Chapter 1

Logica de Ordinul I

Introducere

Logica de ordinul I este o extensie a logicii propoziționale. Această extensie 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.

De exemplu, în logica propozițională, nu putem exprima (într-un mod natural) următoare afirmație: *orice om este muritor*. În continuare, argumentăm de ce logica propozițională nu poate exprima această afirmație.

Încercarea 1. Am putea încerca să modelăm afirmația de mai sus în `LP`, asociind afirmației o variabilă propozițională $\mathtt{p} \in A$. Acum să modelăm afirmația *Socrate este om.* Evident, acestei a doua afirmații trebuie să îi asociem o altă variabilă propozițională $\mathtt{q} \in A$. Să presupunem că știm că $\mathtt{p}$ și $\mathtt{q}$ sunt adevărate. Formal, știm că lucrăm cu o interpretare $S : A \to B$ astfel încât $S(\mathtt{p}) = 1$ și $S(\mathtt{q}) = 1$. Putem trage concluzia ca afirmația *Socrate este muritor* este adevărată în interpretarea S? Nu, deoarece afirmației *Socrate este muritor* ar trebui să îi asociem o a treia variabilă propozițională $\mathtt{r}$ și nu putem trage nicio concluzie asupra lui $S(\mathtt{r})$ din faptul că $S(\mathtt{p}) = 1$ și $S(\mathtt{q}) = 1$. Deci, din semantica logicii propoziționale, nu putem trage concluzia ca $\mathtt{r}$ este adevărată în orice interpretare în care $\mathtt{p}$ și $\mathtt{q}$ sunt adevărate, în ciuda faptului că, în orice lume în care orice om este muritor și Socrate este om putem trage concluzia că Socrate este muritor. Această diferență între realitate și modelarea noastră ne indică faptul că modelarea nu este suficient de bună.

Exercițiul 1.0.1. *Încercați și altă modelare pentru afirmațiile de mai sus în* `LP`. *Argumentați că nici această modelare nu este suficient de bună.*

Logica de ordinul I aduce, în plus față de `LP`, noțiunea de cuantificator (existențial sau universal). Cuantificatorul universal este notat cu $\forall$, iar cuantificatorul existențial este notat cu $\exists$.

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

$$\forall x.(P(x) \to Q(x)),$$

care este citită astfel: *pentru orice x, dacă P de x, atucni Q de x.* P și Q se numesc predicate de ordinul I; predicatele generalizează variabilele propoziționale prin faptul că pot primi argumente (în acest caz, câte un argument fiecare). Predicatul P va fi predicatul care denotă umanitatea: $P(x)$ va fi adevărat când x este om. Afirmația *Socrate este om* va fi modelată prin formula $P(Socrate)$, unde *Socrate* va fi o *constantă* care este asociată obiectului Socrate. De exemplu, $P(Socrate)$ este adevărat (deoarece *Socrate* denotă un om), dar $P(Lăbuș)$ este fals dacă *Lăbuș* este o constantă care modelează cățelul *Lăbuș*.

Afirmația *Socrate este muritor* va fi reprezentată prin $Q(S)$ (țineți minte că constanta *Socrate* îl denotă pe omul Socrate). Într-o structură pentru logica de ordinul I în care $S(\forall x.(P(x) \to Q(x))) = 1$ și $S(P(Socrate))$ vom avea imediat și că $S(Q(Socrate)) = 1$.

În continuare, vom studia sintaxa (definiția matematică a modului în care arată) formulelor din Logica cu Predicate de Ordinul I și semantica acesteia (noțiunea de structură, ce valoare de adevăr asociem fiecărei formule într-o anumită structură).

1.1 Sintaxa

Înainte de a da sintaxa formulelor de ordinul I, avem nevoie să definim *termenii*.

1.1.1 Termeni

Fie $\mathcal{F}$ o mulţime *indexată* de *simboluri funcționale*. Formal, $\mathcal{F} = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \cup \mathcal{F}_n \cup \ldots$, unde $\mathcal{F}_n$ sunt simbolurile funcționale de *aritate* n, iar F_i este disjunctă de F_j dacă $i, j \in \mathbb{N}$ dar $i \neq j$.

Elementele lui $\mathcal{F}$, simbolurile funcţionale, au ataşat un număr natural numit *aritate*. Cu $\mathcal{F}_n$ notăm mulţimea simbolurilor funcţionale de aritate n. Elementele lui $\mathcal{F}$ vor fi notate cu $f, g, h, f', g', f_1, g_2, h''$, etc. Elementele lui $\mathcal{F}_0$ se mai numesc simboluri *constante* şi se notează cu a, b, c, d, a_0, c_0, b', etc.

Fie $\mathcal{X}$ o mulţime infinită, numărabilă, de *variabile*. Elementele lui $\mathcal{X}$ se notează cu x, y, z, x', y_1, z'', etc.

Definiția 1.1.1. Mulţimea *termenilor*, $\mathcal{T}$, este cea mai mică mulţime astfel incât:

1. $\mathcal{F}_0 \subseteq \mathcal{T}$
2. $X \subseteq \mathcal{T}$
3. dacă $f \in \mathcal{F}_n$ (cu $n > 0$) şi $t_1, \ldots, t_n \in \mathcal{T}$, atunci $f(t_1, \ldots, t_n) \in \mathcal{T}$.

Termenii (sau *termii*), sunt notaţi cu t, s, t_1, t_2, s_1, t', etc.

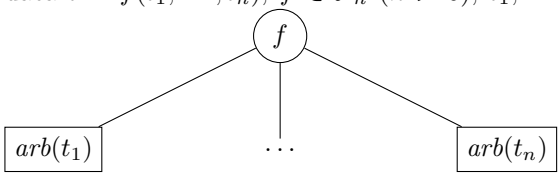
Exercițiul 1.1.1. *Pentru $t = f(g(a,b),y)$, stabiliți ce aritate au simbolurile funcționale f, g, a, b. De ce y nu este simbol funcțional?*

Deşi termenii sunt scrişi în mod uzual ca un şir de caractere, ei trebuie inteleşi ca fiind arbori, definiţi mai jos:

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

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

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



1.1.2 Formule

Fie $\mathcal{P}$ o mulţime indexată de *simboluri predicative* (adică $\mathcal{P} = \mathcal{P}_0 \cup P_1 \cup P_2 \cup \ldots \cup \mathcal{P}_n \cup \ldots$). Fiecare simbol predicativ are ataşat un număr natural numite *aritate*. Mulțimea simbolurilor predicative de aritate n se notează cu $\mathcal{P}_n$.

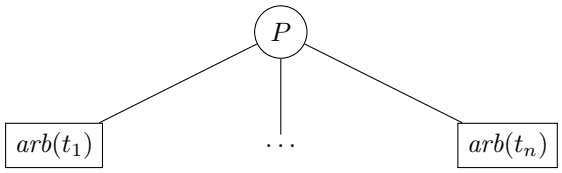
Definiția 1.1.2. Mulţimea formulelor de ordinul I, notată `LP1` este cea mai mică mulţime astfel incât:

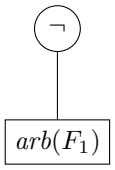
1. $P(t_1, \ldots, t_n) \in$ `LP1` pentru orice simbol predicativ $P \in \mathcal{P}_n$. Dacă $n = 0$, scriem P în loc de $P()$;
2. $(\neg F) \in$ `LP1`, pentru orice $F \in$ `LP1`;
3. $(F_1 \vee F_2) \in$ `LP1` pentru orice $F_1, F_2 \in$ `LP1`;
4. $(F_1 \wedge F_2) \in$ `LP1` pentru orice $F_1, F_2 \in$ `LP1`;
5. $(\forall x.F) \in$ `LP1` pentru orice $F \in$ `LP1` şi pentru orice $x \in \mathcal{X}$;
6. $(\exists x.F) \in$ `LP1` pentru orice $F \in$ `LP1` şi pentru orice $x \in \mathcal{X}$;
7. $(F) \in$ `LP1` pentru orice $F \in$ `LP1`.

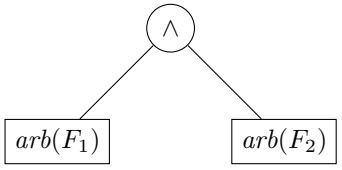
Observația 1.1.1. Simbolurile predicate de aritate 0 țin locul variabilelor propoziționale (deocamtă, la nivel sintactic). Construcțiile $\forall x.F$ și $\exists x.F$ sunt noi.

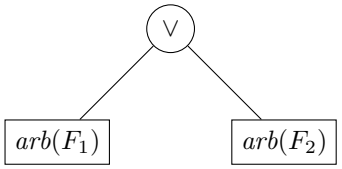
Formulele de forma $P(t_1, \ldots, t_n)$ se numesc *formule atomice*.

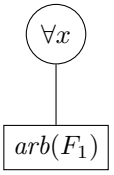
În mod similar cu termenii, deşi formulele sunt scrise ca şiruri de caractere, trebuie întelese ca fiind arbori abstracți:

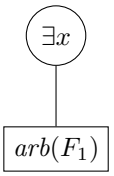
1. dacă $F = P(t_1, \ldots, t_n)$, atunci $arb(F) =$ 

2. dacă $F = (\neg F_1)$, atunci $arb(F) =$ 

3. dacă $F = (F_1 \wedge F_2)$, atunci $arb(F) =$ 

4. dacă $F = (F_1 \vee F_2)$, atunci $arb(F) =$ 

5. dacă $F = (\forall x.F_1)$, atunci $arb(F) =$ 

6. dacă $F = (\exists x.F_1)$, atunci $arb(F) =$ 

7. dacă $F = (F_1)$, atunci $arb(t) = arb(F_1)$.

Observația 1.1.2. Spre deosebire de LP, nu ne va interesa decât arborele . abstract.

Parantezele sunt folosite pentru a marca ordinea efectuării operațiilor logice (și, sau, not, etc.) În continuare, vom renunța la parantezele de prisos, in felul următor: în cazul în care o formulă poate fi interpretată ca un arbore în două sau mai multe moduri, se vor folosi paranteze pentru a stabili arborele dorit. De exemplu, formula $F_1 \vee F_2 \wedge F_3$ ar putea fi înţeleasă ca $(F_1 \vee F_2) \wedge F_3$ sau ca

$F_1 \vee (F_2 \wedge F_3)$. Pentru a nu polua formulele cu prea multe paranteze, se stabilesc *priorităţi*.

De exemplu, în continuare, formula $F = F_1 \vee F_2 \wedge F_3$ va fi întotdeauna înţeleasă ca $F_1 \vee (F_2 \wedge F_3)$, deoarece $\wedge$ este prioritar faţă de $\vee$. Ca analogie, la fel se întâmplă şi în cazul aritmeticii: $1 + 2 * 3$ va fi înteles ca $1 + (2 \times 3)$, deoarece $\times$ are prioritate în faţa lui $+$ ($\times$ este similar cu $\wedge$ şi $+$ cu $\vee$).

În logică, ordinea priorităţii este: $\neg, \forall, \exists, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow$. În cazul în care nu suntem 100% siguri, este de preferat sa folosim paranteze suplimentare.

Conectorii logici $\rightarrow, \leftarrow, \leftrightarrow$ sunt *zahăr sintactic* și vor fi înțeleși după cum urmează:

1. $F_1 \rightarrow F_2 \overset{notație}{=} \neg F_1 \vee F_2$ (conectorul logic *implicație*);
2. $F_1 \leftarrow F_2 \overset{notație}{=} \neg F_2 \vee F_1$ (contectorul logic *implicație inversă*);
3. $F_1 \leftrightarrow F_2 \overset{notație}{=} (F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$ (conectorul logic *dublă implicație*, cunoscut și ca *echivalență*);

1.1.3 Poziţii

Definiția 1.1.3 (Poziție)**.** O *poziţie* p este un şir de numere naturale $n_1 \cdot \ldots \cdot n_k$.

Observația 1.1.3. Pentru $k = 0$, Şirul vid este notat cu ϵ.

Exemplul 1.1.1. Spre exemplu, o poziţie este $p = 0 \cdot 2 \cdot 1$.

Definiția 1.1.4 (Poziții ale unui termen)**.** Mulțimea de *poziţii ale unui termen*, notate $pos(t)$, sunt:

1. $pos(c) = \{\epsilon\}$, dacă $c \in \mathcal{F}_0$
2. $pos(x) = \{\epsilon\}$, dacă $x \in \mathcal{X}$
3. $pos(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \bigcup_{i \in \{1,\ldots,n\}} i \cdot pos(t_i)$, dacă $f \in \mathcal{F}_n$.

