Facultatea de Informatică
Universitatea „Al. I. Cuza” Iaşi
România
(http://www.info.uaic.ro)

LOGICA PENTRU INFORMATICĂ

1-2

- Prof. Dr. Cristian Masalagiu (curs + seminarii)

  mcristy@info.uaic.ro

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

1-3

- Conf. Dr. Ștefan Ciobâcă (curs + seminarii)
- stefan.ciobaca@info.uaic.ro

  http://profs.info.uaic.ro/~stefan.ciobaca

1-4

- Lect. Dr. Andrei Arusoaie (seminarii)
  arusoaie.andrei@info.uaic.ro
  http://profs.info.uaic.ro/~arusoaie.andrei/

1-5

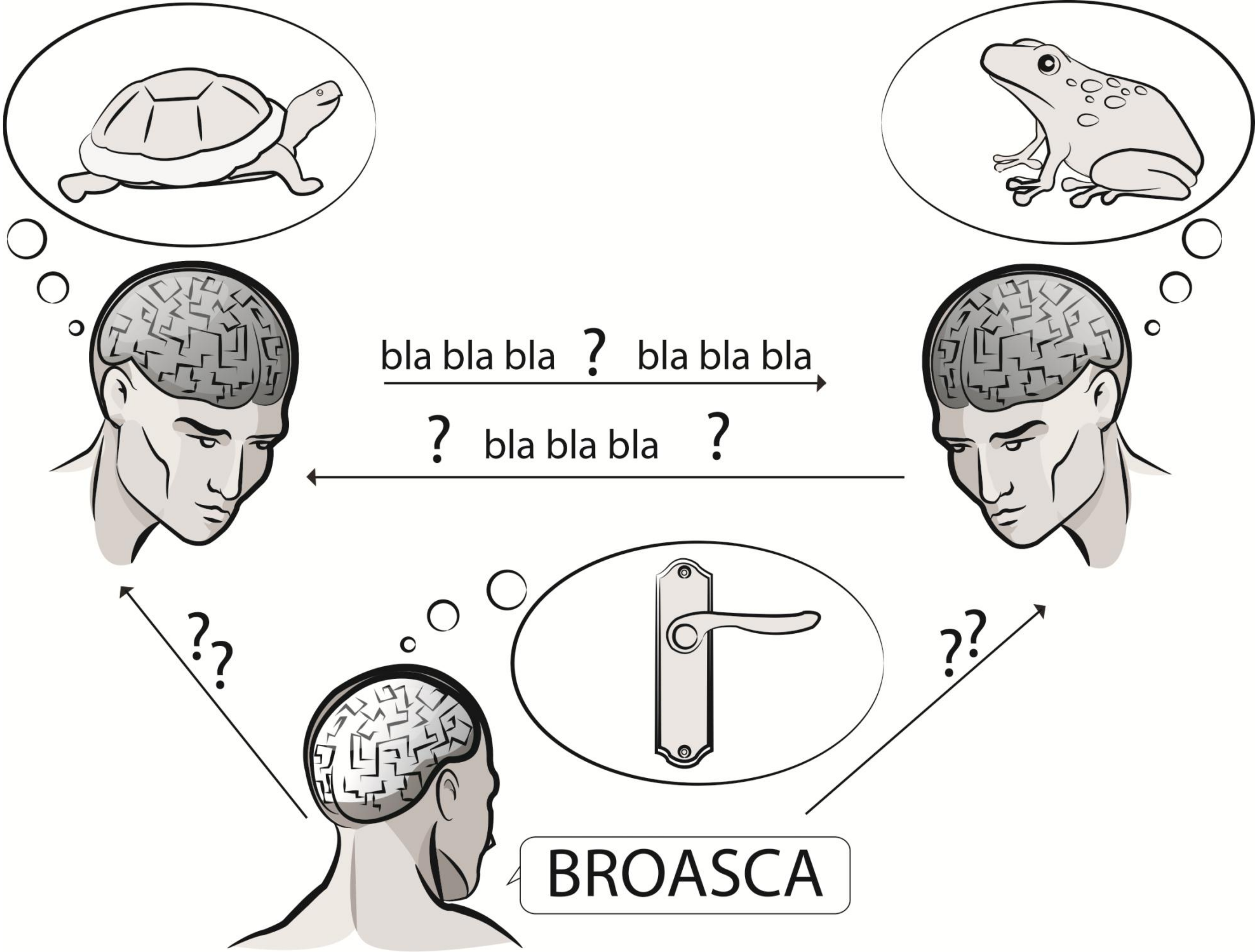
- Asist. Dr. Vasile Alaiba (seminarii)
  alaiba@info.uaic.ro
  http://profs.info.uaic.ro/~alaiba

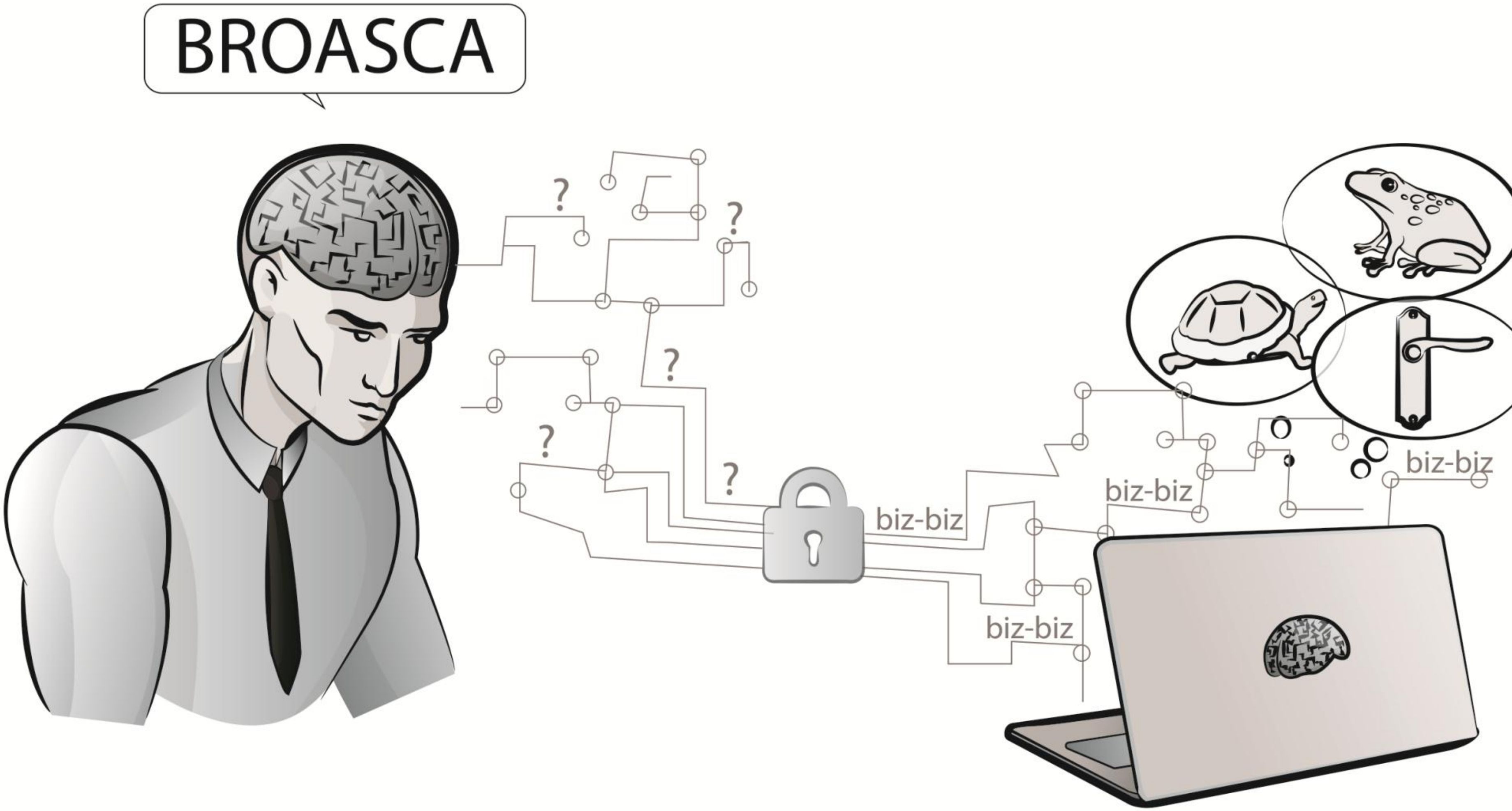
1-6

- Colaborator Radu Rusu
- radu.rusu214@gmail.com
- https://sites.google.com/site/fiiradurusu/

1-7

- **Comentarii**: universitate vs școală, diversitate, independență; dar și constrângeri acceptate; consultații...
- „**Libertatea ta încetează atunci când începe libertatea altuia**” (democracy...)
- **Cum se învață**: învățarea *nu este liniară*, ci un proces complex, de durată, cu reveniri, ramificări, memorări, dar și folosindu-ne imaginația (mai ales când este vorba despre un domeniu nou)
- A căpăta/obține informații (punctuale, gen **GOOGLE**, **Wiki, etc.**), nu înseamnă și a asimila cunoștințe
- „**Success is not final, failure is not fatal: it is the courage to continue that counts.**” (Winston Churchill)
- De predat …predăm (aproape mereu) **cuvinte** (semantica … „realitatea” ...)


bla bla bla ? bla bla bla
? bla bla bla ?
??
??
BROASCA



BROASCA
?
?
?
?
?
biz-biz
biz-biz
biz-biz
biz-biz

Conținut CURS 1

- Logica – este o știință în sine
- Logica - parte a filozofiei
- Logica – parte a matematicii
- Logica – parte a informaticii
- Logica (bivalentă/aristotelică/binară/clasică) – cel mai simplu limbaj de comunicare cu calculatorul (adică de ... programare !)
- Structura generală a Cursului
- Structura semestrului
- Suporturi de informație dedicate vouă (electronice, dar nu numai); alte pagini web
- Evaluare: obținere note/calificative/credite

1-11

Logica - știință în sine:

- Este *Știinţa regulilor generale ale gândirii, cu accent pe aspectele exacte şi de natură structurală ale acesteia* sau, ***Știință a demonstrației, al cărei obiect este stabilirea condițiilor corectitudinii gândirii, a formelor și a legilor generale ale raționării corecte*** (**DEX**online)
- **Realitatea /universul cunoscut:** formată din **obiecte** şi **fenomene** aflate în **relaţii /legături de interdependenţă**
- Realitatea este **dinamică**, orice apariție a unui **eveniment** (echivalent: execuția/derularea unei **acțiuni/activități**), putând schimba realitatea existentă
- În procesul gândirii umane, relaţiile se reflectă „vizual”, apoi prin <u>afirmaţii</u> (*judecăţi*), reformulate apoi în limbaj <u>natural /de discurs</u> (română, engleză ...)

1-12

- La modul cel mai general, orice afirmaţie /propoziție /frază /text, va fi considerată adevărată (**1**) dacă *reflectă în mod adecvat realitatea* şi falsă (**0**) în caz contrar (sensul aristotelic; nimic altceva)
- De asemenea, orice afirmație poate fi **elementară** (**atom**; nu mai poate fi „descompusă” în alte afirmații) sau **compusă**
- Însă „adevărul” (care nici măcar nu este întotdeauna doar negru-alb ...) depinde de emițător, de receptor, de limbajul de discurs; și, în majoritatea cazurilor: de context, de timp etc.
- Limbajul este esențial și orice cuvânt, propoziție, frază, text etc., trebuie prezentat și cercetat din punct de vedere *sintactic* și *semantic*

1-13

- Sintaxa (**DEX**): *Parte a gramaticii care studiază funcțiile cuvintelor și ale propozițiilor în vorbire și care stabilește regulile de îmbinare a cuvintelor în propoziții și a propozițiilor în fraze*
- *Cuvintele* sunt la rândul lor formate din *litere* (care compun *alfabetul* limbii)
- Luând orice limbaj (ex. – lb. română), nu orice secvență de litere formează un cuvânt *corect /admis*
- Cuvintele admise formează *vocabularul* limbii respective
- Elementele de vocabular, împreună cu *spațiul*, *semnele de punctuație* etc. devin litere pentru construcția de propoziții /fraze /texte, corecte lingvistic (sau nu ...)
- Acestea trebuie însă „interpretate” pentru a putea fi folosite în comunicare ...

1-14

- Semantica (**DEX**): *Ramură a lingvisticii care se ocupă cu studierea sensurilor cuvintelor și a evoluției acestor sensuri;* ***teoria interpretării unui sistem formalizat printr-un alt sistem formalizat***
- Prin „analiză semantică de text” putem înțelege: interpretare, semnificație /semantică lingvistică, adevăr lingvistic, adevăr logic /aristotelic (clase de adevăruri…); exemple:

Textul 1 (Pitia)

- *Nu vei muri* (...)

Textul 2

- *Țara mea este mama mea* (...)

1-15

- Deci, din punctul de vedere al oricărei logici, nu ne va interesa sensul lingvistic al textelor acceptate, ci doar cel legat de ceva care, global, ar putea fi numit „adevăr” (a se revedea definiția semanticii din **DEX**)
- Să nu uităm nici de definiția **DEX** a logicii ca știință: nu trebuie doar să stăpânim afirmațiile (textele) din punct de vedere sintactic și semantic, ci și să fim capabili să dezvoltăm raționamente corecte
- Un **raționament corect** este un proces în care, pornind cu niște *afirmații* (vechi), cunoscute a fi *adevărate*, reușim să construim *noi afirmații adevărate*
- Pe parcursul procesului, pentru *păstrarea adevărului*, la fiecare pas de construire a unei noi afirmații se vor folosi *reguli „de deducție” corecte/sound*

1-16

Textul 3

- *Soția unui programator* : *...*

Textul 4

- *Dimineața, tatăl le spune băieților* : *...*

Textul 5

- Admitem că ***timpul*** *înseamnă* ***bani***, că ***munca*** *(„multă”) efectuată într-un* ***timp*** *(cât mai) scurt înseamnă* ***putere*** și că, desigur, *puterea înseamnă (și să dispui de)* ***cunoaștere*** */cunoștințe vaste /profunde* (hm...engleză)
- Deduceți că „*Cei mai bogați oameni sunt cei care nu știu (aproape) nimic, muncind cel mai puțin posibil (dar nu chiar deloc ...)*”

Observație. Regulile de deducție „de folosit” sunt de bun simț și cunoscute de altfel din raționamente matematice uzuale

1-17

Logica - parte a filozofiei (istorie...greci; apoi sec. XIX...)

- Continuând astfel pe aceeași linie, putem spune din nou: logica *Studiază modul de alcătuire și concepere a raţionamentelor corecte, prin care, pornind de la afirmaţii iniţiale (presupuse a fi adevărate) se obţin (utilizând reguli de inferenţă /deducție /derivare) afirmaţii noi (dorite a fi tot adevărate)*
- În context, **logica clasică** (*aristotelică*, *bivalentă,* **0**-**1** etc.) foloseşte doar afirmaţii cărora li se poate asocia în mod *unic* o valoare de adevăr *standard* (independentă de context, moment de timp etc.) şi se bazează în esenţă pe principiul *tertium non datur* (terţiul exclus: *dacă o afirmaţie nu este adevărată, atunci ea este cu certitudine falsă şi reciproc*)
- Nu orice text poate fi considerat a fi „afirmație” (în sensul de mai sus), chiar dacă lingvistic acel text are semnificație/semantică (exemplu clasic imediat: onomatopeele)
- La fel, nu orice text sau chiar rațional corect este lipsit de *confuzii* (vezi exemplele anterioare)

1-18

- Robert Swartz (National Center for Teaching Thinking/S.U.A.; și M.I.T.): **„Aproximativ 90-95% (!) din populația globului nu știe să gândească ... Puțină lume de pe planetă a *învățat* să gândească într-o formă mai largă și mai creativă ... Responsabilitatea revine în special școlii, care (încă) pune accentul pe memorizare și nu pe raționament și rezolvarea creativă a problemelor.”** (aici – specialist **IT** ...)
- Iată câteva dintre noțiunile pe care ar trebui să le stăpâniți deja (din clasa a IX-a ?): *sferă, conţinut, gen proxim, diferenţă specifică, axiomă, teoremă, regulă de inferenţă (de deducție /de demonstrație /de derivare*; de exemplu, *silogismele*, sau regula *modus ponens* (**MP**)), *raţionament* (*deducție /demonstrație /derivare*)
- **Exemple** (comentarii) – revenim la Seminar

1-19

Logica – parte a matematicii

- Mai în profunzime, *Logica matematică*, *formală* (*simbolică*, *abstractă*), *preia problemele logicii filozofice şi le cercetează folosind mijloace specifice*, *punându-se bază pe rigurozitate şi claritate în detrimentul nuanţelor sau intuiţiei*
- În acest context, vorbim despre *limbaje* (*pentru logică /logici*) *precise /formale* prin care se exprimă realitatea în mod direct, similar cu orice limbaj natural: *formule*, *subformule* (sintactic, în loc de cuvinte și texte), *axiome*, *reguli de inferență*, *teoreme*, *demonstrații* (semantic; le știți deja de la matematică), *teorii logice*, *sisteme deductive*, *(meta)teoreme de corectitudine și completitudine* etc.
- Logici *extensionale* și *intensionale*
- **Exemple** – Seminar: exprimare realitate prin formule; *arbori*; *implicația* (*funcții booleene*; „*tabele de adevăr*”); *paradoxuri*; *modus ponens*; *operatori logici* și *priorități*)

1-20

Logica – parte a informaticii

- „Proiectarea” logicii matematice în **Informatică** (privită ca cea mai nouă și, mai ales, dinamică /inovatoare /de „impact” știință), implică o adaptare atât a modului de prezentare a conceptelor/noțiunilor (terminologiei) cât şi a metodelor de demonstraţie, accentul căzând acum pe *constructivism* și *algoritmică*
- Pentru asta ar trebui să stăpâniți (intuitiv, dar și <u>formal</u>) și conceptele: *mulțime* (*finită*, *numărabilă*, *infinită*), *relație*, *grafuri*, *număr cardinal*, *număr ordinal*, *(semi)algoritm* (*pseudocod*, *mașină Türing* etc.), *paradigmă de programare* (*imperativă*, *funcțională*, *orientată pe obiecte*, *logică* etc.), *calculabilitate* și *decidabilitate*, *complexitate* și *tratabilitate*
- **Exemple** (urmează): *definiții constructive /structurale* și *principiul inducției structurale*

1-21

- Din manualele de matematică de liceu sunt bine cunoscute cel puţin două modalităţi de a da o mulţime:

-Prin enumerarea elementelor sale: **N** = {0, 1, 2, ...} este mulţimea numerelor naturale; acest tip de a reprezenta o mulțime este cea mai „potrivită” pentru mulțimile finite, dar „merge” și aici (ce înseamnă „...” ?!)

-Prin specificarea unei proprietăţi caracteristice:

A = $\{x \in \mathbf{R} \mid x^2 + 9x - 8 = 0\}$, este mulţimea rădăcinilor reale ale unei ecuaţii polinomiale de gradul al II-lea; modalitatea de reprezentare este mai „potrivită” pentru mulțimi infinite, chiar *nenumărabile*

- Mai există însă o posibilitate, bazată pe ideea de *constructivism*
- De exemplu, putem defini **N** folosind doar 4 simboluri, să le notăm **0**, **(**, **)**, și **s** (deocamdată, neavând semnificație)

1-22

- Pentru aceasta, sunt necesari 3 pași (din care doar doi sunt „efectivi”):

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”, de fapt textul „**s(***n***)**”, este număr natural).

Nimic altceva nu mai este număr natural (acest pas va fi considerat implicit, la orice definiție constructivă viitoare; *închidere*; *cea mai mică* ...).

- „Traducerea” (semi)algoritmică (pseudocod...)
- Un *prim avantaj* este acela că se poate folosi aceeaşi metodă pentru a introduce alte definiţii, care sunt legate de mulţimea respectivă (aici, **N**), în totalitatea ei
- Putem da astfel o definiţie constructivă/algoritmică (structurală, recursivă...) a *adunării* numerelor naturale („+”; **exemplu**: să se calculeze **2** + **3**):

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

Pas constructiv. *n* + **s(***m***)** = **s(***n* + *m***)**, pentru fiecare *n*, *m* $\in$ **N** (dacă ştim să calculăm *n* + *m* şi cunoaştem succesorul imediat al numărului natural *m*, atunci ştim să calculăm şi suma *n* + **s(***m***)**; mai exact, aceasta coincide cu succesorul imediat al numărului care reprezintă suma *n* + *m*).

- În acest moment, **s(***n***)** se poate nota și cu *n* + **1**...

1-23

- Un *al doilea avantaj*, poate cel mai important, este posibilitatea folosirii în demonstraţii a *principiului inducţiei* (*matematice*, în cazul lui **N**, deocamdată)
- Astfel, dacă vrem să arătăm că o anumită proprietate, notată „P(n)”, este adevărată pentru fiecare $n \in$ N (adică $(\forall n)(\mathrm{P}(n))$), folosind principiul amintit vom arăta că este adevărată afirmația/„formula”

  $\mathrm{P}(\mathbf{0}) \wedge ((\forall n)(\mathrm{P}(n) \to \mathrm{P}(n + \mathbf{1}))$; nu totdeauna sunt echivalente...)
- **Comentarii** (alte forme de inducție, formule, ordine; nu există comutativitate sau proprietăți ale egalității;

  $\mathbf{0} + n = n$ se demonstrează; vezi seminar ...)
- Revăzând definiția constructivă a lui **N**, cele de mai sus se pot generaliza pentru definirea altor mulțimi

1-24

- De exemplu, putem defini constructiv (construi algoritmic!) orice mulțime **M**, *numărabilă* (*intuitiv* ...*cardinalitate*)
- Procesul va avea cei 2/3 pași amintiți; începem cu **M** vidă
- În pasul (*inițial*), **Baza**, se introduc (*explicit*) în **M** un număr oarecare de elemente „de bază” („grupate” în mulțimea **M’**)
- În **Pasul constructiv**, se repetă (*de câte ori este posibil*) un/niște procedeu/e de introducere de elemente *noi* în **M**, folosindu-ne de elementele *vechi*, deja existente (cu ajutorul unor *algoritmi/metode* bine precizați/e)
- **M’** (nevidă) este cunoscută/construită dinainte, ca de altfel și mulțimea **O**, de „algoritmi”
- Fiecare algoritm ***o*** $\in$ **O** este privit în sens *determinist*, *funcțional*: aplicat „intrării” $<m_1, m_2, \ldots, m_k>$, va furniza (unica) „ieșire” *m*

1-25

Definiția structurală a unei mulțimi numărabile M

Baza (*elemente inițiale*). **M'** $\subseteq$ **M** (**M** *conține elementele de bază/inițiale*).

Pas constructiv (*elemente noi din elemente vechi*). Pentru fiecare $k \in$ **N***, pentru fiecare

$m_1, m_2, ..., m_k \in$ **M** și pentru fiecare ***o*** $\in$ **O** (*operator de aritate k*), avem ***o***($<m_1, m_2, ..., m_k>$) = $m \in$ **M**.

Nimic altceva nu mai este element al lui M (adică *singura posibilitate de a obține elemente noi pentru a fi „puse în”* **M**, *este de a aplica algoritmii din* **O**).

- „Traducerea” algoritmică...; [*n*] este notația *ordinală* a mulțimii {1, 2, ..., *n*}); *pairing functions*; revenind la definiția primară a lui **N**: *n* denotă **s(s(**...**(s(**0**))**...**)**

1-26

- Fie acum **M** orice mulțime definită structural ca mai sus (cu ajutorul lui **M’** și **O**) și o afirmație generală de tipul **Q** = ($\forall m$)(**P**(m)), adică proprietatea **P** „privește” întreaga mulțime **M** ($\forall m \in$ **M** ...)
- Fie și afirmația **Q’**, *corespunzătoare definiției structurale a lui* **M**, dată prin **Q’** = ($\forall a \in$ **M’**)(**P**(a)) $\wedge$ ($\forall k \in$ **N***)($\forall m_1, m_2, ..., m_k \in$ **M**)($\forall$***o*** $\in$ **O**)
  (**P**(m_1) $\wedge$ **P**(m_2) $\wedge$ ... $\wedge$ **P**(m_k) $\rightarrow$ **P**(m))
  (unde m = ***o***($<m_1, m_2, ..., m_k>$))

Principiul general al inducţiei structurale

- **Q** este adevărată dacă putem arăta **Q**’, adică:

Baza. **P**(a) este adevărată, pentru fiecare $a \in$ **M’** (*adevărul lui* **P**, *pentru elementele de bază*).

Pas inductiv. Presupunem că sunt adevărate **P**(m_1), **P**(m_2), ..., **P**(m_k). Atunci, arătăm că **P**(m) este adevărată (*presupunând că* **P** *este adevărată în elementele vechi*, *arătăm că* **P** *este adevărată și în elementele noi*).

- Acest ultim pas trebuie demonstrat pentru fiecare $k \in$ **N***, pentru fiecare $m_1, m_2, ..., m_k \in$ **M** și pentru fiecare ***o*** $\in$ **O** (de aritate k) care satisface ***o***($<m_1, m_2, ..., m_k>$) = m

1-27

- Ideea este similară cu cea folosită în cazul **N**: *în loc să arătăm* **Q**, *vom arăta* **Q'** (*principiu* vs *teoremă*)
- Revenind la logicile „informatice", ele (vezi **LP** care urmează) pot fi văzute ca limbaje de programare, și ca limbaje „naturale, exacte" (sintaxă + semantică formală)
- Modelează realitatea, ca sintaxă (formulă „=" program)
- Modelează realitatea și ca semantică generală/valoare de adevăr
- Semnificația unui program este dată (în sens *imperativ*, *operațional*) de *execuțiile* sale: pentru intrarea *x*, se obține ieșirea *y* (prin efectuarea operațiilor indicate de textul programului, în ordinea precizată, asupra valorilor introduse)
- Semnificația unei formule va fi dată, similar, de valorile de adevăr „finale"/„de ieșire", obținute în urma *aplicării* operatorilor logici prezenți în formulă, în ordinea fixată, valorilor de adevăr specificate ca „intrare" (și celor intermediare)

1-28

Sintaxa logicii propoziționale (LP); definiție constructivă

- Fie o mulţime de *variabile propoziţionale* (sau: *formule elementare*, *formule atomice pozitive*, *atomi pozitivi*),

  A = $\{A_1, A_2, \ldots\}$ (**alfabet** numărabil de „litere” /nume generice /constante)
- Fie **C** = {⌉, ∨, ∧} mulţimea *conectorilor logici* (*conectivelor logice*)*: non* (**negaţia**), *sau* (**disjuncţia**), respectiv *şi* (**conjuncţia**)
- Fie **P** = {**(** , **)**} mulţimea *parantezelor rotunde*
- **Formulele** (elementele lui **LP**) vor fi *cuvinte* (*expresii bine formate* – **w**ell **f***ormed* **f***ormulae* /**wff**) peste *alfabetul extins* **L** = **A** ⋃ **C** ⋃ **P** (semne de punctuație...)
- Atunci, mulțimea de formule **LP** va fi *construită* astfel:

1-29

Baza (*formulele elementare sunt formule*): $\mathbf{A} \subseteq \mathbf{LP}$.

Pas constructiv (*obţinere formule noi din formule vechi*, *folosind conectorii*):

(i) Dacă $F \in \mathbf{LP}$ atunci $\mathbf{(}\rceil F\mathbf{)} \in \mathbf{LP}$.

(ii) Dacă $F_1, F_2 \in \mathbf{LP}$ atunci $\mathbf{(}F_1 \vee F_2\mathbf{)} \in \mathbf{LP}$.

(iii) Dacă $F_1, F_2 \in \mathbf{LP}$ atunci $\mathbf{(}F_1 \wedge F_2\mathbf{)} \in \mathbf{LP}$.

(iv) Dacă $F \in \mathbf{LP}$ atunci $\mathbf{(}F\mathbf{)} \in \mathbf{LP}$.

(**Nimic** altceva ...) .

- **Exemple**: Arb(F), subf(F), prop(F)) (revenim; exercițiile ...)
- Pentru a furniza și **semantica** (**formală**), mai trebuie să „muncim”: „**There is no elevator to success. You have to take the stairs!**”
- Dar, în definitiv, mai mereu, „**Nu *atingerea scopului fixat* este cel mai important lucru, ci *puterea și modaliatea de a parcurge tot drumul* până acolo...**”
- Din ce urmează, mai citiți și voi ...

