

## Capitolul 2

### Logica propozițională

#### (Calculul propozițional)

Manualele de *Logică* și *Algebră* ([BIE], [DID]) pot fi privite ca o *introducere (fîravă)* în *logica formală*. Și în alte manuale de matematică (și nu numai), sunt prezente frecvent noțiuni ca *afirmație*, *axiomă*, *teoremă*, *raționament*, *demonstrație*, etc. Aceste noțiuni sunt însă **descrie** 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 și 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 *premisele* cât și *concluziile* sunt afirmații, ideea fiind aceea că regulile sunt astfel construite încât dacă premisele sunt adevărate atunci și 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 științele exacte *noțiuni noi* (utilizând *definițiile*) și *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 și alte tipuri generale de definiții) prin precizarea unei sfere vechi (care caracterizează complet o noțiune anterior definită, numită **gen proxim**) și 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 *patrulat* *convex* care are *două laturi paralele și egale*; un *romb* este un *paralelogram* cu *toate laturile egale*; un *pătrat* este un *romb* având *un unghi de 90°*, ș. 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 și la reprezentări grafice cum ar fi diagramele Venn-Euler – [ȚIP]). Noțiunile primare nu mai sunt definite prin schema „gen proxim și 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 și 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 și 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**. Așa cum în momentul definirii unei noțiuni (noi) trebuie să fim atenți ca sfera acesteia să fie **nevidă** și (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 și datorită „amalgamării” considerațiilor de natură sintactică și semantică, eșafodajul anterior este destul de șubred 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 și 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 și interpretări greșite, „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 șapte 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 își afirmă propria falsitate, sau, un argument care conduce la o concluzie contradictorie deși începe cu niște premise acceptabile și se folosește o deducție validă. Oricum, se acceptă faptul că **un paradox nu înseamnă același lucru cu o contradicție**. Astfel, afirmația „Această cămașă este albastră și această cămașă nu este albastră” este o contradicție, dar *un paradox va apare atunci când o persoană face o anumită presupunere și 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 și reciproc. Mai sus este vorba despre o clasă mai simplă de paradoxuri (numite și *semantice*). Practic, ele ar putea fi „rezolvate” dacă sunt eliminate complet din logica clasică, deoarece pot fi considerate ca afirmații cărora nu li poate atașa o unică valoare de adevăr (contradicțiilor nu li se poate practic atașa 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 însăși ca element?”.

Imediat se obține că dacă răspunsul este „DA”, atunci R nu se conține pe ea însăși și dacă răspunsul este „NU”, atunci R se conține. Contradicția provine aici din acceptarea *axiomei înțelegerii*: „Dacă  $P$  este o proprietate (relație, predicat), atunci  $M = \{x \mid P(x)\}$  este o mulțime” (paradoxul precedent se obține luând  $P(x)$ : „ $x$  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 și broasca țestoasă**, atribuit lui **Zenon**: „Ahile și o broască țestoasă se iau la întrecere într-o alergare de viteză, Ahile aflându-se inițial într-un punct A și broasca în fața sa, la o distanță  $a$ , într-un punct B, dar începând să se deplaseze amândoi în același moment și în aceeași 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 și B; pentru a ajunge în B, Ahile trebuie să ajungă întâi în C; fie acum D mijlocul distanței dintre A și 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 numește și *aporie*, contradicția provenind, în cazul nostru, din utilizarea unui raționament corect într-un „mediu” necorespunzător (drumurile, în legătură cu deplasarea unor ființe, nu pot fi considerate drept reprezentări ale axei reale). Deși nu sunt ele însele „absurdități”, **silogisme** reprezintă o altă sursă generoasă de confuzii. *Inferențele*, adică pașii elementari (considerați a fi indivizibili) ai unui raționament, reprezintă forme

logice complexe. Aceste raționamente „elementare” se împart în *deductive* și *inductive*, iar cele mai simple inferențe sunt cele *immediate*, *cu propoziții categorice* (fiind formate din două asemenea propoziții: *o premiză*, și *o concluzie*). Silogismul este tipul fundamental de inferență deductivă mediată alcătuită din exact trei propoziții categorice: două premize, dintre care una majoră și alta minoră, precum și 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 și în cazul paradoxurilor, să acceptăm niște aberații drept propoziții adevărate. De exemplu:

Albă este adjectiv

Zăpada este albă

Zăpada este adjectiv

Greșeala 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 și numai trei termeni lingvistici distincți. Din păcate însă, în limba

română (dar nu numai) un același cuvânt (sau grup de cuvinte) poate „materializa” mai mult decât o singură noțiune. Astfel, deși 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 *însușire* (care este caracteristică și *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 și câteva idei legate de **inferențele (raționamentele) deductive cu propoziții compuse**. Deși 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*) și apoi (**Pas constructiv**) formăm propoziții noi (*fraze, propoziții compuse*), din propoziții vechi cu ajutorul unor operatori (*conectori*), cum ar fi *sau*, *și*, *negația*, *implicația* (*dacă ... atunci ...*), *echivalența* (*dacă ... atunci ... și reciproc*). Notând cu A și B două propoziții oarecare (elementare sau compuse), putem forma **propozițiile compuse** (de acum înainte,  $\cong$  va nota „egal prin definiție/notație/convenție”):  $C \cong A \text{ sau } B$  (simbolic:  $A \vee B$ );  $D \cong A \text{ și } B$  (simbolic:  $A \wedge B$ );  $E \cong \text{non } A$  (simbolic:  $\neg A$ );  $F \cong \text{dacă } A \text{ atunci } B$  (simbolic:  $A \rightarrow B$ ; A se numește uneori *ipoteză* sau *antecedent* iar B – *concluzie* sau *consecvent*);  $G \cong \text{dacă } A \text{ atunci } B$

și reciproc (sau, A dacă și numai dacă B, sau A atunci și numai atunci când B; simbolic:  $A \leftrightarrow B$ ). Cum A și B pot lua fiecare doar valorile **a** - **adevărat** sau **f** – **fals** (desigur...nu simultan), la fel se va întâmpla și cu propozițiile compuse. Astfel, C va fi **a** atunci și numai atunci când măcar una dintre A și B este **a**, D va fi **a** atunci și numai atunci când atât A cât și B sunt **a**, E va fi **a** atunci și numai atunci când A va fi **f**, F va fi **f** atunci și numai atunci când A este **a** și B este **f**. În sfârșit, G va fi **a** atunci și numai atunci când A și B sunt simultan **a** sau simultan **f**. Ca o consecință, **o implicație va fi adevărată dacă ipoteza este falsă, indiferent de valoarea de adevăr a concluziei**. Acum ne putem referi în mod explicit și la inferențe cu propoziții compuse (se presupune că silogismele utilizează doar propoziții simple), cele conținând **implicația** fiind des utilizate. Cele mai simple inferențe de acest tip sunt cele care conțin *două premize și o concluzie*, dintre ele distingându-se cele *ipotetico-categorice* (*prima premiză este o implicație iar cea de-a doua constă fie din antecedentul sau negația antecedentului, fie din consecventul sau negația consecventului implicației respective*, conform [BIE]). *Schemele valide* care se folosesc în raționamente sunt astfel de forma:

$$A \rightarrow B$$

$$\frac{A}{B}$$

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

sau

$$A \rightarrow B$$

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

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



Validitatea (reamintim: dacă ipotezele sunt adevărate, atunci și concluzia trebuie să fie adevărată) schemelor *modus ponens* (*modul afirmativ*) și *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) și  $A \rightarrow B$  (adevărată) deducem (că) B (adevărată)**. Pe scurt, vom nota acest lucru prin  $A \Rightarrow B$ . Următorul exemplu este edificator pentru greșelile care se pot face fie din necunoașterea 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 și 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*  $x_1, x_2 \in \mathbf{R}$ , *avem: dacă*  $f(x_1) = f(x_2)$  *atunci*  $x_1 = x_2$ . Anticipând notatiile din **Capitolul 3** și presupunând cunoscută (cel puțin la nivel informal) semnificația *cuanficatorilor*, putem scrie acest lucru sub forma condensată  $(\forall x_1, x_2 \in \mathbf{R})(f(x_1) = f(x_2) \rightarrow x_1 = x_2)$ . Există următoarele posibilități:

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

b)  $x_1, x_2 > 0$ . Atunci  $f(x_1) = x_1^2 + 1$  și  $f(x_2) = x_2^2 + 1$ . Prin urmare,  $f(x_1) = f(x_2)$  înseamnă  $x_1^2 + 1 = x_2^2 + 1$ , ceea ce se întâmplă atunci și 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 și 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 și consecventul  $x_1 = x_2$  este adevărat (de fapt, și reciproc).

c)  $x_1 \leq 0, x_2 > 0$ . În acest caz  $f(x_1) = x_1$  și  $f(x_2) = x_2^2 + 1$ . Atunci  $f(x_1) = f(x_2)$  înseamnă  $x_1 = x_2^2 + 1$  și implicația pe care trebuie să o arătăm devine  $x_1 = x_2^2 + 1 \rightarrow x_1 = x_2$ , și aceasta **pentru fiecare**  $x_1 \leq 0$  și  $x_2 > 0$ . Nu mai putem proceda la fel ca în situațiile anterioare, deoarece din  $x_1 = x_2^2 + 1$  nu se poate *deduce*  $x_1 = x_2$ . Totuși, **implicația în cauză din limbajul de bază este adevărată, deoarece antecedentul ei este fals**. Într-adevăr, oricare ar fi  $x_1 \leq 0$  și  $x_2 > 0$ , 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 și o notă optimistă, prezentăm una dintre cele mai „mari” demonstrații cunoscute în

literatura „științifică” a afirmației *Oamenii de știință nu vor câștiga niciodată la fel de mulți bani ca directorii executivi ai unor companii de succes*. În acest scop vom porni de la postulatele (axiomele) *Cunoașterea înseamnă putere și Timpul înseamnă bani*, pe care le vom folosi sub forma prescurtată: **cunoaștere = putere** și 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 și proprietățile relației de egalitate, printr-o inferență simplă deducem:


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


Aplicând acum o proprietate a proporțiilor, găsim:


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

Cititorul poate trage singur concluzia care se impune pentru situația în care **cunoaștere** 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 și ea ipoteze și concluzii. Scrieți simbolic forma generală a unei teoreme (**directe**), utilizând propoziții elementare (**variabile propoziționale**) și **conectori logici**. Scrieți apoi **teorema reciprocă**, **contrara teoremei directe** și **contrara reciprocei**. Există vreo legătură între acestea, în ceea ce privește valoarea lor de adevăr? Dați un exemplu de **teoremă de caracterizare** ( $A$  dacă și numai dacă  $B$ ). Puteți specifica altfel rezultatul exprimat de teoremă, astfel încât să fie – separat – puse în evidență **condiția necesară** și **condiția suficientă**?

