP_T este reuniunea tuturor mulțimilor

$$f_T(\{\}),\ f_T(f_T(\{\})),\ f_T(f_T(f_T(\{\}))), \ldots$$

Cel mai mic model Herbrand

Fie T un program logic definit.

- Exercițiu: f_T este continuă
- Din teorema Knaster-Tarski, f_T are un cel mai mic punct fix FP_T.
- FP_T este reuniunea tuturor mulțimilor

$$f_T(\{\}),\ f_T(f_T(\{\})),\ f_T(f_T(f_T(\{\}))),\ldots$$

Propoziție (caracterizarea $\mathcal{LH}_T$)

Pentru orice $R \in \mathbf{R}$ cu $ari(R) = n$ și pentru orice $t_1, \ldots, t_n$ termeni, avem

$$(t_1, \ldots, t_n) \in R^{\mathcal{LH}_T} \text{ ddacă } R(t_1, \ldots, t_n) \in FP_T$$

Sistem de deducție pentru logica Horn

Sistem de deducție *backchain*

Sistem de deducție pentru clauze Horn

Pentru un program logic definit T avem

Sistem de deducție *backchain*

Sistem de deducție pentru clauze Horn

Pentru un program logic definit T avem

- Axiome: orice clauză din T

Sistem de deducție *backchain*

Sistem de deducție pentru clauze Horn

Pentru un program logic definit T avem

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

$$\frac{\theta(Q_1) \quad \theta(Q_2) \quad \dots \quad \theta(Q_n) \quad (Q_1 \land Q_2 \land \dots \land Q_n \to P)}{\theta(Q)}$$

unde $Q_1 \land Q_2 \land \dots \land Q_n \to P \in T$, iar θ este cgu pentru Q și P.

Sistem de deducție

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

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

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

Sistem de deducție

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

$$Q_1 \land \ldots \land Q_n \to P,$$

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

Exemplu

Pentru ținta

$$ancestor(Z, ken),$$

putem folosi o clauză

$$daughter(X, Y) \to ancestor(Y, X)$$

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

pentru a obține o nouă țintă

$$daughter(ken, Z).$$

Puncte de decizie în programarea logica

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

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.

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

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

Proprietăți ale sistemului de inferență

Vești bune!

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

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

Proprietăți ale sistemului de inferență

Vești bune!

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

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

Vești proaste!

- Strategia de căutare din Prolog este incompletă:

  Poate eșua în a găsi o derivare, chiar și când există o derivare!

Sistemul de inferență backchain

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

Teoremă

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

$$T \models Q \quad \textit{dacă și numai dacă} \quad T \vdash_b Q$$

Sistemul de inferență backchain

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

Teoremă

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

$$T \models Q \quad \textit{dacă și numai dacă} \quad T \vdash_b Q$$

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

$$T \models \exists x Q(x) \text{ dacă și numai dacă } T \vdash_b \theta(Q)$$

pentru o substituție θ.

Corectitudine

Propoziție (Corectitudine)

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

Demonstrație [schiță]

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

$\square$

Completitudine

Teoremă (Completitudine)

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

Completitudine

Teoremă (Completitudine)

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

Trebuie să arătăm că

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

$$\Downarrow$$

există o derivare a lui Q din T.

Completitudine

Teoremă (Completitudine)

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

Trebuie să arătăm că

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

$$\Downarrow$$

există o derivare a lui Q din T.

Demonstrația este mai simplă deoarece

este suficient să ne uităm la modelul Herbrand!

Cel mai mic model Herbrand

Teoremă

Pentru orice T un program logic definit și Q o formulă atomică,

$$T \vdash_B Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q \quad \text{ddacă} \quad T \models Q.$$

Cel mai mic model Herbrand

Teoremă

Pentru orice T un program logic definit și Q o formulă atomică,

$$T \vdash_B Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q \quad \text{ddacă} \quad T \models Q.$$

Demonstrație (schiță)

Demonstrăm numai prima echivalență.

