Logică pentru informatică - note de curs

Universitatea Alexandru Ioan Cuza, Iași
Facultatea de Informatică
Anul universitar 2024-2025

Ștefan Ciobâcă
Andrei Arusoaie
Rodica Condurache
Cristian Masalagiu

Cuprins

1 Introducere **5**

2 Logica propoziţională informală **7**
2.1 Propoziţii 7
2.2 Propoziţii atomice 8
2.3 Conjuncţii 8
2.4 Disjuncţii 9
2.5 Implicaţii 9
2.6 Negaţiile 11
2.7 Echivalenţe 12
2.8 Conectorii logici 12
2.9 Ambiguităţi în limba română 12
2.10 Fişă de exerciţii 13

3 Sintaxa formală a logicii propoziţionale **15**
3.1 Noţiunea de alfabet în informatică 15
3.2 Alfabetul logicii propoziţionale 16
3.3 Formule propoziţionale 16
3.4 Cum arătăm că un cuvânt face parte din $\mathbb{LP}$ 18
3.5 Conectorul principal al unei formule 18
3.6 Cum arătăm că un cuvânt nu face parte din $\mathbb{LP}$ 19
3.7 Proprietatea de citire unică 20
3.8 Limbaj-obiect şi meta-limbaj 21
3.9 Fişă de exerciţii 22

4 Funcţii recursive peste $\mathbb{LP}$ **25**
4.1 Arborele de sintaxă abstractă al unei formule 26
4.2 Alte exemple de funcţii definite recursiv 28
4.3 Demonstraţii prin inducţie structurală 29
4.4 Fişă de exerciţii 31

5 Semantica logicii propoziționale **33**
5.1 Algebra booleană 33
5.2 Atribuiri 33
5.3 Valoarea de adevăr a unei formule într-o atribuire 34
5.4 Satisfiabilitate 36
5.5 Formule valide 37
5.6 Formule satisfiabile, dar care nu sunt valide 37
5.7 Formule echivalente 38
5.8 Consecință semantică 38
5.9 Mulțimi consistente de formule 39
5.10 Aplicația 1 40
5.11 Fișă de exerciții 41

Capitolul 1

Introducere

Dacă aritmetica este știința care studiază numerele și operațiile cu numere, logica este știința în care obiectul studiului este reprezentat de *propoziții* și de operații cu propoziții.

De exemplu, dacă în aritmetică observăm că suma a două numere pare este un număr par, în logică am putea observa că disjuncția a două propoziții adevărate este o propoziție adevărată.

Logica se găsește la intersecția filosofiei, matematicii și informaticii și a cunoscut cea mai mare dezvoltare începând cu anii 1950, datorită aplicațiilor numeroase în Informatică.

În acest curs, vom studia la nivel introductiv *logica propozițională* și *logica de ordinul I*.

Logica propozițională este extrem de simplă, dar conceptele pe care le studiem, metodele pe care le învățăm și problemele pe care le întâlnim în logica propozițională se pot generaliza la alte logici mai complexe. De asemenea, logica propozițională corespunde în mod fidel organizării interne la nivel abstract a calculatoarelor, în sensul în care circuitele electronice se pot modela ca formule din logica propozițională. Logica propozițională are o teorie bogată și interesantă din punct de vedere matematic (exemple: teorema de compactitate, teorema de interpolare a lui Craig). *Problema satisfiabilității* pentru logica propozițională are multe aplicații în informatică. Este deosebit de importantă atât din punct de vedere teoretic (fiind problema canonică NP-completă) cât și practic (cu aplicații în verificarea programelor, în verificarea circuitelor, în optimizare combinatorială ș.a.).

Logica de ordinul I este o extensie a logicii propoziționale și are de asemenea aplicații numeroase în informatică, dar și în matematică. De exemplu, toată matematica pe care ați învățat-o în gimnaziu/liceu se bazează pe o așa numită teorie din logica de ordinul I numită **ZFC** (teoria **Z**ermelo–**F**raenkel a mulțimilor, împreună cu axioma alegerii – Axiom of **C**hoice). În Infor-

matică, aplicații ale logicii de ordinul I apar în domeniul complexității descriptive, în baze de date relaționale, în verificarea automată a hardware-ului și software-ului, ș.a. De asemenea, multe alte logici (de exemplu, logicile de ordin superior) au aplicații în teoria limbajelor de programare, în fundamentele matematicii, teoria tipurilor etc.

Capitolul 2

Logica propozițională informală

Logica propozițională este logica propozițiilor, conectate între ele prin *conectori logici* cum ar fi *sau*, *și* și *non*. În acest capitol, vom trece prin bazele logicii propoziționale.

2.1 Propoziții

O *propoziție* este o afirmație care este sau adevărată, sau falsă. Iată câteva exemple de propoziții:

1. *Port o cămașă albastră*;
2. *Tu deții un laptop și o tabletă, dar nu un telefon inteligent*;
3. $2 + 2 = 4$ (*Doi și cu doi fac patru*);
4. $1 + 1 = 1$ (*Unu plus unu este 1*);
5. $1 + 1 \neq 1$ (*Unu cu unu nu fac 1*);
6. *Dacă* $1 + 1 = 1$*, atunci Pământul este plat*;
7. *Toate numerele naturale sunt întregi*;
8. *Toate numere raționale sunt întregi.*

Iată câteva exemple de expresii care nu sunt propoziții:

1. *Roșu și negru* (nu este o afirmație);

2. π (nu este o afirmație);

3. *Plouă?* (întrebare, nu afirmație);

4. *Pleacă!* (exclamație, nu afirmație);

5. $x > 7$ (aici avem un predicat care depinde de x; după ce fixăm o valoare pentru x, obținem o propoziție);

6. *Această afirmație este falsă.* (deși este o afirmație, nu este propoziție, deoarece nu este sau adevărată sau falsă: dacă ar fi adevărată, atunci ar fi și falsă și invers).

Câteodată nu este foarte clar dacă o afirmație este propoziție cu adevărat, sau nu este foarte clară valoarea ei de adevăr. De exemplu, suntem de acord că *zăpada este albă* în general, dar la fel de bine se poate susține și contrariul: de exemplu, există zăpadă neagră (pe stradă în țările subdezvoltate în timpul iernii), astfel încât valoarea de adevăr a afirmației *zăpada este albă* este pusă în discuție. Faptul că o afirmație este propoziție sau nu este mai mult o problemă de logică filosofică.

2.2 Propoziții atomice

Unele propoziții sunt atomice, în sensul în care nu pot fi descompuse în propoziții mai mici:

1. *Port o cămașă albastră*;

2. *Tu deții un laptop*;

3. $2 + 2 = 4$ (*Doi și cu doi fac patru*).

2.3 Conjuncții

Unele propoziții sunt compuse din altele mai mici. De exemplu, propoziția *afară plouă și eu sunt supărat* este compusă din *afară plouă* și din *sunt supărat*, legate între ele prin *și*. Dacă două propoziții, φ și respectiv ψ, sunt legate printr-un *și*, propoziția rezultată, φ *și* ψ, se numește o *conjuncție* (*conjuncția lui* φ *și* ψ).

O conjuncție este adevărată dacă ambele părți componente sunt adevărate. De exemplu, propoziția *afară plouă și eu sunt supărat* este adevărată dacă atât propoziția *afară plouă* cât și propoziția *sunt supărat* sunt adevărate. În particular, din moment ce nu sunt supărat, această conjuncție este falsă.

O conjuncție nu conține neapărat cuvântul *și* în mod explicit. De exemplu, propoziția *afară plouă, dar eu am o umbrelă* este de asemenea o conjuncție, iar părțile ei componente sunt *afară plouă* și *eu am o umbrelă*. Aceste propoziții sunt legate prin conjuncția adversativă *dar*.

Exercițiul 1. *Găsiți părțile componente ale conjuncției* mă joc acasă și învăț la școală.

Exercițiul 2. *Dați un exemplu de o conjuncție falsă și un exemplu de o conjuncție adevărată.*

2.4 Disjuncții

Disjuncțiile sunt propoziții legate între ele prin *sau*. De exemplu, *afară plouă sau sunt supărat* este o disjuncție a propozițiilor *afară plouă* și *sunt supărat*.

Exercițiul 3. *Găsiți părțile componente ale disjuncției* Voi cumpăra un laptop sau o tabletă. *Atenție! Cele două părți componente trebuie să fie propoziții (anumite cuvinte din cele două părți componente pot să fie implicite și în consecință să nu apară în text).*

O disjuncție este adevărată dacă cel puțin una din părțile sale componente este adevărată. De exemplu, disjuncția $7 > 8$ *sau* $8 > 7$ este adevărată, deoarece $8 > 7$ este adevărată.

Acest înțeles al disjuncțiilor se numește *sau inclusiv* și este standard în matematică. Câteodată cuvântul *sau* este folosit în limbaj natural în sensul de *sau exclusiv*. De exemplu, în propoziția *albul sau negrul câștigă într-un joc de go*, cuvântul *sau* are înțeles de *sau exclusiv*. Înțelesul este că sau albul câștigă, sau negrul, dar nu amândoi simultan (este exclusă opțiunea să fie ambele adevărate în același timp).

În continuare, prin disjuncție vom înțelege *sau inclusiv* (interpretarea standard în matematică).

Exercițiul 4. *Dați un exemplu de o disjuncție falsă și un exemplu de o disjuncție adevărată.*

Exercițiul 5. *Când este disjuncția* φ sau ψ *falsă (în funcție de valorile lui* φ *și* ψ*)?*

Exercițiul 6. *Putem reprezenta* sau exclusiv *utilizând* sau inclusiv*?*

2.5 Implicații

Implicațiile sunt propoziții de forma *dacă* φ *atunci* ψ. Propoziția φ se numește *antecedent* al implicației, iar propoziția ψ se numește *consecvent* (sau *concluzie*) al implicației.

Un exemplu de implicație este *dacă trec la Logică, dau o petrecere.* Antecedentul este *trec la Logică*, iar concluzia este *dau o petrecere.* Când este o implicație adevărată/falsă? O implicație este falsă dacă și numai dacă antecedentul este adevărat, dar consecventul este fals. Să presupunem că trec la Logică și că totuși nu dau o petrecere. Atunci implicația *dacă trec la Logică, dau o petrecere*, în ansamblul său, este falsă (antecedentul este adevărat, dar consecventul fals).