 **Exercițiul 2.2.** Să considerăm definiția limitei unui șir 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 și **cuantificatorii**) această definiție și 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 și apoi rescrieți rezultatul în limbaj natural. Dacă ați fi negat direct propoziția inițială, ați fi obținut același 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, așa cum am sugerat deja, va fi numele unei mulțimi de formule (*propoziționale*), notată  $\mathbf{LP}_{\mathcal{L}}$  sau, prescurtat,  $\mathbf{LP}$  și definită structural în cele ce urmează.

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

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

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

- (i) Dacă  $F \in \mathbf{LP}$  atunci  $(\neg F) \in \mathbf{LP}$ .
- (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.* ■

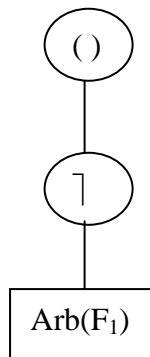
Putem privi o formulă  $F$  ca fiind *reprezentată* de un arbore binar (*arborele atașat lui  $F$ , notat  $\text{Arb}(F)$* ), în modul următor (procedăm structural, conform definiției lui  $\mathbf{LP}$ ):

**Definiția 2.2.**

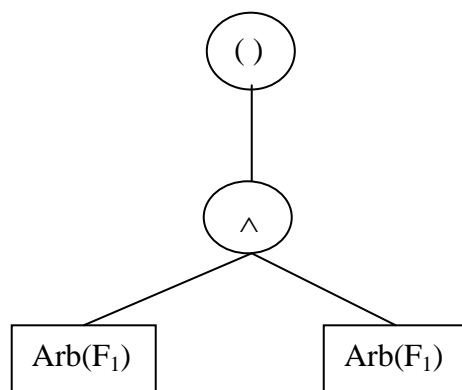
**Baza.**  $F = A \in \mathcal{A}$ . Atunci *arborele atașat* lui  $F$  (sau, *arborele care reprezintă*  $F$ ), este:

**Pas constructiv.**

(i) Fie  $F = (\neg F_1)$  și să presupunem că se cunoaște arborele atașat lui  $F_1$ ,  $\text{Arb}(F_1)$ . Atunci, arborele atașat lui  $F$  va fi (ceva similar se obține pentru (iv), adică pentru cazul  $F = (F_1)$ ):



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



■

Deși 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 același „număr de ordine” cu paranteza deschisă „(” *corespunzătoare*), atunci se observă că fiecărei formule îi corespunde un arbore atașat unic și fiecărui *arbore ordonat*  $G$  (cu nodurile etichetate cu elemente din  $\mathcal{L}$ ) îi corespunde o unică formulă din  $\mathbf{LP}$  (pentru care  $G$  este arborele atașat). Definim structural și *mulțimea subformulelor* oricărei formule date  $F$  (notată  $\text{subf}(F)$ ). *Admitem implicit faptul că  $F' \in \text{subf}(F)$  dacă și numai dacă  $F'$  este subcuvânt al lui  $F$  și  $F' \in \mathbf{LP}$*  (cu alte cuvinte,  $F_1$  și  $F_2$ , în cele ce urmează, sunt tot formule).

**Definiția 2.3.**

**Baza.**  $F = A \in \mathcal{A}$ . Atunci  $\text{subf}(F) = \{A\}$ .


**Pas constructiv.**

- (i)  $F = (\bigwedge F_1)$ . Atunci  $\text{subf}(F) = \text{subf}(F_1) \cup \{(\bigwedge F_1)\}$ .
- (ii)  $F = (F_1 \wedge F_2)$ . Atunci  $\text{subf}(F) = \text{subf}(F_1) \cup \text{subf}(F_2) \cup \{(F_1 \wedge F_2)\}$ .
- (iii) Analog cu (ii) pentru cazul  $F = (F_1 \vee F_2)$  (înlocuind peste tot, simultan,  $\wedge$  cu  $\vee$ ).
- (iv)  $F = (F_1)$ . Atunci  $\text{subf}(F) = \text{subf}(F_1) \cup \{(F_1)\}$  ■

**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  $w \in \mathcal{L}^*$  (adică orice secvență finită de caractere din  $\mathcal{L}$ ) să se decidă dacă  $w \in \mathbf{LP}$* . Conform [JUC] de exemplu, notația  $\mathcal{L}^*$  (algebric,  $\mathcal{L}^*$  este monoidul liber generat de  $\mathcal{L}$ ) 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 juxtaponere 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  $w \in \mathcal{L}^*$ , cu răspunsul (ieșirea) „DA” dacă  $w \in \mathbf{LP}$  și „NU” dacă  $w \notin \mathbf{LP}$ . O **problemă de decizie** are doar alternativa de răspuns „DA/NU” și 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



$\text{subf}(A_1 \vee A_2)$  pentru că  $A_1 \vee A_2$  nu este formulă). Dar, la fel ca și în cazul cunoscut al expresiilor aritmetice care conțin variabile, constante și operatorii „-” (având și sensul de *opus*), „+”, „•” și „/”, 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 și pe unele proprietăți cum ar fi comutativitatea, asociativitatea sau distributivitatea. Prioritățile standard sunt: 0 – pentru  $\neg$ , 1 – pentru  $\wedge$ , 2 – pentru  $\vee$ . Tot o convenție este și aceea de a folosi și 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 ( $A, B, C, \dots$ , cu sau fără indici). Invers, putem adăuga în orice formulă „bine formată” cupluri de paranteze corespondente (la fel cum le-am și eliminat), pentru a îmbunătăți receptarea corectă a sintaxei și fără a schimba semnificația formulei în cauză. Acest lucru este permis de altfel prin (iv), **Definiția 2.1.** ■

 **Exercițiul 2.4.** Fie formula  $F = ((\neg A) \vee (B \wedge C))$ . Construiți arborele atașat (verificând în acest mod și faptul că într-adevăr  $F \in \mathbf{LP}$ ). Eliminați parantezele și 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 și alte câteva prescurtări sintactice, justificate de altfel și de anumite considerente semantice care vor fi prezentate ulterior:

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

Simbolurile  $\rightarrow$  și  $\leftrightarrow$  (numite după cum știm **implicație**, respectiv **echivalență**) pot fi considerate ca și cum ar fi fost introduse de la bun început în mulțimea de conectori  $\mathcal{C}$  (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 și definiția semanticii **LP**, care urmează). Vom numi **literal** o variabilă propozițională sau negația sa.  $A \in \mathcal{A}$  se va numi și **literal pozitiv** iar orice element de forma

$$\neg A,$$

$A \in \mathcal{A}$  va fi un **literal negativ** (vom nota  $\overline{\mathcal{A}} = \{\neg A_1, \neg A_2, \dots\}$ ). Dacă  $L$  este un literal, atunci **complementarul** său,  $\overline{L}$ , va nota literalul  $\neg A$ , dacă  $L = A \in \mathcal{A}$  și respectiv literalul  $A$  dacă  $L = \neg A$ . Sperăm ca această notație sintactică să nu fie confundată cu operația semantică  $\neg$ ,

prezentă în definiția algebrelor booleene, rezultatele privind sintaxa fiind, în general, separate de cele privind semantica. Se numește **clauză** orice disjuncție (finită) de literali. Se numește **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, și literal 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 și pe **f** cu 0, astfel încât să putem lucra cu algebra booleană  $\mathcal{B} = \langle \mathbf{B}, \bullet, +, \bar{\phantom{x}} \rangle$ , noțiunea principală este cea de *asignare* (*interpretare, structură*).

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

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

- (i)  $\mathcal{S}'(A) = \mathcal{S}(A)$ , pentru fiecare  $A \in \mathcal{A}$ .
- (ii)  $\mathcal{S}'(\neg F) = \overline{\mathcal{S}'(F)}$ , 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)  $\mathcal{S}'((F_1 \vee F_2)) = \mathcal{S}'(F_1) + \mathcal{S}'(F_2)$ , pentru fiecare  $F_1, F_2 \in \mathbf{LP}$ .

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

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

**Pas constructiv.**

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

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

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

Este evident că  $\mathcal{S}'$  este o extensie a lui  $\mathcal{S}$ , 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  $\mathcal{S}'$  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  $F \in \mathbf{LP}$ , este adevărat  $P(F)$* , unde  $P(F)$  este: *Oricare ar fi asignarea  $\mathcal{S}$ , valoarea  $\mathcal{S}'(F)$  există (ca element al lui  $\mathbf{B}$ ) și este unică, adică  $\mathcal{S}'$  este funcție, și oricare altă funcție  $\mathcal{S}''$  care satisface (i) – (iv), satisface  $\mathcal{S}'(F) = \mathcal{S}''(F)$ .*

**Baza.** Fie  $F = A \in \mathcal{A}$  și orice asignare  $\mathcal{S}$ . Cum  $\mathcal{S}$  este funcție (totală) prin definiție și avem  $\mathcal{S}'(A) = \mathcal{S}(A)$ , tot prin definiție ( $\mathcal{S}'$  este extensia

lui  $S$ ), este imediat faptul că  $S'(A)$  există și este unică în sensul precizat (orice alt – posibil –  $S''$  trebuie să fie tot o extensie a lui  $S$ ).

**Pas inductiv.** Vom arăta doar cazul  $F = (\bigwedge F_1)$ , celelalte două ( $F = (F_1 \wedge F_2)$  și  $F = (F_1 \vee F_2)$ ) fiind similare. Presupunem prin urmare  $P(F_1)$  ca fiind adevărat și demonstrăm că  $P(F)$  este adevărat. Fie orice asignare  $S$ . Faptul că  $S'(F)$  există (ca element al lui  $\mathbf{B}$ ) și este unică (în sensul precizat), rezultă din nou imediat, din ipoteza inductivă ( $S'(F_1)$  există și este unică), din definiția negației în  $\mathbf{B}$  (știm că  $S'(F) = \overline{S'(F_1)}$ ) și 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 și structură (interpretare)*. Se observă că dată orice formulă  $F \in \mathbf{LP}$  și orice structură  $S$ , este suficient să cunoaștem valorile lui  $S$  în *variabilele propoziționale care apar în  $F$*  (pentru fiecare  $F \in \mathbf{LP}$ , vom nota cu  $\text{prop}(F)$  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  $\text{prop}(F) \subseteq \mathcal{A}$  și cu valori în  $\mathbf{B}$ . Aceasta, în cazul în care  $F$  este cunoscută, poate fi identificată cu o funcție totală pe  $\mathcal{A}$ . Putem conchide chiar că în  $\mathbf{LP}$  *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  $\mathcal{S}(F) \equiv \mathcal{S}(\neg(F))$  pentru fiecare  $F \in \mathbf{LP}$ .

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

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  $F \in \mathbf{LP}$  se numește **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  $F \in \mathbf{LP}$  este validă dacă și numai dacă  $(\neg F)$  este contradicție.

