CURS 1

- Sunt, în anul universitar 2012-2013, semestrul I, 16 săptămâni de şcoală; săptămânile a 8-a şi a 16-a sunt destinate lucrărilor de evaluare; vom face evaluare doar în săptămâna a 16-a; în săptămâna a 8-a vom încerca să recuperăm cursul pierdut pe 1 octombrie (n-am reuşit…)
- Alte recuperări vor fi (re)anunțate pe parcurs
- În consecință, vom avea 14 cursuri şi 14 seminarii
- INFORMAȚI-VĂ LA TIMP (paginile web ale celor cu care lucrați)

DE CE LOGICA ?

- Introducere; explicații

LOGICA PENTRU INFORMATICĂ

Facultatea de Informatică

Universitatea „Al.I.Cuza” Iaşi

(http://www.info.uaic.ro)

Profesori 1

- Prof. Dr. Cristian Masalagiu (Titular curs)
  mcristy@info.uaic.ro

  http://profs.info.uaic.ro/~masalagiu

Profesori 2

- Asist. Drd. Cosmin Vârlan (seminar)

  vcosmin@info.uaic.ro

  http://profs.info.uaic.ro/~vcosmin

Profesori 3

- Colab. Dr. Vasile Alaiba (seminar)

  alaiba@info.uaic.ro

  http://profs.info.uaic.ro/~alaiba

Profesori 4

- Colab. Drd. Marius Barat (seminar)
  marius.barat@info.uaic.ro
  http://students.info.uaic.ro/~marius.barat

Orar 1
(http://www.info.uaic.ro/~orar):

- Curs:

  Luni 10.00 - 12.00 semianul B (C2)

  12.00 - 14.00 semianul A (C2)

Orar 2 (http://www.info.uaic.ro/~orar)

- Seminarii (ştiți cu cine, cf. ORAR):

  Marti:

  14.00 – 16.00 : A4 (C903)

  16.00 – 18.00 : A3 (C903)

  18.00 – 20.00 : A2 (C903)

  Miercuri:

  14.00 – 16.00 : A1 (C903)

  16.00 – 18.00 : B2 (C903)

  18.00 – 20.00 : B7 (C909)

  18.00 – 20.00 : B1 (C903)

Orar 3 (http://www.info.uaic.ro/~orar)

- Seminarii:
  - Joi:
    - 08.00 – 10.00 : B6 (C909)
    - 10.00 – 12.00 : A7 (C909)
    - 12.00 – 14.00 : B5 (C903)
    - 14.00 – 16.00 : A5 (C905)
    - 16.00 – 18.00 : A6 (C905)
    - 16.00 – 18.00 : B3 (C903)
    - 18.00 – 20.00 : B4 (C903)
    - 18.00 – 20.00 : X (C905)

Orar 4 (http://www.info.uaic.ro/~orar)

- Ore de consultații (anunțați în prealabil):
  - C.Masalagiu, V.Alaiba, M.Barat:

    Vineri: 08.00 – 10.00

    Cabinet – C305
  - C.Vârlan:

    Joi: 14.00 – 16.00

    Cabinet – C211

Observație:

1. Orarul se poate modifica în timp; verificați *săptămânal* pe pagina Facultății
2. Se poate veni la consultații şi Vineri, ora 12.00 în C305

Cerințe 1

http://www.info.uaic.ro -> Membri

-> Personal Academic (pentru a cunoaşte şi alți profesori şi a naviga către paginile noastre)

http://www.info.uaic.ro -> Studenți

-> Regulamente

http://profs.info.uaic.ro/~masalagiu/

-> Secțiunea „Logic”

Cerințe 2

- Nota finală se obține în urma acumulării unui număr de puncte (maxim 100) şi în urma aplicării unei ajustări de tip Gauss (conform regulamentului în vigoare)

Cerințe 3

- Cele **100 de puncte** se pot obține în urma activității din timpul semestrului:
  - **10 p** prezența la seminarii (4 verificări a prezenței, realizate aleatoriu, a câte 2,5 p)
  - **40 p** pentru rezolvarea de exerciții (ieşiri la tablă) şi participarea la lucrări (neanunțate?)
  - **50 p** pentru lucrarea de verificare a cunoştințelor de la finalul semestrului

Cerințe 4

- Promovarea este asigurată de un punctaj minim total de **50 p**
- La lucrarea finală este necesară obținerea a minim **25 p**
- A se consulta (deja menționat) măcar pagina http://profs.info.uaic.ro/~masalagiu („Logic”)

Cerințe 5

- Bibliografie „scrisă” minimală:

Masalagiu, C. - ***Fundamentele logice ale Informaticii*** , Editura Universității „Al. I. Cuza”, Iaşi, 2004, ISBN 973-703-015-X

Masalagiu, C. - ***Introducere în programarea logică şi limbajele de programare logică***, Editura Universității „Al. I. Cuza”, Iaşi, 1996

Cazacu, C., Slabu, V. - ***Logica matematica*** , Editura Ştefan Lupascu, Iaşi, 1999, ISBN 973-99044-0-8

Cerințe 6

- Bibliografie principală (din nou…):

Masalagiu, C. - ***Fundamentele logice ale Informaticii*** , Editura Universitatii „Al. I. Cuza", Iasi, 2004, ISBN 973-703-015-X.

http://profs.info.uaic.ro/~masalagiu
(În secțiunea „Logic” - *Bibliografie de bază*; link-urile pot fi accesate doar din cadrul **FII**)

Sugestie: Învățați **<u>şi</u>** după carte, nu numai după slide-uri !!!

Capitolele tematice

(Unele nu vor fi prezentate chiar în această ordine)

- Logica în Informatică
- Algebre booleene
- Logica propoziţională
- Logica cu predicate de ordinul I
- Sisteme deductive şi Teorii logice
- Introducere în Programarea logică
- **(O)BDD**-uri şi verificare

Realitate

- **Realitate: obiecte** şi **fenomene** aflate în **relații de interdependență**
- **Relațiile** (legăturile) se reflectă, în procesul gândirii umane, prin **afirmații** (judecăți)

Logica (intuitiv) 1

- **Logica (în sens filozofic)** este *ştiința regulilor generale ale gândirii, cu accent pe aspectele exacte şi de natură structurală ale acesteia*
- Alte „definiții” – după cum urmează

Logica (intuitiv) 2

- **Logica** studiază *modul de alcătuire a* ***raționamentelor*** *corecte*, prin care, pornind de la *afirmații inițiale* (presupuse a fi *adevărate*) se obțin (utilizând *reguli de inferență*) *afirmații noi* (de asemenea *adevărate*)

Afirmații

- O **afirmație** este *adevărată* dacă *reflectă în mod adecvat realitatea* şi *falsă* în caz contrar
- Este acceptat faptul că valorile de adevăr ataşate unei afirmații pot avea o natură subiectivă şi, în consecință, pot fi şi altceva decât cele *standard* (*adevărat* şi *fals*); dar asta…în „alte logici"
- De exemplu: *necunoscut*, *posibil adevărat*, *adevărat din când în când*, etc.

Logica clasică

- **Logica clasică (aristotelică, bivalentă)** foloseşte doar afirmații cărora li se poate asocia în mod *unic* o valoare de adevăr *standard* (independentă de context, moment de timp, etc.) şi se bazează în esență pe principiul *tertium non datur* (terțiul exclus: *dacă o afirmație nu este adevărată, atunci ea este cu certitudine falsă şi reciproc*)
- Un alt principiu este cel al *extensionalității* (adevărul unei formule…)
- Există *şi logici neclasice* (exemple)

Idei suplimentare 1

- **Logica matematică, formală (simbolică)**, preia problemele logicii filozofice şi le cercetează *folosind mijloace matematice specifice*, *punându-se bază pe rigurozitate şi claritate în detrimentul nuanțelor sau intuiției*

Idei suplimentare 2

- Aplicarea acesteia în **Informatică** necesită însă o anumită adaptare, atât a modului de prezentare a conceptelor (terminologiei) cât şi a metodelor de demonstrație, accentul căzând pe ***constructivism*** (***algoritmică***)

Despre CURS

- Structura Cursului; sugestii privind examinarea
- Dificultăți de învățare (sunt şi avantaje…)
- *Paradoxuri*, greşeli frecvente, exemple
- Sferă, conținut, gen proxim, diferență specifică
- Silogisme, axiome, teoreme, reguli de inferență, raționamente, exemple
- Axiome *adevărate* şi reguli de inferență

  corecte (*sound*) – pot exista raționamente corecte, dar dacă se „pleacă” cu premize false…(vezi şi „paradoxuri”)
- Sintaxă + semantică – Logica este un limbaj de programare: formulă + valoare de adevăr

  (execuție = evaluare)
- Nr. de pagină s-ar putea să nu coincidă cu ceea ce aveți voi pe site (mă refer la cartea publicată/tipărită)

Mulțimi 1

- Clasa funcțiilor booleene (intuitiv)
- Notații
- Din manualele de matematică de liceu sunt bine cunoscute cel puțin două modalități de a *prezenta* o mulțime:

  -Prin enumerarea elementelor sale: **N** = {0, 1, 2, ...} este mulțimea numerelor naturale

Mulțimi 2

-Prin specificarea unei proprietăți caracteristice:

A = $\{x \in \mathbf{R} \mid x^2 + 9x - 8 = 0\}$,
este mulțimea rădăcinilor reale ale unei ecuații polinomiale de gradul al II-lea

-Mai avem o posibilitate, bazată pe *constructivism*:

Mulțimi 3 (Peano)

- **Baza**. $0 \in$ **N** (*zero* este număr natural)
- **Pas constructiv** (**structural**). Dacă $n \in$ **N**, atunci $s(n) \in$ **N** (dacă n este număr natural, atunci *succesorul său imediat* este număr natural)
- **Nimic altceva nu mai este număr natural**

Mulțimi 4

- *Un prim* avantaj este acela că se poate folosi aceeaşi metodă pentru a introduce alte definiții, care sunt legate de mulțimea respectivă în totalitatea ei
- Putem da astfel o definiție constructivă (recursivă) a adunării numerelor naturale:

  -Baza. $x + 0 = x$, pentru fiecare $x \in$ **N** (a aduna 0 la orice număr natural înseamnă a-l lăsa neschimbat)

  -Pas constructiv. $x + s(y) = s(x + y)$, pentru fiecare

  $x, y \in$ **N** (dacă ştim să calculăm $x + y$ şi cunoaştem succesorul imediat al numărului natural y, atunci ştim să calculăm şi suma $x + s(y)$; mai exact, aceasta coincide cu succesorul imediat al numărului care reprezintă suma $x + y$)

  -Nimic altceva…

Mulţimi 5

- *Un al doilea*, şi cel mai important, avantaj este posibilitatea folosirii în demonstraţii a **Principiului inducţiei structurale**:

  -Fie **A** o mulţime definită constructiv,

  A' $\subseteq$ **A** mulţimea **elementelor iniţiale** (definite prin pasul **Baza** al definiţiei) şi **P** o „afirmaţie" care trebuie demonstrată pentru toate elementele lui **A**

  -*Acceptăm* că **P**(a) este adevărată pentru fiecare a $\in$ **A** dacă şi numai dacă:

Mulțimi 6

1) (Baza – elemente inițiale**)**: *Arătăm* că **P**(a) este adevărată pentru fiecare a $\in$ **A'**

2) (Pas inductiv – elemente noi din elemente vechi**)**:

-Fie orice b $\in$ **A**, element nou obținut din elementele deja „construite" $a_1, a_2, \ldots, a_n$, cu ajutorul „operatorului" f (vom conveni să scriem acest lucru prin

$b = f(a_1, a_2, \ldots, a_n)$, deşi relația dintre elementele „vechi" şi cele „noi" nu este întotdeauna de natură funcțională) şi presupunem că este adevărată **P**(a_i) pentru fiecare

$i \in \{1, 2, ..., n\}$

-*Arătăm* că este adevărată **P**(b)

- Probleme suplimentare seminar (nu sunt obligatorii; vă ghidează asistenții; mai există şi o „Culegere de probleme noi", în pregătire…):
  - http://profs.info.uaic.ro/~masalagiu/pub/seminar%20logica%201-3.pdf

CURS 2

- Începem partea consistentă, formală, a cursului

Algebre booleene 1

- Se numeşte **algebră booleană** un 4-uplu **M**, **M** = <**M**, $\perp$, ∇, ~ >, format din orice mulţime nevidă **M** (*suportul* algebrei) două operaţii binare $\perp$, ∇ : **M** $\times$ **M** $\rightarrow$ **M** şi o operaţie unară ~ : **M** $\rightarrow$ **M**, care satisfac condiţiile (*legile*):

1) $x \perp y = y \perp x$ — *comutativitate* (a lui $\perp$)

2) $(x \perp y) \perp z = x \perp (y \perp z)$ — *asociativitate* (a lui $\perp$)

3) $x \perp (y \nabla z) = (x \perp y) \nabla (x \perp z)$ — *distributivitate* (a lui $\perp$ faţă de ∇)

4) $(x \perp y) \nabla y = y$ — *absorbţie*

5) $(x \perp (\sim x)) \nabla y = y$ — *legea contradicţiei*

Algebre booleene 2

şi respectiv

1') $x \nabla y = y \nabla x$ *comutativitate* (a lui ∇)

2') $(x \nabla y) \nabla z = x \nabla (y \nabla z)$ *asociativitate* (a lui ∇)

3') $x \nabla (y \perp z) = (x \nabla y) \perp (x \nabla z)$ *distributivitate* (a lui ∇ față de $\perp$)

4') $(x \nabla y) \perp y = y$ *absorbție*

5') $(x \nabla (\sim x)) \perp y = y$ *legea tautologiei*

Algebre booleene 3

- Dualitate; **principiul** dualităţii
- Notaţii (reamintit: **B**, 2^V; $\perp$ - **•**, ∩; ∇ - +, U; ~ - $\overline{\ }$,**C**$_V$)
- Reprezentare (tabele „de adevăr”, standard)
- Funcţii importante (0, 1, 2 argumente): **0**, **1**, c_0, c_1, 1_B, $\overline{\ }$, +, **•**,⊕, | .
- Numărul de funcţii din **FB**-uri
- Reprezentarea „numerică” a tabelelor „standard”
- Alte considerente algebrice (latici…)
- Alte „legi” (**Tabelul 1.1** – pag.30 – cartea tipărită)

Algebre booleene 4

- Reprezentarea funcțiilor booleene
- Forme normale

Reprezentarea funcțiilor booleene 1

- Notații: $x^1 = x$ şi $x^0 = \overline{x}$
- Indicii (superiori) precedenți nu se supun principiului dualității (de exemplu, nu este adevărat că ($x^1 = x$) coincide cu ($x^0 = x$))
- Dacă x_i, $\alpha_i \in$ **B** atunci, direct din notațiile de mai sus, rezultă că:
- $\mathbf{(x^0)^\alpha = (x^\alpha)^0}$ precum şi ($\mathbf{x^\alpha = 1}$ dacă şi numai dacă $\mathbf{x = \alpha}$)

Reprezentarea funcțiilor booleene 2

- **Teoremă** *(de descompunere, în sumă de „termeni”)*. Pentru fiecare n ∈ **N***, f ∈ **FB**$^{(n)}$ şi fiecare k ∈ [n], avem:

$$f(x_1, x_2, ..., x_n) =$$

$$\sum_{\alpha_1, \alpha_2, ..., \alpha_k \in \mathrm{B}} x_1^{\alpha_1} \bullet x_2^{\alpha_2} \bullet ... \bullet x_k^{\alpha_k} \bullet f(\alpha_1, \alpha_2, ..., \alpha_k, x_{k+1}, ..., x_n)$$

oricare ar fi x_1, x_2, ... , x_n ∈ **B** (demonstrație)

- Enunțați teorema duală

Reprezentarea funcțiilor booleene 3

- **Definiție**. Fie $n \in$ **N*** şi $x_1, x_2, \ldots, x_n \in$ **B** variabile (booleene) distincte. Notăm mulțimea (lista!) acestora cu

  $X = \{x_1, x_2, \ldots, x_n\}$. Se numeşte ***termen (n-ar, peste X)*** orice produs

$$t = x_{i_1}^{\alpha_1} \bullet x_{i_2}^{\alpha_2} \bullet \ldots \bullet x_{i_k}^{\alpha_k}$$

  unde $0 \leq k \leq n$, $\alpha_1, \alpha_2, \ldots, \alpha_k \in$ **B** şi

  $1 \leq i_1 < i_2 < \ldots < i_k \leq n$

Reprezentarea funcțiilor booleene 4

- Termenul generat pentru k=0 este 1 (prin convenție). Pentru k = n obținem aşa-numiții *termeni maximali* ***(maxtermeni)***, adică *acei termeni în care fiecare dintre variabilele considerate apare o dată şi numai o dată (barată sau nebarată), în ordinea precizată*
- Exemple
- Numărul de termeni şi maxtermeni distincți
- Dual: factori şi maxfactori

Forme normale 1

- **Definiție**.

