

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	Motivație și introducere	5
2	Structuri și semnături	7
3	Sintaxa logicii de ordinul I	11
3.1	Alfabetul	11
3.2	Termen	12
3.3	Formule atomice	13
3.4	Formule de ordinul I	14
3.5	Paranteze	16
3.6	Modelarea în LPI a afirmațiilor din limba română	17
3.7	Modelarea în LPI a afirmațiilor despre aritmetică	18
3.8	Variabilele unei formule	19
3.9	Domeniul de vizibilitate al unui cuantificator - analogie cu limbajele de programare	21
3.10	Apariții libere și legate ale variabilelor	22
3.11	Variabile libere și variabile legate	24
3.12	Domeniul de vizibilitate și parantetizarea formulelor	25
3.13	Fișă de exerciții	26
4	Semantica formulelor logicii de ordinul I	29
4.1	Atribuire	30
4.2	Valoarea de adevăr a unei formule de ordinul I	32
4.3	Satisfiabilitate într-o structură fixată	35
4.4	Validitate într-o structură fixată	36
4.5	Satisfiabilitate	36
4.6	Validitate	36
4.7	Consecință semantică	37
4.8	Mulțime consistentă de formule	37
4.9	Formule echivalente	38
4.10	Fișă de exerciții	40

Capitolul 1

Motivație și introducere

Logica de ordinul I, pe care o vom studia în continuare, este o extensie a logicii propoziționale, extensie care aduce un plus de expresivitate. Expresivitatea adițională este necesară pentru a putea modela anumite afirmații care nu pot fi exprimate în logica propozițională.

În logica propozițională, nu putem exprima într-un mod natural următoarea afirmație: *Orice om este muritor*.

Pentru a modela o afirmație în logica propozițională, identificăm întâi propozițiile atomice. Apoi asociem fiecărei propoziții atomice o variabilă propozițională. Propozițiile atomice sunt propozițiile care nu pot fi împărțite în alte propoziții mai mici, care să fie conectate între ele prin conectorii logici \neg , \wedge , \vee , \rightarrow și respectiv \leftrightarrow .

Observăm că afirmația *Orice om este muritor* nu poate fi descompusă în afirmații indivizibile legate între ele prin conectorii logicii propoziționale, după cum este descris mai sus. Așadar, în logica propozițională, afirmația este atomică. Asociem întregii afirmații o variabilă propozițională $p \in A$.

Acum să modelăm afirmația *Socrate este om*. Evident, acestei a doua afirmații trebuie să îi asociem o altă variabilă propozițională $q \in A$. Să presupunem că știm că p și q sunt adevărate. Formal, știm că lucrăm cu o atribuire $\tau : A \rightarrow B$ astfel încât $\tau(p) = 1$ și $\tau(q) = 1$. Putem trage concluzia ca afirmația *Socrate este muritor* este adevărată în atribuirea τ ?

Nu, deoarece afirmației *Socrate este muritor* ar trebui să îi asociem o a treia variabilă propozițională r și nu putem trage nicio concluzie asupra lui $\tau(r)$ din faptul că $\tau(p) = 1$ și $\tau(q) = 1$. Deci, din semantica logicii propoziționale, nu putem trage concluzia că r este adevărată în orice atribuire în care p și q sunt adevărate, în ciuda faptului că, dacă *orice om este muritor* și *Socrate este om* atunci sigur *Socrate este muritor*. Această diferență între realitate și modelarea noastră ne indică faptul că modelarea nu este suficient

de bună.

Logica de ordinul I aduce, în plus față de logica propozițională, noțiunea de *cuantificator* (existențial sau universal) și noțiunea de *predicat*. Cuantificatorul universal este notat cu \forall (de la litera *A* întoarsă – *all* în limba engleză), iar cuantificatorul existențial este notat cu \exists (de la litera *E* întoarsă – *exists* în limba engleză).

Un predicat este o afirmație a cărei valoare de adevăr depinde de zero sau mai mulți parametri. De exemplu, pentru afirmația de mai sus, vom folosi două predicate: $\mathbf{0}$ și \mathbf{M} . Predicatul $\mathbf{0}$ va fi definit astfel: $\mathbf{0}(x)$ va fi adevărat când x este om. Predicatul $\mathbf{M}(x)$ este adevărat când x este muritor. Deoarece predicatele de mai sus au fiecare câte un singur argument/parametru, ele se numesc predicate *unare*. Predicatele generalizează variabilele propoziționale prin faptul că pot primi argumente. De fapt, variabilele propoziționale pot fi văzute ca predicate fără argumente.

Astfel, afirmația *orice om este muritor* va fi modelată prin formula

$$(\forall x. (\mathbf{0}(x) \rightarrow \mathbf{M}(x))),$$

care este citită astfel: *pentru orice x , dacă $\mathbf{0}$ de x , atunci \mathbf{M} de x* . Afirmația *Socrate este om* va fi modelată prin formula $\mathbf{0}(s)$, unde s este o *constantă* prin care înțelegem Socrate, la fel cum prin constanta $\mathbf{0}$ ne referim la numărul natural *zero*. De exemplu, $\mathbf{0}(s)$ este adevărat (deoarece s denotă un om), dar $\mathbf{0}(l)$ este fals dacă, spre exemplu, l este o constantă care ține locul cățelului *Lăbuș*.

Afirmația *Socrate este muritor* va fi reprezentată prin $\mathbf{M}(s)$ (deoarece constanta s se referă la Socrate). Afirmația $\mathbf{M}(s)$ este adevărată deoarece Socrate este muritor; la fel și afirmația $\mathbf{M}(l)$ este adevărată.

Vom vedea că în logica de ordinul I, formula $\mathbf{M}(s)$ este consecință a formulelor $(\forall x. (\mathbf{0}(x) \rightarrow \mathbf{M}(x)))$ și respectiv $\mathbf{0}(s)$. În acest sens, logica de ordinul I este suficient de expresivă pentru a explica din punct de vedere teoretic raționamentul prin care putem deduce că *Socrate este muritor* din faptul că *Orice om este muritor* și din faptul că *Socrate este om*.

Capitolul 2

Structuri și semnături

Cu siguranță ați întâlnit deja mai multe formule din logica de ordinul I, fără să știți neapărat că aveți de a face cu logica de ordinul I. Fie următoare formulă:

$$\varphi = \left(\forall x. (\forall y. (x < y \rightarrow \exists z. (x < z \wedge z < y))) \right).$$

Formula folosește un simbol $<$ căruia îi corespunde un predicat binar $<$ (adică o relație binară) definit astfel: $<(x, y)$ este adevărat dacă x este mai mic strict decât y . Pentru multe predicate binare (inclusiv pentru $<$), pentru a simplifica scrierea, folosim notația infixată ($x < y$) în loc de notația prefixată ($<(x, y)$).

Este formula φ de mai sus adevărată? Formula afirmă că între orice două valori ale variabilelor x, y există o a treia valoare, a variabilei z . Formula este adevărată dacă domeniul variabilelor x, y, z este \mathbb{R} , dar este falsă dacă domeniul este \mathbb{N} (între orice două numere reale există un al treilea, dar între două numere naturale consecutive nu există niciun alt număr natural).

În general, formulele de ordinul I se referă la o anumită *structură matematică*.

Definiția 1 (Structură matematică). *O structură matematică este un triplet $S = (D, Pred, Fun)$, unde:*

- D este o mulțime nevidă numită domeniu;
- fiecare $P \in Pred$ este predicat (de o aritate oarecare) peste mulțimea D ;
- fiecare $f \in Fun$ este funcție (de o aritate oarecare) peste mulțimea D .

Iată câteva exemple de structuri matematice:

1. $(\mathbb{N}, \{<, =\}, \{+, 0, 1\})$;

Domeniul structurii este mulțimea numerelor naturale. Structura conține două predicate: $<$ și $=$, ambele de aritate 2. Predicatul $<$ este predicatul *mai mic* pe numere naturale, iar predicatul $=$ este predicatul de *egalitate* a numerelor naturale.

Funcția binară $+$: $\mathbb{N}^2 \rightarrow \mathbb{N}$ este funcția de adunare a numerelor naturale, iar structura conține și constantele $0 \in \mathbb{N}$ și $1 \in \mathbb{N}$.

2. $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$;

Această structură conține două predicate binare, $<$ și $=$, precum și patru funcții peste \mathbb{R} : funcția binară $+$, funcția unară $-$ și constantele $0, 1 \in \mathbb{R}$.

3. $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$;

Această structură este similară cu structura precedentă, dar domeniul este mulțimea numerelor întregi.

4. $(B, \emptyset, \{., +, \bar{\cdot}\})$;

Această structură este o algebră booleană, unde domeniul este mulțimea valorilor de adevăr, iar funcțiile sunt cele cunoscute din prima jumătate a semestrului. Astfel de structuri, fără niciun predicat, se numesc *structuri algebrice*.

5. $(\mathbb{R}, \{<\}, \emptyset)$.

Această structură conține doar un predicat de aritate 2 (relația *mai mic* peste \mathbb{R}) și nicio funcție. Structurile care nu conțin funcții se numesc structuri relaționale. Structurile relaționale cu domeniul finit se mai numesc baze de date relaționale și se studiază în anul 2.

Când avem o formulă de ordinul I și dorim să îi evaluăm valoarea de adevăr, trebuie să fixăm structura în care lucrăm. Revenind la formula de mai devreme:

$$\varphi = \left(\forall x. (\forall y. (x < y \rightarrow \exists z. (x < z \wedge z < y))) \right),$$

avem că această formulă este adevărată în structura $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$ (între orice două numere reale distincte există cel puțin un număr real) dar este falsă în structura $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$ (deoarece nu între orice două numere întregi putem găsi un alt număr întreg – de exemplu între două numere întregi consecutive nu există niciun întreg). În primul caz, domeniul variabilelor x, y, z este \mathbb{R} și simbolului $<$ îi corespunde predicatul $<_{\subseteq \mathbb{R}^2}$. În al doilea caz, domeniul variabilelor x, y, z este \mathbb{Z} și simbolului $<$ îi corespunde predicatul $<_{\subseteq \mathbb{Z}^2}$.

Este posibil ca două structuri diferite să aibă un set de predicate și de funcții cu același nume. De exemplu, chiar structurile de mai devreme, $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$ și respectiv $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$. Deși predicatul $< \subseteq \mathbb{R}^2$ este diferit de predicatul $< \subseteq \mathbb{Z}^2$, ele au același nume: $<$.

În general, în Matematică și în Informatică, nu facem diferența între un predicat și numele lui, respectiv între o funcție și numele funcției, dar în Logică diferența este extrem de importantă. În particular, dacă ne referim la numele unei funcții vom folosi sintagma *simbol funcțional*, iar dacă ne referim la numele unui predicat vom folosi sintagma *simbol predicativ*. De ce este importantă diferența dintre un simbol predicativ și un predicat? Deoarece vom avea (ne)voie să asociem simbolului predicativ diverse predicate, analog modului în care unei variabile într-un limbaj de programare imperativ îi putem asocia diverse valori.

Când ne interesează doar numele funcțiilor și predicatelor (nu și funcțiile și respectiv predicatele în sine), vom utiliza semnături:

Definiția 2 (Signatură). O semnatură Σ este un tuplu $\Sigma = (\mathcal{P}, \mathcal{F})$ unde \mathcal{P} este o mulțime de simboluri predicative și \mathcal{F} este o mulțime de simboluri funcționale. Fiecare simbol s (predicativ sau funcțional) are asociat un număr natural pe care îl vom numi aritatea simbolului și îl vom nota cu $ar(s)$.