Înțelesul unei implicații merită o discuție mai amănunțită, deoarece poate fi contraintuitiv. Implicațiile, așa cum apar în matematică, pot fi diferite de implicațiile pe care le folosim în viața de zi cu zi. În viața de zi cu zi, când spunem *dacă trec la logică, dau o petrecere*, înțelegem că avem o legătura de cauzalitate între faptul de a trece la Logică și faptul de a da o petrecere. Alte exemple de astfel de legătură de cauzalitate: *dacă am bani, cumpăr o mașină* sau *dacă mă ajuți, te ajut.* În limbajul de zi cu zi, nu ne-am gândi niciodată să conectăm două propoziții printr-o implicație dacă cele două propoziții nu au legătură de cauzalitate între ele. De exemplu, propoziția *dacă pământul este rotund, atunci 2 + 2 = 4* nu ar avea sens (deși este adevărată).

Implicația folosită în matematică se numește *implicație materială.* Valoarea de adevăr a unei implicații depinde doar de valorile de adevăr ale părților componente (antecedentul și consecventul), nu și de legătura de cauzalitate dintre ele. Acest înțeles al implicației materiale nu corespunde tot timpul cu înțelesul din limbajul natural (e.g., limba română), dar practica arată că este singurul înțeles rezonabil în matematică (și informatică).

În particular, atât propoziția *dacă pământul este plat, atunci 2+2=5*, cât și propoziția *dacă pământul este plat, atunci* $2+2=4$ sunt adevărate, deoarece antecedentul este fals.

Exercițiul 7. *Care sunt valorile de adevăr ale propozițiilor* dacă $2+2=4$, atunci Pământul este plat *și* dacă $2+2=5$, atunci Pământul este plat*?*

Valoarea de adevăr a implicației *dacă* φ*, atunci* ψ depinde doar de valorile de adevăr ale antecedentului, φ, și consecventului, ψ, și este prezentată în următorul tabel de adevăr:

φ	ψ	dacă φ, atunci ψ
fals	*fals*	*adevărat*
fals	*adevărat*	*adevărat*
adevărat	*fals*	*fals*
adevărat	*adevărat*	*adevărat*

Următorul exemplu arată că tabelul de adevăr de mai sus este singura intrepretare rezonabilă a implicației. Sunteți cu siguranță de acord că orice număr natural este număr întreg. Altfel spus, propoziția *pentru orice număr* x*, dacă* x *este număr natural, atunci* x *este număr întreg* este adevărată. În

particular veți fi de acord că propoziția este adevărată în cazurile particulare $x = -10$, $x = 10$ și $x = 1.2$ (din moment ce este vorba despre *orice număr* x).

Obținem cazurile particulare *dacă* -10 *este număr natural, atunci* -10 *este număr întreg*, *dacă* 10 *este număr natural, atunci* 10 *este număr întreg* și *dacă* 1.2 *este număr natural, atunci* 1.2 *este număr întreg.*, care trebuie toate să fie adevărate. Aceste cazuri particulare exemplifică rândurile 2, 4 și 1 ale tabelului de adevăr de mai sus (de obicei, studenții nu au încredere în rândul 2). Cât despre a treia linia, o propoziție de forma *dacă* φ*, atunci* ψ, unde φ este adevărată, dar ψ este falsă, nu poate fi decât falsă. Altfel, ar trebui să acceptăm ca fiind adevărate propoziții cum ar fi *Dacă* $2 + 2 = 4$*, atunci* $2 + 2 = 5$. (antecedentul, $2 + 2 = 4$, este adevărat, consecventul, $2 + 2 = 5$, este fals).

Unele implicații pot fi relativ dificil de identificat. De exemplu, în propoziția *trec la Logică doar dacă învăț*, antecedentul este (împotriva aparențelor) *trec la Logică*, iar consecventul este *învăț*. Atenție! Propoziția de mai sus nu are același înțeles cu *dacă învăț, trec la Logică*.

> Atenție! În propozițiile de forma *trec la Logică doar dacă învăț* sau *trec la Logică numai dacă învăț*, antecedentul este *trec la Logică*, iar consecventul este *învăț*. Aceste două propoziții nu au același înțeles cu propoziția *dacă învăț, atunci trec la Logică*.

Implicațiile în limba română pot câteodată să nu folosească șablonul *dacă ... atunci ....* De exemplu, sensul cel mai rezonabil al propoziției *trec la Logică sau renunț la facultate* (aparent o disjuncție), este că *dacă nu trec la Logică, renunț la facultate* (implicație). Din fericire, cele două propoziții sunt echivalente, într-un sens pe care îl vom studia în cursurile următoare.

2.6 Negațiile

O propoziție de forma *nu este adevărat că* φ (sau pur și simplu *nu* φ) se numește *negația* lui φ. De exemplu, *nu plouă* este negația propoziției *plouă*. Valoarea de adevăr a negației unei propoziții φ este opusul valorii de adevăr a propoziției φ. În momentul în care scriu acest text, propoziția *plouă* este falsă, și deci propoziția *nu plouă* este adevărată.

Exercițiul 8. *Dați un exemplu de o propoziție falsă care folosește atât o negație, cât și o conjuncție.*

Exercițiul 9. *În Exercițiul 6 ne întrebam dacă e posibil să reprezentăm* sau exclusiv *utilizând* sau inclusiv*. Dacă utilizăm și conjuncții și negații este posibilă această reprezentare?*

2.7 Echivalențe

O propoziție de forma φ *dacă și numai dacă* ψ se numește *echivalență* sau *dublă implicație*. O astfel de propoziție, în ansamblul său, este adevărată dacă φ și ψ au aceeași valoare de adevăr (ambele false sau ambele adevărate).

De exemplu, în momentul în care scriu acest text, propoziția *plouă dacă și numai dacă ninge* este adevărată. De ce? Deoarece atât propoziția *plouă* cât și propoziția *ninge* sunt false.

Exercițiul 10. *Care este valoarea de adevăr a propoziției* Numărul 7 este impar dacă și numai dacă 7 este număr prim*?*

O echivalență este, din punct de vedere semantic, conjuncția a două implicații: φ *dacă și numai dacă* ψ transmite aceeași informație cu

$$\underbrace{\varphi \text{ dacă } \psi}_{\text{implicația inversă}} \quad \text{și} \quad \underbrace{\varphi \text{ numai dacă } \psi}_{\text{implicația directă}}.$$

Propoziția φ *dacă* ψ este aceeași cu *dacă* ψ*, atunci* φ (doar că are altă topică). Propoziția φ *numai dacă* ψ are același înțeles cu φ *doar dacă* ψ și cu *dacă* φ*, atunci* ψ, după cum am discutat în secțiunea referitoare la implicații.

2.8 Conectorii logici

Cuvintele/expresiile *și*, *sau*, *dacă-atunci*, *doar dacă*, *non*, *dacă-și-numai-dacă* (și altele similare) sunt numite *conectori logici*, deoarece pot fi folosite pentru a conecta propoziții mai mici pentru a obține propoziții mai mari.

Atenție! O propoziție este *atomică* în logica propozițională dacă nu poate fi despărțită în propoziții mai mici separate de conectorii logici discutați mai sus. De exemplu, propoziția *orice număr natural este și număr întreg* este o propoziție atomică (în logica propozițională).

Aceeași propoziție nu mai este neapărat atomică într-o altă logică mai bogată. De exemplu, în logica de ordinul I (pe care o vom studia în partea a doua a cursului), avem doi conectori suplimentari numiți *cuantificatori*. În logica de ordinul I, propoziția *orice număr natural este și număr întreg* nu este atomică.

2.9 Ambiguități în limba română

Am prezentat mai sus limbajul logicii propoziționale: propoziții atomice conectate prin *și*, *sau*, *non* etc. Până în acest moment, am folosit limba română.

Totuși, limba română (și orice alt limbaj natural) nu este potrivită pentru scopul nostru din cauza *ambiguităților*.

Iată exemple de propoziții ambigue:

1. *Ion și Maria sunt căsătoriți* (înțelesul 1: între ei; înțelesul 2: căsătoriți, dar posibil cu alte persoane);
2. *Văd negru* (înțeles 1: sunt supărat; înțeles 2: mă simt rău; înțeles 3: nu este lumină în jur etc.);
3. *Trimit mesajul lui Ion* (înțeles 1: mesajul este al lui Ion și eu îl trimit, nu se știe unde; înțeles 2: am un mesaj și îl trimit către Ion);
4. *Nu vorbesc și mănânc* (înțeles 1: neg faptul că și vorbesc și mănânc; înțeles 2: nu vorbesc, dar mănânc).

Astfel de ambiguități sunt cel puțin neplăcute dacă scopul nostru este să determinăm valoarea de adevăr a unei propoziții (dacă nici măcar nu suntem siguri ce înseamnă propoziția). Pentru peste 2000 de ani, logica a lucrat cu limbaj natural. Nevoia de a introduce un limbaj formal, simbolic, fără ambiguități, a apărut în secolele XVIII - XIX, odată cu dezvoltarea logicii matematice. În logica simbolică/formală, pe care urmează să o studiem, vom lucra cu un limbaj artificial, numit limbaj formal, care este proiectat de o asemenea manieră încât să nu conțină nicio ambiguitate.

În fapt, primul limbaj formal pe care îl vom studia va fi *limbajul formal al logicii propoziționale* (pe scurt, *logica propozițională*).

În cele de mai sus, am observat care sunt dezavantajele și eventualele capcane ale logicii informale. În domeniul programării calculatoarelor, toate cerințele sunt exprimate via limbaj natural. Specificațiile unui produs software sunt de regulă transmise sub această formă. Este foarte important pentru cei care dezvoltă propriu-zis produsul software să identifice posibile lacune și ambiguități din specificații pentru a preveni viitoare comportamente nedorite ale produsului. Unul din scopurile studierii logicii în domeniul informaticii este să dezvolte abilitatea programatorilor de a gândi sistematic și de a formula clar specificațiile unui produs.

2.10 Fișă de exerciții

Exercițiul 11. *Stabiliți care dintre următoarele expresii sunt propoziții:*

1. Tu ai un laptop.

2. Zăpada este albă.

3. Zăpada nu este albă.

4. Tatăl meu merge la serviciu şi eu merg la şcoală.
5. Afară plouă, dar eu am umbrelă.
6. Mâine va ploua sau nu va ploua.
7. Dacă obţin notă de trecere la logică, voi sărbători.
8. $2 + 2 = 4$. *(*Doi plus doi egal cu 4.*)*
9. Roșu și Negru.
10. π.
11. Plouă?
12. Hai la pescuit!
13. x este mai mare decât 7.
14. Această afirmație este falsă.