1-30

- Să terminăm justificarea necesității, pentru un informatician, de a studia logica într-un mod formal (deși ... orice „bucată” hard sau soft este o „bucată de **0**-**1**”...)
- Există sisteme reale care nu pot fi *proiectate* (darămite *create* și *utilizate*...) fără a ști **aprioric**, **cu certitudine,** că ele **vor funcționa conform specificațiilor**
- Acestea sunt așa-numitele **safety critical systems** (există în medicină și sănătate, în domeniul militar, în domeniul economic și bancar etc.)
- Se pot desigur folosi (și) tehnici de modelare, simulare, „baterii” de teste, previziuni statistice etc.
- Acestea nu sunt, în multe cazuri, suficiente (*dacă* sunt posibile); ba, sunt chiar nesigure uneori, având nevoie la rândul lor de o **verificare formală prealabilă**
- **Exemple** (din realitate ...)

1-31

- **Orice limbaj în care se fac asemenea verificări, este în totalitate (sau „aproape”) bazat pe logică**
- Am putea scăpa de logică (deși, nu complet, nu?!), dacă am putea stoca totalitatea informațiilor prin care s-ar putea descrie Universul actual (trecut, prezent, viitor)
- **Se știe** (întrebați-l pe Stephen Hawking, de ex.) că, presupunând că avem nevoie de o unitate de informație pentru a descrie un singur atom, pentru întregul univers este nevoie de 10 la puterea (10 la puterea 123) asemenea unități !!!

1-32

- Însă, ținând cont că trebuie făcute și niște *măsurători* pentru a obține aceste informații, că avem nevoie și de o aparatură specializată și că spațiul „nostru” este finit (asta pentru a nu mai implica și timpul...), dacă am „înghesui” numai aparatura de măsurare în spațiul de care dispunem, *colapsul „în el* însuși” al Universului (gen *gaură neagră*) s-ar produce după stocarea prezumptivă a 10 la puterea (10 la puterea 90) unități
- Dând și alt exemplu, doar pentru memorarea informațiilor care ar descrie complet o picătură de apă, ar fi nevoie de $2 \cdot 10^{20}$ octeți (de aceea...avem treabă și mâine, și ...)

Structura Cursului (principalelele capitole tematice; poate le mai „amestecăm”):

- Logica și Informatica („făcut”)
- Logica propoziţională (**LP**; început ...)
- Algebre booleene și **(O)BDD**-uri
- Teorii logice și Sisteme deductive (în **LP**)
- Logica cu predicate de ordinul I (**LP1**)
- Teorii logice și Sisteme deductive (în **LP1**)
- **Programare logică**, **Verificare**, **Demonstrare automată**, **Logici neclasice**, **Inteligență artificială**, **Securitate** (nu „apucăm” ...)

Structura semestrului, suportul de informație, evaluare (citiți ...)

- Sunt, în anul universitar 2016-2017, semestrul I, 16 săptămâni de „şcoală”; săptămânile a 8-a şi a 15-a (sau a 16-a...) sunt destinate lucrărilor de evaluare (se dau la Logică)
- Perioadele implicate sunt 03.10.2016–23.12.2016 (12 săptămâni de ore și evaluare), apoi 24.12.2016 – 08.01.2017 (vacanță), 09.01.2017–22.01.2017 (alte 2 săptămâni de ore), 23.01.2017-05.02.2017 (evaluare și/sau ore), 06.02.2016-19.02.2017 (vacanță, restanțe/măriri, licență)
- Semestrul II începe deci pe 20.02.2017

1-35

- Recuperările de cursuri (sau de orice altă natură) vor fi anunţate pe parcurs, ca și schimbările de orar
- INFORMAŢI-VĂ PERMANENT (în special: pagina Facultății, paginile web personale ale celor cu care lucraţi, etc.)
- Regulamente, orar, dificultăți/avantaje la învățat (Logica ...)
- Nota finală se obţine în urma acumulării unui număr de puncte (maxim **100**) şi în urma aplicării unei ajustări de tip Gauss conform Regulamentului intern al Universității și al Facultății (aflate în vigoare)
- Nota și clasificarea se obțin practic direct (împărțire la 10 + eventuale rotunjiri)

1-36

- Cele **100 de puncte** se pot obţine astfel (mai concret sau schimbări – discuții la Seminar):
  - **10 p** prezenţa la seminarii: de exemplu, **4** verificări ale prezenţei, realizate aleatoriu, fiecare a câte **2,5** p
  - **30 p** pentru activitatea continuă la seminarii: de exemplu, rezolvarea de exerciţii (ieşiri la tablă), participarea la lucrări scrise (neanunţate), ...
  - **60 p** (maxim) pentru lucrările (2) de verificare a cunoştinţelor de la mijlocul și finalul semestrului (**30** + **30**): în mare, câte 5 subiecte a câte 6 pct.
- Restanță, mărire ...

1-37

- Promovarea este asigurată de un punctaj minim total de **45 p**
- Cumulat, la lucrările de verificare este necesară obţinerea a minim **30 p**
- A se consulta (deja am menţionat) măcar pagina http://profs.info.uaic.ro/~masalagiu (**Logic**)
- Pentru legătura cu **creditele** (la **Logică** sunt **6**) și promovabilitatea generală – citiți Regulamentele din paginile Facultății/Universității

BIBLIOGRAFIE TIPĂRITĂ

- **Masalagiu, C.** - ***Fundamentele logice ale Informaticii*** , Editura Universităţii „Al. I. Cuza”, Iaşi, 2004,

  ISBN 973-703-015-X (o avem și în **format electronic**)
- **Masalagiu, C.** - ***Introducere în programarea logică şi limbajele de programare logică***, Editura Universităţii

  „Al. I. Cuza”, Iaşi, 1996 (nu prea există)
- **Cazacu, C., Slabu, V.** - ***Logica matematica*** , Editura Ştefan Lupascu, Iaşi, 1999, ISBN 973-99044-0-8 (nu ...)
- **M. Huth, M. Ryan -** ***Logic in Computer Science: Modelling and Reasoning about Systems***, Cambridge University Press, England, 2000, ISBN 0-521-65200-6 (o avem și în **format electronic**)

- Desigur că puteți folosi orice alt text din domeniu pe care îl puteți accesa (inclusiv titlurile menționate de mine pentru secția „Engleză”); materia însă ...; carte nouă ...

1-39

- Bibliografie principală (pe parcurs, posibil, se vor mai adăuga alte titluri și link-uri):

  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 secţiunea **Logic** - *Bibliografie de bază*; unele link-uri pot fi accesate doar din laboratoarele **FII**)
Sugestie: Învăţaţi **<u>şi</u>** după (alte) cărți /notițe /seminarii, nu numai după slide-urile /link-urile furnizate în **format electronic**!!!

Link: http://www.info.uaic.ro/~orar

- Ore de consultaţii (**anunţaţi în prealabil** participarea): conform Orarului
- Alte link-uri „de urmărit”

http://www.info.uaic.ro -> Membri (**Members**) -> Personal Academic

http://www.info.uaic.ro -> Studenţi (**Students**) -> Regulamente

http://profs.info.uaic.ro/~masalagiu/ -> Secţiunea **Logic** (de fapt, nu doar asta, de aici...)

1-41 (final 1)

- Citiți cu atenție ***toate*** slide-urile aferente cursului 1 (sunt 41, cu acesta inclusiv)
- ***Important de reținut***:

-Limbaj „natural” vs limbaj „formal”

-**Definiții structurale** /constructive /recursive /inductive ale mulțimilor cel mult numărabile („primare” – mulțimea însăși; „ulterioare”)

-**Principiul inducției structurale**

-Definiția structurală a **sintaxei** logicii propoziționale (limbajul formal **LP**)

2-1 (42)

Continuare sintaxă LP

- *Arbori*, *subformule*, *apariție* „A în F” (definiții constructive; notații: Arb(F), subf(F), prop(F)), altele ...)
- $((\rceil F) \vee G)$ 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** (noi simboluri; multe/puține...; paranteze, asociativitate, comutativitate, ...)

2-2 (43)

- 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 cu **Ā** mulțimea $\{\rceil A_1, \rceil A_2, \ldots\}$)
- Dacă L este un literal (adică $L \in$ **A** $\cup$ **Ā**), atunci **complementarul** său, $\overline{L}$, va denota 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)

2-3 (44)

Semantica generală (0-1) a LP (n-am terminat însă complet cu sintaxa)

- *Semantica (înţelesul)* unei formule propoziţionale este, conform principiilor logicii aristotelice, o valoare de adevăr **a**devărat ($\triangleq$ **1**) sau **f**als ($\triangleq$ **0**), obţinută în mod determinist și independentă de (orice alt) context
- De fapt, vom „lucra” cu *algebra booleană* $\mathcal{B}$ = < **B**, •, +, $\overline{\ }$ >, **B** = {**0**, **1**} (vom reveni: operații booleene ... tabele de „adevăr”...)
- Noţiunea principală este cea de ***asignare*** **(*interpretare*, structură)**
- **Definiţie.** Orice funcţie **S**, **S** : **A** $\rightarrow$ **B** se va numi *asignare*.

2-4 (45)

- **Teoremă (de extensie)**. Pentru fiecare asignare **S** există o *unică extensie* a acesteia, **S'** : **LP** $\rightarrow$ **B** (numită în continuare *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**.

(inducție **constructivă**: în ***metalimbaj*** arătăm ($\forall$F)(P(F)); P(F): ($\forall$**S**)($\exists$!**S'**)(**S'** extinde **S și** ... vezi (i)-(v) de mai sus); de fapt, se arată doar unicitatea: P(A) **și** ...

2-5 (46)

- Vom folosi de acum **S** (în loc de **S’** etc.)
- **Definiţie**. O formulă F $\in$ **LP** se numeşte *satisfiabilă* dacă *există măcar* o structură **S** (***corectă*** pentru ...; prop(F) ...) pentru care formula este adevărată (**S**(F) = **1**); se mai spune în acest caz că **S** este model pentru F (simbolic, se scrie **S** $\models$ F). O formulă este *validă* (*tautologie*) dacă *orice* structură este model pentru ea. O formulă este *nesatisfiabilă* (*contradicţie*) dacă este falsă în *orice* structură (**S**(F) = **0**, pentru fiecare **S**; sau **S** $\not\models$ F).

2-6 (47)

- **Teoremă**. O formulă F $\in$ **LP** este validă dacă şi numai dacă ($\urcorner$ F) este contradicţie. (**demonstraţie**)
- Clasa tuturor formulelor propoziţionale **LP** este astfel partiţionată în trei mulţimi *nevide şi disjuncte*: **tautologii** (formule **valide**), formule **satisfiabile** (dar nevalide), **contradicţii** (formule nevalide); **desen** „oglindă”: F, (G, $\urcorner$G), $\urcorner$F…
- Problema **SAT** este rezolvabilă în timp exponențial (revenim ...)

2-7 (48)

- **Definiţie**. Două formule $F_1, F_2 \in \mathbf{LP}$ se numesc *tare echivalente* dacă *pentru fiecare* structură **S** ele au aceeaşi valoare de adevăr, adică $\mathbf{S}(F_1) = \mathbf{S}(F_2)$ (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* $\mathbf{S_1}(F_1) = \mathbf{1}$, *atunci* există $\mathbf{S_2}$ *astfel încât* $\mathbf{S_2}(F_2) = \mathbf{1}$ şi reciproc).

2-8 (49)

- **Definiție**. O formulă F $\in$ **LP** este *consecinţă semantică* dintr-o mulţime (nu neapărat finită) de formule **G** $\subseteq$ **LP**, dacă: *pentru fiecare structură corectă* **S**, *dacă* ***S*** *satisface* ***G*** (adică avem **S**(G) = **1** pentru fiecare G $\in$ **G**) *atunci* **S** *satisface* F (simbolic, vom scrie **G** $\models$ F).
- **Teoremă**. Fie G $\in$ **LP** şi **G** = { G_1, G_2, …, G_n } $\subseteq$ **LP**. Următoarele afirmaţii sunt echivalente:
- (i) G este consecinţă 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 contradicţie.

(**demonstraţia** – idei)

2-9 (50)

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

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

  (a') $F \vee F \equiv F$. (*idempotenţă*)

  (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') $F \vee (F \wedge G) \equiv F$. (*absorbţie*)

2-10 (51)

(f) $\neg\neg F \equiv F$. (*legea dublei negaţii*)

(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 validităţii*; adevărate ***doar dacă*** F este tautologie)

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

(i') $F \vee G \equiv G$. (*legile contradicţiei*; adevărate **doar dacă** F este contradicţie)

(**demonstraţie** parțială)

- Generalizări pentru mai multe formule; idee *dualitate* (mai revenim …)

2-11 (52)

- **Teoremă (de substituţie)**. 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 obţinută din H prin înlocuirea (unei apariţii fixate a) lui F cu G. Atunci $H \equiv H'$.

(**demonstraţie** prin *inducție structurală* în metalimbaj: $(\forall H)(P(H))$; $P(H)$: $(\forall F)(\forall G)(\forall H')$ (**Dacă** F este subformulă a lui H **și** H' se obține din...**și** $F \equiv G$, **atunci** $H \equiv H'$))

- Urmează câteva definiții și rezultate „combinate" (sintaxă + semantică)

Forme normale

- Vom studia **simultan** *formele normale conjunctive* şi *formele normale disjunctive*
- **Definiţie**. O formulă F $\in$ **LP** se află în **formă normală conjunctivă** (**FNC**, pe scurt) dacă este o *conjuncţie de disjuncţii de literali*, adică o *conjuncţie de clauze*; respectiv,

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

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

- În cele de mai sus $L_{i,\ j} \in$ **A** U **Ā**.

2-13 (54)

- **Teoremă (existență forme normale)**. Pentru fiecare formulă $F \in$ **LP** există cel puţin 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).

(**demonstraţie** – idei: $(\forall F)(P(F))$; $P(F)$: $(\exists F_1)(\exists F_2)(F_1$ este în **FNC și** F_2 este în **FND și** $F_1 \equiv F$ **și** $F_2 \equiv F)$; F – o privim ca arbore...)

- Conform teoremei anterioare, precum şi datorită comutativităţii şi idempotenţei disjuncţiei, comutativităţii şi idempotenţei conjuncţiei (repetarea unui element, fie el literal sau clauză, este nefolositoare aici), este justificată scrierea ca mulţimi a formulelor aflate în **FNC**

2-14 (55)

- Astfel, dacă F este în **FNC**, vom mai scrie

  $F = \{C_1, C_2, \ldots, C_m\}$ (virgula denotă ...)
- Fiecare clauză C_i poate fi la rândul ei reprezentată ca o mulţime, $C_i = \{L_{i,1}, L_{i,2}, \ldots, L_{i,ki}\}$, $L_{i,j}$ fiind literali (virgula denotă acum ...)
- Mai mult, dacă avem $F \in$ **LP** reprezentată ca mulţime (de clauze) sau ca mulţime de mulţimi (de literali), putem elimina clauzele C care conţin atât L cât şi complementarul său, $\overline{L}$, 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**; contradicțiile - cu **0**)
- Iar tautologiile componente nu au nici o semnificaţie pentru stabilirea valorii semantice a unei formule F aflate în **FNC**

  $(\mathbf{1} \wedge C \equiv C)$

2-15 (56)

- Importanța formelor normale (și a echivalențelor semantice existente) rezultă imediat, gândindu-ne la *standardizare*: este posibil să folosim structuri de date și algoritmi unici pentru întregul **LP**, deși formulele ar putea fi scrise în moduri foarte diverse
- Vom exemplifica acest lucru pentru cazul problemei **SAT** (revenim cu amănunte !), atât după tratarea *rezoluției* cât și după studiul *funcțiilor booleene*
- Mai întâi, ne ocupăm de **SAT** pentru o clasă *mai restrânsă* de formule

2-16 (57)

- **Definiţie**. O **formulă Horn** este o formulă aflată în **FNC**, clauzele componente fiind (toate) clauze Horn (conțin cel mult un literal pozitiv).
- Mai jos, A_i, B etc., sunt toate elemente ale lui **A**, nu din **Ā**
- Vom numi (tot) formulă Horn (şi) o formulă care este (tare) echivalentă cu o formulă având forma considerată în definiţia precedentă
- În afară de reprezentarea ca mulţimi, clauzele Horn pot fi reprezentate şi sub aşa-numita **formă implicaţională**
- Vom distinge cazurile („$\triangleq$” înseamnă „egal prin convenţie”, sau „prin definiție”):

-C = A $\in$ **A**; aceasta se mai poate scrie sub forma C $\triangleq$ **1** $\rightarrow$ A, deoarece **1** $\rightarrow$ A $\triangleq$ $\rceil$**1** $\vee$ A $\equiv$ **0** $\vee$ A $\equiv$ A

2-17 (58)

-C = $\rceil A_1 \vee \rceil A_2 \vee \ldots\ldots \vee \rceil A_k$; vom putea scrie
C $\triangleq A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_k \rightarrow$ **0** (folosim din nou definiţia implicaţiei, apoi legile lui deMorgan şi faptul că **0** $\vee$ A $\equiv$ A)

-C = $\rceil A_1 \vee \rceil A_2 \vee \ldots \vee \rceil A_k \vee B$; atunci avem
C$\triangleq A_1 \wedge A_2 \wedge A_3 \ldots \wedge A_k \rightarrow B$, direct din definiţia implicaţiei și „deMorgan”

- Din motive tehnice, admitem și C $\triangleq$ □ (clauza fără *niciun literal*); este ***clauza vidă*** („petit carré”...; în reprezentarea cu mulțimi va fi denotată prin Ø)
- Prin convenţie, □ *este o clauză de orice tip* (inclusiv o clauză Horn), dar **nesatisfiabilă**
- **Teoremă**. Satisfiabilitatea formulelor Horn este decidabilă în timp liniar.

(**demonstraţia** se va baza pe următorul algoritm imperativ):

2-18 (59)

Algoritm Horn

Intrare: Orice formulă Horn, F, reprezentată ca mulţime de clauze, clauzele componente fiind clauze Horn diferite de clauza vidă şi scrise sub formă implicaţională (putem elimina aprioric și „tautologiile depistabile sintactic”).

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 (adică, F nu este satisfiabilă).

- **Observație**. Inițial, toate variabilele care apar se consideră a fi nemarcate. Dacă în F nu există clauze de forma $\mathbf{1} \rightarrow B$, atunci F este satisfiabilă și **S** este **0** pentru fiecare atom din prop(F) (corpul buclei nu se execută niciodată). Clauza $\mathbf{1} \rightarrow B$ se consideră a fi de forma „$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k$”$\rightarrow B$, cu $A_1, A_2, A_3, \ldots, A_k$ *marcaţi* și B nemarcat”. Orice marcare a unui nou literal (pozitiv) înseamnă modificarea valorii lui **S** (pentru acel literal), din **0** în **1**.

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 $A_1, A_2, A_3, \ldots, A_k$ *marcaţi* şi B *nemarcat*, sau de forma $A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \rightarrow \mathbf{0}$, cu $A_1, A_2, A_3, \ldots, A_k$ *marcaţi*) **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

2-20 (61)

Pasul 7. **Dacă** (i = 0)

atunci

Pașii 8-9. Scrie „**DA**” și **S** (**S**(A) = 1 dacă şi numai dacă A apare în F şi este marcat)

altfel

Pasul 10. Scrie „**NU**”

sf_Dacă

- Trebuie să **demonstrăm** *corectitudinea* și *terminarea* algoritmului (idei și **exemplu** la seminar; începem cu **1** $\rightarrow$ B...)

2-21 (62 – final 2)

- Primele 15 slide-uri din acest curs 2 trebuie citite foarte atent (cursul începe la slide 42)
- Ultimele 5 slide-uri (de la 2-16 la 2-20), privind **SAT** pentru formulele Horn (inclusiv **Algoritmul Horn**) pot fi „sărite” pe moment (reluare: după **DP** /**DPLL**)
- ***Important de reținut***:

-Definiția (structurală) pentru Arb(F), subf(F), prop(F)

-Definiția unei *structuri* (Teorema de existență a *extensiei unice*)

-Formule *satisfiabile*, *valide*, *contradicții* (Teoremă)

-*Echivalență* tare și slabă (Teoremă)

-*Consecință semantică* (Teoremă)

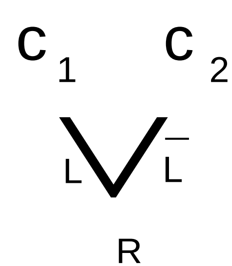
-Teorema de *substituție*

-Forme normale (*clauze*; **FNC**, **FND**)

3-1 (63)

Începem studiul *rezoluției* pentru LP

- Fără a restrânge generalitatea, putem presupune că lucrăm cu formule din **LP** aflate în **FNC**, reprezentate sub formă de mulţimi (finite) de clauze, iar clauzele ca mulţimi (finite) de literali
- **Definiţie (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 obţine prin rezoluţie *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 (1-rezoluția) şi grafic, prin *arborele de rezoluţie*:



3-2 (64)

Observații:

- Rezolventul a două clauze este este tot o clauză (mai mult, rezolventul a două clauze Horn este tot o clauză Horn)
- Clauza vidă (□) poate fi obţinută prin rezoluţie din două clauze de forma $C_1 = \{A\}$ şi $C_2 = \{\rceil A\}$
- Dacă în definiţia anterioară C_1 şi C_2 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, neinteresante d.p.d.v. al satisfiabilității și detectabile, *sintactic*, *aprioric*
- De asemenea vom „rezolva” doar clauzele în care literalul L cu acea proprietate este unic (pentru că...)

3-3 (65)

- **Teoremă (lema rezoluţiei)**. Fie orice formulă $F \in$ **LP** (aflată în **FNC** şi reprezentată ca mulţime de clauze) şi R un rezolvent pentru C_1, $C_2 \in F$. Atunci F este tare echivalentă cu $F \bigcup \{R\}$.

(**demonstraţie**: arătăm că pentru fiecare structură corectă **S**, dacă **S**(F) = **1**, atunci **S**($F \bigcup \{R\}$) = **1** și reciproc)

- În teorema anterioară am fi putut considera, în loc de F, o mulţime oarecare de clauze, chiar infinită, adică numărabilă (vezi **Teorema de compactitate**, care va urma ...)

3-4 (66)

- **Definiţie**. Fie F o mulţime oarecare de clauze din **LP** şi C o (altă) clauză. Spunem că lista C'_1, C'_2 , … , C'_m este o **demonstraţie prin rezoluţie (*în mai mulţi paşi*) a lui C pornind cu (bazată pe) F** dacă sunt satisfăcute condiţiile:

  (i) Pentru fiecare $i \in [m]$, fie $C'_i \in F$, fie C'_i este obţinut prin rezoluţie într-un pas din C'_j, C'_k, cu j, k < i și (desigur) $j \neq k$.

  (ii) $C = C'_m$.

3-5 (67)

- În condiţiile definiţiei, se mai spune că **C este demonstrabilă prin rezoluţie** în mai mulți pași, *pornind cu /bazată pe* F (sau, **în ipotezele date de** F); <u>1-rezoluția „=” regulă de inferență</u>
- Pentru a spune acest lucru, este suficient ca F să poată fi *inserată* (prezentă) într-o demonstraţie şi nu să fie neapărat ultimul element al acesteia
- Intuitiv, o demonstraţie prin rezoluţie în mai mulţi paşi înseamnă o succesiune finită de rezoluţii într-un pas, care poate fi reprezentată şi grafic (printr-un arbore, sau chiar un graf oarecare ... desen; fracții „supraetajate”; mai revenim ...)

3-6 (68)

- În particular, dacă C este clauza vidă, atunci demonstraţia respectivă se numeşte şi **respingere**
- **Numărul de paşi** dintr-o demonstraţie bazată pe F este dat de *numărul de clauze obţinute prin rezoluţie într-un pas* (din clauze anterioare), la care se adaugă de obicei și numărul de clauze din F folosite în demonstrație
- Acesta poate fi considerat ca fiind o *măsură* a „mărimii” (*lungimii*) demonstraţiei
- O (altă) măsură pentru o demonstraţie reprezentată ca text poate fi chiar lungimea listei (numărul total de clauze sau chiar numărul total de clauze distincte)
- Dacă reprezentăm o demonstraţie ca un arbore, putem folosi şi măsuri specifice, cum ar fi *adâncimea /înălțimea* arborelui, *numărul de niveluri,* etc.

3-7 (69)

- **Definiţie (mulţimea rezolvenţilor unei mulţimi de clauze)**. Fie F o mulţime de clauze din **LP** (nu neapărat finită). Notăm succesiv:

  -Res(F) = F U {R | există C_1, $C_2 \in$ F astfel încât R = Res(C_1, C_2)}.

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

  -$\mathrm{Res}^{(0)}(F)$ este o altă notație pentru F şi atunci vom avea şi

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

  Mai mult, vom pune:

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

- $\mathrm{Res}^{(n)}(F)$ se va numi **mulţimea rezolvenţilor lui F obţinuţi în cel mult n paşi**, iar Res*(F) - **mulţimea (tuturor) rezolvenţilor lui F**
- Aceasta constituie definiţia *iterativă* a lui Res*(F)
- Va urma (imediat) și o definiție *structurală* (*iterativ* vs *recursiv*...)

3-8 (70)

- Direct din definiţie urmează că:

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

  $\ldots \subseteq Res^{(n)}(F) \subseteq ... \subseteq \ldots \subseteq Res^{*}(F)$
- Vom nota acum cu Resc(F) mulţimea 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 $C \in Resc(F)$.

  (**Nimic altceva** ...)
- **Exemple** - Seminar (sau în cartea tipărită ...)

3-9 (71)

- **Teoremă**. Pentru fiecare F $\in$ **LP**, avem

  Res*(F) = Resc(F).

(**demonstraţie** - idee: incluziunea

  Res*(F) $\subseteq$ Resc(F) se arată prin inducție matematică; invers, prin inducție structurală generală; vezi de fapt modul de construcție al muțimilor primare în cauză)

- Vom putea astfel folosi ambele notaţii (definiţii) pentru găsirea și /sau manipularea mulţimii rezolvenţilor oricărei mulţimi de clauze (în particular, a unei formule oarecare F $\in$ **LP**, aflată în **FNC** și reprezentată ca o mulțime de clauze)

3-10 (72)

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

(**demonstraţie** – idei: „$\Rightarrow$”; invers, „$\Leftarrow$”, e imediată)

- **Teoremă**. Fie F $\in$ **LP**, aflată în **FNC** şi reprezentată ca mulţime (finită) de clauze. Atunci Res*(F) este finită.

(**demonstraţie** – idei)

- **Teoremă (de compactitate, pentru LP).** Fie **M** o mulţime infinită (numărabilă) de formule din **LP**. Atunci **M** este satisfiabilă dacă şi numai dacă fiecare submulţime **finită** a sa este satisfiabilă.

(**demonstraţie** – idei: **M** satisfiabilă $\Rightarrow$ ... e ușor)

- **Important**. Putem reformula teorema de compactitate astfel: „*O mulțime infinită de formule este nesatisfiabilă dacă și numai dacă există o submulțime finită a sa care este nesatisfiabilă*” (de aici, folosirea cuvântului „nesatisfiabilă” în teorema următoare).

3-11 (73)

- **Teoremă (teorema rezoluţiei pentru LP)**. Fie F o mulţime oarecare de clauze din **LP**. Atunci F este ***nesatisfiabilă*** dacă şi numai dacă $\square \in$ Res*(F).

