Programare Logică – Seminariile II şi III

Claudia MUREŞAN
Universitatea din Bucureşti, Facultatea de Matematică şi Informatică
c.muresan@yahoo.com, cmuresan@fmi.unibuc.ro

2019–2020, Semestrul II

Exerciţiul 1. Fie V mulţimea variabilelor propoziţionale, iar E mulţimea enunţurilor logicii propoziţionale clasice, şi $\alpha, \beta, \gamma, \delta \in E$. Demonstraţi că, în logica propoziţională clasică:

(1) mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \neg\beta \to \gamma,\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ e nesatisfiabilă;

(2) dacă $\vdash \alpha \vee \beta \vee \gamma$, atunci mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ e nesatisfiabilă;

(3) dacă $\alpha, \beta, \gamma, \delta \in V$, atunci mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ e satisfiabilă; să se rezolve această cerinţă:

- demonstrând că această mulţime are un model;
- prin tehnica rezoluţiei;

(4) în cazul în care $\alpha, \beta, \gamma, \delta \in V$, să se rezolve cerinţa de la punctul (1) şi prin tehnica rezoluţiei.

La punctele (3) şi (4), pentru a pune elementele mulţimii sau conjuncţia lor în FNC, să se procedeze prin două metode:

- folosind proprietăţile echivalenţei semantice derivate din proprietăţi booleene;
- folosind tabele semantice.

Rezolvare: Fie $h : V \to \mathcal{L}_2$ o interpretare arbitrară. Cu notaţia din curs, $\widetilde{h} : E \to \mathcal{L}_2$ va fi unica prelungire a lui h la E care transformă conectorii logici în operaţii booleene. (A se revedea primul seminar.)

Vom folosi faptul că h, şi deci $\widetilde{h}$, poate lua orice valori booleene în elementele lui V, dar $\widetilde{h}$ nu poate lua orice valori booleene în elementele lui $E \setminus V$. (În orice tautologie va lua valoarea 1, iar, de exemplu, în negaţia oricărei tautologii va lua valoarea 0.)

(1) Presupunem prin absurd că $h \vDash \{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \neg\beta \to \gamma,\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$, adică:
$h \vDash \alpha \to \beta$, adică $1 = \widetilde{h}(\alpha \to \beta) = \widetilde{h}(\alpha) \to \widetilde{h}(\beta)$, aşadar $\widetilde{h}(\alpha) \leq \widetilde{h}(\beta)$;
$h \vDash \beta \to (\gamma \wedge \delta)$, i.e. $1 = \widetilde{h}(\beta \to (\gamma \wedge \delta)) = \widetilde{h}(\beta) \to (\widetilde{h}(\gamma) \wedge \widetilde{h}(\delta))$, prin urmare $\widetilde{h}(\beta) \leq \widetilde{h}(\gamma) \wedge \widetilde{h}(\delta)$;
$h \vDash \neg\beta \to \gamma$, i.e. $1 = \widetilde{h}(\neg\beta \to \gamma) = \overline{\widetilde{h}(\beta)} \to \widetilde{h}(\gamma)$, deci $\overline{\widetilde{h}(\beta)} \leq \widetilde{h}(\gamma)$;
$h \vDash \gamma \to \alpha$, i.e. $1 = \widetilde{h}(\gamma \to \alpha) = \widetilde{h}(\gamma) \to \widetilde{h}(\alpha)$, aşadar $\widetilde{h}(\gamma) \leq \widetilde{h}(\alpha)$;
$h \vDash \delta \to \neg\alpha$, i.e. $1 = \widetilde{h}(\delta \to \neg\alpha) = \widetilde{h}(\delta) \to \overline{\widetilde{h}(\alpha)}$, prin urmare $\widetilde{h}(\delta) \leq \overline{\widetilde{h}(\alpha)}$.

În consecinţă, $\widetilde{h}(\gamma) \leq \widetilde{h}(\alpha) \leq \widetilde{h}(\beta) \leq \widetilde{h}(\gamma) \wedge \widetilde{h}(\delta) \leq \widetilde{h}(\delta) \leq \overline{\widetilde{h}(\alpha)}$, deci $\widetilde{h}(\alpha) \leq \overline{\widetilde{h}(\alpha)}$. Cum $\widetilde{h}(\alpha) \in \{0, 1\}$, rezultă că $\widetilde{h}(\alpha) = 0$, aşadar $\widetilde{h}(\gamma) \leq 0$, deci $\widetilde{h}(\gamma) = 0$, prin urmare $\widetilde{h}(\gamma) \wedge \widetilde{h}(\delta) = 0 \wedge \widetilde{h}(\delta) = 0$, aşadar $\widetilde{h}(\beta) \leq 0$, deci $\widetilde{h}(\beta) = 0$.

Dar $\overline{\widetilde{h}(\beta)} \leq \widetilde{h}(\gamma)$, deci $\overline{0} \leq 0$, adică $1 \leq 0$; contradicţie. Prin urmare $h \nvDash \{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \neg\beta \to \gamma,\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$. Cum interpretarea h este arbitrară, rezultă că nicio interpretare nu satisface mulţimea de enunţuri $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \neg\beta \to \gamma,\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$, adică această mulţime este nesatisfiabilă.

