Formulă **LP1**:

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

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

(**R** este *densă*; fără „$\in$”; lumea „reală”)

CE se scrie cu: *negru*, *albastru*, *roșu* (vedem în continuare); pt. *distincții mai ușoare* chiar *în cadrul limbajului obiect*

Definiție. O ***structură (matematică)*** este un triplet
S = $\langle$**D**, *Pred*, *Fun*$\rangle$, unde:

- **D** este o mulțime nevidă numită ***domeniu***
- Fiecare P $\in$ *Pred* este un ***predicat*** de o *aritate* oarecare (adică *numărul* de *argumente*, ca *funcție*; sau *numărul* de *elemente* aflate în *relație*) peste mulțimea **D** (***relație***; se scrie și ca *funcție caracteristică*)
- Fiecare f $\in$ *Fun* este o ***funcție*** de o *aritate* oarecare

  peste mulțimea **D**
- Elementele prezente în *Pred*, *Fun*, **D** (ca de altfel și numele generice care se refră la astfel de elemente), se scriu doar cu *negru*
- Într-adevăr rămâne regula standard: elementele din **limbajul obiect/ de discurs** (= formulele **LP1**) sunt scrise „cu culori” iar cele din **metalimbaj** (simboluri, cuvinte, fraze în limbaj natural, reefriri la obiecte oarecare, chiar la cele care fac parte din limbajul obiect) se scriu cu negru

Exemple de structuri (matematice)

1. $\langle \mathbf{N}, \{<, =\}, \{+, 0, 1\}\rangle$
2. $\langle \mathbf{R}, \{<, =\}, \{+, -, 0, 1\}\rangle$ (aceleași nume, dar …)
3. $\langle \mathbf{Z}, \{<, =\}, \{+, -, 0, 1\}\rangle$
4. $\langle \mathbf{B}, \emptyset, \{\bullet, +, \overline{\ \ }\}\rangle$ (fără predicate: **structuri algebrice**)
5. $\langle \mathbf{R}, \{<\}, \emptyset\rangle$ (fără funcții: **structuri relaționale**; în plus, dacă domeniul acestora este finit: **baze de date relaționale**)

- Intuitiv: adevărul unei formule într-o structură (vezi formula φ din primul slide); *overloading* ...

Definiție. O ***signatură*** Σ este un tuplu $\Sigma = \langle \mathcal{P}, \mathcal{F} \rangle$ unde $\mathcal{P}$ este o mulțime de ***simboluri predicative*** și $\mathcal{F}$ este o mulțime de ***simboluri funcționale*** (*nume generice* de *relații între obiecte*, respectiv de *transformări între obiecte*)

- Fiecare simbol **s** (de predicat sau funcție) are asociat un număr natural pe care îl vom numi ***aritatea simbolului*** și îl vom nota cu *ar*(**s**)
- *ar* : $\mathcal{P} \bigcup \mathcal{F} \to$ **N**
- Unei signaturi fixate i se pot *atașa/asocia* oricâte structuri (matematice) astfel:

Definiție. Dacă $\Sigma = \langle \mathcal{P}, \mathcal{F} \rangle$ este o signatură, o Σ-***structură*** este orice structură (matematică)

S = $\langle$**D**, *Pred*, *Fun*$\rangle$ astfel încât:

- Pentru fiecare simbol predicativ P $\in \mathcal{P}$ există un (*unic*) predicat (notat uzual cu $P^{\mathbf{S}}$) $\in$ *Pred,* de aceeași aritate (și reciproc, de fapt)
- Pentru fiecare simbol funcțional *f* $\in \mathcal{F}$ există o (*unică*) funcție (similar, notată cu $f^{\mathbf{S}}$) $\in$ *Fun*, de aceeași aritate (și reciproc)
- **Observație.** Simbolurile (**concrete** !) care fac parte din $\mathcal{P}$ se scriu cu *albastru*, iar cele care fac parte din $\mathcal{F}$ se scriu cu *roșu*; mai sus, P și *f* sunt scrise cu *negru* deoarece sunt „nume generice” pt. elementele acelor mulțimi (fiind astfel simboluri oarecare *din metalimbaj*)

<u>**Exemplu** (nr.4, pag.9</u>, în cursul scris)

- Fie $\Sigma = \langle\{P, Q\}, \{f, i, a, b\}\rangle$, cu $ar(P) = ar(Q) = 2$; $ar(f) = 2$, $ar(i) = 1$, $ar(a) = ar(b) = 0$; aici P, ..., *f*, etc., sunt elemente concrete din ...; deci ...
- <u>$\langle \mathbf{R}, \{<, =\}, \{+, -, 0, 1\}\rangle$ și $\langle \mathbf{Z}, \{<, =\}, \{+, -, 0, 1\}\rangle$ sunt Σ-structuri</u>
- Dată o Σ-structură, mulțimea simbolurilor predicative de aritate *n*, adică $\{P \mid ar(P) = n\}$, se notează cu $\mathcal{P}_n$ ($\mathcal{P}_0$, observație – despre disjuncție, viditate, finitudine …)
- Similar, pentru cazul simbolurilor funcționale, vom folosi $\mathcal{F}_n$ ($\mathcal{F}_0$ …)

4.Sintaxa calculului/logicii cu predicate de ordinul I (LP1)

- Similar cu **LP**, vom *defini inductiv* mulțimea formulelor „de ordinul I” (**LP1**), ca fiind (similar cu cazul **LP**) o mulțime de cuvinte peste un anumit alfabet
- Vom avea nevoie însă de câteva *definiții succesive*, unele imbricate în altele
- **Alfabetul** va fi constituit (tot) din reuniunea câtorva submulțimi distincte, coerente, de simboluri/caractere; mai exact:

8 - Alfabetul

1. ***Conectori logici*** - {$\neg$, $\wedge$,$\vee$, $\rightarrow$, $\leftrightarrow$} (poate - $\perp$)

2. ***Cuantificatori*** - {$\forall$, $\exists$} (de fapt, nu sunt 2 ...)

3. ***Variabile*** – $\mathcal{X}$ = {x, y, z, x', y'', z_1, …}; această mulțime **nu** are același rol cu mulțimea variabilelor propoziționale (ci unul similar cu cel al unei mulțimi de variabile din matematică)

4. ***Simboluri auxiliare/separatori*** –

{(,), ., ,, (,), ,}

5. ***Simboluri suplimentare*** (<u>specifice signaturilor</u>) – mulțimea simbolurilor predicative ($\mathcal{P}$) *și* (adică … $\cup$) cea a simbolurilor funcționale ($\mathcal{F}$) dintr-o *signatură fixată* $\sum = \langle\mathcal{P}, \mathcal{F}\rangle$ (**LP1**$_\Sigma$; similar cu **LP**$_{\neg, \wedge, \vee}$)

9 - Termeni

- Mulțimea termenilor, $\mathcal{T}$, este cea mai mică mulțime care satisface următoarele proprietăți (termenii *concreți* se vor scrie în întregime cu *roșu*):

1. $\mathcal{F}_0 \subseteq \mathcal{T}$, adică orice simbol constant este termen (*caz de bază*).

2. $\mathcal{X} \subseteq \mathcal{T}$, adică orice variabilă este termen (tot *caz de bază*).

3. Dacă $f \in \mathcal{F}_n$ (n > 0) și $t_1, \ldots, t_n \in \mathcal{T}$, atunci $f(t_1, \ldots, t_n) \in \mathcal{T}$, adică un simbol funcțional de aritate *n* „aplicat” (de fapt, textual, citim „urmat de”) unui număr de exact *n* („alți”) termeni (separați prin *simbolul* virgulă), este tot termen (un singur *caz inductiv* – 3., adică construcția termenilor noi din termeni vechi).

- Notație uzuală pentru (nume generice de) termeni: t, s, t_1, t’, … (sau *t*, …); cei concreți vor face evident parte din *limbajul obiect*

- Putem vorbi și despre ***arborele*** (*abstract*, *sintactic* ..., *de sintaxă abstractă*; *atașat formulei*, care *descrie formula* ...) prin care se poate reprezenta/identifica (și) un termen *t*, adică *arb*(*t*) (definiția formală este similară cu cea cunoscută de la **LP -** vezi cursul)

Exemplu (nr.6, pag.12, în cursul scris).

- Să scriem câțiva termeni pentru signatura

$$\Sigma = \langle\{P, Q\}, \{f, i, a, b\}\rangle$$

- Construiți arborii corespunzători

11 – Formule atomice

- O ***formulă atomică*** este orice cuvânt/șir de caractere de forma $P(t_1, \ldots, t_n)$, unde $P \in \mathcal{P}_n$ este un simbol predicativ oarecare (de aritate n), iar $t_1, \ldots, t_n \in \mathcal{T}$ sunt termeni
- Mai sus am inclus și cazul $n = 0$ (elementele din $\mathcal{P}_0$); în loc de $P()$ vom scrie P

<u>**Exemplu**</u> (<u>pag.12-13</u>, în curs))

Putem și continua exemplul de pe slide-ul anterior (formule atomice, inclusiv furnizând arborii sintactici/abstracți atașați); de ce unele cuvinte <u>nu sunt</u> termeni peste o anumită signatură ?

12 – Mulțimea **LP1**

- Formulele vor fi notate, în general, prin litere din alfabetul grecesc (cu sau fără indici …): φ, ψ,…
- Simultan putem defini și *arb*(φ) (vezi **LP**)
- Formulele atomice și formulele se vor scrie fie doar cu *albastru* (caz particular), fie cu *albastru* și *roșu*; ele fac parte desigur din limbajul obiect

Definiție inductivă LP1.

1. Orice formulă atomică este formulă (adică fiecare $P(t_1, \ldots, t_n) \in$ **LP1**, unde …; $n \geq 0$) (este *cazul de bază*)

13 – Mulțimea **LP1** (continuare)

2. Urmează, evident, *cazurile inductive* (în număr de 7, deoarece am introdus explicit și conectorii $\rightarrow$ și $\leftrightarrow$); ele sunt valabile pentru orice formule $\varphi, \varphi_1, \varphi_2 \in$ **LP1** și pentru orice variabilă $x \in \mathcal{X}$ (exemple – pag.14):

(a) Dacă $\varphi \in$ **LP1**, atunci $\rceil \varphi \in$ **LP1**.

(b) Dacă $\varphi_1, \varphi_2 \in$ **LP1**, atunci $(\varphi_1 \wedge \varphi_2) \in$ **LP1**.

(c) Dacă $\varphi_1, \varphi_2 \in$ **LP1**, atunci $(\varphi_1 \vee \varphi_2) \in$ **LP1**.

(d) Dacă $\varphi_1, \varphi_2 \in$ **LP1**, atunci $(\varphi_1 \rightarrow \varphi_2) \in$ **LP1**.

(e) Dacă $\varphi_1, \varphi_2 \in$ **LP1**, atunci $(\varphi_1 \leftrightarrow \varphi_2) \in$ **LP1**.

(f) Dacă $\varphi \in$ **LP1**, atunci $(\forall x.\varphi) \in$ **LP1**.

(g) Dacă $\varphi \in$ **LP1**, atunci $(\exists x.\varphi) \in$ **LP1**.

