Logică pentru Informatică - Săptămâna 2
Sintaxa Logicii Propoziționale

1 Sintaxa logicii propoziționale

Reamintim sintaxa logicii propoziționale (definiția mulțimii *LP*).

Formulele propoziționale sunt cuvinte (șiruri de simboluri) peste alfabetul logicii propoziționale.

Alfabetul logicii propoziționale este reuniunea următoarelor mulțime de simboluri:

1. $A = \{\mathtt{p}, \mathtt{q}, \mathtt{r}, \mathtt{p}', \mathtt{q}_1, \ldots\}$ este o mulțime infinit numărabilă de *variabile propoziționale* pe care o fixăm de la început;
2. $\{\neg, \wedge, \vee\}$ este mulțimea de *conectori logici*;
3. $\{(,)\}$ este mulțimea de simboluri auxiliare.

Așadar, mulțimea $A \cup \{\neg, \wedge, \vee, (,)\}$ este numită *alfabetul logicii propoziționale*. În informatică, o mulțime se numește alfabet dacă îi folosim elementele pentru a construi cuvinte.

Iată câteva exemple de cuvinte peste alfabetul logicii propoziționale: $))\mathtt{p}\vee\wedge$, $\vee\vee\neg(\mathtt{p})$, $\neg(\mathtt{p}\vee\mathtt{q})$. Cuvintele, sau șirurile de simboluri, sunt secvențe de simboluri peste alfabet. Unele dintre aceste cuvinte se numesc formule.

De exemplu, ultimul cuvânt de mai sus, $\neg(\mathtt{p}\vee\mathtt{q})$, este o formulă, dar cuvântul $\vee\vee\neg(\mathtt{p})$ nu este o formulă. Următoarea definiție stabilește cu precizie ce cuvinte sunt formule și ce cuvinte nu sunt formule.

Definiția 1.1 (Mulțimea formulelor propoziționale (*LP*))**.** *Mulțimea formulelor propoziționale, notată LP de aici înainte, este cea mai mică mulțime de cuvinte care satisface următoarele proprietăți:*

1. *(Pasul de Bază) Orice variabilă propozițională (văzută ca un cuvânt cu* 1 *simbol) este în LP (echivalent, $A \subseteq LP$);*
2. *(Pasul Inductiv 1) Dacă $\varphi \in LP$, atunci $\neg\varphi \in LP$ (echivalent, dacă cuvântul φ este o formulă propozițională, atunci și cuvântul care începe cu simbolul $\neg$ și continuă cu simbolurile cuvântului φ este o formulă);*

3. *(Pasul Inductiv 2) Dacă* $\varphi_1, \varphi_2 \in LP$*, atunci* $(\varphi_1 \vee \varphi_2) \in LP$ *(echivalent, exercițiu);*

4. *(Pasul Inductiv 3) Dacă* $\varphi_1, \varphi_2 \in LP$*, atunci* $(\varphi_1 \wedge \varphi_2) \in LP$ *(echivalent, exercițiu).*

Iată exemple de elemente ale mulțimii *LP*:

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

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

Iată exemple de cuvinte care nu sunt în *LP*:

$$\mathtt{pp} \qquad \mathtt{q}\neg\mathtt{q} \qquad \mathtt{q} \wedge \neg\mathtt{p} \qquad \mathtt{p} + \mathtt{q}$$

Definiția mulțimii *LP* este un exemplu de *definiție inductivă*. Astfel de definiții sunt foarte importante în informatică și este obligatoriu să le înțelegem în profunzime. Într-o definiție inductivă a unei mulțimi, avem cazul de bază, care ne precizează un număr de elemente care fac parte din mulțime (elementele "de bază"). De asemenea, într-o definiție inductivă, avem unul sau mai multe cazuri inductive, care explică cum se pot obține elemente "noi" ale mulțimii pornind de la alte elemente "vechi" ale mulțimii. O altă caracteristică importantă a unei definiții prin inducție este constrângerea de minimalitate, care spune că *doar* elementele care sunt cerute de cazul de bază și de cazurile inductive sunt în mulțime, și niciun altul. Se poate arăta că mulțimea *LP* definită mai sus există și este unică, dar demonstrația nu face parte din acest curs.