**Demonstrație.**  $F \in \mathbf{LP}$  este validă dacă și numai dacă pentru fiecare structură  $\mathcal{S}$  avem  $\mathcal{S}(F) = 1$ , adică (conform ii), **Teorema 2.1**) dacă și numai dacă  $\mathcal{S}(\neg F) = 1 = \bar{0}$  (definiția negației), ceea ce înseamnă că  $(\neg F)$  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	( $\neg$ F)

Tabelul 2.1

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

**Definiția 2.6.** Două formule  $F_1, F_2 \in \mathbf{LP}$  se numesc **tare echivalente** dacă *pentru fiecare* structură  $S$  ele au aceeași valoare de adevăr, adică  $\mathcal{S}(F_1) = \mathcal{S}(F_2)$  (simbolic, vom scrie  $F_1 \equiv F_2$ ).  $F_1$  și  $F_2$  se numesc **slab echivalente** dacă  $F_1$  satisfiabilă *implică*  $F_2$  satisfiabilă și reciproc (vom scrie  $F_1 \equiv_s F_2$ , ceea ce înseamnă că *dacă există  $\mathcal{S}_1$  astfel încât  $\mathcal{S}_1(F_1) = 1$ , atunci există  $\mathcal{S}_2$  astfel încât  $\mathcal{S}_2(F_2) = 1$  și reciproc*). O formulă  $F \in \mathbf{LP}$  este **consecință semantică** dintr-o mulțime de formule  $\mathcal{G} \subseteq \mathbf{LP}$ , 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  $\mathcal{G}$ ), *dacă  $S$  satisface  $\mathcal{G}$*  (adică

avem  $S(G) = 1$  pentru fiecare  $G \in \mathcal{G}$  atunci  $S$  satisface  $F$  (simbolic, vom scrie  $\mathcal{G} \sqsubseteq F$ ). ■

**Observație.** Relațiile  $\equiv$  și  $\equiv_s$  sunt *relații de echivalență* (binare) pe  $\mathbf{LP}$ , în sens matematic (adică sunt *reflexive*, *simetrice* și *tranzitive*) și prin urmare  $\mathbf{LP}$  poate fi *partiționată în clase de echivalență* corespunzătoare, obținându-se *mulțimile cât*  $\mathbf{LP}/\equiv$ , respectiv  $\mathbf{LP}/\equiv_s$ . Mai mult, privind  $\wedge, \vee, \top$  ca niște operatori (de fapt, ar trebui să-i considerăm *împreună* cu parantezele pe care le introduc, vezi și **Exercițiul 2.4**), atunci relația  $\equiv$  este *compatibilă (la stânga și la dreapta)* cu aceștia ( $[\top IP]$ ), astfel încât considerând 4-uplul  $\mathcal{LP} = \langle \mathbf{LP}/\equiv, \wedge, \vee, \top \rangle$ , se poate arăta că acesta formează o algebră booleană (homomorfă cu  $\mathcal{B} = \langle \mathbf{B}, \bullet, +, - \rangle$ , după cum sugerează **Teorema de extensie**). ■

**Teorema 2.3.** Fie  $G \in \mathbf{LP}$  și  $\mathcal{G} = \{ G_1, G_2, \dots, G_n \} \subseteq \mathbf{LP}$ . Următoarele afirmații sunt echivalente:

(i)  $G$  este consecință semantică din  $\mathcal{G}$ .

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

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

**Demonstrație.**



(i) *implică* (ii). Presupunem prin reducere la absurd că

$F = (\bigwedge_{i=1}^n G_i) \rightarrow G$  nu este tautologie, deși  $G$  este consecință semantică

din  $\mathcal{G}$ . Rezultă că există o structură  $\mathcal{S}$  pentru care  $F$  este falsă, adică  $\mathcal{S}(\bigwedge_{i=1}^n G_i) = 1$  și  $\mathcal{S}(G) = 0$ . Prin urmare, pentru fiecare  $i \in [n]$  avem  $\mathcal{S}(G_i) = 1$  și ca urmare  $\mathcal{S}(G) = 0$ . În concluzie, există o structură  $\mathcal{S}$  astfel încât  $\mathcal{S}(\mathcal{G}) = 1$  și  $\mathcal{S}(G) = 0$ . 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 deși  $(\bigwedge_{i=1}^n G_i) \rightarrow G$  este tautologie,  $(\bigwedge_{i=1}^n G_i) \wedge \neg G$  nu este contradicție. Această înseamnă că  $F_1 = \neg(\bigwedge_{i=1}^n G_i) \vee G$  este tautologie, dar  $F_2 = (\bigwedge_{i=1}^n G_i) \wedge \neg G$  este satisfiabilă. Prin urmare, există o structură  $\mathcal{S}$  astfel încât  $\mathcal{S}(F_2) = 1$  (și, desigur,  $\mathcal{S}(F_1) = 1$ ). Din  $\mathcal{S}(F_2) = 1$  rezultă  $\mathcal{S}((\bigwedge_{i=1}^n G_i)) \cdot \mathcal{S}(\neg G) = 1$ , adică  $\mathcal{S}((\bigwedge_{i=1}^n G_i)) = 1$  și  $\mathcal{S}(\neg G) = 1$ .

În consecință,  $\mathcal{S}(\neg(\bigwedge_{i=1}^n G_i)) = 0$  și  $\mathcal{S}(G) = 0$ . Pentru că  $\mathcal{S}(F_1) = \mathcal{S}(\neg(\bigwedge_{i=1}^n G_i)) + \mathcal{S}(G)$ , avem  $\mathcal{S}(F_1) = 0$ , ceea ce este absurd deoarece  $F_1$  este tautologie.

(iii) *implică* (i). Presupunem prin reducere la absurd că

$F = (\bigwedge_{i=1}^n G_i) \wedge \neg G$  este contradicție, dar  $G$  nu este consecință semantică

din  $\mathcal{G}$ . Atunci există o structură  $\mathcal{S}$  care satisface toate formulele din  $\mathcal{G}$

dar nu satisface  $G$ . Prin urmare, avem  $\mathcal{S}(\bigwedge_{i=1}^n G_i) = 1$  și  $\mathcal{S}(G) = 0$ , adică

$\mathcal{S}(\bigwedge_{i=1}^n G_i) = 1$  și  $\mathcal{S}(\neg G) = 1$ . Cum  $\mathcal{S}(\bigwedge_{i=1}^n G_i \wedge \neg G) = \mathcal{S}(\bigwedge_{i=1}^n G_i) \cdot \mathcal{S}(\neg G)$ ,

rezultă că există  $\mathcal{S}$  astfel încât  $\mathcal{S}(\bigwedge_{i=1}^n G_i \wedge \neg G) = 1$ , deci  $F$  nu este contradicție (absurd). ■

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

**Teorema 2.4.** Sunt adevărate următoarele echivalențe (tari, pentru oricare  $F, G, H \in \mathbf{LP}$ ):

$$(a) F \wedge F \equiv F$$

$$(a') F \vee F \equiv F \text{ (idempotență)}$$

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

$$(b') F \vee G \equiv G \vee F \text{ (comutativitate)}$$

$$(c) (F \wedge G) \wedge H \equiv$$

$$(c') (F \vee G) \vee H \equiv F \vee (G \vee H)$$

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

$$\text{(asociativitate)}$$

$$(d) F \wedge (G \vee H) \equiv$$

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

$$\equiv (F \wedge G) \vee (F \wedge H)$$

$$\text{(distributivitate)}$$

$$(e) F \wedge (F \vee G) \equiv F \quad (e') F \vee (F \wedge G) \equiv F \text{ (absorbție)}$$

$$(f) \neg \neg F \equiv F \quad (\text{legea dublei negații})$$

$$(g) \neg (F \wedge G) \equiv \neg F \vee \neg G \quad (g') \neg (F \vee G) \equiv \neg F \wedge \neg G \text{ (legile lui deMorgan)}$$

$$(h) F \vee G \equiv F \quad (h') F \wedge G \equiv G \text{ (legile validității, adevărate doar dacă F este tautologie)}$$

$$(i) F \wedge G \equiv F \quad (i') F \vee G \equiv G \text{ (legile contradicției, adevărate doar dacă F este contradicție)}$$

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

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

**Teorema 2.5 (de substituție).** Fie  $H \in \mathbf{LP}$ , oarecare. Fie orice  $F, G \in \mathbf{LP}$  astfel încât  $F$  este o subformulă a lui  $H$  și  $G$  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**  $(\forall H \in \mathbf{LP}) P(H)$ , unde

$P(H)$ :  $(\forall F, G, H' \in \mathbf{LP})(((F \in \text{subf}(H)) \text{ și}$

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

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

**Baza.**  $H = A \in \mathcal{A}$ . Să arătăm că  $P(A)$  este adevărată. Fie  $F, G, H' \in \mathbf{LP}$ , astfel încât  $F \in \text{subf}(H)$ ,  $H'$  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  $F \in \text{subf}(H)$  și  $\text{subf}(H) = \{A\}$ , rezultă că  $F = A$  (care coincide cu  $H$ ). Prin urmare,  $H' = G$ . Avem acum  $F = H, G = H'$  și  $F \equiv G$ , de unde urmează imediat că  $H \equiv H'$ .

**Pas inductiv.** Trebuie tratate separat situațiile care urmează.

(i)  $H = (\bigwedge H_1)$ . Presupunem că  $P(H_1)$  este adevărată și demonstrăm că  $P(H)$  este adevărată. Fie  $F \in \text{subf}(H) = \text{subf}(H_1) \cup \{(\bigwedge H_1)\}$ . Dacă  $F = (\bigwedge H_1)$  ( $= H$ ), 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  $F \in \text{subf}(H_1) \subseteq \text{subf}(H)$  și considerăm orice  $G \in \mathbf{LP}$  astfel încât  $G \equiv F$ . Înlocuind pe  $F$  cu  $G$  în  $H$ , înseamnă în același 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) și obținem că  $H_1 \equiv H_1'$ . Revenind, știm că  $H = (\bigwedge H_1)$ ,  $H' = (\bigwedge H_1')$  și  $H_1 \equiv H_1'$ . Rezultă imediat că  $H \equiv H'$  (vezi și **Observația** care precede imediat **Teorema 2.3** și **V.2.8** din Anexă).