14 – Modelare, 1

- Să ne ocupăm puțin de *modelare*, adică de *reprezentarea realității* („reală”, virtuală, etc.) *prin formule* **LP1**
- Reluând ceea ce am început pe **slide 4** (puțin...), și simplist vorbind, vom considera că *realitatea* este alcătuită dintr-o mulțime de *obiecte diverse*, care pot fi transformate (*doar* într-un alt obiect) prin executarea unor *activități/acțiuni* și între care pot exista anumite *relații/legături* (legături care se exprimă *doar* prin DA = există o anume legătură , sau NU = nu există acea legătură)
- *Această realitate* trebuie *descrisă și tratată* (pentru rezolvarea corectă a problemelor și apelând la un computer) ***formal*** – de exemplu folosind LOGICA (în particuler, **LP** și **LP1**)

15 – Modelare, 2

- Sintactic vorbind, într-o formulă φ obiectele vor fi identificate prin *constante* (simboluri funcționale de aritate 0) sau prin *variabile*; acțiunile prin simboluri *funcționale* (de aritate mai mare ca 0); iar legăturile prin *simboluri predicative*
- Ca de obicei (vezi cazul **LP**), vom lucra cu *formule atomice/simple* pentru a reprezenta textele (care pot fi considerate ca) *indivizibile*; apoi, și cu *formule compuse*
- Formulele compuse se vor forma din cele simple *doar* cu ajutorul *conectorilor logici* (cei cunoscuți deja de la **LP**) și a *cuantificatorilor*

16 – Modelare, 3

- Ca un prim exemplu, să considerăm următorul *text/frază/afirmație compusă* (în limba română)
- *Ion este student. Orice student învață la Logică. Oricine învață la Logică trece examenul. Orice student este om. Există un om care nu a trecut examenul.*
- Să construim o formulă $\varphi \in$ **LP1** a.î. aceasta să *modeleze corespunzător* fraza anterioară
- Intuitiv (nici nu vom putea da o definiție formală; puteți <u>explica</u> DE CE nu ?) acest lucru înseamnă ca *adevărul asociat* (lingvistic, verbal) cu fraza să fie „**aceiași**” cu cel definit formal, prin ***semantica*** **LP1** (revenim la momentul oportun: **S**, $\alpha \vDash \varphi$)

17 – Modelare, 4

- **Obiectele** reale care apar aici, pot fi împărțite (formal: ne-explicit) în (sub)mulțimi: *oameni*, *studenți*, *materii* (în particular – Logica), *examene*
- Ce mai știm, suplimentar, din parcurgerea „la prima vedere” a textului (unele cunoștințe nici nu sunt exprimate explicit – cele subliniate în continuare; ele provin din experiența de viață): *Ion este unul dintre studenți* (<u>Ion este nume propriu pentru oameni</u>); <u>Logica este una dintre materii</u>; **nu** se întrevăd **transformări** (de tipul sugerat) între obiecte; totuși, avem *Ion*, *Logica* $\in \mathcal{F}_0$
- În schimb, **sunt** precizate anumite **legături** între obiecte (de tipul sugerat); unele (necesare a fi) descrise explicit, altele (poate) omise

18 – Modelare, 5

- „Cheia” succesului constă din precizarea legăturilor *cât mai aproape de situația reală* care trebuie studiată; *atenție* la *aritățile* lor – *pot fi determinante*
- **Predicatele** necesare ar fi deci: *Student*(x) – (adevărat doar dacă) x este student; *Om*(x) – (adevărat doar dacă) x este om; *Învață_la*(x, y) - (adevărat doar dacă) x (om, student) învață la (materia) y (din context, deducem că o asemenea relație este absolut necesară; mai mult, ea este *între* 2 *entități*, în mod evident); avem nevoie cu sigeranță și de exprimarea printr-un simbol predicativ a unei legături dintre un student, o materie și rezultatul „examenului” la acea materie;

19 – Modelare, 6

- Continuând, această relație ar putea avea aritate 3, sau chiar 2 (dacă se presupune implicit că se „dă examen” = *o anumită formă de evaluare*, la fiecare materie); alegem această ultimă propunere, deoarece detaliile sunt importante, dar și ... simplificarea „calculelor” (dacă acest lucru nu este clar dăunător); introducem astfel: *Trece_ex*(x, y) - (adevărat doar dacă) x (om, student) „trece” la (evaluarea la materia) y
- ***Concluzionând***: construim mai întâi formulele atomice din care este alcătuit textul inițial

20 – Modelare, 7

- Acestea sunt ușor de identificat: φ_1 = *Student*(*Ion*); φ_2 = *Student*(x); φ_3 = *Învață_la*(x, *Logica*); φ_4 = *Trece_ex*(x, *Logica*); φ_5 = *Om*(x)
- Următorul pas înseamnă introducerea conectorilor logici și a cuantificatorilor, pentru reprezentarea formală a unor subformule (compuse), într-o primă fază; din fericire – în cazul de față – aceste „prime” subformule sunt separate prin „punct”

φ_6 = *Student*(*Ion*) (= φ_1)

φ_7 = ($\forall$x.(*Student*(x) $\rightarrow$ *Învață_la*(x, *Logica*))) =

= ($\forall$x.($\varphi_2 \rightarrow \varphi_3$))

φ_8 = ($\forall$x.(*Învață_la*(x, *Logica*) $\rightarrow$

$\rightarrow$ *Trece_ex*(x, *Logica*))) = ($\forall$x.($\varphi_3 \rightarrow \varphi_4$))

21 – Modelare, 8

$\varphi_9 = (\forall x.(Student(x) \rightarrow Om(x)))$

$= (\forall x.(\varphi_2 \rightarrow \varphi_5))$

$\varphi_{10} = (\exists x.(Om(x) \wedge \neg Trece\_ex(x, Logica))) =$

$= (\exists x.(\varphi_5 \wedge \neg \varphi_4))$

- În final, găsim formula cerută, φ, dacă „luăm" (desigur) conjuncția acestor ultime subformule

$\varphi = \varphi_6 \wedge \varphi_7 \wedge \varphi_8 \wedge \varphi_9 \wedge \varphi_{10}$

- Puteți încerca singuri să „traduceți" din limbaj natural în **LP1**, următoarele texte

Orice număr natural este și număr întreg.

Suma oricăror două numere naturale este număr natural.

Oricum am alege un număr natural, există un număr prim care este mai mare decât numărul respectiv.

22 – Modelare, 9

Dacă orice număr natural este număr prim, atunci zero este număr prim.

- Ca observație ajutătoare: **obiectele** care interesează aici vor fi doar *numere*; subclasele „interesante” vor fi „precizate” prin **relații unare** (*Natural*, *Întreg*, *Prim*, *Par*, etc.); trebuie folosită și **relația binară** „*mai mare*” (poate și „*egal*” ... ; alegeți câte o notație corespunzătoare); sunt – de asemenea – necesare și (măcar ...) simbolurile funcționale 0 ($\in \mathcal{F}_0$) și (să zicem) $+ \in \mathcal{F}_2$ ...

- Continuăm discuția generală dedicată sintaxei cu detalii despre **variabile** și legăturile lor cu formulele **LP1**
- Funcția *vars* : **LP1** $\rightarrow 2^{\mathcal{X}}$
- *vars*(φ), $\varphi \in$ **LP1**, este <u>mulțimea</u> ***variabilelor care apar*** (textual, ca simbol = cuvânt de lungime 1) ***în formula*** φ (măcar o dată)
- Definiția este una inductivă, imbricată (urmând definiția inductivă a lui **LP1**): întâi pt. *termeni*, apoi pt. *formule atomice* și, în final pt. *formule*; <u>păstrăm același nume</u> pt. „*extensiile*” succesive ale funcției *vars* (deși ...)

24 – Funcția *vars*, partea1

Definiție. Pt. mulțimea termenilor ($\mathcal{T}$) avem:

1. *vars*(c) = Ø, pt. fiecare c$\in \mathcal{F}_0$ (*caz de bază*)

2. *vars*(x) = {x}, pt. fiecare x $\in \mathcal{X}$ (*caz de bază*)

3. *vars*(f(t_1, t_2, …, t_n)) = $\bigcup_{i \in [n]}$*vars*(t_i), pt. fiecare număr natural $n > 0$, fiecare $f \in \mathcal{F}_n$ și fiecare $t_1, t_2, \ldots, t_n \in \mathcal{T}$ (*caz inductiv*)

25 – Funcția *vars*, partea 2

Definiție. Pt. formulele atomice (primul caz este *cazul de bază*; apoi sunt *cazurile inductive*):

1. $vars(P(t_1, t_2, \ldots, t_n)) = \bigcup_{i \in [n]} vars(t_i)$

2. $vars(\neg \varphi) = vars(\varphi)$

3. $vars((\varphi_1 \wedge \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$

4. $vars((\varphi_1 \vee \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$

5. $vars((\varphi_1 \rightarrow \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$

6. $vars((\varphi_1 \leftrightarrow \varphi_2)) = vars(\varphi_1) \cup vars(\varphi_2)$

7. $vars((\exists x.\varphi)) = vars(\varphi) \cup \{x\}$

8. $vars((\forall x.\varphi)) = vars(\varphi) \cup \{x\}$

- În cele de mai sus, $t_1, t_2, \ldots, t_n \in \mathcal{T}$ sunt termeni oarecare; în prima definiție, cazul inductiv este valabil pentru $n \geq 1$; în **a doua**, dacă $n = 0$ (în cazul de bază), atunci P va *denota* un element din $\mathcal{P}_0$, caz în care *vars*(P) = Ø; de asemenea, $\varphi, \varphi_1, \varphi_2 \in$ **LP1** sunt formule oarecare

Exemplu. Să calculăm *vars*(φ) pentru formula φ care urmează:

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

27 – Ap.libere/legate ale var, pct.1

Definiție. O(rice) ***apariție liberă*** a unei ***variabile*** x ($\in \mathcal{X}$) într-o formulă φ, este dată de (eticheta unui) un nod din arborele abstract care o descrie și care are proprietatea: „mergând” din acel nod înspre rădăcină („în sus”, cf. reprezentării „uzuale” ...), nu întâlnim niciun alt nod care să fie etichetat cu $\forall$x/$\exists$x.

Definiție. O(rice) ***apariție legată*** a unei ***variabile*** x ($\in \mathcal{X}$) într-o formulă φ, este (dată de) un nod (eticheta) din arborele abstract care descrie φ și care are proprietatea: „mergând” din acel nod înspre rădăcină („inapoi”, pe arcele corespunzătoare, <u>pe unicul drum</u> care există), întâlnim măcar un alt nod care este etichetat cu $\forall$x sau cu $\exists$x.

28 – Ap.libere/legate ale var, pct.2

- Să punctăm că, în ultimul caz, *cuantificatorul* (unic!) *care va lega apariția* în cauză a unui $x \in \mathcal{X}$, este (eticheta din) <u>primul</u> nod întâlnit, adică *primul întâlnit* „în mersul în sus” (spre rădăcină)

<u>**Exemplu**</u> (<u>nr. 23, pag.22</u>, în curs)

Putem continua și exemplul început cu ultima formulă φ (slide 17): întâi <u>construim</u> *arborele abstract* asociat; <u>găsim</u> toate *aparițiile libere ale tuturor variabilelor* și toate *aparițiile legate ale tuturor variabilelor*, folosind arborele; „pe text” <u>le „marcăm”</u> diferit.

- <u>***Atenție***</u>: va exista o ***diferență precisă*** între *aparițiile libere/legate ale variabilelor* (definiție dată deja) și *<u>mulțimea</u> variabilelor libere/<u>mulțimea</u> variabilelor legate* (urmează)

29 – Var. libere/legate, partea 1

- ***Mulțimea variabilelor libere*** dintr-o formulă φ va fi „din" codomeniul funcției *free* : **LP1** $\rightarrow 2^{\mathcal{X}}$, definită inductiv în cele ce urmează; mai precis, *free*(φ)
- Intuitiv, x$\in$*free*(φ), dacă x are măcar o apariție liberă în φ; iar x$\notin$*free*(φ) dacă apare doar ca numele unui cuantificator (chiar în $\forall$x/$\exists$x)
- ***Mulțimea variabilelor legate*** dintr-o formulă φ va fi „din" codomeniul funcției *bound* : **LP1** $\rightarrow 2^{\mathcal{X}}$, definită inductiv în cele ce urmează; mai precis, *bound*(φ)
- Intuitiv, x$\in$*bound*(φ), dacă ea are măcar o apariție legată în φ; se „vede" că, pentru aceasta, este suficient ca în arbore să existe măcar un nod etichetat cu $\forall$x/$\exists$x; variabila x, cea care dă numele cuantorului, este „pusă" (cumva forțat) în *bound*(φ), chiar dacă nu mai are și alte apariții legate

30 - Var. libere/legate, partea 2

Definiție. Funcția *free* : **LP1** $\rightarrow 2^{\mathcal{X}}$ este dată prin:

1. *free*$(P(t_1, t_2, \ldots, t_n)) =$

   $\bigcup_{i \in [n]}$ *vars*(t_i) (*caz de bază*, formule atomice)

2. *free*$(\neg\varphi) =$ *free*(φ) (*caz inductiv*; la fel ca 3.-8.)

3. *free*$((\varphi_1 \wedge \varphi_2)) =$ *free*$(\varphi_1) \cup$ *free*(φ_2)

4. *free*$((\varphi_1 \vee \varphi_2)) =$ *free*$(\varphi_1) \cup$ *free*(φ_2)

5. *free*$((\varphi_1 \rightarrow \varphi_2)) =$ *free*$(\varphi_1) \cup$ *free*(φ_2)

6. *free*$((\varphi_1 \leftrightarrow \varphi_2)) =$ *free*$(\varphi_1) \cup$ *free*(φ_2)

7. *free*$((\exists x.\varphi)) =$ *free*$(\varphi) \setminus \{x\}$

8. *free*$((\forall x.\varphi)) =$ *free*$(\varphi) \setminus \{x\}$

31 - Var. libere/legate, partea 3

Definiție. Similar, funcția *bound* : **LP1** $\rightarrow 2^{\mathcal{X}}$ este:

1. *bound*(P(t$_1$, t$_2$, …, t$_n$)) = Ø (*caz de bază*, pt formule atomice)

2. *bound*($\rceil$ φ)=*bound*(φ) (*caz inductiv*, la fel 3.-8.)

