Programare Logică
Cursurile IX, X, XI, XII, XIII, XIV

Claudia MUREȘAN
cmuresan@fmi.unibuc.ro, c.muresan@yahoo.com

Universitatea din București
Facultatea de Matematică și Informatică
București

2019–2020, Semestrul II

Cuprinsul acestui curs

1. Să reluăm noțiunile din Logica Clasică a Predicatelor care stau la baza funcționării Prologului
2. Cu ce fel de signaturi și structuri algebrice de acele signaturi și cu ce tipuri/cazuri particulare de formule lucrează limbajul Prolog?
3. Modele pentru formule, satisfiabilitate în Logica Clasică a Predicatelor
4. Rezoluția în Logica Clasică a Predicatelor – regula de deducție care stă la baza funcționării limbajului Prolog
5. Rezoluția în Prolog

Structuri algebrice cu operații parțiale și relații

Definiție

Vom numi *structură de ordinul I* o structură algebrică de forma

$$\mathcal{A} = (A; (f_i)_{i \in I}; (R_j)_{j \in J}; (c_k)_{k \in K}),$$

unde:

- A este o mulțime nevidă, numită *universul structurii* $\mathcal{A}$ sau *mulțimea elementelor structurii algebrice* $\mathcal{A}$;
- I, J, K sunt mulțimi oarecare de indici (care pot fi și vide);
- pentru fiecare $i \in I$, există $n_i \in \mathbb{N}^*$, a. î. $f_i : A^{n_i} \dashrightarrow A$ (f_i este o operație parțială pe A cu n_i argumente, adică n_i–ară); pentru fiecare $i \in I$, notăm cu $dom(f_i)$ mulțimea n_i–uplurilor din A^{n_i} în care f_i este definită;
- pentru fiecare $j \in J$, există $m_j \in \mathbb{N}^*$, a. î. $R_j \subseteq A^{m_j}$ (R_j este o *relație* m_j–ară pe A);
- pentru fiecare $k \in K$, $c_k \in A$ (c_k este o constantă din A).

De obicei, operațiilor și relațiilor din componența lui $\mathcal{A}$ li se atașează indicele $\mathcal{A}$, pentru a le deosebi de simbolurile de operații și relații din limbajul pe care îl vom construi în continuare; astfel, structura de ordinul I de mai sus mai se notează sub forma: $\mathcal{A} = (A, (f_i^{\mathcal{A}})_{i \in I}, (R_j^{\mathcal{A}})_{j \in J}, (c_k^{\mathcal{A}})_{k \in K})$.

Vom avea clase de structuri de același **tip**, iar acestor **tipuri** de structuri algebrice le vom asocia **limbaje** ale logicii clasice a predicatelor

Definiție

Tipul sau *signatura* structurii de ordinul I $\mathcal{A}$ din definiția anterioară este tripletul de familii de numere naturale: $\tau = ((n_i)_{i\in I}; (m_j)_{j\in J}; (0)_{k\in K})$.
Orice structură de forma lui $\mathcal{A}$ de mai sus se numește *structură de ordinul I de tipul (sau signatura)* τ.

- Fiecărei signaturi τ a unei structuri de ordinul I (fiecărei clase de structuri de ordinul I de o anumită signatură τ) i se asociază un limbaj, numit **limbaj de ordinul I** și notat, de obicei, cu $\mathcal{L}_\tau$, în care pot fi exprimate proprietățile algebrice ale structurilor de ordinul I de signatura τ.
- Să considerăm o signatură $\tau = ((n_i)_{i\in I}; (m_j)_{j\in J}; (0)_{k\in K})$, pe care o fixăm.

Alfabetul limbajului de ordinul I $\mathcal{L}_\tau$ este format din următoarele **simboluri primitive**:

- o mulțime infinită de **variabile**: $Var = \{x, y, z, u, v, \ldots\}$;
- **simboluri de operații**: $(f_i)_{i\in I}$; pentru fiecare $i \in I$, numărul natural nenul n_i se numește *aritatea* lui f_i;

Limbajul asociat unei signaturi

- **simboluri de relații (simboluri de predicate)**: $(R_j)_{j \in J}$; pentru fiecare $j \in J$, numărul natural nenul m_j se numește *aritatea* lui R_j;
- **simboluri de constante**: $(c_k)_{k \in K}$;
- **simbolul de egalitate**: $=$ (un semn egal îngroșat) (*a nu se confunda cu egalul simplu!*);
- **conectorii logici primitivi**: $\neg$ (*negația*), $\rightarrow$ (*implicația*);
- **cuantificatorul universal**: $\forall$ (*oricare ar fi*)
- paranteze: $(,), [,]$, precum și virgula.

Pentru comoditate, vom spune uneori: "operații", "relații"/"predicate" și "constante" în loc de "simboluri de operații", " simboluri de relații/predicate" și " simboluri de constante", respectiv.

Observație

Majoritatea autorilor consideră virgula ca având semnificație implicită, subînțeleasă, în scrierea termenilor și a formulelor atomice, și nu includ virgula în limbajul $\mathcal{L}_\tau$.
Parantezele care încadrează formule (nu argumentele unei funcții sau relații) au același rol ca în sintaxa logicii propoziționale.

Termeni: variabile, constante, sau funcții cu argumente aplicate unor termeni, recursiv

Definiție

Termenii limbajului $\mathcal{L}_\tau$ sunt cuvintele finite (i.e. de lungime finită, adică formate dintr–un număr finit de litere) peste alfabetul de mai sus (i.e. ale căror litere sunt printre simbolurile de mai sus) definite, recursiv, astfel:

1. variabilele și simbolurile de constante sunt termeni;
2. dacă $n \in \mathbb{N}^*$, f este un simbol de operație n–ară și $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este un termen;
   f se numește *operația dominantă* sau *operatorul dominant* al termenului $f(t_1, \ldots, t_n)$.

Cum termenii sunt cuvinte de lungime finită peste alfabetul de mai sus, rezultă că orice termen se obține prin aplicarea regulilor ① și ② de un număr finit de ori.

Notație

Vom nota cu $Term(\mathcal{L}_\tau)$ mulțimea termenilor din limbajul $\mathcal{L}_\tau$.

Formule atomice: relații aplicate unor termeni

Definiție

Formulele atomice ale limbajului $\mathcal{L}_\tau$ sunt cuvintele finite peste alfabetul de mai sus definite astfel:

1. dacă $t_1, t_2 \in Term(\mathcal{L}_\tau)$, atunci $t_1 = t_2$ este o formulă atomică;
2. dacă R este un simbol de relație m–ară și $t_1, \ldots, t_m \in Term(\mathcal{L}_\tau)$, atunci $R(t_1, \ldots, t_m)$ este o formulă atomică.

Cum formulele atomice au lungimi finite, rezultă că orice formulă atomică se obține prin aplicarea regulilor ① și ② de un număr finit de ori

Formule: obținute din formulele atomice prin aplicarea conectorilor logici și a cuantificatorilor

Definiție

Formulele limbajului $\mathcal{L}_\tau$ sunt cuvintele finite peste alfabetul de mai sus definite, recursiv, astfel:

1. formulele atomice sunt formule;
2. dacă φ este o formulă, atunci $\neg\varphi$ este o formulă;
3. dacă φ și ψ sunt formule, atunci $\varphi \to \psi$ este o formulă;
4. dacă φ este o formulă și x este o variabilă, atunci $\forall x\varphi$ este o formulă.

Cum formulele au lungimi finite, rezultă că orice formulă se obține prin aplicarea regulilor ①, ②, ③ și ④ de un număr finit de ori.

Notație

Se notează cu $Form(\mathcal{L}_\tau)$ mulțimea formulelor limbajului $\mathcal{L}_\tau$.

Formule fără cuantificatori: obținute din formulele atomice prin aplicarea conectorilor logici

Putem defini astfel **formulele fără cuantificatori**:

Definiție

Formulele fără cuantificatori ale limbajului $\mathcal{L}_\tau$ sunt cuvintele finite peste alfabetul de mai sus definite, recursiv, astfel:

1. formulele atomice sunt formule fără cuantificatori;
2. dacă φ este o formulă fără cuantificatori, atunci $\neg\varphi$ este o formulă fără cuantificatori;
3. dacă φ și ψ sunt formule fără cuantificatori, atunci $\varphi \to \psi$ este o formulă fără cuantificatori.

Cum formulele fără cuantificatori au lungimi finite, rezultă că orice formulă fără cuantificatori se obține prin aplicarea regulilor ①, ② și ③ de un număr finit de ori.

Ceilalți conectori logici și celălalt cuantificator

Notație

Introducem abrevierile: pentru orice formule φ, ψ și orice variabilă x:

- **conectorii logici derivați** $\vee$ (*disjuncția*), $\wedge$ (*conjuncția*) și $\leftrightarrow$ (*echivalența*) se definesc astfel:
  $\varphi \vee \psi = \neg \varphi \rightarrow \psi$
  $\varphi \wedge \psi = \neg (\varphi \rightarrow \neg \psi)$
  $\varphi \leftrightarrow \psi = (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
- **cuantificatorul existențial** $\exists$ (*există*) se definește astfel:
  $\exists x \varphi = \neg \forall x \neg \varphi$

Observație (convenție privind scrierea conectorilor logici, a cuantificatorilor și a simbolului de egalitate)

- $\neg, \forall, \exists$ vor avea prioritate mai mare;
- $\rightarrow, \vee, \wedge, \leftrightarrow, =$ vor avea prioritate mai mică.

Variabilele care apar într–un termen

Notație (mulțimile din această notație vor fi definite recursiv mai jos)

Pentru orice termen t și orice formulă φ, notăm:

- $V(t)$ = mulțimea variabilelor termenului t
- $FV(\varphi)$ = mulțimea variabilelor *libere* ale formulei φ
- $V(\varphi)$ = mulțimea variabilelor formulei φ

Definiție

Pentru orice termen t:

- dacă $t = x$, unde x este o variabilă, atunci $V(t) = \{x\}$
- dacă $t = c$, unde c este o constantă, atunci $V(t) = \emptyset$
- dacă $n \in \mathbb{N}^*$ și $t = f(t_1, \ldots, t_n)$, unde f este un simbol de operație n–ară și $t_1, \ldots, t_n$ sunt termeni, atunci $V(t) = \bigcup_{i=1}^{n} V(t_i)$

Notație ($t = t(x_1, \ldots, x_n)$)

Dacă t este un termen și $V(t) = \{x_1, \ldots, x_n\}$, cu $n \in \mathbb{N}^*$, atunci termenul t se mai scrie și sub forma $t(x_1, \ldots, x_n)$.

Variabilele și variabilele libere dintr–o formulă

Definiție

Pentru orice formulă φ:

- dacă $\varphi = (t_1 = t_2)$, unde t_1 și t_2 sunt termeni, atunci $V(\varphi) = FV(\varphi) = V(t_1) \cup V(t_2)$
- dacă $\varphi = R(t_1, \ldots, t_m)$, unde R este un simbol de relație m–ară și $t_1, \ldots, t_m$ sunt termeni, atunci $V(\varphi) = FV(\varphi) = \bigcup_{i=1}^{m} V(t_i)$
- dacă $\varphi = \neg\psi$, pentru o formulă ψ, atunci $V(\varphi) = V(\psi)$ și $FV(\varphi) = FV(\psi)$
- dacă $\varphi = \psi \to \chi$, pentru două formule ψ, χ, atunci $V(\varphi) = V(\psi) \cup V(\chi)$ și $FV(\varphi) = FV(\psi) \cup FV(\chi)$
- dacă $\varphi = \forall x\psi$, pentru o formulă ψ și o variabilă x, atunci $V(\varphi) = V(\psi)$, iar $FV(\varphi) = FV(\psi) \setminus \{x\}$

Notație ($\varphi = \varphi(x_1, \ldots, x_n)$)

Dacă φ este o formulă și, pentru un $n \in \mathbb{N}^*$, $V(\varphi) \subseteq \{x_1, \ldots, x_n\}$, atunci formula φ se mai scrie și sub forma $\varphi(x_1, \ldots, x_n)$.

Variabilele și variabilele libere dintr–o formulă

Remarcă

Este imediat, din definiția anterioară și definiția conectorilor logici derivați și a cuantificatorului existențial, că, pentru orice formule ψ, χ și orice variabilă x:

- $V(\psi \vee \chi) = V(\psi \wedge \chi) = V(\psi \leftrightarrow \chi) = V(\psi) \cup V(\chi)$ și $FV(\psi \vee \chi) = FV(\psi \wedge \chi) = FV(\psi \leftrightarrow \chi) = FV(\psi) \cup FV(\chi)$
- $V(\exists x\psi) = V(\psi)$, iar $FV(\exists x\psi) = FV(\psi) \setminus \{x\}$

Definiție

Pentru orice variabilă x și orice formulă φ:

- dacă $x \in FV(\varphi)$, atunci x se numește *variabilă liberă a lui* φ;
- dacă $x \in V(\varphi) \setminus FV(\varphi)$, atunci x se numește *variabilă legată a lui* φ;
- dacă $FV(\varphi) = \emptyset$ (i. e. φ nu are variabile libere), atunci φ se numește *enunț*.

Remarcă

Formulele fără cuantificatori sunt exact formulele φ cu $V(\varphi) = FV(\varphi)$, adică exact formulele fără variabile legate, i.e. formulele în care toate variabilele sunt libere (adică nu sunt cuantificate, nu li se aplică niciun cuantificator).

1. Să reluăm noțiunile din Logica Clasică a Predicatelor care stau la baza funcționării Prologului

2. Cu ce fel de signaturi și structuri algebrice de acele signaturi și cu ce tipuri/cazuri particulare de formule lucrează limbajul Prolog?