(2) Presupunem că $\vdash \alpha \vee \beta \vee \gamma$. Atunci $\vDash \alpha \vee \beta \vee \gamma$ conform **Teoremei de Completitudine (TC)**, ceea ce înseamnă că orice interpretare satisface enunţul $\alpha \vee \beta \vee \gamma$. Aşadar $h \vDash \alpha \vee \beta \vee \gamma$, adică $1 = \widetilde{h}(\alpha \vee \beta \vee \gamma) = \widetilde{h}(\alpha) \vee \widetilde{h}(\beta) \vee \widetilde{h}(\gamma)$, prin urmare: $\widetilde{h}(\alpha) = 1$ sau $\widetilde{h}(\beta) = 1$ sau $\widetilde{h}(\gamma) = 1$.

Presupunem prin absurd că $h \vDash \{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$, deci, folosind şi calculele de la punctul (1):

$h \vDash \alpha \to \beta$, aşadar $\widetilde{h}(\alpha) \leq \widetilde{h}(\beta)$;
$h \vDash \beta \to (\gamma \wedge \delta)$, prin urmare $\widetilde{h}(\beta) \leq \widetilde{h}(\gamma) \wedge \widetilde{h}(\delta)$;
$h \vDash \gamma \to \alpha$, aşadar $\widetilde{h}(\gamma) \leq \widetilde{h}(\alpha)$;
$h \vDash \delta \to \neg\alpha$, prin urmare $\widetilde{h}(\delta) \leq \overline{\widetilde{h}(\alpha)}$.

Şi aici rezultă că $\widetilde{h}(\gamma) \leq \widetilde{h}(\alpha) \leq \widetilde{h}(\beta) \leq \widetilde{h}(\gamma) \wedge \widetilde{h}(\delta) \leq \widetilde{h}(\delta) \leq \overline{\widetilde{h}(\alpha)}$, deci $\widetilde{h}(\alpha) \leq \overline{\widetilde{h}(\alpha)}$, aşadar $\widetilde{h}(\alpha) = 0$, prin urmare $\widetilde{h}(\gamma) = 0$. Conform celor de mai sus, rezultă că $\widetilde{h}(\beta) = 1$. Dar atunci $1 = \widetilde{h}(\beta) \leq \widetilde{h}(\gamma) \wedge \widetilde{h}(\delta) = 0 \wedge \widetilde{h}(\delta) = 0$, deci $1 \leq 0$; contradicţie. Aşadar $h \nvDash \{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$.

Cum interpretarea h este arbitrară, rezultă că mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ e nesatisfiabilă.

(3) Presupunem că α, β, γ şi δ sunt variabile propoziţionale.

Prima metodă: Cum $\alpha, \beta, \gamma, \delta \in V$, rezultă că există (chiar o infinitate – a se revedea primul seminar – de) interpretări $g : V \to \mathcal{L}_2$ care iau în $\alpha, \beta, \gamma, \delta$ valorile: $g(\alpha) = g(\beta) = g(\gamma) = g(\delta) = 0$.

Cu notaţia din curs, rezultă că orice astfel de interpretare satisface:
$\widetilde{g}(\alpha \to \beta) = \widetilde{g}(\alpha) \to \widetilde{g}(\beta) = 0 \to 0 = 1$, aşadar $g \vDash \alpha \to \beta$;
$\widetilde{g}(\beta \to (\gamma \wedge \delta)) = \widetilde{g}(\beta) \to (\widetilde{g}(\gamma) \wedge \widetilde{h}(\delta)) = 0 \to (0 \wedge 0) = 0 \to 0 = 1$, prin urmare $g \vDash \beta \to (\gamma \wedge \delta)$;
$\widetilde{g}(\gamma \to \alpha) = \widetilde{g}(\gamma) \to \widetilde{g}(\alpha) = 0 \to 0 = 1$, aşadar $g \vDash \gamma \to \alpha$;
$\widetilde{g}(\delta \to \neg\alpha) = \widetilde{g}(\delta) \to \overline{\widetilde{g}(\alpha)} = 0 \to \overline{0} = 0 \to 1 = 1$, prin urmare $g \vDash \delta \to \neg\alpha$.

Aşadar $g \vDash \{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$, prin urmare această mulţime de enunţuri e satisfiabilă.

Metoda rezoluţiei: Amintesc definiţia echivalenţei semantice: $\sim = \{(\varphi, \psi) \mid \varphi, \psi \in E, \vdash \varphi \leftrightarrow \psi\} \in \mathrm{Eq}(E)$ (cu notaţia din curs pentru mulţimea relaţiilor de echivalenţă pe o mulţime; aici, pe mulţimea E a enunţurilor logicii propoziţionale clasice). Conform **TC**, rezultă că, pentru orice enunţuri φ, ψ, avem: $\varphi \sim \psi$ ddacă $\vDash \varphi \leftrightarrow \psi$ ddacă, oricare ar fi interpretarea $g : V \to \mathcal{L}_2$, are loc $1 = \widetilde{g}(\varphi \leftrightarrow \psi) = \widetilde{g}(\varphi) \leftrightarrow \widetilde{g}(\psi)$, deci $\widetilde{g}(\varphi) = \widetilde{g}(\psi)$.