Unei semnături îi putem asocia mai multe structuri:

Definiția 3 (Σ -structuri). Dacă $\Sigma = (\mathcal{P}, \mathcal{F})$ este o semnatură, o Σ -structură este orice structură $S = (D, Pred, Fun)$ astfel încât fiecărui simbol predicativ (sau funcțional) îi corespunde în mod unic un predicat (respectiv, o funcție).

Exemplul 4. Fie $\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\})$ unde \mathbf{P}, \mathbf{Q} sunt simboluri predicative de aritate $ar(\mathbf{P}) = ar(\mathbf{Q}) = 2$ și $\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}$ sunt simboluri funcționale cu aritățile: $ar(\mathbf{f}) = 2$, $ar(\mathbf{i}) = 1$ și $ar(\mathbf{a}) = ar(\mathbf{b}) = 0$.

Avem că $(\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$ și respectiv $(\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$ sunt Σ -structuri.

Observație. Dupa cum se poate observa și în Exemplul 4, pentru simboluri predicative (e.g., \mathbf{P}, \mathbf{Q}) vom utiliza o culoare diferită față de culoarea simbolurilor funcționale (e.g., $\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}$). Pentru predicatele și funcțiile din structuri vom utiliza fontul obișnuit pentru formule matematice.

De reținut!

Structură = domeniu + predicate + funcții

Signatură = simboluri predicative + simboluri funcționale

Unei semnături Σ îi putem asocia mai multe structuri, numite Σ -structuri.

Notăție. *Mulțimea simbolurilor predicative dintr-o Σ -structură de aritate n este notată cu $\mathcal{P}_n = \{P \mid \text{ar}(P) = n\}$, iar mulțimea simbolurilor funcționale de aritate n este notată cu $\mathcal{F}_n = \{f \mid \text{ar}(f) = n\}$. Pentru cazul particular $n = 0$, \mathcal{F}_0 reprezintă mulțimea simbolurilor constante (simboluri funcționale de aritate 0).*

Capitolul 3

Sintaxa logicii de ordinul I

În acest capitol vom prezenta sintaxa formulelor din logica cu predicate de ordinul I. Pentru logica de ordinul I limbajul (mulțimea de șiruri de simboluri) este determinat de alegerea unei semnături Σ . Practic, există mai multe limbaaje de ordinul I, câte un limbaj pentru fiecare semnătură Σ .

În continuare, vom presupune fixată o semnătură Σ cu simboluri predicative \mathcal{P} și simboluri funcționale \mathcal{F} .

3.1 Alfabetul

Ca și formulele din logica propozițională, formulele din logica de ordinul I sunt șiruri de simboluri peste un anumit alfabet. Spre deosebire de logica propozițională, alfabetul este mai bogat și conține următoarele simboluri:

1. *conectori logici* deja cunoscuți: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \perp$, precum și doi *cuan-tificatori*: \forall, \exists ;
2. *variabile*: vom fixa o mulțime infinit numărabilă de variabile, notată $\mathcal{X} = \{x, y, z, x', y', x_1, z'', \dots\}$ (a nu se confunda cu mulțimea variabilelor propoziționale din logica propozițională; sunt două noțiuni fundamental diferite și din acest motiv utilizăm altă culoare pentru a le reprezenta);
3. simboluri auxiliare: $(,), \cdot, \cdot\cdot, (,),$ și $;$;
4. simboluri suplimentare, care sunt specifice fiecărei semnături $\Sigma = (\mathcal{P}, \mathcal{F})$ în parte: simbolurile funcționale din mulțimea \mathcal{F} și respectiv simbolurile predicative din mulțimea \mathcal{P} .

3.2 Termen

Definiția 5. Mulțimea termenilor, \mathcal{T} , este cea mai mică mulțime care satisface următoarele proprietăți:

1. $\mathcal{F}_0 \subseteq \mathcal{T}$ (orice simbol constant este termen);
2. $\mathcal{X} \subseteq \mathcal{T}$ (orice variabilă este termen);
3. dacă $f \in \mathcal{F}_n$ (cu $n > 0$) și $t_1, \dots, t_n \in \mathcal{T}$, atunci $f(t_1, \dots, t_n) \in \mathcal{T}$ (un simbol funcțional de aritate n aplicat unui număr de exact n termeni este termen).

Observație. Deoarece definiția mulțimii termenilor depinde de $\Sigma = (\mathcal{P}, \mathcal{F})$, elementele mulțimii \mathcal{T} se mai numesc Σ -termeni.

Practic, termenii se construiesc aplicând simboluri funcționale peste simboluri constante și variabile.

Exemplul 6. Fie signatura $\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\})$ definită în Exemplul 4, unde $ar(\mathbf{P}) = ar(\mathbf{Q}) = 2$, $ar(\mathbf{f}) = 2$, $ar(\mathbf{i}) = 1$, $ar(\mathbf{a}) = ar(\mathbf{b}) = 0$. Iată câteva exemple de termeni: \mathbf{a} , \mathbf{b} , \mathbf{x} , \mathbf{y} , \mathbf{x}_1 , \mathbf{y}' , $\mathbf{i}(\mathbf{a})$, $\mathbf{i}(\mathbf{x})$, $\mathbf{i}(\mathbf{i}(\mathbf{a}))$, $\mathbf{i}(\mathbf{i}(\mathbf{x}))$, $\mathbf{f}(\mathbf{a}, \mathbf{b})$, $\mathbf{i}(\mathbf{f}(\mathbf{a}, \mathbf{b}))$, $\mathbf{f}(\mathbf{f}(\mathbf{x}, \mathbf{a}), \mathbf{f}(\mathbf{y}, \mathbf{y}))$.

Exercițiul 7. Identificați în lista de mai jos Σ -termenii:

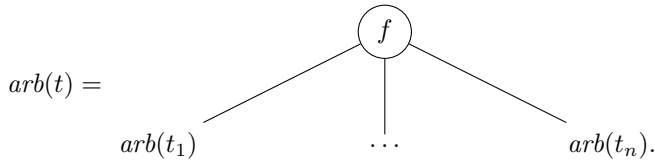
1. $\mathbf{i}(\mathbf{i}(\mathbf{x}))$;
2. \mathbf{i} ;
3. $\mathbf{f}(\mathbf{x}, \mathbf{x})$;
4. $\mathbf{P}(\mathbf{a}, \mathbf{b})$;
5. $\mathbf{i}(\mathbf{a}, \mathbf{a})$;
6. $\mathbf{f}(\mathbf{i}(\mathbf{x}), \mathbf{i}(\mathbf{x}))$;
7. $\mathbf{f}(\mathbf{i}(\mathbf{x}, \mathbf{x}))$;
8. $\mathbf{a}(\mathbf{i}(\mathbf{x}))$.

Termenii (sau, în mod echivalent, *termii*), sunt notați cu t, s, t_1, t_2, s_1, t' , etc. Deși termenii sunt scriși în mod uzual ca un șir de simboluri, ei au asociat un arbore de sintaxă abstractă definit după cum urmează:

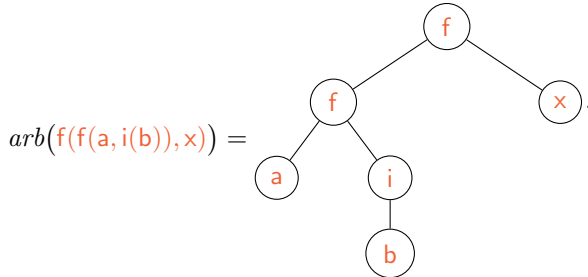
1. dacă $t = c$ și $c \in \mathcal{F}_0$, atunci $arb(t) = \bigcirc_c$

2. dacă $t = x$ și $x \in \mathcal{X}$, atunci $arb(t) = (x)$

3. dacă $t = f(t_1, \dots, t_n)$ și $f \in \mathcal{F}_n$ ($n > 0$), $t_1, \dots, t_n \in \mathcal{T}$, atunci



Observație. Deși formal termenii sunt definiți ca fiind șiruri de simboluri peste alfabetul descris mai sus, aceștia trebuie înțeleși ca fiind arbori. De altfel, în orice software care lucrează cu termeni, aceștia sunt memorati sub formă de arbori cu rădăcină. Iată arborele atașat termenului $f(f(a, i(b)), x)$:



Exercițiul 8. Calculați arborii de sintaxă pentru termenii din Exemplit 6.

3.3 Formule atomice

Definiția 9 (Formulă atomică). O formulă atomică este orice șir de simboluri de forma $P(t_1, \dots, t_n)$, unde $P \in \mathcal{P}_n$ este un simbol predicativ de aritate $n \geq 0$, iar $t_1, \dots, t_n \in \mathcal{T}$ sunt termeni. Dacă $n = 0$, scriem P în loc de $P()$.

Exemplul 10. Continuând Exemplit 6, folosim signatura

$$\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\}),$$

unde $ar(\mathbf{P}) = ar(\mathbf{Q}) = 2$, $ar(\mathbf{f}) = 2$, $ar(\mathbf{i}) = 1$, $ar(\mathbf{a}) = ar(\mathbf{b}) = 0$.

Iată câteva exemple de formule atomice: $\mathbf{P}(\mathbf{a}, \mathbf{b})$, $\mathbf{P}(\mathbf{x}, \mathbf{y})$, $\mathbf{Q}(\mathbf{i}(\mathbf{i}(\mathbf{x})), \mathbf{f}(\mathbf{x}, \mathbf{x}))$, $\mathbf{Q}(\mathbf{a}, \mathbf{b})$, $\mathbf{P}(\mathbf{f}(\mathbf{f}(\mathbf{a}, \mathbf{i}(\mathbf{x})), \mathbf{b}), \mathbf{i}(\mathbf{x}))$.

Exercițiul 11. Explicați de ce $\mathbf{P}(\mathbf{a})$, \mathbf{P} , $\mathbf{i}(\mathbf{i}(\mathbf{x}))$ nu sunt formule atomice peste signatura din Exemplit 10.

3.4 Formule de ordinul I

Definiția 12 (Formule de ordinul I). *Mulțimea formulelor de ordinul I, notată \mathbb{LPI} , este cea mai mică mulțime astfel încât:*

1. (cazul de bază) orice formulă atomică este formulă (adică $P(t_1, \dots, t_n) \in \mathbb{LPI}$ pentru orice simbol predicativ $P \in \mathcal{P}_n$ și orice termeni $t_1, \dots, t_n \in \mathcal{T}$;
2. (cazurile inductive) pentru orice formule $\varphi, \varphi_1, \varphi_2 \in \mathbb{LPI}$, pentru orice variabilă $x \in \mathcal{X}$, avem că:

- (a) $\neg \varphi_1 \in \mathbb{LPI}$;
- (b) $(\varphi_1 \wedge \varphi_2) \in \mathbb{LPI}$;
- (c) $(\varphi_1 \vee \varphi_2) \in \mathbb{LPI}$;
- (d) $(\varphi_1 \rightarrow \varphi_2) \in \mathbb{LPI}$;
- (e) $(\varphi_1 \leftrightarrow \varphi_2) \in \mathbb{LPI}$;
- (f) $(\forall x. \varphi) \in \mathbb{LPI}$;
- (g) $(\exists x. \varphi) \in \mathbb{LPI}$.

Observație. În Definiția 12, regăsim conectorii logici $\neg, \wedge, \vee, \rightarrow$ și respectiv \leftrightarrow din logica propozițională. Locul variabilelor propoziționale (deocamdată, la nivel sintactic) este luat de simbolurile predicative de aritate 0. Construcțiile $(\forall x. \varphi)$ și $(\exists x. \varphi)$ sunt noi.

Exemplul 13. Continuând Exemplul 6, folosim signatura $\Sigma = (\{\mathbf{P}, \mathbf{Q}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{a}, \mathbf{b}\})$, unde $ar(\mathbf{P}) = ar(\mathbf{Q}) = 2$, $ar(\mathbf{f}) = 2$, $ar(\mathbf{i}) = 1$ și $ar(\mathbf{a}) = ar(\mathbf{b}) = 0$.