3. Modele pentru formule, satisfiabilitate în Logica Clasică a Predicatelor

4. Rezoluția în Logica Clasică a Predicatelor – regula de deducție care stă la baza funcționării limbajului Prolog

5. Rezoluția în Prolog

În primul rând, ce tip de signatură avem în Prolog și cu ce fel de structuri algebrice de acea signatură lucrăm?
Am observat că avem nevoie de **operații parțiale**, fiecare având în *domeniul de definiție* elemente de un anumit **tip de date** (sau de mai multe tipuri de date).
Toate tipurile de date sunt înglobate în **tipul termen**, adică sunt **subtipuri** ale tipului termen.
Să considerăm o algebră $\mathcal{A} = (A; (f_i)_{i\in I}; (R_j)_{j\in J}; (c_k)_{k\in K})$ de o signatură $\tau = ((n_i)_{i\in I}; (m_j)_{j\in J}; (0)_{k\in K})$.
Cum trebuie să arate algebra $\mathcal{A}$ ca să corespundă unui program în Prolog?
Predicatele vor fi reprezentate prin **relațiile** din familia de relații $(R_j)_{j\in J}$. Știm că, în Prolog, putem imbrica predicatele, așadar avem nevoie de următoarele:

- mulțimea A trebuie să conțină constantele booleene **true** și **false**;
- o parte dintre operațiile parțiale din familia $(f_i)_{i\in I}$ sunt asociate relațiilor din familia $(R_j)_{j\in J}$, astfel:
  considerăm $J \subseteq I$;
  pentru fiecare $j \in J$, avem: $n_j = m_j$ și $f_j : A^{n_j} \dashrightarrow A$ (cu $Im(f_j) \subseteq \{\text{true}, \text{false}\}$, unde, prin **definiție**, imaginea funcției parțiale f_j coincide cu imaginea funcției $f_j\,|_{dom(f_j)}\colon dom(f_j) \to A$), definită prin: oricare ar fi $(a_1, \ldots, a_{n_j}) \in dom(f_j)$,

$$f(a_1, \ldots, a_{n_j}) = \begin{cases} \text{true}, & \text{dacă } (a_1, \ldots, a_{n_j}) \in R_j; \\ \text{false}, & \text{altfel;} \end{cases}$$

în practică, vom nota funcțiile f_j la fel ca pe relațiile (predicatele) R_j;

- iar **conectorii logici** fac parte dintre operațiile din familia $(f_i)_{i \in I}$, adică există $i_1, i_2, i_3, i_4, i_5 \in I$, astfel încât:

  $n_{i_1} = 1$, $dom(f_{i_1}) = \{\text{true}, \text{false}\}$ și $\begin{cases} f_{i_1}(\text{true}) = \text{false} \text{ și} \\ f_{i_1}(\text{false}) = \text{true}; \end{cases}$

  $n_{i_2} = n_{i_3} = n_{i_4} = n_{i_5} = 2$,

  $dom(f_2) = dom(f_3) = dom(f_4) = dom(f_5) = \{\text{true}, \text{false}\}^2$ și:

  $\begin{cases} f_{i_2}(\text{true}, \text{false}) = \text{false} \text{ și} \\ f_{i_2}(x, y) = \text{true}, \text{ dacă } (x, y) \in \{\text{true}, \text{false}\}^2 \setminus \{\text{true}, \text{false}\}; \end{cases}$

  pentru orice $(x, y) \in \{\text{true}, \text{false}\}^2$, $f_{i_3} = f_{i_2}(f_{i_1}(x), y)$;

  pentru orice $(x, y) \in \{\text{true}, \text{false}\}^2$, $f_{i_4} = f_{i_1}(f_{i_2}(x, f_{i_1}(y)))$;

  pentru orice $(x, y) \in \{\text{true}, \text{false}\}^2$, $f_{i_5} = f_{i_4}(f_{i_2}(x, y), f_{i_2}(y, x))$.

Vom nota funcțiile de mai sus la fel ca pe conectorii logici cărora le corespund: $f_{i_1} = \neg$, $f_{i_2} = \rightarrow$, $f_{i_3} = \vee$, $f_{i_4} = \wedge$ și $f_{i_5} = \leftrightarrow$, cu aceeași regulă a priorităților pentru a evita parantezări excesive și cu scriere infixată pentru f_{i_2}, f_{i_3}, f_{i_4} și f_{i_5}, la fel ca în cazul conectorilor logici.

Pentru orice operație f cu $Im(f) \subseteq \{\text{true}, \text{false}\}$, compunerea $f_{i_1} \circ f$ va fi notată, simplu, $\neg f$. La fel pentru f_{i_2}, f_{i_3}, f_{i_4}, f_{i_5} aplicate unor termeni cu alți operatori dominanți.

Cu ce tipuri/cazuri particulare de formule lucrează Prologul?

De exemplu, Prologul acceptă următoarea bază de cunoștințe:

```
b(1).        % corespunde formulei fără variabile: b(1)

a(0).        % corespunde formulei fără variabile: a(0)
a(false).    % corespunde formulei fără variabile: a(false)
a(b(_)).     % corespunde formulei: ∀X a(b(X))
```

Dar la interogarea:

```
?- a(not(b(1))).    % corespunde formulei fără variabile: a(¬(b(1)))
```

răspunsul este **false**, deși $not(b(1))$ este evaluat la **false**, adică operația (mai precis compunerea de operații) $\neg\, b$ ia în argumentul 1 valoarea **false**;
răspunsul la interogarea de mai sus este **false** pentru că termenul $not(b(1))$, de operator dominant *not* (adică $\neg$ cu notația de mai sus), nu unifică, cu niciunul dintre termenii 0, *false* și $b(X)$, unde $X \in Var$ (termenii care apar ca argumente ale lui a în faptele de mai sus care definesc predicatul a).
În schimb, dacă adăugăm la baza de cunoștințe regula:

```
c(X) :- not(X).    % corespunde formulei care poate fi scrisă astfel pentru a
                   % concorda cu teoria anterioară: ∀X(¬(X=true) → c(X))
```

atunci la interogarea:

?- $c(not(b(1)))$. % corespunde formulei fără variabile: $c(\neg(b(1)))$

Prologul răspunde **true**, pentru că X și $not(b(1))$ unifică, iar $b(1)$ e satisfăcut, prin urmare $not(b(1))$ nu e satisfăcut, așadar $not(not((b(1)))$ (adică $not(X)$ pentru X luând valoarea $not(b(1)))$ este satisfăcut, prin urmare $c(not(b(1)))$ (adică $c(X)$ pentru X luând valoarea $not(b(1)))$ e satisfăcut.
Observăm că Prologul nu acceptă fapte sau membri stângi de reguli conținând conectori logici. De exemplu, nu acceptă regula sau fapta următoare:

$a(X), b(X)$:- $a(b(X))$.
$a(X); b(X)$.

În schimb, acceptă fapte sau reguli de forma:

$d(a(X), b(X))$. % corespunde formulei: $\forall X\, d(a(X) \wedge b(X))$
$e(a(X); b(X))$:- $a(b(X))$.% corespunde formulei: $\forall X\, [a(b(X)) \rightarrow e(a(X) \vee b(X))]$

Cu notațiile de operații de mai sus, argumentele predicatelor (relațiilor) d, respectiv e sunt **termenii** $a(X) \wedge b(X)$, respectiv $a(X) \vee b(X)$. f A se vedea și fișierul *testfaptemsreguli.pl*; vom reveni la modul în care Prologul răspunde interogărilor. Așadar:

Cu ce tipuri/cazuri particulare de formule lucrează Prologul?

Faptele corespund unor formule de tipul:

$|\overline{\forall x_1 \forall x_2 \ldots \forall x_n \varphi}|$, unde $n \in \mathbb{N}^*$ și:

- φ este o formulă atomică;
- $V(\varphi) = FV(\varphi) = \{x_1, x_2, \ldots, x_n\}$, așadar $\forall x_1 \forall x_2 \ldots \forall x_n \varphi$ este un enunț.

Regulile corespund unor formule de tipul:

$|\overline{\forall x_1 \forall x_2 \ldots \forall x_n (\varphi \to \psi)}|$, unde $n \in \mathbb{N}^*$ și:

- φ este o formulă fără cuantificatori (φ este membrul drept al regulii);
- ψ este o formulă atomică (ψ este membrul stâng al regulii);
- așadar $V(\varphi \to \psi) = V(\varphi) \cup V(\psi) = FV(\varphi) \cup FV(\psi) = FV(\varphi \to \psi)$, deci $\varphi \to \psi$ este o formulă fără cuantificatori;
- $V(\varphi \to \psi) = \{x_1, x_2, \ldots, x_n\}$, așadar $\forall x_1 \forall x_2 \ldots \forall x_n (\varphi \to \psi)$ este un enunț.

Interogările corespund unor formule de tipul:

$|\overline{\exists x_1 \exists x_2 \ldots \exists x_n \varphi}|$, unde $n \in \mathbb{N}^*$ și:

- φ este o formulă fără cuantificatori;
- $V(\varphi) = FV(\varphi) = \{x_1, x_2, \ldots, x_n\}$, așadar $\exists x_1 \exists x_2 \ldots \exists x_n \varphi$ este un enunț.

1. Să reluăm noțiunile din Logica Clasică a Predicatelor care stau la baza funcționării Prologului

2. Cu ce fel de signaturi și structuri algebrice de acele signaturi și cu ce tipuri/cazuri particulare de formule lucrează limbajul Prolog?

3. Modele pentru formule, satisfiabilitate în Logica Clasică a Predicatelor

4. Rezoluția în Logica Clasică a Predicatelor – regula de deducție care stă la baza funcționării limbajului Prolog

5. Rezoluția în Prolog

Fie:

- $\tau = ((n_i)_{i\in I}; (m_j)_{j\in J}; (0)_{k\in K})$ o signatură;
- $\mathcal{A} = (A; (f_i)_{i\in I}; (R_j)_{j\in J}; (c_k)_{k\in K})$ o structură de ordinul I de signatură τ;

notăm: $\begin{cases} \mathcal{F} = \{f_i \mid i \in I\}, \\ \mathcal{R} = \{R_j \mid j \in J\}, \\ \mathcal{C} = \{c_k \mid k \in K\}; \end{cases}$ adică:

$\mathcal{F}$ este mulțimea operațiilor cu argumente ale lui $\mathcal{A}$;
$\mathcal{R}$ este mulțimea relațiilor lui $\mathcal{A}$;
$\mathcal{C}$ este mulțimea constantelor lui $\mathcal{A}$.

Amintesc că am notat cu *Var* mulțimea **variabilelor** din limbajul $\mathcal{L}_\tau$ (limbajul de ordinul I asociat signaturii τ).

Elementele lui *Var* nu sunt variabile propoziționale, ci sunt variabilele care apar în predicate, i.e. în propozițiile cu variabile.

În continuare ne vom referi la **termenii** și **formulele** din limbajul $\mathcal{L}_\tau$.

Definiție

O *interpretare* (sau *evaluare*, sau *semantică*) a limbajului $\mathcal{L}_\tau$ în structura algebrică $\mathcal{A}$ este o funcție $s : Var \to A$.
Fiecare variabilă $x \in Var$ este "interpretată" prin elementul $s(x) \in A$.

Prelungim o interpretare s la mulțimea tuturor termenilor, obținând o funcție $s : Term(\mathcal{L}_\tau) \to A$.

Definiție (valorile asociate termenilor de o interpretare)

Pentru orice interpretare s și orice termen t, definim recursiv elementul $s(t) \in A$:

- dacă $t = x \in Var$, atunci $s(t) = s(x)$;
- dacă $t = c \in C$, atunci $s(t) = c$;
- dacă $t = f(t_1, \ldots, t_n)$, unde $f \in F$ are aritatea $n \in \mathbb{N}^*$, iar $t_1, \ldots, t_n \in Term(\mathcal{L}_\tau)$, atunci $s(t) = f(s(t_1), \ldots, s(t_n))$.

Notație

Pentru orice interpretare $s : Var \to A$, orice $x \in Var$ și orice $a \in A$, notăm cu $s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right] : Var \to A$ interpretarea definită prin: oricare ar fi $v \in Var$,

$$s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right](v) = \begin{cases} a, & \text{dacă } v = x, \\ s(v), & \text{dacă } v \neq x. \end{cases}$$

$s\left[\begin{smallmatrix} x \\ a \end{smallmatrix}\right]$ ia valoarea a în variabila x și aceeași valoare ca s în celelalte variabile.

Acum definim valorile unei interpretări s în formule; acestea vor fi valori din **algebra Boole standard**, **algebra Boole a valorilor de adevăr** $\mathcal{L}_2 = (\{0,1\}, \vee, \wedge, \bar{\cdot}, 0, 1)$, unde $0 \neq 1$, $\vee$ este **disjuncția** și $\wedge$ este **conjuncția** (operații de latice), $\bar{\cdot}$ este **complementul** (operație booleană), 0 este **primul element**, iar 1 este **ultimul element**. Notăm cu $\rightarrow$ **implicația booleană**, iar cu $\leftrightarrow$ **echivalența booleană** în $\mathcal{L}_2$ (operații booleene derivate). A se revedea lecția de curs recapitulativă despre algebrele Boole.

Astfel, obținem o funcție $s : Term(\mathcal{L}_\tau) \cup Form(\mathcal{L}_\tau) \rightarrow A \cup \mathcal{L}_2$.

Definiție (valorile booleene asociate formulelor de o interpretare)

Pentru orice interpretare s și orice formulă φ, *valoarea de adevăr a lui φ în interpretarea s* este un element din algebra Boole standard $\mathcal{L}_2 = \{0,1\}$, notat cu $s(\varphi)$, definit, recursiv, astfel:

- dacă $\varphi = (t_1 = t_2)$, pentru doi termeni t_1, t_2, atunci
$$s(\varphi) = \begin{cases} 1, & \text{dacă } s(t_1) = s(t_2), \\ 0, & \text{dacă } s(t_1) \neq s(t_2) \end{cases}$$
- dacă $\varphi = R(t_1, \ldots, t_m)$, unde $R \in \mathcal{R}$ are aritatea $m \in \mathbb{N}^*$, iar $t_1, \ldots, t_m$ sunt termeni, atunci $s(\varphi) = \begin{cases} 1, & \text{dacă } (s(t_1), \ldots, s(t_m)) \in R, \\ 0, & \text{dacă } (s(t_1), \ldots, s(t_m)) \notin R \end{cases}$

Definiție (continuare – am definit mai sus valorile lui φ în formulele atomice; acum extindem definiția lui φ la toate formulele)

- dacă $\varphi = \neg\psi$, pentru o formulă ψ, atunci $s(\varphi) = \overline{s(\psi)}$;
- dacă $\varphi = \psi \to \chi$, pentru două formule ψ, χ, atunci $s(\varphi) = s(\psi) \to s(\chi)$;
- dacă $\varphi = \forall x\psi$, unde $x \in Var$, iar ψ este o formulă, atunci $s(\varphi) = \bigwedge_{a\in A} s[^x_a](\psi)$.

Remarcă

Este imediat că, pentru orice interpretare $s : Var \to A$, orice formule ψ, χ și orice $x \in Var$, au loc egalitățile:

- $s(\psi \vee \chi) = s(\psi) \vee s(\chi)$;
- $s(\psi \wedge \chi) = s(\psi) \wedge s(\chi)$;
- $s(\psi \leftrightarrow \chi) = s(\psi) \leftrightarrow s(\chi)$;
- $s(\exists x\psi) = \bigvee_{a\in A} s[^x_a](\psi)$.

Fie $s : Var \to A$, $x \in Var$ și $\psi \in Form(\mathcal{L}_\tau)$. Cum $s(\alpha) \in \mathcal{L}_2 = \{0, 1\}$ pentru orice $\alpha \in Form(\mathcal{L}_\tau)$, din cele de mai sus rezulta că: $s(\forall x\psi) = 1$ ddacă $s[^x_a](\psi) = 1$ pentru toți $a \in A$, iar $s(\exists x\psi) = 1$ ddacă există un $a \in A$ a.î. $s[^x_a](\psi) = 1$.

Lemă (dacă două interpretări coincid pe variabilele dintr–un termen, atunci ele coincid în acel termen)

Fie $s_1, s_2 : Var \to A$ două interpretări. Atunci, pentru orice termen t, are loc implicația: $s_1 \mid_{V(t)} = s_2 \mid_{V(t)} \Rightarrow s_1(t) = s_2(t)$.