Pentru a aplica tehnica rezoluţiei, trebuie să punem fiecare element al mulţimii $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ în **formă normală conjunctivă (FNC)**, adică trebuie să găsim enunţuri φ, ψ, χ, ξ în FNC astfel încât: $\varphi \sim \alpha \to \beta$, $\psi \sim \beta \to (\gamma \wedge \delta)$, $\chi \sim \gamma \to \alpha$ şi $\xi \sim \delta \to \neg\alpha$.

Prima metodă pentru punerea acestor enunţuri în FNC:
$\alpha \to \beta \sim \neg\alpha \vee \beta$;
$\beta \to (\gamma \wedge \delta) \sim \neg\beta \vee (\gamma \wedge \delta) \sim (\neg\beta \vee \gamma) \wedge (\neg\beta \vee \delta)$;
$\gamma \to \alpha \sim \neg\gamma \vee \alpha$;
$\delta \to \neg\alpha \sim \neg\delta \vee \neg\alpha$.

Cum $\alpha, \beta, \gamma, \delta \in V$, rezultă că enunţurile $\neg\alpha \vee \beta$, $(\neg\beta \vee \gamma) \wedge (\neg\beta \vee \delta)$, $\neg\gamma \vee \alpha$ şi $\neg\delta \vee \neg\alpha$ sunt în FNC.

Aşadar mulţimea de enunţuri $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ este echivalentă cu enunţul în FNC:
$\varepsilon = (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \gamma) \wedge (\neg\beta \vee \delta) \wedge (\neg\gamma \vee \alpha) \wedge (\neg\delta \vee \neg\alpha)$,
adică această mulţime e satisfiabilă ddacă ε e satisfiabil. Considerăm **forma clauzală** a lui ε: enunţul ε este echivalent cu mulţimea de clauze: $\{\{\neg\alpha, \beta\}, \{\neg\beta, \gamma\}, \{\neg\beta, \delta\}, \{\neg\gamma, \alpha\}, \{\neg\delta, \neg\alpha\}\}$.

Pentru a conchide că această mulţime de clauze e **satisfiabilă**, trebuie să efectuăm **toate derivările posibile prin rezoluţie** ale acestei mulţimi şi să constatăm că în niciuna dintre aceste derivări nu apare clauza vidă $\square$. O altă posibilitate, echivalentă cu a efectua toate derivările prin rezoluţie, este să aplicăm **algoritmul Davis–Putnam**. Vom proceda prin această a doua metodă, indicând la fiecare pas variabila propoziţională care va fi eliminată din clauzele curente.

$$\frac{\{\neg\alpha, \beta\}, \{\neg\beta, \gamma\}, \{\neg\beta, \delta\}, \{\neg\gamma, \alpha\}, \{\neg\delta, \neg\alpha\} \quad (\text{ alegem } \delta)}{\{\neg\alpha, \beta\}, \{\neg\beta, \gamma\}, \{\neg\beta, \neg\alpha\}, \{\neg\gamma, \alpha\} \quad (\text{ alegem } \gamma)}$$
$$\frac{\{\neg\alpha, \beta\}, \{\neg\beta, \alpha\}, \{\neg\beta, \neg\alpha\} \quad (\text{ alegem } \beta)}{\{\neg\alpha, \alpha\}, \{\neg\alpha\} \quad (\text{ eliminăm clauza trivială } \{\neg\alpha, \alpha\})}$$
$$\frac{\{\neg\alpha\}}{\emptyset}$$

Nu am obţinut **clauza vidă** $\square$, aşadar enunţul ε e **satisfiabil**, deci mulţimea de enunţuri $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ e **satisfiabilă**.

A DOUA METODĂ: vom pune conjuncţia $\varphi = (\alpha \to \beta) \wedge (\beta \to (\gamma \wedge \delta)) \wedge (\gamma \to \alpha) \wedge (\delta \to \neg\alpha)$ a enunţurilor din această mulţime în FNC folosind un tabel semantic.

Desigur, cu notaţia de mai sus, $\varepsilon \sim \varphi$, dar ε nu este singura FNC echivalentă semantic cu φ.