  -Se numeşte *formă normală disjunctivă (n-ară*, n $\in$ **N***), sau (n-)**FND** pe scurt, orice sumă (finită) de termeni n-ari distincți

  -Se numeşte *formă normală disjunctivă perfectă (n-ară*, n $\in$ **N***), sau (n-)**FNDP** pe scurt, orice sumă de maxtermeni n-ari distincți
- Facem abstracție de ordinea (max)termenilor dintr-o sumă, ***mai exact, considerând oricare două sume care diferă doar prin ordinea termenilor, le vom privi ca fiind identice***

  - Vor exista astfel forme normale disjunctive n-are având k termeni, $0 \leq k \leq 3^n$ (prin convenție, pentru k = 0, unica formă care este acceptată şi este şi perfectă, se notează tot cu 0)

  -Care va fi numărul total al (n -)**FND** – urilor? Dar cel al (n-)**FNDP**–urilor ?

Forme normale 2

- **Teoremă**. Orice funcție booleană f se poate „reprezenta" în mod unic ca o **FNDP**:

$$f(x_1, x_2, ..., x_n) = \sum_{\alpha_1, \alpha_2, ..., \alpha_n \in \mathrm{B}} x_1^{\alpha_1} \bullet x_2^{\alpha_2} \bullet ... \bullet x_n^{\alpha_n} \bullet f(\alpha_1, \alpha_2, ..., \alpha_n)$$

  (demonstrație)

- Mai spunem că expresia din membrul drept al reprezentării *este* ***(o) FND(P) pentru f***

Forme normale 3

- Prin dualizare, folosind noțiunile de ***(n-)factor peste X*** şi ***maxfactor (n-ar, peste X)***, putem defini noțiunea de *formă normală conjunctivă (n-ară)* ((n-)**FNC**, adică orice produs de factori dictincți) şi respectiv *formă normală conjunctivă (n-ară) perfectă* ((n-)**FNCP**, adică orice produs de maxfactori distincți)
- Convenție: două produse nu vor fi considerate distincte dacă diferă doar prin ordinea componentelor
- Enunțați duala teoremei anterioare, pentru **FNCP**
- Care va fi numărul total al (n -)**FNC** – urilor? Dar cel al (n-)**FNCP**–urilor ?

Forme normale 4

- Vom continua cu o modalitate generală de a construi întreaga clasă a funcțiilor booleene

Clase speciale de funcții booleene 1

- Criteriul „numărul total de *apariții* ale variabilelor în reprezentarea unei funcții” (apariția unei aceleiaşi variabile pe *poziții* diferite se numără distinct) poate genera alte forme normale
- Folosind această „măsură” (pe care o vom nota cu n(φ)), putem numi *formă normală disjunctivă minimală* (**FNDM**) pentru f $\in$ **FB** orice **FND**, φ’, astfel încât:

  n(φ’) = *min* {n(φ) | φ este **FND** pentru f}
- Dată o funcție booleană f $\in$ **FB**, se poate pune **problema determinării *tuturor* FNDM pentru f**, sau a uneia *standard* (*algoritmul lui Quine*)

Clase speciale de funcții booleene 2

- Similar, putem reduce în anumite cazuri timpul de procesare a unor texte (expresii, formule etc.) prin **găsirea unui număr minim de operații booleene *convenabile***, cu ajutorul cărora să se „reprezinte” **orice** funcție booleană

Clase speciale de funcții booleene 3

- **Definiție**.

  -Clasa *funcțiilor booleene* ***elementare*** este:

  $\mathbf{E} = \{\ i_p^n \mid n \in \mathbf{N^*},\ 1 \le p \le\ n,\ i_p^n: \mathbf{B}^n \to \mathbf{B},$

  $i_p^n(x_1, x_2, \ldots , x_p, \ldots , x_n) = x_p\}$

  (proiecții)

  -Fie $n \in \mathbf{N^*}$, t un număr natural,

  $f, h_1, h_2, \ldots , h_t \in \mathbf{FB}^{(n)}$ şi $g \in \mathbf{FB}^{(t)}$

Clase speciale de funcții booleene 3

Spunem că *f se obține din g, h_1, h_2, ... , h_t prin* ***superpoziție*** dacă pentru fiecare

$x = \langle x_1, x_2, \ldots , x_n \rangle$ avem:

$f(x) = g(\langle h_1(x), h_2(x), \ldots , h_t(x) \rangle)$

-Fie M $\subseteq$ **FB**. Se numeşte *M-şir* orice *secvență (listă) finită* $f_0, f_1, \ldots , f_r$ de funcții booleene în care fiecare f_i este fie din **E** U M, fie se obține prin superpoziție din alte funcții, aflate în aceeaşi listă dar înaintea lui f_i

Clase speciale de funcții booleene 4

- Alte notații …
- *Mulțimi închise*; *închideri*
- Definiții alternative
- Rezultate importante
- Exemple: $\mathbf{T_0}$, $\mathbf{T_1}$, **Aut**, **Mon**, **Lin**
- *Mulțimi complete*, *precomplete*, *baze*
- Alte rezultate şi exemple

CURS 3

- Vom discuta despre *decidabilitate* şi despre un alt mod de reprezentare a funcțiilor booleene, şi anume *diagramele de decizie binare* (eventual…*orientate*)

Decidabilitate în **FB**

- **Teoremă (decidabilitatea SAT)**. „Satisfiabilitatea” („validitatea”, „nesatisfiabilitatea”) funcțiilor booleene este decidabilă în *timp exponențial* (demonstrație)
- Alte comentarii, etc.

(O)BDD - 1

- Ştim ce înseamnă funcţii booleene şi reprezentarea lor cu ajutorul tabelelor de adevăr sau cu expresii (**FNC(P)**, de exemplu)
- O altă reprezentare a elementelor din **FB** se bazează pe *diagramele de decizie binare* (**BDD**)
- Alegerea celei mai „convenabile” reprezentări depinde de context
- Tot ceea ce putem menţiona în acest moment este că în anumite cazuri o **BDD** poate fi mai „compactă” decât o tabelă de adevăr pentru o aceeaşi funcţie (din cauza anumitor redundanţe care pot fi exploatate mai uşor)

(O)BDD - 2

- **Definiție (B**inary **D**ecision **D**iagram**).** Se numeşte *diagramă de decizie binară* (**BDD** pe scurt) *peste*

  $X = \{x_1, x_2, \dots , x_n\}$ un graf *orientat*, *aciclic*, *etichetat* (pe noduri şi pe arce) în care:

  -există o unică *rădăcină* (nod în care nu intră nici un arc)

  -*frunzele* (nodurile din care nu iese nici un arc) sunt etichetate cu 0 sau 1, iar celelalte noduri (inclusiv rădăcina) sunt etichetate cu elemente din X (se permit etichetări multiple, adică noduri diferite pot avea aceeaşi etichetă); ideea este şi ca fiecare x_i să fie folosit *măcar o dată*; cerința nu este însă mereu obligatorie…

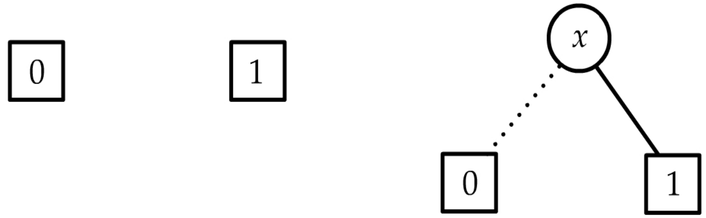
  -fiecare nod care nu este frunză are exact doi succesori imediați, arcele care îi leagă fiind etichetate cu 0 respectiv 1
- O **subBDD** (într-o **BDD** dată) este un *subgraf generat* de un nod fixat împreună cu toți succesorii săi

(O)BDD - 3

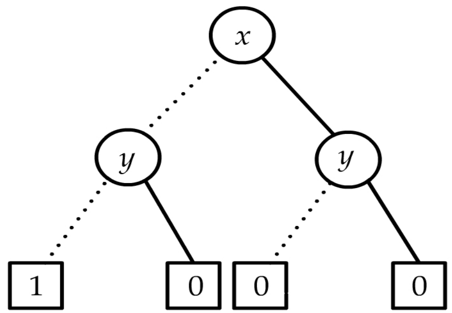
- De obicei, într-un „desen” care reprezintă o **BDD**, frunzele pot fi identificate (şi) prin pătrate (nu cercuri, ca restul nodurilor), orientarea arcelor este implicită („de sus în jos”), arcele etichetate 0 sunt reprezentate prin linii punctate („stânga”), iar cele etichetate 1 sunt linii continue („dreapta”)
- În primele exemple care urmează, grafurile sunt chiar arbori

(O)BDD - 4

- (I) **D0**, **D1** (peste ∅) şi **Dx** (peste X = {x}):



- (II) O **BDD** peste X = {x, y}



(O)BDD - 5

- **Observație.** Orice **BDD** peste $X = \{x_1, x_2, \ldots, x_n\}$ definește/reprezintă/calculează o **unică** funcție booleană $f \in$ $\mathbf{FB}^{(n)}$
- Astfel, pentru $\alpha_1, \alpha_2, \ldots, \alpha_n \in$ **B** (considerate ca fiind valori asignate corespunzător „variabilelor” din X) se „pornește” cu rădăcina (unică) și se „parcurge” un drum (unic) în graf „până” la o frunză (să spunem, etichetată cu $\beta \in$ **B**)
- La fiecare pas, plecând din nodul curent, se alege acel arc (prin urmare și noul nod curent) căruia i se atașează valoarea 0 sau 1 conform valorii α deja atribuite nodului curent x (în asignarea aleasă)
- Valoarea asignată lui β este chiar $f(<\alpha_1, \alpha_2, \ldots, \alpha_n>)$

(O)BDD - 6

- În acest mod, **BDD**-ul din exemplul anterior reprezintă funcția

$$f(x, y) = \overline{x} \cdot \overline{y}$$

- Este clar că putem proceda şi invers, adică pornind cu orice funcție f $\in$ **FB**$^{(n)}$ dată printr-un tabel de adevăr, construim (măcar) un arbore (**BDD**) binar, complet şi având n *nivele*, notate

  0 - rădăcina, 1, … , n-1 – frunzele(alternativ, având *adâncimea* n) care „calculează” f, în modul următor:

(O)BDD - 7

- Se ordonează mulțimea de variabile cu ajutorul căreia este exprimată funcția, să zicem $X = \{x_1, x_2, \dots , x_n\}$, sub forma $<x_{k,1}, x_{k,2}, \dots , x_{k,n}>$, $<k,1; k,2; \dots ; k,n>$ fiind o permutare pentru $<1, 2, \dots , n>$
- Nodurile interioare (care nu sunt frunze) situate pe nivelul i -1 sunt etichetate (**toate**) cu $x_{k,i}$ ($i \in [n]$); rădăcina este etichetată cu $x_{k,1}$ fiind (singurul nod de) pe nivelul 0
- Cele două arce care ies din fiecare nod sunt etichetate (normal) cu 0 şi respectiv 1
- Frunzele sunt etichetate cu 0 sau 1 conform tabelei de adevăr pentru f (drumul de la rădăcină la frunza corespunzătoare furnizează exact linia care trebuie aleasă din tabelă: eticheta fiecărui arc de pe drum reprezintă valoarea atribuită variabilei care este eticheta nodului din care iese arcul)

(O)BDD - 8

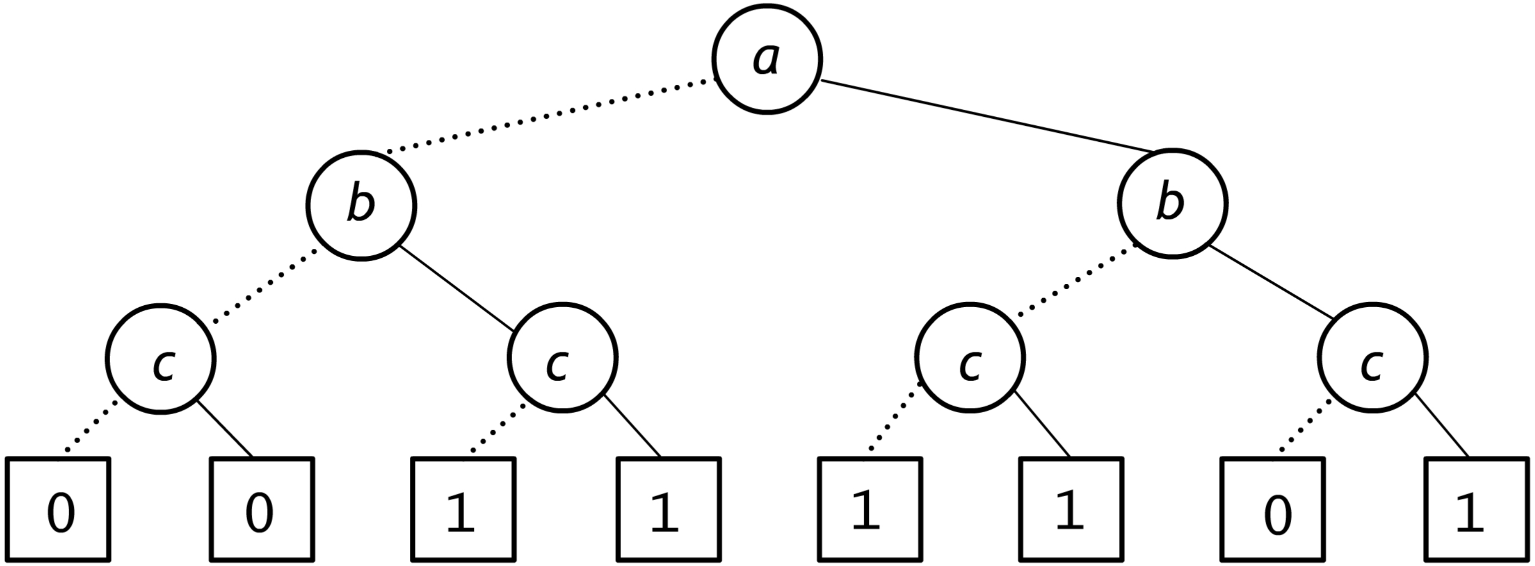
- **Exemplu.** Fie $f \in$ **FB**$^{(3)}$ dată prin

  $$f(a,b,c) = (a+b)\cdot(\overline{a}+\overline{b}+c)$$

  deci exprimată cu ajutorul lui X = {a, b, c},

  <a, b, c> fiind şi ordinea impusă asupra variabilelor. Tabela sa de adevăr este (de făcut voi…)
- **BDD**-ul care calculează f după algoritmul sugerat anterior este (de verificat…):
- **Observație**. De multe ori nu vom face o distincție explicită între funcție booleană şi formulă **LP** (conform cursurilor ulterioare…)

(O)BDD - 9



(O)BDD - 10

- După cum se observă imediat, construirea unui **BDD** care calculează o funcție dată nu este un proces cu rezultat unic (spre deosebire de procesul „invers”), fiind suficient să schimbăm ordinea variabilelor
- Impunerea unei ordini asupra etichetelor nodurilor este însă şi un prim pas spre găsirea unor *forme normale* pentru **BDD**-uri
- Un alt aspect care trebuie avut în vedere pentru atingerea acestui scop este acela că reprezentarea ca arbore a unei **BDD** nu este deloc mai eficientă/compactă decât o tabelă de adevăr (nici decât, de exemplu, o **FNC(P)**): dacă f $\in$ **FB**$^{(n)}$, atunci tabela sa de adevăr va avea 2^n linii iar în reprezentarea **BDD** sugerată de noi (ca arbore, în care fiecare nivel este „destinat” unei variabile şi pe fiecare drum de la rădăcină la o frunză apar toate variabilele exact o dată) vor fi exact $2^{n+1} – 1$ noduri
- Putem compacta o **BDD** dacă îi aplicăm următoarele *procedee de reducere/optimizare* (în cele de mai jos, când ne referim la nodul **n**, **m**, etc. nu ne referim la

(O)BDD - 11

C1 (înlăturarea frunzelor duplicat). Dacă o **BDD** conține mai mult de o frunză etichetată cu 0, atunci:

-păstrăm una dintre ele

-ştergem celelalte frunze etichetate cu 0, împreună cu arcele aferente, care de fapt se „redirecționează” spre singura 0-frunză rămasă (fiecare păstrându-şi nodul sursă)

-se procedează în mod similar cu 1-frunzele; admitem şi înlăturarea unei frunze dacă, în final, ea nu are nici un arc incident cu ea

