Capitolul 2

Logica propozițională

(Calculul propozițional)

Manualele de *Logică* şi *Algebră* ([BIE], [DID]) pot fi privite ca o *introducere (firavă) în logica formală*. Şi în alte manuale de matematică (şi nu numai), sunt prezente frecvent noțiuni ca *afirmație, axiomă, teoremă, raționament, demonstrație*, etc. Aceste noțiuni sunt însă **descrise** sau concepute/receptate/folosite **la modul informal**: o *afirmație* este orice propoziție (frază) care poate căpăta o unică valoare de adevăr (**a** – adevărat, **f** – fals); o *axiomă* este o afirmație care se acceptă a fi adevărată fără a se cere o demonstrație a ei; o *teoremă* este o afirmație (presupusă a fi adevărată) care se obține (din axiome sau teoreme deja acceptate) printr-o demonstrație (formală), numită şi raționament; o *demonstrație (formală)* este transpunerea într-o formă (mai) exactă a unui raționament; un *raționament* este o succesiune (finită) de aplicări ale unor inferențe (reguli de deducție); o *regulă de deducție (inferență)* are forma: premize/concluzii (atât *premizele* cât şi *concluziile* sunt afirmații, ideea fiind aceea că regulile sunt astfel construite încât dacă premizele sunt adevărate atunci şi concluziile sunt adevărate; se mai spune că inferențele sunt în acest caz *valide* sau *corecte*), etc. De altfel, acesta este modul principal prin care se obțin (constructiv) în ştiințele exacte *noțiuni noi* (utilizând *definițiile*) şi *afirmații* (adevărate) *noi* (utilizând *raționamentele*). Din punctul de vedere al logicii filozofice, o noțiune este complet caracterizată de

conținut (element din structura noțiunii alcătuit din mulțimea **proprietăților obiectelor** care formează **sfera noțiunii**) sau **sferă** (element din structura noțiunii alcătuit din mulțimea **obiectelor** ale căror **proprietăți** formează **conținutul noțiunii**). O definiție ar avea astfel rolul de a delimita precis sfera (conținutul) noțiunii. Definirea unei noțiuni noi înseamnă **delimitarea unei noi sfere (sau a unui nou conținut)**, ceea ce se poate face de exemplu (există şi alte tipuri generale de definiții) prin precizarea unei sfere vechi (care caracterizează complet o noțiune anterior definită, numită **gen proxim**) şi a unei mulțimi de proprietăți suplimentare (care nu fac parte din conținutul vechii noțiuni, dar care – împreună cu acesta – vor alcătui noul conținut), numită **diferență specifică**: un *paralelogram* este un *patrulater convex* care are *două laturi paralele şi egale*; un *romb* este un *paralelogram* cu *toate laturile egale*; un *pătrat* este un *romb* având *un unghi de 90*°, ş. a. m. d. În acest mod, „mergând invers", procesul de definire a unor noțiuni ar deveni infinit dacă nu am accepta existența unor **noțiuni primare** (pentru o mai bună înțelegere se poate recurge şi la reprezentări grafice cum ar fi diagramele Venn-Euler – [ȚIP]). Noțiunile primare nu mai sunt definite prin schema „gen proxim şi diferență specifică" ci sunt doar **descrise** cu ajutorul unor elemente considerate a fi suficiente pentru **delimitarea exactă** a sferei curente de sfera altor noțiuni (asemenea definiții sunt cunoscute şi sub numele de **definiții operaționale**): o *mulțime* este o colecție de obiecte distincte două câte două; un *punct* este ceea ce se obține prin apăsarea unui vârf de creion pe o foaie de hârtie, etc. Un proces similar are loc şi în cazul conceptelor de **axiomă** (în „rolul" noțiunilor primare), **teoremă** (în

„rolul” noțiunilor noi), **regulă de inferență** (în „rolul” diferenței specifice), acceptarea axiomelor ca fiind „advărate fără demonstrație” având desigur scopul de **a evita raționamentele infinite**. Aşa cum în momentul definirii unei noțiuni (noi) trebuie să fim atenți ca sfera acesteia să fie **nevidă** şi (în general) **distinctă** de sferele unor noțiuni deja existente (chiar definite operațional), în cazul raționamentelor este de dorit ca **axiomele** să reprezinte „cu certitudine” **afirmații adevărate**, iar **inferențele să fie valide** (inferențele trebuie să fie valide pentru a avea raționamente corecte, adică formate numai din afirmații adevărate). Din păcate, datorită lipsei unei sintaxe clare pentru conceptul de afirmație (lipsei definițiilor formale în general), precum şi datorită „amalgamării” considerațiilor de natură sintactică şi semantică, eşafodajul anterior este destul de şubred putând conduce la apariția unor paradoxuri de gândire sau la acceptarea unor „adevăruri” hilare. Prima parte a capitolului este destinată unei scurte treceri în revistă a unor asemenea anomalii şi introducerii primelor elemente de logică (*informatică*) formală.

§1. Logica, parte a filozofiei

Ambiguitățile permise de limbajul natural, acceptarea utilizării unor noțiuni primare sau a unor axiome având conținut ambiguu în raționamente complexe, tratarea simultană a unor probleme de natură sintactică împreună cu altele care implică semantica, au creat de-a lungul timpului numeroase confuzii şi interpretări greşite, „bruind” comunicarea inter-umană. Un prim tip de asemenea confuzii, cunoscute sub numele de **paradoxuri logice**, sunt deja clasificate, împărțite pe categorii. Nu este simplu să dăm o definiție unanim acceptată (de altfel, B. Russell a împărțit paradoxurile în şapte categorii, având definiții practic diferite). Pentru unii, un *paradox* este *o afirmație care pare să se autocontrazică, sau poate conduce la o situație care contrazice bunul simț*. Mai general, este *orice afirmație surprinzătoare, alambicată, contrară intuiției*, sau, *o argumentație aparent solidă, corectă, dar care conduce la o contradicție*. Pentru alții, este *o propoziție care îşi afirmă propria falsitate*, sau, *un argument care conduce la o concluzie contradictorie deşi începe cu nişte premize acceptabile şi se foloseşte o deducție validă*. Oricum, se acceptă faptul că **un paradox nu înseamnă acelaşi lucru cu o contradicție**. Astfel, afirmația „Această cămaşă este albastră şi această cămaşă nu este albastră" este o contradicție, dar *un paradox va apare atunci când o persoană face o anumită presupunere şi apoi, urmând o argumentație logică, ajunge la contrariul presupunerii inițiale*. „Nu spun niciodată adevărul" este considerat un paradox (***al mincinosului***), deoarece dacă presupunem că propoziția este adevărată atunci rezultă imediat că ea este falsă si reciproc. Mai sus este vorba despre o clasă mai simplă de paradoxuri (numite şi *semantice*). Practic, ele ar putea fi „rezolvate" daca sunt eliminate complet din logica clasică, deoarece pot fi considerate ca afirmații cărora nu li poate ataşa o unică valoare de adevăr (contradicțiilor nu li se poate practic ataşa nici una!). Un paradox mai complicat este paradoxul lui ***B. Russell, legat de teoria mulțimilor*** : „Dacă R este mulțimea tuturor mulțimilor care nu se conțin pe ele însele, atunci R se conține pe sine însăşi ca element?".

Imediat se obține că dacă răspunsul este „DA”, atunci R nu se conține pe ea însăşi şi dacă răspunsul este „NU”, atunci R se conține. Contradicția provine aici din acceptarea ***axiomei înțelegerii***: ***„Dacă P este o proprietate (relație, predicat), atunci M = {x | P(x)} este o mulțime”*** (paradoxul precedent se obține luînd P(x): „x nu este element al lui x”). Matematic vorbind, paradoxul dispare dacă se renunță la axioma înțelegerii (mai exact, **M** de mai sus nu este o mulțime, ci o *clasă*). Un alt paradox, cunoscut încă din antichitate, este paradoxul lui ***Ahile şi broasca țestoasă***, atribuit lui ***Zenon***: „Ahile şi o broască țestoasă se iau la întrecere într-o alergare de viteză, Ahile aflându-se inițial într-un punct A şi broasca în fața sa, la o distanță a, într-un punct B, dar începând să se deplaseze amândoi în acelaşi moment şi în aceeaşi direcție. Afirmație: Ahile nu va ajunge din urmă broasca (chiar dacă broasca ar avea...viteza 0)”. Putem „demonstra” afirmația raționând astfel: fie C mijlocul distanței dintre A şi B; pentru a ajunge în B, Ahile trebuie să ajungă întâi în C; fie acum D mijlocul distanței dintre A şi C, etc. Cum mulțimea numerelor reale este densă, mereu mijlocul unui segment de lungime diferită de zero va genera alte două segmente de lungime nenulă, astfel încât Ahile nu va ajunge niciodată broasca. Acest tip de paradox se numeşte şi *aporie*, contradicția provenind, în cazul nostru, din utilizarea unui raționament corect intr-un „mediu” necorespunzător (drumurile, în legătură cu deplasarea unor ființe, nu pot fi considerate drept reprezentări ale axei reale). Deşi nu sunt ele însele „absurdități”, **silogismele** reprezintă o altă sursă generoasă de confuzii. *Inferențele*, adică paşii elementari (considerați a fi indivizibili) ai unui raționament, reprezintă forme logice complexe. Aceste raționamente „elementare” se împart în ***deductive*** şi ***inductive***, iar cele mai simple inferențe sunt cele ***imediate, cu propoziții categorice*** (fiind formate din două asemenea propoziții: ***o premiză***, şi ***o concluzie)***. Silogismul ***este tipul fundamental de inferență deductivă mediată alcătuită din exact trei propoziții categorice: două premize, dintre care una majoră şi alta minoră, precum şi o concluzie***. Silogismele se pot de altfel împărți în ***ipotetice, categorice, disjunctive***, etc. (nu insistăm asupra altor detalii). Un exemplu de silogism (categoric, corect) este:

> *Premiza majoră*: Toate elementele transuranice sunt radioactive.
> *Premiza minoră*: Plutoniul este element transuranic.
> *Concluzia*: Plutoniul este radioactiv.

Pentru a folosi însă doar silogisme corecte (valide), este necesar un studiu mai aprofundat al acestora. În caz contrar, putem ajunge, ca şi în cazul paradoxurilor, să acceptăm nişte aberații drept propoziții adevărate. De exemplu:

Albă este adjectiv

Zăpada este albă

Zăpada este adjectiv

Greşeala în silogismul anterior constă în aceea că nu se ține cont de o ***lege a silogismelor***, care stipulează că ***într-un silogism valid există trei şi numai trei termeni lingvistici distincți***. Din păcate însă, în limba română (dar nu numai) un acelaşi cuvânt (sau grup de cuvinte) poate „materializa” mai mult decât o singură noțiune. Astfel, deşi în exemplul nostru s-ar părea că avem exact trei termeni distincți (albă, adjectiv, zăpada), în realitate *avem patru*: în premiza majoră cuvântul alb materializează un element al limbajului (*o parte de vorbire*), iar în premiza minoră el redă o *însuşire* (care este caracteristică şi zăpezii). Neconcordanțele pot fi eliminate dacă legile de genul amintit ar putea fi „prinse” în forma sintactică exactă a silogismului. Profităm de prilej pentru a aminti şi câteva idei legate de **inferențele (raționamentele) deductive cu propoziții compuse**. Deşi acestea nu sunt automat generatoare de ambiguități/aberații, folosirea incorectă a **implicației logice** în cadrul lor poate produce neplăceri. Mai întâi, să observăm că putem considera că am definit structural întreaga (sau măcar o parte importantă a sa) mulțime de afirmații pe care le manipulăm în limbaj natural, în modul sugerat de logica clasică: pornim de la anumite afirmații (**Baza** - *propoziții elementare*) şi apoi (**Pas constructiv**) formăm propoziții noi (*fraze, propoziții compuse*), din propoziții vechi cu ajutorul unor operatori (*conectori*), cum ar fi *sau*, *şi*, *negația*, *implicația* (*dacă* ... *atunci* ...), *echivalența* (*dacă* ... *atunci* ... *şi reciproc*). Notând cu A şi B două propoziții oarecare (elementare sau compuse), putem forma **propozițiile compuse** (de acum înainte, **@** va nota „egal prin definiție/notație/convenție”): C **@** A sau B (simbolic: $A \vee B$); D **@** A şi B (simbolic: $A \wedge B$); E **@** non A (simbolic: $\rceil A$); F **@** dacă A atunci B (simbolic: $A \rightarrow B$; A se numeşte uneori *ipoteză* sau *antecedent* iar B – *concluzie* sau *consecvent*); G @ dacă A atunci B şi reciproc (sau, A dacă şi numai dacă B, sau A atunci şi numai atunci când B; simbolic: A $\leftrightarrow$ B). Cum A şi B pot lua fiecare doar valorile **a - adevărat** sau **f – fals** (desigur...nu simultan), la fel se va întâmpla şi cu propoziţiile compuse. Astfel, C va fi **a** atunci şi numai atunci când măcar una dintre A şi B este **a**, D va fi **a** atunci şi numai atunci când atât A cât şi B sunt **a**, E va fi **a** atunci şi numai atunci când A va fi **f**, *F va fi **f** atunci şi numai atunci când A este **a** şi B este **f***. În sfârşit, G va fi **a** atunci şi numai atunci când A şi B sunt simultan **a** sau simultan **f**. Ca o consecinţă, **o implicaţie va fi adevărată dacă ipoteza este falsă, indiferent de valoarea de adevăr a concluziei**. Acum ne putem referi în mod explicit şi la inferenţe cu propoziţii compuse (se presupune că silogismele utilizează doar propoziţii simple), cele conţinând implicaţia fiind des utilizate. Cele mai simple inferenţe de acest tip sunt cele care conţin *două premize* şi o *concluzie*, dintre ele distingându-se cele *ipotetico-categorice* (*prima premiză este o implicaţie iar cea de-a doua constă fie din antecedentul sau negaţia antecedentului, fie din consecventul sau negaţia consecventului implicaţiei respective*, conform [BIE]). *Schemele valide* care se folosesc în raţionamente sunt astfel de forma:

$$\frac{\begin{array}{l} A \to B \\ A \end{array}}{B}$$

modus ponendo-ponens (pe scurt, **modus ponens**)

sau

$$\frac{\begin{array}{l} A \rightarrow B \\ \rceil B \end{array}}{\rceil A}$$

modus tollendo-tollens (pe scurt, **modus tollens**)

Validitatea (reamintim: dacă ipotezele sunt adevărate, atunci şi concluzia trebuie să fie adevarată) schemelor modus ponens (*modul afirmativ*) şi modus tollens (*modul negativ*) rezultă imediat din definiția implicației. Oprindu-ne la modus ponens, am putea spune că acesta poate fi reformulat în: **din A (adevărată) şi A ® B (adevărată) deducem (că) B (adevărată).** Pe scurt, vom nota acest lucru prin $A \Rightarrow B$. Următorul exemplu este edificator pentru greşelile care se pot face fie din necunoaşterea definiției reale a implicației, fie din confundarea lui $A \rightarrow B$ (formulă în *limbajul de bază*) cu $A \Rightarrow B$ (formulă în *metalimbaj*, care sugerează deducerea lui B, pornind de la A şi folosind un raționament).

Exemplu. Să considerăm funcția $f : \mathbf{R} \rightarrow \mathbf{R}$, dată prin:

$$f(x)=\begin{cases} x, & \text{dacă } x \leq 0 \\ x^2+1, & \text{dacă } x>0 \end{cases}$$

Să se arate că **f este injectivă**.

Conform uneia dintre definițiile cunoscute, trebuie să arătăm că *pentru fiecare* x_1, x_2 Î **R**, *avem: dacă* $f(x_1) = f(x_2)$ *atunci* $x_1 = x_2$. Anticipând notatiile din **Capitolul 3** şi presupunând cunoscută (cel puțin la nivel informal) semnificația *cuanficatorilor*, putem scrie acest lucru sub forma condensată $(\forall x_1, x_2 \in R)(f(x_1) = f(x_2) \rightarrow x_1 = x_2)$. Există următoarele posibilități:

a) **x_1, x_2 £ 0**. Atunci $f(x_1) = x_1$ şi $f(x_2) = x_2$. Prin urmare $f(x_1) = f(x_2)$ chiar coincide cu $x_1 = x_2$ şi deci implicația $f(x_1) = f(x_2) \rightarrow x_1 = x_2$ este adevărată.

b) **$x_1, x_2 > 0$**. Atunci $f(x_1) = x_1^2 + 1$ şi $f(x_2) = x_2^2 + 1$. Prin urmare, $f(x_1) = f(x_2)$ înseamnă $x_1^2 + 1 = x_2^2 + 1$, ceea ce se întâmplă atunci şi numai atunci când $(x_1 - x_2)(x_1 + x_2) = 0$. Deoarece variabilele sunt pozitive, ultima egalitate este echivalentă cu $x_1 - x_2 = 0$, deci cu $x_1 = x_2$. Am arătat de fapt că *avem $f(x_1) = f(x_2)$ dacă şi numai dacă $x_1 = x_2$* ceea ce se poate scrie simbolic (**în metalimbaj!**) $f(x_1) = f(x_2) \Leftrightarrow x_1 = x_2$. În consecință, la fel ca la punctul precedent, implicația cerută $f(x_1) = f(x_2) \rightarrow x_1 = x_2$ este la rândul ei adevărată (este adevărată chiar echivalența $f(x_1) = f(x_2) \leftrightarrow x_1 = x_2$), deoarece este clar că dacă antecedentul $f(x_1) = f(x_2)$ este adevărat, atunci şi consecventul $x_1 = x_2$ este adevărat (de fapt, şi reciproc).

c) **x_1 £ 0, $x_2 > 0$**. În acest caz $f(x_1) = x_1$ şi $f(x_2) = x_2^2 + 1$. Atunci $f(x_1) = f(x_2)$ înseamnă $x_1 = x_2^2 + 1$ şi implicația pe care trebuie să o arătăm devine $x_1 = x_2^2 + 1 \rightarrow x_1 = x_2$, şi aceasta **pentru fiecare** $x_1 \leq 0$ şi $x_2 > 0$. Nu mai putem proceda la fel ca în situațiile anterioare, deoarece din $x_1 = x_2^2 + 1$ nu se poate *deduce* $x_1 = x_2$. Totuşi, **implicația în cauză din limbajul de bază este *adevărată*, deoarece antecedentul ei este *fals*.** Într-adevăr, oricare ar fi $x_1 \leq 0$ şi $x_2 > 0$, în egalitatea $x_1 = x_2^2 + 1$, membrul stâng este nepozitiv iar membrul drept este pozitiv, ceea ce face ca relația să devină imposibilă în contextul dat. ■

În finalul paragrafului, pentru a introduce şi o notă optimistă, prezentăm una dintre cele mai „mari” demonstrații cunoscute în literatura „științifică” a afirmației *Oamenii de știință nu vor câştiga niciodată la fel de mulți bani ca directorii executivi ai unor companii de succes*. În acest scop vom porni de la postulatele (axiomele) *Cunoaşterea înseamnă putere* şi *Timpul înseamnă bani*, pe care le vom folosi sub forma prescurtată: cunoaştere = putere şi respectiv timp = bani. Ca inferențe, le vom utiliza pe cele mai simple (imediate, cu afirmații categorice), la care adăugăm altele la fel de simple, cunoscute din matematica elementară. Plecăm astfel de la axioma suplimentară:

$$\frac{\text{muncă}}{\text{timp}} = \text{putere}$$

Folosind axiomele inițiale şi proprietățile relației de egalitate, printr-o inferență simplă deducem:

$$\frac{\text{muncă}}{\text{bani}} = \text{cunoaştere}$$

Aplicând acum o proprietate a proporțiilor, găsim:

$$\frac{\text{muncă}}{\text{cunoaştere}} = \text{bani}$$

Cititorul poate trage singur concluzia care se impune pentru situația în care cunoaştere se apropie de (tinde la) valoarea zero.

