Logică pentru informatică - Săptămâna 10
Semantica logicii de ordinul I

December 5, 2019

1 Introducere

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 precum și de valorile asociate *variabilelor libere*.

2 Variabilele unei formule

Cu $vars(\varphi)$ notăm variabilele care apar în formula φ. De exemplu, vom avea că $vars\big((\forall z.(P(x,y)))\big) = \{x, y, z\}$. Definim funcția $vars : \texttt{LP1} \to 2^{\mathcal{X}}$ în cele ce urmează.

În primul rând, definim o funcție $vars : \mathcal{T} \to 2^{\mathcal{X}}$ (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 și că $2^{\mathcal{X}}$ denotă mulțimea tuturor submulțimilor lui $\mathcal{X}$. Considerând fixată o signatură 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 2.1. *Funcția $vars : \mathcal{T} \to 2^{\mathcal{X}}$ este definită recursiv după cum urmează:*

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

Putem acum defini (inductiv, prin imbricare) funcția extinsă și notată omonim, $vars : \texttt{LP1} \to 2^{\mathcal{X}}$, care asociază unei formule din LP1 mulțimea de variabile ale formulei (adică, variabilele care apar în formulă):

Definiția 2.2. *Funcția vars* : `LP1` $\rightarrow 2^{\mathcal{X}}$ *este definită recursiv după cum urmează:*

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

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 2.1. *Fie formula* $\varphi = \Big(\forall x.\Big(P(x,y) \wedge \exists y.\big(P(z, f(x,y)) \wedge P(x,y)\big)\Big)\Big) \wedge P(x,x)$. *Avem* $vars(\varphi) = \{x, y, z\}$.

2.1 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 două declarații a variabilei `x`. Variabila z este variabilă globală (restul care apar fiind locale).

Un fenomen similar se întâmplă în formulele logicii de ordinul I. De exemplu, în formula $\Big(\forall x.\big(\forall y.(P(x,y) \land P(x,z) \land (\exists x.P(x,y)))\big)\Big)$, 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 al variabilei x cuantificate universal este $\big(\forall y.(P(x,y) \land P(x,z) \land (\exists x.P(x,y)))\big)$, în timp ce domeniul de vizibilitate al variabilei x cuantificate existențial este $P(x,y)$. 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*.

Definițiile următoare stabilesc formal noțiunea/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.

2.2 Apariții libere și legate ale variabilelor

În această secțiune vom utiliza noțiunea de *arbore abstract de sintaxă* pentru formulele din logica de ordinul I.

Definiția 2.3. *O* apariţie 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 2.4. *O* apariţie 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 $\exists x$.*

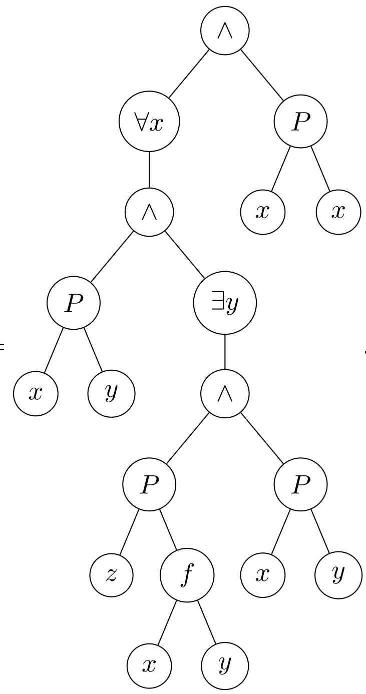
Cel mai apropiat astfel de nod etichetat cu $\forall x$ sau cu $\exists x$ este cuantificarea care leagă *apariția în cauză a variabilei x.*

Exemplul 2.2. *Considerăm în continuare formula*

$$\varphi = \Bigg(\forall x.\Big(P(x,y) \land \exists y.\big(P(z, f(x,y)) \land P(x,y)\big)\Big)\Bigg) \land P(x,x).$$

Arborele ei abstract este:

$arb(\varphi) =$



.