Exercițiul 12. *Pentru toate propozițiile identificate, stabiliți dacă sunt propoziții atomice sau compuse, iar dacă sunt compuse, stabiliți dacă sunt conjuncții, disjuncții, implicații, negații, echivalențe.*

Exercițiul 13. *Stabiliți dacă propozițiile de mai jos sunt propoziții atomice sau compuse, iar dacă sunt compuse, stabiliți tipul acestora (conjuncții, disjuncții, implicații, negații, echivalențe). Pentru fiecare propoziție compusă, discutați care sunt valorile de adevăr în funcție de valorile de adevăr ale componentelor.*

1. Am mâncat atât de mult, încât mi-a fost rău.
2. Nu am rochie, nici pantofi.
3. Ma întorc acasă sau la amicul meu.
4. Merg afară dacă nu plouă.
5. Mă duc la el doar dacă nu mă ascultă.
6. Dacă nu merg la pescuit atunci soarele nu este rotund.
7. Dacă soarele nu este rotund atunci merg la pescuit.

Exercițiul 14. *Reformulați exemplele de propoziții din Secțiunea 2.9 astfel încât ambiguitățile să fie evitate. Puteți să alegeți oricare înțeles.*

Capitolul 3

Sintaxa formală a logicii propoziționale

În continuare, vom studia limbajul formal al logicii propoziționale.

Prin *sintaxă* înțelegem, în general, un ansamblu de reguli pentru scrierea corectă. După cum am promis în capitolul anterior, în acest capitol vom dezvolta un limbaj artificial, pe care îl vom numi *logica propozițională formală* (pe scurt, îl vom numi doar *logica propozițională*). Sintaxa formală a logicii propoziționale se referă la regulile de scriere pentru acest limbaj.

3.1 Noțiunea de alfabet în informatică

În contextul informaticii, prin *alfabet* se înțelege o mulțime. Elementele unui alfabet se numesc *simboluri*. De cele mai multe ori, alfabetul este o mulțime finită (dar nu în cazul logicii propoziționale).

Care este diferența dintre o mulțime și un alfabet? A priori, niciuna. Contează intenția – ce vom face cu elementele mulțimii/alfabetului mai departe. Folosim elementele unui alfabet pentru a crea *cuvinte*. Un *cuvânt* peste un alfabet este o secvență de simboluri din alfabet.

Exemplul 15. *Fie alfabetul* $X = \{0, 1\}$. *Șirurile/secvențele de simboluri* `001010`, `101011` *și* `1` *sunt exemple de cuvinte peste alfabetul* X.

Cuvântul *vid*, format din 0 simboluri ale alfabetului este notat de obicei cu ε.

3.2 Alfabetul logicii propoziționale

Limbajul logicii propoziționale este format din *formule propoziționale*, care modelează propoziții din logica propozițională, propoziții despre care am discutat în capitolul anterior.

Formulele propoziționale sunt cuvinte peste *alfabetul logicii propoziționale* (anumite cuvinte sunt formule; nu toate).

Alfabetul logicii propoziționale este format din reuniunea următoarelor mulțimi:

1. $A = \{\mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{p}', \mathtt{q}_1, \ldots\}$, mulțimea *variabilelor propoziționale*;
2. $\{\neg, \wedge, \vee\}$, mulțimea *conectorilor logici*;
3. $\{(,)\}$, mulțimea simbolurilor auxiliare (un simbol pentru paranteză deschisă și un simbol pentru paranteză închisă).

Astfel, mulțimea $L = A \cup \{\neg, \wedge, \vee\} \cup \{(,)\}$ este alfabetul logicii propoziționale. Iată exemple de cuvinte peste mulțimea L:

1. $))\mathtt{p} \vee \wedge$;
2. $\vee \vee \neg(\mathtt{p})$;
3. $\mathtt{p}$;
4. $\mathtt{ppp}$;
5. $\neg(\mathtt{p} \vee \mathtt{q})$.

În cele ce urmează vom vedea ca doar anumite cuvinte se numesc formule propoziționale. Unii autori folosesc termenul de *well-formed formula* sau *wff*, dar în aceste note de curs vom utiliza doar noțiunea de formulă.

De exemplu, ultimul cuvânt din lista de mai sus, $\neg(\mathtt{p} \vee \mathtt{q})$, este o formulă a logicii propoziționale, dar al doilea cuvânt, $\vee \vee \neg(\mathtt{p})$, nu este formulă. Următoarea definiție surprinde cu precizie care cuvinte sunt formule și care cuvinte nu sunt formule propoziționale.

3.3 Formule propoziționale

Definiția 16 (Mulțimea formulelor propoziționale, notată $\mathbb{LP}$)**.** *Mulțimea* $\mathbb{LP}$ *(mulțimea formulelor propoziționale) este* cea mai mică mulțime *de cuvinte peste alfabetul logicii propoziționale care satisface următoarele proprietăți:*

• *Cazul de Bază. Orice variabilă propozițională, văzută ca un cuvânt de lungime* 1*, este în mulțimea* $\mathbb{LP}$*;*

• *Cazul Inductiv 1. Dacă* $\varphi \in \mathbb{LP}$*, atunci* $\neg\varphi \in \mathbb{LP}$ *(sau: dacă cuvântul* φ *este o formulă propozițională, atunci și cuvântul care începe cu simbolul* $\neg$ *și continuă cu simbolurile din* φ *este formulă propozițională);*

• *Cazul Inductiv 2. Dacă* $\varphi_1, \varphi_2 \in \mathbb{LP}$*, atunci* $(\varphi_1 \wedge \varphi_2) \in \mathbb{LP}$ *(sau: . . . – exercițiu pentru acasă);*

• *Cazul Inductiv 3. Dacă* $\varphi_1, \varphi_2 \in \mathbb{LP}$*, atunci* $(\varphi_1 \vee \varphi_2) \in \mathbb{LP}$ *(sau: . . . – exercițiu pentru acasă).*

Iată câteva exemple de elemente ale mulțimii $\mathbb{LP}$:

$$\mathtt{p},\quad \mathtt{q},\quad \neg\mathtt{p},\quad \neg\mathtt{q},\quad \neg\mathtt{p}',\quad \neg\neg\mathtt{p}_1,\quad (\mathtt{p}\vee\mathtt{q}),\quad (\mathtt{p}\wedge\mathtt{q}),$$
$$\neg(\mathtt{p}\vee\mathtt{q}),\quad (\neg\mathtt{p}\wedge\neg\mathtt{q}),\quad \neg(\neg\neg\mathtt{p}\vee\mathtt{p}),\quad ((\mathtt{p}\vee\mathtt{q})\wedge\mathtt{r}),$$
$$(\mathtt{p}\vee(\mathtt{q}\wedge\mathtt{r})).$$

Iată exemple de cuvinte care nu sunt în $\mathbb{LP}$:

$$\mathtt{pp},\quad \mathtt{q}\neg\mathtt{q},\quad \mathtt{q}\wedge\neg\mathtt{p},\quad \mathtt{p}+\mathtt{q}.$$

Definiția mulțimii $\mathbb{LP}$ este un exemplu de *definiție inductivă*. Astfel de definiții sunt foarte importante în informatică și este obligatoriu să ajungem să le înțelegem foarte bine. Într-o definiție inductivă, există de obicei unul sau mai multe cazuri de bază, care definesc cele mai *mici* elemente ale mulțimii (în cazul nostru, variabilele propoziționale). Cazurile inductive arată cum se pot construi elemente mai *mari* pornind de la alte elemente mai *mici*, despre care știm deja că sunt în mulțime. Alt element important este restricția de minimalitate (subliniată în definiția de mai sus), care indică că **doar** elementele care pot fi construite apelând la cazurile de bază și la cazurile inductive fac parte din mulțime. Această restricție de minimalitate asigură unicitatea mulțimii (nu există o altă mulțime cu cele 4 proprietăți și care să fie minimală) și ne permite să arătăm ca anumite elemente nu fac parte din mulțime (cele care nu pot fi construite prin cazurile de bază/inductive).

De exemplu, cuvântul $\mathtt{pp}$ nu este în mulțimea $\mathbb{LP}$, deoarece nu poate fi construit prin aplicarea niciunui caz dintre cele 4 (cazul de bază sau cele trei cazuri inductive).

3.4 Cum arătăm că un cuvânt face parte din $\mathbb{LP}$

Putem arăta că un cuvânt face parte din $\mathbb{LP}$ explicând cum se pot aplica pasul de bază și pașii inductivi. Iată o demonstrație a faptului că $\neg(p \vee q) \in \mathbb{LP}$:

1. $p \in \mathbb{LP}$ (din pasul de bază, deoarece $p \in A$);
2. $q \in \mathbb{LP}$ (din pasul de bază, deoarece $q \in A$);
3. $(p \vee q) \in \mathbb{LP}$ (din pasul inductiv 3, unde $\varphi_1 = p$ și $\varphi_2 = q$);
4. $\neg(p \vee q) \in \mathbb{LP}$ (din pasul inductiv 1, unde $\varphi = (p \vee q)$).

Putem rearanja lista de mai sus într-un *arbore de construcție adnotat* echivalent pentru formula $\neg(p \vee q)$:

$$\dfrac{\dfrac{\dfrac{}{p \in \mathbb{LP}}\ \textit{pas de bază} \qquad \dfrac{}{q \in \mathbb{LP}}\ \textit{pas de bază}}{(p \vee q) \in \mathbb{LP}}\ \textit{pas inductiv 3}}{\neg(p \vee q) \in \mathbb{LP}}\ \textit{pas inductiv 1}$$

Vom vedea acest tip de notație de mai multe ori în Informatică și de aceea e important să ne familiarizăm cu ea. Fiecare linie orizontală se numește *inferență*; sub fiecare astfel de linie este concluzia inferenței și deasupra ei sunt ipotezele (0, 1 sau mai multe ipoteze). Inferențele cu 0 ipoteze se numesc axiome. Lângă fiecare linie se găsește numele regulii care a fost aplicată.

Dacă renunțăm la adnotații, obținem următorul *arbore de construcție* pentru formula $\neg(p \vee q)$:

$$\dfrac{\dfrac{\dfrac{}{p} \qquad \dfrac{}{q}}{(p \vee q)}}{\neg(p \vee q)}$$

Este ușor de văzut că un cuvânt aparține $\mathbb{LP}$ dacă și numai dacă există un arbore de construcție pentru acel cuvânt.

3.5 Conectorul principal al unei formule

O formulă care constă doar dintr-o variabilă propozițională (cum ar fi formula p sau formula q) se numește *formulă atomică*. Acest lucru explică de ce mulțimea A a variabilelor propoziționale se numește A (A, de la *atomic*).

Formulele mai complexe, cum ar fi $\neg p$ sau $(p \vee q)$, se numesc *compuse* (sau, în engleză, *moleculare*).