Ca o concluzie, situațiile neplăcute descrise anterior trebuie evitate sau eliminate. Acest lucru se poate face doar prin „translatarea” părților de limbaj într-un mecanism formal bine pus la punct, pe care-l vom descrie începând cu secțiunea/paragraful următoare/următor. Enunțul unora dintre exercițiile care urmează este reluat în **§2.10.**

Exercițiul 2.1*. O* ***teoremă****, în sensul matematicii de liceu, are şi ea ipoteze şi concluzii. Scrieți simbolic forma generală a unei teoreme (****directe****), utilizând propoziții elementare (****variabile propoziționale****) şi* ***conectori logic****i. Scrieți apoi* ***teorema reciprocă****,* ***contrara teoremei directe*** *şi* ***contrara reciprocei****. Există vreo legătură între acestea, în ceea ce priveşte valoarea lor de adevăr? Dați un exemplu de* ***teoremă de caracterizare*** *(A dacă şi numai dacă B). Puteți specifica altfel rezultatul exprimat de teoremă, astfel încât să fie – separat - puse în evidență* ***condiția necesară*** *şi* ***condiția suficientă****?*

Exercițiul 2.2*. Să considerăm definiția limitei unui şir dat de numere reale, având ca valoare un număr real dat, definiție exprimată cu ajutorul vecinătăților care sunt intervale simetrice față de punctul considerat. Să se exprime simbolic (în sensul matematicii de liceu, folosind şi* ***cuantificatorii****) această definiție şi să se nege* ***formula*** *astfel găsită.*

Exercițiul 2.3*. Exprimați simbolic, ca o* ***formulă*** *– în sensul exercițiilor anterioare – propoziția* **Dacă mi-e sete, beau apă***. Negați formula şi apoi rescrieți rezultatul în limbaj natural. Dacă ați fi negat direct propoziția inițială, ați fi obținut acelaşi lucru?*

În restul capitolului, câteva dintre concepte/rezultate/exemple sunt din [MAS1] (trebuie să precizăm că o parte dintre acestea provin, la origine, din [SCH]).

§2. Sintaxa logicii propoziționale

Vom trece direct la prezentarea sintaxei formale a logicii propoziționale (calculului propozițional). Logica propozițională, aşa cum am sugerat deja, va fi numele unei mulțimi de formule (*propoziționale*), notată $\mathbf{LP}_L$ sau, prescurtat, **LP** şi definită structural în cele ce urmează.

Definiția 2. 1. Fie o mulțime numărabilă de *variabile propoziționale* (*formule elementare*, *formule atomice pozitive*, *atomi pozitivi*), $A = \{A_1, A_2, \ldots\}$. Fie, de asemenea, $\mathbf{C} = \{\rceil, \vee, \wedge\}$ mulțimea *conectorilor/conectivelor logici/logice* non (**negația**), sau (**disjuncția**), respectiv şi (**conjuncția**) şi $\mathbf{P} = \{ (\ ,\ ) \}$ mulțimea *parantezelor* (rotunde). *Formulele* (elementele lui **LP**) vor fi „cuvinte” (*expresii bine formate*) peste *alfabetul* $L = A \cup \mathbf{C} \cup \mathbf{P}$:

Baza (*formulele elementare sunt formule*). $A \subseteq \mathbf{LP}$.

Pas constructiv (*obținere formule noi din formule vechi*).

(i) Dacă $F \in \mathbf{LP}$ atunci $(\rceil F) \in \mathbf{LP}$.

(ii) Dacă $F_1, F_2 \in \mathbf{LP}$ atunci $(F_1 \vee F_2) \in \mathbf{LP}$.

(iii) Dacă $F_1, F_2 \in \mathbf{LP}$ atunci $(F_1 \wedge F_2) \in \mathbf{LP}$.

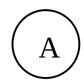
(iv) Dacă $F \in \mathbf{LP}$ atunci $(F) \in \mathbf{LP}$.

(v) *Nimic altceva nu este formulă*. ■

Putem privi o formulă F ca fiind ***reprezentată*** de un arbore binar (***arborele ataşat lui F, notat Arb(F)***), în modul următor (procedăm structural, conform definiției lui **LP**):

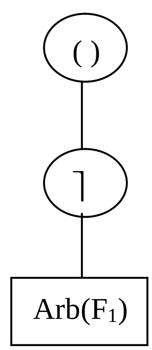
Definiția 2.2.

Baza. F = A ∈ $\mathcal{A}$. Atunci ***arborele ataşat*** lui F (sau, ***arborele care reprezintă*** F), este:

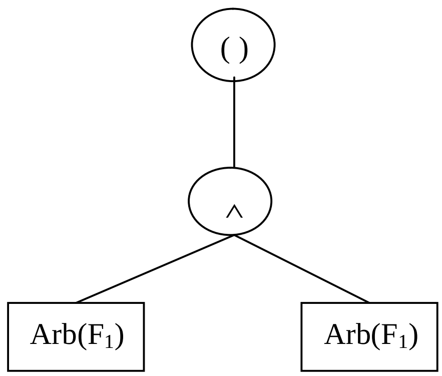


Pas constructiv.

(i) Fie F = ($\rceil$ F_1) şi să presupunem că se cunoaşte arborele ataşat lui F_1, Arb(F_1). Atunci, arborele ataşat lui F va fi (ceva similar se obține pentru (iv), adică pentru cazul F = (F_1)):



(iii) Fie F = ($F_1 \wedge F_2$) şi să presupunem că se cunosc atât arborele ataşat lui F_1 cât şi arborele ataşat lui F_2 , adică Arb(F_1) respectiv Arb(F_2). Atunci arborele ataşat lui F va fi (pentru F = ($F_1 \vee F_2$) se obține ceva similar):



■

Deşi au un rol pur sintactic, neschimbând cu nimic semantica formulelor în care apar, parantezele rotunde au fost – din anumite motive tehnice – privite mai sus ca un *operator pre&post-fixat*. Dacă introducem o *ordine stânga-dreapta* pe mulțimea succesorilor imediați ai fiecarui nod (implicit, pentru o formulă este valabilă ordinea de scriere a „literelor” în cuvântul respectiv, exceptând paranteza închisă „)”, care are acelaşi „număr de ordine” cu paranteza deschisă „(” *corespunzătoare*), atunci se observă că fiecărei formule îi corespunde un arbore ataşat unic şi fiecărui *arbore ordonat* G (cu nodurile etichetate cu elemente din *L*) îi corespunde o unică formulă din **LP** (pentru care G este arborele ataşat). Definim structural şi *mulțimea subformulelor* oricărei formule date F (notată *subf(F)*). *Admitem implicit faptul că F’ Î subf(F) dacă şi numai dacă F’ este subcuvânt al lui F şi F’ Î* **LP** (cu alte cuvinte, F_1 şi F_2, în cele ce urmează, sunt tot formule).

Definiția 2.3.

Baza. F = A ∈ $\mathcal{A}$. Atunci subf(F) = {A}.

Pas constructiv.

(i) F = ($\rceil$ F_1). Atunci subf(F) = subf(F_1) ∪ { ($\rceil$ F_1) }.

(ii) F = ($F_1 \wedge F_2$). Atunci subf(F) = subf(F_1) ∪ subf(F_2) ∪ { ($F_1 \wedge F_2$) }.

(iii) Analog cu (ii) pentru cazul F = ($F_1 \vee F_2$) (înlocuind peste tot, simultan, ∧ cu ∨).

(iv) F = (F_1). Atunci subf(F) = subf(F_1) ∪ { (F_1) } ■

Observație. Nu se admit alte posibilități pentru scrierea unei formule, decât cele fixate prin **Definiția 2.1**. Există de altfel un algoritm care rezolvă *problema de decizie*: *Dat orice cuvânt w Î L^* (adică orice secvență finită de caractere din L) să se decidă dacă w Î* **LP**. Conform [JUC] de exemplu, notația L^* (algebric, L^* este monoidul liber generat de L) se explică prin aceea că mulțimea cuvintelor (secvențelor finite de simboluri) aparținând unui alfabet cel mult numărabil formează un *monoid* față de operația de *concatenare* (de juxtapunere a literelor/cuvintelor). *Elementul neutru*, este cuvântul fără nici o literă (*cuvântul vid*) și este notat cu e. Algoritmul menționat se termină pentru fiecare intrare *w Î L^**, cu răspunsul (ieşirea) „DA” dacă w ∈ **LP** şi „NU” dacă w ∉ **LP**. O **problemă de decizie** are doar alternativa de răspuns „DA/NU” şi aici este un caz particular al *problemei de apartenență pentru un limbaj de tip 2*. Revenind, $A_1 \vee A_2$, nu este formulă pentru că nu are parantezele necesare (nu putem vorbi de subf($A_1 \vee A_2$) pentru că $A_1 \vee A_2$ nu este formulă). Dar, la fel ca şi în cazul cunoscut al expresiilor aritmetice care conțin variabile, constante şi operatorii „**–**” (având şi sensul de *opus*), „+”, „•” şi „/”, putem accepta convenția de a prescurta scrierea unor expresii (formule, cuvinte) prin eliminarea unor paranteze (sau chiar pe toate). Acest lucru se poate face prin atribuirea de *priorități operatorilor*, apoi bazându-ne pe faptul că *aritatea lor (numărul de argumente) este cunoscută*, precum şi pe unele proprietăți cum ar fi comutativitatea, asociativitatea sau distributivitatea. Prioritățile standard sunt: 0 – pentru ⌉, 1 – pentru $\wedge$, 2 – pentru $\vee$. Tot o convenție este şi aceea de a folosi şi alte nume pentru formulele atomice, înafara celor admise deja prin faptul că sunt simboluri desemnate a face parte din $\mathcal{A}$. *În general vom utiliza pentru acestea litere mari de la începutul alfabetului latin (A, B, C, ..., cu sau fără indici).* Invers, putem *adăuga în orice formulă „bine formată” cupluri de paranteze corespondente* (la fel cum le-am şi eliminat), pentru a îmbunătăți receptarea corectă a sintaxei şi fără a schimba semnificația formulei în cauză. Acest lucru este permis de altfel prin (iv), **Definiția 2.1**. ■

Exercițiul 2.4. *Fie formula F = ((ù A) Ú (B Ù C)). Construiți arborele ataşat (verificând în acest mod şi faptul că într-adevăr F Î* **LP***). Eliminați parantezele şi stabiliți o prioritate a operatorilor care intervin, astfel încât semnificația intuitivă a noii secvențe de caractere să nu difere de semnificația inițială (pentru a construi pe F,* *se consideră întâi afirmațiile elementare A, B, C; se consideră apoi negația lui A, notată, să spunem, A' și conjuncția lui B cu C, notată D; în sfârșit, se consideră disjuncția lui A' cu D).*

Vom face şi alte câteva prescurtări sintactice, justificate de altfel şi de anumite considerente semantice care vor fi prezentate ulterior:

- $((\rceil F) \vee G)$ se va nota cu $(F \rightarrow G)$.
- Pentru $(((\rceil F) \vee G) \wedge ((\rceil G) \vee F))$ folosim $(F \leftrightarrow G)$ sau $((F \rightarrow G) \wedge (G \rightarrow F))$.
- $\bigwedge_{i=1}^{n} F_i$ este o prescurtare pentru $F_1 \wedge F_2 \wedge ... \wedge F_n$.
- $\bigvee_{i=1}^{n} F_i$ este prescurtarea lui $F_1 \vee F_2 \vee ... \vee F_n$.

Simbolurile $\rightarrow$ şi $\leftrightarrow$ (numite după cum ştim implicație, respectiv echivalență) pot fi considerate ca şi cum ar fi fost introduse de la bun început în mulțimea de conectori **C** (dacă am fi procedat astfel de la bun început, s-ar fi complicat atât unele lucruri de natură sintactică cum ar fi definițiile constructive, prioritățile, etc., cât şi definiția semanticii **LP**, care urmează). Vom numi **literal** o variabilă propozițională sau negația sa. $A \in \mathcal{A}$ se va numi şi **literal pozitiv** iar orice element de forma $\rceil A$, $A \in \mathcal{A}$ va fi un **literal negativ** (vom nota $\overline{\mathcal{A}} = \{\rceil A_1, \rceil A_2, \dots\}$). Dacă L este un literal, atunci **complementarul** său, $\bar{L}$, va nota literalul $\rceil A$, dacă $L = A \in \mathcal{A}$ şi respectiv literalul A dacă $L = \rceil A$. Sperăm ca această notație sintactică să nu fie confundată cu operația semantică $^{-}$, prezentă în definiția algebrelor booleene, rezultatele privind sintaxa fiind, în general, separate de cele privind semantica. Se numeşte **clauză** orice disjuncție (finită) de literali. Se numeşte **clauză Horn** o clauză care are cel mult un literal pozitiv. O **clauză pozitivă** este o clauză care conține doar literali pozitivi, iar o **clauză negativă** va conține doar literali negativi. O **clauză Horn pozitivă** va conține exact un literal pozitiv (dar, posibil, şi literali negativi).

§3. Semantica logicii propoziționale

Semantica (înțelesul) unei formule propoziționale este, conform principiilor logicii aristotelice, o valoare de adevăr (**a** sau **f**), obținută în mod determinist, care este independentă de context, etc. Notând de la început pe **a** cu 1 şi pe **f** cu 0, astfel încât să putem lucra cu algebra booleană $\mathcal{B} = < \mathbf{B}, \bullet, +, \overline{\ } >$, noțiunea principală este cea de ***asignare*** **(*interpretare*, structură)**.

Definiția 2.4. Orice funcție $\mathcal{S}$, $\mathcal{S} : \mathcal{A} \to \mathbf{B}$ se numeşte **asignare**. ■

Teorema 2.1 (de extensie). Pentru fiecare asignare $\mathcal{S}$ există o *unică extensie* a acesteia, $\mathcal{S}' : \mathbf{LP} \to \mathbf{B}$ (numită tot **structură** sau **interpretare**), care satisface:

(i) $\mathcal{S}'(A) = \mathcal{S}(A)$, pentru fiecare $A \in \mathcal{A}$.

(ii) $\mathcal{S}'((\rceil F)) = \overline{\mathcal{S}'(F)}$, pentru fiecare $F \in \mathbf{LP}$.

(iii) $\mathcal{S}'((F_1 \wedge F_2)) = \mathcal{S}'(F_1) \bullet \mathcal{S}'(F_2)$, pentru fiecare $F_1, F_2 \in \mathbf{LP}$.

(iv) $S'((F_1 \vee F_2)) = S'(F_1) + S'(F_2)$, pentru fiecare $F_1, F_2 \in$ **LP**.

Demonstrație. Fie $S : \mathcal{A} \to \mathbf{B}$. Definim funcția $S' : \mathbf{LP} \to \mathbf{B}$, structural, prin:

Baza. $S'(A) = S(A)$, pentru fiecare $A \in \mathcal{A}$.

Pas constructiv.

(a) Dacă $F = (\rceil F_1)$, atunci $S'(F) = \overline{S'(F_1)}$.

(b) Dacă $F = (F_1 \wedge F_2)$, atunci $S'(F) = S'(F_1) \bullet S'(F_2)$.

(c) Dacă $F = (F_1 \vee F_2)$, atunci $S'(F) = S'(F_1) + S'(F_2)$.

Este evident că S' este o extensie a lui S, proprietatea (i) fiind satisfăcută imediat conform pasului **Baza** de mai sus. De asemenea, definițiile (a) – (c) din **Pasul constructiv** asigură satisfacerea punctelor (ii) – (iv) din enunț, deoarece orice formulă din **LP**, dacă nu este elementară, are una dintre cele trei forme considerate (cazul $F = (F_1)$ este mult prea simplu pentru a fi tratat separat). Mai rămâne să arătăm că S' este funcție *totală* (adică, *atașează fiecărui element din domeniu un element și numai unul din codomeniu*) și că ea este *unica* funcție care satisface (i) – (iv). Acest lucru se face prin **inducție structurală**, trebuind să arătăm că *pentru fiecare F Î* ***LP****, este adevărat P(F)*, unde P(F) este: *Oricare ar fi asignarea S, valoarea $S'(F)$ există (ca element al lui* ***B****) și este unică, adică S' este funcție, și oricare altă funcție S'' care satisface (i) – (iv), satisface $S'(F)= S''(F)$.*

Baza. Fie $F = A \in \mathcal{A}$ și orice asignare S. Cum S este funcție (totală) prin definiție și avem $S'(A) = S(A)$, tot prin definiție (S' este extensia lui S), este imediat faptul că $S'(A)$ există şi este unică în sensul precizat (orice alt – posibil – S'' trebuie să fie tot o extensie a lui S).

