

Logică pentru informatică - Săptămâna 9

Sintaxa logicii de ordinul I

January 7, 2019

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, dar care apar în practică.

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 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 din logica propozițională: *și*, *sau*, *non*, *implică* și respectiv *dacă* și *numai dacă*.

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 muritor*. 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 ca 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 LP, 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: *Om* și *Muritor*. Predicatul *Om* va fi predicatul care denotă umanitatea: $Om(x)$ va fi adevărat când x este om. Predicatul *Muritor* este adevărat când argumentul său 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.

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

$$\forall x.(Om(x) \rightarrow Muritor(x)),$$

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

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

Vom vedea că în logica de ordinul I, formula $Muritor(s)$ este consecință a formulelor $\forall x.(Om(x) \rightarrow Muritor(x))$ și respectiv $Om(s)$. 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*.

2 Mulțimi. Relații. Funcții.

În această secțiune amintim câteva noțiuni de bază pe care le vom utiliza în secțiunile următoare.

Mulțimi. Conceptul de *mulțime* este probabil unul dintre cele mai importante și utilizate în matematică. Formal, noțiunea de mulțime a fost introdusă ca fiind o colecție de obiecte *bine determinate* și *distincte*, colecție în care fiecare obiect apare o singură dată. Aceste obiecte ce constituie mulțimea se mai numesc *elementele* mulțimii.

Există mai multe moduri pentru a specifica o mulțime. O modalitate este enumerarea elementelor mulțimii. Spre exemplu, $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ va reprezenta mulțimea cifrelor (din baza 10). Prescurtat, putem utiliza și notația $\{0, \dots, 9\}$ pentru mulțimea cifrelor. Cealaltă modalitate de a reprezenta o mulțime este să utilizăm proprietăți care caracterizează elementele mulțimii. O formă generală a acestui tip de reprezentare este

$$\{x \mid x \text{ are proprietatea } P\}.$$

Spre exemplu, mulțimea cifrelor mai poate fi scrisă astfel:

$$\{n \mid 0 \leq n < 10, n \text{ este număr natural}\}.$$

Mulțimile se notează de obicei cu litere mari (de exemplu: A , X , etc.), eventual cu litere speciale pentru mulțimile cunoscute (\mathbb{N} , \mathbb{Z} , \mathbb{Q} , etc.). În general, elementele mulțimilor se notează cu litere mici (de exemplu: a , x , n , z etc.). Pentru a exprima faptul că un obiect x este element al mulțimii X este utilizată notația $x \in X$. Spre exemplu, avem că $9 \in \{n \mid 0 \leq n < 10\}$. Atunci când un element x nu aparține unei mulțimi A vom utiliza notația $x \notin A$. Spre exemplu, $10 \notin \{n \mid 0 \leq n < 10\}$. Vom spune ca o mulțime A este o *submulțime* a mulțimii B , și vom nota $A \subseteq B$, dacă pentru fiecare element $a \in A$ avem $a \in B$.

Mulțimile pot avea ca elemente alte mulțimi: $\{\{0\}, \{1\}, \{2\}, \{0,1\}\}$. O mulțime foarte utilizată care are ca elemente alte mulțimi este chiar mulțimea submulțimilor unei mulțimi. Pentru o mulțime X mulțimea submulțimilor lui X se notează cu 2^X și conține ca elemente toate submulțimile lui X .

Printre operațiile peste mulțimi se numără *reuniunea* (\cup), *intersecția* (\cap), *diferența* (\setminus), *diferența simetrică* (Δ), *produsul cartezian* (\times) și altele. Produsul cartezian a două mulțimi A și B este definit astfel:

$$A \times B = \{(a, b) \mid a \in A \text{ și } b \in B\},$$

unde (a, b) este o pereche ordonată de elemente.

Exemplul 2.1. Pentru mulțimile $A = \{1, 2, 3\}$ și $B = \{\alpha, \beta\}$ produsul cartezian este: $A \times B = \{(1, \alpha), (1, \beta), (2, \alpha), (2, \beta), (3, \alpha), (3, \beta)\}$.