Iată câteva exemple de formule din logica de ordinul I:

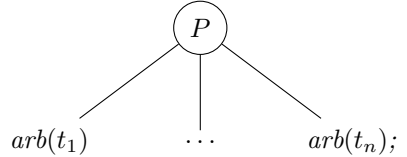
1. $\mathbf{P}(\mathbf{a}, \mathbf{b})$;
2. $\mathbf{Q}(\mathbf{a}, \mathbf{b})$;
3. $\mathbf{P}(\mathbf{a}, \mathbf{x})$;
4. $\neg \mathbf{P}(\mathbf{a}, \mathbf{b})$;
5. $(\mathbf{P}(\mathbf{a}, \mathbf{b}) \wedge \neg \mathbf{Q}(\mathbf{a}, \mathbf{b}))$;
6. $(\mathbf{P}(\mathbf{a}, \mathbf{b}) \vee \neg \mathbf{Q}(\mathbf{x}, \mathbf{y}))$;
7. $(\mathbf{P}(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{P}(\mathbf{a}, \mathbf{b}))$;
8. $((\mathbf{P}(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{P}(\mathbf{a}, \mathbf{b})) \leftrightarrow (\mathbf{P}(\mathbf{a}, \mathbf{b}) \rightarrow \mathbf{P}(\mathbf{a}, \mathbf{b})))$;

9. $(\forall x.P(a, x));$

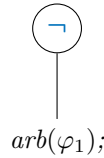
10. $(\exists x.\neg Q(x, y)).$

Definiția 14 (Arborele de sintaxă abstractă asociat formulelor din \mathbb{LPI}).
Formulele au asociat un arbore de sintaxă abstractă definit în cele ce urmează:

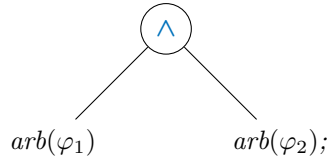
1. dacă $\varphi = P(t_1, \dots, t_n)$, atunci $arb(\varphi) =$



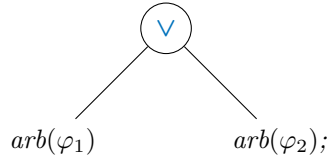
2. dacă $\varphi = \neg\varphi_1$, atunci $arb(\varphi) =$



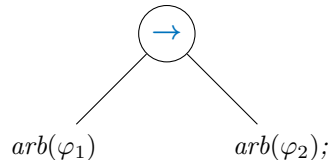
3. dacă $\varphi = (\varphi_1 \wedge \varphi_2)$, atunci $arb(\varphi) =$



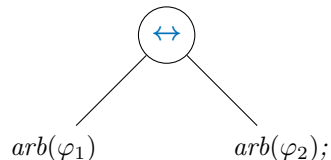
4. dacă $\varphi = (\varphi_1 \vee \varphi_2)$, atunci $arb(\varphi) =$



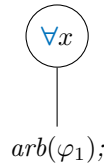
5. dacă $\varphi = (\varphi_1 \rightarrow \varphi_2)$, atunci $arb(\varphi) =$



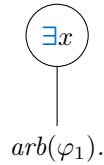
6. dacă $\varphi = (\varphi_1 \leftrightarrow \varphi_2)$, atunci $arb(\varphi) =$



7. dacă $\varphi = (\forall x.\varphi_1)$, atunci $arb(\varphi) =$



8. dacă $\varphi = (\exists x.\varphi_1)$, atunci $arb(\varphi) =$



Exercițiul 15. Calculați arborele de sintaxă asociat formulelor din Exemplul 13.

3.5 Paranteze

Parantezele (și) sunt folosite pentru a marca ordinea efectuării operațiilor logice (și, sau, not, etc.) În continuare, vom renunța la parantezele care nu sunt necesare, la fel ca în cazul logicii propoziționale: dacă o formulă poate fi interpretată ca un arbore de sintaxă abstractă în două sau mai multe moduri, se vor folosi paranteze pentru a stabili arborele dorit.

De exemplu, $\varphi_1 \vee \varphi_2 \wedge \varphi_3$ ar putea fi înțeleasă ca $((\varphi_1 \vee \varphi_2) \wedge \varphi_3)$ sau ca $(\varphi_1 \vee (\varphi_2 \wedge \varphi_3))$. Pentru a evita scrierea formulelor cu prea multe paranteze, se stabilesc *priorități*. Ordinea priorității operatorilor este:

$$\perp, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists,$$

unde \neg este cel mai prioritar, iar \exists este cel mai puțin prioritar. Pentru a evita situațiile de ambiguitate (i.e., atunci când există mai multe moduri de a construi arborele de sintaxă), este recomandată folosirea de paranteze suplimentare.

Datorită priorității conectorilor logici, formula $P(a, a) \vee P(b, b) \wedge P(a, b)$ va fi întotdeauna înțeleasă ca $(P(a, a) \vee (P(b, b) \wedge P(a, c)))$ (deoarece \wedge este prioritar față de \vee). Ca analogie, la fel se întâmplă și în cazul aritmeticii: $1 + 2 * 3$ va fi înțeles ca $1 + (2 * 3)$, deoarece \times are prioritate în fața lui $+$ (\times este similar cu \wedge și $+$ cu \vee).

Exercițiul 16. Scrieți formulele din Exemplul 13 cu cât mai puține paranteze.

În Secțiunea 3.12 vom detalia modul în care cuantificatorii interacționează cu ceilalți conectori logici. Vom vedea că pentru cuantificatori avem niște reguli suplimentare în afară de priorități.

3.6 Modelarea în LPI a afirmațiilor din limba română

În această secțiune vom explica care este signatura folosită pentru a modela în logica de ordinul întâi afirmațiile: *Orice om este muritor*, *Socrate este om* și respectiv *Socrate este muritor*.

În primul rând, identificăm predicatele din text. Avem două predicate unare *este om* și respectiv *este muritor*. Alegem simbolul predicativ **O** pentru primul predicat și simbolul predicativ **M** pentru al doilea predicat. De asemenea, în text avem și o constantă: Socrate. Alegem simbolul funcțional **s** de aritate 0 pentru această constantă. Așadar, pentru a modela afirmațiile de mai sus, vom lucra cu signatura

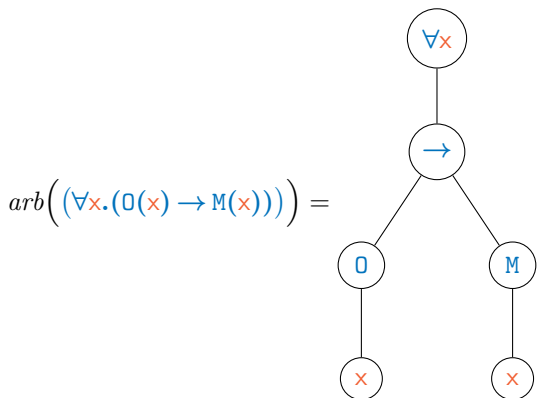
$$\Sigma = (\{\mathbf{O}, \mathbf{M}\}, \{\mathbf{s}\}),$$

unde **O** și **M** sunt simboluri predicative de aritate $ar(\mathbf{O}) = ar(\mathbf{M}) = 1$, iar **s** este un simbol funcțional de aritate $ar(\mathbf{s}) = 0$, adică un simbol constant.

Afirmația *orice om este muritor* va fi modelată prin formula de ordinul I

$$(\forall x. (\mathbf{O}(x) \rightarrow \mathbf{M}(x))),$$

al cărei arbore de sintaxă abstractă este:



Afirmația *Socrate este om* o vom modela prin formula atomică $\mathbf{O}(\mathbf{s})$, iar afirmația *Socrate este muritor* o vom modela prin formula atomică $\mathbf{M}(\mathbf{s})$.

Pentru signatura $\Sigma = (\{\mathbf{O}, \mathbf{M}\}, \{\mathbf{s}\})$ stabilită mai sus, există mai multe Σ -structuri posibile. Un exemplu este structura $S = (D, \{\mathbf{O}^S, \mathbf{M}^S\}, \{\mathbf{s}^S\})$ definită astfel:

1. D este mulțimea tuturor ființelor de pe Pământ;

2. $0^S(x)$ este adevărat pentru orice ființă x care este și om;
3. $M^S(x)$ este adevărat pentru orice ființă x (toate elementele domeniului sunt muritoare);
4. s^S este Socrate (Socrate, fiind o ființă, aparține mulțimii D).

Anticipând puțin (vom discuta despre semantica formulelor de ordinul I în Capitolul 4), toate cele trei formule discutate în această secțiune, adică $(\forall x.(0(x) \rightarrow M(x)))$, $0(s)$ și respectiv $M(s)$, sunt adevărate în structura S definită mai sus. De fapt, calitatea raționamentului *orice om este muritor; Socrate este om; deci: Socrate este muritor* este dată de faptul că formula $M(s)$ este în mod necesar adevărată în *orice* structură în care formulele $0(s)$ și $(\forall x.(0(x) \rightarrow M(x)))$ sunt adevărate, nu doar în structura S de mai sus.

3.7 Modelarea în \mathbb{LPI} a afirmațiilor despre aritmetică

Fie signatura $\Sigma = (\{<, =\}, \{+, -, 0, 1\})$, unde $<$ și $=$ sunt simboluri predicative de aritate 2, $+$ este simbol funcțional de aritate 2, $-$ este simbol funcțional de aritate 1, iar 0 și 1 sunt simboluri constante. Iată câteva formule care fac parte din limbaajul de ordinul I asociat signaturii Σ :

1. $(\forall x.(\forall y.(<(x, y) \rightarrow \exists z.(<(x, z) \wedge <(z, y))))$;
2. $(\forall x.(\forall y.(\exists z.(= (+ (x, y), z))))$;
3. $(\forall x.(<(0, x) \vee =(0, x)))$;
4. $(\forall x.(\exists y.(= (x, -(y))))$;
5. $=(+(x, y), z)$.

De multe ori, în cazul simbolurilor predicative și simbolurilor funcționale binare, se folosește notația infixată (e.g., $x < y$ în loc de $<(x, y)$). În acest caz, putem scrie formulele de mai sus în felul următor:

1. $(\forall x.(\forall y.(x < y \rightarrow \exists z.(x < z \wedge z < y))))$;
2. $(\forall x.(\forall y.(\exists z.(x + y = z))))$;
3. $(\forall x.(0 < x \vee 0 = x))$;

$$4. (\forall x. (\exists y. (x = -(y))));$$

$$5. x + y = z.$$

Două dintre Σ -structurile posibile sunt $S_1 = (\mathbb{R}, \{<, =\}, \{+, -, 0, 1\})$ și $S_2 = (\mathbb{Z}, \{<, =\}, \{+, -, 0, 1\})$, unde predicatele și funcțiile sunt cele cunoscute de la matematică (cu precizarea că $-$ este funcția minus unar).

Anticipând cursul următor referitor la semantica formulelor de ordinul I, prima formulă este falsă în S_2 și adevărată în S_1 . A doua formulă și a patra formulă sunt adevărate atât în S_1 cât și în S_2 . A treia formula este falsă atât în S_1 cât și în S_2 . Valoarea de adevăr a celei de-a cincea formule nu depinde doar de structura în care evaluăm formula, ci și de valorile variabilelor x, y, z . Deoarece variabilele x, y, z nu apar cuantificate în formula numărul 5, acestea se numesc *libere*. Formula 5 este *satisfiabilă* atât în structura S_1 cât și în structura S_2 , deoarece în ambele cazuri există valori pentru variabilele x, y, z care să facă formula adevărată (e.g. valorile 1, 2, 3 pentru x, y și respectiv z).