Pas inductiv. Vom arăta doar cazul F = ($\rceil$ F_1), celelalte două (F = ($F_1 \wedge F_2$) şi F = ($F_1 \vee F_2$)) fiind similare. Presupunem prin urmare $P(F_1)$ ca fiind adevărat şi demonstrăm că $P(F)$ este adevărat. Fie orice asignare S. Faptul că $S'(F)$ există (ca element al lui **B**) şi este unică (în sensul precizat), rezultă din nou imediat, din ipoteza inductivă ($S'(F_1)$ există şi este unică), din definiția negației în **B** (ştim că $S'(F) = \overline{S'(F_1)}$) şi a faptului că orice alt S'' trebuie să satisfacă punctul (ii) din teoremă. ■

N.B. *De acum înainte* ***nu vom face nici o diferență, nici măcar notațională, între asignare şi structură (intrerpretare)****. Se observă că dată orice formulă F Î* **LP** *şi orice structură S, este suficient să cunoaştem valorile lui S în* ***variabilele propoziționale care apar*** *în F (pentru fiecare F Î* ***LP****, vom nota cu* ***prop(F) mulțimea atomilor pozitivi care apar în F****, sau* ***peste care este construită*** *F). Vom numi* ***asignare (structură) completă pentru F****, orice funcție* ***parțială*** *S care este definită exact (sau, măcar) pe prop(F) Í A şi cu valori în* **B***. Aceasta, în cazul în care F este cunoscută, poate fi identificată cu o funcție totală pe A. Putem conchide chiar că în* ***LP valoarea de adevăr a unei formule se deduce în mod unic din valoarea de adevăr a subformulelor*** *(se mai spune că logica propozițională are* ***proprietatea***

de extensionalitate). Vom mai pune $\mathcal{S}$(F) $\cong$ $\mathcal{S}$((F)) pentru fiecare F$\in$ **LP**.

Exercițiul 2.5. *Definiți structural prop(F), pentru fiecare F $\in$ **LP**.*

Fără alte precizări, vom lucra în continuare doar cu structuri complete pentru mulțimile de formule (o structură este completă pentru o mulțime de formule dacă este completă pentru fiecare element din acea mulțime) care ne interesează la momentul dat.

Definiția 2.5. O formulă F $\in$ **LP** se numeşte **satisfiabilă** dacă există măcar o structură $\mathcal{S}$ (completă) pentru care formula este adevărată ($\mathcal{S}$(F) = 1). Se mai spune în acest caz că $\mathcal{S}$ este model pentru F (simbolic, se mai scrie $\mathcal{S} \models$ F). O formulă este **validă (tautologie)** dacă *orice* structură este model pentru ea. O formulă este **nesatisfiabilă (contradicție)** dacă este falsă în orice structură ($\mathcal{S}$(F) = 0, pentru fiecare $\mathcal{S}$, sau $\mathcal{S} \not\models$ F, pentru fiecare $\mathcal{S}$). ■

Teorema 2.2. O formulă F $\in$ **LP** este validă dacă şi numai dacă ($\rceil$F) este contradicție.

Demonstrație. F $\in$ **LP** este validă dacă şi numai dacă pentru fiecare structură $\mathcal{S}$ avem $\mathcal{S}$(F) = 1, adică (conform ii), **Teorema 2.1**) dacă şi numai dacă $S((\rceil F))=1=\overline{0}$ (definiția negației), ceea ce înseamnă că $(\rceil F)$ este o contradicție. ∎

Clasa tuturor formulelor propoziționale **LP**, este astfel partiționată în (mulțimile indicate mai jos *sunt într-adevăr nevide și disjuncte*):

Tautologii	Formule satisfiabile dar nevalide		Contradicții
F	F	$(\rceil F)$	$(\rceil F)$

Tabelul 2.1

În tabelul anterior linia punctată poate fi considerată drept o *oglindă în care se reflectă adevărul.*

Definiția 2.6. Două formule F_1, $F_2 \in$ **LP** se numesc **tare echivalente** dacă *pentru fiecare* structură S ele au aceeaşi valoare de adevăr, adică $S(F_1) = S(F_2)$ (simbolic, vom scrie $F_1 \equiv F_2$). F_1 şi F_2 se numesc **slab echivalente** dacă F_1 satisfiabilă *implică* F_2 satisfiabilă şi reciproc (vom scrie $F_1 \equiv_s F_2$, ceea ce înseamnă că *dacă există S_1 astfel încât $S_1(F_1) = 1$, atunci există S_2 astfel încât $S_2(F_2) = 1$ şi reciproc*). O formulă $F \in$ **LP** este **consecință semantică** dintr-o mulțime de formule $G \subseteq$ **LP**, dacă *pentru fiecare structură corectă* S (aceasta înseamnă aici faptul că *S* este definită cel puțin pentru toate variabilele propoziționale care apar fie în F fie în elementele lui *G*), ***dacă*** *S* ***satisface*** *G* (adică avem *S*(G) = 1 pentru fiecare G ∈ *G*) ***atunci*** *S* ***satisface F*** (simbolic, vom scrie *G* ‘ F). ■

Observație. Relațiile ≡ şi $\equiv_s$ sunt ***relații de echivalență*** (binare) pe **LP**, în sens matematic (adică sunt ***reflexive***, ***simetrice*** şi ***tranzitive***) şi prin urmare **LP** poate fi ***partiționată în clase de ehivalență*** corespunzătoare, obținându-se ***mulțimile cât*** **LP/°**, respectiv **LP/°**$_s$. Mai mult, privind ∧ , ∨ , ⌉ ca nişte operatori (de fapt, ar trebui să-i considerăm ***împreună*** cu parantezele pe care le introduc, vezi şi **Exercițiul 2.4**), atunci relația ≡ este ***compatibilă (la stânga şi la dreapta)*** cu aceştia ([ŢIP]), astfel încât considerând 4-uplul *LP* = <**LP/°**, Ù,Ú,ù>, se poate arăta că acesta formează o algebră booleană (homomorfă cu *B* = < **B**, •, +, ¯ >, după cum sugerează **Teorema de extensie**). ■

Teorema 2.3. Fie G ∈ **LP** şi *G* = { G_1, G_2, …, G_n } ⊆ **LP**. Următoarele afirmații sunt echivalente:

(i) G este consecință semantică din *G*.

(ii) $(\bigwedge_{i=1}^{n} G_i) \to G$ este tautologie.

(iii) $(\bigwedge_{i=1}^{n} G_i) \wedge \rceil G$ este contradicție.

Demonstrație.

(i) ***implică*** **(ii)**. Presupunem prin reducere la absurd că $F = (\bigwedge_{i=1}^{n} G_i) \rightarrow G$ nu este tautologie, deşi G este consecință semantică din $\mathcal{G}$. Rezultă că există o structură S pentru care F este falsă, adică $S(\bigwedge_{i=1}^{n} G_i) = 1$ şi $S(G) = 0$. Prin urmare, pentru fiecare $i \in [n]$ avem $S(G_i) = 1$ şi ca urmare $S(G) = 0$. În concluzie, există o structură S astfel încât $S(\mathcal{G}) = 1$ şi $S(G) = 0$. Acest lucru este absurd pentru că G este consecință semantică din $\mathcal{G}$.

(ii) ***implică*** **(iii)**. Procedăm din nou prin reducere la absurd, adică presupunem că deşi $(\bigwedge_{i=1}^{n} G_i) \rightarrow G$ este tautologie, $(\bigwedge_{i=1}^{n} G_i) \wedge \rceil G$ nu este contradicție. Această înseamnă că $F_1 = \rceil (\bigwedge_{i=1}^{n} G_i) \vee G$ este tautologie, dar $F_2 = (\bigwedge_{i=1}^{n} G_i) \wedge \rceil G$ este satisfiabilă. Prin urmare, există o structură S astfel încât $S(F_2) = 1$ (şi, desigur, $S(F_1) = 1$). Din $S(F_2) = 1$ rezultă $S((\bigwedge_{i=1}^{n} G_i)) \bullet S(\rceil G) = 1$, adică $S((\bigwedge_{i=1}^{n} G_i)) = 1$ şi $S(\rceil G) = 1$. În consecință, $S(\rceil (\bigwedge_{i=1}^{n} G_i)) = 0$ şi $S(G) = 0$. Pentru că $S(F_1) = S(\rceil(\bigwedge_{i=1}^{n} G_i)) + S(G)$, avem $S(F_1) = 0$, ceea ce este absurd deoarece F_1 este tautologie.

(iii) ***implică*** **(i)**. Presupunem prin reducere la absurd că $F = (\bigwedge_{i=1}^{n} G_i) \wedge \rceil G$ este contradicție, dar G nu este consecință semantică din *G*. Atunci există o structură *S* care satisface toate formulele din *G* dar nu satisface G. Prin urmare, avem $S((\bigwedge_{i=1}^{n} G_i)) = 1$ şi $S(G) = 0$, adică $S((\bigwedge_{i=1}^{n} G_i)) = 1$ şi $S(\rceil G) = 1$. Cum $S((\bigwedge_{i=1}^{n} G_i) \wedge \rceil G) = S((\bigwedge_{i=1}^{n} G_i)) \bullet S(\rceil G)$, rezultă că există *S* astfel încât $S((\bigwedge_{i=1}^{n} G_i) \wedge \rceil G) = 1$, deci F nu este contradicție (absurd). ■

În teorema anterioară am renunțat la anumite paranteze, respectând prioritățile/convențiile făcute. Vom face şi pe viitor acest lucru, fără a-l mai menționa explicit.

Teorema 2.4. Sunt adevărate următoarele echivalențe (tari, pentru oricare F, G, H ∈ **LP**):

(a) $F \wedge F \equiv F$ **(a')** $F \vee F \equiv F$ *(idempotență)*

(b) $F \wedge G \equiv G \wedge F$ **(b')** $F \vee G = G \vee F$ *(comutativitate)*

(c) $(F \wedge G) \wedge H \equiv$
$\equiv F \wedge (G \wedge H)$ **(c')** $(F \vee G) \vee H \equiv F \vee (G \vee H)$ *(asociativitate)*

(d) $F \wedge (G \vee H) \equiv$
$\equiv (F \wedge G) \vee (F \wedge H)$ **(d')** $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$ *(distributivitate)*

(e) $F \wedge (F \vee G) \equiv F$ **(e')** $F \vee (F \wedge G) \equiv F$ *(absorbție)*

(f) $\rceil\rceil F \equiv F$ *(legea dublei negații)*

(g) $\rceil(F \wedge G) \equiv$ $\equiv \rceil F \vee \rceil G$

(g') $\rceil(F \vee G) \equiv \rceil F \wedge \rceil G$ *(legile lui deMorgan)*

(h) $F \vee G \equiv F$

(h') $F \wedge G \equiv G$ *(legile validității,* adevărate ***doar dacă*** F este tautologie*)*

(i) $F \wedge G \equiv F$

(i') $F \vee G \equiv G$ *(legile contradicției,* adevărate ***doar dacă*** F este contradicție*)*

Demonstrație. Vom arăta doar una dintre echivalențe şi anume **(i)**. Fie $F \in$ **LP** orice contradicție şi $G \in$ **LP**. Fie orice structură S. Atunci $S(F \wedge G) = S(F) \bullet S(G) = 0$, conform **Tabelului 1.1** (punctul 9)) şi faptului că F este contradicție. Aceeaşi valoare o are şi membrul drept din **(i)**. ■

Se poate arăta, de exemplu, prin inducție matematică, faptul că *asociativitatea, distributivitatea şi legile lui deMorgan se extind pentru orice număr finit de formule*.

Teorema 2.5 (de substituție). Fie $H \in$ **LP**, oarecare. Fie orice $F, G \in$ **LP** astfel încît F este o subformulă a lui H şi G este tare echivalentă cu F. Fie H' formula obținută din H prin înlocuirea (unei apariții fixate a) lui F cu G. Atunci $H \equiv H'$.

Demonstrație. Intuitiv, teorema „spune" că *înlocuind într-o formulă o subformulă cu o formulă echivalentă, obținem o formulă echivalentă cu* *cea inițială*. Vom proceda prin inducție structurală, având de arătat **teorema din metalimbaj** ($\forall$H $\in$ **LP**) P(H), unde

P(H): ($\forall$F, G, H' $\in$ **LP**)(((F $\in$ subf(H)) **şi**

(H' se obține din H înlocuind o apariție fixată a lui F cu G) **şi**

(F $\equiv$ G)) $\Rightarrow$ H $\equiv$ H').

Baza. H = A $\in$ *A*. Să arătăm că *P(A) este adevărată*. Fie F, G, H' $\in$ **LP**, astfel încât F $\in$ subf(H), H' se obține din H înlocuind apariția aleasă a lui F cu G, iar F $\equiv$ G. Trebuie să arătăm că H $\equiv$ H'. Dar, din F $\in$ subf(H) şi subf(H) = {A}, rezultă că F = A (care coincide cu H). Prin urmare, H' = G. Avem acum F = H, G = H' şi F $\equiv$ G, de unde urmează imediat că H $\equiv$ H'.

Pas inductiv. Trebuie tratate separat situațiile care urmează.

(i) **H = (ù H_1)**. *Presupunem că P(H_1) este adevărată şi demonstrăm că P(H) este adevărată*. Fie F $\in$ subf(H) = subf(H_1) $\cup$ {($\rceil$ H_1)}. Dacă F = ($\rceil$ H_1) (= H), suntem într-o situație similară cu cea din **Baza**, deoarece raționamentul se face din nou asupra întregii formule H. Fie o apariție fixată a lui F $\in$ subf(H_1) $\subseteq$ subf(H) şi considerăm orice G $\in$ **LP** astfel încât G $\equiv$ F. Înlocuind pe F cu G în H, înseamnă în acelaşi timp a înlocui pe F cu G în H_1. Notând cu H' respectiv H_1' formulele obținute, putem aplica ipoteza inductivă (P(H_1) este adevărată) şi obținem că H_1 $\equiv$ H_1'. Revenind, ştim că H = ($\rceil$ H_1), H' = ($\rceil$ H_1') şi H_1 $\equiv$ H_1'. Rezultă imediat că H $\equiv$ H' (vezi şi **Observația** care precede imediat **Teorema 2.3** şi **V.2.8** din **Anexă**).

(ii) **H = (H_1 Ù H_2)**. *Presupunem că P(H_1) şi P(H_2) sunt adevărate şi demonstrăm că P(H) este adevărată*. Fie orice F ∈ subf(($H_1 \wedge H_2$)) = subf(H_1) ∪ subf(H_2) ∪{($H_1 \wedge H_2$)}. Dacă F = ($H_1 \wedge H_2$) (= H) suntem din nou într-un caz similar cu cel din **Baza**. Să considerăm că F ∈ subf(H_1) (apariţia deja fixată), cazul F ∈ subf(H_2) tratându-se similar. Fie orice G ∈ **LP** astfel încât G ≡ F. A înlocui pe F cu G în H înseamnă, în acelaşi timp, a înlocui pe F cu G în H_1 (H_2 rămânând neschimbată). Vom nota cu H' respectiv H_1', formulele obţinute după aceste înlocuiri. Aplicând ipoteza inductivă (P(H_1) este adevărată), rezultă imediat că H_1' ≡ H_1. Revenind, ştim că H = ($H_1 \wedge H_2$), H' = ($H_1' \wedge H_2$) şi H_1' ≡ H_1. Obţinem imediat că H ≡ H' (putem folosi direct faptul deja amintit, că º *este compatibilă cu operaţiile*, respectiv cu conjuncţia).

(iii) **H = (H_1 Ú H_2)**. Se demonstrează analog cu cazul precedent. ■

Pentru a nu exista confuzii între limbajul de bază (**LP**) şi metalimbajul în care exprimăm afirmaţiile *despre* elementele lui **LP**, în cele de mai sus (precum şi în continuare) am notat implicaţia cu ⇒ iar conjuncţia prin **şi**. Deocamdată am păstrat notaţia clasică pentru cuantificatorul universal (∀), deoarece el nu apare explicit în **LP**.

N.B. *Rezultatele obţinute ne permit practic să tratăm formulele din* **LP** *într-un mod similar cu funcţiile booleene, dacă ne interesează probleme de natură semantică. Astfel, vom nota cu* **0** *orice contradicţie şi cu* **1** *orice tautologie şi vom accepta principiul dualităţii (rolul lui •*

şi + luându-l Ù respectiv Ú, după cum se poate deduce chiar din ***Teorema 2.4****). De asemenea, vom folosi tabelele de adevăr pentru a găsi în mod direct semantica (valoarea de adevăr a) unei formule într-o structură dată.*

Nu apare astfel surprinzătoare tematica paragrafului următor, privind existența *formelor normale*.

§ 4. Forme normale în LP

Spre deosebire de cazul funcțiilor booleene, vom studia pentru început *formele normale conjunctive* şi *formele normale disjunctive* ***simultan***.

Definiția 2.7. O formulă $F \in$ **LP** se află în **formă normală conjunctivă** (**FNC**, pe scurt) dacă este o *conjuncție de disjuncții de literali*, adică o *conjuncție de clauze*. Simbolic:

$$F = \bigwedge_{i=1}^{m} \Big(\bigvee_{j=1}^{n_i} L_{i,j} \Big) \quad (\text{notăm } C_i = \bigvee_{j=1}^{n_i} L_{i,j},\ i \in [m]\ ).$$

Similar, $F \in$ **LP** este în **formă normală disjunctivă** (**FND**, pe scurt), dacă este o *disjuncție de conjuncții de literali.* ■

În cele de mai sus $L_{i,j} \in A \cup \overline{A}$.