3. *bound*((φ$_1$ $\wedge$ φ$_2$)) = *bound*(φ$_1$) $\cup$ *bound*(φ$_2$)

4. *bound*((φ$_1$ $\vee$ φ$_2$)) = *bound*(φ$_1$) $\cup$ *bound*(φ$_2$)

5. *bound*((φ$_1$ $\rightarrow$ φ$_2$)) = *bound*(φ$_1$) $\cup$ *bound*(φ$_2$)

6. *bound*((φ$_1$ $\leftrightarrow$ φ$_2$)) = *bound*(φ$_1$) $\cup$ *bound*(φ$_2$)

7. *bound*(($\exists$x.φ)) = *bound*(φ) $\cup$ {x}

8. *bound*(($\forall$x.φ)) = *bound*(φ) $\cup$ {x}

32 - Var. libere/legate, partea 4

- **<u>Observație</u>**: am precizat că o variabilă oarecare $x \in \mathcal{X}$, va aparține lui *bound*(φ) deși ea poate apare *doar* ca nume al unui cuantor; totuși, în acest caz, *apariția* sa nu va fi considerată nici ca apariție liberă, nici ca apariție legată; <u>faceți singuri diferența dintre</u> o *apariție* liberă/legată a unei variabile $x \in \mathcal{X}$, într-o formulă φ și prezența sa într-una dintre mulțimile *free*(φ)/*bound*(φ); în plus, să notăm că *free*(φ) și *bound*(φ) pot avea elemente în comun (vezi formulele φ din exemple); cazurile „limită" (*n* = 0 în definiții) „dau" ca rezultat pe Ø

<u>Exemplu</u>. Să găsim *free*(φ) și *bound*(φ), unde φ este dată în <u>slide 26</u>; în curs - <u>nr.29, pag.25</u>).

- Precizări privind priorități, paranteze în plus/minus, domeniu de vizibilitate, etc. ...
- Conectorul $\rceil$ este „cel mai prioritar” (prioritate 0); apoi: $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall x.$, $\exists x.$; $\perp$; vezi și §3.5, pag.16 din curs
- Vom menține și modalitatea de a grupa un șir de operatori de același tip (cei binari: „argumentele - câte 2, la stânga”; „la dreapta” – pt. cei unari)
- Pentru înțegerea exactă a noțiunii de ***domeniu de vizibilitate*** (sau *domeniu sintactic*) ***al unui*** (nume de) ***cuantificator***, citiți singuri: mai întâi despre similaritatea cu limbajele de programare (din cursul scris: *program* **C** cu 3 *for*-uri *imbricate*; *variabile locale și globale în programare*); apoi, §3.12, pag.25, în curs

- În logică (similar cu existența *variabilelor locale* în, de ex., limbajul **C**) *ne interesează* ca atunci când într-o formulă φ apare un cuantificator, să zicem ∀x., să știm *dacă* o *apariție ulterioară*, în textul care formează φ, a lui x (ca variabilă, nu doar ca nume de cuantor), <u>este</u> *legată/binded* <u>*de acest* ∀x.</u>, <u>sau *nu*</u> (în acest ultim caz, apariția respectivă se va numi *liberă* (*free*), și va corespunde folosirii unei variabile globale în **C**)
- Pe scurt, ideea este că domeniul pomenit, în lipsa unor paranteze explicite (similar: *begin – end*), să se „întindă” <u>***cât mai la dreapta posibil***</u>

- Discuție: (∀x. ...); ...(...∀x. ...); (∀x. ...(...) ...)
- Dacă știm că <u>imediat înainte</u> de fiecare ∀x/∃x trebuie pusă o panteză (, re-punem toate parantezele celelalte (conform priorităților) și apoi punem și paranteza) (*corespondentă* cu (), <u>cât mai la dreapta posibil</u> (așa după cum s-a afirmat)
- Teoretic, ar fi (poate) mai bine să admitem (pentru siguranță) și posibilitatea adăugării de paranteze „suplimentare”
- Iar pentru ca *totul* să fie definit formal (și mai ușor de înțeles), ar trebui să apelăm - mai curând - la arborii atașați (însă și pentru ei ar fi necesară o definiție formală ...)
- <u>Atenție însă la difențele posibile dintre *ordinea* în care sunt repuse parantezele</u> conform regulilor și ... cum s-au șters la început ...

- Ținând cont de cele spuse despre priorități și domeniile de vizibilitate, formula φ de mai jos (din care s-au eliminat *toate* parantezele) se reparantetizează corect sub forma φ':

$\varphi = \forall x.P(x, x) \vee \rceil \exists y.P(x, y) \wedge P(x, x)$.

$\varphi' = (\forall x.(P(x, x) \vee (\rceil (\exists y.(P(x, y) \wedge P(x, x))))))$.

- Vezi și (peste tot, de altfel) **Fișa de exerciții** (în acest caz, la pag.26-27 în curs)

Final Seminar 1

- Ce facem în **Cursul 2/LP1** (săptămâna a 10-a de școală, incluzând lucrarea de evaluare)
- **Recapitulare Curs 1**: *signaturi* (∑); ∑*-structuri* (**S**); sintaxa **LP1** (*alfabet*, *termeni*, *formule atomice*, *formule*); reprezentarea formulelor ca *arbori* (*abstracți, de sintaxă ...*); *prioritățile* „operatorilor” (conectori logici și cuantificatori); *eliminarea* și *introducerea parantezelor*; *domenii de vizibilitate* ale cuantorilor; *variabile* (*variabilele unei formule* și funcția *vars*; *apariții libere* și *apariții legate* ale unei variabile într-o formulă; *variabile libere*/funcția *free*; *variabile legate*/funcția *bound*; codomeniile acestora

- **Semantica LP1**: (<u>**S**-)*atribuiri*</u> (α) într-o (Σ-) structură **S**; *valoarea* (din domeniul **D** a) *unui termen*, în **S**-atribuirea α, adică <u>*extensia*</u> unei **S**-atribuiri; <u>*actualizarea*</u> unei **S**-atribuiri; <u>*valoarea de adevăr*</u> a unei *formule atomice* și apoi a unei *formule de ordinul* I (element/ formulă din **LP1**); noțiuni semantice suplimentare: <u>*(ne)satisfiabilitate*</u> și <u>*validitate*</u> într-o structură fixată („<u>local</u>"); *(ne)satisfiabilitate* și *validitate*, la modul „<u>global</u>" (nedepinzând de structură); noțiunea de <u>*consecință semantică*</u> (local, global)

- Pentru exemple, utilizăm o perioadă signatura:

$\Sigma = \langle\{P\}, \{f, i, e\}\rangle$, $ar(P) = 2$;

$ar(f) = 2$, $ar(i) = 1$, $ar(e) = 0$

- Și Σ-structurile

-**S1** = $\langle \mathbf{Z}, \{=\}, \{+, -, 0\}\rangle$

-**S2** = $\langle \mathbf{R}^{+}, \{=\}, \{\times, \bullet^{-1}, 1\}\rangle$ (în curs e scris $\mathbf{R}^{*}$)

-**S3** = $\langle \mathbf{N}, \{=\}, \{+, s, 0\}\rangle$

-**S4** = $\langle \mathbf{N}, \{<\}, \{+, s, 0\}\rangle$

-**S5** = $\langle \mathbf{Z}, \{<\}, \{+, -, 0\}\rangle$

<u>3.Semantica</u> LP1 (*trebuie și valori pt. variabile*)

Definiție. Fie $\sum$ o signatură și **S** o ($\sum$-)structură având domeniul **D**. Se numește (**S**-)***atribuire*** orice funcție $\alpha : \mathcal{X} \to$ **D**.

- Se iau în considerare și atribuirile α_1 și α_2, ambele cu **D** = **Z**, definite prin:

1. $\alpha_1(x_1) = 5$, respectiv $\alpha_2(x_1) = 6$.
2. $\alpha_1(x_2) = 5$, respectiv $\alpha_2(x_2) = 5$.
3. $\alpha_1(x_3) = 6$, respectiv $\alpha_2(x_3) = 6$.
4. $\alpha_1(x) = 0$ și $\alpha_2(x) = 0$,

   pentru orice $x \in \mathcal{X} \setminus \{x_1, x_2, x_3\}$.

Definiție. Fie α o **S**-atribuire ca mai sus și *t* un termen peste signatura Σ. Inductiv - valoarea termenului *t* în α, este un element al lui **D**, notat cu $\alpha^{-}(t)$ și dat, practic, de (***unica*** !) ***extensie*** (α^{-}) a lui α la $\mathcal{T}$:

1. $\alpha^{-}(c) = c^{\mathbf{S}}$, pentru fiecare $c \in \mathcal{F}_0$. (*caz de bază*)
2. $\alpha^{-}(x) = \alpha(x)$, pentru fiecare $x \in \mathcal{X}$. (*caz de bază*)
3. $\alpha^{-}(f(t_1, t_2, \ldots, t_n)) = f^{\mathbf{S}}(\alpha^{-}(t_1), \alpha^{-}(t_2), \ldots, \alpha^{-}(t_n))$, pentru fiecare $f \in \mathcal{F}_n$, n > 0 și $t_1, t_2, \ldots, t_n \in \mathcal{T}$. (*cazul inductiv*)

<u>Să calculăm</u> $\alpha_1^{-}(f(i(x_1), e))$.

Definiție. Se dau α, x și u$\in$**D**, oarecare, ca mai sus. Vom nota prin $\alpha[x \mapsto u]$ o nouă atribuire, numită și ***actualizarea lui*** α (*a „valorii lui* x, *prin* u”), dată inductiv astfel ($\alpha[x \mapsto u] : \mathcal{X} \to$ **D**):

1. $\alpha[x \mapsto u](x) = u$.

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

Cu alte cuvinte: $\alpha[x \mapsto u]$ <u>coincide cu</u> α „peste tot”, <u>exceptând valoarea</u> lui x (care este <u>„pusă pe”</u> u).

- Putem acum, în sfârșit, *defini* (tot) *inductiv* și ***valoarea de adevăr a unei formule*** φ, <u>într-o anumită (signatură Σ), (Σ-)structură **S** și (**S**-) atribuire α</u>, fixate (*citim* …; <u>sau</u> este „exclusiv”)

Definiție. Primul *caz* este *cazul de bază* (vizează formulele atomice), restul fiind desigur *cazurile inductive*:

1. **S**, $\alpha \vDash P(t_1, t_2, \ldots, t_n)$, ddacă $P^{\mathbf{S}}(\overline{\alpha}(t_1), \overline{\alpha}(t_2), \ldots, \overline{\alpha}(t_n))$.
2. **S**, $\alpha \vDash \neg\varphi$, ddacă **S**, $\alpha \nvDash \varphi$.
3. **S**, $\alpha \vDash (\varphi_1 \wedge \varphi_2)$, ddacă **S**, $\alpha \vDash \varphi_1$ și **S**, $\alpha \vDash \varphi_2$.
4. **S**, $\alpha \vDash (\varphi_1 \vee \varphi_2)$, ddacă **S**, $\alpha \vDash \varphi_1$ sau **S**, $\alpha \vDash \varphi_2$.
5. **S**, $\alpha \vDash (\varphi_1 \rightarrow \varphi_2)$, ddacă **S**, $\alpha \nvDash \varphi_1$ <u>sau</u> **S**, $\alpha \vDash \varphi_2$.
6. **S**, $\alpha \vDash (\varphi_1 \leftrightarrow \varphi_2)$ ddacă (atât **S**, $\alpha \vDash \varphi_1$, cât și **S**, $\alpha \vDash \varphi_2$) <u>sau</u> (**S**, $\alpha \nvDash \varphi_1$ și **S**, $\alpha \nvDash \varphi_2$).
7. **S**, $\alpha \vDash (\exists x.\varphi)$, ddacă există $u \in$ **D**, a.î. **S**, $\alpha[x \mapsto u] \vDash \varphi$.
8. **S**, $\alpha \vDash (\forall x.\varphi)$, ddacă pt.orice $u \in$ **D** avem **S**, $\alpha[x \mapsto u] \vDash \varphi$.