3.8 Variabilele unei formule

Cu $\text{vars}(\varphi)$ notăm variabilele care apar în formula φ . De exemplu, vom avea că $\text{vars}((\forall z. (P(x, y)))) = \{x, y, z\}$. Definim funcția $\text{vars} : \mathbb{LPI} \rightarrow 2^{\mathcal{X}}$ în cele ce urmează.

În primul rând, definim o funcție $\text{vars} : \mathcal{T} \rightarrow 2^{\mathcal{X}}$ (atenție, domeniul este \mathcal{T}) ca fiind funcția care asociază unui termen (din mulțimea \mathcal{T}) mulțimea *variabilelor care apar* în acel termen. Toate definițiile care urmează vor fi definiții inductive, care oglindesc definițiile sintactice corespunzătoare. Le vom denumi simplu *definiții recursive*, fără a mai preciza explicit cazurile de bază sau pe cele inductive. Amintim că $2^{\mathcal{X}}$ denotă mulțimea tuturor submulțimilor lui \mathcal{X} . Totodată, amintim că pentru o semnătură fixată Σ , notăm cu \mathcal{P}_n mulțimea simbolurilor predicative de aritate n din Σ , iar cu \mathcal{F}_n mulțimea simbolurilor funcționale de aritate n din Σ .

Definiția 17. *Funcția $\text{vars} : \mathcal{T} \rightarrow 2^{\mathcal{X}}$ este definită recursiv după cum urmează:*

1. $\text{vars}(c) = \emptyset$, dacă $c \in \mathcal{F}_0$ este un simbol constant;
2. $\text{vars}(x) = \{x\}$, dacă $x \in \mathcal{X}$ este o variabilă;
3. $\text{vars}(f(t_1, \dots, t_n)) = \bigcup_{i \in \{1, \dots, n\}} \text{vars}(t_i)$.

Putem acum defini (inductiv, prin imbricare) funcția extinsă și notată omonim, $\text{vars} : \mathbb{LPI} \rightarrow 2^{\mathcal{X}}$, care asociază unei formule din \mathbb{LPI} mulțimea de variabile ale formulei (adică, variabilele care apar în formulă):

Definiția 18. Funcția $\text{vars} : \mathbb{LPI} \rightarrow 2^{\mathcal{X}}$ este definită recursiv după cum urmează:

1. $\text{vars}(P(t_1, \dots, t_n)) = \bigcup_{i \in \{1, \dots, n\}} \text{vars}(t_i)$;
2. $\text{vars}(\neg \varphi) = \text{vars}(\varphi)$;
3. $\text{vars}((\varphi_1 \wedge \varphi_2)) = \text{vars}(\varphi_1) \cup \text{vars}(\varphi_2)$;
4. $\text{vars}((\varphi_1 \vee \varphi_2)) = \text{vars}(\varphi_1) \cup \text{vars}(\varphi_2)$;
5. $\text{vars}((\varphi_1 \rightarrow \varphi_2)) = \text{vars}(\varphi_1) \cup \text{vars}(\varphi_2)$;
6. $\text{vars}((\varphi_1 \leftrightarrow \varphi_2)) = \text{vars}(\varphi_1) \cup \text{vars}(\varphi_2)$;
7. $\text{vars}((\forall x. \varphi)) = \text{vars}(\varphi) \cup \{x\}$;
8. $\text{vars}((\exists x. \varphi)) = \text{vars}(\varphi) \cup \{x\}$.

Să observăm că variabila x este adăugată corespunzător în mulțimea de variabile care se construiește chiar dacă ea apare doar imediat după simbolurile \exists sau \forall .

Exemplul 19. Fie formula φ :

$$\left(\left(\forall x. (P(x, y) \wedge \exists y. (P(z, f(x, y)) \wedge P(x, y))) \right) \wedge P(x, x) \right).$$

$$\text{Avem } \text{vars}(\varphi) = \{x, y, z\}.$$

Exercițiul 20. Fie semnatura $\Sigma = (\{P, Q\}, \{f, i, a, b\})$, unde $ar(P) = ar(Q) = 2$, $ar(f) = 2$, $ar(i) = 1$, $ar(a) = ar(b) = 0$. Calculați $\text{vars}(\varphi)$ pentru fiecare formulă φ de mai jos:

1. $P(x, y)$;
2. $Q(a, b)$;
3. $P(a, x)$;
4. $\neg P(x, z)$;
5. $(P(x, x) \wedge \neg Q(x, z))$;
6. $(P(x, b) \vee \neg Q(z, y))$;
7. $((P(x, b) \rightarrow P(x, z)) \leftrightarrow (P(x, b) \rightarrow P(a, z)))$;
8. $(\forall x. P(a, x))$;
9. $(\exists x. \neg Q(x, y))$;
10. $((\exists x. \neg Q(x, y)) \wedge (\forall y. P(y, x)))$.

3.9 Domeniul de vizibilitate al unui cuantificator - analogie cu limbajele de programare

Într-un limbaj de programare, putem declara mai multe variabile cu același nume. De exemplu, în C, putem avea următorul cod:

```
/* 1:*/ int f()
/* 2:*/ {
/* 3:*/   int s = 0;
/* 4:*/   for (int x = 1; x <= 10; ++x) {
/* 5:*/     for (int y = 1; y <= 10; ++y) {
/* 6:*/       s += x * y * z;
/* 7:*/       for (int x = 1; x <= 10; ++x) {
/* 8:*/         s += x * y * z;
/* 9:*/       }
/* 10:*/     }
/* 11:*/   }
/* 12:*/   return s;
/* 13:*/ }
```

În acest fragment de cod, sunt declarate trei variabile, două dintre variabile având același nume, și anume x . Domeniul de vizibilitate al variabilei x declarate la linia 4 este dat de liniile 4 – 11, iar domeniul de vizibilitate al variabile x declarată la linia 7 este dat de liniile 7 – 9. Astfel, orice apariție a numelui x între liniile 7 – 9 se referă la cea de-a doua declarație a variabilei, în timp ce orice apariție a numelui x între liniile 4 – 11 (cu excepția liniilor 7 – 9) se referă la prima declarație a lui x . De exemplu, apariția lui x de la linia 6 se referă la variabila x declarată la linia 4. Apariția lui x de la linia 8 se referă la variabila x declarată la linia 7.

Liniile 4 – 11 reprezintă domeniul de vizibilitate al primei declarații a variabilei x , iar liniile 7 – 9 reprezintă domeniul de vizibilitate al celei de-a doua declarații a variabilei x .

Un fenomen similar se întâmplă în formulele logicii de ordinul I. De exemplu, în formula $(\forall x.(\forall y.(P(x, y) \wedge P(x, z) \wedge (\exists x.P(x, y)))))$, variabila x este cuantificată de două ori (prima dată universal, a doua oară existențial). O cuantificare a unei variabile se numește *legare* (engl. *binding*), din motive istorice. O *legare* este similară, din punctul de vedere al domeniului de vizibilitate, cu definirea unei variabile într-un limbaj de programare.

Astfel, domeniul de vizibilitate D_1 al variabilei x cuantificate universal este $(\forall y.(P(x, y) \wedge P(x, z) \wedge (\exists x.P(x, y))))$, în timp ce domeniul de vizibilitate D_2 al variabilei x cuantificate existențial este $P(x, y)$:

$$\left(\underbrace{\forall x. (\underbrace{\forall y. (P(x, y) \wedge P(x, z) \wedge (\exists x. \overbrace{P(x, y)}^{D_2}))}_{D_1})}_{D_1} \right)$$

Aparițiile unei variabile cuantificate care sunt prezente în domeniul de vizibilitate al acesteia se numesc *legate*, în timp ce aparițiile din afara domeniului de vizibilitate se numesc *libere*.

3.10 Apariții libere și legate ale variabilelor

În această secțiune vom stabili formal conceptul de apariție/variabilă legată și de apariție/variabilă liberă. Aparițiile libere/legate ale unei variabile în logica de ordinul I sunt, ca o analogie, similare cu variabilele globale/locale într-un limbaj de programare.

Mai departe vom utiliza noțiunea de *arbore de sintaxă abstractă* pentru formulele din logica de ordinul I.

Definiția 21 (Apariție liberă). *O apariție liberă a unei variabile x într-o formulă φ este dată de un nod în arborele formulei etichetat cu x și care are proprietatea că, mergând din nod înspre rădăcină, nu întâlnim niciun nod etichetat cu $\forall x$ sau cu $\exists x$.*

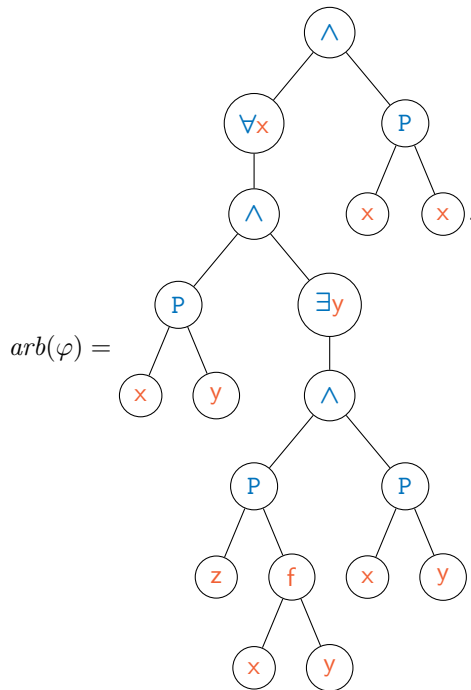
Definiția 22 (Apariție legată). *O apariție legată a unei variabile x într-o formulă φ este dată de un nod în arborele formulei etichetat cu x și care are proprietatea că, mergând din nod înspre rădăcină, întâlnim măcar un nod etichetat cu $\forall x$ sau cu $\exists x$.*

Cel mai apropiat astfel de nod etichetat cu $\forall x$ sau cu $\exists x$ este cuantificarea care leagă apariția în cauză a variabilei x .

Exemplul 23. *Considerăm în continuare formula*

$$\varphi = \left(\left(\forall x. (P(x, y) \wedge \exists y. (P(z, f(x, y)) \wedge P(x, y))) \right) \wedge P(x, x) \right).$$

Arborele de sintaxă abstractă al formulei φ este:



În formula φ de mai sus, variabila x are două apariții libere. Variabila y are o apariție liberă. Variabila z are o apariție liberă. Toate aparițiile libere ale variabilelor în formula φ sunt marcate prin subliniere:

$$\varphi = \left(\left(\forall x. (P(\underline{x}, \underline{y}) \wedge \exists y. (P(\underline{z}, f(\underline{x}, \underline{y})) \wedge P(\underline{x}, \underline{y}))) \right) \right) \wedge P(\underline{x}, \underline{x}).$$

Toate aparițiile legate ale variabilelor în formula φ sunt marcate prin dublă subliniere în următoarea formulă:

$$\varphi = \left(\left(\forall x. (P(\underline{\underline{x}}, \underline{\underline{y}}) \wedge \exists y. (P(\underline{\underline{z}}, f(\underline{\underline{x}}, \underline{\underline{y}})) \wedge P(\underline{\underline{x}}, \underline{\underline{y}}))) \right) \right) \wedge P(\underline{\underline{x}}, \underline{\underline{x}}).$$

Observație. În restul capitolelor vom face o distincție clară între aparițiile (libere sau legate ale) variabilelor într-o formulă și mulțimile variabilelor de acest tip (libere sau legate). Nodurile etichetate cu $\forall x$ și respectiv $\exists x$ nu vor fi considerate ca desemnând nici o apariție liberă, nici o apariție legată a lui x , ci ca fiind simple noduri prin care se fixează/denumeste un cuantificator (sau, prin care se leagă variabila x).