Exemplu. $F = (A \wedge (B \vee C))$ este în **FNC** iar $G = ((A \wedge \rceil B) \vee (A \wedge C))$ este în **FND**, dacă $A, B, C \in A$. ■

Teorema 2.6. Pentru fiecare formulă F $\in$ **LP** există cel puțin două formule F_1, $F_2 \in$ **LP**, F_1 aflată în **FNC** şi F_2 aflată în **FND**, astfel încât $F \equiv F_1$ şi $F \equiv F_2$ (*se mai spune că F_1 şi F_2 sunt o **FNC**, respectiv o **FND**, pentru F*).

Demonstrație. Pentru a demonstra afirmația necesară, $(\forall F)P(F)$ în metalimbaj, unde

$P(F)$: există $F_1 \in$ **LP**, aflată în **FNC** şi există $F_2 \in$ **LP**, aflată în **FND**, astfel încât $F \equiv F_1$ şi $F \equiv F_2$,

procedăm prin inducție structurală.

Baza. $F = A \in \mathcal{A}$. Această formulă este atât în **FNC** cât şi în **FND**, deci putem lua $F_1 = A$ şi $F_2 = A$.

Pas inductiv. Trebuie tratate cazurile corespunzătoare definiției constructive a lui **LP**.

(i) **F = (**ù **G)**. *Presupunem că $P(G)$ este adevărată şi demonstrăm că $P(F)$ este adevărată*. Din ipoteza inductivă rezultă că există formulele G_1, aflată în **FNC** şi G_2, aflată în **FND**, astfel încât $G \equiv G_1$ şi $G \equiv G_2$. Atunci, de exemplu, $\rceil G \equiv \rceil G_1$ şi, aplicând legile lui deMorgan, găsim:

$$\rceil\left(\bigwedge_{i=1}^{m}\left(\bigvee_{j=1}^{n_i} L_{i,j}\right)\right) \equiv \left(\bigvee_{i=1}^{m}\left(\bigwedge_{j=1}^{n_i}\left(\rceil L_{i,j}\right)\right)\right) .$$

În ultima formulă putem aplica – unde este cazul – legea dublei negații şi apoi putem înlocui elementele de forma $\rceil L_{i,j}$ cu $\overline{L}_{i,j}$, obținând astfel o **FND** pentru F. Analog, dacă pornim cu G_2, care este o **FND** pentru G, vom obține o **FNC** pentru F.

(ii) **F = (G Ù H)**. *Presupunem că afirmațiile P(G) şi P(H) sunt adevărate şi arătăm că P(F) este adevărată*. Din faptul că P(G) este adevărată rezultă că există G1, aflată în **FNC** şi satisfăcând $G \equiv G_1$, astfel încât:

$$G_1 = (\bigwedge_{i=1}^{m_1}(\bigvee_{j=1}^{(n_1)_i} L_{i,j}))$$

Cu totul similar, pentru că P(H) este adevărată, înseamnă că există H_1, aflată în **FNC** şi satisfăcând $H \equiv H_1$:

$$H_1 = (\bigwedge_{i=1}^{m_2}(\bigvee_{j=1}^{(n_2)_i} L_{i,j}))$$

Atunci, $G \wedge H \equiv G_1 \wedge H_1$ şi este evident că ultima formulă este tot o conjuncție de disjuncții, adică este o **FNC**, notată F_1, pentru F. Pentru a obține o **FND**, F_2, pentru F, pornim de la o **FND**, G_2, pentru G şi o **FND**, H_2, pentru H. Atunci $F = G \wedge H \equiv G_2 \wedge H_2$, de unde obținem imediat o **FND** pentru F, notată F_2, dacă se aplică mai întâi distributivitatea generalizată a conjuncției față de disjuncție şi apoi, *în interiorul subformulelor*, a disjuncției față de conjuncție.

(iii) **F = (G Ú H)**. Procedăm analog ca în cazul anterior. ■

Teorema precedentă sugerează existența unui *algoritm recursiv* pentru obținerea *simultană* a unei **FNC** şi a unei **FND**, pentru orice formulă propozițională. Putem folosi însă şi tabelele de adevăr şi modalitățile de găsire a formelor normale conjunctive/disjunctive (perfecte) descrise în **Capitolul 1**.

Exemplu. Găsiți o formulă F $\in$ **LP** construită peste mulțimea de variabile propoziționale {A, B, C} şi care să satisfacă condiția: *în tabelul de adevăr standard care o descrie,* ***o schimbare şi numai una*** *în secvența* $<S(A), S(B), S(C)>$ *produce* ***schimbarea valorii*** ***corespunzătoare de adevăr*** *$S(F)$*. Dacă începem secvența S(F) cu 0, atunci F este descrisă de tabelul:

A	B	C	F	
S(A)	S(B)	S(C)	S(F)	
0	0	0	0	
0	0	1	1	*
0	1	0	1	*
0	1	1	0	
1	0	0	0	
1	0	1	1	*
1	1	0	1	*
1	1	1	0	

Se poate construi apoi direct din tabel măcar o formulă (sau două) care îi corespunde semantic, formulă ce se află în **FND** (şi/sau **FNC**). De fapt vom folosi algoritmul de construcție a **FNDP** (**FNCP**) pentru o funcție booleană. Conform **Capitolului 1**, acesta poate fi exprimat astfel: *se fixează liniile având 1 în ultima coloană (cele marcate cu * în tabel); pentru fiecare asemenea linie se construieşte o conjuncție de literali (apar toți, cu bară sau fără): dacă valoarea unei variabile (atom pozitiv) este 0 în tabel, atunci variabila se trece în conjucția* *respectivă negată, iar dacă valoarea ei este 1, atunci ea apare nenegată; formula finală, aflată în* ***FND(P)****, este disjuncția tuturor acestor conjuncții*. Prin urmare, putem pune F = ($\rceil$A $\wedge$ $\rceil$ B $\wedge$ C) $\vee$ $\vee$ ($\rceil$A $\wedge$ B $\wedge$ $\rceil$C) $\vee$ (A $\wedge$ $\rceil$B $\wedge$ C) $\vee$ (A $\wedge$ B $\wedge$ $\rceil$C). Găsiți, analog, o **FNC(P)** ■

Conform teoremei anterioare, precum şi datorită comutativității şi idempotenței disjuncției, comutativității şi idempotenței conjuncției (repetarea unui element, fie el literal sau clauză, este nefolositoare din punctul de vedere al (ne)satisfiabilității unei formule), este justificată ***scrierea ca mulțimi a formulelor aflate în FNC***. Astfel, dacă F este în **FNC** (**Definiția 2.7**), vom mai scrie F = {C_1, C_2, ... , C_m} (nu uităm totuşi că *virgula aici provine dintr-o conjuncție*), unde, pentru fiecare i $\in$ [m], vom pune $C_i = \{L_{i,1}, L_{i,2}, ..., L_{i,n_i}\}$. Mai mult, dacă avem F $\in$ **LP** reprezentată ca mulțime (de clauze) sau ca mulțime de mulțimi (de literali) şi ne interesează doar studiul (ne)satisfiabilității ei, putem elimina clauzele C care conțin atât L cât şi $\overline{L}$, deoarece L $\vee$ $\overline{L}$ $\equiv$ **1**, **1** $\vee$ C $\equiv$ **1** şi deci aceste clauze sunt tautologii (notate generic cu **1**). Tautologiile componente nu au nici o semnificație pentru stabilirea valorii semantice a unei formule F aflate în **FNC** (**1** $\wedge$ C $\equiv$ C).

§ 5. Decidabilitate în LP

LP, cadrul formal propus (*realitatea este modelată prin afirmații, afirmațiile sunt reprezentate ca formule propoziționale*), oferă ca principală metodă de a rezolva problemele, testarea adevărului (satisfiabilității) unor formule. Din punctul de vedere al unui informatician, trebuie ca pentru clasa de formule admisă să existe un algoritm care, având la intrare orice F ∈ **LP**, se termină cu răspunsul „DA", dacă F este satisfiabilă (sau validă, sau contradicție) şi „NU" în rest (ştiind desigur că putem decide dacă un anumit şir de caractere este formulă sau nu). În această situație se spune că **problema satisfiabilității (**pe scurt, **SAT) pentru LP este rezolvabilă (decidabilă)**. Mai mult, am vrea să găsim asemenea algoritmi pentru care **complexitatea timp este „rezonabilă"**.

Teorema 2.7 (decidabilitatea SAT). Satisfiabilitatea (validitatea, nesatisfiabilitatea) formulelor calculului propozițional este decidabilă în *timp exponențial*.

Demonstrație. Practic, demonstrația (exceptând complexitatea) a fost deja făcută, chiar în mai multe moduri. Fie F ∈ **LP** având prop(F) = $\{A_1, A_2, \ldots\ldots, A_n\} = \mathcal{A}_n$. Se formează, de exemplu în **Pasul 1** al unui posibil algoritm (notat tot **SAT**) pentru testarea satisfiabilității (validității, nesatisfiabilității), tabela de adevăr corespunzătoare lui F (*Teorema de extensie*, *Teorema de substituție* şi *legătura dintre* *algebrele booleene* LP şi B stau la baza corectitudinii acestei construcţii) care are forma:

$S(A_1)$	$S(A_2)$	……	$S(A_n)$	$S(F)$
0	0	……	0	v_1
0	0	……	1	v_2
……	……	……	……	……
1	1	……	1	v_m

$2^n = m$

Dacă toţi v_i ($i \in [m]$) sunt egali cu 0 atunci F este contradicţie, dacă toţi v_i sunt 1 atunci F este tautologie, iar în rest F este satisfiabilă dar nevalidă. Pentru a depista acest lucru, trebuie parcurs, în **Pasul 2**, *în cazul cel mai defavorabil*, întregul tabel, linie cu linie şi prin urmare trebuie efectuate 2^n comparaţii (F este construită peste n formule atomice). Deşi mai sus nu avem o explicaţie formală riguroasă a faptului că **SAT** are *timp exponenţial*, se poate arăta că problema este chiar *NP-completă* (conform [AHO]; a se urmări şi comentariile care urmează imediat după demonstraţie). ■

Datorită **Teoremei de extensie** şi **Teoremei de substituţie**, putem construi o tabelă de adevăr pentru o formulă pornind nu de la variabile, ci chiar de la anumite subformule mai complicate (pentru care valorile posibile, finale, sunt tot 0 sau 1). Mai mult (se pot consulta [KNU], [JUC], [LUC] şi, în special, [CRO], [AHO], [COR], [BÖR]), trebuie căutaţi algoritmi „rapizi” (eficienţi, tratabili), adică având complexitate (timp) mică. Astfel, două dintre *măsurile (teoretice, globale) de complexitate* des întrebuințate sunt *complexitatea timp* şi *complexitatea spațiu*. Ideea este aceea că un (orice) pas elementar (instrucțiune) al unui algoritm se execută într-*o unitate de timp* (pentru spațiu, *fiecare dată elementară* se memorează într-*un registru* sau *locație de memorie*, acesta/aceasta ocupând *o unitate de spațiu*), criteriul numindu-se *al costurilor uniforme*. Există şi *criteriul costurilor logaritmice* (pe care însă nu-l vom utiliza aici), în care orice informație de *lungime* i, se prelucrează (respectiv, se memorează) în numărul de unități de timp (unități de spațiu) egal cu $\lfloor \log(i) \rfloor + 1$ (dacă $i = 0$, se convine să luăm $\log(i) = 0$; $\lfloor n \rfloor$ notează *partea întreagă inferioară* a numărului n). Intuitiv, timpul luat de execuția unui algoritm *Alg* este dat de *numărul de instrucțiuni (paşi/operații elementare) efectuate* (să-l notăm cu t^{Alg}), iar spațiul (notat cu s^{Alg}) este dat de *numărul de locații (elementare) de memorie (internă, a calculatorului) ocupate* în cursul execuției. Sigur că *totul se raportează la lungimea fiecarei intrări* (adică, în cazul nostru, la lungimea unei formule $F \in$ **IN** $\subseteq$ **LP**, aceasta putând fi de exemplu n_F = *card (prop(F))*) şi ne interesează de fapt $\sup\{t^{Alg}(F) \mid F \in$ **IN** şi $n_F = n \in \mathbf{N}\}$, margine superioară pe care o vom nota cu $t^{Alg}(n)$. Această abordare (în care se caută *cazul cel mai nefavorabil*, dacă este desigur posibil), ne permite să fim siguri că pentru fiecare intrare de lungime n, timpul de execuție al lui *Alg* nu va depăşi $t^{Alg}(n)$. Cum determinarea acelui supremum este de multe ori destul de dificilă, ne vom mulțumi să studiem aşa-numita *comportare* *asimptotică* (sau *ordinul de creştere*) a (al) lui $t^{Alg}(n)$, adică ne vor interesa doar anumite margini ale sale, cum ar fi marginea sa superioară. Formal, pentru fiecare f : **N** → **N**, notăm *O(f) = {g | g : **N** ® **N**, există c Î **R**, c > 0 şi există k Î **N**, astfel încât pentru fiecare n ³ k avem g(n) £ c•f(n)}* şi vom spune ***că fiecare g Î O(f), este de ordinul lui f***, ceea ce se mai notează şi cu *g = O(f)*. Astfel, vom spune că **SAT** are *complexitatea (timp, asimptotică)* $O(2^n)$, sau, pe scurt, complexitate exponenţială, deoarece că există (măcar) un algoritm *Alg* care rezolvă problema (cel sugerat în demonstraţia **Teoremei 2.7**) şi pentru care $t^{Alg}(n) = O(2^n)$. Similar, vom vorbi de *algoritmi polinomiali* ($t^{Alg}(n) = O(p(n))$, unde p(n) desemnează un polinom în n, de orice grad), sau de *algoritmi liniari* (p(n) de mai sus este un polinom de gradul I), adică de probleme având complexitatea (timp, dar se poate defini ceva asemănător pentru spaţiu) de tipul precedent. Speranţa de a găsi algoritmi *mai performanţi* pentru rezolvarea **SAT**, se poate baza pe ideea de a restrânge **LP** la anumite *subclase stricte, particulare* de formule ale sale, suficient de largi însă pentru a *exprima convenabil* părţi importante ale realităţii. În plus, în condiţiile utilizării calculatorului, găsirea unor algoritmi de natură sintactică pentru rezolvarea **SAT** (în locul celor „semantici", cum este şi cel bazat pe folosirea tabelelor de adevăr) este o prioritate (chiar dacă aceştia nu sunt mai buni din punctul de vedere al teoriei generale a complexităţii).

§ 6. Formule Horn

Reamintim că o clauză Horn este o disjuncție de literali care conține cel mult un literal pozitiv.

Definiția 2.8. O **formulă Horn** este o formulă aflată în **FNC**, clauzele componente fiind (toate) clauze Horn. ■

Uneori, vom numi tot formulă Horn şi o formulă care este (tare) echivalentă cu o formulă de forma considerată în **Definiția 2.8**. Se poate arăta ([MAS1]) că există formule propoziționale care nu sunt tare echivalente cu nici o formulă Horn, apariția a măcar doi literali pozitivi distincți într-o clauză fiind decisivă. Formele posibile pentru o formulă Horn sunt (variabilele propoziționale care apar sunt elemente ale lui $\mathcal{A}$):

(i) $C = \rceil A_1 \vee \rceil A_2 \vee \ldots\ldots \vee \rceil A_k$, $k \geq 1$, $k \in \mathbf{N}$ şi

(ii) $C = \rceil A_1 \vee \rceil A_2 \vee \ldots\ldots \vee \rceil A_k \vee B$, $k \in \mathbf{N}$.

Observație. Înafară de reprezentarea ca mulțimi, clauzele Horn pot fi reprezentate sub şi sub aşa-numita **formă implicațională**. Vom distinge cazurile (reamintim că **0** şi **1** denotă orice contradicție respectiv orice tautologie):

- $C = A \in \mathcal{A}$ (*nici un literal negativ, un literal pozitiv*). Acest lucru se mai poate scrie sub forma $C \cong \mathbf{1} \rightarrow A$, ceea ce se justifică prin aceea că $\mathbf{1} \rightarrow A \cong \rceil\mathbf{1} \vee A \equiv \mathbf{0} \vee A \equiv A$.

- $C = \rceil A_1 \vee \rceil A_2 \vee \ldots\ldots \vee \rceil A_k$ (*nici un literal pozitiv, măcar un literal negativ*). Vom scrie C **@** $A_1 \wedge A_2 \wedge A_3 \ldots\ldots \wedge A_k \rightarrow \mathbf{0}$ (folosim din nou definiția implicației şi faptul că $\mathbf{0} \vee A \equiv A$).
- $C = \rceil A_1 \vee \rceil A_2 \vee \ldots\ldots \vee \rceil A_k \vee B$ (*exact un literal pozitiv, măcar un literal negativ*). Atunci C**@**$A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_k \rightarrow B$, direct din definiția implicației.
- C **@** (*nici un literal negativ, nici un literal pozitiv*). Din motive tehnice vom folosi şi această **clauză vidă** (în reprezentarea clauzelor cu mulțimi vom folosi pentru chiar Ø). Prin convenție, *este o clauză de orice tip* (inclusiv o clauză Horn), dar *nesatisfiabilă*. ■

Teorema 2.8. Satisfiabilitatea formulelor Horn este decidabilă în timp liniar.

Demonstrație. Să considerăm algoritmul:

Algoritm Horn

Intrare: Orice formulă Horn, F, reprezentată ca mulțime de clauze, clauzele componente fiind clauze Horn diferite de clauza vidă şi scrise sub formă implicațională .

Ieşire: „DA”, în cazul în care formula F este satisfiabilă (furnizându-se şi o asignare S care este model pentru F) şi „NU” în caz contrar (F nu este satisfiabilă).

Metodă (*de marcare*):

Pasul 1. i := 0.

Pasul 2.

Cât_timp ((există în F o clauză C de forma $A_1 \wedge A_2 \wedge A_3 \ldots\ldots \wedge A_k \rightarrow B$, cu A_1, A_2, A_3, ... , A_k *marcați* şi B *nemarcat* sau de forma $A_1 \wedge A_2 \wedge A_3 \ldots\ldots \wedge A_k \rightarrow \mathbf{0}$, cu A_1, A_2, A_3, ... , A_k *marcați*) **şi** (i = 0))

execută

Pasul 3. Alege un asemenea C ca mai sus.