Propoziție (dacă două interpretări coincid pe variabilele libere dintr–o formulă, atunci ele coincid în acea formulă)

Fie $s_1, s_2 : Var \to A$ două interpretări. Atunci, pentru orice formulă φ, are loc implicația: $s_1 \mid_{FV(\varphi)} = s_2 \mid_{FV(\varphi)} \Rightarrow s_1(\varphi) = s_2(\varphi)$.

În mod trivial, oricare două interpretări $s_1, s_2 : Var \to A$ coincid pe $\emptyset$: $(\forall\, x)(x \in \emptyset \Rightarrow s_1(x) = s_2(x))$ e adevărată, pentru că $x \in \emptyset$ e falsă pentru orice x, așadar $x \in \emptyset \Rightarrow s_1(x) = s_2(x)$ e adevărată pentru orice x. Prin urmare:

Corolar (cum enunțurile nu au variabile libere (i.e. $FV(\varphi) = \emptyset$ pt. orice enunț φ), rezultă că, într–un enunț, toate interpretările au aceeași valoare)

Dacă φ este un enunț, atunci $s(\varphi)$ nu depinde de interpretarea $s : Var \to A$.

Notație (această valoare nu depinde decât de structura algebrică $\mathcal{A}$)

Corolarul anterior ne permite să notăm, pentru orice enunț φ, cu $||\varphi||_{\mathcal{A}} = s(\varphi)$ valoarea oricărei interpretări $s : Var \to A$ în enunțul φ.

Satisfiabilitate, adevăruri semantice, deducție semantică

Definiție (modele pentru enunțuri și mulțimi de enunțuri)

Pentru orice enunț φ, notăm:

$$\mathcal{A} \models \varphi \text{ ddacă } ||\varphi||_{\mathcal{A}} = 1,$$

și, în acest caz, spunem că $\mathcal{A}$ *satisface* φ sau φ *este adevărat în* $\mathcal{A}$ sau $\mathcal{A}$ *este model pentru* φ.

Pentru orice mulțime Γ de enunțuri, notăm:

$$\mathcal{A} \models \Gamma \text{ ddacă } (\forall\, \gamma \in \Gamma)\, (\mathcal{A} \models \gamma),$$

și, în acest caz, spunem că $\mathcal{A}$ *satisface* Γ sau că $\mathcal{A}$ *este model pentru* Γ.

Definiție (satisfiabilitate)

Spunem că un enunț φ *e satisfiabil* sau *are un model* ddacă există o structură de ordinul I $\mathcal{M}$ de signatură τ astfel încât $\mathcal{M} \models \varphi$.

Spunem că o mulțime Γ de enunțuri *e satisfiabilă* sau *are un model* ddacă există o structură de ordinul I $\mathcal{M}$ de signatură τ astfel încât $\mathcal{M} \models \Gamma$.

Definiție (adevărurile semantice și deducția semantică)

Fie φ un enunț. Spunem că φ este *universal adevărat (adevăr semantic, tautologie)* ddacă $\mathcal{M} \models \varphi$ pentru orice structură de ordinul I $\mathcal{M}$ de signatură τ.

Echivalența semantică pe mulțimea enunțurilor lui $\mathcal{L}_\tau$

Definiție (continuare)

Notăm acest lucru cu $\models \varphi$.
Fie Γ o mulțime de enunțuri. Spunem că φ este *consecință semantică* a lui Γ sau φ enunț *deductibil semantic din* Γ ddacă orice model pentru Γ este model pentru φ, adică, pentru orice structură de ordinul I $\mathcal{M}$ de signatură τ, $\mathcal{M} \models \Gamma$ implică $\mathcal{M} \models \varphi$. Notăm acest lucru cu $\Gamma \models \varphi$.

Amintesc că mulțimea enunțurilor lui $\mathcal{L}_\tau$ este $\{\varphi \in \mathit{Form}(\mathcal{L}_\tau) \mid \mathit{FV}(\varphi) = \emptyset\}$.

Definiție (echivalența semantică)

Considerăm pe pe mulțimea enunțurilor din limbajul $\mathcal{L}_\tau$ o relație binară notată $\models\!\!\models$, definită astfel: oricare ar fi enunțurile φ și ψ, $\varphi \models\!\!\models \psi$ ddacă, oricare ar fi structura algebrică $\mathcal{M}$ de signatură τ, avem: $\mathcal{M} \models \varphi$ ddacă $\mathcal{M} \models \psi$.
Adică o pereche de enunțuri $(\varphi, \psi) \in \models\!\!\models$ ddacă φ și ψ au aceleași modele.
Relația binară $\models\!\!\models$ se numește *echivalența semantică* în limbajul $\mathcal{L}_\tau$. Pentru orice enunțuri φ, ψ, spunem că φ și ψ sunt *echivalente semantic* ddacă $(\varphi, \psi) \in \models\!\!\models$.

Remarcă

$\models\!\!\models$ este o relație de echivalență pe mulțimea enunțurilor lui $\mathcal{L}_\tau$. Într–adevăr, reflexivitatea, simetria și tranzitivitatea lui $\models\!\!\models$ sunt imediate.

Remarcă (două enunțuri sunt echivalente semantic ddacă echivalența lor logică este tautologie)

Pentru orice enunțuri φ, ψ, avem:

$$\varphi \models\!\models \psi \text{ ddacă } \models \varphi \leftrightarrow \psi.$$

Într–adevăr, în primul rând să observăm că $\varphi \leftrightarrow \psi$ este un enunț, întrucât $FV(\varphi \leftrightarrow \psi) = FV(\varphi) \cup FV(\psi) = \emptyset \cup \emptyset = \emptyset$.
Acum, considerând o interpretare arbitrară $s : Var \to A$, avem:
$||\varphi \leftrightarrow \psi||_{\mathcal{A}} = s(\varphi \leftrightarrow \psi) = s(\varphi) \leftrightarrow s(\psi) = ||\varphi||_{\mathcal{A}} \leftrightarrow ||\psi||_{\mathcal{A}}$.
Prin urmare: $\models \varphi \leftrightarrow \psi$ ddacă, pentru orice algebră $\mathcal{M}$ de tip τ, $\mathcal{M} \models \varphi \leftrightarrow \psi$, dacă, pentru orice algebră $\mathcal{M}$ de tip τ, $||\varphi \leftrightarrow \psi||_{\mathcal{M}} = 1$, dacă, pentru orice algebră $\mathcal{M}$ de tip τ, $||\varphi||_{\mathcal{M}} \leftrightarrow ||\psi||_{\mathcal{M}} = 1$, dacă, pentru orice algebră $\mathcal{M}$ de tip τ, $||\varphi||_{\mathcal{M}} = ||\psi||_{\mathcal{M}}$, dacă, pentru orice algebră $\mathcal{M}$ de tip τ, avem echivalența: $||\varphi||_{\mathcal{M}} = 1 \Leftrightarrow ||\psi||_{\mathcal{M}} = 1$, dacă, pentru orice algebră $\mathcal{M}$ de tip τ, avem echivalența: $\mathcal{M} \models \varphi \Leftrightarrow \mathcal{M} \models \psi$, ddacă $\varphi \models\!\models \psi$.
Am folosit următoarele fapte:

- pentru orice elemente a, b ale unei algebre Boole, avem: $a \leftrightarrow b = 1$ ddacă $a = b$;
- oricare ar fi algebra $\mathcal{M}$ de tip τ: $||\varphi||_{\mathcal{M}}, ||\psi||_{\mathcal{M}} \in \mathcal{L}_2 = \{0, 1\}$, așadar $||\varphi||_{\mathcal{M}} = ||\psi||_{\mathcal{M}}$ ddacă fie ambele sunt egale cu 1, fie ambele sunt egale cu 0.

Remarcă

Conform remarcii anterioare, pentru orice enunțuri φ, ψ, avem: $\varphi \models\!\mid \psi$ ddacă, pentru orice algebră $\mathcal{M}$ de tip τ, $||\varphi||_{\mathcal{M}} = ||\psi||_{\mathcal{M}}$.
Din această exprimare a relației $\models\!\mid$ de echivalență semantică pentru $\mathcal{L}_\tau$ rezultă că $\models\!\mid$ are proprietățile relației $\models\!\mid$ de echivalență semantică din logica propozițională clasică, anume cele provenite din proprietățile booleene *(a se vedea mai jos)*.
În plus, au loc proprietăți de tipul următor *(a se vedea o listă extinsă mai jos)*, pentru orice $x, y \in Var$ și orice $\alpha, \beta \in Form(\mathcal{L}_\tau)$:

- dacă $FV(\alpha) \subseteq \{x, y\}$, astfel că următoarele formule sunt enunțuri, atunci: $\forall x \forall y \alpha \models\!\mid \forall y \forall x \alpha$;
- dacă α este un enunț, $x \notin V(\alpha)$ și $FV(\beta) \subseteq \{x\}$, astfel că următoarele formule sunt enunțuri, atunci: $\alpha \models\!\mid \forall x \alpha \models\!\mid \exists x \alpha$ și $\forall x(\alpha \vee \beta) \models\!\mid \alpha \vee \forall x \beta \models\!\mid \forall x \alpha \vee \forall x \beta$.

Într–adevăr, considerând $x, y \in Var$, două **formule arbitrare** α, β, o algebră $\mathcal{M}$ de tip τ, cu mulțimea elementelor M, și o interpretare $s : Var \to M$, avem: notând, pentru orice $a, b \in M$, cu $t_{a,b} : Var \to M$ interpretarea definită prin:

oricare ar fi $v \in Var$, $t_{a,b}(v) = \begin{cases} a, & \text{dacă } v = x, \\ b, & \text{dacă } v = y, \\ s(v), & \text{dacă } v \notin \{x, y\}, \end{cases}$ observăm că

$t_{a,b} = (s[^x_a])[^y_b] = (s[^y_b])[^x_a]$, așadar:

$$s(\forall x \forall y \alpha) = \bigwedge_{a \in A} s[^x_a](\forall y \alpha) = \bigwedge_{a \in A} \bigwedge_{b \in A} (s[^x_a])[^y_b](\alpha) = \bigwedge_{a \in A} \bigwedge_{b \in A} t_{a,b}(\alpha) =$$

$$\bigwedge_{b \in A} \bigwedge_{a \in A} t_{a,b}(\alpha) = \bigwedge_{b \in A} \bigwedge_{a \in A} (s[^y_b])[^x_a](\alpha) = \bigwedge_{b \in A} s[^y_b](\forall x \alpha) = s(\forall y \forall x \alpha);$$

iar, dacă $\forall x \forall y \alpha$ este un **enunț**, adică $FV(\alpha) \subseteq \{x, y\}$, astfel că și $\forall y \forall x \alpha$ este un enunț, atunci, conform calculului anterior,
$||\forall x \forall y \alpha||_{\mathcal{M}} = s(\forall x \forall y \alpha) = s(\forall y \forall x \alpha) = ||\forall x \forall y \alpha||_{\mathcal{M}}$, **și la fel mai jos**;

- dacă $x \notin FV(\alpha)$, în particular dacă $x \notin V(\alpha)$, atunci, pentru orice $a \in A$, $s \mid_{FV(\alpha)} = s[^x_a] \mid_{FV(\alpha)}$, așadar, conform propoziției precedente, pentru orice $a \in A$, $s(\alpha) = s[^x_a](\alpha)$, prin urmare:

$$s(\forall x \alpha) = \bigwedge_{a \in A} s[^x_a](\alpha) = s(\alpha) = \bigvee_{a \in A} s[^x_a](\alpha) = s(\exists x \alpha);$$

întrucât algebra Boole $\mathcal{L}_2$ este finită și, în particular, completă, fapt pe care l–am folosit deja când am admis scrieri de tipul $\bigwedge_{a \in A} s[^x_a](\alpha)$ sau $\bigvee_{a \in A} s[^x_a](\alpha)$, deci am folosit existența conjuncțiilor și disjuncțiilor arbitrare, avem:

$$s(\forall x(\alpha \vee \beta)) = \bigwedge_{a \in A} s[^x_a](\alpha \vee \beta) = \bigwedge_{a \in A} (s[^x_a](\alpha) \vee s[^x_a](\beta)) = \bigwedge_{a \in A} (s(\alpha) \vee s[^x_a](\beta)) =$$

$s(\alpha) \vee \bigwedge_{a \in A} s[^x_a](\beta) = s(\alpha) \vee s(\forall x \beta) = s(\alpha \vee \forall x \beta)$, dar, conform celor de mai sus, avem și: $s(\alpha) \vee s(\forall x \beta) = s(\forall x \alpha) \vee s(\forall x \beta) = s(\forall x \alpha \vee \forall x \beta)$.

Să ne amintim ce sunt substituțiile

Definiție (substituțiile sunt funcțiile de la variabile la termeni)

O *substituție* este o funcție $\sigma : Var \to Term(\mathcal{L}_\tau)$.

Definiție

Fie $\sigma : Var \to Term(\mathcal{L}_\tau)$. Următoarea extindere a lui σ la $Term(\mathcal{L}_\tau)$ se numește tot *substituție* (și, de obicei, se notează tot cu σ): $\widetilde{\sigma} : Term(\mathcal{L}_\tau) \to Term(\mathcal{L}_\tau)$, definită, recursiv, astfel:

- $\widetilde{\sigma}\,|_{Var} = \sigma$;
- oricare ar fi $c \in \mathcal{C}$, $\widetilde{\sigma}(c) = c$;
- pentru orice $n \in \mathbb{N}^*$, orice $f \in \mathcal{F}$ având aritatea n și orice $t_1, \ldots, t_{n_i} \in Term(\mathcal{L}_\tau)$, $\widetilde{\sigma}(f(t_1, \ldots, t_{n_i})) = f(\widetilde{\sigma}(t_1), \ldots, \widetilde{\sigma}(t_n))$.

Observație

Extinderea $\widetilde{\sigma}$ din definiția anterioară este:

- **complet definită**, pentru că toți termenii se scriu ca mai sus, i.e. sunt obținuți prin acea recursie;
- **corect definită**, pentru că orice termen are o unică scriere ca mai sus.

Definiție (definim substituțiile pe formulele atomice, apoi, recursiv, pe toate formulele fără cuantificatori)

Fie $\sigma : Term(\mathcal{L}_\tau) \to Term(\mathcal{L}_\tau)$ o substituție (adică o extindere ca în definiția anterioară a unei funcții de la Var la $Term(\mathcal{L}_\tau)$).
Definim σ pe formulele atomice, recursiv, astfel:

- pentru orice $t_1, t_2 \in Term(\mathcal{L}_\tau)$, $\sigma(t_1{=}t_2) \ = \ \sigma(t_1){=}\sigma(t_2)$;
- pentru orice $m \in \mathbb{N}^*$, orice $R \in \mathcal{R}$ de aritate m și orice $t_1, \ldots, t_m \in Term(\mathcal{L}_\tau)$, $\sigma(R(t_1, \ldots, t_m)) = R(\sigma(t_1), \ldots, \sigma(t_m))$.