- Implicația de la stânga la dreapta rezultă ușor din corectitudinea sistemului de inferență *backchain*.

Cel mai mic model Herbrand

Teoremă

Pentru orice T un program logic definit și Q o formulă atomică,

$$T \vdash_B Q \quad \text{ddacă} \quad \mathcal{LH}_T \models Q \quad \text{ddacă} \quad T \models Q.$$

Demonstrație (schiță)

Demonstrăm numai prima echivalență.

- Implicația de la stânga la dreapta rezultă ușor din corectitudinea sistemului de inferență *backchain*.
- Implicația de la dreapta la stânga este mai complicată.
  - Q apare în interpretările simbolurilor de predicate din $\mathcal{LH}$
  - Deci Q este obținut după un număr finit n de aplicări ale lui f_T
  - Se arată prin inducție după n că pentru fiecare formulă care apare prin aplicări ale lui f_T există o derivare în sistemul de inferență *backchain*.

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 T o mulțime de clauze definite.

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

unde

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din T (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 c.g.u 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)
```

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

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din T
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este c.g.u pentru Q_i și Q.

Rezoluția SLD

Exemplu

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

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

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

$$\frac{\neg stark(jonSnow)}{}$$

$$\theta(X) = jonSnow$$

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

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din T
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este c.g.u pentru Q_i și Q.

Rezoluția SLD

Exemplu

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

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

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

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

$$\theta(X) = jonSnow$$

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

- $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ este o clauză definită din T
- variabilele din $Q \vee \neg P_1 \vee \cdots \vee \neg P_m$ și Q_i se redenumesc
- θ este c.g.u pentru Q_i și Q.

Rezoluția SLD

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

- O derivare din T 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 T,*
- $T \vdash_b Q_1 \wedge \ldots \wedge Q_m$,
- $T \models Q_1 \wedge \cdots \wedge Q_m$.

Rezoluția SLD

Teoremă (Completitudinea SLD-rezoluției)

Sunt echivalente:

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

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 T

ddacă

$$T \vdash_b Q_1 \wedge \ldots \wedge Q_m$$

$\square$

Rezoluția SLD - arbori de căutare

Arbori SLD

- Presupunem că avem o mulțime de clauze definite T 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 $C_i \in T$, 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 T.

Rezoluția SLD

Exemplu

- Fie T 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 T pentru

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

Rezoluția SLD

Exemplu

- Fie T 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 T 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. $mother(diana, brian)$

$$
\begin{tikzcd}
 & \neg grandfather(ken, Y) \arrow[d, "1"] & \\
 & \neg father(ken, V) \vee \neg parent(V, Y) \arrow[d, "4"] & \\
 & \neg parent(diana, Y) \arrow[dl, "2"'] \arrow[dr, "3"] & \\
\neg father(diana, Y) & & \neg mother(diana, Y) \arrow[d, "5"] \\
 & & \square
\end{tikzcd}
$$

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. $mother(diana, brian)$

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

$$\downarrow 1$$

$$\neg father(ken, V) \vee \neg parent(V, Y)$$

$$\downarrow 4$$

$$\neg parent(diana, Y)$$

$$\swarrow 2 \qquad\qquad \searrow 3$$

$$\neg father(diana, Y) \qquad\qquad \neg mother(diana, Y)$$

$$\downarrow 5$$

$$\square$$

Ce lipsește?

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
```

Exemplu (cont.)

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

$$\begin{array}{rcl} albedoDecrease & \to & warmerClimate \\ carbonIncrease & \to & warmerClimate \\ warmerClimate & \to & iceMelts \\ iceMelts & \to & albedoDecrease \\ \top & \to & carbonIncrease \end{array}$$

$$\frac{\dfrac{carbonInc. \qquad carbonInc. \to warmerClim.}{warmerClim.} \qquad warmerClim. \to iceMelts}{iceMelts}$$

Pe săptămâna viitoare!