3.11 Variabile libere și variabile legate

Mulțimea variabilelor unei formule φ care au cel puțin o apariție liberă se notează $free(\varphi)$.

Definiția 24. Funcția $free : \mathbb{LPI} \rightarrow 2^{\mathcal{X}}$ este definită recursiv în modul următor:

1. $free(P(t_1, \dots, t_n)) = vars(t_1) \cup \dots \cup vars(t_n)$;
2. $free(\neg\varphi) = free(\varphi)$;
3. $free((\varphi_1 \wedge \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
4. $free((\varphi_1 \vee \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
5. $free((\varphi_1 \rightarrow \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
6. $free((\varphi_1 \leftrightarrow \varphi_2)) = free(\varphi_1) \cup free(\varphi_2)$;
7. $free((\forall x.\varphi)) = free(\varphi) \setminus \{x\}$;
8. $free((\exists x.\varphi)) = free(\varphi) \setminus \{x\}$.

Exemplul 25. Pentru formula

$$\varphi = \left(\left(\forall x. (P(x, y) \wedge \exists y. (P(z, f(x, y)) \wedge P(x, y))) \right) \right) \wedge P(x, x).$$

avem că $free(\varphi) = \{x, y, z\}$.

Exercițiul 26. Calculați $free(\varphi)$ pentru fiecare formulă φ din Exercițiul 20.

Cu $bound(\varphi)$ notăm mulțimea variabilelor legate într-o formulă, cu alte cuvinte mulțimea acelor variabile x cu proprietatea că există în formulă cel puțin un nod etichetat cu $\forall x$ sau cu $\exists x$.

Definiția 27. Funcția $bound : \mathbb{LPI} \rightarrow 2^{\mathcal{X}}$ este definită recursiv astfel:

1. $bound(P(t_1, \dots, t_n)) = \emptyset$;
2. $bound(\neg\varphi) = bound(\varphi)$;
3. $bound((\varphi_1 \wedge \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;
4. $bound((\varphi_1 \vee \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;
5. $bound((\varphi_1 \rightarrow \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;
6. $bound((\varphi_1 \leftrightarrow \varphi_2)) = bound(\varphi_1) \cup bound(\varphi_2)$;

$$7. \text{bound}((\forall x.\varphi)) = \text{bound}(\varphi) \cup \{x\};$$

$$8. \text{bound}((\exists x.\varphi)) = \text{bound}(\varphi) \cup \{x\}.$$

Exercițiul 28. Calculați $\text{bound}(\varphi)$ pentru fiecare formulă φ din Exercițiul 20.

Exercițiul 29. Calculați $\text{bound}(\varphi)$, unde

$$\varphi = \left(\left(\forall x. (P(x, y) \wedge \exists y. (P(z, f(x, y)) \wedge P(x, y))) \right) \wedge P(x, x) \right).$$

Definiția 30. Variabilele legate ale unei formule φ sunt elementele mulțimii $\text{bound}(\varphi)$.

Definiția 31. Variabilele libere ale unei formule φ sunt elementele mulțimii $\text{free}(\varphi)$.

Observație. Mulțimile $\text{free}(\varphi)$ și $\text{bound}(\varphi)$ pot avea elemente în comun.

Observație. O variabilă poate avea mai multe apariții într-o formulă.

După cum am precizat anterior, trebuie făcută diferența între o apariție liberă a unei variabile într-o formulă și o variabilă liberă a unei formule. Apariția liberă este indicată printr-un nod din arborele formulei, în timp ce variabila liberă este un element al mulțimii \mathcal{X} .

Similar, trebuie făcută diferența între o apariție legată a unei variabile într-o formulă și o variabilă legată a unei formule. Apariția legată este indicată de un nod în arborele formulei, în timp ce variabila este un element al mulțimii \mathcal{X} .

3.12 Domeniul de vizibilitate și parantetizarea formulelor

Acum că am înțeles ce este domeniul de vizibilitate a unei variabile legate (apelând și la arborele formulei), putem clarifica un aspect referitor la ordinea de prioritate a conectorilor logici (dacă privim formula doar ca text/cuvânt). Am stabilit deja că ordinea de prioritate este $\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \forall, \exists$, dar cuantorificatorii \forall și \exists interacționează într-un mod mai subtil cu ceilalți conectori logici. Mai precis, textual, o formulă fără paranteze se (re)parantetizează astfel încât domeniul de vizibilitate al fiecărui cuantorificator să se extindă cât mai mult spre dreapta.

De exemplu, formula:

$$\forall x. P(x, x) \vee \neg \exists y. P(x, y) \wedge P(x, x)$$

se parantezează în felul următor:

$$\left(\forall x. (P(x, x) \vee \neg(\exists y. (P(x, y) \wedge P(x, x)))) \right).$$

3.13 Fișă de exerciții

Exercițiul 32. *Identificați o structură și o semnătură potrivite pentru afirmațiile de mai jos și apoi modelați aceste afirmații ca formule în logica de ordinul I:*

Ion este student. Orice student învață la Logică. Oricine învață la Logică trece examenul. Orice student este om. Există un om care nu a trecut examenul. Deci nu toți oamenii sunt studenți.

Exercițiul 33. *Fie structura $S = (\mathbb{R}, \{Nat, Int, Prim, Par, >\}, \{+, 0, 1, 2\})$, unde Nat , Int , $Prim$, Par sunt predicate unare cu următoarea semnificație:*

- $Nat(u) = u$ este număr natural;
- $Int(u) = u$ este număr întreg;
- $Prim(u) = u$ este număr prim;
- $Par(u) = u$ este număr par.

Predicatul binar $>$ este relația “mai mare” peste numere reale. Funcția $+$ este funcția de adunare a numerelor reale. Constantele $0, 1, 2$ sunt chiar numerele $0, 1, 2$.

1. *Propuneți o semnătură Σ pentru structura S de mai sus.*
2. *Modelați următoarele afirmații ca formule de ordinul I în semnatura asociată structurii S de mai sus:*
 - (a) *Orice număr natural este și număr întreg.*
 - (b) *Suma oricăror două numere naturale este număr natural.*
 - (c) *Oricum am alege un număr natural, există un număr prim care este mai mare decât numărul respectiv.*
 - (d) *Dacă orice număr natural este număr prim, atunci zero este număr prim.*
 - (e) *Oricum am alege un număr prim, există un număr prim mai mare decât el.*
 - (f) *Suma a două numere pare este un număr par.*

- (g) Orice număr prim mai mare decât 2 este impar.
- (h) Orice număr prim poate fi scris ca suma a patru numere prime.
- (i) Suma a două numere pare este un număr impar.

Exercițiul 34. Dați exemplu de 5 termeni peste semnăturile de la Exercițiul 33 și calculați arborele de sintaxă abstractă al acestor termeni.

Exercițiul 35. Dați exemplu de 5 formule peste semnătura de la Exercițiul 33 și calculați arborele de sintaxă abstractă al acestora.

Exercițiul 36. Calculați arborele de sintaxă abstractă al următoarelor formule (indicație: puneți paranteze în jurul formulelor, în ordinea de prioritate a conectorilor):

1. $P(x) \vee P(y) \wedge \neg P(z)$;
2. $\neg \neg P(x) \vee P(y) \rightarrow P(x) \wedge \neg P(z)$;
3. $\forall x. \forall y. \neg \neg P(x) \vee P(y) \rightarrow P(x) \wedge \neg P(z)$;
4. $\forall x. \forall y. \neg \neg P(x) \vee P(y) \rightarrow \exists x. P(x) \wedge \neg P(x)$;
5. $\forall x'. \neg \forall x. P(x) \wedge \exists y. Q(x, y) \vee \neg Q(z, z) \rightarrow \exists z'. P(z')$.

Exercițiul 37. Marcați aparițiile libere și respectiv aparițiile legate ale variabilelor în formulele de mai jos:

1. $\varphi_1 = (\forall x. P(x, x) \wedge P(x, y)) \wedge P(x, z)$;
2. $\varphi_2 = (\forall x. P(f(x, x), i(x)) \wedge \exists y. (P(x, y) \wedge P(x, z)))$.

Exercițiul 38. Identificați domeniul de vizibilitate, pentru fiecare cuantificator, în formulele φ_1 și φ_2 date în Exercițiul 37.

Exercițiul 39. Găsiți variabilele, variabilele libere și respectiv variabilele legate ale formulelor φ_1 și φ_2 date în Exercițiul 37.

Exercițiul 40. Fie $A = \{p, q, r, \dots\}$ mulțimea variabilelor propoziționale. Vom considera semnătura $\Sigma_{\mathbb{LP}} = (A, \emptyset)$, unde variabilele propoziționale din A sunt simboluri predicative de aritate 0.

1. Demonstrați că pentru orice formulă $\varphi \in \mathbb{LP}$ avem $\varphi \in \mathbb{LPI}$ (peste semnătura $\Sigma_{\mathbb{LP}}$).
2. Demonstrați că pentru orice formulă fără cuantificatori $\varphi \in \mathbb{LPI}$ peste semnătura $\Sigma_{\mathbb{LP}}$, avem că $\varphi \in \mathbb{LP}$.

Capitolul 4

Semantica formulelor logicii de ordinul I

Sintaxa logicii de ordinul I explică care sunt, din punct de vedere sintactic, formulele logicii de ordinul I. Semantica logicii de ordinul I se referă la *înțelesul* formulelor. Semantica unei formule (sau înțelesul formulei) va fi o valoare de adevăr. Ca și la logica propozițională, în general, valoarea de adevăr a unei formule depinde nu doar de formulă, ci și de *structura* în care evaluăm formula.

Reamintim că o semnătură $\Sigma = (\mathcal{P}, \mathcal{F})$ este o pereche formată dintr-o mulțime de simboluri predicative \mathcal{P} și o mulțime de simboluri funcționale \mathcal{F} . Fiecare simbol are atașat un număr natural numit aritatea simbolului.

În acest capitol vom utiliza semnătura $\Sigma = (\{\mathbf{P}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{e}\})$, unde \mathbf{P} este simbol predicativ de aritate 2, iar \mathbf{f} , \mathbf{i} și \mathbf{e} sunt simboluri funcționale de aritate 2, 1 și respectiv 0. Altfel spus, $\mathcal{P}_2 = \{\mathbf{P}\}$, $\mathcal{P}_1 = \emptyset$, $\mathcal{P}_0 = \emptyset$, $\mathcal{F}_2 = \{\mathbf{f}\}$, $\mathcal{F}_1 = \{\mathbf{i}\}$ și $\mathcal{F}_0 = \{\mathbf{e}\}$.

Reamintim și că, dacă $\Sigma = (\mathcal{P}, \mathcal{F})$ este o semnătură, prin Σ -structură înțelegem orice tuplu $S = (D, Pred, Fun)$ cu proprietatea că:

1. D este o mulțime nevidă numită domeniul structurii S ;
2. pentru fiecare simbol predicativ $P \in \mathcal{P}$ există un predicat $P^S \in Pred$ de aritate corespunzătoare;
3. pentru fiecare simbol funcțional $f \in \mathcal{F}$ există o funcție $f^S \in Fun$ de aritate corespunzătoare.