1.1 Cum arătăm că un cuvânt face parte din *LP*

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

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

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

$$\dfrac{\dfrac{\dfrac{}{\mathtt{p} \in LP}\ \textit{pas de bază} \qquad \dfrac{}{\mathtt{q} \in LP}\ \textit{pas de bază}}{(\mathtt{p} \vee \mathtt{q}) \in LP}\ \textit{pas inductiv 2}}{\neg(\mathtt{p} \vee \mathtt{q}) \in LP}\ \textit{pas inductiv 1}$$

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 poartă numele de *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 scăpăm de adnotații, obținem următorul *arbore de construcție* pentru formula:

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

Este ușor de văzut ca un cuvânt aparține LP ddacă există un arbore de construcție pentru acel cuvânt.

1.2 Conectorul principal al unei formule

O formulă care constă doar dintr-o variabilă propozițională (cum ar fi formulele $\mathtt{p}$ sau $\mathtt{q}$) 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 $\neg\mathtt{p}$ sau $\mathtt{p}\vee\mathtt{q}$, se numesc *compuse* (sau, pe 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 $\neg(\mathtt{p}\vee\mathtt{q})$ este $\neg$ (negația), în timp ce conectorul principal al formulei $(\neg\mathtt{p}\vee\mathtt{q})$ 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*.

1.3 Cum arătăm că un cuvânt nu face parte din *LP*

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

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

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

Să presupunem, prin reducere la absurd, că $(\mathtt{p}\neg\mathtt{q})\in LP$. În acest caz, prin condiția de minimalitate, avem că $(\mathtt{p}\neg\mathtt{q})\in LP$ 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 $(\mathtt{p}\neg\mathtt{q}) \notin LP$, deoarece $(\mathtt{p}\neg\mathtt{q}) \notin A$.

De asemenea, nu putem aplica nici cazul inductiv 1, deoarece nu există nicio formulă φ astfel încât $(\mathtt{p}\neg\mathtt{q}) = \neg\varphi$ (orice formulă $\varphi \in LP$ 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 $\varphi_1, \varphi_2 \in LP$ astfel încât $(\mathtt{p}\neg\mathtt{q}) = (\varphi_1 \vee \varphi_2)$ (oricum am alege $\varphi_1, \varphi_2 \in LP$, cuvântul din partea stângă nu conține simbolul $\vee$, 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 $(\mathtt{p}\neg\mathtt{q}) \notin LP$.

1.4 Proprietatea de citire unică

Definiția LP are o proprietate importantă, care se numește *proprietatea de citire unică*:

Teorema 1.1 (Proprietatea de citire unică a formulelor propoziționale)**.** *Pentru orice formulă $\varphi \in LP$, 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 LP și vom arată în ce sens această definiție alternativă este ambiguă.

Începutul unei definții greșite pentru *LP*.

Să ne imaginăm că pasul inductiv 2 ar fi:

⋮

3. (pasul inductiv 2) Dacă $\varphi_1, \varphi_2 \in LP$, atunci $\varphi_1 \vee \varphi_2 \in LP$.

⋮

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

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

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 *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, sugerez să 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.

2 Funcții definite inductiv peste *LP*

Reamintire 2.1. *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 această notație. 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:

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

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

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

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

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 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, mai mari,

pentru apelul de funcție, și parantezele obișnuite, scrise cu font `teletype`, 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 $\mathtt{p} \vee \mathtt{q} \notin LP$ nu este o formulă.

Funcția *subf* este prima dintre funcțiile definite recursiv, în funcție de structura formulei $\varphi \in LP$. Aceste funcții se numesc *recursive structural*. Este foarte important să înțelegem cum operează aceste funcții pentru a înțelege restul cursului. Iată alte exemple de astfel de funcții.

Următoarea funcție calculează *arborele abstract de sintaxă* al unei formule: $arb : LP \to Trees$, definită prin:

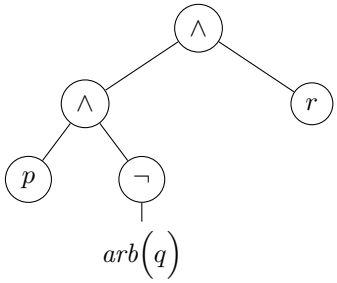
$$arb\Big(\varphi\Big) = \begin{cases} \varphi, & \text{dacă } \varphi \in A; \\ \neg \\ \quad arb\Big(\varphi'\Big) & \text{dacă } \varphi = \neg\varphi' \text{ și } \varphi' \in LP; \\ \vee \\ arb\Big(\varphi_1\Big) \quad arb\Big(\varphi_2\Big) & \text{dacă } \varphi = (\varphi_1 \wedge \varphi_2)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