Pasul 4. **Dacă** ($C = A_1 \wedge A_2 \wedge A_3 \ldots\ldots \wedge A_k \rightarrow B$)

atunci

Pasul 5. Marchează B peste tot în F.

altfel

Pasul 6. i := 1.

Sf_Dacă

Sf_Cât_timp

Pasul 7. **Dacă** (i = 0)

atunci

Pasul 8. Scrie „DA”.

Pasul 9. Scrie S, cu $S(A) = 1$ dacă şi numai dacă A apare în F şi este *marcată*.

altfel

Pasul 10. Scrie „NU”.

Sf_Dacă.

Arătăm mai întâi că **algoritmul *se termină* pentru fiecare intrare**. Să precizăm că acțiunea de *marcare* o privim în sens grafic normal, *marcajul* care poate fi ataşat unei variabile proziționale alegându-se fără criterii speciale (să presupunem că el este *, împreună eventual cu anumiți indici prin care să se identifice în care dintre execuțiile *corpului buclei* s-a făcut marcarea). *Inițial, toate variabilele se presupun a fi nemarcate*. Dacă F conține clauze de forma **1** $\rightarrow$ B (care se *consideră a fi de fapt de forma* $A_1 \wedge A_2 \wedge A_3 \ldots\ldots \wedge A_k \rightarrow B$, cu A_1, A_2, A_3, ... , A_k *marcați* şi B *nemarcat*), se procedează conform algoritmului, adică se marchează toate aparițiile lui B în F şi se trece la pasul următor. Mai departe, la *fiecare* execuție a *corpului buclei* (**Paşii 3.** şi **4.**), fie se marchează o variabilă propozițională nouă (nemarcată încă), fie se iese din execuția buclei. Pentru că numărul de variabile peste care este construită formula F este finit, terminarea algoritmului este evidentă. Dacă nu există deloc clauze de tipul **1** $\rightarrow$ B, algoritmul se termină fără nici o execuție a corpului buclei, cu răspunsul „DA” (formula este satisfiabilă) şi cu asignarea $\mathcal{S}$, în care $\mathcal{S}(A) = 0$ pentru fiecare A (care apare în F).

Arătăm în continuare că **algoritmul este *corect***. Aceasta înseamnă că *ieşirea algoritmului satisface ceea ce am dorit*, adică răspunsul „DA”/$\mathcal{S}$ corespunde faptului că formula F furnizată la intrare este satisfiabilă (şi $\mathcal{S}$ ‘ F) iar răspunsul „NU” corespunde faptului că F este nesatisfiabilă.

Vom separa cazurile:

Cazul a). La terminarea execuției se obține „DA” şi F nu conține clauze C de tipul **1** $\rightarrow$ B. După cum am observat, acest lucru înseamnă că bucla s-a terminat fără să i se execute vreodată corpul având în plus $i = 0$ şi $S(A) = 0$ pentru fiecare A (care apare în F). Atunci există în F (la finalul execuției) ***doar*** clauze de tipul $C_1=A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow B$, sau $C_2 = A_1 \wedge A_2 \wedge A_3 \dots\dots \wedge A_k \rightarrow \mathbf{0}$ $(k \geq 1)$, care n-au nici o variabilă ***marcată***. Avem atunci, pe scurt, $S(C_1) = S(0 \bullet 0 \bullet \dots \bullet 0 \rightarrow \mathbf{0}) = 1$, respectiv $S(C_2) = 1$, de unde găsim $S(F) = 1$.

Cazul b). La terminare se obține „DA” iar F conține şi clauze $C = \mathbf{1} \rightarrow B$. Atunci bucla se termină după un anumit număr de execuții ale corpului său, valoarea lui i este 0 şi F conține în final clauze C având ***marcate*** anumite variabile. Dacă $C = \mathbf{1} \rightarrow B$ (adică $C = B$), unde B este marcat ($S(B) = 1$), avem imediat $S(C) = 1$. Dacă $C = A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow B$ $(k \geq 1)$ este posibil ca, fie toate variabilele din antecedent sunt ***marcate*** (dar atunci B este şi el marcat şi atunci, din nou, $S(C) = 1$ pentru că semantica lui C este de tipul $\mathbf{1} \rightarrow \mathbf{1}$), fie există măcar una dintre variabilele A_i de mai sus care este ***nemarcată***, dar atunci vom avea iarăşi $S(C) = 1$, pentru că semantica sa este de tipul $\mathbf{0} \rightarrow \mathbf{1}$ sau $\mathbf{0} \rightarrow \mathbf{0}$. În sfârşit, dacă $C = A_1 \wedge A_2 \wedge A_3 \dots\dots \wedge A_k \rightarrow \mathbf{0}$ $(k \geq 1)$, unde măcar un A_i este ***nemarcat***, semantica lui C este de forma $\mathbf{0} \rightarrow \mathbf{0}$ şi obținem din nou $S(C) = 1$). Concluzia este că $S(C) = 1$ pentru fiecare C care apare în F, adică $S(F) = 1$.

Cazul c). Algoritmul se termină cu $i = 1$ şi răspunsul „NU”. Acest lucru înseamnă că există în F o clauză $\underline{C} = A_1 \wedge A_2 \wedge A_3 \dots\dots \wedge A_k \rightarrow \mathbf{0}$ cu toți A_i, $i \in [k]$ marcați (obligatoriu, în F există şi clauze de forma

1 $\rightarrow$ B, B marcat), de unde rezultă că semantica lui $\underline{C}$ în *asignarea furnizată de algoritm* este de forma **1** $\rightarrow$ **0** şi prin urmare $\mathcal{S}(\underline{C}) = 0$, de unde $\mathcal{S}(F) = 0$. Acest lucru **nu înseamnă însă că F este nesatisfiabilă**. Pentru a trage această concluzie trebuie să arătăm că pentru *nici o altă asignare, ea nu poate fi model pentru F*. Să presupunem (RA) că există o asignare $\mathcal{S}$**'** (diferită de $\mathcal{S}$, furnizată de algoritm) astfel încât $\mathcal{S}'(F) = 1$. Să observăm, pentru început, că toate variabilele care au fost marcate în algoritm (deci cele care au primit valoarea de adevăr 1 în $\mathcal{S}$), trebuie să primească valoarea 1 în oricare $\mathcal{S}$**'** cu $\mathcal{S}'(F) = 1$. *Altfel spus, asignarea $\mathcal{S}$ conține cel mai mic număr posibil de valori 1 (atribuite evident variabilelor marcate) astfel încît formula* ***să aibă şanse*** *să fie satisfiabilă*. Într-adevăr, pentru fiecare $\mathcal{S}$**'** cu $\mathcal{S}'(F) = 1$, trebuie să avem $\mathcal{S}'(C) = 1$ pentru fiecare clauză C din F. Să ne ocupăm puțin de momentul în care se marchează o variabilă B, ordonând clauzele din F de forma $C = A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_k \rightarrow B$ $(k \geq 1)$ după numărul de variabile din antecedent (chiar în algoritm, selecția unei clauze „*pentru marcare*" se poate face după un asemenea criteriu):

- *Clauze C de tipul* ***1*** *® B º B (nici o variabilă în antecedent, B nemarcat).* De la acestea *începe* procesul de marcare. Din faptul că $\mathcal{S}'(C)$ trebuie să fie egal cu 1, este clar că trebuie pus $\mathcal{S}'(B) = 1$ (B se şi marchează, deci $\mathcal{S}(B) = 1$).
- *Clauze C de forma A ® B º ù A Ú B (o variabilă în antecedent; A este marcat, B nemarcat).* A nu putea fi marcat decât dacă **a apărut deja ca un consecvent** într-o clauză de

tipul anterior, sau în una de acelaşi tip cu aceasta şi care are antecedentul marcat. Prin urmare, în orice S' cu $S'(C) = 1$, trebuie oricum să avem $S'(A) = 1$, deci $S'(\neg A) = 0$ şi atunci $S'(B) = 1$ (concecința este că B se marchează, deci şi $S(B) = 1$).

- Continuăm raționamentul cu $C = A_1 \wedge A_2 \rightarrow B$ *(două variabile în antecedent; ambele variabile marcate; B este, încă, nemarcat)*, ajungând din nou la concluzia că pentru fiecare S', pentru a avea $S'(C) = 1$, trebuie să avem $S'(B) = 1$, precum şi $S(B) = 1$.

Revenind, am arătat într-adevăr că pentru fiecare S' astfel încât $S'(F) = 1$, trebuie să avem $S'(A) = 1$ pentru fiecare A *marcat* de către algoritm, adică pentru fiecare A care satisface şi $S(A) = 1$ (procesul descris mai sus se continuă pentru oricâte variabile prezente în antecedent, iar numărul acestora este finit). Prin urmare, avem şi $S'(\underline{C}) = 0$, de unde rezultă că $S'(F) = 0$, ceea ce este absurd.

Să arătăm în final că **algoritmul Horn are timp de execuție liniar**. Faptul că $t(n) \in O(f(n))$, unde $f(n) = a \cdot n + b$ ($a, b \in \mathbf{N}^*$), rezultă imediat din faptul că la fiecare execuție a *corpului buclei* se *marchează* o *nouă* variabilă. Desigur că pentru a obține în mod real acest lucru algoritmul trebuie detaliat, în sensul că, de exemplu, în **Paşii** de tip **3** (de alegere a unei clauze corespunzătoare C), selecția variabilei de marcat trebuie făcută prin parcurgerea de un număr fix de ori (*independent* de numărul de execuții) a listei variabilelor peste care este construită F. ■

Exemplu. Să aplicăm algoritmul de marcare următoarei formule Horn:
$F = (A \vee \rceil D) \wedge (\rceil C \vee \rceil A \vee D) \wedge (\rceil A \vee \rceil B) \wedge D \wedge \rceil E$.
Scriem întâi F ca o mulțime de implicații, obținând $F = \{D \rightarrow A, C \wedge A \rightarrow D, A \wedge B \rightarrow \mathbf{0}, \mathbf{1} \rightarrow D, E \rightarrow \mathbf{0}\}$. Înainte de prima execuție a corpului buclei, avem i = 0 şi toate variabilele sunt *nemarcate*.

- **Prima execuție**. Alegem clauza $\mathbf{1} \rightarrow D$ (de fapt, nu există altă posibilitate). Toate aparițiile lui D se *marchează* cu $*_1$:
  $D_{*1} \rightarrow A, C \wedge A \rightarrow D_{*1}, A \wedge B \rightarrow \mathbf{0}, \mathbf{1} \rightarrow D_{*1}, E \rightarrow \mathbf{0}$.
- **A doua execuție**. Alegem $D \rightarrow A$ (din nou, nu există decât o unică posibilitate) şi A se *marchează* peste tot, cu $*_2$:
  $D_{*1} \rightarrow A_{*2}, C \wedge A_{*2} \rightarrow D_{*1}, A_{*2} \wedge B \rightarrow \mathbf{0}, \mathbf{1} \rightarrow D_{*1}, E \rightarrow \mathbf{0}$.
- **A treia execuție nu mai are loc**, deoarece nu mai există clauze de tipul cerut. Cum valoarea lui i nu s-a modificat (a rămas 0), răspunsul algoritmului este „DA”.

Prin urmare, F este satisfiabilă şi o structură S, model pentru F, este definită prin $S(A) = 1$, $S(B) = 0$, $S(C) = 0$, $S(D) = 1$, $S(E) = 0$. ■

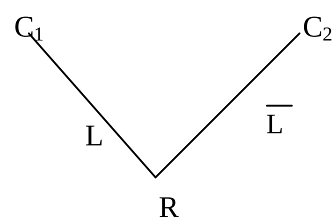
Am găsit prin urmare o subclasă „convenabilă” (acest lucru este cumva subiectiv) de formule propoziționale, si anume clasa formulelor Horn, pentru care testarea satisfiabilității se poate face într-un timp „rezonabil”. Deşi rezultatele teoretice generale ne spun că nu pot exista *metode sintactice mai bune* dacât metoda semantică sugerată de **Algoritmul SAT** (dacă ne referim la *întrega* mulțime **LP**), existența, dovedită de acum, a unor algoritmi care să nu facă apel explicit la semantică, pare deja a fi un câştig.

§7. Rezoluție în LP

Fără a restrânge generalitatea, putem presupune că lucrăm cu formule din **LP** aflate în **FNC**, reprezentate sub formă de mulțimi (finite) de clauze, iar clauzele ca mulțimi (finite) de literali.

Definiția 2.9 (rezolvent). Fie clauzele C_1, C_2 , R. Spunem că **R este rezolventul lui C_1, C_2** (sau că **C_1, C_2 se rezolvă în R**, sau că **R se obține prin rezoluție într-un pas din C_1, C_2**), pe scurt, $R = Res_L(C_1, C_2)$, dacă şi numai dacă există un literal $L \in C_1$ astfel încât $\overline{L} \in C_2$ şi $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$. ■

Vom putea reprezenta acest lucru şi grafic, prin *arborele de rezoluție*:



Vom renunța la scrierea explicită a lui L sau/şi $\overline{L}$ în momentul în care nu există cofuzii.

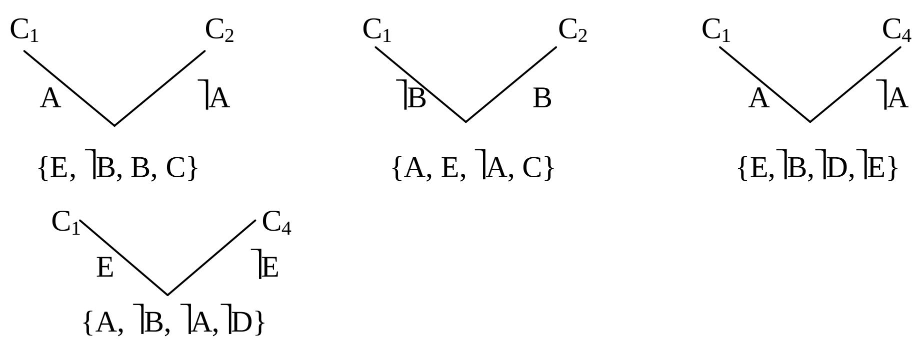
Observație. Rezolventul a două clauze este tot o clauză. Mai mult, rezolventul a două clauze Horn este tot o clauză Horn. Clauza vidă ( )

poate fi obținută prin rezoluție din două clauze de forma $C_1 = \{A\}$ şi $C_2 = \{\neg A\}$. În definiția anterioară putem considera că C_1 şi C_2 sunt diferite între ele. Dacă ele ar coincide, atunci $C_1 = C_2 = \ldots \vee L \vee \overline{L} \vee \ldots \equiv \mathbf{1}$, adică acele clauze sunt tautologii, detectabile sintactic (în acest caz nu ne mai interesează alte metode formale de studiere a satisfiabilității lor).

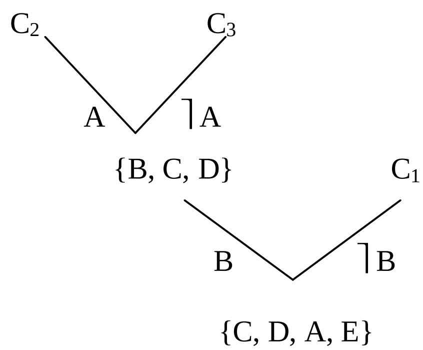
■

Exemplu.

Fie formula $F = \{\{A, E, \neg B\}, \{\neg A, B, C\}, \{A, D\}, \{\neg A, \neg D, \neg E\}\}$. Să găsim câțiva dintre rezolvenții care se pot obține (succesiv) pornind de cele cele patru clauze care compun F, notate respectiv C_1, C_2, C_3, C_4:



Aceştia au fost găsiți apelând de fiecare dată la C_1. Și C_2 poate fi sursa unui întreg ”lanț” de asemenea rezolvenți:



Mulți dintre aceşti rezolvenți „primari” nu sunt interesanți, fiind tautologii *(datorită faptului că acele clauze alese spre rezolvare conțin mai mult de un literal de tipul* L/$\overline{\text{L}}$ *)*. Procesul poate însă continua cu găsirea de noi rezolvenți folosindu-i şi pe cei obținuți din clauzele inițiale (cum este cazul şi mai sus) ş.a.m.d. ■

În acest moment putem să ne punem cel puțin două întrebări:

- Există cazuri în care procesul anterior (de aflare succesivă de rezolvenți noi) nu se termină?
- În caz de răspuns negativ şi presupunând că există o legătură între acest proces sintactic (de obținere de rezolvenți) şi satisfiabilitate, se pot obține algoritmi (sintactici, eventual performanți) de testare a satisfiabilității unor formule?

Răspunsul îl vom da în cele ce urmează.

Teorema 2.9 (lema rezoluției). Fie oricare formulă $F \in$ **LP** (aflată în **FNC** şi reprezentată ca mulțime de clauze) şi R un rezolvent pentru C_1, $C_2 \in F$. Atunci F este tare echivalentă cu $F \cup \{R\}$.

Demonstrație.

„$\Leftarrow$". Dacă $\mathcal{S}$ satisface $F \cup \{R\}$ atunci desigur că $\mathcal{S}$ satisface F, conform definiției (o structură satisface o mulțime de formule dacă satisface fiecare element din mulțime).

„$\Rightarrow$". Să presupunem că $\mathcal{S} \models F$, adică $\mathcal{S} \models C$, pentru fiecare $C \in F$. Fie $C_1, C_2 \in F$ şi R un rezolvent al lor, $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$, unde $L \in C_1$, $\overline{L} \in C_2$.

Cazul 1. $\mathcal{S}(L) = 1$. Atunci $\mathcal{S} \models L$. Dar ştim că $\mathcal{S} \models C_2$. Rezultă că $\mathcal{S} \models C_2 \setminus \{\overline{L}\}$, de unde $\mathcal{S}(R) = 1$.

Cazul 2. $\mathcal{S}(L) = 0$. Analog, arătându-se că $\mathcal{S} \models C_1 \setminus \{L\}$. ∎

În teorema anterioară am fi putut considera, în loc de F, o mulțime oarecare de clauze, chiar infinită.

