UNIVERSITATEA **TITU MAIORESCU**

Facultatea de INFORMATICĂ

Asist. univ. Drd.

CORNACIU VERONICA

LOGICA MATEMATICĂ ŞI COMPUTAŢIONALĂ

Curs pentru învăţământul la distanţă

BUCUREŞTI – 2021

UNIVERSITATEA Titu MAIORESCU BUCUREŞTI
Facultatea de Informatică
Învăţământ la Distanţă

LOGICĂ MATEMATICĂ ŞI COMPUTAŢIONALĂ

Logica matematică şi computaţională este una din disciplinele de pregătire fundamentală care, pentru profilul INFORMATICĂ, este esenţială pentru pregătirea studenţilor şi pentru obţinerea creditelor transferabile prin procedurile de evaluare. Modul de prezentare a acestui material are în vedere particularităţile învăţământului la distanţă, la care studiul individual este determinant. Pentru orice nelămuriri faţă de acest material vă rugăm să contactaţi tutorele de disciplină care are datoria să vă ajute oferindu-vă toate explicaţiile necesare.

Disciplina de *Logica matematică şi computaţională* îşi propune următoarele obiective specifice:

- Însuşirea noţiunilor fundamentale din domeniile *Logică matematică şi computaţională*.
- Formarea deprinderilor de modelare matematică şi de transpunere în programare a unor probleme de natură tehnică, socială sau economică, cu utilizarea cunoştinţelor însuşite.
- Formarea şi dezvoltarea bazei matematice a studenţilor pentru disciplinele fundamentale şi de specialitate din anii superiori;
- Formarea şi dezvoltarea aptitudinilor şi deprinderilor de analiză logică, formulare corectă şi argumentare fundamentată, în rezolvarea problemelor tehnico-economice şi de specialitate;
- O comparaţie critică a metodelor de rezolvare evidenţiind, eventual, calea optimă de soluţionare.
- Înțelegerea modului de funcţionare a unor circuite logice simple care se află în componenta hard a calculatoarelor;
- Înțelegerea modului în care raţionamentul uman şi cel matematic poate fi modelat folosind logica propoziţiilor şi cea a predicatelor;
- Prezentarea aplicabilităţii acestor cunoştinţe pentru: demonstrării automate a teoremelor, programarea logică, inteligenţa artificială.

Vă precizăm de asemenea că, din punct de vedere al verificărilor şi al notării, cu adevărat importantă este capacitatea pe care trebuie să o dobândiţi şi să o probaţi de a rezolva toată tipologia de probleme aplicative aferente materialului teoretic prezentat în continuare. De aceea vă recomandăm să parcurgeţi cu atenţie toate aplicaţiile rezolvate, să rezolvaţi aplicaţiile propuse prin testele de autoevaluare şi temele de control; fiţi convinşi că examenul final apelează la tipurile de aplicaţii prezente în secţiunile menţionate anterior.

Coordonator disciplină: Conf. univ. dr. Daniela JOITA
Tutori: Asist. univ. Veronica CORNACIU

UNITATEA DE ÎNVĂŢARE NR.1

ALGEBRE BOOLE

În acesta unitate de învăţare sunt prezentate principalele noţiuni cu care operează teoria algebrelor Boole. Este pusă în evidenţă şi legătura dintre aceste noţiuni şi aplicaţiile spectaculoase ale algebrelor Boole şi domeniul calculatoarelor electronice şi disciplinelor învecinate.

În prima parte a acestei unităţi sunt prezentate noţiuni generale de teoria mulţimilor necesare abordării noţiunilor din partea doua a unitate de invăţare.

În partea a doua se prezintă teoria algebrelor Boole. Teoria algebrelor Boole s-a născut ca urmare a descoperirii că între legile logicii şi anumite legi ale calculului algebric există o perfectă analogie. Această descoperire este unanim atribuită lui George Boole (*An investigation into the laws of thought*, 1854)

Dintre matematicienii care au adus contribuţii mari la dezvoltarea teoriei algebrelor Boole trebuie menţionat în primul rând M.H. Stone pentru celebra sa teoremă de reprezentare (*The theory of representation for Boolean algebras*, Trans. A.M.S., 40 , 1936, p. 37-111) şi pentru teoria dualităţii a algebrelor Boole. (*Applications of the theory of Boolean rings to general topology*, Trans. A.M.S., 41, 1937, p. 375-481). De asemenea A. Tarski a obtinut rezultate remarcabile atât pe linia algebrică a acestui domeniu, cât mai ales pe linia legăturilor sale cu logica.

Algebrele Boole constituie reflectarea algebrică a calculului propoziţional, fiind modele algebrice ale calculului propoziţional, afirmaţie ce va fi precizată în lectia următoare. În unitatea de învăţare nr.2, metodele folosite pentru demonstrarea completitudinii sistemului formal al calculului predicatelor se vor baza în întregime pe algebrele Boole.

Astăzi, teoria algebrelor Boole se prezintă ca un fragment important al algebrei, având puternice conexiuni cu logica, dar fiind un capitol de sine stătător, atât prin rezultatele obţinute în interiorul său cât şi prin aplicaţiile sale în topologie, analiză, calculul probabilităţilor, etc.

Timpul mediu necesar însuşirii noţiunilor teoretice, formării deprinderilor de calcul şi utilizării metodelor de rezolvare a problemelor specifice teoriei algebrelor Boole este estimat la aproximativ 6-8 ore pentru fiecare din cele trei lecţii ale unităţii de învăţare, într-un ritm de 2-3 ore pe zi.

Lecţia 1- Elemente de teoria mulţimilor

În paragraful întâi al acestui capitol prezentăm câteva noţiuni şi proprietăţi ale calculului propoziţional, absolut necesar pentru demonstrarea propoziţiilor de teoria mulţimilor referitoare la operaţiile finite cu mulţimi. Paragraful al doilea conţine definirea operaţiilor cu mulţimi (reuniune, intersecţie, complementară, etc.) şi proprietăţile lor principale, produsul cartezian şi proprietatea sa de universalitate.

Elementele foarte sumare ale calculului predicatelor sunt expuse în paragraful 1.3, pentru a fi folosite în continuare în stabilirea proprietăţilor operaţiilor infinite cu mulţimi.

Relaţiile şi funcţiile sunt subiectul paragrafului 1.4, cardinalii şi operaţiile cu cardinali sunt prezentate în paragraful 1.5.

Ultimul paragraf se ocupă cu relaţiile de ordine şi preordine. Plasarea acestui paragraf în acest capitol este necesară pentru enunţarea axiomei lui Zorn, care este o axiomă a teoriei mulţimilor.

Nu am dezvoltat extensiv acest capitol, prezentând numai un minim necesar pentru tratarea capitolelor următoare. O serie de proprietăţi au fost date sub formă de exerciţii. Precizăm că punctul de vedere adoptat este acela al "teoriei naive a mulţimilor".

1.1. *Calculul propoziţional*

În calculul propoziţional se studiază propoziţiile din punctul de vedere al adevărului sau falsităţii lor, neluându-le în considerare conţinutul lor. Fără îndoială, legile logicii sunt expresii ale unor legi naturale obiective, însă neconsiderarea conţinutului este necesară pentru a surprinde relaţiile logice ale fenomenelor naturale în toată generalitatea lor.

Vom nota propoziţiile prin literele p, q, r, … . Pentru orice propoziţie p, definim **valoarea ei logică** $v(p)$ prin:

$$v(p) = \begin{cases} 1, \text{ daca propozitia } p \text{ este adevarata} \\ 0, \text{ daca propozitia } p \text{ este falsa} \end{cases}$$

Deci, pentru noi, o propoziţie p este perfect determinată dacă îi cunoaştem valoarea logică $v(p)$.

Dacă p, q sunt două propoziţii oarecare, atunci **conjuncţia** lor $p \wedge q$ este propoziţia "p şi q", iar valoarea ei de adevăr este dată de:

$$v(p \wedge q) = \begin{cases} 1, \text{ daca } p, \ q \text{ sunt simultan adevarate} \\ 0, \text{ daca cel putin una din propozitiile } p, \ q \text{ este falsa} \end{cases}$$

Cu alte cuvinte, $v(p \wedge q) = 1$ dacă şi numai dacă $v(p) = 1$ şi $v(q) = 1$.

Disjuncţia $p \vee q$ a propoziţiilor p, q este propoziţia "p sau q", iar valoarea ei logică este definită prin:

$$v(p \vee q) = \begin{cases} 1, \text{ daca cel putin una din propozitiile } p \text{ si } q \text{ sunt adevarate} \\ 0, \text{ daca ambele propozitii } p \text{ si } q \text{ sunt false} \end{cases}$$

Deci $v(p \vee q) = 1$ dacă şi numai dacă $v(p) = 1$ sau $v(q) = 1$.

Negaţia $\neg p$ a unei propoziţii p are următoarea valoare de adevăr:

$$v(\neg p) = \begin{cases} 0, \text{ daca } p \text{ este adevarata} \\ 1, \text{ daca } p \text{ este falsa} \end{cases}$$

Date fiind două propoziţii p, q, **<u>implicaţia</u>** $p \Rightarrow q$ este propoziţia "p implică q" a cărei valoare de adevăr este:

$$v(p \Rightarrow q) = \begin{cases} 0, & \text{daca } v(p) = 1 \text{ si } v(q) = 0 \\ 1, & \text{in rest} \end{cases}$$

<u>Echivalenţa</u> $p \Leftrightarrow q$ a două propoziţii p, q este propoziţia "p echivalent cu q" a cărei valoare de adevăr este dată de

$v(p \Leftrightarrow q) = 1$ dacă şi numai dacă $v(p) = v(q)$.

Aceste definiţii pot fi concentrate în următoarele <u>tabele de adevăr</u>.

$v(p)$	$v(q)$	$v(p \wedge q)$
1	1	1
1	0	0
0	1	0
0	0	0

disjuncţia

$v(p)$	$v(q)$	$v(p \vee q)$
1	1	1
1	0	1
0	1	1
0	0	0

conjuncţia

$v(p)$	$v(q)$	$v(p \Rightarrow q)$
1	1	1
1	0	0
0	1	1
0	0	1

implicaţia

$v(p)$	$v(q)$	$v(p \Leftrightarrow q)$
1	1	1
1	0	0
0	1	0
0	0	1

echivalenţa

$v(p)$	$v(\neg p)$
1	0
0	1

negaţia

Următoarele propoziţii sunt adevărate, pentru orice propoziţii p, q, r:

1. $(p \vee q) \Leftrightarrow (q \vee p)$; $(p \wedge q) \Leftrightarrow (q \wedge p)$;
2. $[(p \vee q) \vee r] \Leftrightarrow [p \vee (q \vee r)]$; $[(p \wedge q) \wedge r] \Leftrightarrow [p \wedge (q \wedge r)]$;
3. $(p \vee p) \Leftrightarrow p$; $(p \wedge p) \Leftrightarrow p$;
4. $[p \wedge (q \vee r)] \Leftrightarrow [(p \wedge q) \vee (p \wedge r)]$; $[p \vee (q \wedge r)] \Leftrightarrow [(p \vee q) \wedge (p \vee r)]$;
5. $[p \vee (p \wedge q)] \Leftrightarrow p$; $[p \wedge (p \vee q)] \Leftrightarrow p$;
6. $p \vee \neg p$; $\neg(p \wedge \neg p)$;
7. $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$; $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$;
8. $(p \vee q) \Leftrightarrow \neg(\neg p \wedge \neg q)$; $(p \wedge q) \Leftrightarrow \neg(\neg p \vee \neg q)$;
9. $\neg\neg p \Leftrightarrow p$;
10. $(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$;
11. $(p \Leftrightarrow q) \Leftrightarrow (p \Rightarrow q) \wedge (q \Rightarrow p)$;
12. $\neg(p \Rightarrow q) \Leftrightarrow (\neg q \vee \neg p)$;

Vom arăta, de exemlu, că prima propoziţie de la 7 este adevărată. Calculăm valoarea logică a propoziţiei $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$ pentru orice valoare 0 sau 1 pe care o pot lua propoziţiile componente p, q. Sistematizăm acest calcul prin următorul tabel:

$v(p)$	$v(q)$	$v(p\wedge q)$	$v(\neg(p\wedge q))$	$v(\neg p)$	$v(\neg q)$	$v(\neg p\vee\neg q)$	$v(\neg(p\wedge q))\Leftrightarrow v(\neg p\vee\neg q)$
1	1	1	0	0	0	0	1
1	0	0	1	0	1	1	1
0	1	0	1	1	0	1	1
0	0	0	1	1	1	1	1

În toate cazurile am obţinut valoarea 1.

Demonstraţia se face în aceeaşi manieră pentru toate proprietăţile 1- 12.

1.2. *Mulţimi, operaţii cu mulţimi*

Pentru noi, conceptul de ***mulţime*** va avea semnificaţia uzuală de colecţie, grămadă etc. Vom nota mulţimile prin literele $A, B, C, \ldots X, Y, Z$ etc. Obiectele din care este formată o mulţime se vor numi **elemente**. Elementele unei mulţimi vor fi notate $a, b, c, .. x, y, z$ etc.

Faptul că elementul x face parte din mulţimea A va fi notat $x \in A$ şi se va citi: "x aparţine mulţimii A".

Vom extinde conceptul de mulţime prin considerarea **mulţimii vide** $\varnothing$, care este "mulţimea fără nici un element".

Mulţimea A este **inclusă** în mulţimea B, dacă orice element al lui A este şi element al lui B. Scriem aceasta prescurtat $A \subset B$. Definiţia **incluziunii** $A \subset B$ poate fi dată şi astfel:

$$x \in A \Rightarrow x \in B$$

Reuniunea a două mulţimi A şi B este mulţimea $A \cup B$ definită de

$$x \in A \cup B \Leftrightarrow [x \in A] \vee [x \in B]$$

Un alt mod de a scrie această definiţie este:

$$A \cup B = \{x \,/\, (x \in A) \vee (x \in B)\}$$

În cele ce urmează vom omite parantezele, scriind astfel:

$$A \cup B = \{x \,/\, x \in A \vee x \in B\}.$$

Intersecţia a două mulţimi A şi B este mulţimea $A \cap B$ definită de:

$$x \in A \cap B \Leftrightarrow [x \in A] \wedge [x \in B]$$

Această definiţie poate fi dată sub forma:

$$A \cap B = \{x / x \in A \wedge x \in B\}.$$

Diferenţa a două mulţimi A şi B este definită astfel:

$$A - B = \{x / x \in A \wedge x \notin B\}.$$

Observaţia 1.2.1. Prin $x \notin B$ am notat propoziţia $\neg(x \in B)$.

Dacă $A \subset B$ se spune că A este o *parte* (sau o **submulţime**) a lui B. Prin convenţie, $\varnothing$ este submulţime a oricărei mulţimi. Pentru orice mulţime A, vom nota cu $\mathrm{P}(A)$ mulţimea tuturor părţilor lui A.

$$\mathrm{P}(A) = \{B / B \subseteq A\}$$

Fiind dată o mulţime A şi o parte a sa B, definim "complementara $\mathsf{C}_A(B)$ a lui B în raport cu A" prin

$$\mathsf{C}_A(B) = A - B = \{x / x \in A \wedge x \notin B\}.$$

Propoziţia 1.2.2: Pentru orice mulţimi A, B, C sunt verificate următoarele relaţii:

(1) $A \cup B = B \cup A$; $A \cap B = B \cap A$;
(2) $A \cup (B \cup C) = (A \cup B) \cup C$; $A \cap (B \cap C) = (A \cap B) \cap C$;
(3) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$; $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$;
(4) $A \cup A = A$; $A \cap A = A$;
(5) $A \cup (A \cap B) = A$; $A \cap (A \cup B) = A$;
(6) $A \cup \varnothing = A$; $A \cap \varnothing = \varnothing$;
(7) $A = B \Leftrightarrow (A \subseteq B) \wedge (B \subseteq A)$;
(8) $A \subseteq A$;
(9) $[A \subset B] \wedge [B \subset C] \Rightarrow A \subset C$;
(10) $A \cap B \subset A \subset A \cup B$; $A - B \subset A$;
(11) $[A \subset B] \wedge [C \subset D] \Rightarrow [(A \cup C) \subset (B \cup D)] \wedge [(A \cap C) \subset (B \cap D)]$;
(12) $[A \subset B] \Leftrightarrow [A \cup B = B] \Leftrightarrow [A \cap B = A]$;
(13) $A \cap B = A - (A - B)$;
(14) $A \cup (B - A) = A \cup B$;
(15) $A - (A \cap B) = A - B$;
(16) $A \cap (B - C) = (A \cap B) - C$;

Demonstraţie: Vom stabili, de exemplu prima din relaţiile (3):

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;

Aplicând propoziţia 4, din §1.1, rezultă echivalenţele:

$$\begin{aligned} x \in A \cup (B \cap C) &\Leftrightarrow (x \in A) \vee [x \in B \cap C] \\ &\Leftrightarrow (x \in A) \vee [(x \in B) \wedge (x \in C)] \\ &\Leftrightarrow [(x \in A) \vee (x \in B)] \wedge [(x \in A) \vee (x \in C)] \\ &\Leftrightarrow [x \in A \cup B] \wedge [x \in A \cup C] \\ &\Leftrightarrow x \in (A \cup B) \cap (A \cup C) \end{aligned}$$

A rezultat: $x \in A \cup (B \cap C) \Leftrightarrow x \in (A \cup B) \cap (A \cup C)$, ceea ce este acelaşi lucru cu egalitatea ce trebuie demonstrată.

În acelaşi mod se demonstrează toate relaţiile enumerate mai sus.

Propoziţia 1.2.3: Dacă B, C sunt mulţimi ale lui A, atunci avem relaţiile:

(17) $B \subset C \Rightarrow \mathsf{C}_A(C) \subset \mathsf{C}_A(B)$;

(18) $\mathsf{C}_A(B \cup C) = \mathsf{C}_A(B) \cap \mathsf{C}_A(C)$;

(19) $\mathsf{C}_A(B \cap C) = \mathsf{C}_A(B) \cup \mathsf{C}_A(C)$;

(20) $\mathsf{C}_A(A) = \varnothing$; $\mathsf{C}_A(\varnothing) = A$;

Lăsăm demonstraţia acestor relaţii pe seama cititorului.

Dacă sunt date mulţimile $A_1, A_2,..., A_n$ atunci definim intersecţia şi reuniunea lor astfel:

$A_1 \cap A_2 \cap ... \cap A_n = \{x \mid (x \in A_1) \wedge (x \in A_2) \wedge ... \wedge (x \in A_n)\}$

$A_1 \cup A_2 \cup ... \cup A_n = \{x \mid (x \in A_1) \vee (x \in A_2) \vee ... \vee (x \in A_n)\}$

se mai folosesc şi notaţiile:

$$\bigcap_{i=1}^{n} A_i = A_1 \cap ... \cap A_n; \qquad \bigcup_{i=1}^{n} A_i = A_1 \cup ... \cup A_n .$$

Menţionăm următoarele proprietăţi:

(21) $$[\bigcap_{i=1}^{n} A_i] \cup B = \bigcap_{i=1}^{n} [A_i \cup B];$$

(22) $$[\bigcup_{i=1}^{n} A_i] \cap B = \bigcup_{i=1}^{n} [A_i \cap B];$$

(23) $$\mathsf{C}_A[\bigcup_{i=1}^{n} A_i] = \bigcap_{i=1}^{n} \mathsf{C}_A(A_i);$$

(24) $$\mathsf{C}_A[\bigcap_{i=1}^{n} A_i] = \bigcup_{i=1}^{n} \mathsf{C}_A(A_i);$$

Fie A, B două mulţimi oarecare. Produsul cartezian al mulţimilor A şi B este mulţimea $A \times B$ definită astfel :

$$A \times B = \{(x, y) \mid (x \in A) \wedge (y \in B)\}.$$

In general, produsul cartezian a n mulţimi $A_1, A_2,..., A_n$ este:

$$A_1 \times A_2 \times ... \times A_n = \{(x_1, x_2,..., x_n) \mid (x_1 \in A_1) \wedge (x_2 \in A_2) \wedge ... \wedge (x_n \in A_n)\}.$$

Se folosesc notaţiile:

$$\prod_{i=1}^{n} A_i = A_1 \times ... \times A_n \text{ sau dacă } A_1 = A_2 = \cdots = A_n = A \text{ atunci } A^n = \underbrace{A \times A \times \cdots \times A}_{\text{de } n \text{ ori}}.$$

Produsul cartezian are următoarele proprietăţi:

(25) $(A_1 \cup A_2) \times B = (A_1 \times B) \cup (A_2 \times B)$;

(26) $(A_1 \cap A_2) \times B = (A_1 \times B) \cap (A_2 \times B)$;

(27) $(A_1 - A_2) \times B = (A_1 \times B) - (A_2 \times B)$;

(28) Dacă A_1, A_2, B_1, B_2 sunt nevide, atunci

$[A_1 \times B_1 = A_2 \times B_2] \Rightarrow [A_1 = A_2] \wedge [B_1 = B_2]$;

(29) $A \times B = \varnothing$, dacă $A = \varnothing$ sau $B = \varnothing$.

Fie $(A_i)_{i \in I}$ o familie de mulţimi indexată de mulţimea I. Prin produsul cartezian al familiei $(A_i)_{i \in I}$ înţelegem mulţimea:

$$\prod_{i \in I} A_i = \left\{ f : I \to \bigcup_{i \in I} A_i \mid f(i) \in A_i, \text{pentru orice } i \in I \right\}$$

În general, printr-o familie $(x_i)_{i\in I}$ de elemente ale unei mulţimi X se înţelege că fiecărui $i\in I$ îi este asociat un singur element x_i al lui X. I se numeşte mulţimea de indici a familiei $(x_i)_{i\in I}$.

Orice funcţie $f : I \to \bigcup_{i\in I} A_i$ este perfect determinată de familia $(f(i))_{i\in I}$, deci definiţia produsului cartezian $\prod_{i\in I} A_i$ mai poate fi dată astfel:

$$\prod_{i\in I} A_i = \{(x_i)_{i\in I} \mid x_i \in A_i, \text{pentru orice } i \in I\}.$$

Pentru orice $j\in I$ aplicaţia $\pi_j((x_i)_{i\in I}) = x_j$ este surjectivă, $\{\pi_j \mid j\in I\}$ se numesc proiecţiile canonice ale lui $\prod_{i\in I} A_i$.

1.3. *Calculul predicatelor*

În calculul propoziţional nu ne-am interesat de structura propoziţiilor, care au fost considerate ca nişte întregi, preocupându-ne de valoarea lor logică.

Considerând propoziţia "*Socrate este muritor*", observăm în alcătuirea lui un individ, "*Socrate*" şi o proprietate "*muritor*". Propoziţiile "*Platon este muritor*" şi "*Aristotel este muritor*" au aceeaşi formă şi diferă doar individul despre care se afirmă că este muritor.

Toate aceste propoziţii au forma "*x este muritor*". În general, vom considera expresii de forma "*x are proprietatea P*", pe care le vom nota $P(x)$.

Aceste expresii le vom numi ***predicate*** (Hilbert şi Bernays, Grundlagen der Mathematik, vol.1, 1934) sau ***funcţii propoziţionale*** (Russel şi Whitehead, Principia Mathematica, vol.1, 1910). În Principia Mathematica, acest concept este definit astfel: "printr-o funcţie propoziţională înţelegem ceva care conţine o variabilă x şi exprimă o propoziţie de îndată ce lui x i se atribuie o valoare".

Cu alte cuvinte, un predicat $P(x)$ devine o propoziţie $P(a)$ dacă i se atribuie lui x o valoare determinată a. Propoziţia $P(a)$ poate fi adevărată sau falsă.

Vom presupune că x ia valori într-o mulţime A de indivizi, astfel încât pentru orice $a\in A$, $P(A)$ este o propoziţie cu sens.

Pentru exemplificare, să luăm predicatul "*x este muritor*". Propoziţia "*Socrate este muritor* " are sens pe când "*numărul* 7 *este muritor*" este fără sens.

Fie $P(x)$ un predicat oarecare. Din predicatul $P(x)$ putem forma următoarele propoziţii:

$(\exists x)P(x)$: există x care are proprietatea P.

$(\forall x)P(x)$: pentru orice x are loc proprietatea P.

$\forall$ se numeşte ***cuantificator universal***, iar $\exists$ se numeşte ***cuantificator existenţial***.

Vom spune că propoziţia $(\exists x)P(x)$ este adevărată în mulţimea A, dacă există $a\in A$, astfel încât $P(a)$ este o propoziţie adevărată.

Propoziţia $(\forall x)P(x)$ este adevărată în mulţimea A, dacă pentru orice $a\in A$, propoziţia $P(a)$ este adevărată.

În mod analog, pot fi considerate predicate $P(x_1,\ldots,x_n)$ care depind de n variabile. Aceste predicate se numesc predicate n-are; $x_1,\ldots,x_n$ se vor numi variabile.

Dacă $P(x, y)$ este un predicat binar, atunci $(\forall x)P(x, y)$ şi $(\exists x)P(x, y)$ sunt predicate unare în variabila y. Vom spune că în aceste predicate variabila x este ***legată***, iar variabila y este ***liberă***.

Aceste definiţii se pot generaliza pentru predicate în oricâte variabile. În scrierea predicatelor orice variabilă liberă trebuie notată diferit de orice variabilă legată.

De exemplu, nu putem avea $(\exists x)(\forall x)P(x, y)$, însă scrierea $(\exists x)(\forall y)P(x, y)$ este corectă.

Dacă P, Q sunt predicate, atunci

P, $P \vee Q$, $P \wedge Q$, $P \Rightarrow Q$, $P \Leftrightarrow Q$, sunt de asemenea predicate.

Un predicat în care toate variabilele sunt legate se va numi predicat constant sau enunţ.

Pentru orice predicate $P(x)$, $Q(x)$ şi pentru orice mulţime A, în A sunt adevărate următoarele enunţuri:

(a) $\neg(\forall x)P(x) \Leftrightarrow (\exists x)\neg P(x)$
(b) $\neg(\exists x)P(x) \Leftrightarrow (\forall x)\neg P(x)$
(c) $(\forall x)P(x) \Rightarrow (\exists x)P(x)$
(d) $[\ (\forall x)(P(x) \Rightarrow Q(x))] \Rightarrow [(\forall x)P(x) \Rightarrow (\forall x)Q(x)]$
(e) $[(\exists x)P(x) \Rightarrow (\exists x)Q(x)] \Rightarrow [\ (\exists x)(P(x) \Rightarrow Q(x))]$
(f) $(\forall x)[(P(x) \wedge Q(x))] \Leftrightarrow [(\forall x)P(x) \wedge (\forall y)Q(y)]$
(g) $(\exists x)[P(x) \vee Q(x)] \Leftrightarrow [(\exists x)P(x) \vee (\exists x)Q(x)]$

Dacă $P(x, y)$ este un predicat binar, atunci în A sunt adevărate enunţurile:

(h) $(\forall x)(\forall y)P(x, y) \Leftrightarrow (\forall y)(\forall\ x)P(x, y)$
(i) $(\exists x)(\exists\ y)P(x, y) \Leftrightarrow (\exists y)(\exists\ x)P(x, y)$
(j) $(\exists x)(\forall\ y)P(x, y) \Leftrightarrow (\forall y)(\exists\ x)P(x, y)$

1.4. *Relaţii şi funcţii*

Fie A o mulţime. O relaţie n-ară este o submulţime R a lui A^n.

Definiţia 1.4.1 : Fie A, B două mulţimi oarecare. O funcţie definită pe A cu valori în B este o relaţie unară pe $A \times B$ (adică $\Gamma \subset A \times B$) cu proprietatea că pentru orice $x \in A$ există un element $y \in A$ şi numai unul, astfel încât $(x, y) \in \Gamma$.

Vom nota o funcţie $\Gamma \subset$ A × B prin $f : A \to B$, simbolul f având semnificaţia următoare: fiecărui element $x \in A$ îi corespunde un singur element $f(x) \in B$ astfel încât $(x, f(x)) \in \Gamma$.

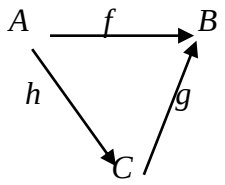
A se numeşte domeniul de definiţie al funcţiei $f : A \to B$ şi B se numeste domeniul valorilor lui f.

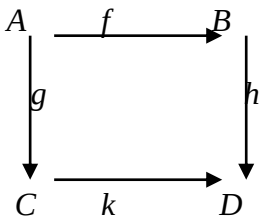
Date funcţiile $f : A \to B$ şi $g : B \to C$, prin compunerea lor se înţelege funcţia $g \circ f : A \to C$, definită de $(g \circ f)(x) = g(f(x))$, pentru orice $x \in A$.

Compunerea funcţiilor este asociativă: pentru funcţiile f : A $\to B$, $g : B \to C$, $h : C \to D$, avem relaţia $h \circ (g \circ f) = (h \circ g) \circ f$.

Pentru orice mulţime A, funcţia identică $1_A : A \to A$ este definită de $1_A(x) = x$, pentru orice $x \in A$.

Vom spune, că diagramele următoare:





sunt comutative, dacă $g \circ f = h$, respeciv $h \circ f = k \circ g$.

În general, o configuraţie compusă din diagrame de tipul de mai sus este o diagramă comutativă dacă diagramele componente sunt comutative.

Funcţia $f : A \to B$ este injectivă dacă pentru orice $x, y \in A$, avem:

$f(x) = f(y) \Rightarrow x = y$.

Evident această relaţie este echivalentă cu

$x \neq y \Rightarrow f(x) \neq f(y)$

Funcţia $f : A \to B$ este surjectivă dacă pentru orice $y \in B$, există $x \in A$, astfel încât $f(x) = y$.

O funcţie injectivă şi surjectivă se numeşte bijectivă. Pentru aceste trei categorii de funcţii se folosesc şi denumirile: injecţie, surjecţie şi bijecţie.

O funcţie $f : A \to B$ este inversabilă, dacă există o funcţie $g : B \to A$ cu proprietăţile $g \circ f = 1_A$ şi $f \circ g = 1_B$.

Exerciţiu. Dacă $f : A \to B$ este inversabilă să se arate că există o singură funcţie $g : B \to A$ cu proprietăţile $g \circ f = 1_A$ şi $f \circ g = 1_B$.

Funcţia $g : B \to A$ cu aceste proprietăţi se numeşte inversa lui f şi se notează f^{-1}. Deci avem relaţiile

$$f^{-1} \circ f = 1_A, \qquad f \circ f^{-1} = 1_B$$

Propoziţia 1.4.1: Pentru o funcţie $f : A \to B$ sunt echivalente afirmaţiile următoare:

(i) f este bijectivă.

(ii) f este inversabilă.

Demonstraţie: (i)$\Rightarrow$(ii). Presupunem că f este bijectivă. Fie $y \in B$. Cum f este surjectivă, există $x \in A$ astfel încât $f(x) = y$. f fiind injectivă, acest element este unic, deci putem defini o funcţie $g : B \to A$ prin $g(y) = x$. Rezultă imediat că această funcţie este inversa lui f.

(ii)$\Rightarrow$(i) este un simplu exerciţiu pentru cititor.

Fie $f : A \to B$ o funcţie oarecare. Dacă $X \subset A$ şi $Y \subset B$, atunci notăm: $f(X)=\{f(x)/x \in X\}$: imaginea directă a lui X prin f.

$f^{-1}(Y) = \{x \in A / f(x) \in Y\}$: imaginea reciprocă a lui Y prin f.

Propoziţia 1.4.2: Fie $f : A \to B$ o funcţie oarecare şi $X_1, X_2 \subset A$, $Y_1, Y_2 \subset B$. Atunci avem următoarele relaţii:

$$f(X_1 \cup X_2) = f(X_1) \cup f(X_2)$$
$$f(X_1 \cap X_2) \subset f(X_1) \cap f(X_2)$$
$$f(X_1) - f(X_2) \subset f(X_1 - X_2)$$
$$f^{-1}(Y_1 \cup Y_2) = f^{-1}(Y_1) \cup f^{-1}(Y_2)$$
$$f^{-1}(Y_1 \cap Y_2) = f^{-1}(Y_1) \cap f^{-1}(Y_2)$$
$$f^{-1}(Y_1 - Y_2) = f^{-1}(Y_1) - f^{-1}(Y_2)$$

Fie I o mulţime nevidă. Dacă fiecărui $i \in I$ îi este asociată o mulţime A_i spunem că avem o familie de mulţimi $(A_i)_{i \in I}$ indexată de mulţimea I.

Reuniunea şi intersecţia familiei $(A_i)_{i \in I}$ sunt definite astfel:

$$\bigcup_{i \in I} A_i = \{x \mid \text{exista } i \in I, \text{astfel incat } x \in A_i\}$$

$$\bigcap_{i \in I} A_i = \{x \mid x \in A_i, \text{pentru orice } i \in I\}$$

Propoziţia 1.4.3: Pentru orice familie $(A_i)_{i \in I}$ de mulţimi şi pentru orice mulţime B, avem relaţiile următoare:

$$(\bigcup_{i \in I} A_i) \cap B = \bigcup_{i \in I} (A_i \cap B); \qquad (\bigcap_{i \in I} A_i) \cup B = \bigcap_{i \in I} (A_i \cup B);$$

Propoziţia 1.4.4: Dacă $(A_i)_{i\in I}$ este o familie de părţi ale unei mulţimi X, atunci

$$\mathsf{C}_X\left(\bigcup_{i\in I} A_i\right) = \bigcap_{i\in I} \mathsf{C}_X(A_i); \quad \mathsf{C}_X\left(\bigcap_{i\in I} A_i\right) = \bigcup_{i\in I} \mathsf{C}_X(A_i)$$

Demonstraţia acestor două propoziţii este simplă. Spre exemplificare, să demonstrăm a doua relaţie a Propoziţiei 1.4.4:

$$x\in \mathsf{C}_X\left(\bigcap_{i\in I} A_i\right) \Leftrightarrow \neg\left(x\in \bigcap_{i\in I} A_i\right)$$
$$\Leftrightarrow \neg(\forall i\in I)[x\in A_i]$$
$$\Leftrightarrow (\exists i\in I)[\neg(x\in A_i)]$$
$$\Leftrightarrow (\exists i\in I)[x\in \mathsf{C}_X(A_i)]$$
$$\Leftrightarrow x\in \bigcup_{i\in I} \mathsf{C}_X(A_i)$$

folosind relaţia (a), § 1.3.

Fie acum R o relaţie binară pe mulţimea A ($R \subset A^2$). R se numeşte relaţie de echivalenţă pe A dacă pentru orice $x, y, z \in A$ sunt satisfăcute proprietăţile:

$(x, x)\in R$ (reflexivitate)

$(x, y)\in R \Rightarrow (y, x)\in R$ (simetrie)

$(x, y)\in R, (y, z)\in R \Rightarrow (x, z)\in R$ (tranzitivitate)

Vom folosi următoarea notaţie: $x \sim y \Leftrightarrow (x, y)\in R$. Proprietăţile de mai sus se transcriu astfel:

$x \sim x$

$x \sim y \Rightarrow y \sim x$

$x \sim y, y \sim z \Rightarrow x \sim z$

Pentru orice $x\in A$ vom nota $\hat{x} = \{y\in A \mid x \sim y\}$. $\hat{x}$ se numeşte clasa de echivalenţă a lui x. Sunt imediate proprietăţile:

$x \sim y \Leftrightarrow \hat{x} = \hat{y}$

$x \sim y \Rightarrow \hat{x} \cap \hat{y} = \varnothing$

O familie $(A_i)_{i\in I}$ de submulţimi ale lui A se numeşte partiţie dacă:

$i, j\in I, i \neq j \Rightarrow A_i \cap A_j = \varnothing; \bigcup_{i\in I} A_i = A.$

Orice partiţie $(A_i)_{i\in I}$ defineşte o relaţie de echivalenţă pe A:

$x \sim y \Leftrightarrow$ există $i\in I$, astfel încât $x, y\in A_i$.

Reciproc, orice relaţie de echivalenţă $\sim$ pe A pune în evidenţă o partiţie dată de mulţimea claselor de echivalenţă.

Se poate arăta că această corespondenţă este bijectivă.

Dată fiind relaţia de echivalenţă $\sim$ pe A, mulţimea claselor de echivalenţă ale elementelor lui A se numeşte mulţimea cât a lui A prin $\sim$ şi se notează prin $A/_\sim$.

Funcţia $p : A \to A/_\sim$ definită de $p(x) = \hat{x}$, pentru orice $x\in A$, este surjectivă.

1.5. *Mulţimi echipotente. Cardinali*

Pentru orice mulţime finită, numărul său de elemente este o noţiune bine precizată. Numărul natural **n** este reprezentarea abstractă a tuturor mulţimilor "cu **n** elemente". Conceptul de număr natural permite compararea mulţimilor finite.

Este evident că a spune că două mulţimi finite au acelaşi număr de elemente este echivalent cu faptul că ele se pot pune în corespondenţă bijectivă.

Această observaţie sugerează introducerea unui concept care să reprezinte "numărul de elemente al unei mulţimi oarecare".

Vom spune că două mulţimi A şi B sunt echipotente sau că au aceeaşi putere dacă există o bijecţie $f : A \to B$. Se scrie acest lucru simbolic: $A \sim B$.

Propoziţia 1.5.1: Echipotenţa este o relaţie reflexivă, simetrică şi tranzitivă.

Demonstraţie. Aplicaţia identică $1_A : A \to B$ este bijectivă, deci $A \sim A$. Dacă $A \sim B$, atunci există o bijecţie $f : A \to B$. Inversa $f^{-1} : A \to B$ este bijectivă deci $B \sim A$. Presupunând că $A \sim B$ şi $B \sim C$, rezultă bijecţiile $f : A \to B$, $g : B \to C$. Funcţia compusă $g \circ f : A \to C$ este bijectivă, deci $A \sim C$.

Observaţia 1.5.1.1: Echipotenţa este o "relaţie de echivalenţă" definită pe clasa tuturor mulţimilor.

Pentru orice mulţime A, vom nota cu $\overline{\overline{A}}$ sau **card(*A*)** clasa de echivalenţă a mulţimilor echipotente cu A:

$\text{card}(A) = \overline{\overline{A}} = \{B \mid \text{există } f : A \to B \text{ bijectivă}\}$.

Vom spune că $\overline{\overline{A}}$ este cardinalul mulţimii A.

Observaţia 1.5.1.2: În cazul în care A este o mulţime finită, $\overline{\overline{A}}$ poate fi asimilat cu numărul **n** al elementelor lui A, în sensul că **n** reprezintă toate mulţimile din $\overline{\overline{A}}$, identificate din punctul de vedere al "numărului lor de elemente".

Mulţimi numărabile. O mulţime A este numărabilă dacă este echipotentă cu mulţimea $\boldsymbol{N}$ a numerelor naturale. Vom nota cardinalul lui $\boldsymbol{N}$ cu $\aleph_0$ (***aleph zero***).

Cu alte cuvinte, o mulţime este numărabilă dacă elementele sale se pot aşeza sub forma unui şir.

Este evident că mulţimea $\boldsymbol{Z}$ a numerelor întregi este numărabilă.

Au loc următoarele proprietăţi pe care le prezentăm fără demonstraţie.

Propoziţia 1.5.2: Orice reuniune numărabilă de mulţimi numărabile este o mulţime numărabilă.

Corolarul 1.5.2.1: Orice reuniune finită de mulţimi numărabile este o mulţime numărabilă.

Corolarul 1.5.2.2: Produsul cartezian al două mulţimi numărabile A, B este o mulţime numărabilă.

Corolarul 1.5.2.3: Produsul cartezian a $\boldsymbol{n}$ multimi numărabile $A_1, \ldots, A_n$ este o mulţime numărabilă.

Corolarul 1.5.2.4: Mulţimea Q a numerelor raţionale este numărabilă.

Corolarul 1.5.2.5: Mulţimea şirurilor finite ai căror termeni apartin unei multimi numărabile este numărabilă.

Corolarul 1.5.2.6: Mulţimea $Q[X]$ a polinoamelor cu coeficienţi raţionali este numărabilă.

Propoziţia 1.5.3. Mulţimea (0,1) nu este numărabilă.

Corolarul 1.5.3.1. Mulţimea R a numerelor reale este nenumărabilă.

Operaţii cu cardinali

Fie A, B două mulţimi oarecare. Atunci putem găsi două mulţimi A_1, B_1 astfel încât $A \sim A_1$, $B \sim B_1$ şi $A_1 \cap B_1 = \varnothing$. Într-adevăr, luând două elemente $a \neq b$ şi punând $A_1 = \{a\} \times A$, $B_1 = \{b\} \times B$ vom avea:

$A \sim A_1$: prin funcţia bijectivă $f : A \to A_1$ dată de $f(x) = (a, x)$, pentru orice $x \in A$;

$B \sim B_1$: prin funcţia bijectivă $g : B \to B_1$ dată de $g(y) = (b, y)$, pentru orice $y \in B$;

$A_1 \cap B_1 = \varnothing$.

Prin definiţie, **suma** cardinalilor $\overline{\overline{A}}$, $\overline{\overline{B}}$ este $\overline{\overline{A}} + \overline{\overline{B}} = \overline{\overline{A_1 \cup B_1}}$

Pentru orice mulţimi A, B, definim **produsul** cardinalilor $\overline{\overline{A}}$, $\overline{\overline{B}}$ prin $\overline{\overline{A}} \cdot \overline{\overline{B}} = \overline{\overline{A \times B}}$

Aceste definiţii se generalizează pentru o familie oarecare de mulţimi $(A_i)_{i \in I}$. Se poate găsi la fel ca mai sus, o familie $(A_i')_{i \in I}$ de mulţimi cu proprietăţile:

$A_i \sim A_i'$, pentru orice $i \in I$.

$A_i \cap A_i' = \varnothing$, pentru orice $i, j \in I$, $i \neq j$.

Atunci suma cardinalilor $(\overline{\overline{A_i}})_{i \in I}$ este:

$$\sum_{i \in I} \overline{\overline{A_i}} = \overline{\overline{\bigcup_{i \in I} A_i'}}.$$

Produsul familiei $(\overline{\overline{A_i}})_{i \in I}$ de cardinali va fi:

$$\prod_{i \in I} \overline{\overline{A_i}} = \overline{\overline{\prod_{i \in I} A_i}}$$

Propoziţia 1.5.4: Pentru orice mulţime A, $\overline{\overline{P(A)}} = 2^{\overline{A}}$.

Demonstraţie. Considerând o mulţime oarecare cu două elemente, fie ea $\{0,1\}$, va trebui să arătăm că $P(A) \sim \{0,1\}^A$.

Pentru orice $B \subset A$, definim funcţia sa caracteristică $X_B : A \to \{0,1\}$ prin:

$$X_B(x) = \begin{cases} 1, & \text{dacă } x \in B \\ 0, & \text{dacă } x \notin B \end{cases}.$$

Considerăm funcţia $\Phi : P(A) \to \{0,1\}^A$ dată de $\Phi(B) = X_B$, pentru orice $B \in P(A)$.

Definim acum o altă funcţie $\Psi : \{0,1\}^A \to P(A)$ prin:

$$\Psi(f) = f^{-1}(\{1\}) = \{ x \in A \mid f(x) = 1 \},$$

pentru orice $f \in \{0,1\}^A$.

Se poate arăta că

$$\Psi \circ \Phi = 1_{\mathfrak{I}(A)}, \quad \Psi \circ \Phi = 1_{\{0,1\}^A}.$$

deci $P(A) \sim \{0,1\}^A$.

Următoarea propoziţie o prezentăm fără demonstraţie.

Propoziţia 1.5.5: (Cantor). Pentru orice mulţime A, avem $\overline{\overline{A}} \neq 2^{\overline{A}}$.

Pentru orice doi cardinali $\overline{\overline{A}}$ şi $\overline{\overline{B}}$, vom spune că $\overline{\overline{A}} \le \overline{\overline{B}}$ decă există o bijecţie $f: A \to B$.

Dacă $\overline{\overline{A}} \le \overline{\overline{B}}$ şi $\overline{\overline{A}} \ne \overline{\overline{B}}$, atunci vom scrie $\overline{\overline{A}} < \overline{\overline{B}}$.

Pentru $\overline{\overline{A}}$, $\overline{\overline{B}}$ finiţi, relaţia $\overline{\overline{A}} \le \overline{\overline{B}}$ revine la relaţia obişnuită de ordine între două numere naturale.

Relaţia $\le$ are proprietăţile următoare:

$A \subset B \Rightarrow \overline{\overline{A}} \le \overline{\overline{B}}$;

$\overline{\overline{A}} \le \overline{\overline{A}}$;

Observaţia 1.5.6:

(i) Din Corolarul Propoziţiei 1.6.3, rezultă $\aleph_0 = \overline{\overline{N}} < \overline{\overline{R}}$.

(ii) Teorema lui Cantor (Propoziţia 1.6.5) se poate formula astfel:

$\overline{\overline{A}} < 2^{\overline{\overline{A}}}$, pentru orice mulţime A.

Teorema Cantor-Bernstein : $\overline{\overline{A}} \le \overline{\overline{B}}$, $\overline{\overline{B}} \le \overline{\overline{A}} \Rightarrow \overline{\overline{A}} = \overline{\overline{B}}$.

Pentru demonstraţie se poate consulta K. Kuratowski, Introducere în teoria mulţimilor şi în topologie, Ed. Tehnică,1969, pag.79-80.

1.6. *Relaţii de ordine*

O relaţie binară R pe o mulţime nevidă A se numeşte **relaţie de preordine** dacă pentru orice x, y, z $\in A$ avem:

($P1$) $x\,R\,x$ (reflexivitate)

($P2$) $x\,Ry, y\,R\,z \Rightarrow x\,R\,z$ (tranzitivitate)

Mulţimea A înzestrată cu o relaţie de preordine R se numeşte **mulţime preordonată**.

Relaţia de preordine R se numeşte **relaţie de ordine** dacă verifică relaţia

($P3$) $x\,R\,y; y\,R\,x \Rightarrow x\,R\,y$ (antisimetrie)

pentru orice $x, y \in A$.

O relaţie de ordine se notează în mod uzual cu $\le$, deci cele trei relaţii ce o definesc se descriu astfel:

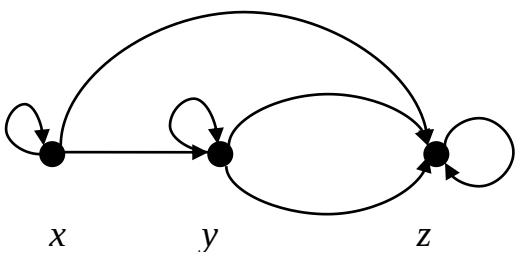
$x \le x$

$x \le y, y \le z \Rightarrow x \le z$

$x \le y, y \le x \Rightarrow x = y$

O **mulţime ordonată** este o mulţime A înzestrată cu o relaţie de ordine $\le$. Vom nota $x < y \Leftrightarrow x \le y$ şi $x \ne y$.

Exemplu de relatie de preordine care nu este o relaţie de ordine. Considerăm o mulţime A=$\{x, y, z\}$ în care relaţia R este definită prin graful următor:



şi anume: $x\,R\,x,\ y\,R\,y,\ z\,R\,z$

$x\,R\,y,\ y\,R\,z,\ z\,R\,y,\ x\,R\,z.$

Se observă că R este reflexivă şi tranzitivă, dar nu este antisimetrică:

$y\,R\,z,\ z\,R\,y$ dar nu şi $y = z$.

O mulţime parţial ordonată $(A, \leq)$ se numeşte **mulţime total ordonată** dacă

(*P*4) pentru orice $x, y \in A$, avem $x\,R\,y$ sau $y\,R\,x$.

Exemplu de mulţime parţial ordonată care nu este total ordonată. În mulţimea Z a numerelor întregi considerăm relaţia:

$x\,R\,y \Leftrightarrow x$ divide pe y

Este evident că R este o relaţie de ordine care nu este totală.

Mulţimea R a numerelor reale înzestrată cu relaţia de ordine naturală este o mulţime total ordonată.

Dacă (A, R) şi (A', R') sunt două mulţimi preordonate, atunci o funcţie $f : A \to A'$ se numeşte izotonă dacă pentru orice $x, y \in A$ avem:

$x\,R\,y \Rightarrow f(x)\,R'\,f(y)$.

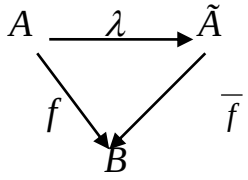
În cazul când $(A, \leq)$ şi $(A', \leq)$ sunt două mulţimi parţial ordonate, $f : A \to A'$ este izotonă dacă:

$x \leq y \Rightarrow f(x) \leq f(y)$

Are loc următorul rezultat important.

Propoziţia 1.6.1: Fie (A, R) o mulţime parţial ordonată. Atunci există o mulţime parţial ordonată, $(\tilde{A}, \leq)$ şi o funcţie izotonă $\lambda : A \to \tilde{A}$ cu proprietatea următoare:

(*) Pentru orice mulţime parţial ordonată $(B, \leq)$ şi pentru orice funcţie izotonă $f : A \to B$ există o unică funcţie izotona $\overline{f} : \tilde{A} \to B$, astfel încât următoarea diagramă este comutativă:



Fie acum $(x_i)_{i \in I}$ o familie oarecare a unei mulţimi parţial ordonate $(A, \leq)$.

Un element $y \in A$ este un majorant al familiei $(x_i)_{i \in I}$ dacă $x_i \leq y$ pentru orice $i \in I$.

$y \in A$ este supremumul familiei $(x_i)_{i \in I}$ dacă pentru orice majorant z al familiei $(x_i)_{i \in I}$ avem $y \leq z$. Supremumul familiei $(x_i)_{i \in I}$ va fi notat: $\bigvee_{i \in I} x_i$.

Deci elementul $\bigvee_{i \in I} x_i$ al lui A este caracterizat de următoarele două relaţii:

(i) $x_i \leq \bigvee_{i \in I} x_i$, pentru orice $i \in I$.

(ii) Dacă $x_i \leq y$ pentru orice $i \in I$, atunci $\bigvee_{i \in I} x_i \leq y$.

Dual, $y \in A$ este infimimul familiei $(x_i)_{i \in I}$ dacă pentru orice minorant z al familiei $(x_i)_{i \in I}$ avem $z \leq y$. Infimimul familiei $(x_i)_{i \in I}$ va fi notat: $\bigwedge_{i \in I} x_i$ şi este caracterizat de

(a) $\bigwedge_{i \in I} x_i \leq x_i$, pentru orice $i \in I$.

(b) Dacă $y \leq x_i$ pentru orice $i \in I$, atunci $y \leq \bigwedge_{i \in I} x_i$.

Supremumul (respectiv infimimul) familiei $\{x_1,\ldots,x_n\}$ se va nota $\bigvee_{i=1}^{n} x_i$ (respectiv $\bigwedge_{i=1}^{n} x_i$).

Pentru mulţimea $\{x, y\}$ notăm:
$x \vee y$ supremumul mulţimii $\{x, y\}$.
$x \wedge y$ infimumul mulţimii $\{x, y\}$.

Definiţia 1.6.2: O mulţime preordonată $(A, \leq)$ se numeşte **latice** dacă pentru orice $x, y \in A$ există $x \vee y$ şi $x \wedge y$. $(A, \leq)$ se numeşte **latice completă** dacă pentru orice familie $(x_i)_{i \in I}$ de elemente ale lui A, există $\bigvee_{i=1}^{n} x_i$ şi $\bigwedge_{i=1}^{n} x_i$.

O mulţime parţial ordonată $(A, \leq)$ se numeşte **inductivă** dacă orice submulţime total ordonată a sa admite cel puţin un majorant.

Fie $(A, \leq)$ o mulţime parţial ordonată. Un element $x \in A$ se numeşte **maximal** dacă nu există nici un element $y \in A$ astfel încât $x < y$; cu alte cuvinte, dacă din $x \leq y$ rezultă $x = y$.

Axioma lui Zorn: Orice mulţime parţial ordonată inductivă admite un element maximal.

Observaţia 1.6.3: Această teoremă a fost impusă de o serie de construcţii ale matematicii care vizează mulţimile infinite. Cunoscută mai ales sub o formă echivalentă (axioma alegerii), ea a generat multe controverse în matematică şi în filosofia matematicii. În prezent situaţia este următoarea:

Pentru teoria mulţimilor s-au propus mai multe sisteme de axiome, mai cunoscute fiind sistemul Zermelo-Fraenkel şi sistemul Gődel-Bernays. Nu s-a reuşit până acum să se demonstreze nici unul din aceste sisteme că este necontradictoriu.

Presupunându-se că sistemul de axiome Zermelo-Fraenkel este necontradictoriu, Kurt Gődel a demonstrat în 1940 că prin adăugarea axiomei lui Zorn se obţine încă un sistem necontradictoriu. Ulterior s-a demonstrat că dacă adăugăm la sistemul Zermelo-Fraenkel negaţia axiomei lui Zorn se obţine încă un sistem necontradictoriu (A. Mostowski, P. Cohen).

Cu alte cuvinte, axioma lui Zorn este independentă de celelalte axiome ale teoriei mulţimilor. Independenţa axiomei lui Zorn este unul din rezultatele de vârf ale matematicii secolului XX.

Se cuvine a preciza că cea mai mare parte a matematicienilor contemporani presupun în cercetările lor că axioma lui Zorn este verificată.

Lecţia 2- Algebre Boole

Teoria algebrelor Boole s-a născut ca urmare a descoperirii că între legile logicii şi anumite legi ale calculului algebric există o perfectă analogie. Această descoperire este unanim atribuită lui George Boole (*An investigation into the laws of thought,* 1854)

Dintre matematicienii care au adus contribuţii mari la dezvoltarea teoriei algebrelor Boole trebuie menţionat în primul rând M.H. Stone pentru celebra sa teoremă de reprezentare (*The theory of representation for Boolean algebras,* Trans. A.M.S., 40 , 1936, p. 37-111) şi pentru teoria dualităţii a algebrelor Boole. (*Applications of the theory of Boolean rings to general topology,* Trans. A.M.S., 41, 1937, p. 375-481). De asemenea A. Tarski a obtinut rezultate remarcabile atât pe linia algebrică a acestui domeniu, cât mai ales pe linia legăturilor sale cu logica.

Algebrele Boole constituie reflectarea algebrică a calculului propoziţional, fiind modele algebrice ale calculului propoziţional, afirmaţie ce va fi precizată în capitolul următor. În capitolul IV, metodele folosite pentru demonstrarea completitudinii sistemului formal al calculului predicatelor se vor baza în întregime pe algebrele Boole.

Astăzi, teoria algebrelor Boole se prezintă ca un fragment important al algebrei, având puternice conexiuni cu logica, dar fiind un capitol de sine stătător, atât prin rezultatele obţinute în interiorul său cât şi prin aplicaţiile sale în topologie, analiză, calculul probabilităţilor, etc.

Este de remarcat faptul că, cele mai spectaculoase aplicaţii ale algebrelor Boole s-au obţinut în domeniul calculatoarelor electronice şi al disciplinelor învecinate.

Paragraful 2.1 al acestui capitol prezintă o serie de proprietăţi generale ale laticilor, care sunt structuri mai generale decât algebrele Boole.

În § 2.2 se dau o serie de definiţii legate de algebrele Boole, se studiază legătura cu inelele Boole, precum şi câteva proprietăţi ale morfismelor de algebre Boole.

Conguenţele, filtrele şi algebrele Boole cât fac obiectul paragrafului 2.3. Paragraful 2.4 este foarte important conţinând, teoria ultrafiltrelor şi demonstraţia teoremei lui Stone. Algebrele Boole finite sunt prezentzate în § 2.5.

2.1. *Latici*

În acest paragraf vom stabili o serie de proprietăţi ale laticilor şi ale laticilor distributive. Să ne reamintim definiţia laticei dată în lecţia 1.1.

O mulţime preordonată $(A, \leq)$ se numeşte **latice** dacă pentru orice $x, y \in A$ există $x \vee y$ şi $x \wedge y$. $(A, \leq)$ se numeşte **latice completă** dacă pentru orice familie $(x_i)_{i \in I}$ de elemente ale lui A, există $\bigvee_{i=1}^{n} x_i$ şi $\bigwedge_{i=1}^{n} x_i$.

Propoziţia 2.1.1: Într-o latice oarecare L sunt verificate următoarele proprietăţi:

(L_1)	$a \wedge a = a,$	$a \vee a = a$	(idempotenţa)
(L_2)	$a \wedge b = b \wedge a,$	$a \vee b = b \vee a$	(comutativitatea)
(L_3)	$(a \wedge b) \wedge c = a \wedge (b \wedge c)$		(asociativitatea)
	$(a \vee b) \vee c = a \vee (b \vee c)$		
(L_4)	$a \wedge (a \vee b) = a,$	$a \vee (a \wedge b) = a$	(absorbţie)

Demonstraţie: Aceste relaţii sunt imediate, pe baza definiţiei infimumului si supremumului. Spre exemplu, să arătăm că $a \wedge (a \vee b) = a$. Conform definiţiei infimumului, va trebui să demonstrăm că:

$a \leq a,\ a \leq a \vee b$

$z \leq a,\ z \leq a \vee b \Rightarrow z \leq a \wedge (a \vee b)$

Se observă însă că aceste relaţii sunt evidente.

Vom stabili acum un rezultat care arată că egalităţile(L_1)-(L_4) caracterizează o latice.

Propoziţia 2.1.2: Fie L o mulţime nevidă oarecare înzestrată cu două operaţii binare $\vee, \wedge$ astfel încât orice elemente $a,b,c \in L$ verifică egalităţile (L_1)-(L_4). Atunci pe mulţimea L se poate defini o relaţie de ordine parţială $\leq$ prin

$a \leq b \Leftrightarrow a \wedge b = a,$

astfel încât $a \wedge b$ (respectiv $a \vee b$) este infimumul (respectiv supremumul) mulţimii $\{a, b\}$ în sensul ordinii astfel definite.

Demonstratie: Verificăm întâi că $\leq$ este o relaţie de ordine parţială:

$a \leq a$ rezultă din $a \wedge a = a$

$a \leq b,\ b \leq a \quad \Rightarrow a = a \wedge b,\ b = b \wedge a \Rightarrow a = b$

$a \leq b,\ b \leq c \quad \Rightarrow a = a \wedge b,\ b = b \wedge c$

$\Rightarrow a = a \wedge b = a \wedge (b \wedge c) = (a \wedge b) \wedge c = a \wedge c$

$\Rightarrow a \leq c$

Pentru a arăta că $a \wedge b$ este infimumul mulţimii $\{a, b\}$ va trebui să stabilim relaţiile:

$a \wedge b \leq a,\ a \wedge b \leq b$

$x \leq a,\ x \leq b \Rightarrow x \leq a \wedge b.$

Primele două relaţii rezultă din

$(a \wedge b) \wedge a = a \wedge (a \wedge b) = (a \wedge a) \wedge b = a \wedge b$

$(a \wedge b) \wedge b = a \wedge (b \wedge b) = a \wedge b.$

Cealaltă relaţie rezultă conform implicaţiei

$x \wedge a = x,\ x \wedge b = x \Rightarrow x \wedge (a \wedge b) = (x \wedge a) \wedge b = x \wedge b = x$

Vom arăta arăta acum că $a \vee b$ este supremumul mulţimii $\{a, b\}$.

$a \leq a \vee b$ rezultă din(L_4): $a \wedge (a \vee b) = a$ şi analog se deduce că şi $b \leq a \vee b$.

Restul rezultă conform implicaţiilor:

$$a \le x,\ b \le x \quad \Rightarrow a \wedge x = a,\ b \wedge x = b$$
$$\Rightarrow a \vee x = (a \wedge x) \vee x = x, \qquad b \vee x = (b \wedge x) \vee x = x$$
$$\Rightarrow (a \vee b) \wedge x = (a \vee b) \wedge (a \vee x) = (a \vee b) \wedge [a \vee (b \vee x)] =$$
$$\Rightarrow (a \vee b) \wedge [(a \vee b) \vee x] = a \vee b \Rightarrow a \vee b \le x.$$

Cu aceasta, demonstraţia este terminată.

Indicăm cititorului să pună în evidenţă toate toate punctele demonstraţiei în care am folosit relaţiile (L1)-(L4).

Observaţi 2.1.2.1: Relaţia de ordine din propoziţia precedentă pote fi definită în mod echivalent şi prin $a \le b \Leftrightarrow a \vee b = b$.

Într-o latice avem implicaţiile:

$$x \le y \quad \Rightarrow a \wedge x \le a \wedge y, \qquad a \vee x \le a \vee y$$
$$x \le y,\ a \le b \quad \Rightarrow x \wedge a \le y \wedge b, \qquad x \vee a \le y \vee b$$

Stabilirea lor este imediată.

Operaţiile unei latici finite pot fi descrise prin tabele.

Spre exemplu, în mulţimea $L = \{0, a, b, 1\}$ putem defini două operaţii de latice în felul următor:

$\wedge$	**0**	***a***	***b***	**1**
0	0	0	0	0
a	0	*a*	0	*a*
b	0	0	*b*	*b*
1	0	*a*	*b*	1

$\vee$	**0**	***a***	***b***	**1**
0	0	*a*	*b*	1
a	*a*	*a*	1	1
b	*b*	1	*b*	1
1	1	1	1	1

Fie L_1, L_2 două latici. O funcţie $f : L_1 \to L_2$ se numeşte morfism de latici dacă pentru orice $x, y \in L_1$, avem

$$f(x \vee y) = f(x) \vee f(y)$$
$$f(x \wedge y) = f(x) \wedge f(y)$$

Un morfism bijectiv de latici $f : L_1 \to L_2$ se numeste izomorfism de latici. Se mai spune în acest caz că laticile L_1, L_2 sunt izomorfe.

Un element 0 al unei latici L se numeşte element prim dacă $0 \le x$, pentru orice $x \in L$. Dual, un element ultim al lui L este definit de: $x \le 1$, pentru orice $x \in L$.

Propoziţia 2.1.3: Într-o latice L sunt echivalente următoarele relaţii:

(i) $(x \wedge y) \vee (x \wedge z) = x \wedge (y \vee z)$ pentru orice $x, y, z \in L$.
(ii) $(x \vee y) \wedge (x \vee z) = x \vee (y \wedge z)$ pentru orice $x, y, z \in L$.
(iii) $(x \vee y) \wedge z \le x \vee (y \wedge z)$ pentru orice $x, y, z \in L$.

Demonstratie:

(i) $\Rightarrow$ (ii). Vom arăta că orice elemente $a, b, c \in L$ verifică (ii).

În (i) vom pune $x = a \vee b, y = a, z = c$:

$$(a \vee b) \wedge (a \vee c) = [(a \vee b) \wedge a] \vee [(a \vee b) \wedge c]$$
$$= a \vee [(a \vee b) \wedge c] \qquad \text{(conform } L_4\text{)}$$
$$= a \vee [(a \wedge c) \vee (b \wedge c)] \qquad \text{(conform (i))}$$
$$= [a \vee (a \wedge c)] \vee (b \wedge c)$$
$$= a \vee (b \wedge c) \qquad \text{(conform } L_4\text{)}$$

(ii) $\Rightarrow$ **(iii).** Din $z \le x \vee z$ rezultă:

$$(x \vee y) \wedge z \le (x \vee y) \wedge (x \vee z) = x \vee (y \wedge z)$$

(iii) $\Rightarrow$ **(i).** Fie $a, b, c \in L$ oarecare. În **(iii)** facem $x = a, y = b, z = a \vee c$:

$$(a \vee b) \wedge (a \vee c) \le a \vee [b \wedge (a \vee c)] = a \vee [(a \vee c) \wedge b]$$

Punând în **(iii)** $x = a, y = c, z = c$ rezultă:

$$(a \vee c) \wedge b \le a \vee (c \wedge b)$$

deci

$$a \vee [(a \vee c) \wedge b] \le a \vee [a \vee (c \wedge b)] = (a \vee a) \vee (b \wedge c) = a \vee (b \wedge c)$$

Din inegalităţile stabilite mai sus se obţine

$$(a \vee b) \wedge (a \vee c) \le a \vee (b \wedge c)$$

Va fi suficient să stabilim inegalitatea inversă, care este valabilă în orice latice:

$$a \vee (b \wedge c) \le (a \vee b) \wedge (a \vee c)$$

Din $a \le a \vee b$, $a \le a \vee c$ rezultă:

$$a \le (a \vee b) \wedge (a \vee c)$$

De asemenea, din $b \le a \vee b$, $c \le a \vee c$, rezultă:

$$b \wedge c \le (a \vee b) \wedge (a \vee c)$$

Din aceste două inegalităţi se obţine, conform definiţiei supremumului, exact inegalitatea căutată. Demonstraţia este terminată.

Definiţia 2.1.3.1. O latice L care satisface una din condiţiile echivalente **(i)** – **(iii)** se numeşte *latice distributivă*.

Fie L o latice cu element prim 0 şi cu element ultim 1. Un element $a \in L$ este un complement al lui $b \in L$ dacă:

$$a \wedge b = 0 \quad \text{şi} \quad a \vee b = 1.$$

Propoziţia 2.1.4. Într-o latice distributivă L orice element poate avea cel mult un complement.

Demonstraţie. Presupunem că b, c sunt două elemente ale lui L care sunt complemente ale lui a atunci ele verifică egalităţile:

$$a \wedge b = 0 \quad a \vee b = 1$$
$$a \wedge c = 0 \quad a \vee c = 1$$

Atunci avem

$$b = b \wedge 1 = b \wedge (a \vee c) = (b \wedge a) \vee (b \wedge c) = 0 \vee (b \wedge c) = b \wedge c .$$

Analog se arată că $c = (b \wedge c)$, deci $b = c$.

Într-o latice distributivă L(cu element prim 0 şi cu element ultim 1) vom nota cu $\neg a$ complementul unui element $a \in L$.