Orice formulă compusă are un *conector principal*, care este dat de ultima inferență din arborele său de construcție. De exemplu, conectorul principal al formulei $\neg(p \vee q)$ este $\neg$ (negația), în timp ce conectorul principal al formulei $(\neg p \vee q)$ este $\vee$ (disjuncția). Numim formulele al căror conector principal este $\wedge$ *conjuncții*. Similar, dacă conectorul principal al unei formule este $\vee$, formula este o *disjuncție*, iar dacă conectorul principal este $\neg$, atunci este o *negație*.

Orice formulă compusă are un conector principal.

3.6 Cum arătăm că un cuvânt nu face parte din $\mathbb{LP}$

Este puțin mai dificil (sau, mai bine spus, lung) să arătăm că un cuvânt nu face parte din $\mathbb{LP}$.

Dacă cuvântul nu este peste alfabetul corect, așa cum este cuvântul cu trei simboluri $p + q$ (care folosește simbolul străin $+$), atunci este evident că acesta nu face parte din $\mathbb{LP}$, deoarece $\mathbb{LP}$ conține doar cuvinte peste alfabetul logicii propoziționale.

Dacă cuvântul este peste alfabetul corect și vrem să arătăm că nu face parte din $\mathbb{LP}$, atunci trebuie să ne folosim de condiția de minimalitate. Iată cum putem arăta că $(p\neg q) \notin \mathbb{LP}$:

Să presupunem, prin reducere la absurd, că $(p\neg q) \in \mathbb{LP}$. În acest caz, din condiția de minimalitate, faptul că $(p\neg q) \in \mathbb{LP}$ trebuie să poată fi explicat prin una dintre cele patru reguli de formare (pasul de bază sau unul dintre cei trei pași inductivi).

Dar nu putem aplica cazul de bază pentru a arăta că $(p\neg q) \in \mathbb{LP}$, deoarece $(p\neg q) \notin A$.

De asemenea, nu putem aplica nici cazul inductiv 1, deoarece nu există nicio formulă φ astfel încât $(p\neg q) = \neg\varphi$ (orice formulă $\varphi \in \mathbb{LP}$ am alege, primul simbol al părții stângi ar fi $($, iar primul simbol al părții drepte ar fi $\neg$.

Nu putem aplica nici cazul inductiv 2, deoarece nu există formulele $\varphi_1, \varphi_2 \in \mathbb{LP}$ astfel încât $(p\neg q) = (\varphi_1 \wedge \varphi_2)$ (oricum am alege $\varphi_1, \varphi_2 \in \mathbb{LP}$, cuvântul din partea stângă nu conține simbolul $\wedge$, în timp ce cuvântul din dreapta îl conține).

Dintr-un motiv similar, nu putem aplica nici cazul inductiv 3.

Din moment ce niciunul dintre cele patru cazuri nu funcționează, presupunerea noastră este falsă, și deci $(p\neg q) \notin \mathbb{LP}$.

3.7 Proprietatea de citire unică

Definiția $\mathbb{LP}$ are o proprietate importantă, care se numește *proprietatea de citire unică*:

Teorema 17 (Proprietatea de citire unică a formulelor propoziționale)**.** *Pentru orice formulă $\varphi \in \mathbb{LP}$, există un unic arbore de construcție.*

Proprietatea de mai sus se mai numește și neambiguitatea gramaticii logicii propoziționale. Demonstrarea proprietății nu face parte din acest curs. Totuși, trebuie să înțelegem semnificația ei. În acest sens, vom considera o altă posibilă definiție (greșită) pentru $\mathbb{LP}$ și vom arată în ce sens această definiție alternativă este ambiguă.

Începutul unei definiții greșite pentru $\mathbb{LP}$.

Să ne imaginăm că pasul inductiv 3 ar fi:

$\vdots$

- Cazul Inductiv 3. Dacă $\varphi_1, \varphi_2 \in \mathbb{LP}$, atunci $\varphi_1 \vee \varphi_2 \in \mathbb{LP}$;

$\vdots$

Singura diferență față de definiția corectă este că nu putem paranteze în jurul disjuncțiilor. Cu această definiție alternativă, fictivă, a lui $\mathbb{LP}$, am avea că $\neg p \vee q \in \mathbb{LP}$. Totuși, acest cuvânt ar avea doi arbori de construcție diferiți:

$$\dfrac{\dfrac{\overline{p} \qquad \overline{q}}{p \vee q}}{\neg p \vee q} \qquad \dfrac{\dfrac{\overline{p}}{\neg p} \qquad \overline{q}}{\neg p \vee q}$$

O astfel de ambiguitate ar fi deosebit de neplăcută pentru scopurile noastre, deoarece nu am putea ști dacă $\neg p \vee q$ este o disjuncție (ca în arborele de construcție din dreapta) sau o negație (ca în arborele de construcție din stânga). De fapt,

> evitarea unor asemenea ambiguități sintactice a fost unul dintre motivele pentru care am părăsit sfera limbajului natural și am început să studiem logica formală.

Sfârșitul unei definiții greșite pentru $\mathbb{LP}$.

După această incursiune, ar trebui să fie clar de ce teorema de citire unică este netrivială (ca un exercițiu pentru cei cu înclinare spre matematică, încercați să o demonstrați). De asemenea, ar trebui să fie clar de ce citirea unică este o proprietate esențială a definiției formulelor: ne arată că orice formulă propozițională poate fi citită într-un singur fel; cu alte cuvinte, orice formulă propozițională nu are ambiguități sintactice.

3.8 Limbaj-obiect și meta-limbaj

O particularitate a logicii este că studiem propoziții (analizăm, de exemplu, valoarea lor de adevăr), iar pentru a efectua studiul, ne folosim de raționament care, în mod natural, sunt și ele alcătuite din propoziții. De exemplu, în cursul studiului nostru, am putea efectua următorul raționament:

> *Propoziția nu merg la școală este falsă dacă propoziția merg la școală este adevărată.*

De observat că afirmațiile care fac obiectul studiului (*nu merg la școală* și *merg la școală*) sunt propoziții, dar și întreaga afirmație *Propoziția nu merg la școală este falsă dacă propoziția merg la școală este adevărată* este la rândul ei o propoziție. Astfel, pare ca facem un raționament circular, în care studiem chiar metoda de studiu.

Această dificultate aparentă nu apare și în cazul aritmeticii, de exemplu. În aritmetică, studiem numere și operații peste numere, iar studiul îl facem folosind propoziții.

Dificultatea își are rezolvarea prin separarea strictă a *limbajului obiect* de *meta-limbaj*. Limbajul obiect este limbajul pe care îl studiem ($\mathbb{LP}$), iar meta-limbajul este limbajul pe care îl folosim pentru a efectua studiul (limba română).

Limbajul-obiect este limbajul care face obiectul studiului (în cazul nostru, logica propozițională). Meta-limbajul este limbajul pe care îl folosim pentru a comunica despre obiectul studiului (în cazul nostru, limba română).

În acest curs, pentru a diferenția ușor între cele două, facem următoarea convenție: toate elementele din limbajul-obiect le scriem cu font sans-serif albastru (de exemplu, $(\mathtt{p} \wedge \mathtt{q})$), iar afirmațiile din meta-limbaj le scriem folosind *font negru obișnuit* (de exemplu, *orice formulă atomică este o formulă*).

> Este extrem de important să facem diferența între limbajul-obiect și meta-limbaj. În acest scop, notele de curs folosesc o convenție tipografică. Elementele meta-limbajului sunt scrise cu font obișnuit, de culoare neagră. Elementele limbajului-obiect sunt scrise astfel: $(\mathtt{p} \wedge \mathtt{q})$. Din acest motiv, dacă imprimați notele de curs, este recomandat să folosiți o imprimantă color.

3.9 Fișă de exerciții

Exercițiul 18. *Arătați că următoarele cuvinte sunt* formule propoziționale *(adică elemente ale mulțimii $\mathbb{LP}$), explicând care sunt pașii de construcție (pas de bază, respectiv unul dintre cei trei pași inductivi):*

1. $\neg \mathtt{q}$;
2. $(\mathtt{p}_1 \wedge \mathtt{q})$;
3. $\neg(\mathtt{p} \vee \mathtt{q})$;
4. $(\neg \mathtt{p} \vee \neg \mathtt{q})$;
5. $(\neg \mathtt{p} \wedge \neg \mathtt{q})$.

Exercițiul 19. *Arătați că următoarele cuvinte nu sunt elemente ale mulțimii $\mathbb{LP}$ (indicație: arătați că niciuna dintre cele 4 reguli de formare nu poate fi aplicată):*

1. $(\neg)\mathtt{q}$;
2. $\mathtt{q} \wedge \neg$;
3. $\mathtt{pq}$;
4. $\mathtt{p} \wedge \mathtt{q}$;
5. $(\mathtt{p}) \wedge (\mathtt{q})$.

Exercițiul 20. *Care dintre următoarele cuvinte sunt formule din $\mathbb{LP}$ și care nu sunt:*

1. p_1;
2. $p_1 \vee q_1$;
3. $(p_1 \vee q_1)$;
4. $(\neg p_1 \vee q_1)$;
5. $\neg(p_1 \vee q_1)$;
6. $((\neg p_1) \vee q_1)$;
7. $(\neg p)$?

Exercițiul 21. *Dați exemple de 5 formule propoziționale interesante (cu mai mulți conectori, mai multe variabile propoziționale etc.) și justificați că sunt într-adevăr formule.*

Capitolul 4

Funcții recursive peste $\mathbb{LP}$

Memento. *Dacă X este o mulțime, prin 2^X notăm* mulțimea părților lui *X, adică mulțimea tuturor submulțimilor lui X. De exemplu, dacă $X = \{1,2,3\}$, atunci $2^X = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$. Când X este finită, avem că $|2^X| = 2^{|X|}$, ceea ce explică notația. În liceu, este posibil să fi folosit notația $\mathcal{P}(X)$ în loc de 2^X.*

Dacă o mulțime este *definită inductiv*, putem defini *funcții recursive* care operează pe elementele mulțimii. Definiția acestor funcții urmează *structura* elementelor din mulțimea definită inductiv.