În definitia de mai sus, am notat cu $i \cdot P$ mulţimea de poziţii obţinută din P prin adăugarea numărului i în faţă: $i \cdot P = \{i \cdot p \mid p \in P\}$.

Exemplul 1.1.2. Pentru $t = f(g(a, b), y)$, avem $pos(t) = \{\epsilon, 1, 2, 1 \cdot 1, 1 \cdot 2\}$.

Definiția 1.1.5 (Termen aflat la o poziție într-un termen)**.** Fie t un termen şi fie $p \in pos(t)$ o poziţie a termenului. *Termenul aflat la poziţia* p în termenul t, notat $t|_p$, este definit astfel:

1. $t|_\epsilon = t$,
2. $f(t_1, \ldots, t_n)|_{i \cdot p} = (t_i)|_p$.

Exemplul 1.1.3. Pentru $t = f(g(a, b), y)$, şi $p = 1$, avem $t|_p = g(a, b)$. Dacă $q = 1 \cdot 2$, atunci $t|_q = b$.

Funcţia *pos* se extinde si asupra formulelor:

Definiția 1.1.6 (Poziții ale unei formule)**.** 1. $pos(P(t_1, \ldots, t_n)) = \{\epsilon\} \cup \bigcup_{i \in \{1,\ldots,n\}} i \cdot pos(t_i)$,

2. $pos(\neg F) = \{\epsilon\} \cup 1 \cdot pos(F)$,

3. $pos(F_1 \wedge F_2) = pos(F_1 \vee F_2) = \{\epsilon\} \cup 1 \cdot pos(F_1) \cup 2 \cdot pos(F_2)$,

4. $pos(\forall x.F_1) = pos(\exists x.F_1) = \{\epsilon\} \cup 1 \cdot pos(F_1)$.

Definiția 1.1.7 (Termen sau formulă aflat(ă) la o poziție într-o formulă)**.** Termenul sau formula aflat/aflată la poziţia p în formula F, notat(ă) $F|_p$ este:

1. $F|_\epsilon = F$,

2. $(\neg F)|_{1 \cdot p} = F|_p$,

3. $(F_1 \wedge F_2)|_{i \cdot p} = (F_1 \vee F_2)|_{i \cdot p} = (F_i)|_p$ $(i \in \{1, 2\})$,

4. $(\forall x.F_1)|_{1 \cdot p} = (\exists x.F_1)|_{1 \cdot p} = (F_1)|_p$.

1.1.4 Variabile

Definiția 1.1.8. *Mulţimea variabilelor* unui termen t este notată $var(t)$ şi este definită astfel:

$var(c) = \emptyset$ (dacă $c \in \mathcal{F}_0$)

$var(x) = \{x\}$ (dacă $x \in \mathcal{F}_0$)

$var(f(t_1, \ldots, t_n)) = \bigcup_{i \in \{1,\ldots,n\}} var(t_i)$

În mod alternativ, $var(t) = \{x \in \mathcal{X} \mid \text{există } p \in pos(t) \text{ a.i. } t|_p = x\}$, pentru orice termen t.

Exercițiul 1.1.2. *Defineţi $var(F)$ ca fiind mulţimea variabilelor dintr-o formulă F.*

1.1.5 Domeniu de vizibilitate - analogie cu limbajele de programare

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

```
1: for (int x = 1; x <= 10; ++x) {
2:   for (int x = 1; x <= 10; ++x) {
3:     s++;
4:   }
5: }
```

În aceast fragment de cod, sunt declarate două variabile, amândouă având același nume, și anume `x`. Domeniul de vizibilitate al variabile `x` declarate la linia 1 este dat de liniile $1-5$, iar domeniul de vizibilitate al variabile `x` declarată la linia 2 este dat de liniile $2-4$. Astfel, orice apariție a numelui `x` între liniile $2-4$ se referă la cea de-a doua declarație a variabilei, în timp ce orice apariție a numelui `x` între liniile $1-5$ (cu excepția liniilor $2-4$) se referă la prima declarație a lui `x`.

Liniile $1-5$ reprezintă domeniul de vizibilitate al primei declarații a variabilei `x`, iar liniile $2-4$ reprezintă domeniul de vizibilitate al celei de-a două declarații a variabilei `x`.

Un fenomen similar se întâmplă în formulele logicii de ordinul I. De exemplu, în formula $\forall x.(P(x) \wedge (\exists x.Q(x)))$, variabila x este cuantificată de două ori (prima dată universal, a doua oară existențial). O cuantificare a unei variabile se numește *legare*, 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 $P(x) \wedge (\exists x.Q(x))$, în timp ce domeniul de vizibilitate al variabilei x cuantificate existențial este $Q(x)$. Aparițiile unei variabile care apar în domeniul de vizibilitate al ei se numesc *legate*, în timp aparițiile din afara domeniului de vizibilitate se numesc *libere*. Definițiile următoare stabilesc formal noțiunea de apariție/variabilă legată și de apariție/variabilă liberă. Aparițiile libere ale unei variabile în logica de ordinul I sunt, ca o analogie, similare cu variabilele globale într-un limbaj de programare.

Definiția 1.1.9. O *apariţie* a unei variabile x într-o formulă F este o pozitie p a.i. $F|_p = x$ sau $F|_p = \forall x.G$ sau $F|_p = \exists x.G$ (pentru o formulă $G \in$ `LP1`).

Definiția 1.1.10. O apariţie p a unei variabile x într-o formulă F este *legată*, dacă există un prefix q a lui p astfel incât $F|_q = \forall x.G$ (dacă pe drumul de la apariţia respectivă spre rădăcina arborelui formulei găsim un nod $\forall x$ sau $\exists x$).

O aparitie p a unei variabile x într-o formulă F este *liberă* dacă apariția nu este legată.

Definiția 1.1.11. Mulţimea variabilelor unei formule care au cel puţin o apariţie legată se notează $bound(F)$:

1. $bound(P(t_1, \ldots, t_n)) = \emptyset$,
2. $bound(\neg F_1) = bound(F_1)$,
3. $bound(F_1 \wedge F_2) = bound(F_1 \vee F_2) = bound(F_1) \cup bound(F_2)$,
4. $bound(\forall x.F_1) = bound(\exists x.F_1) = bound(F_1) \cup \{x\}$.

Definiția 1.1.12. Mulţimea variabilelor unei formule care au cel puţin o apariţie liberă se notează $free(F)$:

1. $free(P(t_1, \ldots, t_n)) = var(t_1) \cup \ldots \cup var(t_n)$,

2. $free(\neg F_1) = free(F_1)$,
3. $free(F_1 \wedge F_2) = free(F_1 \vee F_2) = free(F_1) \cup free(F_2)$,
4. $free(\forall x.F_1) = free(\exists x.F_1) = free(F_1) \setminus \{x\}$.

Atenţie! $free(F)$ şi $bound(F)$ pot avea elemente în comun.

Exemplul 1.1.4. Fie $F = P(x) \wedge \forall x.Q(x, y)$. Prima apariţie a lui x este liberă. Singura apariţie a lui y este liberă. A doua apariţie a lui x este legată. Avem $var(F) = \{x, y\}$, $bound(F) = \{x\}$ şi $free(F) = \{x, y\}$.

1.1.6 Substituţii

Definiția 1.1.13. O *substituție* este o funcție $\sigma : \mathcal{X} \to \mathcal{T}$, cu proprietatea că $\sigma(x) \neq x$ pentru un număr finit de variabile $x \in \mathcal{X}$.

Definiția 1.1.14. Dacă $\sigma : \mathcal{X} \to \mathcal{T}$ este o substituție, atunci mulțimea $dom(\sigma) = \{x \mid \sigma(x) \neq x\}$ se numește domeniul substitutției σ.

Observația 1.1.4. Prin definiție, domeniul unei subtituții este o mulțime finită.

Definiția 1.1.15. Dacă $\sigma : \mathcal{X} \to \mathcal{T}$ este o substituție, atunci *extensia homomorfică* a lui σ la mulțimea termenilor este funcția $\sigma^\sharp : \mathcal{T} \to \mathcal{T}$, definită astfel:

1. $\sigma^\sharp(x) = \sigma(x)$, pentru orice $x \in \mathcal{X}$;
2. $\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n))$, pentru orice simbol funcțional $f \in \mathcal{F}_n$ de aritate $n \in \mathbb{N}$ și orice termeni $t_1, \ldots, t_n \in \mathcal{T}$.

Observația 1.1.5. Pentru orice simbol constant $c \in \mathcal{F}_0$ și pentru orice subtituție $\sigma : \mathcal{X} \to \mathcal{T}$, $\sigma^\sharp(c) = c$.

Dacă $t \in \mathcal{T}$ este un termen, atunci $\sigma^\sharp(t) \in \mathcal{T}$ este termenul obținut din t prin *aplicarea* substituției σ.

Substituţiile se notează cu $\sigma, \tau, \sigma_0, \tau_1, \sigma'$, etc.

Notația 1.1.1. *Pentru orice termen $t \in \mathcal{T}$, termenul $\sigma^\sharp(t)$ se notează cu $\sigma(t)$ sau cu $t\sigma$ și se numește termenul obținut prin* aplicarea subtituției σ asupra termenului t.

Practic, pentru a obține $t\sigma$ din t, fiecare apariţie a unei variabile x din t este înlocuită cu termenul $\sigma(x)$.

Definiția 1.1.16. Dacă $\sigma : \mathcal{X} \to \mathcal{T}$ este o substituție, $x \in \mathcal{X}$ este o variabilă și $t \in \mathcal{T}$ este un termen, atunci $\sigma[x \mapsto t]$ este o (nouă) substituție $\sigma' : \mathcal{X} \to \mathcal{T}$, notată $\sigma' = \sigma[x \mapsto t]$, definită astfel:

1. $\sigma'(y) = \sigma(y)$ pentru orice $y \in \mathcal{X} \setminus \{x\}$;

2. $\sigma'(x) = t$.

Definiția 1.1.17. Pentru orice substituție $\sigma : \mathcal{X} \to \mathcal{T}$, *extensia homomorfică* a lui σ la mulțimea formulelor este funcția $\sigma^\flat : \texttt{LP1} \to \texttt{LP1}$, definită astfel:

1. $\sigma^\flat(P(t_1, \ldots, t_n)) = P(\sigma^x(t_1), \ldots, \sigma^\sharp(t_n))$;
2. $\sigma^\flat(\neg F) = \neg(\sigma^\flat(F))$;
3. $\sigma^\flat(F_1 \texttt{op} F_2) = (\sigma^\flat(F_1))\texttt{op}(\sigma^\flat(F_2))$ (pentru orice operator logic $\texttt{op} \in \{\wedge, \vee\}$);
4. $\sigma^\flat(\forall y.F) = \forall y.((\sigma')^\flat(F))$, unde $\sigma' = \sigma[y \mapsto y]$,
5. $\sigma^\flat(\exists y.F) = \exists y.((\sigma')^\flat(F))$, unde $\sigma' = \sigma[y \mapsto y]$,

Practic, pentru a obține formula $\sigma^\flat(F)$ din formula F, fiecare apariţie **liberă** a variabilei x din formula F este înlocuită cu termenul $\sigma(x)$.

Observația 1.1.6. Atenție! Aparițiile legate ale variabilelor nu sunt înlocuite prin aplicarea substituției.

Notația 1.1.2. *Ca și în cazul termenilor, vom folosi notația* $\sigma^\flat(F) \overset{notație}{=} \sigma(F) \overset{notație}{=} F\sigma$.

Definiția 1.1.18. Prin $range(\sigma)$ notăm mulțimea de termeni

$$range(\sigma) = \{\sigma(x) \mid x \in dom(\sigma)\}.$$

Notația 1.1.3. *Dacă* $U = \{t_1, \ldots, t_n\}$ *este o mulțime de termeni, notăm cu* $var(U) = var(t_1) \cup \ldots \cup var(t_n)$ *mulțimea variabilelor care apar în termenii din* U.

Definiția 1.1.19. O formulă F *acceptă* o substituţie $\sigma : \mathcal{X} \to \mathcal{T}$ dacă

$$bound(F) \cap var(range(\sigma)) = \emptyset.$$

Observația 1.1.7. Avem voie să calculăm formula $F\sigma$ chiar dacă F nu acceptă σ. Noțiunea de *acceptare* va fi folosită în câteva teoreme a căror concluzie va fi adevărată doar dacă o anumită formulă acceptă o anumită substituție.

1.2 Semantica

Până în acest moment, am definit formal *sintaxa* formulelor de ordinul I, adică regulile după care scriem formulele de ordinul I. Nu am discutat deloc, decât la modul intuitiv, ce *înțelegem* printr-o formulă de ordinul I.