Propoziţia 2.1.5. Presupunem că în laticea distributivă L, pentru elementele a şi b există $\neg a$ şi $\neg b$. Atunci există şi $\neg(a \wedge b)$, $\neg(a \vee b)$ şi care sunt daţi de:

$$\neg(a \wedge b) = \neg a \vee \neg b; \quad \neg(a \vee b) = \neg a \wedge \neg b .$$

Demonstraţie: Conform Propoziţiei 2.1.4, pentru verficarea primei relaţii este suficient să arătăm că:

$$(a \wedge b) \wedge (\neg a \vee \neg b) = 0$$
$$(a \wedge b) \vee (\neg a \vee \neg b) = 1.$$

Aceste relaţii se obţin astfel:

$$(a \wedge b) \wedge (\neg a \vee \neg b) = (a \wedge b \wedge \neg a) \vee (a \wedge b \wedge \neg b) = 0 \vee 0 = 0$$
$$(a \wedge b) \vee (\neg a \vee \neg b) = (a \vee \neg a \vee \neg b) \wedge (b \vee \neg a \vee \neg b) = 1 \wedge 1 = 1.$$

Egalitatea a doua a propoziţiei se obţine în mod dual.

Observaţia 2.1.5.1. Pentru cunoaşterea în adâncime a problemelor fundamentale ale teoriei laticilor, indicăm următoarele cărţi de referinţă: G.Birkhoff, *Lattice theory*, American Math. Soc., 1967, (ediţia a 3-a) şi Grätzer, *Lattice theory* (*First concepts and distributive lattice*), San Francisco, 1971.

În cele ce urmează vom nota:

$$x \vee y \vee z = x \vee (y \vee z)$$
$$x \wedge y \wedge z = x \wedge (y \wedge z),$$

pentru orice elemente x, y, z ale unei latici L.

2.2. *Algebre Boole. Proprietăţi generale*

Definiţia 2.2.1. O algebră Boole este o latice distributivă B cu element prim 0 şi cu element ultim 1, astfel încât orice element $x \in B$ are un complement $\neg x$.

Exemplul 2.2.1.1. Mulţimea $L_2 = \{0, 1\}$ este o agebră Boole pentru ordinea naturală:

$$0 \leq 0,\ 0 \leq 1,\ 1 \leq 1.$$

Operaţiile lui L_2 sunt date de:

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

; $\neg 0 = 1$ şi $\neg 1 = 0$.

Exemplul 2.2.1.2. Mulţimea $\mathcal{P}(X)$ a părţilor unei mulţimi nevide X este o algebră Boole în care relaţia de ordine $\leq$ este incluziunea $\subseteq$. Operaţiile lui $\mathcal{P}(X)$ vor fi:

$$A \vee B = A \cup B$$
$$A \wedge B = A \cap B$$
$$\neg A = C_X(A).$$

pentru orice $A, B \in \mathcal{P}(X)$, $\varnothing$ este element prim şi X este elementul ultim al lui $\mathcal{P}(X)$.

Observaţia 2.2.1.3. Dacă B, B' sunt două algebre Boole, atunci un morfism de algebre Boole este o funcţie $f : B \to B'$ care satisface proprietăţile următoare, pentru orice $x, y \in B$:

$$f(x \vee y) = f(x) \vee f(y)$$
$$f(x \wedge y) = f(x) \wedge f(y)$$
$$f(\neg x) = \neg f(x)$$

(1) Orice morfism de algebre Boole $f : B \to B'$ verifică condiţiile:

$$f(0) = 0 \ ; \ f(1) = 1.$$

Cum $B \neq \varnothing$, atunci există $x \in B$, deci vom putea scrie:

$$f(0) = f(x \wedge \neg x) = f(x) \wedge \neg f(x) = 0$$
$$f(1) = f(x \vee \neg x) = f(x) \vee \neg f(x) = 1$$

(2) Orice morfism de algebre Boole este o aplicaţie izotonă:

$$x \leq y \Rightarrow x \wedge y = y \Rightarrow f(x) \wedge f(y) = f(y) \Rightarrow f(x) \leq f(y).$$

În cele ce urmează vom arăta că algebrele Boole sunt echivalente cu o clasă de inele comutative numite inele Boole.

Definiţia 2.2.2. Se numeşte inel Boole orice inel unitar $(A, +, *, 0, 1)$ cu proprietatea că:

$$x^2 = x \text{ , pentru orice } x \in A.$$

Lema 2.2.1. Pentru orice două elemente x, y ale unui inel Boole A, avem relaţiile:

$x + x = 0$

$xy = yx$ (comutativitate)

Demonstraţie. Din

$$x + y = (x + y)^2 = (x + y)(x + y) = x^2 + xy + xy + y^2 = x + xy + yx + y$$

rezultă

$xy + yx = 0.$

Făcând $y = x$, se obţine $x^2 + x^2 = 0$, deci $x + x = 0$.

Pentru orice $z \in A$, vom avea deci $z + z = 0$, adică $z = -z$. Luând $z = xy$, rezultă $xy + xy = 0$ deci $xy = - xy$. Din relaţia stabilită mai sus avem însă că $yx = -xy$ şi deci

$$xy = - xy = yx.$$

Dacă A, A' sunt două inele Boole, atunci un izomorfism de inele Boole $g : A \to A'$ este o funcţie $g : A \to A'$ cu proprietaţile următoare:

$g(x + y) = g(x) + g(y)$

$g(x y) = g(x)g(y)$

$g(1) = 1$

pentru orice $x, y \in A$. Cu alte cuvinte, este un morfism de inele unitare.

Propoziţia 2.2.1. Dacă A este un inel Boole, atunci A poate fi organizat ca o algebră Boole $F(A)$:

$$\begin{cases} x \vee y = x + y + xy \\ x \wedge y = xy \\ \neg x = x + 1 \\ 0 \text{ este element prim al lui } F(A) \\ 1 \text{ este elementul ultim al lui } F(A) \\ x \le y \Leftrightarrow xy = x \end{cases}$$

Demonstraţie. Operaţiile astfel definite verifică axiomele (L_1)-(L_4) din §2.1. Spre exemplu, să arătăm că $x \vee (x \wedge y) = x$, pentru orice $x \in A$.

$x \vee (x \wedge y) = x + xy + xxy = x + xy + x^2y = x + (xy + xy) = x + 0 = 0$, comform Lemei 2.2.1. Deci $F(A)$ este o latice. Printr-un calcul simplu se poate arăta că $F(A)$ este distributivă şi că:

$0 \le x, x \le 1,$ pentru orice $x \in A$.

Să arătăm că $x + 1$ verifică proprietăţile complementului:

$x \vee (x + 1) = x + x + 1 + x(x + 1) = 2x + 1 + x^2 + x = 0 + 1 + (x + x) = 1 + 0 = 1$

$x \wedge (x + 1) = x(x + 1) = x^2 + x = x + x = x.$

Propoziţia 2.2.2. Dacă B este o algebră Boole, atunci B poate fi organizată ca un inel Boole $G(B)$ punând:

$x + y = (x \wedge \neg y) \vee (\neg x \wedge y)$

$xy = x \wedge y$

pentru orice $x, y \in B$. 0 şi 1 vor avea semnificaţia naturală.

Demonstraţia este calculatorie şi o lăsăm pe seama cititorului.

Propoziţia 2.2.3. (i) Dacă $f : A \to A'$ este un morfism de inele Boole, atunci f este şi un morfism de algebre Boole $f : F(A) \to F(A')$.

(ii) Dacă $g : B \to B'$ este un morfism de inele Boole, atunci g este şi un morfism de algebre Boole $g : G(B) \to G(B')$.

Propoziţia 2.2.4. Dacă A este un inel Boole şi B este algebră Boole, atunci:

(i) A şi $G(F(A))$ coincid ca inele Boole.

(ii) B şi $F(G(B))$ coincid ca algebre Boole.

Demonstraţia celor două propoziţii este un exerciţiu util.

Într-o algebră Boole B se defineşte operaţia de implicaţie booleană:

$$x \to y = \neg x \vee y, \qquad x, y \in B$$

şi operaţia de echivalenţă booleana:

$$x \leftrightarrow y = (x \to y) \wedge (y \to x), \quad x, y \in B.$$

Se poate arăta că $x \to y = 1$ dacă şi numai dacă $x \le y$.

Aceste două operaţii au proprietăţile următoare:

$$x \to (y \to x) = 1$$
$$(x \to (x \to y)) \to (x \to y) = 1$$
$$(x \to y) \to ((y \to z) \to (x \to z) = 1$$
$$(x \leftrightarrow y) \to (x \to y) = 1$$
$$(x \leftrightarrow y) \to (y \to x) = 1$$
$$(x \to y) \to ((y \to x) \to (x \leftrightarrow y)) = 1$$
$$(\neg y \to \neg x) \to (x \to y) = 1$$
$$(x \vee y) \leftrightarrow (\neg x \to y) = 1$$
$$(x \wedge y) \to (\neg x \vee \neg y) = 1$$
$$(x \leftrightarrow y) = 1 \Leftrightarrow x = y.$$

Să stabilim, de exemplu proprietatea a doua:

$$(x \to (x \to y)) \to (x \to y) = \neg (x \to (x \to y)) \vee (x \to y)$$
$$= \neg(\neg x \vee \neg x \vee y) \vee (\neg x \vee y)$$
$$= \neg(\neg x \vee y) \vee (\neg x \vee y) = 1$$

Lema 2.2.2: În orice algebră Boole B avem:

(i) $\neg\neg x = x$

(ii) $x \le y \Leftrightarrow \neg y \le \neg x$

(iii) $x \le y \Leftrightarrow x \wedge \neg y = 0$

Demonstraţie.

(i) Rezultă din unicitatea complementului:

$\neg x \vee x = 1$,

$\neg x \wedge x = 0$.

(ii) $x \le y \Rightarrow x \wedge y = x \Rightarrow \neg x \vee \neg y = \neg x \Rightarrow \neg y \le \neg x$

$\neg y \le \neg x \Rightarrow \neg\neg x \le \neg\neg y$ (conform celor demonstrate)

$\Rightarrow x \le y$

(iii) $x \le y \Rightarrow x \wedge \neg y \le y \wedge \neg y = 0 \Rightarrow x \wedge \neg y = 0$

$x \wedge \neg y = 0 \Rightarrow x = x \wedge 1 = x \wedge (y \vee \neg y) = (x \wedge y) \vee (x \wedge \neg y)$

$\Rightarrow (x \wedge y) \vee 0 = x \wedge y$

$\Rightarrow x \le y$

Un morfism de algebra Boole $f : B \to B'$ se numeşte izomorfism de algebre Boole dacă este bijectiv.

Observaţia 2.2.2.1. Compunera a două morfisme (respective izomofisme) de algebre Boole este încă un morfism (respectiv izomorfism) de algebre Boole. Pentru orice algebră Boole B, aplicaţia $1_B : B \to B$ este un izomorfism de algebre Boole.

Propoziţia 2.2.5. Fie $f : B \to B'$ un morfism de algebre Boole. Sunt echivalente afirmaţiile următoare:

(i) f este izomorfism ;

(ii) f este surjectiv şi pentru orice $x, y \in B$, avem
$x \le y \Leftrightarrow f(x) \le f(y)$;

(iii) f este inversabilă şi f^{-1} este un morfism de algebre Boole.

Demonstraţie. (i) $\Rightarrow$ (ii). Amintim că orice morfism de algebre Boole este o aplicaţie izotonă.

$x \le y \Rightarrow f(x) \le f(y)$

Presupunem $f(x) \le f(y)$, deci:

$f(x) \le f(y) \Rightarrow f(x) \wedge f(y) = f(x) \Rightarrow f(x \wedge y) = f(x) \Rightarrow x \wedge y = x \Rightarrow x \le y$.

Am aplicat injectivitatea lui f.

(ii) $\Rightarrow$ (i). Trebuie să arătăm că f este injectivă:

$f(x) = f(y) \Rightarrow f(x) \le f(y)$ şi $f(y) \le f(x)$

$\Rightarrow x \le y$ şi $y \le x \Rightarrow x = y$.

(i) $\Rightarrow$ (iii). Este suficient să arătăm că f^{-1} este morfism de algebre Boole.

Fie $y, y' \in B'$ şi $x = f^{-1}(y) \in B$, $x' = f^{-1}(y') \in B$. Cum f este morfism, rezultă:

$f(x \vee x') = f(x) \vee f(x')$

Dar $f(x) = y$, $f(x') = y'$, deci $f(x \vee x') = y \vee y'$, de unde prin aplicarea lui f^{-1} rezultă:

$f^{-1}(f(x \vee x')) = f^{-1}(y \vee y')$, deci:

$f^{-1}(y \vee y') = x \vee x' = f^{-1}(y) \vee f^{-1}(y')$

Analog se arată că :

$f^{-1}(y \wedge y') = f^{-1}(y) \wedge f^{-1}(y')$

$f^{-1}(\neg y) = \neg f^{-1}(y)$.

(iii) $\Rightarrow$ (i). Evidentă.

Definiţia 2.2.3. O submulţime nevidă B' a unei algebre Boole B se numeşte subalgebră Boole a lui B dacă:

$x, y \in B' \Rightarrow x \wedge y \in B'$ şi $x \vee y \in B'$

$x \in B' \Rightarrow \neg x \in B'$.

Observaţia 2.2.3.1. Dacă B' este subalgebră Boole a lui B, atunci $0 \in B$ şi $1 \in B$:
Într-adevăr, cum $B' \neq \emptyset$, există $x \in B'$, deci:

$0 = x \wedge \neg x \in B'$; $1 = x \vee \neg x \in B'$.

Propoziţia 2.2.6. Dacă $f : B \to B'$ este un morfism de algebre Boole şi B_1 este o subalgebră Boole a lui B, atunci $f(B_1)$ este o subalgebră Boole a lui B'. În particular, imaginea $f(B)$ a lui B prin f este o subalgebră Boole a lui B'.

Demonstraţia este imediată.

Observaţia 2.2.6.1. Dacă $f : B \to B'$ este un morfism de algebre Boole injectiv atunci B este izomorfă cu subalgebra Boole $f(B)$ a lui B'.

Este util să observăm că un morfism de algebre Boole $f : B \to B'$ verifică relaţiile:

$f(x \to y) = f(x) \to f(y)$,

$f(x \leftrightarrow y) = f(x) \leftrightarrow f(y)$,

pentru orice $x, y \in B'$.

Exerciţiu. Pentru ca funcţia $f : B \to B'$ să fie morfism de algebre Boole este necesar şi suficient ca să avem:

$f(x \vee y) = f(x) \vee f(y)$, $\quad x, y \in B$

$f(x \wedge y) = f(x) \wedge f(y)$, $\quad x, y \in B$

$f(1) = 1$; $f(0) = 0$.

2. 3. *Filtre. Algebre Boole cât.*

Definiţia 2.3.1. Fie B o algebră Boole oarecare. O submulţime nevidă F a lui B se numeşte filtru, dacă pentru orice $x, y \in B$ avem :

(a) $x, y \in F \Rightarrow x \wedge y \in F$

(b) $x \in F, x \leq y \Rightarrow y \in F$

Dual, un ideal al lui B este o submulţime nevidă I a lui B pentru care:

(a') $x, y \in I \Rightarrow x \vee y \in I$

(b') $y \in I, x \leq y \Rightarrow x \in I$

Observaţia 2.3.1.1: Pentru orice filtru F, $I \in F$.

Filtrele unei algebre Boole B se pot pune în corespondenţă bijectivă cu idealele sale. Unui filtru F i se asociază idealul

$I_F = \{x \in B | \neg x \in F\}$,

iar idealului I i se asociază filtrul

$F_I = \{y \in B | \neg y \in I\}$.

Se observă cu uşurinţa că funcţiile $F \to I_F$ şi $I \to F_I$ sunt inverse una celeilalte.

Conform acestei observaţii, vom studia numai filtrele unei algebre Boole. Pentru ideale, proprietăţile respective se vor enunţa prin dualizare.

Definiţia 2.3.2. Fie B o algebră Boole. O relaţie de echivalenţă $\sim$ pe B se numeşte congruenţă dacă:

$x \sim y, x' \sim y' \Rightarrow x \vee x' \sim y \vee y'$

$x \sim y, x' \sim y' \Rightarrow x \wedge x' \sim y \wedge y'$

$x \sim y \quad \Rightarrow \quad \neg x \sim \neg y$.

Obsevaţia 2.3.2.1. Dacă $\sim$ este o congruenţă pe B atunci:

$$x \sim y, \quad x' \sim y' \quad \Rightarrow \quad \begin{cases} (x \to x') \sim (y \to y') \\ (x \leftrightarrow x') \sim (y \leftrightarrow y') \end{cases}$$

Propoziţia 2.3.1. Filtrele unei algebre Boole B sunt în corespondenţă bijectivă cu congruenţele sale.

Demonstraţie: Fiecărui filtru F al lui B îi asociem următoarea relaţie binară pe B:

$x \sim_F y \Leftrightarrow (x \leftrightarrow y) \in F$.

Această relaţie se poate scrie echivalent:

$x \sim_F y \Leftrightarrow (x \to y) \in F$ şi $(y \to x) \in F$.

Într-adevăr, aplicăm proprietăţile filtrului şi $x \leftrightarrow y = (x \to y) \wedge (y \to x)$:

$(x \to y) \in F, (y \to x) \in F \Rightarrow (x \to y) \wedge (y \to x) \in F \Rightarrow (x \leftrightarrow y) \in F$

$(x \leftrightarrow y) \in F, (x \leftrightarrow y) \leq (x \to y) \Rightarrow (x \to y) \in F$

şi analog

$(x \leftrightarrow y) \in F \Rightarrow (y \rightarrow x) \in F$.

Vom arăta că $\sim_F$ este o relaţie de echivalenţă:

$x \sim_F x$: deoarece $(x \leftrightarrow x) = 1 \in F$.

$x \sim_F y \Rightarrow (x \leftrightarrow y) \in F \Rightarrow (y \leftrightarrow x) \in F \Rightarrow y \sim_F x$,

folosind egalitatea evidentă $(x \leftrightarrow y) = (y \leftrightarrow x)$.

$x \sim_F y$, $y \sim_F z \Rightarrow (x \leftrightarrow y) \in F$ şi $(y \leftrightarrow z) \in F \Rightarrow (x \leftrightarrow y) \wedge (y \leftrightarrow z) \in F$

Dar

$$\neg x \vee z = \neg x \vee (y \wedge \neg y) \vee z =$$
$$= (\neg x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z) \geq (\neg x \vee y) \wedge (\neg y \vee z) ,$$

deci avem:

$$(x \rightarrow z) \geq (x \rightarrow y) \wedge (y \rightarrow z)$$

În mod analog obţinem:

$$(z \rightarrow x) \geq (z \rightarrow y) \wedge (y \rightarrow x)$$

Din ultimile două relaţii se obţine:

$$(x \rightarrow z) \wedge (z \rightarrow x) \geq (x \rightarrow y) \wedge (y \rightarrow z) \wedge (z \rightarrow y) \wedge (y \rightarrow x)$$

sau

$$x \leftrightarrow y \geq (x \leftrightarrow y) \wedge (y \leftrightarrow z) \in F$$

Rezultă $(x \leftrightarrow z) \in F$, deci $x \sim_F z$.

Relaţia de echivalenţă $\sim_F$ este o congruenţă:

$$\left.\begin{array}{l} x \sim_F y \\ x' \sim_F y' \end{array}\right\} \Rightarrow \left\{\begin{array}{l} x \vee x' \sim_F y \vee y' \\ x \wedge x' \sim_F y \wedge y' \end{array}\right.$$

şi

$$x \sim_F y \quad \Rightarrow \quad \neg x \sim_F \neg y$$

Presupunând că $x \sim_F y$, $x' \sim_F y'$, avem $(x \leftrightarrow y) \in F$, $(x' \leftrightarrow y') \in$F, deci $(x \leftrightarrow y) \wedge (x' \leftrightarrow y') \in F$.

Avem inegalităţile:

$$(x \rightarrow y) \wedge (x' \rightarrow y') = (\neg x \vee y) \wedge (\neg x' \vee y') \leq (\neg x \vee y \vee y') \wedge (\neg x' \vee y \vee y') =$$
$$= (\neg x \wedge \neg x') \vee (y \vee y') = \neg(x \vee x') \vee (y \vee y') = (x \vee x') \rightarrow (y \vee y')$$

şi analog

$$(y \rightarrow x) \wedge (y' \rightarrow x') \leq (y \vee y') \rightarrow (x \vee x')$$

Din aceste două inegalităţi rezultă:

$$(x \rightarrow y) \wedge (y \rightarrow x) \wedge (x' \rightarrow y') \wedge (y' \rightarrow x') \leq [(x \vee x' \rightarrow (y \vee y')] \wedge [(y \vee y') \rightarrow (x \vee x')]$$

adică

$$(x \leftrightarrow y) \wedge (x' \leftrightarrow y') \leq (x \vee y) \leftrightarrow (x' \vee y').$$

Va rezulta $[(x \vee y) \leftrightarrow (x' \vee y')] \in F$, deci $x \vee y \sim_F x' \vee y'$.

Presupunem acum că $x \sim_F y$, deci

$$(\neg x \vee y) \wedge (\neg y \vee x) = (x \leftrightarrow y) \in F,$$

de unde rezultă:

$$(\neg x \leftrightarrow \neg y) = (\neg\neg x \vee \neg y) \wedge (\neg\neg y \vee \neg x) = (x \vee \neg y) \wedge (y \vee \neg x) = (x \leftrightarrow y) \in F.$$

Aşadar am arătat că $\neg x \sim_F \neg y$.

Fie $x \sim_F y$ şi $x' \sim_F y'$. Conform celor arătate, $\neg x \sim_F \neg y$ şi $\neg x' \sim_F \neg y'$, deci:

$(\neg x \vee \neg x') \sim_F (\neg y \vee \neg y')$

$\neg(x \wedge x') \sim_F \neg(y \wedge y')$

Din acestea se obţine $\neg\neg(x \wedge x') \sim_F \neg\neg(y \wedge y')$, adică $x \wedge x' \sim y \wedge y'$. Cu aceasta am stabilit că $\sim_F$ este o congruenţă.

Reciproc, unei congruenţe $\sim$ îi asociem filtrul:

$$\tilde{F} = \{x \in B \mid x \sim 1\}.$$

Într-adevăr, $\tilde{F}$ este filtru:

$$x, y \in \tilde{F} \Rightarrow x \sim 1, y \sim 1 \Rightarrow x \wedge y \sim 1 \wedge 1 \Rightarrow x \wedge y \sim 1 \Rightarrow x \wedge y \in \tilde{F}$$

$$x \le y, x \in \tilde{F} \Rightarrow x \vee y = y, x \sim 1 \Rightarrow y = x \vee y \sim 1 \vee y = 1 \text{ (pentru că } y \sim y)$$

$$\Rightarrow y \in \tilde{F}.$$

Observăm că $F \neq \varnothing$, deoarece $1 \sim 1 \Rightarrow 1 \in \tilde{F}$.

Dacă $\mathcal{F}_B$ este mulţimea filtrelor lui B şi $\mathcal{C}_B$ este mulţimea comgruenţelor lui B, atunci considerăm aplicaţiile:

$\Phi : \mathcal{F}_B \to \mathcal{C}_B$, $\Psi : \mathcal{C}_B \to \mathcal{F}_B$ definite astfel:

$\Phi(F) = \sim_F$, pentru orice $F \in \mathcal{F}_B$

$\Psi(\sim) = \tilde{F}$, pentru orice $\sim$ din $\mathcal{C}_B$.

Vom arăta că Φ, Ψ sunt inverse una celeilalte:

$\Psi(\Phi(F)) = F$

$\Phi(\Psi(\sim)) = \sim$

Într-adevăr avem relaţiile:

$$\Psi(\Phi(F)) = \Psi(\sim_F) = \{x \mid x \sim_F 1\} = \{x \mid (x \leftrightarrow 1) \in F\} = \{x \mid x \in F\} = F,$$

deoarece $(x \leftrightarrow 1) = (\neg x \vee 1) \wedge (0 \vee x) = 1 \wedge x = x$

Pentru stabilirea celeilalte relaţii, observăm că $\Phi(\Psi(\sim)) = \sim_F$, deci

$$x \sim_F y \Leftrightarrow (x \leftrightarrow y) \in \tilde{F}$$

Dacă $(x \leftrightarrow y) \in \tilde{F}$, atunci $(x \to y) \in \tilde{F}$ şi $(y \to x) \in \tilde{F}$. Conform proprietăţilor congruenţelor avem:

$$(x \to y) \in \tilde{F} \Rightarrow \neg x \vee y \sim 1 \Rightarrow (\neg x \vee y) \wedge x \sim 1 \wedge x$$

$$\Rightarrow (\neg x \wedge y) \vee (x \wedge y) \sim x \Rightarrow x \wedge y \sim x$$

şi analog

$$(y \to x) \in \tilde{F} \Rightarrow x \wedge y \sim y$$

Din $x \wedge y \sim x$, $x \wedge y \sim y$, rezultă $x \sim y$. Aşadar a rezultat $x \sim_{\tilde{F}} y \Rightarrow x \sim y$

Reciproc,

$$x \sim y \Rightarrow (x \leftrightarrow y) \sim (y \leftrightarrow y) \Rightarrow (x \leftrightarrow y) \sim 1 \Rightarrow (x \leftrightarrow y) \in \tilde{F} \Rightarrow x \sim_{\tilde{F}} y,$$

deoarece $y \leftrightarrow y = (\neg y \vee y) = 1$.

Am arătat că

$x \sim_{\tilde{F}} y \Leftrightarrow x \sim y$, adică $\Phi(\Psi(\sim)) = \sim$. Demonstraţia este încheiată.

Fie F un filtru în algebra Boole B. Considerând <u>mulţimea cât</u> $B/_F$ înzestrată cu operaţiile:

$$\hat{x} \vee \hat{y} = x \vee y$$

$$\hat{x} \wedge \hat{y} = x \wedge y$$

$$\neg \hat{x} = \neg x$$

şi cu elementul $\hat{0}$ şi $\hat{1}$.

Conform proprietăţilor congruenţei, aceste definiţii nu depind de reprezentanţi:

$$\left.\begin{matrix} x \sim_F x' \\ y \sim_F y' \end{matrix}\right\} \Rightarrow \begin{cases} (x \vee y) \sim_F (x' \vee y') \\ (x \wedge y) \sim_F (x' \wedge y') \end{cases}$$

$$x \sim_F x' \Rightarrow \neg x \sim_F \neg x'$$

Propoziţia 2.3.2. Mulţimea $B/_F = B/\sim_F$ înzestrată cu operaţiile de mai sus este o algebră Boole.

Demonstraţie: Direct din definiţia operaţiilor lui $B/_F$ şi din proprietăţile de algebră Boole a lui B.

$B/_F$ se numeşte algebra Boole cât a lui B prin filtrul F .

Se poate arăta că surjecţia canonică $p : B \to B/_F$ definită după cum ştim: $x \mapsto \hat{x}$, este un morfism de algebre Boole.

Propoziţia 2.3.3. Fie $f : B \to B'$ un morfism de algebre Boole.

(a) $F_f = \{x \in B \mid f(x) = 1\}$ este un filtru al lui B.

(b) f este injectivă $\Leftrightarrow F_f = \{1\} \Leftrightarrow \{ x \mid f(x) = 0 \} = \{ 0 \}$.

(c) $f(B)$ este o subalgebră Boole a lui B' izomorfă cu $B/_{F_f}$

Demonstraţie: (a) F_f are propietăţile filtrului:

$$x, y \in F_f \Rightarrow f(x \wedge y) = f(x) \wedge f(y) = 1 \wedge 1 = 1 \Rightarrow x \wedge y \in F_f.$$

$$x \le y, x \in F_f \Rightarrow 1 = f(x) \le f(y) \Rightarrow f(y) = 1 \Rightarrow y \in F_f$$

$$f(1) = 1 \Rightarrow 1 \in F_f \Rightarrow F_f \neq \varnothing.$$

(b) Presupunem f injectivă. Implicaţiile

$$x \in F_f \Rightarrow f(x) = 1 = f(1) \Rightarrow x = 1$$

ne dau incluziunea $F_f \subseteq \{1\}$. Cealaltă incluziune este evidentă.

Dacă $F_f = \{1\}$, atunci avem:

$$f(x) = f(y) \Rightarrow f(x \vee \neg y) = f(x) \vee \neg f(y) = 1$$
$$\Rightarrow x \vee \neg y = 1$$
$$\Rightarrow \neg x \wedge y = \neg(x \wedge \neg y) = 0$$
$$\Rightarrow y \le x$$

Analog se arată că:

$$f(x) = f(y) \Rightarrow x \le y,$$

deci $x = y$. Am demonstrat că f este injectivă. Cealaltă echivalenţă este evidentă.

(c) Considerăm aplicaţia $g : B/_{F_f} \to f(B)$ definită astfel :

$$g(\hat{x}) = f(x) \in f(B), \text{ pentru orice } \hat{x} \in B/_{F_f}.$$

Definiţia lui g nu depinde de reprezentanţi:

$$x \sim y \Rightarrow (x \leftrightarrow y) \in F_f$$
$$\Rightarrow (f(x) \leftrightarrow f(y)) = f(x \leftrightarrow y) = 1$$
$$\Rightarrow f(x) = f(y)$$

g este un morfism de algebre Boole :

$$g(\hat{x} \vee \hat{y}) = g(x \vee y) = f(x \vee y) = f(x) \vee f(y) = g(\hat{x}) \vee g(\hat{y})$$

Analog se arată că:

$$g(\hat{x} \wedge \hat{y}) = g(\hat{x}) \wedge g(\hat{y}) \; ; \; g(\neg \hat{x}) = \neg g(\hat{x}),$$

g este injectivă:

Conform (b), este suficient să arătăm că $F_g = \{1\}$

$$x \in F_g \Rightarrow g(\hat{x}) = 1 \Rightarrow f(x) = 1 \Rightarrow x \in F_f$$
$$\Rightarrow (x \leftrightarrow 1) = x \in F \Rightarrow x \sim_{F_f} 1 \Rightarrow \hat{x} = \hat{1}$$

Am arătat că $F_g \subseteq \{\hat{1}\}$, deci $F_g = \{1\}$, g este în mod evident şi surjectivă: pentru orice $y = f(x) \in f(B)$, avem elementul $\hat{x} \in B/_{F_f}$ pentru care

$$g(\hat{x}) = f(x) = y.$$

Corolarul 2.3.3.1: Dacă $f : B \to B'$ este un morfism surjectiv de algebre Boole, atunci B' este izomorfă cu $B/_{F_f}$.

2.4. *Teorema de reprezentare a lui Stone*

Scopul acestui paragraf este de-a demonstra că orice algebră Boole este izomorfă cu o algebră Boole ale cărei elemente sunt părţi ale unei mulţimi. Acest rezultat ocupă un loc central în teoria algebrelor Boole şi are importante aplicaţii în logică, calculul probabilităţilor, în topologie etc. Instrumentul principal folosit în demonstraţia acestei teoreme va fi conceptul de ultrafiltru.

Fie B o algebră Boole, fixată pentru întreg paragraful. Un filtru F al lui B este propriu dacă $F \neq B$.

Observaţie. F este propriu $\Leftrightarrow$ $0 \notin B$.

Propoziţia 2.4.1. Dacă $(F_i)_{i \in I}$ este o familie de filtre ale lui B, atunci $\bigcap_{i \in I} F_i$ este un filtru al lui B.

Demonstraţie: $x, y \in \bigcap_{i \in I} F_i \Rightarrow x, y \in F_i$, pentru orice $i \in I$

$$\Rightarrow x \wedge y \in F_i, \text{ pentru orice } i \in I \Rightarrow x \wedge y \in \bigcap_{i \in I} F_i$$

Analog se stabileşte şi cealaltă proprietate din definiţia unui filtru.

Definiţia 2.4.1. Dacă X este o submulţime, atunci filtrul generat de X este intersecţia tuturor filtrelor ce include pe X:

$$\{F \mid F \text{ filtru}, X \subset F\}$$

Filtrul generat de X va fi notat cu (X).

Propoziţia 2.4.2. Dacă $X \neq \Phi$, atunci

$$(X) = \{y \in B \mid \text{există } x_1,...,x_n \in X, \text{ astfel încât } x_1 \wedge ... \wedge x_n \leq y\}$$

Demonstraţie: Dacă F_0 este mulţimea din dreapta, va trebui să arătăm că:

(i) F_0 este filtru

(ii) $X \subset F_0$

(iii) Pentru orice filtru F al lui B, avem $X \subset F \Rightarrow F_0 \subset F$.

Dacă

$y_1, y_2 \in F_0$, atunci există:

$x_1,...,x_m \in X$ şi $z_1,...,z_n \in X$

astfel încât

$$x_1 \wedge ... \wedge x_m \leq y_1 \text{ şi } z_1 \wedge ... \wedge z_n \leq y_2$$

Rezultă

$$x_1 \wedge ... \wedge x_m \wedge z_1 \wedge ... \wedge z_n \leq y_1 \wedge y_2, \text{ deci } y_1 \wedge y_2 \in F_0.$$

Analog se arată:

$$y_1 \leq y_2, y_1 \in F \Rightarrow y_2 \in F_0,$$

deci F_0 este filtru.

Proprietatea (ii) este evidentă. Presupunem acum că F este un filtru astfel încât $X \subset F$, deci

$$y \in F_0 \Rightarrow \text{ există } x_1,...,x_n \in X, \quad x_1 \wedge ... \wedge x_n \leq y$$
$$\Rightarrow \text{ există } x_1,...,x_n \in F, \quad x_1 \wedge ... \wedge x_n \leq y$$
$$\Rightarrow y \in F,$$

ceea ce arată că $F_0 \subset F$. Propoziţia este demonstrată.

Fie $\mathcal{F}(B)$ mulţimea filtrelor propii ale lui B. $\mathcal{F}(B)$ este o mulţime parţial ordonată în raport cu incluziunea $\subset$.

Definiţia 2.4.2: Un element maximal al mulţimii parţial ordonate $(\mathcal{F}(B), \subset)$ se numeşte ultrafiltru.

Cu alte cuvinte, un ultrafiltru este un filtru propriu F al lui B cu proprietatea că pentru orice filtru propriu F' avem

$$F \subset F' \Rightarrow F = F'$$

Propozitia 2.4.3: Pentru orice filtru F există un ultrafiltru F_0 astfel încât $F \subset F_0$.

Demonstraţie. Fie Σ mulţimea filtrelor propri ale lui B ce include pe F.

$$\Sigma = \{F' \mid F' \text{ filtru propriu şi } F \subset F'\}$$

Cum $F \subset F$, avem $F \in \Sigma$, deci $\Sigma \neq \emptyset$. Considerăm mulţimea parţial ordonată $(\Sigma, \subset)$. Vom arăta că $(\Sigma, \subset)$ este inductivă. Pentru aceasta, fie $(F_i)_{i \in I}$ o familie total ordonată de filtre din Σ astfel încât:

$$\text{pentru orice } i, j \in I, \quad F_i \subset F_j \text{ sau } F_j \subset F_i.$$

Demonstrăm că familia $(F_i)_{i \in I}$ admite un majorant.

Fie $F' = \bigcup_{i \in I} F_i$. Atunci F' este filtru:

$$x, y \in F' \Rightarrow \exists\, i, j \in I, \text{ astfel încât } x \in F_i, \ y \in F_j.$$

Presupunând, de exemplu, $F_i \subset F_j$, rezultă $x \in F_j$, deci $x, y \in F_j$. Se deduce că $x \wedge y \in \bigcup_{i \in I} F_i = F'$.

Analog se stabileşte cealaltă proprietate din definiţia filtrului. Observăm că $F \subset F'$, deci $F' \in \Sigma$. Însă $F_i \subset F'$, pentru orice $i \in I$, deci F' este un majorant al familiei total ordonate $(F_i)_{i \in I}$. Aşadar $(\Sigma, \subset)$ este inductivă.

Aplicând axioma lui Zorn, rezultă existenţa unui element maximal al lui $(\Sigma, \subset)$, adică al unui ultrafiltru $F_0 \supset F$.

Observaţia 2.4.3.1: Este primul exemplu în care am folosit explicit axioma lui Zorn.

Corolarul 2.4.3.1: Dacă $x \neq 0$, atunci există un ultrafiltru F_0 astfel încât $x \in F_0$.

Demonstraţie: $F = \{y \in B \mid x \leq y\}$ este un filtru propriu al lui B.

Definiţia 2.4.3. Un filtru propriu F al lui B se numeste prim dacă :

$$x \vee y \in F \Rightarrow x \in F \text{ sau } y \in F.$$

Propoziţia următoare caracterizează ultrafiltrele algebrei Boole B.

Propoziţia 2.4.4. Fie F un filtru propriu al lui B. Sunt echivalente următoarele afirmaţii:

(i) F este ultrafiltru;
(ii) F este filtru prim;
(iii) Pentru orice $x \in B$, avem $x \in F$ sau $\neg x \in B$;
(iv) Algebra Boole cât $B/_F$ este izomorfă cu $L_2 = \{0, 1\}$.

Demonstraţie. (i) $\Rightarrow$ (ii). Presupunem prin absurd că F nu este prim, deci există $x,y \in B$, astfel încât $x \vee y \notin F$ dar $y \notin F$, $y \not\subset F$. Atunci:

$$F \subsetneq (F \cup \{x\}) \Rightarrow (F \cup \{x\}) = A \Rightarrow 0 \in (F \cup \{x\}).$$

Aplicând propoziţia 2.4.2, din $0 \in (F \cup \{x\})$ se deduce existenţa unui element $a \in F$, astfel încât $a \wedge x \leq 0$, deci $a \wedge x = 0$. Analog, există $b \in F$, astfel încât $b \wedge y = 0$. Rezultă:

$$0 = (a \wedge x) \vee (b \wedge y) = (a \vee b) \wedge (a \vee x) \wedge (x \vee b) \wedge (x \vee y)$$

Însă din relaţiile:

$$a \in F,\ b \in F \Rightarrow a \vee b \in F$$
$$a \in F,\ a \leq a \vee y \Rightarrow a \vee y \in F$$
$$b \in F,\ b \leq x \vee b \Rightarrow x \vee b \in F$$
$$x \vee y \in F$$

se obţine:

$$(a \vee b) \wedge (a \vee y) \wedge (x \vee b) \wedge (x \vee y) \in F,$$

deci $0 \in F$, ceea ce contrazice faptul că F este propriu. Deci F este propriu.

(ii) $\Rightarrow$ (iii) Din $x \vee \neg x = 1 \in F$, rezultă $x \in F$ sau $\neg x \in F$.

(iii)$\Rightarrow$ (iv) Aplicaţia $f : B \to L_2$, definită astfel:

$$f(x) = \begin{cases} 1, & dacă \quad x \in F \\ 0, & dacă \quad x \notin F \end{cases}$$

este un morfism de algebre Boole. Într-adevăr, avem:

$$f(x \wedge y) = 1 \Leftrightarrow x \wedge y \in F$$
$$\Leftrightarrow x \in F \text{ şi } y \in F \qquad (F \text{ este filtru})$$
$$\Leftrightarrow f(x) = 1 \text{ şi } f(y) = 1,$$

deci $f(x \wedge y) = f(x) \wedge f(y)$, pentru orice $x, y \in B$.

De asemenea:

$$f(\neg x) = 1 \Leftrightarrow \neg x \in F$$
$$\Leftrightarrow x \notin F \qquad (\text{conform (iii)}).$$
$$\Leftrightarrow f(x) = 0$$
$$\Leftrightarrow \neg f(x) = 1,$$

de unde rezultă $f(\neg x) = \neg f(x)$, pentru orice $x,\ y \in B$.

Cum $1 \in F$, avem $f(1) = 1$. Din $0 \notin F$, rezultă $f(0) = 0$. Pentru orice $x, y \in B$, vom avea:

$$f(x \vee y) = f(\neg(\neg x \wedge \neg y)) = \neg f(\neg x \wedge \neg y) = \neg\,(f(\neg x) \wedge f(\neg y)) =$$
$$= \neg\,(\neg f(x) \wedge \neg f(y)) = f(x) \vee f(y)$$

deci f este morfism de algebre Boole.

Cum $f(1) = 1$, $f(0) = 0$, f este surjectiv. Aplicând corolarul Propoziţiei 2.3.2 rezultă că B/F_f este izomorfă cu L_2.

Dar

$$F_f = \{x \in B \mid f(x) = 1\} = \{x \in B \mid x \in F\} = F,$$

deci $B/_F$ şi L_2 sunt izomorfe.

(iv) $\Rightarrow$ (i). Fie $B/_F \to L_2$ un izomorfism de algebre Boole. Presupunem prin absurd că F nu este propriu, deci $0 \in F$. Cum $(0 \leftrightarrow 1) = 0 \in F$, rezultă $0 \sim_F 1$, deci $\hat{0} = \hat{1}$. Am avea $f(\hat{0}) = f(\hat{1})$, deci $0 = 1$ în algebra Boole $\{0, 1\}$ ceea ce este absurd. Deci F este propriu.

Presupunem că există un filtru propriu F', astfel încât $F \subset_{\neq} F'$. Fie $x \in F'\text{-}F$.

Dacă $f(\hat{x}) = 1 = f(\hat{1})$, atunci $\hat{x} = \hat{1}$, deci $x = (x \leftrightarrow 1) \in F$, ceea ce este o contradicţie. Aşadar $f(\hat{x}) = 0 = f(\hat{0})$, deci $\hat{x} = \hat{0}$.

Rezultă $\neg x = (x \leftrightarrow 0) \in F \subset F'$

Din $x \in F'$, $\neg x \in F'$ se obţine $0 = x \wedge \neg x \in F$, ceea ce ar fi în contradicţie cu faptul că F este propriu. Deci F este ultrafiltru.

Suntem acum în măsură să demonstrăm teorema de reprezentare a lui Stone.

Propoziţia 2.4.5 (Stone). Pentru orice algebră Boole B, există o mulţime nevidă X şi un morfism de algebre Boole injectiv $f : B \to \mathcal{P}(X)$.

Demonstraţie. Vom nota cu X mulţimea ultrafiltrelor lui B şi cu f: $B \to \mathcal{P}(X)$, funcţia definită astfel:

$$f(x) = \{F \in X \mid x \in F\}$$

Pentru orice $x, y \in B$, avem echivalenţele:

$F \in f(x \vee y) \Leftrightarrow x \vee y \in F$
$\Leftrightarrow x \in F$ sau $y \in F$ (F este prim)
$\Leftrightarrow F \in f(x)$ sau $F \in f(y)$
$\Leftrightarrow F \in f(x) \cup f(y)$

$F \in f(x \wedge y) \Leftrightarrow x \wedge y \in F$
$\Leftrightarrow x \in F$ şi $y \in F$ (F este filtru)
$\Leftrightarrow F \in f(x)$ şi $F \in f(y)$
$\Leftrightarrow F \in f(x) \cap f(y)$

$F \in f(\neg x) \Leftrightarrow \neg x \in F$
$\Leftrightarrow x \notin F$ (Propoziţia 2.4.3, (iii))
$\Leftrightarrow F \notin f(x)$
$\Leftrightarrow F \in C_X f(x)$

Am arătat deci că:

$f(x \vee y) = f(x) \cup f(y)$
$f(x \wedge y) = f(x) \cap f(y)$
$f(\neg x) = C_X f(x)$

Pentru a arăta că f este injectivă, vom proba că $F_f = \{1\}$ (vezi Propoziţia 2.3.3, (b)). Presupunem $f(x) = X$, deci $f(\neg x) = \varnothing$.

Dacă $x \neq 1$, atunci $\neg x \neq 0$. Aplicând corolarul Propoziţiei 2.4.3 rezultă un ultrafiltru F astfel încât $\neg x \in F$, deci $F \in f(\neg x) = \varnothing$, ceea ce este o contradicţie. Aşadar $x = 1$.

Observaţie 2.4.5.1: Teorema lui Stone se poate enunţa şi altfel: "Orice algebră Boole este izomorfă cu o subalgebră Boole a unei algebre Boole de forma $\mathcal{P}(X)$".

2.5. *Algebre Boole finite*

Definiţia 2.5.1. Fie B o algebră Boole. Un element $x \in A$ se numeste atom dacă $x \neq 0$ şi dacă pentru orice $y \in B$, avem implicaţia:

$$0 \leq y \leq x \Rightarrow y = 0 \text{ sau } y = x$$

Algebra Boole se numeşte atomică dacă pentru orice $x \in B$ diferit de 0 există un atom a, astfel încât $a \le x$. B se numeste fără atomi dacă nu are nici un atom.

Exemplu: Într-o algebră Boole de forma $\mathcal{P}(X)$, orice parte de forma $\{x\}$, $x \in X$ este un atom.

Noţiunea de atom ne va fi necesară în caracterizarea algebrelor Boole finite.

Propoziţia 2.5.1. Orice algebră Boole finită este atomică.

Demonstraţie. Fie B o algebră Boole finită care nu este atomică, deci există $a_0 \in A$, $a_0 \neq 0$ şi pentru care nu există nici un atom $\le a_0$.

Construim prin inducţie un şir strict descrescător

$$a_0 > a_1 > \ldots > a_n > \ldots > 0.$$

Într-adevăr, presupunând că $a_0 > a_1 > \ldots > a_n$, atunci există a_{n+1}, cu proprietatea că $a_n > a_{n+1}$ (dacă nu există nici un element a_{n+1} cu această proprietate, ar rezulta că a_n este un atom şi $a_n < a_0$, ceea ce contrazice ipoteza făcută). Dar existenţa şirului strict descrescător $a_0 > a_1 > \ldots > a_n > \ldots > 0$ contrazice faptul că B este finită. Deci B este atomică.

Propoziţia 2.5.2. Dacă B este o algebră Boole finită cu n atomi $a_1, \ldots, a_n$, atunci B este izomorfă cu $\mathcal{P}(\{a_1, \ldots, a_n\})$.

Demonstraţie: Fie $A = \{a_1, \ldots, a_n\}$. Considerăm funcţia $f : B \to \mathcal{P}(A)$ definită de

$$f(x) = \{a \in A \mid a \le x\}, \text{ pentru orice } x \in B.$$

Arătăm că

$$f(x \vee y) = f(x) \cup f(y).$$

Incluziunea $f(x) \cup f(y) \subseteq f(x \vee y)$ este evidentă:

$$a \in f(x) \cup f(y) \Rightarrow a \le x \quad \text{sau} \quad a \le y \Rightarrow a \le x \vee y \Rightarrow a \in f(x \vee y).$$

Presupunând prin absurd că incluziunea cealaltă nu are loc, va exista $a \in f(x \vee y)$ şi $a \notin f(x)$, $a \notin f(y)$. Atunci avem $a \not\le x$, $a \not\le y$ deci $a \wedge x < a$, $a \wedge y < a$.

Cum a este atom, rezultă că $a \wedge x = 0$ şi $a \wedge y = 0$ deci $a \wedge (x \vee y) = (a \wedge x) \vee (a \wedge y) = 0$

Din $a \in f(x \vee y)$ rezultă $a \le x \vee y$, deci $a \wedge (x \vee y) = a$. Ar rezulta $a = 0$, ceea ce contrazice faptul că a este atom. Deci şi incluziunea cealaltă este adevărată.

Vom stabili acum egalitatea $f(x \wedge y) = f(x) \wedge f(y)$:

$$\begin{aligned} a \in f(x \wedge y) &\Leftrightarrow a \le x \wedge y \\ &\Leftrightarrow a \le x \text{ şi } a \le y \text{ (conform definiţiei infimului)} \\ &\Leftrightarrow a \in f(x) \text{ şi } a \in f(y) \\ &\Leftrightarrow a \in f(x) \cap f(y) \end{aligned}$$

Avem şi relaţiile :

$f(0) = \varnothing$: deoarece nu există nici un atom a astfel încât $a \le 0$.

$f(1) = A$: deoarece $a \le 1$, pentru orice $a \in A$

Am demonstrat că f este morfism de algebre Boole.

Pentru a arăta că f este injectiv este suficient să arătăm că:

$$f(x) = X \quad \Rightarrow x = 1$$

sau, echivalent,

$$f(x) = \varnothing \quad \Rightarrow x = 0$$

Presupunând $x \neq 0$, atunci, B fiind atomică, există $a \in A$ astfel încât $a \le x$, deci $a \in f(x)$. Cu alte cuvinte, $x \neq 0 \Rightarrow f(x) \neq \varnothing$.

A rămas să arătăm surjectivitatea lui f. Fie $X \subset A$, deci X are forma

$$X = \{a_{i_1}, \ldots, a_{i_K}\}, \ 1 \le i_1 < \ldots < i_k \le n.$$

Notăm $x = a_{i_1} \vee \ldots \vee a_{i_K}$. Vom arăta că $f(x) = X$.

Din $a_{i_1} \leq x, \ldots, a_{i_K} \leq x$ rezultă $a_{i_1} \in f(x), \ldots, a_{i_K} \in f(x)$, deci $X \subseteq f(x)$. Presupunând $a \in f(x)$, avem $a \leq x$, deci

$$a = a \wedge x = a \wedge [a_{i_1} \vee \ldots \vee a_{i_K}] = (a \wedge a_{i_1}) \vee \ldots \vee (a \wedge a_{i_K})$$

Există un indice $j \in \{i_1, \ldots, i_k\}$, astfel încât $a \wedge a_j \neq 0$.

Astfel, am avea

$$a \wedge a_{i_1} = \ldots = a \wedge a_{i_K} = 0 \Rightarrow a = 0 \qquad \text{(absurd)}$$

Cum a, a_j sunt atomi, rezultă $a = a_j$. Într-adevăr, dacă $a \neq a_j$ am avea $0 < a \wedge a_j < a$, ceea ce contrazice faptul că a este atom . Aşadar $a = a_j \in X$, ceea ce stabileşte incluziunea $f(x) \subseteq X$.

În concluzie, f este un izomorfism.

Propoziţia 2.5.3. Pentru orice algebră Boole B, sunt echivalente afirmaţiile:

(i) B este atomică

(ii) Pentru orice $a \in B$, avem $a = \vee \{x \mid x \leq a, x \text{ atom al lui } A\}$

Demonstraţie. (i) $\Rightarrow$ (ii) . Fie $a \in B$. Este evident că a este un majorant al familiei $X_a = \{x \mid x \leq a, x \text{ atom al lui } A\}$.

Presupunem că b este un alt majorant al acestei familii. Dacă $a \not\leq b$, atunci $a \wedge \neg b \neq 0$, dacă există un atom x cu $x \leq a \wedge \neg b$. Atunci $x \leq a$, deci $x \in X_0$, de unde rezultă că $x \leq b$. Am obţinut contradicţia $x \leq b \wedge \neg b = 0$, deci $a \leq b$.

Am arătat că a este cel mai mic majorant al lui X_a.

(ii)$\Rightarrow$ (i). Evident.

Corolarul 2.5.3.1. Dacă B este atomică şi are un număr finit de atomi, atunci B este finită.

Demonstraţie. Conform propoziţiei precedente, orice element $a \in B$ este supremul mulţimii X_a a atomilor $\leq a$. Din ipoteză rezultă că X_a este întotdeauna o submulţime a unei mulţimi finite, deci B este finită.

Exerciţiu. Fie A, B două algebre Boole finite. Atunci A, B sunt izomorfe dacă şi numai dacă $\text{card}(A) = \text{card}(B)$.

Indicaţie: Se aplică Propoziţia 2.5.2.

Lecţia 3 - Logici cu mai multe valori

Logicile cu mai multe valori (polivalente) au fost introduse şi studiate de logicianul polonez J. Lucasiewicz în urma unor cercetări legate de studiul modalităţilor. Legate de aceste logici Gr. C. Moisil a studiat începând din 1940 o clasă de structuri algebrice (numite algebre Lucasiewicz în onoarea logicianului polonez) care sunt reflectarea pe plan algebric a logicilor lui Lucasiewicz. Astăzi teoria algebrelor Lucasiewicz este destul de bogată, atât prin rezultatele lui Moisil şi ale elevilor săi, cât şi prin contribuţia a numeroşi cercetători străini. În primul paragraf prezentăm sumar ideile care l-au condus pe Lucasiewicz în considerarea logicilor cu mai multe valori. Paragraful 2 prezintă o serie de rezultate privind algebrele Lucasiewicz, cel mai important fiind teorema de reprezentare a lui Moisil. În sfârşit, ultimul paragraf conţine câteva exemple incipiente ale logicii trivalente neformalizate, făcându-se legătura cu algebrele Lucasiewicz trivalente.

3.1. Idei care au condus la apariţia logicilor cu mai multe valori

Logica trivalentă a apărut ca urmare a studierii propoziţiilor de forma *"este posibil ca.."* sau *"este necesar ca.."*. Pentru fixarea ideilor, să considerăm o propoziţie oarecare p. Vom nota cu Mp propoziţia *"p este posibil"*(simbolul M derivă de la "möglich" = *posibil*).

Putem forma următoarele combinaţii de propoziţii:

1. *" p este fals "* $\neg p$
2. *" p este posibil "* Mp
3. *" p nu este posibil "* $\neg Mp$
4. *" este posibil non - p "* $M\neg p$
5. *" nu este posibil non -p "* $\neg M\neg p$

Propoziţia (5) este echivalentă cu *" nu este posibil ca p să fie falsă "*, care este totuna cu *" p este necesar adevărată "* sau pe scurt *" p este necesar"*. Vom nota această propoziţie cu Np. Propoziţia (3) se va mai citi *" p este imposibil "*.

Lucasiewicz consideră că următoarele propoziţii trebuiesc acceptate ca evidente:

I. $\neg Mp \rightarrow \neg p$

II. $\neg p \rightarrow \neg Mp$

III. $(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow q)$

IV. $(\neg p \rightarrow q) \rightarrow (\neg q \rightarrow p)$

V. $(p \rightarrow \neg q) \rightarrow (q \rightarrow \neg p)$

VI. $(p \rightarrow q) \rightarrow ((q \rightarrow r) \rightarrow (p \rightarrow r))$

Prima propoziţie este *" dacă p este imposibil, atunci p este fals"*, iar a doua este *" dacă p este fals, atunci p nu este posibil "*. Celelalte patru propoziţii nu fac să intervină conectorul M şi nu comportă nici o discuţie.

La aceste propoziţii, Lucasiewicz adaugă propoziţiile de forma:

" Pentru o anumită propoziţie p, este posibil p şi este posibil non -p". Lucasiewicz dă următorul exemplu: *" Se poate ca acest bolnav să moară, dar se poate să şi nu mo*ară *"*.

Pentru formularea simbolică a acestei propoziţii este necesară introducerea unui cuantificator particular Σ: *" Σp = pentru un anumit p"*.

Propoziţia de mai sus ia următoarea formă simbolică:

VII. $\Sigma p\ (Mp \wedge \neg Mp)$.

Din propoziţiile (I) – (VI) se pot deduce următoarele propoziţii:

a) $p \rightarrow Mp$

b) $Mp \rightarrow p$

Cu alte cuvinte, orice propoziţie p este echivalentă cu propoziţia *" p este posibil "*. Aceasta ar face superfluxă considerarea propoziţiilor de tipul *" p este posibil "*, ceea ce din punct de vedere real nu este acceptabil.

În prezenţa propoziţiei (VII) se poate deduce următoarea propoziţie:

c) *Mp*.

Aplicând regula *modus ponens*, din (b) şi (c) se deduce că orice propoziţie este adevărată, ceea ce arată contradicţia propoziţiilor acceptate ca axiome mai sus.

Aceste constatări l-au condus pe Lucasiewicz la concluzia următoare: principiul *terţiului exclus*, după care orice propoziţie p este adevărată sau falsă, nu funcţionează pentru propoziţii de forma *" p este posibil "*.

De pildă, propoziţia *" Anul viitor, la* 1 *septembrie, este posibil să plouă la Bucureşti "* nu este nici adevărată, nici falsă.

În mod necesar se impune considerarea unei a treia valori de adevăr: *" posibilul "*, obţinându-se astfel punctul de plecare pentru ceea ce se numeşte *" logica trivalentă "*.

Observaţie. Aceste consideraţii aparţin lui J. Lucasiewicz. Pentru consultarea lor în detaliu şi pentru demonstrarea unor afirmaţii făcute în acest paragraf, recomandăm cititorului cartea lui A. Dumitriu *" Logica polivalentă "*, cap.V, pag. 157-207.

3.2 Algebre Lucasiewicz *n*-valente

Algebrele Lucasiewicz n-valente au fost introduse de Grigore C. Moisil în anul 1940 ca modele algebrice pentru logicile cu mai multe valori ale lui Lucasiewicz. Înainte de a prezenta sistemul formal al logicii trivalente, vom studia câteva proprietăţi ale algebrelor Lucasiewicz n-valente.

Vom presupune în continuare că n este un număr natural finit.

Definiţia 1. O algebră Lucasiewicz n-valentă este o latice distributivă $(L, \vee, \wedge, 0, 1)$ cu prim element 0 şi cu ultim element 1, astfel încât:

I. Există o operaţie unară $\neg : L \to L$ cu proprietăţile:

$$\neg(x \vee y) = \neg x \wedge \neg y$$
$$\neg(x \wedge y) = \neg x \vee \neg y$$
$$\neg\neg x = x,$$

pentru orice $x, y \in L$.

II. Există $(n-1)$ aplicaţii $\sigma_i : L \to L$, $i = 1,..., n-1$ cu proprietăţile:

a) $\sigma_i(0) = 0$; $\sigma_i(1) = 1$, pentru orice $i = 1,..., n-1$.

b) $\sigma_i(x \vee y) = \sigma_i(x) \vee \sigma_i(y)$, $\sigma_i(x \wedge y) = \sigma_i(x) \wedge \sigma_i(y)$, pentru orice $i = 1,..., n-1$ şi $x, y \in L$.

c) $\sigma_i(x) \vee \neg\sigma_i(x) = 1$, $\sigma_i(x) \wedge \neg\sigma_i(x) = 0$, pentru orice $i = 1,..., n-1$ şi $x \in L$.

d) $\sigma_k o \sigma_k = \sigma_k$, pentru orice $k = 1,..., n-1$.

e) $\sigma_i(\neg x) = \neg\sigma_j(x)$, pentru $i + j = n$ şi pentru orice $x \in L$.

f) $\sigma_1(x) \le \sigma_2(x) \le \ldots\ldots \le \sigma_{n-1}(x)$, pentru orice $x \in L$.

g) Dacă $\sigma_i(x) = \sigma_j(x)$ pentru orice $i = 1,\ldots., n-1$, atunci $x = y$.

Observaţie: Axioma (g) se numeşte *principiul determinării al lui Moisil*.

$\sigma_1,\ldots, \sigma_{n-1}$ se numesc *endormorfisme chrysipiene*.

Definiţia 2. Dacă L, L' sunt două algebre Lucasiewicz n-valente, atunci o funcţie $f : L \to L'$ se numeşte morfism de algebre Lucasiewiecz n-valente dacă pentru orice $x, y \in L$ avem:

1) $f(0) = 0; \quad f(1) = 1;$
2) $f(x \vee y) = f(x) \vee f(y);$
3) $f(x \wedge y) = f(x) \wedge f(y);$
4) $f(\sigma_i(x)) = \sigma_i(f(x)).$

Prezentăm fără demostraţie un rezultat important.

Lema 1. Dacă $f : L \to L'$ este un morfism de algebre Lucasiewicz n-valente atunci $f(\neg x) = \neg f(x)$, pentru orice $x \in L$.

Exemplul 1. Considerăm în mulţimea

$$L_n = \left\{0, \quad \frac{1}{n-1}, \quad \frac{2}{n-1}, \quad \ldots \quad \frac{n-2}{n-1}, \quad 1\right\}$$

următoarele operaţii:

$x \vee y = max(x, y)$
$x \wedge y = min(x, y)$
$\neg x = 1 - x.$

Definim funcţiile $\sigma_1, \ldots, \sigma_{n-1} : L \to L$ prin următorul tablou:

x	$\sigma_1(x)$	$\sigma_2(x)$	.....	$\sigma_{n-2}(x)$	$\sigma_{n-1}(x)$
0	0	0	.....	0	0
$\frac{1}{n-1}$	0	0	…….	0	1
$\frac{2}{n-1}$	0	0	…….	1	1
. . .	0	…..	…….	1	1
$\frac{n-2}{n-1}$	0	1	……..	1	1
1	1	1	.....	1	1

Se poate verifica uşor că L_n este o algebră Lucasiewicz n-valentă.

3.3. Logica trivalentă

În acest paragraf vom prezenta câteva noţiuni şi proprietăţi ale calculului propoziţional trivalent neformalizat. Se pleacă de la ideia că orice propoziţie p poate fi adevărată, falsă sau posibilă (îndoielnicul). Cu alte cuvinte, vom atribui fiecărei propoziţii p o valoare de adevăr $v(p)$ aparţinând mulţimii

$$L_3 = \{0, \ ½, \ 1 \}$$

astfel încât:

$$v(p) = \begin{cases} 0, \text{ daca } p \text{ este falsa} \\ 1/2, \text{ daca } p \text{ este posibila} \\ 1, \text{ daca } p \text{ este adevarata} \end{cases}$$

Deci dacă P este mulţimea propoziţiilor, atunci v este o funcţie $v : P \to L_3$.

Valoarea de adevăr a conjucţiei $p \wedge q$ şi disjuncţiei $p \vee q$ a două propoziţii p şi q este definită prin tablourile următoare:

$v(q)$ \ $v(p)$	0	1/2	1
0	0	0	0
1/2	0	1/2	1/2
1	0	1/2	1

$v(p \wedge q)$

$v(q)$ \ $v(p)$	0	1/2	1
0	0	1/2	1
1/2	1/2	1/2	1
1	1	1	1

$v(p \vee q)$

Valoarea de adevăr a negaţiei $\neg p$ este dată de:

$v(p)$	0	1/2	1
$v(\neg p)$	1	1/2	0

Valorile de adevăr ale implicaţiei şi echivalenţei sunt date de:

$$v(p \to q) = min(1, 1 - v(p) + v(q))$$
$$v(p \leftrightarrow q) = v(p \to q) \wedge v(q \to p),$$

în ultima relaţie, luând intersecţia din algebra Lucasiewicz L_3.

Valoarea de adevăr a propoziţiei:

$Mp = p$ este *posibil*

este introdusă de tabloul:

$v(p)$	0	1/2	1
$v(Mp)$	0	0	1

Valoarea de adevăr a propoziţiei:

$Np = p$ este *necesar*

este introdusă de tabloul:

$v(p)$	0	1/2	1
$v(Np)$	0	1	1

Propoziţia 1. Funcţia de adevăr v verifică următoarele proprietăţi:

$$v(p \vee q) = v(p) \vee v(q)$$
$$v(p \wedge q) = v(p) \wedge v(q)$$
$$v(\neg p) = \neg v(p)$$
$$v(Mp) = \sigma_1(v(p))$$
$$v(Np) = \sigma_2(v(p))$$

Demostraţia acestei propoziţii se face urmărind în paralel tablourile cu care au fost definite operaţiile lui L_3 şi tablourile de adevăr definite mai înainte.

Exerciţii rezolvate

1. Fie $G(n)$ numărul relaţiilor de ordine parţială ce se pot defini pe o mulţime cu n elemente. Arătaţi că pe o mulţime cu două elemente există exact trei relaţii de ordine parţială.

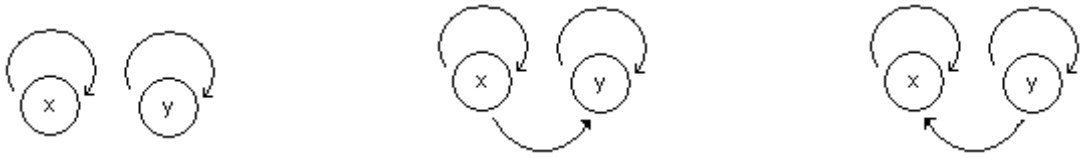
Rezolvare:

R este o relaţie de ordine parţială pe ***M*** dacă:

(i) $(\forall)x \in M, xRx$

(ii) $(\forall)x, y, z \in M, xRy, yRz \Rightarrow xRz$

Dacă ***M={x, y}*** este o mulţime de 2 elemente atunci avem evident:



deci $G(2)=3$.

2. Să se arate că orice mulţime finită poate fi înzestrată cu o relaţie de ordine totală.

Rezolvare:

O mulţime se numeşte total ordonată dacă relaţia de ordine „$\leq$” are următoarele proprietăţi:

(i) $(\forall)x \in M, x \leq x$

(ii) $(\forall)x, y, z \in M, x \leq y, \leq Rz \Rightarrow x \leq z$

(iii) $(\forall)x, y \in M, x \leq y$ şi $y \leq x \Rightarrow x = y$

(iv) $(\forall)x, y \in M, x \leq y$ sau $y \leq x$

Fie $M=\{x_1,x_2,\ldots,x_n\}, n \in \mathbb{N}^*$ astfel încât $x_i \leq x_i$, $i \in \overline{1,n}$ şi $x_i \leq x_j$, dacă $i<j$, $i,j \in \{1,2,\ldots,n\}$. Proprietăţile enunţate sunt verificate.

3. Să se arate că în orice latice ***L*** avem adevărată inegalitatea:

$$(a \wedge c) \vee (b \wedge d) \leq (a \vee b) \wedge (c \vee d).$$

Rezolvare:

L este o latice, dacă **(*L*,≤)** este o mulţime pre ordonată şi pentru orice {x,y} din ***L*** există $x \wedge y$ şi $x \vee y$. Facem la început demonstraţia în ipoteza că laticea este o latice distributivă, adică oricare ar fi x,y,z din ***L*** se satisface una din următoarele relaţii echivalente:

(i) $(x \wedge y) \vee (x \wedge z) = x \wedge (y \vee z)$;

(ii) $(x \vee y) \wedge (x \vee z) = x \vee (y \wedge z)$;

(iii) $(x \vee y) \wedge z \leq x \vee (y \wedge z)$.

