

Curs 3

- 1 Clauze propoziționale definite
- 2 Puncte fixe. Teorema Knaster-Tarski
- 3 Completitudinea sistemului de deducție CDP
- 4 Rezoluție SLD

Bibliografie:

- J.W. Lloyd, Foundations of Logic Programming, Second Edition, Springer, 1987
- R.J. Brachman, H.J. Levesque, Knowledge Representation and Reasoning, Morgan Kaufmann Publishers, San Francisco, CA, 2004
- Logic Programming, The University of Edinburgh,
<https://www.inf.ed.ac.uk/teaching/courses/lp/>

- În principiu, sistemul poate verifica problema consecinței logice construind un **tabel de adevăr**, cu câte o linie pentru fiecare interpretare posibilă.
- În cazul în care programul și ținta conțin n atomi diferiți, tabelul de adevăr rezultat o să aibă 2^n rânduri.
- Această metodă este atât de costisitoare computațional (**timp exponențial**).

Cum salvăm situația?

- În principiu, sistemul poate verifica problema consecinței logice construind un **tabel de adevăr**, cu câte o linie pentru fiecare interpretare posibilă.
- În cazul în care programul și ținta conțin n atomi diferiți, tabelul de adevăr rezultat o să aibă 2^n rânduri.
- Această metodă este atât de costisitoare computațional (**timp exponențial**).

Cum salvăm situația?

- 1 Folosirea **metodelor sintactice** pentru a stabili problema consecinței logice (*proof search*)
- 2 **Restricționarea formulelor** din "programele logice"

Clauze propoziționale definite

Clauze propoziționale definite

- O **clauză propozițională definită** este o formulă care poate avea una din formele:

1 q

2 $p_1 \wedge \dots \wedge p_k \rightarrow q$

unde q, p_1, \dots, p_n sunt variabile propoziționale

Clauze propoziționale definite

- O **clauză propozițională definită** este o formulă care poate avea una din formele:

1 q (un **fapt** în Prolog $q.$)

2 $p_1 \wedge \dots \wedge p_k \rightarrow q$ (o **regulă** în Prolog $q \text{ :- } p_1, \dots, p_k$)

unde q, p_1, \dots, p_n sunt variabile propoziționale

Clauze propoziționale definite

- O clauză propozițională definită este o formulă care poate avea una din formele:
 - 1 q (un fapt în Prolog $q.$)
 - 2 $p_1 \wedge \dots \wedge p_k \rightarrow q$ (o regulă în Prolog $q \text{ :- } p_1, \dots, p_k$)unde q, p_1, \dots, p_n sunt variabile propoziționale
- Numim variabilele propoziționale atomi.

Clauze propoziționale definite

- O **clauză propozițională definită** este o formulă care poate avea una din formele:

1 q (un **fapt** în Prolog $q.$)

2 $p_1 \wedge \dots \wedge p_k \rightarrow q$ (o **regulă** în Prolog $q \text{ :- } p_1, \dots, p_k$)

unde q, p_1, \dots, p_n sunt variabile propoziționale

- Numim variabilele propoziționale **atomi**.

Programare logică – cazul logicii propoziționale

- Un "**program logic**" este o listă Cd_1, \dots, Cd_n de clauze definite.

Clauze propoziționale definite

- O **clauză propozițională definită** este o formulă care poate avea una din formele:

1 q (un **fapt** în Prolog $q.$)

2 $p_1 \wedge \dots \wedge p_k \rightarrow q$ (o **regulă** în Prolog $q \text{ :- } p_1, \dots, p_k$)

unde q, p_1, \dots, p_n sunt variabile propoziționale

- Numim variabilele propoziționale **atomi**.

Programare logică – cazul logicii propoziționale

- Un "**program logic**" este o listă Cd_1, \dots, Cd_n de clauze definite.
- O întrebare este o listă q_1, \dots, q_m de atomi.

Clauze propoziționale definite

- O **clauză propozițională definită** este o formulă care poate avea una din formele:
 - 1 q (un **fapt** în Prolog $q.$)
 - 2 $p_1 \wedge \dots \wedge p_k \rightarrow q$ (o **regulă** în Prolog $q \text{ :- } p_1, \dots, p_k$)unde q, p_1, \dots, p_n sunt variabile propoziționale
- Numim variabilele propoziționale **atomi**.

Programare logică – cazul logicii propoziționale

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

$$Cd_1, \dots, Cd_n \models q_1 \wedge \dots \wedge q_m.$$

Clauze propoziționale definite

- O **clauză propozițională definită** este o formulă care poate avea una din formele:
 - 1 q (un **fapt** în Prolog $q.$)
 - 2 $p_1 \wedge \dots \wedge p_k \rightarrow q$ (o **regulă** în Prolog $q \text{ :- } p_1, \dots, p_k$)unde q, p_1, \dots, p_n sunt variabile propoziționale
- Numim variabilele propoziționale **atomi**.

Programare logică – cazul logicii propoziționale

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

$$Cd_1, \dots, Cd_n \models q_1 \wedge \dots \wedge q_m.$$

Vom studia metode sintactice pentru a rezolva această problemă!

Sistem de deducție CDP

Sistem de deducție CDP pentru clauze definite propoziționale

Pentru o mulțime \mathcal{S} de clauze definite propoziționale, avem

Sistem de deducție CDP

Sistem de deducție CDP pentru clauze definite propoziționale

Pentru o mulțime \mathcal{S} de clauze definite propoziționale, avem

- **Axiome** (premise): orice clauză din \mathcal{S}

Sistem de deducție CDP

Sistem de deducție CDP pentru clauze definite propoziționale

Pentru o mulțime \mathcal{S} de clauze definite propoziționale, avem

□ **Axiome** (premise): orice clauză din \mathcal{S}

□ **Reguli de deducție:**

$$\frac{P \quad P \rightarrow Q}{Q} (MP) \qquad \frac{P \quad Q}{P \wedge Q} (andI)$$

- Aceste reguli ne permit să deducem formula de sub linie din formulele de deasupra liniei.
- Sunt regulile $(\rightarrow e)$ și $(\wedge i)$ din deducția naturală pentru logica propozițională.