(**demonstraţie** – idei: din teorema de compactitate reformulată, este suficient să considerăm că F este finită; implicația „$\Leftarrow$”, numită *corectitudinea rezoluției*, se arată folosind teoremele enunțate anterior și aplicând de un număr finit de ori lema rezoluției; invers, „$\Rightarrow$”, adică *completitudinea*, rezultă demonstrând prin inducție matematică *metateorema*: $(\forall n \in \mathbf{N})$(**dacă** |prop(F)| = n **și** F este nesatisfiabilă, **atunci** $\square \in$ Res*(F)))

3-12 (74)

- Din nou „vedem” că problema **SAT** (revenim !) este rezolvabilă în timp exponențial, mai exact, este **NP**-completă (însă sunt de preferat algoritmii sintactici...; a se vedea și Cursul 8, opțional)
- Exceptând anumite subclase „convenabile” ale **LP** (de exemplu, subclasa alcătuită din formulele Horn), am avea totuși nevoie de **strategii** pentru a ajunge *cât mai repede* la clauza vidă
- **Restricțiile** sunt utile atunci când strategiile nu ne sunt de nici un folos/nu se pot aplica (revenim)
- **Rafinările** *rezoluţiei* (= *strategii* + *restricții*) sunt metode prin care se urmăreşte *obţinerea clauzei vide (dacă acest lucru este posibil) într-un număr cât mai mic de paşi de rezoluţie*

3-13 (75)

- Recapitulând, pornind cu formula F = $\{C_1, C_2, \ldots, C_n\}$, clauzele fiind scrise ca mulţimi de literali, se poate construi *efectiv* mulţimea Res*(F), care este finită și poate fi reprezentată ca un graf (ne)orientat (chiar arbore, dacă repetăm aparițiile unor noduri având o aceeași etichetă)
- Nodurile sunt rezolvenţii succesivi, inclusiv clauzele iniţiale, iar muchiile sunt introduse prin rezoluţiile aplicate într-un pas (**desen**: cu $Res^{(i+1)}(F) \setminus Res^{(i)}(F)$...)
- Practic, acest graf *poate să cumuleze* ***toate*** *posibilele demonstraţii prin rezoluţie care pornesc cu clauzele lui* F (anumiţi rezolvenţi vor fi însă excluşi, deoarece reprezintă – sau generează - tautologii)

3-14 (76)

- Demonstrația **Teoremei rezoluţiei** sugerează *crearea* mai întâi a acestui „graf de rezoluţie total” şi apoi *parcurgerea* lui pentru a vedea dacă □ este (eticheta unui) nod în graf
- Mai exact, algoritmul sintactic *rudimentar* de rezolvare a **SAT** înseamnă crearea și „inspectarea” grafului de rezoluție
- Evident că este mult mai bine să găsim **direct** o respingere în loc de a crea şi apoi parcurge întregul graf
- Altfel spus, putem restrânge de la bun început spațiul de căutare, construind „cât mai repede” acel subgraf (nici măcar el în întregime...) care ar putea să conțină □ (chiar dacă complexitatea generală ...)

3-15 (77)

- **Strategiile** nu restrâng, conceptual, spaţiul de căutare (adică *graful total*) dar folosesc anumite *informaţii suplimentare despre clauze*, astfel încât să crească şansele pentru selectarea rapidă a unei demonstraţii căutate, adică a unui „cel mai scurt drum” pornind de la frunze (elementele lui F), către o rădăcină (sperăm că ea va fi clauza vidă)
- Repetăm: la modul ideal graful total nu se construieşte în întregime, ci doar acele porţiuni din el (cât mai puţine şi cât mai „mici”), care este posibil să „conţină” măcar o respingere

3-16 (78)

- Cel mai simplu exemplu „bun” este **strategia unitară**, în care ***se recomandă*** ca la fiecare pas (efectuat) de rezoluţie măcar una dintre clauze să conţină *un singur* literal*;* dacă însă nu mai poate fi aleasă nicio asemenea „clauză unitară”, se continuă procesul de obţinere de noi rezolvenţi (dacă încă n-am găsit □), în mod obişnuit
- Prin urmare, strategiile nu distrug completitudinea rezoluţiei: dacă o formulă este nesatisfiabilă, atunci se poate demonstra acest lucru prin rezoluţie, găsindu-se o respingere (în cel mai rău caz, este posibil nici să nu conducă la vreo economie semnificativă de timp)

3-17 (79)

- Pe de altă parte, **restricţiile** distrug (în multe situaţii) completitudinea rezoluţiei: există formule nesatisfiabile pentru care nu se pot găsi respingeri, în situaţia în care paşii de rezoluţie sunt supuşi unor condiţii prea restrictive (spaţiul de căutare fiind micşorat într-un mod, să-i spunem, abuziv)
- Astfel, o anumită restricţie poate, de exemplu, *interzice total* folosirea unor clauze având o anumită formă sintactică (există și **restricția unitară**)

3-18 (80)

- Multe dintre restricţii rămân însă complete pentru anumite subclase interesante de formule propoziţionale (de exemplu, pentru clasa formulelor Horn)
- Există mai multe exemple importante de restricţii, folosite cu succes de către limbajele universale („de nivel înalt”), comerciale, „de tip **PROLOG**”: *rezoluţia unitară*, *rezoluția pozitivă /negativă*, *rezoluția liniară*, *rezoluția* **SLD**, *rezoluția bazată pe o mulţime suport*, *rezoluția de intrare* etc.

3-19 (81)

- **Rezoluția liniară** se **bazează pe o clauză inițială**
- Considerăm astfel F $\in$ **LP**, F = {C_1, C_2, ..., C_n} (*clauze de intrare*) și C $\in$ F (*clauză inițială/de bază*)
- **Definiție**. O rezoluție liniară bazată pe C și pornind cu F, este o (demonstrație prin) rezoluție în care, la fiecare pas, se aleg spre a fi rezolvate două clauze C' și C'', unde C' este rezolventul pasului anterior iar C'' este fie o clauză de intrare, fie un rezolvent oarecare obținut anterior, pe parcursul demonstrației.
- La primul pas, C' = C și C'' $\in$ F
- C'' se numește *clauză suplimentară* (sau: *definită*, *exactă*, *precisă*, *de program ...*)

3-20 (82)

- Pe parcursul unei rezoluții liniare, se poate folosi și o așa-numită *funcție de selecție pentru clauzele definite*
- Aceasta este de fapt o metodă/algoritm prin care se aleg clauzele de tip C'' de mai sus, pornind de la anumite informații suplimentare (cum ar fi: forma sintactică a clauzelor „eligibile")
- **Rezoluția liniară cu funcție de selecție pentru clauzele definite**, se mai numește și **SLD**-rezoluție și este completă pentru clasa formulelor Horn (nu și pentru întregul **LP**)
- În acest caz, F este partiționată în F_1 și F_2, unde $F_1 = \{C'_1, C'_2, ..., C'_m\}$ (ea conținând doar clauze Horn pozitive și doar acestea vor fi numite clauze suplimentare) și $F_2 = \{N_1, N_2, ..., N_s\}$ (acestea sunt doar clauze Horn negative, numite și *clauze scop*)

3-21 (83)

- Pentru a obține o **SLD**-rezoluție „clasică” (cu clauze Horn), clauza de bază trebui să fie o clauză scop, iar clauzele suplimentare trebuie să fie clauze pozitive (practic, acestea vor putea fi doar elemente ale lui F_1, deoarece toți rezolvenții din demonstrație sunt clauze Horn negative)
- **Exemple** – Seminar (dacă este timp ...)

3-22 (84)

- Folosind faptul că orice formulă din **LP** „poate fi scrisă în” **FNC**, prezentăm în continuare *algoritmii lui Davis-Putnam* (și *-Logemann-Loveland*) (**DP**(**LL**)) *de rezolvare a problemei* **SAT**
- **DP** are mai multe calități, printre care: este considerat a fi „primul” (de natură sintactică) și permite implementări „imediate” în limbaje care admit concurența explicită (cum ar fi JAVA)
- La final, putem continua discuția asupra complexității generale a **SAT**, în contextul paralelismului (și Curs 8 ...)

3-23 (85)

Problemă. Dată orice formulă din **LP** este ea satisfiabilă /validă /contradicție ? (comentarii ...)

Teoremă (SAT). Problema satisfiabilității /... pentru clasa formulelor logicii propoziționale (**LP**) este rezolvabilă /decidabilă.

Demonstrație. Rezultă imediat dacă demonstrăm *terminarea* și *corectitudinea /soundness* (similar cu Horn ...) algoritmului **DP**(**LL**). Se mai folosește exprimarea: **DP**(**LL**) este corect și complet pentru **SAT**. Vezi și „tabele de adevăr” și rezoluția ...

- **Observație**. Formulele din **LP** cu care lucrăm sunt în **FNC**, reprezentate ca mulțime de mulțimi de literali (deci problema rezolvată ar fi chiar **CNFSAT** ...)

3-24 (86)

- Dacă $F = C_1 \wedge C_2 \wedge \dots \wedge C_n$, unde C_i-urile sunt clauze (= disjuncții de literali), adică

  $F = \{C_1, C_2, \dots, C_n\}$ și fiecare C_i este o mulțime de literali, reamintim că sunt adevărate următoarele:

Din definiția operatorului $\wedge$:

-**S**(F) = **1** ddacă $(\forall i \in [n])(\mathbf{S}(C_i) = \mathbf{1})$, pentru fiecare structură **S**; alternativ, pentru ca F să fie falsă în **S**, este suficient să existe un

$i \in [n]$ astfel încât $\mathbf{S}(C_i) = 0$

3-25 (87)

Din definiția operatorului $\vee$:

-Dacă o clauză componentă a lui F, C_i, conține atât un literal L cât și complementarul său $\overline{L}$, atunci ea este tautologie, deci adevărată în orice structură

-Dacă un C_i este o „supraclauză” pentru o altă componentă C_j a lui F și C_j este adevărată în **S**, atunci și C_i este adevărată în **S**

Din proprietățile mulțimilor (gândindu-ne la idempotența, asociativitatea și comutativitatea lui $\wedge$ și $\vee$):

-Deși F $\in$ **LP** poate conține mai multe clauze care sunt identice, este suficient ca în reprezentarea cu mulțimi să păstrăm doar una

3-26 (88)

-Același lucru este valabil pentru fiecare literal în fiecare clauză din F (peste tot până acum, am avut în vedere *păstrarea valorii de adevăr* a lui F ...)

- Reamintim și că pentru fiecare F $\in$ **LP**, pentru a discuta despre semantica sa, este suficient să considerăm doar structurile corecte, adică definite (doar) peste prop(F) $\subseteq$ **A**
- Din motive istorice și didactice vom prezenta atât prima variantă a algoritmului (**DP**, 1960), cât și pe cea de-a doua (**DPLL**, 1962)
- **DPLL** conține atât o rafinare (la propriu) a unor pași din procedura **DP**, dar este și mai „eficient” (*spațiul* de memorie este liniar în cazul cel mai nefavorabil ...)

3-27 (89)

- Deci, dată $F \in$ **LP**, o aducem la **FNC** și o „periem”, eliminând tautologiile (vizibile sintactic – L / $\overline{L}$) și aparițiile multiple de clauze și /sau literali; vom obține o „formulă” $\{C_1, C_2, ..., C_n\}$ tare echivalentă cu F (pe care o notăm tot cu F)
- Calculăm și V = prop(F); putem chiar ordona V (eventual, alfabetic)
- În timpul execuției lui **DP**, rezolvenții se calculează conform definiției știute; excepție: se rezolvă și clauze care au mai mult de un literal care s-ar putea elimina (în acestea ***se „taie” tot***)
- Să notăm că „formula” { }, diferă de „formula” {□}

3-28 (90)

Algoritm DP

Intrare: „Formula” H = F și V = prop(F) (ca mai sus).

Ieșire: „**DA**”, dacă F este satisfiabilă și „**NU**” în caz contrar.

Metodă:

Pasul 1.

Cât_timp (V $\neq$ Ø și □ $\notin$ H)

execută

Pasul 2. Alege A $\in$ **V**

Pasul 3. Calculează mulțimea **R**, a rezolvenților clauzelor din H, „bazați” pe A

Pasul 4. H := H $\cup$ **R**

Pasul 5. H := H \ {C | C conține A (sau Ā)}

Pasul 6. H := H \ {C | C este tautologie „directă”}

Pasul 7. V := V \ {A}

sf_Cât_timp

Pasul 8.

Dacă (H = { })

atunci

Pasul 9. Scrie „**DA**”

altfel

Pasul 10. Scrie „**NU**”

sf_Dacă

- În cele de mai sus:

-**R** = {R | R = $Res_A(C_1, C_2)$, $C_1, C_2 \in H$}

-Tautologie „directă” (sintactică) desemnează o clauză care conține atât pe L cât și pe $\overline{L}$

-„**DA**” înseamnă că F este satisfiabilă (iar „**NU**” ...)

3-30 (92)

- Deși alegerea unei variabile propoziționale poate fi imediată (prin ordonare) și am putea considera că și intrarea F (mulțime de mulțimi ...) a fost obținută „instantaneu”, pașii din corpul buclei (în special **Pasul 3**.), rămân mari „consumatoare de timp”
- Este evident și că memoria este ocupată exponențial, „în exces”
- Este imediat că **DP *se termină***, corpul buclei executându-se de un număr finit de ori (nu se generează literali noi prin „rezoluție = *tăiere*”):

-Dacă clauza vidă „apare” în H (rezoluție între clauzele {L} și {$\overline{L}$}), la pasul următor se „iese forțat” din buclă

-Oricum, la fiecare pas, cardinalul lui V scade cu 1

3-31 (93)

- Pentru demonstrarea ***corectitudinii***, trebuie arătat că, *într-adevăr*, F este satisfiabilă în cazul ieșirii „**DA**” și contradicție pentru ieșirea „**NU**”
- Acest lucru rezultă imediat din **Teorema rezoluției** și din faptul că, pe parcursul execuției **DP**, dacă „nu apare □ mai devreme” se generează întregul Res*(F) (eventual și niște tautologii suplimentare, ignorate)
- Numerotarea exemplelor este „interioară” fiecărui curs

Exemplul 1. $F = (A \vee B) \wedge (A \vee \rceil B) \wedge (\rceil A \vee B) \wedge (\rceil A \vee \rceil B)$. H: F, ca mulțime; prop(F): {A, B}; H final este {{}}, adică {□}.

Exemplul 2. $F = (A \vee \rceil B) \wedge B$. H final este {}, adică Ø.

3-32 (94)

- „Trecem” la **DPLL**; acesta este o rafinare a **DP** (partea legată de satisfiabilitate), care folosește un spațiu de memorie „liniar” și „promovează” *nedeterminismul* (în mod explicit) și *paralelismul* (implicit); Curs 8 ...
- *Operațiile /regulile* indicate în continuare „acționează” asupra oricărei formule H_1 din **LP**, aflată în **FNC** și reprezentată ca mulțime de mulțimi de literali, rezultând (tot) o (altă) astfel de formulă, H_2 (operațiile sunt date ca funcții ...)

3-33 (95)

- Pentru uniformitate (și păstrarea, cât de cât, a descrierii inițiale a algoritmului), admitem și regulile **UNSAT** și **SAT**, privite ca „*predicate peste* **LP**” (funcții de la **LP** în {**0**, **1**}; **0** denotă „nesatisfiabil(ă)”; similar,**1** denotă „satisfiabil(ă)”)
- De fapt (ca și pentru algoritmul **DP**), dacă **DPLL** se termină cu „**DA**”, se poate construi și o structură **S** care să fie model pentru intrarea F, în cazul în care aceasta este o formulă satisfiabilă
- Nu intrăm în amănunte, dar, pentru idei, se pot folosi demonstrațiile teoremelor care vor urma

3-34 (96)

- Regulile pentru **DPLL**:

<u>UNSAT</u>. **Dacă** H_1 conține clauza □ (va fi suficient să considerăm chiar că $H_1 = \{□\}$), **atunci UNSAT**(H_1) = **0** (adică formula H_1 este nesatisfiabilă).

<u>SAT</u>. **Dacă** $H_1 = \{\ \}$ (adică $H_1 = \emptyset$), **atunci** **SAT**(H_1) = **1** (adică formula H_1 este satisfiabilă).

<u>MULT</u>. **MULT**(H_1) = H_2. Se consideră fiecare clauză **C** $\in H_1$. Dacă în **C** se află mai mult de o apariție a unui literal L (mulțimi, absurd, ...), se păstrează doar o apariție, restul ștergându-se. Se face acest lucru pentru fiecare literal care apare în clauza **C** aleasă. Se repetă apoi procedeul pentru fiecare **C**, rezultatul notându-se **C'**. H_2 se obține din H_1 prin înlocuirea fiecărui **C** cu **C'**.

3-35 (97)

<u>SUBS</u>. **SUBS**(H_1) = H_2. Din nou, se consideră fiecare clauză **C** din H_1. Dacă **C** este o supramulțime a unei alte clauze **C'** din H_1, atunci **C** se șterge din H_1. Repetând procedeul (pentru alți **C**), H_2 se va obține din H_1 prin cumularea tuturor acestor ștergeri.

<u>UNIT</u>. **UNIT**(H_1) = H_2. Se consideră fiecare clauză **C** din H_1. Dacă în H_1 există o clauză „unitară" {L} și dacă **C** conține complementarul lui L, $\overline{L}$, atunci acesta se șterge din **C**, obținându-se o nouă clauză **C'** (clauza {L} nu se șterge). Repetându-se procedeul (pentru asemenea **C**, L, **C'**), din nou H_2 se va obține din H_1 prin cumularea tuturor acestor ștergeri.

3-36 (98)

<u>TAUT</u>. **TAUT**(H_1) = H_2. Se consideră fiecare clauză **C** din H_1. Asemenea **C** se va șterge din H_1, obținându-se (după repetări) în final H_2, dacă clauza **C** conține (măcar) un literal L și complementarul său $\overline{L}$.

<u>PURE</u>. **PURE**(H_1) = H_2. Se consideră fiecare clauză **C** din H_1. Asemenea **C** se va șterge din H_1 dacă conține un literal L, iar $\overline{L}$ nu apare în nicio (altă) clauză din H_1. Repetând procedeul (pentru alți **C**), H_2 se va obține din H_1 prin cumularea tuturor acestor ștergeri.

3-37 (99)

SPLIT. **SPLIT**(H_1) = H_2. Să presupunem că H_1 este reprezentarea cu mulțimi (de mulțimi) a formulei (aflate în **FNC**):

$H = (C_1 \vee L) \wedge \ldots \wedge (C_k \vee L) \wedge (C_{k+1} \vee \overline{L}) \wedge \ldots \wedge$ $\wedge (C_m \vee \overline{L}) \wedge C_{m+1} \wedge \ldots \wedge C_n$, unde clauzele $C_1, \ldots, C_k$ nu conțin pe L (de fapt, putem presupune că nici pe $\overline{L}$, complementarul acestuia) iar $C_{k+1}, \ldots, C_m$ nu conțin pe $\overline{L}$ (și nici pe L ...), iar $C_{m+1}, \ldots, C_n$ nu conțin nici pe L și nici pe $\overline{L}$. În plus, presupunem $k \geq 1$ și $m \geq k + 1$, dar mulțimea $\{C_{m+1}, \ldots, C_n\}$ poate fi și Ø. Se consideră acum formula H', „obținută din H prin ștergerea acestor L, $\overline{L}$; mai exact:

3-38 (100)

$H' = (C_1 \wedge \ldots \wedge C_k \wedge C_{m+1} \wedge \ldots \wedge C_n) \vee$

$\vee (C_{k+1} \wedge \ldots \wedge C_m \wedge C_{m+1} \wedge \ldots \wedge C_n)$

Acum aducem H' la **FNC** (aplicând succesiv distributivitatea), iar formula rezultat o reprezentăm din nou ca mulțime (de mulțimi). Rezultatul va fi chiar H_2. Drept comentariu (deocamdată ...), să spunem că **SPLIT** poate fi considerată ca un fel de „rezoluție într-un pas, generalizată”.

- Sunt necesare câteva **exemple** imediate (pentru fiecare regulă enunțată):

3-39 (101)

- $H_1 = \{\square, C_1, C_2, \ldots, C_n\}$

  UNSAT$(H_1) =$ **0**
- $H_1 = \{C_1\}, C_1 = \{L, L, L_1, \ldots, L_m\}$

  MULT$(H_1) = \{\{L, L_1, \ldots, L_m\}\}$
- $H_1 = \{C_1, C_2\}$,

  $C_1 = \{L_1, \ldots, L_m\}, C_2 = \{L_1, \ldots, L_m, L_{m+1}, \ldots, L_n\}$

  SUBS$(H_1) = \{\{L_1, \ldots, L_m\}\}$
- $H_1 = \{C_1, C_2\}, C_1 = \{\overline{L}, L_1, \ldots, L_m\}, C_1 = \{L\}$

  UNIT$(H_1) = \{\{L_1, \ldots, L_m\}, \{L\}\}$

3-40 (102)

- $H_1 = \{C_1\}$, $C_1 = \{L, \overline{L}, L_1, ..., L_m\}$
  TAUT$(H_1) = \{\ \}$
- $H_1 = \{C_1, C_2\}$, $C_1 = \{L', \overline{L}\}$, $C_2 = \{\rceil A\}$,
  $A \in$ **A** și $A \neq L$, $A \neq \overline{L}$
  PURE$(H_1) = \{\{L', \overline{L}\}\}$
- $H_1 = \{C'_1, C'_2, C'_3\}$,
  $C'_1 = \{A, D\}$, $C'_2 = \{\rceil D\}$, $C'_3 = \{B\}$, $A, B, D \in$ **A**
  Dacă ne uităm la definiția generală a lui **SPLIT**, avem:
  $k = 1$, $L = D$, $C_1 = \{A\}$, $C_2 = \varnothing$ (echivalent, $C_{k+1} = C_m = \{\}$), $C_3 = C_{m+1} = C_n = C'_3$. Atunci obținem (ștergând, de peste tot, pe L și pe complementarul său, adică pe D și pe $\rceil D$):
  $H' = (\{A\} \wedge \{\} \wedge \{B\}) \vee (\{\} \wedge \varnothing \wedge \{B\}) \triangleq (A \wedge B) \vee B$, pe care trebuie să o aducem la **FNC**, găsind, în sfârșit
  SPLIT$(H_1) = (A \vee B) \wedge (B \vee B)$ adică $\{\{A, B\}, \{B\}\}$

3-41 (103)

- **Teoremă**. Fie **OP** $\in$ {**MULT**, **SUBS**, **UNIT**, **TAUT**} și H_1, H_2 $\in$ **LP**, aflate în **FNC**. Fie H'_1 și respectiv H'_2 reprezentările lor ca mulțimi de mulțimi. Atunci, dacă **OP**(H'_1) = H'_2, avem $H_1 \equiv H_2$ (altfel spus, operațiile precizate „păstrează echivalența tare").

  Demonstrație:

-De fiecare dată când trebuiesc luate în considerare mai multe clauze posibile (la aplicarea unui același operator), este suficient să „fie" doar una (generic: **C**)

-H_1 (și H'_1, desigur) se presupune a fi deja „periată" (sintactic): nu conține clauze „identice" (care diferă doar prin „ordinea" literalilor: folosim comutativitatea și asociativitatea lui $\vee$)

3-42 (104)

CAZURI

-**OP** = **MULT**: $H_1 \equiv H_2$ rezultă imediat din idempotența disjuncției

-**OP** = **SUBS**: Avem H'_1 = {..., **C**, ..., **C'**, ...}, unde **C** = **C'** $\vee$ C'' (desigur, în reprezentarea cu mulțimi, iar H'_2 = {..., **C'**, ...}. Pentru a avea $H_1 \equiv H_2$, trebuie să arătăm că: pentru fiecare structură **S**, avem **S**(H_1) = **S**(H_2). Dar, pentru ca **S**(H_1) = **1**, trebuie ca atât **S**(**C**) = **1**, cât și **S**(**C'**) = **1** (ceea ce se întâmplă, desigur, și pentru restul clauzelor din H_1/H_2). De aici rezultă că **S**(H_2) = **1**. Invers, dacă **S**(H_2) = **1**, atunci **S**(**C'**) = **1**, deci **S**(**C**) = **1**; de unde **S**(H_1) = **1**

3-43 (105)

-**OP** = **UNIT**: Prin urmare, H'_1 conține clauzele **C =** {...,$\overline{L}$} și {L}, iar H'_2 va conține **C'** = {...} (obținută prin ștergerea lui $\overline{L}$ din **C**) și {L}. Considerând acum orice structură **S**, din **S**(H_1) = **1**, va trebui ca **S**(L) = **1** și **S**(**C**) = **1**. De unde avem **S**($\overline{L}$) = **0** și, prin urmare, **S**(**C'**) = **1**. În concluzie, avem și **S**(H_2) = **1**. Invers, dacă **S**(H_2) = **1**, atunci **S**(L) = **1** și **S**(**C'**) = **1**. De unde avem imediat că **S**(**C**) = **1** (și **S**(L) = **1**), adică **S**(H_2) = **1**

-**OP** = **TAUT**: Acest caz a mai fost tratat, $H_1 \equiv H_2$ rezultând imediat pornind de la faptul că $L \vee \overline{L}$ este tautologie. (**Q.E.D.**)

3-44 (106)

- Am putea spune că operatorii anteriori „furnizează” niște reguli „polinomial simplificatoare” (o formulă peste n variabile propoziționale, din **LP,** aflată în **FNC**, poate fi interpretată ca un polinom peste n variabile, dacă punem „$\wedge$” $\triangleq$ •, „$\vee$” $\triangleq$ **+** și „$\overline{\ }$” $\triangleq$ -), de unde comentariile legate de „spațiul liniar” pentru **DPLL**
- **Teoremă**. Fie **OP** $\in$ {**PURE, SPLIT**} și H_1, H_2 $\in$ **LP**, aflate în **FNC**. Fie H'_1 și respectiv H'_2 reprezentările lor ca mulțimi de mulțimi. Atunci, dacă **OP**(H'_1) = H'_2, avem: dacă H_1 este nesatisfiabilă, atunci H_2 este nesatisfiabilă, și reciproc (altfel spus, operațiile precizate „păstrează nesatisfiabilitatea”).

  Demonstrație:

-Pornim cu aceleași observații de început ca la demonstația anterioară și avem **CAZURILE**

3-45 (107)

-**OP** = **PURE**: Fie deci $H'_1 = \{$**C**$, C_1, C_2, \ldots, C_n\}$, unde $L \in$ **C** și $\overline{L}$ nu apare în $C_1, C_2, ..., C_n$ (de fapt, nici în **C**, pentru că în acest caz **C** s-ar elimina printr-o aplicare a lui **TAUT** – vezi și **Algoritmul DPLL**), și $H'_2 = \{C_1, C_2, \ldots\}$. Să arătăm că dacă H_1 este nesatisfiabilă atunci H_2 este nesatisfiabilă, și reciproc.

$\Rightarrow$: Fie **S** o structură oarecare (corectă pentru $H'_1 \cup H'_2$),cu **S**(H_1) = **0**. Atunci, fie **S**(**C**) = **0**, fie, neexclusiv, $\exists i \in [n]$ a.î. **S**(C_i) = **0**. În al doilea caz, este clar că avem și **S**(H_2) = **0**. Să presupunem că există **S'** a.î. **S'**(**C**) = **0** (ceea ce implică desigur că **S'**(L) = **0**) dar și ($\forall i \in [n]$)(**S'**(C_i) = **1**). Cum $\overline{L}$ nu apare în H'_2, putem „modifica" **S'** într-un **S''** a.î. **S''**(L) = 1, fără a schimba adevărul niciunui C_i, dar pentru care avem **S''**(**C**) = **1**. Astfel, s-ar găsi un **S''** pentru care **S''**(H_1) = 1, adică H_1 nu ar fi nesatisfiabilă (absurd).

3-46 (108)

$\Leftarrow$: Să presupunem acum că H_2 este nesatisfiabilă și fie orice structură **S** (corectă pentru $H'_1 \cup H'_2$). Avem atunci **S**(H_2) = **0** și este imediat și că **S**(H_1) = **0**. Singurul „dubiu” (ca de altfel și în cazul anterior) ar fi atunci când n = 0, dar atunci **C** (și H_1) ar fi evident satisfiabilă și teorema noastră ar fi adevărată „prin lipsă”

-**OP** = **SPLIT**. Așa cum am stabilit în definiția anterioară a lui **SPLIT**, avem (în condițiile precizate acolo):

$H_1 = (C_1 \vee L) \wedge \ldots \wedge (C_k \vee L) \wedge (C_{k+1} \vee \overline{L}) \wedge \ldots \wedge (C_m \vee \overline{L}) \wedge C_{m+1} \wedge \ldots \wedge C_n$ și

3-47 (109)

$H_2 = (C_1 \wedge \ldots \wedge C_k \wedge C_{m+1} \wedge \ldots \wedge C_n) \vee$

$\vee (C_{k+1} \wedge \ldots \wedge C_m \wedge C_{m+1} \wedge \ldots \wedge C_n)$,

această din urmă formulă neadusă la **FNC** și pe care o lăsăm deocamdată așa. Astfel, practic, „dispar” (în H_2) atât L, cât și complementarul său din toate clauzele lui H_1 (fiecare dintre acești literali având însă înainte măcar o „prezență”). Notăm în continuare primul termen al disjuncției din H_2 cu H_{21} și pe cel de-al doilea cu H_{22}, adică $H_2 = H_{21} \vee H_{22}$. Mai mult, punem și:

$H_{11} = (C_1 \vee L) \wedge \ldots \wedge (C_k \vee L)$

$H_{12} = (C_{k+1} \vee \overline{L}) \wedge \ldots \wedge (C_m \vee \overline{L})$ și

$H_{13} = C_{m+1} \wedge \ldots \wedge C_n$

și avem $H_1 = H_{11} \wedge H_{12} \wedge H_{13}$.