Avem:

$$(a \wedge c) \vee (b \wedge d) \overset{ii}{=} ((a \wedge c) \vee b) \wedge ((a \wedge c) \vee d) = ((a \vee b) \wedge (c \vee b)) \wedge ((a \vee d) \wedge (c \vee d)) =$$
$$= (a \vee b) \wedge (c \vee b) \wedge (a \vee d) \wedge (c \vee d) = ((a \vee b) \wedge (c \vee d)) \wedge ((b \vee c) \wedge (a \vee d)) \leq (a \vee b) \wedge (c \vee d),$$

deoarece $x \wedge y \leq x$.

$$(a\vee b)\wedge(c\vee d)=((a\vee b)\wedge c)\vee((a\vee b)\wedge d)=(a\wedge c)\vee(b\wedge c)\vee(a\wedge d)\vee(b\wedge d)=$$

$$=((a\wedge c)\vee(b\wedge d))\vee((b\wedge c)\vee(a\wedge d))\overset{?}{\leq}(a\wedge c)\vee(b\wedge d).$$

$a\leq a\vee b\Rightarrow a\wedge c\leq(a\vee b)\wedge c$ (1)

$b\leq a\vee b\Rightarrow b\wedge d\leq(a\vee b)\wedge d$ (2)

Din (1) şi (2) rezultă: $(a\wedge c)\vee(b\wedge d)\leq((a\vee b)\wedge c)\vee((a\vee b)\wedge d)$ (3)

Este suficient să demostrăm că $((a\vee b)\wedge c)\vee((a\vee b)\wedge d)\leq(a\vee b)\wedge(c\vee d)$ (4)

$(a\vee b)\wedge c\leq(a\vee b)\wedge(c\vee d)$ pentru că $c\leq c\vee d$ (*)

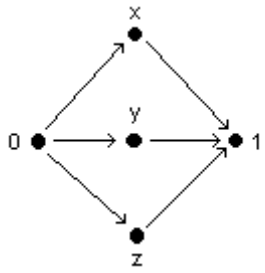
$(a\vee b)\wedge d\leq(a\vee b)\wedge(c\vee d)$ pentru că $d\leq c\vee d$ (**)

Cum x≤z şi y≤z rezultă că $x\vee y\leq z$. Tot astfel, din (*) şi (**) rezultă relaţia (4).

4. O latice ***L*** se numeşte modulară dacă pentru orice ***x,y,z*** din ***L*** avem :

$$x\leq z\Rightarrow x\vee(y\wedge z)=(x\vee y)\wedge z.$$

Să se arate că laticea reprezentată prin graful următor este modulară, dar nu este distributivă.



***Rezolvare*:**

- $0\leq 1\Rightarrow 0\vee(x\wedge 1)=(0\vee x)\wedge 1$, adică $0\vee x=x\wedge 1, x=x$ - adevărat; analog pentru y şi z, în locul lui x. Verificăm şi pentru x=0 sau x=1: $0\vee(0\wedge 1)=(0\vee 0)\wedge 1$, respectiv $0\vee(1\wedge 1)=(0\vee 1)\wedge 1$, care evident sunt adevărate.
- $0\leq x\Rightarrow 0\vee(y\wedge x)=(0\vee y)\wedge x$, adică $0\vee 0=y\wedge x, 0=0$ - adevărat; analog pentru *z* în locul lui *y*; analog pentru *y* şi *z* în locul lui *x*.
- $x\leq 1\Rightarrow x\vee(y\wedge 1)=(x\vee y)\wedge 1$, adică $x\vee y=1\wedge 1, 1=1$ - adevărat; analog pentru *y* şi *z* în locul lui *x*.

Rezultă că laticea este modulară.

L este o latice distributivă dacă $(x\wedge y)\vee(x\wedge z)=x\wedge(y\vee z)$. Adică, $x\wedge(y\vee z)=0\vee 0=0.$ Dar conform proprietăţii de modularitate $x\wedge(y\vee z)=x\wedge 1=x.$ Cum *x* nu este egal cu 0, rezultă că laticea nu este distributivă.

5. Fie ***B, B'*** două algebra Boole şi ***f*** : ***B*** → ***B'*** o aplicaţie oarecare. Să se demonstreze că afirmaţiile următoare sunt echivalente:

(1) *f* este morfism de algebre Boole.

(2) $f(\neg x)=\neg f(x)$ şi $f(x\vee y)=f(x)\vee f(y)$, pentru orice ***x,y*** din ***B***.

(3) $f(\neg x)=\neg f(x)$ şi $f(x\wedge y)=f(x)\wedge f(y)$, pentru orice ***x,y*** din ***B***.

(4) *f*(0)=0, *f*(1)=1, $f(x\vee y)=f(x)\vee f(y)$, pentru orice ***x,y*** din ***B*** şi $f(x)\wedge f(y)=0$ atunci când $x\wedge y=0$.

***Rezolvare*:**

f : ***B*** → ***B'*** este un morfism de algebra Boole, dacă:

(i) $f(x \vee y) = f(x) \vee f(y)$,

(ii) $f(x \wedge y) = f(x) \wedge f(y)$,

(iii) $f(\neg x) = \neg f(x)$.

Rezultă deci că 1 implică 2 şi 1 implică 3 sunt evidente.

Demonstrăm că 2 implică 1:

(i) şi (iii) din definiţia morfismului unei algebre Boole rezultă imediat.

Demonstrăm (ii).

$x \wedge y = \neg(\neg x \vee \neg y)$ rezultă că

$$f(x \wedge y) = f\left(\neg(\neg x \vee \neg y)\right) \overset{2}{=} \neg f(\neg x \vee \neg y) = \neg\left(f(\neg x) \vee f(\neg y)\right) =$$

$$= \neg\left(\neg f(x) \vee \neg f(y)\right) = f(x) \wedge f(y).$$

Analog se demonstrează că 3 implică 1, pornind de la $x \vee y = \neg(\neg x \wedge \neg y)$.

Demonstrăm că 3 implică 4:

$$f(0) = f(x \wedge \neg x) \overset{3}{=} f(x) \wedge f(\neg x) = 0.$$

$$f(1) = f(x \vee \neg x) \overset{3}{=} f(x) \vee f(\neg x) = 1.$$

$f(x \wedge y) = 0$, dar $f(x \wedge y) = f(x) \wedge f(y) \overset{3}{=} f(x \wedge \neg x) = 0.$

Demonstrăm că 4 implică 2:

$x \wedge \neg x = 0 \Rightarrow f(x \wedge \neg x) = f(x) \wedge f(\neg x) = 0.$

$x \vee \neg x = 1 \Rightarrow f(x \vee \neg x) = f(x) \vee f(\neg x) = 1.$

Rezultă că $\neg f(x) = f(\neg x)$.

Cum 4 implică 2 şi 2 implică 1, rezultă că 4 implică 1. Cu aceasta s-a demonstrat echivalenţa celor patru propoziţii.

6. Fie *A* o mulţime dată, finită şi *P*(*A*) mulţimea părţilor lui *A*. Să se găsească toate subalgebrele Boole ale mulţimii *P*(*A*), atunci când *A*={*x*,*y*}

Rezolvare:

Dacă card*A*=*n*, atunci card*P*(A)=2^n. Mulţimea părţilor unei mulţimi *A* se poate organiza ca o algebră Boole dacă considerăm operaţiile: conjuncţia $\wedge$ ca fiind intersecţia de mulţimi $\cap$ şi disjuncţia ca fiind reuniunea de mulţimi $\cup$ iar negaţia $\neg$ ca fiind complementara unei mulţimi. Mulţimea *M* este o subalgebră a mulţimii *P*(*A*) dacă aceasta este închisă la reuniune, intersecţie şi complementară. Aceasta înseamnă şi că Ø şi *A* aparţin obligatoriu lui *M*.

Dacă *A*={*x*,*y*}, atunci cardA=2 şi rezultă că *P*(A)={Ø,{x},{y},A}. Atunci

M_1={ Ø,A };

M_2={ Ø,A,{x},{y}}.

şi deci avem 2 subalgebre.

7. Fie *B* o algebră Boole şi $x \in B$. Să se arate că mulţimea F_x={*y* | *y*≥*x*} este un filtru al lui *B*. Cercetaţi în ce caz F_x este filtru propriu al lui *B*.

***Rezolvare*:**

F_x este un filtru al lui B, dacă:

(i) $(\forall) y_1, y_2 \in F_x \Rightarrow y_1 \wedge y_2 \in F_x$;

(ii) dacă $y \in F_x$ şi $y \le y_1 \Rightarrow y_1 \in F_x$.

F_x este filtru propriu al lui B dacă $F_x \neq B$. Acest lucru se demonstrează arătând că $0 \notin F_x$.

Dacă y_1 şi y_2 aparţin lui F_x, atunci $x \le y_1$ şi $x \le y_2$. Rezultă că $x \wedge x \le y_1 \wedge y_2 \Rightarrow x \le y_1 \wedge y_2 \Rightarrow y_1 \wedge y_2 \in F_x$.

Fie $y \in F_x$; rezultă că $x \le y$. Fie y_1 astfel încât $y \le y_1$. Rezultă prin tranzitivitate că $x \le y_1$şi deci că y_1 aparţine lui F_x.

Cu aceasta am demonstrate că F_x este un filtru al lui B.

Dacă $x=0$, $F_0=\{y \mid 0 \le y\}$, adică $F_0=B$ şi deci F_0 nu este filtru propriu al lui B.

Dacă $x \neq 0$ să presupunem, prin reducere la absurd, că $0 \in F_x$. Aceasta înseamnă, conform definirii lui F_x, $x \le 0$. Contradicţie cu ipoteza că 0 este infimum-ul mulţimii B. Rezultă că 0 nu aparţine lui F_x, şi deci F_x este filtru propriu al lui B.

8. În baza exemplului 1, lecţia 2.2, paragraful 2.2, detaliem cazul $n = 3$.

Rezolvare:

Considerăm în mulţimea

$$L_3 = \{0,\ ½,\ 1\}$$

operaţiile lui L_3 scrise sub formă de tabele:

x / y	0	½	1
0	0	½	1
½	½	½	1
1	1	1	1

$x \vee y$

x / y	0	½	1
0	0	0	0
½	0	½	½
1	0	½	1

$x \wedge y$

x	$\neg x$
0	1
½	½
1	0

$\neg x$

x	$\sigma_1(x)$	$\sigma_2(x)$
0	0	0
½	0	1
1	1	1

Se poate verifica uşor că L_3, cu aceste operaţii, este o algebră Lucasiewicz 3-valentă.

TESTE DE AUTOEVALUARE ŞI TEME DE CONTROL

Testul nr. 1

1. Fie $G(n)$ numărul relaţiilor de ordine parţială ce se pot defini pe o mulţime cu n elemente. Arătaţi că $G(3)=19$.

2. O semilatice este o mulţime A înzestrată cu o operaţie binară cu proprietăţile următoare:

(i) $xox = x$, pentru orice $x \in A$;

(ii) $xoy = yox$, pentru orice $x,y \in A$;

(iii) $xo(yoz)=(xoy)oz$, pentru orice $x,y,z \in A$.

Dacă notăm:

$$x \leq y \Leftrightarrow xoy=x, \qquad (*)$$

atunci arătaţi că $(A,\leq)$ este o mulţime parţial ordonată astfel încât pentru orice $x,y \in A$, $xoy = x \wedge y$.

Testul nr. 2

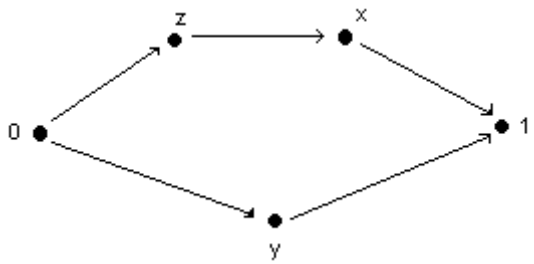
1. Este mulţimea $\boldsymbol{N}$ a numerelor naturale o latice completă faţă de relaţia de ordine definită de divizibilitate?

2. Fie A o mulţime dată, finită şi $P(A)$ mulţimea părţilor lui A. Să se găsească toate subalgebrele Boole ale mulţimii $P(A)$, atunci când $A=\{x,y,z\}$.

Temă de control

1. Fie $G(n)$ numărul relaţiilor de ordine parţială ce se pot defini pe o mulţime cu n elemente. Arătaţi că $G(4) =219$.

2. Să se arate că laticea de mai jos nu este modulară:



3. Arătaţi că într-un inel comutativ A de caracteristică 2, mulţimea $\{x \mid x^2 = x\}$ formează un inel Boole care este subinel al lui A. (A are caracteristica 2 dacă $x + x = 0$, pentru orice x din A).

Indicaţie: $B \subset A$este subinel, dacă:

(i) $(\forall)x, y \in B, x + y \in B$;

(ii) $(\forall)x \in B, -x \in B$;

(iii) $(\forall)x, y \in B, xy \in B$;

(iv) $1 \in$ B.

4. Fie *A* o mulţime dată, finită şi *P*(*A*) mulţimea părţilor lui *A*. Să se găsească toate subalgebrele Boole ale mulţimii *P*(*A*), atunci când *A*={*x*,*y*,*z*.*v*.*w*}.

5. După modelul exerciţiului 8 de la probleme rezolvate, să se trateze cazul $n = 4$ şi $n = 5$, pentru algebre Lukasiewicz..

BIBLIOGRAFIE RECOMANDATĂ LA UNITATEA DE ÎNVĂŢARE NR.1

1. G. Georgescu – *Elemente de logică matematică*, Editura Academiei Tehnice Militare, Bucureşti, 1978
2. G. Metakides, A. Nerode – *Principii de logică şi programare logică*, Editura Tehnică, Bucureşti, 1998
3. D. Busneag, D. Piciu, Probleme de logica si teoria multimilor, Craiova, 2003.
4. G. Georgescu, A. Iorgulescu, Logica matematica, Ed. ASE, Bucuresti, 2010
5. Gr. C Moisil, Elemente de logica matematica si de teoria multimilor, Ed. Stiintifica, Bucuresti, 1968
6. J.D. Monk, Mathematical Logic, Springer Verlag, 1976

UNITATEA DE ÎNVĂŢARE NR.2
LOGICA PROPOZIŢIONALĂ

Lecţia 4 –Logica propoziţională

4.1 *Introducere*

Logica s-a dezvoltat ca o ştiinţă independentă după 1920, prin reprezentaţii ei de marcă Lukasiewicz (1878-1956), Lewis (1883-1964), Gödel (1906-1978), Tarski (1901-1983), Church (1903-1995) şi Kleene (1909-1994).

Începând cu anii ’50, tehnologia calculatoarelor a arătat cum poate fi folosit calculatorul pentru efectuarea prelucrărilor simbolice. O soluţie a acestei probleme a fost dată de programarea funcţională, creată de McCarthy alături de alţi cercetători şi utilizată mai ales în Statele Unite ale Americii. O altă soluţie a fost oferită de programarea logică, creată mai ales în Europa, printre alţii de Colmerauer şi Kowalski. Programarea logică a dat logicii un nou impuls; pe baza programării logice s-au rezolvat numeroase probleme legate de forma expresiilor logice care permit automatizarea realizării deducţiei şi a procedurilor de judecată logică. Programarea logică a oferit, de asemenea, răspunsuri la întrebări referitoare la natura şi mecanismele procedurilor logice executabile pe calculator.

Pentru a înţelege logica trebuie să examinăm, în primul rând, cum sunt formalizate propoziţiile logice şi matematice. Conectorii logici formează elementele de bază ale unei astfel de formalizări, aşa cum se poate vedea din exemplele următoare.

(1) Conjuncţia este formalizată prin $\wedge$. Să presupunem că ştim următoarele două proprietăţi ale unui anume x:

A: $x>3$

B: $x<10$

Atunci ştim despre x că este mai mare decât 3 şi mai mic decât 10. Cu alte cuvinte, cunoaştem propoziţia:

$$A \wedge B$$

unde conectorul logic $\wedge$ corespunde conjuncţiei gramaticale “şi”. Propoziţia $A \wedge B$ indică faptul că “$x > 3$ şi $x < 10$”, ceea ce înseamnă “$3 < x < 10$”.

(2) Negaţia este formalizată cu ajutorul simbolului $\neg$. De exemplu, să considerăm propoziţia:

C: 50 este divizibil cu 7.

$\neg C$ înseamnă în acest caz: “50 nu este divizibil cu 7” .

(3) Disjuncţia este reprezentată prin simbolul $\vee$. Dacă D şi E sunt propoziţiile:

D: 60 este multiplu de 6

E: 60 este multiplu de 5

Atunci propoziţia:

$$D \vee E$$

spune că “60 este multiplu de 6 sau 60 este multiplu de 5”. Simbolul $\vee$ nu este exculsiv, deoarece 60 este atât multiplul lui 6 cât şi al lui 5, aşa cum 60 este şi un multiplu al lui 20, deşi acest fapt nu se menţionează în propoziţiile D sau E.

(4) Implicaţia "dacă … atunci…" este reprezentată în logică prin "→". Dacă F şi G sunt propoziţiile:

F: numărul a este un multiplu de 10
G: numărul a este multiplu de 5

atunci propoziţia $F \to G$ spune că "dacă a este multiplu de 10, atunci este multiplu de 5".

(5) "Dacă … şi numai dacă …" se reprezintă prin simbolul de echivalenţă "↔". De exemplu, dacă H şi I sunt propoziţiile:

H: 16 este multiplu de 2
I: 16 este număr par

Atunci putem scrie formal:

$H \leftrightarrow I$

Conectorii logici pot avea proprietăţi diferite. În exemplul propoziţiilor A şi B, a scrie $A \vee B$ nu diferă conceptual de $B \vee A$. Astfel, intuitiv, conectorul logic $\vee$ pare să fie comutativ şi acelaşi lucru este valabil şi pentru $\wedge$. Nu se poate însă spune la fel despre conectorul $\to$: propoziţia $G \to F$, "dacă a este multiplu de 5, atunci a este multiplu de 10", nu pare a fi corectă deoarece 15 este multiplu de 5, dar nu şi de 10. În consecinţă, trebuie să studiem proprietăţile conectorilor logici.

Trebuie să examinăm sintaxa propoziţiilor şi regulile care determină când şi cum putem deduce concluzii valide din premise pe baza informaţiilor disponibile. Logica propoziţiilor, la fel ca şi logica predicatelor, care este mai complexă, se ocupă de aceste probleme.

În capitolul de logica propoziţiilor (**LP**) vom examina propoziţiile **LP** atât din punctul de vedere al sintaxei cât şi al semanticii. Vom studia de asemenea metode de deducere a concluziilor din premise şi vom stabili formele adecvate ale propoziţiilor **LP** pentru reprezentarea cunoştinţelor, a procedurilor de decizie şi pentru demonstrarea automată a teoremelor în programarea logică.

4.2 *Limbajul logicii propoziţiilor*

Alfabet, sintaxă şi semantică

Limbajul logicii propoziţiilor (**LP**) este un limbaj formal. Prin intermediul acestui limbaj formalizăm acea parte a limbajului cotidian care este necesară conceptelor logice şi matematice. O dată formalizate diverse propoziţii cu ajutorul acestui limbaj, trebuie să examinăm aceste propoziţii din punctul de vedere al structurii şi validităţii lor.

Pentru fiecare limbaj formal există:

(a) un **alfabet** ce conţine toate simbolurile limbajului;

(b) o **sintaxă** care stabileşte cum sunt utilizate simbolurile şi care este forma corectă a propoziţiilor din limbaj;

(c) o **semantică**, pe baza căreia se stabileşte interpretarea şi semnificaţia simbolurilor din limbaj.

Alfabetul limbajului logicii propoziţiilor conţine:

(i) Simboluri propoziţionale: $A, A_1, A_2, \ldots, B, B_1, B_2 \ldots$

(ii) Conectori logici: $\vee$, $\wedge$, $\neg$, $\to$, $\leftrightarrow$

(iii)Virgule şi paranteze: ", " şi "(", ")"

Conectorii logici corespund intuitiv conjuncţiilor gramaticale utilizate în limbajul curent. Avem astfel următoarea corespondenţă:

- $\vee$, disjuncţie: sau
- $\wedge$, conjuncţie: şi
- $\rightarrow$, implicaţie: dacă … atunci …
- $\leftrightarrow$, echivalenţă: …dacă şi numai dacă …
- $\neg$, negaţie: non

Obsevaţie Din motive de ordin istoric, vom considera simbolul "$\leftarrow$" un conector logic, formula "$B \leftarrow A$" având aceeaşi semnificaţie cu "$A \rightarrow B$".

Orice secvenţă de simboluri ce aparţine alfabetului unui limbaj se numeşte expresie. De exemplu, $A \vee\vee B$, $A \vee B$, $\leftrightarrow A$ sunt expresii în limbajul logicii propoziţiilor. Anumite expresii *bine formate* sunt considerate drept propoziţii corecte din punctul de vedere al sintaxei limbajului. Următoarea definiţie stabileşte conceptul de propoziţie şi descrie legile sintactice ale limbajului **LP**.

Definiţia 4.2.1: Definiţia inductivă a propoziţiilor:

(i) Simbolurile propoziţionale sunt propoziţii , numite *propoziţii atomice* sau *atomi*.

(ii) Dacă σ, τ sunt propoziţii, atunci expresiile $(\sigma \wedge \tau)$, $(\sigma \vee \tau)$, $(\sigma \rightarrow \tau)$, $(\sigma \leftrightarrow \tau)$, $(\neg\sigma)$ sunt de asemenea propoziţii numite *propoziţii compuse*.

(iii) Expresiile construite conform regulilor (i) şi (ii) sunt singurele expresii din limbaj care sunt propoziţii.

Expresiile $\vee A \vee B$ şi $\leftrightarrow A$ nu sunt deci propoziţii, deoarece ele nu sunt construite conform regulilor (i) şi (ii), în timp ce $(A \vee B)$ şi $((\neg A) \vee (B \leftrightarrow (\neg C))$ sunt propoziţii.

Definiţia de mai sus foloseşte metoda inducţiei pentru a defini propoziţii compuse; (i) este *primul pas* şi (ii) este *pasul inductiv*. Inducţia se face după "lungimea logică" şi structura propoziţiilor. Prin "lungime logică" a unei propoziţii A înţelegem un număr natural ce reprezintă numărul, tipul şi ordinea de apariţie a conectorilor logici folosiţi pentru construcţia lui A pornind de la propoziţiile atomice ce apar în A.

Am folosit inducţia completă pentru a enunţa definiţia inductivă a propoziţiilor (definiţia 3.2.1) şi o vom folosi din nou pentru a enunţa următoarea definiţie.

Definiţia 4.2.2: Schema generală a inducţiei pentru propoziţii:

Fie P o proprietate propoziţională. Vom scrie $P(\sigma)$ pentru a spune că *propoziţia* σ *are proprietatea* P. Dacă demonstrăm că:

(a) $P(A)$ este adevărată pentru orice atom A din limbaj,

(b) Dacă σ_1, σ_2 sunt propoziţii şi $P(\sigma_1)$, $P(\sigma_2)$, atunci $P((\sigma_1 \wedge \sigma_2))$, $P((\sigma_1 \vee \sigma_2))$, $P((\sigma_1 \rightarrow \sigma_2))$, $P((\sigma_1 \leftrightarrow \sigma_2))$, $P((\neg\sigma_1))$ şi $P((\neg\sigma_2))$,

putem concluziona $P(\sigma)$ pentru orice propoziţie σ din limbaj.

Exemplul 4.2.3:

(i) Expresia E: $(\neg(A \wedge B) \rightarrow C)$ este o propoziţie.

Demonstraţia se bazează pe definiţia 4.2.1:

Pasul 1: $A \wedge B$ este o propoziţie conform definiţiei 4.2.1

Pasul 2: $\neg(A \wedge B)$ este o propoziţie conform definiţiei 4.2.1

Pasul 3: C este o propoziţie conform definiţiei 4.2.1

Pasul 4: $(\neg(A \wedge B) \rightarrow C)$ este o propoziţie coform definiţiei 4.2.1

(ii) Expresiile de mai jos nu sunt propoziţii:

$\neg$: simbolul "$\neg$" nu este un atom

$\wedge \rightarrow A$: simbolul "$\wedge$" nu este o propoziţie
$(A \wedge B$: există un număr incorect de paranteze

Exemplul 4.2.4: Se consideră propoziţia F în limbaj cotidian:
F: "*Dacă nu plouă atunci merg la plimbare*"
Pentru a formaliza această propoziţie în **LP** se pot folosi simboluri propoziţionale auxiliare:
A: "*Plouă*"
B: "*Merg la plimbare*"
atunci F devine $((\neg A) \rightarrow B)$, aceasta fiind o propoziţie.
Dacă nu există riscul unei confuzii, parantezele pot fi omise:
F: $\neg A \rightarrow B$

Fiecare atom care apare într-o formulă bine formată D este considerat o subformulă a lui D. De exemplu, A şi B sunt subformule ale lui F în exemplul 4.2.6. Mai mult, fiecare parte compusă bine formată a unei formule D este o subformulă a lui D. De exemplu, dacă:
D: $(A \wedge \neg B) \rightarrow [(C \vee \neg A) \rightarrow B]$
atunci $\neg A$, $C \vee \neg A$, $A \wedge \neg B$ şi $(C \vee \neg A) \rightarrow B$ sunt subformule ale lui D. În plus, chiar D, adică $(A \wedge \neg B) \rightarrow [(C \vee \neg A) \rightarrow B]$, este considerată o subformulă a lui D.

Mulţimea $subform(\sigma)$ a tuturor subformulelor unei formule bine formate σ este stabilită de următoarea definiţie inductivă:

Definiţia 4.2.5:
(1) dacă σ este un atom A, atunci: $subform(\sigma) = \{A\}$
(2) dacă σ este de forma $\neg\tau$, atunci: $subform(\sigma) = subform(\tau) \cup \{\sigma\}$
(3) dacă σ este de forma $\tau \circ \varphi$, unde $\circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$, atunci:
$subform(\sigma) = subform(\tau) \cup subform(\varphi) \cup \{\tau\}$.

Observaţia 4.2.6: Pentru a evita confuzia în cazul utilizării conectorilor în formule fără paranteze, considerăm $\neg$ ca având cea mai mare prioritate, $\vee$ şi $\wedge$ ca având prioritate mai mare decât $\rightarrow$ şi $\leftrightarrow$, şi $\leftrightarrow$ cu prioritate mai mare decât $\rightarrow$. Astfel, formulele:
$\neg A \rightarrow B \vee C$, $A \wedge B \rightarrow C$, $A \rightarrow B \leftrightarrow C$
se citesc:
$(\neg A) \rightarrow (B \vee C)$, $(A \wedge B) \rightarrow C$, $A \rightarrow (B \leftrightarrow C)$

4.3 *Concepte semantice în logica propoziţiilor*

Valorizări şi valori de adevăr

Propoziţiile, conform definiţiei 4.2.1, sunt obiecte sintactice generale şi abstracte.
Dorim să interpretăm aceste obiecte abstracte cu ajutorul semanticii. Aceasta înseamnă că suntem interesaţi în a determina condiţiile în care o propoziţie este adevărată sau falsă. Pentru aceasta creăm o structură al cărei domeniu este $\{a, f\}$, unde a şi f semnifică valorile de adevăr "adevărat" şi respectiv, "fals", şi încercăm să atribuim una din aceste două valori fiecărei propoziţii. Metoda folosită în atribuirea valorilor de adevăr, împreună cu definiţiile necesare, constituie semantica **LP.**

Definiţia 4.3.1: O *valorizare* este orice funcţie:
$F : Q \rightarrow \{a, f\}$

unde Q este mulţimea de atomi din limbaj.

O valorizare atribuie astfel valori de adevăr atomilor din limbaj.

Acum, asociem mulţimii de valori de adevăr $\{a, f\}$ operaţii algebrice interne, astfel încât să o transformăm într-o structură algebrică. Operatorii interni ai acestei structuri, respectiv $\sim$, $\sqcup$, $\sqcap$, $\sim>$, $<\sim>$, vor corespunde conectorilor logici $\neg$, $\wedge$, $\vee$, $\rightarrow$ şi $\leftrightarrow$. Similaritatea simbolurilor corespunzătoare nu este întâmplătoare, ci pune în evidenţă corespondenţa existentă. Să considerăm, de exemplu, propoziţia $A \vee B$. Prin atribuirea unor valori de adevăr lui A şi B, interpretăm $A \vee B$ folosind interpretările asociate lui A şi B. Operaţia care realizează acest lucru este $\sqcup$, aşa cum se va vedea mai clar din definiţia următoare.

Definiţia 4.3.2: Operaţiile interne $\sim$, $\sqcup$, $\sqcap$, $\sim>$, $<\sim>$ peste mulţimea $\{a, f\}$ sunt definite de următoarele tabele:

	$\sim$
a	f
f	a

$\sqcup$	a	f
a	a	a
f	a	f

$\sqcap$	a	f
a	a	f
f	f	f

$\sim>$	a	f
a	a	f
f	a	a

$<\sim>$	a	f
a	a	f
f	f	a

În tabelele de definire a operaţiilor $\sqcap$, $\sqcup$, $\sim>$, $<\sim>$, prima coloană defineşte prima componentă iar prima linie defineşte a doua componentă a operaţiei corespunzătoare.

Structura $(\{a, f\}, \sim, \sqcap , \sqcup)$ cu operaţiile $\sim$, $\sqcap$, $\sqcup$ definite de tabelele de mai sus este o algebră booleană cu două valori.

Algebrele booleene sunt foarte importante pentru semantica **LP** cât şi pentru studiul general al bazelor teoretice ale ştiinţei calculatoarelor.

Definiţia 4.3.3: Fie S mulţimea de propoziţii din limbajul logicii propoziţionale. Prin *valorizare de adevăr* sau *valorizare booleană* se înţelege funcţia:

$$V : S \rightarrow \{a, f\}$$

astfel încât, pentru orice $\sigma, \tau \in S$:

(a) dacă σ este un atom, atunci $V(\sigma) \in \{a, f\}$

(b) $V(\neg\sigma) = \sim V(\sigma)$

(c) $V(\sigma \vee \tau) = V(\sigma) \sqcup V(\tau)$

(d) $V(\sigma \wedge \tau) = V(\sigma) \sqcap V(\tau)$

(e) $V(\sigma \rightarrow \tau) = V(\sigma) \sim> V(\tau)$

(f) $V(\sigma \leftrightarrow \tau) = V(\sigma) <\sim> V(\tau)$

Aşa cum se poate vedea din ultima definiţie, valorile valorizărilor de adevăr ale atomilor unei propoziţii sunt legate prin operaţiile algebrice $\sim$, $\sqcap$, $\sqcup$, $\sim>$, $<\sim>$ şi determină valoarea valorizării de adevăr a propoziţiei în cauză.

O valorizare atribuie o valoare de adevăr, *a* sau *f,* atomilor din limbaj. O valorizare de adevăr este extensia unei valorizări la mulţimea de propoziţii a limbajului. Aşa cum se

stabileşte prin teorema următoare, fiecare valorizare de adevăr se extinde unic la o valorizare a mulţimii de propoziţii din limbaj.

Teorema 4.3.4: *Pentru fiecare valorizare F există o unică valorizare de adevăr V, astfel încât V extinde F.*

Demonstraţie: Vom folosi inducţia peste lungimea propoziţiilor.

Fie F o valorizare şi Q o mulţime de atomi. Definim inductiv o valorizare de adevăr în următorul mod:

(a) $V(A) = F(A)$ pentru orice $A \in Q$

Fie σ, φ două propoziţii. Dacă $V(\sigma)$ şi $V(\varphi)$ sunt deja definite de (a), impunem:

(b) $V(\neg\sigma)$

(c) $V(\sigma \vee \varphi) = V(\sigma) \sqcup V(\varphi)$

(d) $V(\sigma \wedge \varphi) = V(\sigma) \sqcap V(\varphi)$

(e) $V(\sigma \rightarrow \varphi) = V(\sigma) \sim> V(\varphi)$

(f) $V(\sigma \leftrightarrow \varphi) = V(\sigma) <\sim> V(\varphi)$

V este evident o valorizare de adevăr ce extinde F.

Ceea ce rămâne acum de demonstrat este faptul că dacă valorizările de adevăr V_1 şi V_2 sunt extinderi ale lui F, atunci $V_1(\varphi) = V_2(\varphi)$ pentru orice propoziţie φ. Proprietatea P utilizată pentru inducţie este:

$P(\varphi)$: $V_1(\varphi) = V_2(\varphi)$

(a) Pentru fiecare simbol propoziţional A, avem $V_1(A) = V_2(A)$, deoarece V_1, V_2 sunt extinderi ale lui F.

(b) Să presupunem că $V_1(\sigma) = V_2(\sigma)$ şi $V_1(\varphi) = V_2(\varphi)$; atunci:

$$V_1(\neg\sigma) = \sim V_1(\sigma) = \sim V_2(\sigma) = V_2(\neg\sigma)$$

$$V_1(\sigma \vee \varphi) = V_1(\sigma) \sqcup V_1(\varphi) = V_2(\sigma) \sqcup V_2(\varphi) = V_2(\sigma \vee \varphi)$$

$$V_1(\sigma \wedge \varphi) = V_1(\sigma) \sqcap V_1(\varphi) = V_2(\sigma) \sqcap V_2(\varphi) = V_2(\sigma \wedge \varphi)$$

Tratând similar celelalte condiţii ale definiţiei 4.3.3, demonstrăm că V_1 şi V_2 au aceeaşi valoare pentru orice propoziţie, deci coincid.

Următorul corolar este o consecinţă directă a teoremei 4.3.4.

Corolarul 4.3.5: *Fie σ o propoziţie conţinând numai atomii $A_1,..., A_k$. Dacă două valorizări de adevăr iau aceleaşi valori în mulţimea $\{ A_1, ... ,A_k \}$, adică:*

$$V_1(A_1) = V_2(A_1), \ldots, V_1(A_k) = V_2(A_k)$$

atunci $V_1(\sigma) = V_2(\sigma)$.

Exemplul 4.3.6: Calculul valorizării de adevăr pe baza unei valorizări:

Fie S mulţimea de propoziţii atomice $S = \{ A_1, A_2 \}$ şi F o valorizare pentru care:

$$F(A_1) = a$$

$$F(A_2) = f$$

Conform teoremei 3.3.4, valorizarea de adevăr V_F care extinde F este unic definită.
Să impunem:

$$V_F(A_1) := F(A_1)$$

$$V_F(A_2) := F(A_2)$$

unde ":=" înseamnă "egal prin definiţie".

Putem acum calcula valorile valorizării de adevăr V_F pentru orice mulţime de propoziţii ce conţin numai atomii A_1 şi A_2:

$$V_F(A_1 \wedge A_2) = V_F(A_1) \sqcap V_F(A_2) = a \sqcap f = f$$

$V_F(A_1 \vee A_2) = V_F(A_1) \sqcup V_F(A_2) = a \sqcup f = a$

$V_F((A_1 \vee A_2) \to A_2) = V_F(A_1 \vee A_2) \sim> V_F(A_2) = a \sim> f = f$ etc.

Propoziţiile **LP** pot fi clasificate în funcţie de valorile de adevăr pe care le primesc.

Definiţia 4.3.7: O propoziţie σ din **LP** este *logic adevărată*, sau *tautologie*, dacă pentru orice valorizare de adevăr V, $V(\sigma) = a$. Acest lucru se notează $\models \sigma$. Vom scrie $\not\models \sigma$ ca să indicăm că σ nu este o tautologie, adică există o valorizare de adevăr V pentru care $V(\sigma) = f$.

O propoziţie σ este *realizabilă* sau *verificabilă* dacă există o valorizare de adevăr V, astfel încât $V(\sigma) = a$.

O propoziţie σ se numeşte *logic falsă* sau *neverificabilă* sau *contradicţie* dacă pentru orice valorizare de adevăr V, $V(\sigma) = f$.

Observaţia 4.3.8: Fie V o valorizare de adevăr şi:

$S_V = \{\sigma \in$ propoziţiilor **LP** $\mid V(\sigma) = a\}$

mulţimea propoziţiilor din limbaj realizabile în valorizarea V. Dacă pentru orice valorizare de adevăr V, propoziţia φ aparţine mulţimii S_V, atunci este o *tautologie*. Fiecare mulţime S_V, cu V o valorizare de adevăr , constituie o lume posibilă a mulţimii propoziţilor **LP**. Fiecare lume posibilă este *aristoteliană*, adică pentru orice propoziţie σ, fie σ, fie $\neg\sigma$ este adevărată într-o anume lume posibilă. ($V(\sigma) = a$ sau $V(\sigma) = f$, deci $V(\neg\sigma) = a$ conform principiului aristotelian al *terţului exclus*, obsevaţia 4.4.4).

Definiţia 4.3.9: Două propoziţii σ şi τ cu proprietatea că $V(\sigma) = V(\tau)$ pentru orice valorizare de adevăr V se numesc *logic echivalente*. Aceasta se notează $\sigma \equiv \tau$.

Exemplul 4.3.10: Propoziţiile $A \vee \neg A$ şi $((A \to B) \to A) \to A$ sunt tautologii.

Demonstraţie: Întâi, vom demonstra că $A \vee \neg A$ este o tautologie. Fie V_1 şi V_2 două valorizări de adevăr, astfel încât:

$V_1(A) = a$ şi $V_2(A) = f$

Obsevăm că :

$V_1(A \vee \neg A) = V_1(A) \sqcup V_1(\neg A) = V_1(A) \sqcup (\sim V_1(A)) = a \sqcup (\sim a) = a \sqcup f = a$

$V_2(A \vee \neg A) = V_2(A) \sqcup V_2(\neg A) = V_2(A) \sqcup (\sim V_2(A)) = f \sqcup (\sim f) = f \sqcup a = a$

O valorizare de adevăr aleatoare a lui A stabileşte pentru A fie valoarea dată de V_1, adică a, fie valoarea dată de V_2, adică f.

Astfel pe baza corolarului 4.3.5, avem:

$V(A \vee \neg A) = V_1(A \vee \neg A) = a$ sau

$V(A \vee \neg A) = V_2(A \vee \neg A) = a$

În consecinţă, propoziţia este adevărată pentru orice valorizare de adevăr V, deci este tautologie.

Vom demonstra acum că $((A \to B) \to A) \to A$ este tautologie. În acest caz, avem patru valorizări diferite F_1, F_2, F_3, F_4 peste $Q = \{A, B\}$:

$F_1(A) = a$	şi	$F_1(B) = a$
$F_2(A) = a$	şi	$F_2(B) = f$
$F_3(A) = f$	şi	$F_3(B) = a$
$F_4(A) = f$	şi	$F_4(B) = f$

Pentru fiecare dintre aceste valorizări există o unică valorizare de adevăr V_k , $k \in \{1,2,3,4\}$, care o extinde. Putem uşor verifica:

$$V_k(((A \to B) \to A) \to A) = a \quad \text{pentru } k \in \{1,2,3,4\}.$$

Orice valorizare de adevăr V a simbolurilor propoziţionale A şi B concordă cu una din valorizările V_k , $k \in \{1,2,3,4\}$. Conform corolarului 4.3.5, avem deci:

$$V(((A \to B) \to A) \to A) = a$$

pentru orice $k \in \{1,2,3,4\}$.

În consecinţă, propoziţia $((A \to B) \to A) \to A$ este adevărată pentru orice valorizare de adevăr, deci este tautologie.

Exemplul 4.3.11: Propoziţia $K \equiv [[(\neg A \wedge B) \to C] \vee D]$ este realizabilă.

Într-adevăr, fie F o valorizare peste simbolurile propoziţionale din K pentru care:

$$F(A) = a \qquad F(B) = a \qquad F(C) = f \qquad F(D) = f$$

Valorizările de adevăr care extind F iau următoarele valori:

$$V(A) = a \qquad V(B) = a \qquad V(C) = f \qquad V(D) = f$$

Atunci:

$$V(\neg A) = \sim V(A) = f$$

$$V(\neg A \wedge B) = V(\neg A) \sqcap V(B) = f \sqcap a = f$$

$$V[(\neg A \wedge B) \to C] = V(\neg A \wedge B) \sim> V(C) = f \sim> f = a$$

$$V[[(\neg A \wedge B) \to C] \vee D] = V[(\neg A \wedge B) \to C] \sqcup V(D) = a \sqcup f = a$$

ceea ce înseamnă că $V(K) = a$.

Propoziţia K nu este o tautologie:

Fie G o valorizare cu:

$$G(A) = f \qquad G(B) = a \qquad G(C) = f \qquad G(D) = f$$

Fie G' valorizarea de adevăr care extinde G. Folosind metoda utilizată pentru a demonstra $V(K) = a$, se poate arăta că $G'(K) = f$. Atunci, conform definiţiei 4.3.7, K nu este tautologie.

4.4 *Tabele de adevăr*

În secţiunea precedentă, am dat definiţia valorizării atomilor din limbaj şi apoi, prin extinderea valorizării de la atomi la propoziţii compuse, am definit valorizările de adevăr. Cu această metodă, dorim să aflăm dacă o propoziţie este tautologie, o contradicţie sau este realizabilă. Dar cu cât formula este mai complicată, cu atât metoda devine mai laborioasă.

Pentru a simplifica abordarea, vom grupa toate valorile posibile ale atomilor unei propoziţii într-o tabelă numită "tabela de adevăr" a propoziţiei. Astfel, pentru propoziţiile compuse $\neg A$, $A \vee B$, $A \wedge B$, $A \to B$ şi $A \leftrightarrow B$, avem următoarele tabele de adevăr:

A	$\neg A$
a	f
f	a

A	B	$A \vee B$
a	a	a
a	f	a
f	a	a
f	f	f

A	B	$A \wedge B$
a	a	a
a	f	f
f	a	f
f	f	f

A	B	$A \to B$
a	a	a
a	f	f
f	a	a
f	f	a

A	B	$A \leftrightarrow B$
a	a	a
a	f	f
f	a	f
f	f	a

De acum înainte, vom foloi tabelele de adevăr de mai sus ca definiţii ale valorilor ce rezultă din utilizarea conectorilor logici, fără a ne mai referi la valorizări şi la valorizări de adevăr.

Observaţia 4.4.1:

(1) Disjuncţia **LP** este inclusivă, adică $A \vee B$ poate lua valoarea a şi atunci când atât A cât şi B au valoarea a, spre deosebire de limbajul curent care este de multe ori exclusiv: de exemplu, se spune " Stau acasă sau mă duc la film " şi prin aceasta se înţelege că numai una din variante este posibilă şi nu amândouă.

(2) Propoziţia $A \to B$ ia valoarea f, numai dacă A are valoarea a şi B are valoarea f. Astfel, dacă A este f, $A \to B$ este a oricare ar fi valoarea lui B. Aceste proprietăţi ale conectorului $\to$ şi partea din **LP** care se bazează pe aceste proprietăţi nu au fost întotdeauna acceptate de diverse şcoli ale logicii.

Să recapitulăm ceea ce am discutat despre tabelele de adevăr ale unei propoziţii:

☻ *Pe baza definiţiei inductive a propoziţiilor cât şi pe baza definiţiei valorilor conectorilor logici, se poate construi tabela de adevăr a unei propoziţii prin atriubuirea de valori atomilor constituenţi ai propoziţiei.*

Exemplul 4.4.2: Tabela de adevăr a propoziţiei $A \wedge B \to C$ este:

A	B	C	$A \wedge B$	$A \wedge B \to C$
a	a	a	a	a
a	a	f	a	f
a	f	a	f	a
a	f	f	f	a
f	a	a	f	a
f	a	f	f	a
f	f	a	f	a
f	f	f	f	a

Pentru tripletul de atomi (A, B, C), tripletul de adevăr (f, a, f) face ca formula $A \wedge B \to C$ să fie adevărată, în timp ce pentru tripletul (a, a, f) formula este falsă. *Tabela de adevăr redusă* a propoziţiei $A \wedge B \to C$ este tabela de adevăr ce rezultă prin eliminarea celei de-a patra coloane ce este coloană auxiliară.

Tabela de adevăr redusă a unei propoziţii ce conţine n atomi constă în 2^n linii şi $n+1$ coloane.

Prin intermediul tabelelor de adevăr se poate determina dacă o propoziţie este adevărată sau falsă. Conform corolarului 4.3.5, dacă tabelele de adevăr pentru două propoziţii au aceleaşi valori de adevăr în coloanele atomilor componenţi şi în ultima coloană, atunci cele două propoziţii sunt logic echivalente.

Exemplul 4.4.3:

(i) Propoziţia $((A \to B) \to A) \to A$ este o tautologie:

A	B	$A \to B$	$((A \to B) \to A)$	$((A \to B) \to A) \to A$
a	a	a	a	a
a	f	f	a	a
f	a	a	f	a
f	f	a	f	a

(ii) propozitia $(P \to Q) \wedge (P \wedge \neg Q)$ este nerealizabilă:

P	Q	$\neg Q$	$P \to Q$	$P \wedge \neg Q$	$(P \to Q) \wedge (P \wedge \neg Q)$
a	a	f	a	f	f
a	f	a	f	a	f
f	a	f	a	f	f
f	f	a	a	f	f

Observaţia 4.4.4: Pe baza definiţiei 4.3.7 avem:

(i) O propoziţie este tautologie dacă şi numai dacă negaţia ei nu este realizabilă;

(ii) O propoziţie este realizabilă dacă şi numai dacă negaţia ei nu este o tautologie;

(iii)O propoziţie care este tautologie este realizabilă în timp ce o propoziţie realizabilă nu este neaparat tautologie;

(iv) Există anumite tautologii de bază ce sunt frecvent utilizate:

(1) $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$	Legea lui De Morgan
(2) $\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$	Legea lui De Morgan
(3) $\neg(\neg A) \leftrightarrow A$	Legea dublei negaţii
(4) $(A \to B) \leftrightarrow (\neg B \to \neg A)$	Legea contrapoziţiei
(5) $(B \to C) \to ((A \to B) \to (A \to C))$	Prima lege a silogismului
(6) $(A \to B) \to ((B \to C) \to (A \to C))$	A doua lege a silogismului
(7) $(A \to (B \to C)) \leftrightarrow ((A \wedge B) \to C)$	Legea transportării
(8) $A \vee \neg A$	Legea terţului exclus

Aristotel a fost primul care a enunţat propoziţiile (5),(6),(8). Propoziţiile (5) şi (6) sunt cunoscute sub numele de legile silogistice ale lui Aristotel.

Metoda determinării valorii de adevăr a unei propoziţii compuse cu ajutorul tabelelor de adevăr este destul de simplă atât timp cât propoziţiile conţin un număr mic de atomi. În cazul propozitiilor cu 3 atomi, tabela de adevăr corespunzătoare are $2^3 = 8$ linii. Tabela de adevăr a unei propoziţii cu 4 atomi are $2^4 = 16$ linii, iar dacă propoziţia are 10 atomi, tabela ei de adevăr are $2^{10} = 1024$ linii!

În plus, nu putem utiliza tabele de adevăr în logica predicatelor. Vom studia în secţiunile următoare metode mai avansate şi mai economice de determinare a valorii de adevăr a unei propoziţii sau a unei mulţimi de propoziţii. Aceste metode vor constitui baza studiului programării logice.

4.5 *Consecinţe şi interpretări*

Pentru exemplul 3.4.2 al tabelei de adevăr a propoziţiei $(A \wedge B) \to C$ am văzut că această propozitie ia valoarea a dacă toţi atomii A, B şi C au valoarea a. Aceasta revine la a spune că validitatea propoziţiei $(A \wedge B) \to C$ a fost o consecinţă a faptului că orice propoziţie din $S = \{A, B, C\}$ a luat valoarea a. Dăm astfel următoarea definiţie:

Definiţia 4.5.1: Fie S o mulţime de propoziţii. O propoziţie σ se numeşte *consecinţă a lui* S (notată cu $S \models \sigma$) dacă pentru orice valorizare de adevăr V, pentru care $V(\varphi) = a$ oricare ar fi $\varphi \in S$, putem deduce că $V(\sigma) = a$.

Mulţimea $Con(S) = \{\sigma \mid S \models \sigma\}$ este mulţimea tuturor consecinţelor lui S. Formal:

$\sigma \in Con(S) \Leftrightarrow$ (Pentru fiecare valorizare de adevăr V)
(pentru fiecare $\varphi \in S$)
($V(\varphi) = a \Rightarrow V(\sigma) = a$)

în loc de "(pentru fiecare $\varphi \in S$)($V(\varphi) = a$)" scriem adeseori $V[S] = \{a\}$ sau chiar, informal, $V[S] = a$.

Observaţia 4.5.2: Simbolurile $\Leftrightarrow$ şi $\Rightarrow$ folosite în definiţia de mai sus cu semnificaţia "dacă şi numai dacă" şi "implică" sunt simboluri ale *metalimbajului*. Metalimbajul este folosit pentru a raţiona despre formulele din **LP** şi pentru a investiga proprietăţile acestora. De aceea când spunem " $\models \varphi$ ", propoziţia φ este o tautologie, exprimăm o judecată despre φ. " $\models \varphi$ " este o *metapropoziţie* a **LP** , respectiv o propoziţie a metalimbajului **LP**.

Exemplul 4.5.3: Fie $S = \{A \wedge B, B \rightarrow C\}$ o multime de propoziţii. Atunci propoziţia C este o consecinţă a lui S, adică $S \models C$.

Demonstraţie: Să presupunem că S este o valorizare de adevăr ce validează toate propoziţiile din S:

$V(A \wedge B) = a$ şi $V(B \rightarrow C) = a$

Atunci avem, conform definitiei 4.3.3:

$V(A) \sqcap V(B) = a$ (1) şi

$V(B) \sim> V(C) = a$ (2)

Atunci, din (1) şi definiţia 4.3.2 avem:

$V(A) = V(B) = a$ (3)

(2) devine astfel:

$a \sim> V(C) = a$ (4)

Pe baza definiţiei 4.3.2 şi (4) putem concluziona că $V(C) = a$. Aceasta înseamnă că orice valorizare de adevăr ce verifică toate propoziţiile din S verifică şi pe C. C este deci o consecinţă a lui S, $S \models C$.

Dacă notăm cu *Taut* mulţimea tuturor tautologiilor, atunci:

Observaţia 4.5.4: $Con(Ø)=Taut$, unde $Ø$ este mulţimea vidă.

Astfel, tautologiile nu sunt consecinţe ale nici unei mulţimi particulare S; ele sunt independente de S, pentru orice mulţime de propoziţii S, şi sunt legate numai de tabelele de adevăr din subparagraful 4.3.

Aşa cum s-a spus înainte **LP** se ocupă de studiul propoziţiilor. Următoarele două definiţii sunt legate de realizabilitatea unei mulţimi de propoziţii.

Definiţia 4.5.5: O mulţime de propoziţii S este (semantic) *consistentă*, dacă există o valorizare de adevăr care să verifice orice propoziţie din S. Formal:

$consistent(S) \Leftrightarrow$ (există o valorizare V)[(pentru orice $\sigma \in S$)($V(\sigma) = a$)]

Faptul că S este consistentă se notează cu $V(S) = a$.

Pentru o mulţime consistentă S, folosim de asemenea termenii *realizabilă* sau *verificabilă,* cu aceeaşi semnficaţie.

S este *inconsistentă*, *irealizabilă* sau *neverificabilă* dacă pentru orice valorizare de adevăr există cel puţin o propoziţie nerealizabilă în S:

$inconsistent(S) \Rightarrow$ (pentru orice V)[(exista $\sigma \in S$)($V(\sigma) = f$)]

Exemplul 4.5.6: Mulţimea de propoziţii $S = \{A \wedge \neg B, A \rightarrow B\}$ este inconsistentă.

Demonstraţie:

Să presupunem că există o valorizare de adevăr V astfel încât:

pentru orice $C \in S$, $V(C) = a$

Atunci:

$V(A \wedge \neg B) = a$ şi $V(A \rightarrow B) = a$

ceea ce înseamnă că :

$V(A) \sqcap V(\neg B) = a$ (1) şi

$V(A) \sim> V(B) = a$ (2)

Din (1) şi definiţia 4.3.2 avem $V(A) = V(\neg B) = a$, deci:

$V(A) = a$ şi $V(B) = f$

Atunci (2) devine:

$a \sim> f = a$.

ceea ce contrazice definiţia4.3.2 unde $a \rightarrow f = f$. În consecinţă, nici o valorizare de adevăr nu poate verifica propoziţiile din S, deci S este inconsistentă.

Definiţia 4.5.7: O valorizare de adevăr ce satisface o mulţime de propoziţii S se numeşte *interpretare* a lui S. Mulţimea tuturor interpretărilor lui S se notează cu *Int(S)*, unde:

$Int(S) = \{V \mid V$ valorizare de adevăr şi pentru orice $\sigma \in S, V(\sigma) = a\}$. Prezentăm acum un corolar despre anumite concluzii utile în ceea ce priveşte consecinţele şi interpretările.

Corolarul 4.5.8: *Pentru mulţimile de propoziţii S, S_1, S_2 avem:*

(1) $S_1 \subseteq S_2 \Rightarrow Con(S_1) \subseteq Con(S_2)$

(2) $S \subseteq Con(S)$

(3) $Taut \subseteq Con(S)$, pentru orice mulţime de propoziţii S

(4) $Con(S) = Con(Con(S))$

(5) $S_1 \subseteq S_2 \Rightarrow Int(S_2) \subseteq Int(S_1)$

(6) $Con(S) = \{ \sigma \mid V(\sigma) = a,$ *pentru orice* $V \in Int(S)\}$

(7) $\sigma \in Con(\{\sigma_1,\ldots, \sigma_n\}) \Leftrightarrow \sigma_1 \rightarrow (\sigma_2 \rightarrow \ldots \rightarrow (\sigma_n \rightarrow \sigma)..) \in Taut$

Demonstraţia este evidentă.

Corolarul 4.5.8 (7) oferă o metodă pentru a determina dacă φ este o consecinţă a unei mulţimi finite de propoziţii S, prin verificarea în doi paşi dacă partea dreaptă a lui (7) este o tautologie. Cu toate acestea, pentru o mulţime infinită S, această metodă ar necesita un număr infinit de paşi. Utilizarea tablourilor semantice devine, în acest caz, mult mai potrivită.

4.6 *Mulţimi adecvate de conectori logici-Forme normale*

Determinarea valorii de adevăr a unei propoziţii, ca şi demonstrarea corectitudinii sau a lipsei de corectitudine a unei mulţimi de propoziţii, depinde adeseori de numărul şi tipul conectorilor ce apar în propoziţii. Cei cinci conectori logici pe care îi folosim sunt conectorii utilizaţi mai frecvent în textele matematice. În cele ce urmează se va demonstra că orice mulţime de conectori logici poate fi exprimată pe baza mulţimii $\{\neg, \wedge, \vee\}$ şi vom demonstra astfel că mulţimea de conectori logici $\{\neg, \wedge, \vee\}$ este suficientă pentru a exprima orice propoziţie din **LP**.

Definiţia 4.6.1: Se spune că o muţime Q de conectori logici este *adecvată* dacă pentru orice propoziţie din **LP** există o propoziţie echivalentă logic care nu conţine conectori diferiţi de cei conţinuţi în Q.

Teorema 4.6.2: Fie $\sigma(A_1, A_2,.., A_k)$ o propoziţie în care apar numai atomii $A_1,.., A_k$. Să considerăm tabela de adevăr redusă a lui σ.

A_1	……	A_λ	……	A_k	$\sigma(A_1,\ldots, A_k)$
σ_{1_1}	…….	σ_{1_λ}	……	σ_{1_k}	σ_1
…. ….	…… ……	…….. ……	…… …….	…… ……	…… …….
σ_{ν_1}	…..	σ_{ν_λ}	……	σ_{ν_k}	σ_ν
….. …..	…… ……	…… ……	…… ……	…… ……	…… …….
$\sigma_{2_1^k}$	….	$\sigma_{2_\lambda^k}$	……	$\sigma_{2_k^k}$	σ_{2^k}

σ_{n_l} reprezintă valoarea de adevăr corespunzătoare liniei n şi coloanei l iar σ_n este valoarea de adevăr a propozitiei $\sigma(A_1, A_2,.., A_k)$ în cea de-a n-a linie.

(1) Să presupunem că în ultima coloană se află cel puţin un a. Vom demonstra că există o propoziţie pentru care tabela de adevăr redusă asociată propoziţiei are aceeaşi ultimă coloană ca tabela de mai sus şi care utilizează numai conectorii $\{\neg, \wedge, \vee\}$. În acest caz, propoziţia va fi logic echivalentă cu $\sigma(A_1, A_2,.., A_k)$.

Pentru orice atom A, A^a reprezintă A şi A^f reprezintă $\neg A$. Din cea de-a n-a linie formăm conjuncţia:

$$t_n: (A_1^{\sigma\nu 1} \wedge \ldots \wedge A_k^{\sigma\nu k})$$

care conţine numai conectorii $\neg$ şi $\wedge$. Fie $l_1,.., l_m$ liniile cu un a pe ultima coloană. Atunci propoziţia :

$$t_{l_1} \vee \ldots \vee t_{l_m}$$

este propoziţia pe care o căutăm deoarece tabela ei de adevăr redusă are un a în ultima coloană, exact în acele linii în care tabela de adevăr redusă a propoziţiei $\sigma(A_1, A_2,.., A_k)$ are un a.

(2) Dacă ultima coloană nu contine nici un a, atunci propoziţia este falsă şi este deci logic echivalentă cu $(A \wedge \neg A)$, unde A este orice simbol propoziţional.

Pe această bază, folosind tehnica din teorema anterioară, putem exprima orice propoziţie cu ajutorul conectorilor $\neg$, $\wedge$, $\vee$.

Definiţie: Propoziţia echivalentă ce rezultă se spune a fi *Forma Normal Disjunctivă* (*FND*). Fie F o propoziţie astfel încât atomii lui F sunt $A_1,\ldots, A_n$. Forma generală a lui F în *FND* este:

$$FND(F): (A_{1_1} \wedge \ldots \wedge A_{1_n}) \vee (A_{2_1} \wedge \ldots \wedge A_{2_n}) \vee \ldots \vee (A_{k_1} \wedge \ldots \wedge A_{k_n}),$$

unde $A_{ij} \in \{A_1,\ldots, A_n\}$ sau $A_{ij} \in (\neg A_1,\ldots,\neg A_n)$, adică A_{ij} sunt atomi sau negarea unor atomi din F şi în fiecare componentă conjuctivă a $FND(F)$, fiecare atom a lui F apare numai o singură dată, negat sau nenegat.

Conceptul dual al lui *FND* este numit *Formă Normal Conjunctivă* (*FNC*) şi are următoarea formă:

$$FNC(F): (A_{11} \vee \ldots \vee A_{1n}) \wedge (A_{21} \vee \ldots \vee A_{2n}) \wedge \ldots \wedge (A_{k1} \vee \ldots \vee A_{kn}).$$

Exemplul 4.6.3: Să se găsească *FND* a propoziţiei F cunoscându-se tabela de adevăr redusă a lui F:

	A	B	C	F
1	*a*	*a*	*a*	***a***
2	*a*	*a*	*f*	***f***
3	*a*	*f*	*a*	***f***
4	*a*	*f*	*f*	***f***
5	*f*	*a*	*a*	***a***
6	*f*	*a*	*f*	***f***
7	*f*	*f*	*a*	***f***
8	*f*	*f*	*f*	***a***

Vom urma metoda descrisă în teorema 4.6.2:

Pasul 1: Găsim toate liniile care au un *a* în ultima coloană. Aceste linii sunt 1,5,8.

Pasul 2: $FND(F) \equiv t_1 \vee t_5 \vee t_8 \equiv (A \wedge B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C)$.

Vom enunţa acum două corolare interesante şi utile referitoare la diverse mulţimi adecvate de conectori logici.

Corolarul 4.6.4: $K_1 = \{\neg, \vee\}$ şi $K_2 = \{\neg, \wedge\}$ sunt mulţimi *adecvate de conectori.*

Demonstraţie: Folosind tabelele de adevăr, putem demonstra că:

$$A \to B \equiv \neg A \vee B \qquad A \leftrightarrow B \equiv (A \to B) \wedge (B \to A) \text{ şi}$$

$$A \wedge B \equiv \neg(\neg A \vee \neg B)$$

În consecinţă, orice propoziţie **LP** poate fi exprimată cu conectorii $\{\neg, \vee\}$. K_1 este deci o mulţime adecvată.

Putem demonstra, de asemenea, că mulţimea K_2 este adecvată, astfel:

$$A \vee B \equiv \neg(\neg A \wedge \neg B) \qquad A \to B \equiv \neg A \vee B \equiv \neg(A \wedge \neg B) \text{ şi}$$

$$A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$$

Exemplul 4.6.5: În afara conectorilor $\vee$, $\neg$, $\wedge$, $\to$, $\leftrightarrow$, putem crea şi alţi conectori, cum ar fi „ $\mid$ ” şi „**:**” definiţi de următoarele tabele de adevăr:

A	B	$A \mid B$
a	*a*	*f*
a	*f*	*a*
f	*a*	*a*

f	*f*	*a*

A	B	*A*:*B*
a	*a*	*f*
a	*f*	*f*
f	*a*	*f*
f	*f*	*a*

Corolarul 4.6.6: Mulţimile $\sum_1 = \{ \mid \}$ şi $\sum_2 = \{:\}$ sunt mulţimi adecvate.

Demonstraţie:

($\sum_1$): Intuitiv $A \mid B$ înseamnă că A şi B nu pot fi simultan adevărate.

Atunci

$$A \mid B \equiv \neg (A \wedge B).$$

(Această echivalenţă poate fi uşor verificată pe baza tabelelor de adevăr).

Atunci:

$$A \mid A \equiv \neg (A \wedge A) \equiv \neg A$$

Şi în acest fel negaţia este exprimată prin $\mid$; în acelaşi fel, pentru conjuncţie avem:

$$A \wedge B \equiv \neg\neg(A \wedge B) \equiv \neg(A \mid B) \equiv (A \mid B) \mid (A \mid B)$$

($\sum_2$): Negaţia devine: $\neg A \equiv \neg(A \vee A) \equiv A : A$, şi conjuncţia:

$$A \wedge B \equiv \neg(\neg A \vee \neg B) \equiv (A : A) : (B : B)$$

Să observăm că $\{ \mid \}$ şi $\{ : \}$ sunt singurele mulţimi care conţin un unic conector logic care sunt mulţimi adecvate. Mai mult, orice mulţime cu doi conectori logici, unul dintre aceştia fiind $\neg$ şi celălalt fiind unul dintre $\vee$, $\wedge$, $\rightarrow$, este adecvată.

Observaţia 4.6.7: Transformarea unei propoziţii σ în alta φ, care este logic echivalentă dar conţine conectori diferiţi de cei din σ, este foarte utilă. De exemplu, în metoda de demonstrare prin rezoluţie, care va fi prezentată în secţiunile următoare, exprimăm toate implicaţiile, adică propoziţii de tipul $\sigma \rightarrow \varphi$, folosind conectorii $\vee$, $\wedge$, $\neg$. Pentru această transformare folosim echivalenţa :

$A \rightarrow B \equiv \neg A \vee B$, care poate fi verificată folosind tabelele de adevăr.

În secţiunile următoare vom descrie metodele de demonstrare utilizate în **LP.** Prezentarea lor va simplifica introducerea în programarea logică.

Lecţia 5 –Tablouri semantice si rezoluţie

5.1 *Tablouri semantice*

Metodele de demonstrare sunt proceduri algoritmice care pot fi urmate pentru a determina dacă o propoziţie este sau nu tautologie sau dacă o mulţime de propoziţii poate fi sau nu poate fi satisfacută. Aceste metode sunt incluse în dezvoltarea teoriei demonstrării automate a teoremelor, teorie care reprezintă o aplicaţie de bază a programării logice.

Prima metodă de demonstrare algoritmică ce va fi prezentată foloseşte tablourile semantice. Gentzen (logician german, 1909-1945) a fost primul care a demonstrat în 1943 că toate tautologiile sunt produse prin aplicarea anumitor reguli, ceea ce revine la a spune că pentru orice tautologie φ există o anumită tautologie ce conduce la φ. Teoria demonstrării a fost utilizată în 1955 de Beth şi Hintikka pentru a crea un algoritm de determinare a caracterului tautologic al unei propoziţii.

Cu ajutorul tablourilor semantice Beth, pe scurt tablouri semantice, putem examina care sunt posibilităţile ca o propoziţie dată să ia valoarea de adevăr a sau valoarea de adevăr f.

Tabloul semantic al unei propoziţii compuse K este construit inductiv, pe baza tablourilor semantice ale propoziţiilor ce apar în K. Vom începe astfel prin a defini ce înseamnă tablouri semantice atomice.

Definiţia 5.1.1:

(1) Fie σ o propoziţie. $f\sigma$ reprezintă aserţiunea "σ este *falsă*", iar $a\sigma$ reprezintă aserţiunea "σ este *adevărată*". $f\sigma$ şi $a\sigma$ se numesc *formule cu semn*.

(2) În conformitate cu definiţia inductivă a propoziţiilor, tablourile sematice atomice ale propoziţiilor σ, σ_1, σ_2 sunt cele prezentate în tabelul următor:

aA 1a)	fA 1b)	$a(\sigma_1 \wedge \sigma_2)$ $\vert$ $a\sigma_1$ $\vert$ $a\sigma_2$ 2a)	$f(\sigma_1 \wedge \sigma_2)$ / \\ $f\sigma_1$ $f\sigma_2$ 2b)
$a(\neg\sigma)$ $\vert$ $f\sigma$ 3a)	$f(\neg\sigma)$ $\vert$ $a\sigma$ 3b)	$a(\sigma_1 \vee \sigma_2)$ / \\ $a\sigma_1$ $a\sigma_2$ 4a)	$f(\sigma_1 \vee \sigma_2)$ $\vert$ $f\sigma_1$ $\vert$ $f\sigma_2$ 4b)
$a(\sigma_1 \rightarrow \sigma_2)$ / \\ $f\sigma_1$ $a\sigma_2$ 5a)	$f(\sigma_1 \rightarrow \sigma_2)$ $\vert$ $a\sigma_1$ $\vert$ $f\sigma_2$ 5b)	$a(\sigma_1 \leftrightarrow \sigma_2)$ / \\ $a\sigma_1$ $f\sigma_1$ $\vert$ $\vert$ $a\sigma_2$ $f\sigma_2$ 6a)	$f(\sigma_1 \leftrightarrow \sigma_2)$ / \\ $a\sigma 1$ $f\sigma_1$ $\vert$ $\vert$ $f\sigma_2$ $a\sigma_2$ 6b)

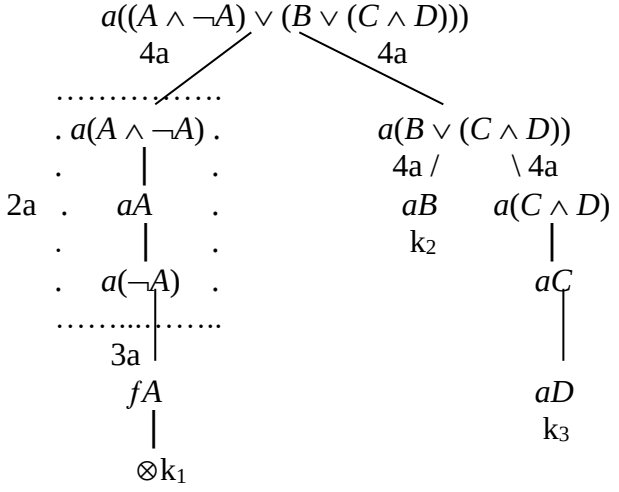
Intuitiv, formula cu semn aA sau fA este privită ca aserţiunea conform căreia A este adevărată, respectiv falsă. Pentru tabloul semantic atomic (4a), aserţiunea "$\sigma 1 \vee \sigma 2$ este adevărată" necesită fie ca σ_1, să fie adevărată, fie ca σ_2 să fie adevărată(ramificare), în timp ce în tabloul(5b), aserţiunea "$\sigma 1 \rightarrow \sigma 2$ este falsă" necesită ca σ_1 să fie adevărată şi σ_2 să fie falsă(secvenţă). Aceasta însemnă că într-un tablou semantic atomic o ramificare reprezintă o disjuncţie, în timp ce o secvenţă reprezintă o conjuncţie. Pentru a construi tabloul semantic al unei propoziţii compuse K, începem prin a scrie formula cu semn aK sau fK, la originea (începutul) tabloului semantic. Dezvoltăm apoi tabloul semantic al lui K conform definiţiei 3.7.1.