Sistemul de deducție CDP

$$\frac{P \quad P \rightarrow Q}{Q} \text{ (MP)}$$

$$\frac{P \quad Q}{P \wedge Q} \text{ (andI)}$$

Exemplu

```
      oslo → windy
      oslo → norway
      norway → cold
cold ∧ windy → winterIsComing
              oslo
```


Sistemul de deducție CDP

$$\frac{P \quad P \rightarrow Q}{Q} \text{ (MP)}$$

$$\frac{P \quad Q}{P \wedge Q} \text{ (andI)}$$

Exemplu

	oslo	→	windy
	oslo	→	norway
	norway	→	cold
	cold ∧ windy	→	winterIsComing
	oslo		
<u>oslo</u>	<u>oslo → norway</u>		
	norway		
		<u>norway → cold</u>	
			<u>oslo oslo → windy</u>
			windy
			<u>cold ∧ windy</u>

Sistemul de deducție CDP

$$\frac{P \quad P \rightarrow Q}{Q} \text{ (MP)}$$

$$\frac{P \quad Q}{P \wedge Q} \text{ (andI)}$$

Exemplu

		oslo	→	windy					
		oslo	→	norway					
		norway	→	cold					
		cold ∧ windy	→	winterIsComing					
		oslo							
<u>oslo</u>	<u>oslo → norway</u>			<u>norway → cold</u>	<u>oslo</u>	<u>oslo → windy</u>			
		norway							
		cold		windy					
		cold ∧ windy							
		cold ∧ windy → winterIsComing							
		winterIsComing							

Sistemul de deducție CDP

$$\frac{P \quad P \rightarrow Q}{Q} \text{ (MP)}$$

$$\frac{P \quad Q}{P \wedge Q} \text{ (andI)}$$

Exemplu

oslo \rightarrow windy
oslo \rightarrow norway
norway \rightarrow cold
cold \wedge windy \rightarrow winterIsComing
oslo

- | | | |
|---|------------------------|------------|
| 1. oslo \rightarrow windy | 6. norway | (MP 5,2) |
| 2. oslo \rightarrow norway | 7. cold | (MP 6,3) |
| 3. norway \rightarrow cold | 8. windy | (MP 5,1) |
| 4. cold \wedge windy \rightarrow winterIsComing | 9. cold \wedge windy | (andI 7,8) |
| 5. oslo | 10. winterIsComing | (MP 9,4) |

Sistemul de deducție CDP

Putem folosi sistemul de deducție CDP pentru a deduce alți atomi:

Sistemul de deducție CDP

Putem folosi sistemul de deducție CDP pentru a deduce alți atomi:

- Atomii $p_i \in \mathcal{S}$ care sunt fapte sunt deductibili.
 - Sunt deduși ca axiome.

Sistemul de deducție CDP

Putem folosi sistemul de deducție CDP pentru a deduce alți atomi:

- Atomii $p_i \in \mathcal{S}$ care sunt fapte sunt deductibili.
 - Sunt deduși ca axiome.
- Un atom r este deductibil dacă
 - p_1, \dots, p_n sunt deductibili, și
 - $p_1 \wedge \dots \wedge p_n \rightarrow r$ este în \mathcal{S} .

O astfel de derivare folosește de $n - 1$ ori (*andI*) și o dată (*MP*).

Sistemul de deducție CDP

Putem folosi sistemul de deducție CDP pentru a deduce alți atomi:

- Atomii $p_i \in \mathcal{S}$ care sunt fapte sunt deductibili.
 - Sunt deduși ca axiome.
- Un atom r este deductibil dacă
 - p_1, \dots, p_n sunt deductibili, și
 - $p_1 \wedge \dots \wedge p_n \rightarrow r$ este în \mathcal{S} .

O astfel de derivare folosește de $n - 1$ ori (*andI*) și o dată (*MP*).

Deci putem construi mulțimi din ce în ce mai mari de atomi care sunt consecințe logice din \mathcal{S} , și pentru care există derivări din \mathcal{S} .

Sistemul de deducție CDP

Putem folosi sistemul de deducție CDP pentru a deduce alți atomi:

- Atomii $p_i \in \mathcal{S}$ care sunt fapte sunt deductibili.
 - Sunt deduși ca axiome.
- Un atom r este deductibil dacă
 - p_1, \dots, p_n sunt deductibili, și
 - $p_1 \wedge \dots \wedge p_n \rightarrow r$ este în \mathcal{S} .

O astfel de derivare folosește de $n - 1$ ori (*andI*) și o dată (*MP*).

Deci putem construi mulțimi din ce în ce mai mari de atomi care sunt consecințe logice din \mathcal{S} , și pentru care există derivări din \mathcal{S} .

Observăm că (*andI*) și (*MP*) pot fi înlocuite cu următoarea regulă derivată:

$$\frac{P_1, \dots, P_n \quad P_1 \wedge \dots \wedge P_n \rightarrow Q}{Q} \text{ (GMP)}$$

Sistemul de deducție CDP

Putem folosi sistemul de deducție CDP pentru a deduce alți atomi:

- Atomii $p_i \in \mathcal{S}$ care sunt fapte sunt deductibili.
 - Sunt deduși ca axiome.
- Un atom r este deductibil dacă
 - p_1, \dots, p_n sunt deductibili, și
 - $p_1 \wedge \dots \wedge p_n \rightarrow r$ este în \mathcal{S} .

O astfel de derivare folosește de $n - 1$ ori (*andI*) și o dată (*MP*).

Deci putem construi mulțimi din ce în ce mai mari de atomi care sunt consecințe logice din \mathcal{S} , și pentru care există derivări din \mathcal{S} .

Observăm că (*andI*) și (*MP*) pot fi înlocuite cu următoarea regulă derivată:

$$\frac{P_1, \dots, P_n \quad P_1 \wedge \dots \wedge P_n \rightarrow Q}{Q} \text{ (GMP)}$$

Sistemul de deducție CDP

O formulă Q se poate deduce din S în sistemul de deducție CDP, notat

$$S \vdash Q,$$

dacă există o secvență de formule Q_1, \dots, Q_n astfel încât $Q_n = Q$ și fiecare Q_i :

- fie aparține lui S
- fie se poate deduce din Q_1, \dots, Q_{i-1} folosind regulile de deducție (MP) și ($andI$)

Completitudinea sistemului de deducție CDP

- Se poate demonstra că aceste reguli sunt corecte, folosind tabelele de adevăr.
 - Dacă formulele de deasupra liniei sunt adevărate, atunci și formula de sub linie este adevărată.

Completitudinea sistemului de deducție CDP

- Se poate demonstra că aceste reguli sunt corecte, folosind tabelele de adevăr.
 - Dacă formulele de deasupra liniei sunt adevărate, atunci și formula de sub linie este adevărată.
- Mai mult, sistemul de deducție este și complet, adică
dacă $\mathcal{S} \models q$, atunci $\mathcal{S} \vdash q$.
 - Dacă q este o consecință logică a lui \mathcal{S} , atunci există o derivare a sa din \mathcal{S} folosind sistemul de deducție CDP

Completitudinea sistemului de deducție CDP

- Se poate demonstra că aceste reguli sunt corecte, folosind tabelele de adevăr.
 - Dacă formulele de deasupra liniei sunt adevărate, atunci și formula de sub linie este adevărată.
- Mai mult, sistemul de deducție este și complet, adică
dacă $\mathcal{S} \models q$, atunci $\mathcal{S} \vdash q$.
 - Dacă q este o consecință logică a lui \mathcal{S} , atunci există o derivare a sa din \mathcal{S} folosind sistemul de deducție CDP
- Pentru a demonstra completitudinea vom folosi teorema Knaster-Tarski.