Relații binare. Fie A și B două mulțimi nevide. Prin definiție, o *relație binară* R de la A la B este o submulțime a produsului cartezian $A \times B$, adică $R \subseteq A \times B$. Dacă pentru un element $(a, b) \in A \times B$ avem $(a, b) \in R$ atunci spunem că a este în relația R cu b și notăm aRb .

Exemplul 2.2. Un exemplu foarte cunoscut de relație binară este relația "mai mic sau egal": $\leq \subseteq \mathbb{N} \times \mathbb{N}$. Perechea $(2, 3) \in \leq$ pe când perechea $(3, 2) \notin \leq$. De obicei utilizăm notația înfixată, adică $2 \leq 3$, iar pentru cazul în care $(3, 2) \notin \leq$ mai scriem și $3 \not\leq 2$.

Pentru o relație $R \subseteq A \times B$ vom numi *domeniul* relației R mulțimea:

$$\{a \in A \mid \text{există } b \in B \text{ astfel încât } (a, b) \in R\}$$

și respectiv *codomeniul* relației R mulțimea:

$$\{b \in B \mid \text{există } a \in A \text{ astfel încât } (a, b) \in R\}.$$

Exemplul 2.3. Pentru mulțimile $A = \{1, 2, 3\}$ și $B = \{\alpha, \beta\}$, mulțimea $R_1 = \{(1, \alpha), (1, \beta), (2, \alpha), (3, \alpha)\}$ este o relație binară. Domeniul relației R_1 este $\{1, 2, 3\}$ iar codomeniul acesteia este $\{\alpha, \beta\}$. Pentru relația $R_2 = \{(1, \alpha), (2, \alpha)\}$, domeniul este $\{1, 2\}$, iar codomeniul este $\{\alpha\}$.

Funcții. *Funcțiile* sunt cazuri particulare de relații. O relație binară $F \subseteq A \times B$ este o *funcție* dacă respectă următoarele două proprietăți:

- domeniul lui F este mulțimea A ;
- pentru orice $a \in A$ și pentru orice $b, b' \in B$ astfel încât $(a, b) \in F$ și $(a, b') \in F$ avem $b = b'$.

Prima proprietate exprimă faptul că pentru fiecare element a din A vom avea un element în B cu care a să fie în relația F . Cea de-a doua proprietate se asigură că pentru fiecare element a din A vom avea un singur element în B cu care a să fie în relația F .

Exemplul 2.4. Fie relația $R_1 = \{(1, \alpha), (1, \beta), (2, \alpha), (3, \alpha)\}$ peste mulțimile $A = \{1, 2, 3\}$ și $B = \{\alpha, \beta\}$. Relația R_1 nu este funcție deoarece ce-a doua proprietate nu este îndeplinită: elementului $1 \in A$ îi sunt asociate două elemente distincte α și β din B .

Nici relația $R_2 = \{(1, \alpha), (2, \alpha)\}$ nu este o funcție, deoarece domeniul funcției nu este A , ci doar o submulțime a lui A .

În schimb, relația $F_1 = \{(1, \alpha), (2, \alpha), (3, \beta)\}$ este o funcție. Ambele condiții de mai sus sunt îndeplinite. Atenție: chiar dacă pentru 1 și 2 asociem același element α , asta nu înseamnă că cea de-a doua proprietate nu este îndeplinită!

În general, dacă o relație binară F este o funcție și $(a, b) \in F$, vom scrie $F(a)$ în loc de b (cu alte cuvinte, $F(a) = b$ dacă și numai dacă $(a, b) \in F$).

Produs cartezian peste n mulțimi. Aritatea relațiilor și funcțiilor. Definiția produsului cartezian pe care am amintit-o mai sus poate fi generalizată la mai multe mulțimi. Fie n mulțimi pe care le notăm A_1, A_2, \dots, A_n . Definim produsul lor cartezian astfel:

$$A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) \mid a_i \in A_i, i \in \{1, \dots, n\}\},$$

unde (a_1, a_2, \dots, a_n) este numit *n -tuplu* sau mai simplu *tuplu*. Prin *n -tuplu* înțelegem o secvență ordonată de n elemente.