$h(\alpha)$	$h(\beta)$	$h(\gamma)$	$h(\delta)$	$\widetilde{h}(\alpha \to \beta)$	$\widetilde{h}(\beta \to (\gamma \wedge \delta))$	$\widetilde{h}(\gamma \to \alpha)$	$\widetilde{h}(\delta \to \neg\alpha)$	$\widetilde{h}(\varphi)$	
0	0	0	0	1	1	1	1	1	
0	0	0	1	1	1	1	1	1	
0	0	1	0	1	1	0	1	0	$\alpha \vee \beta \vee \neg\gamma \vee \delta$
0	0	1	1	1	1	0	1	0	$\alpha \vee \beta \vee \neg\gamma \vee \neg\delta$
0	1	0	0	1	0	1	1	0	$\alpha \vee \neg\beta \vee \gamma \vee \delta$
0	1	0	1	1	0	1	1	0	$\alpha \vee \neg\beta \vee \gamma \vee \neg\delta$
0	1	1	0	1	0	0	1	0	$\alpha \vee \neg\beta \vee \neg\gamma \vee \delta$
0	1	1	1	1	1	0	1	0	$\alpha \vee \neg\beta \vee \neg\gamma \vee \neg\delta$
1	0	0	0	0	1	1	1	0	$\neg\alpha \vee \beta \vee \gamma \vee \delta$
1	0	0	1	0	1	1	0	0	$\neg\alpha \vee \beta \vee \gamma \vee \neg\delta$
1	0	1	0	0	1	1	1	0	$\neg\alpha \vee \beta \vee \neg\gamma \vee \delta$
1	0	1	1	0	1	1	0	0	$\neg\alpha \vee \beta \vee \neg\gamma \vee \neg\delta$
1	1	0	0	1	0	1	1	0	$\neg\alpha \vee \neg\beta \vee \gamma \vee \delta$
1	1	0	1	1	0	1	0	0	$\neg\alpha \vee \neg\beta \vee \gamma \vee \neg\delta$
1	1	1	0	1	0	1	1	0	$\neg\alpha \vee \neg\beta \vee \neg\gamma \vee \delta$
1	1	1	1	1	1	1	0	1	

Tabelul semantic de mai sus ne dă următoarea FNC, care poate fi redusă folosind următoarea proprietate booleană: pentru orice $\zeta, \xi \in E$, $(\zeta \vee \xi) \wedge (\zeta \vee \neg\xi) \sim \zeta \vee (\xi \wedge \neg\xi) \sim \zeta$:

$\varphi \sim (\alpha \vee \beta \vee \neg\gamma \vee \delta) \wedge (\alpha \vee \beta \vee \neg\gamma \vee \neg\delta) \wedge (\alpha \vee \neg\beta \vee \gamma \vee \delta) \wedge (\alpha \vee \neg\beta \vee \gamma \vee \neg\delta) \wedge (\alpha \vee \neg\beta \vee \neg\gamma \vee \delta) \wedge (\alpha \vee \neg\beta \vee \neg\gamma \vee \neg\delta) \wedge (\neg\alpha \vee \beta \vee \gamma \vee \delta) \wedge (\neg\alpha \vee \beta \vee \gamma \vee \neg\delta) \wedge (\neg\alpha \vee \beta \vee \neg\gamma \vee \neg\delta) \wedge (\neg\alpha \vee \neg\beta \vee \gamma \vee \delta) \wedge (\neg\alpha \vee \neg\beta \vee \gamma \vee \neg\delta) \wedge (\neg\alpha \vee \neg\beta \vee \neg\gamma \vee \delta) \sim (\alpha \vee \beta \vee \neg\gamma) \wedge (\alpha \vee \neg\beta) \wedge (\neg\alpha \vee \beta) \wedge (\neg\alpha \vee \neg\beta \vee \gamma) \wedge (\neg\alpha \vee \neg\beta \vee \neg\gamma \vee \delta)$.

Desigur, având tabelul semantic de mai sus, nu mai avem nevoie să efectuăm derivări prin rezoluţie, pentru că acest tabel arată că: $h \vDash \varphi$ ddacă $(h(a), h(b), h(c), h(d)) \in \{(0,0,0,0), (0,0,0,1), (1,1,1,1)\}$, aşadar φ e satisfiabil, deci mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ e **satisfiabilă**, având ca modele toate interpretările cu valorile de mai sus în a, b, c şi d (şi valori arbitrare în orice $v \in V \setminus \{a, b, c, d\}$, deci o infinitate de modele).

Să punem în evidenţă acest fapt observat şi în primul seminar:

Observaţia 1. Cum toate enunţurile sunt de lungime finită (i.e. cuvinte finite peste alfabetul logicii propoziţionale), aşadar fiecare enunţ conţine doar un număr finit de variabile propoziţionale, în timp ce mulţimea V a variabilelor propoziţionale este infinită, iar valoarea unei interpretări într–un enunţ nu depinde decât de variabilele care apar în acel enunţ, rezultă că orice **enunţ satisfiabil**, şi implicit orice **mulţime finită de enunţuri satisfiabilă** are **o infinitate de modele**.

(4) Aici e suficient să găsim **o singură derivare prin rezoluţie** în care apare **clauza vidă** $\square$ pentru a conchide că mulţimea e **nesatisfiabilă**.

Desigur, având tabelul anterior, e suficient să observăm că $(h(a), h(b), h(c), h(d)) \in \{(0,0,0,0), (0,0,0,1), (1,1,1,1)\}$ implică $h \nvDash \neg\beta \to \gamma$, aşadar nicio interpretare nu satisface mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \neg\beta \to \gamma,\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$. Dar să aplicăm tehnica rezoluţiei, conform cerinţei de la acest punct.