Puncte fixe. Teorema Knaster-Tarski

Mulțimi parțial ordonate

- O mulțime parțial ordonată (mpo) este o pereche (M, \leq) unde $\leq \subseteq M \times M$ este o relație de ordine.
 - relație de ordine: reflexivă, antisimetrică, tranzitivă

Mulțimi parțial ordonate

- O mulțime parțial ordonată (mpo) este o pereche (M, \leq) unde $\leq \subseteq M \times M$ este o relație de ordine.
 - relație de ordine: reflexivă, antisimetrică, tranzitivă
- O mpo (L, \leq) se numește lanț dacă este total ordonată, adică $x \leq y$ sau $y \leq x$ pentru orice $x, y \in L$. Vom considera lanțuri numărabile:

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

Mulțimi parțial ordonate complete

O mpo (C, \leq) este **completă** (cpo) dacă:

- C are prim element \perp ($\perp \leq x$ oricare $x \in C$),
- $\bigvee_n x_n$ există în C pentru orice lanț $x_1 \leq x_2 \leq x_3 \leq \dots$

Mulțimi parțial ordonate complete

O mpo (C, \leq) este **completă** (cpo) dacă:

- C are prim element \perp ($\perp \leq x$ oricare $x \in C$),
- $\bigvee_n x_n$ există în C pentru orice lanț $x_1 \leq x_2 \leq x_3 \leq \dots$

Exemplu

Fie X o mulțime și $\mathcal{P}(X)$ mulțimea submulțimilor lui X .

$(\mathcal{P}(X), \subseteq)$ este o cpo:

- \subseteq este o relație de ordine
- \emptyset este prim element ($\emptyset \subseteq Q$ pentru orice $Q \in \mathcal{P}(X)$)
- pentru orice șir (numărabil) de submulțimi ale lui X $Q_1 \subseteq Q_2 \subseteq \dots$ evident $\bigcup_n Q_n \in \mathcal{P}(X)$

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

□ $f_1(Y) = Y \cup \{1\}$

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

□ $f_1(Y) = Y \cup \{1\}$ este monotonă.

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

□ $f_1(Y) = Y \cup \{1\}$ este monotonă.

□ $f_2(Y) = \begin{cases} \{1\} & \text{dacă } 1 \in Y \\ \emptyset & \text{altfel} \end{cases}$

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

□ $f_1(Y) = Y \cup \{1\}$ este monotonă.

□ $f_2(Y) = \begin{cases} \{1\} & \text{dacă } 1 \in Y \\ \emptyset & \text{altfel} \end{cases}$ este monotonă.

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

□ $f_1(Y) = Y \cup \{1\}$ este monotonă.

□ $f_2(Y) = \begin{cases} \{1\} & \text{dacă } 1 \in Y \\ \emptyset & \text{altfel} \end{cases}$ este monotonă.

□ $f_3(Y) = \begin{cases} \emptyset & \text{dacă } 1 \in Y \\ \{1\} & \text{altfel} \end{cases}$

Funcție monotonă

□ Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate.

O funcție $f : A \rightarrow B$ este **monotonă** (**crescătoare**)

dacă $a_1 \leq_A a_2$ implică $f(a_1) \leq_B f(a_2)$ oricare $a_1, a_2 \in A$.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

□ $f_1(Y) = Y \cup \{1\}$ este monotonă.

□ $f_2(Y) = \begin{cases} \{1\} & \text{dacă } 1 \in Y \\ \emptyset & \text{altfel} \end{cases}$ este monotonă.

□ $f_3(Y) = \begin{cases} \emptyset & \text{dacă } 1 \in Y \\ \{1\} & \text{altfel} \end{cases}$ nu este monotonă.

Funcție continuă

- Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate complete.
O funcție $f : A \rightarrow B$ este **continuă** dacă
$$f(\bigvee_n a_n) = \bigvee_n f(a_n)$$
 pentru orice lanț $\{a_n\}_n$ din A .

Funcție continuă

- Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate complete.
O funcție $f : A \rightarrow B$ este **continuă** dacă
$$f(\bigvee_n a_n) = \bigvee_n f(a_n)$$
 pentru orice lanț $\{a_n\}_n$ din A .
- Observăm că **orice funcție continuă este crescătoare**.

Funcție continuă

- Fie (A, \leq_A) și (B, \leq_B) mulțimi parțial ordonate complete.

O funcție $f : A \rightarrow B$ este **continuă** dacă

$$f(\bigvee_n a_n) = \bigvee_n f(a_n) \text{ pentru orice lanț } \{a_n\}_n \text{ din } A.$$

- Observăm că **orice funcție continuă este crescătoare**.

Exemplu

Fie următoarele funcții $f_i : \mathcal{P}(\{1, 2, 3\}) \rightarrow \mathcal{P}(\{1, 2, 3\})$ cu $i \in \{1, 2, 3\}$

- $f_1(Y) = Y \cup \{1\}$ este continuă.

- $f_3(Y) = \begin{cases} \emptyset & \text{dacă } 1 \in Y \\ \{1\} & \text{altfel} \end{cases}$ nu este continuă.

Teorema de punct fix

- Un element $a \in C$ este punct fix al unei funcții $f : C \rightarrow C$ dacă $f(a) = a$.

Teorema de punct fix

- Un element $a \in C$ este **punct fix** al unei funcții $f : C \rightarrow C$ dacă $f(a) = a$.

Teorema Knaster-Tarski pentru CPO

Fie (C, \leq) o mulțime parțial ordonată completă și $\mathbf{F} : C \rightarrow C$ o funcție continuă. Atunci

$$a = \bigvee_n \mathbf{F}^n(\perp)$$

este cel mai mic punct fix al funcției \mathbf{F} .

Teorema de punct fix

- Un element $a \in C$ este **punct fix** al unei funcții $f : C \rightarrow C$ dacă $f(a) = a$.

Teorema Knaster-Tarski pentru CPO

Fie (C, \leq) o mulțime parțial ordonată completă și $\mathbf{F} : C \rightarrow C$ o funcție continuă. Atunci

$$a = \bigvee_n \mathbf{F}^n(\perp)$$

este cel mai mic punct fix al funcției \mathbf{F} .

- Observăm că în ipotezele ultimei teoreme secvența

$$\mathbf{F}^0(\perp) = \perp \leq \mathbf{F}(\perp) \leq \mathbf{F}^2(\perp) \leq \dots \leq \mathbf{F}^n(\perp) \leq \dots$$

este un lanț, deci $\bigvee_n \mathbf{F}^n(\perp)$ există.

Teorema Knaster-Tarski pentru CPO

Demonstrație

Fie (C, \leq) o mulțime parțial ordonată completă și $F : C \rightarrow C$ o funcție continuă.

□ Arătăm că $a = \bigvee_n F^n(\perp)$ este punct fix, i.e. $F(a) = a$

Teorema Knaster-Tarski pentru CPO

Demonstrație

Fie (C, \leq) o mulțime parțial ordonată completă și $F : C \rightarrow C$ o funcție continuă.

□ Arătăm că $a = \bigvee_n F^n(\perp)$ este punct fix, i.e. $F(a) = a$

$$F(a) = F(\bigvee_n F^n(\perp))$$

Teorema Knaster-Tarski pentru CPO

Demonstrație

Fie (C, \leq) o mulțime parțial ordonată completă și $F : C \rightarrow C$ o funcție continuă.