Practic, mai sus, obținem valorile lui σ în termenii obținuți din formulele atomice $t_1{=}t_2$, respectiv $R(t_1, \ldots, t_m)$ prin înlocuirea relațiilor din acești termeni cu operațiile de rezultat boolean asociate acestora, care, într–o algebră $\mathcal{A}$ de signatură τ, având mulțimea elementelor $A \supseteq \mathcal{L}_2 = \{0, 1\}$, sunt definite prin:

- $=^{\mathcal{A}} : A^2 \to \mathcal{L}_2 \subseteq A$, pentru orice $a, b \in A$, $=^{\mathcal{A}}(a, b) = \begin{cases} 1, & \text{dacă } a = b, \\ 0, & \text{dacă } a \neq b; \end{cases}$
- $R^{\mathcal{A}} : A^m \to \mathcal{L}_2 \subseteq A$, pentru orice $a_1, \ldots, a_m \in A$,
  $R^{\mathcal{A}}(a_1, \ldots, a_m) = \begin{cases} 1, & \text{dacă } (a_1, \ldots, a_m) \in R^{\mathcal{A}}, \\ 0, & \text{dacă } (a_1, \ldots, a_m) \notin R^{\mathcal{A}}. \end{cases}$

Definiție (continuare – definiția recursivă a unei substituții pe formulele non–atomice fără cuantificatori, adică formulele cu conectori logici și fără cuantificatori)

După ce obținem valoarea lui σ într–o formulă atomică α asociind, ca mai sus, un termen lui α, facem operația inversă pentru termenul $\sigma(\alpha)$: înlocuim operația dominantă (având rezultatul boolean) a acestui termen cu relația căreia i–am asociat această operație, și astfel termenul $\sigma(\alpha)$ devine formulă atomică.

Pentru orice formule fără cuantificatori φ, ψ, definim:

- $\sigma(\neg\, \varphi) = \neg\, \sigma(\varphi)$;
- $\sigma(\varphi \to \psi) = \sigma(\varphi) \to \sigma(\psi)$.

Din definiția anterioară rezultă:

Remarcă (pentru conectorii logici derivați)

Pentru orice $\sigma : \mathit{Term}(\mathcal{L}_\tau) \to \mathit{Term}(\mathcal{L}_\tau)$ și orice formule fără cuantificatori φ, ψ, au loc:

- $\sigma(\varphi \vee \psi) = \sigma(\varphi) \vee \sigma(\psi)$;
- $\sigma(\varphi \wedge \psi) = \sigma(\varphi) \wedge \sigma(\psi)$;
- $\sigma(\varphi \leftrightarrow \psi) = \sigma(\varphi) \leftrightarrow \sigma(\psi)$.

Ce definește un program în Prolog și ce semnifică interogările?

Considerăm o signatură τ, un $n \in \mathbb{N}^*$ și o bază de cunoștințe în Prolog formată din n fapte și reguli corespunzătoare formulelor $\varphi_1, \ldots, \varphi_n \in Form(\mathcal{L}_\tau)$.

Această bază de cunoștințe definește clasa tuturor algebrelor de signatură τ care satisfac mulțimea de enunțuri $\{\varphi_1, \ldots, \varphi_n\}$, adică mulțimea modelelor lui $\{\varphi_1, \ldots, \varphi_n\}$.

Dacă o interogare corespunde formulei $\varphi = \exists\, x_1\, \exists\, x_2\, \ldots \exists\, x_k\, \psi \in Form(\mathcal{L}_\tau)$, unde ψ este o formulă fără cuantificatori, atunci a rezolva interogarea respectivă semnifică a determina dacă are loc deducția semantică:

$$\{\varphi_1, \ldots, \varphi_n\} \models \varphi,$$

adică dacă orice algebră de signatură τ care satisface mulțimea de enunțuri $\{\varphi_1, \ldots, \varphi_n\}$ satisface și pe φ, adică dacă orice algebră din clasa algebrelor definite de această bază de cunoștințe este model pentru φ.
Mai precis, după cum vom vedea, a rezolva această interogare înseamnă a găsi substituțiile $\sigma = \{x_1/t_1, \ldots, x_k/t_k\}$, unde $t_1, \ldots, t_k$ sunt termeni având $V(t_1) \cup \ldots \cup V(t_k) = \{v_1, \ldots, v_j\}$, cu proprietatea că:

$$\{\varphi_1, \ldots, \varphi_n\} \models \forall\, v_1\, \forall\, v_2\, \ldots \forall\, v_j\, \sigma(\psi).$$

1. Să reluăm noțiunile din Logica Clasică a Predicatelor care stau la baza funcționării Prologului

2. Cu ce fel de signaturi și structuri algebrice de acele signaturi și cu ce tipuri/cazuri particulare de formule lucrează limbajul Prolog?

3. Modele pentru formule, satisfiabilitate în Logica Clasică a Predicatelor

4. **Rezoluția în Logica Clasică a Predicatelor – regula de deducție care stă la baza funcționării limbajului Prolog**

5. Rezoluția în Prolog

Formă prenex și formă normală conjunctivă prenex

Fie τ și $\mathcal{A}$ ca în secțiunea anterioară a cursului.

Definiție

Se numește *formulă prenex* sau *formulă în formă prenex* o formulă de tipul

$$\mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n \varphi,$$

unde $n \in \mathbb{N}^*$, $x_1, \dots, x_n \in Var$, $\mathcal{Q}_1, \dots, \mathcal{Q}_n \in \{\forall, \exists\}$ și φ este o formulă fără cuantificatori.

Observație

Toate formulele cu care lucrează Prologul sunt în formă prenex: atât cele corespunzătoare **faptelor** și **regulilor**, cât și cele corespunzătoare **interogărilor**.

Remarcă (orice enunț poate fi pus în formă prenex)

Pentru orice enunț ε, există o formulă prenex ψ astfel încât $\varepsilon \models\!\models \psi$.

Putem obține o formulă prenex echivalentă semantic cu ε folosind următoarele echivalențe semantice valabile pentru orice enunțuri α, β, γ, orice formule φ, ψ, χ și orice variabile x, y a.î. $FV(\varphi) \cup FV(\psi) \subseteq \{x\}$, iar $FV(\chi) \subseteq \{x, y\}$, astfel că următoarele formule sunt enunțuri:

(proprietăți pentru mutarea cuantificatorilor în față)

- $\forall x \forall y \chi \models\!\models \forall y \forall x \chi$ și $\exists x \exists y \chi \models\!\models \exists y \exists x \chi$
- $\neg \forall x \chi \models\!\models \exists x \neg \chi$ și $\neg \exists x \chi \models\!\models \forall x \neg \chi$
- $\forall x(\varphi \wedge \psi) \models\!\models \forall x \varphi \wedge \forall x \psi$ și $\exists x(\varphi \vee \psi) \models\!\models \exists x \varphi \vee \exists x \psi$
- dacă $x \notin V(\varphi)$, atunci: $\begin{cases} \varphi \models\!\models \forall x \varphi \models\!\models \exists x \varphi \\ \varphi \vee \forall x \psi \models\!\models \forall x(\varphi \vee \psi) \models\!\models \forall x \varphi \vee \forall x \psi \\ \varphi \wedge \exists x \psi \models\!\models \exists x(\varphi \wedge \psi) \models\!\models \exists x \varphi \wedge \exists x \psi \end{cases}$

(proprietăți din calculul propozițional)

1. $\alpha \to \beta \models\!\models \neg \alpha \vee \beta$ și $\alpha \leftrightarrow \beta \models\!\models (\neg \alpha \vee \beta) \wedge (\neg \beta \vee \alpha)$
2. $\alpha \vee \alpha \models\!\models \alpha \wedge \alpha \models\!\models \alpha$
3. $\alpha \vee \beta \models\!\models \beta \vee \alpha$ și $\alpha \wedge \beta \models\!\models \beta \wedge \alpha$
4. $(\alpha \vee \beta) \vee \gamma \models\!\models \alpha \vee (\beta \vee \gamma)$ și $(\alpha \wedge \beta) \wedge \gamma \models\!\models \alpha \wedge (\beta \wedge \gamma)$
5. $\alpha \vee (\alpha \wedge \beta) \models\!\models \alpha \wedge (\alpha \vee \beta) \models\!\models \alpha$
6. $\alpha \vee (\beta \wedge \gamma) \models\!\models (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$ și $\alpha \wedge (\beta \vee \gamma) \models\!\models (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$
7. $\neg \neg \alpha \models\!\models \alpha$
8. $\neg(\alpha \vee \beta) \models\!\models \neg \alpha \wedge \neg \beta$ și $\neg(\alpha \wedge \beta) \models\!\models \neg \alpha \vee \neg \beta$
9. $\alpha \vee (\beta \wedge \neg \beta) \models\!\models \alpha \vee (\beta \wedge \neg \beta \wedge \gamma) \models\!\models \alpha \wedge (\beta \vee \neg \beta) \models\!\models \alpha \wedge (\beta \vee \neg \beta \vee \gamma) \models\!\models \alpha$

Remarcă

Cum, în ultima remarcă din secțiunea anterioară a cursului, α și β sunt **formule arbitrare**, rezultă că **echivalențele semantice** de mai sus au loc și în **subformule** care nu sunt neapărat enunțuri, adică putem aplica faptul că, pentru orice $\varphi, \psi \in Form(\mathcal{L}_\tau)$, orice algebră $\mathcal{M}$ de signatură τ, cu mulțimea elementelor M, și orice interpretare $s : Var \to M$, au loc, de exemplu:

- $s(\neg(\alpha \vee \beta)) = s(\neg\alpha \wedge \neg\beta)$, $s(\exists x \exists y \alpha) = s(\exists y \exists x \alpha)$;
- dacă $x \notin FV(\alpha)$, atunci $s(\alpha) = s(\forall x \alpha) = s(\exists x \alpha)$ și $s(\alpha \wedge \exists x \beta) = s(\exists x(\alpha \wedge \beta)) = s(\exists x \alpha \wedge \exists x \beta)$ etc..

Definiție (FNC prenex)

- Un *literal* este o formulă atomică sau negația unei formule atomice.
- O *clauză* este o disjuncție de literali.

Orice clauză se identifică cu mulțimea literalilor care o compun.

- O *formă normală conjunctivă prenex* sau un *enunț în formă normală conjunctivă prenex* (*FNC prenex*) este un enunț de forma:

$$\forall x_1 \ldots \forall x_n \varphi,$$

unde $n \in \mathbb{N}^*$, $x_1, \ldots, x_n \in Var$ și φ este o conjuncție de clauze, i. e. o

Definiție (continuare)

conjuncție de disjuncții de literali, implicit φ este o formulă fără cuantificatori, așadar $x_1, \ldots, x_n \in Var$ este o formulă prenex.

Orice conjuncție de clauze se identifică, cu mulțimea acelor clauze.
Orice FNC prenex $\forall x_1 \ldots \forall x_n \varphi$ se identifică cu mulțimea clauzelor care îl compun pe φ.

Observație

Întrucât toate enunțurile au lungime finită, conjuncțiile și disjuncțiile la care face referire definiția de mai sus sunt finite.

Remarcă

O mulțime finită și nevidă de enunțuri este satisfiabilă ddacă, conjuncția enunțurilor din acea mulțime e satisfiabilă.
Într–adevăr, dacă $n \in \mathbb{N}^*$ și $\gamma_1, \ldots, \gamma_n$ sunt enunțuri, iar $\mathcal{M}$ este o algebră de signatură τ, atunci: $||\gamma_1 \wedge \ldots \wedge \gamma_n||_{\mathcal{M}} = ||\gamma_1||_{\mathcal{M}} \wedge \ldots \wedge ||\gamma_n||_{\mathcal{M}}$, așadar: $\mathcal{M} \vDash \gamma_1 \wedge \ldots \wedge \gamma_n$ ddacă $||\gamma_1 \wedge \ldots \wedge \gamma_n||_{\mathcal{M}} = 1$ ddacă $||\gamma_1||_{\mathcal{M}} \wedge \ldots \wedge ||\gamma_n||_{\mathcal{M}} = 1$ ddacă $||\gamma_1||_{\mathcal{M}} = \ldots = ||\gamma_n||_{\mathcal{M}} = 1$ ddacă, pentru fiecare $i \in \overline{1, n}$, $\mathcal{M} \vDash \gamma_i$, ddacă $\mathcal{M} \vDash \{\gamma_1, \ldots, \gamma_n\}$.

Remarcă

În mod trivial, mulțimea vidă de enunțuri este satisfiabilă, întrucât următoarea afirmație este adevărată: pentru orice algebră $\mathcal{M}$ de tip τ și orice γ, $\gamma \in \emptyset \Rightarrow \mathcal{M} \models \gamma$.

Definiție

Dacă ε este un enunț, iar ψ este o FNC prenex cu $\varepsilon \models\!\models \psi$, atunci mulțimea de clauze care îl compun pe ψ se numește *formă clauzală* pentru ε.
Dacă $n \in \mathbb{N}^*$, iar $\varepsilon_1, \ldots, \varepsilon_n$ sunt enunțuri, atunci reuniunea unor forme clauzale pentru $\varepsilon_1, \ldots, \varepsilon_n$ se numește *formă clauzală* pentru $\{\varepsilon_1, \ldots, \varepsilon_n\}$.

Definiție

- **Clauza vidă** (i. e. clauza fără literali, clauza fără elemente) se notează cu $\square$ (pentru a o deosebi de **mulțimea vidă de clauze**, $\emptyset$).
- O clauză C se zice *trivială* ddacă există o formulă atomică α astfel încât $\alpha, \neg\, \alpha \in C$.
- O clauză nevidă $C = \{L_1, \ldots, L_n\}$ (cu $n \in \mathbb{N}^*$ și $L_1, \ldots, L_n$ literali) se zice *satisfiabilă* ddacă enunțul $L_1 \vee \ldots \vee L_n$ corespunzător lui C e satisfiabil.
- O mulțime finită de clauze se zice *satisfiabilă* ddacă enunțul în FNC corespunzător acelei mulțimi de clauze e satisfiabil.

Caracterizarea deducțiilor semantice pe care se bazează rezolvarea interogărilor de către Prolog prin tehnica rezoluției

Remarcă

Fie $n \in \mathbb{N}^*$, iar $\varphi_1, \ldots, \varphi_n$ și φ enunțuri.
Să observăm că are loc echivalența:

$$\models \varphi \text{ ddacă } \neg\varphi \text{ e nesatisfiabil (adică nu are model).}$$

Într–adevăr: $\models \varphi$ (i.e. φ e adevăr semantic) ddacă, pentru orice algebră $\mathcal{A}$ de signatură τ, avem $||\varphi||_{\mathcal{A}} = 1$ ddacă, pentru orice algebră $\mathcal{A}$ de signatură τ, avem $||\neg\varphi||_{\mathcal{A}} = \overline{||\varphi||_{\mathcal{A}}} = \overline{1} = 0$ ddacă $\neg\varphi$ e nesatisfiabil.
Mai mult, ca o consecință a Teoremei Deducției și a observației de mai sus, avem următoarea caracterizare a deducției semantice $\{\varphi_1, \ldots, \varphi_n\} \models \varphi$:

$$\{\varphi_1, \ldots, \varphi_n\} \models \varphi \text{ ddacă } \{\varphi_1 \wedge \ldots \wedge \varphi_n\} \models \varphi$$

$$\text{ddacă } \models (\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \varphi$$

$$\text{ddacă enunțul } \neg[(\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \varphi] \text{ e nesatisfiabil.}$$

Definiție (definim substituțiile pe clauze – la fel putem proceda pentru FNC fără cuantificatori reprezentate ca mulțimi de clauze)

Fie $\sigma : Term(\mathcal{L}_\tau) \to Term(\mathcal{L}_\tau)$ o substituție.
Pentru orice clauză $C = \{L_1, \ldots, L_p\}$, unde $p \in \mathbb{N}^*$, iar $L_1, \ldots, L_p$ sunt literali, definim $\sigma(C) = \{\sigma(L_1), \ldots, \sigma(L_p)\}$.