Dacă mulțimile A_1, A_2, \dots, A_n sunt egale, adică $A_1 = A_2 = \dots = A_n = A$, atunci notăm cu A^n produsul cartezian:

$$\underbrace{A \times \dots \times A}_{\text{de } n \text{ ori}}.$$

Deoarece am definit produsul cartezian peste mai multe mulțimi, putem în mod natural să generalizăm și noțiunea de relație binară. Astfel, o relație (nu neapărat binară) peste mulțimile A_1, A_2, \dots, A_n este o submulțime a produsului cartezian $A_1 \times A_2 \times \dots \times A_n$. În acest caz, vom spune că *aritatea* relației este n , deoarece fiecare element al acestei relații este un *n -tuplu*. Pentru cazul particular când o relație este definită ca submulțime a lui A^n , vom ști că relația are aritate n . Relația \leq din Exemplul 2.2 este submulțime a produsului cartezian \mathbb{N}^2 și are aritate 2.

Exemplul 2.5. Fie mulțimile $A = \{1, 2, 3\}$, $B = \{\alpha, \beta\}$ și $C = \{true, false\}$. Mulțimea $R = \{(1, \alpha, true), (1, \beta, true), (1, \alpha, false), (1, \beta, false), (2, \alpha, true), (2, \beta, true), (3, \alpha, false), (3, \beta, false)\}$ este o relație peste mulțimile A , B și C . Această relație mai este numită și relație ternară deoarece fiecare element al relației este o secvență cu 3 elemente - triplet. Mai general, relațiile ale căror elemente sunt secvențe de n elemente se mai numesc n -are.

Fie $R \in A_1 \times A_2 \times \dots \times A_n$. Domeniul lui R este definit astfel:

$$\{(a_1, \dots, a_{n-1}) \in A_1 \times \dots \times A_{n-1} \mid \text{există } a_n \in A_n \text{ astfel încât } (a_1, \dots, a_{n-1}, a_n) \in A_1 \times \dots \times A_{n-1} \times A_n\},$$

iar codomeniul este definit astfel:

$$\{a_n \in A_n \mid \text{există } (a_1, \dots, a_{n-1}) \in A_1 \times \dots \times A_{n-1} \text{ astfel încât } (a_1, \dots, a_{n-1}, a_n) \in A_1 \times \dots \times A_{n-1} \times A_n\}.$$

Generalizarea relațiilor ne permite să definim funcții cu mai multe argumente. Practic, o relație F peste mulțimile A_1, A_2, \dots, A_n este o funcție dacă:

- domeniul lui F este $A_1 \times A_2 \times \dots \times A_{n-1}$ și
- fiecărui element $(a_1, a_2, \dots, a_{n-1})$ din domeniu, F îi asociază un singur element din A_n .

Pentru funcții, în loc de $(a_1, \dots, a_n) \in F$ utilizăm notația $F((a_1, \dots, a_{n-1})) = a_n$. În acest caz, vom spune ca F are aritatea $n - 1$ deoarece ea are $n - 1$ argumente. Totuși, dacă este privită ca o relație, F are aritate n .

Prin convenție, constantele sunt considerate funcții de aritate 0.

Exemplul 2.6. Fie mulțimile A, B, C și relația R din Exemplul 2.5. Relația R nu este o funcție. Deși domeniul lui R este chiar mulțimea $A \times B$, pentru elementul $(1, \alpha) \in A \times B$ relația R asociază atât $true$ cât și $false$ din C .

Pe de altă parte, relația $F' = \{(1, \alpha, true), (1, \beta, true), (2, \alpha, true), (2, \beta, true), (3, \alpha, false), (3, \beta, false)\}$ este o funcție.

Observația 2.1.

Deoarece funcțiile sunt relații, iar relațiile sunt mulțimi, în această secțiune am notat funcțiile cu litere mari (de ex. F, F'). În secțiunile următoare vom diferenția funcțiile de relații prin folosirea literelor mici: f, g, h, f', f_1, f_n , etc. pentru funcții și a literelor mari pentru relații în general.