□ Arătăm că $a = \bigvee_n F^n(\perp)$ este punct fix, i.e. $F(a) = a$

$$\begin{aligned} F(a) &= F(\bigvee_n F^n(\perp)) \\ &= \bigvee_n F(F^n(\perp)) \text{ din continuitate} \end{aligned}$$

Teorema Knaster-Tarski pentru CPO

Demonstrație

Fie (C, \leq) o mulțime parțial ordonată completă și $F : C \rightarrow C$ o funcție continuă.

□ Arătăm că $a = \bigvee_n F^n(\perp)$ este punct fix, i.e. $F(a) = a$

$$\begin{aligned} F(a) &= F(\bigvee_n F^n(\perp)) \\ &= \bigvee_n F(F^n(\perp)) \text{ din continuitate} \\ &= \bigvee_n F^{n+1}(\perp) \end{aligned}$$

Teorema Knaster-Tarski pentru CPO

Demonstrație

Fie (C, \leq) o mulțime parțial ordonată completă și $F : C \rightarrow C$ o funcție continuă.

□ Arătăm că $a = \bigvee_n F^n(\perp)$ este punct fix, i.e. $F(a) = a$

$$\begin{aligned} F(a) &= F(\bigvee_n F^n(\perp)) \\ &= \bigvee_n F(F^n(\perp)) \text{ din continuitate} \\ &= \bigvee_n F^{n+1}(\perp) \\ &= \bigvee_n F^n(\perp) = a \end{aligned}$$

Teorema Knaster-Tarski pentru CPO

Demonstrație (cont.)

- Arătăm că a este cel mai mic punct fix.

Teorema Knaster-Tarski pentru CPO

Demonstrație (cont.)

□ Arătăm că a este cel mai mic punct fix.

Fie b un alt punct fix, i.e. $\mathbf{F}(b) = b$.

Demonstrăm prin inducție după $n \geq 1$ că $\mathbf{F}^n(\perp) \leq b$.

Teorema Knaster-Tarski pentru CPO

Demonstrație (cont.)

□ Arătăm că a este cel mai mic punct fix.

Fie b un alt punct fix, i.e. $\mathbf{F}(b) = b$.

Demonstrăm prin inducție după $n \geq 1$ că $\mathbf{F}^n(\perp) \leq b$.

Pentru $n = 0$, $\mathbf{F}^0(\perp) = \perp \leq b$ deoarece \perp este prim element.

Teorema Knaster-Tarski pentru CPO

Demonstrație (cont.)

□ Arătăm că a este cel mai mic punct fix.

Fie b un alt punct fix, i.e. $\mathbf{F}(b) = b$.

Demonstrăm prin inducție după $n \geq 1$ că $\mathbf{F}^n(\perp) \leq b$.

Pentru $n = 0$, $\mathbf{F}^0(\perp) = \perp \leq b$ deoarece \perp este prim element.

Dacă $\mathbf{F}^n(\perp) \leq b$, atunci $\mathbf{F}^{n+1}(\perp) \leq \mathbf{F}(b)$, deoarece \mathbf{F} este crescătoare.
Deoarece $\mathbf{F}(b) = b$ rezultă $\mathbf{F}^{n+1}(\perp) \leq b$.

Teorema Knaster-Tarski pentru CPO

Demonstrație (cont.)

□ Arătăm că a este cel mai mic punct fix.

Fie b un alt punct fix, i.e. $\mathbf{F}(b) = b$.

Demonstrăm prin inducție după $n \geq 1$ că $\mathbf{F}^n(\perp) \leq b$.

Pentru $n = 0$, $\mathbf{F}^0(\perp) = \perp \leq b$ deoarece \perp este prim element.

Dacă $\mathbf{F}^n(\perp) \leq b$, atunci $\mathbf{F}^{n+1}(\perp) \leq \mathbf{F}(b)$, deoarece \mathbf{F} este crescătoare.
Deoarece $\mathbf{F}(b) = b$ rezultă $\mathbf{F}^{n+1}(\perp) \leq b$.

Știm $\mathbf{F}^n(\perp) \leq b$ oricare $n \geq 1$, deci $a = \bigvee_n \mathbf{F}^n(\perp) \leq b$.

Teorema Knaster-Tarski pentru CPO

Demonstrație (cont.)

□ Arătăm că a este cel mai mic punct fix.

Fie b un alt punct fix, i.e. $\mathbf{F}(b) = b$.

Demonstrăm prin inducție după $n \geq 1$ că $\mathbf{F}^n(\perp) \leq b$.

Pentru $n = 0$, $\mathbf{F}^0(\perp) = \perp \leq b$ deoarece \perp este prim element.

Dacă $\mathbf{F}^n(\perp) \leq b$, atunci $\mathbf{F}^{n+1}(\perp) \leq \mathbf{F}(b)$, deoarece \mathbf{F} este crescătoare.
Deoarece $\mathbf{F}(b) = b$ rezultă $\mathbf{F}^{n+1}(\perp) \leq b$.

Știm $\mathbf{F}^n(\perp) \leq b$ oricare $n \geq 1$, deci $a = \bigvee_n \mathbf{F}^n(\perp) \leq b$.

Am arătat că a este cel mai mic punct fix al funcției \mathbf{F} . □

Completitudinea sistemului de deducție CDP

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Fie $Baza = \{p_i \mid p_i \in \mathcal{S}\}$ mulțimea faptelor din \mathcal{S} .

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Fie $Baza = \{p_i \mid p_i \in \mathcal{S}\}$ mulțimea faptelor din \mathcal{S} .

Exemplu

oslo	→	windy
oslo	→	norway
norway	→	cold
cold ∧ windy	→	winterIsComing
		oslo

$At = \{oslo, windy, norway, cold, winterIsComing\}$

$Baza = \{oslo\}$

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Fie $Baza = \{p_i \mid p_i \in \mathcal{S}\}$ mulțimea atomilor care apar în *faptele* din \mathcal{S} .

Definim funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ prin

$$\begin{aligned} f_{\mathcal{S}}(Y) = & Y \cup Baza \\ & \cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } \mathcal{S}, \\ & \quad s_1 \in Y, \dots, s_n \in Y\} \end{aligned}$$

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Fie $Baza = \{p_i \mid p_i \in \mathcal{S}\}$ mulțimea atomilor care apar în *faptele* din \mathcal{S} .

Definim funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ prin

$$\begin{aligned} f_{\mathcal{S}}(Y) = & Y \cup Baza \\ & \cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } \mathcal{S}, \\ & \quad s_1 \in Y, \dots, s_n \in Y\} \end{aligned}$$

Exercițiu. Arătați că funcția $f_{\mathcal{S}}$ este monotonă.

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \dots$ atunci $f_{\mathcal{S}}(\bigcup_k Y_k) = \bigcup_k f_{\mathcal{S}}(Y_k)$.

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \dots$ atunci $f_{\mathcal{S}}(\bigcup_k Y_k) = \bigcup_k f_{\mathcal{S}}(Y_k)$.