3-48 (110)

Să arătăm, din nou, că dacă H_1 este nesatisfiabilă atunci H_2 este nesatisfiabilă, și reciproc.

$\Rightarrow$. Să presupunem că H_1 este nesatisfiabilă și fie orice structură **S** (corectă pentru $H'_1 \cup H'_2$), pentru care avem desigur $\mathbf{S}(H_1) = \mathbf{0}$. Atunci există măcar una dintre posibilitățile (niciuna dintre situații neexcluzând-o aprioric pe vreo alta):

(a) există $i \in [k]$, a.î. $\mathbf{S}(C_i \vee L) = \mathbf{0}$, ***sau***

(b) există $j \in \{k + 1, ..., m\}$, a.î. $\mathbf{S}(C_j \vee \overline{L}) = \mathbf{0}$, ***sau***

(c) există $l \in \{m + 1, ..., n\}$, a.î. $\mathbf{S}(C_l) = \mathbf{0}$.

3-49 (111)

Pentru a avea **S**(H_2) = **0**, trebuie să avem atât **S**(H_{21}) = **0**, cât și **S**(H_{22}) = **0**. În cazul unei situații care include și (c), egalitățile precedente sunt imediat adevărate. Să presupunem că ***doar*** situația (a) este adevărată, adică avem:

S($C_j \vee \overline{L}$) = **1** (pentru fiecare $j \in \{k + 1, ..., m\}$) și **S**(C_l) = **1** (pentru fiecare $l \in \{m + 1, ..., n\}$). Și, desigur, există (măcar) $i0 \in [k]$ a.î. **S**($C_{i0} \vee L$) = **0**, adică **S**(C_{i0}) + **S**(L) = **0**, și deci **S**(C_{i0}) = **0** și **S**(L) = **0**.

De aici s-ar putea concluziona că **S**(H_{21}) = **0** dar nu și că **S**(H_{22}) = **0**.

3-50 (112)

Am putea însă avea $\mathbf{S}(H_{22}) = \mathbf{1}$, doar dacă $\mathbf{S}(C_{k+1}) = \mathbf{1}$, ..., $\mathbf{S}(C_m) = \mathbf{1}$, $\mathbf{S}(C_{m+1}) = \mathbf{1}$, ... și $\mathbf{S}(C_n) = \mathbf{1}$. Să considerăm atunci structura **S'**, cu $\mathbf{S'}(x) = \mathbf{S}(x)$, pentru fiecare $x \neq L$, și $\mathbf{S'}(L) = \mathbf{1}$.

Cum nici L, nici complementarul său nu apar în clauzele lui H_{22}, acestea vor fi adevărate și în **S'**: $\mathbf{S'}(C_{k+1}) = \mathbf{1}$, ..., $\mathbf{S'}(C_m) = \mathbf{1}$, $\mathbf{S'}(C_{m+1}) = \mathbf{1}$, ... și $\mathbf{S'}(C_n) = \mathbf{1}$. Pentru că L este adevărat în **S'** avem și $\mathbf{S'}(C_{k+1} \vee \overline{L}) = \mathbf{1}$, ..., $\mathbf{S'}(C_m \vee \overline{L}) = \mathbf{1}$, chiar dacă $\mathbf{S'}(\overline{L}) = \mathbf{0}$. Același lucru s-ar întâmpla și cu toate clauzele din H_{11}: $\mathbf{S'}(C_i \vee L) = \mathbf{1}$, pentru fiecare

$i \in [k]$, inclusiv pentru i = i0 (nu uităm: $\mathbf{S'}(L) = \mathbf{1}$).

3-51 (113)

Concluzionăm că dacă s-ar putea să apară ***doar*** situația (a) (în cazul în care H_1 este nesatisfiabilă), pentru ca H_{22} să fie adevărată (ceea ce ar implica faptul, neconvenabil, că H_2 este adevărată), se poate găsi o structură care să facă adevărată H_1, ceea ce este absurd. Prin urmare, (a) ***trebuie*** „acompaniată” fie de (b) fie de (c), caz în care și H_{22} va fi falsă, adică H_2 este și ea nesatisfiabilă (fiind falsă în orice structură în care H_1 este falsă...).

Raționăm similar ca mai înainte pentru cazul în care am presupune că ar fi posibilă ***doar*** situația (b).

3-52 (114)

$\Leftarrow$. Invers, să presupunem că $H_2 = H_{21} \vee H_{22}$ este nesatisfiabilă și să arătăm că $H_1 = H_{11} \wedge H_{12} \wedge H_{13}$ este nesatisfiabilă. Fie astfel orice structură **S**, corectă pentru $H'_1 \cup H'_2$ și satisfăcând (desigur) $\mathbf{S}(H_2) = \mathbf{0}$.

Adică $\mathbf{S}(H_{21} \vee H_{22}) = \mathbf{S}(H_{21}) + \mathbf{S}(H_{22}) = \mathbf{0}$ și atunci $\mathbf{S}(H_{21}) = \mathbf{0}$ și $\mathbf{S}(H_{22}) = \mathbf{0}$. Acest lucru înseamnă că:

(d) există $i \in [k] \cup \{m + 1, ..., n\}$, a.î. $\mathbf{S}(C_i) = \mathbf{0}$ ***și***

(e) există $j \in \{k + 1, ..., m\} \cup \{m + 1, ..., n\}$ a.î.

$\mathbf{S}(C_j) = \mathbf{0}$.

Dacă i sau $j \in \{m + 1, ..., n\}$ (neexclusiv), atunci imediat avem și $\mathbf{S}(H_1) = \mathbf{0}$, adică H_1 este nesatisfiabilă.

3-53 (115)

Să presupunem acum că $i \in [k]$ și $j \in \{k + 1, ..., m\}$. Cum structura **S** este corectă pentru $H'_1 \cup H'_2$, avem fie **S**(L) = **1**, fie **S**(L) = **0**. Dacă **S**(L) = **1**, atunci $S(\overline{L}) = \mathbf{0}$ și astfel $\mathbf{S}(C_j \vee \overline{L}) = \mathbf{0}$. Dacă **S**(L) = **0**, atunci $\mathbf{S}(C_i \vee L) = \mathbf{0}$. Oricum, în ambele situații, rezultă, din nou, că $\mathbf{S}(H_1) = \mathbf{0}$, adică H_1 este nesatisfiabilă. (**Q.E.D.**)

- În continuare, este momentul să prezentăm **Algoritmul DPLL**, care, după cum am menționat, rezolvă **SAT** și este la origine profund nedeterminist, putând fi implementat într-un limbaj concurent (sau, alternativ, pe o „mașină paralelă”)
- Secvențial /determinist, algoritmul rămâne (în cazul cel mai defavorabil) cu timp exponențial, dar (atenție la implementare !), cu spațiu de memorie liniar (Curs 8 și ... alte referințe ...)

Algoritmul DPLL

Intrare: Orice formulă $H_1 \in$ **LP** și aflată în **FNC** (de fapt, se consideră H'_1, adică reprezentarea lui H_1 ca mulțime de mulțimi).

Ieșire: „**DA**” dacă H_1 este satisfiabilă și „**NU**” în caz contrar.

Metodă:

Pasul 1. $H := H'_1$

Pasul 2. **Cât_timp** (nici **UNSAT**, nici **SAT** nu sunt aplicabile lui H)

execută

3-55 (117)

Pasul 3. Alege un **OP** care este aplicabil lui H, din mulțimea {**MULT**, **SUBS**, **UNIT**, **TAUT**, **PURE**, **SPLIT**}

Pasul 4. H'_2 := **OP**(H)

Pasul 5. H := H'_2

sf_Cât_timp

Pasul 6.

Dacă (s-a aplicat **SAT** lui H)

atunci

Pasul 7. Scrie „**DA**”

altfel

Pasul 8. Scrie „**NU**”

sf_Dacă

- **Observații**:

-Chiar dacă nu în mod explicit și de la bun început, avem și acum nevoie de prop(H_1), care e finită (nu uităm, ca și intrarea H_1), deoarece majoritatea operațiilor **OP** fac apel explicit la anumiți literali L din această mulțime

-**Algoritmul DPLL *se termină***, în mod evident: fie nu se mai poate executa nicio operație **OP** asupra lui H curentă (și va urma o ieșire „forțată" din buclă); fie (prin execuția corpului buclei, adică aplicarea unui **OP** ales): se șterge din H curentă măcar un literal, sau o clauză („lungimea" textuală a unui asemenea H scăzând), sau doar cardinalul lui prop(H) scade

3-57 (119)

- Înainte de a demonstra că **Algoritmul DPLL** este ***corect și complet*** (față de problema **SAT**), vom considera un exemplu (poate, la Seminar ...)
- **Exemplu**. Luăm $H_1 \in$ **LP**, considerată deja în forma dorită, $H'_1 = \{C_1, C_2, C_3, C_4\}$, unde
  $C_1 = \{A, B\}$, $C_2 = \{D, \rceil B, \rceil C\}$, $C_3 = \{\rceil A, C\}$,
  $C_4 = \{\rceil D\}$. Fixăm „formula curentă" $H = H'_1$, și aplicăm succesiv operatorii de tip **OP** menționați:
  -**UNIT**$(H) = H'_2 = \{C_1, C', C_3, C_4\}$ (= „noul" H); clauza **C** din definiție este C_2 ($L = \rceil D$), iar **C'** va fi $\{\rceil B, \rceil C\}$. Adică $H = \{\{A, B\}, \{\rceil B, \rceil C\}, \{\rceil A, C\}, \{\rceil D\}\}$

3-58 (120)

-**PURE**(H) = H'_2 = {C_1, C’, C_3} (= „noul” H); clauza **C** din definiție este C_4, care conține L = $\rceil$D și complementarul acestui literal (adică D) nu apare în nicio altă clauză din H curent. Prin urmare continuăm cu (am făcut mici modificări, și anume în ordinea literalilor și a clauzelor) H = {{B, A}, {C, $\rceil$A}, {$\rceil$C, $\rceil$B}}, și se observă că putem aplica **SPLIT** (k = 1, m = 2, n = 3): {B, A} este reprezentarea pentru $C_1 \vee L$ din definiție (C_1 = B, L = A), {C, $\rceil$A} este reprezentarea pentru $C_{k+1} \vee \overline{L}$ (C_2 = C, $\overline{L}$ = $\rceil$A), iar {$\rceil$C, $\rceil$B} este reprezentarea lui C_3 adică

3-59 (121)

- **SPLIT**(H) = H'_2, unde, mai întâi, aplicând **SPLIT** lui H sub forma descrisă, adică lui

  $H = (B \vee A) \wedge (C \vee \neg A) \wedge (\neg C \vee \neg B)$, vom obține formula $(B \wedge (\neg C \vee \neg B)) \vee (C \wedge (\neg C \vee \neg B)) \equiv$

  $\equiv (B \wedge \neg C) \vee (C \wedge \neg B)$ (asociativitate „$\wedge$” în interiorul parantezelor; $L \wedge \overline{L} \equiv \mathbf{0}$, pentru fiecare literal L; $\mathbf{C} \vee \mathbf{0} \equiv \mathbf{C}$, pentru fiecare clauză **C**); apoi:

  $(B \wedge \neg C) \vee (C \wedge \neg B) \equiv (B \vee C) \wedge (B \vee \neg B) \wedge$

  $\wedge (\neg C \vee C) \wedge (\neg C \vee \neg B) \equiv (B \vee C) \wedge (\neg C \vee \neg B)$

  (asociativitate „$\vee$”; $L \vee \overline{L} \equiv \mathbf{1}$, pentru fiecare literal L; $\mathbf{C} \wedge \mathbf{1} \equiv \mathbf{C}$, pentru fiecare clauză **C**)

3-60 (122)

Ca urmare, H'_2, care va deveni „noul” H curent, este $H'_2 = H = \{\{B, C\}, \{\rceil B, \rceil C\}\}$. În această situație se aplică din nou **SPLIT**, luând k = 1, L = C, m = 2, $\mathbf{C_1} = B$, $\mathbf{C_2} =$ (= $\mathbf{C_{k+1}}$) = $\rceil B$ (și nu mai există în H alte clauze, de tipul $\mathbf{C_{m+1}}, \ldots, \mathbf{C_n}$, care să nu conțină B sau $\rceil B$).

-**SPLIT**(H) = H'_2, unde, mai întâi, aplicând **SPLIT** lui H sub forma descrisă, adică lui

$H = \{\{B, C\}, \{\rceil B, \rceil C\}\}$, obținem formula $B \vee \rceil B$, adică H'_2 = **SPLIT**(H) = $\{\{B, \rceil B\}\}$, care devine noul H. Acum

3-61 (123)

-**TAUT**(H) = H'_2, unde H = {{B, $\rceil$B}}, **C** = {B, $\rceil$B}, L = B, ceea ce furnizează H'_2 = {}. În sfârșit, cum „noul” H (ultimul H curent) este {}, putem termina execuția, prin aplicarea lui **SAT**, găsind

-**SAT**(H) = **1**. Acest lucru ne „spune” că formula dată ca intrare este satisfiabilă.

- În demonstrația teoremei următoare, vom arăta că în momentul execuției **Pasului 6** din **DPLL**, nu numai că nu mai putem aplica niciun operator asupra formulei/mulțimii de clauze H (curente, rezultată în urma ieșirii din buclă), ci chiar că ea coincide fie cu {}, fie cu {C_1, …, C_k, □}

3-62 (124)

- Pentru că clauza vidă este nesatisfiabilă (prin definiție), ceea ce implică nesatisfiabilitatea întregii formule aflate în **FNC**, și pentru că orice operație „păstrează nesatisfiabilitatea”, odată „apărută” □ (în cursul execuției), ***putem forța ieșirea din buclă și în acest caz, cu*** H $\triangleq$ {□}
- În consecință, ordinea de efectuare a operațiilor nu este relevantă (presupunând că se pot executa mai multe la un același moment) și nici eventualele schimbări sintactice generate de aplicarea asociativității, comutativității etc. (de exemplu în cursul aplicării unui **SPLIT**)
- Ne-am putea gândi astfel la anumite ***strategii*** (similare cu cele de la rezoluție) prin care să putem ajunge ***cât mai repede*** la {} sau {□} (în sensul de mai sus)

3-63 (125)

- Am putea începe, de pildă, cu operațiile de tip **MULT**, urmate (de exemplu) de **TAUT**, **UNIT** și **SUBS**; când acestea nu se mai pot aplica (nu este exclusă o posibilitate de revenire ulterioară la ele!) putem trece la **PURE**, relua (eventual) precedentele și apoi la cele de tip **SPILT**; continuarea ne va fi oricum ghidată de posibilitatea de aplicare a unei reguli, dar situațiile de „posibilă simultaneitate” devin din ce în ce mai rare
- Construcția unei structuri **S**, care să fie model pentru H (în cazul *ipotetic* că aceasta ar fi satisfiabilă), poate fi făcută simultan cu execuția algoritmului, așa cum am mai sugerat

3-64 (126)

- **Teoremă**. Algoritmul **DPLL** este corect și complet pentru **SAT**.

  Demonstrație: Trebuie să arătăm că o formulă F, din calculul propozițional, este satisfiabilă ddacă **Algoritmul DPLL** se termină cu „**DA**” (echivalent: o formulă F $\in$ **LP** este nesatisfiabilă ddacă **Algoritmul DPLL** se termină cu „**NU**”). Practic, în loc de a demonstra ***corectitudinea*** (terminarea cu „**DA**” implică satisfiabilitate) și ***completitudinea*** (satisfiabilitatea lui F implică faptul că **DPLL**, având pe F la intrare, se termină cu „**DA**”), conform celor două teoreme imediat anterioare trebuie doar să demonstrăm că, momentul în care nu se mai poate aplica nici o operație este ***doar*** acela în care formula H curentă, procesată de algoritm în cursul execuției **Pasului 2**, are exact una dintre formele {} sau $\{C_1, \ldots, C_k, \square\}$ ($\triangleq \{\square\}$).

3-65 (127)

Așa după cum am subliniat deja, datorită teoremelor anterioare, „apariția” lui □ determină ieșirea din buclă cu, practic, H = {{}}/{□} și ***răspunsul corect***, „**NU**”. Dacă nu se obține □, din aceleași teoreme amintite, rezultă că H, la ieșirea din buclă (deci, și intrarea în algoritm) este satisfiabilă (păstrarea echivalenței semantice pe parcurs fiind de tipul „ddacă”). Acest lucru înseamnă că asupra formulei curente H nu se mai poate aplica nicio operație, și ceea ce mai rămâne de demonstrat este că H = {}. Să presupunem (**R.A.**) că H este satisfiabilă și mai conține clauze (nevide) care nu pot fi eliminate prin operațiile permise de algoritm.

3-66 (128)

Dacă H = {C}, C $\neq$ Ø, atunci există în C măcar un literal L, nemultiplicat, și C nu conține complementarul lui L. În acest caz se mai poate aplica **PURE** (absurd). Raționamentul continuă în mod similar, pentru 2 clauze, apoi pentru 3 etc., de fiecare dată rezultând că se mai poate elimina o clauză printr-o operație admisibilă. (**Q.E.D.**)

3-67 (129 – final 3)

- Se citesc f. atent primele 11 slide-uri (se poate „sări” de la 3-12 (74) la 3-21 (83), i.e. peste rafinări): **rezoluție**, rezolvenți, demonstrații, Res*(F), teorema rezoluției)
- Problema **SAT** reluată (slide 3-63 (125)); **Algoritmul DP** (de la 3-22 (84) la 3-31 (93); el însuși: 3-28 (90) și 3-29 (91)); fără demonstrații
- **Algoritmul DPLL** (de la 3-32 (94) la 3-66 (128); el însuși: 3-54 (116) și 3-55 (117)); fără demonstrații; algoritm general „nedeterminist și concurent”; intrare uzuală + un set de „operații posibile”
- Pasul principal al unui asemenea algoritm va fi o buclă „while it is possible”, iar „în corp” se va „alege și executa” una dintre operațiile amintite

4-1 (130)

- Revenim asupra semanticii **LP**, mai exact asupra a ceea ce numim *funcții booleene*
- $\mathbf{B} = \{\mathbf{0}, \mathbf{1}\}$
- $\mathbf{B}^n = \mathbf{B} \times \mathbf{B} \times \ldots \times \mathbf{B}$ (de $n \in \mathbf{N^*}$ ori)
- $\mathbf{FB}^{(n)} = \{f \mid f : \mathbf{B}^n \to \mathbf{B}\}$
- Vom pune și: $$\begin{cases} \mathbf{FB} = \bigcup\limits_{n \geq 0} \mathbf{FB}^{(n)} \\ \mathbf{FB}^{(0)} = \mathbf{B} \end{cases}$$
- Să notăm că orice funcţie booleană n-ară, ca element al lui $\mathbf{FB}^{(n)}$, deci **FB**, poate fi reprezentată prin „*tabele de adevăr*” (2^n, 2 la 2^n; F $\rightsquigarrow$ f…)
- Funcții importante din $\mathbf{FB}^{(n)}$ (n = 0, 1, 2, ...)

4-2 (131)

- Mai mult, 4-uplul $\mathcal{B}$ = < **B**, •, +, ¯ > (**B** este mulțimea *suport*, iar „•”, „+” și „¯” sunt respectiv *produsul*, *suma* și *opusul*) este o *algebră Boole*, adică sunt satisfăcute „legile”:

1) $x \bullet y = y \bullet x$ *comutativitatea* lui „•”

2) $(x \bullet y) \bullet z = x \bullet (y \bullet z)$ *asociativitatea* lui „•”

3) $x \bullet (y + z) = (x \bullet y) + (x \bullet z)$

distributivitatea lui „•” faţă de „+”

4) $(x \bullet y) + y = y$ *absorbţia*

5) $(x \bullet \overline{x}) + y = y$ *legea contradicţiei*

4-3 (132)

1’) $x + y = y + x$ *comutativitatea* lui „+”

2’) $(x + y) + z = x + (y + z)$ *asociativitatea* lui „+”

3’) $x + (y \bullet z) = (x + y) \bullet (x + z)$

distributivitatea lui „+” faţă de „•”

4’) $(x + y) \bullet y = y$ *absorbţia*

5’) $(x + \overline{x}) \bullet y = y$ *legea tautologiei*

- Simbolul egal reprezintă egalitatea de funcții
- *Principiul dualității* și alte considerente algebrice (*latici* ...)
- **Exemple** (la seminar...): alte legi (vezi **Tabelul 1.1**, pag.30, din cartea tipărită)

4-4 (133)

- Pentru partea de hard (descrierea structurii fizice reale a unui computer), partea teoretică similară algebrelor booleene se numește *teoria circuitelor*
- Funcțiile booleene se pot reprezenta și ca *text*
- Notaţii (1 și 0 aici, nu sunt practic din **B**...):

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

4-5 (134)

- **Teoremă** *(de descompunere, în sumă de „termeni”)*. Pentru fiecare $n \in$ **N***, $f \in$ **FB**$^{(n)}$ şi fiecare $k \in [n]$, 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 $x_1, x_2, \ldots, x_n \in$ **B**

(**demonstraţie**)

- **Exerciţiu**: Enunţaţi teorema duală („α cu bară” ...).

4-6 (135)

- **Definiţie**. Fie $n \in \mathbf{N^*}$ şi $x_1, x_2, \dots, x_n \in \mathbf{B}$ variabile /nume (booleene) distincte; notăm mulţimea (lista!) acestora cu

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

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

  unde $0 \le k \le n$, $\alpha_1, \alpha_2, \dots, \alpha_k \in \mathbf{B}$ şi

  $1 \le i_1 < i_2 < \dots < i_k \le n$

4-7 (136)

- Termenul generat pentru k = 0 este **1** (prin convenţie); pentru $k = n$ obţinem aşa-numiţii *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 (adică $x_1, x_2, \ldots, x_n$)
- Numărul de termeni şi maxtermeni n-ari distincţi (și maxtermenii sunt termeni): 3^n și 2^n (de ce?)
- Dual: *factori* şi *maxfactori*

4-8 (137)

- **Definiţie**.

  -Se numeşte *formă normală disjunctivă* (*n-ară*, $n \in \mathbf{N}^*$), (n-)**FND** pe scurt, orice sumă (finită) de termeni *n*-ari distincţi

  -Se numeşte *formă normală disjunctivă perfectă* (*n-ară*, $n \in \mathbf{N}^*$), (*n*-)**FNDP** pe scurt, orice sumă de maxtermeni *n*-ari distincţi
- Facem abstracţie 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 combinări de 3^n luate câte k forme normale disjunctive *n*-are având k termeni, $0 \le k \le 3^n$ (prin convenţie, pentru $k = 0$, unica formă care este acceptată, *fiind şi perfectă*, se notează cu **0**), etc.

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

  (n-)**FNDP**–urilor ? (suma... binomul lui Newton...)

4-9 (138)

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

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

(**demonstraţie** – descompunerea...; apoi, *f*(…) **= 1**, care poate fi „pus jos, la sumă”; se deduce și un algoritm pentru „tabel versus text”, la reprezentarea funcțiilor…)

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

4-10 (139)

- Prin dualizare, folosind noţiunile de (*n*-)factor peste X şi maxfactor (*n*-ar, peste X), putem defini noţiunea de *formă normală conjunctivă* (*n*-ară) ((*n*-)**FNC**: orice produs de factori dictincţi) şi respectiv *formă normală conjunctivă* (*n*-ară) perfectă ((*n*-)**FNCP**: orice produs de maxfactori distincţi)
- **Convenţie**: doi factori nu vor fi considerate distincte dacă diferă doar prin ordinea componentelor
- Enunţaţi duala teoremei anterioare, pentru **FNCP**
- Numărul total al (n -)**FNC** – urilor și cel al

  (n-)**FNCP**–urilor : 2 la 3^n respectiv 2 la 2^n (se schimbă ceva relativ la **FND(P)** ?)

4-11 (140)

- Există și alte modalități de a reprezenta /genera clase întregi de funcții booleene (inclusiv **FB**)
- În funcție de timpul avut la dispoziție, la curs sau/și seminar, se poate vorbi despre:
- *Forme normale disjunctive/conjunctive minimale* și *algoritmul lui Quine*
- *Clasa funcțiilor booleene elementare*, *superpoziții*, M-*șiruri*, *închideri* și *mulțimi închise de funcții booleene* (Ø, **E**, **FB**, $\mathbf{T_0}$, $\mathbf{T_1}$, **Aut**, **Mon**, **Lin**); *mulţimi complete*, *precomplete*, *baze*, etc. (vezi cartea tipărită)

4-12 (141)

- Pentru aprofundarea unor concepte cum ar fi *decidabilitate* și *complexitate*, *probleme* și *algoritmi*, *paradigme*, **P** vs. **NP**, *reduceri*, *automate*, consultați suportul suplimentar de informație (de ex., **Cursul 8**)
- Problema de a decide, pentru o funcție booleană, dacă „ia” doar valoarea **0**, doar valoarea **1**, sau atât valoarea **0** cât și valoarea **1** (**B**oolean **S**atisfiability **P**roblem - **BSP**) este de importanță majoră în teoria complexității
- Din ea derivă varianta pentru **LP** (adesea identificată cu precedenta) și numită **P**ropositional **S**atisfiability **P**roblem (**PSP**) sau **Sat**isfiability problem (pe scurt: **SAT**; cu „versiuni” - **3CNFSAT** etc.)
- **Dată** F $\in$ **LP, este ea satisfiabilă ? Este tautologie ? Este contradicție ?** (am făcut o mare parte ...)