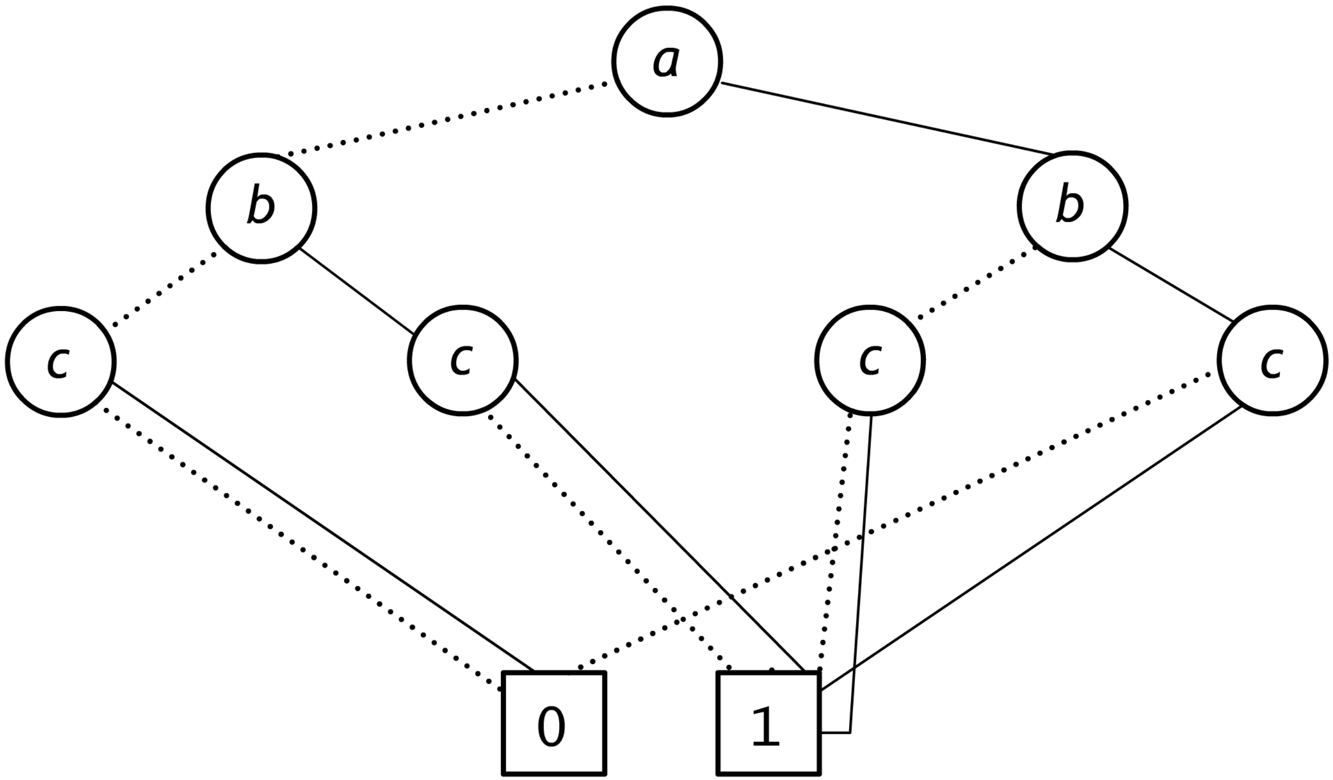
C2 (eliminarea „testelor” redundante). Dacă în **BDD** există un nod (interior) $\mathbf{n}$ pentru care atât 0-arcul cât şi 1-arcul au ca destinație acelaşi nod $\mathbf{m}$ (lucru care se poate întâmpla doar dacă s-a efectuat măcar un pas de tip **C1**), atunci nodul $\mathbf{n}$ se elimină (împreună cu arcele sale care punctează spre $\mathbf{m}$), iar arcele care înainte punctau spre $\mathbf{n}$ sunt „redirecționate” spre $\mathbf{m}$

C3 (eliminarea nodurilor duplicat care nu sunt frunze). Dacă în **BDD** există două noduri interioare distincte, să zicem $\mathbf{n}$ şi $\mathbf{m}$, care sunt rădăcinile a două **subBDD**-uri identice (fiind identice, $\mathbf{n}$ şi $\mathbf{m}$ sunt şi pe acelaşi nivel), atunci se elimină unul dintre ele, să zicem $\mathbf{m}$ (împreună cu arcele care-l au ca sursă), iar arcele care punctau spre $\mathbf{m}$ se „redirecționează” spre $\mathbf{n}$

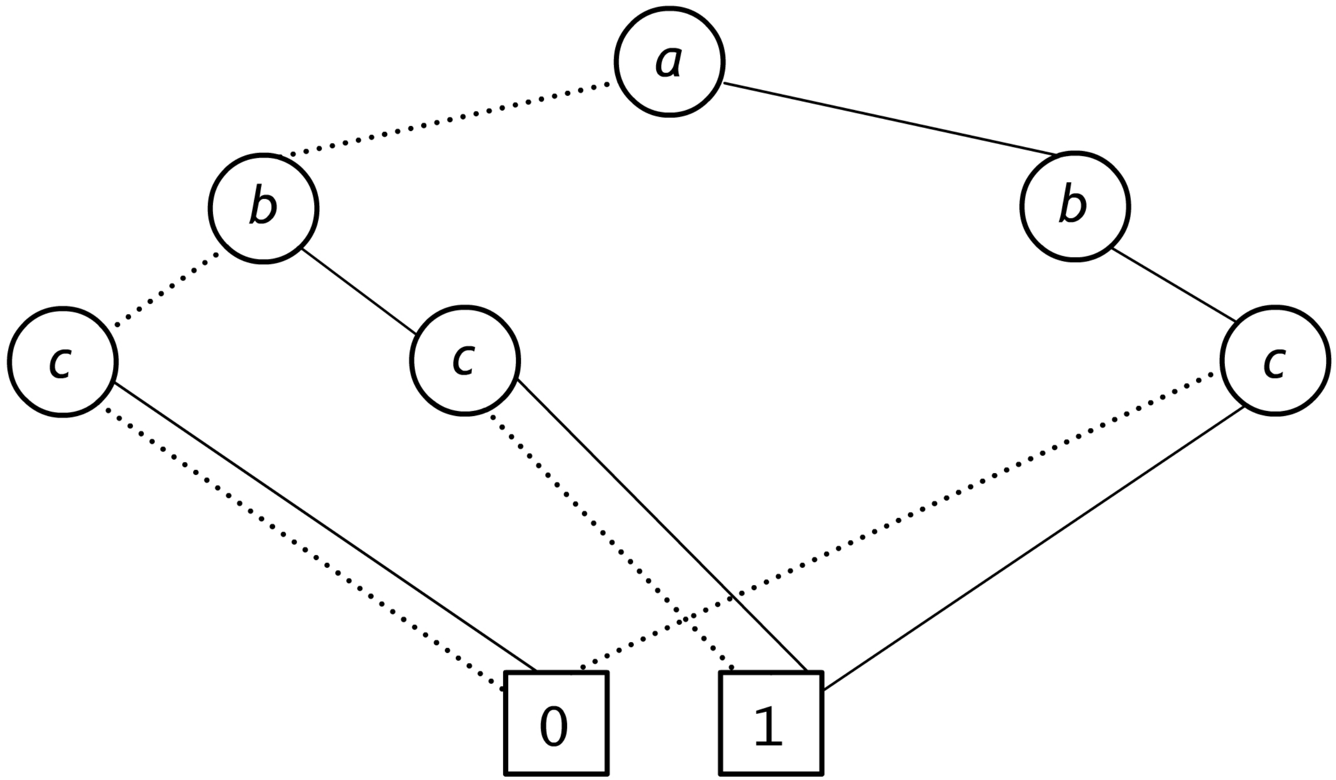
(O)BDD - 12

- **Exemplu.** Plecând de la **BDD**-ul din ultimul exemplu obținem succesiv (mai întâi sunt transformări de tip **C1**, apoi de tip **C3** şi, în sfârşit, de tip **C2**)

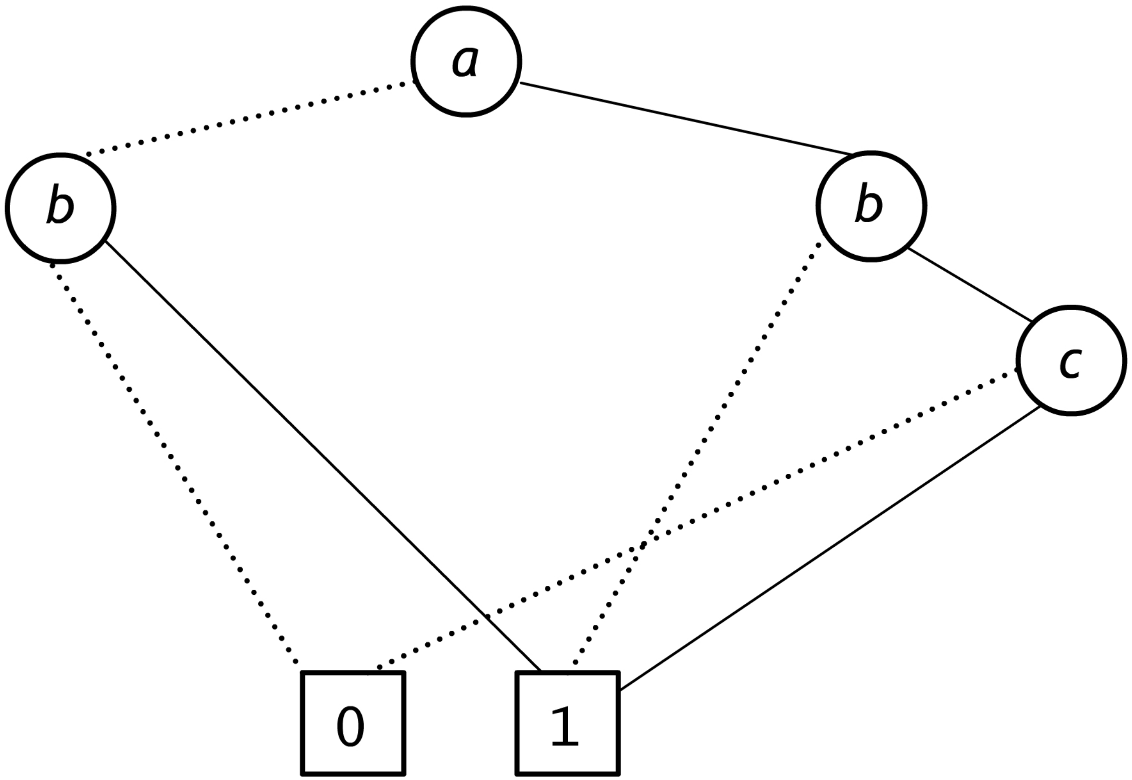
(O)BDD - 13



(O)BDD - 14



(O)BDD - 15



(O)BDD - 16

- Ultima **BDD** este *maximal redusă* (nu se mai pot face alte transformări)
- Desigur că ordinea în care se efectuează eliminările de tipul **C1**-**C3** poate influența aspectul structural al unei **BDD** şi pot exista mai multe transformări distincte care să fie simultan admise pentru o aceeaşi structuri
- În schimb, nici o transformare **nu modifică** funcția booleană calculată

(O)BDD - 17

- Concluzionând, **BDD**-urile pot fi uneori „convenabil de compacte”
- Mai mult, putem reformula anumite definiții ale funcțiilor booleene pentru noua reprezentare, căpătând, de exemplu, idei noi pentru rezolvarea problemei **SAT**
- **Diagramele de decizie binare ordonate vor fi studiate opțional, deocamdată**

(O)BDD - 18

- **Definiție (drumuri consistente şi satisfiabilitate).** Fie o **BDD** D peste mulțimea de variabile X, care calculează o funcție f $\in$ **FB**. Se numeşte *drum consistent pentru* f *în* D, un drum (d) de la rădăcină la o frunză care satisface condiția că pentru fiecare x $\in$ X, (d) conține doar arce reprezentate ca linii punctate sau doar arce reprezentate ca linii continue, care ies din fiecare nod etichetat cu x (acest lucru echivalează cu a stipula că pentru a afla valoarea de adevăr a lui f într-o asignare dată, unei variabile x nu i se pot atribui simultan valorile 0 şi 1, ceea ce este absolut normal)
- Atunci, f este *satisfiabilă* dacă şi numai dacă există o **BDD** D care o reprezintă, precum şi un drum consistent pentru ea în D astfel încât el „leagă” rădăcina de o frunză etichetată cu 1 (similar cu **LP** se definesc noțiunile de formulă *validă* şi *contradicție*)

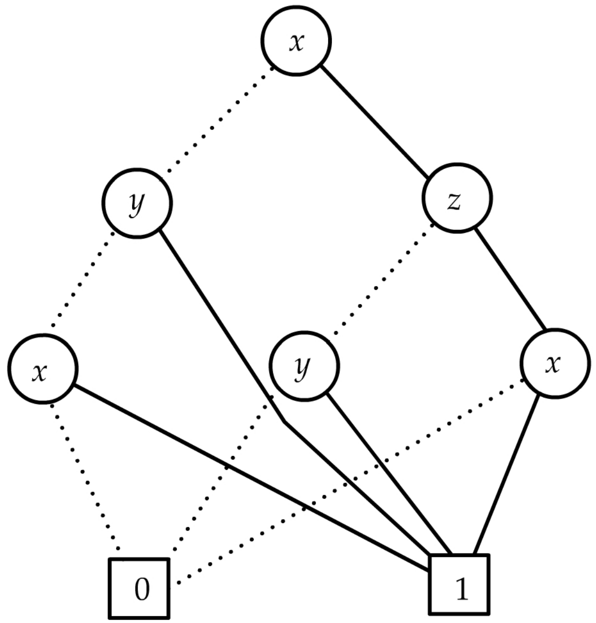
(O)BDD – 19

- **Definiție (diagrame de decizie binare ordonate).** Fie $X = \{x_1, x_2, \dots , x_n\}$ o mulțime de variabile considerată a fi deja total (strict) ordonată (X este de fapt lista $< x_1, x_2, \dots , x_n >$) şi o **BDD** D peste X. Spunem că D *are ordinea* $< x_1, x_2, \dots , x_n >$ dacă şi numai dacă pentru fiecare drum (d) în D de la rădăcină la orice frunză şi fiecare apariție a etichetelor distincte x_i şi x_j pe noduri din (d), dacă x_i apare înaintea lui x_j atunci $i < j$
- O **BDD** D se numeşte *ordonată* (pe scurt, **OBDD**), dacă există o listă de variabile X (inclusiv lista vidă sau cea care conține un unic element) astfel încât D are ordinea X

(O)BDD - 20

- Deşi esenţiale pentru testarea satisfiabilităţii unei formule date (a cărei semantică este dată de funcţia din **FB** reprezentată ca **(O)BDD**) sunt doar variabilele care apar într-o **BDD** care o calculează, să notăm că nu am cerut în mod explicit ca lista să conţină *exact* etichetele care apar într-o **OBDD**
- Astfel, dacă un **OBDD** are ordinea <x, y, z> atunci ea are şi ordinea <u, x, y, v, z, w>, etc.
- Deoarece am presupus că ordinea este totală şi strictă, relaţia respectivă „$<$” nu este reflexivă, este antisimetrică (în sensul că dacă x $<$ y atunci nu putem avea şi y $<$ x) şi tranzitivă
- Datorită acestor proprietăţi, într-o **OBDD** nu există apariţii multiple ale unei variabile pe un drum şi este clar că există măcar o **BDD** care nu este şi o **OBDD**:

(O)BDD - 21



(O)BDD - 22

- Cu toate aceste posibile avantaje, se pare totuşi că reprezentarea cu ajutorul **(O)BDD** a funcțiilor booleene nu este încă destul de „convingătoare”
- Nu sunt astfel „vizibili” algoritmi simpli pentru a testa echivalența semantică a două **(O)BDD** deja reduse dar diferite (ca în cazul tabelelor de adevăr) şi nici pentru a le „compune” uşor (ca în cazul formelor normale din **LP**)

(O)BDD - 23

- Exemplele anterioare ne furnizează atât o **OBDD** neredusă cât şi una maximal redusă
- **Definiție (ordini compatibile).** Fie **O1** şi **O2** două ordini (totale, stricte) peste mulțimile de variabile X şi respectiv Y (notăm Z = X U Y). **O1** şi **O2** se numesc *compatibile* dacă nu există variabilele distincte x, y $\in$ Z astfel încât x precede y în **O1** iar y precede x în **O2**

(O)BDD - 24

- Restrângând clasa ordinilor posibile la ordinile compatibile, obținem imediat adevărul următoarei afirmații
- **Teoremă (unicitatea OBDD-urilor maximal reduse)**. Fie f $\in$ **FB** orice funcție booleană. Atunci **OBDD**-ul maximal redus care o reprezintă este unic via ordinile compatibile (mai exact, fie D şi D' două **OBDD**-uri maximal reduse care reprezintă f - adică semantic echivalente). Atunci D şi D' coincid

(O)BDD - 25

- Din teorema precedentă rezultă că **verificarea echivalenței semantice devine banală** în acest context (într-o anumită implementare ar trebui să verificăm pur şi simplu egalitatea a doi pointeri)
- Mai rezultă şi faptul că **indiferent de ordinea în care aplicăm reducerile vom obține aceeaşi diagramă maximal redusă**
- **Definiție (forma canonică a diagramelor ordonate).** Fie orice $n \in \mathbf{N}$, orice $f \in \mathbf{FB}^{(n)}$ şi orice mulțime de „variabile" total şi strict ordonată $X = \langle x_1, x_2, \ldots, x_n \rangle$. Fie D o **OBDD** peste X, care are ordinea X, este maximal redusă şi reprezintă f. Atunci D se numeşte

  (**OBDD**-)*forma canonică a lui* f

(O)BDD - 26

- Am putea deduce că dimensiunea unei **OBDD** este independentă de ordinea aleasă
- Din păcate acest lucru nu este valabil în general şi dependența dimensiunii unei **OBDD** de ordinea aleasă este prețul pe care îl plătim pentru avantajele pe care **OBDD**-rile le au față de **BDD**-uri şi alte tipuri de reprezentări

(O)BDD - 27

- În concluzie, chiar dacă nu trebuie să supraestimăm importanța **OBDD**-urilor şi a existenței reprezentărilor canonice pentru funcțiile booleene, putem enumera următoarele avantaje ale utilizării acestora:

-Formele canonice sunt în multe cazuri reprezentări mai compacte decât cele folosite în mod uzual (tabele, forme normale)