Din faptul că $f_{\mathcal{S}}$ este crescătoare rezultă $f_{\mathcal{S}}(\bigcup_k Y_k) \supseteq \bigcup_k f_{\mathcal{S}}(Y_k)$

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \dots$ atunci $f_{\mathcal{S}}(\bigcup_k Y_k) = \bigcup_k f_{\mathcal{S}}(Y_k)$.

Din faptul că $f_{\mathcal{S}}$ este crescătoare rezultă $f_{\mathcal{S}}(\bigcup_k Y_k) \supseteq \bigcup_k f_{\mathcal{S}}(Y_k)$

Demonstrăm în continuare că $f_{\mathcal{S}}(\bigcup_k Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$. Fie $a \in f_{\mathcal{S}}(\bigcup_n Y_k)$. Sunt posibile trei cazuri

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \dots$ atunci $f_{\mathcal{S}}(\bigcup_k Y_k) = \bigcup_k f_{\mathcal{S}}(Y_k)$.

Din faptul că $f_{\mathcal{S}}$ este crescătoare rezultă $f_{\mathcal{S}}(\bigcup_k Y_k) \supseteq \bigcup_k f_{\mathcal{S}}(Y_k)$

Demonstrăm în continuare că $f_{\mathcal{S}}(\bigcup_k Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$. Fie $a \in f_{\mathcal{S}}(\bigcup_n Y_k)$. Sunt posibile trei cazuri

□ $a \in \bigcup_k Y_k$

Există un $k \geq 1$ astfel încât $a \in Y_k$, deci $a \in f_{\mathcal{S}}(Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$.

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \dots$ atunci $f_{\mathcal{S}}(\bigcup_k Y_k) = \bigcup_k f_{\mathcal{S}}(Y_k)$.

Din faptul că $f_{\mathcal{S}}$ este crescătoare rezultă $f_{\mathcal{S}}(\bigcup_k Y_k) \supseteq \bigcup_k f_{\mathcal{S}}(Y_k)$

Demonstrăm în continuare că $f_{\mathcal{S}}(\bigcup_k Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$. Fie $a \in f_{\mathcal{S}}(\bigcup_n Y_k)$. Sunt posibile trei cazuri

□ $a \in \bigcup_k Y_k$

Există un $k \geq 1$ astfel încât $a \in Y_k$, deci $a \in f_{\mathcal{S}}(Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$.

□ $a \in Baza \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$

Clauze definite și funcții monotone

Fie At mulțimea atomilor p_1, p_2, \dots care apar în \mathcal{S} .

Propoziție

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este continuă.

Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \dots$ atunci $f_{\mathcal{S}}(\bigcup_k Y_k) = \bigcup_k f_{\mathcal{S}}(Y_k)$.

Din faptul că $f_{\mathcal{S}}$ este crescătoare rezultă $f_{\mathcal{S}}(\bigcup_k Y_k) \supseteq \bigcup_k f_{\mathcal{S}}(Y_k)$

Demonstrăm în continuare că $f_{\mathcal{S}}(\bigcup_k Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$. Fie $a \in f_{\mathcal{S}}(\bigcup_n Y_k)$. Sunt posibile trei cazuri

- $a \in \bigcup_k Y_k$
Există un $k \geq 1$ astfel încât $a \in Y_k$, deci $a \in f_{\mathcal{S}}(Y_k) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$.
- $a \in Baza \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$
- Există s_1, \dots, s_n în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \dots \wedge s_n \rightarrow a)$ este în \mathcal{S} .

Clauze definite și funcții monotone

Demonstrație (cont.)

- Există s_1, \dots, s_n în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \dots \wedge s_n \rightarrow a)$ este în \mathcal{S} .

Clauze definite și funcții monotone

Demonstrație (cont.)

- Există s_1, \dots, s_n în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \dots \wedge s_n \rightarrow a)$ este în \mathcal{S} .
Pentru fiecare $i \in \{1, \dots, n\}$ există $k_i \in \mathbb{N}$ astfel încât $s_i \in Y_{k_i}$.

Clauze definite și funcții monotone

Demonstrație (cont.)

□ Există s_1, \dots, s_n în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \dots \wedge s_n \rightarrow a)$ este în \mathcal{S} .

Pentru fiecare $i \in \{1, \dots, n\}$ există $k_i \in \mathbb{N}$ astfel încât $s_i \in Y_{k_i}$.

Dacă $k_0 = \max\{k_1, \dots, k_n\}$ atunci $Y_{k_i} \subseteq Y_{k_0}$ pentru orice $i \in \{1, \dots, n\}$.

Clauze definite și funcții monotone

Demonstrație (cont.)

- Există s_1, \dots, s_n în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \dots \wedge s_n \rightarrow a)$ este în \mathcal{S} .

Pentru fiecare $i \in \{1, \dots, n\}$ există $k_i \in \mathbb{N}$ astfel încât $s_i \in Y_{k_i}$.

Dacă $k_0 = \max\{k_1, \dots, k_n\}$ atunci $Y_{k_i} \subseteq Y_{k_0}$ pentru orice $i \in \{1, \dots, n\}$.

Rezultă că $s_1, \dots, s_n \in Y_{k_0}$, deci $a \in f_{\mathcal{S}}(Y_{k_0}) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$.

Clauze definite și funcții monotone

Demonstrație (cont.)

- Există s_1, \dots, s_n în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \dots \wedge s_n \rightarrow a)$ este în \mathcal{S} .

Pentru fiecare $i \in \{1, \dots, n\}$ există $k_i \in \mathbb{N}$ astfel încât $s_i \in Y_{k_i}$.

Dacă $k_0 = \max\{k_1, \dots, k_n\}$ atunci $Y_{k_i} \subseteq Y_{k_0}$ pentru orice $i \in \{1, \dots, n\}$.

Rezultă că $s_1, \dots, s_n \in Y_{k_0}$, deci $a \in f_{\mathcal{S}}(Y_{k_0}) \subseteq \bigcup_k f_{\mathcal{S}}(Y_k)$.

Am demonstrat că $f_{\mathcal{S}}$ este continuă.



Clauze definite și funcții monotone

Pentru funcția continuă $f_S : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$

$$f_S(Y) = Y \cup Baza \\ \cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

aplicând **Teorema Knaster-Tarski pentru CPO**, obținem că

$$\bigcup_n f_S^n(\emptyset)$$

este cel mai mic punct fix al lui f_S .

Clauze definite și funcții monotone

- Analizați ce se întâmplă când considerăm succesiv

$$\emptyset, \quad f_S(\emptyset), \quad f_S(f_S(\emptyset)), \quad f_S(f_S(f_S(\emptyset))), \dots$$

La fiecare aplicare a lui f_S , rezultatul fie se mărește, fie rămâne neschimbat.

Clauze definite și funcții monotone

- Analizați ce se întâmplă când considerăm succesiv

$$\emptyset, \quad f_S(\emptyset), \quad f_S(f_S(\emptyset)), \quad f_S(f_S(f_S(\emptyset))), \dots$$

La fiecare aplicare a lui f_S , rezultatul fie se mărește, fie rămâne neschimbat.