(ii)  $\mathbf{H} = (\mathbf{H}_1 \wedge \mathbf{H}_2)$ . Presupunem că  $P(H_1)$  și  $P(H_2)$  sunt adevărate și demonstrăm că  $P(H)$  este adevărată. Fie orice  $F \in \text{subf}((H_1 \wedge H_2)) = \text{subf}(H_1) \cup \text{subf}(H_2) \cup \{(H_1 \wedge H_2)\}$ . Dacă  $F = (H_1 \wedge H_2) (= H)$  suntem din nou într-un caz similar cu cel din **Baza**. Să considerăm că  $F \in \text{subf}(H_1)$  (apariția deja fixată), cazul  $F \in \text{subf}(H_2)$  tratându-se similar. Fie orice  $G \in \mathbf{LP}$  astfel încât  $G \equiv F$ . A înlocui pe  $F$  cu  $G$  în  $H$  înseamnă, în același timp, a înlocui pe  $F$  cu  $G$  în  $H_1$  ( $H_2$  rămânând neschimbată). Vom nota cu  $H'$  respectiv  $H_1'$ , formulele obținute după aceste înlocuiri. Aplicând ipoteza inductivă ( $P(H_1)$  este adevărată), rezultă imediat că  $H_1' \equiv H_1$ . Revenind, știm că  $H = (H_1 \wedge H_2)$ ,  $H' = (H_1' \wedge H_2)$  și  $H_1' \equiv H_1$ . Obținem imediat că  $H \equiv H'$  (putem folosi direct faptul deja amintit, că  $\equiv$  este compatibilă cu operațiile, respectiv cu conjuncția).

(iii)  $\mathbf{H} = (\mathbf{H}_1 \wedge \mathbf{H}_2)$ . Se demonstrează analog cu cazul precedent. ■

Pentru a nu exista confuzii între limbajul de bază (**LP**) și metalimbajul în care exprimăm afirmațiile *despre* elementele lui **LP**, în cele de mai sus (precum și în continuare) am notat implicația cu  $\Rightarrow$  iar conjuncția prin și. Deocamdată am păstrat notația clasică pentru cuantificatorul universal ( $\forall$ ), deoarece el nu apare explicit în **LP**.

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

și + luându-l  $\wedge$  respectiv  $\vee$ , 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* și *formele normale disjunctive simultan*.

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

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

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

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

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

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

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

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

procedăm prin inducție structurală.

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

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

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

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

În ultima formulă putem aplica – unde este cazul – legea dublei negații și apoi putem înlocui elementele de forma  $\neg L_{i,j}$  cu  $\bar{L}_{i,j}$ , 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)  $F = (G \wedge H)$ . Presupunem că afirmațiile  $P(G)$  și  $P(H)$  sunt adevărate și arătăm că  $P(F)$  este adevărată. Din faptul că  $P(G)$  este adevărată rezultă că există  $G_1$ , aflată în **FNC** și satisfăcând  $G \equiv G_1$ , astfel încât:

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

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

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

Atunci,  $G \wedge H \equiv G_1 \wedge H_1$  și 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  $G$  și 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 și apoi, în interiorul subformulelor, a disjuncției față de conjuncție.

(iii)  $F = (G \vee H)$ . Procedăm analog ca în cazul anterior. ■

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



**Exemplu.** Găsiți o formulă  $F \in \mathbf{LP}$  construită peste mulțimea de variabile propoziționale  $\{A, B, C\}$  și care să satisfacă condiția: în tabelul de adevăr standard care o descrie, *o schimbare și numai una în secvența  $\langle S(A), S(B), S(C) \rangle$  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** (și/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 construiește o conjuncție de literalii (apar toți, cu bară sau fără): dacă valoarea unei variabile (atom pozitiv) este 0 în tabel, atunci variabila se trece în conjuncția*

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  $F = (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$ . Găsiți, analog, o **FNC(P)** ■

Conform teoremei anterioare, precum și datorită comutativității și idempotenței disjuncției, comutativității și 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  $F = \{C_1, C_2, \dots, C_m\}$  (nu uităm totuși că *virgula aici provine dintr-o conjuncție*), unde, pentru fiecare  $i \in [m]$ , vom pune  $C_i = \{L_{i,1}, L_{i,2}, \dots, L_{i,n_i}\}$ . Mai mult, dacă avem  $F \in \mathbf{LP}$  reprezentată ca mulțime (de clauze) sau ca mulțime de mulțimi (de literal) și ne interesează doar studiul (ne)satisfiabilității ei, putem elimina clauzele  $C$  care conțin atât  $L$  cât și  $\bar{L}$ , deoarece  $L \vee \bar{L} \equiv \mathbf{1}$ ,  $\mathbf{1} \vee C \equiv \mathbf{1}$  și deci aceste clauze sunt tautologii (notate generic cu  $\mathbf{1}$ ). Tautologiile componente nu au nici o semnificație pentru stabilirea valorii semantice a unei formule  $F$  aflate în **FNC** ( $\mathbf{1} \wedge C \equiv C$ ).

## § 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  $F \in \mathbf{LP}$ , se termină cu răspunsul „DA”, dacă  $F$  este satisfiabilă (sau validă, sau contradicție) și „NU” în rest (știind desigur că putem decide dacă un anumit șir 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  $F \in \mathbf{LP}$  având  $\text{prop}(F) = \{A_1, A_2, \dots, A_n\} = \mathcal{A}_n$ . 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 și legătura dintre algebrele booleene  $\mathcal{LP}$  și  $\mathcal{B}$  stau la baza corectitudinii acestei construcții*) care are forma:

$\mathcal{S}(A_1)$	$\mathcal{S}(A_2)$	.....	$\mathcal{S}(A_n)$	$\mathcal{S}(F)$	$2^n = m$
0	0	.....	0	$v_1$	
0	0	.....	1	$v_2$	
.....	.....	.....	.....	.....	
1	1	.....	1	$v_m$	

Dacă toți  $v_i$  ( $i \in [m]$ ) sunt egali cu 0 atunci  $F$  este contradicție, dacă toți  $v_i$  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 și prin urmare trebuie efectuate  $2^n$  comparații ( $F$  este construită peste  $n$  formule atomice). Deși mai sus nu avem o explicație formală riguroasă a faptului că **SAT** are  *timp exponențial*, se poate arăta că problema este chiar *NP-completă* (conform [AHO]; a se urmări și comentariile care urmează imediat după demonstrație). ■

Datorită **Teoremei de extensie** și **Teoremei de substituție**, 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] și, în special, [CRO], [AHO], [COR], [BÖR]), trebuie căutați algoritmi „rapizi” (eficienți, tratabili), adică având complexitate (timp) mică. Astfel, două dintre *măsurile (teoretice, globale) de complexitate* des întrebuințate sunt *complexitatea timp* și

*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 și *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 (pași/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 fiecărei intrări* (adică, în cazul nostru, la lungimea unei formule  $F \in \mathbf{IN} \subseteq \mathbf{LP}$ , aceasta putând fi de exemplu  $n_F = \text{card}(\text{prop}(F))$ ) și ne interesează de fapt  $\sup\{t^{Alg}(F) \mid F \in \mathbf{IN} \text{ și } n_F = n \in \mathbf{N}\}$ , 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 depăși  $t^{Alg}(n)$ . Cum determinarea aceluia supremum este de multe ori destul de dificilă, ne vom mulțumi să studiem așa-numita *comportare asimptotică* (sau *ordinul de creștere*) a (al) lui  $t^{Alg}(n)$ , adică ne vor interesa doar anumite margini ale sale, cum ar fi marginea sa superioară. Formal, pentru fiecare  $f : \mathbf{N} \rightarrow \mathbf{N}$ , notăm

$O(f) = \{g \mid g : \mathbb{N} \rightarrow \mathbb{N}, \text{ există } c \in \mathbb{R}, c > 0 \text{ și există } k \in \mathbb{N}, \text{ astfel încât}$   
*pentru fiecare*  $n \geq k$  *avem*  $g(n) \leq c \cdot f(n)\}$  și vom spune **că fiecare**  
 $g \in O(f)$ , **este de ordinul lui**  $f$ , ceea ce se mai notează și cu  $g = O(f)$ .  
 Astfel, vom spune că **SAT** are *complexitatea (timp, asimptotică)*  $O(2^n)$ ,  
 sau, pe scurt, complexitate exponențială, deoarece că există (măcar) un  
 algoritm  $Alg$  care rezolvă problema (cel sugerat în demonstrația  
**Teoremei 2.7**) și 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 1), adică de probleme având complexitatea  
 (timp, dar se poate defini ceva asemănător pentru spațiu) de tipul  
 precedent. Speranța de a găsi algoritmi *mai performanți* 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* părți importante ale realității. În plus, în  
 condițiile utilizării calculatorului, găsirea unor algoritmi de natură  
 sintactică pentru rezolvarea **SAT** (în locul celor „semantici”, cum este  
 și cel bazat pe folosirea tabelor de adevăr) este o prioritate (chiar dacă  
 aceștia nu sunt mai buni din punctul de vedere al teoriei generale a  
 complexității).

## § 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 și 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)  $C = \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_k$ ,  $k \geq 1$ ,  $k \in \mathbf{N}$  și
- (ii)  $C = \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_k \vee B$ ,  $k \in \mathbf{N}$ .

**Observație.** Înafară de reprezentarea ca mulțimi, clauzele Horn pot fi reprezentate sub și sub așa-numita **formă implicațională**. Vom distinge cazurile (reamintim că **0** și **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  $\mathbf{1} \rightarrow A \cong \neg \mathbf{1} \vee A \equiv \mathbf{0} \vee A \equiv A$ .
- $C = \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_k$  (nici un literal pozitiv, măcar un literal negativ). Vom scrie  $C \cong A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow \mathbf{0}$  (folosim din nou definiția implicației și faptul că  $\mathbf{0} \vee A \equiv A$ ).
- $C = \neg A_1 \vee \neg A_2 \vee \dots \vee \neg A_k \vee B$  (exact un literal pozitiv, măcar un literal negativ). Atunci  $C \cong A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow B$ , direct din definiția implicației.

- $C \cong \square$  (nici un literal negativ, nici un literal pozitiv). Din motive tehnice vom folosi și această **clauză vidă** (în reprezentarea clauzelor cu mulțimi vom folosi pentru  $\square$  chiar  $\emptyset$ ). Prin convenție,  $\square$  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 și scrise sub formă implicațională .

**Ieșire:** „DA”, în cazul în care formula  $F$  este satisfiabilă (furnizându-se și o asignare  $S$  care este model pentru  $F$ ) și „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  $A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow B$ , cu  $A_1, A_2, A_3, \dots, A_k$  *marcați* și  $B$  *nemarcată* sau de forma  $A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow 0$ , cu  $A_1, A_2, A_3, \dots, A_k$  *marcați*) și ( $i = 0$ ))

**execută**

**Pasul 3.** Alege un asemenea  $C$  ca mai sus.

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

**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 și numai dacă  $A$  apare în  $F$  și 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 atașat 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  $A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow B$ , cu  $A_1, A_2, A_3, \dots, A_k$  marcați și  $B$  nemarcat), se procedează conform algoritmului, adică se marchează toate aparițiile lui  $B$  în  $F$  și se trece la pasul următor. Mai departe, la *fiecare* execuție a corpului buclei (**Pașii 3. și 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) și cu asignarea  $S$ , în care  $S(A) = 0$  pentru fiecare  $A$  (care apare în  $F$ ).