Prezentăm în continuare un exemplu ce va fi urmat şi în definiţia formală a regulii generale de construcţie a tablourilor semantice.

Exemplul 5.1.2:

Fie K: $((A \wedge \neg A) \vee (B \vee (C \wedge D)))$ o propoziţie.

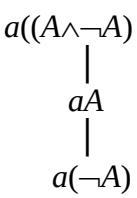
Atomii lui K sunt A, B, C şi D. Începem tabloul semantic cu originea aK.



Acesta este un tablou semantic complet cu 3 ramuri, respectiv trei secvenţe k_1, k_2 şi k_3. Ramurile încep din origine. Ramura stângă, k_1, este contradictorie deoarece conţine formulele cu semn mutual contradictorii aA şi fA. Caracterul contradictoriu al unei ramuri se marchează cu simbolul $\otimes$ la capătul ramurii. Celelalte două ramuri nu sunt contradictorii.

Prin intermediul tabloului semantic al propoziţiei K, observăm că ipoteza aK este adevărată în anumite condiţii, de exemplu aB pe ramura k_2 sau aD pe ramura k_3. Poate fi însă şi falsă, cum ar fi cazul ramurii k_1.

Sunt uşor de identificat tablourile sematice atomice constituente ale tabloului semantic anterior. De exemplu:



în zona încerciută cu linie punctată este tabloul semantic atomic (2a).

Definim acum conceptele necesare pentru construcţia tablourilor semantice.

Definiţia 5.1.3:

(1) *Nodurile* unui tablou semantic sunt toate formulele cu semn care apar în tablou.

(2) Un nod al unui tablou semantic se numeşte *folosit* dacă apare ca origine a unui tablou semantic atomic; în caz contrar, nodul se numeşte *nefolosit.*

(3) O ramură a unui tablou semantic se numeşte *contradictorie* dacă pentru o anumită propoziţie σ, $a\sigma$ şi $f\sigma$ sunt noduri ale ramurii respective.

(4) Un tablou semantic se numeşte *complet* dacă nici *una* din ramurile necontradictorii din tablou nu are noduri nefolosite; în caz contrar se numeşte tablou *incomplet.*

(5) Un tablou semantic este *contradictoriu* dacă toate ramurile sale sunt contradictorii.

Definiţia 5.1.4: Construcţia inductivă a tablourilor semantice:

Vom construi un tablou semantic pentru o propoziţie K după cum urmează:

Vom începe cu formula cu semn aK (sau fK) ca origine a tabloului şi continuăm inductiv .

Pasul n: Avem un tablou semantic atomic T_n.

Pasul n+1: *Tabloul* semantic atomic T_n va fi extins la tabloul T_{n+1} prin utilizarea anumitor noduri ale T_n care nu vor mai fi utilizate în continuare. Dintre nodurile neutilizate ale lui T_n aflate cel mai aproape de origine, selectăm pe cel mai din stânga. Fie X acest nod.

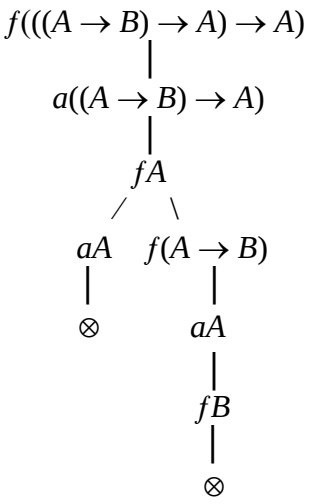
Extindem acum fiecare ramură necontradictorie ce trece prin X prin concatenarea unui tablou atomic semantic T_{n+1} (în practică nu se va mai scrie nodul X din nou deoarece el este deja un nod al ramurii necontradictorii).

Construcţia se termină atunci când fiecare ramură necontradictorie nu mai are noduri nefolosite.

Următorul exemplu, ce prezintă un tablou semantic contradictoriu, clarifică modul de construire definit anterior.

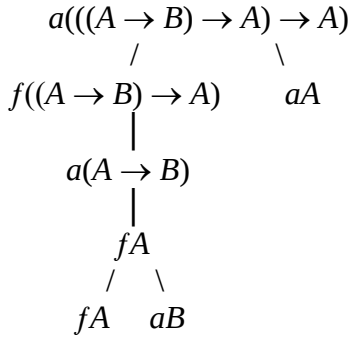
Exemplul5.1.5: Legea lui Peirce: $((A \to B) \to A) \to A$

(1) începem cu aserţiunea conform căreia $((A \to B) \to A) \to A$ este falsă:



Ipoteza că legea lui Peirce este falsă a condus la un tablou semantic contradictoriu, în consecinţă formula este adevărată.

(2) Dacă începem cu presupunerea că propoziţia este adevărată, concluzia rămâne în continuare aceeaşi:



Observăm că nu există nici o ramură contradictorie. Atunci, dacă fA sau aB şi fA, sau aA, $((A \to B) \to A) \to A$ devine adevărată. Chiar dacă fB, atunci fie aA fie fA se îndeplineşte. Deci legea lui Peirce este logic adevărată.

Intuitiv:

◇ *Dacă un tablou semantic complet cu originea fK este contradictoriu, acesta înseamnă că am încercat toate modurile posibile în care propoziţia K poate deveni falsă şi am eşuat; în consecinţă, K este o tautologie.*

Această idee cheie este exprimtă de următoarea definiţie:

Definiţia 5.1.6: *O demonstraţie Beth* a unei propoziţii K este un tablou semantic contradictoriu complet cu originea fK. Un tablou semantic contradictoriu complet cu originea aK se numeşte *o respingere Beth* a lui K.

Se spune că propoziţia K este *demonstrabilă Beth* dacă K admite o demonstraţie Beth. K se numeşte *respinsă Beth* dacă există o respingere Beth pentru K.

Faptul că propoziţia K este demonstrabilă Beth se notează cu $\vdash_B K$.

Aşa cum vom demonstra în 6.1.7 şi în teorema 6.1.9, orice tautologie este demonstrabilă Beth (completitudinea demonstrabilităţii Beth) şi invers, orice propoziţie care este demonstrabilă Beth este o tautologie (corectitudine a demonstrabilităţii Beth).

5.2 *Demonstraţii axiomatice*

Logica propoziţiilor, ca şi alte sisteme matematice, poate fi prezentată, de asemenea, ca un sistem axiomatic cu axiome logice şi reguli de derivare în loc de tablouri sematice. Axiomele sistemului sunt o parte dintre tautologii şi o regulă de derivare *R derivă* o propoziţie σ dintr-o secvenţă de propoziţii $\sigma_1,\sigma_2,\ldots,\sigma_n$. Vom face acum o scurtă descriere a unei astfel de prezentări axiomatice a **LP**.

Definiţia 5.2.1: Axiomele. Privim ca axiomă oricare din propoziţiile de forma următoare:

(1) $\varphi \to (\tau \to \varphi)$

(2) $(\varphi \to (\tau \to \sigma)) \to ((\varphi \to \tau) \to (\varphi \to \sigma))$

(3) $(\neg\varphi \to \neg\tau) \to (\tau \to \varphi)$

Trebuie remarcat că propoziţiile φ, σ, τ pot fi înlocuite cu orice alte propoziţii. În acest fel avem scheme axiomatice (sau scheme de axiome) ce conduc la un număr nelimitat de axiome. Se poate verifica uşor că toate aceste axiome sunt formule bine formate ale **LP** şi, bineînţeles, tautologii.

Definiţia 5.2.2: Regula *Modus Ponens*:

Folosim o regulă de derivare, regula *Modus Ponens*, care spune că propoziţia τ poate fi derivată din propoziţiile φ şi $\varphi \to \tau$. Regula *Modus Ponens* (*mode* conform lui Diogenes Laertius) se notează cu:

$$\left.\begin{array}{c}\varphi \\ \varphi \to \tau \\ \hdashrule{2cm}{0.4pt}{} \\ \tau\end{array}\right\} \qquad (1)$$

sau chiar cu :

$$\varphi, \varphi \to \tau \vdash \tau \qquad (2)$$

Pentru (1), care este o definiţie caracteristică a unei reguli logice, linia orizontală separă ipotezele de concluzie. Pentru (2), simbolul "$\vdash$" reprezintă derivarea în sistemul axiomatic. Considerăm cele trei axiome ale definiţiei 5.2.1 ca formule derivate în acest sistem axiomatic. Pe baza acestor trei axiome şi a regulii *Modus Ponens* se pot produce noi propoziţii. Următorul exemplu arată cum se pot aplica axiomele şi *Modus Ponens* pentru a deriva formula **LP** $A \to A$.

Exemplul 5.2.3: Să se demonstreze că $\vdash A \to A$.

Demostratie:

$$\vdash A \to ((B \to A) \to A) \qquad (1)$$

pe baza primei axiome. Pe baza celei de-a doua axiome avem:

$$\vdash A \to ((B \to A) \to A) \to [((A \to (B \to A)) \to (A \to A)] \qquad (2)$$

Din (1), (2) şi *Modus Ponens* rezultă:

$$\vdash (A \to (B \to A)) \to (A \to A) \qquad (3)$$

Dar $\vdash A \to (B \to A)$, conform primei axiome şi, cu regula *Modus Ponens*, (3) conduce la $\vdash A \to A$. Astfel, propoziţia $A \to A$ este derivată în sistemul axiomatic descris.

Următoarea teoremă ne permite să înlocuim una din subformulele unei propoziţii **LP** cu o subformulă logic echivalentă.

Teorema 5.2.4: *Substituţia echivalenţelor*:

Dacă o propoziţie $\sigma \leftrightarrow \sigma_1$ *este derivată în* **LP** *şi este o subformulă a propoziţiei* φ, *atunci propoziţia* $\varphi \leftrightarrow \varphi_1$ *poate fi de asemenea derivată în* **LP**, *unde* φ_1 este *propoziţia obtinută din* φ *prin înlocuirea a zero, una sau mai multe apariţii ale propoziţiei* σ *cu echivalentul ei* σ_1. *Formal*:

$$\vdash \sigma \leftrightarrow \sigma_1 \quad \Rightarrow \quad \vdash \varphi \leftrightarrow \varphi_1$$

Vom enunţa acum definiţia formală a demonstraţiei unei propozitii pe baza metodei axiomatice.

Definiţia 5.2.5: Fie S o mulţime de propoziţii.

(1) O demonstraţie din S este o secvenţă finită de propoziţii $\sigma_1, \sigma_2, \ldots, \sigma_n$

astfel încât pentru fiecare $1 \le i \le n$:

(i) σ_i aparţine lui S, sau

(ii) σ_i este o axiomă, sau

(iii) σ_i urmează din σ_j, σ_k , $1 \le j,k \le i$, prin aplicarea regulii *Modus Ponens*.

(2) O propoziţe σ este S-demonstrabilă dintr-o mulţime de propoziţii S dacă există o demonstraţie $\sigma_1, \sigma_2, \ldots, \sigma_n$ din S astfel încât σ_n coincide cu σ. Formal se scrie $S \vdash \sigma$.

(3) Propoziţia σ este demonstrabilă dacă $\vdash \sigma$, adică dacă σ este derivată în sistemul axiomatic din definiţia 5.2.1 prin utilizarea regulii *Modus Ponens*. Evident, conceptul de propoziţie S-demonstrabilă coincide cu conceptul de propoziţie demonstrabilă pentru $S = \varnothing$.

Exemplul 5.2.6: Vom prezenta demonstraţia formulei $\neg B \to (C \to A)$ din $S = \{A\}$:

(1)	A	$A \in S$
(2)	$A \to (C \to A)$	axioma 1
(3)	$(C \to A)$	*Modus Ponens* din (1) şi (2)
(4)	$(C \to A) \to (\neg B \to (C \to A))$	axioma 1
(5)	$\neg B \to (C \to A)$	*Modus Ponens* din (3) şi (4)

Să observăm că dacă o propoziţie σ este demonstrabilă din S şi S este o mulţime infinită, atunci σ este demonstrabilă dintr-o submulţime finită a lui S, deoarece demonstraţiile sunt întotdeauna finite.

Următoarea teoremă, pe care o dăm fără demonstraţie, este fundamentală în teoria demonstrării.

Teorema 5.2.7: *Teorema deducţiei*:
Fie S o mulţime de propoziţii şi fie K, L două propoziţii **LP**. Atunci:

$$S \cup \{K\} \vdash L \quad \Leftrightarrow \quad S \vdash K \to L$$

Axiomele **LP** sunt deseori numite axiome logice. De obicei, definim o teorie prin extinderea axiomatizării **LP** printr-o multime S de axiome suplimentare care caracterizează teoria respectivă. Teoremele teoriei S sunt elementele mulţimii $\{\varphi \mid S \vdash \varphi\}$ (vezi observaţia 5.2.8).

Observaţia 5.2.8:

(1) ***Sistemul axiomatic***: Axiomele *LP* şi regula *Modus Ponens* constituie sistemul axiomatic Frege-Lukasiewicz. Frege (german, 1848-1925) a fost primul filozof şi logician care a definit un limbaj formal adecvat logicii. Lukasiewicz (polonez, 1878-1956) s-a ocupat de axiomatizarea **LP.** .

(2) ***Modus Ponens***: Dacă σ şi $\sigma \to \tau$ sunt demonstrabile Beth, atunci σ şi $\sigma \to \tau$ sunt logic adevărate şi astfel şi τ este adevarată (de ce?). Există o metodă algoritmică ce reprezintă o demonstraţie Beth a lui τ din demonstraţiile Beth ale lui σ şi $\sigma \to \tau$. Această metodă este o aplicare a teoremei lui Gentzen (Gentzen Hauptsatz), dar demonstraţia ei este în afara cadrului acestei cărţi.

(3**)** ***Teoreme***: O teoremă este orice propoziţie ce apare într-o demonstraţie. Aceasta înseamnă că teoremele teoriei S sunt exact elementele mulţimii $\{\varphi \mid S \vdash \varphi\}$. De obicei, concluzia apare ca ultima propoziţie din secvenţă dar, de fapt, orice parte iniţială din

demonstraţie este de asemenea o demonstraţie.

(4) ***Selecţia axiomelor***: Metoda axiomatică de demonstrare este corectă şi completă, aşa cum se va arăta în capitolele următoare. Sistemul axiomatic ales este astfel complet, ceea ce înseamnă că orice tautologie poate fi demonstrată din axiome prin aplicarea succesivă a regulii *Modus Ponens*.

(5) ***Demonstrabilitate Beth***: Deoarece axiomele sunt propoziţii logic adevărate, ele sunt de asemenea demonstrabile Beth şi regula *Modus Ponens* conduce de la propoziţii logic adevărate la propoziţii care sunt tot logic adevărate. În acest fel, orice teoremă este demonstrabilă Beth.

(6) ***Axiome şi reguli***: Putem înlocui una sau mai multe din axiomele sistemului axiomatic **LP** cu reguli. De exemplu, cea de a treia axiomă:

$$(\neg\varphi \rightarrow \neg\tau) \rightarrow (\tau \rightarrow \varphi)$$

poate fi înlocuită cu regula:

$$\frac{\neg\varphi \rightarrow \neg\tau}{\tau \rightarrow \varphi}$$

Alegerea între axiome şi reguli este de obicei o opţiune bazată pe evaluarea personală a necesităţilor teoretice specifice.

(7) ***Derivarea din axiome***: Pentru a demonstra o propoziţie din axiome, trebuie să încercăm diverse combinaţii pentru a putea determina combinaţia adecvată de propoziţii în aplicarea axiomelor şi a regulii *Modus Ponens*. O singură derivare, de exemplu:

$$\vdash (A \rightarrow B) \rightarrow ((C \rightarrow A) \rightarrow (C \rightarrow B))$$

devine astfel dificilă şi consumatoare de timp, chiar având indicaţia că prima şi cea de a doua axiomă trebuie utilizate ambele, de două ori.

În schimb, demonstraţiile Beth, aşa cum au fost fixate de definiţia 5.1.6, oferă o metodă algoritmică sistematică de producere a rezultatului. Pentru acest motiv preferăm utilizarea demonstraţiior Beth.

5.3 *Rezoluţie*

Terminologie şi notaţie în programarea logică

Metoda rezoluţiei este cea mai eficace metodă de demonstraţie algoritmică a **LP** precum şi, după cum vom vedea în capitolul al patrulea, a logicii predicatelor. Ea constituie metoda de demonstraţie pe care se bazează limbajul de programare logică PROLOG. Metoda rezoluţiei este o metodă de demonstraţie prin respingere, la fel ca demonstraţiile Beth. În ansamblu, ea are un şir de asemănări cu metoda de demonstraţie Beth, dar este mai potrivită pentru scrierea de programe logice, unde limbajul de programare este foarte apropiat de limbajul **LP**.

Pentru a expune rezoluţia, trebuie să definim unele concepte de bază ale programării logice moderne.

Definitia 5.3.1: Un literal este orice atom sau negaţia acestuia.

De exemplu: $\neg A$, B, $\neg C$ sunt literali.

Ştim că putem dezvolta orice propoziţie din **LP** într-o formă normală conjunctivă **FNC**, care este echivalentă cu propoziţia iniţială. O componentă a **FNC** este de fapt o

disjuncţie de literali, astfel că în fiecare disjuncţie nici un literal nu apare mai mult decât o dată.

Prezentăm acum un algoritm pentru construcţia unei **FNC** pentru o propoziţie dată, care este mult mai rapid decât construirea tabelei de adevăr a propoziţiei, urmată de alegerea coloanelor etc..

Acest algoritm rezultă a fi o aplicaţie bazată pe:

(i) legile lui De Morgan:

$\neg (A \wedge B) \leftrightarrow \neg A \vee \neg B$ şi $\neg (A \vee B) \leftrightarrow \neg A \wedge \neg B$

(ii) proprietăţile de asociativitate ale lui $\wedge$ şi $\vee$:

$(A \wedge B) \wedge C \leftrightarrow A \wedge (B \wedge C)$ şi $(A \vee B) \vee C \leftrightarrow A \vee (B \vee C)$

(iii) proprietăţile de comutativitate ale lui $\wedge$ şi $\vee$:

$A \wedge B \leftrightarrow B \wedge A$ şi $A \vee B \leftrightarrow B \vee A$

(iv) proprietăţile de distributivitate ale lui $\wedge$ faţă de $\vee$ şi ale lui $\vee$ faţă de $\wedge$:

$A \wedge (B \vee C) \leftrightarrow (A \wedge B) \vee (A \wedge C)$ şi

$A \vee (B \wedge C) \leftrightarrow (A \vee B) \wedge (A \vee C)$

(v) propoziţiile:

$A \vee A \leftrightarrow A$ $A \wedge A \leftrightarrow A$ $A \wedge (B \vee \neg B) \leftrightarrow A$

$A \vee (B \wedge \neg B) \leftrightarrow A$ şi $\neg\neg A \leftrightarrow A$

precum şi pe teorema substituţiei echivalenţelor. (Ca exerciţiu demonstraţi că propoziţiile de mai sus sunt tautologii.)

Această metodă va fi prezentată cu ajutorul unui exemplu.

Exemplul 5.3.2 : Dezvoltaţi propoziţia S într-o **FNC** unde:

S: $\neg ((A \vee B) \wedge (\neg A \vee \neg B) \wedge C$

Pasul 1: Mutăm negaţiile spre interiorul parantezelor folosind legile lui De Morgan:

a: $S \leftrightarrow [\neg (A \vee B) \vee \neg (\neg A \vee \neg B)] \wedge C$

b: $S \leftrightarrow [(\neg A \wedge \neg B) \vee (\neg\neg A \wedge \neg\neg B)] \wedge C$

Pasul 2: Folosim proprietăţile de asociativitate şi comutativitate pentru a aduce la un loc literalii aceluiaşi atom. Apoi putem simplifica dublele negaţii, termenii dubli de forma $A \vee A$ şi $A \wedge A$ precum şi termenii inutili de forma $B \wedge \neg B$ sau $B \vee \neg B$ folosind teorema substituţiei echivalenţelor:

$S \leftrightarrow [(\neg A \wedge \neg B) \vee (A \wedge B)]$

Pasul 3**:** Conform proprietăţilor de distributivitate avem:

$S \leftrightarrow [((\neg A \wedge \neg B) \vee A) \wedge ((\neg A \wedge \neg B) \vee B)] \wedge C$

Continuăm prin repetarea paşilor al doilea şi al treilea. până ce stabilim **FNC** finală:

Pasul 1': $S \leftrightarrow ((\neg A \wedge \neg B) \vee A) \wedge ((\neg A \wedge \neg B) \vee B) \wedge C$

Pasul 3': $\leftrightarrow (\neg A \wedge A) \wedge (\neg B \vee A) \wedge (\neg A \vee B) \wedge (\neg B \vee B) \wedge C$

PasuI 2': $\leftrightarrow (\neg B \vee A) \wedge (\neg A \vee B) \wedge C$

care este **FNC** a lui S pe care o cautăm.

Ultima formă a lui S este o conjuncţie de disjuncţii de literali şi este echivalentă cu formula iniţială. Acest algoritm se termină în general când se obţine următoarea forma a lui S:

$$(A_1^1 \vee A_2^1 \vee \ldots \vee A_k^1) \wedge \ldots \wedge (A_1^v \vee A_2^v \vee \ldots \vee A_k^v) \qquad (*)$$

unde elementele lui $\{ A_1^1, A_2^1, \ldots, A_k^1, \ldots, A_1^v, A_2^v, \ldots, A_k^v\}$ sunt literali.

În contextul metodei de demonstraţie prin rezoluţie, exprimarea unei propoziţii ca o mulţime de literali se vădeşte foarte utilă. De exemplu, propoziţia din prima paranteză din (*) devine:

$\{ A_1, A_2, \ldots, A_v \}$

Considerăm că o astfel de mulţime reprezintă o disjuncţie de literali şi anume, o propoziţie din **LP**.

Vom da acum definiţia riguroasă a formei unei propoziţii exprimată pe baza teoriei mulţimilor.

Definitia 5.3.3: Disjuncţia unui număr finit de *literali* poate fi reprezentată conform *teoriei mulţimilor* ca o mulţime ale cărei elemente sunt literalii în cauză. Această mulţime se numeşte *clauză*. Astfel, o clauză este echivalentă cu o propoziţie disjunctivă din **LP**.

Din motive tehnice vom introduce şi noţiunea de *clauză vidă*, o clauză care nu conţine literali şi este întotdeauna *neverificabilă.* Clauza vidă se notează prin $\square$.

Definiţia 5.3.4: Conjuncţia unui număr finit de *clauze* poate fi reprezentată *conform teoriei mulţimilor* ca o mulţime ale cărei elemente sunt aceste clauze. Aceasta mulţime se numeşte *mulţime de clauze*. O mulţime de clauze constituie astfel o conjuncţie de disjuncţii, în speţă o propoziţie conjunctivă din **LP**.

Exemplul 5.3.5: Mulţimea de clauze:

$$\{ \underbrace{\{A, B\}}_{1}, \underbrace{\{\neg B, \neg C\}}_{2}, \underbrace{\{D\}}_{3} \}$$

reprezintă propoziţia:

$$(\underbrace{(A \vee B)}_{1} \wedge \underbrace{(\neg B \vee \neg C)}_{2} \wedge \underbrace{D}_{3})$$

Observaţia 5.3.6:

(1) Este evident că o valorizare de adevăr verifică o mulţime de clauze dacă verifică fiecare clauză din mulţime. De exemplu, fie $S = \{\{A, B\}, \{\neg C\}\}$ o multime de clauze şi fie V o valorizare de adevăr astfel încât:

$$V(A) = V(B) = V(C) = a.$$

Atunci, V nu verifică S întrucât nu verifică una din componentele sale si anume pe $\{\neg C\}$.

(2) Fireşte putem considera şi mulţimea de clauze vidă $\{\varnothing\}$, care trebuie să nu se confunde cu clauză vidă $\square$. În mod formal, fiecare valorizare de adevăr verifică mulţimea de clauze vidă, deoarece validează fiecare din componentele sale (nu există nici o propoziţie conţinută în clauzele din $\{\varnothing\}$(vezi demonstraţia corolarului 4.5.4).

Dimpotrivă, orice mulţime de clauze care conţine clauza vidă nu poate fi verificată de nici o valorizare de adevăr, deoarece $\square$ nu este verificabilă.

În mod intuitiv, mulţimea de clauze vidă arată că nu există *nici o "aserţiune"*(propoziţie) privind "lumea" (mulţimea de propoziţii), în timp ce clauza vidă arată că avem cel puţin o propoziţie privind "lumea" noastră, clauza $\square$, care creează întotdeauna contradicţii, întrucât o face inconsistentă, şi anume neverificabilă.

În programarea logică, precum şi în majoritatea versiunilor de PROLOG, predomină reprezentarea de mai sus.

Fie S propoziţia:

$$A_1 \vee \ldots \vee A_k \vee (\neg B_1) \vee \ldots \vee (\neg B_l)$$

unde $A_1, \ldots, A_k, B_1, \ldots, B_l$ sunt atomi. Atunci avem:

$$S \leftrightarrow A_1 \vee \ldots \vee A_k \vee \neg(B_1 \wedge \ldots \wedge B_l) \qquad \text{conform De Morgan}$$
$$\leftrightarrow (B_1 \wedge \ldots \wedge B_l) \rightarrow (A_1 \vee \ldots \vee A_k) \qquad \text{conform } (\neg B \vee A) \leftrightarrow (B \rightarrow A)$$

şi, în sfârsit:

$S \leftrightarrow ((A_1 \vee \ldots \vee A_k) \leftarrow (B_1 \wedge \ldots \wedge B_l))$ (1)

Pentru folosirea lui $\leftarrow$ drept conector logic vezi şi observaţia 4.2.8. Dacă dorim acum ca în locul conectorilor logici $\wedge$, $\vee$ şi $\leftarrow$ să folosim simbolurile analoage "**,**" (virgula), "**;**" (punctul şi virgula) şi "**:-**" (simbolul "gâtului"), atunci S poate fi reprezentată echivalent prin:

A_1 **;**… **;** A_k **:-** B_1**,**…**,** B_l (2)

Dacă în propoziţia (2) este valabil k= 1, atunci avem:

A_1 **:-** B_1**,**…**,** B_l (3)

Definţia 5.3.7: Fiecare clauză de forma (3) este o clauză Horn. Atomul A este capul sau scopul lui S, iar componentele conjunctive B_1**,**…**,** B_l sunt *coada* sau *corpul* sau subscopurile lui S.

*Interpretarea intuitivă a unei clauze Horn este că pentru ca un scop A să fie valid, trebuie ca şi subscopurile B_1**,**…**,** B_l să fie valide.*

Definiţia 5.3.8: Dacă într-o clauză de forma (2) k = 0, atunci clauza:

:- B_1**,**…**,** B_l (4)

se numeşte *scop de program* sau *scop definit* (determinat).

Dacă l = 0, clauza:

A_1 **:-** (5)

se numeşte *clauză unitate* sau *fapt*.

Observatia 5.3.9:

(1) O mulţime de clauze S dată, poate fi considerată în sens larg drept o bază de date, deoarece clauzele din S reprezintă informaţii privind relaţiile dintre atomii pe care îi conţin.

(2) Verificarea scopului A în clauza A **:-** B_1**,**…**,** B_l este dedusă din validarea subscopurilor B_1**,**…**,** B_l . Într-un astfel de caz se spune că scopul A reuşeste sau că există o demonstraţie a lui A. În caz contrar, se spune că A eşuează.

(3) Forma (4) a unei clauze Horn, denotând absenţa unui scop, afirmă că cel puţin unul dintre B_i ,$1 \le i \le l$ eşuează. Forma (5) a unei clauze Horn înseamnă că A_1 reuşeste întotdeauna. În acest caz, A_1 constituie o afirmaţie, un fapt al bazei de date.

Privind în ansamblu, *rezoluţia* este o regulă deductivă prin care, într-o clauză, putem deduce o propoziţie din alte două propoziţii.

Să considerăm ca fiind date următoarele clauze:

$C_1 = \{ A_1,\ldots, A_{k_1}, \neg B_1,\ldots, \neg B_{l_1} \}$

$C_2 = \{ D_1,\ldots, D_{k_2}, \neg F_1,\ldots, \neg F_{l_2} \}$

unde $A_1,\ldots, A_{k_1}, \neg B_1,\ldots, \neg B_{l_1}, D_1,\ldots, D_{k_2}, \neg F_1,\ldots, \neg F_{l_2}$ sunt atomi. Să presupunem că A_1 coincide cu F_1.

Putem atunci rescrie cele două clauze după cum urmează:

$C_1 = \{ A_1\} \cup C_1'$ unde $C_1' = \{ A_2,\ldots, A_{k_1}, \neg B_1,\ldots, \neg B_{l_1} \}$

$C_2 = \{ \neg A_1\} \cup C_2'$ unde $C_2' = \{ D_1,\ldots, D_{k_2}, \neg F_2,\ldots, \neg F_{l_2} \}$

Atunci, regula rezoluţiei pe care vrem să o stabilim va trebui să ne permită să producem drept deducţie clauza următoare:

$C = C_1' \cup C_2'$

Cu alte cuvinte:

fiind date: $C_1 = \{ A_1\} \cup C_1^{'}$ şi $C_2 = \{ \neg A_1\} \cup C_2^{'}$

concluzia: $C_1^{'} \cup C_2^{'} = (C_1 - \{A_1\}) \cup (C_2 - \{\neg A_1\})$ (*)

Putem considera că cele două clauze, C_1 şi C_2, "sunt în conflict" sau "se ciocnesc" pentru că C_1 conţine literalul A_1 iar C_2 literalul $\neg A_1$. Înlăturarea cauzei de conflict conduce la clauza (*), care rezolvă ciocnirea. Metoda îşi datorează numele acestei rezolvări (rezoluţii). Putem defini acum riguros metoda rezoluţiei.

Definiţia 5.3.10: Rezoluţia, o descriere formalizată:

Fie C_1 şi C_2 două clauze şi fie L un literal astfel încât $L \in C_1$ şi $(\neg L) \in C_2$. Putem atunci deduce **r***ezolventul* D al lui C_1 şi C_2:

$$D = (C_1 - \{L\}) \cup (C_2 - \{\neg L\})$$

Exemplul 5.3.11: Să considerăm următoarele clauze:

$$\{\neg A, B\}$$
$$\{A, C\}$$

Aplicând rezoluţia, putem deduce: $\{B, C\}$

Sensul intuitiv al folosirii unei astfel de reguli devine foarte clar când reexprimăm clauza precedentă conform formulărilor clasice din **LP**:

propoziţii date: $(\neg A \vee B)$

$(A \vee C)$

concluzia: $(B \vee C)$

Această regulă este o aplicare a tautologiei:

$$(\neg A \vee B) \wedge (A \vee C) \rightarrow (B \vee C)$$

Din teorema completitudinii 3.10.9 ştim că tautologiile se pot deduce folosind axiomele şi *Modus Ponens*. Astfel:

$$\vdash (\neg A \vee B) \wedge (A \vee C) \rightarrow (B \vee C)$$

Atunci, din teorema deducţiei (teorema 5.2.7), obţinem :

$$(\neg A \vee B) \wedge (A \vee C) \vdash (B \vee C)$$

Regula rezoluţiei este deci derivabilă în **LP**.

Exemplul 5.3.12:

fiind date: $C_1 = \{P, Q\}$

$C_2 = \{\neg P, \neg Q\}$

concluzia: $D = \{Q, \neg Q\}$

Dacă avem o mulţime care conţine mai mult decât două clauze, putem introduce noţiunea de mulţime rezolvent:

Definiţia 5.3.13: Fie $S = \{C_1, C_2, \ldots, C_n\}$ *o* multime de clauze. Atunci mulţimea:

$$R(S) = S \cup \{D \mid D \text{ este rezolventul clauzelor } C_i, C_j \in S,\ i \neq j,\ 1 \leq i,j \leq n\}$$

este *rezolventul* lui S.

Firește, putem continua cu aplicarea metodei, luând succesiv mulţimile următoare:

$$R^0 = S, \quad R^1(S) = R(S), \quad R^2(S) = R(R(S)), \ldots, R^n(S) = R(R^{n-1}(S)),$$

şi, în final:

$$R^*(S) = \bigcup_{n=1}^{\infty} R^n(S) = \{C_i \mid C_i \in R^j(S) \quad \text{şi} \quad j \in N\}$$

unde C_i sunt clauzele conţinute în al j-lea rezolvent al lui S.
De notat că $R^*(S)$ este o mulţime finită dacă şi numai dacă S este finită.

Exemplul 5.3.14: Fie S o mulţime de clauze:

$$S = \{\underbrace{\{A, \neg B, \neg C\}}_{1}, \underbrace{\{B, D\}}_{2}, \underbrace{\{\neg A, \neg D\}}_{3}\}$$

Aplicând regula rezoluţiei perechilor de clauze ale lui S, avem:

1	$\{A, \neg B, \neg C\}$	2	$\{B, D\}$	3	$\{\neg A, \neg D\}$
2	$\{B, D\}$	3	$\{\neg A, \neg D\}$	1	$\{A, \neg B, \neg C\}$
4	$\{A, D, \neg C\}$	5	$\{B, \neg A\}$	6	$\{\neg B, \neg C, \neg D\}$

şi, în final:

$$R(S) = \{\underbrace{\{A, \neg B, \neg C\}}_{1}, \underbrace{\{B, D\}}_{2}, \underbrace{\{\neg A, \neg D\}}_{3}, \underbrace{\{A, D, \neg C\}}_{4}, \underbrace{\{B, \neg A\}}_{5}, \underbrace{\{\neg B, \neg C, \neg D\}}_{6}\}$$

Observaţia 5.3.15:

(1) În exemplul 5.3.12, am ales să aplicăm rezoluţia cu ajutorul literalului P. Am fi putut să facem aceasta cu ajutorul lui Q, deoarece şi Q este evident o cauză de conflict.

(2) Este intuitiv că la fiecare aplicare a rezoluţiei, dacă o valorizare de adevăr verifică pe C_1 şi C_2, atunci verifică şi rezolventul lor D. De asemenea, oricând dacă o valorizare de adevăr verifică S, verifică şi $R(S)$.

(3) De notat că rezolventul D al lui C_1 şi C_2 cuprinde mai puţină informaţie decât C_1 şi C_2. Aceasta se vede clar din următorul exemplu.

Exemplul 5.3.16. Fie $S = \{\{A, B\}, \{\neg B\}\}$ o mulţime de clauze. Atunci, prin rezoluţie avem :

date: $C_1 = \{A, B\}$
$C_2 = \{\neg B\}$
———————
rezoluţia: $D = \{A\}$

Aplicând rezoluţia asupra lui S, producem $D = \{A\}$, care nu conţine nici o informaţie despre literalul B.

Vom da acum definiţia riguroasă a demonstraţiilor prin metoda rezoluţiei.

Definiţia 5.3.17: Fie S o mulţime de clauze. O demonstraţie prin rezoluţie din S este o secvenţă finită de clauze $C_1, C_2, \ldots, C_n$, astfel încât pentru fiecare C_i, $i = 1, \ldots, n$, avem:

$C_i \in (S)$ sau $C_i \in R(\{C_j, C_k\})$ $1 \le j, k \le i \le n$.

O clauză C este *demonstrabilă prin rezoluţie dintr-o mulţime de clauze* S, formalizat $S \vdash_R C$, dacă există o demonstraţie prin rezoluţie din S a cărei ultimă clauză este C. Evident, $C \in R^*(S)$

Exemplul 5.3.18. Aflaţi toţi rezolvenţii mulţimii de clauze:

$S = \{\{A, B\}, \{\neg A, \neg B\}\}$

Să numerotăm toate clauzele în S:

1. $\{A, B\}$
2. $\{\neg A, \neg B\}$

3. $\{B, \neg B\}$ din 1 şi 2
4. $\{A, \neg A\}$ din 1 şi 2

Atunci:

$R^1(S) = \{\{A, B\}, \{\neg A, \neg B\}, \{B, \neg B\}, \{A, \neg A\}\}$

La sfârşit:

$R^*(S) = R^0(S) \cup R^1(S) = \{\{A, B\}, \{\neg A, \neg B\}, \{B, \neg B\}, \{A, \neg A\}\}$

Clauze de tipul $\{A, \neg A\}$, adică $A \vee \neg A$, sunt tautologii.

Exemplul 5.3.19: Se dă următoarea propoziţie:

$$S: ((A \leftrightarrow (B \rightarrow C)) \wedge (A \leftrightarrow B) \wedge (A \leftrightarrow \neg C)$$

Demonstraţi că S nu este verificabilă.

Demonstraţie:

Pasul 1: Se stabileşte **FNC** a lui S:

$$S \leftrightarrow ((A\rightarrow(B\rightarrow C))\wedge((B\rightarrow C)\rightarrow A)\wedge(A\rightarrow B)\wedge(B\rightarrow A)\wedge(A\rightarrow\neg C)\wedge(\neg C\rightarrow A)$$

$$\leftrightarrow (\underbrace{\neg A \vee \neg B \vee C}_{1}) \wedge (\underbrace{B \vee A}_{2}) \wedge (\underbrace{\neg C \vee A}_{3}) \wedge (\underbrace{\neg A \vee B}_{4}) \wedge (\underbrace{\neg B \vee A}_{5}) \wedge$$

$$\wedge (\underbrace{\neg A \vee \neg C}_{6}) \wedge (\underbrace{C \vee A}_{7})$$

Pasul 2: Se alcătuieşte mulţimea de clauze corespunzătoare:

$$S = \{\{\underbrace{\neg A, \neg B, C}_{1}\}, \{\underbrace{B, A}_{2}\}, \{\underbrace{\neg C, A}_{3}\}, \{\underbrace{\neg A, B}_{4}\}, \{\underbrace{\neg B, A}_{5}\}, \{\underbrace{\neg A, \neg C}_{6}\}, \{\underbrace{C, A}_{7}\}\}$$

Pasul 3: Se stabilesc feluriţii rezolvenţi:

8. $\{A\}$ din 2 şi 5
9. $\{\neg A, \neg B\}$ din 1 şi 6
10. $\{\neg A\}$ din 4 şi 9
11. $\square$ din 8 şi 10

(vezi şi definiţia 5.3.3. Literalul $\neg A$ este eliminat, iar clauza 11 nu conţine nici un literal). Întrucât clauza vidă aparţine rezolventului conform 11, mulţimea de clauze S nu este verificabilă. Astfel, propozitia S nu este verificabilă.

Lecţia 6- CORECTITUDINE SI COMPLETITUDINE

6.1 *Corectitudinea şi completitudinea tablourilor*

În secţiunile următoare vom da teoremele de bază privind corectitudinea şi completitudinea metodelor de demonstraţie pe care le-am prezentat.Vom începe cu corectitudinea şi completitudinea demonstraţiilor Beth.

Pentru fiecare din definiţiile sau teoremele următoare care conţin noţiunile de "*demonstraţie Beth*" şi "*logic adevărat*", există definiţii şi teoreme duale, corespunzând noţiunilor duale de "*respingere Beth*" şi respectiv "*logic fals*".

Vom demonstra că toate propoziţiile demonstrabile Beth sunt, adevărate (corectitudine) şi, reciproc, că oricare propoziţie logic adevărată este demonstrabilă Beth (completitudine). Demonstraţiile pe care le vom prezenta sunt inductive asupra lungimii unei propoziţii sau asupra lungimii unui tablou semantic. Am descris deja schema de inducţie pentru propoziţii. Acum vom descrie schema de inducţie generală pentru tablouri semantice.

Definiţia 6.1.1: Schema de inducţie pentru tablouri semantice.

Fie P *o* proprietate a unui tablou semantic oarecare T, simbolic $P(T)$. Dacă demonstrăm că.:

(a) fiecare tablou semantic atomic are proprietatea P,

(b) dacă P este o proprietate a tabloului semantic T iar dacă T' este un nou tablou semantic format prin concatenarea unui tablou semantic atomic la capătul uneia din ramurile lui T, atunci P este şi o proprietate a lui T' ,

putem deduce atunci ca P *este o* proprietate a tuturor tablourilor semantice, adică $P(T)$ este *valabilă pentru fiecare tablou semantic* T.

Pentru un tablou semantic T, inducţia este realizată pe lungimea lui T, în speţă pe numărul de tablouri semantice atomice din T.

Analogia dintre schema de inducţie pentru propoziţii, definiţia 4.2.2, şi schema de inducţie pentru tablouri semantice este evidentă.

Exemplul 6.1.2: Fie P proprietatea "*numărul de noduri al unui tablou semantic este mai mare sau egal cu numărul de ramuri*".

Demonstraţie:

(a) În toate tablourile semantice atomice numărul de noduri este mai mare sau egal cu numărul de ramuri.

(b) Să presupunem că în tabloul semantic T, numărul de noduri este mai mare sau egal cu numărul de ramuri. Mergem mai departe concatenând un tablou semantic la capătul uneia din ramurile lui T. Dacă nodul de concatenare este $a(\neg\sigma)$ sau $f(\sigma)$, atunci avem un nod în plus în timp ce numărul de ramuri a rămas acelaşi. Pentru orice altfel de nod de concatenare vor fi cel mult două noi ramuri şi cel puţin două noduri noi, conform definiţiei 5.1.1. Astfel, în noul tablou semantic T', numărul de noduri este iarăşi mai mare sau egal cu numărul de ramuri.

Dăm acum unele definiţii şi leme auxiliare privind teoremele de corectitudine şi completitudine pentru demonstraţiile Beth.

Definiţia 6.1.3: Fie K o ramură a unui tablou semantic T şi fie $P = \{P_1,\ldots, P_n\}$ mulţimea nodurilor lui K, unde pentru fiecare propoziţie σ, ori $P_i = a\sigma$ ori $P_i = f\sigma$. Atunci, valorizarea de adevăr V concordă cu ramura K dacă pentru fiecare $P_i \in P$:

$P_i = a\sigma \Rightarrow V(\sigma) = a$ şi $P_i = f\sigma \Rightarrow V(\sigma) = f$

Lema 6.1.4: Fie V o valorizare de adevăr care concordă cu originea unui tablou semantic, ceea ce înseamnă că dacă originea este $a\sigma$, atunci $V(\sigma) = a$, iar dacă originea este $f\sigma$, atunci $V(\sigma) = f$. Atunci V concordă cu o ramură a tabloului semantic.

Lema 6.1.5: Lema lui Hintikka:

Fie K o ramură necontradictorie a unui tablou semantic complet. Definim o atribuire de adevăr şi astfel, valorizarea de adevăr corespunzătoare, după cum urmează:

$V(A) = a$ dacă aA este nod al lui K

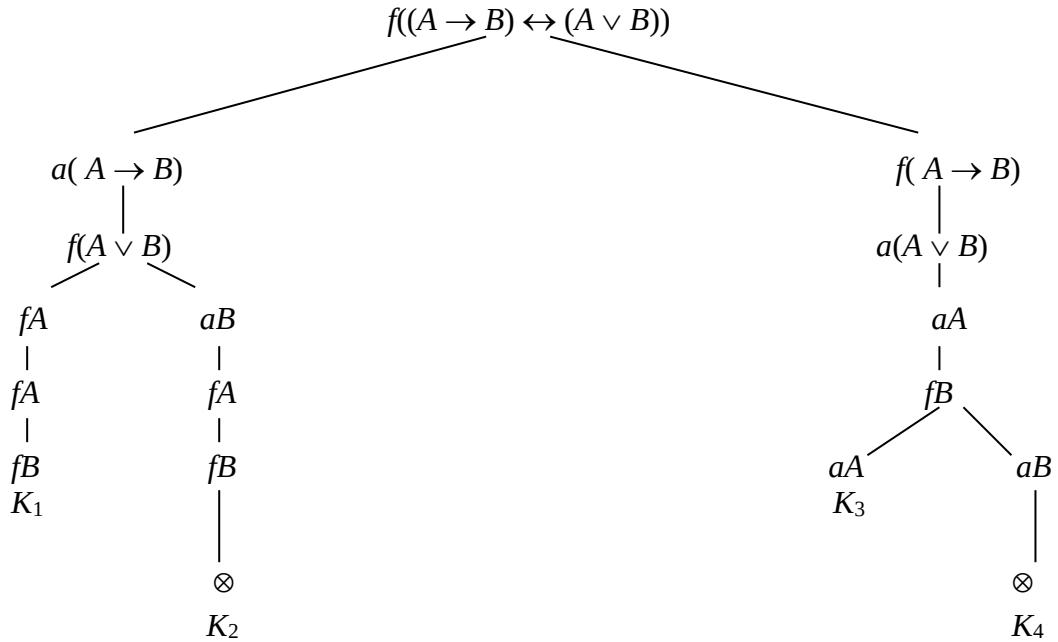
$V(A) = f$ dacă aA nu este nod al lui K.

Atunci V concordă cu ramura K.

* **Lema lui Hintikka** ne oferă practic un algoritm pentru construirea unui contraexemplu la aserţiunea că o propoziţie este logic adevărată. Să presupunem că se dă o propoziţie σ. Construim atunci un tablou semantic complet cu fσ drept origine. Dacă tabloul semantic este contradictoriu, atunci propoziţia este într-adevăr, logic adevărată. Dacă tabloul semantic nu este contradictoriu, atunci are cel puţin o ramură necontradictorie K. Lema lui Hintikka ne indică modul în care să construim pe baza lui K o valorizare de adevăr, astfel încât V(σ) = f.

Să examinăm acest procedeu cu un exemplu.

Exemplul 6.1.6: Aflaţi o valorizare de adevăr V astfel încât:



Ramurile K_2 şi K_4 sunt contradictorii. Putem folosi oricare din ramurile necontradictorii K_1 şi K_3 pentru a aplica lema lui Hintikka. Avem, de exemplu, valorizările de adevăr V_1 şi V_3 cu:

$V_1(A) = f$ $V_1(B) = f$ şi $V_3(A) = a$ $V_3(B) = f$

astfel încât:

$V_1((A \to B) \leftrightarrow (A \vee B)) = V_3((A \to B) \leftrightarrow (A \vee B)) = f$

Asta înseamnă că am găsit două valorizări de adevăr care dau propoziţiei $(A \to B) \leftrightarrow (A \vee B)$ valoarea de adevăr f.

Teorema 6.1.7: Teorema de corectitudine:

Dacă o propoziţie σ *este demonstrabilă Beth, atunci este şi logic adevărată.. Formalizat*:

$$\vdash_B \sigma \Rightarrow \models \sigma$$

Demonstraţie: Dacă propoziţia σ nu este logic adevărată, atunci există o valorizare de adevăr astfel încât $V(\sigma) = f$. Conform lemei 3.10.4, fiecare tablou semantic cu $f\sigma$ la origine are cel puţin o ramură K care concordă cu V şi, prin urmare, nu este contradictorie (de ce?). Aşadar, σ nu este demonstrabilă Beth.

Observaţia 6.1.8: În demonstraţia teoremei 6.1.7 am folosit în metalimbajul din axioma a treia:

$$(\neg P_1 \Rightarrow \neg P_2) \Rightarrow (P_2 \Rightarrow P_1)$$

În loc de a demonstra $P_2 \Rightarrow P_1$ adică dacă σ este demonstrabilă Beth, atunci σ este logic adevărată, am demonstrat că $\neg P_1 \Rightarrow \neg P_2$, cu alte cuvinte, dacă σ nu este logic adevărată, atunci σ nu este demonstrabilă Beth. Această metodă de demonstraţie este folosită destul de des şi se numeşte demonstraţie *indirectă* sau prin *contrapoziţie*. (Metoda de demonstraţie directă ar însemna să se demonstreze nemijlocit $P_2 \Rightarrow P_1$)

Teorema 6.1.9: Teorema de completitudine:

Dacă o propoziţie σ *este logic adevărată, atunci este şi demonstrabilă Beth. Formalizat*:

$$\models \sigma \Rightarrow \vdash \sigma$$

Demonstraţie: Dacă propoziţia σ este logic adevărată, atunci pentru fiecare valorizare de adevăr V, este valabil $V(\sigma) = a$. Să presupunem că nu există o demonstraţie Beth pentru σ. Construim un tablou semantic complet cu o origine $f\sigma$. Acest tablou semantic trebuie să aibă o ramură necontradictorie. Conform lemei 6.1.4, există o valorizare de adevăr V definită adecvat care concordă cu această ramură şi, prin. urmare, cu originea $f\sigma$. Dar atunci $V(\sigma) = f$, ajungând la o contradicţie. Aşadar, există o demonstraţie Beth pentru σ.

Din demonstraţia de completitudine rezultă evident că, dacă am fi încercat să construim o demonstraţie Beth pentru o propoziţie σ (în speţă un tablou semantic complet având drept origine $f\sigma$) şi am fi eşuat, adică dacă tabloul semantic complet construit are cel puţin o ramură necontradictorie, atunci, exact ca în lema 6.1.5, poate fi definită o valorizare de adevăr care este un contraexemplu la aserţiunea că σ este logic adevărată. Cu alte cuvinte, *construind un tablou semantic al lui* σ *avem garanţia fie de a obţine o demonstraţie a lui* σ*, fie un contraexemplu la aserţiunea că* σ *este adevărată.*

6.2 Deducţii din ipoteze

Teorema de compactitate

Demonstrarea automată de teoreme din ipoteze şi date este un obiectiv de seamă al programării logice. În subparagraful 4.5 am discutat consecinţele unei mulţimi de ipoteze S. O propozitie σ s-a numit c*onsecinţă* a lui S, $S \models \sigma$, dacă fiecare valorizare de adevăr care face valide toate propoziţiile din S atribuie şi lui σ valoarea de adevăr a. Putem acum defini ce înseamnă deducerea unei propoziţii dintr-o mulţime de propoziţii şi de ipoteze.

Definiţia 6.2.1: Fie:

$\sigma, \varphi_1, \varphi_2,.....,\varphi_n,.....$

un şir finit sau infinit de propoziţii. Se spune că σ este o *deducţie Beth* din $\varphi_1, \varphi_2,.....,\varphi_n,.....$, dacă există un tablou semantic contradictoriu construit după cum urmează:

Pasul 0: începem cu $f\sigma$ drept origine;

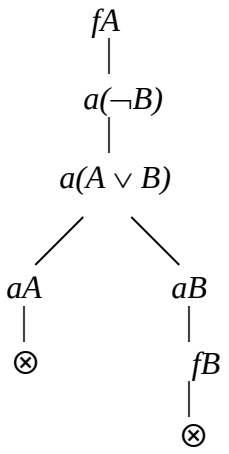
Pasul P_{2n}: plasăm $a\varphi_n$ la capătul fiecărei ramuri necontradictorii;

Pasul P_{2n+1}: aplicăm regulile de dezvoltare tabloului semantic T_{2n} de la pasul precedent.

Dacă şirul de propoziţii este finit, această construcţie poate să nu se termine niciodată. σ este o deducţie Beth numai dacă construcţia se termină şi dacă se obţine un tablou semantic contradictoriu. Atunci, în mod intuitiv, nu există o valorizare care să atribuie valoarea de adevăr a tuturor propoziţiilor φ_n folosite în construirea tabloului semantic şi care să atribuie valoarea de adevăr f lui σ. Dacă şirul iniţial este finit, atunci construcţia se va termina cu siguranţă, oferind un tablou semantic complet.

Să ilustrăm construcţia de mai sus cu un exemplu.

Exemplul 6.2.2: Dorim să demonstrăm că propoziţia A este o deducţie Beth din propoziţiile $\neg B$ şi $(A \vee B)$. Începem cu fA şi concatenăm succesiv aserţiunile $a(\neg B)$ şi $a(A \vee B)$, ca în tabloul de mai jos:



Examinăm pe $a(A \vee B)$. Ramura din stânga este contradictorie şi nu mai trebuie examinată mai departe. În ramura din dreapta examinăm pe $a(\neg B)$. Şi această ramură este contradictorie. Aşadar, A este o deducţie Beth din $\neg B$ şi $(A \vee B)$.

Cele două teoreme care urmează se referă la şiruri finite de ipoteze. Ele corespund teoremelor de corectitudine şi completitudine din secţiunea precedentă.

Teorema 6.2.3: Corectitudinea deducţiilor:

Dacă o propoziţie σ *este o deducţie Beth din* $\varphi_1, \varphi_2,.....,\varphi_n,\ldots\ldots$, *atunci* σ *este o consecinţă a* $\varphi_1, \varphi_2,.....,\varphi_n$. *Formalizat*:

$$\{\varphi_1, \varphi_2,.....,\varphi_n\} \vdash_B \sigma \quad \Rightarrow \quad \{\varphi_1, \varphi_2,.....,\varphi_n\} \models \sigma$$

Demonstraţie: *Indirect*:

Să presupunem că σ nu este o consecinţă a $\varphi_1, \varphi_2,.....,\varphi_n$. Atunci există o valorizare de adevăr V cu proprietatea:

$$V(\varphi_1) = \ldots = V(\varphi_n) = a$$

în timp ce $V(\sigma) = f$. Conform lemei 6.1.4 fiecare tablou semantic cu o origine $f\sigma$ are cel puţin o ramură care concordă cu V şi deci necontradictorie. Atunci σ nu este demonstrabil Beth din $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$.

Teorema 6.2.4: Completitudinea deducţiilor:

Dacă o propoziţie σ *este o consecinţă a* $\varphi_1, \varphi_2, \ldots, \varphi_n$, *atunci* σ *este o deducţie Beth din* $\varphi_1, \varphi_2, \ldots, \varphi_n$. *Formalizat*:

$$\{\varphi_1, \varphi_2, \ldots, \varphi_n\} \models \sigma \qquad \Rightarrow \qquad \{\varphi_1, \varphi_2, \ldots, \varphi_n\} \vdash_B \sigma$$

Demonstraţie: Să presupunem că σ nu este o deducţie Beth din $\varphi_1, \varphi_2, \ldots, \varphi_n$. Atunci, putem să construim un tablou semantic complet necontradictoriu având drept origine $f\sigma$, fiecare ramură a acestuia conţinând nodurile $a\varphi_1, a\varphi_2, \ldots, a\varphi_n$. Aşadar, există o ramură a acestui tablou care este necontradictorie şi, conform lemei 3.10.5, există o valorizare de adevăr V care concordă cu această ramură. Prin urmare avem, $V(\varphi_1) = V(\varphi_2) \ldots = V(\varphi_n) = a$ şi $V(\sigma) = f$. Totuşi, aceasta este o contradicţie, deoarece σ este o consecinţă a $\varphi_1, \varphi_2, \ldots, \varphi_n$.

Observaţia 6.2.5: Teoremele de corectitudine 6.2.3 şi de completitudine 6.2.4 sunt valabile chiar dacă şirul $\{\varphi_k, k \in N\}$ are un număr infinit de termeni.

Acum putem formula, demonstra şi aplica teorema de compacitate a **LP.** Mai întâi vom da o definiţie de bază.

Definiţia 6.2.6: Un şir $\sigma_1, \sigma_2, \ldots, \sigma_n$, se numeşte *verificabil* dacă există o valorizare de adevăr V astfel încât $V(\sigma_1) = V(\sigma_2) = \ldots = V(\sigma_n) = a$. În acest caz se spune că V *verifică* (sau *satisface*) şirul $\sigma_1, \sigma_2, \ldots, \sigma_n$.

Exemplul 6.2.7: Dacă $A_1, A_2, \ldots$ este un şir de atomi, atunci şirul infinit $A_1, A_2, A_1 \wedge A_2, A_3, A_1 \wedge A_3, A_2 \wedge A_3, \ldots$ este *verificabil*. (O valorizare de adevăr V astfel încât $a = V(A_1) = V(A_2) = \ldots = V(A_3) = \ldots$ verifică şirul.) Dimpotrivă, $A_1, A_2, (A_1 \rightarrow A_3), (\neg A_3)$ este un şir finit care nu este verificabil. Într-adevăr, dacă presupunem că este verificabil, atunci există o valorizare de adevăr V astfel încât:

$$V(A_1) = V(A_2) = V(A_1 \rightarrow A_3) = V(\neg A_3) = a$$

Cu alte cuvinte, $V(A_3) = f$ în timp ce $V(A_1) \sim\sim> V(A_3) = a$, ceea ce înseamnă că $V(A_1) = f$, iar aceasta este o contradicţie.

Înainte de a formula teorema de compacitate, vom da o definiţie şi o lemă care sunt necesare demonstraţiei în cauză.

Definiţia 6.2.8:

(1) Fie X,Y două noduri ale unui tablou semantic. Y este un *descendent* al lui X dacă există o ramură care trece prin X şi Y, iar X este mai apropae de origine decât Y. Y este un *descendent imediat* al lui X dacă Y este un descendent al lui X şi nu există nici un alt nod între X şi Y în ramura care trece prin X şi Y. Se spune că X este ***adecvat*** dacă are un număr infinit de descendenţi.

(2) Un tablou semantic se numeşte a fi de *grad finit* dacă fiecare din nodurile sale are numai un număr *finit* de descendenţi imediaţi.

Lema 6.2.9: Lema lui König:

Un tablou de grad finit cu număr infinit de noduri are cel puţin o ramură infinită.

Teorema 6.2.10: Teorema de compacitate:

Fie $\sigma_1, \sigma_2, \ldots, \sigma_n$ un şir finit de propoziţii. Dacă pentru fiecare *n,* şirul finit $\sigma_1, \sigma_2, \ldots, \sigma_n$ este verificabil, atunci şirul $\sigma_1, \sigma_2, \ldots, \sigma_n$ este verificabil.

Demonstraţie: Vom descrie inductiv construirea unui tablou semantic care poate fi infinit:

Pasul 1: Începeţi cu $f(\neg\sigma_1)$ drept origine.

Pasul P_{2n}: Plasaţi $a(\sigma_n)$ la capătul fiecărei ramuri necontradictorii a tabloului T_{2n-1}, construind astfel tabloul T_{2n}.

Pasul P_{2n+1}: Alegeţi nodul nefolosit al lui T_{2n} care se află cel mai la stânga şi aplicaţi regulile de dezvoltare din definiţia 3.7.1.

Să presupunem că construcţia se termină dacă, la un pas impar $2n+1$, toate ramurile sunt contradictorii. (Dacă nu sunt contradictorii, trebuie să continuăm construcţia cu pasul următor $2n+1$.) Dacă se întâmplă aşa, avem un tablou semantic contradictoriu cu originea $f(\neg\sigma_1)$ şi cu $\sigma_1, \sigma_2, \ldots, \sigma_n$ ca ipoteze. Conform teoremei 6.2.3, ştim că $\neg\sigma_1$ este o consecinţă a lui $\sigma_2, \sigma_3, \ldots, \sigma_n$ şi, prin urmare, $\sigma_1, \sigma_2, \ldots, \sigma_n$ nu este verificabil, ceea ce contrazice premisele.

Aşadar construcţia nu se termină niciodată şi astfel, va rezulta un tablou semantic cu un număr infinit de noduri. Acest tablou semantic este de grad finit deoarece fiecare din nodurile sale are un număr finit de descendenţi direcţi. Aplicând lema lui König avem, aşadar, un tablou semantic cu o ramură infinită *k,* unde orice $a\sigma_i$ este un nod al lui *k* pentru orice $i = 1,2,\ldots$

Definim acum o valorizare de adevăr *V* astfel încât:

$V(A) = a \Leftrightarrow A$ este un simbol propoziţional, iar aA este un nod al lui *k.*

Apoi, din lema lui Hintikka putem deduce că $V((\sigma_i) = a$ pentru orice *i,* iar $\sigma_1, \sigma_2, \ldots, \sigma_n$ este verificabil.

6.3 *Corectitudinea şi completitudinea demonstraţiilor axiomatice*

Prezentăm acum teoremele de corectitudine, completitudine şi compacitate ale metodei axiomatice. Demonstraţiile acestor teoreme nu se dau aici, dar cititorul poate să le afle în majoritatea cărţilor clasice de logică.

Teorema 6.3.1: σ *este demonstrabilă dintr-o mulţime S de propoziţii dacă şi numai dacă* σ *este o consecinţă a lui S. Formalizat:*

$$S \vdash \sigma \Leftrightarrow S \models \sigma.$$

Corolarul 6.3.2:

σ *este demonstrabilă din* $S = \varnothing \Leftrightarrow \sigma$ *este logic adevărată.*

Teorema 6.3.3: Compacitate:

S este o mulţime de propoziţii verificabilă dacă şi numai dacă orice submulţime finită a lui S este verificabilă.

6.4 *Corectitudinea şi completitudinea rezoluţiei*

În continuare, ne ocupăm de teoremele de corectitudine şi completitudine ale metodei rezoluţiei.

Teorema 6.4.1: Corectitudinea şi completitudinea rezoluţiei:

S este o mulţime de clauze neverificabilă dacă şi numai dacă $R^*(S)$ *conţine clauza vidă. Formalizat:*

$\square \in R^*(S) \quad \Leftrightarrow \quad S$ *este verificabilă*

Pentru demonstraţia teoremei de corectitudine trebuie să demonstrăm mai întâi o lemă auxiliară.

Lema 6.4.2: *Dacă* $\{C_1, C_2\}$ *este o mulţime de clauze veificabilă şi dacă* C *este rezolventul lui* C_1, C_2 *atunci* C *este verificabil.*

Demonstraţie: Pentru un literal oarecare p, avem $C_1 = \{p\} \cup C_1^{'}$, $C_2 = \{\neg p\} \cup C_2^{'}$ şi $C = C_1^{'} \cup C_2^{'}$. Dacă V este o valorizare de adevăr care verifică $\{C_1, C_2\}$, atunci $V(p) = a$ sau $V(\neg p) = a$. Să presupunem că $V(p) = a$. Deoarece $V(C_2) = a$ şi $V(\neg p) = f$, atunci $V(C_2^{'}) = a$ şi astfel $V(C) = a$. Dacă $V(\neg p) = a$, pur şi simplu îl înlocuim pe C_2 cu C_1.

Teorema 6.4.3: Corectitudinea rezoluţiei:

Dacă $\square \in R^*(S)$, S *este neverificabilă. Formalizat:*

$\square \in R^*(S) \quad \Leftrightarrow \quad S$ *este neverificabilă*

Demonstraţie: Fie $C_1,\ldots,C_k$ o demonstraţie prin rezoluţie a lui S. Atunci, prin folosirea inductivă a lemei de mai sus, demonstrăm că orice valorizare de adevăr care verifică pe S, verifică şi pe C_1. Dacă concluzia este clauza vidă, atunci C_k coincide cu $\square$. Deoarece clauza vidă $\square$ este neverificabilă, S este neverificabilă.

Pentru a demnostra completitudinea rezoluţiei vom folosi următoarea lemă auxiliară.

Lema 6.4.4: *Fie* S *o mulţime de clauze neverificabilă în care apar numai literalele* $A_1, A_2, \ldots, A_k$. *Fie* S^{k-1} *mulţimea finită de clauze care sunt demonstrabile prin rezoluţie şi în care singurele propoziţii atomice care apar sunt* $A_1, A_2 \ldots, A_{k-1}$. *Atunci* S^{k-1} *este neverificabilă.*

Teorema 6.4.5: Completitudinea rezoluţiei:

Daca S este o mulţime de clauze neverificabilă, atunci clauza vidă este demonstrabilă prin rezoluţie din S. Formalizat:

S neverificabilă $\Rightarrow \quad \square \in R^*(S)$

Demonstraţie: Conform definiţiei 5.3.3, S conţine un număr finit de literali. Fie aceşti literali $A_1, A_2, \ldots, A_k$. Aplicând lema precedentă o dată, conchidem că S^{k-1} este neverificabilă. Aplicând aceeaşi lemă de k ori, conchidem că S este neverificabilă, deoarece în clauzele sale nu apar nici un fel de propoziţii atomice iar $S^0 \subseteq R^*(S)$. Deci, $\square \in S^0 \subseteq R^*(S)$.

Observaţia 6.4.6: Dacă σ este o propoziţie demonstrabilă prin rezoluţie din mulţimea S de propoziţii, atunci σ este demonstrabilă din S (definiţia 5.2.5). Avem atunci conform teoremei completitudinii 6.3.1:

$$S \vdash_R \sigma \quad \Leftrightarrow \quad S \vdash \sigma \quad \Leftrightarrow \quad S \models \sigma \qquad (1)$$

Conform teoremelor de corectitudine şi completitudine 6.1.6 şi 6.1.8, (1) se poate generaliza:

$$S \vdash_R \sigma \quad \Leftrightarrow \quad S \vdash_B \sigma \quad \Leftrightarrow \quad S \vdash \sigma \quad \Leftrightarrow \quad S \models \sigma$$

Exerciţii rezolvate

1. Fie propoziţiile:

A_1: „3 este număr prim”;
A_2: „15 se împarte cu 3”;
A_3: „2 se împarte cu 3”;
A_4: „13 se împarte cu 3”.

a) Stabiliţi o valorizare V pentru propoziţiile de mai sus.
b) Fie W valorizarea de adevăr care îl extinde pe V. Să se calculeze $W((A_1 \wedge A_2) \to (A_3 \vee A_4))$.