- Să presupunem că în S avem k atomi. Atunci după $k + 1$ aplicări ale lui f_S , trebuie să existe un punct în șirul de mulțimi obținute de unde o nouă aplicare a lui f_S nu mai schimbă rezultatul (punct fix):

$$f_S(X) = X$$

Clauze definite și funcții monotone

- Analizați ce se întâmplă când considerăm succesiv

$$\emptyset, \quad f_S(\emptyset), \quad f_S(f_S(\emptyset)), \quad f_S(f_S(f_S(\emptyset))), \dots$$

La fiecare aplicare a lui f_S , rezultatul fie se mărește, fie rămâne neschimbat.

- Să presupunem că în S avem k atomi. Atunci după $k + 1$ aplicări ale lui f_S , trebuie să existe un punct în șirul de mulțimi obținute de unde o nouă aplicare a lui f_S nu mai schimbă rezultatul (**punct fix**):

$$f_S(X) = X$$

- Dacă aplicăm f_S succesiv ca mai devreme până găsim un X cu proprietatea $f_S(X) = X$, atunci găsim **cel mai mic punct fix** al lui f_S .

Cel mai mic punct fix

Exemplu

$cold \rightarrow wet$
 $wet \wedge cold \rightarrow scotland$

$$f_S(Y) = Y \cup Baza$$

$$\cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

Se observă că $f_S(\emptyset) =$

Cel mai mic punct fix

Exemplu

$cold \rightarrow wet$
 $wet \wedge cold \rightarrow scotland$

$$f_S(Y) = Y \cup Baza$$

$\cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$

Se observă că $f_S(\emptyset) = \emptyset$, deci \emptyset este cel mai mic punct fix.

De aici deducem că niciun atom nu este consecință logică a formulelor de mai sus.

Exemplu

Exemplu

cold
cold → *wet*
windy → *dry*
wet ∧ *cold* → *scotland*

$$f_S(Y) = Y \cup \text{Baza}$$

$\cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S,$
 $s_1 \in Y, \dots, s_n \in Y\}$

Exemplu

Exemplu

cold
cold \rightarrow *wet*
windy \rightarrow *dry*
wet \wedge *cold* \rightarrow *scotland*

$$f_S(Y) = Y \cup Baza$$

$$\cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

$$f_S(\emptyset) = \{ cold \}$$

Exemplu

Exemplu

cold
cold → *wet*
windy → *dry*
wet ∧ *cold* → *scotland*

$$f_S(Y) = Y \cup \text{Baza}$$

$$\cup \{a \in \text{At} \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

$$f_S(\emptyset) = \{ \text{cold} \}$$

$$f_S(\{ \text{cold} \}) = \{ \text{cold}, \text{wet} \}$$

Exemplu

Exemplu

cold
cold → *wet*
windy → *dry*
wet ∧ *cold* → *scotland*

$$f_S(Y) = Y \cup \text{Baza}$$

$$\cup \{a \in \text{At} \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

$$f_S(\emptyset) = \{ \text{cold} \}$$

$$f_S(\{ \text{cold} \}) = \{ \text{cold}, \text{wet} \}$$

$$f_S(\{ \text{cold}, \text{wet} \}) = \{ \text{cold}, \text{wet}, \text{scotland} \}$$

Exemplu

Exemplu

cold
cold → *wet*
windy → *dry*
wet ∧ *cold* → *scotland*

$$f_S(Y) = Y \cup \text{Baza}$$

$$\cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

$$f_S(\emptyset) = \{ \text{cold} \}$$

$$f_S(\{ \text{cold} \}) = \{ \text{cold}, \text{wet} \}$$

$$f_S(\{ \text{cold}, \text{wet} \}) = \{ \text{cold}, \text{wet}, \text{scotland} \}$$

$$f_S(\{ \text{cold}, \text{wet}, \text{scotland} \}) = \{ \text{cold}, \text{wet}, \text{scotland} \}$$

Exemplu

Exemplu

cold
cold \rightarrow *wet*
windy \rightarrow *dry*
wet \wedge *cold* \rightarrow *scotland*

$$f_S(Y) = Y \cup \text{Baza}$$

$$\cup \{a \in \text{At} \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, \\ s_1 \in Y, \dots, s_n \in Y\}$$

$$f_S(\emptyset) = \{ \text{cold} \}$$

$$f_S(\{ \text{cold} \}) = \{ \text{cold}, \text{wet} \}$$

$$f_S(\{ \text{cold}, \text{wet} \}) = \{ \text{cold}, \text{wet}, \text{scotland} \}$$

$$f_S(\{ \text{cold}, \text{wet}, \text{scotland} \}) = \{ \text{cold}, \text{wet}, \text{scotland} \}$$

Deci cel mai mic punct fix este $\{ \text{cold}, \text{wet}, \text{scotland} \}$.

Programe logice și cel mai mic punct fix

Teoremă

Fie X este cel mai mic punct fix al funcției f_S . Atunci

$$q \in X \quad \text{dacă} \quad S \models q.$$

Intuiție: Cel mai mic punct fix al funcției f_S este mulțimea tuturor atomilor care sunt consecințe logice ale programului.

Funcția $f_S : \mathcal{P}(At) \rightarrow \mathcal{P}(At)$ este definită prin

$$f_S(Y) = Y \cup Baza$$

$$\cup \{a \in At \mid (s_1 \wedge \dots \wedge s_n \rightarrow a) \text{ este în } S, s_1 \in Y, \dots, s_n \in Y\}$$

unde At este mulțimea atomilor din S și $Baza = \{p_i \mid p_i \in S\}$ este mulțimea atomilor care apar în faptele din S .

Demonstrație

$(\Rightarrow) q \in X \Rightarrow \mathcal{S} \models q.$

- Funcția $f_{\mathcal{S}}$ conservă atomii adevărați.
- Deci, dacă fiecare clauză unitate din \mathcal{S} este adevărată, după fiecare aplicare a funcției $f_{\mathcal{S}}$ obținem o mulțime adevărată de atomi.

Programe logice și cel mai mic punct fix

Demonstrație

$(\Rightarrow) q \in X \Rightarrow \mathcal{S} \models q.$

- Funcția $f_{\mathcal{S}}$ conservă atomii adevărați.
- Deci, dacă fiecare clauză unitate din \mathcal{S} este adevărată, după fiecare aplicare a funcției $f_{\mathcal{S}}$ obținem o mulțime adevărată de atomi.

$(\Leftarrow) \mathcal{S} \models q \Rightarrow q \in X.$

- Fie $\mathcal{S} \models q$. Presupunem prin absurd că $q \notin X$.
- Căutăm o evaluare e care face fiecare clauză din \mathcal{S} adevărată, dar q falsă.

Programe logice și cel mai mic punct fix

Demonstrație (cont.)

□ Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

Programe logice și cel mai mic punct fix

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.
- Arătăm că $e^+(P) = 1$, pentru orice clauză $P \in \mathcal{S}$.