4-13 (142 – final 4)

- Toate slide-urile din acest curs trebuie citite și aprofundate: algebre booleene, termeni, factori, teoreme de descompunere și reprezentare, **FNCP** și **FNDP**; mai puțin informațiile despre mulțimi complete de funcții booleene, baze, superpoziții etc. (deși ...)
- E vorba de fapt de o revenire (pe un plan superior) la semantica formală a **LP**, inclusiv la conexiunile acesteia cu sintaxa și rezolvările problemei **SAT**
- Acest curs este puternic legat de următorul

5-1 (143)

- O ultimă modalitate importantă de reprezentare a funcțiilor booleene este cea folosind *diagramele de decizie binare (ordonate)* (**O***rdered* **B***inary* **D***ecision* **D***iagrams*, pe scurt **(O)BDD**)
- Ştim ce înseamnă *funcţie booleană* şi reprezentarea acestor funcții cu ajutorul tabelelor de adevăr /matrici sau cu expresii /texte
- Alegerea celei mai „convenabile” reprezentări (electronice!) depinde însă de context (problema de rezolvat, în principal)
- Tot ceea ce putem menţiona în acest moment este că în anumite cazuri o (**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)

5-2 (144)

- **Definiţie.** Se numeşte *diagramă de decizie binară peste* lista

  $X = \{x_1, x_2, \ldots , x_n\}$ 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*; cerinţa nu este însă obligatorie întotdeauna…;

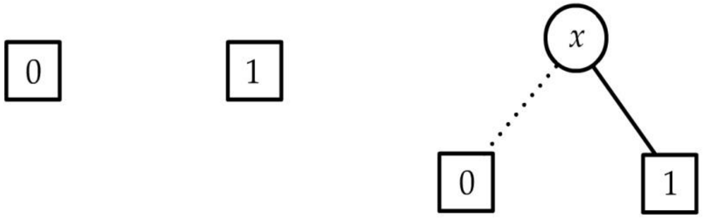
  -fiecare nod care nu este frunză are exact doi succesori imediaţi, arcele care îi leagă fiind și ele etichetate: cu **0** (*stânga*) respectiv **1** (*dreapta*)
- O **subBDD** (într-o **BDD** dată) este un *subgraf generat* de un nod fixat împreună cu *toţi* succesorii săi (desigur, inclusiv arcele care le leagă)

5-3 (145)

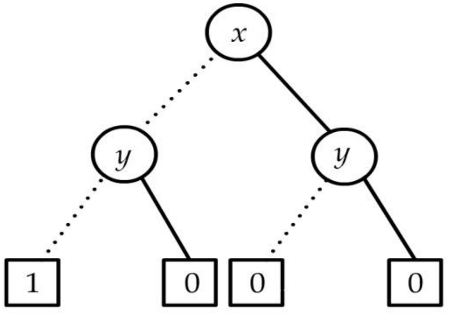
- De obicei, într-un „desen” care reprezintă o **BDD**, frunzele pot fi identificate (şi) prin pătrate (nu cercuri, ca restul nodurilor), orientarea arcelor este implicită („de sus în jos”), arcele etichetate **0** sunt reprezentate prin linii punctate (stânga), iar cele etichetate **1** sunt linii continue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

5-4 (146)

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



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



5-5 (147)

- **Observaţie.** Orice **BDD** peste $X = \{x_1,x_2,\dots ,x_n\}$ defineşte /reprezintă /calculează o **unică** funcţie booleană $f \in \mathbf{FB}^{(n)}$
- Astfel, pentru $\alpha_1,\alpha_2,\dots,\alpha_n \in \mathbf{B}$ (considerate ca fiind valori asignate corespunzător „variabilelor booleene” din X) se „porneşte” cu rădăcina (unică) şi se „parcurge” un drum (unic) în graf „până” la o frunză (să spunem că aceasta este etichetată cu $\beta \in \mathbf{B}$)
- La fiecare pas, plecând din nodul curent, se alege acel arc (prin urmare şi noul nod curent) care are ataşată valoarea **0** sau **1** conform valorii α deja atribuite ex-nodului curent x (în asignarea aleasă)
- Valoarea β este chiar $f(<\alpha_1, \alpha_2, \dots , \alpha_n>)$

5-6 (148)

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

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

- Este clar că putem proceda şi invers, adică pornind cu orice funcţie f $\in$ **FB**$^{(n)}$ dată printr-un tabel de adevăr, construim (măcar) un arbore (**BDD**) binar, complet şi având n + 1 *niveluri*, notate 0 - *rădăcina*, 1, … , n-1, n - frunzele (alternativ, arborele are *adâncimea* n) care „calculează” *f*, în modul următor:

5-7 (149)

- Se ordonează mulţimea de variabile cu ajutorul căreia este exprimată funcţia, 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 $x_{k,1}$ ea 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)

5-8 (150)

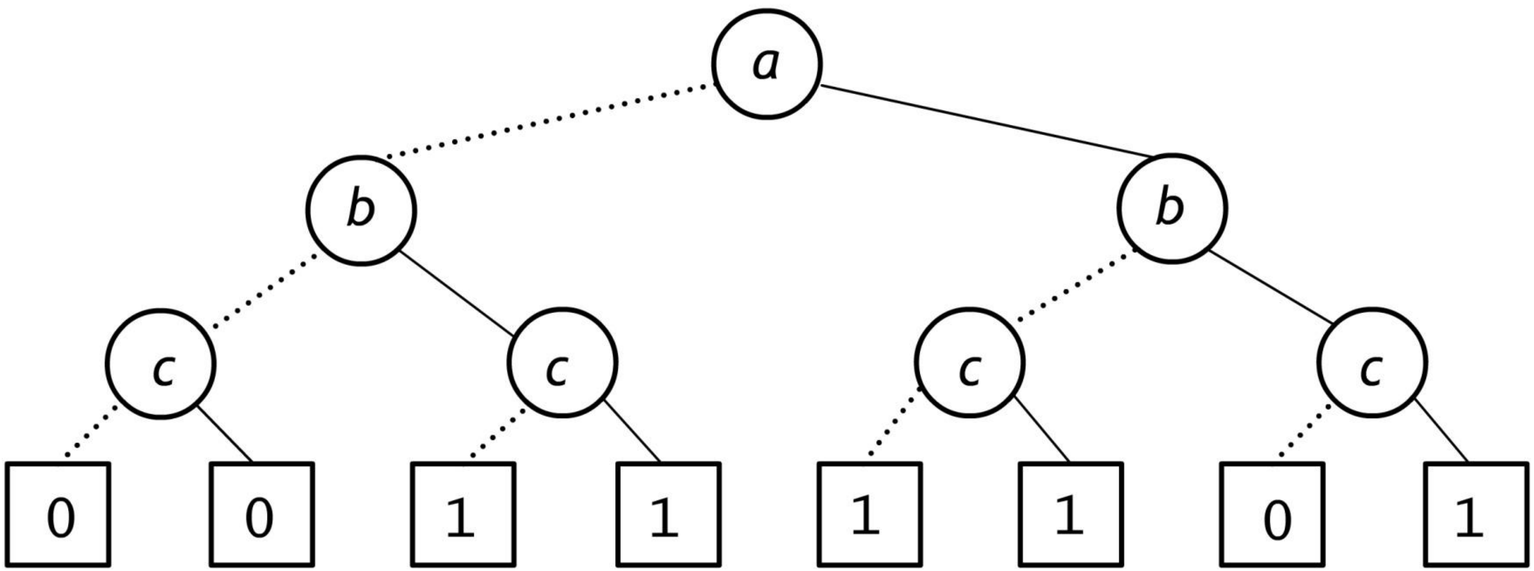
- **Exemplu.** Fie $f \in$ $\mathbf{FB}^{(3)}$ dată prin

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

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

  <a, b, c> fiind ordinea impusă asupra variabilelor; tabela sa de adevăr este…
- **BDD**-ul care calculează *f* după algoritmul sugerat anterior este... (urmează pe alt slide)
- **Observaţie**. De multe ori nu vom face o distincţie explicită între o funcţie booleană şi o formulă din **LP** (deși…)

5-9 (151)



5-10 (152)

- După cum se observă imediat, construirea unui **BDD** care calculează o funcţie 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 $\mathbf{n}$, $\mathbf{m}$, etc. nu ne referim la eticheta din X; sunt doar nişte nume noi):

5-11 (153)

C1 (înlăturarea frunzelor duplicat). Dacă o **BDD** conţine 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 „redirecţionează” 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 destinaţie 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 „redirecţionate” spre $\mathbf{m}$.

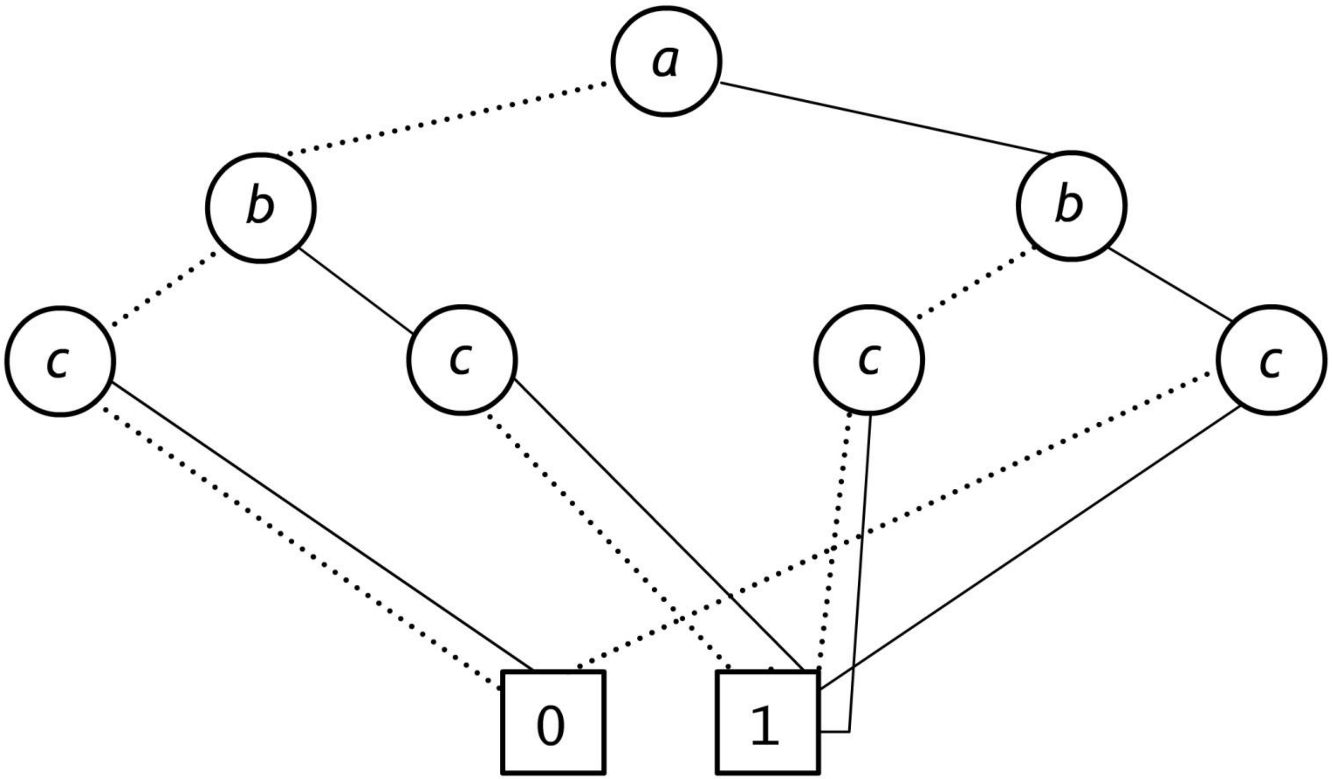
C3 (eliminarea nodurilor duplicat care nu sunt frunze). Dacă în **BDD** există două noduri interioare distincte, să zicem $\mathbf{n}$ şi $\mathbf{m}$, care sunt rădăcinile a două

subBDD-uri identice (fiind identice, $\mathbf{n}$ şi $\mathbf{m}$ sunt şi pe acelaşi nivel, $\mathbf{m}$ „mai în dreapta”), 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 „redirecţionează” spre $\mathbf{n}$.

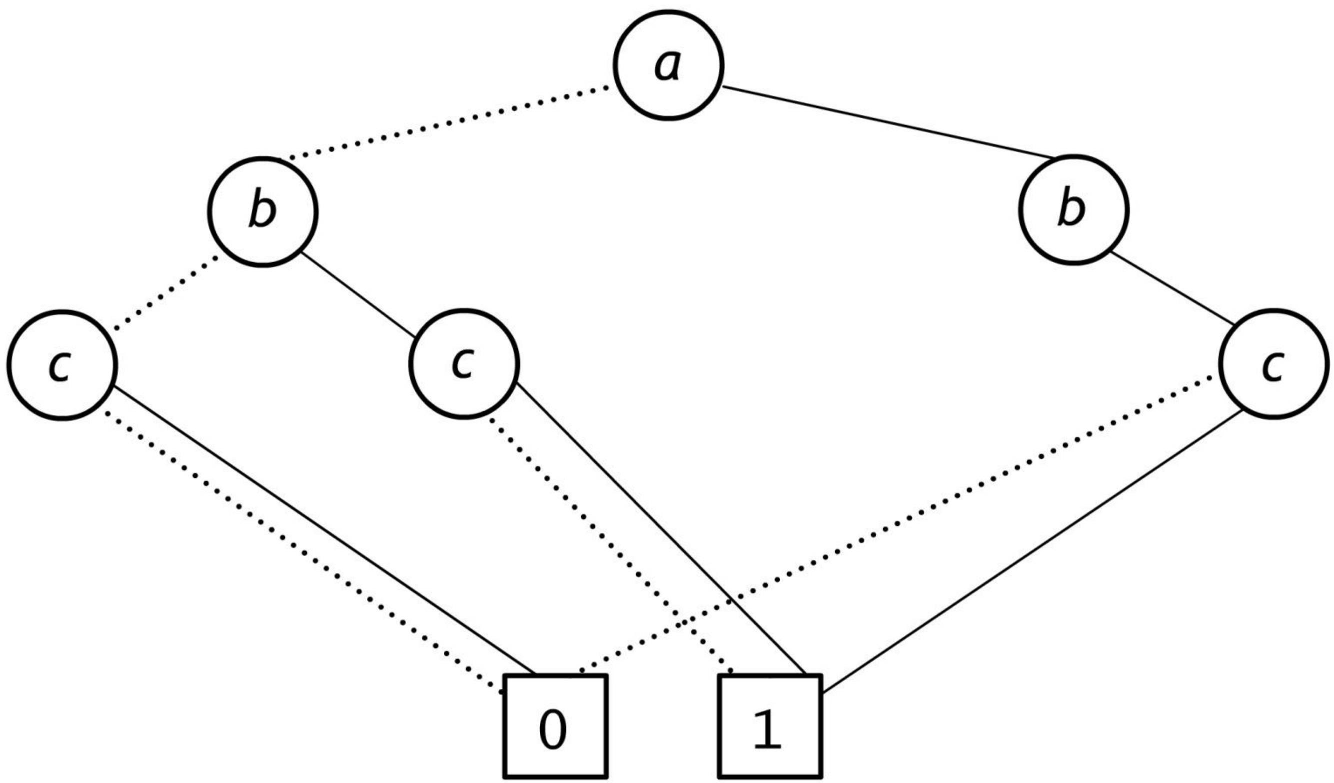
5-12 (154)

- **Exemplu.** Plecând de la **BDD**-ul din ultimul exemplu obţinem succesiv (mai întâi sunt transformările de tip **C1**, apoi toate cele de tip **C3** şi, în sfârşit, toate cele de tip **C2**):

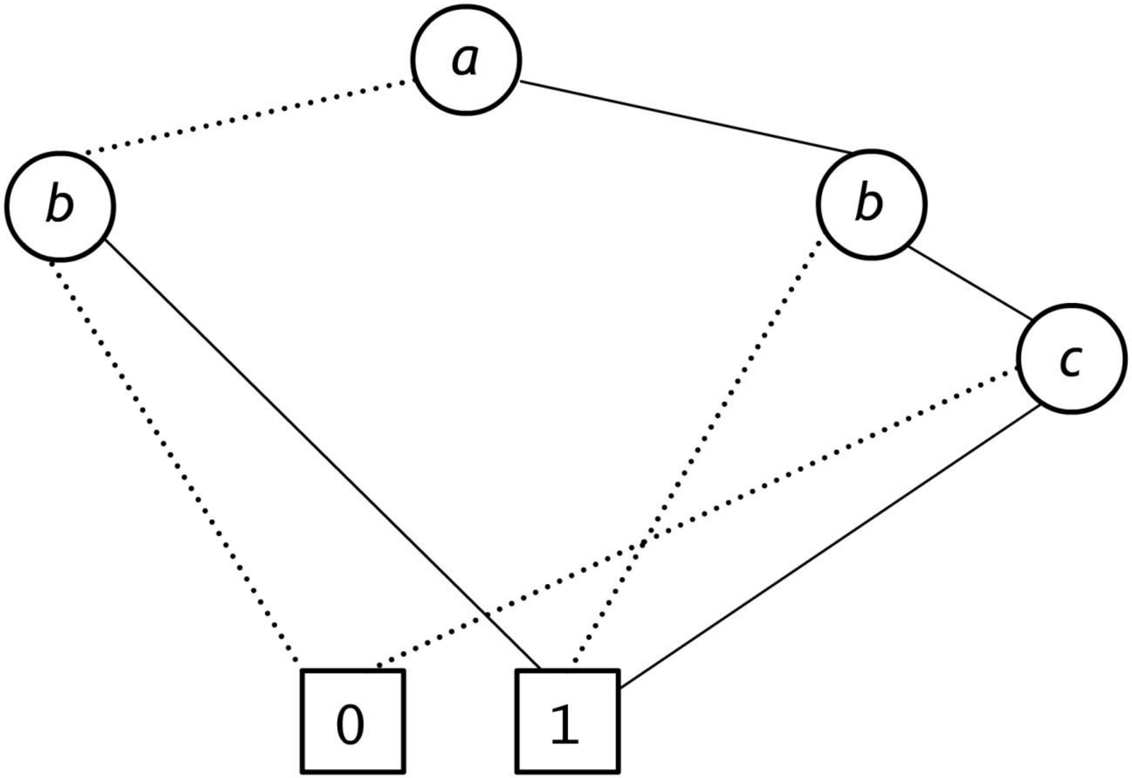
5-13 (155)



5-14 (156)



5-15 (157)



5-16 (158)

- Ultima **BDD** este *maximal redusă* (nu se mai pot face alte transformări de tipul precizat)
- Desigur că ordinea în care se efectuează eliminările de tipul **C1**-**C3** poate influenţa 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** funcţia booleană calculată

5-17 (159)

- Concluzionând, **BDD**-urile pot fi uneori „convenabil de compacte”
- Mai mult, putem reformula anumite definiţii ale funcţiilor booleene pentru noua reprezentare, căpătând, de exemplu, idei noi de algoritmi pentru rezolvarea problemei **BSP/SAT**
- **Diagramele de decizie binare ordonate, precum și anumite completări vor fi studiate/exemplificate** în funcție de timp
- Dacă celelalte reprezentări ale funcțiilor booleene sunt utile pentru „zona software” (programare ...),

  (O)BDD-urile sunt utile în special pentru „zona hardware” (circuite logice); dar și proiectarea sistemelor multiagent, verificare automată folosind teoria modelelor și anumite logici specializate

5-18 (160 – final 5)

- Și aceste ultime slide-uri (legate intrinsec de Cursul 4) trebuie citite în totalitate
- Ideea de bază (în ceea ce privește revenirea de fapt la semantica **LP**, într-o formă mai aprofundată /granulată) este legată de introducerea unei modalitatăți de a „lega” formulele propoziționale de funcțiile booleene
- **LP** $\rightsquigarrow$ clase „$\equiv$” (și structuri) $\rightsquigarrow$ **FB** /(**O**)**BDD**, și reciproc
- De asemenea de a furniza /reprezenta constructiv (în mai multe moduri) întreaga clasă **FB** (la fel și **LP**)

6-1 (161)

- În cursul de față începem tratarea globală a claselor de formule (deocamdată, din **LP**), vorbind despre *sisteme deductive* (noțiune **sintactică**) și despre *teorii logice* (noțiune **semantică**)
- O **teorie logică** este o mulţime de formule închisă la consecinţă semantică (exemplu clasic și de interes: *clasa formulelor valide* dintr-o logică dată)
- Un **sistem deductiv**, este format din *axiome* + *reguli de inferenţă*

  (eventual, plus „câteva” *ipoteze suplimentare …*)
- O **teoremă de corectitudine şi completitudine** certifică legătura dintre „adevăr” și „demonstrație”; <u>se dorește ca într-o logică dată /construită corespunzător, să avem</u>: *tot ceea ce este demonstrabil este adevărat* (*corectitudine*) și, reciproc, *tot ceea ce este adevărat este demonstrabil* (*completitudine*)

6-2 (162)

- Deci, noțiunea de teorie logică (**TE**) este un concept semantic pentru definirea şi tratarea globală a unei mulţimi de „formule"
- O teorie logică este astfel, formal, o (sub)clasă de formule (dintr-o logică dată), care este închisă la consecinţă semantică
- Cu alte cuvinte, o mulţime **TE** de formule este teorie logică dacă pentru fiecare submulţime M $\subseteq$ **TE** şi fiecare (altă) formulă G care este consecinţă semantică din M, avem şi G $\in$ **TE**

6-3 (163)

- Un exemplu imediat de teorie logică este dat de clasa formulelor valide (din **LP**)
- Notaţii pentru „consecinţă semantică”:

  $\mathcal{I} \vDash F$; $\vDash F$, sau $\emptyset \vDash F$ (pentru „F – validă”; deși... „adevărat prin lipsă”)
- $\mathcal{Cs}(\mathcal{F})$ – mulţimea consecinţelor semantice din $\mathcal{F} \subseteq$ **FORM** (la modul „cel mai general”; nu insistăm...)
- Nu insistăm nici asupra altor concepte cum ar fi cele privind *tipurile de teorii*: *(ne)degenerate*, *(in)consistente*, *complete*, *recursive (recursiv enumerabile)* etc. (a se consulta, eventual, bibliografia)

6-4 (164)

- Noțiunea de sistem deductiv (*de deducţie*, *de demonstraţie*, *inferenţial*), pe scurt **SD**, este un concept sintactic pentru definirea şi tratarea globală a unei mulţimi de „formule”
- Se numeşte sistem deductiv în **FORM**, care reprezintă mulţimea „(meta)formulelor” dintr-o logică dată, un cuplu **SD** = $<\mathcal{A}, \mathcal{R}>$ unde

  \- $\mathcal{A} \subseteq$ **FORM** este o mulţime de *axiome* iar

  \- $\mathcal{R} \subseteq$ $\mathbf{FORM}^{+} \times \mathcal{C}$ este o mulţime de *reguli de inferenţă* (*de deducţie*, *de demonstraţie*)

6-5 (165)

- În cele de mai sus, **FORM⁺** denotă mulţimea relaţiilor de oricâte argumente (cel puţin unul) peste **FORM**, iar $\mathcal{C}$ reprezintă o mulţime de *condiţii de aplicabilitate*
- Fiecare regulă de inferenţă r $\in \mathcal{R}$, are astfel aspectul

  $r = < < G_1, G_2, \ldots , G_n, G>, c>$, unde $n \in \mathbf{N}$,

  iar $G_1, G_2, \ldots , G_n, G \in$ **FORM;** $c \in \mathcal{C}$
- $G_1, G_2, \ldots , G_n$ sunt *ipotezele (premizele) regulii*, G reprezintă *concluzia (consecinţa)* iar c desemnează *cazurile (modalităţile) în care regula poate fi aplicată*. Vom scrie chiar $r = < < \{G_1, G_2, \ldots , G_n\}, G>, c>$ deoarece *ordinea ipotezelor nu este esenţială*

6-6 (166)

- Mulţimea $\mathcal{C}$ nu a fost specificată formal, din cauza generalităţii ei; putem spune totuşi că elementele sale sunt (meta)*predicate* (condiții „de satisfăcut” petru formule, demonstrații, etc.)
- Similar cu situaţia rezoluţiei, regulile vor fi folosite pentru a construi demonstraţii în paşi succesivi, la un pas aplicându-se o regulă

6-7 (167)

- O regulă r = < < {G_1, G_2, … , G_n}, G>, *c*> va fi scrisă şi ca:

$$\frac{G_1, G_2, \ldots, G_n}{G}, c$$

- Câteodată, alături de *c*, sunt explicitate separat şi restricţiile sintactice locale asupra (formei) (meta)formulelor
- În cazul în care n = 0 şi *c* lipseşte, r poate fi identificată ca fiind o axiomă, după cum rezultă din definiţia care va urma

6-8 (168)

- Există însă posibilitatea ca în afara restricţiilor sintactice „locale”, date de forma formulelor implicate (ceea ce face ca regulile să fie, de fapt, *scheme generale*) să se interzică aplicarea regulii (schemei) pe considerente semantice „globale” (forma demonstraţiei, apariţia în demonstraţie a unei formule nedorite la acel pas, păstrarea completitudinii unei teorii etc.)
- Dacă *c* este ataşată unei reguli r (*c* poate lipsi, mai exact ea poate fi „condiţia permanent adevărată indiferent de context”), înseamnă că în **orice** demonstraţie, r va putea fi aplicată la un moment dat **doar dacă** *c* este adevărată la momentul respectiv

6-9 (169)

- **Definiţie**. Fie un sistem deductiv **SD** = <$\mathcal{A}$, $\mathcal{R}$> în **FORM**. Se numeşte *demonstraţie* (pentru F_k, pornind cu $\mathcal{A}$) în **SD** o listă de (meta)formule, ($\mathcal{D}$), F_1, F_2, … , F_k astfel încât pentru fiecare i $\in$ [k], fie $F_i \in \mathcal{A}$, fie F_i este obţinut din F_{j1}, F_{j2}, … , F_{jm} folosind o regulă r = < < {F_{j1}, F_{j2}, … , F_{jm}}, F_i>, c> din $\mathcal{R}$,, unde j_1, j_2, ... , j_m < i
- Prin urmare, fiecare element al listei ($\mathcal{D}$) este fie o axiomă, fie este concluzia unei reguli de inferenţă ale cărei ipoteze sunt elemente anterioare din listă (toate acestea se numesc și **teoreme**)

6-10 (170)

- O demonstrație (**raționament formal** !), se poate reprezenta ca un graf, sau chiar ca un arbore
- Putem considera că un sistem de demonstrație poate conține, pe lângă axiome (de obicei acestea sunt formule *valide*, „știute” ca fiind așa printr-o altă metodă credibilă), și, eventual, niște ipoteze suplimentare (considerate, de obicei, a fi formule, *măcar satisfiabile*)
- Vorbim atunci de **SD’** = <$\mathcal{A}$’, $\mathcal{R}$>, $\mathcal{A}$’ = $\mathcal{A} \cup \mathcal{I}$, $\mathcal{I}$ reprezentând „axiomele suplimentare”
- Notăm $\mathcal{Th}$(**SD**) = {F $\in$ **FORM** | există o demonstraţie ($\mathcal{D}$) pentru F în **SD**} (se poate da şi o definiţie constructivă pentru $\mathcal{Th}$(**SD**))

6-11 (171)

- Și în legătură cu sistemele deductive putem discuta despre: *tipuri* de sisteme deductive (*boolean complete*, *finit axiomatizabile* etc.), sau ddespre *clasificarea generală* a acestora (Hilbert, Gentzen, etc.); nu insistăm
- **Exemplele** tratate, corecte și complete: **SD3**, **SD0**, **SD1**; detaliile vor fi lăsate în mare parte pentru studiu individual; după, vom și reveni cu anumite precizări /generalizări
- Oricum, ceea ce ne propunem să folosim (pentru **IA**, în general), sunt <u>sistemele corecte și complete pentru clasa formulelor valide ($\mathcal{Val}$(X)) din orice logică dată, X</u>

6-12 (172)

- **Definiție** (*regulă de inferență derivată*)**.** Considerând orice prefix al oricărei demonstraţii (privită textual) ($\mathcal{D}$) dintr-un sistem **SD**, acesta poate fi considerat ca o nouă regulă de inferenţă („derivată” din cele iniţiale): concluzia noii reguli este ultima formulă din demonstraţia respectivă, iar ipotezele sunt reprezentate de restul formulelor care apar.
- **Definiţie.** Două sisteme **SD’** şi **SD’’** sunt *echivalente* dacă pentru fiecare mulţime de (meta)formule $\mathcal{I}$ (poate fi chiar vidă) şi fiecare (meta)formulă F avem: $\mathcal{I} \vdash_{\mathbf{SD'}}$ F dacă şi numai dacă $\mathcal{I} \vdash_{\mathbf{SD''}}$ F.

6-13 (173)

- Dacă un sistem are „mai multe” reguli de inferență decât axiome, el se numește de tip Gentzen(-Jaskowski)
- Un asemenea sistem va fi **SD0**, sau **deducţia naturală**, care nu are nicio axiomă (!); revenim
- În cazul în care balanţa este inversată (există „mai multe” axiome decât reguli de inferenţă, ca în cazul **SD3**), sistemele sunt cunoscute sub numele de sisteme (de tip) Hilbert
- **SD0** este echivalent cu **SD3** (și cu **SD1** de altfel; sistem tratat pe final de curs)

6-14 (174)

- Sistemul **SD3**, exemplificat pe scurt în continuare, este un sistem deductiv *standard* (de tip Hilbert), *finit specificat*, care generează întreaga clasă (şi numai pe aceasta) a formulelor valide din **LP** („completându-l”, vom obține același lucru pentru **LP1**, după cum se va vedea, dacă va fi timp)
- Sistemul a fost introdus pentru prima dată de către A. Church (1954)
- **Axiome** ($\mathcal{A}_{\mathbf{SD3}}$). Condiţiile sintactice sunt doar F, G, H $\in$ **LP**:

1. $F \rightarrow (G \rightarrow F)$
2. $(F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$
3. $(\rceil F \rightarrow \rceil G) \rightarrow ((\rceil F \rightarrow G) \rightarrow F)$

6-15 (175)

- Să remarcăm deja faptul că mulțimea **LP** trebuie considerată ca fiind construită peste alfabetul care conţine drept conectori doar pe „$\rceil$” şi „$\rightarrow$”
- Dacă dorim să utilizăm şi ceilalţi conectori, putem face acest lucru utilizându-i doar ca notaţii (de exemplu, $A \vee B$ va reprezenta de fapt $\rceil A \rightarrow B$; etc.)
- **Reguli de inferenţă ($\mathcal{R}_{SD3}$).** Există doar restricţii de natură sintactică (lipsind condiţiile de aplicabilitate): F, G $\in$ **LP**
- Schema de regulă (unică) este *modus ponens* (pe scurt, **(MP)**):

1. $$\frac{F \rightarrow G, F}{G}$$

6-16 (176)

Exemplu. Să se arate că în **SD3** se poate demonstra teorema $T = (A \rightarrow A)$.