Rezolvare:

a) Fie $V(A_1)=a$, $V(A_2)=a$, $V(A_3)=f$ şi $V(A_4)=f$.

b) $W((A_1 \wedge A_2) \to (A_3 \vee A_4)) = W(A_1 \wedge A_2) \sim> W(A_3 \vee A_4) =$

$W(A_3 \vee A_4) = (V(A_1) \cap V(A_2)) \sim> (V(A_3) \cup V(A_4)) =$

$= (a \cap a) \sim> (f \cup f) = a \sim> f = f.$

2. Demonstraţi că următoarea propoziţie este o tautologie:

$P = (A \wedge \neg A) \to A$;

Rezolvare:

Formăm tabloul:

A	$\neg A$	$A \wedge \neg A$	P
a	f	f	a
f	a	f	a

3. Demonstraţi că următoarea propoziţie este o contradicţie:

$P = A \wedge \neg A$;

Rezolvare:

Formăm tabloul:

A	$\neg A$	$P = A \wedge \neg A$
a	f	f
f	a	f

4. *Demonstraţi că următoarele propoziţii sunt logic echivalente:*

$\neg(A \wedge B)$ *şi* $\neg A \vee \neg B$;

Rezolvare:

Formăm tabloul:

A	B	$\neg(A \wedge B)$	$\neg A \vee \neg B$
a	a	f	f
a	f	a	a
f	a	a	a
f	f	a	a

5. Dacă $S=\{A\vee B, A\to C\}$ să se demonstreze că $S \models B\vee C$.

Rezolvare:

O propoziţie σ este consecinţă a unei mulţimi de propoziţii $S\in$ LP dacă şi numai dacă pentru orice valorizare de adevăr W a lui S şi pentru orice $\tau\in S$, astfel încât $W(\tau)=a$, $W(\sigma)=a$. Adică: pentru fiecare valorizare de adevăr a propoziţiilor lui S pentru care ele sunt simultan adevărate, σ este adevărată.

A	B	C	$A\vee B\ (\tau_1)$	$A\to C\ (\tau_2)$	$B\vee C\ (\sigma)$
a	a	a	a	a	a
a	a	f	a	f	a
a	f	a	a	a	a
a	f	f	a	f	f
f	a	a	a	a	a
f	a	f	a	a	a
f	f	a	f	a	a
f	f	f	f	a	f

6. *Dacă* $S=\{A\leftrightarrow C, B\leftrightarrow D, (A\vee B)\wedge(C\vee D)\}$ *să se demonstreze că* $S \not\models (A\wedge B)\vee(C\wedge D)$.

Rezolvare:

Formăm tabloul:

A	B	C	D	$A\leftrightarrow C$	$B\leftrightarrow D$	$A\vee B$	$C\vee D$	$(A\vee B)\wedge (C\vee D)$	$A\wedge B$	$C\wedge D$	$(A\wedge B)\vee (C\wedge D)$
a	a	a	a	a	A	a	a	a	a	a	a
a	a	a	f	a	F	a	a	a	a	f	a
a	a	f	a	f	A	a	a	a	a	f	a
a	a	f	f	f	F	a	f	f	a	f	a
a	f	a	a	a	F	a	a	a	f	a	a
a	f	a	f	a	A	a	a	a	f	f	f
a	f	f	a	f	F	a	a	a	f	f	f
a	f	f	f	f	A	a	f	f	f	f	f
f	a	a	a	f	A	a	a	a	f	a	a
f	a	a	f	f	F	a	a	a	f	f	f
f	a	f	a	a	A	a	a	a	f	f	f
f	a	f	f	a	F	a	f	f	f	f	f
f	f	a	a	f	F	f	a	f	f	a	a
f	f	a	f	f	A	f	a	f	f	f	f
f	f	f	a	a	F	f	a	f	f	f	f
f	f	f	f	a	A	f	f	f	f	f	f

7. Sa se determine forma normal disjunctiva pentru o propoziţie ştiind ca tabela sa de adevăr (redusa) este:

A	B	C	S
a	a	a	a
a	a	f	f
a	f	a	f
a	f	f	f
f	a	a	a
f	a	f	f
f	f	a	f
f	f	f	a

***Rezolvare*:**

Avem conform tabelei de adevăr redusă (liniile 1, 5 şi 8):

$$FND(S) = (A \wedge B \wedge C) \vee (\neg A \wedge B \wedge C) \vee (\neg A \vee \neg B \vee \neg C)$$

8. Sa se determine forma normal conjunctiva pentru propoziţia:

$$P = \neg((A \vee B) \wedge (\neg A \vee \neg B)) \wedge C$$

Rezolvare:

$P = \neg((A \vee B) \wedge (\neg A \vee \neg B)) \wedge C =$

$[\neg(A \vee B) \vee \neg(\neg A \vee \neg B)] \wedge C =$

$[(\neg A \wedge \neg B) \vee (A \wedge B)] \wedge C =$

$[(\neg A \wedge \neg B) \vee A] \wedge [(\neg A \wedge \neg B) \vee B] \wedge C =$

$(\neg A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C) = FND(P)$

$(\neg A \vee A) \wedge (\neg B \vee A) \wedge (\neg A \vee B) \wedge (\neg B \vee B) \wedge C =$

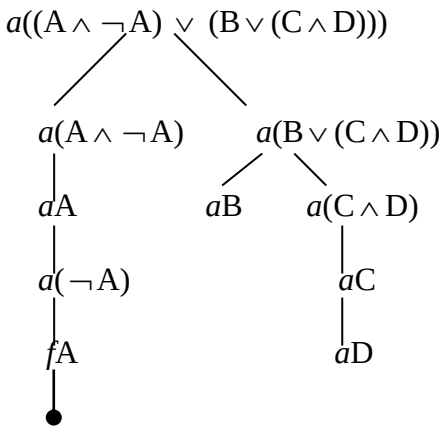
$(\neg B \vee A) \wedge (\neg A \vee B) \wedge C = FNC(P)$

Deci $FNC(P) = \{ \{\neg A, B\}, \{\neg B, A\}, \{C\} \}$

9. Sa se construiască tablourile semantice pentru propoziţia:

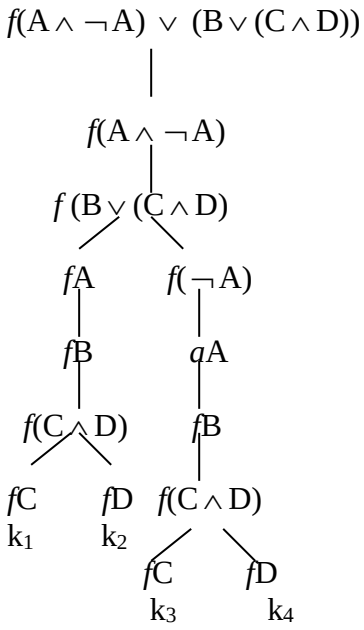
S: $(A \wedge \neg A) \vee (B \vee (C \wedge D))$

Rezolvare:



(ramura contradictorie)

şi



Sunt 4 ramuri: k_1,k_2,k_3,k_4.

10. Să presupunem că următoarele propoziţii sunt adevărate.
(1) *George o iubeşte pe Maria sau George o iubeşte pe Ecaterina*.
(2) *Dacă George o iubeşte pe Maria atunci el o iubeşte pe Ecaterina*.
Pe cine iubeşte George de fapt?

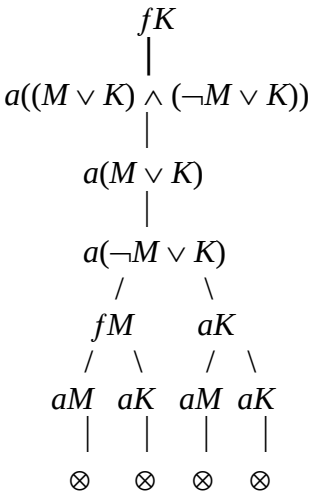
Rezolvare. Să notăm ” *George o iubeşte pe Maria*” cu M şi “*George o iubeşte pe Ecaterina*” cu K. Atunci,

$$(1) \equiv M \vee K \quad \text{şi} \qquad (2) \equiv M \to K \equiv \neg M \vee K.$$

Formăm propoziţia

$$A \equiv (M \vee K) \wedge (\neg M \vee K) \equiv (1) \wedge (2),$$

care conform ipotezei, este adevărată. Dorim să aflăm dacă *George o iubeşte pe Ecaterina* sau, echivalent, dacă aK. Să presupunem că nu o iubeşte, adică fK. Atunci putem construi arborele semantic cu originea fK.



În pasul 2 am adăugat $a((M \vee K) \wedge (\neg M \vee K))$, deoarece (1) şi (2) sunt, conform presupunerii iniţiale, adevărate. Începând cu aK, obţinem un tablou semantic contradictoriu, ceea ce însemană că propoziţia K este întodeauna adevărată, cu alte cuvinte *George o iubeşte pe Ecaterina*.

Dacă construim un tablou semantic cu originea în fM, vom obţine un tablou semantic necontradictoriu şi astfel nu vom putea concluziona dacă *George o iubeşte pe Maria* sau *nu*.

11. Demonstraţi că propoziţia $\neg B$ este demonstrabilă prin rezoluţie din mulţimea:

$S = \{\{A, \neg B\}, \{\neg A, \neg B, \neg C\}, \{\neg A, \neg B, C\}\}$

Rezolvare. Avem

1. $\{A, \neg B\}$
2. $\{\neg A, \neg B, \neg C\}$
3. $\{\neg A, \neg B, C\}$
4. $\{\neg A, \neg B\}$ din 2 şi 3
5. $\{\neg B\}$ din 4 şi 1

Demonstraţia prin rezoluţie pe care o căutăm este succesiunea clauzelor 1,2,3,4 şi 5.

Demonstraţia ar fi putea fi înfăptuită şi după cum urmează:

Aplicăm rezoluţia asupra lui:

$$S_1 = \{\{\underbrace{A, \neg B}_{1}\}, \{\underbrace{\neg A, \neg B, \neg C}_{2}\}, \{\underbrace{\neg A, \neg B, C}_{3}\}, \{\underbrace{B}_{4}\}\} = S \cup \{B\}$$

pentru a da:

5. $\{A\}$ din 1 şi 4
6. $\{\neg A, \neg B\}$ din 2 şi 3
7. $\{\neg A\}$ din 4 şi 6
8. $\square$ din 5 şi 7

Anume avem $S \cup \{B\} \vdash_R \square$. Întrucât regula rezoluţiei este o regulă derivabilă în **LP** după cum am văzut în exemplul 5.3.10, avem de asemenea $S \cup \{B\} \vdash \square$. Dar, în acest caz, avem, conform teoremei deducţiei 5.2.6, că $S \vdash B \rightarrow \square$.

Conform tautologiei $(B \rightarrow \square) \leftrightarrow \neg B$, putem conchide $S \vdash \neg B$; cu alte cuvinte $\neg B$ este demonstrabil din S.

TESTE DE AUTOEVALUARE ŞI TEME DE CONTROL

Testul nr. 1

1. Demonstraţi că următoarea propoziţie este o tautologie:
$P=(A \to B) \vee (A \to \neg B)$;

2. Demonstraţi că următoarea propoziţie este o contradicţie:
$P=(A \to B) \wedge (B \to C) \wedge (A \to \neg A)$;

3. *Demonstraţi că următoarele propoziţii sunt logic echivalente:*
$A \vee (B \wedge C)$ *şi* $(A \vee B) \wedge (A \vee C)$;

Testul nr. 2

1. Demonstraţi că următoarea propoziţie este o tautologie:
$P= A \to \neg\neg A$;

2. Demonstraţi că următoarea propoziţie este o contradicţie:
$P=(\neg A \vee (B \wedge \neg B)) \leftrightarrow A$;

3. *Demonstraţi că următoarele propoziţii sunt logic echivalente:*
$A \to B$ *şi* $\neg B \to \neg A$.

Temă de control

1. Demonstraţi că următoarea propoziţie este o tautologie:
$P= ((A \wedge B) \to C) \leftrightarrow (A \to (B \to C))$.

2. Demonstraţi că următoarea propoziţie este o contradicţie:
$P= \neg(A \wedge B) \wedge (A \to B) \wedge A$.

3. *Demonstraţi că următoarele propoziţii sunt logic echivalente:*
$A \to (B \vee C)$ *şi* $(\neg B \wedge \neg C) \to \neg A$.

4. Sa se determine forma normal disjunctivă pentru propoziţia:
$P = \neg((A \vee B) \wedge (\neg A \vee \neg B)) \wedge C$

5. Aplicând metoda tablourilor semantice, să se verifice dacă următoarele propoziţii sunt tautologii:

a) $\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$

Indicaţie: Se va folosi demonstraţia Beth pentru $f(\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B))$

b) $(A \wedge (A \to B)) \leftrightarrow B$

Indicaţie: Se va folosi demonstraţia Beth pentru $f((A \wedge (A \to B)) \leftrightarrow B)$

BIBLIOGRAFIE RECOMANDATĂ LA UNITATEA DE INVATARE NR.2

7. G. Metakides, A. Nerode – *Principii de logică şi programare logică*, Editura Tehnică, Bucureşti, 1998
8. G. Georgescu – *Elemente de logică matematică*, Editura Academiei Tehnice Militare, Bucureşti, 1978
9. D. Busneag, D. Piciu, *Probleme de logica si teoria multimilor*, Craiova, 2003.
10. G. Georgescu, A. Iorgulescu, *Logica matematica*, Ed. ASE, Bucuresti, 2010
11. Gr. C Moisil, *Elemente de logica matematica si de teoria multimilor*, Ed. Stiintifica, Bucuresti, 1968
12. J.D. Monk, *Mathematical Logic*, Springer Verlag, 1976
13. V. E. Cazanescu, *Curs de bazele informaticii*, Tipografia Universitatii din Bucuresti, 1976
14. S. Rudeanu, *Curs de bazele informaticii*, Tipografia Universitatii din Bucuresti, 1982
15. M. Huth, M. Ryan, *Logic in Computer Science*: Modelling and Reasoning about Systems, Cambridge Univ. Press, 2009
16. A.R. Bradley, Z. Manna, *The Calculus of Computation Decision Procedures with Applications to Verification*, Springer,2007
17. M. Ben-Ari, *Mathematical Logic For Computer Science*, Springer, 2003

UNITATEA DE INVĂŢARE NR.3
LOGICA PREDICATELOR

Lecţia 7-Logica predicatelor

7.1 *Introducere*

În capitolul precedent s-a făcut o descriere analitică a limbajului **LP,** un limbaj formal, pe baza căruia putem exprima atât propoziţii simple cât şi propoziţii compuse. Mai mult, am examinat metodele de derivare a concluziilor din mulţimi de propoziţii **LP**.

Deşi limbajul **LP** este destul de bogat, el permite numai o exprimare limitată a proprietăţilor şi relaţiilor. Să considerăm următorul exmplu de propoziţie în limbaj natural:

S: "Dacă George este om atunci George este muritor"

Dacă *A* reprezintă propoziţia:

"George este om"

iar *B* reprezintă:

"George este muritor"

atunci, în contextul **LP**, *S* devine:

$S: A \rightarrow B$

S exprimă anumite caracteristici ale unei anumite persoane particulare, respectiv George. Cum putem însă exprima proprietăţi similare ale altor persoane, cum ar fi Socrate sau Petre? O soluţie ar fi să introducem tot atâtea simboluri propoziţionale diferite câţi oameni există! Dar acest lucru este imposibil în practică.

În acest capitol vom descrie limbajul "Logicii Predicatelor", care oferă o soluţie acestei probleme. Elementul de noutate al acestui limbaj este introducerea **variabilelor** şi a **cuantificatorilor.**

A) Variabile

Dacă considerăm propoziţia *S* şi dacă presupunem că *x* este o variabilă care ia valori în mulţimea numelor de persoane, de exemplu:

x = George sau x = Ion sau x = …

şi dacă "Om" şi "Muritor" sunt simboluri ce reprezintă proprietăţi, atunci putem reprezenta relaţia generală între aceste proprietăţi prin:

P : Om(x) → Muritor(x)

Astfel de reprezentări, cum ar fi "Om(x)" sau "Muritor(x)", care exprimă relaţii generale sub formă de proprietăţi, se numesc **predicate.** O **formulă**, de exemplu *P* de mai sus, este o reprezentare care conţine predicate legate prin conectori logici.

Substituţia variabilei *x* cu constanta "George" transformă *P* în formula:

S' : Om(George) → Muritor(George)

În plus, dacă variabila *x* ia valoarea "Socrate", rezultatul va fi o nouă formulă ce reprezintă relaţia dinte Socrate şi proprietatea de a fi muritor. "Ion", "George" şi "Petre" sunt **constante** în noul limbaj formal.

În general, corespondenţa între variabile şi constante, pe de o parte, şi simboluri ale limbajului natural, pe de altă parte, poate fi reprezentată intuitiv astfel:

Limbaj natural		Limbaj formal
pronume	⇛	variabilă
nume propriu	⇛	constantă

simbolurile speciale “Om” şi”Muritor” se numesc **simboluri predicative.** Predicatele pot conţine mai multe variabile, exprimând astfel nu numai proprietăţi, dar şi relaţii între mai multe obiecte. De exemplu, dacă variabilele *x* şi *y* iau valori în mulţimea numerelor întregi şi dacă introducem predicatul *I*, “mai_mare”, putem exprima una dintre relaţiile fundamentale între întregi:

$I(X,Y)$: mai_mare(x, y)

care este **interpretată** drept “*x* este mai mare decât *y*”.

Dacă în expresia de mai sus înlocuim *x* cu 5 şi *y* cu 3, avem evident o versiune particulară a lui *I*:

$I(5, 3)$: mai_mare$(5, 3)$

care este adevărată pentru respectivele numere întregi.

B) Cuantificatori

Introducerea variabilelor rezultă în schimbarea validităţii unei formule. Să considerăm, de exemplu, formula:

$Q(x, y)$: zbor_$X(x, y)$

care este **interpretată** ca:

Există un zbor al companiei *X* între oraşele *x* şi *y*.

Validitatea formulei este numai *parţială*, deoarece poate să nu existe un zbor al companiei *X* între New York şi Bucureşti, de exemplu. În schimb, formula:

$P(x)$: Om$(x) \rightarrow$ Muritor(x)

are o validitate universală, deoarece se îndeplineşte pentru orice variabilă *x*.

În **Logica Predicatelor**, pe scurt **LPr**, validitatea generală sau parţială este reprezentată prin două simboluri speciale numite “cuantificatori”; avem astfel un **cuantificator universal** şi un **cuantificator existenţial**, notaţi prin “$\forall$” şi, respectiv “$\exists$”. Astfel, formula iniţială *P* devine:

$P(x)$: $(\forall x)$(Om$(x) \rightarrow$ Muritor(x))

şi *Q* devine:

$Q(x, y)$: $(\exists (x, y))$ zbor_$X(x, y)$

În secţiunile următoare, vom introduce formal limbajul logicii predicatelor.

7.2 *Limbajul logicii predicatelor*

Enunţăm acum descrierea formală a unui limbaj al **LPr.**

Definiţia 7.2.1: Un limbaj **LPr** conţine următoarele simboluri fundamentale:

(I) Simboluri logice:

- **(i)** *Variabile* $x, y, z, \ldots, x_0, y_0, z_0, \ldots, x_i, \ldots$
- **(ii)** *Conectori logici* $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$
- **(iii)** *Virgule, paranteze* , ( )
- **(iv)** *Cuantificatori* $\forall, \exists$

(II) Simboluri specifice:

- **(i)** *Simboluri predicative* $P, Q, R, \ldots, P_0, Q_0, R_0, \ldots, P_1, \ldots$
- **(ii)** *Simboluri pentru constante* $a, b, \ldots, a_0, b_0, \ldots, a_1, \ldots, a_2, \ldots$
- **(iii)** *Simboluri funcţionale* $f, g, f_0, g_0, f_1, \ldots$

Numărul de variabile diferite ce apare într-un predicat reprezintă **gradul** sau **aritatea** predicatului. De exemplu, $Q(x, y, z)$ este un predicat de gradul 3 sau un predicat de arietate 3.

Fiecare cuantificator este **dual** celuilalt: $\forall$ este echivalent secventei de simboluri $\neg\exists\neg$ şi $\exists$ este echivalent cu $\neg\forall\neg$. Pentru formula $(\forall x)Q(x)$ avem, de exemplu,

$$(\forall x)Q(x) \leftrightarrow \neg(\exists\, x)\neg Q(x)$$

Fiecare limbaj al **LPr**, adică fiecare mulţime de simboluri specifice conţine toate simbolurile logice. Astfel, pentru a determina un limbaj, este suficientă definirea simbolurilor lui specifice.

Exemplul 7.2.2: $\mathcal{L}_A = (=, \leq, +, *, 0, 1)$ este un limbaj pentru aritmetică.

a) = şi ≤ sunt simboluri predicative binare (de aritate 2):
$=(x, y)$ se citeşte "$x = y$", iar $\leq(x, y)$ reprezintă "$x \leq y$".

b) + şi * sunt predicate ternare (de aritate 3):
$+(x, y, z)$ se citeşte "$x + y = z$", iar $*(x, y, z)$ se citeşte "$x * y = z$"

c) 0 şi 1 sunt simboluri constante

Definitia 7.2.3: Un **termen** este definit inductiv astfel:

(i) O **constantă** este un termen.

(ii) O **variabilă** este un termen.

(iii) Dacă f este o **funcţie** n-ară şi $t_1, \ldots, t_n$ sunt termeni, atunci $f(t_1, \ldots, t_n)$ este un termen.

Definiţia 7.2.4: O **formulă atomică** sau **atom** este orice secvenţă de simboluri $P(t_1,\ldots, t_n)$, unde P este un simbol predicativ n-ar şi t_i este un termen, pentru orice i=1,2, …, n.

Defniţia 7.2.5: O **formulă** este definită inductiv astfel:

(i) Orice **atom** este o formulă.

(ii) Dacă σ_1, σ_2 sunt formule, atunci $(\sigma_1 \wedge \sigma_2)$, $(\sigma_1 \vee \sigma_2)$, $(\sigma_1 \rightarrow \sigma_2)$, $(\neg\sigma_1)$ şi $(\sigma_1 \leftrightarrow \sigma_2)$ sunt formule.

(iii) Dacă v este o variabilă şi φ este o formulă, atunci $((\exists v)\varphi)$, $((\forall v)\varphi)$ sunt de asemenea formule.

(iv) Numai secvenţele de simboluri formate conform regulilor (i), (ii), (iii) sunt formule.

Exemplul 7.2.6: Următoarele expresii sunt formule:

(i) $\varphi_1 : (\forall y)\,(\exists x)\,[P(x, f(y)) \vee Q(x)]$

(ii) $\varphi_2 : (\forall y)\,(\exists x)\,[P(x) \vee Q(x, y) \rightarrow \neg(R(x))]$

Observaţia 7.2.7: Observăm că definiţia formulelor limbajului permite utilizarea trivială a cuantificatorilor, de exemplu:

$(\exists y)\,[y = 3]$

care este echivalentă cu formula:

$y = 3$

Aceste utilizări triviale sunt formal acceptate; cu toate acestea ele sunt utilizate numai în demonstraţii tehnice.

Exemplul 7.2.8: Prezentăm diverse formule ale limbajului $\mathcal{L}_A$ definit în exemplul 4.2.2.

(1) $(\forall x)\,(x = x)$ = este reflexivă
(2) $(\forall x)\,(\forall y)\,(x = y \rightarrow y = x)$ = este simetrică
(3) $(\forall x)\,(\forall y)\,(\forall v)\,[(x = y \wedge y = v) \rightarrow x = v]$ = este tranzitivă
(4) $(\forall x)\,(x \leq x)$ ≤ este reflexivă

Vom continua prin enunţarea anumitor definiţii care sunt necesare pentru descrierea completă a contextului **LPr** şi a programării logice.

Definiţia 7.2.9:

(i) O subsecvenţa t_1 de simboluri a unui termen t, astfel încât t_1 este un termen, se numeşte un **subtermen** al lui t.

(ii) O subsecevnţă φ_1 de simboluri a unei formule φ, astfel încât φ_1 este o formulă, se numeşte o **subformulă** a lui φ.

Exemplul 7.2.10:

(i) dacă $f(x, y)$ este un termen, atunci x, y şi $f(x, y)$ sunt subtermeni ai lui $f(x,y)$.

(ii) $P(x)$, $\neg R(x)$, $R(x)$, $P(x) \vee Q(x, y)$ sunt subformule ale formulei φ_2 din exemplul 4.2.6.

Observaţia 7.2.11: În capitolul destinat **LP**, am examinat propoziţiile numai în funcţie de construcţia lor bazată pe propoziţiile simple, atomice şi conectori logici. Am presupus astfel că propoziţiile atomice nu necesită o analiză mai amănunţită. **Logica predicatelor,** în schimb, utilizează un concept mai general, "formulele atomice". În acest caz se presupune că formulele atomice sunt reprezenate de predicate şi că fiecare predicat n-ar $P(t_1, \ldots, t_n)$ exprimă o relaţie între termenii $t_1, \ldots, t_n$.

Definiţia 7.2.12: Apariţii libere şi legate ale variabilelor:

(i) O apariţie a unei variabile v într-o formulă φ se spune că este legată dacă există o subformulă ψ a lui φ care conţine variabila v şi începe cu $(\forall v)$ sau cu $(\exists v)$.

(ii) O apariţie a unei variabile v într-o formulă se spune că este liberă dacă nu este legată.

Definiţia 7.2.13: Variabile libere şi legate:

O variabilă v ce apare într-o formulă φ se spune că este **liberă** dacă are cel puţin o apariţie liberă în φ. Se spune că v este legată dacă nu este liberă.

Exemplul 7.2.14: În exemplul 7.2.6, variabila x are o apariţie liberă în subformula:

φ': $(\forall y)\,[P(x) \vee Q(x, y) \rightarrow \neg(R(x))]$

din φ_2, dar este legată în formula φ_2 însăşi. Deci x este liberă în φ' (deoarece are cel puţin o apariţie liberă) dare este legată în φ_2.

Definiţia 7.2.15: Un termen ce nu conţine variabile se numeşte **termen de bază.**

Exemplul 7.2.16: Dacă a, b sunt constante şi f este un simbol funcţional de aritate 2, atunci a, b, $f(a, b)$, $f(f(a, b), b)$, … sunt termeni de bază.

Definiţia 7.2.17: O **frază** sau **formulă închisă** este o formulă fără variabile libere.

Conform definiţiei anterioare, pentru a forma o formulă închisă dintr-o formulă dată, trebuie să legăm toate variabilele libere din acea formulă prin cuantificatori.

Exemplul 7.2.18: Din formula $\varphi(x,y)$: $(x+y=x*y)$ putem forma formula închisă:
$\sigma(x, y) : (\forall x)\ (\exists y)\ (x+y = x*y)$

O posibilitate de a forma propoziţii este aceea de a substitui apariţiile libere ale variabilelor prin constante. În general, avem următoarea definiţie pentru substituţie:

Definiţia 7.2.19: O **mulţime substituţie**, sau mai simplu **substituţie**, este o mulţime:
$\theta = \{x_1/t_1\ ,\ x_2/t_2\ ,\ \ldots\ ,\ x_n/t_n\}$
unde x_i şi t_i, $1 \le i \le n$, sunt variabile şi termeni corespondenţi pentru care dacă $x_i = x_j$ atunci $t_i = t_j$, $1 \le j \le n$.

Dacă φ este o expreie (atom, termen sau formulă), atunci $\varphi\theta$ reprezintă expresia ce rezultă prin substituţia apariţiilor variabilelor $x_1, x_2, \ldots, x_n$ cu termenii corespondenţi $t_1, t_2, \ldots, t_n$.

Substituţia vidă se notează cu E, cu alte cuvinte $E = \{\ \}$.

Exemplul 7.2.20: Dacă aplicăm substituţia $\theta = \{x/2\ ,\ y/2\}$ formulei:
$K : \varphi(x, y)$
în exemplul anterior, se obţine formula:
$K :$ (2+2 = 2*2)

Exemplul 7.2.21: Să considerăm formula:
$\Phi(x, y, z) : (\exists y)\ R(x, y)\ \wedge\ (\forall z)\ (\neg Q(x, y))$
şi termenul de bază $f(a,b)$. Dacă aplicăm substituţia $\{x\ /f(a,\ b)\}$ formulei $\varphi(x, y, z)$ se obţine formula:
$\varphi(f(a, b), y, z) : (\exists y)\ R(f(a, b), y)\ \wedge\ (\forall z)\ (\neg Q(f(a, b), z))$
Operaţia de bază asupra substituţiilor este compunerea.

Definiţa 7.2.22: Fie:
$\theta = \{u_1/s_1,\ \ldots,\ u_m/s_m\}$ şi $\psi = \{v_1/t_1,\ \ldots,\ v_n/t_n\}$
Compunerea substituţiilor θ şi ψ este substituţia:
$\theta\psi = \{u_1/s_1\psi,\ \ldots,\ u_m/s_m\psi,\ v_1/t_1,\ \ldots,\ v_n/t_n\}$
$-\ (\{u_i/s_i\psi \mid u_i = s_i\psi\}\ \cup\ \{v_i/t_i \mid v_i \in \{\ u_1,\ \ldots,\ u_m\}\})$
Cu alte cuvinte, pentru a obţine compunerea substituţiilor aplicăm mai întâi ψ asupra termenilor $s_1, \ldots, s_m$ din θ înlocuind variabilele u_i din θ cu temenii $s_i\psi$ corespunzători conform definiţiei 7.2.19, şi completăm cu elementele lui ψ. Omitem termenii u_i din θ care sunt de forma $s_i\psi$ şi termenii v_i din ψ care conţin variabile din θ.

Exemplul 7.2.23:
(1). Fie $\theta = \{x / f(y),\ y / z\}$ (adică $\{u_i / s_i\}$) şi $\psi = \{x / a,\ y / b,\ z / y\}$ (adică $\{v_i/t_i\}$) două substituţii.Compunerea lui θ şi ψ este:
$\theta\psi = \{x / f(y)\psi,\ y / z\psi,\ x / a,\ y / b,\ z / y\} - (\{u_i / s_i\psi \mid u_i = s_i\psi\}\ \cup\ \{v_i/t_i \mid v_i \in \{\ u_i\}\}) = \{x / f(\mathrm{b}),\ y / y,\ x / a,\ y / b,\ z / y\} - (\{y /y\}\ \cup \{\ x / a,\ y / b\}) = \{x / f(\mathrm{b}),\ z / y\}$
(2). Să considerăm termenul t: $w(f(v_1),\ h(x),\ f(v_2),\ v_3)$ şi substituţiile:
$\theta = \{v_1 / f(g(x)),\ v_2 / h(v_1),\ v_3 / h(v_3)\}$ şi
$\psi = \{x / z,\ v_1 / v_2,\ v_3 / v_1\}$
Atunci:
$\theta\psi = \{v_1 / f(g(x))\psi,\ v_2 / h(v_1)\psi,\ v_3 / h(v_3)\psi,\ x / z,\ v_1 / v_2,\ v_3 / v_1\}$

$-\{\varnothing \cup \{ v_1 / v_2 , v_3 / v_1\}\} = \{v_1 / f(g(z)), v_2 / h(v_2), v_3 / h(v_1), x / z, v_1 / v_2, v_3 / v_1\} - \{v_1 / v_2, v_3 / v_1\} = \{v_1 / f(g(z)), v_2 / h(v_2), v_3 / h(v_1), x / z\}$

În consecinţă:

$t(\theta\psi) : w(f(f(gz))), h(z), f(h(v_2)), h(v_1))$

Mai mult, se observă că:

$t(\theta\psi) = w(f(f(g(x))), h(x), f(h(v_1)), h(v_3))\psi$

$= w(f(f(g(z))), h(z), f(h(v_2)), h(v_3))$

$= (t\ \theta)\ \psi$

Putem deci concluziona că proprietatea de asociativitate a compunerii substituţiilor se îndeplineşte pentru substituţia din exemplul 7.2.23. În general avem:

Teorema 7.2.24: *Pentru orice substituţii* θ, ψ *şi* γ *şi pentru orice formulă închisă* σ *avem:*

(i) $\theta E = E\theta = \theta$

(ii) $(\sigma\theta)\ \psi = \sigma(\theta\psi)$

(iii) $(\theta\psi)\gamma = \theta(\psi\gamma)$

Definiţia 7.2.25: Fie φ o formulă fără cuantificatori şi fie θ o substituţie. Fie $\varphi\theta$ formula ce rezultă prin substituirea fiecărui termen t ce apare în φ cu $t\theta$.

Corespunzător, dacă $S = \{C_1, ..., C_k\}$ este o mulţime de formule **LPr** fără cuantificatori, atunci $S\theta = \{C_1\theta, ..., C_k\theta\}$ rezultă din substituţia *elementelor* C_i, $1 \leq i \leq k$, cu formulele $C_i\theta$.

Definiţia 7.2.26: Fie S_1 şi S_2 două mulţimi de formule fără cuantificatori. S_1 şi S_2 se numesc **varianţi** dacă există două substituţii θ şi ψ astfel încât:

$S_1 = S_2\theta$ şi $S_2 = S_1\psi$

Exemplul 7.2.27: $S_1 = P(f(x, y), h(z), b)$ şi $S_2 = P(f(y, x), h(u), b)$, unde b este o constantă, sunt varianţi. Într-adevăr, dacă:

$\theta = \{x / z, y / x, z / u\}$ şi $\psi = \{x / y, y / x, u / z\}$ atunci:

$S_1\theta = P(f(x, y), h(z), b)\theta = P(f(y, x), h(u), b) = S_2$

$S_2\psi = P(f(x, y), h(z), b)\psi = P(f(x, y), h(z), b) = S_1$

Definiţia 7.2.28: O **substituţie de redenumire** este o substituţie de forma:

$\{v_1 / u_1, ..., v_n / u_n\}$ unde v_i şi u_i, $1 \leq i \leq n$, sunt variabile.

7.3 *Bazele axiomatice ale logicii predicatelor*

În subparagraful 5.2 am axiomatizat logica propoziţională cu ajutorul unui sistem axiomatic format din trei axiome şi o regulă. Logica predicatelor poate fi similar axiomatizată. Să începem prin a da o definiţie auxiliară.

Definiţia 7.3.1: O variabilă x este liberă pentru termenul t în formula σ, formal liber(x, t, σ), dacă nici una din variabilele libere din t nu devine legată după substituţia lui x cu t pentru toate apariţiile libere ale lui x în σ.

Exemplul 7.3.2: Fie $\sigma : (\forall y)\ P(x, y)$. Atunci x nu este liberă pentru termenul y în σ deoarece, după aplicarea substituţiei x / y pentru apariţia liberă a lui x, variabila y a termenului y este legată.

Invers, *x* este liberă pentru termenul *z*, unde *z* este o variabilă diferită de *y* deoarece, după substituţia *x / z* în σ, variabilele termenului *z*, şi anume *z*, nu sunt legate. În plus, *y* este liberă pentru *y* în σ (σ nu conţine apariţii libere ale lui *y*).

Definiţia 7.3.3: Pentru formulele φ, τ, σ din **LPr**, axiomele **LPr** sunt:

(1) $\varphi \to (\tau \to \varphi)$

(2) $(\varphi \to (\tau \to \sigma)) \to ((\varphi \to \tau) \to (\varphi \to \sigma))$

(3) $(\neg\varphi \to \neg\tau) \to (\tau \to \varphi)$

(4) Dacă liber(*x, t,* φ), atunci formula:

$(\forall x)\varphi \to \varphi(x / t)$

este o axiomă.

(5) Dacă *x* nu este liberă în formula φ, atunci formula:

$(\forall x)(\varphi \to \tau) \to (\varphi \to (\forall x)\tau))$

este o axiomă.

La fel ca în **LP**, simbolul ⊢ reprezintă derivarea formulelor în sistemul axiomatic al **LPr**. Acest sistem axiomatic conţine două reguli:

(1) Modus Ponens: $\varphi, \varphi \to \tau \vdash \tau$

(2) Generalizare: $\varphi \vdash (\forall x)\varphi$

Observaţia 7.3.4:

(1) Conform regulii generalizării, dacă φ este o formulă derivată din axiomele şi regulile **LPr**, atunci $(\forall x)\varphi$ este de asemenea derivabilă în sistemul axiomatic. Să presupunem, de exemplu, că s-a derivat formula:

"Om(*x*) → Muritor(*x*)"

Atunci formula:

"$(\forall x)$ (Om(*x*) → Muritor(*x*))"

este de asemenea derivată.

Cu alte cuvinte, putem întotdeauna să obţinem validitatea unei formule generalizate $(\forall x)\varphi$ pe baza validităţii formulei φ. Anumite erori în discuţiile comune sunt generate de o aplicare incorectă a regulii generalizării. De exemplu, spunem de multe ori că:

"toţi politicienii sunt escroci"

deoarece ştim că politicienii *a* şi *b* sunt escroci.

Cu toate acestea, acest enunţ nu este logic valid: pentru a putea generaliza, adică pentru a caracteriza toţi politicienii şi nu numai *a* şi *b*, trebuie să ne asigurăm că următoarea formulă este derivabilă în sistemul nostru axiomatic:

"politician(*x*) → escroc(*x*)"

Ceea ce (sperăm!) nu este cazul.

(2) Comparând sistemele axiomatice ale **LP** şi **LPr**, observăm că axiomele şi regulile **LP** sunt incluse în axiomele şi regulile **LPr**. Cu toate acestea, logica propoziţiilor tratează propoziţii, în timp ce logica cu predicate se referă la un concept compus, respectiv formulele **LPr.**

(3) Cuantificatorul ∃ nu este inclus în sistemul axiomatic deoarece, aşa cum s-a văzut în subcapitolul 4.1, ∃ se defineşte ca $\neg\forall\neg$.

(4) Din (2) şi corolarul 6.3.2 concluzionăm că toate tautologiile **LP** sunt derivate în **LPr** prin considerarea formulelor **LPr** în locul propoziţiilor **LP.** De exemplu, pentru propoziţia $A \leftrightarrow \neg\neg A$ care este derivabilă în **LP** avem formula **LPr** $\varphi\leftrightarrow\neg\neg\varphi$ care este derivabilă în sistemul axiomatic specificat de definiţia 7.3.3. Putem astfel aplica toate tautologiile din **LP** formulelor din **LPr** (teorema de completitudine 6.3.1) şi obţine formule ale **LPr** care sunt derivabile în sistemul axiomatic al **LPr.**

Teorema substituţiei echivalenţelor este validă atât în **LP**, cât şi în **LPr**. Demonstraţia este similară celei date teoremei corespunzătoare din **LP**.

Teoremă 7.3.5: Teorema substituţiei echivalenţelor pentru **LPr**:

Dacă o formulă A_1 este derivată dintr-o formulă A, după substituţia formulei B cu formula B_1 pentru zero, una sau mai multe apariţii ale lui B în A, dacă $\{x_1,..., x_n\}$ *sunt variabile libere în B şi B_1 şi sunt în acelaşi timp variabile legate în A, şi dacă:*

$\vdash (\forall x_1)...(\forall x_n)(B \leftrightarrow B_1)$

atunci $\vdash A \leftrightarrow A_1$.

Formulele **LPr** care sunt derivate în sistemul axiomatic al **LPr** sunt singurele formule „legale" cu care putem lucra în contextul **LPr.** Următoarea teoremă prezintă o listă de formule des utilizate. Aceste formule exprimă asociativitatea şi distributivitatea cuantificatorilor faţă de conectorii logici. Aşa cum se arată în această teoremă, unele proprietăţi ale cuantificatorilor sunt valide numai în anumite condiţii.

Teoremă 7.3.6:*Dacă φ şi σ sunt formule ale* **LPr,** atunci următoarele formule sunt derivabile în **LPr:**

$(\forall x)(\varphi \to \sigma) \to ((\forall x)\varphi \to (\forall x)\sigma)$
$((\forall x)\varphi \to (\forall x)\sigma) \to (\exists x)(\varphi \to \sigma)$
$((\exists x)\varphi \to (\exists x)\sigma) \to (\exists x)(\varphi \to \sigma)$
$(\exists x)(\varphi \leftrightarrow \sigma) \to ((\forall x)\varphi \to (\exists x)\varphi)$
$((\forall x)\varphi \vee (\forall x)\varphi) \to (\forall x)(\varphi \vee \sigma)$
$(\forall x)(\varphi \vee \sigma) \to ((\exists x)\varphi \vee (\forall x)\varphi)$
$(\exists x)(\varphi \vee \sigma) \leftrightarrow ((\exists x)\varphi \vee (\exists x)\sigma)$
$(\exists x)(\varphi \wedge \sigma) \to ((\exists x)\varphi \wedge (\exists x)\sigma)$
$(\forall x)(\varphi \wedge \sigma) \to ((\forall x)\varphi \wedge (\exists x)\sigma)$
$(\forall x)(\varphi \wedge \sigma) \leftrightarrow ((\forall x)\varphi \wedge (\exists x)\sigma)$
$(\exists y)(\forall x)\varphi \to (\forall x)(\exists y)\varphi$
$(\forall x)(\forall y) \leftrightarrow (\forall y)(\forall x)\varphi$
$(\exists x)(\exists y) \leftrightarrow (\exists y)(\exists x)\varphi$
$(\forall x)\varphi \leftrightarrow \varphi$ dacă nu există nici o apariţie liberă a lui x în φ
$(\exists x)\varphi \leftrightarrow \varphi$ dacă nu există nici o apariţie liberă a lui x în φ

Anumite erori care apar frecvent în demonstraţiile formale din diverse ramuri ale ştiinţei se datorează unei utilizări incorecte a distributivităţii cuantificatorilor faţă de conectorii logici. De exemplu, formulele:

$$(\forall x)\varphi \vee (\forall x)\sigma \to (\forall x)(\varphi \vee \sigma)$$

şi

$$(\exists x)(\varphi \vee \sigma) \to ((\exists x)\varphi \wedge (\exists x)\sigma)$$

luată din lista anterioară sunt derivabile în **LPr.** Dar formulele:

$$((\forall x)\varphi \vee (\forall x)\varphi) \to (\forall x)(\varphi \vee \sigma)$$

şi

$$(\exists x)(\varphi \vee \sigma) \leftrightarrow ((\exists x)\varphi \vee (\exists x)\sigma)$$

care exprimă distributivitatea totală a cuantificatorilor $\forall$ şi $\exists$ faţă de $\vee$ şi respectiv $\wedge$, nu sunt valide.

Formulele :

$(\forall x)(\varphi \vee \sigma) \to ((\forall x)\varphi \vee (\forall x)\sigma)$

şi

$((\exists x)\varphi \wedge (\exists x)\sigma) \leftrightarrow ((\exists x) \wedge (\exists x)\sigma)$

care exprimă distributivitatea totală a cuantificatorulor $\forall$ şi $\exists$ faţă de $\vee$ şi respectiv $\wedge$, nu sunt valide.

Formulele:

$(\forall x)(\varphi \vee \sigma) \rightarrow ((\exists x)\varphi \vee (\forall x)\sigma)$

şi

$((\exists x)\varphi \wedge (\exists x)\sigma) \rightarrow (\exists x)(\varphi \wedge \sigma)$

nu sunt derivabile în sistemul axiomatic al ***LPr*** *şi nu sunt valide.*

De exemplu, formula:

$(\forall x)[(x = 2x) \vee (x \neq 2x)]$

este adevărată (vezi exemplul 4.5.4.).

Cu toate acestea, formula:

$[(\forall x)(x = 2x) \vee (x \neq 2x)]$

NU este o formulă adevărată. Dacă ar fi adevărată, atunci cel puţin una din formulele:

$(\forall x)(x = 2x)$, $(\forall x)(x \neq 2x)$

(definiţia 7.5.5.) ar trebui să fie adevărată. Acest lucru nu se întâmplă deoarece dacă $x = 1$, $x = 2x$ nu este adevărată şi dacă $x = 0$, $x \neq 2x$ nu este adevărată. În concluzie, trebuie să fim foarte atenţi la utilizarea comutativităţii şi distributivităţii cuantificatorilor deoarece pot apărea frecvent greşeli de raţionament.

Observaţia 7.3.7: Dacă am extinde limbajul **LPr** cu un simbol logic particular de paritate 2, simbolul „=”, ar exista proprietăţi ale lui „=” care nu ar putea fi exprimate pe baza sistemului axiomatic din definiţia 7.3.4; această axiomatizare exprimă proprietăţi generale ale predicatelor şi nu poate descrie proprietăţile particulare ale unor predicate, cum ar fi „=”. De exemplu, egalitatea trebuie să fie reflexivă, simetrică şi tranzitivă. Pentru a exprima aceste proprietăţi axiomatic, trebuie să extindem sistemul axiomatic din definiţia 7.3.3 cu două axiome:

(6) Pentru toţi termenii *x*, formula:

$x = x$

este o axiomă.

(7) Dacă A_1 este o formulă derivată din formula *A* prin substituirea a zero, o parte sau a tuturor apariţiilor unui termen *x* printr-un termen *y*, atunci formula:

$(x = y) \rightarrow (A \leftrightarrow A_1)$

este o axiomă.

Am axiomatizat astfel reflexivitatea relaţiei de egalitate şi o regulă de substituţie a termenilor egali. Proprietăţile de simetrie şi tranzitivitate ale egalităţii sunt derivabile pe baza axiomelor (1) până la (7) şi a regulilor generalizării şi Modus Ponens.

Definiţia 7.3.8:

(i) Dacă S este o mulţime de formule, posibil vidă, şi A este o formulă a **LPr, o demonstraţie a formulei** A **din** S, notată prin $S \vdash A$, este o secvenţă finită de formule $B_1,...,B_n$ ale **LPr,** unde fiecare $B_i, 1 \leq i \leq k$, este fie o axiomă, fie aparţine lui S, fie urmează din două formule anterioare în secvenţă B_j, B_l, $1 \leq j$, $l \leq i$, folosind regula Modus ponens sau regula generalizării.

(ii) A **este demonstrabilă din** S dacă există o demonstraţie a lui A din S.

(iii) A **este demonstrabilă** dacă există o demonstraţie a lui A din mulţimea vidă.

În **LPr**, teorema deducţiei este de mare interes. Aplicarea teoremei corespunzătoare din **LP**, anume teorema 5.2.7, formulelor **LPr** poate duce la rezultate neaşteptate:

Să presupunem că ştim că există oameni bogaţi:

bogat(x)

Atunci, pe baza regulii generalizării, avem:

bogat(x) $\vdash$ ($\forall$x) bogat(x)

şi pe baza teoremei deducţiei **LPr** „echivalente" celei din **LP** putem concluziona:

bogat(x) $\rightarrow$ ($\forall$x) bogat(x)

Cu alte cuvinte, existenţa oamenilor bogaţi implică faptul că toţi oamenii sunt bogaţi, ceea ce nu corespunde, evident, realităţii! De aceea, pentru a obţine concluzii corecte, teorema deducţiei trebuie să includă limitări în utilizarea regulii generalizării.

Teorema 7.3.9: Teorema deducţiei:

Dacă S este o mulţime de formule ale **LPr**, *A, B formule ale* **LPr** *astfel încât* $S \cup \{A\} \vdash B$, *şi dacă regula generalizării nu a fost utilizată în derivarea lui B din* $S \cup \{A\}$ *pentru o variabilă liberă ce apare în A, atunci* $S \vdash A \rightarrow B$.

Demonstraţia teoremei deducţiei a **LPr** este similară cu demonstraţia teoremei **LP** corespunzătoare.

Pe baza regulii Modus Ponens, inversa este evident adevărată. Atunci:

$S \vdash A \rightarrow B \Leftrightarrow S \cup \{A\} \vdash B$.

7.4 *Notaţii în programarea logică*

În subparagraful 5.3 am definit mai multe concepte de bază ale programării logice în **LP.** Vom extinde aceste definiţii la **LPr** .

Definiţia 7.4.1:

(i) Un **literal** este un atom (definiţia 7.2.4.) sau un atom negat.

(ii) O secvenţă de simboluri de forma:

($\forall x_1$)($\forall$ x_2)...($\forall x_k$)($C_1 \vee C_2 \vee ... \vee C_n$)

unde C_i, i = 1,..., n sunt literali şi x_1,..., x_k sunt toate variabile ce apar în C_i, $1 \leq i \leq n$, se numeşte **clauză**. Dacă n = 0, avem clauză vidă, care se notează prin □.

Parantezele cuantificatorilor vor fi omise ori de câte ori poziţia variabilelor şi a cuantificatorilor este explicită.

O clauză poate fi scrisă echivalent şi în una din următoarele forme:

(a) $\forall x_1 ... \forall x_k (A_1 \vee ... \vee A_m \vee \neg B_1 \vee ... \vee \neg B_l)$

(b) $\forall x_1 ... \forall x_k (A_1 \vee ... \vee A_m \leftarrow B_1 \wedge ... \wedge B_l)$

(c) $\forall x_1 ... \forall x_k (A_1, ... , A_m \leftarrow B_1, ... , B_l)$

(d) $\{C_1, C_2, ..., C_n\}$, *formă mulţime-teoretică,* unde pentru orice $1 \leq i \leq n$, C_i este A_j, $1 \leq j \leq m$, sau C_i este $\neg B_j$, $1 \leq j \leq l$

(e) $A_1, \ldots, A_m \leftarrow B_1, \ldots, B_l$.

Fiecare clauză este astfel o frază, definiţia 7.2.27. Inversa nu este valabilă datorită existenţei posibile a cuantificatorului $\exists$. Cu toate acestea, programarea logică include toate frazele **LPr**. Vom vedea cum se poate rezolva această problemă în subparagraful 8.1 unde vom discuta formele Skolem.

Exemplul 7.4.2: Următoarele secvenţe de simboluri sunt clauze:

(i) $\forall x \forall y \forall z(P(x) \vee \neg Q(x, y) \vee R(x, y, z))$

(ii) $\forall x \forall y(\neg P(f(x, y), a) \vee Q(x, y))$

Definiţia 7.4.3: O frază ce rezultă prin eliminarea cuantificatorilor dintr-o clauză φ şi substituţia tuturor variabilelor cu constante este o **instanţă de bază** a lui φ.

De exemplu, fraza:

$P(a) \vee Q(b) \vee \neg R(a, b)$

este o instanţă de bază a clauzei:

$\forall x \forall y(P(x) \vee Q(y) \vee \neg R(x, y))$

Definiţia 7.4.4: O **clauză Horn** este o clauză de forma:

$\forall x_1 \ldots \forall x_k (A \leftarrow B_1, \ldots, B_l)$

unde $A, B_1, \ldots, B_l$ sunt atomi şi $l \geq 1$. Atomii B_i, $i = 1, \ldots, l$, sunt premisele clauzei Horn şi A este concluzia.

Definiţia 7.4.5: Un **scop** este o clauză Horn fără concluzie, adică o clauză de forma:

$\forall x_1 \ldots \forall x_k (\leftarrow B_1, \ldots, B_l)$

Atomii B_i sunt **subscopurile** scopului.

Interpretarea intuitivă a unui scop devine clară dacă îl rescriem în forma **LPr** formală şi dacă utilizăm dualitatea cuantificatorilor $\forall$ şi $\exists$ şi legile lui De Morgan (vezi observaţia 73.4(3)):

$\forall x_1 \ldots \forall x_k (\neg B_1 \vee \ldots \vee \neg B_l) \leftrightarrow \forall x_1 \ldots \forall x_k \neg(B_1 \wedge \ldots \wedge B_l)$

$\leftrightarrow \neg (\exists x_1 \ldots \exists x_k)(B_1 \wedge \ldots \wedge B_l)$

Cu alte cuvinte, nu există $x_1, \ldots, x_k$ astfel încât toate presupunerile $B_1, \ldots, B_l$ să fie adevărate.

Definiţia 7.4.6: Un **fapt** este o clauză Horn fără presupuneri, adică este o clauză de forma:

$\forall x_1 \ldots \forall x_k (A \leftarrow)$

Observaţia 7.4.7: În secţiunile următoare, vom discuta în special frazele **LPr**, adică formulele fără variabile libere (definiţiile 7.2.12 şi 7.2.13).

Există două motive care ne determină să ne ocupăm de fraze:

(1) Clauzele (definiţiile 7.4.1, 7.4.4, 7.4.5 şi 7.4.6) care sunt folosite în programarea logică sunt fraze ale **LPr** în care toate variabilele sunt legate prin cuantificatori universali;

(2) Pentru fiecare formulă φ a **LPr** care conţine numai variabilele libere $\{x_1, \ldots, x_k\}$ avem:

$\vdash \phi \Leftrightarrow \vdash (\forall x_1) \ldots (\forall x_k)\varphi$

(unde implicaţia ($\Rightarrow$) rezultă prin regula generalizării, iar implicaţia ($\Leftarrow$) din axioma (4) şi regula Modus Ponens).

Considerăm astfel fraza $(\forall x_1)\ldots(\forall x_k)\varphi$ în locul formulei φ.

Definiţia 7.4.8: Un **program logic** este o mulţime finită de clauze Horn.

Să ilustrăm conceptele precedente printr-un exemplu.

Exemplul 7.4.9: Se dau următoarele propoziţii în limbaj natural:

S_1: *Petre este hoţ*
S_2: *Mariei îi place mâncarea*
S_3: *Mariei îi place vinul*
S_4: *Lui Petre îi plac banii*
S_5: *Petre îl place pe* ***x*** *dacă lui* ***x*** *îi place vinul*
S_6: ***x*** *poate fura* ***y*** *dacă* ***x*** *este hoţ şi dacă lui* ***x*** *îi place* ***y***

Prin introducerea constantelor "*Petre*", "*Maria*", "*vin*", "*mâncare*" şi "*banii*", a variabilelor *x* şi *y*, a predicatelor "*hoţ*" (aritate 1), "*place*" (aritate 2) şi "*poate_fura*" (aritate 2) formăm următorul program:

C_1: *hoţ*(*Petre*) ←
C_2: *place*(*Maria, mâncare*) ←
C_3: *place*(*Maria, vin*) ←
C_4: *place*(*Petre, bani*) ←
C_5: *place*(*Petre, bani*) ← *place*(*x, vin*)
C_6: *poate_fura*(*x, y*) ← *hoţ*(*x*), *place*(*x, y*)

Clauzele Horn C_1, C_2, C_3 şi C_4 sunt fapte. Clauzele Horn C_5 şi C_6 constituie partea procedurală a programului.

Să presupunem că dorim să aflăm dacă *Petre poate fura* (dacă există ceva ce *Petre poate fura*); avem de formulat următotul scop:

G: ← *poate_fura*(*Petre, y*)

Vom afla răspunsul în exemplul 9.2.4.

Vom discuta mai târziu, sistematic, modurile în care se pot deriva concluzii din mulţimi de clauze similare celor din exemplul anterior. În secţiunile următoare se va dezvolta teoria interpretărilor şi vom descrie metode de atribuire a valorilor de adevăr frazelor unui limbaj **LPr**, adică formulelor fără variabile libere, pe baza interpretărilor.

7.5 *Interpretări în logica predicatelor*

În contextul **LP**, determinarea valorii de adevăr a unei propoziţii compuse s-a bazat pe conceptul de valorizare. Prin atribuirea de valori de adevăr formulelor atomice dintr-o propoziţie dată A, am determinat inductiv valoarea de adevăr a lui A.

În cele ce urmează, vom generaliza conceptul de interpretare, pe care vom baza dezvoltarea metodelor de determinare a valorii de adevăr a unei fraze **LPr**.

A) Interpretări: o descriere informală

Dorim să interpretăm, adică să asociem valori de adevăr frazelor din **LPr**. Elementele de bază ale acestor fraze sunt termenii, respectiv variabilele şi constantele. De aceea, avem nevoie mai întâi să interpretăm termenii prin formarea unei mulţimi de obiecte ce reprezintă interpretarea termenilor unei fraze. Putem apoi să definim valorile de adevăr ale predicatelor şi frazelor din limbaj.

Se observă astfel că semantica **LPr** este cert mai complexă decât semantica **LP**. Pentru a înţelege temeinic conceptele pe care le vom introduce acum, vom prezenta mai întâi câteva exemple.

Exemplul 7.5.1: Fie limbajul:

$\mathcal{L} = \{Q, \alpha\}$

unde Q este un predicat şi α este o constantă, şi o frază în acest limbaj:

S: $(\exists x)(\forall y)Q(x, y)$

Orice interpretare a limbajului de mai sus trebuie să determine o mulţime de obiecte, ale cărei elemente trebuie să corespundă simbolurilor din $\mathcal{L}$ şi să atribuie valori de adevăr frazelor din limbaj. Vom considera **N,** mulţimea numerelor naturale, ca această mulţime de obiecte. Putem acum defini interpretarea ca o structură:

$\mathcal{A} = (\mathbf{N}, \leq, 1)$

unde:

N:	este mulţimea numerelor naturale
$\leq$:	este relaţia "mai mic decât sau egal cu" în **N**
1:	este numărul natural 1

Pe baza interpretării $\mathcal{A}$, atribuim următoarele asocieri simbolurilor din $\mathcal{L}$:

- simbolul Q corespunde relaţiei $\leq$ în **N**
- simbolul α corespunde numărului natural 1

Variabilele din *S, x* şi *y* iau valori în **N**. Atunci, semnificaţia evidentă a lui *S* în relaţie cu $\mathcal{A}$ este:

S: "Există un număr natural *x* astfel încât, pentru orice număr natural *y*, $x \leq y$"

S specifică astfel că există un element minimal în $\mathcal{A}$ care este evident adevărat în $\mathcal{A}$, 1 fiind elementul minimal din $\mathcal{A}$.

Să definim acum o interpretare diferită pentru acelaşi limbaj $\mathcal{L}$:

$\mathcal{A}' = (\mathbf{N}, >, 1)$

unde Q este predicatul corespunzător relaţiei „>", relaţia „mai mare decât" peste numerele naturale, şi unde α corespunde lui 1.

Pe baza acestei interpretări, semnificaţia lui *S* devine:

S: „Există un număr natural *x* astfel încât, pentru orice număr natural y, $x>y$".

S enunţă că există un element maximal în $\mathcal{A}$, şi acesta nu este, evident, adevărat pentru $\mathcal{A}'$ deoarece $\mathcal{A}'$ nu are un element maximal.

Cu alte cuvinte, observăm că putem atribui interpretări diferite pentru acelaşi limbaj, asociind asfel valori de adevăr diferite.

B) Interpretări şi adevăr: Descriere formală

Vom continua cu descrierea formală a interpretării şi adevărului unei fraze în contextul **LPr**.

Definiţia 7.5.2: Fie

$\mathcal{L} = \{R_0, R_1,\ldots,f_0, f_1,\ldots, c_0, c_1,\ldots\}$

un limbaj al **LPr**, unde R_i sunt simboluri predicative, f_i sunt simboluri funcţionale şi c_i sunt simboluri de constante, i = 0, 1,…

O **interpretare** a lui $\mathcal{L}$:

$\mathcal{A} = \{ A, \varepsilon(R_0), \varepsilon(R_1),\ldots,\varepsilon(f_0), \varepsilon(f_1),\ldots, \varepsilon(c_0), \varepsilon(c_1),\ldots\}$

constă în:

(i) o mulţime $A \neq \emptyset$, universul interpretării

(ii) o relaţie n-ară $\varepsilon(R_i) \subseteq A^n$ pentru orice simbol predicativ R_i de aritate n

(iii)o funcţie $\varepsilon(f_i) : A^n \mapsto A$ pentru fiecare funcţie f_i

(iv)un element $\varepsilon(c_i) \in A$ pentru orice simbol de constantă c_i

$\varepsilon(R_i)$, $\varepsilon(f_i)$ şi $\varepsilon(c_i)$ sunt **interpretările** lui R_i, f_i şi respectiv c_i în $\mathcal{A}$.

Exemplul 7.5.3: Fie

$\mathcal{L} = \{=, \leq, +, *, 0, 1\}$

limbajul **LPr** din exemplul 7.2.2. O interpretare a acestui limbaj este:

$\mathcal{A} = (\mathbf{N}, =, \leq, +, *, \ldots, 0, 1\}$

unde **N** este mulţimea numerelor naturale, "=" relaţia de egalitate în **N,** "≤" relaţia mai mic sau egal" şi "+", "*" adunarea şi înmulţirea în **N**.

Să observăm că, până în acest punct, simbolul "+" din $\mathcal{L}$ este doar un simbol predicativ ternar, de exemplu +(2, 3, 5), 2 + 3 = 5. În interpretare, acest simbol este **ineterpretat** ca ε(+), simbolul adunării numerelor naturale. Vom nota de obicei ε(@) obiectul interpretat de simbolul "@" al limbajului într-o interpretare $\mathcal{A}$.

Avem atunci următoarea interpretare a simbolurilor din $\mathcal{L}$:

$\varepsilon(=) \subseteq N \times N$
$\varepsilon(\leq) \subseteq N \times N$
$\varepsilon(+) \subseteq N \times N \times N$
$\varepsilon(*) \subseteq N \times N \times N$
$\varepsilon(0) \in N$
$\varepsilon(1) \in N$

Se observă că un limbaj al **LPr** şi interpretarea lui diferă semnificativ. Cu toate acestea, pentru simplitate, vom trata de multe ori un limbaj şi interpretarea lui indentic. De exemplu, ≤ va fi utilizat atât ca un predicat al limbajului cât şi ca o interpretare, în loc de ε(≤), care este o submulţime a lui $N \times N$.

Aşa cum s-a văzut în exemplul 7.5.1, un limbaj poate avea multe interpretări.

Exemplul 7.5.4: O interpretare diferită a limbajului $\mathcal{L}$ descris în exemplul precedent este:

$\mathcal{A}' = (\{2n \mid n \in N_0\}, =', *', +', 0')$ (numerele naturale pare)

unde:

$\varepsilon(=)$ este $='$, egalitatea în $\{2n \mid n \in N_0\}$
$\varepsilon(*)$ este $*'$, înmulţirea în $\{2n \mid n \in N_0\}$
$\varepsilon(+)$ este $+'$, adunarea în $\{2n \mid n \in N_0\}$
$\varepsilon(0)$ este $0'$, elementul neutru al lui $+'$

Aşa cum s-a specificat anterior, o frază a unui limbaj poate fi "adevărată" într-o interpretare a limbajului şi "falsă" în altă interpretare. De exemplu,

$(\exists y)(\forall x)(x*y = x)$

care indică existenţa unui element neutru pentru * este adevărată în $\mathcal{A}$ şi falsă în $\mathcal{A}'$.

Vom defini acum inductiv adevărul unei fraze într-o interpretare $\mathcal{A}$.

Definiţia 7.5.5: Definiţia inductivă a adevărului unei fraze într-o interpretare:

Fie:

$\mathcal{L} = \{R_0, R_1, \ldots, f_0, f_1, \ldots, c_0, c_1, \ldots\}$

un limbaj şi $\mathcal{A}$ una din interpretările lui.

(a) Fraza atomică $R_i(c_{i_1}, c_{i_2}, ..., c_{i_n})$ este adevărată în $\mathcal{A}$ dacă şi numai dacă:

$$(\varepsilon(c_{i_1}), ..., \varepsilon(c_{i_n})) \in \varepsilon(R_i)$$

Formal, vom scrie:

$$\mathcal{A} \models R_i(c_{i_1}, c_{i_2}, ..., c_{i_n})$$

Fie φ, φ_1 şi φ_2 trei fraze din $\mathcal{L}$. Atunci:

(b) $\mathcal{A} \models \neg\varphi \Leftrightarrow \mathcal{A} \not\models \varphi$

(c) $\mathcal{A} \models \varphi_1 \vee \varphi_2 \Leftrightarrow \mathcal{A} \models \varphi_1$ sau $\mathcal{A} \models \varphi_2$

(d) $\mathcal{A} \models \varphi_1 \wedge \varphi_2 \Leftrightarrow \mathcal{A} \models \varphi_1$ şi $\mathcal{A} \models \varphi_2$

(e) $\mathcal{A} \models \varphi_1 \rightarrow \varphi_2 \Leftrightarrow \mathcal{A} \models \varphi_2$ sau $\mathcal{A} \not\models \varphi_1$

(f) $\mathcal{A} \models \varphi_1 \leftrightarrow \varphi_2 \Leftrightarrow (\mathcal{A} \models \varphi_1$ şi $\mathcal{A} \models \varphi_2)$ sau $(\mathcal{A} \not\models \varphi_1$ şi $\mathcal{A} \not\models \varphi_2)$

(g) $\mathcal{A} \models (\exists u)\varphi(u) \Leftrightarrow$ există un simbol de constantă $c \in \mathcal{L}$ astfel încât $\mathcal{A} \models \varphi(c)$

(h) $\mathcal{A} \models (\forall u)\varphi(u) \Leftrightarrow$ pentru orice simbol de constantă $c \in \mathcal{L}$, $\mathcal{A} \models \varphi(c)$

Observaţia 7.5.6:

(1) Din definiţia 7.5.5 şi observaţia 7.3.7 avem:

(i) $\mathcal{A} \models c_1 = c_2 \Leftrightarrow \varepsilon(c_1)$ este identică cu $\varepsilon(c_2)$

(2) Dacă fraza atomică $R_i(c_{i_1}, c_{i_2}, ..., c_{i_n})$ este adevărată în $\mathcal{A}$, atunci R_i ia valoarea a în $\mathcal{A}$. Dacă nu este adevărată atunci ia valoarea f

În definiţia de mai sus, inducţia se face după lungimea frazelor. De exemplu, din (c), orice frază disjunctivă $(\varphi_1 \vee \varphi_2)$ este adevărată în $\mathcal{A}$ dacă şi numai dacă cel puţin una din φ_1, φ_2 este adevărată în $\mathcal{A}$.