**Arătăm** în continuare că **algoritmul este corect**. Aceasta înseamnă că ieșirea algoritmului satisface ceea ce am dorit, adică răspunsul „DA”/ $S$  corespunde faptului că formula  $F$  furnizată la intrare este satisfiabilă (și  $S \models F$ ) iar răspunsul „NU” corespunde faptului că  $F$  este nesatisfiabilă. Vom separa cazurile:

**Cazul a).** La terminarea execuției se obține „DA” și  $F$  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  $i = 0$  și  $\mathcal{J}(A) = 0$  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  $C_2 = A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow 0$  ( $k \geq 1$ ), care n-au nici o variabilă *marcată*. Avem atunci, pe scurt,  $\mathcal{J}(C_1) = \mathcal{J}(0 \bullet 0 \bullet \dots \bullet 0 \rightarrow 0) = 1$ , respectiv  $\mathcal{J}(C_2) = 1$ , de unde găsim  $\mathcal{J}(F) = 1$ .

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

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

$1 \rightarrow B$ ,  $B$  marcat), de unde rezultă că semantica lui  $\underline{C}$  în *asignarea furnizată de algoritm* este de forma  $1 \rightarrow 0$  și 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 încât formula să aibă șanse 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  $C = A_1 \wedge A_2 \wedge A_3 \dots \wedge A_k \rightarrow B$  ( $k \geq 1$ ) 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  $1 \rightarrow B \equiv B$  (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 și marchează, deci  $\mathcal{S}(B) = 1$ ).
- *Clauze  $C$  de forma  $A \rightarrow B \equiv \neg A \vee B$  (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 același tip cu aceasta și care are antecedentul marcat. Prin urmare, în orice  $\mathcal{S}$  cu  $\mathcal{S}(C) = 1$ , trebuie oricum să avem  $\mathcal{S}(A) = 1$ , deci  $\mathcal{S}(\neg A) = 0$  și atunci  $\mathcal{S}(B) = 1$  (consecința este că B se marchează, deci și  $\mathcal{S}(B) = 1$ ).

- 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  $\mathcal{S}$ , pentru a avea  $\mathcal{S}(C) = 1$ , trebuie să avem  $\mathcal{S}(B) = 1$ , precum și  $\mathcal{S}(B) = 1$ .

Revenind, am arătat într-adevăr că pentru fiecare  $\mathcal{S}$  astfel încât  $\mathcal{S}(F) = 1$ , trebuie să avem  $\mathcal{S}(A) = 1$  pentru fiecare A marcat de către algoritm, adică pentru fiecare A care satisface și  $\mathcal{S}(A) = 1$  (procesul descris mai sus se continuă pentru oricâte variabile prezente în antecedent, iar numărul acestora este finit). Prin urmare, avem și  $\mathcal{S}(\underline{C}) = 0$ , de unde rezultă că  $\mathcal{S}(F) = 0$ , 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 **Pașii** 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:  
 $F = (A \vee \neg D) \wedge (\neg C \vee \neg A \vee D) \wedge (\neg A \vee \neg B) \wedge D \wedge \neg E$ .  
 Scriem întâi  $F$  ca o mulțime de implicații, obținând  
 $F = \{D \rightarrow A, C \wedge A \rightarrow D, A \wedge B \rightarrow 0, 1 \rightarrow D, E \rightarrow 0\}$ . Înainte de  
 prima execuție a corpului buclei, avem  $i = 0$  și toate variabilele sunt  
*nemarcate*.

- **Prima execuție.** Alegem clauza  $1 \rightarrow D$  (de fapt, nu există altă posibilitate). Toate aparițiile lui  $D$  se *marchează* cu  $*_1$ :  
 $D_{*1} \rightarrow A, C \wedge A \rightarrow D_{*1}, A \wedge B \rightarrow 0, 1 \rightarrow D_{*1}, E \rightarrow 0$ .
- **A doua execuție.** Alegem  $D \rightarrow A$  (din nou, nu există decât o unică posibilitate) și  $A$  se *marchează* peste tot, cu  $*_2$ :  
 $D_{*1} \rightarrow A_{*2}, C \wedge A_{*2} \rightarrow D_{*1}, A_{*2} \wedge B \rightarrow 0, 1 \rightarrow D_{*1}, E \rightarrow 0$ .
- **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 și o structură  $\mathcal{J}$ , model pentru  $F$ , este definită prin  $\mathcal{J}(A) = 1, \mathcal{J}(B) = 0, \mathcal{J}(C) = 0, \mathcal{J}(D) = 1, \mathcal{J}(E) = 0$ . ■

Am găsit prin urmare o subclasă „convenabilă” (acest lucru este cumva subiectiv) de formule propoziționale, și anume clasa formulelor Horn, pentru care testarea satisfiabilității se poate face într-un timp „rezonabil”. Deși rezultatele teoretice generale ne spun că nu pot exista *metode sintactice mai bune* decât metoda semantică sugerată de **Algoritmul SAT** (dacă ne referim la *întreaga* mulțime **LP**), existența,

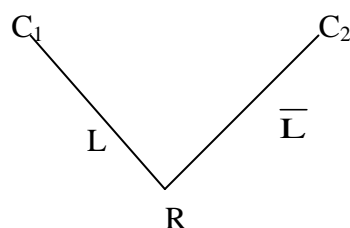
dovedită de acum, a unor algoritmi care să nu facă apel explicit la semantică, pare deja a fi un câștig.

## §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,  $R = \text{Res}_L(C_1, C_2)$ , dacă și numai dacă există un literal  $L \in C_1$  astfel încât  $\bar{L} \in C_2$  și  $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\bar{L}\})$ . ■

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



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

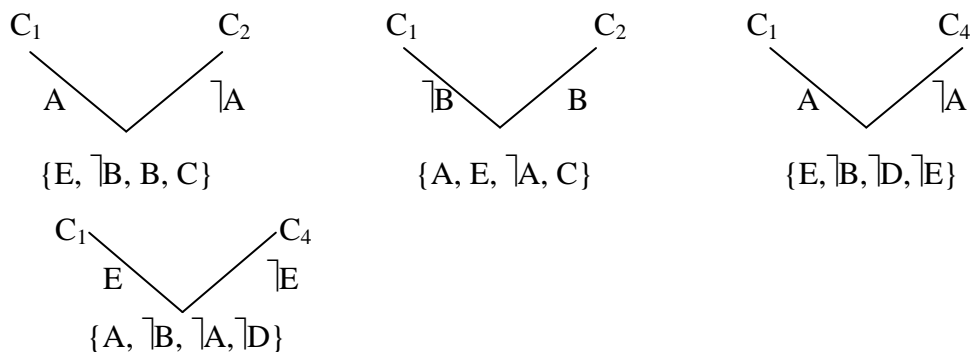
**Observație.** Rezolventul a două clauze este tot o clauză. Mai mult, rezolventul a două clauze Horn este tot o clauză Horn. Clauza vidă ( $\square$ )

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

■

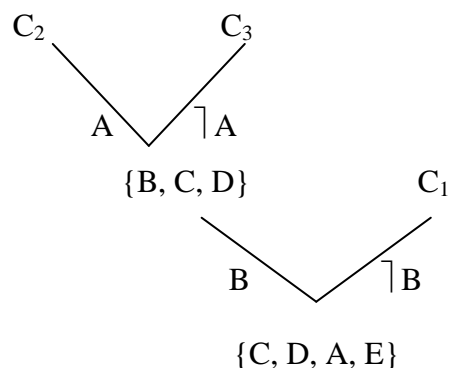
### Exemplu.

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  $C_1, C_2, C_3, C_4$ :



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





Mulți dintre acești rezolvenți „primari” nu sunt interesați, fiind tautologii (*datorită faptului că acele clauze alese spre rezolvare conțin mai mult de un literal de tipul  $L/\bar{L}$* ). Procesul poate însă continua cu găsirea de noi rezolvenți folosindu-i și pe cei obținuți din clauzele inițiale (cum este cazul și mai sus) ș.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 și presupunând că există o legătură între acest proces sintactic (de obținere de rezolvenți) și 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  $F \in \mathbf{LP}$  (aflată în **FNC** și reprezentată ca mulțime de clauze) și  $R$  un rezolvent pentru  $C_1, C_2 \in F$ . Atunci  $F$  este tare echivalentă cu  $F \cup \{R\}$ .

**Demonstrație.**

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

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

**Cazul 1.**  $S(L) = 1$ . Atunci  $S \models L$ . Dar știm că  $S \models C_2$ . Rezultă că  $S \models C_2 \setminus \{\bar{L}\}$ , de unde  $S(R) = 1$ .

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

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

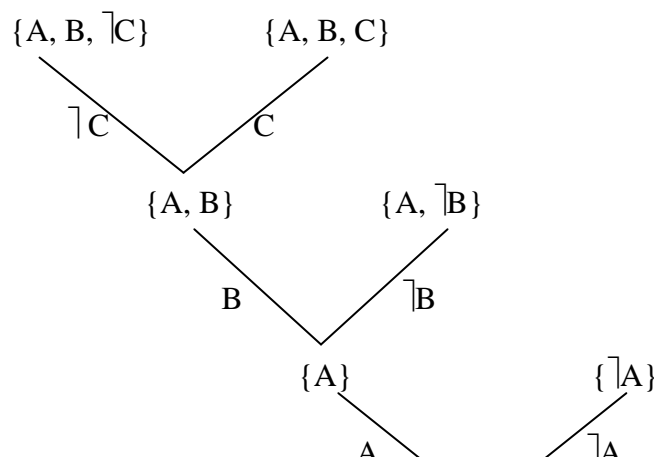
**Definiția 2.10.** Fie  $F$  o mulțime oarecare de clauze din **LP** și  $C$  o clauză. Spunem că lista  $C'_1, C'_2, \dots, C'_m$  este o **demonstrație prin rezoluție (în mai mulți pași) a lui  $C$  pornind cu  $F$**  dacă sunt satisfăcute condițiile:

- (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 și nu să fie neapărat ultimul element al acesteia. Intuitiv, o demonstrație prin rezoluție în mai mulți pași înseamnă o succesiune finită de rezoluții într-un pas, care poate fi reprezentată și 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 aceleiași clauze). În particular, dacă C este clauza vidă, atunci demonstrația respectivă se numește și **respingere**.