În acest capitol, vom studia *semantica* formulelor de ordinul I, adică înțelesul lor.

Înainte de a trece propriu-zis la semantică, este bine să amintim câteva concepte:

1.2.1 Funcții

O funcție n-ară peste o mulțime U este notată astfel:

$$f : \underbrace{U \times \ldots \times U}_{n} \to U.$$

Câteodată notăm $\underbrace{U \times \ldots \times U}_{n} = U^n$ și scriem $f : U^n \to U$.

De exemplu, $+$ este o funcție binară peste mulțimea numerelor întregi: $+ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$.

Un caz particular sunt funcțiile 0-are. Conform celor de mai sus, o astfel de funcție ar trebui să fie definită pe mulțimea U^0 și să ia valori în U. Dar mulțimea U^0 este, prin convenție, mulțimea alcătuită din exact un element (de obicei se consideră că acest element este mulțimea vidă și deci $U^0 = \{\emptyset\}$). Dacă avem o funcție $c : U^0 \to U$, atunci funcția c este unic determinată de valoarea ei în singurul element al mulțimii U^0. Vom conveni ca în acest caz să facem un abuz de notație și să scriem c în loc de $c(\emptyset)$ (unde $\emptyset$ este singurul element din U^0). Astfel, dacă $c : U^0 \to U$ este o funcție 0-ară peste U, vom considera că $c \in U$. Funcțiile de 0 argumente (sau 0-are) se numesc și constante.

1.2.2 Predicate

Un predicat este o funcție care întoarce o valoare de adevăr. În acest curs, mulțimea valorilor de adevăr am notat-o cu $\mathbb{B}$ și am convenit că $\mathbb{B} = \{0, 1\}$, unde 0 reprezintă valoarea de adevăr *fals* și 1 reprezintă valoarea de adevăr *adevărat*.

Un predicat n-ar peste o mulțime U este așadar o funcție

$$P : U^N \to \mathbb{B}.$$

Un exemplu de predicat binar este predicatul $\leq$, pe care l-ați învățat încă din clasele primare (fără să știți neapărat că este un predicat):

$$\leq : \mathbb{N} \times \mathbb{N} \to \mathbb{B}.$$

Pentru a simplifica notația, ați folosit scrierea $x \leq y$ în loc de $\leq (x, y) = 1$ și $x \not\leq y$ în loc de $\leq (x, y) = 0$.

Un predicat 0-ar peste o mulțime U este o funcție $P : U^0 \to \mathbb{B}$. Vom face aceeași convenție ca la funcțiile 0-are și vom considera că aceste predicate $P : U^0 \to mathbbB$, fără niciun argument, sunt valoari de adevăr $P \in \mathbb{B}$.

Unii autori folosesc și noțiunea de *relație* în loc de *predicat* și vorbesc despre relații 0-are, unare, binare, ternare, etc. Trebuie știut că *relația* și *predicatul* sunt unul și același lucru.

1.2.3 Funcții versus simboluri funcționale, predicate versus simboluri predicative

În cursul precedent, am discutat despre *simboluri funcționale* și despre *simboluri predicative*. Care este diferența dintre un simbol funcțional și o funcție?

Care este diferența dintre un simbol predicativ și un predicat? Un simbol este doar un nume, similar cu un identificator într-un limbaj de programare. Simbolul funcțional este un nume care ține locul unei funcții, în timp ce simbolul predicativ ține locul unui predicat. Aritatea unui simbol funcțional/predicativ reprezintă numărul de argumente al funcției/predicatului căreia/căruia îi ține locul.

1.2.4 Noțiunea de structură pentru logica de ordinul I

În cursul precedent, am notat cu $\mathcal{F} = \mathcal{F}_0 \cup F_1 \cup F_2 \cup \ldots$ mulțimea simbolurilor funcționale și cu $\mathcal{P} = \mathcal{P}_0 \cup \mathcal{P}_1 \cup \mathcal{P}_2 \cup \ldots$ mulțimea simbolurilor predicative.

În continuare, pentru a asocia fiecărei formule o valoare de adevăr, trebuie să stabilim ce predicat asociem fiecărui simbol predicativ și ce funcție asociem fiecărui simbol funcțional. De asemenea, trebuie să stabilim și care este *domeniul* peste care iau valori variabilele. Toate aceste aspecte sunt conținute în noțiunea de *structură*:

Definiția 1.2.1 (Structură)**.** O *structură* este o pereche $S = (U, I)$, formată din:

1. o mulțime nevidă U, numită *univers* (sau domeniu) al structurii;
2. o funcție I, care asociază:
   (a) fiecărui simbol funcțional $c \in \mathcal{F}_0$ de aritate 0 din signatură un element al universului $I_c \in U$;
   (b) fiecărui simbol funcțional $f \in \mathcal{F}_n$ de aritate n o funcție n-ară peste univers: $I_f : U \times \ldots \times U \to U$;
   (c) fiecărui simbol predicativ $P \in \mathcal{P}_0$ de aritate 0 o valoare de adevăr $I_P \in \mathbb{B}$;
   (d) fiecărui simbol predicativ $P \in \mathcal{F}_n$ de aritate n un predicat n-ar peste univers: $I_P : U \times \ldots \times U \to \mathbb{B}$;
   (e) fiecărei variabile $x \in \mathcal{X}$ un element al universului: $I_x \in U$.

Folosind convențiile anterioare privind funcțiile și predicatele de aritate 0, putem scrie mai simplu că $I_f : U^n \to U$ pentru orice simbol funcțional n-ar $f \in \mathcal{F}_n$ (tratând unitar cazurile (a) și (b)) și respectiv că $I_P : U^n \to B$ pentru orice simbol predicativ n-ar $P \in \mathcal{P}_n$ (tratând unitar cazurile (c) și (d)), unde $n \in \mathbb{N}$.

Exemplul 1.2.1. Fie $\mathcal{F}_0 = \{e\}$, $\mathcal{F}_1 = \{i\}$, $\mathcal{F}_2 = \{f\}$ și $\mathcal{P}_2 = \{\mathit{equals}\}$.

Vom considera structura $S = (U, I)$, unde $U = \mathbb{Z}$ și:

1. $I_e : \mathbb{Z}^0 \to \mathbb{Z}$ (sau, echivalent, $I_e \in \mathbb{Z}$), definită prin

$$I_e = 0;$$

2. $I_i : \mathbb{Z}^1 \to \mathbb{Z}$, definită prin
$$I_i(u) = -u,$$
pentru orice $u \in \mathbb{Z}$;

3. $I_f : \mathbb{Z}^2 \to \mathbb{Z}$, definită prin
$$I_f(u, v) = u + v,$$
pentru orice $u, v \in \mathbb{Z}$;

4. $I_{equals} : \mathbb{Z}^2 \to \mathbb{B}$, definit prin
$$I_{equals}(u, v) = \begin{cases} 1 & \text{dacă } u = v \\ 0 & \text{dacă } u \neq v; \end{cases}$$

5. $I_x \in \mathbb{Z}$, definită prin
$$I_x = 7,$$
pentru orice variabilă $x \in \mathcal{X}$.

Observația 1.2.1. Este important (vom vedea de ce mai târziu) ca universul unei structuri să nu fie mulțimea vidă. În exemplul de mai sus, am ales $U = \mathbb{Z}$, mulțime care, în mod evident, nu este vidă.

Definiția 1.2.2 (Valoarea unui termen într-o structură)**.** Dacă $S = (U, I)$ este o structură și $t \in T$ este un termen, atunci *valoarea termenului t în structura S* (sau *interpretarea termenului t în structura S*) se notează cu $S(t)$ și este definită inductiv astfel:

1. $S(c) = I_c$, pentru orice simbol funcțional $c \in \mathcal{F}_0$ de aritate 0;

2. $S(x) = I_x$, pentru orice variabilă $x \in \mathcal{X}$;

3. $S(f(t_1, \ldots, t_n)) = I_f(S(t_1), \ldots, S(t_n))$, pentru orice simbol funcțional $f \in \mathcal{F}_n$ de aritate $n \geq 1$ și pentru orice termeni $t_1, \ldots, t_n \in T$.

Exemplul 1.2.2. Continuând exemplul precedent, fie termenul $t = S(f(f(x, e), i(x)))$. Avem că
$$\begin{aligned} S(t) &= S\Big(f\big(f(x, e), i(x)\big)\Big) \\ &= I_f\Big(S\big(f(x, e)\big), S\big(i(x)\big)\Big) \\ &= \Big(S\big(f(x, e)\big) + S\big(i(x)\big)\Big) \\ &= I_f\big(S(x), S(e)\big)\big) + I_i\big(S(x)\big) \\ &= \big(S(x) + S(e)\big) + \big(-S(x)\big) \\ &= (I_x + I_e) + (-I_x) \\ &= (7 + 0) + \big(-7\big) \\ &= 0. \end{aligned}$$

Definiția 1.2.3 (Actualizarea valorii unei variabile într-o structură)**.** Fie $S = (U, I)$ o structură, $x \in \mathcal{X}$ o variabilă și $u \in U$ un element al universului.

Cu $S[x \mapsto u]$ notăm o nouă structură $S[x \mapsto u] = (U', I')$, definită astfel:

1. $U' = U$;
2. $I'_x = u$;
3. $I'_o = I_o$, pentru orice $o \in \mathcal{F} \cup \mathcal{P} \cup \mathcal{X} \setminus \{x\}$.

Numim $S[x \mapsto u]$ structura S după actualizarea valorii variabilei x la u.

Exemplul 1.2.3. Continuând exemplul precedent, calculați $S[y \mapsto 3](f(x, f(e, y)))$ (răspunsul trebuie să fie 10).

Definiția 1.2.4 (Valoarea de adevăr a unei formule într-o structură)**.** Dacă S este o structură și $F \in \texttt{LP1}$ este o formulă, atunci *valoarea de adevăr a formulei* F *în structura* S se notează cu $S(F)$ (vom avea $S(F) \in \mathbb{B}$) și este definită inductiv astfel:

1. $S(P(t_1, \ldots, t_n)) = I_P(S(t_1), \ldots, S(t_n))$, pentru orice predicat P de aritate $n \geq 0$;
2. $S(F_1 \vee F_2) = S(F_1) + S(F_2)$, pentru orice formule $F_1, F_2 \in \texttt{LP1}$ (funcția $+ : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ este aici funcția *sau* logic);
3. $S(F_1 \wedge F_2) = S(F_1) \cdot S(F_2)$, pentru orice formule $F_1, F_2 \in \texttt{LP1}$ (funcția $\cdot : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ este aici funcția *și* logic);
4. $S(\neg F) = \overline{S(F)}$, pentru orice formulă $F \in \texttt{LP1}$ (funcția $\bar{\cdot} : \mathbb{B} \to \mathbb{B}$ este aici funcția *negație* logică);
5. $S(\exists x.F) = \begin{cases} 1 & \text{dacă există un element } u \in U \text{ a.î. } S[x \mapsto u](F) = 1, \\ 0 & \text{altfel.} \end{cases}$
6. $S(\forall x.F) = \begin{cases} 1 & \text{dacă } S[x \mapsto u](F) = 1, \text{ pentru orice element } u \in U, \\ 0 & \text{altfel.} \end{cases}$

Exemplul 1.2.4. Continuând exemplul precedent, avem că:

1. $S\Big(\mathit{equals}(x, y)\Big) = 1$;
2. $S\Big(\mathit{equals}(x, e)\Big) = 0$;
3. $S[x \mapsto 0]\Big(\mathit{equals}(x, e)\Big) = 1$;
4. $S[x \mapsto 0]\Big(\mathit{equals}(x, e) \wedge \mathit{equals}(y, z)\Big) = 1$;
5. $S[x \mapsto 0]\Big(\mathit{equals}(x, e) \wedge \mathit{equals}(x, y)\Big) = 0$;

6. $S[x \mapsto 0]\Big(equals(x, e) \vee equals(x, y)\Big) = 1$;
7. $S[x \mapsto 0]\Big(equals(x, f(y, i(z)))\Big) = 1$;
8. $S\Big(equals(x, f(y, i(z)))\Big) = 0$;
9. $S\Big(\neg(equals(x, f(y, i(z))))\Big) = 1$;
10. $S\Big(\exists x.(equals(x, e))\Big) = 1$;
11. $S\Big(\forall x.(equals(x, e))\Big) = 0$;
12. $S\Big(\forall x.(equals(x, y))\Big) = 0$;
13. $S[y \mapsto 3]\Big(\exists x.(equals(x, y))\Big) = 0$;
14. $S\Big(\forall x.\forall y.(equals(f(x, y), f(y, x)))\Big) = 1$;
15. $S\Big(\forall x.\forall y.\forall z.(equals(f(x, f(y, z)), f(f(x, y), z)))\Big) = 1$;

Exercițiul 1.2.1. *Scrieți formulele care exprimă că e este element neutru la dreapta pentru f și că i este funcția invers.*

Exercițiul 1.2.2. *Dați exemplu de o structură în care formula* $\forall x.\forall y.(equals(f(x, y), f(y, x)))$ *să nu aibă valoarea de adevăr* 1.