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.
- Arătăm că $e^+(P) = 1$, pentru orice clauză $P \in \mathcal{S}$.
- Fie $P \in \mathcal{S}$. Avem două cazuri:

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.
- Arătăm că $e^+(P) = 1$, pentru orice clauză $P \in \mathcal{S}$.
- Fie $P \in \mathcal{S}$. Avem două cazuri:

1 P este un fapt. Atunci $P \in X$, deci $e(P) = 1$.

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.
- Arătăm că $e^+(P) = 1$, pentru orice clauză $P \in \mathcal{S}$.
- Fie $P \in \mathcal{S}$. Avem două cazuri:

1 P este un fapt. Atunci $P \in X$, deci $e(P) = 1$.

2 P este de forma $p_1 \wedge \dots \wedge p_n \rightarrow r$. Atunci avem două cazuri:

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.
- Arătăm că $e^+(P) = 1$, pentru orice clauză $P \in \mathcal{S}$.
- Fie $P \in \mathcal{S}$. Avem două cazuri:

1 P este un fapt. Atunci $P \in X$, deci $e(P) = 1$.

2 P este de forma $p_1 \wedge \dots \wedge p_n \rightarrow r$. Atunci avem două cazuri:

- există un p_i , $i = 1, \dots, n$, care nu este în X . Deci $e^+(P) = 1$.

Demonstrație (cont.)

- Fie evaluarea

$$e(p) = \begin{cases} 1, & \text{dacă } p \in X \\ 0, & \text{altfel} \end{cases}$$

- Evident, această interpretare face q falsă.
- Arătăm că $e^+(P) = 1$, pentru orice clauză $P \in \mathcal{S}$.
- Fie $P \in \mathcal{S}$. Avem două cazuri:

1 P este un fapt. Atunci $P \in X$, deci $e(P) = 1$.

2 P este de forma $p_1 \wedge \dots \wedge p_n \rightarrow r$. Atunci avem două cazuri:

- există un p_i , $i = 1, \dots, n$, care nu este în X . Deci $e^+(P) = 1$.
- toți p_i , $i = 1, \dots, n$, sunt în X . Atunci $r \in f_{\mathcal{S}}(X) = X$, deci $e(r) = 1$.
În concluzie $e^+(P) = 1$.



Sistemul de deducție

Corolar

Sistemul de deducție pentru clauze definite propoziționale este complet pentru a arăta clauze unitate:

dacă $\mathcal{S} \models q$, atunci $\mathcal{S} \vdash q$.

Sistemul de deducție

Corolar

Sistemul de deducție pentru clauze definite propoziționale este complet pentru a arăta clauze unitate:

dacă $\mathcal{S} \models q$, atunci $\mathcal{S} \vdash q$.

Demonstrație

- Presupunem $\mathcal{S} \models q$.
- Atunci $q \in X$, unde X este cel mai mic punct fix al funcției $f_{\mathcal{S}}$.
- Fiecare aplicare a funcției $f_{\mathcal{S}}$ produce o mulțime demonstrabilă de atomi.
- Cum cel mai mic punct fix este atins după un număr finit de aplicări ale lui $f_{\mathcal{S}}$, orice $a \in X$ are o derivare.

□

Rezoluție SLD

Metodă de decizie

Avem o **metodă de decizie** (*decision procedure*) pentru a verifica $\mathcal{S} \vdash q$

Metoda constă în:

- calcularea celui mai mic punct fix X al funcției $f_{\mathcal{S}}$
- dacă $q \in X$ atunci returnăm **true**, altfel returnăm **false**

Metodă de decizie

Avem o **metodă de decizie** (*decision procedure*) pentru a verifica $\mathcal{S} \vdash q$

Metoda constă în:

- calcularea celui mai mic punct fix X al funcției $f_{\mathcal{S}}$
- dacă $q \in X$ atunci returnăm **true**, altfel returnăm **false**

Această metodă se termină.

Exercițiu. De ce?

Metodă de decizie

Avem o **metodă de decizie** (*decision procedure*) pentru a verifica $\mathcal{S} \vdash q$

Metoda constă în:

- calcularea celui mai mic punct fix X al funcției $f_{\mathcal{S}}$
- dacă $q \in X$ atunci returnăm **true**, altfel returnăm **false**

Această metodă se termină.

Exercițiu. De ce?

Program Prolog = baza de cunoștințe

- Un program Prolog reprezintă o bază de cunoștințe (knowledge base) KB. Cel mai mic punct fix al funcției f_{KB} definește totalitatea cunoștințelor care pot fi deduse din KB.

Metodă de decizie

Avem o **metodă de decizie** (*decision procedure*) pentru a verifica $\mathcal{S} \vdash q$

Metoda constă în:

- calcularea celui mai mic punct fix X al funcției f_S
- dacă $q \in X$ atunci returnăm **true**, altfel returnăm **false**

Această metodă se termină.

Exercițiu. De ce?

Program Prolog = baza de cunoștințe

- Un program Prolog reprezintă o bază de cunoștințe (knowledge base) KB. Cel mai mic punct fix al funcției f_{KB} definește totalitatea cunoștințelor care pot fi deduse din KB.
- Pentru o bază de cunoștințe formată numai din clauze propoziționale definite, cel mai mic punct fix poate fi calculat în timp liniar.

Calculul celui mai mic punct fix

KB:

$\{oslo\}$

$\{\neg oslo, windy\}$

$\{\neg oslo, norway\}$

$\{\neg norway, cold\}$

$\{\neg cold, \neg windy, winter\}$

LFP:

Calculul celui mai mic punct fix

LFP:

$\{\neg oslo, windy\}$

$\{\neg oslo, norway\}$

$\{\neg norway, cold\}$

$\{\neg cold, \neg windy, winter\}$

$\{oslo\}$

Calculul celui mai mic punct fix

LFP:

$\{\neg oslo, windy\}$

$\{\neg oslo, norway\}$

$\{\neg norway, cold\}$

$\{\neg cold, \neg windy, winter\}$

$\{oslo\}$

Calculul celui mai mic punct fix

$\{\neg \text{norway}, \text{cold}\}$

$\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$

LFP:

$\{\text{oslo}\}$

$\{\text{windy}\}$

$\{\text{norway}\}$

Calculul celui mai mic punct fix

$\{\neg \text{norway}, \text{cold}\}$

$\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$

LFP:

$\{\text{oslo}\}$

$\{\text{windy}\}$

$\{\text{norway}\}$

Calculul celui mai mic punct fix

$\{\neg cold, \neg windy, winter\}$

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

$\{cold\}$

Calculul celui mai mic punct fix

$\{\neg cold, \neg windy, winter\}$

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

$\{cold\}$

Calculul celui mai mic punct fix

$\{\neg \textit{windy}, \textit{winter}\}$

LFP:

$\{\textit{oslo}\}$

$\{\textit{windy}\}$

$\{\textit{norway}\}$

$\{\textit{cold}\}$

Calculul celui mai mic punct fix

$\{\neg \textit{windy}, \textit{winter}\}$

LFP:

$\{\textit{oslo}\}$

$\{\textit{windy}\}$

$\{\textit{norway}\}$

$\{\textit{cold}\}$

Calculul celui mai mic punct fix

LFP:

{*oslo*}

{*windy*}

{*norway*}

{*cold*}

{**winter**}

Propagarea unității

- În procedeul anterior am folosit o metodă asemănătoare rezoluției în care una din clauze are un singur literal.
- Clauzele formate dintr-un singur literal se numesc **clauze unitate** (*unit clause*), iar metoda anterioară se numește **propagarea unității** (*unit propagation*).
- Printr-o reprezentare adecvată a datelor, propagarea unității poate fi implementată în timp liniar în raport cu dimensiunea bazei de cunoștințe inițiale.
- **Clauzele Horn propoziționale** sunt clauze care au cel mult un literal pozitiv. Clauzele propoziționale definite sunt clauze Horn care au exact un literal pozitiv. Folosind metoda de propagare a unității problema satsfiabilității pentru clauze Horn propoziționale HORNSAT poate fi rezolvată în timp liniar.