3 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:

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

Formula folosește un predicat binar (adică o relație binară), $<$, care este 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 3.1. 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 trei 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. $(B, \emptyset, \{\cdot, +, \neg\})$;

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.

Revenind la formula de mai devreme:

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

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).

Când avem o formulă de ordinul I și dorim să îi evaluăm valoarea de adevăr, trebuie să fixăm structura în care lucrăm.

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 între 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 lucra cu semnături:

Definiția 3.2. O semnătură 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 3.3. Dacă $\Sigma = (\mathcal{P}, \mathcal{F})$ este o semnătură, o Σ -structură este orice structură $S = (D, Pred, Fun)$ astfel încât fiecărui simbol predicativ/funcțional îi corespunde în mod unic un predicat/o funcție.

Exemplul 3.1. Fie $\Sigma = (\{P, Q\}, \{f, i, a, b\})$ unde P, Q sunt simboluri predicative de aritate $ar(P) = ar(Q) = 2$ și f, i, a, b sunt simboluri funcționale cu aritățile: $ar(f) = 2$, $ar(i) = 1$ și $ar(a) = ar(b) = 0$.

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

De reținut!

Structură = domeniu + predicate + funcții

Signatură = simboluri predicative + simboluri funcționale
 Unei semnături Σ îi putem asocia mai multe structuri, numite Σ -structuri.

Mulțimea simbolurilor predicative dintr-o Σ -structură de aritate n este notată cu $\mathcal{P}_n = \{P \mid ar(P) = n\}$, iar mulțimea simbolurilor funcționale de aritate n este notată cu $\mathcal{F}_n = \{f \mid ar(f) = n\}$.

4 Sintaxa logicii de ordinul I

În continuare, vom studia sintaxa (definiția matematică a modului în care se scriu) formulelor din logica cu predicate de ordinul I și semantica acestora (cum calculăm valoarea de adevăr a unei formule).

Pentru logica de ordinul I limbajul (mulțimea de șiruri de simboluri) este determinat de alegerea lui Σ . Practic, există mai multe limbafe 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} .

4.1 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. Alfabetul logicii de ordinul I conține următoarele simboluri:

1. *conectori logici* deja cunoscuți: $\wedge, \vee, \neg, \rightarrow, \leftrightarrow$, precum și doi *cuantificatori* noi: \forall, \exists ;
2. *variabile*: vom presupune că avem la dispoziție o mulțime infinit numărabilă de variabile notată $\mathcal{X} = \{x, y, z, x', y', x_1, z'', \dots\}$ (a nu se confunda cu, variabilele propoziționale din logica propozițională – sunt două noțiuni fundamentale diferite);
3. simboluri auxiliare: “(”, “)”, “.” și respectiv “,”;
4. simboluri suplimentare, care sunt specifice fiecărei semnături în parte: simbolurile funcționale din mulțimea \mathcal{F} și respectiv simbolurile predicate din mulțimea \mathcal{P} .

4.2 Termen

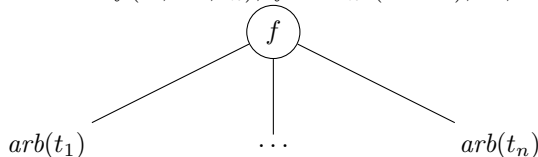
Definiția 4.1. *Mulțimea termenilor, \mathcal{T} , este cea mai mică mulțime care satisface următoarele proprietăți:*

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).

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 abstract de sintaxă definit după cum urmează:

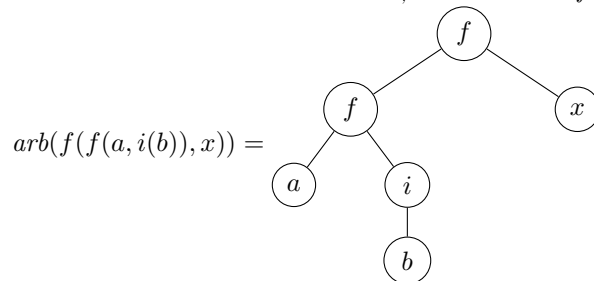
1. dacă $t = c$, $c \in \mathcal{F}_0$, atunci $arb(t) = \bigcirc c$
2. dacă $t = x$, $x \in \mathcal{X}$, atunci $arb(t) = \bigcirc x$
3. dacă $t = f(t_1, \dots, t_n)$, $f \in \mathcal{F}_n$ ($n > 0$), $t_1, \dots, t_n \in \mathcal{T}$, atunci $arb(t) =$