1.2.5 Validitate, satisfiabilitate, consecință semantică

În general, valoarea de adevăr a unei formule depinde de structura în care este *evaluată* formula respectivă. Mai devreme am văzut că formula $\forall x.\forall y.(equals(f(x, y), f(y, x)))$ este adevărată în structura dată ca exemplu la începutul cursului, dar am găsit într-un exercițiu și o structură în care formula să nu fie adevărată.

Ca și la logica propozițională, există formule care sunt adevărate în orice structură și respectiv formule care sunt false în orice structură.

Exemplul 1.2.5. Formula $equals(x, y) \vee \neg equals(x, y) \in \texttt{LP1}$ este adevărată în orice structură S.

Definiția 1.2.5 (Formulă validă)**.** Fie $F \in \texttt{LP1}$ o formulă. Spunem că F este o *formulă validă* (sau, în mod echivalent, o *tautologie*) dacă, pentru orice structură $S = (U, I)$, avem că $S(F) = 1$.

Exercițiul 1.2.3. *Dați exemplu de o formulă care* nu este validă.

Definiția 1.2.6 (Formulă satisfiabilă)**.** Fie $F \in \texttt{LP1}$ o formulă. Spunem că F este o *formulă satisfiabilă* dacă există o structură $S = (U, I)$ a.î. $S(F) = 1$.

Exercițiul 1.2.4. *Dați exemplu de o formulă satisfiabilă.*

Observația 1.2.2. O formulă care nu este satisfibilă se numește *formulă nesatisfiabilă*, sau, în mod echivalent, *contradicție*.

Exercițiul 1.2.5. *Dați exemplu de o formulă nesatisfiabilă.*

Definiția 1.2.7 (Formule echivalente)**.** Două formule $F_1, F_2 \in \texttt{LP1}$ se numesc *(tare) echivalente* dacă au aceeași valoare de adevăr în orice structură: pentru orice structură $S = (U, I)$, $S(F_1) = S(F_2)$. Faptul că F_1 este echivalentă cu F_2 se notează cu $F_1 \equiv F_2$.

Exemplul 1.2.6. 1. $P(x) \vee Q \equiv Q \vee P(x)$;

2. $\forall x.P(x) \equiv \forall y.P(y)$.

theorem 1.2.1 (Teorema Echivalențelor)**.** *Pentru orice $F, G, H \in \texttt{LP1}$, pentru orice $x \in \mathcal{X}$,*

1.

$$F \wedge G \equiv G \wedge F,$$
$$F \vee G \equiv G \vee F;$$

2.

$$F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H,$$
$$F \vee (G \vee H) \equiv (F \vee G) \vee H;$$

3.

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H),$$
$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H),$$
$$\neg(F \wedge G) \equiv (\neg F) \vee (\neg G),$$
$$\neg(F \vee G) \equiv (\neg F) \wedge (\neg G);$$

4.

$$\neg(\forall x.F) \equiv \exists x.(\neg F),$$
$$\neg(\exists x.F) \equiv \forall x.(\neg F);$$

5. dacă $x \notin free(G)$, atunci:

$$(\forall x.F) \vee G \equiv \forall x.(F \vee G),$$
$$(\exists x.F) \vee G \equiv \exists x.(F \vee G),$$
$$(\forall x.F) \wedge G \equiv \forall x.(F \wedge G),$$
$$(\exists x.F) \wedge G \equiv \exists x.(F \wedge G).$$

Observația 1.2.3. În noțiunea de structură există condiția ca universul să nu fie mulțimea vidă. Dacă am permite ca universul să fie vid, ar mai fi adevărate echivalențele de la penultimul punct din teorema de mai sus?

Definiția 1.2.8 (Consecință semantică)**.** Fie $n+1$ formule $F_1, \ldots, F_n, F \in \texttt{LP1}$ ($n \geq 0$). Formula F este *consecință semantică* a formulelor $F_1, \ldots, F_n$ dacă, pentru orice structură $S = (U, I)$, dacă formulele $F_1, \ldots F_n$ sunt adevărate în S, atunci și F este adevărată în S.

Faptul că F este consecință semantică din $F_1, \ldots, F_n$ se notează cu $F_1, \ldots, F_n \models F$.

Exercițiul 1.2.6. *Arătați că* $\forall x.P(x) \models P(c)$*, unde* P *este un simbol predicativ unar,* c *este un simbol funcțional de aritate* 0*, iar* x *este o variabilă.*

Observația 1.2.4. Dacă $n = 0$, faptul că F este consecință semantică din formulele $F_1, \ldots, F_n$ (cele 0 formule) este echivalent cu faptul că F este validă. Această observație justifică notația $\models F$ pentru faptul că F este validă.

Definiția 1.2.9 (Model)**.** O structură S este *model pentru formula* F dacă $S(F) = 1$.

Exercițiul 1.2.7. *Dați exemple de modele pentru formulele:*

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

Definiția 1.2.10. O formulă $F \in \texttt{LP1}$ este *închisă* dacă $\mathit{free}(F) = \emptyset$. O formulă F este *deschisă* dacă nu este închisă (are cel puțin o variabilă liberă).

Definiția 1.2.11. *Închiderea universală* a formulei F este formula $\forall x_1.\forall x_2.\ldots.\forall x_n.F$, unde $\mathit{free}(F) = \{x_1, \ldots, x_n\}$.

Definiția 1.2.12. *Închiderea existențială* a formulei F este formula $\exists x_1.\exists x_2.\ldots.\exists x_n.F$, unde $\mathit{free}(F) = \{x_1, \ldots, x_n\}$.

Exercițiul 1.2.8. *Arătați că o formulă* F *este satisfiabilă ddacă închiderea existențială a formulei* F *este satisfiabilă.*

Definiția 1.2.13. Fie $F \in \texttt{LP1}$ o formulă. *Matricea* lui F este formula notată cu F^*, obținută din F prin ștergerea tuturor cuantificatorilor. Formal, avem că:

1. $(\forall x.G)^* = G^*$ pentru orice $G \in \texttt{LP1}$;

2. $(\exists x.G)^* = G^*$ pentru orice $G \in \texttt{LP1}$;
3. $(\neg G)^* = (\neg(G^*))$ pentru orice $G \in \texttt{LP1}$;
4. $(G_1 \wedge G_2)^* = (G_1^* \wedge G_2^*)$ pentru orice $G_1, G_2 \in \texttt{LP1}$;
5. $(G_1 \vee G_2)^* = (G_1^* \vee G_2^*)$ pentru orice $G_1, G_2 \in \texttt{LP1}$;
6. $P(t_1, \ldots, t_n)^* = P(t_1, \ldots, t_n)$ pentru orice simbol predicativ $P \in \mathcal{F}_n$ și orice termeni $t_1, \ldots t_n \in \mathcal{T}$.

Exemplul 1.2.7. $\Big(P(a) \wedge \forall x.(P(x) \rightarrow \exists y.R(x,y))\Big)^* = P(a) \wedge (P(x) \rightarrow R(x,y))$

În continuare, cursul este dedicat rezolvării următoarei probleme:
`Input`: O formulă $F \in \texttt{LP1}$.
`Output`: Da, dacă formula este validă.
Din păcate, problema de mai sus este *nedecidabilă*: nu există niciun algoritm care să o rezolve.
Totuși, vom găsi un *semialgoritm* pentru problema de mai sus:

1. Semialgoritmul se oprește cu răspunsul "da" ddacă formula este validă;
2. Dacă răspunsul este "nu", atunci formula nu este validă;
3. Dacă formula nu este validă, există posibilitatea ca semialgoritmul să bucleze la infinit.

De ce este mai puțin util un semialgoritm decât un algoritm? Un algoritm se oprește pentru orice date de intrare. Dacă rulăm semialgoritmul de mai sus pentru o anumită formulă și algoritmul nu se oprește în timp rezonabil, nu putem ști dacă formula nu este validă sau dacă doar nu l-am lăsat suficient de mult timp pentru a se opri cu răspunsul "da".
De fapt, din rațiuni istorice, ne vom concentra asupra problemei nesatisfiabilității. Dar F este validă ddacă $\neg F$ este nesatisfiabilă și deci dacă rezolvăm una din probleme am rezolvat-o și pe cealaltă.

1.3 Forme normale

Definiția 1.3.1 (Literal)**.** O formulă $F \in \texttt{LP1}$ se numește *literal* dacă există un predicat $P \in \mathcal{P}_n$ de aritate $n \geq 0$ și n termeni $t_1, \ldots, t_n$ astfel încât

$$F = P(t_1, \ldots, t_n)$$

sau

$$F = \neg P(t_1, \ldots, t_n).$$

Informal, spunem că un literal este o formulă atomică sau negația unei formule atomice.

Exemplul 1.3.1. Exemple de literali:

$$P \qquad \neg Q(x, f(y)) \qquad R(a, f(x), b)$$

$$\neg P \qquad Q(x, f(y)) \qquad R(a, f(x), b)$$

Exemple de formule care nu sunt literali:

$$P \wedge Q(x, y) \qquad \neg\neg P(x) \qquad \forall x.Q(x, x)$$

Definiția 1.3.2 (Clauză)**.** O formulă $F \in \texttt{LP1}$ se numește *clauză* dacă există n literali $L_1, \ldots, L_n \in \texttt{LP1}$ astfel încât

$$F = L_1 \vee L_2 \vee \ldots \vee L_n.$$

Exemplul 1.3.2.

$$F_1 = P(x) \vee Q(x, a) \vee \neg R(x, f(a, y)); \quad F_2 = P(x); \quad F_3 = \square.$$

Definiția 1.3.3 (FNC)**.** O formulă F este *în formă normală clauzală* (sau, echivalent, *în formă normală conjunctivă*) dacă există n clauze $C_1, \ldots, C_n$ astfel încât

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_n.$$

Exemplul 1.3.3. Următoarele formule sunt în FNC:

$$F_1 = (P(x) \vee Q(x)) \wedge (\neg P(x) \vee R(x, y))$$

$$F_2 = (P(x) \vee Q(x) \vee Q(a)) \wedge (\neg P(x)) \wedge (\neg Q(x) \vee R(x, z)).$$

Definiția 1.3.4 (FNS)**.** O formulă F este în formă normală Skolem (FNS) dacă

$$F = \forall x_1. \ldots . \forall x_n.G,$$

unde:

1. $G^* = G$ (G nu conține cuantificatori) și
2. $\mathit{free}(G) = \{x_1, \ldots, x_n\}$.

Exemplul 1.3.4. Exemple de formule în FNS:

$$F_1 = \forall x.P(x) \qquad F_2 = \forall x. \forall y. \Big(P(a) \wedge \neg (R(x) \vee Q(y)) \Big)$$

Exemple de formule care nu sunt în FNS:

$$F_1 = \exists x.P(x) \qquad F_2 = \forall x. \Big(P(a) \wedge \neg (R(x) \vee Q(y)) \Big)$$

$$F_3 = P(a) \wedge \forall x.P(x)$$

Definiția 1.3.5. O formulă F este în formă normală Skolem clauzală (FNSC) dacă

1. F este în formă normală Skolem și
2. F^* este în formă normală clauzală.

Exemplul 1.3.5.

$$F_1 = \forall x.\forall y.\Big((P(x) \vee \neg Q(x,y)) \wedge (Q(a,y) \vee \neg P(a))\Big)$$

$$F_2 = \forall x.\forall y.\Big(P(a) \wedge (\neg R(x) \vee Q(y))\Big)$$

Exemple de formule care nu sunt în FNSC:

$$F_1 = \exists x.P(x) \qquad F_2 = \forall x.\Big(P(a) \wedge (\neg R(x) \vee Q(y))\Big)$$

$$F_3 = \forall x.\forall y.\Big(P(a) \wedge \neg(R(x) \vee Q(y))\Big)$$

Definiția 1.3.6. Fie $F_1, F_2 \in$ `LP1`. Formula F_1 este *slab echivalentă* cu F_2 (notat $F_1 \equiv_s F_2$) dacă exact una din următoarele condiții este adevărată:

1. F_1 și F_2 sunt satisfiabile
2. F_1 și F_2 sunt nesatisfiabile

Cu alte cuvinte, două formule sunt slab echivalente dacă sunt *echisatisfiabile* (dacă una dintre ele este satisfiabilă atunci și cealaltă este satisfiabilă și invers).

theorem 1.3.1. *Pentru orice formulă $F \in$ `LP1` există o formulă $G \in$ `LP1` aflată în FNSC astfel încât*

$$F \equiv_s G.$$

Proof. Vom demonstra constructiv teorema de mai sus, printr-o serie de pași care transformă o formulă oarecare într-o formulă FNSC echisatisfiabilă cu formula de la care am pornit.