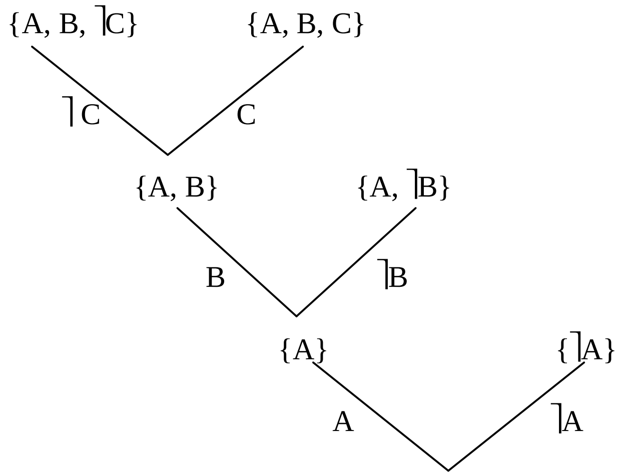
Definiția 2.10. Fie F o mulțime oarecare de clauze din **LP** şi C o clauză. Spunem că lista $C'_1, C'_2, \ldots, C'_m$ este o **demonstrație prin rezoluție (în mai mulți paşi) a lui C pornind cu F** dacă sunt satisfăcute condițiile:

(i) Pentru fiecare $i \in [m]$, fie $C'_i \in F$, fie C'_i este obținut prin rezoluție într-un pas din C'_j, C'_k, cu $j, k < i$.

(ii) $C = C'_m$. ∎

În condițiile definiției, se mai spune că **C este demonstrabilă prin rezoluție** (pornind cu F, sau, **în ipotezele date de F**). Mai mult, pentru a spune acest lucru, este suficient ca F să fie **inserată** (prezentă) într-o demonstrație şi nu să fie neapărat ultimul element al acesteia. Intuitiv, o demonstrație prin rezoluție în mai mulți paşi înseamnă o succesiune finită de rezoluții într-un pas, care poate fi reprezentată şi grafic, printr-un arbore (a se vedea exemplul care urmează), sau chiar ca un graf oarecare (dacă nu folosim noduri diferite pentru aparițiile distincte ale unei aceleiaşi clauze). În particular, dacă C este clauza vidă, atunci demonstrația respectivă se numeşte şi **respingere**. **Numărul de paşi** dintr-o demonstrație este dat de *numărul de clauze obținute prin rezoluție într-un pas* (din clauze anterioare). Acesta poate fi considerat ca fiind o măsură a „mărimii” (*lungimii*) demonstrației. O altă măsură pentru o demonstrație reprezentată ca text poate fi chiar lungimea listei (numărul total de clauze, sau chiar numărul total de clauze distincte). Dacă reprezentăm o demonstrație ca un arbore, putem folosi şi măsuri specifice, cum ar fi *adâncimea* arborelui, *numărul de nivele* ([IVA]), etc. *Convenim să eliminăm din orice demonstrație rezolvenții care conțin atât L cât şi ù L, aceste clauze fiind tautologii şi deci neinteresante din punctul de vedere al studiului satisfiabilității unei formule aflate în **FNC**.*

Exemplu. Fie F = {{A, B, ⌉C}, {⌉A}, {A, B, C}, {A, ⌉B}}. O respingere poate fi descrisă prin arborele:



■

Definiția 2.11 (mulțimea rezolvenților unei mulțimi de clauze). Fie F o mulțime de clauze din **LP** (nu neapărat finită). Notăm succesiv:

- $Res(F) = F \cup \{R \mid \text{există } C_1, C_2 \in F \text{ astfel încât } R = Res(C_1, C_2)\}$.
- $Res^{(n+1)}(F) = Res(Res^{(n)}(F))$, $n \in \mathbf{N}$. Prin $Res^{(0)}(F)$ vom înțelege F şi atunci vom putea pune şi $Res^{(1)}(F) = Res(F)$.
- $Res^{*}(F) = \bigcup_{n \in \mathbf{N}} Res^{(n)}(F)$.

$Res^{(n)}(F)$ se va numi **mulțimea rezolvenților lui F obținuți în cel mult n paşi**, iar $Res^{*}(F)$ **mulțimea (tuturor) rezolvenților lui F**. ■

Observație. Direct din definiție rezultă că:

$$F = Res^{(0)}(F) \subseteq Res^{(1)}(F) \subseteq \ldots \subseteq Res^{(n)}(F) \subseteq \ldots \subseteq Res^{*}(F).$$

Putem da atunci şi o ***definiție structurală a lui Res*(F)***. Vom nota astfel cu Resc mulțimea definită prin:

Baza. F $\subseteq$ Resc.

Pas constructiv: Dacă C_1, C_2 $\in$ Resc şi C = Res(C_1, C_2), atunci C $\in$ Resc. ■

Rămâne să arătăm că cele două definiții introduc aceeaşi mulțime.

Teorema 2.10. Pentru fiecare F $\in$ **LP**, avem Res*(F) = Resc.

Demonstrație. Arătăm egalitatea prin dublă incluziune.

„Í”. Demonstrăm prin inducție matematică adevărul afirmației din metalimbaj ($\forall$n)P(n), unde P(n): Res$^{(n)}$(F) $\subseteq$ Resc.

Baza. n = 0. Trebuie arătat că F = Res$^{(0)}$(F) $\subseteq$ Resc, ceea ce este imediat din definiția lui Resc.

Pas inductiv. Presupunem că Res$^{(n)}$(F) $\subseteq$ Resc şi arătăm că Res$^{(n+1)}$(F) $\subseteq$ Resc, ceea ce este din nou imediat din definiția lui Resc şi **Definiția 2.11**. În sfârşit, avem Res*(F) $\subseteq$ Resc, direct din **Definiția 2.11** şi observația care urmează acesteia.

„Ê”. Procedăm prin inducție structurală, mai exact arătăm că afirmația din metalimbaj ($\forall$C $\in$ Resc)(C $\in$ Res*(F)) este adevărată.

Baza. C $\in$ F. Adevărat, deoarece F = Res$^{(0)}$(F) $\subseteq$ Res*(F).

Pas inductiv. Fie C = Res(C_1, C_2), C_1, $C_2 \in$ Resc şi resupunem că C_1, $C_2 \in$ Res^*(F). Să arătăm că C $\in$ Res^*(F). Acest fapt urmează imediat, conform **Definiției 2.11**. ■

De acum înainte vom folosi ambele notații pentru mulțimea rezolvenților unei mulțimi de clauze. Şi în **Teorema 10** se putea considera că F reprezintă o mulțime oarecare de clauze.

Teorema 2.11. Fie F o mulțime de clauze din **LP** (nu neapărat finită). O clauză C $\in$ **LP** se poate demonstra prin rezoluție pornind cu clauzele lui F dacă şi numai dacă există k $\in$ **N**, asfel încât C $\in$ $Res^{(k)}$(F).

Demonstrație. Fie F şi C fixate ca în enunț.

„Þ”. Să presupunem că există o demonstrație prin rezoluție a lui C pornind cu F, C'_1, C'_2, ... , C'_m = C. Este îndeplinită condiția (i) din **Definiția 2.10** şi atunci înseamnă că pentru fiecare i $\in$ [m], avem C'_i $\in$ Resc, care coincide cu Res^*(F), conform **Teoremei 2.10**. Prin urmare, conform definiției lui Res^*(F) există k $\in$ **N**, asfel încât C $\in$ $Res^{(k)}$(F).

„Ü”. Să presupunem că există k $\in$ **N**, asfel încât C $\in$ $Res^{(k)}$(F) (pe *k îl considerăm a fi cel mai mic număr natural care satisface condiția*). Conform **Definiției 2.11**, avem $Res^{(j)}$(F) = Res($Res^{(j-1)}$(F)) = $Res^{(j-1)}$(F) $\cup$ {R | există C_1, C_2 $\in$ $Res^{(j-1)}$(F) astfel încât R = Res(C_1, C_2)}, pentru fiecare j $\in$ [k]. *Putem conveni chiar ca în a* *doua mulțime din reuniunea de mai sus să nu punem decât rezolvenții noi, care nu apar în* $Res^{(j-1)}(F)$. Atunci C apare efectiv în $Res^{(k)}(F)$ dar nu şi în $Res^{(k-1)}(F)$. Dacă k = 0, am terminat ($C \in F$ şi lista formată doar din *C constituie o demonstrație prin rezoluție a lui C*). În caz contrar, mai întâi *construim algoritmic un graf neorientat* în felul următor: la **Pasul 1** punem ca noduri elementele din $Res^{(k)}(F)$, care nu sunt şi în $Res^{(k-1)}(F)$; la **Pasul 2** punem nodurile din $Res^{(k-1)}(F)$ care nu sunt în $Res^{(k-2)}(F)$, precum şi *muchiile* corespunzătoare care unesc nodurile puse deja în graf, conform rezoluțiilor într-un pas din care ele provin, ş. a. m. d. În cel mult **k + 1 paşi**, vom plasa în graf şi elementele (folosite) ale lui F, precum şi toate muchiile corespunzătoare rezoluțiilor într-un pas cu ajutorul cărora se construieşte $Res^{(k)}(F)$. Considerăm acum *subgraful generat de nodul C şi toate nodurile aflate pe lanțuri de la C la frunze (*[IVA]*)*. Acesta este un *arbore cu rădăcina C, care reprezintă o demonstrație a lui C pornind cu o submulțime a lui F, deci şi cu F* (desigur că demonstrația se obține prin „listarea” corespunzătoare a nodurilor, ultimul element din listă fiind C). Dacă subgraful considerat nu este arbore, acest lucru se datorează faptului că măcar o clauză C' este utilizată în mai mulți paşi de rezoluție. Graful poate fi uşor transformat în arbore prin multiplicarea nodurilor de tipul C' şi a arcelor aferente. ■

După cum probabil s-a putut observa, în cele de mai sus am folosit în majoritatea cazurilor termenul *mulțimea de clauze F* şi nu *formula F (aflată în FNC)*. Deşi pe noi ne interesează doar formulele (care pot fi privite ca mulțimi **finite** de clauze în cazul în care ne interesează doar satisfiabilitatea lor), aproape toate rezultatele sunt valabile şi pentru mulţimi infinite (numărabile) de formule (clauze). Teorema următoare stabileşte o legătură importantă, privind satisfiabilitatea, între mulţimile infinite şi cele finite de formule oarecare din **LP**.

Teorema 2.12 (de compactitate pentru LP). Fie **M** o mulţime infinită (numărabilă) de formule din **LP**. M este satisfiabilă dacă şi numai dacă fiecare submulţime finită a sa este satisfiabilă.

Demonstraţie.

„ Þ ”. Dacă există structura *S* astfel încât *S* ‘ **M**, adică *S*(F) = 1 pentru fiecare F ∈ **M**, atunci evident că acelaşi lucru se întâmplă pentru fiecare submulţime (finită) **M**' ⊆ **M**.

„ Ü ”. Pentru fiecare n ∈ **N**, vom nota $\mathbf{M_n}$ **@** {F ∈ **M** | subf(F) ∩ *A* = prop(F) ⊆ A_n}, adică mulţimea formulelor din **M** care sunt construite peste (cel mult) mulţimea de variabile propoziţionale A_n = {A_1, A_2, … , A_n}. Cum mulţimea funcţiilor booleene de n variabile ($FB^{(n)}$) are cardinalul 2^{2^n}, în $\mathbf{M_n}$ există cel mult 2^{2^n} formule cu tabele de adevăr distincte. Mai mult, direct din definiţii avem $\mathbf{M_1} \subseteq \mathbf{M_2} \subseteq \ldots \subseteq \mathbf{M_n} \subseteq \ldots \subseteq \mathbf{M}$ şi $\mathbf{M} = \bigcup_{\mathbf{n}=1}^{\infty} \mathbf{M_n}$. Revenind, să presupunem că fiecare submulţime finită a lui **M** este satisfiabilă şi să arătăm că **M** este satisfiabilă. Fie $\mathbf{K} \subseteq \mathbf{M}$ orice submulțime finită (satisfiabilă) a lui **M**. Atunci există n, natural, astfel încât $\mathbf{K} \subseteq \mathbf{M_n}$. Fie $\mathbf{M'_n} = \{F_1, F_2, \dots, F_{k_n}\}$, $k_n \leq 2^{2^n}$, mulțimea elementelor lui $\mathbf{M_n}$ care au tabele de adevăr distincte. Pentru fiecare formulă G din **K** alegem o formulă şi numai una, F_i, din $\mathbf{M'_n}$, astfel încât $G \equiv F_i$. Fie $\mathbf{M''_n}$ mulțimea tuturor acestor formule, pentru care este satisfăcută condiția: ***K** este satisfiabilă dacă şi numai dacă $\mathbf{M''_n}$ este satisfiabilă*. Fie atunci $\mathcal{S}_n$ un model pentru $\mathbf{M''_n}$. Avem şi $\mathcal{S}_n \models M_n$ pentru că pentru fiecare $F \in \mathbf{M_n} \setminus \mathbf{M''_n}$, există $G \in \mathbf{M''_n}$ astfel încât $\mathcal{S}_n(G) = \mathcal{S}_n(F)$. Din ipoteza noastră (fiecare submulțime finită a lui **M** este satisfiabilă) rezultă aşadar că există un şir de structuri care satisfac:

$$\mathcal{S}_1 \models \mathbf{M_1}, \mathcal{S}_2 \models \mathbf{M_2}, \dots, \mathcal{S}_n \models \mathbf{M_n}, \dots$$

Renumerotând dacă este cazul variabilele inițiale, putem presupune că fiecare dintre mulțimile de mai sus este nevidă şi că modele sunt „construite succesiv”, după cum este descris în ceea ce urmează. $\mathcal{S}_1$ există şi este indiferent modul său de obținere. Apoi, pentru fiecare $i \in \{1, 2, ...\}$, construim $\mathcal{S}_{i+1}$ pornind de la $\mathcal{S}_i$ (şi modificând eventual şi pe $\mathcal{S}_{i-1}, \dots, \mathcal{S}_1$), în felul următor: $\mathcal{S}_{i+1}$ „pleacă” cu valorile de adevăr pentru $A_1, A_2, \dots, A_i$ stabilite de către $\mathcal{S}_i$ şi „fixează” o valoare pentru A_{i+1}, în mod aleator. Dacă nu avem $\mathcal{S}_{i+1} \models \mathbf{M_{i+1}}$, atunci revenim, alegând o structură $\mathcal{S}_{i+1}$ care să satisfacă $\mathbf{M_{i+1}}$ (ştim că există), prin schimbarea, eventual, şi a structurilor anterioare $\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_i$ (acestea vor fi simple restricții ale lui $\mathcal{S}_{i+1}$, lucrul fiind evident posibil deoarece $\mathbf{M_i} \subseteq \mathbf{M_{i+1}}$). Ca urmare, putem defini structura $\mathcal{S} : \mathcal{A} \to \{0,1\}$, dată prin $\mathcal{S}(A_i) = \mathcal{S}_i(A_i)$, pentru fiecare $i \in$ **N***. Faptul că $\mathcal{S}$ este funcție și model pentru **M** este imediat. ■

Teorema 2.13. Fie $F \in$ **LP**, aflată în **FNC** și reprezentată ca mulțime (finită) de clauze. Atunci $Res^*(F)$ este finită.

Demonstrație. Arătăm mai întâi că *există un k Î N astfel încât* $Res^{(k)}(F) = Res^{(k+1)}(F)$. Fie $|\,prop(F)| = n$. Numărul total (m, să spunem) al clauzelor peste (cel mult) n variabile atomice date este finit (de fapt, $m = 3^n$). *Orice rezoluție într-un pas „șterge" câte un literal*. Prin urmare, indiferent câte dintre cele m posibile clauze sunt prezente inițial în F și oricâți pași de rezoluție am efectua, cardinalul *oricărui* $Res^{(i)}(F)$ *nu poate depăși m*. Datorită acestui fapt și existenței incluziunii $Res^{(i)}(F) \subseteq Res^{(i+1)}(F)$ (pentru fiecare $i \in$ **N**), afirmația noastră se deduce imediat. Mai mult, notând cu j pe cel mai mic k cu proprietatea de mai sus, *avem* $Res^{(j)}(F) = Res^{(j+l)}(F)$, *pentru fiecare l Î N* (lucru care rezultă printr-o simplă inducție matematică și folosind **Definiția 2.11**). De aici conchidem imediat că $Res^{(j)}(F) = Res^*(F)$, de unde $card(Res^*(F)) \leq m$. ■

Reamintind că vom elimina din orice mulțime de forma $Res^*(F)$, pe măsură ce se obțin, toate clauzele care conțin o subformulă de tipul $A \vee \rceil A$, enunțăm cea mai importantă teoremă din acest capitol.

Teorema 2.14 (teorema rezoluției pentru calculul propozițional). Fie F o mulțime oarecare de clauze din calculul propozițional. Atunci F este nesatisfiabilă dacă şi numai dacă $\in$ Res*(F).

Demonstrație. Conform **Teoremei de compactitate**, ştim că F este nesatisfiabilă dacă şi numai dacă *există o submulțime finită* a sa care este nesatisfiabilă. Din acest motiv, în cele de mai jos vom presupune că F este o mulțime finită de clauze, sau, alternativ, o formulă propozițională aflată în **FNC**. Fără a restrânge generalitatea, putem presupune deci că F este o formulă oarecare din **LP** (**Teorema 2.6**).

„Ü” (corectitudine). Să presupunem că $\in$ Res*(F) şi să arătăm că F este nesatisfiabilă. Conform **Definiției 2.11** şi aplicării repetate (de un număr finit de ori) a **Lemei rezoluției**, avem F $\equiv$ Res$^{(1)}$(F) $\equiv$ Res$^{(2)}$(F) $\equiv$ … $\equiv$ Res$^{(n)}$(F) $\equiv$ … . Dacă $\in$ Res*(F) atunci există k $\in$ **N**, astfel încât $\in$ Res$^{(k)}$(F), adică Res$^{(k)}$(F) este nesatisfiabilă (este nesatisfiabilă prin convenție). Cum F $\equiv$ Res$^{(k)}$(F), rezultă că F este nesatisfiabilă.