Remarcă (la fel va fi în cazul FNC fără cuantificatori)

Cu notațiile din definiția anterioară, clauza $\sigma(C) = \{\sigma(L_1), \ldots, \sigma(L_p)\}$ corespunde enunțului $\sigma(L_1) \vee \ldots \vee \sigma(L_p) = \sigma(L_1 \vee \ldots \vee L_p)$ dat de valoarea lui σ în enunțul $L_1 \vee \ldots \vee L_p$ corespunzător clauzei C.

Notație

Fie $n \in \mathbb{N}^*$, $x_1, \ldots, x_n \in Var$, două câte două distincte, și $t_1, \ldots, t_n \in Term(\mathcal{L}_\tau)$.
Notăm cu

$$\{x_1/t_1, \ldots, x_n/t_n\} : Var \to Term(\mathcal{L}_\tau)$$

substituția definită prin:

$$\begin{cases} (\forall\, i \in \overline{1,n})\,(\{x_1/t_1, \ldots, x_n/t_n\}(x_i) = t_i); \\ (\forall\, x \in Var \setminus \{x_1, \ldots, x_n\})\,(\{x_1/t_1, \ldots, x_n/t_n\}(x) = x). \end{cases}$$

Notație

Fie $n, k \in \mathbb{N}^*$, $x_1, \ldots, x_n \in Var$, două câte două distincte, $i_1, \ldots, i_k \in \overline{1, n}$, astfel încât $1 \leq i_1 < i_2 < \ldots < i_k \leq n$, $t_{i_1}, \ldots, t_{i_k} \in Term(\mathcal{L}_\tau)$ și $\varphi(x_1, \ldots, x_n)$ o formulă fără cuantificatori cu $V(\varphi) \subseteq \{x_1, \ldots, x_n\}$.
Atunci notăm cu

$$\varphi(x_1, \ldots, x_{i_1-1}, t_{i_1}, x_{i_1+1}, \ldots, x_{i_2-1}, t_{i_2}, x_{i_2+1}, \ldots, x_{i_k-1}, t_{i_k}, x_{i_k+1}, \ldots, x_n)$$

formula $\{x_{i_1}/t_{i_1}, \ldots, x_{i_k}/t_{i_k}\}(\varphi)$.

Exemplu

Dacă v, x, y, z sunt variabile distincte, s, t, u sunt termeni, ρ este un simbol de relație binară, f e un simbol de operație binară, g e un simbol de operație unară și c este un simbol de constantă, atunci rezultatul aplicării substituției $\{x/s, y/t, z/u\}$ asupra formulei fără cuantificatori:

$$\varphi(v, x, y, z) = \neg\,[\rho(g(x), f(v, y)) \to \neg\,\rho(f(g(y), g(z)), f(v, g(c))]$$

este formula fără cuantificatori: $\{x/s, y/t, z/u\}(\varphi(v, x, y, z)) = \varphi(v, s, t, u) =$

$$\neg\,[\rho(g(s), f(v, t)) \to \neg\,\rho(f(g(t), g(u)), f(v, g(c))].$$

Pentru a aplica rezoluția unui enunț, enunțul trebuie pus într–un anumit tip de FNC, numit **formă Skolem**, despre care se poate demonstra că există pentru orice enunț și satisfiabilă ddacă acel enunț e satisfiabil.
Fie ε un enunț. Folosind proprietățile de mai sus și, eventual, redenumind variabilele pentru a nu avea variabile cuantificate și universal, și existențial (mai multe detalii în SEMINAR), se determină o formă prenex pentru ε:

$$\varepsilon \models\!\models \mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n \varphi(x_1, \dots, x_n),$$

unde $n \in \mathbb{N}^*$, $x_1, \dots, x_n \in Var$, $\mathcal{Q}_1, \dots, \mathcal{Q}_n \in \{\forall, \exists\}$ și $\varphi(x_1, \dots, x_n)$ este o formulă fără cuantificatori. Apoi, folosind proprietățile ①–⑨ de mai sus, folosite în calculul propozițional pentru a pune enunțurile în FNC, se pune $\varphi(x_1, \dots, x_n)$ într–o FNC ψ, astfel obținându–se o altă formă prenex pentru ε:

$$\varepsilon \models\!\models \mathcal{Q}_1 x_1 \dots \mathcal{Q}_n x_n \psi(x_1, \dots, x_n),$$

cu formula fără cuantificatori ψ în FNC.
Pentru fiecare cuantificator existențial $\mathcal{Q}_i = \exists$, cu $i \in \overline{1, n}$, se adaugă la signatura τ câte o operație "fictivă" h_i, numită *funcție Skolem*, de aritate $i - |\{j \in \overline{1, i} \mid \mathcal{Q}_j = \exists\}|$ (astfel că, în cazul în care $\mathcal{Q}_1 = \exists$, h_1 este o constantă), și se înlocuiește fiecare apariție a variabilei x_i în $\psi(x_1, \dots, x_n)$ cu termenul h_i având $V(h_i) = \{x_1, \dots, x_i\} \setminus \{x_j \mid j \in \overline{1, i}\}$, iar cuantificatorii existențiali sunt eliminați din forma prenex anterioară.

Astfel obținem următoarea FNC: $\chi = \forall x_1, \ldots \forall x_{i_1-1} \forall x_{i_1+1} \ldots \forall x_{i_2-1} \forall x_{i_2+1} \ldots \forall x_{i_k-1} \forall x_{i_k+1} \ldots \forall x_n \varphi(x_1, \ldots, x_{i_1-1}, h_{i_1}(x_1, \ldots, x_{i_1-1}), x_{i_1+1}, \ldots, x_{i_2-1}, h_{i_2}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}), x_{i_2+1}, \ldots, x_{i_k-1}, h_{i_k}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2+1}, \ldots, x_{i_k-1}), x_{i_k+1}, \ldots, x_n)$.

Fie $p = n - k$ și (doar pentru a simplifica următoarea scriere) să redenumim variabilele $x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2+1}, \ldots, x_{i_k-1}, x_{i_k+1}, \ldots, x_n$ în $y_1, \ldots, y_p$, respectiv. Dacă $C_1(y_1, \ldots, y_p), \ldots, C_r(y_1, \ldots, y_p)$ sunt clauzele formulei fără cuantificatori în FNC $\varphi(x_1, \ldots, x_{i_1-1}, h_{i_1}(x_1, \ldots, x_{i_1-1}), x_{i_1+1}, \ldots, x_{i_2-1}, h_{i_2}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}), x_{i_2+1}, \ldots, x_{i_k-1}, h_{i_k}(x_1, \ldots, x_{i_1-1}, x_{i_1+1}, \ldots, x_{i_2-1}, x_{i_2+1}, \ldots, x_{i_k-1}), x_{i_k+1}, \ldots, x_n)$, atunci:

$$\chi = \forall y_1 \ldots \forall y_p \, C_1(y_1, \ldots, y_p) \wedge \ldots \wedge C_r(y_1, \ldots, y_p) \models$$

$$(\forall y_1 \ldots \forall y_p \, C_1(y_1, \ldots, y_p)) \wedge \ldots \wedge (\forall y_1 \ldots \forall y_p \, C_r(y_1, \ldots, y_p)),$$

iar această din urmă formulă este satisfiabilă ddacă fiecare dintre formulele $\forall y_1 \ldots \forall y_p \, C_1(y_1, \ldots, y_p)$, ..., $\forall y_1 \ldots \forall y_p \, C_r(y_1, \ldots, y_p)$ este satisfiabilă.

Acum redenumim variabilele $\{y_1, \ldots, y_p\}$ în fiecare dintre clauzele $C_1, \ldots, C_r$ astfel încât mulțimile $V(C_1), \ldots, V(C_r)$ să devină două câte două disjuncte: $V(C_1) = \{z_{11}, \ldots, z_{1p}\}$, ..., $V(C_r) = \{z_{r1}, \ldots, z_{rp}\}$, și, folosind, ca și mai sus, distributivitatea cuantificatorului universal ($\forall$) față de conjuncție ($\wedge$), obținem:

$$\xi \models \gamma = \forall z_{11} \ldots \forall z_{1p} \ldots \forall z_{r1} \ldots \forall z_{rp}(C_1(z_{11}, \ldots, z_{1p}) \wedge \ldots \wedge C_r(z_{r1}, \ldots, z_{rp})).$$

FNC γ se numește *formă Skolem* pentru ε, și este satisfiabilă ddacă ε e satisfiabil. Nu putem spune că $\varepsilon \models\!\models \gamma$, întrucât ε și γ nu sunt satisfăcute de aceleași algebre: la signatura τ trebuie să adăugăm simboluri de operații/constante corespunzătoare funcțiilor Skolem pentru a obține signatura algebrelor în care este evaluată valoarea de adevăr a enunțului γ.

Definiție (**unificare** între **literali**)

Prin *unificare* între **formule atomice** înțelegem unificarea acelor formule privite ca termeni cu operația dominantă de rezultat boolean asociată relației din acele formule atomice, ca în Prolog.
Dacă φ și ψ sunt formule atomice, vom spune că $\neg\varphi$ și $\neg\psi$ *unifică* ddacă φ și ψ unifică. Un *(cel mai general) unificator* pentru $\neg\varphi$ și $\neg\psi$ este un *(cel mai general) unificator* pentru φ și ψ, adică un (cel mai general) unificator pentru termenii obținuți din φ și ψ prin înlocuirea relațiilor din aceste formule atomice cu operațiile de rezultat boolean asociate acestor relații.

Definiție (**clauze triviale**)

Numim *clauză trivială* o clauză care conține doi literali φ și $\neg\psi$, unde φ și ψ sunt formule atomice care **unifică**.

(Rezoluția (regulă de deducție pentru logica clasică a predicatelor))

Pentru orice clauze C și D cu $V(C) \cap V(D) = \emptyset$, dacă φ și ψ sunt formule atomice astfel încât $\varphi, \neg\varphi$ nu unifică, cu niciun literal din C și $\psi, \neg\psi$ nu unifică, cu niciun literal din D, iar $\sigma : Term(\mathcal{L}_\tau) \to Term(\mathcal{L}_\tau)$ este o substituție cu proprietatea că $\sigma(\varphi) = \sigma(\psi)$, atunci:

$$\frac{C \cup \{\varphi\}, D \cup \{\neg\psi\}}{\sigma(C) \cup \sigma(D)}.$$

Definiție (derivări prin rezoluție)

Fie o mulțime finită de clauze $\{D_1, \ldots, D_k\}$.
Dacă $i, j \in \overline{1,k}$ a. î. $i \neq j$ și există două formule atomice φ și ψ astfel încât $\varphi, \neg\varphi$ nu unifică, cu niciun literal din D_i, $\psi, \neg\psi$ nu unifică, cu niciun literal din D_j, iar $\sigma : Term(\mathcal{L}_\tau) \to Term(\mathcal{L}_\tau)$ este o substituție cu proprietatea că $\sigma(\varphi) = \sigma(\psi)$, atunci mulțimea de clauze $R = \{\sigma(D_1), \ldots, \sigma(D_k)\}$ se numește *rezolvent* al mulțimii de clauze $Q = \{D_i \cup \{\varphi\}, D_j \cup \{\neg\psi\}\} \cup \{D_t \mid t \in \overline{1,k} \setminus \{i,j\}\}$.
Deducția $\frac{Q}{R}$ se numește *derivare prin rezoluție* a mulțimii Q. Vom numi orice succesiune de derivări prin rezoluție tot *derivare prin rezoluție*. O succesiune de derivări prin rezoluție care începe cu o FNC/mulțime de clauze μ și se termină cu o FNC/mulțime de clauze ν se numește *derivare prin rezoluție a lui ν din μ*.

(Algoritmul Davis–Putnam (abreviat *DP*) pentru Logica Predicatelor)

INPUT: $m \in \mathbb{N}^*$ și o mulțime $S = \{C_1, \ldots, C_m\}$ de clauze netriviale având $V(C) \cap V(D) = \emptyset$ pentru orice clauze $C, D \in S$ cu $C \neq D$;

$i := 1$; $S_1 := S$;

PASUL 1: considerăm o formulă atomică α_i care apare în măcar una dintre clauzele din S_i;

$T_i^0 := \{j \in \overline{1,m} \mid C_j \in S_i, (\exists\, \beta_i)(\neg\, \beta_{i,j} \in C_j$ și $\beta_{i,j}$ unifică, cu $\alpha_i)\}$;

$T_i^1 := \{j \in \overline{1,m} \mid C_j \in S_i, (\exists\, \beta_i)(\beta_{i,j} \in C_j$ și $\beta_{i,j}$ unifică, cu $\alpha_i)\}$;

$T_i := T_i^0 \cup T_i^1$;

fie σ_i un unificator pentru elementele mulțimii $\{\beta_{i,j} \mid j \in T_i\}$;

PASUL 2: dacă $T_i^0 \neq \emptyset$ și $T_i^1 \neq \emptyset$,

atunci $U_i := \{\sigma_i(C_j \setminus \{\neg\, \beta_{i,j}\}) \cup \sigma_i(C_k \setminus \{\beta_{i,k}\}) \mid j \in T_i^0, k \in T_i^1\}$;

altfel $U_i := \emptyset$;

PASUL 3: $S_{i+1} := (S_i \setminus \{C_j \mid j \in T_i\}) \cup U_i$;

$S_{i+1} := S_{i+1} \setminus \{C \in S_{i+1} \mid (\exists\, \beta \in V)\,(\beta$ formulă atomică și $\beta, \neg\, \beta \in C)\}$ (eliminăm din S_{i+1} clauzele triviale);

PASUL 4: dacă $S_{i+1} = \emptyset$,

atunci OUTPUT: S *e satisfiabilă*;

altfel, dacă $\square \in S_{i+1}$,

atunci OUTPUT: S *nu e satisfiabilă*;

altfel $i := i + 1$ și mergi la PASUL 1.

În derivările prin rezoluție, inclusiv în unificările efectuate în cadrul acestor derivări, deci și în aplicarea algoritmului DP, funcțiile Skolem sunt tratate ca orice funcții. Rezultatul algoritmului DP e influențat de FORMA SKOLEM obținută pentru enunțul despre care dorim să aflăm dacă e satisfiabil sau nu. A se vedea un exemplu mai jos.