Exemplul 41. *Mai jos avem câteva exemple de Σ -structuri:*

1. $S_1 = (\mathbb{Z}, \{=\}, \{+, -, 0\})$;

$$2. S_2 = (\mathbb{R}^*, \{=\}, \{\times, \cdot^{-1}, 1\});$$

$$3. S_3 = (\mathbb{N}, \{=\}, \{+, s, 0\});$$

$$4. S_4 = (\mathbb{N}, \{<\}, \{+, s, 0\});$$

$$5. S_5 = (\mathbb{Z}, \{<\}, \{+, -, 0\}).$$

Structura S_1 are domeniul \mathbb{Z} (mulțimea numerelor întregi), predicatul asociat simbolului predicativ \mathbf{P} este = (predicatul de egalitate pentru numere întregi), funcția $+$ este funcția de adunare a numerelor întregi asociată simbolului funcțional \mathbf{f} , $-$ este funcția minus unar asociată simbolului funcțional \mathbf{i} , iar simbolul constant \mathbf{e} are asociată constanta 0.

Structura S_2 are domeniul \mathbb{R}^* (mulțimea numerelor reale strict pozitive), predicatul asociat simbolului predicativ \mathbf{P} este = (predicatul de egalitate pentru numere reale pozitive), funcția \times este funcția de înmulțire a numerelor reale asociată simbolului funcțional \mathbf{f} , \cdot^{-1} este funcția unară asociată simbolului funcțional \mathbf{i} care calculează inversul unui număr real (e.g. $5^{-1} = \frac{1}{5}$, iar $\frac{1}{10}^{-1} = 10$), iar simbolul constant \mathbf{e} are asociată constanta 1.

Structura S_3 are domeniul \mathbb{N} (mulțimea numerelor naturale), predicatul asociat simbolului predicativ \mathbf{P} este = (predicatul de egalitate pentru numere naturale), funcția $+$ este funcția de adunare a numerelor naturale asociată simbolului funcțional \mathbf{f} , s este funcția succesor (care asociază unui număr natural următorul număr natural – e.g., $s(7) = 8$) asociată simbolului funcțional \mathbf{i} , iar simbolul constant \mathbf{e} are asociată constanta 0.

Structura S_4 are domeniul \mathbb{N} (mulțimea numerelor naturale), predicatul asociat simbolului predicativ \mathbf{P} este $<$ (relația mai mic peste numere naturale), funcția $+$ este funcția de adunare a numerelor naturale asociată simbolului funcțional \mathbf{f} , s este funcția succesor (care asociază unui număr natural următorul număr natural – e.g., $s(7) = 8$) asociată simbolului funcțional \mathbf{i} , iar simbolul constant \mathbf{e} are asociată constanta 0.

Structura S_5 este similară cu S_1 , doar că simbolul predicativ \mathbf{P} are asociată relația mai mic în loc de egal.

Folosind notațiile de mai sus, avem că $\mathbf{P}^{S_4} = <$, $\mathbf{f}^{S_2} = \times$, iar $\mathbf{e}^{S_1} = 0$.

4.1 Atribuiri

Asemănător cu logica propozițională, pentru a obține valoarea de adevăr a unei formule într-o structură, trebuie să pornim cu fixarea unor valori concrete pentru simbolurile sintactice din alfabetul peste care este construită formula. În cazul de față, începem cu variabilele.

Definiția 42 (Atribuire). *Fie Σ o semnătură și S o Σ -structură cu domeniul D . Se numește S -atribuire orice funcție*

$$\alpha : \mathcal{X} \rightarrow D.$$

Exemplul 43. *Funcția $\alpha_1 : \mathcal{X} \rightarrow \mathbb{Z}$, definită ca mai jos, este o S_1 -atribuire:*

1. $\alpha_1(\mathbf{x}_1) = 5$;
2. $\alpha_1(\mathbf{x}_2) = 5$;
3. $\alpha_1(\mathbf{x}_3) = 6$;
4. $\alpha_1(x) = 0$ pentru orice $x \in \mathcal{X} \setminus \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$.

Exemplul 44. *Funcția $\alpha_2 : \mathcal{X} \rightarrow \mathbb{Z}$, definită ca mai jos, este o S_1 -atribuire:*

1. $\alpha_2(\mathbf{x}_1) = 6$;
2. $\alpha_2(\mathbf{x}_2) = 5$;
3. $\alpha_2(\mathbf{x}_3) = 6$;
4. $\alpha_2(x) = 0$ pentru orice $x \in \mathcal{X} \setminus \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$.

Observație. *Atenție! A nu se confunda noțiunea de atribuire din Logica de Ordinul I cu noțiunea de atribuire de valori de adevăr din Logica Propozițională.*

Acum, având la dispoziție o atribuire α , putem calcula valoarea unui termen într-o asemenea atribuire. Pentru aceasta, vom folosi de fapt extensia lui α , notată $\bar{\alpha}$,

$$\bar{\alpha} : \mathcal{T} \rightarrow D,$$

dată în definiția care urmează.

Definiția 45 (Valoarea unui termen într-o atribuire). *Dându-se o S -atribuire α și un termen $t \in \mathcal{T}$ peste semnatura Σ , valoarea termenului t în atribuirea α este un element al domeniului D notat cu $\bar{\alpha}(t)$ și calculat recursiv astfel:*

1. $\bar{\alpha}(c) = c^S$ dacă $c \in \mathcal{F}_0$ (i.e., c este un simbol constant);
2. $\bar{\alpha}(x) = \alpha(x)$ dacă $x \in \mathcal{X}$ (i.e., x este o variabilă);
3. $\bar{\alpha}(f(t_1, \dots, t_n)) = f^S(\bar{\alpha}(t_1), \dots, \bar{\alpha}(t_n))$ dacă $f \in \mathcal{F}_n$ este un simbol funcțional de aritate n , iar t_1, \dots, t_n sunt termeni.

Exemplul 46. Continuând Exemplul 43, unde α_1 este o S_1 -atribuire, avem:

$$\begin{aligned} \overline{\alpha_1}(f(i(x_1), e)) &= \overline{\alpha_1}(i(x_1)) + \overline{\alpha_1}(e) \\ &= -(\overline{\alpha_1}(x_1)) + e^{S_1} \\ &= -(\alpha_1(x_1)) + 0 \\ &= -5 + 0 \\ &= -5. \end{aligned}$$

Așadar, valoarea termenului $f(i(x_1), e)$ în atribuirea α_1 este -5 .

Observație. Intuiția este că atribuirile asociază variabilelor libere dintr-o formulă valori din domeniul structurii.

În punctul în care o variabilă este legată în formulă, valoarea variabilei este umbrată (engl. shadows) de o nouă valoare în domeniul de vizibilitate al variabilei legate. Această schimbare a valorii se face prin intermediul operației de actualizare a unei atribuiți, definită mai jos.

Definiția 47 (Actualizarea unei atribuiți). Dându-se o atribuire α , o variabilă $x \in \mathcal{X}$ și un element $u \in D$, notăm cu $\alpha[x \mapsto u]$ o nouă atribuire, care coincide cu α , exceptând valoarea variabilei x , care devine acum u :

$$\alpha[x \mapsto u] : \mathcal{X} \rightarrow D, \text{ a.î.}$$

1. $(\alpha[x \mapsto u])(x) = u$;
2. $(\alpha[x \mapsto u])(y) = \alpha(y)$, pentru orice $y \in \mathcal{X} \setminus \{x\}$.

Exemplul 48. De exemplu, atribuirea $\alpha_1[x_1 \mapsto 6]$ este exact atribuirea α_2 definită în exemplele de mai sus. Valoarea termenului $f(i(x_1), e)$ în atribuirea $\alpha_1[x_1 \mapsto 6]$, notată cu $\overline{\alpha_1[x_1 \mapsto 6]}(f(i(x_1), e))$, este -6 .

Exercițiul 49. Calculați valorile de mai jos:

1. $\overline{\alpha_1[x_1 \mapsto 10]}(f(i(x_1), e))$;
2. $\overline{\alpha_1[x_2 \mapsto 10]}(f(i(x_1), e))$;
3. $\overline{\alpha_1[x_2 \mapsto 10]}[x_1 \mapsto 10](f(i(x_1), e))$.

4.2 Valoarea de adevăr a unei formule de ordinul I

În acest moment avem ingredientele pentru a defini formal valoarea de adevăr a unei formule de ordinul I, construită peste o semnătură Σ . Această valoare se poate calcula doar într-o Σ -structură S , cu ajutorul unei S -atribuiți α .

Notațiile folosite sunt similare cu cele pentru logica propozițională. Astfel, notăm faptul că o formulă φ este adevărată într-o structură S cu o atribuire α prin $S, \alpha \models \varphi$. Faptul că o formulă φ nu este adevărată într-o structură S cu o atribuire α se notează cu $S, \alpha \not\models \varphi$.

Notația $S, \alpha \models \varphi$ se mai citește *S satisface φ cu atribuirea α* , iar $S, \alpha \not\models \varphi$ se mai citește *S nu satisface φ cu atribuirea α* .

Definiția 50. *Faptul că o structură S satisface o formulă φ cu o anumită atribuire α (echivalent, φ este adevărată în structura S cu atribuirea α) se definește inductiv astfel (prima linie din enumerarea care urmează desemnează cazul de bază, restul reprezentând cazurile inductive):*

1. $S, \alpha \models P(t_1, \dots, t_n)$ ddacă $P^S(\bar{\alpha}(t_1), \dots, \bar{\alpha}(t_n))$;
2. $S, \alpha \models \neg\varphi$ ddacă $S, \alpha \not\models \varphi$;
3. $S, \alpha \models (\varphi_1 \wedge \varphi_2)$ ddacă $S, \alpha \models \varphi_1$ și $S, \alpha \models \varphi_2$;
4. $S, \alpha \models (\varphi_1 \vee \varphi_2)$ ddacă $S, \alpha \models \varphi_1$ sau $S, \alpha \models \varphi_2$;
5. $S, \alpha \models (\varphi_1 \rightarrow \varphi_2)$ ddacă $S, \alpha \not\models \varphi_1$ sau $S, \alpha \models \varphi_2$;
6. $S, \alpha \models (\varphi_1 \leftrightarrow \varphi_2)$ ddacă (1) atât $S, \alpha \models \varphi_1$, cât și $S, \alpha \models \varphi_2$, sau (2) $S, \alpha \not\models \varphi_1$ și $S, \alpha \not\models \varphi_2$;
7. $S, \alpha \models (\exists x.\varphi)$ ddacă există $u \in D$ astfel încât $S, \alpha[x \mapsto u] \models \varphi$;
8. $S, \alpha \models (\forall x.\varphi)$ ddacă pentru orice $u \in D$, avem că $S, \alpha[x \mapsto u] \models \varphi$.

Exemplul 51. *Vom lucra în continuare peste signatura $\Sigma = (\{\mathbf{P}\}, \{\mathbf{f}, \mathbf{i}, \mathbf{e}\})$, Σ -structura $S_1 = (\mathbb{Z}, \{=\}, \{+, -, 0\})$ definită la începutul capitoului și S_1 -atribuirile α_1, α_2 .*

Avem că

$$\begin{array}{ll}
 S_1, \alpha_1 \models \mathbf{P}(\mathbf{x}_1, \mathbf{x}_1) & \text{ddacă} \quad \mathbf{P}^{S_1}(\bar{\alpha}_1(\mathbf{x}_1), \bar{\alpha}_1(\mathbf{x}_1)) \\
 & \text{ddacă} \quad \bar{\alpha}_1(\mathbf{x}_1) = \bar{\alpha}_1(\mathbf{x}_1) \\
 & \text{ddacă} \quad \alpha_1(\mathbf{x}_1) = \alpha_1(\mathbf{x}_1) \\
 & \text{ddacă} \quad 5 = 5.
 \end{array}$$

Din moment ce $5 = 5$, rezultă că $S_1, \alpha_1 \models \mathbf{P}(\mathbf{x}_1, \mathbf{x}_1)$, adică formula $\mathbf{P}(\mathbf{x}_1, \mathbf{x}_1)$ este adevărată în structura S_1 cu atribuirea α_1 . Altfel spus, S_1 satisface $\mathbf{P}(\mathbf{x}_1, \mathbf{x}_1)$ cu atribuirea α_1 .