Cum $\neg\beta \to \gamma \sim \beta \vee \gamma$, conform rezolvării punctului (3) rezultă că o FNC echivalentă semantic cu mulţimea $\{\alpha \to \beta,\ \beta \to (\gamma \wedge \delta),\ \neg\beta \to \gamma,\ \gamma \to \alpha,\ \delta \to \neg\alpha\}$ este $\psi = (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \gamma) \wedge (\neg\beta \vee \delta) \wedge (\beta \vee \gamma) \wedge (\neg\gamma \vee \alpha) \wedge (\neg\delta \vee \neg\alpha)$, având forma clauzală: $\{\{\neg\alpha, \beta\}, \{\neg\beta, \gamma\}, \{\neg\beta, \delta\}, \{\beta, \gamma\}, \{\neg\gamma, \alpha\}, \{\neg\delta, \neg\alpha\}\}$.

În următoarele derivări prin rezoluţie pentru mulţimea de clauze de mai sus, voi marca literalii folosiţi la fiecare pas. Dacă luăm mereu prima pereche de literali utilizabilă din stânga, atunci obţinem următoarea derivare:

$$\{\neg\alpha,\cancel{\beta}\},\{\neg\cancel{\beta},\gamma\},\{\neg\beta,\delta\},\{\beta,\gamma\},\{\neg\gamma,\alpha\},\{\neg\delta,\neg\alpha\}$$
$$\{\neg\alpha,\gamma\},\{\neg\cancel{\beta},\delta\},\{\cancel{\beta},\gamma\},\{\neg\gamma,\alpha\},\{\neg\delta,\neg\alpha\}$$
$$\{\neg\alpha,\cancel{\gamma}\},\{\delta,\gamma\},\{\neg\cancel{\gamma},\alpha\},\{\neg\delta,\neg\alpha\}$$
$$\{\neg\alpha,\alpha\},\{\delta,\gamma\},\{\neg\delta,\neg\alpha\} \quad \text{eliminăm clauza trivială } \{\neg\alpha,\alpha\}$$
$$\{\cancel{\delta},\gamma\},\{\neg\cancel{\delta},\neg\alpha\}$$
$$\{\gamma,\neg\alpha\}$$

Nu am întâlnit clauza vidă în această derivare, dar nu putem conchide că mulţimea e satisfiabilă până nu efectuăm **toate derivările prin rezoluţie** (sau aplicăm **algoritmul Davis–Putnam**).

Iată o derivare prin rezoluţie în care apare **clauza vidă** $\square$:

$$\{\neg\alpha,\cancel{\beta}\},\{\neg\beta,\gamma\},\{\neg\cancel{\beta},\delta\},\{\beta,\gamma\},\{\neg\gamma,\alpha\},\{\neg\delta,\neg\alpha\}$$
$$\{\neg\alpha,\delta\},\{\neg\cancel{\beta},\gamma\},\{\cancel{\beta},\gamma\},\{\neg\gamma,\alpha\},\{\neg\delta,\neg\alpha\}$$
$$\{\neg\alpha,\cancel{\delta}\},\{\gamma\},\{\neg\gamma,\alpha\},\{\neg\cancel{\delta},\neg\alpha\}$$
$$\{\neg\cancel{\alpha}\},\{\gamma\},\{\neg\gamma,\cancel{\alpha}\}$$
$$\{\neg\cancel{\gamma}\},\{\cancel{\gamma}\}$$
$$\square$$

Derivarea de mai sus este suficientă pentru a concluziona că mulţimea de enunţuri $\{\alpha\to\beta,\ \beta\to(\gamma\wedge\delta),\ \neg\beta\to\gamma,\ \gamma\to\alpha,\ \delta\to\neg\alpha\}$ e **nesatisfiabilă**, dar, ATENŢIE: **tehnica rezoluţiei** nu poate fi aplicată mai sus decât dacă $\alpha,\beta,\gamma,\delta\in V$! Observaţi tehnica rezoluţiei aplicată la punctul (3) pentru cazul $\alpha,\beta,\gamma,\delta\in V$, care nu acoperă şi cazul de la punctul (2)!

Satisfacerea interogărilor de către Prolog

Amintesc din primul curs că, pentru a satisface un scop, Prologul încearcă să unifice predicatul sau predicatele care compun acel scop cu predicatul dintr–un fapt sau din membrul stâng al unei reguli, şi:

- dacă reuşeste o unificare cu un fapt, atunci conchide satisfacerea sau nesatisfacerea acelui scop;
- dacă reuşeste o unificare cu membrul stâng al unei reguli, atunci va trece la un următor scop, constând din membrul drept al acelei reguli, în care variabilele care au fost unificate cu constante sau funcţii cu argumente pentru a realiza unificarea cu membrul drept al regulii sunt înlocuite cu acele valori rezultate prin unificare.

Exemplul 1. Fie programul, ale cărui clauze le numerotăm după cum urmează ((1) şi (7) sunt **fapte**, iar (2), (3), (4), (5) şi (6) sunt **reguli**):

(1) a.

(2) q :- s.

(3) s :- t.

(4) t :- d.

(5) t :- a.

(6) t :- b.

(7) b.

Predicatele a, b, d, q, s, t sunt zeroare, adică nu au argumente (din punct de vedere logic, sunt propoziţii: predicate fără variabile).

Predicatul d apare doar în membrul drept al regulii (4), aşadar trebuie să adăugăm la programul de mai sus următoarea directivă, care specifică faptul că predicatul zeroar d nu este întotdeauna definit (numărul 0 de mai jos este aritatea lui d, i. e. numărul de argumente al predicatului d):

:- dynamic $d/0$.