Iată un exemplu de funcție care calculează mulțimea *subformulelor unei formule*:

$subf : \mathbb{LP} \to 2^{\mathbb{LP}}$, definită prin

$$subf\Big(\varphi\Big) = \begin{cases} \{\varphi\}, & \text{dacă } \varphi \in A; \\ \{\varphi\} \cup subf\Big(\varphi'\Big), & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in \mathbb{LP}; \\ \{\varphi\} \cup subf\Big(\varphi_1\Big) \cup subf\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}; & \\ \{\varphi\} \cup subf\Big(\varphi_1\Big) \cup subf\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \vee \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}. & \end{cases}$$

Iată cum se poate calcula $subf\Big(\varphi\Big)$ pentru $\varphi = \neg(\mathtt{p} \vee \mathtt{q})$:

$$\begin{aligned} subf\Big(\neg(\mathtt{p} \vee \mathtt{q})\Big) &= \{\neg(\mathtt{p} \vee \mathtt{q})\} \cup subf\Big((\mathtt{p} \vee \mathtt{q})\Big) \\ &= \{\neg(\mathtt{p} \vee \mathtt{q})\} \cup \{(\mathtt{p} \vee \mathtt{q})\} \cup subf\Big(\mathtt{p}\Big) \cup subf\Big(\mathtt{q}\Big) \\ &= \{\neg(\mathtt{p} \vee \mathtt{q})\} \cup \{(\mathtt{p} \vee \mathtt{q})\} \cup \{\mathtt{p}\} \cup \{\mathtt{q}\} \\ &= \{\neg(\mathtt{p} \vee \mathtt{q}), (\mathtt{p} \vee \mathtt{q}), \mathtt{p}, \mathtt{q}\}. \end{aligned}$$

Observați cele două tipuri de paranteze care apar în calculul de mai sus. Pe de o parte, avem parantezele care mărginesc argumentul funcției *subf*, iar pe de altă parte avem parantezele din alfabetul logicii propoziționale. Primul tip de paranteze sunt paranteze obișnuite din matematică, în timp ce al doilea tip sunt simboluri ale alfabetului. Pentru a le diferenția, vom adopta convenția următoare în aceste note de curs: folosim paranteze cu font obișnuit, de obicei mai mari, pentru apelul de funcție (de exemplu (sau $\Big($), și parantezele $\mathtt{(}$ și $\mathtt{)}$, scrise cu font $\mathtt{sans-serif\ albastru}$, pentru simbolurile alfabetului.

Ambele tipuri de paranteze sunt importante. De exemplu, este o greșeală să scriem $subf\Big(\mathtt{p} \vee \mathtt{q}\Big)$ în loc de $subf\Big((\mathtt{p} \vee \mathtt{q})\Big)$, din moment ce cuvântul $\mathtt{p} \vee \mathtt{q}$ nu este o formulă.

Este o greșeală să scriem $subf\Big(\mathtt{p} \vee \mathtt{q}\Big)$ în loc de $subf\Big((\mathtt{p} \vee \mathtt{q})\Big)$, din moment ce cuvântul $\mathtt{p} \vee \mathtt{q}$ nu este o formulă.

Funcția *subf* este prima dintre funcțiile pe care le vom defini recursiv, în funcție de structura formulei $\varphi \in \mathbb{LP}$. Aceste funcții se numesc *recursive structural*. Este foarte important să înțelegem cum operează aceste funcții pentru a înțelege restul cursului. În secțiunile următoare prezentăm și alte exemple de funcții recursive structural.

4.1 Arborele de sintaxă abstractă al unei formule

Următoarea funcție calculează *arborele de sintaxă abstractă* al unei formule:

$arb : \mathbb{LP} \to Trees$, definită prin:

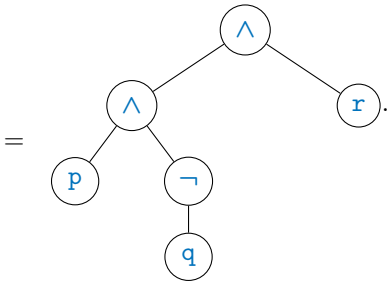
$$arb\Big(\varphi\Big) = \begin{cases} \varphi, & \text{dacă } \varphi \in A; \\ \neg \\ arb\Big(\varphi'\Big) , & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in \mathbb{LP}; \\ \wedge \\ arb\Big(\varphi_1\Big) \quad arb\Big(\varphi_2\Big) , & \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}; \\ \vee \\ arb\Big(\varphi_1\Big) \quad arb\Big(\varphi_2\Big) , & \text{dacă } \varphi = (\varphi_1 \vee \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}. \end{cases}$$

Iată un calcul al arborelui abstract de sintaxă al formulei $((\mathtt{p} \wedge \neg\mathtt{q}) \wedge \mathtt{r})$.

$$arb\Big(((\mathtt{p} \wedge \neg\mathtt{q}) \wedge \mathtt{r})\Big) = \wedge \quad arb\Big((\mathtt{p} \wedge \neg\mathtt{q})\Big) \quad arb\Big(\mathtt{r}\Big)$$

$$= \wedge \quad \mathtt{r} \quad arb\Big(\mathtt{p}\Big) \quad arb\Big(\neg\mathtt{q}\Big)$$

$$= \wedge \quad \mathtt{r} \quad \mathtt{p} \quad \neg \quad arb\Big(\mathtt{q}\Big)$$



Mulțimea *Trees* este mulțimea arborilor cu rădăcină, etichetați.

Arborii de sintaxă abstractă sunt extrem de importanți deoarece, din punct de vedere conceptual, o formulă propozițională *este* arborele abstract de sintaxă. Totuși, deoarece scrierea arborilor nu este convenabilă (pentru persoane), recurgem la notația convențională, și scriem formulele în stil tradițional, de la stânga la dreapta, sub formă de cuvinte. În schimb, la implementarea pe un calculator a diferiților algoritmi care prelucrează formule logice, vom prefera reprezentarea formulelor sub forma arborelui de sintaxă abstractă în dauna reprezentării sub forma unui cuvânt (șir de caractere).

> Din punct de vedere conceptual, o formulă propozițională *este* arborele abstract de sintaxă.

4.2 Alte exemple de funcții definite recursiv

Iată și alte exemple de funcții definițe prin recursie structurală peste formule.

Prima funcție, $height : \mathbb{LP} \to \mathbb{N}$, calculează *înălțimea* arborelui abstract de sintaxă al unei formule:

$$height\Big(\varphi\Big) = \begin{cases} 1, & \text{dacă } \varphi \in A; \\ 1 + height\Big(\varphi'\Big), & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in \mathbb{LP}; \\ 1 + max\Big(height\Big(\varphi_1\Big), height\Big(\varphi_2\Big)\Big), & \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}; \\ 1 + max\Big(height\Big(\varphi_1\Big), height\Big(\varphi_2\Big)\Big), & \text{dacă } \varphi = (\varphi_1 \vee \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}. \end{cases}$$

Funcția $max : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ care apare în definiția funcției *height* este funcția obișnuită care calculează maximul dintre două numere naturale.

Următoarea funcție, $size : \mathbb{LP} \to \mathbb{N}$, calculează *dimensiunea* arborelui abstract de sintaxă al unei formule (adică numărul de noduri):

$$size\Big(\varphi\Big) = \begin{cases} 1, & \text{dacă } \varphi \in A; \\ 1 + size\Big(\varphi'\Big), & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in \mathbb{LP}; \\ 1 + size\Big(\varphi_1\Big) + size\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}; & \\ 1 + size\Big(\varphi_1\Big) + size\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \vee \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}. & \end{cases}$$

Ultimul exemplu, $prop : \mathbb{LP} \to 2^A$, calculează toate variabilele propoziționale care apar într-o formulă:

$$prop\Big(\varphi\Big) = \begin{cases} \{\varphi\}, & \text{dacă } \varphi \in A; \\ prop\Big(\varphi'\Big), & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in \mathbb{LP}; \\ prop\Big(\varphi_1\Big) \cup prop\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}; & \\ prop\Big(\varphi_1\Big) \cup prop\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \vee \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}. & \end{cases}$$

Asigurați-vă că înțelegeți și că știți să definiți și să calculați cu astfel de funcții.

4.3 Demonstrații prin inducție structurală

Câteodată, vom face demonstrații prin *inducție structurală*. Sunteți deja familiarizați din liceu cu demonstrațiile prin inducție matematică.

Pentru a demonstra o teoremă de forma

$$\textit{Pentru orice } n \in \mathbb{N}\textit{, este adevărat că } P(n),$$

principiul inducției matematice postulează că este suficient să arătăm că:

- (cazul de bază) $P(0)$ *este adevărat*;
- (cazul inductiv) $P(k)$ *implică* $P(k+1)$, *pentru orice* $k \in \mathbb{N}$.

Demonstrațiile prin inducție structurală generalizează principiul de mai sus și pot fi aplicate oricărei mulțimi definite inductiv, cum ar fi $\mathbb{LP}$.

Pentru cazul $\mathbb{LP}$, principiul inducției structurale este că, pentru a demonstra o teoremă de forma

Pentru orice formulă propozițională $\varphi \in \mathbb{LP}$, *este adevărat că* $P\Big(\varphi\Big)$,

este suficient să arătăm că:

- (cazul de bază) $P\Big(\varphi'\Big)$ *are loc pentru orice formula atomică* $\varphi' \in A$ (altfel spus, proprietatea P este adevărată pentru orice formulă atomică);
- (cazul inductiv 1) $P\Big(\neg\varphi'\Big)$ *are loc dacă* $P\Big(\varphi'\Big)$ *are loc*;
- (cazul inductiv 2) $P\Big((\varphi_1 \wedge \varphi_2)\Big)$ *are loc dacă* $P\Big(\varphi_1\Big)$ *și* $P\Big(\varphi_2\Big)$ *au loc*;
- (cazul inductiv 3) $P\Big((\varphi_1 \vee \varphi_2)\Big)$ *are loc dacă* $P\Big(\varphi_1\Big)$ *și* $P\Big(\varphi_2\Big)$ au loc.

Iată un exemplu de teoremă și de demonstrația ei prin inducție structurală:

Teorema 22 (Exemplu de teoremă care poate fi demonstrată prin inducție structurală)**.** *Pentru orice formulă propozițională* φ, *avem că* $height\Big(\varphi\Big) \leq size\Big(\varphi\Big)$.

Demonstrație: Facem demonstrația prin inducție structurală, unde proprietatea P este definită astfel:

$$P(\varphi) = height\Big(\varphi\Big) \leq size\Big(\varphi\Big).$$

Este suficient să arătăm că:

1. (cazul de bază) $height\Big(\varphi'\Big) \leq size\Big(\varphi'\Big)$ pentru orice variabilă propozițională $\varphi' \in A$.

Dar, prin definiție, $height\Big(\varphi'\Big) = 1$ și $size\Big(\varphi'\Big) = 1$ (deoarece $\varphi' \in A$). Și $1 \leq 1$, deci am terminat de demonstrat acest caz.

2. (cazul inductiv 1) Presupunând că $height\Big(\varphi'\Big) \leq size\Big(\varphi'\Big)$, arătăm că $height\Big(\neg\varphi'\Big) \leq size\Big(\neg\varphi'\Big)$.

Dar, din definiție, $height\Big(\neg\varphi'\Big) = 1 + height\Big(\varphi'\Big)$ și $size\Big(\neg\varphi'\Big) = 1 + size\Big(\varphi'\Big)$. Și $1 + height\Big(\varphi'\Big) \leq 1 + size\Big(\varphi'\Big)$ (ceea ce trebuia să arătăm), din moment ce știm deja că $height\Big(\varphi'\Big) \leq size\Big(\varphi'\Big)$.