(O)BDD - 28

- -Formele canonice se pot construi efectiv şi în mod unic pornind de la alte reprezentări (şi reciproc)
- -Nu conțin apariții nenecesare de variabile; dacă valoarea lui $f \in$ **FB**$^{(n)}$ în $<x_1, x_2, \ldots , x_n>$ nu depinde de un x_i atunci forma canonică care reprezintă f nu va conține nici un x_i-nod (nod etichetat cu x_i)
- -Dacă două funcții $f, g \in$ **FB**$^{(n)}$ sunt reprezentate de **Df** respectiv **Dg**, **OBDD**-uri canonice cu ordini compatibile, atunci se poate decide simplu echivalența semantică a lui **Df** şi **Dg** adică egalitatea $f = g$

(O)BDD - 29

- Putem testa dacă o funcție este satisfiabilă „lucrând” pe forma sa canonică: forma canonică nu este **D0**; validitatea/contradicția este la fel de simplu de testat: forma canonică a funcției coincide cu **D1**/**D0**
- Putem testa dacă f „implică” g (f, g $\in$ **FB**$^{(n)}$), adică dacă pentru fiecare $<a_1, a_2, \ldots , a_n> \in$ **B**^{n}, din

  $f(a_1, a_2, \ldots , a_n) = 1$ rezultă $g(a_1, a_2, \ldots , a_n) = 1$. Asta înseamnă să calculăm forma canonică pentru $f \cdot \overline{g}$

  şi aceasta trebuie să coincidă cu **D0** în cazul în care implicația este adevărată
- Şi în cazul acestei reprezentări se pot defini, prin algoritmi eficienți, anumite operații asupra **OBDD**-urilor (formelor canonice) prin care putem „construi” întreaga clasă a funcțiilor booleene (şi nu numai), pornind cu anumite funcții de bază (elementare)

Final **FB**

- Index recapitulativ
- Exerciții suplimentare
- Alte comentarii (ce a fost, ce va fi, în acest curs)

CURS 4

- Continuăm cu Logica propozițională (**LP**)

LP - Sintaxa 1

- Logica propozițională, d.p.d.v. sintactic, este o mulțime de „formule” (*propoziționale*), notată **LP**
- **Definiție (constructivă)**:

  -Fie o mulțime numărabilă de *variabile propoziționale* (*formule elementare*, *formule atomice pozitive*, *atomi pozitivi*), **A** = $\{A_1, A_2, \ldots\}$
- Fie, de asemenea, **C** = $\{\rceil, \vee, \wedge\}$ mulțimea *conectorilor/conectivelor logici/logice:* non (**negația**), sau (**disjuncția**), respectiv şi (**conjuncția**) şi **P** = { (,) } mulțimea *parantezelor* (rotunde)
- *Formulele* (elementele lui **LP**) vor fi „cuvinte” (*expresii bine formate - wff*) peste *alfabetul* **L** = **A** U **C** U **P**

LP - Sintaxa 2

-Baza (*formulele elementare sunt formule*):

A $\subseteq$ **LP**

-Pas constructiv (*obținere formule noi din formule vechi*):

(i) Dacă $F \in$ **LP** atunci $(\neg F) \in$ **LP**

(ii) Dacă F_1, $F_2 \in$ **LP** atunci $(F_1 \vee F_2) \in$ **LP**

(iii) Dacă F_1, $F_2 \in$ **LP** atunci $(F_1 \wedge F_2) \in$ **LP**

(iv) Dacă $F \in$ **LP** atunci $(F) \in$ **LP**

(v) *Nimic altceva nu mai este formulă*

LP - Sintaxa 3

- Arbori, subformule
- $((\rceil F) \vee G)$ se va nota cu $(F \rightarrow G)$
- Pentru $(((\rceil F) \vee G) \wedge ((\rceil G) \vee F))$ folosim $(F \leftrightarrow G)$ sau $((F \rightarrow G) \wedge (G \rightarrow F))$
- $\bigwedge_{i=1}^{n} F_i$ este o prescurtare pentru $F_1 \wedge F_2 \wedge \ldots \wedge F_n$
- $\bigvee_{i=1}^{n} F_i$ este prescurtarea lui $F_1 \vee F_2 \vee \ldots \vee F_n$
- Comentarii asupra noilor simboluri

LP - Sintaxa 4

- Vom numi **literal** o variabilă propoziţională sau negaţia sa
- $A \in$ **A** se va numi **literal pozitiv** iar orice element de forma $\rceil A$, $A \in$ **A** va fi un **literal negativ** (vom nota $\overline{\mathbf{A}} = \{\rceil A_1, \rceil A_2, \ldots\}$)
- Dacă L este un literal, atunci **complementarul** său, $\overline{L}$, va nota literalul $\rceil A$, dacă $L = A \in$ **A** şi respectiv literalul A dacă $L = \rceil A$
- Se numeşte **clauză** orice disjuncţie (finită) de literali
- Se numeşte **clauză Horn** o clauză care are cel mult un literal pozitiv
- O **clauză pozitivă** este o clauză care conţine doar literali pozitivi, iar o **clauză negativă** va conţine doar literali negativi
- O **clauză Horn pozitivă** va conţine exact un literal pozitiv (dar, posibil, şi literali negativi)

LP – Semantica 1

- *Semantica (înțelesul)* unei formule propoziționale este, conform principiilor logicii aristotelice, o valoare de adevăr (**a**(=1) sau **f** (=0)), obținută în mod determinist, care este independentă de context. Notând de la început pe **a** cu 1 şi pe **f** cu 0, astfel încât să putem „lucra” cu algebra booleană $\mathcal{B}$ = < **B**, •, +, $\bar{\ }$ >, noțiunea principală este cea de ***asignare*** **(*interpretare*, structură)**
- **Definiție.** Orice funcție **S**, **S** : **A** $\rightarrow$ **B** se numeşte **asignare**

LP – Semantica 2

- **Teoremă (de extensie)**. Pentru fiecare asignare **S** există o *unică extensie* a acesteia, **S'** : **LP** $\rightarrow$ **B** (numită tot **structură** sau **interpretare**), care satisface:

  (i) **S'**(A) = **S**(A), pentru fiecare A $\in$ **A**

  (ii) **S'**(($\rceil$ F)) = $\overline{\mathbf{S'}(F)}$ pentru fiecare F $\in$ **LP**

  (iii) **S'**(($F_1 \wedge F_2$)) = **S'**(F_1) • **S'**(F_2),

  pentru fiecare F_1, F_2 $\in$ **LP**

  (iv) **S'**(($F_1 \vee F_2$)) = **S'**(F_1) + **S'**(F_2), pentru fiecare F_1, F_2 $\in$ **LP**

  (v) **S'**((F)) = **S'**(F), pentru fiecare F $\in$ **LP**

  (demonstrație)

LP – Semantica 3

- **Definiție** (folosim tot **S** în loc de **S'**).

  -O formulă F $\in$ **LP** se numeşte **satisfiabilă** dacă există măcar o structură **S** (completă) pentru care formula este adevărată (**S**(F) = 1). Se mai spune în acest caz că **S** este model pentru F (simbolic, se mai scrie **S** $\models$ F)

  -O formulă este **validă (tautologie)** dacă *orice* structură este model pentru ea

  -O formulă este **nesatisfiabilă (contradicție)** dacă este falsă în orice structură (**S**(F) = 0, pentru fiecare **S**, sau **S** $\not\models$ F, pentru fiecare **S**)

LP – Semantica 4

- **Teoremă**. O formulă F $\in$ **LP** este validă dacă şi numai dacă ($\rceil$F) este contradicție (demonstrație)
- Clasa tuturor formulelor propoziționale **LP** este astfel partiționată în trei mulțimi *nevide şi disjuncte*: tautologii (formule valide), formule satisfiabile dar nevalide, contradicții (formule nevalide); desen…

LP – Semantica 5

- **Definiție**.

  -Două formule F_1, $F_2 \in$ **LP** se numesc **tare echivalente** dacă *pentru fiecare* structură S ele au aceeaşi valoare de adevăr, adică $S(F_1) = S(F_2)$ (simbolic, vom scrie

  $F_1 \equiv F_2$)

  -F_1 şi F_2 se numesc **slab echivalente** dacă F_1 satisfiabilă *implică* F_2 satisfiabilă şi reciproc (vom scrie $F_1 \equiv_s F_2$, ceea ce înseamnă că *dacă* există $\mathbf{S_1}$ *astfel încât* $\mathbf{S_1}(F_1) = 1$*, atunci* există $\mathbf{S_2}$ *astfel încât* $\mathbf{S_2}(F_2) = 1$ şi reciproc)

LP – Semantica 6

-O formulă F $\in$ **LP** este **consecință semantică** dintr-o mulțime de formule **G** $\subseteq$ **LP**, dacă: *pentru fiecare structură corectă* ***S*** (aceasta înseamnă …), *dacă* ***S*** *satisface* ***G*** (adică avem **S**(G) = 1 pentru fiecare G $\in$ **G**) *atunci* **S** *satisface* F (simbolic, vom scrie **G** $\models$ F)

LP – Semantica 7

- **Teoremă**. Fie G $\in$ **LP** şi
  G = { G_1, G_2, …, G_n } $\subseteq$ **LP**.
  Următoarele afirmații sunt echivalente:
- (i) G este consecință semantică din **G**
- (ii) $(\bigwedge_{i=1}^{n} G_i) \rightarrow G$ este tautologie
- (iii) $(\bigwedge_{i=1}^{n} G_i) \wedge \rceil G$ este contradicție
- (demonstrație)

LP – Semantica 8

- **Teoremă (de echivalență)**. Sunt adevărate următoarele echivalențe (tari, pentru oricare F, G, H $\in$ **LP**):

  (a) $F \wedge F \equiv F$

  (a') $F \vee F \equiv F$ *(idempotență)*

  (b) $F \wedge G \equiv G \wedge F$

  (b') $F \vee G \equiv G \vee F$ *(comutativitate)*

  (c) $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$

  (c') $(F \vee G) \vee H \equiv F \vee (G \vee H)$ *(asociativitate)*

  (d) $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$

  (d') $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$ *(distributivitate)*

  (e) $F \wedge (F \vee G) \equiv F$

  (e') $F \vee (F \wedge G) \equiv F$ *(absorbție)*

LP – Semantica 9

(f) $\neg\neg F \equiv F$ *(legea dublei negații)*

(g) $\neg(F \wedge G) \equiv \neg F \vee \neg G$

(g') $\neg(F \vee G) \equiv \neg F \wedge \neg G$ *(legile lui deMorgan)*

(h) $F \vee G \equiv F$

(h') $F \wedge G \equiv G$ *(legile validității,* adevărate *doar dacă* F este tautologie*)*

(i) $F \wedge G \equiv F$

(i') $F \vee G \equiv G$ *(legile contradicției,* adevărate *doar dacă* F este contradicție*)*

- Generalizări pentru mai multe formule
- Demonstrație

LP – Semantica 10

- **Teoremă (de substituție)**. Fie H $\in$ **LP**, oarecare. Fie orice F, G $\in$ **LP** astfel încît F este o subformulă a lui H şi G este tare echivalentă cu F. Fie H’ formula obținută din H prin înlocuirea (unei apariții fixate a) lui F cu G. Atunci H $\equiv$ H’ (demonstrație)

CURS 5

- Continuăm cu FORME NORMALE

LP – Forme normale 1

- Spre deosebire de cazul funcțiilor booleene, studiem *formele normale conjunctive* şi *formele normale disjunctive* ***simultan*** (pentru început…)
- **Definiție**. O formulă $F \in$ **LP** se află în **formă normală conjunctivă** (**FNC**, pe scurt) dacă este o *conjuncție de disjuncții de literali*, adică o *conjuncție de clauze:*

$$F = \bigwedge_{i=1}^{m} \Big(\bigvee_{j=1}^{n_i} L_{i,j} \Big)$$

Similar, $F \in$ **LP** este în **formă normală disjunctivă** (**FND**, pe scurt), dacă este o *disjuncție de conjuncții de literali:*

$$F = \bigvee_{i=1}^{m} \Big(\bigwedge_{j=1}^{n_i} L_{i,j} \Big)$$

În cele de mai sus $L_{i,j} \in \mathbf{A} \cup \overline{\mathbf{A}}$

LP – Forme normale 2

- **Teoremă**. Pentru fiecare formulă $F \in$ **LP** există cel puțin două formule F_1, $F_2 \in$ **LP**, F_1 aflată în **FNC** şi F_2 aflată în **FND**, astfel încât $F \equiv F_1$ şi $F \equiv F_2$ (se mai spune că F_1 şi F_2 sunt *o* **FNC**, respectiv *o* **FND**, *pentru* F) (demonstrație)

LP – Forme normale 3

- Conform teoremei anterioare, precum şi datorită comutativităţii şi idempotenţei disjuncţiei, comutativităţii şi idempotenţei conjuncţiei (repetarea unui element, fie el literal sau clauză, este nefolositoare din punctul de vedere al (ne)satisfiabilităţii unei formule), este justificată ***scrierea ca mulţimi a formulelor aflate în FNC***
- Astfel, dacă F este în **FNC**, vom mai scrie

  $F_L = \{C_1, C_2, \ldots , C_m\}$ (nu uităm totuşi că *virgula aici provine dintr-o conjuncţie,* C_i fiind clauze)

LP – Forme normale 4

- Fiecare clauză C_i poate fi la rândul ei reprezentată ca o mulțime, $C_i = \{L_{i,1}, L_{i,2}, \ldots, L_{i,ki}\}$, $L_{i,j}$ fiind literali
- Mai mult, dacă avem F $\in$ **LP** reprezentată ca mulțime (de clauze) sau ca mulțime de mulțimi (de literali) şi ne interesează doar studiul (ne)satisfiabilității ei, putem elimina clauzele C care conțină atât L cât şi $\overline{L}$, deoarece $L \vee \overline{L} \equiv \mathbf{1}$, $\mathbf{1} \vee C \equiv \mathbf{1}$ şi deci aceste clauze sunt tautologii (notate generic cu **1**)
- Tautologiile componente nu au nici o semnificație pentru stabilirea valorii semantice a unei formule F aflate în **FNC** ($\mathbf{1} \wedge C \equiv C$)

Formule Horn în **LP** - 1

- O clauză Horn este o disjuncție de literali care conține *cel mult* un literal pozitiv
- **Definiție**. O **formulă Horn** este o formulă aflată în **FNC**, clauzele componente fiind (toate) clauze Horn
- Vom numi (tot) formulă Horn (şi) o formulă care este (tare) echivalentă cu o formulă având forma considerată în definiția precedentă
- Se poate arăta că există formule propoziționale care nu sunt tare echivalente cu nici o formulă Horn, apariția a măcar doi literali pozitivi distincți într-o clauză fiind decisivă
- Formele posibile pentru o formulă Horn sunt (variabilele propoziționale care apar sunt elemente ale lui **A**):

  (i) $C = \rceil A_1 \vee \rceil A_2 \vee \ldots \vee \rceil A_k$, $k \geq 1$, $k \in \mathbf{N}$ şi

  (ii) $C = \rceil A_1 \vee \rceil A_2 \vee \ldots \vee \rceil A_k \vee B$, $k \in \mathbf{N}$

Formule Horn în **LP** - 2

- În afară de reprezentarea ca mulțimi, clauzele Horn pot fi reprezentate şi sub aşa-numita **formă implicațională**
- Vom distinge cazurile (reamintim că **0** şi **1** denotă orice *contradicție* respectiv orice *tautologie*; $\triangleq$ - „egal” prin convenție):

  -C = A $\in$ **A** (*nici un literal negativ, un literal pozitiv*). Acest lucru se mai poate scrie sub forma C $\triangleq$ **1** $\rightarrow$ A, ceea ce se justifică prin aceea că

  $\mathbf{1} \rightarrow A \triangleq \rceil \mathbf{1} \vee A \equiv \mathbf{0} \vee A \equiv A$

  -C = $\rceil A_1 \vee \rceil A_2 \vee \ldots\ldots \vee \rceil A_k$ (*nici un literal pozitiv, măcar un literal negativ*). Vom scrie

  $C \triangleq A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_k \rightarrow \mathbf{0}$ (folosim din nou definiția implicației şi faptul că $\mathbf{0} \vee A \equiv A$)

Formule Horn în **LP** - 3

-C = $\rceil A_1 \vee \rceil A_2 \vee \ldots \vee \rceil A_k \vee B$ (*exact un literal pozitiv, măcar un literal negativ*). Atunci $C \triangleq A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_k \rightarrow B$, direct din definiția implicației

-C $\triangleq$ □ (*nici un literal negativ, nici un literal pozitiv*)

- Din motive tehnice vom folosi şi această **clauză vidă** (în reprezentarea clauzelor cu mulțimi vom folosi pentru □ simbolul Ø, al mulțimii vide). Prin convenție, □ *este o clauză de orice tip* (inclusiv o clauză Horn), dar *nesatisfiabilă*

Formule Horn în **LP** - 4

- **Teoremă.** Satisfiabilitatea formulelor Horn este decidabilă în timp liniar.