□

1.3.1 Poziții în formule

Reminder:

1. noțiunea de poziție (Definiția 1.1.3);
2. noțiunea de poziții ale unui termen (Definiția 1.1.4) și noțiunea de poziții ale unei formule (Definiția 1.1.6);
3. noțiunea de (sub)termen/(sub)formulă aflată la o poziție într-un termen sau într-o formulă (Definiția 1.1.5 și Definiția 1.1.7);

Definiția 1.3.7 (Apariție a unei formule)**.** Fie $F \in \texttt{LP1}$ o formulă și $p \in pos(F)$ o poziție a formulei F la care se află o formulă $G \in \texttt{LP1}$: $F|_p = G$. Atunci spunem că *p este o apariție a lui G* în F.

Definiția 1.3.8 (Înlocuirea apariției unei formule)**.** Fie $F, G \in \texttt{LP1}$ două formule și $p \in pos(F)$ o apariție a formulei G în F. Atunci formula obținută din F prin înlocuirea aparației p a lui G cu H, notată cu $F[H]_p$, este definită inductiv astfel:

1. $(F_1)[H]_\epsilon = H$;
2. $(F_1 \wedge F_2)[H]_{1\cdot p} = (F_1[H]_p) \wedge F_2$;
3. $(\neg F_1)[H]_{1\cdot p} = \neg((F_1)[H]_p)$;
4. $(\forall x.F_1)[H]_{1\cdot p} = \forall x.((F_1)[H]_p)$;
5. ....

Exemplul 1.3.6. $\Big(\big(P(x)\vee Q(y)\big) \to \forall z.R(x,y,z)\Big)[P(a)]_{2\cdot 1} = \Big(\big(P(x)\vee Q(y)\big) \to \forall z.P(a)\Big)$

Exercițiul 1.3.1. *Completați și celelalte cazuri în Definiția 1.3.8.*

theorem 1.3.2 (Teorema substituției)**.** *Fie $F, G, H \in \texttt{LP1}$, $p \in pos(F)$ astfel încât:*

1. *$G \equiv H$.*
2. *$G = F|_p$ și*

Atunci $F = F[G]_p \equiv F[H]_p$.

Exemplul 1.3.7. Fie $G = P(x)\vee Q(x)$ și $H = Q(x)\vee P(x)$. Fie $F = \forall x.(P(x)\vee Q(x))$ și poziția $p = 1$. Avem că $F|_p = G$ și că $G \equiv H$. Prin teorema substituției, putem concluziona că $\forall x.(P(x) \vee Q(x)) \equiv \forall x.(Q(x) \vee P(x))$.

Notația 1.3.1. *Notație: prin $[x \mapsto t]$ vom nota substituția $\sigma : \mathcal{X} \to T$ de domeniu $dom(\sigma) = \{x\}$ definită prin $\sigma(x) = t$.*

lemma 1.3.3 (Lema de redenumire a variabilelor legate)**.** *Dacă $x, y \in \mathcal{X}$ sunt două variabile, $F \in \texttt{LP1}$ este o formulă astfel încât $y \notin free(F)$, atunci*

$$\forall x.F \equiv \forall y.(F[x \mapsto y]) \qquad \text{și} \qquad \exists x.F \equiv \exists y.(F[x \mapsto y]).$$

Exemplul 1.3.8. Exemple de aplicare ale lemei redenumirii de variabile:

1. $\forall x.P(x,z) \equiv \forall y.P(y,z)$.
2. $\forall x.P(x,z) \not\equiv \forall z.P(z,z)$.

3. $\forall x.(P(x,z) \wedge \exists y.Q(y)) \equiv \forall y.(P(y,z) \wedge \exists y.Q(y))$.

Definiția 1.3.9 (FNP)**.** O formulă $F \in \texttt{LP1}$ este în *formă normală prenex* dacă există variabilele distincte $x_1, \ldots, x_n$, cuantificatorii $Q_1, \ldots, Q_n \in \{\forall, \exists\}$ și o formulă fără cuantificatori $G \in \texttt{LP1}$, astfel încât:

$$F = Q_1 x_1 . \ldots . Q_n x_n . G.$$

theorem 1.3.4 (Teorema de aducere în FNP)**.** *Pentru orice formulă $F_1 \in$ LP1, există o formulă $F_2 \in$ LP1 aflată în FNP astfel încât $F_1 \equiv F_2$.*

Schiță de demonstrație. Aplicăm în mod repetat următoarele transformări, la orice poziție s-ar putea aplica:

1. $\neg(\forall x.F) \equiv \exists x.(\neg F)$;
2. $\neg(\exists x.F) \equiv \forall x.(\neg F)$;
3. $(\forall x.F) \vee G \equiv \forall y.(F[x \mapsto y] \vee G)$ (unde $y \in \mathcal{X} \setminus \mathit{free}(G)$);
4. $(\exists x.F) \vee G \equiv \exists y.(F[x \mapsto y] \vee G)$ (unde $y \in \mathcal{X} \setminus \mathit{free}(G)$);
5. $(\forall x.F) \wedge G \equiv \forall y.(F[x \mapsto y] \wedge G)$ (unde $y \in \mathcal{X} \setminus \mathit{free}(G)$);
6. $(\exists x.F) \wedge G \equiv \exists y.(F[x \mapsto y] \wedge G)$ (unde $y \in \mathcal{X} \setminus \mathit{free}(G)$);

Dacă formula nu este în FNP, cel puțin una din transformări se poate aplica. De asemenea, transformările nu se pot aplica la infinit, deoarece orice transformare "mută" poziția unui cuantificator mai "sus" în arborele formulei (și nu îl poate muta la infinit în sus).

De asemenea, conform Teoremei Echivalenței (1.2.1) și aplicând Teorema Substituției (1.3.2), toate transformările păstrează echivalența cu formula de la care am plecat. Rezultă că procesul de transformare se va termina, formula la care se ajunge fiind în FNP și echivalentă cu formula de la care am plecat.

□

Definiția 1.3.10 (FNS)**.** O formulă F este în *formă normală Skolem* dacă există variabilele distincte $\{x_1, \ldots, x_n\}$ astfel încât:

$$F = \forall x_1 . \forall x_2 . \ldots . \forall x_n . G$$

pentru o formulă $G \in \texttt{LP1}$ astfel încât $\mathit{free}(G) = \{x_1, \ldots, x_n\}$ și $\mathit{bound}(G) = \emptyset$.

Exemplul 1.3.9. Formule în FNS:

1. $\forall x.\forall y.(P(x,y))$
2. $\forall x.\forall y.(P(x,y) \wedge Q(x))$
3. $\forall x.\forall y.\forall z.((P(x,y) \wedge Q(x)) \vee R(x,y,z))$

4. $P(c) \wedge Q(f(a,b))$

lemma 1.3.5 (Skolem)**.** *Fie $F = \forall x_1.\forall x_2.\ldots.\forall x_n.\exists y.G$, unde $n \geq 0$, $G \in$ LP1 (G poate conține alți cuantificatori).*

Fie $f \in \mathcal{F}_n$ un simbol funcțional de aritate n care nu apare în G. Atunci:

$$F \equiv_s \forall x_1.\forall x_2.\ldots\forall x_n.G[y \mapsto f(x_1,\ldots,x_n)].$$

Schiță de demonstrație. Implicația directă:

Presupunem că exists o structură S astfel încât $S(F) = 1$.

Găsim o structură S' astfel încât $S'(\forall x_1.\forall x_2.\ldots\forall x_n.G[y \mapsto f(x_1,\ldots,x_n)]) = 1$.

Implicația inversă:

Presupunem că exists o structură S' astfel încât $S'(\forall x_1.\forall x_2.\ldots\forall x_n.G[y \mapsto f(x_1,\ldots,x_n)]) = 1$.

Găsim o structură S astfel încât $S(F) = 1$.

□

Exercițiul 1.3.2. *Completați demonstrația de mai sus.*

theorem 1.3.6. *Pentru orice formulă $F \in$ LP1, există o formulă $F' \in$ LP1 în formă normală Skolem astfel încât*

$$F \equiv_s F'.$$

Schiță de demonstrație.

1. Calculăm o formulă F', aflată în FNP și echivalentă cu formula F (folosind Teorema 1.3.4);
2. Calculăm o formulă F'', închiderea existențială a lui F' (F'' va fi echisatisfabilă cu F);
3. Aplicăm în mod repetat lema de Skolemizare (Lema 1.3.5).

Rezultatul va fi o formulă aflată în FNS, echisatisfabilă cu formula de la care am plecat.

□

Definiția 1.3.11 (FNSC, reminder)**.** O formulă F este în formă normală Skolem clauzală dacă F este în FNS și F^* este în FNC.

Definiția 1.3.12 (FNC (reminder))**.** O formulă $F \in$ LP1 este în *formă normală conjunctivă* (sau *formă normală clauzală*) (FNC) dacă există n clauze $C_1,\ldots,C_n$ astfel încât:

$$F = C_1 \wedge \ldots \wedge C_n.$$

Exemplul 1.3.10. Formulă aflată în FNSC: $\forall x.\forall y.\forall z.(P(x,y) \wedge (Q(x) \vee R(x,y,z)) \wedge \neg Q(x))$.

Exercițiul 1.3.3. *Găsiți un algoritm, bazat pe echivalențele demonstrate în Teorema Echivalenței (Teorema 1.2.1), care să transforme orice formulă aflată în FNS într-o formulă echivalentă, aflată în FNSC.*

Rezumat:

F	orice formulă $\in$ LP1
$\equiv$	
F_p	o formulă în FN prenex
$\equiv_s$	
F_s	o formulă în FN Skolem
$\equiv$	
F_{FNSC}	o formulă în FN Skolem clauzală

1.4 Rezoluție în LP1

1.4.1 Structuri Herbrand

Definiția 1.4.1 (Domeniu Herbrand)**.** Domeniul Herbrand asociat signaturii $\mathcal{F}$, notat $\mathcal{D}(\mathcal{F})$, este mulțimea de termeni $\mathcal{D}(\mathcal{F}) \subseteq T$ definită inductiv astfel:

1. $\mathcal{F}_0 \subseteq \mathcal{D}(\mathcal{F})$;
2. dacă $\mathcal{F}_0 = \emptyset$, atunci $a \in \mathcal{D}(\mathcal{F})$ (se consideră un simbol constant a);
3. dacă $f \in \mathcal{F}_n$ și $t_1, \ldots, t_n \in \mathcal{D}(\mathcal{F})$, atunci $f(t_1, \ldots, t_n) \in \mathcal{D}(\mathcal{F})$.

Observația 1.4.1. Domeniul Herbrand este mulțimea *termenilor închiși* (termeni fără variabile).

Exemplul 1.4.1. Continuând exemplul precedent, $f(e, f(e, e)) \in \mathcal{D}(\mathcal{F})$, dar $f(x, f(e, y)) \notin \mathcal{D}(\mathcal{F})$.

Definiția 1.4.2. O structură $S = (U, I)$ este *structură Herbrand* dacă:

1. $U = \mathcal{D}(\mathcal{F})$;
2. $S(t) = t$ pentru orice termen $t \in \mathcal{T} \cap \mathcal{D}(\mathcal{F})$.

theorem 1.4.1 (Herbrand)**.** *O formulă $F \in$ LP1, aflată în FNS, are model ddacă are model Herbrand.*

Schiță de demonstrație. Implicația inversă este trivială. Implicația directă o vom arăta în continuarea cursului.

□

Teorema de mai sus este importantă deoarece, dacă dorim să verificăm satisfiabilitatea unei formule, este suficient să analizăm "mai puține" structuri (este suficient să analizăm structurile Herbrand), ceea ce conduce la o procedură de semidecizie pentru logica de ordinul I.