Exemplul 52. Continuând exemplul anterior, avem

$$\begin{array}{ll}
 S_1, \alpha_1 \models P(x_1, x_3) & \text{dacă} \quad P^{S_1}(\overline{\alpha_1}(x_1), \overline{\alpha_1}(x_3)) \\
 & \text{dacă} \quad \overline{\alpha_1}(x_1) = \overline{\alpha_1}(x_3) \\
 & \text{dacă} \quad \alpha_1(x_1) = \alpha_1(x_3) \\
 & \text{dacă} \quad 5 = 6.
 \end{array}$$

Din moment ce $5 \neq 6$, rezultă că $S_1, \alpha_1 \not\models P(x_1, x_3)$, adică formula $P(x_1, x_3)$ este falsă în structura S_1 cu atribuirea α_1 . Altfel spus S_1 nu satisface $P(x_1, x_3)$ cu atribuirea α_1 .

Exemplul 53. Continuând exemplul anterior, avem

$$\begin{array}{ll}
 S_1, \alpha_1 \models \neg P(x_1, x_3) & \text{dacă} \quad S_1, \alpha_1 \not\models P(x_1, x_3) \\
 & \text{dacă} \quad \text{nu } P^{S_1}(\overline{\alpha_1}(x_1), \overline{\alpha_1}(x_3)) \\
 & \text{dacă} \quad \text{nu } \overline{\alpha_1}(x_1) = \overline{\alpha_1}(x_3) \\
 & \text{dacă} \quad \overline{\alpha_1}(x_1) \neq \overline{\alpha_1}(x_3) \\
 & \text{dacă} \quad \alpha_1(x_1) \neq \alpha_1(x_3) \\
 & \text{dacă} \quad 5 \neq 6.
 \end{array}$$

Din moment ce $5 \neq 6$, rezultă că $S_1, \alpha_1 \models \neg P(x_1, x_3)$, adică formula $\neg P(x_1, x_3)$ este adevărată în structura S_1 cu atribuirea α_1 . Altfel spus, S_1 satisface $\neg P(x_1, x_3)$ cu atribuirea α_1 .

Exemplul 54. Continuând exemplul anterior, avem

$$\begin{array}{ll}
 S_1, \alpha_1 \models P(x_1, x_1) \wedge \neg P(x_1, x_3) & \text{dacă} \\
 S_1, \alpha_1 \models P(x_1, x_1) \text{ și } S_1, \alpha_1 \models \neg P(x_1, x_3) & \text{dacă} \\
 \dots \text{ și } \dots & \text{dacă} \\
 5 = 5 \text{ și } 5 \neq 6. &
 \end{array}$$

Din moment ce $5 = 5$ și $5 \neq 6$, rezultă că $S_1, \alpha_1 \models P(x_1, x_1) \wedge \neg P(x_1, x_3)$.

Exemplul 55. Continuând exemplul anterior, avem

$$S_1, \alpha_1 \models P(x_1, x_3) \vee P(x_1, x_1) \text{ dacă } S_1, \alpha_1 \models P(x_1, x_3) \text{ sau } S_1, \alpha_1 \models P(x_1, x_1).$$

Am stabilit deja că $S_1, \alpha_1 \models P(x_1, x_3)$, deci $S_1, \alpha_1 \models P(x_1, x_3) \vee P(x_1, x_1)$ (chiar dacă $S_1, \alpha_1 \not\models P(x_1, x_1)$).

Exemplul 56. Continuând exemplul anterior, avem

$S_1, \alpha_1 \models \exists x_1. P(x_1, x_3)$	ddacă
există $u \in D$ a.î. $S_1, \alpha_1[x_1 \mapsto u] \models P(x_1, x_3)$	ddacă
există $u \in D$ a.î. $P^{S_1}(\overline{\alpha_1[x_1 \mapsto u]}(x_1), \overline{\alpha_1[x_1 \mapsto u]}(x_3))$	ddacă
există $u \in D$ a.î. $\overline{\alpha_1[x_1 \mapsto u]}(x_1) = \overline{\alpha_1[x_1 \mapsto u]}(x_3)$	ddacă
există $u \in D$ a.î. $\alpha_1[x_1 \mapsto u](x_1) = \alpha_1[x_1 \mapsto u](x_3)$	ddacă
există $u \in D$ a.î. $u = \alpha_1(x_3)$	ddacă
există $u \in D$ a.î. $u = 6$.	

Din moment ce există u (putem alege $u = 6$) a.î. $u = 6$, avem că $S_1, \alpha_1 \models \exists x_1. P(x_1, x_3)$.

Exemplul 57. Continuând exemplul anterior, avem

$S_1, \alpha_1 \models \forall x_1. \exists x_3. P(x_1, x_3)$	ddacă
pentru orice $u \in D$, avem că $S_1, \alpha_1[x_1 \mapsto u] \models \exists x_3. P(x_1, x_3)$	ddacă
pt. orice $u \in D$, există $v \in D$ a.î. $S_1, \alpha_1[x_1 \mapsto u][x_3 \mapsto v] \models P(x_1, x_3)$	ddacă
...	ddacă
pentru orice $u \in D$, avem că există $v \in D$ a.î. $u = v$.	

Din moment ce pentru orice număr întreg u , există un număr întreg v a.î. $u = v$, avem că $S_1, \alpha_1 \models \forall x_1. \exists x_3. P(x_1, x_3)$.

Exercițiul 58. Arătați că $S_1, \alpha_1 \models \forall x_1. \exists x_3. P(x_1, i(x_3))$.

4.3 Satisfiabilitate într-o structură fixată

Definiția 59 (Satisfiabilitate într-o structură fixată). O formulă φ este satisfiabilă într-o structură S dacă există o S -atribuire α cu proprietatea că

$$S, \alpha \models \varphi.$$

Exemplul 60. Formula $P(x_1, x_3)$ este satisfiabilă în structura S_1 , deoarece există o atribuire, de exemplu α_2 , cu proprietatea că $S_1, \alpha_2 \models P(x_1, x_3)$.

Exercițiul 61. Arătați că formula $\neg P(x_1, x_1)$ nu este satisfiabilă în structura S_1 (deoarece, pentru orice atribuire α aleasă, avem că $S_1, \alpha \not\models \neg P(x_1, x_1)$).

4.4 Validitate într-o structură fixată

Definiția 62 (Validitate într-o structură fixată). *O formulă φ este validă într-o structură S dacă pentru orice S -atribuire α , avem că*

$$S, \alpha \models \varphi.$$

Exemplul 63. *Formula $P(x_1, x_3)$ nu este validă în structura S_1 , deoarece există o atribuire, și anume α_1 , cu proprietatea că $S_1, \alpha_1 \not\models P(x_1, x_3)$.*

Exercițiul 64. *Arătați că formula $P(x_1, x_1)$ este validă în structura S_1 (deoarece orice atribuire α am alege, $S_1, \alpha \models P(x_1, x_1)$).*

4.5 Satisfiabilitate

Definiția 65 (Satisfiabilitate). *O formulă φ este satisfiabilă dacă există o structură S și o S -atribuire α cu proprietatea că*

$$S, \alpha \models \varphi.$$

Exemplul 66. *Formula $\neg P(x_1, x_1)$ este satisfiabilă, deoarece există o structură (de exemplu S_5) și o S_5 -atribuire (de exemplu α_1) astfel încât $S_5, \alpha_1 \models \neg P(x_1, x_1)$ (deoarece $5 \not\prec 5$).*

Să subliniem faptul că, deoarece S_5 și S_1 au același domeniu, atribuirea α_1 este atât o S_1 -atribuire cât și o S_5 -atribuire.

Observație. *O formulă poate să nu fie satisfiabilă într-o structură fixată (de exemplu $\neg P(x_1, x_1)$ nu este satisfiabilă în structura S_1) și totuși să fie satisfiabilă (vezi Exemplul 66 unde aceeași formulă $\neg P(x_1, x_1)$ este satisfiabilă).*

4.6 Validitate

Definiția 67 (Validitate). *O formulă φ este validă dacă, pentru orice structură S și pentru orice S -atribuire α , avem*

$$S, \alpha \models \varphi.$$

Exemplul 68. *Formula $P(x_1, x_1)$ nu este validă, deoarece $S_5, \alpha_1 \not\models P(x_1, x_1)$. Pe de altă parte, formula $P(x_1, x_1) \rightarrow P(x_1, x_1)$ este validă.*

Observație. *O formulă poate să fie validă într-o structură fixată (de exemplu $P(x_1, x_1)$ este validă în structura S_1) și totuși să nu fie validă (de exemplu, $P(x_1, x_1)$ nu este validă, deoarece $S_5, \alpha_1 \not\models P(x_1, x_1)$).*

4.7 Consecință semantică

Definiția 69. O formulă φ este consecință semantică a formulelor $\varphi_1, \dots, \varphi_n$ într-o structură fixată S , notat $\varphi_1, \dots, \varphi_n \models_S \varphi$, dacă, pentru orice S -atribuire α pentru care $S, \alpha \models \varphi_1$, $S, \alpha \models \varphi_2$, \dots , $S, \alpha \models \varphi_n$, avem că $S, \alpha \models \varphi$.

Exemplul 70. Avem că $P(x, y) \models_{S_1} P(y, x)$, deoarece, pentru orice S_1 -atribuire α cu proprietatea că $S_1, \alpha \models P(x, y)$ (adică $\alpha(x) = \alpha(y)$), avem și că $S_1, \alpha \models P(y, x)$ (adică $\alpha(y) = \alpha(x)$).

Avem că $P(x, y) \not\models_{S_5} P(y, x)$, deoarece, pentru atribuirea $\alpha(x) = 5$, $\alpha(y) = 6$, avem că $S_5, \alpha \models P(x, y)$ (adică $5 < 6$), dar $S_5, \alpha \not\models P(y, x)$ ($6 \not< 5$).

Definiția 71. O formulă φ este consecință semantică a formulelor $\varphi_1, \dots, \varphi_n$, notat $\varphi_1, \dots, \varphi_n \models \varphi$, dacă

$$\varphi_1, \dots, \varphi_n \models_S \varphi$$

pentru orice structură S .

Exemplul 72. Avem că $P(x, y) \not\models P(y, x)$, deoarece există o structură (și anume S_5) astfel încât $P(x, y) \models_{S_5} P(y, x)$.

Exercițiul 73. Arătați că:

$$\forall x. \forall y. \forall z. (P(x, y) \wedge P(y, z) \rightarrow P(x, z)), P(x_1, x_2), P(x_2, x_3) \models P(x_1, x_3).$$

Desigur că, în cele de mai sus (similar cu logica propozițională), lista $\varphi_1, \varphi_2, \dots, \varphi_n$ denotă de fapt o mulțime având respectivele elemente.

4.8 Mulțime consistentă de formule

Definiția 74. O mulțime de formule Γ este consistentă într-o structură fixată S dacă există o S -atribuire α cu proprietatea că $S, \alpha \models \varphi$ pentru fiecare formulă $\varphi \in \Gamma$.

Exemplul 75. Avem că $\Gamma = \{P(x_1, x_2), P(x_2, x_1)\}$ este consistentă în S_1 : S_1 face ambele formule adevărate cu atribuirea α_1 .

Aceeași mulțime de formule, $\Gamma = \{P(x_1, x_2), P(x_2, x_1)\}$, nu este consistentă în S_5 : presupunând prin reducere la absurd că ar exista o atribuire α cu care S_5 să facă ambele formule din Γ adevărate, am avea $\alpha(x_1) < \alpha(x_2) < \alpha(x_1)$, ceea ce este imposibil.