  Demonstrația se bazează pe următorul algoritm:

Algoritm Horn

Intrare: Orice formulă Horn, F, reprezentată ca mulțime de clauze, clauzele componente fiind clauze Horn diferite de clauza vidă şi scrise sub formă implicațională

Ieşire: „DA”, în cazul în care formula F este satisfiabilă (furnizându-se şi o asignare **S** care este model pentru F) şi „NU” în caz contrar (F nu este satisfiabilă)

Formule Horn în **LP** – 5

- **Metodă** (*de „marcare*”):

Pasul 1. i := 0

Pasul 2.

Cât_timp ((există în F o clauză C de forma $A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow B$, cu A_1, A_2, A_3, ... , A_k *marcați* şi B *nemarcat* sau de forma $A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow$ **0**, cu A_1, A_2, A_3, ... , A_k *marcați*) şi (i = 0))

execută

Pasul 3. Alege un asemenea C ca mai sus.

Pasul 4. **Dacă** ($C = A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow B$)

atunci

Pasul 5. Marchează B peste tot în F

altfel

Pasul 6. i := 1

Sf_Dacă

Sf_Cât_timp

Formule Horn în **LP** - 6

Pasul 7. **Dacă** (i = 0)

atunci

Pasul 8. Scrie „DA”

Pasul 9. Scrie **S**, cu **S**(A) = 1 dacă şi numai dacă A apare în F şi este *marcat*

altfel

Pasul 10. Scrie „NU”

Sf_Dacă

- Urmează demonstrația ***corectitudinii***

Formule Horn în **LP** - 7

- **Arătăm** mai întâi că **algoritmul *se termină* pentru fiecare intrare**. Să precizăm că acțiunea de *marcare* o privim inițial în sensul că *toate variabilele se presupun a fi nemarcate*

  -Dacă F conține clauze de forma **1** $\rightarrow$ B (care se *consideră a fi de fapt de forma*

  $A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow B$, cu $A_1, A_2, A_3, \ldots, A_k$ *marcați* şi B *nemarcat*), se procedează conform Algoritmului, adică se marchează toate aparițiile lui B în F şi se trece la pasul următor. Mai departe, la *fiecare* execuție a *corpului buclei* (**Paşii 3.** şi **4.**), fie se marchează o variabilă propozițională nouă (nemarcată încă), fie se iese din execuția buclei (şi algoritmul se termină); pentru că numărul de variabile peste care este construită formula F este finit, terminarea algoritmului este evidentă

  -Dacă nu există deloc clauze de tipul **1** $\rightarrow$ B, algoritmul se termină fără nici o execuție a corpului buclei, cu răspunsul „DA” (formula este satisfiabilă) şi cu asignarea **S**, în care **S**(A) = 0 pentru fiecare A (care apare în F)

Formule Horn în **LP** - 8

- **Arătăm** în continuare că **algoritmul este *corect***; aceasta înseamnă că *ieşirea algoritmului satisface ceea ce am dorit*, adică răspunsul „DA”/**S** corespunde faptului că formula F furnizată la intrare este satisfiabilă (şi **S** este model pentru F) iar răspunsul „NU” corespunde faptului că F este nesatisfiabilă
- Vom separa cazurile:

Formule Horn în **LP** - 9

- **Cazul a)** La terminarea execuției se obține „DA” şi F nu conține clauze C de tipul **1** $\rightarrow$ B (sunt doar clauze de forma…şi **S**(A) = 0 pentru fiecare A le face pe toate adevărate)
- **Cazul b)** La terminare se obține „DA” iar F conține şi clauze C = **1** $\rightarrow$ B; atunci bucla se termină după un anumit număr de execuții ale corpului său, valoarea lui i este 0 şi F conține în final clauze C având *marcate* anumite variabile (din nou, există doar următoarele posibilități…; de fapt, **1** $\rightarrow$ **0** nu poate fi printre aceste posibilități)

Formule Horn în **LP** - 10

- **Cazul c)** Algoritmul se termină cu i = 1 şi răspunsul „NU”; acest lucru înseamnă că există în F o clauză $\underline{C} = A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow \mathbf{0}$ cu toți A_i, $i \in [k]$ marcați (obligatoriu, în F există şi clauze de forma $\mathbf{1} \rightarrow B$, B marcat), de unde rezultă că semantica lui C în *asignarea furnizată de algoritm* este de forma $\mathbf{1} \rightarrow \mathbf{0}$ şi prin urmare $\mathbf{S}(\underline{C}) = 0$, de unde $\mathbf{S}(F) = 0$
- Acest lucru **nu înseamnă însă că F este nesatisfiabilă**
- Pentru a trage această concluzie trebuie să arătăm că *nici o altă asignare nu poate fi model pentru* F

Formule Horn în **LP** - 11

- Să presupunem **(RA)** că există o asignare **S'** (diferită de **S**, cea furnizată de algoritm) astfel încât **S'**(F) = 1
- Să observăm, pentru început, că toate variabilele care au fost marcate în algoritm (deci cele care au primit valoarea de adevăr 1 în **S**), trebuie să primească valoarea 1 în oricare **S'** cu S'(F) = 1; a*ltfel spus, asignarea* **S** *conține cel mai mic număr posibil de valori 1 (atribuite evident variabilelor marcate) astfel încît formula* ***să aibă şanse*** *să fie satisfiabilă*; într-adevăr, pentru fiecare **S'** cu **S'**(F) = 1, trebuie să avem **S'**(C) = 1 pentru fiecare clauză C din F
- Să ne ocupăm puțin de momentul în care se marchează o variabilă B, ordonând clauzele din F de forma

  $C = A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow B$ ($k \geq 1$) după numărul de variabile din antecedent (chiar în algoritm, selecția unei clauze „*pentru marcare*" se poate face după un asemenea criteriu)

Formule Horn în **LP** - 12

- Începem cu *clauze C de tipul* $\mathbf{1} \rightarrow B \equiv B$ *(nici o variabilă în antecedent, B nemarcat)*; de la acestea *începe* de fapt procesul de marcare; pentru că **S'**(C) trebuie să fie egal cu 1, este clar că trebuie pus **S'**(B) = 1 (B se şi marchează, deci **S**(B) = 1)
- Continuăm cu *clauzele C de forma* $A \rightarrow B \equiv \rceil A \vee B$ *(o variabilă în antecedent; A este marcat, B nemarcat)*; A nu putea fi marcat decât dacă **a apărut deja ca un consecvent** într-o clauză de tipul anterior, sau în una de acelaşi tip cu aceasta şi care are antecedentul marcat; prin urmare, în orice **S'** cu **S'**(C) = 1, trebuie oricum să avem **S'**(A) = 1, deci **S'**($\rceil$ A) = 0 şi atunci **S'**(B) = 1 (consecința este că B se marchează, deci din nou **S**(B) = 1)
- Continuăm raționamentul cu $C = A_1 \wedge A_2 \rightarrow B$ *(două variabile în antecedent; ambele variabile marcate; B este, încă, nemarcat)* şi ajungem iar la concluzia că pentru fiecare **S'**, pentru a avea **S'**(C) = 1, trebuie să avem **S'**(B) = 1, precum şi **S**(B) = 1

Formule Horn în **LP** - 13

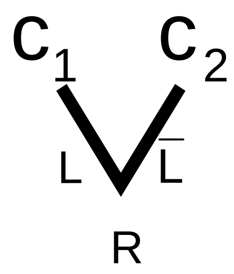
- Revenind, am arătat că pentru fiecare **S'** astfel încât **S'**(F) = 1, trebuie să avem şi **S'**(A) = 1 pentru fiecare A *marcat* de către algoritm, adică pentru fiecare A care satisface **S**(A) = 1 (procesul descris mai sus se continuă pentru oricâte variabile prezente în antecedent, iar numărul acestora este finit)
- Prin urmare, avem şi **S'**($\underline{C}$) = 0, de unde rezultă că **S'**(F) = 0, ceea ce este absurd
- Să arătăm în final că **algoritmul Horn are timp de execuție liniar**
- Faptul că t(n) $\in$ O(f(n)), unde f(n) = a•n + b (a, b $\in$ **N***), rezultă imediat din faptul că la fiecare execuție a *corpului buclei* se *marchează* o *nouă* variabilă
- Desigur că, pentru a obține în mod real acest lucru, algoritmul trebuie detaliat, în sensul că, de exemplu, în **Paşii** de tip **3** (de alegere a unei clauze corespunzătoare C), selecția variabilei de marcat trebuie făcută prin parcurgerea de un număr fix de ori (*independent* de numărul de execuții) a listei variabilelor peste care este construită F
- Exemplu

CURS 6

- Continuăm cu „Metoda rezoluției”

Rezoluție în **LP** - 1

- Fără a restrânge generalitatea, putem presupune că lucrăm cu formule din **LP** aflate în **FNC**, reprezentate sub formă de mulțimi (finite) de clauze, iar clauzele ca mulțimi (finite) de literali
- **Definiție (rezolvent)**. Fie clauzele C_1, C_2 , R. Spunem că **R este rezolventul lui C_1, C_2** (sau că **C_1, C_2 se rezolvă în R**, sau că **R se obține prin rezoluție *într-un pas* din C_1, C_2**), pe scurt, $R = Res_L(C_1, C_2)$, dacă şi numai dacă există un literal $L \in C_1$ astfel încât $\overline{L} \in C_2$ şi
  $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\})$
- Vom putea reprezenta acest lucru şi grafic, prin *arborele de rezoluție*:



Rezoluție în **LP** - 2

- Rezolventul a două clauze este tot o clauză

  (mai mult, rezolventul a două clauze Horn este tot o clauză Horn)
- Clauza vidă (□) poate fi obținută prin rezoluție din două clauze de forma $C_1 = \{A\}$ şi $C_2 = \{\rceil A\}$
- În definiția anterioară putem considera că C_1 şi C_2 sunt diferite între ele. Dacă ele ar coincide, atunci

  $C_1 = C_2 = C = \ldots \vee L \vee \ldots \vee \overline{L} \vee \ldots \equiv \mathbf{1}$, adică acele clauze sunt tautologii, detectabile sintactic (în acest caz nu ne mai interesează alte metode formale de studiere a satisfiabilității lor)
- De asemenea vom „rezolva” doar clauzele în care literalul L cu acea proprietate este unic

Rezoluție în **LP** - 3

- **Teoremă (lema rezoluției)**. Fie oricare formulă $F \in$ **LP** (aflată în **FNC** şi reprezentată ca mulțime de clauze) şi R un rezolvent pentru C_1, $C_2 \in F$. Atunci F este tare echivalentă cu

  F U {R} (demonstrație)
- În teorema anterioară am fi putut considera, în loc de F, o mulțime oarecare de clauze, chiar infinită (vezi **Teorema de compactitate** care urmează peste câteva slide-uri)

Rezoluție în **LP** - 4

- **Definiție**. Fie F o mulțime oarecare de clauze din **LP** şi C o clauză. Spunem că lista $C'_1, C'_2, \ldots, C'_m$ este o **demonstrație prin rezoluție (în mai mulți paşi) a lui C pornind cu F** dacă sunt satisfăcute condițiile:

  (i) Pentru fiecare $i \in [m]$, fie $C'_i \in F$, fie C'_i este obținut prin rezoluție într-un pas din C'_j, C'_k, cu $j, k < i$

  (ii) $C = C'_m$

Rezoluție în **LP** - 5

- În condițiile definiției, se mai spune că **C este demonstrabilă prin rezoluție** (pornind cu F sau **în ipotezele date de F**)
- Mai mult, pentru a spune acest lucru, este suficient ca F să poată fi **inserată** (prezentă) într-o demonstrație şi nu să fie neapărat ultimul element al acesteia
- Intuitiv, o demonstrație prin rezoluție în mai mulți paşi înseamnă o succesiune finită de rezoluții într-un pas, care poate fi reprezentată şi grafic, printr-un arbore, sau chiar ca un graf oarecare (dacă nu folosim noduri diferite pentru aparițiile distincte ale unei aceleiaşi clauze)

Rezoluție în **LP** - 6

- În particular, dacă C este clauza vidă, atunci demonstrația respectivă se numeşte şi **respingere**
- **Numărul de paşi** dintr-o demonstrație este dat de *numărul de clauze obținute prin rezoluție într-un pas* (din clauze anterioare)
- Acesta poate fi considerat ca fiind o măsură a „mărimii” (*lungimii*) demonstrației
- O altă măsură pentru o demonstrație reprezentată ca text poate fi chiar lungimea listei (numărul total de clauze sau chiar numărul total de clauze distincte)
- Dacă reprezentăm o demonstrație ca un arbore, putem folosi şi măsuri specifice, cum ar fi *adâncimea* arborelui, *numărul de nivele,* etc.

Rezoluție în **LP** - 7

- **Definiție (mulțimea rezolvenților unei mulțimi de clauze).** Fie F o mulțime de clauze din **LP** (nu neapărat finită). Notăm succesiv:

  -$Res(F) = F \cup \{R \mid$ există $C_1, C_2 \in F$ astfel încât $R = Res(C_1, C_2)\}$

  -$Res^{(n+1)}(F) = Res(Res^{(n)}(F))$, $n \in \mathbf{N}$

  -Prin $Res^{(0)}(F)$ vom înțelege F şi atunci vom putea pune şi

  $Res^{(1)}(F) = Res(F)$; mai mult:

$$\mathrm{Res}^{*}(F) = \bigcup_{n \in N} \mathrm{Res}^{(n)}(F)$$

- $Res^{(n)}(F)$ se va numi **mulțimea rezolvenților lui F obținuți în cel mult n paşi**, iar $Res^{*}(F)$ **mulțimea (tuturor) rezolvenților lui F**; aceasta este o definiție **iterativă**
- Exemplu

Rezoluție în **LP** - 8

- Direct din definiție rezultă că:

  $F = Res^{(0)}(F) \subseteq Res(F) = Res^{(1)}(F) \subseteq ...$

  $\subseteq Res^{(n)}(F) \subseteq ... \subseteq \ldots \subseteq Res^*(F)$

- Putem da şi o definiție **structurală** a lui $Res^*(F)$
- Vom nota astfel cu Resc(F) mulțimea definită prin:

  Baza. $F \subseteq Resc(F)$

  Pas constructiv: Dacă $C_1, C_2 \in Resc(F)$ şi $C = Res(C_1, C_2)$, atunci $C \in Resc(F)$

Rezoluție în **LP** - 9

- **Teoremă**. Pentru fiecare $F \in$ **LP**, avem Res*(F) = Resc(F)

  (schiță demonstrație)
- Vom putea astfel folosi ambele notații (definiții) pentru manipularea mulțimii rezolvenților unei mulțimi de clauze (în particular, a unei formule oarecare $F \in$ **LP**, aflată în **FNC**)

Rezoluție în **LP** - 10

- **Teoremă**. Fie F o mulțime de clauze din **LP** (nu neapărat finită). O clauză $C \in$ **LP** se poate demonstra prin rezoluție pornind cu clauzele lui F dacă şi numai dacă există $k \in \mathbf{N}$, asfel încât $C \in Res^{(k)}(F)$

  (schiță demonstrație)

Rezoluție în **LP** - 11

- În cele de mai sus am folosit în majoritatea cazurilor termenul *mulțimea de clauze* F şi nu *formula* F (aflată în **FNC**)
- Deşi pe noi ne interesează doar formulele (care pot fi privite ca mulțimi **finite** de clauze în cazul în care ne interesează doar satisfiabilitatea lor), aproape toate rezultatele sunt valabile şi pentru mulțimi infinite (**numărabile**) de formule (clauze)
- Teorema următoare stabileşte o legătură importantă, privind satisfiabilitatea, între mulțimile infinite şi cele finite de formule oarecare din **LP**

Rezoluție în **LP** - 12

- **Teoremă (de compactitate, pentru LP).** Fie **M** o mulțime infinită (numărabilă) de formule din **LP**. Atunci M este satisfiabilă dacă şi numai dacă fiecare submulțime **finită** a sa este satisfiabilă (fără demonstrație)
- Altă formulare…
- **Teoremă**. Fie F $\in$ **LP**, aflată în **FNC** şi reprezentată ca mulțime (finită) de clauze. Atunci Res*(F) este finită (schiță demonstrație)

Rezoluție în **LP** - 13

- **Teoremă (teorema rezoluției pentru calculul propozițional)**. Fie F o mulțime oarecare de clauze din calculul propozițional. Atunci F este nesatisfiabilă dacă şi numai dacă $\square \in$ Res*(F)

  (demonstrație – poate, seminar…)
- *Corectitudine* şi *completitudine*
- **Ceea ce urmează în legătură cu „rafinări şi restricții", este – deocamdată - opțional**

Rafinări ale rezoluţiei şi „complemente”

- Dacă ne gândim doar la satisfiabilitate (de spus despre **SAT**, **3CNFSAT**, …) avem nevoie de **strategii** pentru a ajunge *cât mai repede* la clauza vidă

Rafinări ale rezoluției - 1

- **Rafinările rezoluției** sunt metode prin care se urmăreşte *obținerea clauzei vide (dacă acest lucru este posibil) într-un număr cât mai mic de paşi de rezoluție*
- Pornind cu formula F = $\{C_1, C_2, \ldots, C_n\}$ (aflată în **FNC** şi scrisă ca o mulțime de clauze, la rândul lor clauzele fiind scrise ca mulțimi de literali), se poate construi *efectiv* mulțimea Res*(F), care poate fi reprezentată (fiind finită), după cum deja ştim, ca un ***graf (chiar arbore…) neorientat*** (nodurile sunt rezolvenții succesivi, inclusiv clauzele inițiale, iar muchiile sunt introduse prin rezoluțiile într-un pas care au fost aplicate)
- Practic, *acest graf ar trebui să cumuleze* ***toate*** *posibilele demonstrații prin rezoluție care pornesc cu clauzele lui F* (reamintim că anumiți rezolvenți sunt totuşi excluşi, deoarece reprezintă – sau generează - tautologii)