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 liberere ale variabilelor în formula φ sunt marcate cu text îngroșat:

$$\varphi = \Big(\forall x.\big(P(x, \boldsymbol{y}) \land \exists y.(P(\boldsymbol{z}, f(x, y)) \land P(x, y))\big)\Big) \land P(\boldsymbol{x}, \boldsymbol{x}).$$

Toate aparițiile legate ale variabilelor în formula φ sunt marcate cu text îngroșat în următoarea formulă:

$$\varphi = \Big(\forall x.\big(P(\boldsymbol{x}, y) \land \exists y.(P(z, f(\boldsymbol{x}, \boldsymbol{y})) \land P(\boldsymbol{x}, \boldsymbol{y}))\big)\Big) \land P(x, x).$$

Observația 2.1. *Începând cu acest curs 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/denumește un cuantificator (sau, prin care se leagă variabila x).*

2.3 Variabile libere și variabile legate

Mulţimea variabilelor unei formule φ care au cel puţin o apariţie liberă se notează *free*(φ).

Definiția 2.5. *Funcția free* : $\texttt{LP1} \to 2^{\mathcal{X}}$ *este definită recusiv în modul următor:*

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

Exemplul 2.3. *Continuând exemplul precedent, pentru formula*

$$\varphi = \Bigg(\forall x.\Big(P(x,y) \wedge \exists y.\big(P(z, f(x,y)) \wedge P(x,y)\big)\Big)\Bigg) \wedge P(x,x),$$

avem că

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

Cu $\mathit{bound}(\varphi)$ 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 2.6. *Funcția bound* : $\texttt{LP1} \to 2^{\mathcal{X}}$ *este definită recursiv astfel:*

1. $\mathit{bound}(P(t_1, \ldots, t_n)) = \emptyset$;
2. $\mathit{bound}(\neg\varphi) = \mathit{bound}(\varphi)$;
3. $\mathit{bound}(\varphi_1 \wedge \varphi_2) = \mathit{bound}(\varphi_1) \cup \mathit{bound}(\varphi_2)$;
4. $\mathit{bound}(\varphi_1 \vee \varphi_2) = \mathit{bound}(\varphi_1) \cup \mathit{bound}(\varphi_2)$;
5. $\mathit{bound}(\varphi_1 \to \varphi_2) = \mathit{bound}(\varphi_1) \cup \mathit{bound}(\varphi_2)$;
6. $\mathit{bound}(\varphi_1 \leftrightarrow \varphi_2) = \mathit{bound}(\varphi_1) \cup \mathit{bound}(\varphi_2)$;
7. $\mathit{bound}\big((\forall x.\varphi)\big) = \mathit{bound}(\varphi) \cup \{x\}$;
8. $\mathit{bound}\big((\exists x.\varphi)\big) = \mathit{bound}(\varphi) \cup \{x\}$.

Definiția 2.7. *Variabilele legate ale unei formule* φ *sunt elementele mulțimii* $\mathit{bound}(\varphi)$.

Definiția 2.8. *Variabilele libere ale unei formule* φ *sunt elementele mulțimii* $\mathit{free}(\varphi)$.

Observația 2.2. *$free(\varphi)$ şi $bound(\varphi)$ pot avea elemente în comun.*

Observația 2.3. *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 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 $\mathcal{X}$.*

2.4 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 cuantificatorii $\forall$ 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 cuantificator 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:

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

3 Semantica formulelor logicii de ordinul I

Reamintim că o signatură $\Sigma = (\mathcal{P}, \mathcal{F})$ 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.

Exemplul 3.1. *În exemplele care urmează, vom lucra peste signatura $\Sigma = (\{P\}, \{f, i, e\})$, unde P este simbol predicativ de aritate 2, iar f, i și e sunt simboluri funcționale de aritate 2, 1 și respectiv 0.*

Exemplul 3.2. *Continuând exemplul anterior, $\mathcal{P}_2 = \{P\}, \mathcal{P}_1 = \emptyset, \mathcal{P}_0 = \emptyset, \mathcal{F}_2 = \{f\}, \mathcal{F}_1 = \{i\}, \mathcal{F}_0 = \{e\}$.*

Reamintim și că, dacă $\Sigma = (\mathcal{P}, \mathcal{F})$ este o signatură, 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 3.3. *Continuând exemplele anterioare, unde $\Sigma = (\{P\}, \{f, i, e\})$, fie Σ-structurile:*

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 P este $=$ (predicatul de egalitate pentru numere întregi), funcția $+$ este funcția de adunare a numerelor întregi, $-$ este funcția minus unar, iar simbolul constant e are asociată constanta 0.