3. (cazul inductiv 2) Presupunând că $height\Big(\varphi_1\Big) \leq size\Big(\varphi_1\Big)$ și $height\Big(\varphi_2\Big) \leq size\Big(\varphi_2\Big)$, arătăm că $height\Big((\varphi_1 \wedge \varphi_2)\Big) \leq size\Big((\varphi_1 \wedge \varphi_2)\Big)$.

Dar, prin definiție, $height\Big((\varphi_1 \wedge \varphi_2)\Big) = 1+max\Big(height\Big(\varphi_1\Big), height\Big(\varphi_2\Big)\Big)$ și, din moment ce $max(a, b) \leq a + b$ (pentru orice numere naturale $a, b \in \mathbb{N}$), avem că $height\Big((\varphi_1 \wedge \varphi_2)\Big) \leq 1+height\Big(\varphi_1\Big)+height\Big(\varphi_2\Big)$. Dar $height\Big(\varphi_i\Big) \leq size\Big(\varphi_i\Big)$ $(1 \leq i \leq 2)$ din ipoteză, și deci $height\Big((\varphi_1 \wedge \varphi_2)\Big) \leq 1 + size\Big(\varphi_1\Big) + size\Big(\varphi_2\Big)$. Dar, din definiție, $size\Big((\varphi_1 \wedge \varphi_2)\Big) = 1 + size\Big(\varphi_1\Big) + size\Big(\varphi_2\Big)$, și deci $height\Big((\varphi_1 \wedge \varphi_2)\Big) \leq size\Big((\varphi_1 \wedge \varphi_2)\Big)$, ceea ce trebuia să arătăm.

4. (cazul inductiv 3) analog cazului inductiv 2.

q.e.d.

4.4 Fișă de exerciții

Exercițiul 23. *Calculați, folosind funcția subf, mulțimea de subformule ale formulelor:*
1. $((\mathtt{p} \wedge \neg\mathtt{q}) \wedge \mathtt{r})$*;* *2.* $((\mathtt{p} \vee \neg\mathtt{q}) \wedge \mathtt{r})$*;* *3.* $\neg((\mathtt{p} \vee \neg\mathtt{q}) \wedge \mathtt{r})$*.*

Exercițiul 24. *Calculați arborii abstracți ai următoarelor formule:*

1. $((\mathtt{p} \wedge \neg\mathtt{q}) \wedge \mathtt{r})$*;*
2. $((\mathtt{p} \vee \neg\mathtt{q}) \wedge \mathtt{r})$*;*
3. $\neg((\mathtt{p} \vee \neg\mathtt{q}) \wedge \mathtt{r})$*;*
4. $(\neg(\mathtt{p} \vee \neg\mathtt{q}) \wedge \mathtt{r})$*.*

Exercițiul 25. *Calculați, utilizând funcția height* : $\mathbb{LP} \to \mathbb{N}$*, înălțimea arborelui abstract de sintaxă pentru fiecare dintre formulele de la Exercițiul 24.*

Exercițiul 26. *Calculați, utilizând funcția size* : $\mathbb{LP} \to \mathbb{N}$*, numărul de noduri al arborelui abstract de sintaxă pentru fiecare dintre formulele de la Exercițiul 24.*

Exercițiul 27. *Calculați, utilizând funcția prop* : $\mathbb{LP} \to 2^A$*, mulțimea variabilelor propoziționale care apar în fiecare dintre formulele de la Exercițiul 24.*

Exercițiul 28. *Demonstrați utilizând principiul inducției structurale că pentru orice formulă propozițională* φ*, avem că* $height\Big(\varphi\Big) < size\Big(\varphi\Big) + 1$.

Capitolul 5

Semantica logicii propoziționale

5.1 Algebra booleană

Mulțimea $B = \{0, 1\}$ se numește mulțimea valorilor booleene (sau mulțimea valorilor de adevăr). Valorea 0 reprezintă falsul și valoarea 1 reprezintă adevărul.

Funcția $\bar{}\ : B \to B$ se numește *negație logică* și este definită astfel: $\overline{0} = 1$ și $\overline{1} = 0$.

Funcția $+ : B \times B \to B$ se numește *disjuncție logică* și este definită astfel: $0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 1$.

Funcția $\cdot : B \times B \to B$ se numește *conjuncție logică* și este definită astfel: $0 \cdot 0 = 0, 0 \cdot 1 = 0, 1 \cdot 0 = 0, 1 \cdot 1 = 1$.

Tuplul $(B, +, \cdot, \bar{})$ formează o *algebră booleană*.

5.2 Atribuiri

O *atribuire de valori de adevăr* (sau pur și simplu *atribuire* de aici înainte) este orice funcție $\tau : A \to B$. Cu alte cuvinte, o atribuire este o funcție care asociază fiecărei variabile propoziționale o valoare de adevăr.

Exemplul 29. *Fie $\tau_1 : A \to B$ o funcție definită după cum urmează:*

1. $\tau_1(\mathtt{p}) = 1$;
2. $\tau_1(\mathtt{q}) = 0$;

3. $\tau_1(\mathtt{r}) = 1$;
4. $\tau_1(a) = 0$ *pentru orice* $a \in A \setminus \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$.

Din moment ce este o funcție de la A *la* B, τ_1 *este o atribuire de valori de adevăr.*

Exemplul 30. *Fie* $\tau_2 : A \to B$ *o funcție definită după cum urmează:*

1. $\tau_2(\mathtt{p}) = 0$;
2. $\tau_2(\mathtt{q}) = 0$;
3. $\tau_2(\mathtt{r}) = 1$;
4. $\tau_2(a) = 1$ *pentru orice* $a \in A \setminus \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$.

Din moment ce este o funcție de la A *la* B, τ_2 *este de asemenea o atribuire.*

Exemplul 31. *Fie* $\tau_3 : A \to B$ *o funcție definită după cum urmează:*

1. $\tau_3(a) = 0$ *pentru orice* $a \in A$.

Din moment ce este o funcție de la A *la* B, τ_3 *este o atribuire.*

5.3 Valoarea de adevăr a unei formule într-o atribuire

Valoarea de adevăr a unei formule φ într-o atribuire τ este notată cu $\hat{\tau}\Big(\varphi\Big)$ și este definită astfel:

$$\hat{\tau}\Big(\varphi\Big) = \begin{cases} \tau(\varphi), & \text{dacă } \varphi \in A; \\ \overline{\hat{\tau}\Big(\varphi'\Big)}, & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in \mathbb{LP}; \\ \hat{\tau}\Big(\varphi_1\Big) \cdot \hat{\tau}\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}; & \\ \hat{\tau}\Big(\varphi_1\Big) + \hat{\tau}\Big(\varphi_2\Big), & \\ \qquad \text{dacă } \varphi = (\varphi_1 \vee \varphi_2) \text{ și } \varphi_1, \varphi_2 \in \mathbb{LP}. & \end{cases}$$

De fapt, funcția $\hat{\tau} : \mathbb{LP} \to B$ se numește *extensia homomorfică* a atribuirii $\tau : A \to B$ la mulțimea de formule $\mathbb{LP}$.

Un exemplu de calcul a valorii de adevăr a formulei $(p \vee q)$ în atribuirea τ_1 este prezentat mai jos:

$\hat{\tau}_1\Big((p \vee q)\Big) = \hat{\tau}_1\Big(p\Big) + \hat{\tau}_1\Big(q\Big) = \tau_1(p) + \tau_1(q) = 1 + 0 = 1.$

Concluzionăm că valoarea de adevăr a formulei $(p \vee q)$ în τ_1 este 1.

Iată un alt exemplu, în care calculăm valoarea de adevăr a formulei $\neg(p \wedge q)$ în atribuirea τ_1:

$\hat{\tau}_1\Big(\neg(p \wedge q)\Big) = \overline{\hat{\tau}_1\Big((p \wedge q)\Big)} = \overline{\hat{\tau}_1\Big(p\Big) \cdot \hat{\tau}_1\Big(q\Big)} = \overline{\tau_1(p) \cdot \tau_1(q)} = \overline{1 \cdot 0} = \overline{0} = 1.$

Concluzionăm că valoarea de adevăr a formulei $\neg(p \wedge q)$ în τ_1 este 1.

Iată încă un exemplu, în care calculăm valorea de adevăr a formulei $\neg\neg q$ în atribuirea de valori de adevăr τ_2:

$\hat{\tau}_2\Big(\neg\neg q\Big) = \overline{\hat{\tau}_2\Big(\neg q\Big)} = \overline{\overline{\hat{\tau}_2\Big(q\Big)}} = \overline{\overline{\tau_2(q)}} = \overline{\overline{0}} = \overline{1} = 0.$

Așadar, valoarea de adevăr a formulei $\neg\neg q$ în τ_2 este 0.

Observație. • *În general, nu are sens exprimarea* valoarea de adevăr a unei formule*, deoarece o formulă poate fi adevărată într-o anumită atribuire, dar falsă în altă atribuire. Are sens să spunem* valoarea de adevăr a unei formule într-o atribuire.

• *Din același motiv, nu are sens să spunem* formula este adevărată *sau* formula este falsă*. În schimb, are sens să spunem* formula este adevărată în această atribuire *sau* formula este falsă în această atribuire*. De exemplu, formula $\neg\neg p$ este adevărată în τ_1, dar falsă în τ_2.*

• *Dacă spunem doar* valoarea de adevăr a formulei φ*, înțelegem* o funcție de la mulțimea tuturor atribuirilor la mulțimea B *și nu* o valoare de adevăr din mulțimea B*.*

> În general, nu are sens exprimarea *valoarea de adevăr a unei formule*, deoarece o formulă poate fi adevărată într-o anumită atribuire, dar falsă în altă atribuire. Are sens să spunem *valoarea de adevăr a unei formule într-o atribuire.*

Definiția 32 (Noțiunea de satisfacere)**.** *O* atribuire τ *satisface* φ *dacă* $\hat{\tau}\Big(\varphi\Big) = 1$.

În loc de τ *satisface* φ, putem folosi oricare dintre următoarele forme echivalente:

1. τ este model al formulei φ;
2. φ ține în τ;
3. τ face φ adevărată;

Exemplul 33. *Atribuirea τ_1 definită mai sus este model pentru $\neg(\mathtt{p} \wedge \mathtt{q})$. Atribuirea τ_1 nu este model al formulei $(\neg\mathtt{p} \wedge \mathtt{q})$.*

Scriem $\tau \models \varphi$ (și citim: τ este model al formulei φ; sau: τ satisface φ etc.) ddacă $\hat{\tau}\Big(\varphi\Big) = 1$. Scriem $\tau \not\models \varphi$ (și citim: τ nu este model al formulei φ; sau: τ nu satisface φ) ddacă $\hat{\tau}\Big(\varphi\Big) = 0$.