Rafinări ale rezoluției - 2

- **Teorema rezoluției** sugerează *crearea* mai întâi a acestui ***graf de rezoluție total*** şi apoi *parcurgerea* lui pentru a vedea dacă □ este (eticheta unui) nod în graf
- *Este suficient să găsim direct o respingere* în loc de *a crea* şi *apoi parcurge* întregul graf
- Rafinările se împart în două mari categorii: ***strategii*** şi ***restricții***

Rafinări ale rezoluției - 3

- **Strategiile** nu restrâng, în general, spațiul de căutare (adică *graful total*) dar folosesc anumite *informații suplimentare despre clauze*, astfel încât să crească şansele pentru selectarea rapidă a unei demonstrații căutate, adică a unui „cel mai scurt drum” pornind de la frunze (elementele lui F), către o rădăcină (clauza vidă)
- Astfel, cel puțin la modul ideal, *graful total nu se construieşte în întregime, ci doar acele porțiuni din el (cât mai puține şi cât mai mici), care este posibil să „conțină” măcar o respingere*
- Cel mai cunoscut exemplu este **strategia unitară**, în care ***se recomandă*** ca *la fiecare pas al rezoluției măcar una dintre clauze să conțină un singur literal* (dacă însă nu mai poate fi aleasă nici o asemenea *clauză unitară*, se continuă procesul de obținere de noi rezolvenți în mod obişnuit)
- Prin urmare, *strategiile nu distrug completitudinea rezoluției* (*dacă o formulă este nesatisfiabilă, atunci se poate demonstra acest lucru prin rezoluție, găsindu-se o respingere*), dar, în cel mai rău caz, este posibil să nu conducă la nici o economie de timp

Rafinări ale rezoluției - 4

- Pe de altă parte, **restricțiile** *distrug în multe situații completitudinea rezoluției (există formule nesatisfiabile pentru care nu se pot găsi respingeri, în situația în care paşii de rezoluție sunt supuşi unor condiții prea restrictive)*, deoarece *spațiul de căutare este practic micşorat* într-un mod, să-i spunem, abuziv
- Astfel, o anumită restricție poate *interzice total* folosirea (într-un pas de rezoluție) a unor clauze având o anumită formă sintactică
- *Restricțiile rămân însă complete pentru anumite subclase de formule propoziționale* (de exemplu, pentru clasa formulelor Horn)
- Există mai multe exemple importante de restricții
- Exemple absolut necesare (rezoluția *unitară*, *pozitivă*, *negativă*, *liniară*, ***SLD***, *bazată pe o mulțime suport*, *de intrare*, etc.)

CURS 7

- O nouă logică: **LP1**
- Sper să nu confundați **LP1** cu **LP**, sau cu **LP-1** (ca titulatură de curs…)
- Sper, de asemenea, să nu confundați diversele notații pentru simbolurile folosite în mod curent şi care provin din overloading sau/şi din diverse tipuri diferite de fonturi (aici, dar şi alte materiale; *rond* sau nu, *arial* sau nu, *tipărit* sau nu, etc.)

LP1 – Sintaxa 1

- Pentru a construi mulțimea de formule a *logicii cu predicate de ordinul I*, **LP1**, vom porni cu următoarele mulțimi de simboluri:

  -$X = \{x_1, x_2, \ldots\}$: o mulțime cel mult numărabilă de **variabile funcționale** sau, pe scurt, **variabile**

  -$\mathcal{P} = \{P_0, P_1, \ldots\}$: o mulțime cel mult numărabilă de **simboluri predicative** (sau **predicate**, sau **relații**), cu arități

  -La rândul său, fiecare P_i este o mulțime cel mult numărabilă de **predicate de aritate i** ($i \in \mathbf{N}$); elementele lui P_0 se mai numesc şi **variabile predicative**

LP1 – Sintaxa 2

-$\mathcal{F}$ = {F_0, F_1, ...}: o mulțime cel mult numărabilă de **simboluri funcționale** (sau **funcții**) cu arități, fiecare F_i fiind o mulțime cel mult numărabilă de **funcții de aritate i** (i $\in$ **N**); elementele lui F_0 se numesc şi **constante (funcționale)**

-$\mathcal{C}1$ = {$\rceil$, $\vee$, $\wedge$}: o mulțime de **conectori logici (conective logice)**, la care se pot adăuga, opțional, şi alte simboluri cum ar fi $\rightarrow$, $\leftrightarrow$ etc.

-$\mathcal{C}2$ = {($\forall$x) | x $\in$ X} U {($\exists$ x) | x $\in$ X}: o mulțime de **cuantificatori (cuantori)**, **universali**, respectiv **existențiali** (($\forall$x) se citeşte „*pentru fiecare x*”, sau „*pentru oricare (orice) x*”, iar ($\exists$ x) – „*există x*”, „*există măcar un x*” etc.)

LP1 – Sintaxa 3

- Ca şi în cazul **LP**, vom porni cu alfabetul total, adică reuniunea mulțimilor precedente, împreună cu mulțimea formată din parantezele rotunde şi *virgula specială* (pe scurt, **P**):

  Alf $= X \cup (\cup P_i) \cup (\cup F_i) \cup \mathcal{C}1 \cup \mathcal{C}2 \cup$ **P**

- Mulțimile $\cup P_i$ şi $\cup F_i$ vor fi notate tot cu $\mathcal{P}$, respectiv $\mathcal{F}$, atunci când nu există confuzii

LP1 – Sintaxa 4

- **Definiție (sintaxa LP1)**. Fie **Alf** alfabetul fixat anterior. Atunci mulțimea **formulelor calculului cu predicate de ordinul I**, $\mathbf{LP1_{Alf}}$, este dată constructiv prin:

  -Baza. Se defineşte mulțimea **formulelor atomice**, notată cu **At**, prin:

  (i) $P_o \subseteq$ **At** (variabilele predicative sunt formule atomice)

  (ii) Pentru fiecare $n \in \mathbf{N^*}$, pentru fiecare $P \in P_n$, pentru fiecare $t_1, t_2, \ldots, t_n \in \mathbf{T}$, avem $P(t_1, t_2, \ldots, t_n) \in$ **At**

  (iii) Nimic altceva nu mai este formulă atomică

  -În cele de mai sus, **T** denotă mulțimea **termilor (funcționali)**, care este la rândul ei definită constructiv astfel:

LP1 – Sintaxa 5

-Baza. $X \subseteq$ **T** şi $F_0 \subseteq$ **T** (*variabilele* şi *constantele* sunt *termi*; se mai numesc şi *elementari*)

-Pas constructiv. Pentru fiecare $n \in$ **N***, pentru fiecare $f \in F_n$, pentru fiecare $t_1, t_2, \dots, t_n \in$ **T**,

avem $f(t_1, t_2, \dots\dots, t_n) \in$ **T**

- Concluzionăm această etapă a definiției prin: **At** $\subseteq$ **LP1** (*formulele atomice sunt formule*)

  -Pas constructiv. Continuăm definirea lui $\mathbf{LP1_{Alf}}$ cu partea *„formule noi din formule vechi”:*

  (i) Dacă $F \in$ **LP1** atunci $(\rceil F) \in$ **LP1**

  (ii) Dacă $F_1, F_2 \in$ **LP1** atunci $(F_1 \wedge F_2), (F_1 \vee F_2) \in$ **LP1** (dacă dorim, putem introduce şi $(F_1 \rightarrow F_2)$, sau

  $(F_1 \leftrightarrow F_2) \in$ **LP1** etc.)

  (iii) Dacă $F \in$ **LP1** atunci $(\forall x)(F) \in$ **LP1** (punem şi $(\exists x)(F) \in$ **LP1**), pentru fiecare $x \in X$

  (iv) Dacă $F \in$ **LP1** atunci $(F) \in$ **LP1**

  -Nimic altceva nu mai este formulă

LP1 – Sintaxa 6

- Similar cu cazul **LP**, se definesc constructiv *subf(*F*)*, *mulțimea subformulelor* formulei F şi *Arb(*F*)*, *arborele ataşat* lui F (cuantorii sunt operatori de *aritate unu*)
- Singura subformulă a unei formule atomice este ea însăşi şi *Arb*(F) va fi constituit în acest caz dintr-un simplu nod
- Un term poate fi, la rândul său, privit ca un arbore (ca de altfel şi orice formulă atomică), astfel încât arborele unei formule poate fi „detaliat”, dacă înlocuim fiecare nod corespunzător unui term cu arborele ataşat acestuia (similar pentru o subformulă atomică)
- Din motive tehnice, toate simbolurile care apar în **Alf** sunt considerate a fi diferite (nu admitem overloading)

LP1 – Sintaxa 7

- **Definiție (apariții libere şi legate ale variabilelor)**. Fie $F \in$ **LP1** şi $x \in X$, astfel încât x apare în F, la o poziție oarecare j (în sens textual, stânga/ dreapta, ca *literă într-un cuvânt*, apariția menționată *nefiind parte a numelui unui cuantificator* $(\forall x)$ sau $(\exists x)$)
- Apariția fixată a lui x se numeşte **legată** dacă este într-o parte (subformulă) G a unei (alte) subformule a lui F de forma $G_1 = (\forall x)(G)$ (sau $(\exists x)(G)$)
- În restul cazurilor, apariția considerată se numeşte **liberă**

LP1 – Sintaxa 8

- Orice apariție a unei variabile într-o formulă poate fi definită formal într-un mod simplu (structural)
- Vom nota, pentru fiecare F $\in$ **LP1**, cu *free(*F*)* - mulțimea variabilelor care au apariții libere în F, şi cu

  *leg(*F*)* - mulțimea variabilelor care au apariții legate în F
- Desigur că, pentru fiecare x $\in$ X, este posibil ca x să nu apară în F, sau să aibă doar apariții libere, sau doar apariții legate, sau şi apariții libere, şi apariții legate
- Putem nota cu **var(**F**) = free(**F**)** U **leg(**F**)**
- O situație nenaturală din punct de vedere semantic, dar posibilă sintactic, este aceea în care o variabilă x nu apare de loc în F (în sensul considerat), dar este prezent ca nume al unui cuantificator
- Vom conveni să notăm mulțimea acestor variabile cu *restvar(*F*)* şi să includem şi această mulțime în *var(*F*)*

LP1 – Sintaxa 9

- **Definiție (închideri)**. O formulă $F \in$ **LP1** se numeşte **închisă** dacă nu conține apariții libere de variabile (altfel spus, $free(F) = \emptyset$)
- Pentru o formulă F, se numeşte **închiderea sa universală** formula $(\forall x_1)((\forall x_2)(\ldots((\forall x_k)(F)) \ldots))$ (notată pentru simplitate şi cu $(\forall *)(F)$ sau chiar $(\forall F)$), unde

  $\{x_1, x_2, \ldots , x_k\} = free(F)$
- Analog (înlocuind $\forall$ cu $\exists$) se defineşte (notează) **închiderea existențială** a lui F
- Se va numi **matricea** lui F (notată F*) acea formulă obținută din F prin ştergerea (sintactică, textuală) a tuturor cuantificatorilor $(\forall x)$ şi $(\exists x)$
- O formulă care nu este închisă se numeşte **deschisă** (o formulă în care $var(F) = \emptyset$ se consideră a fi închisă)
- **Definiție (substituții)**. Prin **substituție** vom înțelege o secvență finită de elemente de tipul [x/t] (numite şi **substituții elementare**), unde $x \in X$, $t \in$ **T**

LP1 – Sintaxa 10

- O substituție va avea astfel forma

  $s = [x_1/t_1]\bullet[x_2/t_2]\bullet \ldots \bullet[x_n/t_n]$ fiind practic un *cuvânt* peste alfabetul **S** = {[x/t] | x ∈ X, t ∈ **T**}, adică un element al lui **S*** (*monoidul liber* generat de **S**)
- **O substituție s** (ca mai sus) **se aplică unei formule F**, rezultând o formulă G, notată (F)s, care se obține din F prin *înlocuirea fiecărei apariții libere* a variabilei x_1 cu termul t_1, *apoi* a fiecărei apariții libere a variabilei x_2 cu t_2, etc.
- De obicei, se utilizează doar **substituții permise** pentru o formulă F ∈ **LP1**
- *Substituția elementară* [x/t] *este permisă pentru* F *(sau,* F ***acceptă*** [x/t]*)* dacă t *nu conține variabile (libere) care au apariții legate în* F (s de mai sus va fi permisă pentru F dacă va fi permisă pentru fiecare *componentă* a sa, $[x_i/t_i]$, $i \in [n]$)

LP1 – Sintaxa 11

- Ordinea (fixată deja prin modul de scriere) aplicării substituțiilor elementare dintr-o substituție s este esențială în majoritatea cazurilor
- O substituție s este **normalizată** (pentru F) dacă *ordinea de aplicare a substituțiilor elementare componente nu contează*
- Mai precis, s este normalizată dacă avem (F)s = (F)s’, pentru fiecare s’ care este obținută din s printr-o *permutare* a componentelor acesteia
- **Substituția vidă** (ca element neutru al lui **S***), notată [ ], nu face desigur nici o transformare în formula F căreia îi este aplicată, adică avem (F)[ ] = F
- Dacă avem (F)s = (F)s’ atunci s şi s’ se mai numesc şi *echivalente pentru* F

LP1 – Sintaxa 12

- Păstrăm notațiile, convențiile, rezultatele, conceptele din **LP**, dacă nu precizăm altfel (de ex.: *literal*, *clauză*, *clauză Horn*, *formă normală*, etc.*)*
- Putem renunța la paranteze; fixa priorități operatorilor (cea mai „mare”: cuantificatorii, restul se păstrează cf. **LP**), *domeniul sintactic al unui operator* (de detaliat...)
- **Definiție.** Un term care nu conține variabile se numeşte **term de bază**. Analog, vom avea **formule de bază**, **substituții de bază**, etc.
- EXEMPLE

CURS 8 - 9

LP1 – Semantica 1

- **Înțelesul (semantica)** unei formule F $\in$ **LP1** va fi, la fel ca în logica propozițională, o valoare de adevăr 0, 1 $\in$ **B**, valoare obținută într-un mod extensional
- Elementul principal în definirea semanticii va rămâne noțiunea de *structură*

LP1 – Semantica 2

- **Definiție.** Se numeşte **structură** un cuplu **S** = <$\mathcal{U}_{\mathbf{S}}$, $\mathcal{I}_{\mathbf{S}}$> în care $\mathcal{U}_{\mathbf{S}}$ este o mulțime **nevidă** numită **univers**, iar $\mathcal{I}_{\mathbf{S}}$ este o funcție (numită şi **interpretare**)
- $\mathcal{I}_{\mathbf{S}}$: X U $\mathcal{F}$ U $\mathcal{P} \rightarrow \mathcal{U}_{\mathbf{S}}$ U [$\mathcal{U}_{\mathbf{S}}$*$\rightarrow$ **B**] U [$\mathcal{U}_{\mathbf{S}}$* $\rightarrow \mathcal{U}_{\mathbf{S}}$], care satisface condițiile:

  -Dacă x $\in$ X, atunci $\mathcal{I}_{\mathbf{S}}(x) \in \mathcal{U}_{\mathbf{S}}$

  -Dacă P $\in$ $\mathbf{P}_{\mathbf{n}}$, atunci $\mathcal{I}_{\mathbf{S}}(P) : \mathbf{U}_S^n \rightarrow \mathbf{B}$

  -Dacă F $\in$ $\mathbf{F}_{\mathbf{n}}$, atunci $\mathcal{I}_{\mathbf{S}}(F) : \mathbf{U}_S^n \rightarrow \mathcal{U}_{\mathbf{S}}$
- [C $\rightarrow$ D] desemnează mulțimea tuturor funcțiilor totale având domeniul C şi codomeniul D, iar [C* $\rightarrow$ D] denotă mulțimea tuturor funcțiilor de oricâte argumente (inclusiv 0) peste C, cu valori în D

LP1 – Semantica 3

- Prin urmare, **interpretarea (semantica) unei variabile în structura S este un element din univers, interpretarea unui simbol predicativ n-ar este o funcție de la U_S^n la {0, 1} (sau, uneori, *mulțimea elementelor din* U_S^n *pentru care valoarea în cauză este 1*), iar semantica unui simbol funcțional de aritate n este o funcție de la U_S^n la U_S**
- Pentru simplificarea exprimării, vom renunța la indici dacă nu există confuzii şi **vom nota pe I_S tot cu S**
- Similar cu cazul logicii propoziționale, orice structură va putea fi unic extinsă astfel încât să fie definită pentru toate elementele lui **LP1**

LP1 – Semantica 4

- **Definiție.** Pentru fiecare structură **S** = <$\mathcal{U}_{\mathbf{S}}$, $\mathcal{I}_{\mathbf{S}}$>, vom numi **extensia sa imediată funcția**

  S' : X U $\mathcal{F}$ U $\mathcal{P}$ U **T** U **LP1** $\to$ $\mathcal{U}_{\mathbf{S}}$ U [$\mathcal{U}_{\mathbf{S}}$* $\to$ $\mathcal{U}_{\mathbf{S}}$] U

