# Curs 3

2019-2020

Programare Logică

# Cuprins



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/

# SAT

- 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. $q$ (un fapt în Prolog `q.`)
  2. $p_1 \wedge \ldots \wedge p_k \rightarrow q$ (o regulă în Prolog `q :- p1,...,pk`)

  unde $q, p_1, \ldots, p_n$ sunt variabile propoziționale
- Numim variabilele propoziționale atomi.

## Programare logică – cazul logicii propoziționale

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

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

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

- Axiome (premise): orice clauză din $\mathcal{S}$
- Reguli de deducție:

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

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

# Sistemul de deducție CDP

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

## Exemplu

$$\begin{array}{rcl} \texttt{oslo} & \to & \texttt{windy} \\ \texttt{oslo} & \to & \texttt{norway} \\ \texttt{norway} & \to & \texttt{cold} \\ \texttt{cold} \wedge \texttt{windy} & \to & \texttt{winterIsComing} \\ & & \texttt{oslo} \end{array}$$

$$\dfrac{\dfrac{\dfrac{oslo \quad oslo \to norway}{norway} \quad norway \to cold}{cold} \quad \dfrac{oslo \quad oslo \to windy}{windy}}{cold \wedge windy}$$

$$\frac{cold \wedge windy \quad cold \wedge windy \to winterIsComing}{winterIsComing}$$

# Sistemul de deducție CDP

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

## Exemplu

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

1. *oslo* $\to$ *windy*
2. *oslo* $\to$ *norway*
3. *norway* $\to$ *cold*
4. *cold* $\wedge$ *windy* $\to$ *winterIsComing*
5. *oslo*
6. *norway* (MP 5,2)
7. *cold* (MP 6,3)
8. *windy* (MP 5,1)
9. *cold* $\wedge$ *windy* (andI 7,8)
10. *winterIsComing* (MP 9,4)

# 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
  - $p_1, \ldots, p_n$ sunt deductibili, și
  - $p_1 \wedge \ldots \wedge p_n \rightarrow r$ este în $\mathcal{S}$.

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

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:

$$\frac{P_1, \ldots, P_n \quad P_1 \wedge \cdots \wedge P_n \rightarrow Q}{Q} \ (GMP)$$

# Sistemul de deducție CDP

O formulă $Q$ se poate deduce din $\mathcal{S}$ în sistemul de deducție CDP, notat

$$\mathcal{S} \vdash Q,$$

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

- fie aparține lui $S$
- fie se poate deduce din $Q_1, \ldots, Q_{i-1}$ 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.
- Mai mult, sistemul de deducție este și complet, adică

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

  - 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
- 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 $\bot$ ($\bot \leq x$ oricare $x \in C$),
- $\bigvee_n x_n$ există în $C$ pentru orice lanț $x_1 \leq x_2 \leq x_3 \leq \ldots$

## 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 $Q_1 \subseteq Q_2 \subseteq \ldots$ evident $\bigcup_n Q_n \in \mathcal{P}(X)$

# Funcție monotonă

- Fie $(A, \leq_A)$ și $(B, \leq_B)$ mulțimi parțial ordonate.
  O funcție $f : A \to B$ 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 $f_i : \mathcal{P}(\{1,2,3\}) \to \mathcal{P}(\{1,2,3\})$ 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 $f : A \to B$ 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 $f_i : \mathcal{P}(\{1,2,3\}) \to \mathcal{P}(\{1,2,3\})$ 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 $f : C \to C$ dacă $f(a) = a$.

## Teorema Knaster-Tarski pentru CPO

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

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

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

- Observăm că în ipotezele ultimei teoreme secvența

$$\mathbf{F}^0(\bot) = \bot \leq \mathbf{F}(\bot) \leq \mathbf{F}^2(\bot) \leq \cdots \leq \mathbf{F}^n(\bot) \leq \cdots$$

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

# Teorema Knaster-Tarski pentru CPO

## Demonstrație

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

- Arătăm că $a = \bigvee_n \mathbf{F}^n(\bot)$ este punct fix, i.e. $\mathbf{F}(a) = a$

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

# 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 $\mathbf{F}^n(\bot) \leq b$.

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

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

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

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 $p_1, p_2, \ldots$ care apar în $\mathcal{S}$.

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

## Exemplu

$$\begin{array}{rcl} \texttt{oslo} & \rightarrow & \texttt{windy} \\ \texttt{oslo} & \rightarrow & \texttt{norway} \\ \texttt{norway} & \rightarrow & \texttt{cold} \\ \texttt{cold} \wedge \texttt{windy} & \rightarrow & \texttt{winterIsComing} \\ & & \texttt{oslo} \end{array}$$

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

$Baza = \{oslo\}$

# Clauze definite și funcții monotone

Fie $At$ mulțimea atomilor $p_1, p_2, \ldots$ 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 $f_{\mathcal{S}} : \mathcal{P}(At) \to \mathcal{P}(At)$ prin

$$f_{\mathcal{S}}(Y) = Y \cup Baza \cup \{a \in At \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } \mathcal{S}, s_1 \in Y, \ldots, s_n \in Y\}$$

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

# Clauze definite și funcții monotone

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

## Propoziție

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

## Demonstrație

Arătăm că dacă $Y_1 \subseteq Y_2 \subseteq Y_3 \subseteq \ldots$ 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 $s_1, \ldots, s_n$ în $\bigcup_k Y_k$ astfel încât $(s_1 \wedge \ldots \wedge s_n \to a)$ este în $\mathcal{S}$.