„Þ” (completitudine). Să presupunem că F este nesatisfiabilă şi să arătăm că $\in$ Res*(F). Fie n = card(prop(F)). Procedăm prin inducție asupra lui n, adică demonstrăm astfel adevărul metateoremei ($\forall$n $\in$ **N**) (| prop(F)| = n şi F este nesatisfiabilă $\Rightarrow$ $\in$ Res*(F)).

Baza. n = 0. Aceasta înseamnă că F = { } = Res*(F) şi concluzia este evidentă.

Pas inductiv. *Presupunem afirmația adevărată pentru formule construite peste n variabile propoziționale şi o demonstrăm pentru* *formule construite peste n+1 formule atomice*. Fie F $\in$ **LP**, construită peste $\mathcal{A}_{n+1}$ = {A_1, A_2, … , A_n, A_{n+1}}. Pornind de la această formulă vom construi alte două formule, notate $\mathbf{F}_0^{A_{n+1}}$ şi respectiv $\mathbf{F}_1^{A_{n+1}}$, în modul următor:

- $\mathbf{F}_0^{A_{n+1}}$ se formează din F prin eliminarea sintactică a oricărei apariţii a literalului pozitiv A_{n+1} din orice clauză şi apoi eliminarea *în totalitate* a tuturor clauzelor care conţin o apariţie negativă a literalului A_{n+1}.
- $\mathbf{F}_1^{A_{n+1}}$ se obţine prin *dualizare*, adică din F se scot din toate clauzele apariţiile negative ale lui A_{n+1} şi se elimină apoi toate clauzele care conţin apariţii pozitive ale aceleiaşi variabile.

Afirmaţie. *Dacă F este nesatisfiabilă, atunci atât* $\mathbf{F}_0^{A_{n+1}}$ *cât şi* $\mathbf{F}_1^{A_{n+1}}$ *sunt nesatisfiabile.* Să presupunem că F este nesatisfiabilă şi nu are clauze care sunt tautologii. Fie $\mathbf{F}_0^{A_{n+1}}$ şi fie $\mathcal{S}$ orice structură corectă (definită pentru toate variabilele propoziţionale care intervin în formulele considerate). Considerând clauzele C ale lui $\mathbf{F}_0^{A_{n+1}}$, avem următoarele posibilităţi :

- *C este o clauză din F, nemodificată*. Evident că valoarea lui $\mathcal{S}$ pentru această clauză nu se modifică şi astfel nu modifică valoarea de adevăr a lui $\mathbf{F}_0^{A_{n+1}}$ faţă de cea a lui F (dacă luăm în considerare doar această clauză).
- *C provine dintr-o clauză din F, care conţinea în plus o apariţie a lui A_{n+1}*. Dacă $\mathcal{S}(A_{n+1}) = 0$, atunci $\mathcal{S}(C \cup \{A_{n+1}\}) = \mathcal{S}(C)$, din

nou valoarea de adevăr a lui $\mathbf{F}_0^{A_{n+1}}$ nemodificându-se față de cea a lui F (relativ la C). Dacă $\mathcal{S}(A_{n+1}) = 1$, avem $\mathcal{S}(C \cup \{A_{n+1}\}) = 1$. Cum F este nesatisfiabilă, înseamnă că există o altă clauză C', $C \cup \{A_{n+1}\} \neq C' \in F$ cu $\mathcal{S}(C') = 0$. Este evident că C' nu poate conține pe A_{n+1} deoarece $\mathcal{S}(A_{n+1}) = 1$ şi nici pe $\rceil A_{n+1}$ (în acest caz C nu ar mai fi apărut în $\mathbf{F}_0^{A_{n+1}}$). Prin urmare, C' apare şi în $\mathbf{F}_0^{A_{n+1}}$, de unde urmează imediat că $\mathcal{S}(\mathbf{F}_0^{A_{n+1}}) = 0$.

Mai există posibilitatea ca nesatisfiabilitatea lui F să fi provenit din faptul că $\mathcal{S}(C) = 0$ pentru o clauză $C \in F$ care conține neapărat $\rceil A_{n+1}$, restul clauzelor lui F, ca şi cele ale lui $\mathbf{F}_0^{A_{n+1}}$, fiind adevărate în $\mathcal{S}$. Acest lucru înseamnă că în această structură avem $\mathcal{S}(A_{n+1}) = 1$. Să considerăm structura $\mathcal{S}'$ care coincide cu $\mathcal{S}$, exceptând valoarea lui A_{n+1}, care este pusă pe 0. Conform celor de mai sus, avem imediat S'(F) =1, ceea ce este absurd, F fiind nesatisfiabilă. Rezultă că $\mathbf{F}_0^{A_{n+1}}$ este nesatisfiabilă. Se procedează similar pentru $\mathbf{F}_1^{A_{n+1}}$. **(q. e. d.)**

În acest moment ştim că formulele $\mathbf{F}_0^{A_{n+1}}$ şi $\mathbf{F}_1^{A_{n+1}}$ sunt nesatisfiabile şi, mai mult, sunt construite peste cel mult n variabile. Aplicând ipoteza inductivă pentru aceste formule rezultă că $\in \mathrm{Res}^*(\mathbf{F}_0^{A_{n+1}})$ şi $\in \mathrm{Res}^*(\mathbf{F}_1^{A_{n+1}})$. Conform **Teoremelor 2.11** şi **2.13**, există o respingere (D_0) $C_1, C_2, \ldots, C_l =$, pornind cu elementele lui $\mathbf{F}_0^{A_{n+1}}$, precum şi o respingere (D_1) B_1, B_2, … B_t = , pornind cu clauzele lui $\mathbf{F}_1^{A_{n+1}}$. Adăugăm acum la fiecare clauză din (D_0) pe A_{n+1}, peste tot de unde acesta a fost scos (inclusiv la clauzele rezultate în urma aplicării rezoluției într-un pas), obținând o demonstrație prin rezoluție notată (D_0') și, analog, adăugăm la fiecare element din (D_1) pe $\rceil A_{n+1}$ acolo unde este necesar, obținând o altă demonstrație prin rezoluție, notată (D_1') (odată A_{n+1} respectiv $\rceil A_{n+1}$ introduse, ele nu vor mai fi șterse). Sunt posibile următoarele situații:

- Ultima clauză a lui (D_0') este $C'_l = \{A_{n+1}\}$ și ultima clauză a lui (D_1') rămâne $B'_t = B_t =$ (sau invers, $C'_l = C_l =$ și $B'_t = B_t = \{\rceil A_{n+1}\}$). Atunci concatenăm cele două liste care reprezintă demonstrațiile (D_0') și (D_1'), rezultând evident o respingere pornind cu clauzele lui F.
- Ultima clauză lui (D_0') este $C'_l = \{A_{n+1}\}$și ultima clauză a lui (D_1') este $B'_t = \{\rceil A_{n+1}\}$. Atunci concatenăm din nou cele două liste și apoi mai facem un pas de rezoluție obținând clauza finală $C = Res(C'_l, B'_t) =$. Din nou avem o respingere pornind cu clauzele lui F.

În ambele situații, conform **Teoremei 2.11**, rezultă că $\in Res^*(F)$. ■

În urma acestui rezultat, putem concluziona că *există algoritmi sintactici pentru a testa nesatisfiabilitatea formulelor* din logica propozițională, ei rămânând (din păcate, dar nesurprinzător) exponențiali ca timp de execuție. Lucrările [MAS4] și [MAS5] pot fi consultate pentru alte detalii legate de complexitatea rezoluției. Să remarcăm şi faptul că ***testarea satisfiabilității*** nu implică nimic special (în acest caz, condiția de verificat va fi $\notin$ Res*(F)), ca de altfel nici ***testarea validității*** (F este validă dacă şi numai dacă $\rceil$ F este contradicție; prin urmare, putem aplica **Teorema 2.14** lui $\rceil$ F). ***A testa dacă o formulă F este satisfiabilă dar nevalidă*** impune însă aplicarea teoremei anterioare atât pentru F (F este satisfiabilă dacă $\notin$ Res*(F)) cât şi pentru $\rceil$ F ($\rceil$ F este satisfiabilă dacă $\notin$ Res*($\rceil$ F)). Singura şansă (oricât de puțin probabilă ar părea) de a găsi algoritmi performanți rămâne aceea de a căuta subclase ale lui **LP** „suficient de interesante din punct de vedere practic”, pentru care asemenea algoritmi să existe (avem deja un exemplu: clasa formulelor Horn). Avantajul în acest moment este că aceste subclase pot fi selecționate ținându-se cont (numai) de motivații sintactice.

§8. Rafinări ale rezoluției: strategii şi restricții

Rafinările rezoluției sunt metode prin care se urmăreşte *obținerea clauzei vide (dacă acest lucru este posibil) într-un număr cât mai mic de paşi de rezoluție.* Pornind cu formula F = {C_1, C_2, … , C_n} (aflată în **FNC** şi scrisă ca o mulțime de clauze, la rândul lor clauzele fiind scrise ca mulțimi de literali), se poate construi efectiv mulțimea Res*(F), care poate fi reprezentată (fiind finită), după cum deja ştim, ca un *graf neorientat* (nodurile sunt rezolvenții succesivi, inclusiv clauzele inițiale, iar muchiile sunt introduse prin rezoluțiile într-un pas aplicate). Practic, *acest graf ar trebui să cumuleze* ***toate*** *posibilele demonstrații prin rezoluție care pornesc cu clauzele lui F* (reamintim că anumiți rezolvenți sunt totuşi excluşi, deoarece reprezintă tautologii). **Teorema rezoluției** sugerează crearea mai întâi a acestui *graf de rezoluție total* şi apoi *parcurgerea* lui pentru a vedea dacă   este (eticheta unui) nod în graf. **Teorema 2.11** ne indică faptul că *este suficient să găsim o respingere* în loc de ***a creea*** şi *apoi parcurge* întregul graf. Rafinările se împart în două mari categorii: *strategii* şi *restricții*.

Strategiile nu restrâng, în general, spațiul de căutare (adică *graful total*) dar folosesc anumite *informații suplimentare despre clauze*, astfel încât să crească şansele pentru selectarea rapidă a unei demonstrații căutate, adică a unui „cel mai scurt drum" pornind de la frunze (elementele lui F), către o rădăcină (clauza vidă). Astfel, cel puțin la modul ideal, *graful total nu se construieşte în întregime, ci doar acele porțiuni din el (cât mai puține şi cât mai mici), care este posibil să „conțină" măcar o respingere*. Cel mai cunoscut exemplu este **strategia unitară**, în care ***se recomandă*** ca *la fiecare pas al rezoluției măcar una dintre clauze să conțină un singur literal* (dacă însă nu mai poate fi aleasă nici o asemenea *clauză unitară*, se continuă procesul de obținere de noi rezolvenți în mod obişnuit). Prin urmare, *strategiile nu distrug completitudinea rezoluției* (*dacă o formulă este nesatisfiabilă, atunci se poate demonstra acest lucru prin rezoluție, găsindu-se o respingere*), dar, în cel mai rău caz, este posibil să nu conducă la nici o economie de timp.

Pe de altă parte, **restricțiile** *distrug în multe situații completitudinea rezoluției (există formule nesatisfiabile pentru care nu se pot găsi respingeri, în situația în care paşii de rezoluție sunt supuşi unor condiții prea restrictive)*, deoarece *spațiul de căutare este practic* *micşorat* într-un mod, să-i spunem, abuziv. Astfel, o anumită restricție poate *interzice total* folosirea (într-un pas de rezoluție) a unor clauze având o anumită formă sintactică. *Restricțiile rămân însă complete pentru anumite subclase de formule propoziționale*. Există mai multe exemple importante de restricții, câteva dintre ele fiind trecute în continuare în revistă.

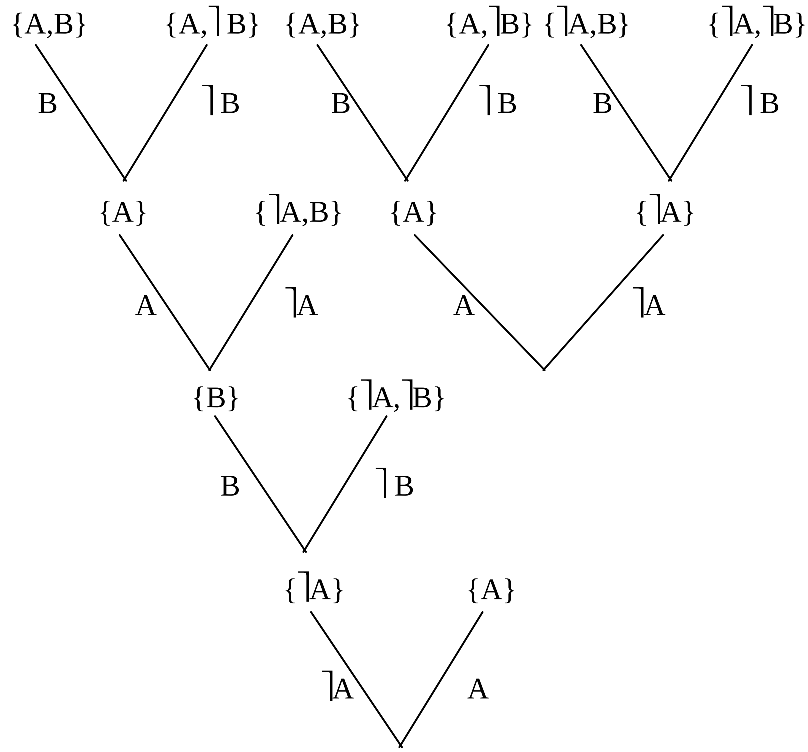
Rezoluția pozitivă (P-rezoluția). La fiecare pas al rezoluției (al unei demonstrații prin rezoluție), măcar una dintre clauze trebuie să fie o clauză pozitivă.

Rezoluția negativă (N-rezoluția). La fiecare pas al rezoluției măcar una dintre clauze se cere să fie negativă.

Rezoluția liniară bazată pe o clauză inițială. Fie $F \in$ **LP**, $F = \{C_1, C_2, \dots , C_n\}$ o mulțime de clauze (numite şi *clauze de intrare*) şi o clauză fixată $C \in F$ (numită *clauză inițială* sau *clauză de bază*). O *rezoluție liniară bazată pe C* este o (demonstrație prin) rezoluție în care la fiecare pas se aleg spre a fi rezolvate două clauze C_1 şi C_2, dintre care C_1 este rezolventul pasului anterior, iar C_2 (*clauza suplimentară, definită, exactă, precisă, de program*) este fie o clauză de intrare, fie un rezolvent obținut anterior în demonstrație. La primul pas, avem $C_1 = C$ şi $C_2 \in F$. În particular, se poate introduce în plus o *funcție de selecție* pentru clauzele definite, adică o modalitate precisă (bazată pe eventuale informații suplimentare, sau pe forma sintactică a clauzelor) de *alegere* a clauzelor de tip C_2. Obținem astfel aşa-numita **SLD-rezoluție** (*rezoluție* ***L****iniară cu funcție de* ***S****elecție pentru clauzele* ***D****efinite*).

SLD-rezoluția se utilizează cu succes pentru **clauzele Horn** (alte detalii sunt în **Capitolul 5**). În acest caz, ea va fi atât o rezoluție liniară cât şi una de intrare (a se vedea mai jos). Astfel, putem considera că F este partiționată în $F_1 = \{C'_1, C'_2, ... , C'_m\}$, care sunt clauze Horn pozitive (doar acestea numindu-se aici clauze *definite*, *program*, etc.) şi $F_2 = \{N_1, N_2, ... , N_s\}$, care sunt clauze Horn negative (numite şi **clauze scop**). Pentru a obține o **SLD**-rezoluție, clauza de bază este o clauză scop, iar clauzele suplimentare trebuie să fie clauze pozitive (practic, elemente ale lui F_1, pentru că toți rezolvenții obținuți pe parcurs sunt clauze negative).

Exemplu. Să se găsească o respingere liniară bazată pe C = {A, B} şi pornind cu F = { {A, B}, {⌉A, B}, {⌉A, ⌉B}}. În cele de mai jos (sunt reprezentați doi arbori de rezoluție distincți), în stânga avem ceea ce am cerut, iar în dreapta o respingere oarecare, aceasta din urmă fiind „mai scurtă”.



■

Rezoluția bazată pe o mulțime suport. Se pornește cu formula F, precum și cu **mulțimea suport** T $\subseteq$ F, singura condiție fiind că F \ T trebuie să fie satisfiabilă (desigur că, în principiu, acest lucru trebuie să fie cunoscut aprioric și nu după aplicarea unuia dintre algoritmii deja descriși, care folosesc tot rezoluția). O demonstrație din F folosind (bazată pe) T satisface cerința că *în fiecare pas de rezoluție măcar una dintre clauze trebuie să nu aparțină lui F \ T* (situație foarte convenabilă dacă numărul de elemente din T este mic, preferabil egal cu unu). Astfel, se poate modela o situație reală în care dispunem de o anumită *bază de cunoştințe*, exprimată printr-o mulțime satisfiabilă de formule F' = $\{F_1, F_2, ... , F_n\}$, noi dorind să aflăm dacă o altă afirmație, exprimată, să zicem, prin formula G, este *consecință semantică* din F'. Conform **Teoremei 2.3**, acest lucru este echivalent cu a arăta că F = $\{F_1, F_2, ... , F_n, \rceil G\}$ este nesatisfiabilă, ceea ce se poate face utilizând rezoluția, după ce toate formulele lui F sunt „aduse" la FNC (atunci mulțimea suport T va fi constituită din formulele $G_1, G_2, ... , G_k$, adică din reprezentarea formulei $\rceil G$ ca mulțime de clauze).

Rezoluția de intrare. În orice pas al acestui tip de rezoluție, *măcar una dintre clauze trebuie să fie o clauză de intrare* (adică din mulțimea inițială F). Se observă imediat că acest tip de rezoluție poate fi considerat şi ca o rezoluție liniară, bazată pe (oricare) $C \in F$.