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

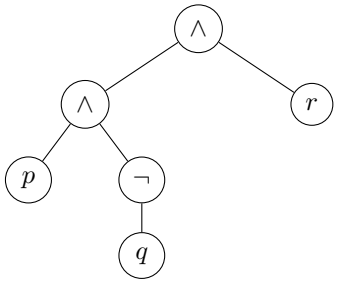
$$arb\Big(((\mathtt{p} \wedge \neg\mathtt{q}) \wedge \mathtt{r})\Big) = \begin{array}{c} \wedge \\ arb\Big((p \wedge \neg q)\Big) \quad arb\Big(r\Big) \end{array}$$

$$= \begin{array}{ccc} & \wedge & \\ & & \\ \wedge & & r \\ & & \\ arb\Big(p\Big) & arb\Big(\neg q\Big) & \end{array}$$

$=$



$=$



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

Arborii abstracți de sintaxă sunt foarte importanți. De fapt, conceptual, o formula 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.

Iată încă trei exemple de funcții definite prin recursie structurală peste formule.

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

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

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

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

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

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

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

2.1 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

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

principiul inducției matematice postulează că este suficient să arătăm că:

1. (cazul de bază) $P(0)$ este adevărat;
2. (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 LP.

Pentru cazul LP, principiul inducției structurale este că, pentru a demonstra o teoremă de forma

$$\text{Pentru orice formula propozițională } \varphi \in LP, \text{ este adevărat că } P\Big(\varphi\Big),$$

este suficient să arătăm că:

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

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

Teorema 2.1 (Exemplu de teoremă care poate fi demonstrată prin inducție structurală)**.** *Pentru orice formulă propozițională φ, avem că $height\Big(\varphi\Big) \leq size\Big(\varphi\Big)$.*

Proof. Facem demonstrația prin inducție structurală (proprietatea $P(\varphi)$ este $height\Big(\varphi\Big) \leq size\Big(\varphi\Big)$). 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 $height\Big((\varphi_1 \vee \varphi_2)\Big) \leq size\Big((\varphi_1 \vee \varphi_2)\Big)$.

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

4. (cazul inductiv 3) analog.

□

3 Semantica logicii propoziționale

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

Funcția $\_ : B \to B$ 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: $0 \cdot 0 = 0, 0 \cdot 1 = 0, 1 \cdot 0 = 1, 1 \cdot 1 = 1$.

Tuplul $(B, +, \cdot, -)$ se numește *algebră booleană*.

3.1 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 3.1. *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 3.2. *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 3.3. *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 de asemenea o atribuire.

3.2 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:

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

De fapt, funcția $\hat{\tau} : LP \to B$ se numește *extensia homomorfică* a atribuirii $\tau : A \to B$ la mulțimea de formule LP, dar nu este necesar să rețineți acest aspect.

Iată un exemplu de calcul a valorii de adevăr a formulei $(\mathtt{p} \vee \mathtt{q})$ în atribuirea τ_1:

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

Concluzionăm că valoarea de adevăr a formulei $(\mathtt{p} \vee \mathtt{q})$ în $\hat{\tau}_1$ este 1.

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

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

Concluzionăm că valoarea de adevăr a formulei $\neg(\mathtt{p} \wedge \mathtt{q})$ 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:

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

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

Observație 3.1. *Important!*

Nu are sens să spunem "valoarea de adevăr a unei formule". Are sens să spunem "valoarea de adevăr a unei formule într-o atribuire".

De asemenea, 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".

Acest lucru deoarece formula ar putea fi adevărată într-o anumită atribuire, dar falsă în altă atribuire. De exemplu, formula $\neg\neg\mathtt{p}$ este adevărată în τ_1, dar falsă în τ_2.

O *atribuire τ satisface φ* dacă $\hat{\tau}\Big(\varphi\Big) = 1$. În loc de *$\tau$ 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;

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

Exemplul 3.4. *Atribuirea τ_1 definită mai sus este model pentru $\neg(\mathtt{p} \wedge \mathtt{q})$.*

Atribuirea τ_1 nu este model al formulei $(\neg\mathtt{p} \wedge \mathtt{q})$.