Toate axiomele din definiţia 7.3.3 şi observaţia 7.3.7 sunt formule adevărate în orice interpretare $\mathcal{A}$. Mai mult, regulile generalizării şi Modus Ponens conduc de la formule adevărate tot la formule adevărate, pentru orice interpretare a **LPr** (corectitudinea sistemului axiomatic al **LPr**).

Exemlul 7.5.7: Fie $\mathcal{L} = \{Q, f\}$ limbajul aritmetic şi fie:

$\mathcal{A} = (\mathbf{Q}_+, =, \varphi)$ şi $\mathcal{A}' = (\mathbf{R}_+, =, g)$

două interpretări ale lui $\mathcal{L}$, unde:

$\mathbf{Q}_+$: mulţimea numerelor raţionale pozitive
$\mathbf{R}_+$: mulţimea numerelor reale pozitive
g: $g(x) = x^2$
$\varepsilon'(Q)$: =, egalitatea în $\mathbf{R}_+$
$\varepsilon(Q)$: =, egalitatea în $\mathbf{Q}_+$
$\varepsilon'(g)$: g definit în $\mathbf{R}_+$
$\varepsilon(g)$: g definit în $\mathbf{Q}_+$

Atunci fraza S: $(\forall x)(\exists y)Q(x, g(y))$, şi anume $(\forall x)(\exists y)$ $(x = y^2)$, este adevărată în $\mathcal{A}'$ dar nu este adevărată în $\mathcal{A}$ deoarece ecuaţia $x = y^2$, cu **x** fixat şi pozitiv, are întotdeauna soluţie în $\mathbf{R}_+$, dar nu are întotdeauna soluţie în $\mathbf{Q}_+$.

Exemplul 7.5.8: Pentru limbajul $\mathcal{L} = \{Q, f, a, b\}$, putem defini următoarea interpretare:

$\mathcal{A}$ = (*A, copil, mama, Ion, Maria, Napoleon*)

unde:

A: mulţimea fiinţelor umane
$\varepsilon(Q)$: relaţia "*copil*", adică $\varepsilon(Q)(x_1, x_2) = copil(x_1, x_2)$ = "x_1 este copilul lui x_2"
$\varepsilon(f)$: relaţia "*mama*", adică $\varepsilon(f)(x) = mama(x)$ = "mama lui x"
$\varepsilon(a)$: *persoana Maria*
$\varepsilon(b)$: *persoana Ion*
$\varepsilon(c)$: *persoana Napoleon*

Observăm că, în $\mathcal{A}$, un simbol care nu apare ca element al limbajului, şi anume c, este totuşi interpretat. Vom vedea în teorema 7.5.14 că o asemenea interpretare a simbolurilor ce nu aparţin limbajului nu reprezintă o problemă.

Fie S: $Q(b, f(b)) \vee (\exists x)\,(Q(a, x))$. Conform interpretării $\mathcal{A}$, interpretarea lui S este:

S: " *Ion este copilul mamei lui Ion*
sau
există o persoană **x** *pentru care Maria este copilul lui* **x**"

Ion este evident copilul mamei lui Ion. S este deci adevărată în interpretarea $\mathcal{A}$ (cazul (c) din definiţia 4.5.5).

Definiţia 7.5.9: Fie $\mathcal{L}$ un limbaj şi **σ** o frază. Interpretarea $\mathcal{A}$ a lui $\mathcal{L}$ este un **model** al formulei **σ** dacă şi numaim dacă $\mathcal{A} \models \sigma$.

Definiţia 7.5.10: Dacă $\mathcal{L}$ este un limbaj şi $\mathcal{A}$ o interpretare a lui $\mathcal{L}$ atunci mulţimea:

$$\theta(\mathcal{A}) = \{\sigma \mid \mathcal{A} \models \sigma\}$$

se numeşte **teoria interpretării**

Cu alte cuvinte, teoria unei interpretări este mulţimea tuturor frazelor care sunt adevărate în acea interpretare.

În exemplul precedent am văzut că este posibil să interpretăm simboluri care nu apar ca elemente ale limbajului. Vom enunţa o definiţie utilă în astfel de situaţii.

Definiţia 7.5.11: Să presupunem că avem un limbaj $\mathcal{L}$ şi o interpretare $\mathcal{A}$ a lui $\mathcal{L}$ pentru care elementele $a_1, a_2, \ldots$ din universul lui $\mathcal{A}$ nu sunt interpretări ale unor simboluri de constante din $\mathcal{L}$.

(1) Formăm un *nou limbaj*:

$\mathcal{L}^* = \mathcal{L} \cup \{c_1, c_2, ..\}$

unde simbolurile c_1, c_2,… sunt constante noi care aparţin lui $\mathcal{L}$. $\mathcal{L}^*$ se numeşte **extensia elementară a lui** $\mathcal{L}$.

(2) Dacă avem interpretarea:

$\mathcal{A} = (A, R_1, R_2, \ldots, P_1, P_2, \ldots, d_1, d_2, \ldots)$

putem forma o nouă interpretare:

$\mathcal{A}^* = (A, R_1, R_2, \ldots, P_1, P_2, \ldots, d_1, d_2, \ldots, a_1, a_2, \ldots)$

a limbajului $\mathcal{L}^* = \mathcal{L} \cup \{c_1, c_2, \ldots\}$, unde $c_1, c_2, \ldots \notin \mathcal{L}$, atribuind astfel o interpretare constantelor c_1, c_2,… şi impunând de fapt $\varepsilon(c_1) = a_1$, $\varepsilon(c_2) = a2\ldots$

$\mathcal{A}^*$ se numeşte atunci **extensia elementară a lui** $\mathcal{A}$.

Exemplul 7.5.12: Fie:

$\mathcal{L} = \{=, \leq, +, *, 0, 1\}$ şi $\mathcal{A} = (\mathbf{N}, =, \leq, +, *, 0, 1)$

Putem atunci formula limbajul:

$\mathcal{L}^* = \{=, \leq, +, *, 0, 1, c_1, c_2, \ldots\}$

şi interpretarea lui:

$\mathcal{A}^* = (\mathbf{N}, =, \leq, +, *, 0, 1, 2, 3, \ldots)$

Observaţia 7.5.13: Am îmbogăţit deja limbajul $\mathcal{L}$ cu constante. $\mathcal{L}$ poate fi îmbogăţit şi cu simboluri funcţionale sau predicative noi. $\mathcal{L}^*$, extensia limbajului $\mathcal{L}$ cu simboluri funcţionale şi predicative noi, este o extensie neelementară a lui $\mathcal{L}$. Extensia corespunzătoare a lui $\mathcal{A}$, anume $\mathcal{A}^*$, este o extensie neelementară a lui $\mathcal{A}$.

Prezentăm acum o teoremă despre realizabilitatea unei fraze în contextul **LPr.**

Teorema 7.5.14: *Fie $\mathcal{L}$ un limbaj, $\mathcal{A}$ o interpretare a lui $\mathcal{L}$, $\mathcal{L}^*$ şi $\mathcal{A}^*$ extensiile elementare asociate, astfel încât toate elementele universului lui $\mathcal{A}^*$ sunt interpretări ale simbolurilor din $\mathcal{L}^*$. Fie* **σ** *o frază în $\mathcal{L}$. Atunci* **σ** *este adevărată în interpretarea $\mathcal{A}$ dacă şi numai dacă este adevărată în $\mathcal{A}^*$. Formal*:

$$\mathcal{A} \models \sigma \Leftrightarrow \mathcal{A}^* \models \sigma$$

Demonstraţie: Prin inducţie după lungimea lui **σ**.

Dacă **σ** este o frază $P(c_1,\ldots,c_k)$, atunci:

$\mathcal{A}^* \models P(c_1,\ldots,c_k) \Leftrightarrow (\varepsilon(c_1), \ldots, \varepsilon(c_k)) \in \varepsilon(P)$

şi $\varepsilon(c_1), \ldots, \varepsilon(c_k) \in \mathcal{A}^*$ $\varepsilon(P) \subseteq (\mathcal{A}^*)^k$

$\Leftrightarrow (\varepsilon(c_1), \ldots, \varepsilon(c_k)) \in \varepsilon(P)$

şi $\varepsilon(c_1), \ldots, \varepsilon(c_k) \in \mathcal{A}$, $\varepsilon(P) \subseteq \mathcal{A}^*$

deoarece **σ** este o frază în $\mathcal{L}$.

$\Leftrightarrow \mathcal{A} \models P(c_1,\ldots,c_k)$.

Cazurile $\sigma: \neg\varphi$, $\sigma: \varphi \vee \varphi'$, $\sigma: \varphi \wedge \varphi'$ şi $\sigma: \varphi \rightarrow \varphi'$ se tratează similar.

Să presupunem că σ este o frază $(\exists x)\varphi(x)$, unde x este singura variabilă liberă din σ. Atunci:

$\mathcal{A}^* \models (\exists x)\varphi(x) \Leftrightarrow$ pentru un simbol de constantă c din $\mathcal{L}^*$, $\mathcal{A}^* \models \varphi(c)$

$\Leftrightarrow \mathcal{A} \models \varphi(c)$ datorită presupunerii inductive şi a faptului că φ este o frază în $\mathcal{L}$.

Cazul σ: $(\forall x)\varphi(x)$ se tratează similar.

Pe baza teoremei 7.5.14, adevărul unei fraze într-o interpretare $\mathcal{A}^*$ nu depinde de alegerea noilor simboluri de constante şi a interpetării lor.

Definiţia 7.5.15: O frază σ a unui limbaj $\mathcal{L}$ este realizabilă sau verificabilă sau consistentă dacă şi numai dacă există o interpretare $\mathcal{A}$ a lui $\mathcal{L}$ în care fraza este adevărată, adică $\mathcal{A} \models \sigma$.

Definiţia 7.5.16: O mulţime de fraze S este **realizabilă** sau **verificabilă** sau **consistentă** dacă există o interpretare în care toate frazele din S sunt adevărate. În caz contrar, S este o mulţime **nerealizabilă** sau **neverificabilă** sau **inconsistentă.**

Exemplul 7.5.17:

Fie: $\mathcal{A} = (\mathbf{N}, =, \leq, +, *, 0, 1)$.

Atunci $\mathcal{A} \models \sigma$, unde σ este oricare din frazele (1) - (4) din exemplul 7.2.8. Vom demonstra că $\mathcal{A} \models (\forall x)(\forall y)(x = y \rightarrow y = x)$ nu este adevărată. Atunci, pe baza definiţiei 7.5.5(h) şi (b) există c_1, c_2 astfel încât:

$\mathcal{A} \models \neg[(c_1 = c_2) \rightarrow (c_1 = c_2)]$

Atunci:

$\mathcal{A} \models \neg[\neg(c_1 = c_2) \vee (c_2 = c_1)]$

sau

$\mathcal{A} \models [(c_1 = c_2) \wedge \neg(c_2 = c_1)]$

şi pe baza definiţiei 4.5.5(d):

$\mathcal{A} \models (c_1 = c_2)$ (1)

şi

$\mathcal{A} \models \neg(c_2 = c_1)$ (2)

Pe baza celei de a şaptea axiome din observaţia 7.3.7, adică axioma substituţiei termenilor egali, folosim (1) pentru a substitui c_2 în locul valorii lui c_1 în (2). Atunci:

$\mathcal{A} \models \neg(c_2 = c_2)$

sau, pe baza definiţiei 7.5.5(g):

$\mathcal{A} \models (\exists x)\neg(x = x)$

şi pe baza dualităţii cuantificatorilor $\forall$ şi $\exists$ avem:

$\mathcal{A} \models \neg(\forall x)(x = x)$

ceea ce este o contradicţie, deoarece contravine reflexivităţii relaţiei de egalitate.

Exemplul 7.5.18: Interpretarea standard a limbajului $\mathcal{L} = \{=, \leq, +, *, 1, 0\}$ este:

$\mathcal{A} = (\mathbf{N}, =, \leq, +, *, 0, 1)$

Altă interpretare a lui $\mathcal{L}$, unde **Q** este mulţimea numerelor raţionale, este:

$\mathcal{B} = (\mathbf{Q}, =, \leq, +, *, 0, 1)$

Fie **σ** fraza (ce reprezintă de fapt definiţia formală a unei relaţii de ordine dense):

$(\forall x)(\forall y)[\neg(x = y) \rightarrow (\exists z)[\neg(z = x) \wedge \neg(z = y) \wedge (x \leq z) \wedge (z \leq y)]]$

Atunci $\mathcal{A} \not\models \sigma$, dar $\mathcal{B} \models \sigma$.

La fel ca şi în **LP,** există fraze în orice limbaj al **LPr** care sunt adevărate în toate interpretările limbajului.

Definiţia 7.5.19: Formule adevărate logic:

Dacă formula σ din limbajul $\mathcal{L}$ este adevărată în orice interpretare a lui $\mathcal{L}$, atunci σ este **logic adevărată.**

Exemplul 7.5.20: Fie φ o frază în $\mathcal{L}$ şi ψ o formulă în $\mathcal{L}$.

Atunci fraza:

S: $((\forall v)(\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow (\forall v)\psi))$

este adevărată în orice interpretare a lui $\mathcal{L}$.

Demonstraţie: Să presupunem că *S* nu este adevărată în toate interpretările lui $\mathcal{L}$. Atunci există o interpretare $\mathcal{A}$ a lui $\mathcal{L}$ astfel încât:

$\mathcal{A} \not\models (\forall v)(\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow (\forall v)\psi)$

Atunci, conform definiţiei 7.5.5, avem:

not$[\mathcal{A} \not\models (\forall v)(\varphi \rightarrow \psi)$ sau $\mathcal{A} \models \varphi \rightarrow (\forall v)\psi]$

şi, pe baza legii lui De Morgan:

not$[\mathcal{A} \not\models (\forall v)(\varphi \rightarrow \psi)]$ şi not$[\mathcal{A} \models \varphi \rightarrow (\forall v)\psi]$

Din regula dublei negaţii obţinem:

$\mathcal{A} \models (\forall v)(\varphi \rightarrow \psi)$ (1) şi $\mathcal{A} \not\models \varphi \rightarrow (\forall v)\psi$ (2)

Atunci, din (2) se obţine $\mathcal{A} \not\models \neg\varphi \vee (\forall v)\psi$ şi, conform definiţiei 7.5.5(b), avem:

$\mathcal{A} \models \varphi$ şi $\mathcal{A} \not\models (\forall v)\psi$ (3)

Din (3) şi definiţia 7.5.5(h) ştim că există un $c \in \mathcal{L}$ astfel încât:

$\mathcal{A} \models \neg\psi(v / c)$

şi din $\mathcal{A} \models \varphi$ şi defiţia 7.5.5(d) avem:

$\mathcal{A} \models \varphi \vee \neg\psi(v / c)$

sau, echivalent:

$\mathcal{A} \models \neg(\neg\varphi \vee \psi(v / c))$

sau, echivalent:

$\mathcal{A} \models \neg((\varphi \rightarrow \psi)(v / c))$ (4)

Din (1) şi definiţia 7.5.5(h), pentru orice $c \in \mathcal{L}$,

$\mathcal{A} \models (\varphi \rightarrow \psi)(v / c)$ (5)

se îndeplineşte. Dar φ este o frază, deci nu are variabile libere. În consecinţă, nu există nici o substituţie posibilă pentru φ. Atunci (5) ia forma $\mathcal{A} \models \varphi \rightarrow \psi(v / c)$ pentru orice $c \in \mathcal{L}$, ceea ce este în contradicţie cu (4). *S* este deci adevărată în toate interpretările lui $\mathcal{L}$.

Definiţia 7.5.21: Fie $\mathcal{L}$ un limbaj al **LPr** şi *S* o mulţime de fraze ale **LPr**. Fraza σ este o **consecinţă** a lui *S*, formal $S \models \sigma$, dacă şi numai dacă orice interpretare $\mathcal{A}$ a lui $\mathcal{L}$ ce verifică toate propoziţiile din *S* verifică σ. Acest lucru se notează:

$\sigma \in Con(S) \Leftrightarrow (\forall \mathcal{A})[\ \mathcal{A} \models S \Rightarrow \mathcal{A} \models \sigma]$

7.6 *Forme normale în logica predicatelor*

În **LP** am discutat despre două forme echivalente ale unei propoziţii, **FNC** (Forma Normal Conjunctivă) şi **FND** (Forma Normal Disjunctivă). Fraze de forme similare se întâlnesc şi în **LPr**. În contextul **LPr** există două forme suplimentare: *Forma Normală Prenex* **(FNP)** şi *Forma Normală Skolem* **(FNS)**, în funcţie de cuantificatorii ce apar în frază. Prin transformarea a două propoziţii într-una din aceste forme normale, putem să le comparăm cu uşurinţă şi să determinăm dacă sunt echivalente, să vedem dacă una este negaţia celeilalte sau dacă prezintă vreo proprietate semnificativă. *Forma Normală Skolem* are o importanţă deosebită în programarea logică.

În secţiunile următoare vom investiga analitic aceste forme.

Definiţia 7.6.1: Formula φ este o **formulă prenex**, pe scurt **FNP**, dacă φ este de forma:

φ: $(Q_1x_1)(Q_2x_2)\ldots(Q_nx_n)\sigma$

unde fiecare Q_i, i = 1,…, n, este unul din cuantificatorii ∀, ∃ şi σ este o formulă fără cuantificatori. $(Q_1x_1)(Q_2x_2)\ldots(Q_nx_n)$ se numeşte **prefixul** lui φ, iar σ se numeşte **matricea** lui φ.

Exemplul 7.6.2: Următoarele fraze sunt în **FNP**:

(i) $(\forall x)(\forall y)[P(x, y) \rightarrow Q(x)]$

(ii) $(\forall x)(\exists y)\,[Q(x, y) \vee P(x, y)]$

Pentru a transforma o formulă sau o frază în **FNP**, pe lângă formulele utilizate în **LP**, în **LPr** folosim de asemenea următoarele formule:

(1) $(Qx)P(x) \wedge G \leftrightarrow (Qx)\,[P(x) \wedge G]$ unde x nu apare liberă în G

(2) $(Qx)P(x) \vee G \leftrightarrow (Qx)\,[P(x) \vee G]$ unde x nu apare liberă în G

(3) $\neg(\forall x)P(x) \leftrightarrow (\exists x)(\neg\, P(x))$

(4) $\neg(\exists x)P(x) \leftrightarrow (\forall x)(\neg P(x))$

(5) $(\forall x)\mathrm{P}(x) \wedge (\forall x)G(x) \leftrightarrow (\forall x)[P(x) \wedge G(x)]$

(6) $(\exists x)P(x) \vee (\exists x)G(x) \leftrightarrow (\exists x)[P(x) \vee G(x)]$

unde (Qx) este $(\forall x)$ sau $(\exists x)$.

Formulele de mai sus sunt derivabile în sistemul axiomatic al **LPr** (teorema 7.3.6). Formulele (1) şi (2) specifică faptul că domeniul de acţiune a cuantificatorilor include conjuncţia şi disjuncţia, cu condiţia ca variabilele cuantificate să nu apară ca variabile libere. Formulele (3) şi (4) sunt cazuri evidente, considerând dualitatea lui ∀ şi ∃. Cazurile (5) şi (6) arată distributivitatea cuantificatorului universal faţă de conjuncţie şi a cuantificatorului existenţial faţă de disjuncţie. Distributivitatea inversă nu este însă adevărată:

(8) $(\forall x)P(x) \wedge (\forall x)G(x) \not\Leftrightarrow (\exists x)[P(x) \wedge G(x)]$

(9) $(\forall x)P(x) \vee (\forall \mathrm{x})G(x) \not\Leftrightarrow (\forall x)[P(x) \vee G(x)]$

Pentru a obţine echivalenţe în cazuri similare cazurilor (7) şi (8), trebuie întâi să redenumim toate apariţiile lui x în formulele $(\forall x)G(\mathrm{x})$ şi $(\exists x)G(x)$. Acest lucru este permis deoarece x nu este variabilă legată. Atunci, conform teoremei 7.3.6, (5) şi (6), obţinem (cum?) următoarele echivalenţe:

(10) $(Q_1x)\mathrm{P}(x) \wedge (Q_2x)G(x) \leftrightarrow (Q_1x)(Q_2z)[P(x) \wedge G(z)]$

(11) $(Q_1x)P(x) \vee (Q_2x)G(x) \leftrightarrow (Q_1x)(Q_2\mathrm{z})[P(x) \vee G(z)]$

unde Q_1, $Q_2 \in \{\forall, \exists\}$ şi unde variabila z nu apare în $G(x)$ şi nu apare ca variabilă liberă în $P(x)$, în (10) şi (11).

Vom descrie acum procedura formală pentru a transforma o frază în **FNP**:

Construcţia 7.6.3:

Pasul 1: Eliminarea simbolurilor $\leftrightarrow$ şi $\rightarrow$ pe baza formulelor:

(1a) $(A \leftrightarrow B) \leftrightarrow ((A \rightarrow B) \wedge (B \rightarrow A))$

(1b) $(A \rightarrow B) \leftrightarrow (\neg A \vee B)$

Pasul 2: Transferarea negaţiei în faţa atomilor pe baza formulelor:

(2a) $\neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$

(2b) $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$

(2c) $\neg(\neg A) \leftrightarrow A$

(2d) $\neg(\forall x)A \leftrightarrow (\exists x)\ \neg A$

(2e) $\neg(\exists x)A \leftrightarrow (\forall x)\ \neg A$

Pasul 3: Transferarea cuantificatorilor la stânga pe baza formulelor:

(3a) $(\forall x)A(x) \wedge (\forall x)B(x) \leftrightarrow (\forall x)\ [A(x) \wedge B(x)]$

$(\exists x)A(x) \vee (\exists x)B(x) \leftrightarrow (\exists x)\ [A(x) \vee B(x)]$

(3b) $(Qx)A(x) \wedge B \leftrightarrow Q(x)\ [A(x) \wedge B]$

$(Qx)A(x) \vee B \leftrightarrow Q(x)\ [A(x) \vee B]$

unde x nu apare ca variabilă liberă în B în formulele (3b).

(3c) $(Q_1x)A(x) \vee (Q_2x)B(x) \leftrightarrow (Q_1x)(Q_2z)\ [A(x) \vee B(z)]$

$(Q_1x)A(x) \wedge (Q_2x)B(x) \leftrightarrow (Q_1x)(Q_2z)\ [A(x) \wedge B(z)]$

unde x nu apare ca variabilă liberă în B în formulele (3c), z nu apare în B şi nu apare ca o variabilă liberă în A, iar $B(z)$ este rezultatul înlocuirii fiecărei apariţii libere a lui x cu z.

Exemplul 7.6.4:

φ:	$(\forall x)P(x) \rightarrow (\exists x)R(x)$	$\leftrightarrow \neg(\forall x)P(x) \vee (\exists x)R(x)$	(1b)
		$\leftrightarrow (\forall x)\ \neg P(x) \vee (\exists x)R(x)$	(2d)
		$\leftrightarrow (\exists x)[\neg P(x) \vee R(x)]$	(3a)

Exemplul 7.6.5:

φ:	$(\forall x)(\forall y)[(\exists z)(P(x, z) \wedge P(y, z)) \rightarrow (\exists u)R(x, y, u)]$	
$\leftrightarrow$	$(\forall x)(\forall y)[\neg(\exists z)(P(x, z) \wedge P(x, z)) \vee (\exists u)R(x, y, u)]$	(1b)
$\leftrightarrow$	$(\forall x)(\forall y)[(\forall z)(\neg P(x, z) \vee \neg P(y, z)) \vee (\exists u)R(x, y, u)]$	(2e, 2b)
$\leftrightarrow$	$(\forall x)(\forall y)(\forall z)[\neg P(x, z) \vee \neg P(y, z) \vee (\exists u)R(x, y, z)]$	(3b)
$\leftrightarrow$	$(\forall x)(\forall y)(\forall z)(\exists u)[\neg P(x, z) \vee \neg P(y, z) \vee R(x, y, u)]$	(3b)

Vom discuta acum transformarea unei fraze φ într-o frază φ^* care este **universală**, adică conţine numai cuantificatori universali, şi într-o *Formă Prenex*, astfel încât φ^* este realizabilă dacă şi numai dacă φ este realizabilă.

7.7 *Forma Normală Skolem*

Teorema 7.7.1: Loewenheim, Skolem:

Pentru orice frază φ *al* ***LPr,*** *putem forma o frază universală* φ^* *astfel încât:*

φ este realizabilă $\Leftrightarrow$ φ^* este realizabilă

Vom descrie acum cum se formează φ^*, cu φ o frază arbitrară a **LPr**.

Definiţia 7.7.2: Fie φ o frază a **LPr**.

Pasul 1:

Determinăm **FNP** a lui φ.

Pasul 2:

Eliminăm treptat fiecare cuantificator existenţial ($\exists y$), înlocuind fiecare apariţie a lui y printr-un simbol funcţional nou, neutilizat, f, cu argumente toate variabilele legate prin cuantificatori universali ce preced ($\exists y$). f se numeşte **funcţie Skolem** a formulei φ, pe scurt **FNS.**

Exemplul 7.7.3: Fie fraza:

φ: $(\forall x)(\exists y)(\forall z)(\exists \upsilon)P(x, y, z, \upsilon)$

care este în **FNP.**

(1). Eliminăm ($\exists y$) şi înlocuim y cu funcţia Skolem $f(x)$. Obţinem astfel fraza:

φ_1: $(\forall x)(\forall z)(\exists \upsilon)P(x, f(x), z, \upsilon)$

(2). Eliminăm ($\exists \upsilon$) din φ_1 şi înlocuim υ cu funcţia Skolem $g(x, z)$, deoarece ($\forall x$), ($\forall z$) precede ($\exists \upsilon$). Obţinem astfel:

φ*: $(\forall x)(\forall z)P(x, f(x), z, g(x, z))$

Intuitiv, în exemplul de mai sus, $f(x)$ indică existenţa unei variabile y care depinde de x. (Conform definiţiei 7.5.5(h) şi (g), pentru fiecare x există o constantă care poate înlocui y). Deci y este o variabilă existenţială, cu alte cuvinte este legată de un cuantificator existenţial. Astfel, φ_1 şi φ* implică φ pe baza lui $f(x)$ şi a lui $g(x, z)$.

Exemplul 7.7.4: FNS a formulei

φ: $(\exists y)(\forall x)(\forall z)\psi(x, y, z)$

este

φ*: $(\forall x)(\forall z)\psi(x, c, z)$

unde c este o constantă deoarece funcţia Skolem f corespunzătoare nu are nici o variabilă (($\exists y$) este primul cuantificator al lui φ, deci y nu este determinată de nici o variabilă şi f este funcţia constantă c).

Transformarea unei fraze în **FNS** este de mare importanţă în programarea logică, acest lucru devenind evident în momentul în care vom prezenta metoda demonstrării prin rezoluţie în **LPr**.

În **LP** (definiţia 3.9.4) am văzut că o frază de forma:

$$(A_{1_1} \vee ... \vee A_{1_n}) \wedge ... \wedge (A_{k1} \vee ... \vee A_{k_n})$$

deci o conjuncţie de disjuncţii, poate fi reprezentată conform teoriei mulţimilor ca o mulţime de forma:

$$\{\{A_{1_1} \vee ... \vee A_{1_n}\},...,\{A_{k1} \vee ... \vee A_{k_n}\}\}$$

Luând în considerare definiţia 7.4.1 cât şi cuantificatorii, prezentăm acum această definiţie în contextul **LPr**.

Definiţia 7.7.5: Fie φ o frază în **LPr** în **FNS**:

φ: $(\forall x_1) \ldots (\forall x_l)A(x_1, \ldots, x_l)$ (*)

unde $A(x_1, \ldots, x_l)$ este conjuncţia:

$$C_1(x_1, \ldots, x_l) \wedge \ldots \wedge C_k(x_1, \ldots, x_l)$$

şi fiecare C_i, $1 \leq i \leq k$, este o disjuncţie de atomi sau de atomi negaţi $P_{1_1},...,P_{1_p}$ în **LPr**.

Atunci φ este reprezentată conform teoriei mulţimilor ca mulţimea:

$S = \{\{P_{1_1},...,P_{1_n}\},...,\{P_{k1},...,P_{k_n}\}\}$ sau $S = \{C_1,\ldots,C_k\}$

Fiecare C_i este o clauză, S fiind mulţimea de clauze.

Exemplul 7.7.6: Fraza:

$$(\forall x)(\forall z)[P_1(x, z) \wedge (P_2(x) \vee P_3(z)) \wedge P_4(z, x)]$$

se reprezintă ca mulţime de clauze prin:

$$S = \{\{P_1(x, z)\}, \{P_2(x), P_3(z)\}, \{P_4(z, x)\}\}$$

Lecţia 8- Interpretări Herbrand şi metode de demonstraţie in logica predicatelor

În cele ce urmează, vom prezenta **interpretările Herbrand** cât şi metodele de demonstraţie a realizabilităţii unei fraze sau a unei mulţimi de fraze a **LPr.**

8.1 *Interpretări Herbrand*

În această secţiune vom descrie un tip special de interpretări, *interpretările Herbrand*, care au un rol catalizator în bazele teoretice ale programării logice.

Interpretările Herbrand au fost subiectul tezei de doctorat a lui J. Herbrand în 1930. Fără contribuţia lui Herbrand, programarea logică ar putea fi încă un vis de neatins. Problema de bază a demonstrării automate a teoremelor este determinarea unei proceduri generale pe baza căreia putem demonstra dacă o frază a **LPr** este adevărată sau nu. În 1936, Turing şi Church, fiecare lucrând independent, au demonstrat că nu există o astfel de procedură. Însă Herbrand a rezolvat această problemă indirect, pe baza unui algoritm de construcţie a unei intepretări ce respinge o formulă dată φ. Dacă φ este adevărată, nu există nici o interpretare care s-o respingă şi algoritmul se opreşte după un număr finit de paşi.

Primele încercări de a utiliza ideile lui Herbrand în programarea logică se situează în anul 1960 şi aparţin lui Gilmore cât şi lui Davis şi Putnam; însă aceste încercări nu au fost încununate de succes. De abia în 1965, Robinson realizează punerea în practică a metodei lui Herbrand prin introducerea şi utilizarea rezoluţiei.

Descrierea universului Herbrand

Fiind dată o frază φ a **LPr**, dorim să determinăm dacă φ este verificabilă sau nu. Decidem astfel despre posibila realizabilitate a unei fraze prin inspectarea corespunzătoare a mulţimii de clauze *S* asociată lui φ.

Cu toate acestea, demonstrarea realizabilităţii sau a nerealizabilităţii tuturor termenilor de bază ce apar într-o clauză este aproape imposibilă. Din acest motiv, creem o mulţime, un univers, în care termenii frazei φ să ia valori. Această mulţime se numeşte *universul Herbrand.* Construcţia **universului Herbrand** se face inductiv, după cum urmează:

Construcţia 8.1. l: Construcţia *universului Herbrand:*
Fie *S* o mulţime de clauze corespunzătoare unei fraze φ.
Pasul l:

$$H_0 = \begin{cases} \{c \mid c \text{ o constanta ce apare in S}\} \\ \{c_0\} \text{ daca S nu contine nici o constanta.} \end{cases}$$

c_0 este o constantă nouă (nouă însemnând că nu apare în *S*), introdusă arbitrar.
Pasul i+1:

$H_{i+1} = H_i \cup \{ f(a_1,\ldots,a_n) / a_j,\ 1 \le j \le n$, sunt termeni din H_i, iar f este o funcţie sau o constantă ce apare în $S\}$.

În final, impunem:

$$H = \bigcup_{i \in N} H_i$$

Atunci H, mulţimea tuturor termenilor formaţi cu constantele din H_0 şi funcţiile care apar în S, se numeşte **universul Herbrand** al mulţimii de clauze S. Mulţimile H_i, i=0,1,2,… se numesc **mulţimi Herbrand** ale lui S.

Exemplul 8.1.2: Fie a o constantă şi S o mulţime de clauze:

$S = \{\{P(a)\},\{\neg P(a), P(f(x))\}\}$

Atunci:

$H_0 = \{a\}$

$H_1 = \{a, f(a)\}$

$H_2 = \{a, f(a), f(f(a))\}$

……

şi în final :

$H = \{a, f(a), f(f(a)), f(f(f(a))),\ldots\}$

Exemplul 8.1.3: Fie programul:

$P(x) \leftarrow Q(f(x), g(x))$

$R(x) \leftarrow$

Introducem constanta a şi avem:

$H = \{a, f(a), g(a), f(f(a)), f(g(a)), g(f(a)), g(g(a)), f(f(f(a))),\ldots\}$

Exemplul 8.1.4:

$S = \{\{P(x), Q(x)\},\{T(x), \neg R(x)\}\}$

Introducem o constantă nouă c_0. Atunci $H_0 = \{c_0\}$. Deoarece nu există simboluri funcţionale în S, avem $H = H_0 = \{c_0\}$.

Teoretic, pentru a determina dacă o frază este realizabilă, trebuie să determinăm dacă există o interpretare (dintr-o mulţime eventual infinită de interpretări) care verifică acea frază. Lucrurile devin mult mai uşoare dacă ne limităm la fraze universale, adică la fraze ce conţin numai cuantificatori universali; în acest caz, trebuie să considerăm numai un anumit grup de interpretări numite **interpretări Herbrand.**

Vom enunţa acum definiţia formală a unei interpretări Herbrand.

Definiţia 8.1.5: *Interpretare Herbrand:*

Fie S o mulţime de clauze şi H **universul Herbrand** corespunzător lui S. O *interpretare Herbrand* $\mathcal{A}_H$ pentru S se defineşte astfel:

(i) H este universul interpretării;

(ii) Interpretarea fiecărui simbol de constantă este constanta însăşi;

(iii) Interpretarea fiecărui termen $f(t_1,\ldots,t_n)$, unde f este o funcţie sau o constantă, este $f(t'_1,\ldots,t'_n)$, unde $t'_1,\ldots,t'_n$ sunt interpretările termenilor $t_1,\ldots,t_n$;

(iv) Interpretarea fiecărui simbol predicativ de aritate n, $P(t_1,\ldots,t_n)$, este o relatie n-ară $P(t'_1,\ldots,t'_n)$ peste H, unde $t'_1,\ldots,t'_n$ sunt interpretările termenilor $t_1,\ldots,t_n$.

Exemplul 8.1.6 : Fie $\mathcal{L} = \{\leq, +, *, s, 0\}$ un limbaj aritmetic, unde s este funcţia succesor. Avem:

$H_0 = \{0\}$

$H_1 = \{0, s(0)\}$

$H_2 = \{0, s(0), s(s(0))\}$

şi, în final,

$H = \{0, s(0), s(s(0)), s(s(s(0))),\ldots\}$

Interpretările elementelor din $\mathcal{L}$ diferite de 0 sunt relaţiile corespunzătoare peste H. De exemplu, pentru s şi $\leq$ avem, conform definiţiei 7.5.2:

$\varepsilon(s) : H \mapsto H = a \mapsto s(a) \in H$

$\varepsilon(\leq) \subseteq H_2 : \varepsilon(\leq) = \{(0, s(0)), (s(0), s^2(0)),\ldots,(s^n(0), s^{n+1}(0)),\ldots\}$

Enunţăm acum teorema de bază a interpretărilor Herbrand.

Teorema 8.1.7: *Fie* φ *o frază universală şi S mulţimea de clauze asociată acestei fraze. Atunci* φ *este realizabilă (în anumite interpretări) dacă şi numai dacă este realizabilă într-o interpretare Herbrand.*

Demonstraţie:

($\Leftarrow$): O interpretare Herbrand este o interpretare (cazul trivial).

($\Rightarrow$): Dorim să demonstrăm că dacă φ este verificabilă într-o interpretare $\mathcal{A}$ atunci putem defini relaţii în universul Herbrand care satisfac clauzele din S. Să presupunem deci că φ este verificată într-o interpretare $\mathcal{A}$ cu A universul acesteia. Pentru a specifica interpretarea diverselor simboluri din $\mathcal{L}$, adică a limbajului care conţine fraza φ, folosim:

- $\varepsilon(c) = \hat{c} \in A$, pentru fiecare simbol de constantă c
- $\varepsilon(f) = \hat{f} : A^n \mapsto A$, pentru fiecare funcţie de n variabile
- $\varepsilon(t) = \varepsilon(f(t_1, ...,t_n)) = \hat{f}(\hat{t}_1,...,\hat{t}_n)$, pentru fiecare termen $f(t_1, ...,t_n)$
- $\varepsilon(R) = \hat{R} \subseteq A^n$, pentru fiecare predicat n-ar R

Pentru fiecare simbol predicativ R de aritate n definim relaţia R_H în H (definiţia 8.1.5, definiţia 7.5.2 (ii), definiţia 7.5.5) după cum urmează:

$R_H(t_1, ...,t_n) \Leftrightarrow (\hat{t}_1,...,\hat{t}_n) \in \hat{R}$

Avem astfel o interpretare Herbrand A_H.

Să impunem $A' = \{ t \mid t \in H\}$. Structura $\mathcal{A}'$, cu A' universul asociat şi cu restricţiile relaţiilor lui $\mathcal{A}$ în A', este evident o interpretare a mulţimii S. Mai mult, $\mathcal{A} \models \varphi$ şi orice apare în φ apare, prin definiţie, şi în A. Deci $\mathcal{A}' \models \varphi$. Atunci, pe baza definiţiei lui $\mathcal{A}_H$, $\mathcal{A}_H \models \varphi$.

În consecinţă, φ este într-adevăr realizabilă într-o *interpretare Herbrand.*

Pe baza teoremei 8.1.7, dacă φ nu este verificabilă într-o *interpretare Herbrand,* atunci φ nu este realizabilă; nu există nici o interpretare care satisface φ.

Cu alte cuvinte:

- *Pe baza interpretărilor Herbrand, reducem nerealizabilitatea unei mulţimi de clauze la nerealizabilitatea unei mulţimi de instanţe de bază a acestor clauze în universul Herbrand. Deoarece în instanţa de bază a unei clauze nu apar variabile, realizabilitatea poate fi demonstrată pe baza metodelor* **LP***, de exemplu metoda tablourilor semantice sau rezoluţie.*

Demonstraţiile Beth, la fel ca şi rezoluţia, sunt demonstraţii algoritmice (spre deosebire de metodele uzuale de realizabilitate în **LPr**, exemplul 7.5.17), Acest rezultat este cunoscut ca *teorema lui Herbrand,* indiferent de forma sub care se găseşte în bibliografia

clasică sau modernă. Teorema lui Herbrand va fi analizată în secţiunea următoare, ce se ocupă de demonstraţii pe baza arborilor semantici.

8.2 *Demonstraţii cu tablouri sematice*

În **LPr,** la fel ca şi în **LP,** putem determina dacă o frază sau o mulţime de fraze este realizabilă. Metodele de demonstrare discutate anterior pot fi utilizate şi în contextul **LPr.**

Să începem cu metoda tablourilor semantice. Tablourile semantice sunt folosite pentru determinarea valorii de adevăr a unei fraze compuse a unui limbaj $\mathcal{L}$ a LPr.

Definiţia 8.2.1: Demonstraţii cu tablouri sematice complete:

Să presupunem că avem un limbaj $\mathcal{L}$ şi că c_0, c_1,..sunt simbolurile constante din limbaj, formând o listă de constante. (Semnificaţia acestei liste va fi clarificată în construcţia 8.2.5).

Fie σ, σ_1, σ_2 fraze din $\mathcal{L}$ .Tablourile semantice corespunzătoare sunt prezentate în tabelul următor.

Tablourile semantice din **LPr** sunt o extensie a tablourilor semantice din **LP,** care include cazuri suplimentare pentru cuantificatori.

Cu ajutorul tabloului semantic:

$a(\forall x)\varphi(x)$
|
$a\varphi(c)$
pentru oricare c

reprezentăm faptul că "pentru ca $(\forall x)\ \varphi(x)$ să fie adevărată, $\varphi(x)$ trebuie să fie adevărată pentru orice constantă c".

Corespunzător, cu tabloul semantic

$a(\exists x)\varphi(x)$
|
$a\varphi(c)$
pentru c nou

reprezentăm faptul că "pentru ca $(\exists x)\varphi(x)$să fie adevărată trebuie să existe o constantă c, care nu a apărut încă în tablou, astfel încât $\varphi(c)$ să fie adevărată".

1	2
$a\ (\neg\sigma)$ | $f\sigma$	$f\ (\neg\sigma)$ | $a\sigma$

3 $a\,(\sigma_1 \vee \sigma_2)$ $a\sigma_1 \quad a\sigma_2$	**4** $f\,(\sigma_1 \vee \sigma_2)$ $f\sigma_1$ $f\sigma_2$	**5** $a\,(\sigma_1 \wedge \sigma_2)$ $a\sigma_1$ $a\sigma_2$	**6** $f\,(\sigma_1 \wedge \sigma_2)$ $f\sigma_1 \quad f\sigma_2$
7 $a\,(\sigma_1 \to \sigma_2)$ $f\sigma_1 \quad a\sigma_2$	**8** $f\,(\sigma_1 \to \sigma_2)$ $a\sigma_1$ $f\sigma_2$	**9** $a\,(\sigma_1 \leftrightarrow \sigma_2)$ $a\sigma_1 \quad f\sigma_1$ $a\sigma_2 \quad f\sigma_2$	**10** $f\,(\sigma_1 \leftrightarrow \sigma_2)$ $a\sigma_1 \quad f\sigma_1$ $f\sigma_2 \quad a\sigma_2$
11 $a\,(\forall x)\varphi(x)$ $a\varphi(c)$ pentru oricare c	**12** $f\,(\forall x)\varphi(x)$ $f\varphi(c)$ pentru c nou	**13** $a\,(\exists x)\varphi(x)$ $a\varphi(c)$ pentru c nou	**14** $f\,(\exists x)\varphi(x)$ $f\varphi(c)$ pentru oricare c

Tablourile semantice ale **LPr** se numesc **tablouri sistematice complete**, pe scurt **TSC**. Construcţia unui tablou semantic complet al unei fraze în **LPr** este analoagă cu construcţia **LP** corespunzătoare. Să începem prin a studia câteva exemple.

Exemplul 8.2.2: Să presupunem că dorim să demonstrăm că:

σ : $(\forall x)\varphi(x) \to (\exists x)\varphi(x)$

este logic adevărată, unde φ este o frază în **LPr**.

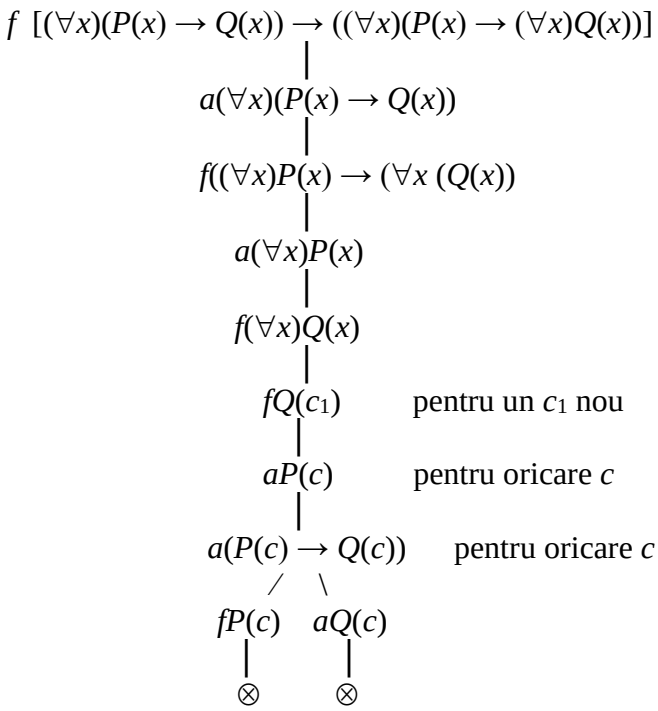
Începem cu un tablou care are $f\varphi$ ca origine:

	$f((\forall x)\varphi(x) \to (\exists x)\varphi(x)$	nod 1
tablou 8	$a(\forall x)\varphi(x)$	nod 2
	$f(\exists x)\varphi(x)$	nod 3
din nodul 3	$a\varphi(c)$ pentru oricare c	nod 4
din nodul 2	$f\varphi(c)$ pentru oricare c	nod 5
contradicţie între 4, 5	$\otimes$	

Pentru ultimul nod al tabloului semantic am utilizat aceeaşi constantă c pentru a genera contradicţia. Acest lucru este permis deoarece tabloul frazei $(\forall x)\varphi(x)$ permite introducerea oricărei constante.

Intuitiv, demonstraţia Beth de mai sus arată că $(\forall x)\varphi(x) \to (\exists x)\varphi(x)$ este o frază logic adevărată deoarece orice încercare de a demonstra că este falsă a rezultat într-o contradicţie.

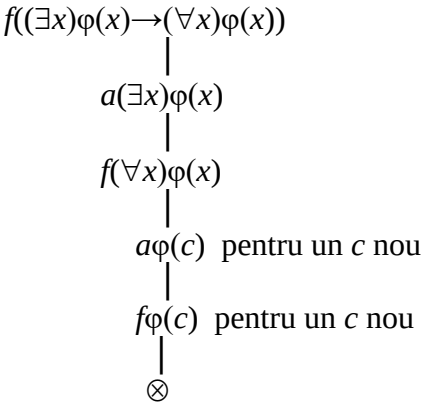
Exemplul 8.2.3:



Tabloul semantic al lui $a(\forall x)\varphi(x)$ - dual, al lui $f(\exists x)\varphi(x)$ - ne permite să declarăm $\varphi(c)$ adevărată - dual, falsă - pentru orice constantă c. Tabloul semantic $a(\exists x)\varphi(x)$ ne permite să declarăm $\varphi(c)$ adevărată numai pentru acele constante c care nu au apărut anterior în tabloul semantic.

Următorul exemplu arată ce s-ar întâmpla fără această restricţie.

Exemplul 8.2.4: Fie fraza $(\exists x)\varphi(x) \rightarrow (\forall x)\varphi(x)$. Această frază nu este logic adevărată deoarece existenţa unui x pentru care $\varphi(x)$ este adevărată nu implică adevărul lui $\varphi(x)$ pentru orice x (de exemplu, existenţa lui $x > 3$ nu implică faptul că pentru orice x este adevărat $x > 3$). Dar:



În nodul 5 nu am fi avut dreptul să utilizăm aceeaşi constantă c ca în nodul precedent 4. Am "demonstrat" astfel că $(\exists x)\varphi(x) \rightarrow (\forall x)\varphi(x)$ este o frază logic adevărată, ceea ce, evident, nu este corect.

Datorită tablourilor 11 şi 14(din definiţia tablourilor semantice, definiţia 8.2.1), un tablou semantic se poate dezvolta la infinit dacă nu există o contradicţie pe una din ramurile acestuia. (În exemplele 8.2.2 şi 8.2.3 nu a fost nevoie să scriem toate constantele din tablourile 4 şi 13). Acest lucru va fi înţeles mai bine din următoarea construcţie formală a unui tablou sematic complet pentru o frază φ.

Construcţia 8.2.5: Construcţia unui tablou sematic complet:

Construcţia începe cu formulele cu semn $f\varphi$ sau $a\varphi$ ca origine a tabloului. Continuăm apoi inductiv.

Pasul n:

Am format deja un tablou T_n.

T_n va fi extins la un nou tablou T_{n+1} prin utilizarea anumitor noduri din T_n.

Pasul $n+1$:

Fie X nodul neutilizat şi neatomic cel mai depărtat la stânga dintre nodurile echidistante faţă de origine. Dacă nu există un astfel de nod X, tabloul sematic este complet. Dacă există un astfel de nod, construim tabloul T_{n+1} prin extinderea fiecărei ramuri necontradictorii ce trece prin X cu concatenarea (la sfârşitul fiecărei ramuri) a tabloului corespunzător lui X. Analizăm noile cazuri:

Cazul 1: X este $a((\forall x)\varphi(x))$.

Fie c_k primul simbol de constantă din lista tuturor constantelor din limbaj astfel încât c_n nu apare în nici o ramură ce trece prin X. Adăugăm atunci $a\varphi(c_n)$ la sfârşitul fiecărei ramuri necontradictorii ce trece prin X, conform tabloului:

$$a(\forall x)\varphi(x)$$
$$\Big|$$
$$a\varphi(c_n)$$

Cazul 2: X este $f((\forall x)\varphi(x))$.

Fie c_k primul simbol de constantă din listă ce nu apare în nici un nod din ramura ce trece prin X. Atunci adăugăm $f\varphi(c_k)$ la sfârşitul fiecărei ramuri ce trece prin X, conform tabloului:

$$f(\forall x)\varphi(x)$$
$$\Big|$$
$$f\varphi(c_k)$$

Cazurile 3, 4: X este $f((\exists x)\varphi(x))$ şi, respectiv, $a((\exists x)\varphi(x))$. Aceste cazuri sunt duale cazurilor l şi 2.

Intuitiv, în cazurile l şi 3 (şi dual în 2 şi 4) dorim să evităm repetarea şi declarăm (φ(c) adevărat pentru constante noi, utilizând astfel toată lista de constante (nu într-o perioadă de timp finită, evident).

Definiţia 8.2.6: Un **tablou sematic complet,** pe scurt **TSC**, este reuniunea tuturor tablourilor T_n din construcţia precedentă, adică:

$$T = \cup_{n\in N} T_n$$

Un **TSC** poate avea un număr infinit de noduri, în timp ce tablourile semantice din **LP** sunt întotdeauna finite.

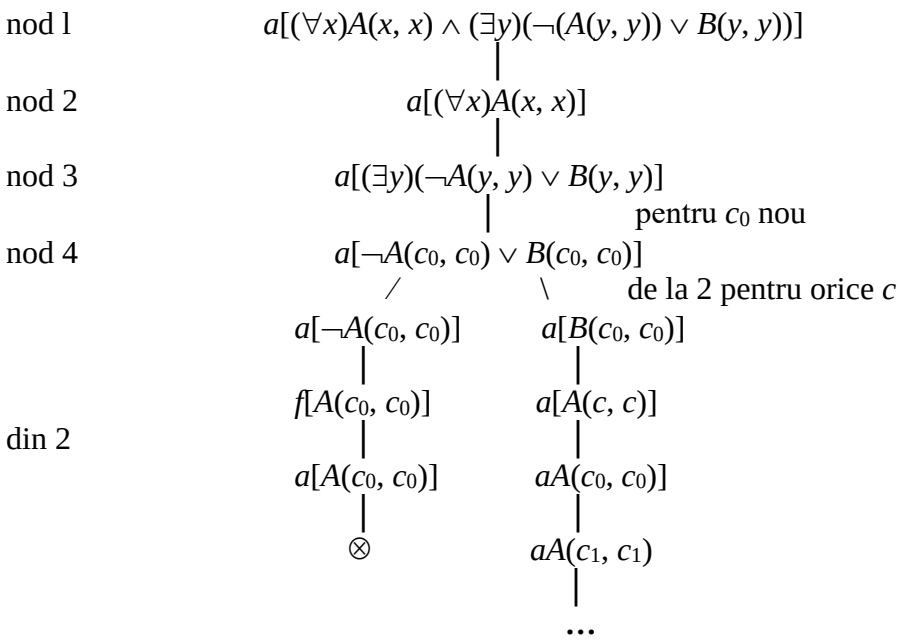
Definiţia 8.2.7:

(i) Un **TSC** este **contradictoriu** dacă toate ramurile lui sunt contradicţii.

(ii) O frază φ este **demonstrabilă Beth(respinsă Beth)** dacă există un **TSC** contradictoriu cu originea în $f\sigma(a\sigma)$. Faptul că σ este demonstrabilă Beth se notează $\vdash_B \sigma$.

(iii) O frază σ este **demonstrabilă Beth** dintr-o mulţime de fraze *S* dacă există un **TSC** contradictoriu cu originea în $f\sigma$ şi un nod următor aP, unde *P* este conjuncţia frazelor din *S*. Acest lucru se notează $S \vdash_B \sigma$.

Exemplul 8.2.8:



În acest exemplu, ramura din stânga este contradictorie în timp ce ramura din dreapta continuă la infinit.

Demonstraţiile pe baza arborilor semantici sunt destul de asemănătoare cu demonstraţiile Beth. Să vedem o descriere informaiă a acestei probleme şi metode.

Arbori semantici: Descriere informală

Fie φ o frază şi *S* mulţimea de clauze corespunzătoare lui φ. Dacă φ este realizabilă, atunci este adevărată într-o interpretare Herbrand (teorema 8.1.7). În consecinţă, toate instanţele de bază ale clauzelor din *S* sunt adevărate în această interpretare. Dacă φ este nerealizabilă, orice încercare de a verifica toate instanţele de bază ale clauzelor din *S* prin intermediul unei valorizări a atomilor de bază $R(t_1, t_2,..., t_n)$ unde $t_1, t_2,..., t_n$ aparţin universului Herbrand, trebuie să eşueze. Acest eşec este confirmat într-un număr finit de paşi prin construcţia unei mulţimi finite de instanţe de bază nerealizabile într-un univers Herbrand. Astfel, conform teoremei 8.1.7, aceste instanţe sunt nerealizabile în orice interpretare.

Se constantă astfel că problema este aceea de a construi o procedură care, pornind de la o frază φ şi mulţimea corespunzătoare de clauze *S*:

(i) dacă φ este nerealizabilă, procedura se termină după un număr finit de paşi pentru o mulţime finită de instanţe de bază;

(ii) dacă φ este realizabilă, procedura nu produce nimic într-o perioadă finită de timp dar constituie totuşi construcţia unei interpretări Herbrand ce satisface φ.

Cu alte cuvinte, cu ajutorul acestei proceduri dorim să obţinem demonstraţia unui contraexemplu a nerealizabilităţii frazei. Construcţia unei astfel de proceduri necesită utilizarea arborilor semantici.

Definiţia 8.2.9: Un **arbore** este o structură $T = \{X, r\}$, unde X este mulţimea nodurilor lui T, iar r este o relaţie binară în X astfel încât:

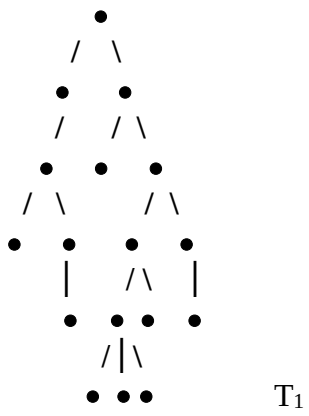
(1) dacă $x, y \in X$ şi $x\ r\ y$, atunci x se numeşte nodul predecesor al lui y şi y **nodul succesor** al lui x;

(2) există exact un singur nod în T care nu are un nod predecesor. Acest nod se **numeşte originea** sau **rădăcina** lui T;

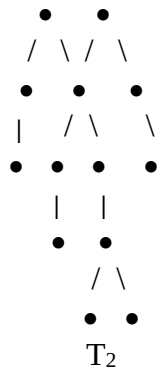
(3) fiecare nod care diferă de originea lui T are exact un unic predecesor.

Linia ce uneşte un nod cu nodul lui succesor se numeşte un **arc** al lui T. Un **nod final** este un nod fără succesori. Secvenţa de arce ce leagă originea arborelui de un nod final se numeşte **ramură** a lui T.

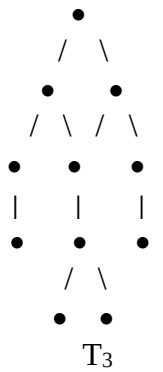
Exemplul 8.2.10:



T_1



T_2



T_3

Diagrama T_1 este un arbore. Se observă că ramurile arborelui sunt orientate în jos.

T_2 nu este un arbore deoarece are două origini şi, mai mult, există un nod cu doi predecesori. T_3 nu este, de asemenea, un arbore deoarece are un nod cu două noduri predecesoare.

Observaţia 8.2.11: Tabloul semantic al unei propoziţii a **LP** şi tabloul sematic complet al unei fraze din **LPr** sunt arbori.

Definiţia 8.2.12: Arbori semantici:

Fie S *o* mulţime de clauze, $S = \{C_1,...,C_n\}$, $P_1,...,P_\lambda$ atomii ce apar în clauzele din S şi $\{a_1,...,a_n,...\}$ universul Herbrand al lui S. Un **arbore semantic** al lui S este un arbore care satisface următoarele condiţii:

(1) Originea lui T este un punct arbitrar. Nodurile diferite de origine sunt instanţe de bază ale $P_1,...,P_\lambda$ în universul $\{a_1,...,a_n,...\}$. Fiecare nod are exact doi succesori, respectiv $P_i(a_{i1},...,a_{ik})$ şi $\neg P_i(a_{i1},...,a_{ik})$;

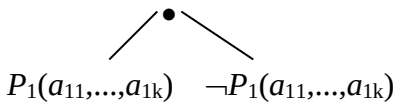
(2) Fiecare ramură a lui T conţinând exact instanţele de bază:
$P_{ip}(a_{i1},...,a_{ik})$ şi $P_{ip}(a_{p1},...,a_{pk})$
reprezintă conjuncţia:
$P_{ip}(a_{i1},...,a_{ik}) \wedge ... \wedge P_{ip}(a_{p1},...,a_{pk})$;

(3) Disjuncţia conjuncţiilor asociate oricărei ramuri din T este logic adevărată;

(4) Dacă un nod din T este $P_i(a_{i1},...,a_{ik})$, atunci nodul succesor nu poate fi $\neg P_i(a_{i1},...,a_{ik})$;

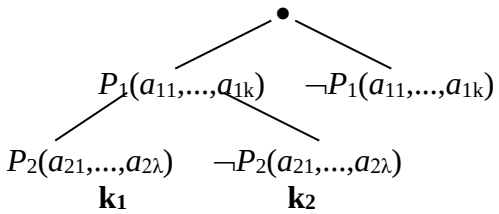
(5) Dacă în timpul construcţiei lui T se obţine un nod k care contrazice o instanţă de bază a uneia din clauzele $C_1,...,C_n$ din S, atunci k este un nod final şi ramura corespunzătoare se numeşte **contradictorie**.

În practică, pentru a construi un arbore semantic pentru o mulţime de clauze S, începem cu $P_1(a_{11},...,a_{1k})$:



Dacă una din clauzele din S conţine $P_1(a_{11},...,a_{1k})$ atunci ramura din dreapta este contradictorie, $\neg P_1(a_{11},...,a_{1k})$ este un nod final şi construcţia continuă pe ramura din stânga. Dacă una din clauzele din S conţine $\neg P_1(a_{11},...,a_{1k})$, atunci ramura din stânga este contradictorie, $P_1(a_{11},...,a_{1k})$ este un nod final şi construcţia continuă cu ramura din dreapta.

Să presupunem că $P_1(a_{11},...,a_{1k})$ nu este un nod final. Continuăm cu nodul $P_2(a_{21},...,a_{2\lambda})$. (Alegerea atomului cu care continuăm construcţia este a noastră).



Ramura k_1 este conjuncţia $P_1(a_{11},...,a_{1k}) \wedge P_2(a_{21},...,a_{2\lambda})$. Ramura k_2 este conjuncţia $P_1(a_{11},...,a_{1k}) \wedge \neg P_2(a_{21},...,a_{2\lambda})$. Verificăm dacă una din clauzele din S conţine $\neg P_1(a_{11},...,a_{1k})$ sau $\neg P_2(a_{21},...,a_{2\lambda})$. Dacă se întâmplă aşa, k_1 este ramură contradictorie şi continuăm cu k_2. Continuăm astfel inspecţia şi construcţia arborelui semantic în scopul de a ajunge la noduri finale, folosind toţi atomii din S.

Definiţia 8.2.13:

(1) O mulţime de clauze S se numeşte **respinsă de un arbore semantic** dacă există un arbore semantic pentru S pentru care toate ramurile sunt contradictorii.

(2) O ramură ***k*** a unui arbore semantic a unei mulţimi de clauze *S* se numeşte **completă** dacă pentru oricare instanţă de bază $P_1(a_1,...,a_n)$ a fiecărui atom *P*, fie $P_1(a_1,...,a_n)$ fie $\neg P_1(a_1,...,a_n)$ este conţinută în ***k***.

Exemplul 8.2.14:

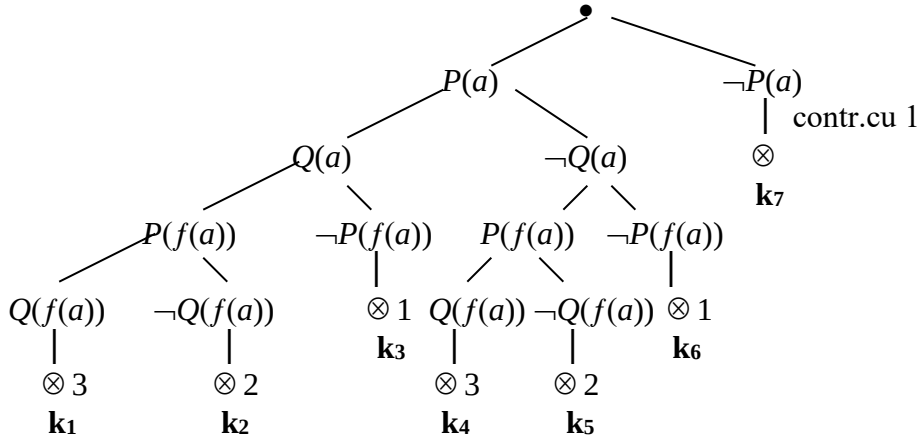
Fie $\varphi : (\forall x)[([(\forall x)(P(x) \wedge (\neg(P(x) \vee Q(f(x))) \wedge \neg Q(f(x))]$

Atunci $S = \{ \underbrace{\{P(x)\}}_{1}, \underbrace{\{\neg P(x), Q(f(x))\}}_{2}, \underbrace{\{\neg Q(f(x))\}}_{3} \}$

şi H = universul Herbrand = $\{a, f(a), f(f(a)), f(f(f(a))),...\}$

Instanţe de bază = $\{P(a), Q(a), P(f(a), Q(f(a)),...\}$

Un arbore semantic pentru *S* este T_1:



Să vedem de ce **k2** este o ramură contradictorie: **k2** reprezintă conjuncţia:

$$\neg Q(f(a)) \wedge P(f(a)) \wedge Q(a) \wedge P(a)$$

Cea de a doua clauză a lui *S* este $\{\neg P(x), Q(f(x))\}$. Pe baza lui *S*, cerem ca:

$$\neg P(x) \vee Q(f(x))$$

să fie validă pentru orice *x*. Cu toate acestea **k2** spune că există un *x* în universul Herbrand, *x* care este egal cu *a* şi pentru care avem:

$$\neg Q(f(a)) \wedge P(f(a)) \wedge Q(a) \wedge P(a)$$

Atunci avem de asemenea:

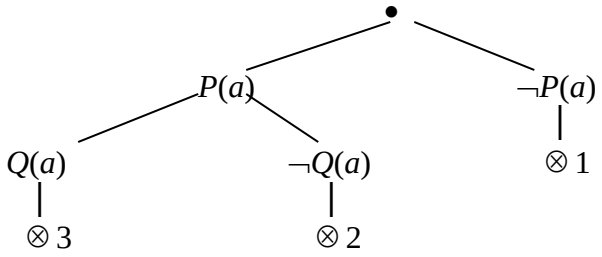
$$\neg Q(f(a)) \wedge P(a) \qquad \text{(pe baza A} \wedge \text{B} \rightarrow \text{A)}$$

şi conform De Morgan:

$$\neg(Q(f(a)) \wedge P(a))$$

cu alte cuvinte **k2** determină o contradicţie cu cea de a doua clauză din *S*.

Aşa cum am spus mai înainte, noi alegem ordinea în care atribuim valori din *H* atomilor din *S*. Astfel, dacă alegem *Q(f(a))* după *P*(*a*), avem un arbore semantic foarte simplu pentru *S*, T_2, cu toate ramurile contradictorii:



Exemplul 8.2.15:

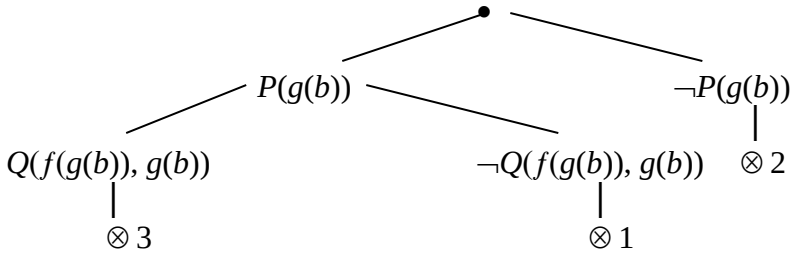
Fie $\varphi : (\forall x)(\forall z)((\neg P(x) \vee Q(f(x), x))) \wedge P(g(b)) \wedge \neg Q(x, z)$

Atunci $S = \{ \underbrace{(\neg P(x), Q(f(x), x)\}}_{1}, \underbrace{\{P(g(b))\}}_{2}, \underbrace{\{\neg Q(x, z)\}}_{3}\}$

şi $H = \{b, f(b), g(b), f(f(b)), f(g(b)),...\}$

Instanţe de bază = $\{P(b), Q(f(b), b), P(g(b)), Q(f(g(b)), g(b),...\}$

Arborele semantic pentru *S* este:



Aşa cum am mai spus, ordinea în care utilizăm atomii în construcţia arborelui are un efect semnificativ asupra numărului de paşi necesari pentru atingerea nodurilor finale şi producerea ramurilor contradictorii.

În observaţia 8.2.11, am văzut că tablourile semantice şi tablourile sematice complete sunt arbori. Pe baza lemei lui Koenig (lema 6.2.9), fiecare arbore finit cu un număr de noduri infinite are cel puţin o ramură infinită. Atunci, dacă *S* nu este realizabilă, construcţia arborelui ei semantic va conduce, într-un număr finit de paşi, la găsirea unei interpretări Herbrand *H* astfel încât $\mathcal{A}_H \not\models S$. Dacă *S* este realizabilă, atunci construcţia corespunzătoare va conduce la un arbore infinit, fiecare ramură a acestui arbore determinând interpretarea Herbrand ce satisface *S*. Această concluzie reprezintă esenţa **teoremei lui Herbrand**, pe care o vom demonstra în continuare prin folosirea unei metode de construcţie a unui arbore semantic corespunzător unei mulţimi de clauze *S*.