- În cele ce urmează vom renunţa pe parcurs la unele paranteze (dacă înţelegerea nu este afectată), deşi, formal, acest lucru nu este admis
- În derivare, folosim întâi instanţa schemei 1., obţinută luând $F = A$ şi $G = (A \rightarrow A)$ (regula aplicată va fi evident, mereu, **(MP)**)
- Primul element al listei care reprezintă demonstraţia va fi atunci: $E1 = A \rightarrow ((A \rightarrow A) \rightarrow A)$
- Folosim acum schema 2., punând $F = A$, $G = (A \rightarrow A)$ şi $H = A$ și obţinem: $E2 = (A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A))$
- Aplicând regula avută pentru $E2 = F \rightarrow G$ şi $E1 = F$ (se observă imediat că $G = (A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$), găsim: $E3 = (A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$
- Punem acum $F = A$ şi $G = A$, în schema de axiome 1., rezultând: $E4 = A \rightarrow (A \rightarrow A)$
- În sfârşit, putem folosi regula pentru E3 şi E4 (luând $F = A \rightarrow (A \rightarrow A)$ şi $G = (A \rightarrow A)$), pentru a obţine ceea ce doream, adică: $E5 = T = (A \rightarrow A)$

6-17 (177)

- Continuăm prezentarea cu un sistem de tip Gentzen, și anume **SD0** (*deducția naturală)*
- Prima abordare este orientată pe introducerea anumitor aspecte teoretice generale (aici bibliografia este puțin accesibilă: C. Cazacu ...)
- Ne vom axa apoi (în principal) pe abordarea lui **M. Huth/M. Ryan**, din cartea menționată (e mai accesibilă; și științific ...); încercați să le comparați în final
- Clasa **FORM** este desigur **LP**
- Alfabetul peste care sunt construite formulele conține în acest caz doar conectorii ⌉ (notat uneori și prin $\neg$) și $\wedge$

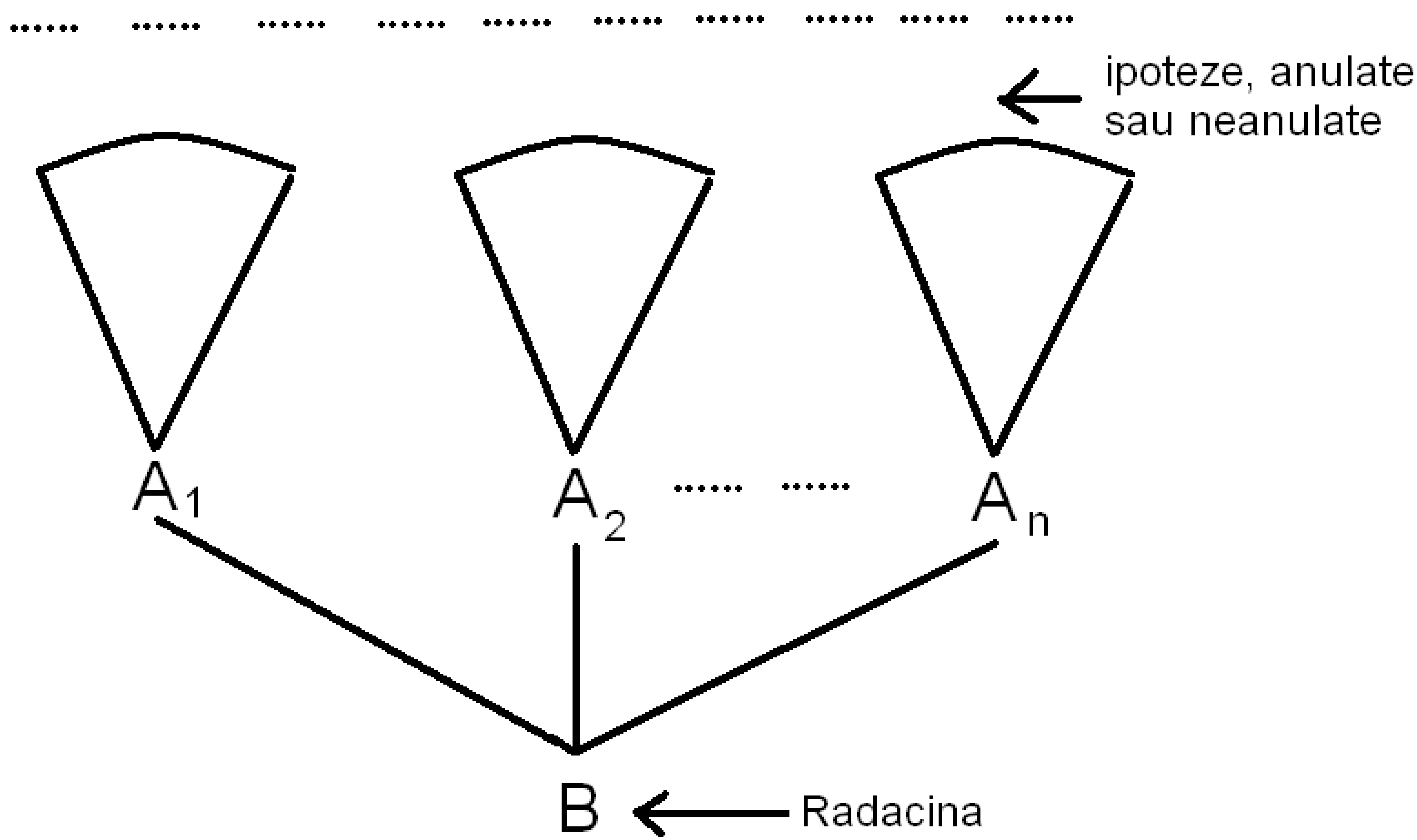
6-18 (178)

- După cum am mai spus, în sistemele Gentzen regulile de inferență sunt „mai multe” decât axiomele
- **SD0 nu are nicio axiomă**
- O demonstrație, sensul *deducției naturale* va fi definită în mod direct ca fiind un arbore (și nu o listă)
- Un ***arbore de deducție naturală*** are pe nivelul 0 (cel al frunzelor) formule oarecare (numite și *ipoteze*, sau *premize*, pentru anumite reguli de inferență din sistem)
- Următoarele niveluri sunt obținute constructiv din precedentele
- Astfel, nivelul *i* va conține *concluziile* inferențelor având ca premize elemente ale nivelurilor anterioare 0,1,...,*i*-1 (rădăcina fiind „rezultatul final“ al demonstrației)

6-19 (179)

- O caracteristică importantă a acestui sistem este aceea că acele condiții de aplicabilitate ***c*** asociate regulilor (dacă există), au aspectul *„înlătură ipoteza* F*”* (termenul *ipoteză* se referă aici exclusiv la frunzele arborelui curent)
- Înlăturarea nu este una efectivă: doar *marcăm* frunzele corespunzătoare (de exemplu, folosind numere; diferite de la pas la pas)
- Corectitudine/completitudine **SD0** (**TCC0**): F va fi o ***formulă validă*** *în* **LP** ddacă este rădăcina unui arbore de deducție naturală ***având toate ipotezele*** înlăturate (pe parcursul demonstrației)

6-20 (180)



6-21 (181)

- Vom furniza toate regulile, ($\mathcal{R}_{\mathbf{SD0}}$), corespunzătoare acestui sistem **SD0**, folosind notațiile generale deja introduse
- Orice regulă este de fapt *o schemă* (valabilă pentru fiecare formulă; acestea sunt notate aici cu

  A, B, … $\in$ **LP**, și nu cu F,G, ...); ele au asociate atât un *număr de ordine*, cât și un *mnemonic*
- Schemele 3. şi 4. au și alternative, deoarece trebuie să „captăm” comutativitatea conjuncției la nivel sintactic (ele se vor numi 3’., și respectiv 4’.)
- Mnemonicele „vin” de la următoarele cuvinte:

  E – eliminare; **I** – introducere; **N** – negare; **C** – conjuncție (pentru primele patru reguli)

6-22 (182)

- 1. (**EN**) $\dfrac{B,\ \neg B}{A}$, ***c***: *ipoteza* $\neg$A *este înlăturată*
- 2. (**IN**) $\dfrac{B,\ \neg B}{\neg A}$, ***c***: *ipoteza* A *este înlăturată*
- 3. (**EC**) $\dfrac{A \wedge B}{A}\ si\ \dfrac{A \wedge B}{B}$
- 4. (**IC**) $\dfrac{A, B}{A \wedge B}\ si\ \dfrac{A, B}{B \wedge A}$

6-23 (183)

- Acest **SD0** este un *sistem predicativ*, *finit specificat* și *boolean complet*; dacă introducem direct și conectorii $\vee$ și $\rightarrow$ în alfabet, putem folosi și următoarele *reguli derivate* (revenim):
- 5. (**ED**) $\dfrac{A \vee B, \neg A}{B} \, si \, \dfrac{A \vee B, \neg B}{A}$ 6. (**ID**) $\dfrac{A}{A \vee B} \, si \, \dfrac{A}{B \vee A}$
- 7. (**EI**) $\dfrac{A, A \rightarrow B}{B}$ 8. (**II**) $\dfrac{B}{A \rightarrow B}$
- 9. (**DN**) $\dfrac{A}{\neg \neg A} \, si \, \dfrac{\neg \neg A}{A}$, ***c***: *ipoteza* A *este înlăturată*

6-24 (184)

- Noile mnemonice, folosite mai sus, sunt: **ED -** **e**liminarea **d**isjuncției, **ID** – **i**ntroducerea **d**isjuncției, **EI** – **e**liminarea **i**mplicației, **II** – **i**ntroducerea **i**mplicației, și, în sfârșit, **DN** – **d**ubla **n**egație
- Pentru a înțelege următorul exemplu mai sunt necesare câteva precizări, pe care le vom prezenta într-o formă particulară (pentru **SD0** și **LP**)
- Vom reveni în final, după cum am mai menționat, cu anumite generalizări

6-25 (185)

- **Definiție** (*consecință sintactică*). Fie $\mathcal{I} \subseteq$ **LP** și A $\in$ **LP**. A este consecință sintactică din $\mathcal{I}$ ($\mathcal{I} \vdash$ A) dacă există o deducție naturală (un arbore de ...) având *rădăcina* A și *toate ipotezele neanulate aparținând lui* $\mathcal{I}$.
- De altfel, **TCC0** are formularea mai generală: $\mathcal{I} \vdash$ A ddacă $\mathcal{I} \vDash$ A; anterior enunțul reprezenta cazul $\mathcal{I} = \emptyset$
- În exemplul următor se arată de fapt că formula C este consecință semantică și sintactică din $\mathcal{I} = \{\rceil(\rceil\rceil A \wedge \rceil C), \rceil(\rceil\rceil B \wedge \rceil C), \rceil(\rceil A \wedge \rceil B)\}$

6-26 (186)

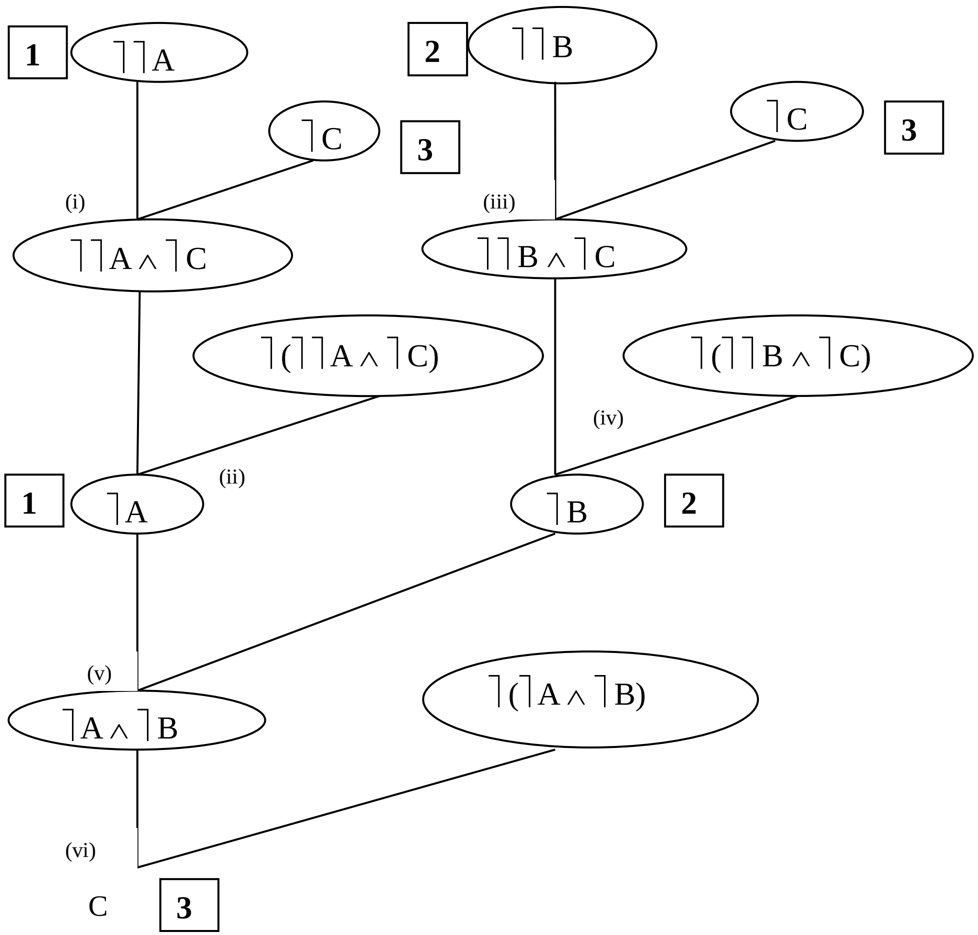
- Desigur că mai sunt ipoteze (frunze) în arbore (și anume $\rceil\rceil A$, $\rceil C$ (de două ori) și $\rceil\rceil B$), dar ele se anulează în cursul demonstrației (un pas al demonstrației reprezintă folosirea unei noi reguli de inferență, adică construirea unui nou nod în graf, împreună cu arcele corespunzătoare)
- Cu (i) – (vi), practic am numerotat pașii de demonstrație (și, de exemplu, (ii) și (iii) puteau fi „inversați”)
- „Mărcile” 1, 2, 3 (încercuite, atașate nodurilor), reprezintă „momentul” în care s-a anulat o ipoteză

6-27 (187)

- O aceeași marcă este atașată atât concluziei regulii de inferență care „aplică o anulare”, cât și ipotezei anulate
- De exemplu, marca 1 este atașată nodului $\neg A$ obținut în pasul (ii) după aplicarea (unei instanțe a) regulii (**EN**) (în care B este $\neg\neg A \wedge \neg C$, iar A este $\neg A$, anulându-se astfel ipoteza $\neg\neg A$), dar și ipotezei care se anulează, $\neg\neg A$
- Tot ca precizare pentru exemplul care urmează, în pasul notat (i) s-a aplicat (o instanță a) regula (regulii) (**IC**), cu $\neg\neg A$ „pe post de A” și cu $\neg C$ „pe post de B”
- Etc.

6-28 (188)

- **Exemplu**. Să arătăm în **SD0** „*validitatea secvenței*” (vezi și ceea ce mai urmează) $\rceil(\rceil\rceil A \wedge \rceil C), \rceil(\rceil\rceil B \wedge \rceil C), \rceil(\rceil A \wedge \rceil B) \vdash C$:



6-29 (189)

- Vom prezenta un ultim exemplu interesant, *calculul cu secvențe*, **SD1**, prezentat pentru elementele lui **LP1** ca formule „de bază” în construcția metaformulelor manipulate
- Eliminând părțile care implică cuantorii, slide-urile pot fi citite de pe acum („rămânând” cu **SD1** „proiectat” în **LP**)
- Pornim însă cu **LP1**, construit peste un alfabet care conţine toţi conectorii şi cuantificatorii cunoscuţi (desigur că unii dintre ei pot fi adoptaţi prin notaţie, dar îi vom folosi fără restricţie): $\urcorner$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$, etc.
- Se numeşte **secvenţă** orice formulă care are forma:

  $A_1 \wedge A_2 \wedge \ldots \wedge A_m \rightarrow B_1 \vee B_2 \vee \ldots \vee B_n$, unde n, m $\in$ **N**, $A_1, A_2, \ldots, A_m, B_1, B_2, \ldots, B_n \in$ **LP1** (m, n pot fi şi egali cu 0, dar nu simultan)

6-30 (190)

- Prin urmare, vom lucra practic cu „clauze”, dar notaţia adoptată mai jos ne conduce la ideea că *secvenţele sunt mai degrabă metaformule (alt tip de obiect, oricum* ***mai complex****) decât formulele cu care am fost familiarizaţi până în prezent* (elemente ale lui **LP**, de exemplu)
- Mai mult, o asemenea notație „separă” în mod explicit formulele (considerate a fi) *pozitive* de cele *negative* (nu presupunem însă neapărat folosirea unor forme normale de la început)
- Astfel, vom scrie o secvenţă sub forma:

$A_1, A_2, \ldots , A_m \Rightarrow B_1, B_2, \ldots\ldots , B_n$

6-31 (191)

- Vom considera cei doi membri ai relaţiei de mai sus ca fiind mulţimi și nu liste (atunci când ordinea elementelor va fi esenţială, vom specifica explicit acest lucru)
- Prin urmare, **o secvenţă va fi de forma *U* $\Rightarrow$ *V*** (repetăm, ***U* şi *V***, ca submulțimi de formule din **LP1**, **pot fi şi mulţimea vidă, dar nu simultan**) şi vom putea scrie *U'* = *U*, A în loc de *U'* = *U* U {A}, în ideea că, din anumite motive, elementul A din *U'* trebuie „pus în evidenţă"
- Vom extinde notaţia la submulţimi oarecare, adică vom putea scrie (de exemplu) *V*, *W* în loc de *V* U *W* şi *V*, A, B în loc de *V* U {A} U {B}

6-32 (192)

- În cele ce urmează, vom „pune”

  FORM = {*U* | *U* este secvenţă în **LP1**}
- Sistemul **SD1**, atribuit de obicei lui Gentzen (1934) şi având *o singură schemă de axiome* (este drept, foarte generală), se apropie însă practic (în privinţa modalităţii de utilizare) mai mult de un sistem de tip Hilbert
- Sistemele deductive bazate pe secvenţe („de tip **SD1**”) au şi o răspândită utilizare în situaţii nestandard, legate *doar* de definirea constructivă a unor mulţimi „în mod axiomatic”, fără a se face apel la vreo „semantică de adevăr” asociată (de exemplu: ***acceptăm ca fiind „adevărată” orice secvență care se poate demonstra***)
- **SD1** este un *sistem predicativ finit specificat şi boolean complet*, având:

6-33 (193)

- **Axiome ($\mathcal{A}_{\mathbf{SD1}}$).** Pentru fiecare *U*, *V* $\subseteq$ **LP1** şi pentru fiecare A $\in$ **LP1**: *U*, A $\Rightarrow$ *V*, A
- **Reguli de inferenţă ($\mathcal{R}_{\mathbf{SD1}}$).** Schemele de mai jos (care sunt numerotate, dar au ataşat şi un nume mnemonic (nu necesită explicaţii suplimentare), exceptând, poate, (**RT**) care înseamnă regula tăieturii), sunt valabile pentru fiecare *U*, *V* $\subseteq$ **LP1**, fiecare A, B $\in$ **LP1**, fiecare x $\in$ **X** şi fiecare t $\in$ **T** (a se vedea a doua parte a cursului)
- În regula 5., substituţia [x/t] trebuie să fie permisă pentru A, iar în 6., x nu trebuie să apară liber în nici o formulă din *U* sau *V*
- Atât axiomele, cât și premizele și concluzia fiecărei reguli de inferență, sunt metaformule (elemente din **FORM**)

6-34 (194)

- În momentul în care vor exista mai multe premize într-o regulă, vom folosi pentru separarea lor „;” (pentru evitarea confuziilor)
- Atenţie la faptul că regulile 5. şi 7. **au o infinitate de premize** (de exemplu, *U*, (A)[x/t] $\Rightarrow$ *V* din ($\forall\Rightarrow$) trebuie înţeles ca reprezentând

  U, (A)[x/t1] $\Rightarrow$ *V*; *U*, (A)[x/t2] $\Rightarrow$ *V*; ... , adică se iau în considerare ***toate*** elementele t din **T**, pentru care substituția [x/t] este permisă pentru A; rolul lui A în (**RT**) este similar, instanţele unei scheme

  referindu-se la celelalte elemente având statutul de a fi „oarecare” (adică nume generice)

6-35 (195)

1. $(\neg \Rightarrow)$ $\dfrac{U \Rightarrow V, A}{U, \neg A \Rightarrow V}$

2. $(\wedge \Rightarrow)$ $\dfrac{U, A, B \Rightarrow V}{U, A \wedge B \Rightarrow V}$

3. $(\Rightarrow \neg)$ $\dfrac{U, A \Rightarrow V}{U \Rightarrow V, \neg A}$

4. $(\Rightarrow \wedge)$ $\dfrac{U \Rightarrow V, A; U \Rightarrow V, B}{U \Rightarrow V, A \wedge B}$

5. $(\forall \Rightarrow)$ $\dfrac{U, (A)[x/t] \Rightarrow V}{U, (\forall x)A \Rightarrow V}$

6. $(\Rightarrow \forall)$ $\dfrac{U \Rightarrow V, A}{U \Rightarrow V, (\forall x)A}$

7. (**RT**) $\dfrac{U, A \Rightarrow V; U \Rightarrow V, A}{U \Rightarrow V}$

- **Exemple** - slide-ul care urmează; voi – pentru alte detalii
- *Corectitudine și completitudine* în **SD1**: $\vdash_{\mathbf{SD1}} F$ (adică $F \in$ **LP1** este formulă validă) dacă şi numai dacă $\emptyset \Rightarrow F$ este teoremă în **SD1**

6-36 (196)

- **Exemplu**. Regula (↔⇒) se poate obţine pe baza deducţiei (metaformulele *U*, B $\Rightarrow$ *V*, B şi *U*, A $\Rightarrow$ *V*, A sunt axiome):

$$\dfrac{\dfrac{\dfrac{U, A, B \Rightarrow V \qquad U, B \Rightarrow V, B}{U, A \to B, B \Rightarrow V} \qquad \dfrac{U, A \Rightarrow V, A \qquad U \Rightarrow V, A, B}{U, A \to B \Rightarrow V, A}}{U, A \to B, B \to A \Rightarrow V}}{U, (A \to B) \wedge (B \to A) \Rightarrow V}$$

6-37 (197)

- Având și exemple, putem încheia secțiunea privind legătura dintre teoriile logice și sistemele de demonstrație cu câteva precizări/reluări importante
- Să ne gândim astfel la reprezentarea prin (meta)formule a unei ***baze de cunoştinţe***
- Din păcate nu există metode semantice efective (algoritmice) convenabile pentru a testa aprioric dacă o mulţime dată de (meta)formule este sau nu închisă la consecinţă semantică, sau dacă o anumită (meta)formulă este satisfiabilă sau validă
- Alternativa este de a folosi metode sintactice, care au avantajul că pot fi totuşi *automatizate* (măcar parţial)

6-38 (198)

- În acest context, se poate pune de exemplu problema *axiomatizării teoriilor logice, cu ajutorul sistemelor de demonstraţie*
- Acest lucru înseamnă că având dată o teorie logică **TE** $\subseteq$ **FORM** (de exemplu, mulţimea $\mathcal{Val}$(**LP1**), a formulelor valide din **LP1**), trebuie să găsim o submulţime $\mathcal{A}'$ **=** $\mathcal{A}$ U $\mathcal{I} \subseteq$ **TE** de *axiome şi/sau ipoteze suplimentare, precum şi o mulţime de reguli de inferenţă* $\mathcal{R}$ (adică un *sistem de demonstraţie* **SD' =** <$\mathcal{A}'$, $\mathcal{R}$>**)** astfel încât **TE =** $\mathcal{Th}$**(SD')**

6-39 (199)

- În acest caz, se impune de obicei ca $\mathcal{A}'$ să fie măcar o mulţime satisfiabilă (există măcar o structură **S** astfel încât pentru fiecare F $\in \mathcal{A}'$ avem **S**(F) = 1), sau chiar validă (*dacă $\mathcal{A}'$ conţine măcar o contradicţie, atunci orice (meta)formulă este consecinţă semantică din $\mathcal{A}'$*)
- Forma generală a lui $\mathcal{A}'$ se explică prin aceea că, *de obicei*, *se așteaptă ca $\mathcal{A}$ să conţină formule **valide** iar $\mathcal{I}$ formule **satisfiabile*** (odată fixată o noțiune formală de ***adevăr și de structură***)

6-40 (200)

- Mai general, să presupunem că pornim cu o mulţime de (meta)formule $\mathcal{A}' \subseteq$ **FORM**, de *cunoştinţe primare, unanim acceptate ca fiind „adevărate"*, adică despre care *ştim* (nu ne interesează deocamdată prin ce metodă am aflat acest lucru) că reprezintă formule valide /satisfiabile, în contextul descris mai sus
- Pentru a axiomatiza teoria dată, trebuie să mai găsim şi o mulţime (scheme de) *reguli de inferenţă* $\mathcal{R}$ astfel încât să avem $\mathcal{Cs}(\mathcal{A}')$ **=** $\mathcal{Th}$(**SD'**) (am notat cu $\mathcal{Cs}(\mathcal{A}')$ mulţimea tuturor consecinţelor semantice din $\mathcal{A}'$, în raport cu noţiunea de adevăr adoptată, şi cu **SD'** sistemul deductiv $<\mathcal{A}',\mathcal{R}>$)

6-41 (201)

- Putem considera şi situaţia inversă, în care avem dat un sistem **SD’** = $<\mathcal{A}', \mathcal{R}>$ şi dorim să ne convingem că Th(**SD’**) este într-adevăr o teorie logică, şi, mai mult, să știm dacă $\mathit{Cs}(\mathcal{A}')$ **=** Th**(SD’**)
- **Definiţie**. Un *sistem de demonstraţie* **SD’** = $<\mathcal{A}', \mathcal{R}>$ se numeşte ***corect şi complet*** *pentru o teorie* **TE** dacă **TE** = Th(**SD’**) = $\mathit{Cs}(\mathcal{A}')$ şi $\mathcal{A}' \subseteq$ **TE**. O *teorie* **TE** este *axiomatizabilă* dacă există un sistem deductiv **SD’** = $<\mathcal{A}', \mathcal{R}>$ corect şi complet pentru ea, adică satisfăcând condiţiile anterioare.
- Dacă **SD’** este finit (schemele ...) specificabil (axiomatizabil), atunci teoria corespunzătoare se numeşte *finit axiomatizabilă*

6-42 (202)

- În cele de mai sus, dacă $\mathcal{I}$ este mulţimea vidă atunci **TE** este/ar trebui să fie alcătuită doar din (meta) formule valide
- În cazul teoriilor „reale”, $\mathcal{I}$ cuprinde în general cunoştinţele primare ale lumii respective (vezi „aritmetica Presburger” …), iar $\mathcal{A}$ axiomele „(pur) logice” (de genul celor „puse” în **SD3**)
- **Teoremă** (**de corectitudine şi completitudine – forma generală**). Fie o clasă de (meta)formule **FORM**, o clasă de structuri „admisibile”, $\mathcal{S}tr$, pentru **FORM** (prin care se defineşte de fapt noţiunea de *adevăr*), un sistem deductiv **SD’** = <$\mathcal{A}$’, $\mathcal{R}$> în **FORM**, cu $\mathcal{A}$’ = $\mathcal{A}$ U $\mathcal{I}$ ($\mathcal{A}$ fiind alcătuită din formule valide şi $\mathcal{I}$ din formule satisfiabile) şi o teorie logică **TE** $\subseteq$ **FORM**, astfel încât **TE** = $\mathcal{C}s$($\mathcal{A}$’). Atunci $\mathcal{T}h$(**SD’**) = $\mathcal{C}s$($\mathcal{A}$’).

6-43 (203)

- **Observaţie**. A demonstra corectitudinea înseamnă a arăta că $\mathcal{Th}$(**SD'**) $\subseteq$ $\mathcal{Cs}$($\mathcal{A}$') iar completitudinea, că $\mathcal{Th}$(**SD'**) $\supseteq$ $\mathcal{Cs}$($\mathcal{A}$')
- Teorema se mai poate enunţa şi sub forma (destul de des întâlnită): ***Teoria*** **TE** (de multe ori, ea coincide cu $\mathcal{Val}$(X), X fiind o „logică dată”) ***admite un sistem deductiv corect şi complet***
- Sau chiar: ***pentru fiecare (meta)formulă*** F$\in$ **FORM**, ***avem*** $\mathcal{I} \vdash_{\mathbf{SD}}$ F ***ddacă*** $\mathcal{I} \models$ F
- Practic, este de dorit ca teoria ***dorită*** **TE** să fie (eventual, chiar finit) axiomatizabilă

6-44 (204)

- În cazul în care este vorba de o teorie formată doar din formule valide (atunci va lipsi $\mathcal{I}$), teorema capătă forma simplificată:

  Pentru fiecare F$\in$ **FORM**, ***avem***: $\vdash_{\mathbf{SD}}$ F ***ddacă***

  $\models$ F
- În cele de mai sus am folosit notaţia $\vdash_{\mathbf{SD}}$ F pentru a exprima faptul că F $\in \mathcal{Th}$(**SD**), unde

  SD = <$\mathcal{A}$, $\mathcal{R}$>, sau, în momentul în care **SD** este implicit, corect și și complet, $\vdash$ F poate nota chiar faptul că F este o formulă validă
- **Teoremă (teorema de completitudine a lui K. Gödel, 1930)**. $\mathcal{I} \vdash_{\mathbf{SD3}}$ F dacă şi numai dacă

  $\mathcal{I} \models$ F (pentru fiecare $\mathcal{I} \subseteq$ **LP1**).