Să întrebăm Prologul:

?- t.

Pentru a răspunde la interogarea de mai sus, Prologul consideră fiecare dintre cele 7 clauze:

(1) false. (t nu unifică cu a)

(2) false. (şi unificarea $t = q$ eşuează; scriem prescurtat: $t \backslash = q$)

(3) false. ($t \backslash = s$)

(4) $t = t$: t unifică cu membrul stâng, t, al regulii (4), aşa că noul scop, subscop al scopului iniţial t, este:

?- d.

(1) false. ($d \backslash = a$)

(2) false. ($d \backslash = q$)

(3) false. ($d \backslash = s$)

(4), (5), (6) false. ($d \backslash = t$)

(7) false. ($d \backslash = b$)

false.

(5) $t = t$; noul scop (subscop al scopului iniţial t):

?- a.

(1) true. ($a = a$ în faptul (1))

true.

true: acesta e primul răspuns la interogarea ?- t.

Dacă mai cerem variante de satisfacere a predicatului t (";" în versiunea desktop şi "Next" în versiunea online a SWI Prolog), atunci Prologul continuă cu parcurgerea clauzelor, folosind algoritmul backtracking pe care îl are încorporat:

(6) $t = t$; noul subscop:

?- b.

(1) false. ($b \backslash = a$)

(2) false. ($b \backslash = q$)

(3) false. ($b \backslash = s$)

(4), (5), (6) false. ($b \backslash = t$)

(7) true. ($b = b$ în faptul (7))

true.

true.

Dacă mai cerem o soluţie:

false. (bactrackingul s–a încheiat; toate clauzele au fost parcurse).

Dacă dăm interogarea:

?- q.

atunci, la fel ca mai sus, se trece la subscopurile:

?- s.

?- t.

?- d.

false.

?- a.

true.

true.

true.
true.
Dacă mai cerem o soluţie:
?- b.
true.
true.
true.
Dacă mai cerem o soluţie:
false.

Exemplul 2. Fie programul:

(1) a :- b.

(2) a :- c.

(3) c :- d.

(4) c.

(5) d :- e.

(6) e.

Predicatul b apare doar în membrul drept al regulii (1), aşadar, ca în Exemplul 1, trebuie să adăugăm directiva:
:- dynamic $b/0$.
Dăm interogarea:
?- a.

(1) $a = a$.
?- b.
(1), (2) false. ($b \backslash = a$)
(3), (4) false. ($b \backslash = c$)
(5) false. ($b \backslash = d$)
(6) false. ($b \backslash = e$)
false.
(2) $a = a$.
?- c.
(1), (2) false. ($c \backslash = a$)
(3) $c = c$.
?- d.
(1), (2) false. ($d \backslash = a$)
(3), (4) false. ($d \backslash = c$)
(5) $d = d$.
?- e.
(1), (2) false. ($e \backslash = a$)
(3), (4) false. ($e \backslash = c$)
(5) false. ($e \backslash = d$)
(6) true. ($e = e$)
true.
true.

true.
true.
Dacă mai cerem soluţii: bactrackingul s–a întrerupt în timpul satisfacerii scopului d, aşadar continuă cu:
(6) false. ($d \backslash = e$)
false.
?- c.
(4) true. ($c = c$)
true.
true.
Dacă mai cerem soluţii:
(5) false. ($c \backslash = d$)
(6) false. ($c \backslash = e$)
false.
(3), (4) false. ($a \backslash = c$)
(5) false. ($a \backslash = d$)
(6) false. ($a \backslash = e$)
false.

Exemplul 3. Fie programul:

(1) $s(X)$:- $t(X)$.

(2) $t(1)$.

(3) $t(2)$.

(4) $t(Y)$:- $a(Y)$.

(5) $a(21)$.

Clauzele de mai sus (**fapte** şi **reguli**) formează o **bază de cunoştinţe**. În aceste clauze, variabilele sunt cuantificate universal. Astfel:

- clauza (1) corespunde enunţului: $(\forall X)\,(t(X) \to s(X))$;
- clauza (4) corespunde enunţului: $(\forall Y)\,(a(Y) \to t(Y))$.

În interogări, variabilele sunt cuantificate existenţial. Astfel, interogarea:
?- $s(X)$.
cere Prologului să satisfacă enunţul: $(\exists X)\,(s(X))$, adică să găsească o valoare v a variabilei X pentru care $s(v)$ este adevărat. La fel pentru scopuri constând în predicate cu mai multe variabile.

Pentru unificările următoare, amintesc că domeniul de valabilitate al unui nume de variabilă în Prolog este limitat la clauza în care apare acel nume de variabilă.