- Mai sus, am scris $P^S(\overline{\alpha}(t_1),\overline{\alpha}(t_2),\ldots,\overline{\alpha}(t_n))$ în loc de $\langle\overline{\alpha}(t_1),\overline{\alpha}(t_2),\ldots,\overline{\alpha}(t_n)\rangle \in P^S$, adică am scris P^S ca un *predicat/relație* (prescurtat)
- Puteam scrie și $P^S(\overline{\alpha}(t_1),\overline{\alpha}(t_2),\ldots,\overline{\alpha}(t_n)) = 1$ (așa, P^S ar fi privit ca o *funcție caracteristică*)
- Fie signatura (dată deja) $\Sigma = \langle\{P\}, \{f, i, e\}\rangle$, cu aritățile ... și Σ-structura **S1** = $\langle$**Z**, {=}, {+, -, 0}$\rangle$
- Fie și **S**-atribuirea $\alpha_1 : \mathcal{X} \to \mathbf{Z}$, cu $\alpha_1(x_1) = 5$, $\alpha_1(x_2) = 5$, $\alpha_1(x_3) = 6$, $\alpha_1(x) = 0$ și $\alpha_1(x) = 0$ în rest (pentru orice $x \in \mathcal{X} \setminus \{x_1, x_2, x_3\}$). <u>Să calculăm</u> valoarea de adevăr pentru formula:

  $\varphi_7 = \forall x_1.\exists x_3.P(x_1, x_3)$; mai exact, să verificăm dacă **S1**, $\alpha_1 \vDash \varphi_7$

- Recomandarea este să *rezolvați toate exercițiile* rămase nerezolvate (curs + seminar); sau ... măcar câte unul „de același tip”

Alte concepte semantice

- *Satisfiabilitate*, *nesatisfiabilitate*, *validitate, echivalență* (din cursul anterior); *consecință semantică*; (totul, atât local cât și global); legături între noțiuni

Definiție. O formulă $\varphi \in$ **LP1** ***este satisfiabilă într-o*** (Σ-)***structură*** **S**, dacă existā o (**S**-)atribuire α, a.î. **S**, $\alpha \vDash \varphi$. (*concept local*)

Definiție. O formulă $\varphi \in$**LP1** ***este validă într-o*** $\sum$***-structură*** **S**, dacă <u>pentru fiecare</u> **S**-atribuire α, avem **S**, $\alpha \vDash \varphi$. (*local*)

Definiție. O formulă $\varphi \in$**LP1** ***este satisfiabilă***, dacă <u>există</u> o $\sum$-structură **S** <u>și</u> o **S**-atribuire α, a.î. **S**, $\alpha \vDash \varphi$. (*concept global*)

Definiție. O formulă $\varphi \in$**LP1** ***este validă***, dacă <u>pentru orice</u> $\sum$-structură **S** <u>și</u> orice **S**-atribuire α, a.î. **S**, $\alpha \vDash \varphi$. (*global*)

- Mai sunt și alte „combinații” posibile, dar nu sunt „interesante” și deci nu au toate „nume” speciale, la care să ne putem referi

Exemple

Fie $\Sigma = \langle\{P\}, \{f, i, e\}\rangle$, $\mathbf{S1} = \langle \mathbf{Z}, \{=\}, \{+, -, 0\}\rangle$, α_i – urile deja amintite, și formulele

$\varphi_5 = P(x_1, x_3) \vee P(x_1, x_1)$ și $\varphi_6 = \exists x_3.P(x_1, x_3)$.

- Să notăm că există formule satisfiabile, dar care sunt false într-o anumită structură
- De asemenea, este evident că există și formule care să nu fie valide (global), dar care sunt valide într-o anumită structură particulară

Definiție. O formulă $\varphi \in$ **LP1** este ***consecință semantică*** din mulțimea de formule/a formulelor $\varphi_1, \varphi_2, \ldots, \varphi_n$, într-o ($\Sigma$-)***structură*** (fixată) **S**, notat $\varphi_1, \varphi_2, \ldots, \varphi_n \vDash_{\mathbf{S}} \varphi$, dacă <u>pentru orice</u> **S**-atribuire α, a.î. **S**, $\alpha \vDash \varphi_i$ (pt. fiecare $i \in [n]$), <u>avem</u> și **S**, $\alpha \vDash \varphi$. (*concept local*).

Definiție. O formulă $\varphi \in$ **LP1** este ***consecință semantică*** din mulțimea de formule $\varphi_1, \varphi_2, \ldots, \varphi_n$, notat $\varphi_1, \varphi_2, \ldots, \varphi_n \vDash \varphi$, dacă <u>avem</u>

$\varphi_1, \varphi_2, \ldots, \varphi_n \vDash_{\mathbf{S}} \varphi$ <u>pentru orice</u> Σ-structură **S**. (*concept global*).

<u>Exemplu</u>

Să arătăm că $P(x, y) \vDash_{\mathbf{S1}} P(y, x)$ (dar, global, avem $P(x, y) \nvDash P(y, x)$: în **S1,** „luăm” < ...).

Final Seminar 2

- Ce facem în **Cursul 3**, **LP1** (săptămâna a 11-a de școală, incluzând lucrarea de evaluare)

1. Recapitulare Curs 2: *semantica* **LP1** (adevărul unei formule într-o $\sum$–structură **S** și o **S**-atribuire α); concepte semantice suplimentare: *satifiabilitate*, *nesatisfiabilitate*, *validitate, consecință semantică* (precum și *legăturile* care există între acestea): la *nivel* de *structură* (*local*), apoi la *nivel global* (vizând toate structurile)

2. Substituții în LP1: *domeniul* unei substituții; *extensii*; substituții în formulele **LP1**; notații suplimentare (oricum, peste tot, $\sum$ este fixat)

3. Deducția naturală pentru LP1: *secvențe*; (*scheme* de) *reguli de inferență* (*ipoteze*, *concluzie*, *nume*, *condiție* de aplicabilitate); *sisteme deductive* și *demonstrații formale*; *secvențe valide* într-un sistem deductiv **D** (vezi **LP**)

4. Sistemul deductiv notat **DN** (pt. **LP1** …): *extensia* la **LP1** al **DN** (pentru **LP**); *reguli suplimentare*: ***i***ntroducerea și **e**liminarea cuantificatorului universal, precum și ***i***ntroducerea și **e**liminarea cuantificatorului existențial; *corectitudinea și completitudinea* sistemului deducției naturale pentru **LP1**

<u>2.Substituții</u>

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

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

- Observăm că *dom*(σ) este o mulțime finită
- Vom extindem mai întâi orice substituție σ la $\mathcal{T}$ ($\sigma^{\#}$ - sigma <u>*diez*</u>) și apoi la **LP1** (σ^{b} – sigma <u>*bemol*</u>)
- Prin aplicarea lui $\sigma^{\#}$ unui termen vom obține (tot) un termen, iar prin aplicarea lui σ^{b} unei formule obținem (tot) o formulă

Definiție. Dacă $\sigma : \mathcal{X} \to \mathcal{T}$ este o substituție, atunci ***extensia*** (unică !) a ***lui*** σ ***la mulțimea termenilor*** este funcția $\sigma^{\#} : \mathcal{T} \to \mathcal{T}$, definită inductiv (cf. def. lui $\mathcal{T}$) astfel:

1. $\sigma^{\#}(x) = \sigma(x)$, pentru orice $x \in \mathcal{X}$. (*caz de bază*)

2. $\sigma^{\#}(c) = c$, pentru orice $c \in \mathcal{F}_0$. (*caz de bază*)

3. $\sigma^{\#}(f(t_1, \ldots, t_n)) = f(\sigma^{\#}(t_1), \ldots, \sigma^{\#}(t_n))$, pentru orice $f \in \mathcal{F}_n$, $n \in \mathbf{N}^*$ și orice termeni $t_1, \ldots, t_n \in \mathcal{T}$. (*caz inductiv*)

- Substituțiile se vor nota cu σ, τ, σ_0, τ_1, σ', ρ etc.
- Dacă $t \in \mathcal{T}$ este un termen, atunci prin $\sigma^{\#}(t) \in \mathcal{T}$ vom nota ***termenul obținut din*** *t* prin ***aplicarea substituției*** σ (alternativ: *termenul obținut prin aplicarea substituției* σ *asupra termenului t*)
- Practic, pentru a obține $\sigma^{\#}(t)$ din *t*, *toate aparițiile* variabilei fixate x din *t* sunt înlocuite *simultan* cu termenul corespunzător σ(x)

<u>Exemplu</u> (în curs, <u>89, pag. 40</u>; e făcut și acolo).

Fie substituția $\sigma_1 : \mathcal{X} \to \mathcal{T}$ definită astfel ($dom(\sigma_1) = \{x_1, x_2\}$):

1. $\sigma_1(x_1) = x_2$.
2. $\sigma_1(x_2) = f(x_3, x_4)$.
3. $\sigma_1(x) = x$ pentru orice $x \in X \setminus \{x_1, x_2\}$.

Fie termenul $t = f(f(x_1, x_2), f(x_3, e))$. Să calculăm $(\sigma_1)^\#(t)$.

- Dacă $dom(\sigma) = \{x_1, \ldots, x_k\}$, atunci substituția sigma se mai poate scrie în felul următor:

$$\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_k \mapsto \sigma(x_k)\}$$

- *Atenție*, mai sus nu este vorba de o mulțime, ci, în primul rând, *de o notație* pentru substituții (practic, este o listă)

Definiție. Dacă σ : $\mathcal{X} \to \mathcal{T}$ este o substituție și V $\subseteq \mathcal{X}$ este o submulțime de variabile, atunci ***restricția substituției*** σ ***la mulțimea*** V este o nouă substituție, notată $\sigma|_V : \mathcal{X} \to \mathcal{T}$, și definită astfel:

1. $\sigma|_V(x) = \sigma(x)$ pentru orice $x \in V$.
2. $\sigma|_V(x) = x$ pentru orice $x \in \mathcal{X} \setminus V$.

- În general, este suficient să luăm V $\subseteq$ *dom*(σ)
- Evident, restricționând o substituție la o (sub)mulțime (de variabile) se scot celelalte variabile din domeniul (inițial al) ei

Definiție. Pentru orice substituție $\sigma : \mathcal{X} \to \mathcal{T}$, ***extensia lui*** σ ***la mulțimea formulelor*** este funcția σ^b : **LP1** → **LP1**, definită inductiv (cf. **Def. LP1**) astfel:

1. $\sigma^b(P(t_1, \ldots, t_n)) = P(\sigma^\#(t_1), \ldots, \sigma^\#(t_n))$. (*cazul de bază, pentru formulele atomice*)

- Restul cazurilor (în slide-ul care urmează) vor reprezenta desigur *cazurile inductive* (în număr de **7**, conform definiției inductive a sintaxei **LP1**)

2. $\sigma^b(\neg\varphi) = \neg\sigma^b(\varphi)$.

3. $\sigma^b((\varphi_1 \wedge \varphi_2)) = (\sigma^b(\varphi_1) \wedge \sigma^b(\varphi_2))$.

4. $\sigma^b((\varphi_1 \vee \varphi_2)) = (\sigma^b(\varphi_1) \vee \sigma^b(\varphi_2))$.

5. $\sigma^b((\varphi_1 \rightarrow \varphi_2)) = (\sigma^b(\varphi_1) \rightarrow \sigma^b(\varphi_2))$.

6. $\sigma^b((\varphi_1 \leftrightarrow \varphi_2)) = (\sigma^b(\varphi_1) \leftrightarrow \sigma^b(\varphi_2))$.

7. $\sigma^b((\forall x.\varphi)) = ((\forall x.\rho^b(\varphi)))$, unde $\rho = \sigma|_{dom(\sigma)\setminus\{x\}}$.

8. $\sigma^b((\exists x.\varphi)) = ((\exists.\rho^b(\varphi)))$, unde $\rho = \sigma|_{dom(\sigma)\setminus\{x\}}$.

- Practic, pentru a obține formula $\varphi' = \sigma^b(\varphi)$ din formula φ, adică ***formula*** φ' ***obținută prin aplicarea*** (***extensiei*** σ^b ***a***) ***substituției*** σ ***lui*** φ, procedăm astfel: *fiecare <u>apariție liberă</u> a variabilei* x *în formula* φ *este <u>înlocuită simultan (unde apare)</u>, cu termenul* σ(x)

<u>Exemplu</u> (în curs - <u>94, pag.41</u>; făcut și acolo).

Fie formula:

$\varphi = (\forall x_2.P(x_1, x_2)) \wedge P(x_2, x_2)$.

Să-i aplicăm substituția $\sigma_1 = \{x_1 \mapsto x_2, x_2 \mapsto f(x_3, x_4)\}$, de fapt extensia $(\sigma_1)^b$ a acesteia; găsim astfel formula φ' (desigur, pe parcurs se va folosi și $(\sigma_1)^\#$).

- *Repetăm*: într-o asemenea înlocuire (*atenție*, simultană !) sunt „vizate” *doar aparițiile libere ale variabilelor*
- *Notație alternativă*: o substituție foarte simplă, care vizează doar o variabilă – să zicem că ea este $\sigma = \{x \mapsto t\}$ – se mai notează cu $[t/x]$ sau, (noi am preferat) cu $[x \mapsto t]$ (a **NU se confunda** cu notația similară de la semantică); mai mult, în aceste ultime cazuri, vom adopta *notația post-fixată* pentru a indica aplicarea lui σ unei formule φ, adică $\varphi[x \mapsto t]$, (respectiv $\varphi[t/x]$) în loc de mai complicatul $\{x \mapsto t\}(\varphi)$) (și orice *apariție liberă* a lui x se ...)