Teorema 8.2.16: Teorema lui Herbrand:

Dacă S este o mulţime de clauze nerealizabilă, atunci S esîe respinsă de un arbore semantic.

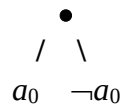
Demonstraţie: Vom descrie un algoritm de construcţie a unui arbore semantic pentru *S*. Formăm universul Herbrand al lui *S* şi mulţimea:

$\{a_0, a_1,...\}$

a instanţelor de bază din *S*. Atunci:

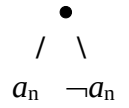
Pasul 0:

Construim arborele:



Pasul n:

Concatenăm la fiecare nod final a unei ramuri necontradictorii **K** extensia:



Să presupunem că *S* nu este respinsă de un arbore semantic. Atunci, construcţia descrisă de algoritmul anterior nu se va termina niciodată. Cu toate acestea, în acest caz *Iema lui Koenig* garantează că arborele va conţine o ramură infinită **K**. Pentru fiecare instanţă de bază a_n, va conţine fie a_n fie negaţia $\neg a_n$ ca nume a nodului.

Definim acum o *interpretare Herbrand*, după cum urmează:

Pentru fiecare simbol predicativ *n*-ar *P* şi termeni t_1, t_2,..., t_n cu interpretări ce aparţin universului Herbrand al lui *S*, interpretarea lui *P* este relaţia:

$$\varepsilon(P)(\varepsilon(t_1),\ldots,\varepsilon(t_n))$$

unde $P(t_1,...,t_n)$ este numele unui nod de-a lungul unei ramuri infinite. Această interpretare satisface, evident, toate clauzele din *S*. În consecinţă, *S* este realizabilă.

De fapt, *teorema lui Herbrand* oferă un algoritm ce investighează realizabilitatea unei fraze sau a unei mulţimi de clauze S pe baza metodelor logicii propoziţiilor; de exemplu, tablouri semantice sau rezoluţie. Dacă *S* este nerealizabilă, atunci există o mulţime de instanţe de bază a clauzelor din *S* care nu este realizabilă. Această mulţime finită constă în propoziţii **LP** şi nerealizabilitatea ei poate fi pusă în evidenţă cu metode care ne sunt deja cunoscute. Astfel, pentru orice mulţime de clauze *S*, începem prin a număra toate instanţele de bază ale clauzelor din *S*. În timpul acestui proces, verificăm sistematic realizabilitatea fiecărei submuţimi finite de instanţe de bază prin metode ale logicii propoziţiilor. Dacă *S* nu este realizabilă, atunci această verificare va arăta că una dintre submulţimi, finită, nu este realizabilă. Dacă *S* este realizabilă, verificarea poate continua la infinit.

Exemplul 8.2.17: Fie fraza din exemplul 8.2.15:

$$\varphi: \quad (\forall x)(\forall z)((\neg P(x) \vee Q(f(x), x))) \wedge P(g(b)) \wedge \neg Q(x, z)$$

Să se determine dacă φ este realizabilă. Mulţimea corespunzătoare de clauze este:

$$S = \{ \underbrace{(\neg P(x), Q(f(x), x)\}}_{1} , \underbrace{\{P(g(b))\},}_{2} \underbrace{\{\neg Q(x, z)\}}_{3} \}$$

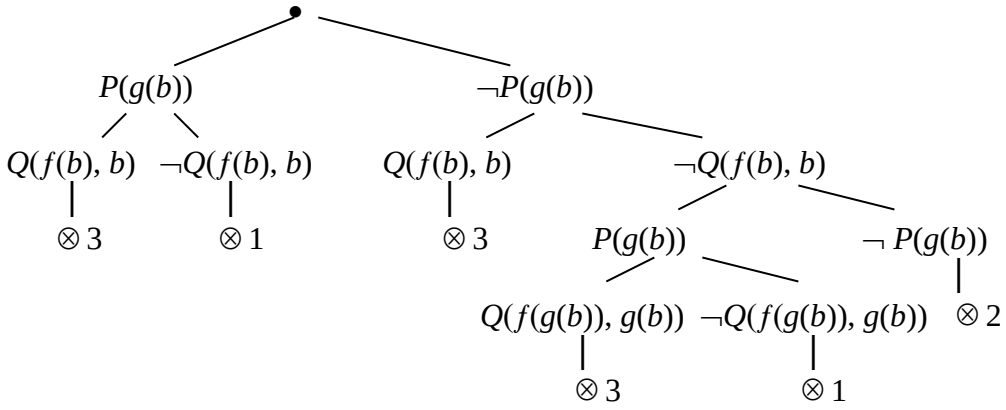
Universul Herbrand al lui S este:

$$H = \{b, g(b), f(b), f(g(b)), g(f(b)),...\}$$

Mulţimea instanţelor de bază ale atomilor din S este

$$\{P(b), Q(f(b), b), P(g(b)), Q(f(g(b)), g(b), P(f(b)),...\}$$

Construim arborele semantic sistematic pentru *S* conform metodei descrise în teorema 4.8.16.



Arborele semantic reprezintă demonstraţia faptului că *S* nu este realizabilă şi stabileşte o anumită submulţime de instanţe de bază ale clauzelor din *S* care nu este realizabilă. Aceste

instanţe de bază sunt acelea şi numai acelea care sunt utilizate pentru a arăta că o anumită ramură este contradictorie. Concret:

$$\{\{\neg Q(f(b), b)\}, \quad \{\neg Q(f(g(b)), g(b))\}, \quad \{P(g(b))\},$$
$$\{\neg P(b), Q(f(b), b)\}, \quad \{\neg P(g(b)), Q(f(g(b)), g(b))\}\}$$

Primele două instanţe provin din clauza 3 din *S*, cea de a treia din clauza 2 şi ultimele două din clauza 1. Nerealizabilitatea acestei mulţimi finite se poate într-adevăr demonstra, de exemplu, prin metoda rezoluţiei în logica propoziţiilor. Să notăm:

A: $Q(f(b), b)$
B: $Q(f(g(b)), g(b))$
C: $P(g(b))$
D: $P(b)$

Atunci, putem scrie submulţimea finită a instanţelor de bază ale clauzelor din *S* după cum urmează:

$$S' = \{\underbrace{\{\neg A\}}_{1}, \underbrace{\{\neg B\}}_{2}, \underbrace{\{C\}}_{3}, \underbrace{\neg D, A}_{4}, \underbrace{\{\neg C, B\}}_{5}\}$$

Folosind rezoluţia, obţinem:

(1) $\neg A$
(2) $\neg B$
(3) C
(4) $\neg D, A$
(5) $\neg C, B$
(6) B din (3) şi (5)
(7) □ din (2) şi (6)

Trebuie să observăm în acest moment că algoritmul de construcţie a unui arbore semantic nu produce întotdeauna mulţimea minimală de instanţe de bază.

Astfel, în exemplul precedent:

$$S_1 = \{\underbrace{\{\neg B\}}_{1}, \underbrace{\{C\}}_{2}, \underbrace{\{\neg C, B\}}_{3}\}$$

este deja o mulţime nerealizabilă.

(1) ¬B
(2) C
(3) ¬C, B
(4) B din (2) şi (3)
(5) □ din (1) şi (4)

Exemplul 8.2.18: Să examinăm acum o mulţime de clauze realizabilă. Fie fraza:

φ : $(\forall x)[(\exists y)P(x, y) \rightarrow (\exists y)P(a, y)]$.

Forma normală Skolem a lui φ este $(\forall x)(\forall y)[\neg P(x, y) \vee P(a, f(x, y)]$ (De ce?).
Mulţimea de clauze corespunzătoare este:

$$S = \{\{\neg P(x, y), P(a, f(x, y)\}\}$$

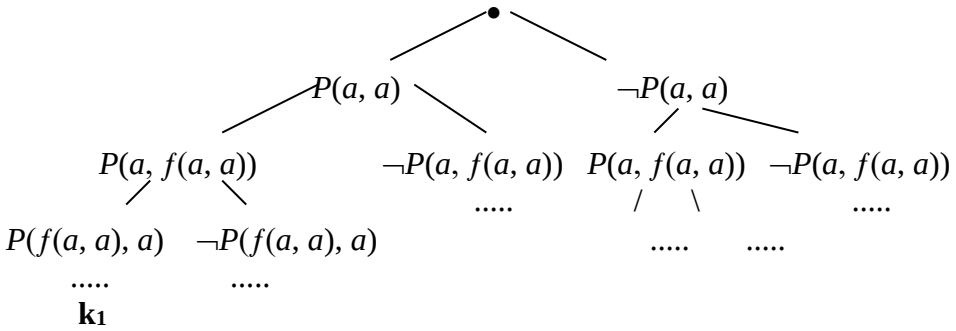
şi *universul Herbrand* corespunzător:

$$H = \{a, f(a, a), f(a, f(a, a)), f(f(a, a), a), f(f(a, a), f(a, a)),...\}$$

Instanţele de bază ale atomilor din *S* sunt:

$$\{P(a, a), P(a, f(a, a)), P(f(a, a), a), P(f(a, a), f(a, a)),...\}$$

Un arbore semantic pentru *S* este:



Acest arbore semantic infinit conţine ramuri care nu sunt contradictorii prin construcţie, deoarece conjuncţia atomilor acestor ramuri nu invalidează nici o clauză din *S*. De exemplu, ramura $\mathbf{k_1}$, care nu conţine negarea lui *P*, este necontradictorie. $\mathbf{k_1}$ oferă o interpretare Herbrand a frazei φ ce satisface *S*.

8.3 *Unificare şi rezoluţie în LPr*

Am discutat anterior despre formele normale Prenex şi Skolem ale unei fraze. Fie φ o frază a unui limbaj **LPr**. Atunci, urmând paşii:

A: Formă Prenex
B: **FNC** a frazei fără cuantificatori
C: Formă normală Skolem
D: Formă clauzală

putem reduce φ la o mulţime de clauze şi determina valoarea ei de adevăr pe baza metodelor specifice **LPr.**

Exemplul 8.3.1: Fie φ o frază în formă normală Prenex *(A)*:

$$\varphi: \quad (\forall x)(\exists y)(\exists z)((\neg P(x, y) \wedge Q(x, z) \vee R(x, y, z)) \leftrightarrow (\forall x)(\exists y)(\exists z)((\neg P(x, y) \vee R(x, y, z)) \wedge (Q(x, z)) \vee R(x, y, z))$$

Introducem acum simbolurile funcţionale Skolem f, g, unde $y = f(x)$ şi $z = g(x)$.
Atunci:

$$\varphi \leftrightarrow (\forall x)[(\neg P(x, f(x)) \vee R(x, f(x), g(x))) \wedge (Q(x, g(x)) \vee R(x, f(x), g(x)))$$

şi în final determinăm forma clauzală (D) a frazei φ:

$$S = \{\{\neg P(x, f(x)), R(x, f(x), g(x))\}, \quad \{(Q(x, g(x)), R(x, f(x), g(x))\}\}$$

În forma clauzală a lui φ trebuie să considerăm instanţieri ale variabilelor şi funcţii Skolem. Arborii semantici definiţi în secţiunea precedentă ne ajută în rezolvarea acestei probleme. Folosind arbori semantici, putem selecta simboluri pentru substituţia variabilelor din universul Herbrand corespunzător şi apoi putem aplica metoda rezoluţiei pentru a găsi contradicţii între instanţele de bază ale clauzelor implicate. O astfel de procedură este însă consumatoare de timp. În plus, nu poate fi folosită cu uşurinţă într-un mecanism deductiv structurat implementabil la nivel de program. Avem nevoie, deci, de o metodă algoritmică care poate fi uşor programată. Procedura de unificare ce va fi prezentată în continuare oferă o metodă care va facilita determinarea contradicţiilor.

A) Unificare: Descriere informală

Am definit deja (definiţia 7.2.19) conceptul de substituţie. Fie următoarele două clauze:

$C_1 : \{P(f(x), y), Q(a, b, x)\}$ şi $C_2 : \{\neg P(f(g(c)), g(d))\}$

Dorim să aplicăm rezoluţia asupra clauzelor C_1 şi C_2 substituind x cu $g(c)$ şi y cu $g(d)$.

Pentru aceasta:

(1) trebuie să verificăm dacă C_1 şi C_2 pot rezolva, şi

(2) trebuie să găsim mulţimile substituţie adecvate care permit rezoluţia.

Algoritmul de unificare permite realizarea acestor operaţii. Să vedem cum unifică C_1 şi C_2 pe baza acestui algoritm.

Pasul 1:

Identificăm în cele două clauze doi atomi de semn contrar (în exemplu, $P(f(x),y)$ din C_1 şi $\neg P(f(g(c)), g(d))$ în C_2) şi comparăm termenii lor de la stânga la dreapta, până la întâlnirea primei perechi de termeni care diferă, în cazul de faţă, este vorba de o pereche de termeni cu aceleaşi simboluri funcţionale, dar care diferă prin argumentele acestor funcţii. Construim o mulţime ce conţine aceste argumente, numită **mulţime de neconcordanţă**. Pentru C_1 şi C_2, prima mulţime de neconcordanţă este $\{x,g(c)\}$.

Pasul 2:

Pentru fiecare variabilă din *mulţimea de neconcordanţă* verificăm dacă apare într-un alt termen din aceeaşi mulţime.

Pasul 3:

Dacă testul anterior este pozitiv, atunci cele două clauze nu unifică şi algoritmul se termină cu eşec. Dacă testul este negativ, înlocuim variabila cu termenul corespunzător din mulţimea de neconcordantă. Aplicăm substiţia $\theta_1 = \{x / g(c)\}$ asupra clauzelor C_1 şi C_2, care devin astfel:

$$C_1^1 = \{P(f(g(c)), y), Q(a, b, g(c))\}$$
$$C_2^1 = C_2$$

Pasul 4:

Reluăm comparaţia termenilor din pasul l şi continuăm cu paşii 2 şi 3. Noua mulţime de neconcordanţă este $\{y, g(d)\}$. Aplicăm substituţia $\theta_2 = (y / g(d)\}$ asupra lui C_1^1 şi C_2^1 şi obţinem:

$$C_1^2 = \{P(f(g(c)),g(d)), Q(a,b,g(c))\}$$
$$C_2^2 = C_2$$

În acest moment, se poate aplica rezoluţia între C_1^2 şi C_2^2.

În final, algoritmul se termină cu producerea unei mulţimi de substituţii pe baza cărora C_1 şi C_2 sunt unificate şi rezolvă conform regulii rezoluţiei din **LP**. Această mulţime de substituţii se numeşte **unificator general**, UG, al clauzelor rezolvate. Pentru C_1 şi C_2, *unificatorul general* este $\theta = \{x / g(c), y / g(d)\}$. Dacă clauzele nu pot rezolva, algoritmul se termină în pasul 2.

B) Unificare: Descriere formală

Vom prezenta în continuare descrierea formală a algoritmului de unificare şi definiţiile asociate.

Definiţia 8.3.2: *Mulţime de neconcordanţă:*
Fie $S = \{C_1, C_2, \ldots, C_n\}$ o mulţime de clauze. Atunci mulţimea:

MN(S) = $\{t_i, t_{j1}, ..., t_{j\lambda} \mid t_i$ este primul termen din stânga ce apare într-un subtermen c_k al clauzelor din S şi termenii $t_{j1}, ..., t_{j\lambda}$ apar în subtermenii $c_{1j}, ...c_{j\lambda}$ ai clauzelor din S, astfel încât pentru substituţia $\theta_1 = \{t_i / t_{j\mu}\}$ sau substituţia $\theta_2 = \{t_{j\mu} / t_i\}$, $1 \le \mu \le \lambda$, $c_k\theta_1$ este identic cu $c_{j\mu}\theta_1$ sau $c_k\theta$ este identic cu $c_{j\mu}\theta_2\}$ se numeşte **mulţime de neconcordanţa** a lui S , pe scurt **MN.**

Mulţimea de neconcordanţă nu este definită unic ci depinde de ordinea de enumerare a clauzelor din S.

În exemplul prezentat la descrierea informală a unificării avem:

$S = \{C_1, C_2\} = \{\{P(f(x), y), Q(a, b, x)\}, \{\neg P(f(g(c)), g(d))\}\}$

Dacă substituim subtermenul x din $f(x)$ cu $g(c)$ în $P(f(x), y)$, cu alte cuvinte $\theta_1 = \{x/g(c)\}$, atunci $f(x)\theta_1$ este identic cu $f(g(c))\theta_1$. Prima mulţime de neconcordanţă este astfel $MN_1 = \{x, g(c)\}$. După aplicarea substituţiei θ_1, neconcordanţa între x şi $g(c)$ este eliminată şi următoarea neconcordanţă localizată de algoritm este dată de mulţimea $MN_2 = \{y, g(d)\}$.

Definiţia 8.3.3: Fie $C = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ o mulţime de clauze sau de termeni, o substituţie θ se numeşte unificator al mulţimii C dacă $\sigma_1\theta = \sigma_2\theta = \ldots = \sigma_n\theta$.

Definiţia 8.3.4: Un unificator θ se numeşte **cel mai general unificator**, pe scurt **MGU** (*Most General Unifier*), dacă pentru orice alt unificator ψ există o substituţie γ astfel încât $\psi = \theta\gamma$.

Pentru o mulţime de clauze sau termeni, *cel mai general unificator* este unic determinat.

Exemplu 8.3.5: Fie mulţimea de atomi $S = \{Q(g(x), w), Q(y, b)\}$, unde b este un simbol de constantă. Un unificator al lui S este:

$\psi = \{y / g(b), x / b, w / b\}$

Dacă ψeste aplicată lui S, se crează următoarele instanţe de bază:

$Q(g(x), w)\psi = Q(g(b), b)$

$Q(y, b)\psi = Q(g(b), b)$

Dacă impunem acum $\gamma = \{x / b\}$şi $\theta = \{y / g(x), w / b\}$, putem demonstra uşor că $\psi = \theta\gamma$. În plus, θ unifică atomii de mai sus. θ este deci un **MGU**, cel mai general unificator al mulţimii S.

Exemplu 8.3.6: Fie C o mulţime de termini $C = \{f(x, g(x)), f(h(y, g(h(y))))\}$.

$\psi = \{x / h(g(c)), y / g(c)\}$ este un unificator al lui C. $\theta = \{x / h(y)\}$ este, de asemenea, un unificator al lui C. Dacă impunem $\gamma = \{y / c\}$, atunci se poate uşor demonstra că $\psi = \theta\gamma$. Deci θ este *cel mai general unificator* al lui C.

Acum, putem trece la descrierea formală a algoritmului de unificare.

Algoritmul 8.3.7: Algoritmul de unificare:
Fie $T = \{P_1, P_2, ..., P_n\}$o mulţime de formule atomice.

Pasul 0:

$\theta_0 = E$ (substituţia identică)

Pasul k:

Avem deja substituţiile $\theta_0, \theta_1,...,\theta_k$.

Pasul k+1 (pas recursiv):

(i) Dacă $P_1\theta_0\theta_1...\theta_k = P_2\theta_0\theta_1...\theta_k =....= P_n\theta_0\theta_1...\theta_k$ atunci algoritmul se termină demonstrând că $\theta = \theta_0\theta_1...\theta_k$ este un unificator general.

(ii) Dacă $P_i\theta_0\theta_1...\theta_k \neq P_j\theta_0\theta_1...\theta_k$ pentru anumiţi *i* şi *j,* atunci:

(a) Formăm mulţimea de neconcordanţă:

$MN(P_1\theta_0\theta_1...\theta_k ,...., P_n\theta_0\theta_1...\theta_k) = MN(T\theta_0\theta_1...\theta_k) = MN_k$

(b) Efectuăm **verificarea apariţiei** variabilelor (**VA**), cu alte cuvinte verificăm dacă există vreo variabilă *v* într-un element din MN_k care apare şi într-un alt element din MN_k.

Dacă testul **VA** este afirmativ, (o variabilă apare în mai mult de un element din MN_k) atunci ne oprim şi concluzionăm că *T* nu este o mulţime unificabilă. În caz contrar, deci pentru un rezultat negativ al testului, pentru $v,t \in MN_k$ impunem $\theta_{k+1} = \{v / t\}$ pentru a elimina neconcordanţa între $P_1,...,P_n$ în termenii *v, t.*

Pasul k+2:

Reluăm paşii *k*+l, *k+2* pentru *k* = *k*+1.

Aşa cum se va vedea din enunţul teoremei următoare, dacă *T* este unificabilă, algoritmul se termină întotdeauna prin construcţia celui mai general unificator:

$MGU = \theta = \theta_0\theta_1...\theta_k$

Considerăm următoarea teoremă.

<u>Teorema 8.3.8</u>: (J. A. Robinson)

Dacă aplicăm algoritmul de unificare mulţimii $T = \{P_1,...,P_n\}$ *atunci:*

Dacă T este unificabil, atunci algoritmul se termină şi construieşte cel mai general unificator al mulţimii T.

Dacă T nu este unificabilă, atunci algoritmul se termină şi anunţă că nu există unificator.

Teorema precedentă spune că dacă *T* este unificabilă, atunci algoritmul de unificare se termină determinând **MGU** (şi nu doar un unificator oarecare) al lui *T.*

<u>Exemplul 8.3.9</u>: Fie mulţimea de formule:

$S = \{Q(a, x, f(g(z))), Q(z, f(y), f(y))\}$

Este S unificabilă? Dacă da, să se determine **MGU**.

Pasul 0: Impunem $\theta_0 = E$

Pasul 1: $S\theta_0 = S$

$MN(S\theta_0) = MN_1 = \{a, z\}$

VA negativ

Impunem $\theta_1 = \{z / a\}$

Pasul 2: $S\theta_0\theta_1 = \{ Q(a, x, f(g(a))), Q(a, f(y), f(y))\}$

$MN(S\theta_0\theta_1) = MN_2 = \{x / f(y)\}$
VA negativ
Impunem $\theta 2 = \{x / f(y)\}$

Pasul 3: $S\theta_0\theta_1\theta_2 = \{ Q(a, f(y), f(g(a))), Q(a, f(y), f(y))\}$
$MN(S\theta_0\theta_1\theta_2) = MN_3 = \{g(a), y\}$
VA negativ
Impunem $\theta_3 = \{y / g(a)\}$

Pasul 4: $S\theta_0\theta_1\theta_2\theta_3 = \{ Q(a, f(g(a)), f(g(a))), Q(a, f(g(a)), f(g(a)))\}=\{ \}$

S este deci unificabilă cu cel mai general unificator:

$MGU = S\theta_0\theta_1\theta_2\theta_3 = \{z / a, x / f(y), y / g(a)\}$

Exemplul 8.3.10: Fie mulţimea de formule:

$S = \{Q(y, y), Q(z, f(z))\}$

Este *S* unificabilă? Dacă da, determinaţi un unificator general.

Pasul 0: Impunem $\theta_0 = E$

Pasul 1: $S\theta_0 = S$
$MN(S\theta_0) = MN_1 = \{y, z\}$
VA negativ
Impunem $\theta 1 = \{y / z\}$

Pasul 2: $S\theta_0\theta_1 = \{Q(z, z), Q(z, f(z))\}$
$MN(S\theta_0\theta_1) = MN_2 = \{z, f(z)\}$
VA afirmativ, *z* apare în f(*z*).

Deci *S* nu este unificabilă.

Trebuie să observăm că în multe aplicaţii, pentru a obţine o eficienţă crescută, mecanismul de inferenţă PROLOG bazat pe algoritmul de unificare ignoră testul de verificare a apariţiei variabilelor. Cu alte cuvinte, substituie prima variabilă *x* cu primul termen a unei mulţimi de neconcordanţă date **MN**. Acest lucru poate conduce evident la erori şi este sarcina programatorului să creeze mecanismele de control şi securitate corespunzătoare pentru a evita apariţia unor astfel de erori în program.

Acum, avem elementele necesare pentru a descrie rezoluţia în logica predicatelor.

C) Rezoluţia în LPr

Metoda rezoluţiei în **LPr** este, de fapt, o combinaţie între unificarea **LPr** şi rezoluţia în **LP**. Astfel, la fel ca în **LP** (definiţia 5.3.17), dacă *S* este o mulţime de clauze **LPr**, atunci o demonstraţie prin rezoluţie din *S* este o secvenţă finită de clauze $C_1,...,C_n$ astfel încât pentru fiecare $C_i (1 \le i \le n)$ avem $C_i \in S$ sau $C_i \in R(\{C_j, C_k\})$ $(1 \le j,k \le i)$ unde $R(\{C_j, C_k\})$ este rezolventul clauzelor C_j şi C_k. Trebuie remarcat că, deoarece variabilele clauzelor sunt toate legate prin cuantificatorul universal, putem redenumi variabilele pentru a evita confuziile la unificare. Această procedură de redenumire se numeşte normalizarea variabilelor.

A se vedea modul de lucru al metodei rezoluţiei prin exercitiul 3 (exer. **3. Exemplul 8.3.11**), de la aplicaţii.

Observaţia 8.3.12: Utilizarea rezoluţiei este, de fapt, o aplicare şi simplificare a metodei corespunzătoare din **LPr**. Astfel, dacă fraza σ din **LPr** are o demonstraţie prin rezoluţie din mulţimea *S* în **LPr**, notată $S \vdash_R \sigma$, atunci σ este demonstrabilă din *S*, conform definiţiei 7.3.8. Formal:

$$S \vdash_R \sigma \Rightarrow S \vdash \sigma$$

Inversa acestei afirmaţii este, de asemenea, validă pentru orice mulţime de clauze *S:*

$$S \vdash \sigma \Rightarrow S \vdash_R \sigma$$

Să investigăm acum o serie de rezultate de corectitudine şi completitudine a metodelor prezentate.

Lecţia 9- Corectitudinea şi completitudinea demonstraţiilor LPr

În cele ce urmează vom discuta, la fel ca şi în cazul **LP**, rezultate referitoare la completitudinea şi corectitudinea demonstraţiilor logicii predicatelor. Demonstraţiile sunt asemănătoare cu cele din **LP** .

9.1 *Corectitudinea şi completitudinea demonstraţiilor cu tablouri*

Reamintim notaţiile $\vdash_B \sigma$ pentru o frază σ ce este demonstrabilă Beth şi $\models \sigma$ pentru o frază σ ce este logic adevărată. Vom începe prin a enunţa nişte leme şi teoreme auxiliare referitor la corectitudinea şi completitudinea demonstraţiilor Beth.

Lema 9.1.1: *Fie **R** un simbol predicativ de aritate **n** a unui limbaj $\mathcal{L}$ şi fie σ o frază din $\mathcal{L}$. Să presupunem că există o ramură necontradictorie κ într-un tablou sistematic cu $f\sigma$ în origine. Formăm o interpretare $\mathcal{A}$, al cărei univers este orice mulţime $\{a_1, a_2,...\} = A$ care stabileşte o corespondenţă de unu la unu cu simbolurile de constante din $\mathcal{L}$. Interpretarea oricărui simbol de constantă c_i este elementul $a_i=\varepsilon(c_i)$.*

Definim relaţia $\varepsilon(R) \subseteq A^n$:

$$\varepsilon(R)(a_{i1}, a_{i2},..., a_{in}) \Leftrightarrow aR(c_{i1}, c_{i2},..., c_{in})$$ *este un nod al ramurii* κ

Atunci:

(i) *dacă* $f\sigma$ *este un nod al lui* κ *, atunci* σ *este falsă în* $\mathcal{A}$.

(ii) *dacă* $a\sigma$ *este un nod al lui* κ *, atunci* σ *este adevărată în* $\mathcal{A}$.

Teorema 9.1.2: Completitudine:

Dacă σ *este o consecinţă logică a unei mulţimi de fraze S din **LPr**, atunci este, de asemenea, demonstrabilă Beth din S*:

$$S \models \sigma \Rightarrow S \vdash_B \sigma$$

Corolarul 9.1.3: *Dacă* σ *este logic adevărată, atunci este, de asemenea, demonstrabilă Beth:*

$$\models \sigma \Rightarrow \vdash_B \sigma$$

Definiţia 9.1.4: Fie κ o ramură a unui tablou şi fie $\mathcal{A}$ a interpretare a lui $\mathcal{L}$. Se spune că $\mathcal{A}$ **corespunde lui** κ dacă:

(i) $a\sigma$ este un nod al lui $\kappa \Rightarrow \mathcal{A} \models \sigma$

(ii) $f\sigma$ este un nod a lui $\kappa \Rightarrow \mathcal{A} \not\models \sigma$

Lema 9.1.5: Fie ***T** un tablou sistematic complet cu $f\sigma$ în origine, £ un limbaj şi $\mathcal{A}$ restricţia unei interpretări a lui $\mathcal{L}$ la simbolurile de constante ce apar în σ, astfel încât* $\mathcal{A} \models$

$\neg\sigma$. *Atunci, există cel puţin o ramură a lui* ***T*** *care corespunde cu o extensie a lui* A .

Teorema 9.1.6: Corectitudine:

Dacă o frază σ *este demonstrabilă Beth dintr-o mulţime de fraze S din* **LPr**, *atunci* σ *este o consecinţă a lui S*:

$$S \vdash_B \sigma \Rightarrow S \models \sigma$$

Corolarul 9.1.7: *Dacă fraza* σ *este demonstrabilă Beth, atunci ea este logic adevărată:*

$$\vdash_B \sigma \Rightarrow \models \sigma$$

Teorema 9.1.8: Completitudine:

O mulţime de fraze S este realizabilă dacă şi numai dacă orice submuhime a lui S este realizabilă.

9.2 *Corectitudinea şi completitudinea demonstraţiilor prin rezoluţie*

Teorema 9.2.1: Corectitudine: *Fie S o mulţime de clauze şi R*(S) mulţimea rezolvenţilor lui S. Dacă R*(S) conţine clauza vidă, atunci S este nerealizabilă:*

$\square \in R^*(S) \Rightarrow$ *S este nerealizabilă*

Lema 9.2.2: *Dacă* C_1' *şi* C_2' *sunt instanţe de bază ale clauzelor* C_1 *şi* C_2 *şi dacă C' este rezolventul lui* C_1' *şi* C_2' *atunci există un rezolvent C al clauzelor* C_1 *şi* C_2, *astfel încât C' este o instanţă de bază a lui C.*

Teorema 9.2.3: Completitudine: Fie S o mulţime de clauze şi R*(S) mulţimea rezolvenţilor lui S. Dacă S nu este realizabilă, atunci R*(S) conţine clauza vidă:

S este nerealizabilă $\Rightarrow \square \in R^*(S)$

Interpretarea intuitivă a acestei teoreme este:

♦ *Pentru a demonstra nerealizabilitatea unei fraze cu ajutorul rezoluţiei, trebuie să demonstrăm clauza vidă din mulţimea de clauze corespunzătoare frazei respective. Dacă dorim să demonstrăm că o clauză* φ *este o consecinţă a unei mulţimi consistente de clauze S dată, încercăm demonstrarea clauzei vide din mulţimea de clauze* $S \cup S'$, *unde S' este mulţimea de clauze corespunzătoare lui* $\neg\varphi$. *Dacă obţinem clauza vidă, atunci inconsistenţa este generată de presupunerea* $\neg\varphi$, *deci* φ *este consecinţă a lui S.*

Exemplul 9.2.4: Să considerăm mulţimea de clauze din exemplul 4.4.9. Să se dea răspunsuri analitice, folosind metoda rezoluţiei, la următoarele întrebări:

(a) "*Ce poate fura Petre*?" (b) "*Poate Petre fura de la Maria*?"

Răspuns:

Reformulăm clauzele Horn din exemplu şi întrebările *în formă mulţime-teoretică.* Considerăm astfel următoarea mulţime de clauze Horn.

C_1 :{*hoţ*(*Petre*)}

C_2 : {*place*(*Maria, mâncare*)}

C_3 : { *place*(*Maria, vin*)}

C_4 : { *place*(*Petre, bani*)}

C_5 : { *place*(*Petre, x*), ¬ *place*(*x, vin*)}

C_6 : {*poate_fura*(*x, y*), ¬*hoţ*(*x*), ¬*place*(*x, y*)}

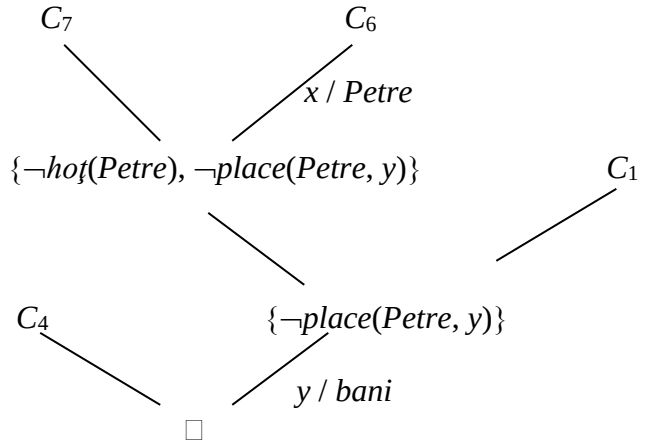
întrebările iau următoarea formă:

C_7 : {¬*poate_fura*(*Petre, y*)}

C_8 : {¬*poate_fura*(*Petre, Maria*)}

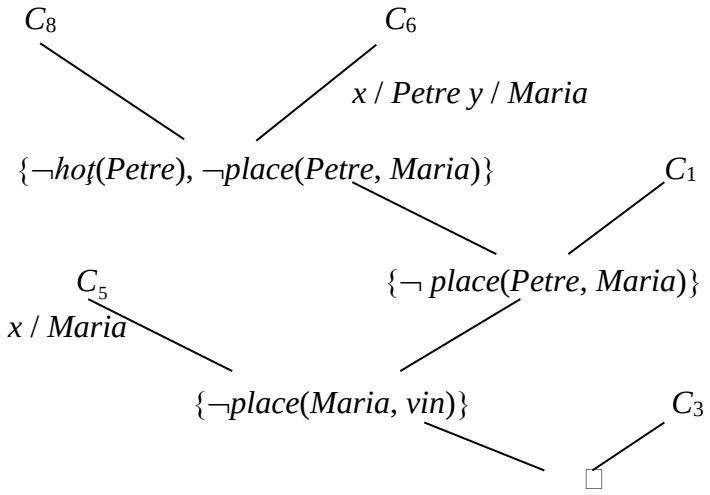
Atunci:

(a)



Cu alte cuvinte, pornind de la negarea întrebării obţinem clauza vidă, deci C_7 nu este o afirmaţie adevărată, în plus, demonstraţia de mai sus produce valoarea variabilei *y,* deci răspunsul la întrebarea (a), *y = bani*, ceea ce înseamnă că *Petre poate fura bani*.

(b)



C_8 conduce la demonstrarea clauzei vide, deci *Petre poate fura de la Maria*!

Observaţia 9.2.5: Dacă în timpul rezoluţiei, algoritmul de unificare se termină fără a produce cel mai general unificator, cu alte cuvinte nu putem demonstra □ din enunţuri, atunci se spune că scopul nostru **eşuează,** în caz contrar, se spune că scopul reuşeşte (observaţia 9.2.4). De exemplu, în exemplul 7.4.9, scopul "*poate_fura*(*Petre, Maria*)" reuşeşte.

9.3 *Completitudinea demonstraţiilor axiomatice*

Demonstraţia următoarei teoreme de completitudine a metodei demonstraţiilor axiomatice este în afara contextului acestei cărţi.

Teorema 9.3.1: Corectitudine şi completitudine, Godel, 1930:

O formulă φ *în* ***LPr*** *este derivabilă dintr-o mulţime de fraze S din* ***LPr*** *dacă şi numai dacă* φ *este o consecinţă a lui S. Formal*:

$$S \vdash \varphi \Leftrightarrow S \models \varphi$$

Corolarul 9.3.2: *O formulă* φ *în* ***LPr*** *este derivabilă din axiomele* ***LPr*** *dacă şi numai dacă* φ *este logic adevărată. Formal*:

$$\vdash \varphi \Leftrightarrow \models \varphi$$

9.4 *Metode de decizie în logică*

Multe probleme matematice se încadrează în aceeaşi schemă generală şi pot fi astfel rezolvate pe baza unei proceduri generale aplicate unei probleme specifice. De exemplu, putem răspunde la întrebarea:

"este polinomul *g(x)* divizibil prin polinomul *f(x)*?"

folosind algoritmul de împărţire, împărţind *f(x)* la *g(x).* Dacă restul împărţirii este polinomul nul, atunci răspunsul este "da", iar dacă restul este diferit de polinomul nul, atunci răspunsul este "nu".

Definiţia 9.4.1: O metodă care ne permite să răspundem cu "da" sau "nu" unei instanţe specifice a unei întrebări generale se numeşte **procedură de decizie**. Problema găsirii unei astfel de metode pentru o întrebare generală se numeşte **problema deciziei** întrebării.

Multe probleme de decizie în matematică nu pot fi rezolvate în forma lor generală sau au numai soluţii specifice, ceea ce înseamnă că sunt rezolvate numai în anumite condiţii. Problema deciziei unui sistem logic *L* este o astfel de problemă. O logică *L* este determinată de un limbaj, care constă în simbolurile logice şi cele speciale din limbaj, şi din axiomele şi regulile cu ajutorul cărora putem demonstra şi analiza fraze bine formate din logică.

Definiţia 9.4.2: Problema deciziei unui sistem logic *L* constă în găsirea unei metode algoritmice cu ajutorul căreia putem decide dacă o frază bine formată a limbajului este derivabilă în *L* sau nu, cu alte cuvinte dacă este demonstrabilă sau nu în *L*..

Problema deciziei logicii propoziţiilor **(LP)** este rezolvată folosind tabelele de adevăr: pentru o propoziţie *A* din **LP**, construim tabela de adevăr a lui *A* şi verificăm dacă *A* este o tautologie. Dacă *A* este tautologie atunci, conform corolarului 6.3.2, *A* este derivabilă în **LP**, iar dacă *A* nu este o tautologie, *A* nu este derivabilă în **LP**. În consecinţă, se poate enunţa următoarea teoremă.

Teorema 9.4.3: Problema deciziei în **LP** este rezolvabilă.

Situaţia este însă mai complicată pentru cazul logicii predicatelor **(LPr).** Conform corolarului 9.1.15, ştim că o formulă φ a unui limbaj $\mathcal{L}$ a **LPr** este derivabilă dacă şi numai dacă φ este logic adevărată, adică dacă este adevărată în toate interpretările lui $\mathcal{L}$. Dar interpretările unui limbaj $\mathcal{L}$ nu sunt numărabile şi, deci, nu pot fi toate verificate. Următoarea teoremă este cunoscută din 1936.

Teorema 9.4.4: Problema deciziei în **LPr** nu este rezolvabilă. Cu alte cuvinte, nu există o metodă algoritmică care să ne permită să decidem dacă o formulă din **LPr** este derivabilă în **LPr** sau nu.

Deşi problema deciziei **LPr** nu poate fi rezolvată în general, există soluţii specifice.

Teorema 9.4.5: Problema deciziei **LPr** poate fi rezolvată pentru formule *în formă normală Prenex,* în care nu există cuantificatori existenţiali ce preced un cuantificator universal. Deci există o procedură algoritmică care ne permite să decidem dacă o formulă de forma:

$$\underbrace{(\exists y_1)....(\exists y_\lambda)}_{numai\ \exists}\underbrace{(\forall x_1)....(\forall x_k)}_{numai\ \forall}\varphi$$

Teorema 9.4.6: Problema deciziei LPr poate fi rezolvată pentru toate formulele ce conţin exclusiv predicate de ordin mai mic sau egal cu l, adică predicate care au cel mult o variabilă.

În paralel cu eforturile de a găsi soluţii specifice problemei deciziei **LPr**, s-au desfăşurat cercetări referitor la problema deciziei realizabilităţii unei formule a **LPr**, adică găsirea unei metode algoritmice care să ne permită să determinăm dacă o formulă din **LPr** este realizabilă sau nu. Teoremele 7.6.6. (Loewenheim Skolem), 8.1.7 şi 9.1.6 precizează soluţiile specifice ale problemei deciziei realizabilităţii unei fraze din **LPr**.

Exerciţii rezolvate

1. Sa se găsească apariţiile libere de variabila in următoarele propoziţii:

a) $\forall$ (x) P(x,y) $\rightarrow$ V(z)Q(z,x) **rezultă**: x liber, y liber, z legat

b) Q(z) $\rightarrow \neg$($\forall$ x)($\forall$ y)P(x,y,a) **rezultă**:z liber; x,y legate

c) ($\forall$ x) P(x) $\wedge$ ($\forall$ y) Q(x,y) **rezultă**: x liber, y legat

2. Folosind simbolul „<”si limbajul LPr, sa se formuleze următoarele afirmaţii:

a) exista un număr mai mic decât 5 si mai mare ca 3;
b) pentru orice număr x, exista un număr y mai mic ca x;
c) pentru orice număr x, exista un număr mai mare ca x;
d) pentru orice doua numere x si y suma x+y este egala cu suma y+x;
e) pentru orice număr x exista un număr y, a.i pentru orice z pentru care, daca diferenţa z-5 este mai mica decât y, atunci diferenţa z-7 este mai mica decât 3 .

Rezolvare:

Predicate: număr(x) = x este număr;
<(x,y) = x<y
+(x,y) = x+y
- (x,y) = x-y
=(x,y) = x=y

a) $(\exists x)[numar(x) \wedge <(x,5) \wedge <(3,x)]$

b) $(\forall x)(\exists y)[numar(x) \wedge numar(y) \wedge <(y,x)]$

c) $(\forall x)(\exists y)[numar(x) \wedge numar(y) \wedge <(x,y)]$

d) $(\forall x)(\forall y)[numar(x) \wedge numar(y) \Rightarrow =(+(x,y),(+y,z))]$

e) $(\forall x)(\exists y)(\forall z)[numar(x) \wedge numar(y) \wedge numar(z) \wedge (-(z,5),y) \rightarrow <(-(z,7),3)]$

3. Fie următoarele clauze:

$C_1 = \{\neg P(x, y), \neg P(y, z), P(x, z)\}$

$C_2 = \{\neg P(u, v), P(v, u)\}$

Dorim să concluzionăm:

$C_3 = \{\neg P(x, y), \neg P(z, y), P(x, z)\}$

Notaţiile corespunzătoare clauzelor C_1, C_2 şi C_3 în contextul **LPr** sunt:

$(\forall x)(\forall y)(\forall z)[P(x, y) \wedge P(y, z) \rightarrow P(x, z)]$ pentru C_1

$(\forall u)(\forall v)[P(u, v) \rightarrow P(v, u)]$ pentru C_2

$(\forall x)(\forall y)(\forall z)[P(x, y) \wedge P(z, y) \rightarrow P(x, z)]$ pentru C_3

Reyolvare.

Metoda I:

Lucrăm direct în contextul LPr:

C_1, C_2 şi C_3 sunt echivalente cu:

$(\forall x)(\forall y)(\forall z)[\neg P(x, y) \vee \neg P(y, z) \vee P(x, z)]$ (1)

$(\forall u)(\forall v)[\neg P(u, v) \vee P(v, u)]$ (2)

$(\forall x)(\forall y)(\forall z)[\neg P(x, y) \vee \neg P(z, y) \vee P(x, z)]$ (3)

Redenumim variabilele din (2):

$(\forall x)(\forall z)[\neg P(x, z) \vee P(z, x)]$ (4)

şi, conform teoremei 4.3.6, (4) devine:

$(\forall x)(\forall y)(\forall z)[\neg P(x, z) \vee P(z, x)]$ (5)

Conjuncţia formulelor (1) şi (5), conform teoremei 4.3.6, este:

$(\forall x)(\forall y)(\forall z)[(\neg P(x, z) \vee P(z, x)) \wedge (\neg P(x, y) \vee \neg P(y, z) \vee P(x, z))]$ (6)

Şi pe baza formulei

$(\neg A \vee B) \wedge (A \vee C) \rightarrow (B \vee C)$ (*)

care este derivabilă în **LPr** (de ce?), (6) ia forma:

$(\forall x)(\forall y)(\forall z)[(P(z, x)) \vee \neg P(x, y) \vee \neg P(y, z)]$ (7)

Redenumim variabilele din (2)

$(\forall z)(\forall x)[\neg P(z, x) \vee P(x, z)]$ (8)

şi, conform 4.3.6, (8) devine:

$(\forall z)(\forall y)(\forall x)[\neg P(z, x) \vee P(x, z)]$ (9)

Conjuncţia formulelor (9) şi (7) este, pe baza (*):

$(\forall x)(\forall y)(\forall z)[\neg P(x, y) \vee \neg P(y, z) \vee P(x, z)]$ (10)

Redenumim variabilele din (2)

$(\forall z)(\forall y)[\neg P(z, y) \vee P(y, z)]$ (11)

Conform teoremei 4.3.6, (l1) devine:

$(\forall x)(\forall y)(\forall z)[\neg P(z, y) \vee P(y, z)]$ (12)

Conjuncţia lui (10) şi (12), conform (*), este:

$(\forall x)(\forall y)(\forall z)[\neg P(x, y) \vee \neg P(z, y) \vee P(x, z)]$

care este formula (3) pe care o căutăm.

Metoda a II-a:

Construim demonstraţia lui C_3 prin rezoluţie.

(1) C_1

(2) C_2

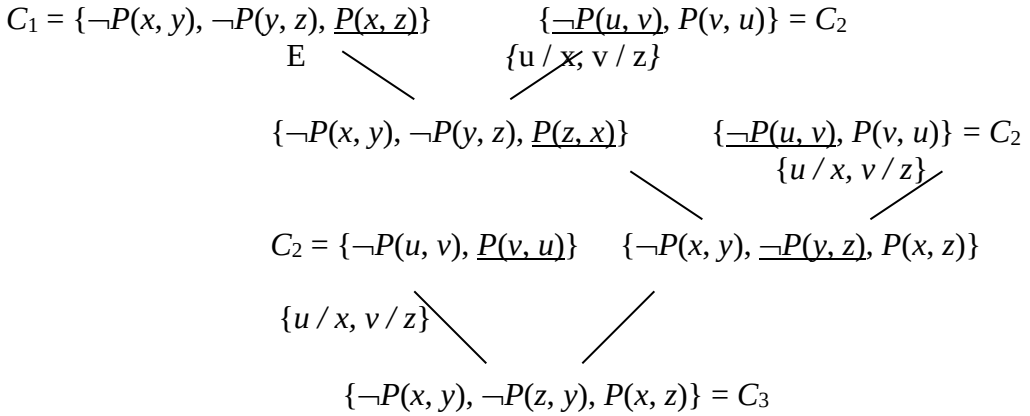
(3) $\{\neg P(x, z), P(z, x)\}$ din (2), prin $\{u / x, v / z\}$

(4) $\{\neg P(x, y), \neg P(y, z), P(z, x)\}$ din (1) şi (3), prin rezoluţie
(5) $\{\neg P(z, x), P(x, z)\}$ din (2), prin $\{u / z, v / x\}$
(6) $\{\neg P(x, y), \neg P(y, z), P(x, z)\}$ din (4) şi (5), prin rezoluţie
(7) $\{\neg P(z, y), P(y, z)\}$ din (2), prin $\{u / z, v / y\}$
(8) C_3 din (6) şi (7) prin rezoluţie

Metoda a II-a oferă o procedură mecanică de demonstrare, care este, în mod clar, mai rapidă şi mai eficientă.

Metoda a III-a:

Demonstraţia poate fi construită şi pe baza unui arbore inversat, cu rădăcina C_3. Clauzele selectate pentru rezoluţie apar în aceleaşi linii cu variabilele lor normalizate. Unificările aplicate apar pe ramurile arborelui şi atomii corespunzători care rezolvă sunt subliniaţi.



Metoda de demonstrare bazată pe arbore, exemplificată pentru demonstrarea clauzei C_3, este folosită şi în limbajul PROLOG.

Dinamismul limbajului PROLOG şi, mai general, eficienţa programării logice în domeniul programării simbolice sunt evidente.

TESTE DE AUTOEVALUARE ŞI TEME DE CONTROL

Testul nr. 1

1. Sa se determine aplicaţiile libere ale variabilelor din următoarele propoziţii. Care din propoziţii sunt fraze?

a) $(\forall x),(\forall y),(\forall z)[x > y \wedge z > z] \rightarrow (\exists w)[w > w]$

b) $(\exists x)(este\_rosu(x)) \vee (\forall y)[este\_albastru(y) \vee este\_galben(x)]$

c) x+x=x+x

d) $(\exists y)[x + x = x + x]$

e) $(\exists x)(\exists y)[este\_profesor(x, y) \wedge invata(x, y, z)]$.

2. Sa se determine forma mulţime teoretica a propoziţiilor:

a) Lui Ion ii place mâncare;
b) Merele sunt mâncare;
c) Pasările sunt mâncare;
d) Orice poate fi mâncat, fara a omora pe cineva, este mâncare;
e) Barbu mananca si este viu;
f) Maria mananca tot ceea ce mananca Barbu;

Indicaţie:
Predicate: mâncare(x) $\rightarrow$ X este mâncare
place(x,y) $\rightarrow$ lui x ii place mâncarea y;
este viu(x) $\rightarrow$ x este viu;
mananca(x,y) $\rightarrow$ x mananca felul de mâncare y;

Testul nr. 2

1. Sa se determine care din următoarele expresii sunt termeni sau formule:

a) Nicu
b) Matematician(x)
c) Număr(6)
d) Este planeta(x)
e) (3H)+10-termen
f) ($\forall$ x) [număr(x) $\wedge$ x=x+x]
g) =[+(x,y),z]+
h) $(x+y)+j^2$
i) uraste(x,y) $\wedge$ iubeşte(x,y)

2. Sa presupunem ca dragonii exista si ca tocmai am capturat un dragon mare. Sa se formuleze următoarele afirmaţii din limbajul cotidian folosind clauze Horn.

g) orice dragon care traieste la o gradina zoologica nu este fericit;
h) orice animal care intalneste oameni politicoşi este fericit;
i) oamenii care vizitează gradina zoologica sunt politicoşi;
j) animalele care trăiesc la gradina zoologica întâlnesc oamenii care o vizitează;

Indicaţie:

Predicate: traieste(x,y) $\rightarrow$ x traieste in locul y ; fericit(x); politicos(x); intalneste(x,y); dragon(x); animal(x); vizitează(x,y) – x vizitează locul y; persoana(x)

Temă de control

1. Fie L={a,b,P} un limbaj din LPr si A o interpretare a lui L cu univers D={a,b}a.i (a,a) si $(b,b) \in \varepsilon(P) \subset D \times D$, in timp ce (a,b) si (b,a) $\notin \varepsilon(p)$. Sa se determina daca urmatoarele formule sunt adevărate in A:

a) $(\forall x)(\exists y)P(x,y)$

b) $(\exists x)(\forall y)P(x,y)$

c) $(\forall x)(\exists y[P(x,y) \rightarrow P(y,x)]$

d) $(\forall x)(\forall y)P(x,y)$

e) $(\exists y)\neg P(a,y)$

f) $(\forall x)P(x,x)$

Indicaţie: Alcătuiţi o tabela de adevăr a universului interpretării:

P(a,a)	P(a,b)	P(b,a)	P(b,b)
a	f	f	a

şi deduceţi cerinţele.

2. Fie fraza $\sigma : (\exists x)P(x) \rightarrow (\forall x)P(x)$.

a) Sa se demonstreze ca σ este întotdeauna adevărat in interpretări in care x are un unic element.

b) Sa se găsească o interpretare cu un univers ce conţine doua elemente in care σ nu e adevărata.

3. Fie L={ Δ,c_1,c_2, …., c_9} un limbaj unde Δ este simbolul predicativ de aritate, c_1-c_9 sunt simboluri de constante si fie pentru L interpretarea: A={/, 1,2,…,9}, /-divizibilitatea, $\varepsilon(\Delta) = 1$, $\varepsilon(c_i) = i$, i = 1,…9. Care din următoarele fraze sunt adevărate in aceasta interpretare:

a) $(\forall y)\Delta(c_1, y) \rightarrow a$

b) $(\forall x)[\Delta(x, c_5) \leftrightarrow (x = c_1) \vee (x = c_5)] \rightarrow a$

c) $(\exists x)(\forall y)\Delta(x, y) \rightarrow a$

d)($(\exists x)(\forall y)\Delta(y, x) \rightarrow a$

BIBLIOGRAFIE RECOMANDATĂ LA UNITATEA DE ÎNVĂŢARE NR.3

18. G. Metakides, A. Nerode – *Principii de logică şi programare logică*, Editura Tehnică, Bucureşti, 1998
19. G. Georgescu – *Elemente de logică matematică*, Editura Academiei Tehnice Militare, Bucureşti, 1978
20. D. Busneag, D. Piciu, Probleme de logica si teoria multimilor, Craiova, 2003.
21. G. Georgescu, A. Iorgulescu, Logica matematica, Ed. ASE, Bucuresti, 2010
22. Gr. C Moisil, Elemente de logica matematica si de teoria multimilor, Ed. Stiintifica, Bucuresti, 1968
23. J.D. Monk, Mathematical Logic, Springer Verlag, 1976
24. V. E. Cazanescu, Curs de bazele informaticii, Tipografia Universitatii din Bucuresti, 1976
25. S. Rudeanu, Curs de bazele informaticii, Tipografia Universitatii din Bucuresti, 1982
26. M. Huth, M. Ryan, Logic in Computer Science: Modelling and Reasoning about Systems, Cambridge Univ. Press, 2009
27. A.R. Bradley, Z. Manna, The Calculus of Computation Decision Procedures with Applications to Verification, Springer,2007
28. M. Ben-Ari, Mathematical Logic For Computer Science, Springer, 2003

UNITATEA DE ÎNVĂŢARE NR.4
MAŞINI TURING ŞI ALGORITMI MARKOV

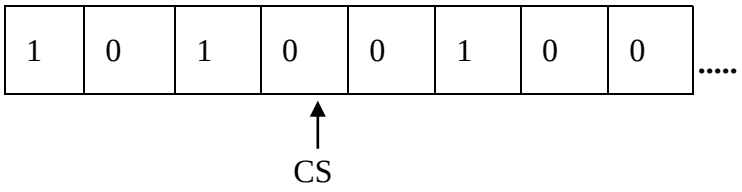
Lecţia 10 -Maşini Turing

Maşinile Turing reprezintă o altă teorie matematică care formalizează noţiunea de algoritm. Principiul filosofic care fundamentează această teorie este teza lui Turing prin care se identifică funcţiile efectiv calculabile (concept intuitiv) cu funcţiile calculabile Turing (concept matematic). Maşinile Turing, definite de Turing în 1936, s-au dovedit a fi echivalente cu funcţiile recursive. În timp ce tehnica funcţiilor recursive este mai simplă, maşinile Turing reprezintă varianta matematică a noţiunii de algoritm cea mai apropiată de calculatoarele electronice.
Acest capitol va studia maşinile Turing şi funcţiile calculabile Turing. Se dau numeroase exemple de calcule efectuate cu ajutorul maşinilor Turing.

10.1. *Maşini Turing. Funcţii calculabile Turing*

Vom începe cu o descriere intuitivă a maşinii Turing, ceea ce va sugera şi definiţia matematică. Considerăm o bandă semiinfinită B şi un dispozitiv automat CS(citeşte/scrie) care are un număr finit de stări. Banda B este împărţită în celule, iar automatul CS se mişcă faţă de banda B cu câte un pas (corespunzător unei celule) înainte şi înapoi. Vom interzice deplasarea lui CS spre stînga, atunci când se află în dreptul primei celule. Dispozitivul CS se va numi cap de citire-scriere.
Vom arăta acum modul de funcţionare al maşinii Turing. Pe bandă se pot înregistra diferite caractere (pentru noi va fi suficient 0 şi 1), care pot fi şi şterse. Capul de citire-scriere CS citeşte ce este scris pe celula din dreptul său, după care poate şterge şi scrie alte caractere sau poate să lase ceea ce a fost. În cazul nostru sunt posibile patru situaţii:
- a fost **0** şi a rămas **0**;
- a fost **0** şi a apărut **1**;
- a fost **l** şi a apărut **0**;
- a fost **l** şi a rămas **1**.

A doua etapă este mutarea capului de citire - scriere CS la stînga, la dreapta sau rămînerea lui pe loc. A treia etapă este trecerea maşinii în altă stare sau rămînerea în aceeaşi stare. Aceste trei etape formează un pas de calcul.



Reamintim că N^+ = (l, 2, 3,...}; dacă poziţiile celulelor vor fi numerotate cu l,2,3,..., atunci ceea ce este scris pe bandă poate fi asimilat cu un şir de caractere a_1, a_2, a_3..., în care a_k reprezintă ceea ce este scris în celula numerotată cu **k**. In cazul nostru a_k, este 0 sau 1.
Aceasta sugerează următoarea definiţie:

Definiţia 10.1.1: O bandă este un şir a_1, a_2, a_3..... în care fiecare a_k, este 0 sau 1. O poziţie a benzii este formată dintr-o pereche (j, k) cu $j, k \in N^+$ şi dintr-o bandă

(j, k): a_1 a_2 a_3,.....,a_j…

j reprezintă poziţia capului de citire-scriere, iar **k** este starea pentru această poziţie a benzii.

Exemplu10.1.2: Considerăm poziţia **t** a benzii:

(3, 2): 1 1 1 1 0 1 0 .....

Semnul de citire-scriere se află în dreptul celulei a treia, iar starea este 2.

Definiţia 10.1.3: O maşină Turing este un triplet ordonat de funcţii (d, p, s) care au acelaşi domeniu de definiţie D, unde D este o mulţime finită de perechi de forma (i, k) cu $i \in \{0,1\}$ şi $k \in N^+$ iar cele trei funcţii sunt precizate astfel:

$d : D \to \{0, 1\}$

$p : D \to \{-1, 0, 1\}$

$s : D \to N^+$

Dacă M = (d, p, s), atunci domeniul DomM al maşinii Turing M este D.

Observaţia 10.1.4: Cele trei funcţii **d**, **p**, **s** ce intră în componenţa unei maşini Turing M corespund celor trei etape ale unui pas de calcul ce le-am pus în evidenţă în descrierea intuitivă a unei maşini Turing.

Funcţia **d** ne dă operaţia de „citire-scriere". Dacă $(i, k) \in D$, şi $d(i, k) = \varepsilon \in \{0,1\}$, atunci această relaţie se traduce astfel în cuvinte: presupunând că, capul de citire-scriere se află pe poziţia **i** şi ne aflăm în starea **k**, atunci:

- în locul lui a_k apare 0, dacă $\varepsilon = 0$
- în locul lui a_k apare l, dacă $\varepsilon = 1$.

Funcţia **p** descrie modul cum se mută capul de citire-scriere. Dacă $p(i, k) = \mu$ atunci:

- CS se mută la celula din stînga, dacă $\mu = -1$.
- CS rămâne pe loc, dacă $\mu = 0$.
- CS se mută la celula din dreapta, dacă $\mu = 1$.

Funcţia **s** ne va da starea în care trece CS: dacă $s(i, k) = \boldsymbol{l} \in N^+$ atunci CS va trece în starea ***l***.

Observatia 10.1.5: Există numeroase alte definiţii ale maşinilor Turing (nu neapărat echivalente), dar care răspund aceleaşi probleme: aceea de-a da un concept matematic pentru funcţiile efectiv calculabile.

Exemplul 10.1.6: Considerăm maşina Turing M = (d, p, s) definită astfel:

$D = \{ (1, 2) \}$

$d(1, 2) = 0$

$p(1, 2) = -1$

$s(1, 2) = 4$

Dacă avem o poziţie a benzii **t**:

(3, 2): 1 1 1 1 0 1 0 .....

Atunci dacă aplicăm o dată maşina vom obţine o nouă poziţie a benzii:

(2, 4): 1 1 0 1 0 ...

Prin aplicarea lui **d** pe poziţia 3 a apărut 0 în loc de l; acţiunea lui **p** a fost mutarea capului de citire-scriere cu o celulă la stînga. În fine, noua stare este 4.

Definiţia 10.1.7: Fie **t** o poziţie a benzii dată de:
(j, k): a_1, a_2, a_3.....,a_j…

Poziţia următoare a benzii M(t) este definită de:
(j+ p(a_j, k), s(a_j, k)): a_1, a_2,..... a_{j-1} d(a_j, k) a_{j+1} a_{j+2}….
cu condiţia ca (a_j, k)∈DomM şi j + p(a_j,k) > 0.

Un calcul parţial al lui M este un şir t_1, t_2,....,t_m de poziţii ale benzii, astfel încât t_{i+1} = M(t_i) pentru orice i = l, 2,...,m-l. Un calcul al lui M este un calcul parţial t_1,...,t_m pentru care M(t_m) nu este definit. Dacă t_1, t_2,....,t_m este un calcul, atunci t_1 este intrarea şi t_m este ieşirea. Când t_1, t_2,....,t_r este un calcul parţial, atunci vom scrie M′(t_1) pentru t_r.
Vom găsi un mod convenabil de a reprezenta o maşină Turing. Spre exemplu, tabelul următor:

M:		0	1
	1	1 L 2	0 R 1
	2	1 0 2	

reprezintă maşina Turing M = (d, p, s) definită astfel:
Dom M = { (0, l), (1, 1), (0, 2) }
d(0, l) = l, p(0, l) = - l, s(0, l) = 2,
d(1, 1) = 0, p(1, 1) = l s(1, 1) = l,
d(0, 2) = l, p(0, 2) = 0, s(0, 2) = 2.
Observaţia 10.1.8: Vom folosi notaţii R, 0, L în loc de l, 0, -l pentru a indica modul cum se mută capul de citire-scriere(R este prescurtarea de la „right", iar L de la „left").
În general, o maşină Turing va fi scrisă sub forma unui tablou pe două coloane:

M:		0	1
	j_1		
	j_2		
	.		
	.		
	j_i		
	j_m		

În stânga sunt trecute stările j_1, j_2,...,j_m. O serie de locuri ale tabloului sunt completate, altele nu. Dacă, de exemplu, pe linia **i**, coloana întâi apare $\alpha\beta\gamma$, aceasta semnifică următoarele:
(0, j_i)∈DomM, d(0, j_i) = α, p(0, j_i) = β, s(0, j_i) = γ.
Presupunând că pe linia **i**, coloana a doua nu este scris nimic, perechea (l, j_i) nu aparţine lui DomM.
Reluăm exemplul de mai sus. Considerăm următoarea poziţie **t** a benzii:

(2, 1): 0 1 0 0 0 0.....

Atunci M(t) va fi:

(3, 1): 0 0 0 0 0 0.....

conform modului cum funcţionează maşina M. Este clar că următorul şir de poziţii ale benzii este un calcul al lui M:

(2, 1): 0 1 0 0 0 0.....

(3, 1): 0 0 0 0 0 0.....

(2, 2): 0 0 1 1 0 0.....

(2, 2): 0 1 1 1 0 0.....

Prima poziţie a benzii este intrarea, iar ultima poziţie a benzii este ieşirea.
Pentru a avea o scriere prescurtată a unei benzi vom nota:
$\underbrace{0\,0...0}_{n-ori}$ cu 0^n ; $\underbrace{1\,1...1}_{n-ori}$ cu 1^n

De exemplu, banda:
0 1 1 1 0 0 1 1 1 1 0 .....
se va scrie:
0 1^3 0^2 1^4
suprimând ultimii 0 care apar.

Observaţia 10.1.9: Vom conveni ca să apară întotdeauna un număr finit de **l** pe o bandă.
Vom scrie $0^m(n_1,..., n_k)$ în loc de 0^m 0 l^{n_1} 0 l^{n_2} 0...0 l^{nk}, unde $m \in N$. Dacă m = 0 vom scrie aceasta cu $(n_1,...,n_k)$ şi dacă k = l vom scrie n_1 în loc de (n_1).
Banda $(n_1,...,n_k)$ va fi scrisă uneori $\overline{n}$.
Exemplu: banda 0 l^3 0 l^4 se prescurtează (3,4)

Definiţia 10.1.10: Fie M o maşină Turing şi $g : (N^+)^k \to N^*$ o funcţie cu proprietatea că pentru orice $(n_1,...,n_k) \in (N^+)^k$ există un calcul (în raport cu M) care are intrarea:
(2, l): 0 $\underbrace{1....1}_{n_1-ori}$ 0 $\underbrace{1....1}_{n_2-ori}$ 0 ... 0 $\underbrace{1....1}_{nk-ori}$ 0 0 ...
şi ieşirea de foma:
(k, ℓ): $\underbrace{0.....0}_{(k-1)-ori}$ $\underbrace{1.....1}_{t-ori}$ 0 0 .....
unde $t = g(n_1,...,n_k)$. Spunem în acest caz că funcţia **g** este calculabilă Turing şi că M calculează pe g.

Observaţia 10.1.11: Conform scrierii prescurtate adoptate, intrarea de mai sus are forma (2, 1): $(n_1,.....n_k)$, iar ieşirea are forma $0^{k-1}\,1^{g(n_1,.....,n_b)}$. Este vizibil deci cum $(n_1,...,n_k) \in (N^+)^k$ apare scris pe bandă la intrare şi cum valoarea $g(n_1,...,n_k)$ a funcţiei în acest punct apare pe bandă la ieşire în urma efectuării calculului cu maşina Turing.
Vom arăta acum că funcţia f(n) = 2, pentru orice $n \in N^+$, este calculabilă Turing cu maşina Turing.

	0	1
1	1 L 2	0 R 1

	2	1 0 2	

Într-adevăr, avem următorul calcul cu M:

(2, 1): 0 1^n 0 0 ....
(3, 1): 0 0 1^{n-1} 0 0...
(4, 2): 0 0 1^{n-2} 0 0...
……
(n+1, 2): 0^n 1 1 0 0 0

Ca să înţelegem mai bine acest exemplu, vom da în întregime calculul pentru n =3

(2, 1): 0 1 1 1 0 0 0....
(3, 1): 0 0 1 1 0 0 0....
(4, 1): 0 0 0 1 0 0 0....
(5, 1): 0 0 0 0 0 0 0....
(4, 2): 0 0 0 0 1 0 0....
(4, 2): 0 0 0 1 1 0 0....

Exerciţiu: Să se găsească maşina Turing cu ajutorul căreia putem calcula funcţia constantă f (n) = 4.