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

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

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

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

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

3.1 Noțiunea de atribuire

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 3.1 (Atribuire)**.** *Fie Σ o signatură și S o Σ-structură cu domeniul D. Se numește S-atribuire este orice funcție*

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

Exemplul 3.4. *Funcția $\alpha_1 : \mathcal{X} \to \mathbb{Z}$, definită ca mai jos, este o S_1-atribuire:*

1. $\alpha_1(x_1) = 5$;
2. $\alpha_1(x_2) = 5$;
3. $\alpha_1(x_3) = 6$;
4. $\alpha_1(x) = 0$ *pentru orice* $x \in \mathcal{X} \setminus \{x_1, x_2, x_3\}$.

Exemplul 3.5. *Funcția $\alpha_2 : \mathcal{X} \to \mathbb{Z}$, definită ca mai jos, este o S_1-atribuire:*

1. $\alpha_2(x_1) = 6$;
2. $\alpha_2(x_2) = 5$;
3. $\alpha_2(x_3) = 6$;
4. $\alpha_2(x) = 0$ *pentru orice* $x \in \mathcal{X} \setminus \{x_1, x_2, x_3\}$.

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 $\overline{\alpha}$,

$$\overline{\alpha} : \mathcal{T} \to D,$$

dată în definiția care urmează.

Definiția 3.2 (Valoarea unui termen într-o atribuire)**.** *Dându-se o S-atribuire α și un termen $t \in \mathcal{T}$ peste signatura Σ,* valoarea termenului t în atribuirea α *este un element al domeniului D, notat cu $\overline{\alpha}(t)$ și calculat recursiv astfel:*

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

Exemplul 3.6. *Continuând exemplele precedente, avem:*

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

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

Definiția 3.3 (Actualizarea unei atribuiri)**.** *Dându-se o atribuire α, o variabilă $x \in \mathcal{X}$ și o valoare $u \in D$, notăm cu $\alpha[x \mapsto u]$ o nouă atribuire, care coincide în totalitate cu α, exceptând valoarea variabilei x, care devine acum u:*

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

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

Exemplul 3.7. *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 $\alpha_1[x \mapsto 6]$ este $\overline{\alpha_1[x \mapsto 6]}(f(i(x_1), e)) = -6$.*

3.2 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 signatură Σ. Această valoare se poate calcula doar într-o Σ-structură S, cu ajutorul unei S-atribuiri 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 atribure 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 3.4. *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. $S, \alpha \models P(t_1, \ldots, t_n)$, *dacă* $P^S(\overline{\alpha}(t_1), \ldots, \overline{\alpha}(t_n)) = 1$;
2. $S, \alpha \models \neg\varphi$, *dacă* $S, \alpha \not\models \varphi$;
3. $S, \alpha \models \varphi_1 \land \varphi_2$, *dacă* $S, \alpha \models \varphi_1$ *și* $S, \alpha \models \varphi_2$;
4. $S, \alpha \models \varphi_1 \lor \varphi_2$, *dacă* $S, \alpha \models \varphi_1$ *sau* $S, \alpha \models \varphi_2$;
5. $S, \alpha \models \varphi_1 \to \varphi_2$, *dacă* $S, \alpha \not\models \varphi_1$ *sau* $S, \alpha \models \varphi_2$;
6. $S, \alpha \models \varphi_1 \leftrightarrow \varphi_2$ *dacă (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. $S, \alpha \models \exists x.\varphi$ *dacă există* $u \in D$ *astfel încât* $S, \alpha[x \mapsto u] \models \varphi$;
8. $S, \alpha \models \forall x.\varphi$ *dacă pentru orice* $u \in D$, *avem că* $S, \alpha[x \mapsto u] \models \varphi$.

Exemplul 3.8. *Vom lucra în continuare peste signatura* $\Sigma = (\{P\}, \{f, i, e\})$, Σ-*structura* $S_1 = (\mathbb{Z}, \{=\}, \{+, -, 0\})$ *definită la începutul cursului și* S_1*-atribuirile* α_1, α_2.