Definiția 34 (Relația de satisfacere (notată $\models$))**.** *Relația $\models$, dintre atribuiri și formule, se numește* relația de satisfacere. *Relația este definită astfel: $\tau \models \varphi$ ddacă $\hat{\tau}\Big(\varphi\Big) = 1$.*

> Relația $\models$ este definită astfel: $\tau \models \varphi$ ddacă $\hat{\tau}\Big(\varphi\Big) = 1$.

5.4 Satisfiabilitate

Definiția 35. *O formulă φ este* satisfiabilă *dacă există cel puțin o atribuire τ astfel încât $\tau \models \varphi$ (i.e., dacă φ are cel puțin un model).*

Exemplul 36. *Formula $(\mathtt{p} \vee \mathtt{q})$ este satisfiabilă, din moment ce are un model (de exemplu, atribuirea τ_1, definită mai sus).*

Exemplul 37. *Formula $\neg\mathtt{p}$ este de asemenea satisfiabilă: de exemplu, atribuirea τ_3, definită mai sus, face formula adevărată.*

Exemplul 38. *Formula $(\mathtt{p} \wedge \neg\mathtt{p})$ nu este satisfiabilă, deoarece are valoarea de adevăr* fals *în orice atribuire.*

Demonstrație: *Considerăm o atribuire $\tau : A \to B$ oarecare.*
Avem că $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \hat{\tau}\Big(\mathtt{p}\Big) \cdot \hat{\tau}\Big(\neg\mathtt{p}\Big) = \tau(\mathtt{p}) \cdot \overline{\hat{\tau}\Big(\mathtt{p}\Big)} = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})}$.
Dar $\tau(\mathtt{p})$ poate fi 0 sau 1:

1. *în primul caz ($\tau(\mathtt{p}) = 0$), avem că $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \ldots = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})} = 0 \cdot \overline{0} = 0 \cdot 1 = 0$;*

2. *în al doilea caz* ($\tau(\mathtt{p}) = 1$), *avem că* $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = \ldots = \tau(\mathtt{p}) \cdot \overline{\tau(\mathtt{p})} = 1 \cdot \overline{1} = 1 \cdot 0 = 0$.

Observăm că în oricare dintre cele două cazuri, avem că $\hat{\tau}\Big((\mathtt{p} \wedge \neg\mathtt{p})\Big) = 0$. *Dar* τ *a fost o atribuire aleasă arbitrar, și din acest motiv* $(\mathtt{p} \wedge \neg\mathtt{p})$ *este falsă* în orice atribuire τ. *Adică formula este nesatisfiabilă.* *q.e.d.*

Definiția 39 (Contradicție)**.** *O formulă care este nesatisfiabilă se numește* contradicție.

Exemplul 40. *După cum am văzut mai sus,* $(\mathtt{p} \wedge \neg\mathtt{p})$ *este o contradicție.*

5.5 Formule valide

Definiția 41 (Formulă validă)**.** *O formulă* φ *este* validă *dacă orice atribuire* τ *are proprietatea că* $\tau \models \varphi$ *(orice atribuire este model pentru formulă).*

Definiția 42 (Tautologie)**.** *O formulă validă se mai numește și* tautologie.

Exemplul 43. *Formula* $(\mathtt{p} \vee \neg\mathtt{p})$ *este validă, deoarece este adevărată în orice atribuire: fie* τ *o atribuire oarecare; avem că* $\hat{\tau}\Big((\mathtt{p} \vee \neg\mathtt{p})\Big) = \tau(\mathtt{p}) + \overline{\tau(\mathtt{p})}$. *Deoarece* $\tau(\mathtt{p})$ *poate lua doar 2 valori posibile, adică 0 sau 1, vom avea că* $\tau(\mathtt{p}) + \overline{\tau(\mathtt{p})}$ *este sau* $0 + 1$, *sau* $1 + 0$, *deci* 1 *în ambele cazuri.*

Notație. *Faptul că formula* φ *este validă se mai notează cu* $\models \varphi$.

Exemplul 44. *Formula* $\mathtt{p}$ *nu este validă (deoarece există o atribuire, de exemplu* τ_3, *care face formula falsă).*

5.6 Formule satisfiabile, dar care nu sunt valide

Observație. *În engleză, o formulă care nu este nici contradicție și nici tautologie se numește* contingent formula. *În limba română, nu avem un cuvânt precis pentru acest concept, și vom folosi exprimarea* formulă contingentă *(sau* contingență*), supraîncărcând înțelesul cuvântului contingent din DEX.*

Definiția 45. *O formulă care nu este nici contradicție și nici tautologie se numește* contingență.

Fiecare formulă poate fi clasificată ca fiind o contradicție, o tautologie, sau o contingență.

Exemplul 46. *Iată exemple din fiecare astfel de clasă de formule:*

1. $(\mathtt{p} \wedge \neg \mathtt{p})$ *este o contradicție;*
2. $\mathtt{p}$ *este o contingență;*
3. $(\mathtt{p} \vee \neg \mathtt{p})$ *este o tautologie.*

5.7 Formule echivalente

Definiția 47. *Spunem că două formule* $\varphi_1, \varphi_2 \in \mathbb{LP}$ *sunt* echivalente, *și scriem* $\varphi_1 \equiv \varphi_2$, *dacă pentru orice atribuire* $\tau : A \to B$, $\hat{\tau}\left(\varphi_1\right) = \hat{\tau}\left(\varphi_2\right)$.

Intuitiv, formulele care sunt echivalente au același înțeles (adică exprimă același lucru).

Exemplul 48. *La începutul cursului, am văzut că formulele* $\mathtt{p}$ *și* $\neg\neg\mathtt{p}$ *nu sunt egale din punct de vedere sintactic. Evident că formulele nu sunt egale, deoarece prima are* 1 *simbol, iar cealaltă* 3 *simboluri (dacă ar fi fost egale, ar fi avut același număr de simboluri). Totuși, acum suntem pregătiți să înțelegem relația dintre ele:* $\mathtt{p} \equiv \neg\neg\mathtt{p}$. *Cu alte cuvinte, chiar dacă nu sunt egale, ele sunt echivalente: exprimă același lucru.*
Pentru a demonstra $\mathtt{p} \equiv \neg\neg\mathtt{p}$, *este suficient să arătăm că formulele au aceeași valoare de adevăr în orice atribuire. Fie* τ *o atribuire oarecare. Avem că* $\hat{\tau}\left(\neg\neg\mathtt{p}\right) = \overline{\overline{\tau(\mathtt{p})}} = \tau(\mathtt{p}) = \hat{\tau}\left(\mathtt{p}\right)$. *Pe scurt,* $\hat{\tau}\left(\neg\neg\mathtt{p}\right) = \hat{\tau}\left(\mathtt{p}\right)$. *Din moment ce* τ *a fost aleasă arbitrar, înseamnă că* $\hat{\tau}\left(\neg\neg\mathtt{p}\right) = \hat{\tau}\left(\mathtt{p}\right)$ *pentru* orice atribuire τ *și de acest motiv* $\mathtt{p}$ *este echivalent cu* $\neg\neg\mathtt{p}$.

Exemplul 49. *Următoarea echivalență are loc:* $(\mathtt{p} \vee \mathtt{q}) \equiv \neg(\neg\mathtt{p} \wedge \neg\mathtt{q})$ *(exercițiu: verificați că într-adevăr așa este).*

Iată încă două echivalențe importante, cunoscute sub denumirea de *legile lui De Morgan*:

Teorema 50. *Pentru orice formule* $\varphi_1, \varphi_2 \in \mathbb{LP}$, *avem că:*

1. $\neg(\varphi_1 \vee \varphi_2) \equiv (\neg\varphi_1 \wedge \neg\varphi_2)$;
2. $\neg(\varphi_1 \wedge \varphi_2) \equiv (\neg\varphi_1 \vee \neg\varphi_2)$.

5.8 Consecință semantică

Definiția 51. *Fie* $\Gamma = \{\varphi_1, \ldots, \varphi_n, \ldots\}$ *o mulțime (posibil infinită) de formule. Spunem că* φ *este o* consecință semantică *a mulțimii* Γ, *și scriem* $\Gamma \models \varphi$, *dacă orice atribuire care este model pentru toate formulele din* Γ *este model și pentru formula* φ.

Spunem de asemenea că φ *este o consecință logică a mulțimii* Γ sau că φ *este o consecință tautologică a mulțimii* Γ în loc de φ *este o consecință semantică a mulțimii* Γ.

Exemplul 52. *Fie* $\Gamma = \{\mathtt{p}, (\neg\mathtt{p} \vee \mathtt{q})\}$. *Avem că* $\Gamma \models \mathtt{q}$.

Într-adevăr, fie τ *un model al formulelor* $\mathtt{p}$ *și* $(\neg\mathtt{p} \vee \mathtt{q})$. *Din moment ce* τ *este model pentru* $\mathtt{p}$, *avem prin definiție că* $\tau(\mathtt{p}) = 1$.

Din moment ce τ *este model al* $(\neg\mathtt{p} \vee \mathtt{q})$, *avem că* $\hat{\tau}\Big((\neg\mathtt{p} \vee \mathtt{q})\Big) = 1$. *Dar* $\hat{\tau}\Big((\neg\mathtt{p} \vee \mathtt{q})\Big) = \overline{\tau(\mathtt{p})} + \tau(\mathtt{q})$. *Dar* $\tau(\mathtt{p}) = 1$, *și deci* $\hat{\tau}\Big((\neg\mathtt{p} \vee \mathtt{q})\Big) = 0 + \tau(\mathtt{q}) = \tau(\mathtt{q})$. *Înseamnă că* $\tau(\mathtt{q}) = 1$.