3.Deducția naturală LP1

- Se păstrează absolut identic definițiile de la **LP** pentru: *secvență, regulă de inferență, sistem deductiv, demonstrație formală, secvență validă*
- Completări *doar dacă* va fi cazul; pentru exemple se ia $\sum' = \langle\{P, Q\}, \{f, i, a, b\}\rangle$; *ar* sunt 1,1; 2,1, 0, 0

4.Sistemul deductiv pentru **LP1** (notat tot **DN**)

- Toate regulile – inclusiv exemplele și comentariile - de la **LP** se păstrează identic, doar înlocuind $\varphi\in$**LP** cu $\varphi\in$**LP1**
- Să reținem că (și) așa-numitele formule atomice „*de bază*" (*ground = nu conțin variabile*) – $P(a)$, $Q(a)$, de ex. – pot fi considerate ca *formule atomice* din **LP**

- Regulile („corecte”) din „noul” **DN** sunt cele din **_LP + cuantori_** (*scheme*; „f. multe” instanțe ...)

$$(\forall \mathbf{e})\ \frac{\Gamma \vdash (\forall x.\varphi)}{\Gamma \vdash \varphi[x \mapsto t]} \qquad (\exists \mathbf{i})\ \frac{\Gamma \vdash \varphi[x \mapsto t]}{\Gamma \vdash (\exists x.\varphi)}$$

$$(\forall \mathbf{i})\ \frac{\Gamma \vdash \varphi[x \mapsto x_0]}{\Gamma \vdash (\forall x.\varphi)}\quad (c_1), \text{ unde } \underline{(c_1)\ x_0 \notin vars(\Gamma, \varphi)}$$

$$(\exists e)\ \frac{\Gamma \vdash (\exists x.\varphi)\ ;\ \Gamma \cup \{\varphi[x \mapsto x_0]\} \vdash \psi}{\Gamma \vdash \psi}\ (c_2),\ \text{unde}$$

<u>$(c_2)\ x_0 \notin vars(\Gamma, \varphi, \psi)$</u>

- În *schemele* de mai sus, $\Gamma \subseteq$ **LP1**, φ, $\psi \in$ **LP1**, x, $x_0 \in \mathcal{X}$, $t \in \mathcal{T}$ sunt oarecare
- Nu uităm nici că <u>$\Gamma \cup \{\varphi\}$</u> se scrie și: <u>Γ,φ</u>, sau <u>$\Gamma \cup \varphi$</u>
- Regulile „pentru $\forall$" de <u>***i***</u>ntroducere/<u>***e***</u>liminare, sunt cumva ***duale*** cu cele „pentru $\exists$", de <u>***e***</u>liminare/<u>***i***</u>ntroducere
- Primele două reguli duale sunt simple

- Pentru prima regulă **<u>(eliminarea cuantificatorului universal)</u>**, *ipoteza* ne „spune” că $(\forall x.\varphi)$ este consecință sintactică din Γ
- *Intuitiv*, putem deci *instanția variabila legată* x cu orice *valoare* (adică, *abstract*, cu orice *termen t*)
- În *concluzie*, *orice* asemenea *nou* φ va fi consecință sintactică din aceeași mulțime Γ

<u>Exemplu</u> (în curs - <u>119, pag.52-53</u>; făcut).

Demonstrația validității secvenței (în Cursul „vorbit” se folosește de obicei metoda „*backward thinking*”):

$\{\forall x.(Om(x) \rightarrow Muritor(x)), Om(s)\} \vdash Muritor(s)$.

- Să comentăm a doua regulă simplă, adică **introducerea cuantificatorului existențial**
- În această situație, „știm” (din ipoteză) că formula $\varphi[x \mapsto t]$ este consecință semantică din mulțimea de formule Γ („indiferent” de cine sunt Γ, φ, x, *t*)
- Intuitiv, ar rezulta în mod evident că $(\exists x.\varphi)$ poate fi, la rândul ei consecință din (aceeași) Γ, „valoarea” lui x putând fi considerată ca fiind „egală” cu cea a oricărui *t* (care se „trece” pe linia unde se folosește regula, în demonstrații)

Exemplu (în curs - 121, pag.53; făcut).

Să arătăm că secvența

$\{(\forall x.(P(x) \rightarrow Q(x))), P(a)\} \vdash (\exists x.Q(x))$ este validă.

- Să explicăm în continuare cea de-a treia regulă folosită pentru manipularea sintactică a cuantorilor, și anume **<u>introducerea cuantificatorului universal</u>**
- Ea ne „spune” că vom putea deriva *concluzia* de acolo, $\Gamma \vdash (\forall x.\varphi)$, dacă vom putea „arăta înainte” că $\varphi[x \mapsto x_0]$ (*ipoteză*) este consecință semantică din Γ
- Regula în sine pare mai complicată deoarece are și o condiție, pe care am notat-o (c_1)
- Condiția „în sine” este însă simplă, deoarece „afirmă” *doar* faptul că x_0 este o *variabilă nouă <u>oarecare</u>* (practic, că ***nu*** *mai apare* în ceea ce aveam până atunci, adică în $\Gamma \cup \varphi$)
- Numele x_0 se „trece” pe linia unde se aplică regula

<u>Exemplu</u> (în curs - <u>122, pag.54</u>; făcut).

Să arătăm că secvența

$\{(\forall x.(P(x) \to Q(x))), (\forall x.P(x))\} \vdash (\forall x.Q(x))$ este validă.

- Se poate observa că în demonstrație utilizăm (desigur) variabila x_0, și că asupra ei nu mai facem nicio altă presupunere (înafară de faptul că *nu mai apare …*)
- Deci, intuitiv, $Q(x_0)$ va putea fi derivat, în aceeași manieră, <u>pentru orice x_0 posibil</u>

- În sfârșit, să trecem și la ultima regulă pe care am introdus-o, adică **<u>eliminarea cuantificatorului existențial</u>** (duala regulii ...)

- Prima ipoteză a regulii este $\Gamma \vdash (\exists x.\varphi)$, care ne asigură că există cel puțin un termen *t* (pot fi mai mulți, desigur) care poate înlocui variabila x astfel încât φ să fie consecință sintactică din Γ
- Nu știm însă care este/sunt acest/acești termen/termeni
- Știm însă, totuși, că *măcar unul există*
- Generic, *toți/fiecare* acești *t* vor *purta numele* x_0
- Pentru a demonstra concluzia, adică faptul că ψ este consecință sintactică din Γ, va trebui să facem, practic, o *analiză pe cazuri* după toți asemenea posibili x_0

- Acest lucru este practic sumarizat de a doua ipoteză a regulii, care ne „spune” că, înainte de a trage vreo concluzie, trebuie arătat (și) că ***orice*** ψ este consecință sintactică din $\Gamma \cup \{\varphi[x \mapsto x_0]\}$ (similaritate cu eliminarea lui „$\vee$”; ar putea fi „mai mulți de $\vee$”, chiar „o infinitate” ...)

Exemplu (în curs - 124, pag.55). Să arătăm că este validă secvența $\{(\forall x.(P(x) \rightarrow Q(x))), (\exists x.P(x))\} \vdash (\exists x.Q(x))$. (făcut)

- După cum am mai precizat, ***teorema de corectitudine și completitudine rămâne adevărată*** și în cazul **LP1** (e vorba desigur despre **DN**); altfel spus, ADEVĂR „=” DEMONSTRAȚIE; relațiile consecință semantică ($\vDash$) și consecință sintactică ($\vdash$) coincid, etc.

- Recapitulăm puțin și **DN** pentru **LP1**
- De cele mai multe ori, o demonstrație se construiește „în stil” *backward* și se verifică (pentru siguranță) în modul *forward*
- (**Atenție**: regula „∃**e**” înseamnă „eliminarea unui cuantificator ∃”, dar ... **nu** din formula ψ !)
- Să arătăm că este validă secvența:

$$\Gamma = \{(\exists x.(\exists y.P(x, y)))\} \vdash (\exists y.(\exists x.P(x, y))).$$

Notăm cu $\Gamma' = \Gamma, (\exists y.P(x_0, y))$ și cu

$$\Gamma'' = \Gamma', P(x_0, y_0),$$

unde x_0, y_0 sunt variabile „noi”, fixate deci; ideea este că trebuie să-i „*punctăm*” pe cei 2 de „∃”, pentru a-i putea elimina, și apoi să-i reintroducem „altfel” (în altă ordine ...); aici vom „face” totul (direct) *forward*, inspectând desigur regulile necesare

1. $\Gamma'' \vdash P(x_0, y_0)$ (IPOTEZĂ)
2. $\Gamma'' \vdash (\exists x.P(x, y_0))$ ($\exists$**i**, 1, $t = x_0$)
3. $\Gamma'' \vdash (\exists y.(\exists x.P(x, y)))$ ($\exists$**i**, 1, $t = y_0$)
4. $\Gamma' \vdash (\exists y.P(x_0, y))$ (IPOTEZĂ)
5. $\Gamma' \vdash (\exists y.(\exists x.P(x, y)))$ ($\exists$**e**, 4, 3)

Observația *1*). În 4., care este prima ipoteză pentru o instanță a regulii „$\exists$**e**": Γ' joacă rolul lui Γ; y este x de acolo; iar $\varphi = P(x_0, y)$; apoi, 3. desemnează cea de-a doua ipoteză a aceleiași instanțe a regulii, unde: $\Gamma'' = \Gamma' \cup \{\varphi[y \mapsto y_0]\}$, i.e. $\Gamma'' = \Gamma'$, $P(x_0, y_0)$; avem și $\psi = (\exists y.(\exists x.P(x, y)))$.

Observația *2)*. Până acum, am dedus formula care ne trebuie, adică ψ, din Γ'' (ușor, pentru că am „pus corect" ipotezele suplimentare) și din Γ' (mai greu, apelând la **e**liminarea unui „∃"). Vom apela la aceeași regulă „∃**e**" (cu altă instanțiere, dar în același „stil") pentru a o obține pe ψ și din Γ, ceea ce se cerea de la bun început.

6. $\Gamma \vdash (\exists x.(\exists y.P(x, y)))$ (IPOTEZĂ)

7. $\Gamma \vdash (\exists y.(\exists x.P(x, y)))$ (∃**e**, 6, 5)

Observația *3)*. În 6., prima ipoteză a instanței alese, Γ este el însuși; la fel x; iar $\varphi = (\exists y.P(x, y))$; apoi, în 5., adică cea de-a doua ipoteză a aceleiași instanțe, avem: $\Gamma' = \Gamma \cup \{\varphi[x \mapsto x_0]\}$, i.e. $\Gamma' = \Gamma, (\exists y.P(x_0, y))$; și desigur, după cum am spus, $\psi = (\exists y.(\exists x.P(x, y)))$.

Final Seminar 3

74

- Ce facem în seminarul atașat **Cursului 4**, **LP1** (săptămâna a 12-a de școală, incluzând lucrarea de evaluare)

***1*.Recapitulare**: *satifiabilitate*, *nesatisfiabilitate*, *validitate* și *consecință semantică*; legăturile dintre acestea (la nivel „local” = de structură, și la nivel „global” = „toate structurile”); *substituții*.

2.Formule echivalente în LP1: la nivel local și la nivel global.

3.Forme normale (prima parte) **pentru** (formulele) **LP1**: *forma normală prenex* (**FNP**); *lema de redenumire*; **algoritm** pentru „*aducerea cuantificatorilor în fața unei formule*”.

4.Clase importante de formule în **LP1** (aflate, eventual, în **FNP**): formule *închise/deschise*; *închideri* ale formulelor: *închiderea existențială* și *închiderea universală*; *formule echisatisfiabile*; echivalența/echisatisfiabilitatea – legătura cu satisfiabilitatea/validitatea și închiderile.

2. Formule echivalente

Definiție. Două formule φ_1 și $\varphi_2 \in$**LP1** se numesc ***echivalente în*** (Σ-) ***structura*** **S** dacă pentru fiecare (**S**-)atribuire α avem:

$$\mathbf{S}, \alpha \vDash \varphi_1 \text{ ddacă } \mathbf{S}, \alpha \vDash \varphi_2.$$

Acest lucru se notează prin $\varphi_1 \equiv^{\mathbf{S}} \varphi_2$ (simbolul **S** se pune de obicei *exact deasupra* simbolului $\equiv$).

<u>Exemplu</u> (făcut și în curs)

Fie signatura $\Sigma = \langle\{P\}, \{f, i, e\}\rangle$ (arități: 2; 2, 1, 0) și Σ-structura **S1** = $\langle$**Z**, {=}, {+, -, 0}$\rangle$. Să arătăm că $P(x, y) \equiv^{\mathbf{S1}} P(y, x)$ (deoarece pentru orice **S1**-atribuire α avem …) și, respectiv, $P(x_1, x_3) \not\equiv^{\mathbf{S1}} P(x_2, x_3)$ (adică ***nu*** este adevărat că ...).

Definiție. Două formule φ_1 și $\varphi_2 \in$ **LP1** se numesc (Σ-)***echivalente*** dacă <u>pentru fiecare</u> (Σ-)structură **S** <u>și</u> <u>pentru fiecare</u> (**S**-)atribuire α <u>avem</u>:

$$\mathbf{S}, \alpha \vDash \varphi_1 \text{ ddacă } \mathbf{S}, \alpha \vDash \varphi_2.$$

Notăm aceasta prin $\varphi_1 \equiv \varphi_2$ (reamintim că asta este același lucru cu: $\varphi_1 \equiv^{\mathbf{S}} \varphi_2$ <u>*pentru fiecare*</u> **S**).