Avem că

$$\begin{aligned}
S_1, \alpha_1 \models P(x_1, x_1) \ \textit{dacă} \quad & P^{S_1}(\overline{\alpha_1}(x_1), \overline{\alpha_1}(x_1)) = 1, \\
\textit{dacă} \quad & \overline{\alpha_1}(x_1) = \overline{\alpha_1}(x_1), \\
\textit{dacă} \quad & \alpha_1(x_1) = \alpha_1(x_1), \\
\textit{dacă} \quad & 5 = 5.
\end{aligned}$$

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

Exemplul 3.9. *Continuând exemplul anterior, avem*

$$\begin{aligned}
S_1, \alpha_1 \models P(x_1, x_3) \ \textit{ddacă} \quad & P^{S_1}(\overline{\alpha_1}(x_1), \overline{\alpha_1}(x_3)) = 1, \\
\textit{ddacă} \quad & \overline{\alpha_1}(x_1) = \overline{\alpha_1}(x_3), \\
\textit{ddacă} \quad & \alpha_1(x_1) = \alpha_1(x_3), \\
\textit{ddacă} \quad & 5 = 6.
\end{aligned}$$

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 3.10. *Continuând exemplul anterior, avem*

$$\begin{aligned}
S_1, \alpha_1 \models \neg P(x_1, x_3) \ \textit{dacă} \quad & S_1, \alpha_1 \not\models P(x_1, x_3) \\
\textit{dacă} \quad & P^{S_1}(\overline{\alpha_1}(x_1), \overline{\alpha_1}(x_3)) = 0, \\
\textit{dacă} \quad & \overline{\alpha_1}(x_1) \neq \overline{\alpha_1}(x_3), \\
\textit{dacă} \quad & \alpha_1(x_1) \neq \alpha_1(x_3), \\
\textit{dacă} \quad & 5 \neq 6.
\end{aligned}$$

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 3.11. *Continuând exemplul anterior, avem*

$$\begin{aligned}
S_1, \alpha_1 \models P(x_1, x_1) \wedge \neg P(x_1, x_3) \ \textit{dacă} \quad & S_1, \alpha_1 \models P(x_1, x_1) \ \textit{și} \ S_1, \alpha_1 \models \neg P(x_1, x_3), \\
\textit{dacă} \quad & \ldots \ \textit{și} \ \ldots, \\
\textit{dacă} \quad & 5 = 5 \ \textit{și} \ 5 \neq 6.
\end{aligned}$$

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 3.12. *Continuând exemplul anterior, avem*

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

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 3.13. *Continuând exemplul anterior, avem* $S_1, \alpha_1 \models \exists x_1.P(x_1, x_3)$

$$\begin{array}{lr}
\text{dacă} & \text{există } u \in D \text{ a.î. } S_1, \alpha_1[x_1 \mapsto u] \models P(x_1, x_3), \\
\text{dacă} & \text{există } u \in D \text{ a.î. } P^{S_1}(\overline{\alpha_1[x_1 \mapsto u]}(x_1), \overline{\alpha_1[x_1 \mapsto u]}(x_3)) = 1, \\
\text{dacă} & \text{există } u \in D \text{ a.î. } \overline{\alpha_1[x_1 \mapsto u]}(x_1) = \overline{\alpha_1[x_1 \mapsto u]}(x_3), \\
\text{dacă} & \text{există } u \in D \text{ a.î. } \alpha_1[x_1 \mapsto u](x_1) = \alpha_1[x_1 \mapsto u](x_3), \\
\text{dacă} & \text{există } u \in D \text{ a.î. } u = \alpha_1(x_3), \\
\text{dacă} & \text{există } u \in D \text{ a.î. } u = 6.
\end{array}$$

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 3.14. *Continuând exemplul anterior, avem* $S_1, \alpha_1 \models \forall x_1.\exists x_3.P(x_1, x_3)$

$$\begin{array}{lr}
\text{dacă} & \text{pentru orice } u \in D, \text{ avem că } S_1, \alpha_1[x_1 \mapsto u] \models \exists x_3.P(x_1, x_3), \\
\text{dacă} & \text{pt. orice } u \in D, \text{ există } v \in D \text{ a.î. } S_1, \alpha_1[x_1 \mapsto u][x_3 \mapsto v] \models P(x_1, x_3), \\
\text{dacă} & \ldots \\
\text{dacă} & \text{pentru orice } u \in D, \text{ avem că există } v \in D \text{ a.î. } u = v.
\end{array}$$

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

3.3 Noțiuni semantice

Satisfiabilitate într-o structură fixată

Definiția 3.5 (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 3.15. *Formula* $P(x_1, x_3)$ *este satisfiabilă în structura* S_1, *deoarece există o atribuire, și anume* α_2, *cu propritatea că* $S_1, \alpha_2 \models P(x_1, x_3)$.