# Clauze definite și funcții monotone

## Demonstrație (cont.)

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

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

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

  Rezultă că $s_1, \ldots, s_n \in Y_{k_0}$, 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 $f_{\mathcal{S}} : \mathcal{P}(At) \to \mathcal{P}(At)$

$$f_{\mathcal{S}}(Y) = Y \cup Baza \cup \{a \in At \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } \mathcal{S}, s_1 \in Y, \ldots, s_n \in Y\}$$

aplicând Teorema Knaster-Tarski pentru CPO, obținem că

$$\bigcup_n {f_{\mathcal{S}}}^n(\emptyset)$$

este cel mai mic punct fix al lui $f_{\mathcal{S}}$.

# Clauze definite și funcții monotone

- Analizați ce se întamplă când considerăm succesiv

$$\emptyset, \quad f_{\mathcal{S}}(\emptyset), \quad f_{\mathcal{S}}(f_{\mathcal{S}}(\emptyset)), \quad f_{\mathcal{S}}(f_{\mathcal{S}}(f_{\mathcal{S}}(\emptyset))), \ldots$$

  La fiecare aplicare a lui $f_{\mathcal{S}}$, rezultatul fie se mărește, fie rămâne neschimbat.

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

$$f_{\mathcal{S}}(X) = X$$

- Dacă aplicăm $f_{\mathcal{S}}$ succesiv ca mai devreme până găsim un $X$ cu proprietatea $f_{\mathcal{S}}(X) = X$, atunci găsim cel mai mic punct fix al lui $f_{\mathcal{S}}$.

# Cel mai mic punct fix

## Exemplu

$$\begin{array}{rcl} cold & \rightarrow & wet \\ wet \wedge cold & \rightarrow & scotland \end{array}$$

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

Se observă că $f_{\mathcal{S}}(\emptyset) = \emptyset$, 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

$$
\begin{array}{rcl}
cold & & \\
cold & \rightarrow & wet \\
windy & \rightarrow & dry \\
wet \wedge cold & \rightarrow & scotland
\end{array}
$$

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

$$
\begin{aligned}
f_S(\emptyset) &= \{ cold \} \\
f_S(\{ cold \}) &= \{ cold, wet \} \\
f_S(\{ cold, wet \}) &= \{ cold, wet, scotland \} \\
f_S(\{ cold, wet, scotland \}) &= \{ cold, wet, scotland \}
\end{aligned}
$$

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

# Programe logice și cel mai mic punct fix

**Teoremă**

*Fie $X$ este cel mai mic punct fix al funcției $f_{\mathcal{S}}$. Atunci*

$$q \in X \quad \textit{ddacă} \quad \mathcal{S} \models q.$$

Intuiție: Cel mai mic punct fix al funcției $f_{\mathcal{S}}$ este mulțimea tuturor atomilor care sunt consecințe logice ale programului.

Funcția $f_{\mathcal{S}} : \mathcal{P}(At) \to \mathcal{P}(At)$ este definită prin

$$f_{\mathcal{S}}(Y) = Y \cup Baza \cup \{a \in At \mid (s_1 \wedge \ldots \wedge s_n \to a) \text{ este în } \mathcal{S}, s_1 \in Y, \ldots, s_n \in Y\}$$

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

# 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.

## 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 $p_1 \wedge \ldots \wedge p_n \to r$. Atunci avem două cazuri:
   - există un $p_i$, $i = 1, \ldots, n$, care nu este în $X$. Deci $e^+(P) = 1$.
   - toți $p_i$, $i = 1, \ldots, n$, 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:

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

## 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**

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 cunoștintelor 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

$\{\neg oslo, windy\}$

$\{\neg oslo, norway\}$

$\{\neg norway, cold\}$

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

LFP:

$\{oslo\}$

# Calculul celui mai mic punct fix

$\{\neg oslo, windy\}$

$\{\neg oslo, norway\}$

$\{\neg norway, cold\}$

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

LFP:

$\{oslo\}$

# Calculul celui mai mic punct fix

$\{\neg norway, cold\}$

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

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

# Calculul celui mai mic punct fix

$\{\neg norway, cold\}$

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

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

# 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

$\{\neg windy, winter\}$

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

$\{cold\}$

# Calculul celui mai mic punct fix

$\{\neg windy, winter\}$

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

$\{cold\}$

# Calculul celui mai mic punct fix

LFP:

$\{oslo\}$

$\{windy\}$

$\{norway\}$

$\{cold\}$

$\{\mathbf{winter}\}$

# 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.

Nu acesta este algoritmul folosit de Prolog!

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

# Backward chaining

- În *backward chaining* pornim de la întrebare (`-? winter`) și analizăm baza de cunoștinte, 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 metod a 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.
```

- Formă clauzală:

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

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

| | |
|---|---|
| $\{\neg winter\}$ | $\{\neg cold, \neg windy, winter\}$ |
| $\{\neg cold, \neg windy\}$ | $\{\neg norway, cold\}$ |
| $\{\neg norway, \neg windy\}$ | $\{\neg oslo, norway\}$ |
| $\{\neg oslo, \neg windy\}$ | $\{oslo\}$ |
| $\{\neg windy\}$ | $\{\neg oslo, windy\}$ |
| $\{\neg oslo\}$ | $\{oslo\}$ |

$\square$

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

Pe săptămâna viitoare!