Exemplu (făcut și în curs)

Continuăm cu $\sum$ și **S1** din exemplul anterior, precum și cu **S5** = $\langle$**Z**, {<}, {+, -, 0}$\rangle$. Să arătăm că: $P(x, y) \not\equiv P(y, x)$. În aceleași condiții, *arătați voi* că $(\forall x.P(x, z)) \equiv (\forall y.P(y, z))$.

3.Forme normale

Definiție. O formulă $\varphi \in$**LP1** este ***în formă normală prenex*** (**FNP**), dacă toți cuantificatorii care apar (textual) în φ *sunt „în fața" formulei*.

Exemplu.

$\varphi_1 = (\forall x.(\exists y.(P(x, y) \wedge \neg P(z, y))))$ este în **FNP**.

$\varphi_2 = ((\forall x.(\exists y.P(x, y))) \wedge \neg P(z, y))$ ***nu*** este în **FNP**.

Formal (ultima definiție - **FNP**):

$\varphi = Q_1x_1.Q_2x_2. \ldots .Q_nx_n.\varphi'$, unde

1. Pentru fiecare $i \in [n]$, $Q_i \in \{\forall, \exists\}$.
2. φ' nu conține niciun cuantificator.

- Deoarece domeniile de vizibilitate sunt „clare”, nu am pus explicit parantezele
- Legat de substituții, reamintim că am introdus funcția $\sigma : \mathcal{X} \rightarrow \mathcal{T}$; *dom*($\sigma$); extensia $\sigma^{\#} : \mathcal{T} \rightarrow \mathcal{T}$; extensia σ^{b} : **LP1** $\rightarrow$ **LP1**; restricția $\sigma|_V : \mathcal{X} \rightarrow \mathcal{T}$ ($V \subseteq \mathcal{X}$, aici fiind suficient să luăm cazul $V \subseteq$ *dom*(σ)); alte notații: $\varphi[x \mapsto t]$, care „provine” din $\sigma = \{x_1 \mapsto \sigma(x_1), \ldots, x_k \mapsto \sigma(x_k)\}$ (în cursul scris, căutați între pag. 39 – pag. 42)

Teoremă (de existență a **FNP).** Pentru orice formulă $\varphi \in$**LP1**, există (măcar) o altă formulă $\varphi' \in$**LP1** (numită și ***o*** **FNP *a lui/pentru*** φ), a.î.:

1. φ' este în **FNP**.
2. $\varphi \equiv \varphi'$.

- În scopul prezentării unui ***algoritm general*** *pentru calculul* ***unei*** **FNP** (intrare - o(rice) formulă din **LP1**)*,* avem nevoie (și) de ***lema de redenumire*** (demonstrația ei – prin *inducție structurală* asupra definiției inductive a lui **LP1**)
- Poate ar fi bine să vă reamintiți și de (măcar) funcția *free*(φ) ... (în curs, la pag. 24)

Lemă (de redenumire - LR). Fie $\varphi \in$ **LP1** o formulă (oarecare) și x, y $\in \mathcal{X}$ două variabile distincte *oarecare*, cu proprietatea că y $\notin$ *free*(φ). Atunci avem:

-$(\forall x.\varphi) \equiv (\forall y.\sigma^b(\varphi))$ și

-$(\exists x.\varphi) \equiv (\exists y.\sigma^b(\varphi))$,

unde $\sigma = \{x \mapsto y\}$.

- Cu alte cuvinte, în $(\forall x.\varphi)/(\exists x.\varphi)$ putem înlocui cuantorul $\forall$x/$\exists$x cu altul, $\forall$y/$\exists$y, cu condiția ca y să nu apară liber în φ (de ex., acesta poate fi *nou*, ca la **DN** – e mai „sigur” ...)
- În plus, în acest caz, *toate aparițiile libere* ale lui x în φ trebuie înlocuite cu y (se aplică σ^b lui φ)

<u>Exemplu.</u> (făcut și în curs)

Să aplicăm lema precedentă formulei $\varphi = (\forall x.P(x, y))$. Ce „s-ar schimba” dacă „luăm” y (în loc de „o altă” variabilă z) ?

- Algoritmul menționat anterior cu ajutorul căruia, practic, se **demonstrează Teorema** (de existență a **FNP**), este dat printr-o *secvență de echivalențe*, care se vor aplica *nedeterminist*, de preferință în ordinea sugerată mai jos, formulei *inițiale/intrării* φ (precum și celor *intermediare*), pentru a obține, *în final* (adică atunci când *nu se mai poate face nicio înlocuire)*, formula φ' – care este clar în **FNP**):

1. $(\forall x.\varphi_1) \wedge \varphi_2 \equiv (\forall x.(\varphi_1 \wedge \varphi_2))$, dacă $x \notin free(\varphi_2)$.
2. $(\forall x.\varphi_1) \vee \varphi_2 \equiv (\forall x.(\varphi_1 \vee \varphi_2))$, dacă $x \notin free(\varphi_2)$.
3. $(\exists x.\varphi_1) \wedge \varphi_2 \equiv (\exists x.(\varphi_1 \wedge \varphi_2))$, dacă $x \notin free(\varphi_2)$.
4. $(\exists x.\varphi_1) \vee \varphi_2 \equiv (\exists x.(\varphi_1 \vee \varphi_2))$, dacă $x \notin free(\varphi_2)$.
5. $\rceil(\forall x.\varphi) \equiv (\exists x.\rceil\varphi)$.
6. $\rceil(\exists x.\varphi) \equiv (\forall x.\rceil\varphi)$.

- Dacă 1.-4. nu pot fi aplicate datorită restricției, mai întâi aplicăm corect **LR** (sunt și forme particulare ale lor, mai simple – vezi **Fișele de exerciții**)
- Pe parcurs putem apela, deși am demonstrat formal echivalențe doar pt. **LP**), la *comutativitatea* lui $\wedge$/$\vee$; *deMorgan*; la *exprimarea* $\rightarrow$/$\leftrightarrow$ cu ajutorul …; adică pct. 7.-10., pag. 64, din curs

- Este evident că se face apel și la o **Teoremă de înlocuire** (identică cu cea enunțată pentru **LP**), și că (repetăm) *algoritmul este*, în esență, *o buclă nedeterministă*, la fiecare pas executându-se (e vorba despre „corpul” buclei) – printr-o anumită alegere de moment – o înlocuire indicată de una dintre echivalențele din secvența 1.-10. de mai sus (până când „nu se mai poate”); *cu ordini diferite de aplicare, putem obține o altă formă normală*

Exemplu. (inclusiv „cum scriem” – în curs)

Să aplicăm algoritmul descris anterior formulei (nu vom explicita chiar fiecare înlocuire, cum ar fi, de ex., cele legate de comutativitate, etc.):

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

4. Clase importante

Formule *închise/deschise* și *închideri*.

Definiție. O formulă $\varphi \in$ **LP1** este ***închisă*** (eng.: ***sentence***) dacă *free*(φ) = Ø.

Definiție. O formulă $\varphi \in$ **LP1** este ***deschisă*** dacă ***nu*** este închisă.

Definiție. Fie $\varphi \in$ **LP1** o formulă oarecare, cu *free*(φ) = $\{x_1, x_2, \ldots, x_n\}$. ***Închiderea existențială*** a lui φ, va fi formula $(\exists x_1.(\exists x_2. \ldots .(\exists x_n.\varphi) \ldots))$.

Exemplu.

a)Formula

$\varphi = (\forall z.(\exists y.(\rceil(P(z, z) \wedge \rceil P(z, y))) \wedge P(x, x))$ ***nu*** este închisă.

b)Închiderea existențială a formulei precedente este ...

- Evident: închiderea existențială a oricărei formule este o formulă închisă

Definiție. Două formule φ_1 și $\varphi_2 \in$ **LP1** se numesc ($\sum$-)***echisatisfiabile*** dacă:

1.Atât φ_1 cât și φ_2 sunt satisfiabile, sau

2.Nici φ_1, nici φ_2 nu sunt satisfiabile.

- Cu alte cuvinte: singura posibilitate ca 2 formule să *nu fie* echisatisfiabile, este ca una dintre ele să fie satisfiabilă, iar cealaltă să nu fie satisfiabilă
- Am putea restrânge noțiunea de echisatisfiabilitate (definiție locală) la o structură… (pe moment, nu ne folosește)

Teoremă. Orice formulă este echisatisfiabilă cu închiderea sa existențială.

- Rezultatul nu este dificil de demonstrat (încercați singuri ...)
- Prin „dualizare”, avem imediat:

Definiție. Fie $\varphi \in$ **LP1** o formulă oarecare, cu *free*(φ) = $\{x_1, x_2, \ldots, x_n\}$. ***Închiderea universală*** a lui φ, este formula $(\forall x_1.(\forall x_2. \ldots .(\forall x_n.\varphi) \ldots))$.

- Desigur că (și) închiderea universală a oricărei formule este o formulă închisă
- Nu este – de asemenea – greu (încercați singuri) de demonstrat:

Teoremă. O formulă este validă ddacă închiderea sa universală este validă.

Exemplu.

Închiderea universală a formulei din exemplul anterior (aici, în slide 85) este …

Final Seminar 4

- Ce facem în **Cursul 5**, **LP1** (săptămâna a 13-a de școală, inclusiv cea de evaluare)

1.Recapitulare Curs 4: formule *închise* în **LP1**; *închiderea existențială* și *închiderea universală* pt. $\varphi \in$**LP1**; relația de *echisatisfiabilitate* în **LP1**; *validitatea/echisatisfiabilitatea* închiderilor; *legături*; *forma normală prenex* (**FNP**) în **LP1**; algoritm pentru aducerea formulelor la ***o*** **FNP**.

2.Forma normală Skolem (FNS): eliminarea cuantificatorilor existențiali din **FNP**-uri; teorema/algoritmul de existență/aducere la ***o*** **FNS** (plecând de la ***o*** **FNP** a unei formule).

3.Forma normală conjunctivă/clauzală (FNC): *literali* și *clauze* în **LP1**.

4.Forma normală Skolem clauzală (FNSC) pentru formulele **LP1**: teorema/*algoritmul* de existență/aducere a/la ***o*** **FNSC** a unei $\varphi \in$**LP1** (pornind de la ***o*** **FNP** sau de la ***o*** **FNS** a lui φ); relația dintre φ și diversele sale forme normale.

5.Rezoluția de bază în LP1: este un **sistem deductiv** (corect, ne-complet, analog cu cazul **LP**), folosit pentru ***testarea nesatisfiabilității*** unei $\varphi \in$**LP1**; *termeni*, *substituții* și *formule „de bază”* (eng. = *ground*); ***Teorema rezoluției de bază***.

1.Recapitulare Curs 4

- Reamintirea *algoritmului nedeterminist general* „de aducere la (o) **FNP** echivalentă” pentru o formulă dată $\varphi \in$**LP1** (de spus *en avant*: *închiderea existențială și păstrarea echisatisfiabilității*); folosirea a 10 *echivalențe* + **Teorema de înlocuire** (**Teorema 138**, pag. 63-64).

2.FNS

Definiție. O formulă φ este în ***formă normală Skolem*** (prescurtat, **FNS**) dacă $\varphi = \forall x_1.\forall x_2. \ldots .\forall x_n.\varphi'$,

unde:

1. φ' nu conține cuantificatori și

2. *free*$(\varphi') \subseteq \{x_1, x_2, \ldots, x_n\}$ (bine - chiar „=”, în loc de „ $\subseteq$”).

Cerința intuitivă (pt. ca o formulă φ să fie în **FNS**): φ are ***doar cuantificatori universali***, aceștia ***sunt toți „în față”*** și ***toate variabilele*** (care apar) ***sunt cuantificate***.

Exemplu. (am dat aritățile și am pus toate parantezele)
Fie signatura $\sum = \langle\{P/2, Q/1, R/3\}, \{f/2, i/1, e/0\}\rangle$ și:

$\varphi_1 = (\forall x.P(x, i(e)))$;

$\varphi_2 = (\forall x.(\forall y.(P(f(x, e), y) \wedge \rceil (R(x, i(f(y, y)), e) \vee Q(y))))))$;

$\varphi_3 = (\exists x.P(x, x))$, $\varphi_4 = (\forall x.(Q(e) \wedge \rceil (Q(x) \vee Q(y))))$;

$\varphi_5 = (Q(e) \wedge (\forall x.Q(x)))$.

Care dintre ele este în **FNS** ? (care NU este, de ce ?)

Teoremă (*de aducere în* **FNS**). Pentru orice formulă $\varphi \in$**LP1**, există o formulă $\varphi' \in$**LP1** astfel încât:

a) φ' este în formă normală Skolem;

b) φ și φ' sunt echisatisfiabile.