Exemplul 4.1. De exemplu, pentru semnatura $\Sigma = (\{P, Q\}, \{f, i, a, b\})$ definită în Exercițiul 3.1 (unde $ar(P) = ar(Q) = 2$, $ar(f) = 2$, $ar(i) = 1$, $ar(a) = ar(b) = 0$), iată câteva exemple de termeni: a , b , x , y , x_1 , y' , $i(a)$, $i(x)$, $i(i(a))$, $i(i(x))$, $f(a, b)$, $i(f(a, b))$, $f(f(x, a), f(y, y))$, etc.

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

Observația 4.1. 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 memorați sub formă de arbori cu rădăcină. Iată arborele atașat termenului $f(f(a, i(b)), x)$:



4.3 Formule atomice

Definiția 4.2 (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 4.2. Continuând Exemplul 4.1, folosim signatura $\Sigma = (\{P, Q\}, \{f, i, a, b\})$, unde $ar(P) = ar(Q) = 2$, $ar(f) = 2$, $ar(i) = 1$, $ar(a) = ar(b) = 0$.

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

4.4 Formule de ordinul I

Definiția 4.3 (Formule de ordinul I). Mulțimea formulelor de ordinul I, notată $LP1$, este cea mai mică mulțime astfel încât:

1. (cazul de bază) orice formulă atomică este formulă (adică $P(t_1, \dots, t_n) \in LP1$ pentru orice simbol predicativ $P \in \mathcal{P}_n$ și orice termeni t_1, \dots, t_n ;
2. (cazurile inductive) pentru orice formule $\varphi_1, \varphi_2 \in LP1$, pentru orice variabilă $x \in \mathcal{X}$, avem că:
 - (a) $\neg\varphi_1 \in LP1$;
 - (b) $(\varphi_1 \vee \varphi_2) \in LP1$;
 - (c) $(\varphi_1 \wedge \varphi_2) \in LP1$;
 - (d) $(\varphi_1 \rightarrow \varphi_2) \in LP1$;
 - (e) $(\varphi_1 \leftrightarrow \varphi_2) \in LP1$;
 - (f) $\forall x.\varphi \in LP1$;
 - (g) $\exists x.\varphi \in LP1$.

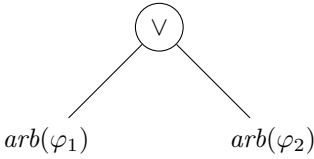
Observația 4.2. Simbolurile predicative de aritate 0 țin locul variabilelor propoziționale (deocamdată, la nivel sintactic). Construcțiile $\forall x.\varphi$ și $\exists x.\varphi$ sunt noi.

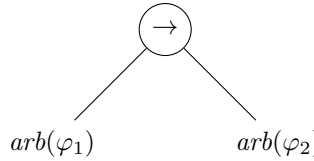
Formulele au asociat un arbore abstract de sintaxă definit în cele ce urmează:

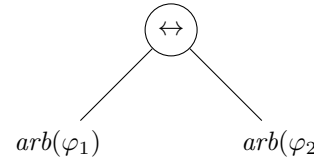
$$1. \text{ dacă } \varphi = P(t_1, \dots, t_n), \text{ atunci } arb(\varphi) = \begin{array}{c} \textcircled{P} \\ \swarrow \quad \downarrow \quad \searrow \\ arb(t_1) \quad \dots \quad arb(t_n) \end{array} ;$$

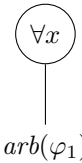
$$2. \text{ dacă } \varphi = \neg\varphi_1, \text{ atunci } arb(\varphi) = \begin{array}{c} \textcircled{\neg} \\ \downarrow \\ arb(\varphi_1) \end{array} ;$$

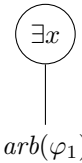
$$3. \text{ dacă } \varphi = (\varphi_1 \wedge \varphi_2), \text{ atunci } arb(\varphi) = \begin{array}{c} \textcircled{\wedge} \\ \swarrow \quad \searrow \\ arb(\varphi_1) \quad arb(\varphi_2) \end{array} ;$$

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 $arb(\varphi) =$  .