Astfel, când Prologul caută să satisfacă scopul $s(X)$ prin aplicarea regulii (1), va încerca să unifice $s(X)$ cu membrul stâng $s(X)$ al acestei reguli în care variabila X este redenumită (în cele ce urmează, nu vom redenumi variabilele din fapte sau membrii stângi ai unor reguli decât dacă au aceleaşi nume ca unele variabile din scop sau din unificări anterioare, provenite din satisfacerea unui scop pentru care scopul curent este subscop):
(1) true; are loc unificarea $s(X) = s(X')$ ddacă avem unificarea $X = X'$, astfel că următorul scop este:
?- $t(X)$.
(1) false. ($t(X) \backslash = s(X')$)
(2) true, $X = 1$. (are loc unificarea $t(X) = t(1)$ ddacă avem unificarea $X = 1$)
true, $X = 1$.
true, $X = 1$.

Dacă mai cerem soluţii:
(3) true, $X = 2$. ($t(X) = t(2)$ ddacă $X = 2$)
true, $X = 2$.
true, $X = 2$.
Dacă mai cerem soluţii:
(4) $X = Y$: are loc unificarea $t(X) = t(Y)$ ddacă avem unificarea $X = Y$, aşadar noul scop este:
?- $a(X)$.
(1) false. ($a(X) \backslash = s(X')$)
(2) false. ($a(X) \backslash = t(1)$)
(3) false. ($a(X) \backslash = t(2)$)
(4) false. ($a(X) \backslash = t(Y)$)
(5) true, $X = 21$. ($a(X) = a(21)$ ddacă $X = 21$)
true, $X = 21$.
true, $X = 21$.
Dacă mai cerem soluţii:
(5) false. ($t(X) \backslash = a(21)$)
false.
(2) false. ($s(X) \backslash = t(1)$)
(3) false. ($s(X) \backslash = t(2)$)
(4) false. ($s(X) \backslash = t(Y)$)
(5) false. ($s(X) \backslash = a(21)$)
false.

Exemplul 4. Să considerăm următorul program, constând în definirea unui predicat ternar (i.e. cu 3 argumente) pentru concatenarea a două liste: $concat(L, M, C)$ e satisfăcut ddacă L şi M sunt liste, iar C este lista obţinută prin concatenarea lui L cu M:

(1) $concat([], L, L)$.

(2) $concat([H|T], L, [H|M])$:- $concat(T, L, M)$.

Considerăm următoarele interogări:
?- $concat([1, 2], [3, 4], [1, 2, 3, 4])$.
?- $concat([1, 2], [3, 4], X)$.
?- $concat([1, 2], [3, 4], [1])$.
?- $concat(X, Y, [1, 2, 3, 4])$.

Pentru cele ce urmează, a se observa că lista vidă [] este o **constantă**, aşadar nu unifică, cu nicio listă nevidă, de exemplu $[] \backslash = [H|T]$, pentru că lista $[H|T]$, care este nevidă întrucât conţine măcar capul H, este specificată cu ajutorul operatorului binar $[\_|\_]$, care nu unifică, cu [] pentru nicio pereche de valori ale argumentelor sale.

- Considerăm prima interogare: ?- $concat([1, 2], [3, 4], [1, 2, 3, 4])$.

(1) false. ($concat([1, 2], [3, 4], [1, 2, 3, 4])$ nu unifică, cu $concat([], L, L)$, pentru că lista $[1, 2]$ nu unifică, cu lista vidă [])
(2) true; are loc unificarea $concat([1, 2], [3, 4], [1, 2, 3, 4]) = concat([H|T], L, [H|M])$ ddacă au loc unificările: $H = 1$, $T = [2]$, $L = [3, 4]$ şi $M = [2, 3, 4]$, aşadar următorul scop este:
?- $concat([2], [3, 4], [2, 3, 4])$.
(1) false. ($concat([2], [3, 4], [2, 3, 4]) \backslash = concat([], L, L)$, pentru că $[2] \backslash = []$)
(2) true; $concat([2], [3, 4], [2, 3, 4]) = concat([H'|T'], L', [H'|M'])$ ddacă $H' = 2$, $T' = []$, $L' = [3, 4]$.
?- $concat([], [3, 4], [3, 4])$.
(1) true. ($concat([], [3, 4], [3, 4]) = concat([], L'', L'')$ ddacă $L'' = [3, 4]$)

true.

true.

true.

Dacă mai cerem soluţii:

(2) false. ($concat([], [3,4], [3,4]) \backslash = concat([H|T], L, [H|M])$, pentru că $[] \backslash = [H|T]$)

(2) false.

false.

- Acum să vedem cum e satisfăcută interogarea: ?- $concat([1,2], [3,4], X)$.

(1) false. ($concat([1,2], [3,4], X) \backslash = concat([], L, L)$, pentru că $[1,2] \backslash = []$)

(2) true; $concat([1,2], [3,4], X) = concat([H|T], L, [H|M])$ ddacă $H = 1$, $T = [2]$, $L = [3,4]$ şi $X = [H|M] = [1|M]$.

?- $concat([2], [3,4], M)$.

(1) false. ($concat([2], [3,4], M) \backslash = concat([], L, L)$, pentru că $[2] \backslash = []$)

(2) true; $concat([2], [3,4], M) = concat([H'|T'], L', [H'|M'])$ ddacă $H' = 2$, $T' = []$, $L' = [3,4]$ şi $M = [H'|M'] = [2|M']$.

?- $concat([], [3,4], M')$.