Deci τ *este model pentru* $\mathtt{q}$. *Am presupus că* τ *este model pentru* $\mathtt{p}$ *și* $(\neg\mathtt{p} \vee \mathtt{q})$ *și am arătat că în mod necesar* τ *este și model pentru* $\mathtt{q}$. *Dar aceasta este fix definiția faptului că* $\{\mathtt{p}, (\neg\mathtt{p} \vee \mathtt{q})\} \models \mathtt{q}$, *ceea ce voiam să arătăm.*

Exemplul 53. *Avem că* $\{\mathtt{p}, (\mathtt{p} \vee \mathtt{r})\} \not\models \neg\mathtt{r}$, *adică* $\neg\mathtt{r}$ *nu este consecință logică a formulelor* $\mathtt{p}, (\mathtt{p} \vee \mathtt{r})$. *Pentru a arăta această* neconsecință, *este suficient să găsim un model pentru formulele* $\mathtt{p}$ *și* $(\mathtt{p} \vee \mathtt{r})$, *dar care să nu fie model al formulei* $\neg\mathtt{r}$. *Orice atribuire* τ *cu* $\tau(\mathtt{p}) = 1$ *și* $\tau(\mathtt{r}) = 1$ *(de exemplu,* τ_1*) satisface cele două proprietăți.*

Notație. *Câteodată scriem*

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

în loc de

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

Observație. *Dacă* $n = 0$, $\varphi_1, \ldots, \varphi_n \models \varphi$ *se reduce la faptul că* φ *este consecință semantică a mulțimii vide. Acest lucru înseamnă că* φ *este validă, ceea ce justifică notația* $\models \varphi$ *pentru faptul că* φ *este validă.*

5.9 Mulțimi consistente de formule

O altă noțiune semantică care apare relativ des în practică și care are o legatură strânsă cu conectorul $\wedge$ este noțiunea de *mulțime (in)consistentă de formule.*

Definiția 54 (Mulțime consistentă de formule)**.** *O mulțime* $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ de formule se numește consistentă *dacă există o atribuire* τ *care să fie model pentru toate formulele* φ_i $(1 \leq i \leq n)$.

Observație. *Atenție! Aceeași atribuire* τ *pentru toate formulele din mulțime (nu câte o atribuire potențial diferită pentru fiecare formulă)!*

O mulțime de formule este inconsistentă (sau neconsistentă) dacă nu este consistentă.

Lema 55 (Legătura dintre mulțimile consistente și conectorul logic $\wedge$)**.** *O mulțime de formule $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ este consistentă ddacă formula*

$$(((\varphi_1 \wedge \varphi_2) \wedge \ldots) \wedge \varphi_n)$$

este satisfiabilă.

Lema 56 (Legătura dintre mulțimile inconsistente și conectorul logic $\wedge$)**.** *O mulțime de formule $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ este inconsistentă ddacă formula*

$$(((\varphi_1 \wedge \varphi_2) \wedge \ldots) \wedge \varphi_n)$$

nu este satisfiabilă.

Exercițiul 57. *Arătați că, pentru orice formulă φ, dacă $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ este o mulțime inconsistentă de formule, avem că*

$$\{\varphi_1, \varphi_2, \ldots, \varphi_n\} \models \varphi.$$

Exercițiul 58. *Arătați că dacă*

$$\{\varphi_1, \varphi_2, \ldots, \varphi_n\} \models (\mathtt{p} \wedge \neg \mathtt{p}),$$

atunci $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ este o mulțime inconsistentă de formule.

5.10 Aplicația 1

Ion scrie următorul cod:

```
if (((y % 4 == 0) && (y % 100 != 0)) || (y % 400 == 0))
   printf("%d is a leap year.", y);
else
   printf("%d is not a leap year.", y);
```

Ioana încearcă să simplifice codul în felul următor:

```
if (((y % 4 != 0) || (y % 100 == 0)) && (y % 400 != 0))
   printf("%d is not a leap year.", y);
else
   printf("%d is a leap year.", y);
```

Transformarea făcută de Ioana păstrează comportamentul programului? E dificil să spunem cu certitudine acest lucru dacă doar ne uităm la cod, dar putem să ne folosim de conceptele pe care le-am învățat pentru a modela problema de mai sus folosind uneltele logicii propoziționale și să determinăm dacă cele două programe au același comportament.

Înainte de toate, vom *traduce* condițiile din instrucțiunea if-else în logica propozițională. Vom identifica *propozițiile atomice* și le vom înlocui cu variabile propoziționale după cum urmează. Fie `y` un an fixat:

1. variabila propozițională $\mathtt{p}$ va ține locul propoziției atomice `(y % 4 == 0)`;
2. variabila propozițională $\mathtt{q}$ va ține locul propoziției atomice `(y % 100 == 0)`;
3. variabila propozițională $\mathtt{r}$ va ține locul propoziției atomice `(y % 400 == 0)`.

Ținând cont de traducerea de mai sus, vedem că condiția din programul lui Ion este, în limbajul logicii propoziționale, $((\mathtt{p} \wedge \neg \mathtt{q}) \vee \mathtt{r})$.

Formula Ioanei este, în limbajul logicii propoziționale, $((\neg \mathtt{p} \vee \mathtt{q}) \wedge \neg \mathtt{r})$.

Observați de asemenea că ramura `if` a primului program corespunde ramurii `else` a celui de-al doilea program și invers. Pentru ca cele două programe să aibă același comportament, este suficient ca negația formulei lui Ion să fie echivalentă cu formula Ioanei. Cu alte cuvinte, are loc echivalența:

$$\neg((\mathtt{p} \wedge \neg \mathtt{q}) \vee \mathtt{r}) \equiv ((\neg \mathtt{p} \vee \mathtt{q}) \wedge \neg \mathtt{r})?$$

Aplicând legile lui De Morgan, vedem că echivalența are într-adevăr loc și deci transformarea propusă de Ioana este corectă.

5.11 Fișă de exerciții

Exercițiul 59. *Fie $\tau : A \to B$ atribuirea definită după cum urmează: $\tau(\mathtt{p}) = 1$, $\tau(\mathtt{q}) = 0$, $\tau(\mathtt{r}) = 0$, $\tau(a) = 0$ pentru orice altă variabilă propozițională $a \in A \setminus \{\mathtt{p}, \mathtt{q}, \mathtt{r}\}$.*

Stabiliți valoarea de adevăr în atribuirea τ de mai sus a următoarelor formule:

1. $(\mathtt{p} \wedge \mathtt{q})$;
2. $(\mathtt{q} \wedge \mathtt{p})$;
3. $\neg \mathtt{q}$;

4. $(\neg q \wedge r)$;
5. $((\neg q \wedge r) \vee \neg p)$.

Exercițiul 60. *Găsiți câte o atribuire τ în care următoarele formule să fie adevărate (câte o atribuire pentru fiecare formulă):*

1. $(p \wedge q)$;
2. $(p \wedge \neg q)$;
3. $((p \wedge \neg q) \vee q)$.

Există o (singură) atribuire care să facă toate cele trei formule de mai sus adevărate?

Exercițiul 61. *Găsiți câte o atribuire τ în care următoarele formule să fie false (câte o atribuire pentru fiecare formulă):*

1. $(p \vee q)$;
2. $(q \wedge (p \vee \neg q))$;
3. $((p \wedge \neg q) \vee q)$.

Exercițiul 62. *Care dintre următoarele formule sunt satisfiabile?*

1. $(p \wedge \neg p)$;
2. $(p \vee \neg p)$;
3. $((p \vee \neg p) \wedge \neg q)$;
4. $((p \vee \neg p) \wedge (\neg p \wedge q))$;
5. $((p \vee \neg q) \wedge (\neg p \vee r))$.

Exercițiul 63. *Care dintre următoarele formule sunt valide?*

1. $(p \wedge \neg p)$;
2. $(p \vee \neg p)$;
3. p;
4. $((p \vee \neg p) \wedge \neg q)$;
5. $((p \wedge q) \vee (\neg p \wedge r))$.

Exercițiul 64. *Dați exemple de 5 contradicții interesante.*

Exercițiul 65. *Dați exemple de 5 tautologii interesante.*

Exercițiul 66. *Demonstrați că, pentru orice formule $\varphi_1, \varphi_2, \varphi_3 \in \mathbb{LP}$, au loc următoarele echivalențe:*

1. $(\varphi_1 \wedge (\varphi_2 \wedge \varphi_3)) \equiv ((\varphi_1 \wedge \varphi_2) \wedge \varphi_3)$;
2. $(\varphi_1 \wedge \varphi_2) \equiv (\varphi_2 \wedge \varphi_1)$;
3. $(\varphi_1 \vee (\varphi_2 \vee \varphi_3)) \equiv ((\varphi_1 \vee \varphi_2) \vee \varphi_3)$;
4. $(\varphi_1 \vee \varphi_2) \equiv (\varphi_2 \vee \varphi_1)$;
5. $\neg\neg\varphi_1 \equiv \varphi_1$;
6. $\neg(\varphi_1 \wedge \varphi_2) \equiv (\neg\varphi_1 \vee \neg\varphi_2)$;
7. $\neg(\varphi_1 \vee \varphi_2) \equiv (\neg\varphi_1 \wedge \neg\varphi_2)$.

Exercițiul 67. *Putem demonstra că, pentru orice formule $\varphi_1, \varphi_2 \in \mathbb{LP}$, $(\varphi_1 \vee \varphi_2) \equiv \varphi_1$ dacă și numai dacă $\varphi_1 \in \mathbb{LP}$ este tautologie?*

Exercițiul 68. *Putem demonstra că, pentru orice formule $\varphi_1, \varphi_2 \in \mathbb{LP}$, $(\varphi_1 \wedge \varphi_2) \equiv \varphi_2$ dacă și numai dacă $\varphi_1 \in \mathbb{LP}$ este tautologie?*

Exercițiul 69. *Putem demonstra că, pentru orice formule $\varphi_1, \varphi_2 \in \mathbb{LP}$, $(\varphi_1 \wedge \varphi_2) \equiv \varphi_1$ dacă și numai dacă $\varphi_1 \in \mathbb{LP}$ este contradicție?*

Exercițiul 70. *Putem demonstra că, pentru orice formule $\varphi_1, \varphi_2 \in \mathbb{LP}$, $(\varphi_1 \vee \varphi_2) \equiv \varphi_2$ dacă și numai dacă $\varphi_1 \in \mathbb{LP}$ este contradicție?*

Exercițiul 71. *Arătați că $\neg\mathtt{p}$ este consecință semantică din $(\neg\mathtt{p} \vee \neg\mathtt{p})$.*

Exercițiul 72. *Arătați că $\mathtt{p}$ nu este consecință semantică din $(\neg\mathtt{q} \vee \mathtt{p})$.*

Exercițiul 73. *Arătați că $\mathtt{p}$ este consecință semantică din $(\neg\mathtt{q} \vee \mathtt{p})$ și $\mathtt{q}$.*

Exercițiul 74. *Arătați că $\mathtt{p}_3$ este consecință semantică din $(\neg\mathtt{p}_1 \vee (\mathtt{p}_2 \vee \mathtt{p}_3))$, $((\neg\neg\mathtt{p}_2 \vee \neg\mathtt{p}_4) \wedge (\neg\neg\mathtt{p}_4 \vee \neg\mathtt{p}_2))$ și $(\mathtt{p}_1 \wedge \neg\mathtt{p}_4)$.*