  U [$\mathcal{U}_{\mathbf{S}}$* $\to$ **B**] U **B**, dată constructiv în continuare
- Pentru început, vom pune S'(a) = S(a) (= $\mathcal{I}_{\mathbf{S}}$(a)), pentru fiecare a $\in$ X U $\mathcal{F}$ U $\mathcal{P}$, ceea ce înseamnă că S' s-a definit, în particular, *şi pentru fiecare term elementar*
- Fie acum orice t $\in$ **T**, adică orice n$\in$ **N***,orice $t_1, t_2, \ldots, t_n \in$ **T** şi orice f $\in$ $\mathbf{F_n}$, astfel încât t = f(t_1, t_2, …, t_n); atunci S'(t) = S(f)(S'(t_1), S'(t_2), ... , S'(t_n)) ($\in$ $\mathcal{U}_{\mathbf{S}}$)
- *Am încheiat astfel procesul de definire al lui S' pe*

  X U $\mathcal{F}$ U $\mathcal{P}$ U **T** şi rămâne să definim S' pe **LP1**
- Vom face acest lucru în mod constructiv:

LP1 – Semantica 5

-Baza. Fie $F = A \in$ **At**; în această situație avem fie $A = P \in \mathbf{P_0}$ fie $A = P(t_1, t_2, \ldots, t_n)$, $n \in \mathbf{N^*}$, $t_1, t_2, \ldots\ldots, t_n \in \mathbf{T}$; în primul caz S' este deja definită ($S'(P) = S(P) \in \mathbf{B}$), iar în al doilea caz punem desigur $S'(P) = S(P)(S'(t_1), S'(t_2), \ldots, S'(t_n)) \in \mathbf{B}$ (paranteze <>…)

-Pas constructiv. Vom avea de considerat cazurile:

(i) $F = (\rceil F_1)$. Atunci $S'(F) = \overline{S'(F)}$

(ii) $F = (F_1 \wedge F_2)$. Atunci $S'(F) = S'(F_1) \bullet S'(F_2)$

(iii) $F = (F_1 \vee F_2)$. Atunci $S'(F) = S'(F_1) + S'(F_2)$

(iv) $F = (\forall x)(G)$. Atunci $S'(F) = 1$ dacă şi numai dacă pentru fiecare $u \in \mathcal{U}_{\mathbf{S}}$ avem $S'_{[x/u]}(G) = 1$ unde $S'_{[x/u]}$ este o interpretare care coincide în totalitate cu S' exceptând faptul că $S'(x) = u$

(v) $F = (\exists x)(G)$. Atunci $S'(F) = 1$ dacă şi numai dacă există (măcar) un element $u \in \mathcal{U}_{\mathbf{S}}$ astfel încât $S'_{[x/u]}(G) = 1$

- Alte notații, observații (F „la” S; **LP1** „cu” egal, etc.; **LP2**; **LP** $\subseteq$ **LP1**)

CURS 10

- Continuăm într-un mod similar cu modul în care am procedat la **LP**
- Mai exact, va trebui să discutăm despre: *decidabilitate*, *forme normale*, *rezoluție*, etc.

LP1 – Semantica 6

- **Definiție (universuri** şi **structuri Herbrand)**. Fie F $\in$ **LP1**. Se numeşte **univers Herbrand (ataşat lui** F**)**, mulțimea **D(**F**)** definită constructiv prin:

  -Baza. În **D(**F**)** se pun toate elementele din F_0 care apar în F; dacă F nu conține nici o constantă, atunci se pune **forțat** în **D(**F**)** un element oarecare din $\mathbf{F_0}$ (numele rezervat standard, de obicei, este a)

  -Pas constructiv. Fie orice n $\in$ **N***, orice f $\in$ $\mathbf{F_n}$ care apare în F şi termii oarecare

  $t_1, t_2, \ldots, t_n \in$ **D(**F**)**; atunci $f(t_1, t_2, \ldots, t_n) \in$ **D(**F**)**

  -**Nimic** altceva nu mai este în **D(**F**)**

LP1 – Semantica 7

- O **structură Herbrand (pentru** F**)** este o structură (corectă pentru F), $H = <\mathcal{U}_H, \mathcal{I}_H>$, în care $\mathcal{U}_H$ = **D(**F**)**, iar $\mathcal{I}_H$ satisface condiția că „*interpretează fiecare term prin el însuşi*”. Mai exact,

  $H(f(t_1, t_2, \ldots , t_n)) = f^H(<t_1^H, t_2^H, \ldots , t_n^H >) =$
  $= f(t_1^H, t_2^H, \ldots , t_n^H)$, pentru fiecare $n \in \mathbf{N}$ şi fiecare $t = f(t_1, t_2, \ldots , t_n) \in \mathbf{T}$

LP1 – Semantica 8

- Se poate spune că **D(**F**)** *este mulțimea termilor de bază construiți cu simboluri din* F
- Într-o structură Herbrand, dacă f $\in$ $\mathbf{F_0}$ atunci H(f) = f şi în consecință dacă t este un term de bază avem şi $t^H = t$
- Interpretarea unei variabile este cea uzuală ($x^H \in$ **D(**F**)** pentru fiecare x $\in$ X), la fel ca şi interpretarea simbolurilor predicative (ele vor fi funcții oarecare peste **D(**F**)** cu valori în **B**). **A nu se confunda**

  $f^H(<t_1^H, t_2^H, \ldots, t_n^H>)$, care denotă aplicarea efectivă a funcției f^H : D(F**)^n $\rightarrow$ D(**F**) n-uplului $<t_1^H, t_2^H, \ldots, t_n^H>$,** cu **$f(t_1^H, t_2^H, \ldots, t_n^H)$ (valoarea aplicării anterioare), care este un term fără variabile aparținând lui D(**F**), adică, în ultimă instanță, un şir de caractere**
- Dacă există o structură Herbrand care este model pentru o formulă F, atunci spunem şi că F **admite model Herbrand**

LP1 – Semantica 9

- **Teoremă (de extensie)**. Pentru fiecare structură **S** = <$\mathcal{U}_{\mathbf{S}}$, $\mathcal{I}_{\mathbf{S}}$>, extensia sa imediată **S'** este funcție şi este unica funcție având domeniul

  X U P U T U F U **LP1** şi codomeniul

  $\mathcal{U}_{\mathbf{S}}$ U [$\mathcal{U}_{\mathbf{S}}^* \to \mathcal{U}_{\mathbf{S}}$] U [$\mathcal{U}_{\mathbf{S}}^* \to$ **B**] U **B** unde **S'** extinde **S** şi satisface condițiile:

  (i) **S'**(($\rceil$ F)) = $\overline{S'(F)}$, pentru fiecare F $\in$ **LP1**

  (ii) **S'**(($F_1 \wedge F_2$)) = **S'**(F_1) • **S'**(F_2), pentru fiecare F_1, $F_2 \in$ **LP1**

  (iii) **S'**(($F_1 \vee F_2$)) = **S'**(F_1) + **S'**(F_2), pentru fiecare F_1, $F_2 \in$ **LP1**

  (iv) **S'**(($\forall$x)(G)) = 1 dacă şi numai dacă pentru fiecare u $\in \mathcal{U}_{\mathbf{S}}$ avem **S'**$_{[x/u]}$(G) = 1 unde **S'**$_{[x/u]}$ este o interpretare care coincide cu **S'** exceptând faptul că

  S'(x) = u (pentru fiecare x $\in$ X şi fiecare G $\in$ **LP1**)

LP1 – Semantica 10

- Mai sus putem adăuga (conform presupunerilor de până acum) şi

  (v) **S'**$((\exists x)(G))$ = 1 dacă şi numai dacă există (măcar) un element

  $u \in \mathcal{U}_{\mathbf{S}}$ astfel încât $\mathbf{S}'_{[x/u]}(G) = 1$ (pentru fiecare $x \in X$ şi fiecare $G \in$ **LP1**)
- Relația **S'**((F)) = **S'**(F) o putem considera ca fiind adevărată chiar prin convenție

  (demonstrație similară cu **LP**)

LP1 – Forme normale 1

- **Teoremă (de substituție)**. Fie H $\in$ **LP1**, oarecare. Fie orice F, G $\in$ **LP1** astfel încât F este o subformulă a lui H şi G este tare echivalentă cu F. Fie H' formula obținută din H prin înlocuirea (unei apariții fixate a) lui F cu G. Atunci H $\equiv$ H'

  (demonstrație - schiță)

- Echivalențele deja cunoscute din **LP** pot fi completate cu altele, care se referă la cuantori:

  Teoremă. Pentru fiecare F, G $\in$ **LP1** şi fiecare x, y $\in$ X, sunt adevărate următoarele echivalențe:

  1. $\rceil(\forall x)F \equiv (\exists x)(\rceil F)$

     $\rceil(\exists x)F \equiv (\forall x)(\rceil F)$

LP1 – Forme normale 2

2. Dacă x nu apare liber în G, atunci:

   $(\forall x)(F \wedge G) \equiv (\forall x)(F) \wedge G$

   $(\forall x)(F \vee G) \equiv (\forall x)(F) \vee G$

   $(\exists x)(F \wedge G) \equiv (\exists x)(F) \wedge G$

   $(\exists x)(F \vee G) \equiv (\exists x)(F) \vee G$

3. $(\forall x)(F) \wedge (\forall x)(G) \equiv (\forall x)(F \wedge G)$

   $(\exists x)(F) \vee (\exists x)(G) \equiv (\exists x)(F \vee G)$

4. $(\forall x)(\forall y)F \equiv (\forall y)(\forall x)F$

   $(\exists x)(\exists y)F \equiv (\exists y)(\exists x)F$

5. Dacă x nu apare liber în F, atunci:

   $(\forall x)F \equiv F$

   $(\exists x)F \equiv F$

(demonstrație - schiță)

LP1 – Forme normale 3

- Ca o consecință a teoremei anterioare rezultă că sunt adevărate şi echivalențele $(\forall x)(\forall x)F \equiv (\forall x)F$ şi $(\exists x)(\exists x)F \equiv (\exists x)F$ (chiar şi variantele $(\exists x)(\forall x)F \equiv (\forall x)F$, respectiv $(\forall x)(\exists x)F \equiv (\exists x)F$), care se pot generaliza pentru oricâte apariții ale cuantorilor (F ar putea conține la rândul ei cuantificatori)
- Se poate arăta însă că **nu sunt adevărate** echivalențele (mai exact, există $x \in X$, există $F, G \in$ **LP1** astfel încât, mai jos, formulele din membrul stâng nu sunt echivalente cu cele corespunzătoare din membrul drept):

  $(\forall x)F \vee (\forall x)G \equiv (\forall x)(F \vee G)$

  $(\exists x)F \wedge (\exists x)G \equiv (\exists x)(F \wedge G)$
- Nici echivalența $(\forall x)(\exists y)F \equiv (\exists y)(\forall x)F$ **nu este adevărată** pentru fiecare formulă F

LP1 – Forme normale 4

- **Teoremă (lema de translație)**. Fie $F \in$ **LP1**, $x \in X$, $t \in T$ un term de bază şi **S** orice structură corectă pentru formulele care apar. Atunci:

$$\mathbf{S}((F)[x/t]) = \mathbf{S}_{[x/\mathbf{S}(t)]}(F)$$

(demonstrație - schiță)

- **Teoremă (lema de redenumire a variabilelor legate)**. Fie $F = (\circ x)G$, $\circ \in \{\forall, \exists\}$, o formulă oarecare din **LP1**. Fie y o variabilă nouă (în sensul că ea nu apare în G). Atunci :

$$F \equiv (\circ y)(G[x/y])$$

(demonstrație - schiță)

LP1 – Forme normale 5

- **Definiție (forma normală rectificată)**. O formulă $F \in$ **LP1** se numeşte *rectificată* (sau se află în *formă normală rectificată*, pe scurt **FNR**) dacă nu conține variabile care apar atât libere, cât şi legate şi nu are cuantificatori care să lege aceeaşi variabilă, dar pe poziții diferite în formulă (indiferent dacă este vorba de cuantificatori existențiali sau universali)
- **Teoremă.** Pentru orice formulă din $F \in$ **LP1**, există o formulă rectificată $F' \in$ **LP1,** astfel încât $F' \equiv F$

  (demonstrație - schiță)

LP1 – Forme normale 6

- **Definiție (forma normală prenex)**. O formulă $F \in$ **LP1** este în *formă normală prenex* (**FNP**, pe scurt) dacă $F = (\circ_1 y_1) \ldots (\circ_n y_n)G$, unde $n \in \mathbf{N}$, $\circ_i \in \{\exists, \forall\}$ ($i \in [n]$), iar $y_1, \ldots , y_n$ sunt (posibil, toate) variabilele distincte care apar (liber) în G
- În plus, G nu mai conține cuantificatori
- În cele de mai sus, cazul $n = 0$ se referă la absența cuantificatorilor ($[0] = \emptyset$)
- **Teoremă**. Pentru fiecare formulă $F \in$ **LP1**, există o formulă $F' \in$ **LP1**, care este în **FNP** şi este tare echivalentă cu F (demonstrație - schiță)

LP1 – Forme normale 7

- Am arătat că pentru fiecare formulă din **LP1**, există o altă formulă din **LP1**, care este tare echivalentă cu ea şi care este în **FNP** *rectificată* (pe scurt, **FNPR**)
- Putem presupune şi că nu există în această formulă cuantificatori care să lege variabile care nu apar în ea şi nici cuantificatori (relativ la o aceeaşi variabilă) cu apariții multiple
- Cu alte cuvinte, din punctul de vedere al satisfiabilității, ne putem limita la studiul formulelor din **LP1** de forma

  $F = (\circ_1 y_1) \ldots (\circ_k y_k)F'$, unde $free(F') = \{y_1, y_2, \ldots , y_k\}$ (poate fi doar incluziune, pentru moment), iar F' este chiar matricea lui F, nemaiconținând alți cuantori ($\circ_1, \circ_2, \ldots, \circ_k \in \{\forall, \exists\}$)
- Prin urmare (avem şi: *o formulă este satisfiabilă dacă şi numai dacă închiderea sa existențială este satisfiabilă*), pentru testarea satisfiabilității unei formule din **LP1** putem să ne limităm la clasa de formule având aspectul sintactic

  $F = (\circ_1 y_1)(\circ_2 y_2) \ldots (\circ_k y_k)F^*$, unde F^* este matricea lui F iar $leg(F) = var(F) = free(F^*) = \{y_1, y_2, \ldots y_k\}$; această formulă este şi închisă, neconținând apariții libere de variabile

LP1 – Forme normale 8

- **Definiție (forma normală Skolem)**. O formulă F $\in$ **LP1** este în *formă normală Skolem* (**FNS**, pe scurt), dacă are aspectul F = $(\forall x_1) \ldots (\forall x_k)$G unde G nu mai conține cuantificatori (este chiar matricea lui F), iar $x_1, x_2, \ldots, x_k$ sunt variabile distincte şi reprezintă *exact* variabilele care apar în G (free(G) = $\{x_1, x_2, \ldots, x_k\}$). F este în **formă normală Skolem clauzală** (**FNSC**, pe scurt), dacă este în **FNS** şi matricea sa este în **FNC** (forma normală conjuctivă) într-un sens similar cu **LP** (literalii reprezentând acum formule atomice din **LP1** sau negații ale lor)
- **Teoremă.** Pentru fiecare formulă F din **LP1**, există o altă formulă F'$\in$ **LP1**, care este în **FNSC** şi care este slab echivalentă cu ea (demonstrație – schiță; urmează)

  Demonstrație. Vom prezenta un algoritm prin care formula F' va fi construită *efectiv* din formula F

LP1 – Forme normale 9

Algoritm Skolem

- **Intrare**: F $\in$ **LP1**. Fără a restrânge generalitatea, putem presupune că F este în **FNPR**, închisă
- **Ieşire**: F' $\in$ **LP1**, aflată în **FNS** (şi închisă), slab echivalentă cu F
- **Metodă**:
- **Pasul 1**. F' : = F
- **Pasul 2**. **Cât_timp** (există cuantificatori existenţiali în F')

execută

2.1. Alege un asemenea cuantificator şi elimină-l

2.2. Transformă formula F'

Sf_Cât_timp

LP1 – Forme normale 10

- **Comentarii legate de demonstrație.** Orice formulă intermediară prelucrată de algoritm are forma

  $F' = (\forall y_1) \ldots (\forall y_n)(\exists z)G$, unde G poate să conțină şi alți cuantificatori (am pus în evidență primul cuantificator existențial, alegerea fiind acum deterministă dacă ne gândim că parcurgem formula simbol cu simbol, de la stânga la dreapta)
- Atunci, în urma **Paşilor 2.1** şi **2.2**, F' va căpăta forma $F' = (\forall y_1) \ldots(\forall y_n)((G)[z/f(y_1, \ldots , y_n)])$ unde f este un simbol funcțional nou (în sensul că el nu mai apare în formulele considerate – atenție la cardinalitățile alfabetului inițial), $f \in \mathcal{F}_n$
- Să notăm cu $H \triangleq (\forall y_1) \ldots (\forall y_k)(\exists z)G$, formula de tip F' existentă înainte de execuția unui pas al algoritmului precedent şi cu $H' \triangleq (\forall y_1) \ldots (\forall y_k)(G)[z/f (y_1, y_2, \ldots y_k)]$ formula rezultată după execuție
- Este suficient să arătăm că *H este slab echivalentă cu H'*