Observație. În anumite situații particulare, φ și φ' pot fi chiar echivalente. Se observă ușor că: dacă 2 formule sunt echivalente, atunci ele sunt și echisatisfiabile; reciproca însă nu este adevărată „mereu”. Oricum, φ‘ este în **FNP** și închisă.

Demonstrație (schiță):

1. „Calculăm” o formulă φ_1, aflată în **FNP** și echivalentă cu formula φ (folosind algoritmul știut).

2. Obținem φ_2, închiderea existențială a lui φ_1 (dacă φ_1 nu este închisă; cele două formule vor fi doar echisatisfiabile).

3. Aplicăm repetat **Lema** de skolemizare (urmează imediat mai jos) asupra formulei φ_2; în final obținem o formulă aflată în **FNS**, închisă și echisatisfabilă (uneori - echivalentă) cu formula φ.

Lemă (*de skolemizare*). Fie $\varphi = \forall x_1.\forall x_2. \ldots .\forall x_k.\exists x.\varphi'$, $k \geq 0$, unde φ' poate conține și alți cuantificatori (cu alte cuvinte, există exact k cuantificatori universali înainte de, textual, *primul* cuantificator existențial din formulă, st-dr, pus aici în evidență). Fie $f \in \mathcal{F}_k$ un simbol funcțional oarecare, dar care nu apare *deloc* în φ (nou/proaspăt/*fresh*). Atunci φ este echisatisfiabilă cu (știm cine e σ^b) formula

$\varphi'' = \forall x_1.\forall x_2. \ldots .\forall x_k.(\sigma^b(\varphi'))$, unde substituția σ este dată prin

$\sigma = \{x \mapsto f(x_1, x_2, \ldots, x_k)\}$ (aparițiile libere ale lui x din ... se ...).

Demonstrație (schiță; în curs, pag.69-70):

Practic, fiecare cuantor $\exists x.$ este precedat textual (st-dr) în formula curentă (în **FNP**, dar nu neapărat închisă), de un număr de cuantori universali *k* (care poate fi chiar 0); acesta se șterge efectiv din text și apoi toate aparițiile libere (din formula ...) ale lui x se înlocuiesc cu termenul $f(x_1, x_2, \ldots, x_k)$, *f* fiind un simbol funcțional **nou**, de aritate *k*, iar $x_1, x_2, \ldots, x_k$ sunt chiar variabilele cuantificate menționate. Dacă φ este construită peste signatura $\Sigma=\langle \mathcal{P}, \mathcal{F} \rangle$, atunci formula φ'' va fi construită peste o signatură „mai bogată”, $\Sigma' = \langle \mathcal{P}, \mathcal{F} \cup \{f\} \rangle$.

Presupunem mai întâi că există o Σ-structură **S** și o **S**-atribuire α astfel încât **S**, $\alpha \vDash \varphi$. Arătăm că există o Σ'-structură **S'** astfel încât **S'**, $\alpha \vDash \varphi''$ (este într-adevăr *același* α; iar **S'** este peste signatura Σ' care este mai bogată decât signatura structurii inițiale, Σ.

Invers, trebuie desigur presupus că există o Σ'-structură **S'** și o **S'**-atribuire α, astfel încât **S'**, $\alpha \vDash \varphi''$, și găsită o Σ-structură **S** și o **S**-atribuire α' (eventual – din nou - aceeași cu α, deoarece *mulțimea de variabile nu se schimbă*) astfel încât **S'**, $\alpha' \vDash \varphi$.

De obicei, se „lucrează” de la început cu „cea mai completă” signatură necesară (care conține toate simbolurile noi necesare).

Exemplu.

Să se calculeze o **FNS** pentru formula

$\varphi = \forall x.\underline{\exists} y.\forall z.\exists z'.(P(x, y) \leftrightarrow P(z, z'))$ (este-?- în **FNP**, și este și închisă; de pus paranteze ...)

Avem succesiv, aplicând de 2 ori lema anterioară, că φ este echisatisfiabilă cu φ_1, apoi cu φ_2, unde (lăsăm totuși pe $\leftrightarrow$; aici „merge” ...):

$\varphi_1 = \forall x.\forall z.\underline{\exists} z'.(P(x, g(x)) \leftrightarrow P(z, z'))$ și

$\varphi_2 = \forall x.\forall z.(P(x, g(x)) \leftrightarrow P(z, h(x,z)))$,

unde *g*/1 și *h*/2 de mai sus sunt simboluri funcționale noi, de arități corespunzătoare.

3.FNC

- Ca și în cazul **LP**, un *literal* va fi o formulă atomică sau negația unei formule atomice; mai exact (în noul context):

Definiție. O formulă $\varphi \in$ **LP1** se numește ***literal*** dacă există un simbol predicativ $P \in \mathcal{P}_n$ (cu $n \geq 0$) și n termeni $t_1, \ldots, t_n \in \mathcal{T}$ astfel încât

$\varphi = P(t_1, \ldots, t_n)$, sau $\varphi = \rceil P(t_1, \ldots, t_n)$.

Exemple (pag. 70 în curs; aici, folosim ultima signatură): $\rceil P(x, x)$, $Q(i(y))$, $R(e, f(x, y), e)$, …

- De remarcat mai sus: P și P ... (mici scăpări în curs)

Definiție. O formulă $\varphi \in$ **LP1** se numește ***clauză*** dacă există *n* literali $\varphi_1, \ldots, \varphi_n$ (adică, formule atomice din **LP1** sau negații ale lor), astfel încât $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ (paranteze ...)

- Desigur că admitem în continuare existența „clauzei vide” (notată tot prin $\square$), care este considerată a fi o formulă/clauză nesatisfiabilă (nu uităm că un unic literal poate fi privit atât ca o clauză cât și ca o „succesiune” de $\wedge$-uri)

Definiție. O formulă $\varphi \in$ **LP1** este ***în formă normală clauzală/conjunctivă*** (**FNC**) dacă există *m* clauze

$\psi_1, \ldots, \psi_m \in$ **LP1** astfel încât $\varphi = \psi_1 \wedge \ldots \wedge \psi_m$.

Exemple (vezi cursul scris, pag.71-72).

4.FNSC

Definiție. O formulă $\varphi \in$ **LP1** este în ***formă normală Skolem clauzală*** (prescurtat **FNSC**) dacă φ este în formă normală Skolem (**FNS**) și φ' este în formă normală clauzală (**FNC**); mai precis:

$\varphi = \forall x_1.\forall x_2. \ldots .\forall x_n.\varphi'$, unde φ' nu conține cuantificatori (deci φ' este chiar subformula obținută din φ prin ștergerea efectivă a cuantificatorilor, care sunt toți „în față”); și, desigur, φ' este (și) în **FNC**.

Observație. Practic, avem și $vars(\varphi') = free(\varphi') = \{x_1, x_2, \ldots, x_n\}$. Cuantificările „duble” și cele „fără rost” se pot elimina.

Teoremă. Pentru orice formulă $\varphi \in$**LP1** aflată în **FNS** există o formulă $\varphi' \in$**LP1** astfel încât:

1. φ' este în **FNSC** și

2. $\varphi \equiv \varphi'$.

Demonstrație (schiță):

Se aplică succesiv următoarele înlocuiri de subformule (membrul stâng al echivalențelor se înlocuiește cu membrul drept), formulei inițiale (și apoi celor rezultate în urma transformărilor efectuate), *preferabil, dar nu esențial* (textual, de la stânga la dreapta) în ordinea dată mai jos (a se revedea **LP** și *algoritmul (nedetermist) general* pentru găsirea unei **FNC** de acolo):

1. $(\varphi_1 \leftrightarrow \varphi_2) \equiv (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$
2. $(\varphi_1 \rightarrow \varphi_2) \equiv (\rceil \varphi_1 \vee \varphi_2)$
3. $\rceil \rceil \varphi \equiv \varphi$
4. $\rceil (\varphi_1 \vee \varphi_2) \equiv (\rceil \varphi_1 \wedge \rceil \varphi_2)$
5. $\rceil (\varphi_1 \wedge \varphi_2) \equiv (\rceil \varphi_1 \vee \rceil \varphi_2)$
6. $(\varphi_1 \vee (\varphi_2 \wedge \varphi_3)) \equiv ((\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3))$

- <u>Să re-punctăm</u> că (și) aici se permite folosirea liberă a asociativității și comutativității pentru $\wedge$/$\vee$, că operațiile se execută „atât timp cât este posibil” și că rămâne valabilă **Teorema de înlocuire** de la **LP**. De asemenea, că algoritmul este practic identic cu cel folosit în **LP** pentru aflarea **FNC**, variabilele propoziționale putând fi „înlocuite” cu formule atomice din **LP1** (acestea ar putea fi chiar denotate „local” prin *p*, *q*, ..., etc.)

Observație. În urma rezultatelor obținute, putem concluziona că pentru orice formulă $\varphi \in$**LP1** există (*măcar*) ***o*** formulă $\varphi' \in$**LP1** astfel încât φ' este în **FNSC** iar formulele φ și φ' sunt echisatisfiabile (uneori, ele pot fi chiar echivalente). Pentru a găsi o asemenea φ', se urmează pașii următori:

1. Din φ, calculăm ***o*** **FNP** a sa, φ_1.

2. „Găsim" ***o*** φ_2, ***închiderea existențială*** a lui φ_1.

3. Aplicăm **Lema de skolemizare** lui φ_2, eliminând cuantificatorii existențiali; obținem o nouă formulă, φ_3, aflată în **FNS** (care e închisă, fără „există", etc.).

4. Aplicăm teorema de aducere la **FNC** a lui φ_3 (dacă mai este cazul) și găsim φ_4, în **FNSC** (φ_4 este φ').

<u>Exemplu concret</u> de „aducere în” **FNSC** (vezi și cursul scris).

Ne interesează să vedem dacă formula φ:

$$\varphi = (\forall x.(Q(x) \leftrightarrow \rceil Q(i(x)))) \rightarrow$$
$$\rightarrow (\forall x.(Q(x) \rightarrow Q(i(i(x))))) \text{ este validă.}$$

Pentru aceasta, (este suficient să) arătăm că $\rceil\varphi$ este nesatisfiabilă. Vom obține mai întâi o formulă aflată în **FNSC**, și echisatisfiabilă cu $\rceil\varphi$:

1.O **FNP**, φ_1, corespunzătoare este: …

2.-3.Din φ_1 (care este închisă) obținem (aplicând **Lema de skolemizare**) o formulă φ_2, aflată în **FNS** (se introduce doar un simbol funcțional nou, $g \in \mathcal{F}_1$)

4.În sfârșit, din φ_2 se obține φ_3, în **FNSC** (și echisatisfiabilă cu φ), procedând conform „celor știute”:

$$\varphi_3 = \forall x.((\rceil Q(x) \vee \rceil Q(i(x))) \wedge (Q(i(x)) \vee Q(x)) \wedge Q(g(x)) \wedge$$
$$\wedge \rceil Q(i(i(g(x)))))$$

<u>**5.Rezoluția de bază (*ground resolution*).**</u> Pentru a testa (ne)satisfiabilitatea unei formule din **LP1**, aflată în **FNSC**, putem folosi „sistemul deductiv *de tip rezoluție*” descris în continuare.

Definiție. Un termen $t \in \mathcal{T}$ se numește ***termen de bază*** dacă *vars*(t) = Ø. Similar, o formulă $\varphi \in$**LP1** este ***formulă de bază*** dacă *vars*(φ) = Ø.

Definiție. O substituție $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ se numește ***substituție de bază*** dacă termenii $t_1, \ldots, t_n$ sunt **toți** termeni de bază.

Definiție. În mod similar cu ceea ce am făcut în cazul rezoluției propoziționale, ***rezoluția de bază*** este introdusă printr-un sistem deductiv, având o unică (schemă de) regulă de inferență (**RB**); de fapt mai poate fi una numită ***factorizare*** (<u>încă nu</u> ...):

$$\frac{C_1 \vee P(t_1, \ldots, t_n) \qquad C_2 \vee \rceil P((t_1)', \ldots, (t_n)')}{(\sigma_1)^b(C_1) \vee (\sigma_2)^b(C_2)}$$

În cele de mai sus, <u>σ_1 is σ_2 sunt substituții de bază alese astfel încât să avem $(\sigma_1)^\#(t_i) = (\sigma_2)^\#((t_i)')$</u>, pentru fiecare $i \in [n]$ (știm cine este extensia $\sigma^\#$ pt. σ).

Teoremă (a *rezoluției de bază*). O formulă $\varphi \in$ **LP1**, aflată în **FNSC**, este nesatisfiabilă ddacă se poate obține □, printr-o *demonstrație utilizând rezoluția de bază*, pornind cu mulțimea clauzelor care compun φ, ca „axiome” (similar cu rezoluția din **LP**, <u>neglijând prezența cuantificatorilor din față</u>); reprezentăm apoi <u>φ</u> - ștergând cuantificatorii din față, dar notând la fel rezultatul - <u>ca mulțime de clauze</u>; putem <u>**reprezenta și clauzele ca mulțimi de literali**</u>, etc.).