Pentru **tipurile de enunțuri** pentru care **Prologul** aplică tehnica rezoluției, algoritmul DP este **corect și complet**: se termină într–un număr finit de pași și dă rezultatul corect: determină dacă un astfel de enunț e satisfiabil sau nu.
În general însă, spre deosebire de cazul logicii propozițiilor, rezultatul aplicării algoritmului DP unui enunț cu sau fără cuantificatori existențiali nu este neapărat corect.

Revenind la o **formă Skolem** pentru un enunț ε: ce rol au funcțiile Skolem?

Dacă $x, y \in Var$, iar φ este o formulă cu $FV(\varphi) \subseteq \{x, y\}$, de ce avem: $\exists x \forall y \varphi \vDash \forall y \exists x \varphi$, dar nu și $\forall y \exists x \varphi \nvDash \exists x \forall y \varphi$? Pentru că, în enunțul $\forall y \exists x \varphi$, o valoare a a lui x care satisface acest enunț poate să depindă de valoarea b a lui y pentru care perechea (a, b) satisface acest enunț.

Exemplu

- $(\exists x \in \mathbb{R})(\forall y \in \mathbb{R})(x + y = 0)$: **fals**: nu există un astfel de x comun tuturor valorilor lui $y \in \mathbb{R}$;
- $(\forall y \in \mathbb{R})(\exists x \in \mathbb{R})(x + y = 0)$: **adevărat**;

Exemplu (continuare)

- $(\exists\, x \in \mathbb{N})\,(\forall\, y \in \mathbb{Z})\,(x \in \{y, -y\})$: **fals**: nu există un astfel de x comun tuturor valorilor lui $y \in \mathbb{Z}$;
- $(\forall\, y \in \mathbb{Z})\,(\exists\, x \in \mathbb{N})\,(x \in \{y, -y\})$: **adevărat**.

Fie $f : \mathbb{R} \to \mathbb{R}$, având cel puțin 3 valori distincte, i.e. cardinalul imaginii sale $f(\mathbb{R})$ mai mare sau egal cu 3.

- $(\exists\, x \in [0, +\infty))\,(\forall\, y \in \mathbb{R})\,(x \in \{f(y), -f(y)\})$: **fals**: nu există un astfel de x comun tuturor valorilor lui $y \in \mathbb{R}$, pentru că alegerea lui f ne asigură de faptul că există $y_0, y_1 \in \mathbb{R}$ cu proprietatea că $f(y_1) \notin \{f(y_0), -f(y_0)\}$, așadar perechea $(x_1 = f(y_1), y_0)$ nu satisface proprietatea $x_1 \in \{f(y_0), -f(y_0)\}$;
- $(\forall\, y \in \mathbb{R})\,(\exists\, x \in [0, +\infty))\,(x \in \{f(y), -f(y)\})$: **adevărat**.

O **formă Skolem** pentru enunțul $(\forall\, y)\,(\exists\, x)\,(x + y = 0)$ este: $(\forall\, y)\,(g(y) + y = 0)$, unde g este o funcție Skolem unară, iar acest enunț în formă Skolem, pentru domeniul valorilor $\mathbb{R}$ pentru ambele variabile, x și y, semnifică faptul că există o funcție $g : \mathbb{R} \to \mathbb{R}$ astfel încât $(\forall\, y \in \mathbb{R})\,(g(y) + y = 0)$.
O **formă Skolem** pentru enunțul $(\forall\, y)\,(\exists\, x)\,(x \in \{y, -y\})$ este: $(\forall\, y)\,(h(y) \in \{y, -y\})$, unde h este o funcție Skolem unară, iar acest enunț în formă Skolem, pentru domeniile valorilor $\mathbb{N}$, respectiv $\mathbb{Z}$ pentru variabilele x, respectiv y, semnifică faptul că există o funcție $h : \mathbb{Z} \to \mathbb{N}$ astfel încât $(\forall\, y \in \mathbb{Z})\,(h(y) \in \{y, -y\})$.

O **formă Skolem** pentru enunțul $(\forall\, y)\,(\exists\, x)\,(x \in \{f(y), -f(y)\})$ este: $(\forall\, y)\,(k(y) \in \{f(y), -f(y)\})$, unde k este o funcție Skolem unară, iar acest enunț în formă Skolem, pentru pentru domeniile valorilor $[0, +\infty)$, respectiv $\mathbb{R}$ pentru variabilele x, respectiv y, semnifică faptul că există o funcție $k : \mathbb{R} \to [0, +\infty)$ astfel încât $(\forall\, y \in \mathbb{R})\,(k(y) \in \{f(y), -f(y)\})$. Într–adevăr, acest enunț este satisfăcut de funcția $k : \mathbb{R} \to [0, +\infty)$, definită prin:

$$(\forall\, y \in \mathbb{R})\left(k(y) = |f(y)| = \begin{cases} f(y), \text{ dacă } f(y) \geq 0, \\ -f(y), \text{ altfel.} \end{cases}\right).$$

Deci faptul că funcția k nu coincide nici cu f, nici cu $-$ nu influențează satisfiabilitatea enunțului anterior.

Exemplu

Să considerăm un limbaj de ordinul I conținând două simboluri de relații unare p și q. Fie x, y variabile distincte. Considerăm enunțurile:

- $\varphi = \forall\, x\, p(x) \wedge \exists\, y\, q(y)$;
- $\psi = \exists\, x\, [(p(x) \vee q(x)) \wedge \forall\, y\, ((p(x) \vee \neg\, q(y)) \wedge \neg\, p(x))]$.

$\varphi \not\models \forall\, x \exists\, y\, (p(x) \wedge q(y))$, dar avem și
$\varphi \not\models \exists\, y\, q(y) \wedge \forall\, x\, p(x) \not\models \exists\, y \forall\, x\, (q(y) \wedge p(x)) \not\models \exists\, y \forall\, x\, (p(x) \wedge q(y))$.

Așadar două forme Skolem diferite pentru enunțul φ sunt:

- $\forall x\,(p(x) \wedge q(c))$, unde c este o constantă Skolem; această formă Skolem corespunde mulțimii de clauze $\{\{p(x)\}, \{q(c)\}\}$, care nu are derivări prin rezoluție, deci algoritmul DP determină satisfiabilitatea acestei mulțimi de clauze;
- $\forall x\,(p(x) \wedge q(f(x)))$, unde f este o funcție Skolem unară; această formă Skolem corespunde mulțimii de clauze $\{\{p(x)\}, \{q(f(x))\}\}$, care nu are derivări prin rezoluție, deci algoritmul DP determină și satisfiabilitatea acestei mulțimi de clauze.

La fel ca mai sus:

- $\psi \models\!\!\!\models \exists x \forall y\,[(p(x) \vee q(x)) \wedge (p(x) \vee \neg\, q(y)) \wedge \neg\, p(x)]$, având drept formă Skolem enunțul $\forall y\,[(p(c) \vee q(c)) \wedge (p(c) \vee \neg\, q(y)) \wedge \neg\, p(c)]$, unde c este o constantă Skolem, corespunzător mulțimii de clauze $\{\{p(c), q(c)\}, \{p(c), \neg\, q(y)\}, \{\neg\, p(c)\}\}$;
- $\psi \models\!\!\!\models \forall y \exists x\,[(p(x) \vee q(x)) \wedge (p(x) \vee \neg\, q(y)) \wedge \neg\, p(x)]$, având drept formă Skolem enunțul $\forall y\,[(p(f(y)) \vee q(f(y))) \wedge (p(f(y)) \vee \neg\, q(y)) \wedge \neg\, p(f(y))]$, unde f este o funcție Skolem unară, corespunzător mulțimii de clauze $\{\{p(f(v)), q(f(v))\}, \{p(f(y)), \neg\, q(y)\}, \{\neg\, p(f(z))\}\}$.

Pentru prima dintre mulțimile de clauze de mai sus avem următoarea derivare prin rezoluție a clauzei vide, prin urmare algoritmul DP determină nesatisfiabilitatea acestei mulțimi de clauze:

$$\frac{\{p(c), q(c)\},\{p(c), \neg q(y)\} \text{ (cu unificatorul } \{y/c\}),\{\neg p(c)\}}{\frac{\{p(c)\},\{\neg p(c)\}}{\square}}$$

Pentru a doua dintre mulțimile de clauze de mai sus avem următoarele derivări prin rezoluție, dintre care niciuna nu ajunge la clauza vidă, prin urmare algoritmul DP determină satisfiabilitatea acestei mulțimi de clauze:

$$\frac{\{p(f(v)), q(f(v))\},\{p(f(y)), \neg q(y)\} \text{ (cu unificatorul } \{y/f(v)\}),\{\neg p(f(z))\}}{\frac{\{p(f(v)), p(f(f(v)))\},\{\neg p(f(z))\} \text{ (cu unificatorul } \{z/v\})}{\{p(f(f(v)))\}}}$$

$$\frac{\{p(f(v)), q(f(v))\},\{p(f(y)), \neg q(y)\} \text{ (cu unificatorul } \{y/f(v)\}),\{\neg p(f(z))\}}{\frac{\{p(f(v)), p(f(f(v)))\},\{\neg p(f(z))\} \text{ (cu unificatorul } \{z/f(v)\})}{\{p(f(v))\}}}$$

$$\frac{\{p(f(v)), q(f(v))\},\{p(f(y)), \neg q(y)\},\{\neg p(f(z))\} \text{ (cu unificatorul } \{z/v\})}{\frac{\{q(f(v))\},\{p(f(y)), \neg q(y)\} \text{ (cu unificatorul } \{y/f(v)\})}{\{p(f(f(v)))\}}}$$

Vom vedea că Prologul aplică rezoluția numai pentru FNC prenex (FNC fără cuantificatori precedate numai de cuantificatori universali) obținute fără Skolemizare (i.e. fără înlocuirea unor variabile cu funcții Skolem), mai precis pentru un tip particular de astfel de FNC prenex: **Prologul** întotdeauna aplică **regula rezoluției** pentru o **clauză Horn** și o **clauză scop**: definițiile mai jos.

1. Să reluăm noțiunile din Logica Clasică a Predicatelor care stau la baza funcționării Prologului
2. Cu ce fel de signaturi și structuri algebrice de acele signaturi și cu ce tipuri/cazuri particulare de formule lucrează limbajul Prolog?
3. Modele pentru formule, satisfiabilitate în Logica Clasică a Predicatelor
4. Rezoluția în Logica Clasică a Predicatelor – regula de deducție care stă la baza funcționării limbajului Prolog
5. Rezoluția în Prolog

Amintesc că:

- *formulele atomice* sunt formulele fără cuantificatori și fără conectori logici;
- *literalii* sunt formulele atomice și negațiile formulelor atomice;
- *clauzele* sunt disjuncțiile finite (nu neapărat nevide) de literali; clauzele se identifică, cu mulțimile literalilor pe care îi conțin; dacă sunt nevide, clauzele sunt formule fără cuantificatori;
- *FNC prenex* sunt enunțurile date de conjuncții finite și nevide de clauze nevide precedate de cuantificatori universali; FNC prenex se identifică, cu mulțimile clauzelor pe care le conțin.

Definiție

O *clauză Horn* sau *clauză definită* este o clauză care conține exact o formulă atomică nenegată (implicit e o clauză nevidă), iar ceilalți literali pe care îi conține sunt formule atomice negate.
O *clauză scop* este o clauză nevidă care conține numai formule atomice negate.

Amintesc cu ce tipuri de formule (mai precis enunțuri) lucrează Prologul:

- **faptele** corespund unor enunțuri de tipul următor: $\forall x_1 \forall x_2 \ldots \forall x_n \alpha$, unde $n \in \mathbb{N}^*$ și α este o formulă atomică;
- **regulile** corespund unor enunțuri de tipul:$\forall y_1 \forall y_2 \ldots \forall y_k (\varphi \to \beta)$, unde $k \in \mathbb{N}^*$, φ este o formulă fără cuantificatori, iar β este o formulă atomică;
- **interogările** corespund unor formule de tipul $\exists z_1 \exists z_2 \ldots \exists z_p \psi$, unde $p \in \mathbb{N}^*$ și ψ este o formulă fără cuantificatori.

Pentru cele ce urmează, cu notațiile de mai sus, vom presupune că φ și ψ sunt conjuncții finite (desigur, nevide) de formule atomice.
Dacă în φ sau ψ apare negația unei formule atomice, de exemplu $\rho = \neg R(t_1, \ldots, t_m)$, unde $m \in \mathbb{N}^*$, $t_1, \ldots, t_m$ sunt termeni și R este un simbol de relație m–ară, atunci transformăm pe ρ într–o formulă atomică introducând în signatură simbolul de relație m–ară $\neg R$ astfel încât, în orice algebră $\mathcal{A}$ de signatura definită de respectivul program în Prolog care satisface enunțurile date de faptele și regulile din programul respectiv, dacă A este mulțimea de elemente ale lui $\mathcal{A}$, atunci $(\neg R)^{\mathcal{A}} = A^m \setminus R^{\mathcal{A}} = \{(a_1, \ldots, a_m) \in A^m \mid (a_1, \ldots, a_m) \notin R\}$.
Se adaptează teoria următoare și pentru cazul în care, în formule precum φ, ψ apar disjuncții de formule atomice, nu numai conjuncții. De exemplu, o regulă de forma:

concluzie :- *posibilitatea*1; *posibilitatea*2.

se poate înlocui cu următoarele două reguli:

concluzie :- *posibilitatea*1.
concluzie :- *posibilitatea*1.

O regulă de forma următoare, de exemplu:

concluzia :- *premisa*1, (*posibilitate*1; *posibilitate*2; *posibilitate*3), *premisa*2.

se poate înlocui cu următoarele trei reguli:

concluzia :- *premisa*1, *posibilitate*1, *premisa*2.
concluzia :- *premisa*1, *posibilitate*2, *premisa*2.
concluzia :- *premisa*1, *posibilitate*3, *premisa*2.

Un scop de forma următoare, de exemplu:

?- *subscop*1, (*varianta*1; *varianta*2), *subscop*2, *subscop*3.

e satisfăcut ddacă e satisfăcut măcar unul dintre scopurile:

?- *subscop*1, *varianta*1, *subscop*2, *subscop*3.
?- *subscop*1, *varianta*2, *subscop*2, *subscop*3.

iar soluțiile primei interogări de mai sus sunt date de reuniunea soluțiilor celor două interogări anterioare.
Să presupunem, așadar, că: $\varphi = \gamma_1 \wedge \ldots \wedge \gamma_q$ și $\psi = \delta_1 \wedge \ldots \wedge \delta_s$, unde $q, s \in \mathbb{N}^*$, iar $\gamma_1, \ldots, \gamma_q, \delta_1, \ldots, \delta_s$ sunt formule atomice. Atunci:

$$\forall\, y_1 \forall\, y_2 \ldots \forall\, y_k\, (\varphi \to \beta) = \forall\, y_1 \forall\, y_2 \ldots \forall\, y_k\, ((\gamma_1 \wedge \ldots \wedge \gamma_q) \to \beta) \models\!\models$$

$$\forall\, y_1 \forall\, y_2 \ldots \forall\, y_k\, (\neg\, (\gamma_1 \wedge \ldots \wedge \gamma_q) \vee \beta) \models\!\models \forall\, y_1 \forall\, y_2 \ldots \forall\, y_k\, (\neg\, \gamma_1 \vee \ldots \vee \neg\, \gamma_q \vee \beta),$$

iar $\neg\, \gamma_1 \vee \ldots \vee \neg\, \gamma_q \vee \beta$ este o **clauză Horn**.
Cum α este o formulă atomică, în particular α este o **clauză Horn**.