**Numărul de pași** 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 și măsuri specifice, cum ar fi *adâncimea* arborelui, *numărul de nivele* ([IVA]), etc. *Convenim să eliminăm din orice demonstrație rezolvenții care conțin atât L cât și  $\neg L$ , aceste clauze fiind tautologii și deci neinteresante din punctul de vedere al studiului satisfiabilității unei formule aflate în FNC.*

**Exemplu.** Fie  $F = \{\{A, B, \neg C\}, \{\neg A\}, \{A, B, C\}, \{A, \neg B\}\}$ . 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:

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

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

**Observație.** Direct din definiție rezultă că:

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

Putem da atunci și o *definiție structurală a lui  $\text{Res}^*(F)$* . Vom nota astfel cu  $\text{Resc}$  mulțimea definită prin:

**Baza.**  $F \subseteq \text{Resc}$ .

**Pas constructiv:** Dacă  $C_1, C_2 \in \text{Resc}$  și  $C = \text{Res}(C_1, C_2)$ , atunci  $C \in \text{Resc}$ . ■

Rămâne să arătăm că cele două definiții introduc aceeași mulțime.

**Teorema 2.10.** Pentru fiecare  $F \in \mathbf{LP}$ , avem  $\text{Res}^*(F) = \text{Resc}$ .

**Demonstrație.** Arătăm egalitatea prin dublă incluziune.

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

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

**Pas inductiv.** Presupunem că  $\text{Res}^{(n)}(F) \subseteq \text{Resc}$  și arătăm că  $\text{Res}^{(n+1)}(F) \subseteq \text{Resc}$ , ceea ce este din nou imediat din definiția lui  $\text{Resc}$  și

**Definiția 2.11.** În sfârșit, avem  $\text{Res}^*(F) \subseteq \text{Resc}$ , direct din **Definiția 2.11** și observația care urmează acesteia.

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

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

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

De acum înainte vom folosi ambele notații pentru mulțimea rezolvenților unei mulțimi de clauze. Și 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  $C \in \text{LP}$  se poate demonstra prin rezoluție pornind cu clauzele lui  $F$  dacă și numai dacă există  $k \in \mathbf{N}$ , astfel încât  $C \in \text{Res}^{(k)}(F)$ .

**Demonstrație.** Fie  $F$  și  $C$  fixate ca în enunț.

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

„ $\Leftarrow$ ”. Să presupunem că există  $k \in \mathbf{N}$ , astfel încât  $C \in \text{Res}^{(k)}(F)$  (pe  $k$  îl considerăm a fi cel mai mic număr natural care satisface condiția). Conform **Definiției 2.11**, avem  $\text{Res}^{(j)}(F) = \text{Res}(\text{Res}^{(j-1)}(F)) = \text{Res}^{(j-1)}(F) \cup \{R \mid \text{există } C_1, C_2 \in \text{Res}^{(j-1)}(F) \text{ astfel încât } R = \text{Res}(C_1, C_2)\}$ , 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 și în  $Res^{(k-1)}(F)$ . Dacă  $k = 0$ , am terminat ( $C \in F$  și 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 și în  $Res^{(k-1)}(F)$ ; la **Pasul 2** punem nodurile din  $Res^{(k-1)}(F)$  care nu sunt în  $Res^{(k-2)}(F)$ , precum și muchiile corespunzătoare care unesc nodurile puse deja în graf, conform rezoluțiilor într-un pas din care ele provin, ș. a. m. d. În cel mult  $k + 1$  pași, vom plasa în graf și elementele (folosite) ale lui  $F$ , precum și toate muchiile corespunzătoare rezoluțiilor într-un pas cu ajutorul cărora se construiește  $Res^{(k)}(F)$ . Considerăm acum subgraful generat de nodul  $C$  și 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 și 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 pași de rezoluție. Graful poate fi ușor transformat în arbore prin multiplicarea nodurilor de tipul  $C'$  și a arcelor aferente. ■

După cum probabil s-a putut observa, în cele de mai sus am folosit în majoritatea cazurilor termenul *mulțimea de clauze*  $F$  și nu *formula*  $F$  (aflată în  $FNC$ ). Deși 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 și pentru mulțimi infinite (numărabile) de formule (clauze). Teorema următoare stabilește o legătură importantă, privind satisfiabilitatea, între mulțimile infinite și cele finite de formule oarecare din **LP**.

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

**Demonstrație.**

„ $\Rightarrow$ ”. Dacă există structura  $\mathcal{S}$  astfel încât  $\mathcal{S} \models \mathbf{M}$ , adică  $\mathcal{S}(F) = 1$  pentru fiecare  $F \in \mathbf{M}$ , atunci evident că același lucru se întâmplă pentru fiecare submulțime (finită)  $\mathbf{M}' \subseteq \mathbf{M}$ .

„ $\Leftarrow$ ”. Pentru fiecare  $n \in \mathbb{N}$ , vom nota  $\mathbf{M}_n \cong \{F \in \mathbf{M} \mid \text{subf}(F) \cap \mathcal{A} = \text{prop}(F) \subseteq \mathcal{A}_n\}$ , adică mulțimea formulelor din **M** care sunt construite peste (cel mult) mulțimea de variabile propoziționale  $\mathcal{A}_n = \{A_1, A_2, \dots, A_n\}$ . Cum mulțimea funcțiilor booleene de  $n$  variabile ( $\text{FB}^{(n)}$ ) 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 definiții avem  $\mathbf{M}_1 \subseteq \mathbf{M}_2 \subseteq \dots \subseteq \mathbf{M}_n \subseteq \dots \subseteq \mathbf{M}$  și  $\mathbf{M} = \bigcup_{n=1}^{\infty} \mathbf{M}_n$ . Revenind, să presupunem că fiecare submulțime finită a lui **M** este satisfiabilă și să arătăm că **M** este satisfiabilă. Fie  $\mathbf{K} \subseteq \mathbf{M}$  orice submulțime finită



(satisfiabilă) a lui  $\mathbf{M}$ . 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  $\mathbf{K}$  alegem o formulă și 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:  *$\mathbf{K}$  este satisfiabilă dacă și numai dacă  $\mathbf{M}''_n$  este satisfiabilă*. Fie atunci  $S_n$  un model pentru  $\mathbf{M}''_n$ . Avem și  $S_n \sqsubseteq \mathbf{M}_n$  pentru că pentru fiecare  $F \in \mathbf{M}_n \setminus \mathbf{M}''_n$ , există  $G \in \mathbf{M}''_n$  astfel încât  $S_n(G) = S_n(F)$ . Din ipoteza noastră (fiecare submulțime finită a lui  $\mathbf{M}$  este satisfiabilă) rezultă așadar că există un șir de structuri care satisfac:

$$S_1 \sqsubseteq \mathbf{M}_1, S_2 \sqsubseteq \mathbf{M}_2, \dots, S_n \sqsubseteq \mathbf{M}_n, \dots$$

Renumerotând dacă este cazul variabilele inițiale, putem presupune că fiecare dintre mulțimile de mai sus este nevidă și că modele sunt „construite succesiv”, după cum este descris în ceea ce urmează.  $S_1$  există și este indiferent modul său de obținere. Apoi, pentru fiecare  $i \in \{1, 2, \dots\}$ , construim  $S_{i+1}$  pornind de la  $S_i$  (și modificând eventual și pe  $S_{i-1}, \dots, S_1$ ), în felul următor:  $S_{i+1}$  „pleacă” cu valorile de adevăr pentru  $A_1, A_2, \dots, A_i$  stabilite de către  $S_i$  și „fixează” o valoare pentru  $A_{i+1}$ , în mod aleator. Dacă nu avem  $S_{i+1} \sqsubseteq \mathbf{M}_{i+1}$ , atunci revenim, alegând o structură  $S_{i+1}$  care să satisfacă  $\mathbf{M}_{i+1}$  (știm că există), prin schimbarea, eventual, și a structurilor anterioare  $S_1, S_2, \dots, S_i$  (acestea vor fi simple restricții ale lui  $S_{i+1}$ , lucrul fiind evident posibil deoarece  $\mathbf{M}_i \subseteq \mathbf{M}_{i+1}$ ). Ca urmare, putem defini structura  $S : \mathcal{A} \rightarrow \{0,1\}$ , dată prin

$S(A_i) = S_i(A_i)$ , pentru fiecare  $i \in \mathbf{N}^*$ . Faptul că  $S$  este funcție și model pentru  $\mathbf{M}$  este imediat. ■

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

**Demonstrație.** Arătăm mai întâi că *există un  $k \in \mathbf{N}$  astfel încât  $\text{Res}^{(k)}(F) = \text{Res}^{(k+1)}(F)$* . Fie  $|\text{prop}(F)| = n$ . 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*  $\text{Res}^{(i)}(F)$  *nu poate depăși  $m$* . Datorită acestui fapt și existenței incluziunii  $\text{Res}^{(i)}(F) \subseteq \text{Res}^{(i+1)}(F)$  (pentru fiecare  $i \in \mathbf{N}$ ), afirmația noastră se deduce imediat. Mai mult, notând cu  $j$  pe cel mai mic  $k$  cu proprietatea de mai sus, *avem  $\text{Res}^{(j)}(F) = \text{Res}^{(j+1)}(F)$ , pentru fiecare  $l \in \mathbf{N}$*  (lucru care rezultă printr-o simplă inducție matematică și folosind **Definiția 2.11**). De aici conchidem imediat că  $\text{Res}^{(j)}(F) = \text{Res}^*(F)$ , de unde  $\text{card}(\text{Res}^*(F)) \leq m$ . ■

Reamintind că vom elimina din orice mulțime de forma  $\text{Res}^*(F)$ , pe măsură ce se obțin, toate clauzele care conțin o subformulă de tipul  $A \vee \neg A$ , 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 și numai dacă  $\square \in \text{Res}^*(F)$ .

**Demonstrație.** Conform **Teoremei de compactitate**, știm că  $F$  este nesatisfiabilă dacă și 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**).

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

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

**Baza.**  $n = 0$ . Aceasta înseamnă că  $F = \{\square\} = \text{Res}^*(F)$  și concluzia este evidentă.

**Pas inductiv.** Presupunem afirmația adevărată pentru formule construite peste  $n$  variabile propoziționale și o demonstrăm pentru

formule construite peste  $n+1$  formule atomice. Fie  $F \in \mathbf{LP}$ , construită peste  $\mathcal{A}_{n+1} = \{A_1, A_2, \dots, A_n, A_{n+1}\}$ . Pornind de la această formulă vom construi alte două formule, notate  $\mathbf{F}_0^{A_{n+1}}$  și 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 apariții a literalului pozitiv  $A_{n+1}$  din orice clauză și apoi eliminarea în totalitate a tuturor clauzelor care conțin o apariție negativă a literalului  $A_{n+1}$ .
- $\mathbf{F}_1^{A_{n+1}}$  se obține prin *dualizare*, adică din  $F$  se scot din toate clauzele aparițiile negative ale lui  $A_{n+1}$  și se elimină apoi toate clauzele care conțin apariții pozitive ale aceleiași variabile.