LP1 – Forme normale 11

- Putem sintetiza rezultatele obținute până în prezent în:

  Teoremă. Pentru fiecare formulă F $\in$ **LP1**, există o formulă F' $\in$ **LP1** astfel încât

  F' $\equiv_s$ F, F' fiind în **FNSC** închisă
  (F' = $(\forall y_1) \ldots (\forall y_n)$F*,$\{y_1, \ldots , y_n\}$=free(F*), F*este matricea lui F şi este în formă normală conjunctivă)
- Exemple, exerciții

CURS 11

Decidabilitate şi rezoluție de bază în LP1 ($LP1_=$) 1

- **Teoremă**. Fie F o formulă din calculul cu predicate de ordinul I fără egalitate, închisă şi aflată în **FNS**. Atunci F este satisfiabilă dacă şi numai dacă F admite un model Herbrand

  (demonstrație - schiță)

Decidabilitate şi rezoluţie de bază în **LP1 ($LP1_=$)** 2

- **Teoremă (Löwenheim – Skolem)**. Fiecare formulă satisfiabilă din **LP1** admite model cel mult numărabil

  (demonstraţie - schiţă)

- **Definiţie (extensia Herbrand)**. Pentru fiecare formulă F închisă, aflată în **FNS,**

  $F = (\forall y_1) \ldots (\forall y_n)F^*$, $\{y_1, \ldots , y_n\} = free(F^*)$, F^* fiind matricea lui F, extensia sa Herbrand este mulţimea

  $\mathbf{E(}F\mathbf{)} = \{(F^*)[y_1/t_1]\bullet[y_2/t_2]\bullet \ldots \bullet[y_n/t_n] \mid t_1, t_2, \ldots, t_n \in \mathbf{D}(F)\}$

Decidabilitate şi rezoluție de bază în **LP1 (LP1$_=$)** 3

- Dacă F este în **FNSC** (F are forma de mai sus, în plus F* fiind în **FNC**, F* = $C_1 \wedge C_2 \wedge \ldots\ldots \wedge C_k$, $C_1, C_2, \ldots, C_k$ reprezentând clauze, adică literali din **LP1**), mulțimea se numeşte **extensia Herbrand generalizată** (notată **E'(**F**)**)
- Extensia Herbrand generalizată a unei formule este obținută practic prin considerarea tuturor instanțelor clauzelor care compun matricea sa (formula fiind deja în **FNSC** şi considerată în reprezentarea cu mulțimi), instanțe obținute prin aplicarea tuturor substituțiilor posibile cu termi de bază din **D(**F**)** (pentru ca F* să fie nesatisfiabilă este suficient ca măcar o clauză C_j să fie nesatisfiabilă)

Decidabilitate şi rezoluție de bază în **LP1 (LP1$_=$)** 4

- **Teoremă (Church)**. Problema validității pentru logica cu predicate de ordinul I (fără egalitate) este nedecidabilă, dar semidecidabilă

  (demonstrație)
- **Teoremă (Gödel-Herbrand-Skolem)**. O formulă

  F $\in$ **LP1** este satisfiabilă dacă şi numai dacă E(F) este satisfiabilă

  (demonstrație)
- **Teoremă (teorema lui Herbrand; teorema de compactitate pentru LP1)**. O formulă F $\in$ **LP1** este nesatisfiabilă dacă şi numai dacă există o submulțime finită a lui **E'(**F**)** care să fie nesatisfiabilă

  (demonstrație)

Decidabilitate şi rezoluţie de bază în **LP1 (LP1$_=$)** 5

- În acest moment putem spune că procedura următoare (intitulată şi **Procedura lui Gilmore**) poate fi *folosită pentru testarea nesatisfiabilităţii oricărei formule din* **LP1**
- **Pasul 1** este un algoritm în sine, formula găsită la sfârşitul execuţiei sale fiind slab echivalentă cu formula iniţială şi având aspectul F = $(\forall^*)F^*$, unde $F^* = C_1 \wedge C_2 \wedge \ldots \wedge C_k$
- Extensia Herbrand generalizată **E'(**F**)** rezultată în urma aplicării **Pasului 2** trebuie interpretată ca fiind o listă de clauze din **LP**
- Datorită faptului că **E'(**F**)** nu este recursivă ci doar recursiv enumerabilă, **Pasul 2** reprezintă un semialgoritm
- După cum se observă, nici n-ar fi nevoie de obţinerea acestei liste dintr-o dată. Este nevoie doar de a putea selecta câte un *nou* element din ea „atunci când este necesar”, conform **Pasului 3.3.2**, care ar putea fi reformulat prin *Obţine un nou element din* **E'(**F**)**

Decidabilitate şi rezoluţie de bază în **LP1 (LP1$_{=}$)** 6

- Practic, pornind de la ordinea pe **D(**F**)** deja sugerată (bazată pe lungimea termilor), se poate defini o ordine totală şi pe **E'(**F**)** (acest lucru nu ar implica decât o simplă extensie a unei relaţii de ordine definită pe o mulţime „suport" la un produs cartezian al acelei mulţimi cu ea însăşi, de oricâte ori)
- De fapt, doar **Pasul 3**, şi numai el, este semialgoritmul cunoscut în literatura de specialitate ca *Procedura lui Gilmore* (sau ***Procedura rezoluţiei de bază***)
- Aceasta nu se termină în cazul în care F este satisfiabilă şi conţine măcar un simbol funcţional de aritate cel puţin 1

Decidabilitate şi rezoluție de bază în **LP1 ($LP1_=$)** 7

Semialgoritmul lui Gilmore

(Procedura rezoluției de bază)

- **Intrare**: Orice formulă F $\in$ **LP1**
- **Ieşire**: „DA”, doar dacă F este nesatisfiabilă
- **Metodă**:

Pasul 1. Se transformă F într-o formulă aflată în **FNSC** (închisă), succesiv, prin rectificare (redenumire), găsirea **FNP** (**FNPR**), obținerea închiderii existențiale, obținerea **FNS** şi apoi **FNSC**, formula rezultat notându-se, pentru simplitate, tot cu F

Pasul 2. Se „obține” **E’(F)** = $\{G_1, G_2, \ldots , G_n, \ldots \}$

Decidabilitate şi rezoluţie de bază în **LP1 ($LP1_=$)** 8

Pasul 3.

3.1. M := Ø

3.2. i := 0

3.3. **Repetă**

3.3.1. i := i + 1

3.3.2. Alege $G_i \in$ **E'(F)**

3.3.3. M := M U $\{G_i\}$

3.3.4. M' := Res*(M)

Până_când ($\square \in$ M')

Pasul 4. Tipăreşte **DA**

Decidabilitate şi rezoluție de bază în **LP1 (LP1$_=$)** 9

- Trebuie însă să arătăm că (semi)algoritmul precedent „face ceea ce dorim”
- Să precizăm de la bun început că vom lua în considerare doar formule F $\in$ **LP1** pentru care **E’(**F**)** este infinită
- În caz contrar, rezultatele obținute până în prezent ne „spun” că F poate fi privită ca o formulă din **LP**, nemaifiind necesară o tratare a acesteia în noul context
- **Teoremă**. Procedura rezoluției de bază pentru **LP1** este corectă

  (demonstrație - schiță)

Decidabilitate şi rezoluție de bază în **LP1 (LP1$_=$)** 10

- **Teoremă (a rezoluției de bază)**. Fie F $\in$ **LP1** şi **E'(**F'**)** extensia Herbrand generalizată a unei formule F', slab echivalente cu F şi aflată în **FNSC** (închisă)
- Atunci F este nesatisfiabilă dacă şi numai dacă există o demonstrație prin rezoluție (în sensul logicii propoziționale) a lui $\square$, pornind cu (o parte finită din) elementele lui **E'(**F'**)**

  (demonstrație - schiță)
- Exemple (rezoluția de bază), exerciții, comentarii

LP1 – Rezoluție pură 1

- **Rezoluția specifică** (numită şi **pură**) pentru **LP1**, deşi diferită de rezoluția de bază, păstrează ideea principală a rezoluției din **LP** şi anume că **la fiecare pas de rezoluție se aleg două clauze şi se obține o altă clauză (rezolvent), eliminând anumiți literali prin „reducere” cu negațiile lor**
- Eliminările devin mai complicate, iar „esența” lor este (sub)procedura de **unificare**

LP1 – Rezoluție pură 2

- A unifica două sau mai multe formule atomice din **LP1** înseamnă a găsi o substituție pentru variabilele care intervin în acele formule (substituția nefiind neapărat elementară sau de bază) astfel încât în urma aplicării substituției formulele atomice respective să coincidă (textual, ca şiruri de caractere)
- Obținerea unui rezolvent nu va însemna numai o simplă reducere a unui literal cu complementarul său, ci şi o „identificare” prealabilă a unor literali

LP1 – Rezoluție pură 3

- Unificarea se „face” cu ajutorul **Algoritmului lui J. Robinson**
- **Exemplu**.Fie:

$F = (\forall x)(\forall y)((\urcorner P(x) \vee \urcorner P(f(c)) \vee Q(y)) \wedge P(y) \wedge (\urcorner P(g(b, x)) \vee \urcorner Q(b)))$, unde

- $C_1 = \urcorner P(x) \vee \urcorner P(f(c)) \vee Q(y)$
- $C_2 = P(y)$
- $C_3 = \urcorner P(g(b, x)) \vee \urcorner Q(b)$

$F^* = \{C_1, C_2, C_3\} = \{\{\urcorner P(x), \urcorner P(f(c)), Q(y)\}, \{P(y)\}, \{\urcorner P(g(b, x)), \urcorner Q(b)\}\}$

Considerăm pe rând următoarele cupluri de clauze:

LP1 – Rezoluție pură 4

-C_1 şi C_2. Din motive tehnice, nu trebuie să existe variabile comune în clauzele considerate în momentul în care încercăm să aplicăm un pas al rezoluției pure

Facem *substituția de redenumire* [y/z] în C_2, găsind $C'_2 = \{P(z)\}$

Prin [x/z], *unificăm* mulțimea $\{\neg P(x), \neg P(z)\}$ (acest din urmă literal fiind complementarul celui conținut de C'_2), găsind

$Res(C_1, C'_2) = \{\neg P(f(c)), Q(y)\} = C_4$

-C_4 şi C_2. Redenumim în C_2 şi lucrăm tot cu C'_2. Aplicând [z/f(c)], vom unifica mulțimea

$\{\neg P(f(c)), \neg P(z)\}$, găsind $Res(C_4, C'_2) =$

$= \{Q(y)\} = C_5$

LP1 – Rezoluție pură 5

-C_3 şi C_2. Nu mai avem nevoie de redenumiri, putând unifica $\{\neg P(g(b, x)), \neg P(y)\}$, prin substitutia $s = [y/g(b, x)]$. Găsim $Res(C_3,C_2) = \{\neg Q(b))\} = C_6$

-C_5 şi C_6. Nu avem nevoie de redenumiri si vom unifica $\{\neg Q(y), \neg Q(b)\}$ prin $s = [y/b]$ şi obținem în final $\square \in Res(C_5, C_6)$

- Astfel, am găsit o respingere utilizând rezoluția pură, pornind cu clauzele lui F, adică demonstrația

  $(\mathcal{D})$: C_1, C_2, C_4, C_5, C_3, C_6, $\square$

LP1 – Rezoluție pură 6

- **Definiție (unificare)**. Fie $\mathcal{L} = \{L_1, L_2, \ldots , L_k\}$ o mulțime finită, nevidă, de literali din **LP1**. Ea se numeşte **unificabilă** dacă există o substituție s astfel încât card(($\mathcal{L}$)s) = 1
- În acest caz, s se numeşte **unificator pentru** $\mathcal{L}$
- O substituție s se numeşte **cel mai general unificator** (**m.g.u.**) pentru o mulțime unificabilă $\mathcal{L}$ dacă orice alt unificator s' se „obține" din s, adică pentru fiecare unificator s' există o substituție **sub** astfel încât s' = s • **sub**

LP1 – Rezoluție pură 7

- Să presupunem acum că avem două clauze distincte din **LP1** (nu neapărat clauze Horn şi conținând eventual şi variabile)
- *Ideea rezoluției pure se bazează pe faptul că putem unifica „cât mai mulți” literali din cele două clauze şi apoi îi putem elimina, într-un mod similar cu rezoluția propozițională*

CURS 12

$\overline{L}_1$

LP1 – Rezoluție pură 8

- **Definiție (rezoluția pură într-un pas, în LP1)**. Fie C_1, C_2 şi R clauze în **LP1**, $C_1 \neq C_2$. R se numeşte **rezolvent (pur)** pentru $\mathbf{C_1}$ şi $\mathbf{C_2}$, obținut **într-un pas**, dacă sunt îndeplinite condițiile:

  (i) Există substituțiile de redenumire s_1 şi s_2 astfel încât $(C_1)s_1$ şi $(C_2)s_2$ nu au variabile comune

  (ii) Există literalii $L_1, L_2, \ldots , L_m \in (C_1)s_1$ şi $L'_1, L'_2, \ldots , L'_n \in (C_2)s_2$ astfel încât mulțimea $\mathcal{L} = \{L_1, L_2, \ldots , L_m \text{ (barate)}, L'_1, L'_2, \ldots , L'_n\}$ este unificabilă. Fie **sub** un cel mai general unificator pentru $\mathcal{L}$

  (iii) $R = (((C_1)s_1 \setminus \{L_1, L_2, \ldots , L_m\})\ \mathrm{U}\ ((C_2)s_2 \setminus \{L'_1, L'_2, \ldots , L'_n\}))\mathbf{sub}$

LP1 – Rezoluție pură 9

- Deoarece nu există pericol de confuzii, vom folosi aceleaşi notații pentru rezoluția pură ca si cele adoptate pentru rezoluția propozițională (ceea ce se schimbă este practic doar definiția rezolvenților obținuți într-un pas)
- **Teoremă (Julia Robinson)**. Orice mulțime $\mathcal{L}$, finită, nevidă, unificabilă, de literali din **LP1**, admite un cel mai general unificator. Problema testării faptului că o mulțime de literali este unificabilă, precum şi găsirea unui cel mai general unificator, sunt decidabile (vezi algoritmul de mai jos)
- Există astfel o metodă relativ simplă pentru unificarea unei mulțimi date de literali
- Ideea este de a *identifica porțiuni de text, având un format special*; acest lucru nu se poate face decât prin intermediul variabilelor, folosind substituțiile
- În plus, „apelurile" recursive, de genul *„x este înlocuit cu t, care conține x"*, sunt interzise (altfel ar fi necesară o procedură de tip *occurrence checking*, care este ineficientă)

LP1 – Rezoluție pură 10
Algoritmul Juliei Robinson

Intrare. Orice mulțime finită, nevidă de literali din **LP1**, $\mathcal{L} = \{L_1, ..., L_n\}$

Ieşire.

-„DA” în caz că $\mathcal{L}$ este unificabilă şi **sub** un m.g.u. (normalizat)

-„NU” în caz că $\mathcal{L}$ nu este unificabilă

Metodă.

Pas1. **sub**:=[ ] {**sub** va fi cel mai general unificator în caz de răspuns afirmativ, iar [ ] reprezintă substituția vidă}

Pas2. j:=0 {j este o variabilă (*flag*) prin care se testează dacă mulțimea respectivă este sau nu unificabilă}

LP1 – Rezoluție pură 11

Pas 3. Cât timp (j = 0 şi card(($\mathcal{L}$)sub)>1) **execută**

{Există L_1, $L_2 \in \mathcal{L}$ a.î. $L_1 \neq L_2$ (textual). Fie i poziția simbolurilor diferite din L_1, L_2 (prima de la stânga la dreapta)}

Pas 3.1. Dacă (nici unul dintre cei doi simboli diferiti de pe poziția i din cei doi literali nu este o variabilă)

atunci j:=1

altfel {fie x simbolul care este o variabilă (să admitem că este din L_1) şi că în L_2 pe poziția i corespunzătoare se găseşte (în mod neapărat) un simbol funcțional de aritate 0 sau mai mare; aceasta înseamnă că la poziția i din L_2 „începe” un term t}

LP1 – Rezoluție pură 12

```
                    Dacă (x apare în t)
                     atunci j:=1
                     altfel
                          sub := sub • [x/t]
                    sf
                sf
         sf
Pas4.  Dacă (j=1)
           atunci
              „NU” {mulțimea inițială nu este unificabilă}
           altfel
              „DA” {mulțimea inițială este unificabilă} şi sub
           sf
```

LP1 – Rezoluție pură 12

- Comentarii asupra CORECTITUDINII algoritmului
- **Teoremă (a rezoluției pure pentru LP1)**.

  Fie F $\in$ **LP1** o formulă închisă, aflată în **FNSC**,

  F = $(\forall_*)$F*. Atunci F este nesatisfiabilă dacă şi numai dacă $\square \in$ Res*(F*), adică dacă şi numai dacă există o demonstrație prin rezoluție pură a clauzei vide (o respingere), pornind cu clauzele lui F

LP1 – Rezoluție pură 13

- Demonstrația se face prin intermediul teoremei rezoluției de bază
- Indexul, exercițiile …