Propoziţia 10.1.12: Următoarele funcţii sunt calculabile Turing:

(i) funcţia sumă: Sum (m, n) = m + n.
(ii) proiectiile pr_n^i $(m_1,.....m_n) = m_i$.
(iii) funcţia constantă: $Ct_k^d(n_1,…..n_k) = d$.
(iv) funcţia predecesor :

$$\text{Pred (m)} = \begin{cases} m-1 & dacă \quad m \geq 2 \\ 1 & dacă \quad m = 1 \end{cases}$$

Observaţia 10.1.13 Toate funcţiile de mai sus le considerăm definite numai pentru argumente din N^+ .

Demonstraţie: (i) Considerăm, maşina Turing:

	0	1
1		0 R 2
2	1 L 3	1 R 2
3	0 R 4	1 L 3

Următorul şir este un calcul al lui **m + n** cu ajutorul acestei maşini Turing:

(2, 1): 0 1^m 0 1^n
(3, 2): 0 0 1^{m-1} 0 1^n
.....
(m+2, 2): 0 0 1^{m-1} 0 1^n
(m+1, 3): 0 0 1^{m-1} 1 1^n
.....

(2, 3): 0 0 1^{m+n}
(3, 4): 0 0 1^{m+n}

Este instructiv să dăm acest calcul pentru m = 2, n = 3.

(2, 1): 0 1 1 0 1 1 1 0 0....
(3, 2): 0 0 1 0 1 1 1 0 0....
(4, 2): 0 0 1 0 1 1 1 0 0....
(3, 3): 0 0 1 1 1 1 1 0 0....
(2, 3): 0 0 1 1 1 1 1 0 0....
(3, 4): 0 0 1 1 1 1 1 0 0....

(ii). Vom demonstra numai în cazul n = 4, i = 3. Considerăm maşina Turing:

	0	1
1	0 R 2	0 R 1
2	0 R 3	0 R 2
3	0 R 4	1 R 3
4	0 L 5	0 R 4
5	0 L 5	1 L 6
6	0 R 7	1 L 6

Vom scrie calculul pentru (2, 1, 3, 1):

(2, 1): 0 1 1 0 1 0 1 1 1 0 1 0 0 0...
(3, 1): 0 0 1 0 1 0 1 1 1 0 1 0 0 0...
(4, 1): 0 0 0 0 1 0 1 1 1 0 1 0 0 0...
(5, 2): 0 0 0 0 1 0 1 1 1 0 1 0 0 0...
(6, 2): 0 0 0 0 0 0 1 1 1 0 1 0 0 0...
(7, 3): 0 0 0 0 0 0 1 1 1 0 1 0 0 0...
(8, 3): 0 0 0 0 0 0 1 1 1 0 1 0 0 0...
(9, 3): 0 0 0 0 0 0 1 1 1 0 1 0 0 0...
(10, 3): 0 0 0 0 0 0 1 1 1 0 1 0 0 0...
(11, 4): 0 0 0 0 0 0 1 1 1 0 1 0 0 0...

(12, 4): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(11, 5): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(10, 5): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(9, 5): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(8, 6): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(7, 6): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(6, 6): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(7, 7): 0 0 0 0 0 0 1 1 1 0 0 0 0 0...

(iii). Pentru d = 2 , vom lua următoarea maşină Turing:

	0	1
1	0 R 2	0 R 1
2	1 L 3	0 R 1
3	1 0 3	

Vom scrie calculul valorii lui Ct_3^2 în punctul (2, 1, 1):

(2, 1): 0 1 1 0 1 0 1 0 0.....

(3, 1): 0 0 1 0 1 0 1 0 0.....

(4, 1): 0 0 0 0 1 0 1 0 0.....

(5, 2): 0 0 0 0 1 0 1 0 0.....

(6, 1): 0 0 0 0 0 0 1 0 0.....

(7, 2): 0 0 0 0 0 0 1 0 0.....

(8, 1): 0 0 0 0 0 0 0 0 0.....

(9, 2): 0 0 0 0 0 0 0 0 0.....

(8, 3): 0 0 0 0 0 0 0 0 1.....

(8, 3): 0 0 0 0 0 0 0 1 1.....

iv). Se poate verifica uşor că Pred. se calculează cu următoarea maşină Turing:

	0	1
1		0 R 2

2	1 0 3	

Cu aceasta, demonstraţia este încheiată.

Observaţi 10.1.14: Două maşini Turing diferite pot calcula aceiaşi funcţie. Se poate verifica uşor că următoarea maşină Turing:

	0	1
1		1 0 2

poate calcula funcţia identică f(x) = x pentru orice $x \in N^+$. Aceeaşi funcţie poate fi calculată cu maşina Turing folosită pentru calculul sumei a două numere:

(2, 1): 0 1^n 0 0.....
(3, 2): 0 0 1^{n-1} 0 0.....
.....
(n+2, 2): 0 0 1^{n-1} 0 0...
(n+1, 3): 0 0 1^{n-1} 1 0...
.....
(2, 3): 0 0 1^n 0 0.....
(3, 4): 0 0 1^n 0 0.....

Observaţia 10.1.15: O maşină Turing nu poate să calculeze o funcţie $f(x_1,.....,x_n)$ pentru unul din următoarele motive:

(a) Există o intrare **t** astfel încât $M^r(t)$ există pentru orice $r \in N^+$, deci calculul parţial nu se termină niciodată. Maşina următoare ne oferă un asemenea exemplu:

	0	1
1	0 R 1	1 R 1

(b) Există un calcul cu intrarea **t**, dar ieşirea nu verifică condiţiile Definiţiei 4, cum rezultă din exemplul următor:

	0	1
1		1 L 1

Definiţia 10.1.16: Fie $X \subseteq (N^+)^k$. Vom nota cu $R_X : (N^+)^k \to \{1, 2\}$ funcţia definită astfel:

$$R_X(n_1,....., n_k) = \begin{cases} 1, & dacă \quad (n_1,..., n_k) \in X \\ 2, & dacă \quad (n_1,..., n_k) \notin X. \end{cases}$$

Vom spune că relaţia *X* este calculabilă Turing dacă R_X este calculabilă Turing.

Observaţia10.1.17: Nu am putut folosi funcţia caracteristică pentru că am eliminat pe **0** ca argument şi ca valoare a funcţiilor calculabile Turing.

Exemplul 10.1.18: Fie *X* = mulţimea numerelor impare. Atunci:

$$R_X(n) = \begin{cases} 1, & dacă \quad n\ este\ impar \\ 2, & dacă \quad n\ este\ par \end{cases}$$

Considerăm maşina Turing următoare:

	0	1
1	1 L 2	0 R 2
2	1 0 3	0 R 1

Această maşină Turing poate calcula funcţia R_X . Vom scrie calculul pentru n = 3 şi n = 2.
Pentru n = 3:

(2, 1): 0 1 1 1 0 0 0....

(3, 2): 0 0 1 1 0 0 0....

(4, 1): 0 0 0 1 0 0 0....

(5, 2): 0 0 0 0 0 0 0....

(5, 3): 0 0 0 0 1 0 0....

Pentru n = 2:

(2, 1): 0 1 1 0 0 0 0....

(3, 2): 0 0 1 0 0 0 0....

(4, 1): 0 0 0 0 0 0 0....

(3, 2): 0 0 0 1 0 0 0....

(3, 3): 0 0 1 1 0 0 0....

Deci *X* este o mulţime Calculabilă Turing.

Exemplul 10.1.19: Fie $X = \{m \in N^+ \mid m \geq 2\}$. Atunci:

$$R_X(m) = \begin{cases} 1, & dacă \quad m \geq 2 \\ 2, & dacă \quad m = 1 \end{cases}$$

R_X este calculabilă Turing cu ajutorul următoare maşini Turing:

	0	1
1		1 R 2
2	1 L 5	0 R 3
3	0 L 4	0 R 3
4	0 L 4	1 0 5

Vom scrie calculul pentru m = 3:

(2, 1): 0 1 1 1 0 0 0....

(3, 2): 0 1 1 1 0 0 0....

(4,3): 0 1 0 1 0 0 0....

(5, 3): 0 1 0 0 0 0 0....

(4, 4): 0 1 0 0 0 0 0....

(3, 4): 0 1 0 0 0 0 0....

(2, 4): 0 1 0 0 0 0 0....

(2, 5): 0 1 0 0 0 0 0....

Calculul pentru m = 1 este următorul:

(2, 1): 0 1 0 0….

(3, 2): 0 1 0 0….

(2, 5): 0 1 1 0….

Deci X este o mulţime calculabilă Turing.

Exemplu 10.1.20: Fie *P* relaţia " ≤ ", adică *P* = {(m, n) | m ≤ n}. Atunci:

$$R_P(m, n) = \begin{cases} 1, & dacă \quad m \le n \\ 2, & dacă \quad m > n. \end{cases}$$

Considerăm maşina Turing următoare:

	0	1
1		0 R 2
2	0 R 5	1 R 3
3	0 R 4	1 R 3
4	0 L 6	1 R 4
5	1 0 11	0 R 5
6		0 L 7
7	0 L 10	1 L 8
8	0 L 9	1 L 8
9	0 R 1	1 L 9
10	1 L 5	0 L 10

Cu această maşină Turing se poate calcula R_P. Vom exemplifica în cazul $R_P(1,2)$=l.

(2, 1): 0 1 0 1 1 0 0....

(3, 2): 0 0 0 1 1 0 0....

(4, 5): 0 0 0 1 1 0 0....

(5, 5): 0 0 0 0 1 0 0....

(6, 5): 0 0 0 0 0 0 0....

(6, 11): 0 0 0 0 0 1 0....

Propoziţia 10.1.21: Orice funcţie calculabilă Turing este efectiv calculabilă.

Demonstraţie. Fie $f : (N^+)^k \to N^+$ o funcţie calculabilă Turing şi $(n_1,...,n_k) \in (N^+)^k$. $(n_1,...,n_k)$ se scrie pe bandă sub forma:

(2, 1): $0\ \underbrace{1....1}_{n_1-ori}\ 0\ \underbrace{1....1}_{n2-ori}\ 0 \ldots 0\ \underbrace{1....1}_{nk-ori}\ 0 \ldots$

Conform definiţiei calculabilităţii Turing există o maşină Turing M şi un calcul în raport cu această maşină care începe cu intrarea de mai sus şi care are ieşirea:

(k, n) : $0...0\ \underbrace{1....1}_{m-ori}\ 0 \ldots$

unde $m = f(n_1,...,n_k)$, pentru orice $(n_1,...,n_k) \in (N^+)^k$. După un număr finit de paşi putem să determinăm pe $f(n_1,...,n_k)$. Cu alte cuvinte f este efectiv calculabilă.

Reciproca acestei propoziţii este următoarea aserţiune:

Teza lui Turing *Orice funcţie calculabilă Turing este efectiv calculabilă*.

Observaţia 10.1.22: Teza lui Turing este o afirmaţie matematică prin care funcţiile calculabile Turing sunt identificate cu funcţiile efectiv calculabile. Noţiunea de funcţie calculabilă Turing este o noţiune riguros definită matematic, în timp ce noţiunea de funcţie efectiv calculabilă este dată în planul empirico-matematic, fiind sugerată de o lungă practică cu funcţiile X_n ce se pot "calcula". Această teză este analoagă tezei lui Church prezentată în capitolul de funcţii recursive. După cum vom vedea în cele ce urmează aceste două teze sunt reflectarea unui acelaşi principiu filosofic: găsirea unei noţiuni matematice care să reflecte "corect" noţiunea empirică de algoritm.

Un prim argument în favoarea tezei lui Turing este faptul că până acum nu s-a găsit nici o funcţie efectiv calculabilă care să nu fie calculabilă Turing.

Teorema următoare stabileşte legătura dintre funcţiile recursive şi funcţiile calculabile Turing.

Teorema 10.1.23 Pentru orice funcţie $f : (N^+)^k \to N^+$ sunt echivalente următoarele afirmaţii:

(i) f este o funcţie recursivă.

(ii) este calculabilă Turing.

Demonstraţia acestei teoreme depăşeşte cadrul acestui manual, presupunând o serie de construcţii dificile. Cititorul o poate găsi de exemplu în J. Malitz, *Introduction to Mathematical Logic*, Springer-Verlag, 1979 sau în J. Donald Monk, *Mathematical Logic*, North-Holland, 1978.

Această teoremă este cu totul remarcabilă, semnificaţia ei fiind profundă atât ca rezultat matemetic, cât şi ca sens filosofic. Pe plan matematic ea furnizează un procedeu comod de-a găsi funcţii calculabile Turing (am văzut că nu este uşor totdeauna să verificăm că o funcţie este calculabilă Turing).

Sensul filosofic poate depăşeste pe cel matematic. Funcţiile recursive(cu numeroase definiţii echivalente) au fost inventate înainte de maşinile Turing. Faptul că cei doi pretendenţi pentru a defini matematic noţiunea de funcţie calculabilă(funcţiile recursive şi funcţiile calculabile Turing) s-au dovedit a fi echivalenţi este un argument foarte serios în favoarea tezei lui Church şi a tezei lui Turing. Vom observa că acest mod de-a "măsura" fidelitatea reflectării unor noţiuni şi fapte apărute în practică prin concepte şi teorii matematice este frecvent în matematică, ca şi în alte ştiinţe.

Lecţia 11 - Algoritmi Markov

Algoritmii Markov reprezintă o a treia noţiune matematică care reflectă noţiunea intuitivă de algoritm. Scopul acestui capitol este de a reprezenta concepte şi proprietăţi fundamentale ale algoritmurilor Markov, clarificate printr-o serie de exemple. Un rezultat important al lui Detlovs identifică funcţiile recursive cu funcţiile ce se pot calcula cu algoritmii Markov (funcţiile algoritmice), ceea ce arată că cele trei tehnici prezentate în acest capitol –funcţii recursive, algoritmi Markov şi maşini Turing -sunt echivalente.

11.1-*Definiţia algoritmilor Markov*

Fie **Λ** o mulţime finită, pe care o vom numi alfabet. Un şir finit de elemente ale lui **Λ** se va numi cuvânt. Dacă A, B sunt două cuvinte:

$A = \alpha_1\alpha_2...\alpha_n$
$B = \beta_1\beta_2...\beta_m$

atunci vom nota cu AB cuvântul:

$$AB = \alpha_1\alpha_2...\alpha_n\beta_1\beta_2...\beta_m.$$

Fie $\mathcal{M}(\Lambda)$ mulţimea cuvintelor construite cu elemente din **Λ** la care adăugăm "cuvântul vid" Ø. Prin operaţia:

$(A, B) \in \mathcal{M}(\Lambda) \times \mathcal{M}(\Lambda) \quad \Rightarrow \quad AB \in \mathcal{M}(\Lambda)$

mulţimea $\mathcal{M}(\Lambda)$ este înzestrată cu o structură algebrică de monoid.

Observaţia 11.1.1:. Reamintim că un monoid este o mulţime înzestrată cu o operaţie binară asociativă care are un element neutru.
Elementul neutru al lui $\mathcal{M}(\Lambda)$ este Ø, dacă adoptăm convenţia:
$\emptyset A = A\emptyset = A$
pentru orice cuvânt A.
$\mathcal{M}(\Lambda)$ se numeşte monoidul liber generat de Λ.

Dacă A, B sunt două cuvinte, vom spune că A este inclus în B dacă există cuvintele P, Q astfel încât $B = PAQ$. Vom scrie aceasta prin $A \subset B$. Este evident că descompunerea lui B sub forma PAQ nu este unică. Atunci când $B = PAQ$ şi lungimea cuvântului P este minimă avem prima incluziune a lui A în B.
Definim funcţia
$\Sigma : \mathcal{M}(\Lambda) \times \mathcal{M}(\Lambda) \times \mathcal{M}(\Lambda) \to \mathcal{M}(\Lambda)$
în modul următor:
• dacă $A \not\subset B$, punem $\Sigma(A, B, C) = B$
• dacă $A \subset B$ şi prima incluziune a lui A în B este $B = PAQ$, vom pune:
$\Sigma(A, B, C) = PCQ$

Definiţia 11.1.2: Fie Γ un alfabet astfel încât $\Lambda \subset \Gamma$. Un algoritm Markov pe Γ este o funcţie $\mathcal{F}$ definită pe o parte $\mathcal{A}$ a lui $\mathcal{M}(\Lambda)$ cu valori în $\mathcal{M}(\Lambda) \times \{0, 1\}$
$\mathcal{F}: \mathcal{A} \to \mathcal{M}(\Lambda) \times \{0, 1\}$

Observaţia11.1.3: După cum se va vedea în cele ce urmează este necesar să introducem anumite simboluri în plus fata de cele ale lui Λ, de aceea consideram un alfabet mai mare Γ.
Un algoritm este perfect determinat de un tablou de forma:

A_1 B_1 ε_1
A_2 B_2 ε_2
. . .
. . .
. . .
A_n B_n ε_n

unde $A_i \in \mathcal{M}(\Lambda)$, $Bi \in \mathcal{M}(\Lambda)$ şi $\varepsilon_i \in \{0, 1\}$ pentru orice $i = 1,\ldots, n$.
Într-adevăr, dat fiind un algoritm Markov $\mathcal{F}$ domeniul său va fi exact mulţimea:
$\mathcal{A} = \{ A_1, A_2,\ldots,A_n \}$
iar un şir $A_i B_i \varepsilon_i$ este definit de:
$\mathcal{F}(A_i) = (B_i, \varepsilon_i)$, $i = 1,\ldots, n$.
Dacă se dă un tablou ca mai sus, algoritmul Markov asociat este definit în mod natural.
Vom prefera să reprezentăm şirurile $A_i B_i \varepsilon_i$ prin săgeţi:

$A_i B_i 0$ prin $A_i \to B_i$
$A_i B_i 1$ prin $A_i \to \bullet B_i$

Convenţie. Vom scrie:

$\to B_i$ în loc de $\varnothing \to B_i$
$A_i \to$ în loc de $A_i \to \varnothing$

Definiţia 11.1.4: Un pas algoritmic cu algoritmul Markov $\mathcal{F}$ este o pereche de cuvinte (A, B) cu următoarele proprietăţi:

(i) Există $i \le n$ astfel încât $A_i \subset A$.
(ii) $B = \sum (A_{i_1}, A, B_{i_1})$, unde i_1 este primul indice pentru care $A_{i_1} \subset A$.

Definiţia 11.1.5:. Un pas algoritmic (A, B) este final dacă $A_{i_1} \to B_{i_1}$ în cazul când i_1 este primul indice pentru care $A_{i_1} \subset A$.

Definiţia 11.1.6: Un calcul cu algoritmul Markov $\mathcal{F}$ este un şir finit de cuvinte:

$(P_0, P_1,...., P_m)$

cu proprietăţile următoare:

a). $P_0 = P$
b). (P_j, P_{j+1}) este un pas algoritmic care nu este final pentru $j = 0, 1,..., m\text{-}2$.
c). (P_{m-1}, P_m) este un pas algoritmic final sau $A_i \not\subset P_{m-1}$ pentru orice $i = 1,..., n$.

Vom nota în acest caz $\mathcal{F}(P) = P_m$.

Observaţia 11.1.7: Deci $\mathcal{F}$ operează asupra unui cuvânt P în felul următor: Se caută cel mai mic indice i astfel încât $A_i \subset P$. Dacă un astfel de indice nu există vom pune $\mathcal{F}(P) = P$. În cazul când i_1 este un asemenea indice se înlocuieşte cuvântul A_{i_1}, în prima sa incluziune în P, prin cuvântul B_{i_1}.

Când avem $A_{i_1} \to \bullet B_{i_1}$ operaţiunea este încheiată şi $\mathcal{F}(\mathrm{P}) = \sum (A_{i_1}, P, B_{i_1})$

Dacă avem $A_{i_1} \to B_{i_1}$, se reia operaţiunea de la capăt asupra cuvântului $\sum (A_{i_1}, P, B_{i_1})$, etc.

Aplicarea unui algoritm Markov asupra unui cuvânt *P* conduce la un lanţ de cuvinte:

$$P \to P_1 \to P_2 \to ... \to P_m \to .....$$

în care $P_j \to P_{j+1}$ corespunde situaţiei

$$P_j \to \sum (A_{i_{j+1}}, P_i, B_{i_{j+1}}) = P_{j+1}$$

unde i_{j+1} este cel mai mic indice astfel încât $A_{i_{j+1}} \subset P_i$.

Observaţia 11.1.8: Pot exista situaţii când un asemenea lanţ este infinit; spunem în acest caz că $\mathcal{F}(P)$ nu este definit.

Un lanţ ca mai sus va fi finit dacă întâlnim una din următoarele două situaţii:

- se ajunge la o substituţie de forma $A_i \to \bullet B_i$;
- nu există nici un indice i_{n+1} astfel încât $A_{i_{n+1}} \subset P_m$.

Exemplul 11.1.9: Fie alfabetul

$\Lambda = \{\xi_1, \xi_2, \ldots, \xi_p\}$

Vom lua $\Gamma = \Lambda \cup \{\alpha\}$. Tabloul substituţiilor este următorul:

$$\begin{aligned}
\xi_i\alpha &\to \alpha\xi_i \\
\alpha\xi_i &\to \alpha \\
\alpha &\to \bullet C \\
A\xi_i &\to \alpha \\
\xi_i A &\to \alpha \\
A &\to \bullet B \\
&\to \alpha
\end{aligned}$$

unde A, B, C sunt trei cuvinte finite(din $\mathcal{M}(\Lambda)$).

Observaţia 11.1.10: $\xi_i\alpha \to \alpha\xi_i$ reprezintă un şir de substituţii:

$\xi_1\alpha \rightarrow \alpha\xi_1$
$\xi_2\alpha \rightarrow \alpha\xi_2$
……
$\xi_n\alpha \rightarrow \alpha\xi_n$

Vom cerceta cum acţionează acest algoritm Markov $\mathcal{F}$ asupra unui cuvânt $P \in \mathcal{M}(\Gamma)$. Distingem următoarele cazuri:

(I). $\boldsymbol{P = A}$.
Cum $P \in \mathcal{M}(\Lambda)$ şi $\alpha \notin \Lambda$ rezultă că $\xi_i\alpha$, $\alpha\xi_i$, α nu sunt incluse în P. Este evident că $A\xi_i$, $\xi_i A$ nu sunt incluse în P. A este primul cuvânt din lista de mai sus ce este inclus în P, deci:
$\sum(A, P, B) = \sum(A, A, B) = B$
Cum $A \rightarrow \bullet B$ aplicarea algoritmului se termină şi vom avea:
$\mathcal{F}(A) = B$.

(II). $\boldsymbol{P \neq A, A \subset P}$ şi $\boldsymbol{P}$ este de forma:
$P = \xi_{i1}\,\xi_{i2} \ldots \xi_{ip}\, A\xi_{i0}\, \xi_{ip+1} \ldots \xi_{ip+j}$
unde A apare în prima sa incluziune în P.
$\xi_i\alpha$, $\alpha\xi_i$, α nu sunt incluse în P. $A\xi_{i0}$ este primul cuvânt din lista tabloului substituţiilor ce apare în P deci:
$P_1 = \sum(A\xi_{i0}, P, \alpha) = \xi_{i1} \ldots \xi_{ip}\alpha\xi_{ip+1} \ldots \xi_{ip+j}$
Vom avea în continuare:
$P_2 = \sum(\xi_{ip}\alpha, P_1, \alpha\xi_{ip}) = \xi_{i1}\xi_{i2} \ldots \xi_{ip-1}\alpha\xi_{ip} \ldots \xi_{ip+j}$
şi continuând în acest fel:
$P_{p+1} = \alpha\xi_{i1}\xi_{i2} \ldots \xi_{ip+j}$
$\alpha\xi_{i1}$ este primul cuvânt din listă inclus în P:
$P_{p+2} = \sum(\alpha\xi_{i1}, P_{p+1}, \alpha) = \alpha\xi_{i2}\xi_{i3} \ldots \xi_{ip+j}$

$P_{p+3} = \sum(\alpha\xi_{i2}, P_{p+1}, \alpha) = \alpha\xi_{i3} \ldots \xi_{ip+j}$
……
$P_{p+j+1} = \alpha$
În sfârşit vom avea:
$P_{p+j+2} = \sum(\alpha, P_{p+j+1}, C) = C.$
Cu aceasta am terminat aplicarea algoritmului, deoarece $\alpha \rightarrow \bullet C$. Obţinem deci $\mathcal{F}(P) = C$.

(III). $\boldsymbol{P \neq A,\ A \subset P}$ şi $\boldsymbol{P}$ este de forma:
$P = \xi_{i_1} \ldots \xi_{i_p} A$
Vom avea
$$P_1 = (\xi_{ip} A, P, \alpha) = \xi_{i1}, \ldots, \xi_{ip-1}\alpha$$
conform substituţiei $\xi_i A \rightarrow \alpha$. Aplicăm mai departe substituţiile
$\xi_i\alpha \rightarrow \alpha\xi_i$

deci

$$P_2 = \sum(\xi_{i_{p-1}}\alpha, P_1, \alpha\xi_{i_{p-1}}) = \xi_{i1}...\xi_{i_{p-2}}\alpha\xi_{i_{p-2}}$$

$$P_3 = \sum(\xi_{i_{p-2}}\alpha, P_2, \alpha\xi_{i_{p-2}}) = \xi_{i1}...\alpha\xi_{i_{p-2}}\xi_{i_{p-1}}$$

……

$$P_p = \sum(\xi_{i_1}\alpha, P_{p-1}, \alpha\xi_{i_1}) = \alpha\xi_{i_1}...\xi_{i_{p-1}}$$

Mai departe, prin substituţiile $\alpha\xi_i \to \alpha$:

$$P_{p+1} = \sum(\alpha\xi_{i1}, P_p, \alpha) = \alpha\xi_{i2}...\xi_{i_{p-1}}.$$

$$P_{p+2} = \sum(\alpha\xi_{i2}, P_{p+1}, \alpha) = \alpha\xi_{i3}...\xi_{i_{p-1}}.$$

……

$$P_{2p-1} = \sum(\alpha\xi_{ip-1}, P_{2p-2}, P) = \alpha.$$

Aplicând mai departe substituţia $\alpha \to \bullet C$ obţinem:

$P_{2p} = C$ şi deci $\mathcal{F}(P) = C$.

(IV). $A \subset P, P = \xi_{i_1}...\xi_{i_p}$.

În acest caz singura substituţie posibilă este $\to\alpha$, deci :

$$P_1 = \sum(\varnothing, P, \alpha) = \alpha\xi_{I_1}...\xi_{i_p}$$

$$P_2 = \sum(\alpha\xi_{i1}, P_1, \alpha) = \alpha\xi_{i2}...\xi_{ip}$$

.....

$$P_{p+1} = \sum(\alpha\xi_{ip}, P_p, \alpha) = \alpha$$

$$P_{p+2} = \sum(\alpha, P_{p+1}, C) = C$$

Rezultă $\mathcal{F}(P) = C$.

În concluzie, vom avea:

$$\mathcal{F}(P) = \begin{cases} B, \text{ dacă } P = A \\ C, \text{ dacă } P \neq A. \end{cases}$$

Pentru a reprezenta acţiunea unui algoritm Markov asupra unui cuvânt vom numerota substituţiile tabloului.

(1). $A_1 \to B_1$
(2). $A_2 \to B_2$
……
(n). $A_n \to B_n$

şi vom scrie lanţul $P \to P_1 \to P_2 \to ... \to P_m \to .....$ sub forma următoare:

$P \to P_1$ (j_1)
$\to P_2$ (j_2)
…….
$\to P_m$ (j_m)

……

indicând în paranteză numărul substituţiei.
Dacă aplicăm de p ori aceiaşi substituţie (j):

$$P_i \to P_{i+1} \quad (j_{i+1} = j)$$
$$P_{i+1} \to P_{i+2} \quad (j_{i+2} = j)$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$P_{i+p-1} \to P_{i+p} \quad (j_{i+p} = j)$$

vom nota aceasta prescurtat: $P_i \Rightarrow P_{i+p} \quad (j)$
Dacă folosim de p ori două substituţii $j_1, j_2,$ vom nota aceasta:
$P_i \Rightarrow P_{i+p} \quad (j_1, j_2)$.
Vom ilustra toate acestea pe următorul exemplu:

Exemplul 11.1.11: Fie alfabetul:

$$\Lambda = \{\xi_1, \xi_2, \ldots, \xi_n\}$$
$$\Gamma = \Lambda \cup \{\alpha, \beta, \gamma\}$$

şi algoritmul Markov dat de:

$$\xi_i \xi_j \beta \to \xi_j \beta \xi_i \quad (I)$$
$$\alpha \xi_i \to \xi_i \beta \xi_i \alpha \quad (II)$$
$$\beta \to \gamma \quad (III)$$
$$\gamma \to \quad (IV)$$
$$\alpha \to \bullet \quad (V)$$
$$\to \alpha \quad (VI)$$

Fie $P \in \mathcal{M}(\Gamma)$ cuvântul următor: $P = \xi_{i1}\xi_{i2}\ldots\xi_{ip}$.
Aplicând algoritmul respectiv asupra lui P vom avea:

$$P \to \underline{\alpha\xi_{i1}}\xi_{i2}...\xi_{ip} \qquad (VI)$$
$$\to \xi_{i1}\beta\xi_{i1}\underline{\alpha\xi_{i2}}...\xi_{ip} \qquad (II)$$
$$\to \xi_{i1}\underline{\beta\xi_{i1}}\underline{\xi_{i2}}\beta\xi_{i2}\alpha\xi_{i3}...\xi_{ip} \qquad (II)$$
$$\to \xi_{i1}\beta\xi_{i2}\beta\xi_{i1}\xi_{i2}\underline{\alpha\xi_{3}}...\xi_{ip} \qquad (I)$$
$$\to \xi_{i1}\beta\xi_{i2}\beta\xi_{i1}\underline{\xi_{i2}\xi_{i3}}\underline{\beta\xi_{3}}\alpha\xi_{i4}...\xi_{ip} \qquad (II)$$
$$\Rightarrow \xi_{i1}\beta\xi_{i2}\beta\xi_{3}\beta\xi_{i1}\xi_{i2}\xi_{3}\underline{\alpha\xi_{i4}}...\xi_{ip} \qquad (I)$$
$$\Rightarrow \xi_{i1}\underline{\beta}\xi_{i2}\beta\xi_{3}\beta...\xi_{ip}\beta\xi_{i1}\xi_{i2}...\xi_{ip}\alpha \qquad (II, I)$$
$$\to \xi_{i1}\gamma\xi_{i2}\underline{\beta}\xi_{i3}\beta...\xi_{ip}\beta\xi_{i1}...\xi_{ip}\alpha \qquad (III)$$
$$\Rightarrow \xi_{i1}\underline{\gamma}\xi_{i2}\gamma\xi_{i3}\gamma...\xi_{ip}\gamma\xi_{i1}...\xi_{ip}\alpha \qquad (III)$$
$$\to \xi_{i1}\xi_{i2}\underline{\gamma}\xi_{i3}\gamma...\xi_{ip}\gamma\xi_{i1}...\xi_{ip}\alpha \qquad (IV)$$
$$\Rightarrow \xi_{i1}\xi_{i2}...\xi_{ip}\xi_{i1}....\xi_{ip}\underline{\alpha} \qquad (IV)$$
$$\to \xi_{i1}\xi_{i2}...\xi_{ip}\xi_{i1}...\xi_{ip} \qquad (V)$$

În cele de mai sus am subliniat prima apariţie a unui cuvânt din lista din stânga a tabloului substituţiilor.
Deducem din calculul de mai sus că:
$\mathcal{F}(P) = PP$, pentru orice $P \in \mathcal{M}(\Lambda)$.

Exemplul 11.1.12: Fie $\Lambda = \{\xi_1, \xi_2, \ldots, \xi_n\}$ şi $\Gamma = \Lambda \cup \{\ |\ ,\ \beta\ \}$. Considerăm tabloul de substituţii:

$\xi_i \to |$ (I)
$|\ | \to |$ (II)
$| \to \bullet$ (III)
$\to \beta$ (IV)

Fie $P = \xi_{i1}\xi_{i2}...\xi_{ip}$ un cuvânt oarecare al lui $\mathcal{M}(\Lambda)$. Dacă $P \neq 0$, atunci numai substituţia (IV) este posibilă:

$$\varnothing \to \beta$$

deci $\mathcal{F}(\varnothing) = \beta$.
Pentru $P \neq \varnothing$:

$$P \to I\,\xi_{i2}...\xi_{ip} \qquad (I)$$
$$\to I\,I\,\xi_{i3}...\xi_{ip} \qquad (I)$$
$$\Rightarrow \underbrace{I\,I...I}_{p-ori} \qquad (I)$$
$$\to \underbrace{I\,I...I}_{(p-1)-ori} \qquad (II)$$
$$\Rightarrow I \qquad (II)$$
$$\to \bullet\varnothing \qquad (III)$$

deci $\mathcal{F}(P) = \beta$ pentru $P \neq \varnothing$.

Exemplul 11.1.13: Algoritmul care are tabloul $\to \bullet$ realizează aplicaţia identică a monoidului $\mathcal{M}(\Lambda)$.

Observaţia11.1.14: Doi algoritmi Markov pot fi egali ca funcţii, însă tablourile lor de substituţie să difere.

Vom spune că algoritmii Markov $\mathcal{F}_1$, $\mathcal{F}_2$ sunt egali ($\mathcal{F}_1 = \mathcal{F}_2$) dacă au aceleaşi tablouri de substituţii; dacă $\mathcal{F}_1(P) = \mathcal{F}_2(P)$ pentru orice cuvânt P, spunem că $\mathcal{F}_1$, $\mathcal{F}_2$ sunt echivalenţi ($\mathcal{F}_1 \equiv \mathcal{F}_2$).

Se vede imediat că $\mathcal{F}_1 = \mathcal{F}_2 \Rightarrow \mathcal{F}_1 \equiv \mathcal{F}_2$.

11.2. *Proprietăţi ale algoritmilor Markov*

În § 1 am văzut că un calcul $P \to P_1 \to P_2 \to ... \to P_m$ se poate termina în două moduri:

a) avem o substituţie cu punct $A_{i_n} \to \bullet B_{i_n}$

b) nu există nici un indice i_{n+1} cu proprietatea că $A_{i_{n+1}} \subset P_n$.

Următoarea propoziţie va reduce problema la cazul a).

Propoziţia 11.2.1: Pentru orice algoritm Markov $\mathcal{F}$ există un algoritm Markov $\mathcal{F}^*$ echivalent cu $\mathcal{F}$ cu proprietatea că atunci când $\mathcal{F}$ (P) este definit lanţul:

$P \to P_1 \to P_2 \to ... \to \bullet P_n = \mathcal{F}^*(P)$ se termină printr-o substituţie cu punct.

Demonstraţie. Considerăm tabloul substituţiilor lui $\mathcal{F}$:

$A_1 \to B_1$ (1)
$A_2 \to B_2$ (2)
…….....
$A_n \to B_n$ (n)

În partea dreaptă putem avea eventual puncte. Algoritmul $\mathcal{F}^*$ se obţine prin adăugarea substituţiei $\to\bullet$.

$A_1 \to B_1$ (1)
$A_2 \to B_2$ (2)
…….....
$A_n \to B_n$ (n)
$\to \bullet$ (n+1)

Dacă P este un cuvânt pentru care $\mathcal{F}(P)$ este definit, lanţul:

$$P \to P_1 \to P_2 \to ... \to P_m = \mathcal{F}(P)$$

este finit şi avem unul din cele două cazuri menţionate mai sus.

a) Ultimul pas al lanţului $P_{m-1} \to P_m$ este dat de o substituţie cu punct (din tabloul lui $\mathcal{F}^*$).

$$A_k \to \bullet B_k$$

Cum substituţia respectivă face parte şi din tabloul lui $\mathcal{F}$rezultă $\mathcal{F}^*(P) = P_m = \mathcal{F}(P)$.

b) Nu există nici un indice *i* astfel încât $A_i \subset P_m$. Şirul $P \to P_1 \to P_2 \to ... \to P_m$ poate fi obţinut şi prin aplicarea lui $\mathcal{F}^*$, după care intră în acţiune substituţia (n+1) care dă $P_m \to \bullet P_m$ şi calculul cu algoritmul $\mathcal{F}^*$ s-a terminat. Observăm că $\mathcal{F}^*(P) = \mathcal{F}(P) = P_m$.

Deci $\mathcal{F}^* \equiv \mathcal{F}$. Demonstraţia s-a terminat.

Fie acum două alfabete:

$$\Lambda = \{\xi_1,...,\xi_n\}$$
$$\Lambda' = \{\theta_1,...,\theta_m\}$$

Propoziţia 11.2.2: Pentru orice funcţie f : $\Lambda \to \Lambda'$ există o unică funcţie $\overline{f} : \mathcal{M}(\Lambda) \to \mathcal{M}(\Lambda')$ astfel încât următoarele proprietăţi sunt verificate.
(1). Următoarea diagramă este comutativă:

$$\begin{array}{ccc} \Lambda & \xrightarrow{\quad f \quad} & \Lambda' \\ \downarrow & & \downarrow \\ \mathcal{M}(\Lambda) & \xrightarrow[\overline{f}]{} & \mathcal{M}(\Lambda') \end{array}$$

unde cu $\downarrow$ am notat scufundările canonice.
(2) $\overline{f}$ este un morfism de monoizi, adică:
$\overline{f}(\varnothing) = \varnothing$.

$$\overline{f}(PP') = \overline{f}(P)\overline{f}(P')$$

pentru orice cuvinte P, P′$\in \mathcal{M}(\Lambda)$.

Demonstraţie. Existenţă: Pentru orice cuvânt P = $\xi_{i1}\xi_{i2}...\xi_{ip}$ al lui $\mathcal{M}(\Lambda)$ vom nota:

$$\overline{f}(P) = f(\xi_{i1})f(\xi_{i2})...f(\xi_{ip}) \qquad (1)$$

Dacă P = $\varnothing$, notăm $\overline{f}(P) = \varnothing$, cu aceasta, proprietăţile cerute sunt imediate.
Unicitate: Presupunem că ar mai exista o funcţie g : $\mathcal{M}(\Lambda) \to \mathcal{M}(\Lambda')$ cu proprietăţile (1) şi (2). Conform lui (1), pentru orice $\xi \in \Lambda$, avem g(ξ) = f(ξ). Ţinând seama de acest lucru şi de (2), vom avea, pentru orice cuvânt P = $\xi_{i1}\xi_{i2}...\xi_{ip}$:

$$g(P) = g(\xi_{i1}\xi_{i2}...\xi_{ip}) = g(\xi_{i1})g(\xi_{i2})...g(\xi_{ip})$$

$$= f(\xi_{i1})f(\xi_{i2})...f(\xi_{ip}) = \overline{f}(P)$$

Deci: g = $\overline{f}$.

Propoziţia 11.2.3:. Fiind dată o funcţie f : $\Lambda \to \Lambda'$ şi $\alpha \notin \Lambda \cup \Lambda'$ există un algoritm Markov $\mathcal{H}$ pe $\Lambda \cup \Lambda' \cup \{\alpha\}$ astfel încât pentru orice P$\in \mathcal{M}(\Lambda)$ să avem $\mathcal{H}(P) = \overline{f}(P)$, unde $\overline{f}$ este prelungirea lui f dată de propoziţia precedentă.
Demonstraţie. Vom nota $\delta_i = f(\xi_i), \quad i = 1,...,n$.

Dacă $P = \xi_{i_1}\xi_{i_2}...\xi_{i_p}$ atunci $\overline{f}(P) = f(\xi_{i_1})f(\xi_{i_2})...f(\xi_{i_p}) = \delta_{i_1}\delta_{i_2}...\delta_{i_p}$

Deci un calcul al lui $\mathcal{H}$ trebuie să transforme cuvântul $\xi_{i_1}, \xi_{i_2}, ..., \xi_{i_p}$ în cuvântul $\delta_{i_1}\delta_{i_2}...\delta_{i_p}$. Aceasta se obţine prin înlocuirea succesivă a lui ξ_{i_h} cu δ_{i_k}.

Fie $\mathcal{H}$ algoritmul Markov dat de următorul tablou de substituţii:

$\alpha\xi i \to \delta i\alpha$ (I)
$\alpha \to \bullet$ (II)
$\to \alpha$ (III)

Vom arăta că $\mathcal{H}(P) = \overline{f}(P)$ pentru orice P$\in \mathcal{M}(\Lambda)$. Luând $P = \xi_{i_1}...\xi_{i_p}$ avem:

$$P \to \alpha\xi_{i_1}\xi_{i_2}...\xi_{i_p} \qquad (III)$$
$$\to \delta_{i_1}\alpha\xi_{i_2}...\xi_{i_p} \qquad (I)$$
$$\to \delta_{i_1}\delta_{i_2}\alpha...\xi_{i_p} \qquad (I)$$
$$\Rightarrow \delta_{i_1}\delta_{i_2}...\alpha\xi_{i_p} \qquad (I)$$
$$\to \delta_{i_1}\delta_{i_2}...\delta_{i_p}\alpha \qquad (I)$$
$$\to \delta_{i_1}\delta_{i_2}...\delta_{i_p}$$

Observaţie: $(\mathcal{F}) = \mathcal{F}$.

În cele ce urmează vom considera $\Lambda = \{0, 1\}$ şi $\Gamma = \{0, 1, 2\}$.

Definiţia 11.2.4: O funcţie $f : N^m \to N$ se numeşte algoritmică dacă există un algoritm Markov $\mathcal{F}$ pe Λ astfel încât pentru orice $x_1,...,x_m \in N$ să existe un calcul al lui $\mathcal{F}$:

$$P_0 \to P_1 \to P_2 \to ... \to P_n$$

astfel încât să fie satisfăcute proprietăţile următoare:

(1) $P_0 = 0\ 1^{x1+1}\ 0\ 1^{x2+1}\ 0...0\ 1^{xm+1}\ 0\ 2$

(2) $P_1,..., P_{n-1}$ nu conţine pe 2.

(3) $0\ 1^{k+1}\ 0\ \ 2$ este inclus în P_n, unde $k = f(x_1,...,x_n)$.

Se spune că algoritmul Markov calculează funcţia f.

Observaţia 11.2.5: Dacă $f : N^m \to N$ este o funcţie algoritmică, atunci f este efectiv calculabilă. Într-adevăr, pentru orice $(x_1,...,x_m) \in N^m$ există un algoritm Markov cu ajutorul căruia putem obţine pe $f(x_1,...,x_n)$ într-un număr finit de paşi.

Teza lui Markov. Orice funcţie efectiv calculabilă este algoritmică. Bineînţeles că teza lui Markov eate o aserţiune nematematică care stabileşte echivalenţa între conceptul intuitiv de funcţie efectiv calculabilă cu acela matematic de funcţie algoritmică. Ea este similară cu teza lui Church pentru funcţiile recursive şi cu teza lui Turing pentru funcţii calculabile Turing.

Următorul rezultat, dat de V. Detlovs, arată echivalenţa între funcţiile recursive şi funcţiile algoritmice:

Teorema 11.2.6: Fie $f : N^k \to N$ o funcţie oarecare. Sunt echivalente afirmaţiile:

(1) f este algoritmică

(2) f este recursivă.

Nu dăm demonstraţia acestei teoreme, fiind foarte tehnică.

În concluzie, cele trei concepte propuse pentru a defini matematic funcţiile efectiv calculabile - funcţiile recursive, funcţiile calculabile Turing, funcţiile algoritmice sunt echivalente. Apreciem că acesta este unul din cele mai importante rezultate obţinute în logica matematică pînă acum. Toate cele trei direcţii au fost introduse şi studiate independent, iar faptul că s-au dovedit a fi echivalente matematic atestă - dacă mai era nevoie - puterea matematicii de a oglindi fidel fragmente ale realităţii.

Exerciţii rezolvate

Exemple de masini Turing

În acest paragraf vom pune în evidenţă o serie de maşini Turing.
Dacă M este o maşină Turing şi **t**, **s** sunt două poziţii ale benzii, atunci vom nota M | t ⇝ s dacă există un calcul în raport cu M cu intrarea **t** şi cu ieşirea **s**.
În cazul când **t** este:
(k + j, 1): $a_1 \ldots a_k\ b_1 \ldots b_j \ldots b_r .. c_1\ c_2 \ldots$

iar **s** este:
(k + *l*, n): $a_1 \ldots a_k\ b_1' .. b_l' \ldots b_r' .. c_1\ c_2 \ldots$

atunci vom nota faptul ca M | t ⇝ s prin:

$b_1 \ldots b_j \ldots b_r .. \rightsquigarrow b_1' .. b_l' \ldots b_r' \ldots$

Lema 1. Există o maşină Turing, notată:

compress

astfel încât pentru orice $m, n \in N^+$ sa avem

compress | $1\ 0^m\ 1\ 1^{(n-1)}\ 0 \ldots \rightsquigarrow \ldots 1\ 0\ 1\ 1^{(n-1)}\ 0^m \ldots$

Demonstrţie: Considerăm următoarea maşina Turing:

	0	1
1		1 L 2
2	1 L 3	0 R 7
3	0 R 4	1 R 2
4	0 L 5	1 R 4
5		0 L 6
6	0 R 1	1 L 6

Această maşină Turing cercetează întâi dacă m ≥ 1. Dacă nu, atunci intrarea are aceiaşi bandă ca şi ieşirea:

(k, 1): $\ldots 1\ \ 0\ \ 1\ \ 1^{(n-1)}\ \ 0\ldots\ldots$

(k-1, 2): $\ldots 1\ \ 0\ \ 1\ \ 1^{(n-1)}\ \ 0\ldots\ldots$

(k-2, 3): $\ldots 1\ \ 1\ \ 1\ \ 1^{(n-1)}\ \ 0\ldots\ldots$

(k-1, 2): $\ldots 1\ \ 1\ \ 1\ \ 1^{(n-1)}\ \ 0\ldots\ldots$

(k, 7): $\ldots 1\ \ 0\ \ 1\ \ 1^{(n-1)}\ \ 0\ldots\ldots$

Dacă m ≥ 2, atunci un calcul parţial va da următoarea poziţie a benzii:
(k+m, 1): $\ldots 1\ \ 0^{(n-1)}\ \ 1^n\ \ 0\ \ 0\ldots\ldots$

şi acest proces de tipărire a lui 1 la stânga lui 1^n şi de ştergere a lui 1 la dreapta lui 1^n va da în final …1 0 1 $1^{(n-1)}$ 0^m……

Vom ilustra această afirmaţie în cazul m = 2, n = 2.

(k, 1): …1 0 0 1 1 0……

(k-1, 2): …1 0 0 1 1 0……

(k-2, 3): …1 0 1 1 1 0……

(k-1, 4): …1 0 1 1 1 0……

(k, 4): …1 0 1 1 1 0……

(k+1, 4): …1 0 1 1 1 0……

(k+2, 4): …1 0 1 1 1 0……

(k+1, 5): …1 0 1 1 1 0……

(k, 6): …1 0 1 1 0 0……

(k-1, 6): …1 0 1 1 0 0……

(k-2, 6): …1 0 1 1 0 0……

(k-1, 1): …1 0 1 1 0 0……

(k-2, 2): …1 0 1 1 0 0……

(k-3, 3): … 1 1 1 1 0 0……

(k-2, 2): …1 1 1 1 0 0……

(k-1, 7): …1 0 1 1 0 0……

Fie M = (d, p, s). Vom scrie M(i, k) = (d(i, k), p(i, k), s(i, k)). Notăm cu S(M) multimea tuturor **k** astfel încât (0, k)∈DomM, (1, k)∈DomM sau k∈s(DomM).

Observaţie. s(DomM) este imaginea lui DomM prin funcţia *s*.

Fie **u** un număr natural strict mai mare decât cardinalul lui S(M), care este evident finit. Definim maşina Turing $\underset{\downarrow}{M}_u$ în felul următor:

$$Dom(\underset{\downarrow}{M}_u) = \{(i,k)/i \in \{0,1\} \text{şi } k \in S(M)\}$$

$$\underset{\downarrow}{M}_u(i,k) = \begin{Bmatrix} M(i,k), daca(i,k) \in DomM \\ (i,0,u), \text{în } celelalte\ cazuri \end{Bmatrix}$$

(j, k): *a* este o ieşire a lui M având intrarea **t** dacă şi numai dacă (j, u): *a* este o ieşire a lui $\underset{\downarrow}{M_u}$ având intrarea **t**.

Vom defini acum o altă maşină Turing [M, *l*] cu domeniul:
{(i, *l*+k) | (i, k)∈DomM}
prin relaţia:
[M, *l*](i, *l*+k) = (d(i, k), p(i, k), s(i, k), s(i, k) + *l*)
pentru orice (i, k)∈DomM.
Deci [M, *l*] este maşina Turing obţinută din M înlocuind în tabloul lui M starea **k** cu starea ***l*+k**.
Fie u = card(S(M))+1. Vom defini acum o serie de maşini Turing:

(I) Prin $\underset{M_1}{\overset{M}{\downarrow}}$ vom înţelege maşina Turing $\underset{\downarrow}{M_u} \cup [M_1, u-1]$. Această maşină Turing este rezultatul alăturării lui M_1 la M astfel încât o ieşire a lui M este o intrare a lui M_1.

(II) Fie M_0 maşina Turing:

u	0 0 u+1	

Definim $\overset{M}{\swarrow}$ M_1 ca fiind maşina Turing $\underset{\downarrow}{M_u} \cup M_0 \cup [M_1, u]$.

Referitor la această maşină Turing, toate ieşirile (i, j): *a* ale lui M pentru care $a_j = 0$ sunt convertite în intrări pentru M_1.

(III) Daca M_0 este maşina Turing:

u		1 0 u+1

Atunci $M \searrow$ M_1 va fi maşina Turing $\underset{\downarrow}{M_u} \cup M_0 \cup [M_1, u]$.

Toate ieşirile lui *M* în care este **l** pe locul indicat de capul de citire-scriere ↑ sunt aici intrări pentru M_1 .

(IV) Fie $\overset{M}{\nearrow}$ maşina Turing $\underset{\downarrow}{M_u} \cup M_0$ unde:

M_0 = u	0 0 1	

O ieşire (j, k) a lui M pentru care $a_j = 0$ devine o intrare a lui $\overset{M}{\nearrow}$.

(V) Dacă:

M_0 = u		1 0 1

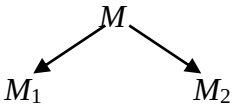
vom defini M ca fiind maşina Turing $\underset{\downarrow}{M_u} \cup M_0$. O ieşire (j, k) a lui M pentru care

a_j =1 este convertită într-o intrare a lui M.

(VI) Fie M_0 maşina Turing:

u	0 0 u+1	1 0 u+v+1

unde v = card (S(M_1)). Definim maşina Turing următoare:



ca fiind $\underset{\downarrow}{M_u} \cup M_0 \cup [M_1,\ u] \cup [M_2,\ u+v]$.

O ieşire a lui M devine o intrare a lui M_1 dacă pe bandă apare **0** în locul marcat de ↑, respectiv o intrare lui M_2 dacă ↑ indica **1** pe bandă.

Următoarele exemple vor facilita înţelegerea modului de funcţionare al maşinilor Turing definite mai sus.

Considerăm maşina Turing următoare:

	0	1
M: 1	1 L 2	0 R 1
2	1 0 2	

Scrisă explicit această maşină Turing este data de M = (d, p, s), cu:

DomM ={(0,1), (1, 1), (0, 2)}.
d(0, 1) = 1, p(0, 1) = -1, s(0, 1) = 2.
d(1, 1) = 0, p(1, 1) = 1, s(1, 1) = 1,
d(0, 2) = 1, p(0, 2) = 0, s(0, 2) = 2.
Considerăm următorul calcul cu maşina Turing M:

(2, 1): 0 1 0 0…
↑

(3, 1): 0 0 0 0…
↑

(2, 2): 0 0 0 1…
↑

(2, 2): 0 1 1 1…
↑

Aplicând exact definiţia avem S(M) = {1, 2}. Să determinăm acum maşina Turing $\underset{\downarrow}{M_u}$ pentru u = 3:

$Dom\, \underset{\downarrow}{M_3} = \{(0,1),(1,1),(0,2),(1,2)\}$

Singura adăugire faţă de M este:

$\underset{\downarrow}{M_s}(1, 2) = (1, 0, 3)$

deci tabloul ce reprezintă pe $\underset{\downarrow}{M_3}$ este:

$\underset{\downarrow}{M_3}$:

	0	1
1	1 L 2	0 R 1
2	1 0 2	1 0 3

Un calcul cu $\underset{\downarrow}{M_3}$ care are intrarea (2, 1): 0 1 0 0… va arăta astfel:

(2, 1): 0 1 0 0…

(3, 1): 0 0 0 0…

(2, 2): 0 0 1 0…

(2, 2): 0 1 1 0…

(2, 3): 0 1 1 0…

M fiind maşina Turing de mai sus, [*M*, 4] va fi maşina Turing dată de:

[M, 4]:

	0	1
5	1 L 6	0 R 5
6	1 0 6	

Fie M_1 următoarea maşină Turing:

M_1:

	0	1
1	1 L 2	0 R 1

Un calcul cu aceasta masina Turing care are intrarea (2, 1): 0 1 1 0… va fi:

(2, 1): 0 1 1 0…

(3, 1): 0 0 1 0…

(4, 1): 0 0 0 0…

(3, 2): 0 0 0 1…

Să determinăm acum maşiniile Turing (I) - (VI) definite anterior:

(I) Având în vedere că [M_1, 2] are forma:

0	1

$[M_1, 2]$:

1	1 L 4	0 R 3

şi că $\overset{M}{\underset{M_1}{\downarrow}} = M_3 \cup [M_1, 2]$ vom avea:

$\overset{M}{\underset{M_1}{\downarrow}}$:

	0	1
1	1 L 2	0 R 1
2	1 0 2	1 0 3
3	1 L 4	0 R 3

Un calcul cu această maşină având intrarea (2, 1): 0 1 1 0…

(2, 1): 0 1 0 0…

(3, 1): 0 0 0 0…

(2, 2): 0 0 1 0…

(2, 2): 0 1 1 0…

(2, 3): 0 1 1 0…

(3, 3): 0 0 1 0…

(4, 3): 0 0 0 0…

(3, 4): 0 0 0 1 …

(II). Având în vedere că:

$M = \underset{\downarrow}{M_3} \cup M_0 \cup [M_1, 3]$

M_1

M_0:

	0	1
3	0 0 4	

$[M_1,3]$:

	0	1
4	1 L 5	0 R 4

M : M_1		0	1	
	1	1 L 2	0 R 1	M_3
	2	1 0 2	1 0 3	↓
	3	0 0 4		M_0
	4	1 L 5	0 R 4	$[M_1, 3]$

Invităm cititorul să execute un calcul cu această maşină Turing:

(III). Ţinând seama că:

M_0:		0	1
	3		1 0 4

şi aplicând definiţia lui M (cu M_1) rezultă:

M : M_1		0	1	
	1	1 L 2	0 R 1	M_3
	2	1 0 2	1 0 3	↓
	3		1 0 4	M_0
	4	1 L 5	0 R 4	$[M_1, 3]$

Scriem calculul cu intrarea (2, 1): 0 1 0 0…

(2, 1): 0 1 0 0…

(3, 1): 0 0 0 0…

(2, 2): 0 0 1 0…

(2, 2): 0 1 1 0…

(2, 3): 0 1 1 0…

(2, 4): 0 1 1 0…

(3, 4): 0 0 1 0…

(4, 4): 0 0 0 0…

(3, 5): 0 0 0 1…

Din acest calcul se vede modul de funcţionare al lui M (cu M_1)

(IV) Din $M = M_3 \cup M_0$ (cu M ↓ M_3), unde:

M_0 :		0	1
	3	0 0 1	

rezultă:

M:		0	1
	1	1 L 2	0 R 1
	2	1 0 2	1 0 3
	3	0 0 1	

Un exerciţiu util va fi efectuerea unui calcul cu această maşină Turing.

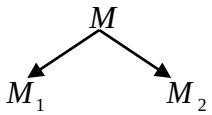
(V). Ţinând seama de definiţie rezultă:

M :		0	1
	1	1 L 2	0 R 1
	2	1 0 2	1 0 3
	3		1 0 1

(VI). Fie maşina Turing următoare:

M_2 :		0	1
	1	0 R 1	1 R 2

Vrem să calculăm maţina Turing:

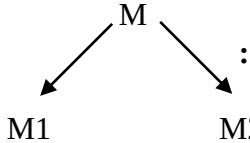


Atunci S(M) = {1}, deci v = 1, de unde rezultă:

M_0 :		0	1
	3	0 0 4	1 0 5

$[M_2,u+v]=$ $[M_2,4]$:		0	1
	5	0 R 5	1 R 6

Aplicând definiţia vom avea:



:		0	1
	1	1 L 2	0 R 1
	2	1 0 2	1 0 3
	3	0 0 4	1 0 5
	4	1 L 5	0 R 4
	5	0 R 5	1 R 6

Un calcul cu această maşină arată astfel:

(2, 1): 0 1 0 0…

(3, 1): 0 0 0 0…

(2, 2): 0 0 1 0…

(2, 2): 0 1 1 0…

(2, 3): 0 1 1 0…

(2, 5): 0 1 1 0…

(3, 6): 0 1 1 0…

Lema 2. Pentru orice $k \in N^+$ există o maşină Turing

copy k

astfel încât un calcul care are intrarea:

$$(2, 1): 0\ \underbrace{1...1}_{n_1-ori}\ 0\ \underbrace{1...1}_{n_2-ori}\ 0...0\ \underbrace{1...1}_{n_k-ori}\ 0...$$

va avea ieşirea de forma:

$$(n_1+3, ?): 0\ \underbrace{1...1}_{n_1-ori}\ 0\ \underbrace{1...1}_{n_2-ori}\ 0...0\ \underbrace{1...1}_{n_k-ori}\ 0\ \underbrace{1...1}_{n_1-ori}\ 0...$$

Demonstratie: Vom lucra numai cu cazul n = 2. Fie M_1 următoarea maşină Turing:

M_1 :

	0	1
1		1 R 2
2	0 L 3	0 R 2
3	0 L 3	1 R 4

M_1 are proprietatea următoare:

$$M_1 \mid \quad 0\ \underset{\uparrow}{1}\ 1^{n_1-1}\ 0\ 1^{n_2}\ 0... \rightarrow 0\ 1\ \underset{\uparrow}{0}\ 0^{n_1-1}\ 1^{n_2}\ 0...$$

Vom pune această proprietate în evidenţă pe un caz particular:

(2, 1): 0 1 1 0 1 1 0…

(3, 2): 0 1 1 0 1 1 0…

(4, 2): 0 1 0 0 1 1 0…

(3, 3): 0 1 0 0 1 1 0…

(2, 3): 0 1 0 0 1 1 0…

(3, 4): 0 1 0 0 1 1 0…

Fie M_2 maşina Turing următoare:

M_2 :

	0	1
1	0 R 1	1 R 2
2	1 R 3	0 R 2
3	0 L 4	1 R 3
4	0 L 5	1 L 4
5	0 L 6	0 L 5

6	0 L 1	1 R 7

Considerăm maşina Turing:

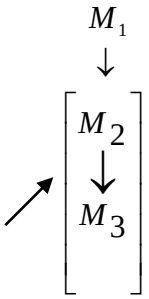
M_3 :

	0	1
1	0 R 2	
2	1 L 3	1 0 3

Vom lua

copy 2

ca fiind maşina Turing următoare:



Exerciţiu: Să se scrie în detaliu tabloul maşinii Turing

copy 2

şi să se efectueze cu această maşină Turing calculul care începe cu intrarea:

(2, 1): 0 1 1 0 1 1 0 …

Exerciţiu. Considerăm algoritmul $\Gamma = \Lambda \cup \{\alpha, \beta\}$ şi fie $\mathcal{F}$ algoritmul Markov dat de următorul tablou de substituţie:

$\alpha\alpha \to \beta$ (I)
$\alpha\xi_i\xi_j \to \xi_j\alpha\xi_i$ (II)
$\beta\alpha \to \beta$ (III)
$\beta\xi_i \to \xi_j\beta$ (IV)
$\beta \to \bullet$ (V)
$\to \alpha$ (VI)

Să se aplice acest algoritm cuvântului $P = \xi_{i_1}\xi_{i_2}...\xi_{i_p}$

Rezolvare.
Avem următorul calcul:

$\xi_{i_1}\xi_{i_2}...\xi_{i_p} \to \alpha\underline{\xi_{i_1}\xi_{i_2}}...\xi_{i_p}$ (VI)
$\to \xi_{i_2}\underline{\alpha\xi_{i_1}\xi_{i_3}}...\xi_{i_p}$ (II)
$\to \xi_{i_2}\xi_{i_3}\alpha\xi_{i_4}...\xi_{i_p}$ (II)
$\Rightarrow \xi_{i_2}\xi_{i_3}...\xi_{i_p}\alpha\xi_{i_1}$ (II)
$\to \alpha\xi_{i_2}\xi_{i_3}...\xi_{i_p}\alpha\xi_{i_1}$ (VI)
$\Rightarrow \alpha\xi_{i_p}\alpha\xi_{i_{p-1}}...\xi_{i_2}\alpha\xi_{i_1}$ (II, VI)
$\to \underline{\alpha\alpha}\xi_{i_p}\alpha\xi_{i_{p-1}}...\xi_{i_2}\alpha\xi_{i_1}$ (VI)
$\to \underline{\beta\xi_{i_p}}\alpha\xi_{i_{p-1}}...\xi_{i_2}\alpha\xi_{i_1}$ (I)
$\to \xi_{i_p}\underline{\beta\alpha}\xi_{i_{p-1}}...\xi_{i_2}\alpha\xi_{i_1}$ (VI)
$\to \xi_{i_p}\underline{\beta\xi_{i_{p-1}}}...\xi_{i_2}\alpha\xi_{i_1}$ (III)
$\Rightarrow \xi_{i_p}\xi_{i_{p-1}}...\xi_{i_2}\xi_{i_1}\underline{\beta}$ (III, IV)
$\to \bullet\xi_{i_p}\xi_{i_{p-1}}...\xi_{i_2}\xi_{i_1}$ (V)

A rezultat deci $\mathcal{F}(P) = \xi_{i_p} \xi_{i_{p-1}} ... \xi_{i_2} \xi_{i_1}$.

Observaţie: Algoritmul $\mathcal{F}$ scrie cuvintele lui $\mathcal{M}(\Lambda)$ în ordine inversă.

TESTE DE AUTOEVALUARE ŞI TEME DE CONTROL

Testul nr. 1

1. Considerăm maşina Turing M = (d, p, s) definită astfel:
D = { (1, 1), (0, 2), (1, 2), (0, 3), (1, 4)},
d(1, l) = l, p(1, l) = l, s(1, l) = 2,
d(0, 1) = 0, p(0, 2) = 0, s(0, 2) = 2,
d(1, 2) = l, p(1, 1) = 1, s(1, 2) = 1,
d(0, 3) = 0, p(0, 3) = -1, s(0, 3) = 3,
d(1, 4) = 1, p(1, 4) = 0, s(1, 4) = 4.
a) Să se scrie sub formă de tablou această maşină Turing.
b) Să se scrie calculul cu această maşină Turing care ţncepe cu:

(2,1): 0 1 1 0 0 0....
↑

2. Fie alfabetul $\Lambda = \{A, B, C, ..., X, Y, Z\}$ un şi $\mathcal{F}$ un algoritm Markov pe Λ. Fie $\Gamma = \Lambda \cup \{\alpha, \beta, \gamma\}$ şi fie $\mathcal{F}$ algoritmul Markov dat de următorul tablou de substituţie:

$\xi_i \xi_j \beta \rightarrow \xi_j \beta \xi_i$ (I)
$\alpha \xi_i \rightarrow \xi_i \beta \xi_i \alpha$ (II)
$\beta \rightarrow \gamma$ (III)
$\gamma \rightarrow$ (IV)
$\alpha \rightarrow \bullet$ (V)
$\rightarrow \alpha$ (VI)

unde $\xi_i \in \Lambda$ (deci ξ_i este una din literele $A, B, C, ..., X, Y, Z$).

Aplicaţi algoritmul $\mathcal{F}$ definit pe alfabetul Γ cuvântului $P = MA$ şi deduceţi cine este $\mathcal{F}(P)$.

Temă de control

1. Se dă maşina Turing:

	0	1
1	1 R 2	
2	0 R 2	1 R 3
3	0 L 3	1 R 4
4	1 R 5	0 L 1

a) Să se scrie explicit această maşină Turing.
b) Să se determine calculul cu intrarea:

(2,3): 0 0 1 1 0 0 0....
↑

2. Considerăm maşina Turing următoare:

		0	1
M:	1	1 L 2	0 R 1
	2	1 0 2	

a)) Să se scrie calculul cu intrarea:

(2,1): 0 1 1 0 1 1 0....
↑

b) Să se determine $\overset{M}{\downarrow}_{M}$ şi să se scrie calculul cu această maşină Turing care începe cu intrarea dată la a).

3. Fie alfabetul $\Lambda = \{A, B, C, ..., X, Y, Z, -\}$ şi $\mathcal{F}$ un algoritm Markov pe Λ. Fie $\Gamma = \Lambda \cup \{\alpha, \beta\}$ şi fie $\mathcal{F}$ algoritmul Markov dat de următorul tablou de substituţie:

$\alpha\alpha \to \beta$ (I)
$\alpha\xi_i\xi_j \to \xi_j\alpha\xi_i$ (II)
$\beta\alpha \to \beta$ (III)
$\beta\xi_i \to \xi_j\beta$ (IV)
$\beta \to \bullet$ (V)
$\to \alpha$ (VI)

unde $\xi_i \in \Lambda$ (deci ξ_i este una din literele $A, B, C, ..., X, Y, Z,$ sau -).
Aplicaţi algoritmul $\mathcal{F}$ definit pe alfabetul Γ cuvântului $P = ERAM - LEC - NAFETS$ şi deduceţi cine este $\mathcal{F}$(P).

BIBLIOGRAFIE RECOMANDATĂ LA UNITATEA DE ÎNVĂŢARE NR.4

1. G. Georgescu – *Elemente de logică matematică*, Editura Academiei Tehnice Militare, Bucureşti, 1978
2. G. Georgescu – *Teoria algoritmilor*, Editura Academiei Tehnice Militare, Bucureşti, 1980