Definiția 76. O mulțime de formule Γ este consistentă dacă există o structură S și o S -atribuire α cu proprietatea că $S, \alpha \models \varphi$ pentru fiecare formulă $\varphi \in \Gamma$.

Exemplul 77. *Avem că $\Gamma = \{P(x_1, x_2), P(x_2, x_1)\}$ este consistentă, deoarece este consistentă în S_1 .*

Avem că $\Gamma' = \{P(x_1, x_2), \neg P(x_1, x_2)\}$ nu este consistentă.

Exercițiul 78. *Arătați că Γ' este inconsistentă.*

4.9 Formule echivalente

În diverse contexte, anumite formule pot avea același înțeles. De exemplu, formulele $(\forall x.P(x, x))$ și $(\forall y.P(y, y))$ au același înțeles în orice context. Un alt exemplu de formule cu același înțeles este $\neg(\forall x.Q(x))$ și $(\exists x.\neg Q(x))$. Vom numi astfel de formule *echivalente*.

Anumite formule au același înțeles doar pentru o anumită interpretare a simbolurilor predicative și funcționale. De exemplu, dacă lucrăm într-o structură în care simbolul predicativ P este interpretat printr-un predicat simetric, formulele $P(x, y)$ și respectiv $P(y, x)$ au același înțeles. Astfel de formule se numesc *echivalente în structura respectivă*.

Acest aspect este surprins de următoarele definiții.

Definiția 79. *Două formule $\varphi_1 \in \mathbb{LPI}$ și $\varphi_2 \in \mathbb{LPI}$ sunt echivalente în structura S dacă, pentru orice S -atribuire α ,*

$$S, \alpha \models \varphi_1 \text{ dacă } S, \alpha \models \varphi_2.$$

Faptul că φ_1 și φ_2 sunt echivalente în structura S se notează $\varphi_1 \stackrel{S}{\equiv} \varphi_2$.

Cu alte cuvinte, două formule sunt echivalente într-o anumită structură S dacă, evaluând valoarea de adevăr a formulelor în structura S , obținem același rezultat pentru ambele formule (ambele adevărate sau ambele false), indiferent de atribuirea α cu care lucrăm.

Exemplul 80. *Continuăm exemplele din cursurile anterioare. Considerăm semnatura $\Sigma = (\{P\}, \{f, i, e\})$ și Σ -structura $S_1 = (\mathbb{Z}, \{=\}, \{+, -, 0\})$.*

1. *Avem că $P(x, y) \stackrel{S_1}{\equiv} P(y, x)$. De ce?*

Fie α o S_1 -atribuire oarecare.

Avem $S_1, \alpha \models P(x, y)$ dacă (prin definiția relației \models) $P^{S_1}(\bar{\alpha}(x), \bar{\alpha}(y))$

ddacă $\bar{\alpha}(x) = \bar{\alpha}(y)$

ddacă $\alpha(x) = \alpha(y)$

ddacă (prin simetria relației de egalitate) $\alpha(y) = \alpha(x)$

ddacă $\bar{\alpha}(y) = \bar{\alpha}(x)$

ddacă (prin definiția relației \models) $S_1, \alpha \models P(y, x)$.

Deci, pentru orice S_1 -atribuire α , avem: $S_1, \alpha \models P(x, y)$ dacă $S_1, \alpha \models P(y, x)$, care este chiar definiția $P(x, y) \stackrel{S_1}{\equiv} P(y, x)$.

2. Avem că $P(x_1, x_3) \not\stackrel{S_1}{\equiv} P(x_2, x_3)$. De ce?

Deoarece există o S_1 -atribuire $\alpha : \mathcal{X} \rightarrow \mathbb{Z}$, definită prin $\alpha(x_1) = 42, \alpha(x_2) = 7, \alpha(x_3) = 42$ (pentru restul variabilelor nu este relevantă valoarea lor în atribuire) cu proprietatea că

$$\begin{aligned} S_1, \alpha &\models P(x_1, x_3) \text{ (deoarece } 42 = 42), \text{ dar} \\ S_1, \alpha &\not\models P(x_2, x_3) \text{ (deoarece } 42 \neq 7). \end{aligned}$$

În cazul în care structura nu este fixată, avem următoarea definiție:

Definiția 81. Două formule $\varphi_1 \in \text{LPI}$ și $\varphi_2 \in \text{LPI}$ sunt echivalente dacă, pentru orice structură S și pentru orice S -atribuire α ,

$$S, \alpha \models \varphi_1 \text{ dacă } S, \alpha \models \varphi_2.$$

Faptul că φ_1 și φ_2 sunt echivalente se notează $\varphi_1 \equiv \varphi_2$.

Exemplul 82. Continuăm exemplul anterior. Avem că $P(x, y) \not\equiv P(y, x)$. De ce?

Deoarece există o Σ -structură și o atribuire în structura respectivă astfel încât cele două formule să ia valori de adevăr diferite.

Fie structura $S_5 = (\mathbb{Z}, \{<\}, \{+, -, 0\})$ definită în cursul anterior și S_5 -atribuirea $\alpha_6 : \mathcal{X} \rightarrow \mathbb{Z}$, definită prin $\alpha_6(x) = 2, \alpha_6(y) = 3$ și $\alpha_6(z) = 1$ pentru orice $z \in \mathcal{X} \setminus \{x, y\}$.

Observați că singura diferență între S_1 și S_5 este faptul că simbolul predicativ P este interpretat prin predicatul $=$ în S_1 , în timp ce în S_5 este interpretat prin $<$ (relația mai mic strict peste numere întregi).

Avem $S_5, \alpha_6 \models P(x, y)$, deoarece $2 < 3$, dar $S_5, \alpha_6 \not\models P(y, x)$, deoarece $3 \not< 2$. Deci formulele $P(x, y)$ și $P(y, x)$ nu sunt echivalente (chiar dacă sunt echivalente în structura S_1).

Exemplul 83. Avem că $(\forall x.P(x, z)) \equiv (\forall y.P(y, z))$. De ce?

Fie S o Σ -structură oarecare cu domeniul D și $\alpha : \mathcal{X} \rightarrow D$ o S -atribuire oarecare. Avem că

$$\begin{aligned} S, \alpha &\models (\forall x.P(x, z)) && \text{dacă} \\ \text{pentru orice } u \in D, \text{ avem } S, \alpha[x \mapsto u] &\models P(x, z) && \text{dacă} \\ \text{pentru orice } u \in D, \text{ avem } P^S \left(\overline{\alpha[x \mapsto u]}(x), \overline{\alpha[x \mapsto u]}(z) \right) && \text{dacă} \\ \text{pentru orice } u \in D, \text{ avem } P^S \left(u, \alpha(z) \right) && \text{dacă} \\ \text{pentru orice } u \in D, \text{ avem } P^S \left(\overline{\alpha[y \mapsto u]}(y), \overline{\alpha[y \mapsto u]}(z) \right) && \text{dacă} \\ \text{pentru orice } u \in D, \text{ avem } S, \alpha[y \mapsto u] &\models P(y, z) && \text{dacă} \\ S, \alpha &\models (\forall y.P(y, z)). \end{aligned}$$

Deci, pentru orice Σ -structură S , pentru orice S -atribuire α , avem că

$$S, \alpha \models (\forall x.P(x, z)) \text{ dacă } S, \alpha \models (\forall y.P(y, z)),$$

care este chiar definiția faptului că $(\forall x.P(x, z)) \equiv (\forall y.P(y, z))$.

4.10 Fișă de exerciții

Amintim mai jos structurile din Exemplul 41:

1. $S_1 = (\mathbb{Z}, \{=\}, \{+, -, 0\})$;
2. $S_2 = (\mathbb{R}^*, \{=\}, \{\times, \cdot^{-1}, 1\})$;
3. $S_3 = (\mathbb{N}, \{=\}, \{+, s, 0\})$;
4. $S_4 = (\mathbb{N}, \{<\}, \{+, s, 0\})$.
5. $S_5 = (\mathbb{Z}, \{<\}, \{+, -, 0\})$.

Aceste structuri vor fi utilizate în exercițiile de mai jos.

Exercițiul 84. *Stabiliți dacă:*

1. $S_1, \alpha_1 \models P(x_2, x_3)$;
2. $S_1, \alpha_1 \models \neg P(x_2, x_3)$;
3. $S_1, \alpha_1 \models \neg P(x_2, x_3) \wedge P(x_1, x_1)$;
4. $S_1, \alpha_1 \models \exists x_3.P(x_2, x_3)$;
5. $S_1, \alpha_1 \models \forall x_2.\exists x_3.P(x_2, x_3)$;
6. $S_1, \alpha_1 \models \exists x_3.\forall x_2.P(x_2, x_3)$;
7. $S_1, \alpha_2 \models \forall x_2.\exists x_3.P(x_2, i(x_3))$;

Exercițiul 85. *Găsiți pentru fiecare dintre itemii de mai jos câte o S_2 -atribuire α_3 astfel încât:*

1. $S_2, \alpha_3 \models P(x_1, x_2)$;
2. $S_2, \alpha_3 \models P(f(x_1, x_2), x_3)$;
3. $S_2, \alpha_3 \models P(f(x_1, x_2), i(x_3))$;
4. $S_2, \alpha_3 \models P(x, e)$;

$$5. S_2, \alpha_3 \models \exists y.P(x, i(y));$$

$$6. S_2, \alpha_3 \models \forall y.\exists x.P(x, i(y)).$$

Exercițiul 86. Arătați că următoarele formule sunt valide în S_2 :

$$1. \forall x.\exists y.P(x, i(y));$$

$$2. \forall x.P(f(x, e), x);$$

$$3. \forall x.P(x, i(i(x))).$$

Exercițiul 87. Arătați că formula $\forall x.\exists y.P(x, i(y))$ nu este validă în S_3 .

Exercițiul 88. Găsiți o formulă care să fie satisfiabilă în S_1 dar nu în S_3 .

Exercițiul 89. Găsiți o formulă fără variabile libere care să fie satisfiabilă în S_5 dar nu în S_4 .

Exercițiul 90. Arătați că formula $\forall x.\exists y.P(x, y)$ nu este validă.

Exercițiul 91. Arătați că formula $(\forall x.P(x, x)) \rightarrow \exists x_2.P(x_1, x_2)$ este validă.

Exercițiul 92. Arătați că formula $\forall x.\exists y.P(y, x)$ nu este validă.

Exercițiul 93. Arătați că formula $\forall x.\neg P(x, x)$ este satisfiabilă.

Exercițiul 94. Arătați că formula $\forall x.\neg P(x, x) \wedge \exists x.P(x, x)$ nu este satisfiabilă.

Exercițiul 95. În Exercițiul 40 (din Capitolul 3) am arătat că \mathbb{LPI} este o extensie sintactică a lui \mathbb{LP} . Pe scurt, dacă $A = \{p, q, r, \dots\}$ este o mulțime de variabile propoziționale atunci construim o semnatura $\Sigma_{\mathbb{LP}} = \{A, \emptyset\}$, unde variabilele propoziționale din A sunt simboluri predicative de aritate 0.

Semantica formulelor \mathbb{LPI} construite peste semnatura $\Sigma_{\mathbb{LP}}$ este consistentă cu semantica formulelor \mathbb{LP} . Fie $\tau : A \rightarrow B$ o atribuire. Fie $S = (D, \{a^S \mid a \in A\}, \emptyset)$ o $\Sigma_{\mathbb{LP}}$ -structură, unde D este orice mulțime nenulă și $a^S = \tau(a)$, pentru orice $a \in A$.

Demonstrați că pentru orice $\varphi \in \mathbb{LP}$, avem că $\tau \models \varphi$ dacă și numai dacă $S, \alpha \models \varphi$ pentru orice S -atribuire $\alpha : \mathcal{X} \rightarrow D$.