Formula $\neg P(x_1, x_1)$ *nu este satisfiabilă în structura* S_1, *deoarece, pentru orice atribuire* α *aleasă, avem că* $S_1, \alpha \not\models \neg P(x_1, x_1)$ *(verificați formal afirmația).*

Validitate într-o structură fixată

Definiția 3.6 (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 3.16. *Formula* $P(x_1, x_3)$ *nu este validă în structura* S_1*, deoarece există o atribuire, și anume* α_1*, cu propritatea că* $S_1, \alpha_1 \not\models P(x_1, x_3)$*.*

Formula $P(x_1, x_1)$ *este validă în structura* S_1*, deoarece orice atribuire* α *am alege,* $S_1, \alpha \models P(x_1, x_1)$ *(verificați din nou).*

Satisfiabilitate

Definiția 3.7 (Satisfiabilitate)**.** *O formulă* φ este satisfiabilă *dacă există o structură* S *și o* S*-atribuire* α *cu propritatea că*

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

Exemplul 3.17. *Formula* $\neg P(x_1, x_1)$ *este satisfiabilă, deoarece există o structură (și anume* S_5*) și o* S_5*-atribuire (și anume* α_1*) astfel încât* $S_5, \alpha_1 \models \neg P(x_1, x_1)$ *(deoarece* $5 \not< 5$*).*

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.*

Observația 3.1. *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 (de exemplu* $\neg P(x_1, x_1)$*).*

Validitate

Definiția 3.8 (Validitate)**.** *O formulă* φ este validă *dacă, pentru orice structură* S *și pentru orice* S*-atribuire* α*, avem*

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

Exemplul 3.18. *Formula* $P(x_1, x_1)$ *nu este validă, deoarece* $S_5, \alpha_1 \not\models P(x_1, x_1)$*.*

Formula $P(x_1, x_1) \rightarrow P(x_1, x_1)$ *este validă.*

Observația 3.2. *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,* $P(x_1, x_1)$*).*

Consecință semantică

Definiția 3.9. *O formulă* φ este consecință semantică a formulelor $\varphi_1, \ldots, \varphi_n$ într-o structură fixată S*, notat* $\varphi_1, \ldots, \varphi_n \models_S \varphi$*, dacă, pentru orice* S*-atribuire* α *pentru care* $S, \alpha \models \varphi_1$*,* $S, \alpha \models \varphi_2$*, ...,* $S, \alpha \models \varphi_n$*, avem că* $S, \alpha \models \varphi$*.*

Exemplul 3.19. *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 3.10. *O formulă* φ *este consecință semantică a formulelor* $\varphi_1, \ldots, \varphi_n$*, notat* $\varphi_1, \ldots, \varphi_n \models \varphi$*, dacă*

$$\varphi_1, \ldots, \varphi_n \models_S \varphi$$

pentru orice structură S*.*

Exemplul 3.20. *Avem că* $P(x, y) \not\models P(y, x)$*, deoarece există o structură (și anume* S_5*) astfel încât* $P(x, y) \not\models_{S_5} P(y, x)$*.*

De asemenea, este adevărat că $\forall x.\forall y.\forall z.(P(x, y) \land P(y, z) \to P(x, z)), P(x_1, x_2), P(x_2, x_3) \models P(x_1, x_3)$*.*

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

Material suplimentar

În materialul suplimentar din cursul 1 am arătat că `LP1` este o extensie sintactică a lui `LP`.

Pe scurt, dacă $A = \{\mathtt{p}, \mathtt{q}, \mathtt{r}, \ldots\}$ este o mulțime de variabile propoziționale atunci construim o signatura $\Sigma_{\mathtt{LP}} = \{A, \emptyset\}$, unde variabilele propoziționale din A sunt simboluri predicative de aritate 0.

Semantica formulelor `LP1` construite peste signatura $\Sigma_{\mathtt{LP}}$ este consistentă cu semantica formulelor `LP`. Fie $\tau : A \to B$ o atribuire. Fie $S = (D, \{a^S \mid a \in A\}, \emptyset)$ o $\Sigma_{\mathtt{LP}}$-structură, unde D este orice mulțime nenulă și $a^S = \tau(a)$.

Teorema 3.1. *Pentru orice* $\varphi \in \mathtt{LP}$*,* $\tau \models \varphi$ *ddacă* $S, \alpha \models \varphi$ *pentru orice* S*-atribuire* $\alpha : \mathcal{X} \to D$*.*

Exercițiul 3.2. *Demonstrați Teorema 3.1.*