Fie $F = \forall x_1 \ldots . \forall x_n . (C_1 \wedge \ldots \wedge C_m)$ o formulă în FNSC. Scriem și

$$F = \{C_1, \ldots, C_m\}.$$

Dacă $C_1 = L_1^1 \vee \ldots \vee L_{k_1}^1$, $C_2 = L_1^2 \vee \ldots \vee L_{k_2}^2$, ..., $C_n = L_1^m \vee \ldots \vee L_{k_m}^m$, scriem și

$$C_i = \{L_i^i, \ldots, L_{k_i}^i\}$$

și respectiv:

$$F = \{\{L_1^1, \ldots, L_{k_1}^1\}, \ldots, \{L_1^m, \ldots, L_{k_m}^m\}, \}.$$

Exemplul 1.4.2. Fie $F = \forall x.\forall y.(P(x) \wedge (\neg Q(x) \vee R(y)))$. Mai notăm $F = \Big\{\{P(x)\}, \{\neg Q(x), R(y)\}\Big\}$.

Definiția 1.4.3. Fie $F = \forall x_1.\ldots.\forall x_n.G$, cu $G = F^*$, o formulă în FNS. Atunci *extensia Hebrand* a lui F este mulțimea:

$$E(F) = \Big\{G\sigma \mid dom(\sigma) = \{x_1, \ldots, x_n\}, \sigma(x_i) \in \mathcal{D}(\mathcal{F})\Big\}.$$

Exemplul 1.4.3. Fie $\mathcal{F}_0 = \{c\}, \mathcal{F}_1 = \{h\}, \mathcal{F}_2 = \{f\}$. Atunci:

$$\mathcal{D}(\mathcal{F}) = \{c, h(c), h(h(c)), f(c,c), f(h(c),c), \ldots\},$$

$$E(\forall x.P(x)) = \{P(c), P(h(c)), P(h(h(c))), P(f(c,c)), P(f(h(c),c)), \ldots\}.$$

1.4.2 Rezoluția de bază

Regula rezoluției de bază este următoarea regulă de inferență:

$$\frac{P(t_1, \ldots, t_n) \vee C \qquad \neg P(t_1, \ldots, t_n) \vee D}{C \vee D}\ \textsc{Rezolutie de baza}$$

Regula rezoluției de bază permite obținerea concluziei $C \vee D$ din premisele $P(t_1, \ldots, t_n) \vee C$ și $\neg P(t_1, \ldots, t_n) \vee D$.

theorem 1.4.2 (Herbrand)**.** *F este nesatisfiabilă dacă și numai dacă există o submulțime finită M a lui $E'(F)$ astfel încât se poate obține $\square$ din M aplicând regula rezoluției de bază.*

Exemplul 1.4.4. Fie următoarea formulă:

$$F = \forall x.\forall y.\forall z.(P(x) \wedge (\neg P(y) \vee Q(y)) \wedge \neg Q(z)).$$

Avem că

$$E'(F) = \{P(c), \neg P(c) \vee Q(c), \neg Q(c), \ldots\}.$$

Să arătăm că F este nesatisfiabilă, folosind Teorema 1.4.2. Alegem mulțimea $M = \{P(c), \neg P(c) \vee Q(c), \neg Q(c)\} \subseteq E'(F)$. Următoarea derivare prin rezoluție de bază a clauzei vide ($\square$), pornind de la clauzele mulțimii M, arată că F este nesatisfiabilă.

1. $P(c)$ parte din M
2. $\neg P(c) \vee Q(c)$ parte din M

3. $\neg Q(c)$ parte din M
4. $Q(c)$ rezoluție între 1 și 2
5. $\square$ rezoluție între 3 și 4

Semialgoritmul lui Gilmore este o procedură pentru următoarea problemă computațională:

- Input: o formulă F în FNSC
- Output: da, dacă formula nu este satisfiabilă (altfel nu se oprește)

Semialgoritmul lui Gilmore:

1. $M \leftarrow \emptyset$ o mulțime de clauze
2. $i \leftarrow 0$
3. cât timp $\square \notin Res^i(M)$:
   (a) alege $C \in E'(F) \setminus M$ (dacă nu există un astfel de C și $Res^i(M) = Res^{i+1}(M)$, întoarce NU)
   (b) $M \leftarrow M \cup \{C\}$
   (c) $i \leftarrow i + 1$
4. întoarce DA

Semialgoritmul lui Gilmore folosește două variabile: M (o mulțime de clauze) și un număr i (numărul de pași de rezoluție). Prin Res^i se notează rezolvenții în i pași, similar cu LP.

Dacă algoritmul se oprește cu răspunsul DA, înseamnă că se poate obține clauza vidă din mulțimea M de clauze (în cel mult i pași) și deci, prin Teorema 1.4.2, formula este nesatisfiabilă. Invers, dacă formula este nesatisfiabilă, există o submulțime a mulțimii $E'(F)$ din care se poate obține clauza vidă (prin teorema lui Herbrand). În acest caz, după un anumit număr de pași, variabila M va conține (cel puțin) acele clauze iar valoarea din variabila i va fi suficient de mare pentru a permite derivarea clauzei vide în cel mult i pași. În acel punct, algoritmul se va opri și va întoarce DA. Am arătat că algoritmul se oprește cu răspunsul DA ddacă formula F este nesatisfiabilă.

Invers, dacă formula F este satisfiabilă, algoritmul nu se oprește (în general). Singurul caz în care formula este satisfiabilă și algoritmul se oprește este când alegerea de la linia 3.1 nu se poate efectua (deci $E'(F)$ este finită) și nu se pot obține noi rezolvenți în $i + 1$ pași.

Observația 1.4.2. Când este $E'(F)$ finită?

Cum determinăm dacă o formulă F este validă? Construim $\neg F$ (F este validă dacă și numai dacă $\neg F$ nu este satisfiabilă). Calculăm F', o formă normală Skolem clauzală pentru $\neg F$. Aplicăm semialgoritmul lui Gilmore pentru F'

- Dacă algoritmul se termină cu răspunsul *NU*, atunci $\neg F$ nu este satisfiabilă și deci F este validă (suntem bucuroși)
- Altfel...

Exercițiul 1.4.1. *Arătați că* $\forall x.(P(x) \to Q(x)) \wedge P(s) \to Q(s)$ *este validă.*

1.4.3 Unificare

Semialgoritmul lui Gilmore este doar de interes teoretic, dar avem nevoie de o metodă mai practică pentru a testa (ne)satisfiabilitatea unei formule. Semialgoritmul lui Gilmore (sau algoritmi similari) a fost folosit în anii de debut ai domeniului Inteligență Aritificială pentru verificarea nesatisfiabilității unor formule, dar apoi Robinson a inventat o metodă mult mai eficientă, bazată pe unificare.

Noutate: vom nota $\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$ substituția σ de domeniu $\{x_1, \ldots, x_n\}$.

Definiția 1.4.4. O substituție σ este unificator pentru t_1 și t_2 dacă $t_1\sigma = t_2\sigma$.

Exemplul 1.4.5. Fie termenii $t_1 = f(x, h(y))$ și $t_2 = f(h(z), z')$. Un unificator pentru t_1 și t_2 ar fi:

$$\sigma = \{x \mapsto h(z), z' \mapsto h(y)\} \qquad (t_1\sigma = f(h(z), h(y)) = t_2\sigma)$$

Un altul:

$$\sigma' = \{z \mapsto a, x \mapsto h(a), z' \mapsto h(y)\} \qquad (t_1\sigma' = f(h(a), h(y)) = t_2\sigma').$$

Termenii $t_1 = f(x, y)$ și $t_2 = h(z)$ nu au unificator. (De ce? Pentru orice substituție σ avem $t_1\sigma = f(x, y)\sigma = f(x\sigma, y\sigma) \neq h(z\sigma) = h(z)\sigma = t_2\sigma$.)

Termenii $t_1 = x$ și $t_2 = h(x)$ nu au unificator (De ce? Pentru orice substituție σ, avem că $len(x\sigma) < len(h(x\sigma)) = len(h(x)\sigma)$).

Definiția 1.4.5. O problemă de unificare P este:

- sau o mulțime
$$P = \{t_1 \doteq t_1', \ldots, t_n \doteq t_n'\}$$
formată din n perechi de termeni
- sau simbolul special
$$P = \bot.$$

Definiția 1.4.6. O problemă de unificare are soluție (sau are unificator) dacă este de forma
$$P = \{t_1 \doteq t_1', \ldots, t_n \doteq t_n'\}$$
și există o substituție σ care să fie unificator pentru t_i și t_i' pentru orice $i \in \{1, \ldots, n\}$, adică $t_1\sigma = t_1'\sigma, \ldots, t_n\sigma = t_n'\sigma$.

Exemplul 1.4.6. Dacă σ și σ' sunt doi unificatori pentru termenii t_1 și t_2, σ este un *unificator mai general* decât σ' dacă există o substituție σ'' astfel încât $t_1\sigma' = t_1\sigma\sigma'' = t_2\sigma\sigma'' = t_2\sigma'$.

Exemplul 1.4.7. Fie $t_1 = f(x, a)$ și $t_2 = f(y, a)$. Unificatorul $\{y \mapsto x\}$ este mai general decât $\{x \mapsto a, y \mapsto a\}$.

Definiția 1.4.7. Substituția σ este cel mai general unificator pentru o problemă de unificare $P = \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$ dacă:

1. σ este unificator pentru P: $t_i\sigma = t'_i\sigma$, pentru orice $1 \leq i \leq n$;
2. σ este mai general decât orice alt unificator pentru P.

Definiția 1.4.8. Cu $mgu(P)$ notăm mulțimea unificatorilor cei mai generali pentru P. Pentru $P = \bot$, $mgu(P) = \emptyset$.

Exemplul 1.4.8. Fie $P = \{f(x, a) \doteq f(y, a)\}$. Avem că $mgu(P) = \{\{x \mapsto z, y \mapsto z\}, \{x \mapsto y\}, \ldots\}$.

Definiția 1.4.9. O problemă de unificare P este în formă rezolvată dacă $P = \bot$ sau $P = \{x_1 \doteq t'_1, \ldots, x_n \doteq t'_n\}$ și $x_i \notin vars(t_j)$ pentru orice $i, j \in \{1, \ldots, n\}$.

lemma 1.4.3. *Dacă $P = \{x_1 \doteq t'_1, \ldots, x_n \doteq t'_n\}$ este în formă rezolvată, atunci $\{x_1 \mapsto t'_1, \ldots, x_n \mapsto t'_n\} \in mgu(P)$.*

Următoarele reguli pot fi folosite pentru aducerea unei probleme de unificare în formă rezolvată:

STERGERE	$P \cup \{t \doteq t\} \Rightarrow P$
DESCOMPUNERE	$P \cup \{f(t_1, \ldots, t_n) \doteq f(t'_1, \ldots, t'_n)\} \Rightarrow$
	$P \cup \{t_1 \doteq t'_1, \ldots, t_n \doteq t'_n\}$
ORIENTARE	$P \cup \{f(t_1, \ldots, t_n) \doteq x\} \Rightarrow P \cup \{x \doteq f(t_1, \ldots, t_n)\}$
ELIMINARE	$P \cup \{x \doteq t\} \Rightarrow P\{x \mapsto t\} \cup \{x \doteq t\}$
	daca $x \notin vars(t), x \in vars(P)$
CONFLICT	$P \cup \{f(t_1, \ldots, t_n) \doteq g(t'_1, \ldots, t'_m)\} \Rightarrow \bot$
OCCURS CHECK	$P \cup \{x \doteq f(t_1, \ldots, t_n)\} \Rightarrow \bot$
	daca $x \in vars(f(t_1, \ldots, t_n))$

lemma 1.4.4. *Dacă P nu este în formă rezolvată, atunci există P' astfel încât $P \Rightarrow P'$.*

lemma 1.4.5. *Dacă $P \Rightarrow P'$, atunci $mgu(P) = mgu(P')$.*

lemma 1.4.6. *Nu există o secvență infinită $P \Rightarrow P_1 \Rightarrow P_2 \Rightarrow \ldots \Rightarrow P_i \Rightarrow \ldots$.*

Corolarul 1.4.7. *Regulile precedente constituie un algoritm de calcul al celui mai general unificator pentru o problemă de unificare, dacă acesta există.*

Exemplul 1.4.9.

$$P = \{f(g(x_1, a), x_2) \doteq x_3, f(x_2, x_2) \doteq f(a, x_1)\} \overset{\text{DESCOMPUNERE}}{\Rightarrow}$$

$$\{f(g(x_1, a), x_2) \doteq x_3, x_2 \doteq a, x_2 \doteq x_1\} \overset{\text{ELIMINARE}}{\Rightarrow}$$

$$\{f(g(x_1, a), a) \doteq x_3, x_2 \doteq a, a \doteq x_1\} \overset{\text{ORIENTARE}}{\Rightarrow}$$

$$\{f(g(x_1, a), a) \doteq x_3, x_2 \doteq a, x_1 \doteq a\} \overset{\text{ELIMINARE}}{\Rightarrow}$$

$$\{f(g(a, a), a) \doteq x_3, x_2 \doteq a, x_1 \doteq a\} \overset{\text{ORIENTARE}}{\Rightarrow}$$

$$\{x_3 \doteq f(g(a, a), a), x_2 \doteq a, x_1 \doteq a\}$$

Concluzie: $\{x_3 \mapsto f(g(a, a), a), x_2 \mapsto a, x_1 \mapsto a\} \in mgu(P)$.