4.5 Paranteze

Parantezele 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 abstract de sintaxă în două sau mai multe moduri, se vor folosi paranteze pentru a stabili arborele dorit.

De exemplu, formula $\varphi_1 \vee \varphi_2 \wedge \varphi_3$ 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 nu polua formulele cu prea multe paranteze, se stabilesc *priorități*. În logică, ordinea priorității este: $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \forall, \exists$. În cazul în care nu suntem 100% siguri, este de preferat să folosim paranteze suplimentare.

Din cauza priorității conectorilor logici, formula $\varphi = P(a) \vee P(b) \wedge P(c)$ va fi întotdeauna înțeleasă ca $(P(a) \vee (P(b) \wedge P(c)))$ (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 $*$ are prioritate în fața lui $+$ ($*$ este similar cu \wedge și $+$ cu \vee).

4.6 Exemplul 1

În continuare, 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 Om pentru primul predicat și simbolul predicativ $Muritor$ 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

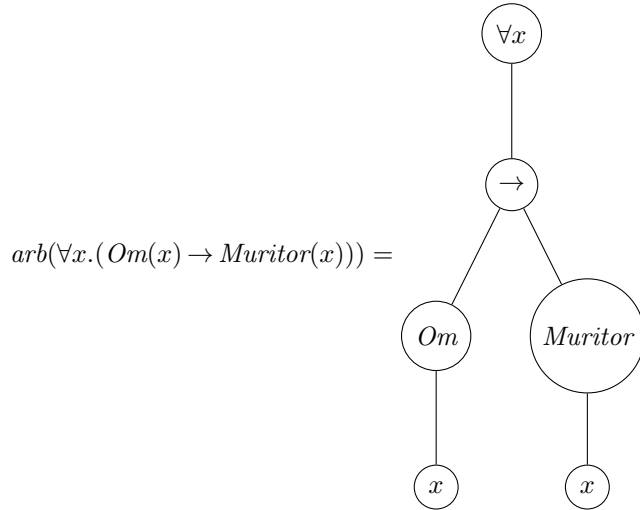
$$\Sigma = (\{Om, Muritor\}, \{s\}),$$

unde Om și $Muritor$ sunt simboluri predicative de aritate $ar(Om) = ar(Muritor) = 1$, iar s este un simbol funcțional de aritate $ar(s) = 0$, adică un simbol constant.

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

$$\forall x.(Om(x) \rightarrow Muritor(x)),$$

al cărei arbore abstract de sintaxă este:



Afirmația *Socrate este om* o vom modela prin formula atomică $Om(s)$, iar afirmația *Socrate este muritor* o vom modela prin formula atomică $Muritor(s)$.

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

1. D este mulțimea tuturor ființelor de pe Pământ;
2. $Om^S(x)$ este adevărat pentru orice ființă x care este și om;

3. $Muritor^S(x)$ 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 cursul următor), toate cele trei formule discutate în această secțiune, adică $\forall x.(Om(x) \rightarrow Muritor(x))$, $Om(s)$ și respectiv $Muritor(s)$, 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 $Muritor(s)$ este în mod necesar adevărată în *orice* structură în care formulele $Muritor(s)$ $\forall x.(Om(x) \rightarrow Muritor(x))$ și respectiv $Om(s)$ sunt adevărate, nu doar în structura S de mai sus.

4.7 Exemplul 2

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 limbajul de ordinul I asociat signaturii Σ :

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

De multe ori, în cazul simbolurilor predicative și simbolurilor functionale binare, se folosește notația infixată (e.g., $x < y$ în loc de $\langle(x, y)$). 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 e vorba de funcția $-$ 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 formulă 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 protejate de un cuantificator 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).