6-45 (205)

- Iată alte câteva rezultate interesante
- **Teoremă**. Sistemul **SD0** este corect şi complet pentru $\mathcal{Val}$(**LP1**).
- **Teoremă**. Sistemele **SD0** şi **SD3** sunt echivalente. Mai mult, pentru fiecare mulţime de formule închise $\mathcal{J} \subseteq$ **LP1** şi fiecare formulă F $\in$ **LP1**, avem:
  $\mathcal{J} \vdash_{\mathbf{SD0}}$ F dacă şi numai dacă $\mathcal{J} \vdash_{\mathbf{SD3}}$ F.
- **Teoremă**. Fie orice F $\in$ **LP1**. Atunci:
  $\vdash_{\mathbf{SD1}}$ F dacă şi numai dacă $\models$F (de fapt, în sensul precizat acolo ...).

6-46 (206)

- Ca o concluzie, d.p.d.v. practic avem nevoie de teorii „puternice” (care să conţină măcar „aritmetica”, „egalitatea” şi axiomele logice „primare” Presburger…); acestea, deși „mai simple” sunt însă ***incomplete*** (*nu tot ceea este adevărat este demonstrabil* ...), din păcate (există și o teoremă de incompletitudine a lui K. Gödel ...)

6-47 (207 – final 6)

- Din acest curs, nu vor fi exerciții explicite pentru lucrarea de verificare (a se vedea **link-ul** la „Material seminarii")
- Defințiile și conceptele „primare" trebuie însă **reținute**
- Se studiază în ***mod global clase de formule*** (**LP**; uneori, **FORM** sau **LP1**)
- **Sintactic**: *sistemele deductive* (**SD**)
- **Semantic**: *teoriile logice* (**TE**; în special cazul **TE** = $\mathcal{Val}$(**FORM**))
- Legăturile dintre **SD** și **TE**: **TCC** - *teoreme de corectitudine și completitudine*; *rezolvarea* **SAT** *semantic și sintactic*
- Construcția unei logici (**FORM**); în cadrul dat, a unei **TE** pornind de la un **SD** și reciproc
- Sunt prevăzute exemple edificatoare (**SD3**, **SD0** și **SD1**), folosind un cadru foarte general
- Acestea nu trebuie memorate explicit, iar la **SD0**, sub o formă ușor modificată (***deducția naturală***) revenim în cursul următor

7-1 (208)

- Cartea lui **Huth /Ryan** (paginile numerotate 1-30)
- Reluăm **SD0** (***deducția naturală***) dintr-o nouă perspectivă
- **Exemplul 1**. Dacă trenul ajunge mai târziu și nu sunt taxiuri în stație, atunci Ion va întârzia la întâlnirea fixată. Ion nu întârzie la întâlnire. Trenul ajunge într-adevăr mai târziu. *Prin urmare*, erau taxiuri în stație. (*Informal, e „bun”*)

p: trenul ajunge mai târziu

q: sunt taxiuri în stație

r: Ion întârzie la întâlnirea fixată

- **Raționament = demonstrație**
- $(p \wedge (\neg q)) \rightarrow r, \neg r, p \vdash q$ (***secvențe***, ***premize***, ***concluzie***; *atomi*, *negația*, *clauza vidă*, *formule*, *clase de formule*; ce știam, *consecință sintactică* ... chiar fără „ceva” precizat cu exactitate a fi formal)

7-2 (209)

- În general, acum: $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ (secvență); p, q, … (înainte: A, B, …); $\neg$ ($\rceil$); $\bot$ ($\square$); φ, ψ, ... (F, G, …); Φ, Ψ,… ($\mathcal{F}$, $\mathcal{G}$, …)
- Demonstrațiile prin deducție naturală sunt „mai complexe”, „lucrând” chiar asupra raționamentelor
- Ele pot include (implicit, sau chiar explicit, fără a apela la noțiunea de arbore) alte demonstrații, care pot fi exprimate prin noi reguli de inferență (*reguli derivate*, în sensul anterior)
- **H/R**: *By applying proof rules to the premises, we hope to get some more formulae, and by applying more proof rules to those, to eventually obtain the conclusion* (atenție – constructivism ...)

7-3 (210)

- Relativ la sintaxă (***semantica o vom utiliza doar intuitiv***, *în substrat*), prezentăm și *forma Backus-Naur* (**BNF**) a **LP** folosită în **H/R** (până la „dezvoltarea” lui **LP** „în” **LP1**):

  $\varphi ::= p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$

- Gramatici, cuvinte, limbaje (sub „altă viziune”)...
- **Observație**. Pentru o scriere „textuală” mai simplă, în „Tabelul tuturor celor 16 reguli” (care va urma imediat) vom denota un *dreptunghi /cutie* (vezi exemplele care vor urma și ele) prin

  box(σ, τ), unde σ este *presupunerea*, iar τ este „concluzia” demonstrației înscrise în cutie

7-4 (211)

- Revenind, nu este simplu să știm care reguli/*șabloane* trebuie aplicate și în ce ordine
- Sau, am putea folosi în raționamente șabloane *unsound /invalide* (p, q $\vdash$ p $\wedge$ $\neg$q; !?)
- Prezentăm cele 16 reguli de inferență **H/R** (Fig. 1.2, p.27) pentru deducția naturală (ultimele 4 sunt însă reguli derivate), „numite”:

  ($\wedge$**i**), ($\wedge$**e**$_1$), ($\wedge$**e**$_2$), ($\vee$**i**$_1$), ($\vee$**i**$_2$), ($\vee$**e**), ($\rightarrow$**i**), ($\rightarrow$**e**), ($\neg$**i**), ($\neg$**e**), ($\perp$**e**), ($\neg\neg$**e**); respectiv (**MT**), ($\neg\neg$**i**), (**PBC**), și (**LEM**)
- Cum știm (?), **i** „vine” de la **i***ntroducere* și **e** – de la **e***liminare* (*restul simbolurilor ...*)
- În cele derivate: **M**odus **T**ollens (*modul negativ*), **P**roof **B**y **C**ontradiction (*reductio ad absurdum*), și respectiv **L**aw (of the) **E**xcluded **M**iddle (*tertium non datur*)
- Toate sunt *sound /valide /corecte* ... („discuții” ...)

7-5 (212)

$$\frac{\varphi \qquad \psi}{\varphi \wedge \psi}\ (\wedge\mathbf{i}) \qquad \frac{\varphi \wedge \psi}{\varphi}\ (\wedge\mathbf{e_1}) \qquad \frac{\varphi \wedge \psi}{\psi}\ (\wedge\mathbf{e_2})$$

$$\frac{\varphi}{\varphi \vee \psi}\ (\vee\mathbf{i_1}) \qquad \frac{\psi}{\varphi \vee \psi}\ (\vee\mathbf{i_2}) \qquad \frac{\varphi \vee \psi \quad \mathbf{box}(\varphi, \theta) \quad \mathbf{box}(\psi, \theta)}{\theta}\ (\vee\mathbf{e})$$

$$\frac{\mathbf{box}(\varphi, \psi)}{\varphi \to \psi}\ (\to\mathbf{i}) \qquad \frac{\varphi \qquad \varphi \to \psi}{\psi}\ (\to\mathbf{e})$$

7-6 (213)

$$\frac{\textbf{box}(\varphi, \perp)}{\neg\varphi}\ (\neg \textbf{i}) \qquad\qquad \frac{\varphi \qquad \neg\varphi}{\perp}\ (\neg \textbf{e})$$

$$\frac{\perp}{\varphi}\ (\perp \textbf{e}) \qquad\qquad \frac{\neg\neg\varphi}{\varphi}\ (\neg\neg \textbf{e})$$

- În sfârșit, iată cele 4 reguli *derivate* amintite

7-7 (214)

$$\frac{\varphi \to \psi \qquad \neg\psi}{\neg\varphi}\ (\mathbf{MT}) \qquad\qquad \frac{\varphi}{\neg\neg\varphi}\ (\neg\neg\mathbf{i})$$

$$\frac{\mathbf{box}(\neg\varphi, \bot)}{\varphi}\ (\mathbf{PBC}) \qquad\qquad \frac{}{\varphi \vee \neg\varphi}\ (\mathbf{LEM})$$

- Înainte de exemplificări, vom prezenta *intuiția procedurală* (**CUM** …) din „spatele” regulilor
- Pe parcursul exemplelor, va deveni transparentă și *intuiția* de tip *declarativ* (**CE** …)

7-8 (215)

- ($\wedge$**i**): pentru a demonstra $\varphi \wedge \psi$, trebuie mai întâi să demonstrăm separat φ și ψ (apoi se folosește regula)
- ($\wedge$**e**$_1$): pentru a demonstra φ, se încearcă a se demonstra $\varphi \wedge \psi$ și apoi se folosește regula; acesta nu pare a fi un sfat prea bun, deoarece va fi probabil mai greu de demonstrat $\varphi \wedge \psi$ decât de a demonstra doar φ; totuși, dacă „ne uităm în jur” și „vedem” că $\varphi \wedge \psi$ a fost deja demonstrată ...
- ($\vee$**i**$_1$): pentru a arăta $\varphi \vee \psi$, se încearcă să se demonstreze φ; ca mai înainte, ar putea fi mai greu să se demonstreze φ (doar dacă nu cumva ... era) decât să se demonstreze $\varphi \vee \psi$; astfel, dacă vrem să arătăm $q \vdash p \vee q$, nu vom putea folosi doar pe ($\vee$**i**$_1$), simplu, dar ($\vee$**i**$_2$) probabil va „funcționa” …

7-9 (216)

- ($\vee$**e**): dacă „avem” deja $\varphi \vee \psi$ și dorim să „arătăm o propoziție” θ, e nevoie să să arătăm θ atât „din” φ, cât și „din” ψ (folosind, eventual, și celelalte *premize /*sau *presupuneri* avute deja la dispoziție); nu știm care este adevărată ...
- ($\rightarrow$**i**): este ceva similar cu ceea ce aveam mai înainte; adică, dacă vrem să demonstrăm $\varphi \rightarrow \psi$, pare mai util să încercăm să demonstrăm pe ψ, pornind cu faptul că avem deja demonstrat φ (mai ... simplă ...)
- ($\rceil$**i**): pentru a arăta $\rceil\varphi$, este (poate mai) bine să demonstrăm $\perp$ ***/false*** „din” φ (adică **box**(φ, $\perp$))

7-10 (217)

- Începem exemplele cu regulile ($\wedge$**i**), ($\wedge$**e**$_1$), și ($\wedge$**e**$_2$)
- În context, $\perp$ („clauza” /formula vidă, □) reprezintă *orice* formulă ***false***, adică de forma $\varphi \wedge \neg\varphi$ (sau $\neg\varphi \wedge \varphi$; diferență de sintaxă; n-avem încă semantică ...)
- **Exemplul 2**. Arătați că *secvența* $p \wedge q, r \vdash q \wedge r$ *este validă* (cuvântul nu are încă semnificația anterioară, legată de adevăr); sau, *construiți demonstrația* ...
- Mod de transcriere a unei secvențe: $p \wedge q$

  r

  spații (interlinii)

  $q \wedge r$
- ***A construi o demonstrație înseamnă a completa spațiile aflate între premize și concluzie*** (*prin aplicarea unei secvențe „potrivite” de reguli*)

7-11 (218)

- Obținem (ușor ? cum ?)

1	$p \wedge q$	*premiză*	
2	r	*premiză*	
3	q	**($\wedge$e$_2$)**	1
4	$q \wedge r$	**($\wedge$i)**	3, 2

- Desigur că φ și ψ din the „tabelul cu reguli” (**H/R**, pag. 27), pot fi (în general) „instanțiate” nu doar cu formule atomice (sunt „scheme” ...; „substituții” ...)
- Demonstrația precedentă poate fi prezentată și printr-un arbore etichetat (desen …), având rădăcina (etichetată cu) $q \wedge r$ și frunzele $p \wedge q$, și r (surpriză ?!):

7-12 (219)

$$\dfrac{\dfrac{p \wedge q}{q}\,(\wedge e_2) \qquad r}{q \wedge r}\,(\wedge i)$$

- Desigur că pot exista diferite modalități de a *demonstra o concluzie*, *construi o demonstrație*, *găsi o secvență validă* /raționament corect (liste diferite /arbori diferiți, o ordine diferită de aplicare a regulilor, reguli diferite…)
- **Important**: orice „presupusă” demonstrație poate fi **verificată** că este **corectă** (în sensul aplicării corecte a tuturor regulilor, pe parcursul dezvoltării ei), prin „controlul” fiecărei linii în parte (începând de sus, în reprezentarea liniară)

7-13 (220)

- În exemplul precedent am folosit doar *„și”-regulile*
- Să continuăm cu explicarea regulilor referitoare la dubla negație (i.e. ($\neg\neg$**e**) și „derivata” ($\neg\neg$**i**))
- Nu avem semantică formală, dar, intuitiv, nu există vreo diferență între formulele φ și $\neg\neg\varphi$: *The sentence* “It is **not** true that it does **not** rain” *is just a more sophisticated way of saying* “It rains” (*and*…*conversely*…)
- **Exemplul 3**. Demonstrați singuri validitatea secvenței:

  $p, \neg\neg(q \wedge r) \vdash \neg\neg p \wedge r.$
- **Indicație**: Se pornește cu ambele premize ...

7-14 (221)

- Iată, totuși, o demonstrație pentru **Exemplul 3**:

1	p	*premiză*
2	$\neg\neg(q \wedge r)$	*premiză*
3	$\neg\neg p$	($\neg\neg$**i**) 1
4	$q \wedge r$	($\neg\neg$**e**) 2
5	r	($\wedge$**e$_2$**) 4
6	$\neg\neg p \wedge r$	($\wedge$**i**) 3, 5

- **Exemplul 4**. Demonstrați secvența (preferabil singuri; apoi comparați ceea ce ați făcut voi cu soluția ce urmează): $(p \wedge q) \wedge r, s \wedge t \vdash q \wedge s$.

7-15 (222)

1	$(p \wedge q) \wedge r$	*premiză*
2	$s \wedge t$	*premiză*
3	$p \wedge q$	**($\wedge$e$_1$)** 1
4	q	**($\wedge$e$_2$)** 3
5	s	**($\wedge$e$_1$)** 2
6	$q \wedge s$	**($\wedge$i)** 4, 5

- A voastră care este ? Sunt diferențe ? Care ? De ce ?!
- Aici – de dat „copii” ale paginii /paginilor (26-)27 din **H/R** (regulile deducției naturale)

7-16 (223)

- Mai este mult de „muncit” la explicarea (declarativă - **CE** /imperativă, computațională – **CUM**, a) regulilor de inferență și construcția raționamentelor corecte, chiar înainte de introducerea (în carte a) sintaxei /semanticii formale (am făcut câteva „**CUM**”)
- Vom insista mai mult asupra explicării lui (**→i**) și a regulilor folosite pentru disjuncție și negație
- Vom analiza (**MT**), ca regulă derivată și vom discuta *echivalența* (*sintactică a raționamentelor*) *prin demonstrație*
- Începem cu (**→i**) deoarece construirea unei implicații corecte este evident mai provocatoare decât „distrugerea”/eliminarea uneia ((**MP**) /(**→e**))

7-17 (224)

- Ignorând pe moment faptul că (**MT**) este o regulă derivată, datorită ei putem spune că secvența $p \to q, \urcorner q \vdash \urcorner p$ este validă
- *If Abraham Lincoln was Ethiopean, then he was African. Abraham Lincoln was not African; therefore he was not Ethiopean.*
- Deci, la fel ca (**MT**), pare la fel de plauzibil că și secvența $p \to q \vdash \urcorner q \to \urcorner p$ este validă, ele „spunând cam același lucru”
- Mai în amănunt, plecând cu $p \to q$ ca premiză și *presupunând* ***temporar*** că $\urcorner q$ este „adevărată” (altă premiză, dar ...), putem chiar demonstra afirmația anterioară, sub forma (**Exemplul 5**):

7-18 (225)

1 $p \to q$ *premiză*

2 $\neg q$ *presupunere (premiză temporară)*

3 $\neg p$ (**MT**) 1, 2

4 $\neg q \to \neg p$ ($\to$**i**) 2-3

- În cele de mai sus, prin subliniere s-a „marcat” faptul că $\neg q$ și $\neg p$ constituie (pe verticală, în această ordine) dreptunghiul care reprezintă ipoteza regulii ($\to$**i**), iar 2-3 specifică prima și ultima formulă din dreptunghi
- Dreptunghiul specifică de fapt domeniul sintactic al presupunerii temporare $\neg q$ (similar: *subprogram*, *variabilă locală*)

7-19 (226)

- În concluzie, am procedat astfel: am făcut presupunerea $\neg q$, „deschizând” un dreptunghi și „punând $\neg q$ deasupra”; am continuat să aplicăm reguli în mod normal (ca în construcția oricărei demonstrații, dar în „interiorul dreptunghiului”), ultima *care depinde de* $\neg q$ (aici - și singura) fiind (**MT**), prin care am creat $\neg p$ (adică, și $\neg p$ „merge” în interiorul dreptunghiului); apoi putem „ieși” din dreptunghi, deoarece nici regula aplicată, ($\rightarrow$**i**), nici concluzia $\neg q \rightarrow \neg p$, nu mai depind de presupunerea temporară $\neg q$

7-20 (227)

- Ca exemplu, (re)faceți raționamentul pentru:

  p = You are French

  q = You are European
- Într-un alt context, ne putem gândi la p $\rightarrow$ q ca denotând tipul unei proceduri date; de exemplu, *propoziția* p *ar putea exprima faptul că procedura considerată așteaptă să primească la intrare o valoare intreagă* x, iar *propoziția* q *faptul că, la ieșire, aceeași procedură va returna o valoare booleană* y
- Atunci, „validitatea” lui p $\rightarrow$ q apare acum ca un fel de „adevăr” al unei aserțiuni de forma presupune-garantează:

7-21 (228)

Dacă intrarea procedurii este un număr întreg, atunci ieșirea va fi o valoare booleană

- Acest lucru nu înseamnă că, aceeași procedură, nu poate face „lucruri trăznite” sau nu se poate „bloca” dacă intrarea a nu va fi un număr întreg
- Din motivele amintite mai sus (mai există și altele !), regula ($\rightarrow$**i**) arată ca având drept ipoteză un „prim” dreptunghi (care *începe cu formula indicată prin* φ *și se termină cu formula indicată prin* ψ, *între ele putând însă exista alte demonstrații*), iar drept concluzie implicația

$\varphi \rightarrow \psi$

7-22 (229)

- Adică (intuitiv !): ***Pentru a introduce o implicație*** *într-un raționament /„demonstra validitatea” / presupune „adevărul”* (etc.) *lui*

  $\varphi \rightarrow \psi$, ***se face o presupunere temporară asupra lui*** φ (că este „adevărată”, ...) și ***apoi se demonstrează*** ψ („renunțându-se” apoi la φ)
- Pentru că partea cu „se demonstrează” este conținută în acele „puncte, puncte” din interiorul dreptunghiului, trebuie respectate niște reguli sintactice concrete (suplimentare), începând cu faptul că „acolo”, se poate folosi φ și orice alte formule „valide”, inclusiv premize sau concluzii provizorii făcute /obținute până la momentul /punctul respectiv al demonstrației

7-23 (230)

- În general, o formulă θ se poate folosi la un punct din (linie de) demonstrație, doar dacă acea formulă a „apărut” (măcar ca premiză) în demonstrație înainte de punctul de folosire și dacă niciun dreptunghi în interiorul căruia se găsește (acea apariție a lui) θ nu a fost deja închis
- În cursul unei demonstrații oarecare, pot exista doar dreptunghiuri imbricate sau disjuncte, și nu care se intersectează (*deschide un nou dreptunghi de-abia după ce s-a închis precedentul*, sau *deschide-l și închide-l, dacă precedentul deschis nu a fost încă închis*)

7-24 (231)

- Linia care urmează imediat celei prin care se „închide” un dreptunghi, trebuie să respecte „forma” (*pattern*-ul) concluziei (sintactic, ca formulă) regulii de inferență care utilizează acel dreptunghi ca ipoteză
- De exemplu, pentru regula r = ($\rightarrow$**i**), aceasta înseamnă că imediat după un dreptunghi închis (care începe cu φ și se termină cu ψ), care este ipoteza unei instanțe a lui r, trebuie să continuăm numai cu $\varphi \rightarrow \psi$ (și pentru celelalte două inferențe „primare”, ($\vee$**e**) și ($\urcorner$**i**), și cea derivată (**PBC**), lucrurile sunt simple; mai avem și alte exemple în Curs...)

- **Exemplul 6**. Arătați că secvența $\neg q \to \neg p \vdash p \to \neg\neg q$ este validă

  (în demonstrația de mai jos am putut folosi, în linia 4, (**MT**), pentru formule „din dreptunghi”, sau „de deasupra” lui, deoarece înaintea acestei linii 4 nu exista niciun alt dreptunghi închis care să conțină liniile referite, adică 1 și 3)

1 $\neg q \to \neg p$ *premiză*

2 p *presupunere*

3 $\neg\neg p$ ($\neg\neg$**i**) 2

4 $\neg\neg q$ (**MT**) 1, 3

5 $p \to \neg\neg q$ ($\to$**i**) 2-4

7-26 (233)

- **Observație**. Ce ar însemna demonstrația (neinterzisă!) de o linie:

  1 $\quad p \quad$ *premiză* $\quad$?

  O interpretare imediată (corectă „prin lipsă”) este evident aceea că ea dovedește validitatea secvenței

  $p \vdash p$.

- Pe de altă parte, regula ($\rightarrow$**i**), care este o schemă de regulă (cu concluzia $\varphi \rightarrow \psi$), nu exclude posibiltatea ca φ să coincidă cu ψ și ambele să fie instanțiate la p
- În acest mod, am putea „extinde” demonstrația imediat anterioară, la:

  1 $\quad$ <u>p $\qquad$ *presupunere*</u>

  2 $\quad p \rightarrow p \qquad$ ($\rightarrow$**i**) 1-1

7-27 (234)

- Iar așa ceva s-ar putea interpreta (tot „prin lipsă”) ca fiind ***demonstrația validității secvenței*** $\vdash p \rightarrow p$ (sau ... $\varnothing \vdash p \rightarrow p$), prin aceasta „argumentându-se” faptul că „adevărul” unei formule/ afirmații de genul $p \rightarrow p$ este „universal”, el nedepinzând de vreo premiză sau presupupunere suplimentară
- **Definiție**. Orice formulă φ (din **LP**) pentru care secvența $\vdash \varphi$ este (s-a demonstrat a fi) validă, se numește ***teoremă***.
- Prin urmare (*repetăm*: fără a folosi încă o semantică formală), putem „reconstrui” conceptele sintactice ale unei logici (deocamdată, vorbim doar de **LP**, dar ...) pornind direct cu analiza conceptului de *raționament* (și nu cu cel de *formulă*)

7-28 (235)

- **Exemplul 7**. Arătați că este teoremă formula

  $\varphi = (q \to r) \to ((\rceil q \to \rceil p) \to (p \to r))$

  box1...

1 $q \to r$ *presupunere*

box2...

2 $\rceil q \to \rceil p$ *presupunere*

box3...

3 p *presupunere*

4 $\rceil\rceil p$ ($\rceil\rceil$**i**) 3

5 $\rceil\rceil q$ (**MT**) 2, 4

6 q ($\rceil\rceil$**e**) 5

7 r ($\to$**e**) 1, 6

sfbox3...

8 $p \to r$ ($\to$**i**) 3-7

sfbox2...

9 $(\rceil q \to \rceil p) \to (p \to r)$ ($\to$**i**) 2-8

sfbox1...

10 φ ($\to$**i**) 1-9

7-29 (236)

- **Observație**. Din exemplul precedent vedem cum o demonstrație pentru validitatea secvenței $\varphi_1, \varphi_2, ..., \varphi_n \vdash \psi$ se poate transforma într-o demonstrație a teoremei $\theta = \varphi_1 \to (\varphi_2 \to (\varphi_3 \to (... (\varphi_n \to \psi)...)))$; acest lucru se realizează prin „îmbogățirea” primei demonstrații cu *n* linii generate de regula (**→i**) care trebuie aplicată, pe rând și în această ordine, formulelor $\varphi_n, \varphi_{n-1}, ..., \varphi_1$.
- În plus, dreptunghiurile imbricate din **Exemplul 7** sugerează un tipar de demonstrație: acela de a folosi mai întâi regulile de eliminare (pentru a „distruge” /renunța la presupunerile făcute, care nu sunt premize obligatorii), și abia apoi de a folosi regulile de introducere (pentru a construi concluzia finală)

7-30 (237)

- Desigur că pentru demonstrațiile mai complicate ar fi nevoie să apelăm la tiparul descris în diverse faze de construcție, și că, de multe ori, o demonstrație apare la fel ca „un iepure din pălărie”
- Discuția merită aprofundată (vezi și **H/R** !) și „morala” ar fi că ***structura sintactică a unei posibile teoreme ne „spune o mulțime de lucruri” despre structura unei posibile demonstrații de validitate*** (și deci, acea structură merită „disecată și exploatată” oricât de mult)

7-31 (238)

- „Conform planului”, este momentul să comentăm puțin modalitățile de introducere a regulilor de inferență pentru *disjuncție* și *negație*
- Poate surprinzător, „spiritul” regulilor de inferență aferente disjuncției diferă profund de cel al conjuncției
- Cazul conjuncției este clar și concis: pentru a obține o demonstrație pentru $\varphi \wedge \psi$, trebuie folosită, la final, regula ($\wedge$**i**), practic doar pentru a concatena o demonstrație pentru φ cu o demonstrație pentru ψ
- În privința disjuncției însă, introducerea ei este, de departe, mult mai ușoară decât eliminarea

7-32 (239)

- Avem astfel nevoie atât de ($\vee$**i**$_1$) cât și de ($\vee$**i**$_2$) doar din motive sintactice, comutativitatea disjuncției nefiind stipulată explicit
- Apoi, conform semanticii intuitive a lui „*sau* neexclusiv”,

  $\varphi \vee \psi$ va fi „adevărată” dacă măcar una dintre φ și ψ este „adevărată” (indiferent de situația reală, alegerea regulii depinde de ceea ce avem la dispoziție la un moment dat în demonstrație)
- Cât despre eliminarea lui *sau*, ce putem să spunem despre folosirea unei formule de forma $\varphi \vee \psi$ într-o demonstrație, în ideea că ne păstrăm principiul călăuzitor de a „dezasambla presupunerile” în componentele de bază, pentru că niște elemente mai simple pot conduce mai ușor la o concluzie dorită ?

7-33 (240)

- Să presupunem că vrem să demonstrăm o „propoziție” θ având deja demonstrată (printre altele) formula $\varphi \vee \psi$
- Deoarece nu știm care dintre φ, ψ este /a fost „adevărată” (se poate să fi fost și ambele), trebuie să „prindem” ambele posibilități prin furnizarea a două demonstrații separate, care apoi să poată fi combinate într-o unică argumentație:

1. Mai întâi, vom presupune că φ este „adevărată”; apoi va trebui să „concepem” o demonstrație pentru θ
2. Același lucru ca la 1., cu ψ în loc de φ

7-34 (241)

3. Având în mod real la dispoziție aceste două demonstrații, este evident că putem deduce θ din $\varphi \lor \psi$, și introduce regula ($\lor$**e**) (așa cum este listată în tabelul general), deoarece analiza noastră este acum exhaustivă

- **Exemplul 8**. Arătați că secvența $p \lor q \vdash q \lor p$ este validă.