Așadar **clauzele** din **baza de cunoștințe** constau în enunțuri date de **clauze Horn** precedate de cuantificatori universali.

Amintesc că, dacă baza de cunoștințe este formată din enunțurile $\varepsilon_1, \ldots, \varepsilon_r$, cu $r \in \mathbb{N}^*$, iar ε este un enunț corespunzător unei interogări, atunci cerința adresată Prologului prin această interogare este de a determina dacă:

$$\{\varepsilon_1, \ldots, \varepsilon_r\} \models \varepsilon,$$

sau, echivalent:

$$\{\varepsilon_1 \wedge \ldots \wedge \varepsilon_r\} \models \varepsilon,$$

sau, echivalent, conform Teoremei Deducției Semantice:

$$\models (\varepsilon_1 \wedge \ldots \wedge \varepsilon_r) \rightarrow \varepsilon,$$

fapt echivalent cu:

$$\begin{array}{c} \text{enunțul } \neg\left[(\varepsilon_1 \wedge \ldots \wedge \varepsilon_r) \rightarrow \varepsilon\right] \text{ e nesatisfiabil,} \\ \text{adică enunțul } \varepsilon_1 \wedge \ldots \wedge \varepsilon_r \wedge \neg\, \varepsilon \text{ e nesatisfiabil,} \end{array}$$

întrucât

$$\neg\left[(\varepsilon_1 \wedge \ldots \wedge \varepsilon_r) \rightarrow \varepsilon\right] \models\!\!\!\models \neg\left[\neg\,(\varepsilon_1 \wedge \ldots \wedge \varepsilon_r) \vee \varepsilon\right] \models\!\!\!\models \varepsilon_1 \wedge \ldots \wedge \varepsilon_r \wedge \neg\, \varepsilon,$$

iar acest din urmă enunț poate fi transformat într–o **FNC prenex** prin mutarea cuantificatorilor în față (a se vedea mai jos).

Definiție

O derivare prin rezoluție a clauzei vide $\square$ din FNC prenex $\varepsilon_1 \wedge \ldots \wedge \varepsilon_r \wedge \neg\, \varepsilon$ se numește *SLD–respingere* a clauzei ε din mulțimea de clauze $\{\varepsilon_1, \ldots, \varepsilon_r\}$.

(Principala regulă de deducție utilizată de Prolog)

Pentru a satisface scopul ε, Prologul caută o SLD–respingere a clauzei ε din mulțimea de clauze corespunzătoare bazei de cunoștințe.
Acest caz al rezoluției aplicat de Prolog este numit *rezoluție SLD* (*Selective Linear Definite Clause Resolution*: rezoluție liniară selectivă pe clauze definite).

Baza de cunoștințe se identifică, cu, conjuncția enunțurilor corespunzătoare faptelor și regulilor ($\varepsilon_1 \wedge \ldots \wedge \varepsilon_r$ cu notația de mai sus), care, prin mutarea cuantificatorilor în față, devine o **FNC prenex** constând din **clauze Horn**.

Într–adevăr, de exemplu, considerând faptul și regula de mai sus, dacă mulțimea de variabile $\{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_k\} = \{v_1, \ldots, v_j\}$, avem:

$$\forall\, x_1\ \ldots \forall\, x_n\, \alpha \wedge \forall\, y_1\ \ldots \forall\, y_k\, (\varphi \to \beta) \models\!\models \forall\, v_1\ \ldots \forall\, v_j\, [\alpha \wedge (\varphi \to \beta)] =$$

$$\forall\, v_1\ \ldots \forall\, v_j\, [\alpha \wedge ((\gamma_1 \wedge \ldots \wedge \gamma_q) \to \beta)] \models\!\models \forall\, v_1\ \ldots \forall\, v_j\, [\alpha \wedge (\neg\, \gamma_1 \vee \ldots \vee \neg\, \gamma_q \vee \beta)].$$

La fel pentru mai multe fapte și/sau reguli.
De exemplu, dacă baza de cunoștințe este formată din faptul și regula de mai sus, iar $\{v_1, \ldots, v_j\} \cup \{z_1, \ldots, z_p\} = \{w_1, \ldots, w_l\}$ atunci interogarea de mai sus presupune a verifica nesatisfiabilitatea enunțului:

$$\forall x_1 \ldots \forall x_n \alpha \wedge \forall y_1 \ldots \forall y_k (\varphi \rightarrow \beta) \wedge \neg (\exists z_1 \exists z_2 \ldots \exists z_p \psi) \models\!\models$$

$$\forall x_1 \ldots \forall x_n \alpha \wedge \forall y_1 \ldots \forall y_k (\varphi \rightarrow \beta) \wedge \forall z_1 \ldots \forall z_p \neg \psi \models\!\models$$

$$\forall w_1 \ldots \forall w_l [\alpha \wedge (\varphi \rightarrow \beta) \wedge \neg \psi] \models\!\models \forall w_1 \ldots \forall w_l [\alpha \wedge (\varphi \rightarrow \beta) \wedge \neg (\delta_1 \wedge \ldots \wedge \delta_s)] \models\!\models$$

$$\forall w_1 \ldots \forall w_l (\alpha \wedge (\neg \gamma_1 \vee \ldots \vee \neg \gamma_q \vee \beta) \wedge (\neg \delta_1 \vee \ldots \vee \neg \delta_s)).$$

Observăm că negația interogării de mai sus, anume enunțul $\forall z_1 \ldots \forall z_p (\neg \delta_1 \vee \ldots \vee \neg \delta_s)$ este un enunț dat de o **clauză scop** precedată de cuantificatori universali.
Tehnica descrisă mai sus presupune adăugarea acestei clauze scop la mulțimea de clauze Horn corespunzătoare bazei de cunoștințe și aplicarea rezoluției pentru mulțimea de clauze astfel obținută:

- dacă această mulțime de clauze este **nesatisfiabilă**, atunci răspunsul la interogarea de mai sus este **true**: enunțul $\exists z_1 \ldots \exists z_p (\delta_1 \wedge \ldots \wedge \delta_s)$ se deduce semantic din baza de cunoștințe, deci scopul dat de acest enunț este satisfăcut;
- dacă această mulțime de clauze este **satisfiabilă**, atunci răspunsul la interogarea de mai sus este **false**: enunțul $\exists z_1 \ldots \exists z_p (\delta_1 \wedge \ldots \wedge \delta_s)$ nu este consecință semantică a bazei de cunoștințe, deci scopul dat de acest enunț eșuează.

Exemplu

Considerăm baza de cunoștințe:

$a.\ \ b.$

c :- $a, b.$

și interogarea: ?- c.

Regula din programul de mai sus corespunde formulei fără variabile:

$$(a \wedge b) \to c \models\!\!\models \neg (a \wedge b) \vee c \models\!\!\models \neg a \vee \neg b \vee c,$$

așadar întreaga bază de cunoștinte corespunde FNC:

$$a \wedge b \wedge [(a \wedge b) \to c] \models\!\!\models a \wedge b \wedge (\neg a \vee \neg b \vee c),$$

având forma clauzală: $\{\{a\}, \{b\}, \{\neg a, \neg b, c\}\}$.

Să vedem dacă, adăugând la această mulțime de clauze clauza $\{\neg c\}$, obținem o mulțime de clauze satisfiabilă:

$$\dfrac{\dfrac{\dfrac{\{a\}, \{b\}, \{\neg a, \neg b, \cancel{c}\}, \{\cancel{\neg c}\}}{\{a\}, \{\cancel{b}\}, \{\neg a, \cancel{\neg b}\}}}{\{\cancel{a}\}, \{\cancel{\neg a}\}}}{\square}$$

Am ajuns la **clauza vidă**, așadar mulțimea de clauze $\{\{a\}, \{b\}, \{\neg a, \neg b, c\}, \{\neg c\}\}$ e **nesatisfiabilă**, prin urmare:

$$\{a, b, (a \wedge b) \to c\} \models c,$$

Să ne amintim ce sunt **substituțiile**

adică scopul c e satisfăcut, deci răspunsul la interogarea ?- c este **true**.

Cum, în exemplul anterior, nu avem variabile, ne situăm în cazul **rezoluției propoziționale**. Cazul general al **rezoluției pentru logica predicatelor** presupune determinarea unui **unificator** la fiecare aplicare a **regulii rezoluției** și aplicarea acestui unificator la clauzele rămase după aplicarea acelui pas de rezoluție.

Definiție (o *substituție* este o funcție de la variabile la termeni)

O *substituție* este o funcție $\sigma : Var \to Term(\mathcal{L}_\tau)$.

Definiție și notație (prelungirea unei substituții la termeni)

Fie $\sigma : Var \to Term(\mathcal{L}_\tau)$. Următoarea extindere a lui σ la întreaga mulțime a termenilor se numește tot *substituție* (și, de obicei, se notează tot cu σ): $\widetilde{\sigma} : Term(\mathcal{L}_\tau) \to Term(\mathcal{L}_\tau)$, definită, recursiv, astfel:

- pentru orice $v \in Var$, $\widetilde{\sigma}(v) = \sigma(v)$;
- pentru orice (simbol de) constantă c, $\widetilde{\sigma}(c) = c$;
- pentru orice $h \in \mathbb{N}^*$, orice (simbol de) operație h–ară f și orice $t_1, \ldots, t_h \in Term(\mathcal{L}_\tau)$, $\widetilde{\sigma}(f(t_1, \ldots, t_h)) = f(\widetilde{\sigma}(t_1), \ldots, \widetilde{\sigma}(t_h))$.

Un **unificator** pentru o mulțime de termeni este o substituție care are aceeași valoare în acei termeni, i.e., aplicată acelor termeni, are ca rezultat termeni identici

Definiție

O *substituție* $\sigma : Var \to Term(\mathcal{L}_\tau)$ se numește *unificator* pentru $h \in \mathbb{N}^*$ termeni $t_1, \ldots, t_h \in Term(\mathcal{L}_\tau)$ ddacă $\widetilde{\sigma}(t_1) = \ldots = \widetilde{\sigma}(t_h)$.

Algoritmul de unificare aplicat unor termeni $t_1, \ldots, t_h$ $(h \in \mathbb{N}^*)$ întoarce un **cel mai general unificator** al termenilor $t_1, \ldots, t_h$, adică un unificator μ al acestor termeni cu proprietatea că orice unificator al termenilor $t_1, \ldots, t_h$ este dat de compunerea unei substituții cu μ.

Amintesc și această:

Notație

Dacă $h \in \mathbb{N}^*$, $v_1, \ldots, v_h \in Var$ și $t_1, \ldots, t_h \in Term(\mathcal{L}_\tau)$, atunci substituția pe care o notăm $\{v_1/t_1, \ldots, v_h/t_h\} : Var \to Term(\mathcal{L}_\tau)$ este definită prin:

$$\begin{cases} \{v_1/t_1, \ldots, v_h/t_h\}(v_i) = t_i, & \text{pentru orice } i \in \overline{1,h}, \\ \{v_1/t_1, \ldots, v_h/t_h\}(v) = v, & \text{pentru orice } v \in Var \setminus \{v_1, \ldots, v_h\}. \end{cases}$$

Cu notațiile de mai sus, avem, în Prolog:

- **faptul** $\forall x_1 \forall x_2 \ldots \forall x_n \alpha$:

α. % poate fi privit ca o **regulă** cu **membrul stâng** α și **premisa vidă**

- **regula** $\forall y_1 \forall y_2 \ldots \forall y_k [(\gamma_1 \wedge \ldots \wedge \gamma_q) \rightarrow \beta]$:

β :- $\gamma_1, \ldots, \gamma_q$. % cu **membrul stâng** β și **premisa** $\gamma_1, \ldots, \gamma_q$

- **interogarea** $\exists z_1 \exists z_2 \ldots \exists z_p (\delta_1 \wedge \ldots \wedge \delta_s)$:

?- $\delta_1, \ldots, \delta_s$.

unde α, β, $\gamma_1, \ldots, \gamma_q$, $\delta_1, \ldots, \delta_s$ sunt **formule atomice**, adică relații aplicate la termeni, astfel că, în Prolog, pot fi considerate **termeni** cu operatorii dominanți dați de **operațiile de rezultat boolean asociate acelor relații** (a se vedea mai sus).

(Cum rezolvă Prologul interogarea ?- $\delta_1, \ldots, \delta_s$.?)

Prologul rezolvă o interogare ?- $\delta_1, \ldots, \delta_s$ (adică satisface **scopul compus** $\delta_1, \ldots, \delta_s$) satisfăcând, PE RÂND, **subscopurile** $\delta_1, \ldots, \delta_s$, DE LA STÂNGA LA DREAPTA.

Să detaliem modul în care Prologul răspunde interogărilor

Cum satisface Prologul un **subscop** δ_i, cu $i \in \overline{1,s}$? Considerând **faptele** și **regulile** din **baza de cunoștințe** DE SUS ÎN JOS și unificând (prin aplicarea **algoritmului de unificare**), cu un **cel mai general unificator** μ, pe δ_i cu **formula atomică** dintr–un:

- **fapt** α, caz în care următorul scop compus este:
  ?- $\mu(\delta_1), \ldots, \mu(\delta_{i-1}), \mu(\delta_{i+1}), \ldots, \mu(\delta_s)$.
  adică se scoate din scopul compus $\delta_1, \ldots, \delta_s$ subscopul δ_i, iar la celelalte subscopuri se aplică unificatorul μ; acesta este un **caz particular** al aplicării unei **reguli**, pentru **premisa vidă**;

sau din

- **membrul stâng** β **al unei reguli** β :- $\gamma_1, \ldots, \gamma_q$, caz în care următorul scop compus este:
  ?- $\mu(\delta_1), \ldots, \mu(\delta_{i-1}), \mu(\gamma_1), \ldots, \mu(\gamma_q), \mu(\delta_{i+1}), \ldots, \mu(\delta_s)$.
  adică, în scopul compus $\delta_1, \ldots, \delta_s$, se înlocuiește subscopul δ_i, cu **membrul drept al regulii** β :- $\gamma_1, \ldots, \gamma_q$, iar la subscopurile din lista astfel obținută se aplică unificatorul μ.

Procedeul de mai sus este aplicat în manieră **backtracking**. Mai precis, Prologul aplică:

Algoritmul **backward chaining**

Mai jos, **cazul particular** $q = 0$ este cazul aplicării unui **fapt**.
Notăm cu $id_{Var} : Var \to Term(\mathcal{L}_\tau)$ incluziunea canonică: pentru fiecare $v \in Var$, $id_{Var}(v) = v$.