Rezoluția unitară. Într-o demonstrație de acest tip, *orice rezolvent poate fi obținut* ***doar dacă*** *(aici este diferența față de strategia cu acelaşi nume) măcar una dintre cele două clauze este alcătuită dintr-un unic literal*. Deoarece „lungimea" (numărul de literali ai) unui rezolvent scade cu o unitate la aplicarea oricărui pas de rezoluție, şansele de a obține „repede" clauza vidă sunt suficient de mari.

Teorema următoare o dăm fără demonstrație deoarece este importantă pentru anumite detalii legate de implementările programelor logice, detalii care nu sunt tratate nici măcar în **Capitolul 5**.

Teorema 2.16. **P**-rezoluția, **N**-rezoluția, rezoluția liniară şi rezoluția bazată pe o mulțime suport sunt complete. Rezoluția unitară, rezoluția de intrare şi **SLD**-rezoluţia sunt complete doar pentru clasa formulelor Horn. ■

§9. Recapitulare şi Index

În acest capitol am introdus **sintaxa şi semantica formală a unui limbaj logic**, privit în sensul unei **mulţimi de formule**. Acestea reprezintă, într-un mod precis, cunoştinţele noastre despre anumite părţi ale realităţii. Sunt transpuse într-o formă exactă conceptele şi principiile principale ale logicii aristotelice, printre care *bivalenţa/tertium non datur* şi *extensionalitatea.* **Sintactic**, mulţimea formulelor logicii propoziţionale – notată **LP** – poate fi definită constructiv, pornind de la o mulţime numărabilă de formule atomice şi utilizând conectorii logici şi, sau, non, eventual şi implică. O formulă poate fi „recunoscută fără dubii" (problema apartenenţei unui şir finit de caractere la **LP** este decidabilă) şi poate fi reprezentată ca un arbore. Există şi reprezentări standard ale formulelor, prin **forme normale**. **Semantica** unei formule este o valoare de adevăr (0 – **adevărat**, 1 – **fals**), valoare care se determină tot într-un mod standard. Această valoare este unică, odată ce este cunoscută o structură corectă, adică o asignare pentru formulele atomice componente. Definiţia semanticii se bazează şi pe rezultatele cunoscute despre **funcţiile booleene**. Am evidenţiat apoi clasele de formule satisfiabile, valide şi nesatisfiabile, introducându-se şi studiindu-se alte concepte de natură semantică, cum ar fi cele de echivalenţă (slabă, tare) sau de consecinţă semantică. Problema cea mai importantă de care ne-am ocupat a fost problema **SAT** (a satisfiabilităţii formuleleor din **LP**), despre care am arătat că **este decidabilă** dar **de complexitate (timp) exponențială**. Primii algoritmi descrişi erau bazați pe semantică. Din punctul de vedere al tratării automate (cu ajutorul unui calculator) a problemei **SAT**, sunt mai convenabili **algoritmii de decizie bazați pe sintaxă**, deşi aceştia nu sunt mai rapizi, la nivel global. Am prezentat un asemeanea algoritm (**Teoremele 2.11**, **2.13**, **2.14**), care foloseşte conceptul de **rezoluție (propozițională)**. Deşi metoda rezoluției nu aduce îmbunătățiri semnificative ale timpului necesar pentru rezolvarea **SAT** (în sensul teoriei generale a complexității şi pentru întreaga clasă **LP**), s-au putut pune în evidență **strategii şi restricții ale rezoluției**, care măresc şansele găsirii rapide a răspunsului, precum şi subclase de formule pentru care problema poate fi rezolvată în **timp liniar** (cum ar fi **clasa formulelor Horn**, pentru care putem aplica atât **algoritmul de marcare**, cât şi **o variantă convenabilă de implementare a SLD-rezoluției**). Se poate argumenta că mulțimea **LP** este „prea simplă" în privința „puterii" de a reprezenta lumea reală şi în consecință nu merită atenție specială. Este adevărat că **LP** poate fi inclusă în mulțimea de formule a **calculului cu predicate de ordinul I** (care va constitui obiectul de studiu al următorului capitol), dar am considerat ca benefică introducerea şi prezentarea ei separată, din motive didactice. De altfel, deşi destul de restrictivă, **LP** este suficient de „bogată" pentru a putea **exprima** (şi deci, **studia formal**) afirmații intreresante privind lumea reală. Urmărind exemplul de mai jos, vom înțelege poate mai uşor/mai exact conținutul de până în prezent al cărții.

Exemplu. Fie următoarea afirmație:

Dacă există petrol în Patagonia atunci fie experții au dreptate, fie guvernul minte. Nu există petrol în Patagonia sau experții greşesc, aşadar guvernul nu minte.

O primă întrebare ar fi: ***Formulează ea un adevăr?***

Logica ne oferă o modalitate de a răpunde la întrebare cât mai exact. Ideea este de a exprima afirmația anterioară, **cât mai adecvat**, ca o formulă F din **LP** şi apoi de a vedea dacă formula respectivă este satisfiabilă, validă sau contradicție, pentru a putea trage concluziile de rigoare. Pentru aceasta, vom izola următoarele propoziții, pe care le vom privi drept variabile (elemente din A):

P - *Există petrol în Patagonia*

E - *Experții au dreptate*

G - *Guvernul minte*

Înainte de a concepe formula, ar trebui ca afirmația să fie rescrisă, tot în limbaj natural, astfel încât să transpară mai clar ceea ce vrea să exprime. În continuare, 1., 2. şi 3. sintetizează cunoştințele prezente în afirmație:

1. Dacă *există petrol în Patagonia* atunci, sau *experții au dreptate,* sau *guvernul minte* (F_1) şi
2. *Nu există petrol în Patagonia* sau *experții greşesc* (F_2),

Înafară de aceste *ipoteze*, există şi o *concluzie*:

3. Aşadar *guvernul nu minte* (F_3).

Formula noastră ar putea fi deci:

$F = (F1 \wedge F2) \rightarrow F3$, unde $F_1 = P \rightarrow (E \vee G)$, $F_2 = \rceil P \vee \rceil E$ şi $F3 = \rceil G$, adică:

$F = ((P \rightarrow (E \vee G)) \wedge (\rceil P \vee \rceil E)) \rightarrow \rceil G$.

Să determinăm formal valoarea de adevăr a lui F.

Metoda 1 (încercăm să folosim algoritmul de marcare). Pentru aceasta, F ar trebui să fie tare echivalentă cu o formulă Horn:

$F \equiv$

(*definiția implicației)*

$\equiv \rceil((\rceil P \vee E \vee G) \wedge (\rceil P \vee \rceil E)) \vee \rceil G \equiv$

(de Morgan)

$\equiv (\rceil\rceil P \wedge \rceil E \wedge \rceil G) \vee (\rceil\rceil P \wedge \rceil\rceil E) \vee \rceil G \equiv$

(dubla negație,
asociativitate)

$\equiv (P \wedge \rceil E \wedge \rceil G) \vee ((P \wedge E) \vee \rceil G) \equiv$

(distributivitate)

$\equiv (P \wedge \rceil E \wedge \rceil G) \vee ((P \vee \rceil G) \wedge (E \vee \rceil G)) \equiv$

(distributivitate,
idempotență)

$\equiv (P \vee \rceil G) \wedge (P \vee E \vee \rceil G) \wedge (\rceil E \vee P \vee \rceil G) \wedge (\rceil E \vee E \vee \rceil G) \wedge$
$\wedge (\rceil G \vee P) \wedge (\rceil G \vee E) \equiv$

(legea tautologiei,
idempotență,
comutativitate)

$\equiv (P \vee \rceil G) \wedge (P \vee \rceil G \vee E) \wedge (P \vee \rceil G \vee \rceil E) \wedge (\rceil G \vee E) \equiv$

(asociativitate, distributivitate)

$\equiv (P \vee \rceil G) \wedge ((P \vee \rceil G) \vee (E \wedge \rceil E)) \wedge (\rceil G \vee E) \equiv$

(legea contradicției, idempotență, comutativitate)

$\equiv (P \vee \rceil G) \wedge (E \vee \rceil G)$.

Prin urmare, F poate fi considerată ca fiind formula Horn $\{G \rightarrow P, G \rightarrow E\}$ şi, printr-o aplicare imediată a **Algoritmului Horn**, obținem că F este satisfiabilă (asignarea $\mathcal{S}$ fiind dată de $\mathcal{S}(P) = \mathcal{S}(E) = \mathcal{S}(G) = 0$). Este însă F validă? Pentru aceasta putem cerceta dacă $\rceil F$ este contradicție. Obținem însă imediat (folosind legile lui de Morgan, distributivitatea şi idempotența) că $\rceil F \equiv (\rceil P \vee \rceil E) \wedge (\rceil P \vee G) \wedge (G \vee \rceil E) \wedge G$, adică, în reprezentarea cu mulțimi găsim că $\rceil F = \{P \wedge E \rightarrow 0, P \rightarrow G, E \rightarrow G, 1 \rightarrow G\}$. Aplicarea **Algoritmului Horn** conduce la *marcarea* doar a lui G şi la răspunsul că $\rceil$ F este tot satisfiabilă. În concluzie, **F este satisfiabilă dar nevalidă**, ceea ce poate apare ca un fapt ciudat dacă ne-am fi luat după forma inițială a afirmației în limbaj natural. Dacă am cunoaşte ***toate* structurile care sunt model pentru F**, acest lucru ar deveni poate explicabil. Aplicarea metodei următoare poate fi o soluție.

Metoda 2 (a tabelelor de adevăr). Ştim deja că $F \equiv (P \vee \rceil G) \wedge (E \vee \rceil G)$, de aceea nu vom folosi forma inițială, care este mai complicată. Avem:

P	E	G	$\rceil$G	P $\vee$ $\rceil$G	E $\vee$ $\rceil$G	F
0	0	0	1	1	1	1
0	0	1	0	0	0	0
0	1	0	1	1	1	1
0	1	1	0	0	1	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	1	0	1	1	1	1
1	1	1	0	1	1	1

Se observă din nou că formula este satisfiabilă, nevalidă, dar acum avem şi valorile lui F pentru toate structurile posibile. Penultima linie de exemplu, se poate rescrie în limbaj natural ca „*Afirmația* **Dacă există petrol în Patagonia, experții au dreptate şi guvernul nu minte** *este adevărată*”. Să reținem însă ***deosebirea esențială dintre afirmațiile din limbaj natural şi cele din limbajul formal ales*** (**LP**). În limbajul natural afirmația „elementară” P (adică, de fapt, *Există petrol în Patagonia*) ***este considerată ad literam,*** **împreună** ***cu semantica sa*** (adică se acceptă cumva intuitiv şi implicit, ca fiind un adevăr: ***în*** *Patagonia există petrol*), pe când în **LP**, ***P este un simplu nume sintactic de variabilă, care semantic poate avea orice valoare de adevăr***.

Metoda 3 (a rezoluției). Pornim din nou cu forma mai simplă F ≡ (P $\vee$ $\rceil$G) $\wedge$ (E $\vee$ $\rceil$G) şi dorim să vedem dacă F este nesatisfiabilă, utilizând rezoluția. Prin urmare, F = {{P, $\rceil$G}, {E, $\rceil$G}} şi practic se găseşte imediat că Res*(F) = F, $\square \notin$ Res*(F), adică F este satisfiabilă. *Lăsăm pe seama cititorului* să arate, folosind tot această metodă, că F este nevalidă. ■

Nu am avea acelaşi succes dacă am dori să „exprimăm convenabil" în **LP** o afirmaţie ca: „*O relaţie binară f Í A ´ B este funcţie atunci şi numai atunci când pentru fiecare element a Î A, dacă există b_1, b_2 Î B cu f(a) = b_1 şi f(a) = b_2, rezultă b_1 = b_2*". Am putea încerca ceva de genul (începem tot cu „delimitarea" subformulelor atomice):

- D: $f \subseteq A \times B$ este o relaţie binară.
- E: A' este o submulţime a lui A formată din toţi $a \in A$ pentru care există măcar două elemente distincte $b_1, b_2 \in B$, care satisfac $f(a) = b_1$ şi $f(a) = b_2$, sau, pentru care nu există nici un element $b \in B$ castfel încât $f(a) = b_1$.
- H: A' $\subseteq$ A este vidă, A' fiind cea de mai sus.
- G: relaţia $f \subseteq A \times B$ este funcţie, f fiind cea de mai sus, în altă notaţie.

Atunci formula F $\in$ **LP** va fi $F = (D \wedge E \wedge H) \rightarrow G$ şi va exprima cumva afirmaţia iniţială. Sunt însă numeroase **inconveniente**, printre care: *variabilele propoziţionale nu prea sunt elementare (indivizibile, cu valoare clară de adevăr)*; *se amestecă prea mult sintaxa cu semantica*; *înafară de valorile 0 şi 1 parcă ar mai trebui ceva*, etc. *Soluţia ar fi să putem* ***exprima direct în limbajul formal*** **cuantificatorii** (pentru orice, există), *aşa cum de altfel* ***am şi făcut***

până acum, însă doar în metalimbaj. S-ar ajunge atunci la ceva de forma (presupunând că relația inițială este deja exprimată ca fiind o mulțime de perechi de tipul <x, f(x)>):

$$F' = (\forall x)(\exists y)(f(x) = y) \wedge (\forall x)(\forall y)(\forall z)(f(x) = y \wedge f(x) = z \rightarrow y = z),$$

lucru care ar permite păstrarea tuturor principiilor logicii aristotelice, iar „puterea de exprimare directă” a limbajului este clar „mai mare” ($F' \notin$ **LP**, dar poate aparține unei supramulțimi alese corespunzător). Atuci va exista într-adevăr posibilitatea de a lucra cu afirmații elementare cu semnificație clară, de a avea o semantică explicită pentru cuantificatori, de a exprima direct relații (cum ar fi relația de egalitate), sau, dacă se doreşte neapărat, relația de apartenență a unor obiecte la anumite mulțimi, etc. **Capitolul 3** este destinat studiului unei extensii posibile a lui **LP** şi anume **LP1**, *logica (calculul) cu predicate de ordinul I*.

Indexul termenilor importanți:

§10. Exerciții

1. Rezolvați **Exercițiul 2.1**.
2. Rezolvați **Exercițiul 2.2**.
3. Rezolvați **Exercițiul 2.3**.
4. Rezolvați **Exercițiul 2.4**.
5. Completați demonstrația **Teoremei 2.1**.
6. Rezolvați **Exercițiul 2.5**.
7. Arătați că în **LP** există formule satisfiabile (dar nevalide), formule valide, contradicții.
8. Arătați că sunt adevărate afirmațiile:

   (a) $\equiv$ şi $\equiv_s$ sunt relații de echivalență pe **LP**.

   (b) $\equiv$ este compatibilă la dreapta şi la stânga cu $\wedge$, $\vee$ şi compatibilă cu $\rceil$.

   (c) $\mathcal{LP}$ = < **LP**/≡, Ù, Ú, ù > formează o algebră booleană.

   (d) Între $\mathcal{LP}$ şi $\mathcal{B}$ = < **B**, •, +, ¯ > există măcar un homomorfism de algebre booleene.
9. Completați demonstrația **Teoremei 2.4**.
10. Să se aplice **Algoritmul Horn** formulei:

    $F = (\rceil B \vee \rceil D) \wedge \rceil E \wedge \rceil C \wedge B \wedge (\rceil B \vee D \wedge B)$.
11. Să se exprime ca formulă în **LP** şi să se studieze satisfiabilitatea afirmației: *Vom câştiga alegerile în condițiile în care Popescu*

va fi liderul Partidului. Dacă Popescu nu este ales liderul Partidului, atunci fie Ionescu fie Rădulescu va părăsi partidul şi vom pierde alegerile.

12. Se dă formula:

    $F = ((A_1 \wedge A_3) \rightarrow (A_2 \rightarrow A_4)) \rightarrow ((A_1 \rightarrow A_2) \wedge (A_2 \rightarrow A_4))$.

    Să se elimine conectorii $\rightarrow$ care apar în F şi apoi să se elimine „cât mai multe” paranteze (fără a schimba semantica formulei), ținîndu-se cont de prioritățile atribuite operatorilor $\rceil$, $\vee$, $\wedge$, precum şi de alte proprietăți ale acestor operatori.

13. Să se găsească o respingere (dacă există) pornind cu clauzele:

    $F = \{\{A, \rceil B, C\}, \{B, C\}, \{\rceil A, C\}, \{B, \rceil C\}, \{\rceil C\}\}$.

14. Arătați că formula:

    $F = (\rceil B \wedge \rceil C \wedge D) \vee (\rceil B \wedge \rceil D) \vee (C \wedge D) \vee B$

    este tautologie, folosind metoda rezoluției.

15. Arătați că formula $G = (A \wedge B \wedge C)$ este consecință semantică din mulțimea de formule $\mathcal{G} = \{\rceil A \vee B, \rceil B \vee C, A \vee \rceil C, A \vee B \vee C\}$ folosind metoda rezoluției.

16. Mulțimea infinită de formule $M = \{A_1 \vee A_2, \rceil A_2 \vee \rceil A_3, A_3 \vee A_4, \rceil A_4 \vee \rceil A_5, \ldots\}$ este satisfiabilă?

17. Demonstrați adevărul sau falsitatea următoarelor afirmații (există şi alte variante, pe care le puteți deduce singuri):

    (a) Dacă $F \rightarrow G$ este validă şi F este validă, atunci G este validă.

    (b) Dacă $F \rightarrow G$ este satisfiabilă şi F este satisfiabilă, atunci G este satisfiabilă.

(c) Dacă $F \rightarrow G$ este validă şi F este satisfiabilă, atunci G este satisfiabilă.

18. Folosind cele menționate despre necesitatea de a utiliza un limbaj mai complex pentru reprezentarea realității prin formule, găsiți o formulă F care să conțină un simbol funcțional f de aritate 1 şi care să exprime faptul că f este funcție injectivă şi surjectivă.
19. Fie formula $F \in$ **LP**, $F = (\rceil((A \vee B) \wedge C))$, A, B, C $\in \mathcal{A}$. Să se găsească arborele care descrie formula şi, simultan, o **FNC** şi o **FND** pentru F (conform algoritmului recursiv sugerat de demonstrația **Teoremei 2.6**).