Exemplu. Continuați ultimul exemplu („concret”), de unde l-am lăsat. Se poate obține □ în maxim 7 pași de demonstrație (vezi și cursul scris).

Final Seminar 5

- Ce facem în **Curs 6**, **LP1** (ultimul; săptămâna a 14-a de școală, incluzând lucrarea de evaluare)

1.Recapitulare Curs 5: rezoluția de bază pentru **LP1**.

2.Unificare: *unificatori*; *termeni unificabili*; unificatori „*mai generali* ” și *cel mai general unificator* (**m***ost* **g***eneral* **u***nifier* – m.g.u./mgu); *problemă de unificare*; cea mai generală *soluție* a unei probleme de unificare și *probleme rezolvate*.

3.Rezoluția de ordinul I (sau, eng.: „*pure resolution*”): rezoluția pură ca **SD** cu 2 (scheme de) reguli de inferență; **Teorema rezoluției LP1**.

1.Recapitulare

- Principiul general al rezoluției; termeni de bază; rezoluția de bază ca **SD** cu o (singură) regulă de inferență; exemple de demonstrații pentru validitatea/nesatisfiabilitatea formulelor **LP1** utilizând rezoluția de bază

2.Unificare

Definiția 1.1 (Unificator)**.** O substituție σ este un ***unificator*** al termenilor $t_1, t_2 \in \mathcal{T}$ dacă

$\sigma^{\#}(t_1) = \sigma^{\#}(t_2)$.

<u>**Exemplu.**</u> (cf. ultima signatură $\sum$, deși aici ...)

$t_1 = f(x, h(y))$, $t_2 = f(h(z), z')$.

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

$\sigma_1 = \{x \mapsto h(z), z' \mapsto h(y)\}$.

Definiție. (Termeni unificabili). Doi termeni t_1, $t_2 \in \mathcal{T}$ sunt ***unificabili*** dacă „au” (măcar) *un* unificator. (adică, există măcar ***o*** σ, a.î. ...)

<u>**Exemplu.**</u>

a)Termenii $t_1 = f(x, y)$ și $t_2 = h(z)$ <u>nu</u> sunt unificabili. (puteam lua și $ar(h) = 2$, că ...)

b)Termenii t_1 = x și t_2 = h(x) <u>nu</u> sunt unificabili. (nr. de noduri din arbore ...)

Definiție. (Compunerea a două substituții). Fie σ_1 și σ_2 două substituții. Substituția $\sigma = \sigma_2 \circ \sigma_1$, $\sigma_2 \circ \sigma_1 : \mathcal{X} \to \mathcal{T}$, este ***compunerea substituțiilor*** σ_1 și σ_2 dacă $(\sigma_2 \circ \sigma_1)(x) = (\sigma_2)^{\#}(\sigma_1(x))$, pentru fiecare $x \in \mathcal{X}$. (vezi diagrama ...)

<u>Exemplu.</u>

Fie σ și σ_1 substituțiile de pe <u>slide 108</u>. Atunci $\sigma_2 \circ \sigma_1 = \sigma$, unde $\sigma_2 = \{z \mapsto a\}$.

Definiție. (substituție *mai generală*). O substituție σ_1 este ***mai generală*** decât o substituție σ, dacă σ se poate obține prin compunerea lui σ_1 cu o altă substituție σ_2, adică: $\sigma = \sigma_2 \circ \sigma_1$.

Exemplu. σ_1 este mai generală decât σ (cf. exemplelor anterioare).

Definiție. (Cel mai general unificator). O substituție σ este (***un***) ***m.g.u.*** al termenilor t_1, $t_2 \in \mathcal{T}$ dacă:

1. σ este unificator pentru t_1, t_2.

2. σ este o substituție mai generală decât orice (alt) unificator al lui t_1, t_2.

Exemplu. (E mai greu de arătat proprietatea asta)

Substituția $\sigma_1 = \{z \mapsto h(z), z' \mapsto h(y)\}$ este (un) m.g.u. pentru $t_1 = f(x, h(y))$, $t_2 = f(h(z), z')$.

Teoremă. (de existență a m.g.u.). Oricare 2 termeni unificabili admit un m.g.u. (*ne-unic* de obicei).

<u>Exemplu.</u>

Un unificator pentru $t_1 = h(x)$ și $t_2 = h(y)$ este substituția $\sigma = \{x \mapsto a, y \mapsto a\}$, care însă nu este un m.g.u. pentru ei. În schimb $\sigma_1 = \{x \mapsto y\}$ este m.g.u.; ca de altfel și $\{y \mapsto x\}$; ca și ... $\{x \mapsto z; y \mapsto z\}$; ...

- Pentru a dezvolta un algoritm de calcul al unui m.g.u., este nevoie să generalizăm noțiunea de unificare în cazul mai multor perechi de termeni

Definiție. (Problemă de unificare). O ***problemă de unificare***, P, este de forma:

- fie, o *mulțime* $P = \{t_1 \doteq (t_1)', \ldots, t_n \doteq (t_n)'\}$ ($n \geq 1$)
- fie, $P = \perp$. (nu confundați P cu *predicatele*)

(t_i și $(t_j)'$ nu trebuie să fie neapărat distincți între ei).

Definiție. (Soluția unei probleme de unificare). O substituție σ este o ***soluție*** a unei probleme de unificare P, dacă:

1. Problema este $P = \{t_1 \doteq (t_1)', \ldots, t_n \doteq (t_n)'\}$ *și*

2. σ este unificator pentru *toate* ($i \in [n]$) perechile de termeni $(t_i, (t_i)')$.

Definiție. ***Mulțimea soluțiilor*** unei probleme de unificare este (evident) dată de:

$$unif(P) = \{\sigma \mid \sigma \text{ este soluție a lui } P\}.$$

- Prin definiție, punem *unif*($\perp$) = Ø

<u>Exemplu.</u> (dacă e nevidă, e infinită ...)

Fie $P = \{f(x, a) \doteq f(y, a)\}$. Atunci:

$unif(P) = \{\{x \mapsto z, y \mapsto z\}, \{x \mapsto y\}, \ldots\}$.

Definiție. (Cea mai generală soluție). Substituția σ este ***cea mai generală soluție*** pentru problema $P = \{t_1 \doteq (t_1)' \ldots, t_n \doteq (t_n)'\}$, dacă:

1. σ este soluție pentru P: $\sigma^\#(t_i) = \sigma^\#((t_i)')$, $i \in [n]$ <u>*și*</u>
2. σ este mai generală decât orice altă soluție.

- Dacă P are soluție, cu $mgu(P)$ se notează ***<u>o</u>*** cea mai generală soluție a sa
- Prin $mgu(t_1, t_2)$ vom nota cel mai general unificator al termenilor t_1, t_2, în cazul în care aceștia sunt unificabili
- Desigur că „punem”: $mgu(t_1, t_2) = mgu(\{t_1 \doteq t_2\})$

Definiție. (*forma rezolvată* a unei probleme de unificare). O problemă de unificare P este în ***formă rezolvată***, dacă fie $P = \bot$, fie problema este de forma $P = \{x_1 \doteq (t_1)', \ldots, x_n \doteq (t_n)'\}$, $n \geq 1$, cu $x_i \notin vars(t_j)$, pentru fiecare $i, j \in [n]$; iar x_i sunt ***toți diferiți***.

Lemă. Dacă $P = \{x_1 \doteq (t_1)', \ldots, x_n \doteq (t_n)'\}$ este în formă rezolvată, atunci cea mai generală soluție a lui P este $\sigma = \{x_1 \mapsto (t_1)', \ldots, x_n \mapsto (t_n)'\}$, $n \geq 1$.

- Următoarele reguli sunt folosite pentru „*a aduce o problemă de unificare la o formă rezolvată*”

(ȘT)ERGERE: $P \cup \{t \doteq t\} \Rightarrow P.$

(DE)SCOMPUNERE: $P \cup \{f(t_1, \ldots, t_n) \doteq f((t_1)', \ldots, (t_n)')\} \Rightarrow P \cup \{t_1 \doteq (t_1)', \ldots, t_n \doteq (t_n)'\}.$

(OR)IENTARE: $P \cup \{f(t_1, \ldots, t_n) \doteq x\} \Rightarrow P \cup \{x \doteq f(t_1, \ldots, t_n)\}.$

(EL)IMINARE: $P \cup \{x \doteq t\} \Rightarrow \sigma(P) \cup \{x \doteq t\}$,

doar dacă $x \notin vars(t)$, dar $x \in vars(P)$; $\sigma = \{x \mapsto t\}$.

(CO)NFLICT: $P \cup \{f(t_1, \ldots, t_n) \doteq g((t_1)', \ldots, (t_m)')\} \Rightarrow \perp$

(OC)CURS CHECK: $P \cup \{x \doteq f(t_1, \ldots, t_n)\} \Rightarrow \perp$, dacă $x \in vars(f(t_1, \ldots, t_n))$.

- Formal, *vars*(*P*) este ...
- Se pot demonstra următoarele rezultate

Lema 1. (*progres*). Dacă *P* nu este în formă rezolvată, atunci există P' astfel încât $P \Rightarrow P'$.

Lema 2. (*păstrarea soluțiilor = invariant*). Dacă avem $P \Rightarrow P'$, atunci avem *unif*(P) = *unif*(P').

Lema 3. (*terminare*). Nu există o secvență infinită de transformări $P \Rightarrow P_1 \Rightarrow P_2 \ldots \Rightarrow P_i \ldots$.

Corolar. Cu regulile precedente se poate construi un ***algoritm*** de calcul a unei cele mai generale soluții pentru o problemă de unificare *P*, dacă o asemenea soluție există (<u>explicații</u> …).

- <u>Exemplu</u> ...
- <u>Observație</u>: conceptele de compunere, unificator, unificator mai general, mgu, etc., folosesc *doar la* demonstrarea corectitudinii algoritmului

Observații. a)când se aplică regulile, numărul de perechi dintr-o problemă *P* nu rămâne constant (*n*); nici nu trebuie să coincidă cu aritatea vreunui simbol, etc.; b)pot exista secvențe infinite, dar formate dintr-o ***aceeași*** problemă, de la un loc ...

Exemplu.

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

$\sigma = \{x_3 \mapsto f(g(a, a), a), x_2 \mapsto a, x_1 \mapsto a\}$ este ***o*** cea mai generală soluție. (curs: pag.83-84)

Exemplu.

$P = \{f(g(x_1, a), x_2) \doteq x_3, f'(x_2) \doteq f'(x_3)\}$.

$unif(P) = \emptyset$. (tot acolo)

Exemplu.

$P = \{f(g(x_1, a), x_2) \doteq x_3, f'(g(x_4, x_5)) \doteq f'(x_3)\}$.

$unif(P) = \emptyset$. (tot acolo)

3.Rezoluția pură

- După cum am mai precizat, rezoluția specifică limbajului **LP1** poate fi prezentată printr-un sistem deductiv având două (scheme de) reguli de inferență, numite REZOLUȚIE BINARĂ (regula (I)) și, respectiv, FACTORIZARE POZITIVĂ (regula (II))

$$(\mathrm{I})\quad \frac{C_1 \vee Q(t_1, \ldots, t_n) \qquad C_2 \vee \rceil Q((t_1)', \ldots, (t_n)')}{(\sigma)^b(C_1 \vee C_2)}\quad (\boldsymbol{c})$$

- Mai sus, (***c***) este dată prin:

(a) $V_1 \cap V_2 = \emptyset$

(b) $\sigma = mgu(\{t_1 \doteq (t_1)' \ldots, t_n \doteq (t_n)'\})$

(c) $V_1 = vars(C_1 \vee Q(t_1, \ldots, t_n))$, (*definiție vars* - ...)

$V_2 = vars(C_2 \vee \rceil Q((t_1)', \ldots, (t_n)'))$.

Și:

$$\text{(II)}\ \frac{C \vee Q(t_1, \ldots, t_n) \vee Q((t_1)', \ldots, (t_n)')}{(\sigma)^b(C \vee Q(t_1, \ldots, t_n))}\ (\boldsymbol{c}')$$

- Unde, pentru (***c***') avem doar restricția:

$$\sigma = mgu(\{t_1 \doteq (t_1)', \ldots, t_n \doteq (t_n)'\})$$

(Putem „pune” însă și o *factorizare negativă* ...)

Observații:

1. Dacă clauzele din ipotezele regulii (I) au variabile în comun ($V_1 \cap V_2 \neq \emptyset$), atunci toate variabilele în cauză trebuie redenumite înainte a aplica regula (nu uităm că <u>acestea sunt de fapt „cuantificate universal *în față*”</u>)
2. În cazul în care problema de unificare care apare în regula aleasă spre aplicare nu are soluție, acea regulă nu poate fi (de fapt) aplicată

Final (și pentru **Seminar 6)**