Forward chaining / Backward chaining

- Metoda anterioară este centrată pe *lărgirea bazei de cunoștințe*.
- Pentru a afla răspunsul la o întrebare (-? winter) adăugăm pas cu pas cunoștințe noi, verificând de fiecare dată dacă am răspuns la întrebare.
- Acest procedeu se numește **forward chaining**.

Forward chaining / Backward chaining

- Metoda anterioară este centrată pe *lărgirea bazei de cunoștințe*.
- Pentru a afla răspunsul la o întrebare (-? winter) adăugăm pas cu pas cunoștințe noi, verificând de fiecare dată dacă am răspuns la întrebare.
- Acest procedeu se numește **forward chaining**.

Nu acesta este algoritmul folosit de Prolog!

Forward chaining / Backward chaining

- Metoda anterioară este centrată pe *lărgirea bazei de cunoștințe*.
- Pentru a afla răspunsul la o întrebare (-? winter) adăugăm pas cu pas cunoștințe noi, verificând de fiecare dată dacă am răspuns la întrebare.
- Acest procedeu se numește **forward chaining**.

Nu acesta este algoritmul folosit de Prolog!

- Metoda folosită de Prolog se numește **backward chaining**. Această metodă este centrată pe *găsirea răspunsului la întrebare*.

Backward chaining

- În *backward chaining* pornim de la întrebare (`-? winter`) și analizăm baza de cunoștințe, căutând o regulă care are drept concluzie scopul (`winter :- cold, windy`).
- În continuare vom încerca să satisfacem scopurile noi (`cold` și `windy`) prin același procedeu.
- Această metodă este realizată printr-o implementare particulară a rezoluției - rezoluția SLD.

Rezoluția SLD

Baza de cunoștințe KB:

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

Întrebarea:

```
-? winter.
```

Rezoluția SLD

Baza de cunoștințe KB:

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

Întrebarea:

-? winter.

□ Formă clauzală:

$$KB = \{\{oslo\}, \{\neg oslo, windy\}, \{\neg oslo, norway\}, \{\neg norway, cold\}, \{\neg cold, \neg windy, winter\}\}$$

□ $KB \vdash winter$ dacă și numai dacă $KB \cup \{\neg winter\}$ este satisfiabilă.

Rezoluția SLD

Baza de cunoștințe KB:

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

Întrebarea:

```
-? winter.
```

- Formă clauzală:

$$KB = \{\{oslo\}, \{\neg oslo, windy\}, \{\neg oslo, norway\}, \{\neg norway, cold\}, \{\neg cold, \neg windy, winter\}\}$$

- $KB \vdash winter$ dacă și numai dacă $KB \cup \{\neg winter\}$ este satisfiabilă.
- Satisfiabilitatea este verificată prin rezoluție

SLD = Linear resolution with Selected literal for Definite clauses

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \textit{winter}$ prin rezoluție SLD:

$\{\neg \textit{winter}\}$

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$ $\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$

$\{\neg \text{cold}, \neg \text{windy}\}$

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$ $\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$

$\{\neg \text{cold}, \neg \text{windy}\}$ $\{\neg \text{norway}, \text{cold}\}$

$\{\neg \text{norway}, \neg \text{windy}\}$

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$ $\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$

$\{\neg \text{cold}, \neg \text{windy}\}$ $\{\neg \text{norway}, \text{cold}\}$

$\{\neg \text{norway}, \neg \text{windy}\}$ $\{\neg \text{oslo}, \text{norway}\}$

$\{\neg \text{oslo}, \neg \text{windy}\}$

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$	$\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$
--------------------------	--

$\{\neg \text{cold}, \neg \text{windy}\}$	$\{\neg \text{norway}, \text{cold}\}$
---	---------------------------------------

$\{\neg \text{norway}, \neg \text{windy}\}$	$\{\neg \text{oslo}, \text{norway}\}$
---	---------------------------------------

$\{\neg \text{oslo}, \neg \text{windy}\}$	$\{\text{oslo}\}$
---	-------------------

$\{\neg \text{windy}\}$	
-------------------------	--

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$	$\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$
--------------------------	--

$\{\neg \text{cold}, \neg \text{windy}\}$	$\{\neg \text{norway}, \text{cold}\}$
---	---------------------------------------

$\{\neg \text{norway}, \neg \text{windy}\}$	$\{\neg \text{oslo}, \text{norway}\}$
---	---------------------------------------

$\{\neg \text{oslo}, \neg \text{windy}\}$	$\{\text{oslo}\}$
---	-------------------

$\{\neg \text{windy}\}$	$\{\neg \text{oslo}, \text{windy}\}$
-------------------------	--------------------------------------

$\{\neg \text{oslo}\}$	
------------------------	--

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$	$\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$
--------------------------	--

$\{\neg \text{cold}, \neg \text{windy}\}$	$\{\neg \text{norway}, \text{cold}\}$
---	---------------------------------------

$\{\neg \text{norway}, \neg \text{windy}\}$	$\{\neg \text{oslo}, \text{norway}\}$
---	---------------------------------------

$\{\neg \text{oslo}, \neg \text{windy}\}$	$\{\text{oslo}\}$
---	-------------------

$\{\neg \text{windy}\}$	$\{\neg \text{oslo}, \text{windy}\}$
-------------------------	--------------------------------------

$\{\neg \text{oslo}\}$	$\{\text{oslo}\}$
------------------------	-------------------

□

Clause Horn propoziționale - rezoluția SLD

Exemplu

Demonstrăm $KB \vdash \text{winter}$ prin rezoluție SLD:

$\{\neg \text{winter}\}$	$\{\neg \text{cold}, \neg \text{windy}, \text{winter}\}$
--------------------------	--

$\{\neg \text{cold}, \neg \text{windy}\}$	$\{\neg \text{norway}, \text{cold}\}$
---	---------------------------------------

$\{\neg \text{norway}, \neg \text{windy}\}$	$\{\neg \text{oslo}, \text{norway}\}$
---	---------------------------------------

$\{\neg \text{oslo}, \neg \text{windy}\}$	$\{\text{oslo}\}$
---	-------------------

$\{\neg \text{windy}\}$	$\{\neg \text{oslo}, \text{windy}\}$
-------------------------	--------------------------------------

$\{\neg \text{oslo}\}$	$\{\text{oslo}\}$
------------------------	-------------------

□

În cursurile următoare vom studia aceste mecanisme în logica de ordinul I.



Pe săptămâna viitoare!