(1) true; ($concat([], [3,4], M') = concat([], L'', L'')$ ddacă $L'' = M' = [3,4]$.

true; $M' = [3,4]$.

true; $M = [2|M'] = [2|[3,4]] = [2,3,4]$.

true; $X = [1|M] = [1|[2,3,4]] = [1,2,3,4]$.

Dacă mai cerem soluţii:

(2) false. ($concat([], [3,4], M') \backslash = concat([H|T], L, [H|M])$, pentru că $[] \backslash = [H|T]$)

(2) false.

false.

- Acum să considerăm interogarea: ?- $concat([1,2], [3,4], [1])$.

(1) false. ($concat([1,2], [3,4], [1]) \backslash = concat([], L, L)$, pentru că lista $[1,2] \backslash = []$)

(2) true; $concat([1,2], [3,4], [1]) = concat([H|T], L, [H|M])$ ddacă $H = 1$, $T = [2]$, $L = [3,4]$ şi $M = []$, aşadar următorul scop este:

?- $concat([2], [3,4], [])$.

(1) false. ($concat([2], [3,4], [2,3,4]) \backslash = concat([], L, L)$, pentru că $[2] \backslash = []$)

(2) false. ($concat([2], [3,4], []) \backslash = concat([H'|T'], L', [H'|M'])$, pentru că lista nevidă $[H'|M'] \backslash = []$)

false.

false.

- Şi acum să considerăm interogarea: ?- $concat(X, Y, [1,2,3,4])$.

(1) true; $concat(X, Y, [1,2,3,4]) = concat([], L, L)$, ddacă $X = []$ şi $Y = L = [1,2,3,4]$.

true; $X = []$, $Y = [1,2,3,4]$.

Dacă mai cerem soluţii:

(2) true; $concat(X, Y, [1,2,3,4]) = concat([H|T], L1, [H|M])$ ddacă $H = 1$, $M = [2,3,4]$, $X = [H|T] = [1|T]$ şi $Y = L1$, aşadar următorul scop este:

?- $concat(T, L1, [2,3,4])$.

(1) true; $concat(T, L1, [2,3,4]) = concat([], L2, L2)$ ddacă $T = []$ şi $L1 = L2 = [2,3,4]$.

true; $T = []$, $Y = [2,3,4]$.

true; $X = [1|T] = [1|[]] = [1]$, $Y = L1 = [2,3,4]$.

Dacă mai cerem soluţii:

(2) true; $concat(T, L1, [2,3,4]) = concat([H'|T'], L', [H'|M'])$ ddacă $H' = 2$, $M' = [3,4]$, $T = [H'|T'] = [2|T']$ şi $L1 = L'$, aşadar următorul scop este:

?- $concat(T', L', [3,4])$.

(1) true; $concat(T', L', [3,4]) = concat([], L3, L3)$ ddacă $T' = []$ şi $L' = L3 = [3,4]$.

true; $T' = []$, $L' = [3,4]$.

true; $T = [2|T'] = [2|[]] = [2]$, $L1 = L' = [3,4]$.

true; $X = [1|T] = [1|[2]] = [1,2]$, $Y = L1 = [3,4]$.

Dacă mai cerem soluţii:

(2) true; $concat(T', L', [3,4]) = concat([H''|T''], L'', [H''|M''])$ ddacă $H'' = 3$, $M'' = [4]$, $T' = [H''|T''] = [3|T'']$ şi $L' = L''$, aşadar următorul scop este:

?- $concat(T'', L'', [4])$.

(1) true; $concat(T'', L'', [4]) = concat([], L4, L4)$ ddacă $T'' = []$ şi $L'' = L4 = [4]$.

true; $T'' = []$, $L'' = [4]$.

true; $T' = [3|T''] = [3|[]] = [3]$, $L' = L'' = [4]$.

true; $T = [2|T'] = [2|[3]] = [2,3]$, $L1 = L' = [4]$.

true; $X = [1|T] = [1|[2,3]] = [1,2,3]$, $Y = L1 = [4]$.

Dacă mai cerem soluţii:

(2) true; $concat(T'', L'', [4]) = concat([H'''|T'''], L''', [H'''|M'''])$ ddacă $H''' = 4$, $M''' = []$, $T'' = [H'''|T'''] = [4|T''']$ şi $L'' = L'''$, aşadar următorul scop este:

?- $concat(T''', L''', [])$.

(1) true; $concat(T''', L''', []) = concat([], L5, L5)$ ddacă $T''' = []$ şi $L''' = L5 = []$.

true; $T''' = []$, $L''' = []$.

true; $T'' = [4|T'''] = [4|[]] = [4]$, $L'' = L''' = []$.

true; $T' = [3|T''] = [3|[]] = [3,4]$, $L' = L'' = []$.

true; $T = [2|T'] = [2|[3,4]] = [2,3,4]$, $L1 = L' = []$.

true; $X = [1|T] = [1|[2,3,4]] = [1,2,3,4]$, $Y = L1 = []$.

Dacă mai cerem soluţii:

(2) false; $concat(T''', L''', [])\backslash = concat([H''''|T''''], L'''', [H''''|M''''])$, pentru că $[]\backslash = [H''''|M'''']$.

false.

false.

false.

false.

false.