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

- $C$  este o clauză din  $F$ , nemodificată. Evident că valoarea lui  $S$  pentru această clauză nu se modifică și astfel nu modifică valoarea de adevăr a lui  $\mathbf{F}_0^{A_{n+1}}$  față de cea a lui  $F$  (dacă luăm în considerare doar această clauză).
- $C$  provine dintr-o clauză din  $F$ , care conținea în plus o apariție a lui  $A_{n+1}$ . Dacă  $S(A_{n+1}) = 0$ , atunci  $S(C \cup \{A_{n+1}\}) = S(C)$ , 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  $\mathcal{S}(A_{n+1}) = 1$  și nici pe  $\neg A_{n+1}$  (în acest caz  $C$  nu ar mai fi apărut în  $\mathbf{F}_0^{A_{n+1}}$ ). Prin urmare,  $C'$  apare și 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  $\neg A_{n+1}$ , restul clauzelor lui  $F$ , ca și 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  $\mathcal{S}'(F) = 1$ , 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 știm că formulele  $\mathbf{F}_0^{A_{n+1}}$  și  $\mathbf{F}_1^{A_{n+1}}$  sunt nesatisfiabile și, mai mult, sunt construite peste cel mult  $n$  variabile. Aplicând ipoteza inductivă pentru aceste formule rezultă că  $\square \in \text{Res}^*(\mathbf{F}_0^{A_{n+1}})$  și  $\square \in \text{Res}^*(\mathbf{F}_1^{A_{n+1}})$ . Conform **Teoremelor 2.11** și **2.13**, există o respingere  $(D_0) C_1, C_2, \dots, C_l = \square$ , pornind cu elementele lui  $\mathbf{F}_0^{A_{n+1}}$ , precum și o

respingere  $(D_1) B_1, B_2, \dots B_t = \square$ , pornind cu clauzele lui  $F_1^{A_{n+1}}$ .

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  $\neg A_{n+1}$  acolo unde este necesar, obținând o altă demonstrație prin rezoluție, notată  $(D_1')$  (odată  $A_{n+1}$  respectiv  $\neg A_{n+1}$  introduse, ele nu vor mai fi șterse).

Sunt posibile următoarele situații:

- Ultima clauză a lui  $(D_0')$  este  $C'_1 = \{A_{n+1}\}$  și ultima clauză a lui  $(D_1')$  rămâne  $B'_t = B_t = \square$  (sau invers,  $C'_1 = C_1 = \square$  și  $B'_t = B_t = \{\neg A_{n+1}\}$ ). Atunci concatenăm cele două liste care reprezintă demonstrațiile  $(D_0')$  și  $(D_1')$ , rezultând evident o respingere pornind cu clauzele lui  $F$ .
- Ultima clauză a lui  $(D_0')$  este  $C'_1 = \{A_{n+1}\}$  și ultima clauză a lui  $(D_1')$  este  $B'_t = \{\neg A_{n+1}\}$ . Atunci concatenăm din nou cele două liste și apoi mai facem un pas de rezoluție obținând clauza finală  $C = \text{Res}(C'_1, B'_t) = \square$ . Din nou avem o respingere pornind cu clauzele lui  $F$ .

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

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 și faptul că *testarea satisfiabilității* nu implică nimic special (în acest caz, condiția de verificat va fi  $\Box \notin \text{Res}^*(F)$ ), ca de altfel nici *testarea validității* ( $F$  este validă dacă și numai dacă  $\neg F$  este contradicție; prin urmare, putem aplica **Teorema 2.14** lui  $\neg F$ ). A *testa dacă o formulă  $F$  este satisfiabilă dar nevalidă* impune însă aplicarea teoremei anterioare atât pentru  $F$  ( $F$  este satisfiabilă dacă  $\Box \notin \text{Res}^*(F)$ ) cât și pentru  $\neg F$  ( $\neg F$  este satisfiabilă dacă  $\Box \notin \text{Res}^*(\neg F)$ ). Singura șansă (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 și restricții

**Rafinările rezoluției** sunt metode prin care se urmărește *obținerea clauzei vide (dacă acest lucru este posibil) într-un număr cât mai mic de pași de rezoluție*. Pornind cu formula  $F = \{C_1, C_2, \dots, C_n\}$  (aflată în **FNC** și scrisă ca o mulțime de clauze, la rândul lor clauzele fiind scrise ca mulțimi de literali), se poate construi efectiv mulțimea  $\text{Res}^*(F)$ , care poate fi reprezentată (fiind finită), după cum deja știm, 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 totuși excluși, deoarece reprezintă tautologii). **Teorema rezoluției** sugerează crearea mai întâi a acestui *graf de rezoluție total* și apoi *parcurea* lui pentru a vedea dacă  $\square$  este (eticheta unui) nod în graf. **Teorema 2.11** ne indică faptul că *este suficient să găsim o respingere* în loc de *a creea și apoi parcurge* întregul graf. Rafinările se împart în două mari categorii: *strategii* și *restricții*.

**Strategiile** nu restrâng, în general, spațiul de căutare (adică *graful total*) dar folosesc anumite *informații suplimentare despre clauze*, astfel încât să crească șansele 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 construiește în întregime, ci doar acele porțiuni din el (cât mai puține și 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 obișnuit). 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 nesatisfiabale pentru care nu se pot găsi respingeri, în situația în care pașii de rezoluție sunt supuși unor condiții prea restrictive)*, deoarece *spațiul de căutare este practic*



*micșorat* 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.

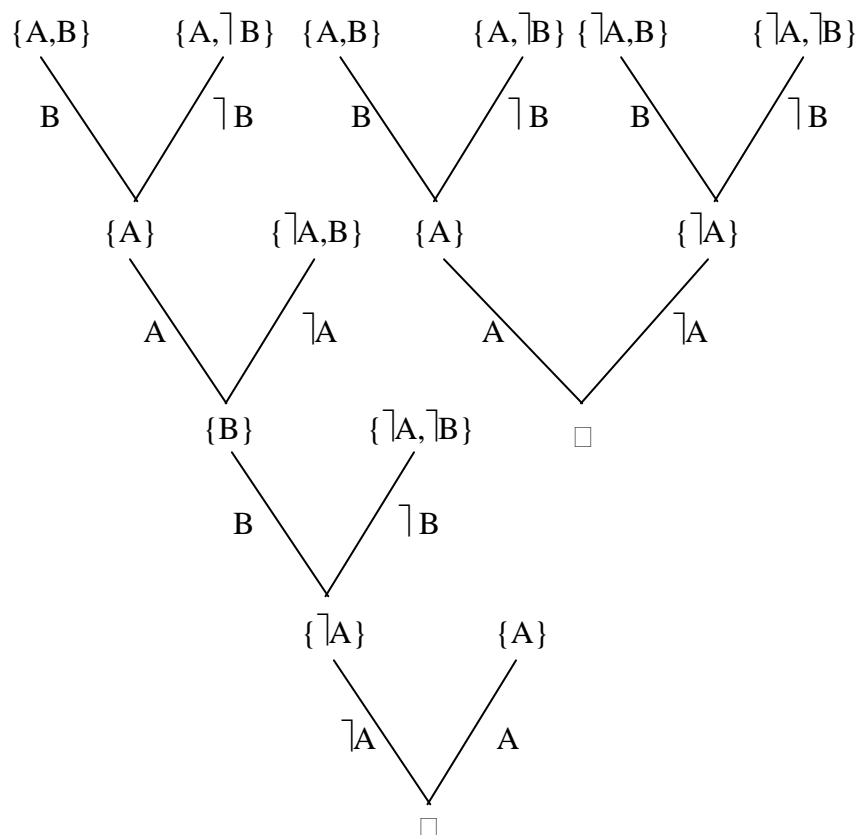
**Rezoluția pozitivă (P-rezoluția).** La fiecare pas al rezoluției (al unei demonstrații prin rezoluție), măcar una dintre clauze trebuie să fie o clauză pozitivă.

**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  $F \in \mathbf{LP}$ ,  $F = \{C_1, C_2, \dots, C_n\}$  o mulțime de clauze (numite și *clauze de intrare*) și 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  $C_1$  și  $C_2$ , 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  $C_1 = C$  și  $C_2 \in F$ . 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 așa-numita **SLD-rezoluție** (*rezoluție Liniară cu funcție de Selecție pentru clauzele Definite*).

**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 și una de intrare (a se vedea mai jos). Astfel, putem considera că  $F$  este partiționată în  $F_1 = \{C'_1, C'_2, \dots, C'_m\}$ , care sunt clauze Horn pozitive (doar acestea numindu-se aici clauze *definite*, *program*, etc.) și  $F_2 = \{N_1, N_2, \dots, N_s\}$ , care sunt clauze Horn negative (numite și **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  $C = \{A, B\}$  și pornind cu  $F = \{ \{A, B\}, \{ \neg A, B\}, \{ \neg A, \neg B\} \}$ . 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  $F \setminus T$  trebuie să fie satisfiabilă (desigur că, în principiu, acest lucru trebuie să fie cunoscut aprioric și nu după aplicarea unuia dintre algoritmi de rezoluție). 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  $F \setminus T$*  (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 cunoștințe*, exprimată printr-o mulțime satisfiabilă de formule  $F' = \{F_1, F_2, \dots, F_n\}$ , 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  $F = \{F_1, F_2, \dots, F_n, \neg G\}$  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  $G_1, G_2, \dots, G_k$ , adică din reprezentarea formulei  $\neg G$  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 și 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 același nume) măcar una dintre cele două clauze este alcătuită dintr-un unic literal*. Deoarece „lungimea” (numărul de literal ai) unui rezolvent scade cu o unitate la aplicarea oricărui pas de rezoluție, șansele 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 și rezoluția bazată pe o mulțime suport sunt complete. Rezoluția unitară, rezoluția

de intrare și **SLD**-rezoluția sunt complete doar pentru clasa formulelor Horn. ■