(apelat din programul principal cu următorii parametri: **backwardChaining**(număr inițial de subscopuri date de formule atomice; lista acestor subscopuri inițiale; substituția id_{Var})

backwardChaining(INTEGER s; formule atomice $\delta_1, \ldots, \delta_s$; unificator μ)
  FOR $i = 1$ TO s
    $\delta'_i := \mu(\delta_i)$
  FOR $i = 1$ TO s
    FOR toate clauzele β :- $\gamma_1, \ldots, \gamma_q$ din baza de cunoștințe
      IF ν este un cel mai general unificator pentru δ'_i și β THEN
        $s' := s - 1 + q$
        IF $s' = 0$ THEN RETURN **true**, WRITE soluție: unificatorul $\nu \circ \mu$
        ELSE **backwardChaining**(s'; $\nu(\delta'_1), \ldots, \nu(\delta'_{i-1})$, $\nu(\gamma_1), \ldots, \nu(\gamma_q)$, $\nu(\delta'_{i+1}), \ldots, \nu(\delta'_s)$; $\nu \circ \mu$)
RETURN **false**

Observăm că algoritmul de mai sus constă într–un **backtracking recursiv**.

Dezavantaj: căutarea de mai sus este de tip **depth first**, așadar, de exemplu, dacă inversăm ultimele două clauze din următoarea bază de cunoștințe:

$arc(N, M)$:- M is $N + 1$.
$drum(X, Y)$:- $arc(X, Y)$.
$drum(X, Y)$:- $arc(X, Z)$, $drum(Z, Y)$.

atunci, de exemplu, la interogarea: ?- $drum(1, 3)$, recursia nu se termină.
Cu notațiile de mai sus, amintesc că:

enunțul $\forall\, x_1 \ldots \forall\, x_n\, \alpha$ are forma clauzală $\{\alpha\}$,

$$\forall\, y_1 \ldots \forall\, y_k\, [(\gamma_1 \wedge \ldots \wedge \gamma_q) \to \beta] \models\!\!\mid \forall\, y_1 \ldots \forall\, y_k\, [\neg\, (\gamma_1 \wedge \ldots \wedge \gamma_q) \vee \beta]$$
$$\models\!\!\mid \forall\, y_1 \ldots \forall\, y_k\, (\neg\, \gamma_1 \vee \ldots \vee \neg\, \gamma_q \vee \beta),$$

având forma clauzală $\{\neg\, \gamma_1, \ldots, \neg\, \gamma_q, \beta\}$,

$$\neg\, [\exists\, z_1\, \exists\, z_2 \ldots \exists\, z_p\, (\delta_1 \wedge \ldots \wedge \delta_s)] \models\!\!\mid \forall\, z_1\, \forall\, z_2 \ldots \forall\, z_p\, \neg\, (\delta_1 \wedge \ldots \wedge \delta_s)$$
$$\models\!\!\mid \forall\, z_1\, \forall\, z_2 \ldots \forall\, z_p\, (\neg\, \delta_1 \vee \ldots \vee \neg\, \delta_s),$$

având forma clauzală $\{\neg\, \delta_1, \ldots, \neg\, \delta_s\}$.

Pasul de aplicare a unui **fapt** sau a unei **reguli** descris mai sus din backtracking–ul urmat de Prolog pentru a răspunde interogării ?- $\delta_1, \ldots, \delta_s$ constă în **regula**

aplicată clauzei corespunzătoare negației acestei interogări și clauzei corespunzătoare faptului, respectiv regulii de mai sus, în modul următor: pentru a satisface **subscopul** δ_i pentru un $i \in \overline{1,s}$:

- cu **faptul** α:

$$\frac{\{\neg\,\delta_1, \ldots, \neg\,\delta_{i-1}, \neg\not{\delta_i}, \neg\,\delta_{i+1}, \ldots, \neg\,\delta_s\},\ \{\not{\alpha}\ \}\ \text{(cu unificatorul } \mu)}{\{\neg\,\mu(\delta_1), \ldots, \neg\,\mu(\delta_{i-1}), \neg\,\mu(\delta_{i+1}), \ldots, \neg\,\mu(\delta_s)\}}$$

- prin **regula** β :- $\gamma_1, \ldots, \gamma_q$:

$$\frac{\{\neg\,\delta_1, \ldots, \neg\,\delta_{i-1}, \neg\not{\delta_i}, \neg\,\delta_{i+1}, \ldots, \neg\,\delta_s\},\ \{\neg\,\gamma_1, \ldots, \neg\,\gamma_q, \not{\beta}\ \}\ \text{(cu unificatorul } \mu)}{\{\neg\,\mu(\delta_1), \ldots, \neg\,\mu(\delta_{i-1}), \neg\,\mu(\gamma_1), \ldots, \neg\,\mu(\gamma_q), \neg\,\mu(\delta_{i+1}), \ldots, \neg\,\mu(\delta_s)\}}$$

Într–adevăr, următorul scop compus pe care trebuie să îl satisfacă Prologul după aplicarea **faptului**, respectiv **regulii** de mai sus este:

$$\mu(\delta_1), \ldots, \mu(\delta_{i-1}), \mu(\delta_{i+1}), \ldots, \mu(\delta_s),$$

respectiv

$$\mu(\delta_1), \ldots, \mu(\delta_{i-1}), \mu(\gamma_1), \ldots, \mu(\gamma_q), \mu(\delta_{i+1}), \ldots, \mu(\delta_s).$$

Toate derivările posibile prin rezoluție respectând strategia de mai sus pot fi reprezentate printr–un **arbore de derivare prin rezoluție SLD**:

- **nodurile** acestui arbore sunt etichetate cu **negațiile scopurilor compuse curente**;
- **rădăcina** arborelui de derivare este etichetată cu **negația scopului compus** din care constă interogarea;
- pentru fiecare nod al arborelui și fiecare clauză din baza de cunoștințe care, împreună cu eticheta acelui nod, are o derivare prin rezoluție, se introduce în arbore un fiu al acelui nod, etichetat cu **negația noului scop compus** obținută prin acea derivare prin rezoluție;
- muchiile arborelui se etichetează cu clauzele din baza de cunoștințe pentru care s–a aplicat regula rezoluției și unificatorii folosiți la fiecare astfel de derivare prin rezoluție.

Conform celor de mai sus:

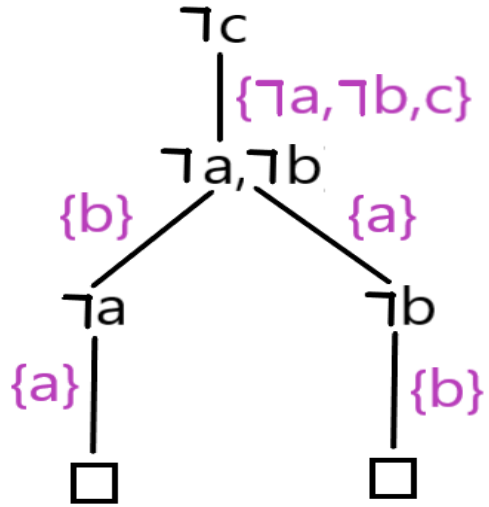
- dacă măcar una dintre frunze este etichetată cu $\square$, atunci răspunsul la acea interogare este **true**, iar soluția interogării este compunerea unificatorilor de pe drumul de la rădăcină la acea frunză;
- dacă niciuna dintre frunze nu este etichetată cu $\square$, atunci răspunsul la acea interogare este **false**.

Exemplu

Pentru exemplul de mai sus în care am aplicat rezoluția propozițională, avem arborele de derivare prin rezoluție SLD:

a. {a}
b. {b}
c :- a,b. {¬a,¬b,c}

?- c.



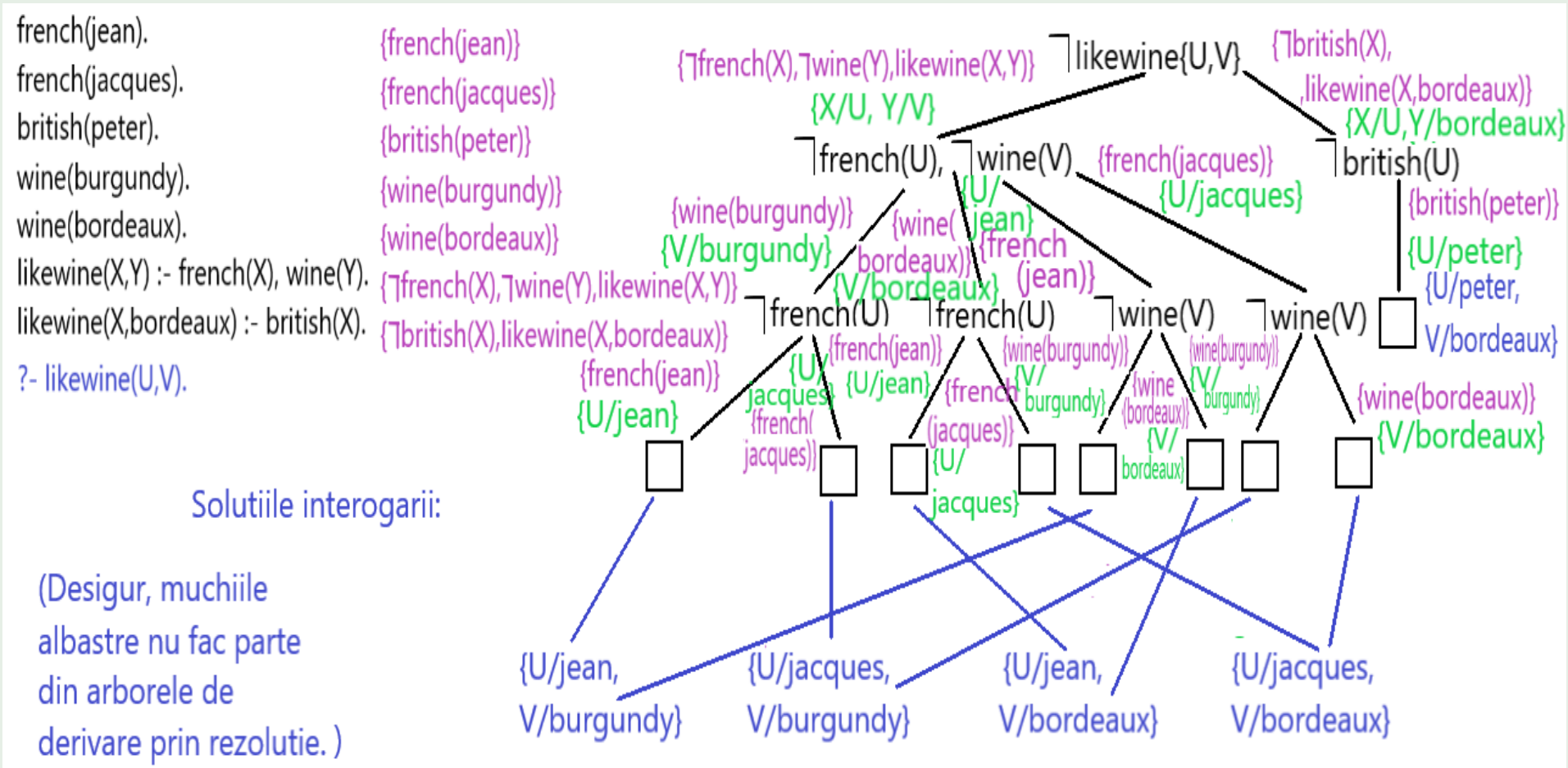
Pentru simplitate, putem omite acoladele din etichetele nodurilor arborelui de derivare, ca mai sus.

(Rezoluția SLD pentru o **clauză scop** și o **mulțime de clauze Horn** este **corectă și completă)**

Rezoluția SLD aplicată ca mai sus, pornind de la o **clauză scop** și aplicând, la fiecare pas, **regula rezoluției** pentru **clauza scop curentă** și o **clauză Horn**, întotdeauna se termină într–un număr finit de pași și dă răspunsul corect: există o astfel de derivare prin rezoluție a clauzei vide □ ddacă negația acelei clauze scop se deduce semantic din acea mulțime de clauze Horn.

În plus, **arborele de derivare prin rezoluție** dă **toate soluțiile** unei interogări, sub forma compunerilor unificatorilor pe fiecare drum de la rădăcina arborelui la o frunză etichetată cu □, de fapt de substituțiile date prin valorile în variabilele din interogare ale acestor compuneri compuse la stânga cu substituții arbitrare.

Exemplu (nu neapărat soluții distincte pentru diferitele frunze etichetate cu □)



Cum obține Prologul soluțiile unei interogări?

Să presupunem că FNC prenex corespunzătoare bazei de cunoștințe este γ.
Fie $n \in \mathbb{N}^*$, și să considerăm n variabile distincte $X_1, \ldots, X_n$.
Atunci, pentru o interogare constând dintr–un scop $\varphi(X_1, \ldots, X_n)$, așadar corespunzând formulei $\exists\, X_1 \ldots \exists\, X_n\, \varphi(X_1, \ldots, X_n)$:

• **toate soluțiile** acestei interogări constau în substituțiile
$\sigma = \{X_1/t_1, \ldots, X_n/t_n\}$, cu $t_1, \ldots, t_n$ termeni, având proprietatea că:

$$\{\gamma\} \vDash \forall\, V_1 \ldots \forall\, V_j\, \sigma(\varphi(X_1, \ldots, X_n)) = \forall\, V_1 \ldots \exists\, V_j\, \varphi(t_1, \ldots, t_n),$$

unde $\{V_1, \ldots, V_j\} = V(t_1) \cup \ldots \cup V(t_n)$;

• **soluțiile** *date acestei interogări de către* **Prolog** constau în substituții $\mu_1, \ldots \mu_k$ ($k \in \mathbb{N}$) cu proprietatea că orice soluție a acestei interogări este o compunere între o substituție (arbitrară) și una dintre substituțiile $\mu_1, \ldots \mu_k$, mai precis cu proprietatea că mulțimea tuturor soluțiilor acestei interogări este:
$\{\, \{X_1/(\sigma \circ \mu_i)(X_1), \ldots, X_n/(\sigma \circ \mu_i)(X_n)\} \mid i \in \overline{1, k}, \sigma : Var \to Term(\mathcal{L}_\tau)\}$.
Amintesc că fiecare soluție μ_i este dată de compunerea $\nu_i = \nu_{i,1} \circ \nu_{i,2} \ldots \circ \nu_{i,p}$ a unificatorilor de pe drumul de la rădăcină la câte o frunză etichetată cu clauza vidă $\square$ în arborele de derivare prin rezoluție SLD, mai precis de substituția $\mu_i = \{X_1/\nu_i(X_1), \ldots, X_n/\nu_i(X_n)\}$. Putem spune, simplu, că soluțiile date de Prolog sunt valorile substituțiilor $\mu_1, \ldots \mu_k$ în variabilele $X_1, \ldots, X_n$:
$\mu_i(X_1) = \nu_i(X_1), \ldots, \mu_i(X_n) = \nu_i(X_n)$, pentru fiecare $i \in \overline{1, k}$.