Exemplul 1.4.10.

$$P = \{f(g(x_1, a), x_2) \doteq x_3, f(x_2) \doteq f(x_3)\} \overset{\text{DESCOMPUNERE}}{\Rightarrow}$$

$$\{f(g(x_1, a), x_2) \doteq x_3, x_2 \doteq x_3\} \overset{\text{ORIENTARE}}{\Rightarrow}$$

$$\{x_3 \doteq f(g(x_1, a), x_2), x_2 \doteq x_3\} \overset{\text{ELIMINARE}}{\Rightarrow}$$

Explicati de ce nu se mai poate aplica orientare

$$\{x_3 \doteq f(g(x_1, a), x_3), x_2 \doteq x_3\} \overset{\text{OCCURS CHECK}}{\Rightarrow}$$

$$\perp$$

Concluzie: $mgu(P) = \emptyset$.

Exemplul 1.4.11.

$$P = \{f(g(x_1, a), x_2) \doteq x_3, f(g(x_4, x_5)) \doteq f(x_3)\} \overset{\text{DESCOMPUNERE}}{\Rightarrow}$$

$$\{f(g(x_1, a), x_2) \doteq x_3, g(x_4, x_5) \doteq x_3\} \overset{\text{ORIENTARE}}{\Rightarrow}$$

$$\{f(g(x_1, a), x_2) \doteq x_3, x_3 \doteq g(x_4, x_5)\} \overset{\text{ELIMINARE}}{\Rightarrow}$$

$$\{f(g(x_1, a), x_2) \doteq g(x_4, x_5), x_3 \doteq g(x_4, x_5)\} \overset{\text{CONFLICT}}{\Rightarrow}$$

$$\bot$$

Concluzie: $mgu(P) = \emptyset$.

(BINARY) RESOLUTION

$$\frac{P(t_1, \ldots, t_n) \vee C \quad \neg P(t'_1, \ldots, t'_n) \vee D \quad \sigma \in mgu\{t_1 \doteq t'_1, \ldots t_n \doteq t'_n\} \quad var(P(t_1, \ldots, t_n) \vee C) \cap var(\neg P(t'_1, \ldots, t'_n) \vee D) = \emptyset}{(C \vee D)\sigma}$$

(POSITIVE) FACTORING

$$\frac{P(t_1, \ldots, t_n) \vee P(t'_1, \ldots, t'_n) \vee C \qquad \sigma \in mgu\{t_1 \doteq t'_1, \ldots t_n \doteq t'_n\}}{(P(t_1, \ldots, t_n) \vee C)\sigma}$$

Să demonstrăm că $\forall x.(P(x) \wedge (\neg P(h(x)) \vee Q(f(x))) \wedge (\neg Q(f(g(a)))))$ este nesatifiabilă prin rezoluție:

1. $P(x)$
2. $\neg P(h(x)) \vee Q(f(x))$
3. $\neg Q(f(g(a)))$
4. $Q(f(x))$ rezoluție între 1 și 2: $\dfrac{P(x') \qquad \neg P(h(x)) \vee Q(f(x)) \qquad \{x' \mapsto h(x)\} \in mgu\{x' \doteq h(x)\}}{Q(f(x))\{x' \mapsto h(x)\}}$
5. $\square$ rezoluție între 3 și 4: $\dfrac{Q(f(g(a))) \qquad Q(f(x)) \qquad \{x \mapsto g(a)\} \in mgu\{f(g(a)) \doteq f(x)\}}{\square\{x \mapsto g(a)\}}$

1. Fie F în FNSC reprezentată ca mulțime de clauze: $F = \{C_1, \ldots, C_n\}$.
2. F este nesatisfiabilă dacă și numai dacă $\square$ se poate obține din $\{C_1, \ldots, C_n\}$ aplicând regulile RESOLUTION și FACTORING.

Exemplul 1.4.12. Arătați că $\forall x.(P(x) \to Q(x)) \wedge P(s) \to Q(s)$ este validă folosind rezoluția.

1.5 Logica de ordinul I cu egalitate

Logica de ordinul I cu egalitate este o extensie a logicii de ordinul I.

Definiția 1.5.1. Mulțimea formulelor din logica de ordinul I cu egalitate este notată cu $\texttt{LP1}_{=}$. Mulțimea $\texttt{LP1}_{=}$ este definită exact ca mulțimea LP1, dar este obligatoriu să existe un simbol predicativ $= \in \mathcal{P}_2$ (un simbol predicativ special).

Exemplul 1.5.1. Formula $\forall x.\exists y.(P(x) \to =(x, f(y))) \in \texttt{LP1}_{=}$.

Simbolul predicativ $=$ se numește simbol predicativ de egalitate.

Observația 1.5.1. Deseori folosim notația $=(x, y) \overset{notație}{\equiv} x = y$. Astfel, formula de mai sus se poate scrie:

$$\forall x.\exists y.(P(x) \to x = f(y)) \in \texttt{LP1}_{=}.$$

Definiția 1.5.2. O structură $S = (U, I)$ pentru $\texttt{LP1}_{=}$ este o structură pentru LP1 cu proprietatea că:

- $I_{=}(u, v) = 1$ ddacă u este aceeeași valoare cu v, oricum am alege $u, v \in U$.

Observația 1.5.2. Deoarece restricționăm structurile la cele în care simbolul predicativ $=$ este interpretat într-un anumit fel, spunem că simbolul predicativ are o *interpretare standard* (engl. *intended interpretation*). În cazul nostru, interpretarea standard pentru simbolul predicativ $=$ este predicatul de egalitate peste domeniul U.

Exercițiul 1.5.1. *Arătați că* $\forall x.\forall y.(x = y)$ *este adevărată într-o structură* $S = (U, I)$ *ddacă* $|U| = 1$.

Exercițiul 1.5.2. *Arătați că următoarele formule din* $\texttt{LP1}_{=}$ *sunt valide:*

1. $\forall x.(x = x)$;
2. $\forall x.\forall y.(x = y \to y = x)$;
3. $\forall x.\forall y.\forall z.\Big((x = y \wedge y = z) \to x = z\Big)$;
4. $\forall x.\forall y.\forall z.\Big((t_1 = t'_1 \wedge \ldots t_n = t'_n) \to f(t_1, \ldots, t_n) = f(t'_1, \ldots, t'_n)\Big)$, *pentru orice simbol funcțional* f *și pentru orice termeni* $t_1, \ldots t_n \in \mathcal{T}$.

1.6 Teorii logice

Memento 1.6.1. *O formulă* $F \in \texttt{LP1}$ *(sau* $F \in \texttt{LP1}_{=}$*) este* închisă *(engl.* closed formula*) dacă, prin definiție,* $free(F) = \emptyset$.

Exemplul 1.6.1. 1. $\forall x.\forall y.P(x, y)$ este închisă;

2. $\forall x.P(x,y)$ este deschisă.

Definiția 1.6.1. O mulțime $M \subseteq \texttt{LP1}$ (sau $M \subseteq \texttt{LP1}_{=}$) de formule închise se numește *teorie logică* dacă:

- pentru orice $F_1, \ldots, F_n \in M$ și pentru orice F astfel încât $F_1 \ldots, F_n \models F$, $F \in M$.

Observația 1.6.1. În mod echivalent, spunem că M este teorie logică ddacă M este *închisă* la consecință semantică.

Exemplul 1.6.2. Mulțimea $M_1 = \{P(a)\}$ nu este teorie logică (dați exemplu de o consecință semantică pentru $P(a)$ care nu apare în M_1.

Mulțimea $M_2 = \{P(x)\}$ nu este teorie logică.

Mulțimea $M_3 = \{\forall x.P(x), \forall x.\Big(P(x) \to Q(x)\Big)\}$ nu este teorie logică.

Definiția 1.6.2. Fie M o mulțime de formule închise. Mulțimea $\bar{M}$ se numește închiderea mulțimii M la consecință semantică și este definită inductiv astfel:

1. $M \subseteq \bar{M}$;
2. dacă $F_1, \ldots, F_N \in \bar{M}$ și $F_1, \ldots, F_n \models F$, atunci $F \in \bar{M}$.

Observația 1.6.2. Pentru orice mulțime M de formule închise, $\bar{M}$ este o teorie logică.

Definiția 1.6.3. Fie M o mulțime de formule închise. Spunem că $\bar{M}$ este teoria logică *generată* de mulțimea M.

Observația 1.6.3. Câteodată, prin abuz de limbaj, spunem că M este o teorie logică, deși ne referim la $\bar{M}$. În acest caz, spunem că M sunt *axiomele* teoriei.

1.6.1 Aritmetica Presburger

Aritmetica Presburger este teoria logică (în logica de ordinul I cu egalitate), definită peste signatura funcțională $\mathcal{F}_0 = \{0, 1\}$, $\mathcal{F}_2 = \{+\}$, generată de următoarele axiome $M_{Presburger}$:

1. $\forall x.\Big(x + 0 = x\Big)$;
2. $\forall x.\Big(\neg(0 = x + 1)\Big)$;
3. $\forall x.\forall y.\Big((x + 1 = y + 1) \to x = y\Big)$;
4. $\forall x.\forall y.\Big(x + (y + 1) = (x + y) + 1\Big)$;
5. $\varphi\{x \mapsto 0\} \wedge \forall x.(\varphi\{x \mapsto x\} \to \varphi\{x \mapsto x + 1\}) \to \forall y.(\varphi\{x \mapsto y\})$, pentru orice formulă φ cu $free(\varphi) = \{x\}$ (câte o axiomă pentru fiecare astfel de formulă).

În formulele de mai sus, am folosit notația:

$$+(t_1, t_2) \overset{notație}{=} t_1 + t_2.$$

Ultimul punct din enumerarea de mai sus nu conține o axiomă, ci o *schemă de axiome* (câte o axiomă pentru fiecare formula φ cu proprietățile respective). Astfel, mulțimea de axiome din aritmetica Presburger este infinită.

Definiția 1.6.4. O teorie este finit axiomatizabilă dacă există un număr finit de axiome care generează teoria respectivă.

theorem 1.6.1. *Aritmetica Presburger nu este finit axiomatizabilă (în* $LP1_=$*) (altfel spus, nu putem avea un număr finit de axiome care să fie echivalente cu axiomele de mai sus).*

Definiția 1.6.5. O teorie M este *consistentă* dacă nu există nicio formulă F astfel încât atât $F \in M$, cât și $\neg F \in M$.

theorem 1.6.2. *Aritmetica Presburger este consistentă.*

Definiția 1.6.6. O teorie M este *completă* dacă, pentru orice formulă închisă F, avem că $F \in M$ sau $\neg F \in M$.

theorem 1.6.3. *Aritmetica Presburger este completă.*

Observația 1.6.4. Dacă M este o teorie și logică și $F \in M$ este o formulă închisă din teorie, spunem că F este o *teoremă* a teoriei logice.

Definiția 1.6.7. O teorie M este *decidabilă* dacă există un algoritm care, dându-se la intrare o formulă închisă F, întoarce *DA* dacă $F \in M$ și *NU* dacă $F \notin M$ (e.g. algoritmul determină dacă F este, sau nu, o teoremă).

theorem 1.6.4. *Aritmetica Presburger este decidabilă (Presburger, 1929).*

Observația 1.6.5. Dându-se o formulă închisă F, există un algoritm care determină dacă F este o teoremă în timp $O(2^{2^n})$ (dublu exponențial), unde n este lungimea formulei F. Spunem că acest algoritm decide teoria. Nu există niciun algoritm care decide teorie și care rulează mai rapid de $O(2^{2^{cn}})$ în cazul cel mai defavorabil (teorema Fischer-Rabin, 1974).

Exercițiul 1.6.1. *Arătați că* $\forall x.(x + 1 = 1 + x)$ *este o teoremă în aritmetica Presburger.*

Definiția 1.6.8. O relație (predicat) $R : U^n \to B$ este *definibilă* într-o teorie M dacă există o formulă φ cu variabile libere $x_1, \ldots, x_n$ astfel încât, oricum am alege o structură S model pentru teorie, $R(S(x_1), \ldots, S(x_n)) = 1$ ddacă $S(\varphi) = 1$.