## §9. Recapitulare și Index

În acest capitol am introdus **sintaxa și semantica formală a unui limbaj logic**, privit în sensul unei **mulțimi de formule**. Acestea reprezintă, într-un mod precis, cunoștințele noastre despre anumite părți ale realității. Sunt transpuse într-o formă exactă conceptele și principiile principale ale logicii aristotelice, printre care *bivalența/tertium non datur* și *extensionalitatea*. **Sintactic**, mulțimea formulelor logicii propoziționale – notată **LP** – poate fi definită constructiv, pornind de la o mulțime numărabilă de formule atomice și utilizând conectorii logici și, sau, non, eventual și implică. O formulă poate fi „recunoscută fără dubii” (problema apartenenței unui șir finit de caractere la **LP** este decidabilă) și poate fi reprezentată ca un arbore. Există și 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. Definiția semanticii se bazează și pe rezultatele cunoscute despre **funcțiile booleene**. Am evidențiat apoi clasele de formule satisfiabile, valide și nesatisfiabile, introducându-se și studiindu-se alte concepte de natură semantică, cum ar fi cele de echivalență (slabă, tare) sau de consecință semantică. Problema cea mai importantă de care ne-am ocupat a fost problema **SAT** (a satisfiabilității

formulelor din **LP**), despre care am arătat că **este decidabilă dar de complexitate (timp) exponențială**. Primii algoritmi descriși 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**, deși aceștia nu sunt mai rapizi, la nivel global. Am prezentat un asemenea algoritm (**Teoremele 2.11, 2.13, 2.14**), care folosește conceptul de **rezoluție (propozițională)**. Deși metoda rezoluției nu aduce îmbunătățiri semnificative ale timpului necesar pentru rezolvarea **SAT** (în sensul teoriei generale a complexității și pentru întreaga clasă **LP**), s-au putut pune în evidență **strategii și restricții ale rezoluției**, care măresc șansele găsirii rapide a răspunsului, precum și 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 și **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 și 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 și prezentarea ei separată, din motive didactice. De altfel, deși destul de restrictivă, **LP** este suficient de „bogată” pentru a putea **exprima** (și deci, **studia formal**) afirmații interesante privind lumea reală. Urmărind exemplul de mai jos, vom înțelege poate mai ușor/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 greșesc, așadar guvernul nu minte.*

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

Logica ne oferă o modalitate de a răspunde la întrebare cât mai exact. Ideea este de a exprima afirmația anterioară, **cât mai adecvat**, ca o formulă  $F$  din **LP** și 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  $\mathcal{A}$ ):

$P$  - 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. și 3. sintetizează cunoștințele prezente în afirmație:

1. Dacă există petrol în Patagonia atunci, sau experții au dreptate, sau guvernul minte ( $F_1$ ) și
2. Nu există petrol în Patagonia sau experții greșesc ( $F_2$ ),

În afară de aceste ipoteze, există și o concluzie:

3. Așadar guvernul nu minte ( $F_3$ ).

Formula noastră ar putea fi deci:

$F = (F_1 \wedge F_2) \rightarrow F_3$ , unde  $F_1 = P \rightarrow (E \vee G)$ ,  $F_2 = \neg P \vee \neg E$  și  $F_3 = \neg G$ , adică:

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

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)

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

(de Morgan)

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

(dubla negație,  
asociativitate)

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

(distributivitate)

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

(distributivitate,  
idempotență)

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

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

(legea tautologiei,  
idempotență,  
comutativitate)



$$\begin{aligned}
&\equiv (P \vee \neg G) \wedge (P \vee \neg G \vee E) \wedge (P \vee \neg G \vee \neg E) \wedge (\neg G \vee E) \equiv \\
&\quad \text{(asociativitate,} \\
&\quad \text{distributivitate)} \\
&\equiv (P \vee \neg G) \wedge ((P \vee \neg G) \vee (E \wedge \neg E)) \wedge (\neg G \vee E) \equiv \\
&\quad \text{(legea contradicției,} \\
&\quad \text{idempotență,} \\
&\quad \text{comutativitate)} \\
&\equiv (P \vee \neg G) \wedge (E \vee \neg G).
\end{aligned}$$

Prin urmare,  $F$  poate fi considerată ca fiind formula Horn  $\{G \rightarrow P, G \rightarrow E\}$  și, printr-o aplicare imediată a **Algoritmului Horn**, obținem că  $F$  este satisfiabilă (asignarea  $S$  fiind dată de  $S(P) = S(E) = S(G) = 0$ ). Este însă  $F$  validă? Pentru aceasta putem cerceta dacă  $\neg F$  este contradicție. Obținem însă imediat (folosind legile lui de Morgan, distributivitatea și idempotența) că  $\neg F \equiv (\neg P \vee \neg E) \wedge (\neg P \vee G) \wedge (G \vee \neg E) \wedge G$ , adică, în reprezentarea cu mulțimi găsim că  $\neg F = \{P \wedge E \rightarrow 0, P \rightarrow G, E \rightarrow G, 1 \rightarrow G\}$ . Aplicarea **Algoritmului Horn** conduce la *marcarea* doar a lui  $G$  și la răspunsul că  $\neg F$  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 cunoaște **toate structurile care sunt model pentru  $F$** , acest lucru ar deveni poate explicabil. Aplicarea metodei următoare poate fi o soluție.

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

P	E	G	$\neg G$	$P \vee \neg G$	$E \vee \neg G$	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 și 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 și guvernul nu minte este adevărată*”. Să reținem însă *deosebirea esențială dintre afirmațiile din limbaj natural și 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 și 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  $F \equiv (P \vee \neg G) \wedge (E \vee \neg G)$  și dorim să vedem dacă F este nesatisfiabilă, utilizând rezoluția. Prin urmare,  $F = \{ \{P, \neg G\}, \{E, \neg G\} \}$  și practic se

găsește imediat că  $\text{Res}^*(F) = F$ ,  $\square \notin \text{Res}^*(F)$ , adică  $F$  este satisfiabilă. Lăsăm pe seama cititorului să arate, folosind tot această metodă, că  $F$  este nevalidă. ■

Nu am avea același succes dacă am dori să „exprimăm convenabil” în **LP** o afirmație ca: „O relație binară  $f \subseteq A \times B$  este funcție atunci și numai atunci când pentru fiecare element  $a \in A$ , dacă există  $b_1, b_2 \in B$  cu  $f(a) = b_1$  și  $f(a) = b_2$ , 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 relație binară.
- E:  $A'$  este o submulțime a lui  $A$  formată din toți  $a \in A$  pentru care există măcar două elemente distincte  $b_1, b_2 \in B$ , care satisfac  $f(a) = b_1$  și  $f(a) = b_2$ , sau, pentru care nu există nici un element  $b \in B$  astfel încât  $f(a) = b$ .
- H:  $A' \subseteq A$  este vidă,  $A'$  fiind cea de mai sus.
- G: relația  $f \subseteq A \times B$  este funcție,  $f$  fiind cea de mai sus, în altă notație.

Atunci formula  $F \in \mathbf{LP}$  va fi  $F = (D \wedge E \wedge H) \rightarrow G$  și va exprima cumva afirmația inițială. Sunt însă numeroase **inconveniente**, printre care: *variabilele propoziționale nu prea sunt elementare (indivizibile, cu valoare clară de adevăr); se amestecă prea mult sintaxa cu semantica; înafară de valorile 0 și 1 parcă ar mai trebui ceva, etc.* Soluția ar fi să putem **exprima direct în limbajul formal cuantificatorii** (pentru orice, există), așa cum de altfel **am și 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  $\langle x, f(x) \rangle$ ):

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

lucru care ar permite păstrarea tuturor principiilor logicii aristotelice, iar „puterea de exprimare directă” a limbajului este clar „mai mare” ( $F' \notin \mathbf{LP}$ , dar poate aparține unei supramulțimi alese corespunzător). Atunci 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 dorește neapărat, relația de apartenență a unor obiecte la anumite mulțimi, etc. **Capitolul 3** este destinat studiului unei extensii posibile a lui **LP** și anume **LP1**, *logica (calculul) cu predicate de ordinul I*.

**Indexul** termenilor importanți:

*inferențe valide*, 42

*sfera și conținutul unei noțiuni*, 43

*diferență specifică*, 43

*definiții operaționale*, 43

*paradoxuri*, 45

*silogisme*, 47

*conectori logici (non, și, sau, implică, echivalență)*, 48

*inferențe ipotetico-categorice*, 49

*modus ponens, modus tollens*, 50

*formule propoziționale*, 53  
*variabile propoziționale*, 53  
*formule atomice*, 54  
*arbore atașat unei formule*, 55  
*subformulă*, 56  
*problemă de decizie*, 57  
*literali (pozitivi, negativi)*, 59  
*clauze, clauze pozitive, clauze negative*, 59  
*clauze Horn, clauze Horn pozitive*, 59  
*asignare, interpretare, structură*, 60  
*structură completă*, 62  
*formule satisfiabile (model), valide (tautologii)*, 63  
*formule nesatisfiabile (contradicții)*, 63  
*formule tare și slab echivalente*, 64  
*consecință semantică*, 65  
*forme normale (disjunctive, conjunctive)*, 71  
*clauza vidă*, 80  
*forma implicațională a clauzelor Horn*, 80  
*rezolvent, arbore de rezoluție*, 87  
*rezoluție într-un pas*, 87  
*demonstrație prin rezoluție*, 90  
*respingere*, 91  
*mulțimea rezolvenților unei mulțimi de clauze*, 93  
*rafinări, restricții și strategii ale rezoluției*, 104  
*strategia unitară*, 105  
*rezoluția pozitivă, rezoluția negativă*, 106

rezoluția liniară bazată pe o clauză inițială, SLD-rezoluția, 106

rezoluția bazată pe o mulțime suport, 108

rezoluția de intrare, 109

## §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)  $\equiv$  și  $\equiv_s$  sunt relații de echivalență pe **LP**.
  - (b)  $\equiv$  este compatibilă la dreapta și la stânga cu  $\wedge$ ,  $\vee$  și compatibilă cu  $\neg$ .
  - (c)  $\mathcal{LP} = \langle \mathbf{LP}/\equiv, \wedge, \vee, \neg \rangle$  formează o algebră booleană.
  - (d) Între  $\mathcal{LP}$  și  $\mathcal{B} = \langle \mathbf{B}, \bullet, +, \bar{\phantom{x}} \rangle$  există măcar un homomorfism de algebre booleene.
9. Completați demonstrația **Teoremei 2.4**.
10. Să se aplice **Algoritmul Horn** formulei:
 
$$F = (\neg B \vee \neg D) \wedge \neg E \wedge \neg C \wedge B \wedge (\neg B \vee D \wedge B).$$
11. Să se exprime ca formulă în **LP** și să se studieze satisfiabilitatea afirmației: *Vom câștiga 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 și 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  $F$  și apoi să se elimine „cât mai multe” paranteze (fără a schimba semantica formulei), ținându-se cont de prioritățile atribuite operatorilor  $\neg$ ,  $\vee$ ,  $\wedge$ , precum și de alte proprietăți ale acestor operatori.

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

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

14. Arătați că formula:

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

este tautologie, folosind metoda rezoluției.

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

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

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

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

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

- (c) Dacă  $F \rightarrow G$  este validă și  $F$  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 și care să exprime faptul că  $f$  este funcție injectivă și surjectivă.
19. Fie formula  $F \in \mathbf{LP}$ ,  $F = (\bigwedge ((A \vee B) \wedge C))$ ,  $A, B, C \in \mathcal{A}$ . Să se găsească arborele care descrie formula și, simultan, o **FNC** și o **FND** pentru  $F$  (conform algoritmului recursiv sugerat de demonstrația **Teoremei 2.6**).