Exemplul 1.6.3. Predicatul de paritate este definibil în aritmetica Presburger:

$$\exists y.(y + y = x_1).$$

Exercițiul 1.6.2. *Arătați că următoarele relații sunt definibile în aritmetica Presburger:*

1. *relația* $\leq$*;*
2. *predicatul de imparitate;*
3. *predicatul de egalitate;*
4. *predicatul "multiplu de* 5*".*

1.6.2 Aritmetica Peano

Aritmetica Peano este teoria logică de ordinul I peste signatura funcțională $\mathcal{F}_0 = \{0, 1\}, \mathcal{F}_2 = +, \times$ generată de axiomele:

1. $\forall x.\Big(x + 0 = x\Big)$;
2. $\forall x.\Big(\neg(0 = x + 1)\Big)$;
3. $\forall x.\forall y.\Big((x + 1 = y + 1) \to x = y\Big)$;
4. $\forall x.\forall y.\Big(x + (y + 1) = (x + y) + 1\Big)$;
5. $\forall x.\Big(x \times 0 = 0\Big)$;
6. $\forall x.\forall y.\Big(x \times (y + 1) = x \times y + x\Big)$;
7. $\varphi\{x \mapsto 0\} \wedge \forall x.(\varphi\{x \mapsto x\} \to \varphi\{x \mapsto x + 1\}) \to \forall y.(\varphi\{x \mapsto y\})$, pentru orice formulă φ cu $\mathit{free}(\varphi) = \{x\}$ (câte o axiomă pentru fiecare astfel de formulă).

Practic, aritmetica Peano îmbogățește aritmetica Presburger cu operația de înmulțire. Totuși, aritmetica Peano nu se bucură de aceleași proprietăți:

theorem 1.6.5. *Aritmetica Peano este nedecidabilă.*

Exercițiul 1.6.3. *Arătați că relația de divizibilitate poate fi exprimată în aritmetica Peano.*

Arătați că predicatul de primalitate poate fi exprimat în aritmetica Peano.

1.7 Demonstratoare automate de teoreme

Demonstratoarele automate pentru logica de ordinul I (cu egalitate) primesc o mulțime de axiome $F_1, \ldots, F_n$ și o conjectură F și încearcă să demonstreze că $F_1, \ldots, F_n \models F$. Cum problema este nedecidabilă, este posibil ca execuția unui demonstrator automat să:

1. se termine cu răspunsul *DA*;
2. se termine cu răspunsul *NU*;
3. nu se termine (în practică se va epuiza memoria);
4. se termine cu răspunsul *nu știu*.

1.7.1 Cum funcționează demonstratoarele automate?

Cele mai multe (și cele mai bune) demonstratoare automate sunt bazate pe rezoluție. În primul rând, ele se bazează pe faptul că $F_1, \ldots, F_n \models F$ ddacă $\models (F_1 \wedge \ldots \wedge F_n) \rightarrow F$. Calculează o formă normală Skolem clauzală pentru negația formulei $(F_1 \wedge \ldots \wedge F_n) \rightarrow F$ și apoi încearcă sa găsească clauza vidă prin rezoluție.

Evident că există nenumărate optimizări (atât la nivel de implementare, prin structuri de date care implementează conceptul de *term indexing*, cât și la nivel de algoritmi/reguli de inferență) față de ce am învățat în curs.

În continuare, vom folosi demonstratorul SPASS, versiunea 3.5 (`http://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/classic-spass-theorem-prover/download/`) pentru a demonstra câteva teoreme interesante. Vom scrie datele de intrare pentru SPASS în format propriu, dar există posibilitatea și de a folosit formatul TPTP (`www.tptporg`), format care poate fi procesat și de alte demonstratoare.

1.7.2 Socrate

Primul lucru pe care vrem să îl verificăm este că $P(s), \forall x.(P(x) \rightarrow Q(x)) \models Q(s)$, care este exemplul cu Socrate pe care l-am văzut la începutul cursului.

Fișierul de intrare este:

```
begin_problem(Socrate).

list_of_descriptions.
name({*socrate.spass*}).
author({*Stefan Ciobaca*}).
status(unknown).
description({*Problema cu Socrate - muritor*}).
end_of_list.

list_of_symbols.
```

```
functions[(s, 0)].
predicates[(Om, 1), (Muritor, 1)].
end_of_list.

list_of_formulae(axioms).
formula(Om(s)).
formula(forall([x], implies(Om(x), Muritor(x)))).
end_of_list.

list_of_formulae(conjectures).
formula(Muritor(s)).
end_of_list.

end_problem.
```

Demonstratorul SPASS poate fi rulat pe fișierul de mai sus (socrate.spass) apelând următoarea comandă la shell-ul sistemului de operare:

```
spass socrate.spass
```

Pentru fișierul de mai sus, demonstratorul produce următorul output:

```
-------------------------SPASS-START-----------------------------
Input Problem:
1[0:Inp] ||  -> Om(s)*.
2[0:Inp] || Muritor(s)* -> .
3[0:Inp] || Om(U) -> Muritor(U)*.
 This is a monadic Horn problem without equality.
 This is a problem that has, if any, a finite domain model.
 There are no function symbols.
 This is a problem that contains sort information.
 The conjecture is ground.
 The following monadic predicates have finite extensions: Om.
 Axiom clauses: 2 Conjecture clauses: 1
 Inferences: IEmS=1 ISoR=1 IORe=1
 Reductions: RFMRR=1 RBMRR=1 RObv=1 RUnC=1 RTaut=1 RSST=1 RSSi=1 RFSub=1 RBSub=1 RCon=1
 Extras    : Input Saturation, Always Selection, No Splitting, Full Reduction,  Ratio: 5, FuncWei
 Precedence: Muritor > Om > s
 Ordering  : KBO
Processed Problem:

Worked Off Clauses:

Usable Clauses:
1[0:Inp] ||  -> Om(s)*.
2[0:Inp] || Muritor(s)* -> .
3[0:Inp] Om(U) ||  -> Muritor(U)*.
```

```
SPASS V 3.5
SPASS beiseite: Proof found.
Problem: socrate.spass
SPASS derived 1 clauses, backtracked 0 clauses, performed 0 splits and kept 4 clauses.
SPASS allocated 26886 KBytes.
SPASS spent 0:00:00.04 on the problem.
0:00:00.01 for the input.
0:00:00.01 for the FLOTTER CNF translation.
0:00:00.00 for inferences.
0:00:00.00 for the backtracking.
0:00:00.00 for the reduction.

-------------------------SPASS-STOP------------------------------
```

În prima jumătate, sunt afișate clauzele din FNSC (în formă implicațională), precum și o serie de observații în legătură cu specificitatea acestor clauze (de exemplu, SPASS observă că toate clauzele sunt clauze Horn în această problemă). În a doua jumătate, linia cea mai importantă este:

```
SPASS beiseite: Proof found.
```

"Proof found" înseamnă că s-a găsit o demonstrație pentru clauza vidă pornind de la FNSC-ul negației formulei de început și deci formula de la care am plecat este validă.

Putem modifica problema astfel încât să verificăm dacă $P(s), \forall x.(P(x) \rightarrow Q(x)) \models \forall x.Q(x)$:

```
begin_problem(Socrate).

list_of_descriptions.
name({*socrate.spass*}).
author({*Stefan Ciobaca*}).
status(unknown).
description({*Problema cu Socrate - muritor*}).
end_of_list.

list_of_symbols.
functions[(s, 0)].
predicates[(Om, 1), (Muritor, 1)].
end_of_list.

list_of_formulae(axioms).
formula(Om(s)).
formula(forall([x], implies(Om(x), Muritor(x)))).
end_of_list.

list_of_formulae(conjectures).
```

```
formula(forall([x], Muritor(x))).
end_of_list.

end_problem.
```

De această dată, SPASS răspunde:

```
SPASS beiseite: Completion found.
```

Acest lucru înseamnă că, prin rezoluție, s-a generat o mulțime de clauze închisă la rezoluție (nu se mai pot obține clauze noi) care nu conține clauza vidă. Concluzionăm că formula de la care am plecat nu este validă și deci consecința semnatică $P(s), \forall x.(P(x) \rightarrow Q(x)) \models \forall x.Q(x)$ nu este adevărată.

1.7.3 Dreadbury Mansion Mystery

Următorul exemplu este un puzzle arhicunoscut în cursurile de demonstrare automată:

> Someone who lives in Dreadbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein. A killer always hates his victim, and is never richer than his victim. Charles hates no one that Aunt Agatha hates. Agatha hates everyone except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone Aunt Agatha hates. No one hates everyone. Agatha is not the butler.

Autorul puzzle-ului este Len Schubert. Puzzle-ul de mai sus face parte dintr-o colecție de probleme pentru demonstratoare automate descrisă de Francis Jeffry Pelletier în *Seventy-Five Problems for Testing Automatic Theorem Provers* (Journal of Automated Reasoning, 1986).

Modelați puzzle-ul de mai sus sub forma unei probleme pentru SPASS și stabiliți cine este criminalul.

Soluție:

```
begin_problem(PuzzleAgatha).

list_of_descriptions.
name({*agatha.spass*}).
author({*Stefan Ciobaca*}).
status(unknown).
description({*Problema cu Dreadbury Mansion*}).
end_of_list.

list_of_symbols.
functions[(agatha, 0), (butler, 0), (charles, 0)].
predicates[(lives, 1), (killed, 2), (richer, 2), (hates, 2)].
```

```
end_of_list.

list_of_formulae(axioms).
% Someone who lives in Dreadbury Mansion killed Aunt Agatha.

  formula(exists([X], and(lives(X),killed(X, agatha)))).

% Agatha, the butler, and Charles live in Dreadbury Mansion,

  formula(and(lives(agatha), lives(butler), lives(charles))).

% and are the only people who live therein.
  formula(forall([X], implies(lives(X),
                              or(equal(X, agatha), equal(X, butler), equal(X, charles)

% A killer always hates his victim,
  formula(forall([X,Y], implies(killed(X, Y), hates(X, Y)))).

% and is never richer than his victim.
  formula(forall([X,Y], implies(killed(X, Y), not(richer(X, Y))))).

% Charles hates no one that Aunt Agatha hates.
  formula(forall([X], implies(hates(agatha, X), not(hates(charles, X))))).

% Agatha hates everyone except the butler.
  formula(forall([X], implies(not(equal(X, butler)), hates(agatha, X)))).

% The butler hates everyone not richer than Aunt Agatha.
  formula(forall([X], implies(not(richer(X, agatha)), hates(butler, X)))).

% The butler hates everyone Aunt Agatha hates.
  formula(forall([X], implies(hates(agatha, X), hates(butler, X)))).

% No one hates everyone.
  formula(forall([X], exists([Y], not(hates(X, Y))))).

% Agatha is not the butler.
  formula(not(equal(agatha, butler))).

end_of_list.

list_of_formulae(conjectures).
  formula(killed(XXXXX, agatha)).
end_of_list.

end_problem.
```

Înlocuiți `XXXXX` cu fiecare dintre cele trei personaje și stabiliți cine este criminalul.

1.7.4 Puzzle-ul lui Einstein

Următorul puzzle (cu autor necunoscut) îi este atribuit lui Einstein (foarte probabil, în mod eronat) și este cunoscut sub numele de "Einstein's riddle" / "Einstein's puzzle".

> There are five houses in five different colors in a row. In each house lives a person with a different nationality. The five owners drink a certain type of beverage, smoke a certain brand of cigar and keep a certain pet. No owners have the same pet, smoke the same brand of cigar, or drink the same beverage. Other facts:
>
> 1. The Brit lives in the red house.
> 2. The Swede keeps dogs as pets.
> 3. The Dane drinks tea.
> 4. The green house is on the immediate left of the white house.
> 5. The green house's owner drinks coffee.
> 6. The owner who smokes Pall Mall rears birds.
> 7. The owner of the yellow house smokes Dunhill.
> 8. The owner living in the center house drinks milk.
> 9. The Norwegian lives in the first house.
> 10. The owner who smokes Blends lives next to the one who keeps cats.
> 11. The owner who keeps the horse lives next to the one who smokes Dunhill.
> 12. The owner who smokes Bluemasters drinks beer.
> 13. The German smokes Prince.
> 14. The Norwegian lives next to the blue house.
> 15. The owner who smokes Blends lives next to the one who drinks water.
>
> The question is: who owns the fish?

Exercițiul 1.7.1. *Modelați problema de mai sus în SPASS (și găsiți soluția).*