1	$p \lor q$	*premiză*
	box1...	
2	p	*presupunere*
3	$q \lor p$	($\lor$**i_2**) 2
	sfbox1..	
	box2...	
4	q	*presupunere*
5	$q \lor p$	($\lor$**i_1**) 4
	sfbox2..	
6	$q \lor p$	($\lor$**e**) 1, 2-3, 4-5

7-35 (242)

- Urmează câteva aspecte care trebuie neapărat reținute atunci când se aplică regula ($\vee$**e**)
- Pentru ca raționamentul să fie într-adevăr „sound”, trebuie să ne asigurăm că concluziile din cele două cazuri distincte (formulele denotate prin θ) coincid cu adevărat
- „Munca” efectuată prin aplicarea lui ($\vee$**e**) reprezintă cu adevărat combinarea argumentațiilor pentru pentru cele două cazuri distincte într-unul singur

7-36 (243)

- În fiecare dintre cele două cazuri distincte știute trebuie să fim atenți ***să nu utilizăm*** și presupunerea temporară folosită în celălalt caz (exceptînd situația în care vreuna dintre presupuneri a fost deja ***demonstrată înainte*** de deschiderea dreptunghiurilor aferente cazurilor amintite)
- Revenind, să notăm și faptul că dacă folosim într-o argumentație o formulă de tipul $\varphi \vee \psi$ *doar* ca o presupunere sau o premiză, se pierde „ceva” din informația avută la dispoziție (nu „știm” **care** dintre φ sau /și ψ este „adevărată”)

7-37 (244)

- Să aprofundăm puțin și regulile pentru *negație*
- „Ciudățenia”, vizibilă (sintactic !) de la bun început, este că avem reguli pentru introducerea /eliminarea dublei negații, nu și pentru negația simplă
- Dacă „gândim semantic” însă, dacă o formulă θ ar însemna „adevăr”, negația ei ar însemna „contradicție”, iar noi suntem preocupați ca ***prin raționament să păstrăm adevărul***
- În consecință nici nu ar trebui să existe vreo modalitate directă de a deduce $\rceil\theta$ „știind” θ

7-38 (245)

- Să reamintim că pentru noi, deocamdată, o contradicție este dată doar prin definiție sintactică: $\neg\theta \wedge \theta$ și $\theta \wedge \neg\theta$ (pentru *orice* formulă θ)
- Odată cu introducerea regulilor pentru negație, ar trebui să fim capabili să demonstrăm validitatea unei secvențe de tipul:

$$\neg(r \vee s \to q) \wedge (r \vee s \to q) \vdash (p \to q) \wedge \neg(p \to q)$$

  și reciproc
- Mai mult, *orice* formulă ar trebui să poată fi „derivată” dintr-o contradicție, nu ?

7-39 (246)

- Ideea este desigur cunoscută: într-o secvență, după „$\vdash$”, poate /trebuie să apară orice afirmație care ar putea fi dedusă (presupunând că premizele dinainte de „$\vdash$” au fost deja deduse); și aceasta indiferent dacă acele premize au vreo legătură (semantică, intuitivă, ...) cu concluzia
- De unde ajungem (mai târziu ... semantic, în **H/R**) și la ideea de reguli /raționament corect /sound: dacă toate premizele sunt „adevărate” atunci și concluzia este adevărată (lucru valabil, prin lipsă, și dacă vreuna dintre premize este – poate, mereu - „falsă”)

7-40 (247)

- În tabelul general al regulilor, aceste ultime observații sunt „prinse” prin introducerea simbolului $\perp$ (pe „post” *și* de $\square$), ca denotație generală pentru o contradicție și, în consecință, introducerea regulilor($\perp$**e**) și ($\neg$**e**)
- **Exemplul 9**. Arătați că secvența $\neg p \vee q \vdash p \rightarrow q$ este validă.

1 $\neg p \vee q$ *premiză*

box1..|box1......................................

2 $\neg p$ *premiză* | q *premiză*

box2..|box2......................................

3 p *presupunere* | p *presupunere*

4 $\perp$ ($\neg$**e**) 3, 2 | q *premiză* (copie a lui 2)

5 q ($\perp$**e**) 4 |**sfbox2**…………………………

5’ **sfbox2..**|$p \rightarrow q$ ($\rightarrow$**i**) 3-4

6 $p \rightarrow q$ ($\rightarrow$**i**) 3-5 |

sfbox1..|..

7 $p \rightarrow q$ ($\vee$**e**) 1, 2-6

7-41 (248)

- În exemplul anterior am „pus alături” (grafic) dreptunghiurile /demonstrațiile care sunt ipoteze pentru regula ($\vee$**e**) (împreună cu premiza $\neg p \vee q$); desigur că (pentru a „respecta tradiția”) le puteam pune, ca și până acum, una sub alta (oricum, ordinea nu contează)
- De asemenea, 5’ este „pe post” de 5 pentru dreptunghiul din dreapta și „intră” în enumerarea notată prin 2-6
- Să subliniem în continuare că „intuiția din spatele” regulii ($\neg$**i**) este: *dacă facem o presupunere care ne „duce” într-o „stare” contradictorie (obținem prin demonstrație $\perp$), înseamnă că de fapt acea presupunere este „nerealistă” (nu poate fi „adevărată”); deci ea trebuie să fie „falsă”*

7-42 (249)

- **Exemplul 10**. Arătați că secvența $p \to q, p \to \neg q \vdash \neg p$ este validă.

1	$p \to q$	*premiză*
2	$p \to \neg q$	*premiză*
	box1..	
3	p	*presupunere*
4	q	($\to$**e**) 1, 3
5	$\neg q$	($\to$**e**) 2, 3
6	$\perp$	($\neg$**e**) 4, 5
	sfbox1....................................	
7	$\neg p$	($\neg$**i**) 3-6

7-43 (250)

- **Exemplul 11**. Arătați că secvența $p \to \neg p \vdash \neg p$ este validă.

1 $p \to \neg p$ *premiză*

box1..

2 p *presupunere*

3 $\neg p$ ($\to$**e**) 1, 2

4 $\perp$ ($\neg$**e**) 2, 3

sfbox1......................................

5 $\neg p$ ($\neg$**i**) 2-4

- Să trecem în revistă și câteva „intuiții” legate de *regulile derivate*

7-44 (251)

- Prezentăm mai întâi demonstrația validității secvenței $\varphi \to \psi \vdash \neg\psi \vdash \neg\varphi$
- De aici va rezulta faptul că regula (**MT**) poate fi văzută ca un „macro” care înlocuiește o „combinație” de aplicări ale regulilor ($\to$**e**), ($\neg$**e**) și ($\neg$**i**)
- **Exemplul 12**. Regula (**MT**).

1	$\varphi \to \psi$	*premiză*
2	$\neg\psi$	*premiză*
	box1.......................................	
3	φ	*presupunere*
4	ψ	($\to$**e**) 1, 3
5	$\perp$	($\neg$**e**) 4, 2
	sfbox1....................................	
6	$\neg\varphi$	($\neg$**i**) 3-5

7-45 (252)

- **Exemplul 13**. Ceva similar se poate arăta pentru regula ($\neg\neg$**i**). Ea poate fi derivată din ($\neg$**i**) și ($\neg$**e**):

1 φ *premiză*

box1..

2 $\neg\varphi$ *presupunere*

3 $\bot$ ($\neg$**e**) 1, 2

sfbox1....................................

4 $\neg\neg\varphi$ ($\neg$**i**) 2-3

- Ideea de a folosi „un număr minim” de reguli (de fapt, pentru demonstrarea unei teoreme de corectitudine și completitudine; niciuna derivată) este uneori benefică

7-46 (253)

- Să trecem la (**PBC**): *dacă pornind cu* $\neg\varphi$ *obținem o contradicție, atunci suntem îndreptățiți să spunem că am demonstrat* φ
- Mai jos, ca o alternativă generală pentru prezentarea „pe verticală” a unei demonstrații de validitate a unei secvențe, în loc de a porni cu premiza „dreptunghi care are *sus* pe $\neg\varphi$ și *jos* $\perp$” vom „scrie” (pentru simplitate) că am făcut demonstrația lui $\neg\varphi \to \perp$ (de fapt, folosim (**→i**) în mod implicit)
- Astfel, avem demonstrația faptului că (**PBC**) este derivată din (**→i**), (**¬i**), (**→e**) și (**¬¬e**):

7-47 (254)

1 $\neg\varphi \to \bot$ *demonstrație făcută deja*

box1..

2 $\neg\varphi$ *presupunere*

3 $\bot$ (**$\to$e**) 1, 2

sfbox1...................................

4 $\neg\neg\varphi$ (**$\neg$i**) 2-3

5 φ (**$\neg\neg$e**) 4

- **Definiție**. Fie $\varphi, \psi \in$ **LP**. Spunem că ele sunt ***demonstrabil echivalente*** (scris $\varphi \dashv\vdash \psi$) ddacă ambele secvențe, $\varphi \vdash \psi$ și $\psi \vdash \varphi$, sunt valide.

7-48 (255)

- Conform ultimei **Observații**, relativă la teoremele formate doar cu ajutorul implicației (și folosind regulile de inferență relative la $\wedge$), acest lucru este identic cu a spune că secvența $\vdash (\varphi \to \psi) \wedge (\psi \to \varphi)$ este validă (sau, desigur, că $\theta = (\varphi \to \psi) \wedge (\psi \to \varphi)$ este teoremă)
- Există evident o legătură clară cu anumite concepte deja introduse (consecință sintactică, legătura, încă imposibilă, cu consecințele semantice etc.) și pe care ar trebui să o „reconsiderați” singuri ...
- Sugerăm, în paralel, și „**CE**” fac regulile ...

7-49 (256)

- În final, să ne reamintim modalitatea prin care am răspuns la întrebarea: ***Cum procedăm în realitate pentru a construi efectiv o demonstrație*** ?
- Adică, să recapitulăm pe scurt metoda sugerată până în prezent, de a „construi” o demonstrație în **SD0** /prin deducție naturală, pornind cu o secvență dată
- Metoda, care, „în cuvinte” *demonstrează validitatea secvenței*, *nu poate fi automatizată în întregime*, dar sunt anumite momente în care dacă se ține cont de anumite aspecte structurale (și având „în spate” o semantică intuitivă), putem restrânge foarte mult aria de selecție a următoarei reguli de infrență care ar trebui aplicată în vederea atingerii scopului final

7-50 (257)

- Nu uităm că la fiecare stadiu /pas al demonstrației se permite introducerea „oricărei” formule ca *presupunere* (desigur, dacă alegem o regulă de inferență care „deschide” un dreptunghi /cutie în ipotezele sale; cutia delimitează de fapt *domeniul de „veridicitate”* al presupunerii făcute)
- Când se introduce o (nouă) presupunere, se deschide o (nouă) cutie; o presupunere nu reprezintă o premiză (inițială) necesară
- Închiderea „fizică” a cutiei se face *numai prin* scrierea unei (noi) linii (deja înafara cutiei), conform „tiparului” concluziei regulii care a permis deschiderea
- Odată cu închiderea cutiei, veridicitatea presupunerii se pierde și trebuie s-o eliminăm din considerațiile ulterioare

7-51 (258)

- Revenind, pornim cu o secvență (a cărei validitate trebuie demonstrată), „scrisă (virtual, poate !) pe verticala unei coli de hârtie”, cu premizele plasate pe rândurile de sus (ordinea nu contează) și concluzia pe ultimul rând
- Rândurile „dintre” premize și concluzie („goale”) trebuie completate cu alte formule (prin aplicarea unor reguli de inferență „potrivite”) pentru a construi demonstrația „completă”
- Trebuie să „lucrăm” *simultan* „de sus în jos și de jos în sus”, adică să încercăm să „aducem” premizele „spre” concluzie, dar și concluzia „spre” premize

7-52 (259)

- Mai întâi este indicat să ne uităm la concluzie
- Dacă aceasta este de forma $\theta = \varphi \to \psi$, atunci „încercăm” aplicarea (unei instanțe a) regulii ($\to$**i**), ceea ce înseamnă de fapt să desenăm un dreptunghi având „sus” pe φ (ca *presupunere*) și „jos” pe ψ
- Mai precis, dacă demonstrația inițială avea forma:

.

premize

.

$\varphi \to \psi$

7-53 (260)

acum va fi:

.

premize

.

box1...............................

φ *presupunere*

ψ

sfbox1...........................

$\varphi \to \psi$ ($\to$**i**)

7-54 (261)

- **Observație**. Ca o excepție (mai degrabă, ca un caz particular) pentru situația tratată mai sus, ar fi util să încercăm întâi aplicarea unei reguli de tipul ($\rightarrow$**e**) (în loc de ($\rightarrow$**i**)), care s-ar putea să conducă mai repede la finalizare (producând o demonstrație mai simplă /scurtă); luați ca exemplu cazul secvenței
  $p \rightarrow (q \rightarrow \neg r), p \vdash q \rightarrow \neg r$
- Revenind acum la aplicarea lui ($\rightarrow$**i**), trebuie în continuare să „acoperim golul" dintre φ și ψ
- Suntem însă într-o situație mai bună: dispunem de încă o formulă asupra căreia putem „lucra", iar concluzia „intermediară" ψ este mai simplă

7-55 (262)

- Să notăm că regula ($\rceil$**i**) seamănă mult cu ($\rightarrow$**i**) și are aceleași efecte benefice asupra așteptărilor de a construi o demonstrație validă: furnizează o nouă „premiză” cu care se poate lucra și „noua” concluzie (intermediară) este mai simplă
- Ideea, în mare, este aceea că, deoarece la fiecare pas al demonstrației există doar câteva reguli de inferență care ar putea fi aplicate să o selectăm pe cea mai „simplificatoare” din lista „completă” (în sensul sugerat mai înainte: situația analizei demonstrației se „îmbunătățește”)

7-56 (263)

- Regulile „aproape” neatinse (ca explicații **CUM /CE**) sunt: ($\rightarrow$**e**), ($\urcorner\urcorner$**e**), ($\urcorner$**e**) și ($\perp$**e**)
- ($\rightarrow$**e**) este de fapt (**MP**) și nu mai comentăm
- ($\urcorner\urcorner$**e**) e comentată puțin pe slide 7-13 (220), iar, în plus, „duala” sa ($\urcorner\urcorner$**i**) este arătată a fi „derivată din ($\urcorner$**i**) și ($\urcorner$**e**)” pe slide 7-45 (252)
- Regulile ($\urcorner$**e**) și ($\perp$**e**) sunt „evidente”: dacă φ **și** negația sa sunt ***true***, atunci ***false*** este ***true***; respectiv, dacă ***false*** este ***true*** atunci **orice** este ***true***
- Ca ultimă concluzie: mai mult decât a spune „*Faceți, punctual, la fiecare pas, alegeri judicioase, bazate atât pe structura premizelor și pe structura concluziei, cât și pe intuiția semantică*”, ***folosiți oridecâteori este posibil regulile*** ($\rightarrow$**i**) ***și*** ($\urcorner$**i**)

7-57 (264 - final 7)

- În acest curs am prezentat detaliat sistemul deductiv numit al „deducției naturale” și notat de noi prin **SD0**, urmând linia generală din **H /R**
- Toate slide-urile sunt importante, existând multe exemple, interpretări intuitive și explicații de natură semantică (atât de natură ***declarativă*** – CE?, cât și ***procedurală*** – CUM?) privind ***forma regulilor de inferență*** și ***modalitățile de a construi demonstrații*** în **SD0** (i.e. ***secvențe valide***)
- Mai întâi, fiți siguri că ați înțeles cele 12 + 4 reguli; apoi, că știți cum trebuie completat (mecanic + creativ) spațiul dintre premizele și concluzia unei secvențe, pentru ca ea să fie validă: folosind reguli pentru ipoteze obținute anterior + aplicând corespunzător „mecanismul” de construire și folosire a „căsuțelor”

8-1 (265)

- Acest „curs” ***suplimentar*** grupează doar câteva dintre conceptele și rezultatele de bază privind ***constructivismul în teoria mulțimilor, numerele cardinale și ordinale, calculabilitatea și decidabilitatea, complexitatea calculului și tratabilitatea problemelor, modelele formale ale noțiunii de algoritm***

8-2 (266) Mulțimi 1

- Nu vom „intra adânc" în ***teoria axiomatică a mulțimilor***, dar vom apela de câteva ori, explicit sau nu, la *axiomatica Zermelo-Fränkel* (**ZF**, pe scurt, fără axioma alegerii; **ZFC**(hoice) este **ZF cu** axioma alegerii): vorbim despre *axioma alegerii*, *axioma înlocuirii*, *axioma separării*, *axioma regularității*, sau *axioma infinitului*
- Necesitatea utilizării unor formalisme pornește de la anumite ***paradoxuri*** „primordiale", începând cu arhicunoscutul paradox al lui B. Russell, din care rezultă faptul că, formal, noțiunea de mulțime nu este suficientă, fiind nevoie de un concept mai elaborat, implicând noțiunea de ***clasă***
- Acest paradox ne „spune", în esență, că nu orice *formulă /proprietate* poate defini /descrie o mulțime, de exemplu proprietatea **P**(x) : „*x este mulțime și $x \notin x$*"
- Dacă ar fi adevărat, ar însemna că există mulțimea R = {x **|** **P**(x)} și s-ar deduce imediat că „R $\in$ R *dacă și numai dacă* R $\notin$ R"

8-3 (267) Mulțimi 2

- Vom „așeza” teoria mulțimilor (chiar la nivel intuitiv) „înaintea matematicii”, adică o vom privi ca un domeniu având origini „precursoare” (tot în) logica filozofică
- În cazul în care obiectele dintr-o colecție sunt ele însele mulțimi, colecția se va numi și ***familie***
- Ceea ce este important pentru noi va fi să admitem existența ***mulțimilor infinite***, simultan cu posibilitatea ca acestea să poată fi ***manipulate algoritmic***
- Considerăm necesar să enunțăm (măcar) axioma alegerii (deși nu o vom folosi efectiv în demonstrații formale), datorită importanței ei în *înțelegerea mai profundă a legăturii dintre numerele cardinale și numerele ordinale*

8-4 (268) Mulțimi 3

Axioma alegerii. Pentru orice familie A de mulțimi nevide și disjuncte două câte două, există mulțimi de ***reprezentanți***.

- Ideea în cele de mai sus este că se poate „alege /selecta” (fără a exista un procedeu de selecție unic și precizat formal) câte un element *a*, numit și *reprezentant al lui A*, din fiecare $A \in \mathrm{A}$, astfel încât toate aceste elemente să fie distincte (colecția lor formează o mulțime inclusă în $\bigcup_{A \in \mathrm{A}} A$)

8-5 (269) Mulțimi 4

- Nu insistăm acum asupra operațiilor pe mulțimi, fie ele definite constructiv sau nu (*intersecție*, *reuniune*, *produs cartezian*, etc.)
- Să precizăm doar că *mulțimea părților* (a tuturor submulțimilor) unei mulțimi A, se va nota cu **P**(A), sau cu 2^A
- O ***relație k-ară*** ($k \in$ **N***), ***peste mulțimile*** M_1, M_2, ..., M_k este orice submulțime R $\subseteq M_1 \times M_2 \times \ldots \times M_k$
- Dacă $k = 2$, R se numește ***binară***, caz în care, dacă avem și $M_1 = M_2 = M$, relația va fi ***pe*** M
- Atunci M_1 se mai numește domeniul lui R (notat **dom**(R)), iar M_2 va fi codomeniul acesteia (notat **ran**(R))
- Dacă R este o relație binară pe M, se va numi R***-lanț finit*** (***infinit***) peste M, o secvență /cuvânt finit (infinit) de elemente $a_i \in M$ ($i \in$ **N'** $\subseteq$ **N**) cu proprietatea că $<a_i, a_{i+1}> \in$ R (se mai scrie a_i R a_{i+1}, sau $a_{i+1} \in$ R(a_i)), pentru fiecare $i \in$ **N'**

8-6 (270) Mulțimi 5

- Dacă **N'** = {0, 1, …, n - 1}, $n \geq 2$, atunci R-lanțul este ***de la*** a_0 (***începe cu*** a_0) ***la*** a_{n-1} (***se termină cu*** a_{n-1})
- O ***funcție /operație /aplicație*** (***de la*** mulțimea A ***la*** mulțimea B) este un triplet $<f, A, B>$, unde $f \subseteq A \times B$ este o relație binară ***surjectivă***
- Dacă în plus este satisfăcută condiția
  $(\forall a \in A)(\exists b \in B)(f(a) = b)$, funcția f se numește ***totală*** (altfel, f este ***parțială***)
- O funcție $<f, A, B>$ se mai denotă prin $f : A \rightarrow B$, iar ***inversa*** *sa* (dacă există), prin f^{-1}
- Nu vom insista nici asupra operațiilor cu relații /funcții (*reuniune*, *produs cartezian*, *star*, *închideri*, *compunere*, etc.)

8-7 (271) Mulțimi 6

- O relație binară R, pe *M*, care este *reflexivă*, *antisimetrică* şi *tranzitivă*, se numește ***relație de echivalență***
- Orice relație de echivalență pe *M* ***partiționează*** mulțimea în ***clase*** (***de ehivalență***), a căror mulțime formează așa-numita ***mulțime cât*** (notată *M*/R)
- Operațiile/***operatorii*** definite/definiți pe *M* se pot *extinde* (în anumite condiții suplimentare, privind ***compatibilitatea***) *la operații pe mulțimea cât* (pe clase, cu ajutorul unor ***reprezentanți*** ai acestora)
- Presupunem cunoscută (adică, nu insistăm), noțiunea de *operație compatibilă* (*la stânga* sau/și *la dreapta*) cu o relație dată
- Exemple importante imediate de relații de echivalență sunt *egalitatea* (identitatea dintre un element și el însuși, relație posibilă a fi definită pe orice mulțime) sau ***echipotența*** (două mulțimi *A*, *B*, sunt echipotente dacă există o funcție totală $f : A \rightarrow B$, care este ***bijectivă***, adică surjectivă și ***injectivă***)

8-8 (272) Algoritmi 1

- O funcție totală $f : A \to B$ poate fi privită (numită) ca (o) ***problemă algoritmică***
- În acest caz, A constituie *mulțimea informațiilor inițiale* ale problemei (***intrărilor***), iar B – *mulțimea informațiilor finale* (***ieșirilor /rezultatelor /răspunsurilor***)
- Dacă B are exact 2 elemente, problema se numește ***problemă de decizie***
- Elementul a aparținând domeniului funcției se mai numește și ***instanță*** *a problemei* (prin abuz de notație, vom scrie și $a \in f$)
- Un ***algoritm*** **(*secvențial*)** ***care rezolvă problema*** f va „porni” cu o ***codificare*** a oricărei instanțe $a \in f$, și va „calcula” o codificare a rezultatului, adică a lui $f(a)$

8-9 (273) Algoritmi 2

- Un algoritm (***pseudocod***, ***program*** într-un limbaj de programare, etc.) va fi privit în sensul paradigmei ***imperative***, conform ideilor lui D. E. Knuth
- Presupunem că fiecărei instanțe $a \in f$ i se poate asocia un număr natural $g_a(f)$, numit ***dimensiunea problemei*** (***pentru*** a)
- Dimensiunea poate fi gândită, de exemplu, ca *lungimea* (ca număr de simboluri) a unei codificări (să zicem, *binare*) pentru instanța considerată
- De asemenea, $g_a(f)$ poate reprezenta uneori o ***dimensiune structurală*** a informației inițiale a, în ideea că lungimea codificării va fi mărginită (superior) de un polinom având ca argument pe $g_a(f)$

8-10 (274) Algoritmi 3

- Lungimea /dimensiunea unui obiect ***o*** se mai notează cu **|*o*|**
- *Resursele de calcul* (principale) *asociate execuției* unui algoritm, sunt legate de ***spațiul*** (de memorie) utilizat în decursul execuției și ***timpul*** necesar finalizării acesteia (ne vom ocupa, puțin mai pe larg, de resursa „timp”; resursa „spațiu” tratându-se similar
- Este bine să subliniem însă faptul că aceste resurse sunt puternic corelate, astfel încât, de obicei, dacă o problemă are timp de execuție „convenabil”, spațiul necesar va fi „mare” , și reciproc

8-11 (275) Algoritmi 4

- Fie astfel o problemă *P* și un algoritm *K* (de la Mohammed ibn-Musa al–Khwarizmi, sec.VIII – IX, a.d.) care rezolvă *P*
- Vom nota cu $T_K(p)$ ***timpul necesar lui*** *K* ***pentru a calcula /rezolva instanța*** $p \in P$; $T_K(p)$ va fi de fapt ***numărul*** operațiilor elementare /***instrucțiunilor*** efectuate de *K* în decursul execuției, pentru găsirea lui $P(p)$
- Presupunem și că resursa „timp” este studiată independent de sistemul de calcul /limbajul pe /în care se face *implementarea* algoritmului
- Aceasta înseamnă că execuția unei instrucțiuni nu depinde în niciun fel de operanzii implicați, sau de timpul efectiv de memorare a rezultatelor
- ***Comportarea*** (în sens temporal, nu uităm) ***în cazul cel mai nefavorabil a lui*** *K* ***pe o intrare de dimensiune*** *n* este dată de $T_K(n) = \sup\{T_K(p) \mid p \in P \text{ și } g_p(P) = n\}$

8-12 (276) Algoritmi 5

- *Analizând* algoritmii într-o asemenea manieră, avem avantajul de a ne asigura de faptul că timpul de lucru este mărginit superior de $T_K(n)$, indiferent de dimensiunea n a intrării
- În practică este posibil însă ca $T_K(n)$ să fie determinat numai de anumite instanțe speciale, care apar foarte rar; de aceea, o alternativă ar fi să apelăm la *teoria probabilităților* (nu insistăm), și anume la studiul ***comportării în medie a unui algoritm***
- Aceasta impune precizarea unei *distribuții de probabilitate* pe mulțimea instanțelor $p \in P$ și determinarea *mediei* pentru $T_K(p)$, privită ca o *variabilă aleatoare, care este*:

$T_{K,med}(n) = \mathbf{media}(\{T_K(p)\ \mathbf{|}\ p \in P \text{ și } g_p(P) = n\})$

8-13 (277) Algoritmi 6

- Calculul mediei de mai sus se reduce de obicei la determinarea valorii unor sume (finite), câteodată existând totuși dificultăți mari de evaluare
- Problema cea mai complicată nu este însă aceasta, ci efectuarea într-un mod „realist”, a etapei precedente, notată cu (a); din acest motiv, ne vom axa doar pe determinarea lui $T_K(n)$ (deși și acest lucru este uneori foarte dificil, nemaivorbind de iminența considerării unor detalii de implementare)
- Suntem nevoiți astfel să căutăm margini superioare (sau chiar inferioare) pentru $T_K(n)$, care sunt mai „accesibile”, și vom studia ***comportarea asimptotică*** a acestuia sau ***ordinul său de creștere*** (adoptăm și anumite notații uzuale pentru clasa funcțiilor (totale) de la **N** la **N**, pe care o notăm pe scurt cu [**N** $\rightarrow$ **N**])

8-14 (278) Algoritmi 7

- Adică, pentru fiecare $f \in [\mathbf{N} \to \mathbf{N}]$, numită în acest context și ***funcție de complexitate***, punem:

  $\boldsymbol{O}(f) = \{g \mid (g : \mathbf{N} \to \mathbf{N})(\exists c \in \mathbf{R}, c > 0)(\exists n_0 \in \mathbf{N})$

  $(g(n) \leq c \bullet f(n)$, pentru fiecare $n \geq n_0)\}$.

  $\boldsymbol{\Omega}(f) = \{g \mid (g : \mathbf{N} \to \mathbf{N})(\exists c \in \mathbf{R}, c > 0)(\exists n_0 \in \mathbf{N})$

  $(g(n) \geq c \bullet f(n)$, pentru fiecare $n \geq n_0)\}$.

  $\boldsymbol{\Theta}(f) = \{g \mid (g : \mathbf{N} \to \mathbf{N})(g \in \boldsymbol{O}(f) \cap \Omega(f)\}$.
- În loc de $g \in \boldsymbol{O}(f)$ (respectiv $\boldsymbol{\Omega}(f)$, $\boldsymbol{\Theta}(f)$), se poate scrie și

  $g = \boldsymbol{O}(f)$ (respectiv $\boldsymbol{\Omega}(f)$, $\boldsymbol{\Theta}(f)$)
- În sfârșit, ***comportarea asimptotică pentru*** $T_K(n)$ definită mai sus ***se va numi complexitatea timp a algoritmului*** *K*
- Revenind, dacă *P* este o problemă algoritmică, atunci ***o margine superioară pentru complexitatea ei*** („de tip” ***O***) ***se poate stabili în practică prin proiectarea și analiza unui algoritm care să o rezolve***

8-15 (279) Algoritmi 8

- De exemplu, vom spune că *P* ***are complexitatea timp O***$(f(n))$ dacă ***există*** un ***algoritm*** *K* care ***rezolvă*** *P* și *K* ***are complexitatea*** $T_K(n) = \boldsymbol{O}(f(n))$
- Analog, *P* ***are complexitatea*** (***timp***) $\boldsymbol{\Omega}(f(n))$ dacă ***orice algoritm*** *K* care ***rezolvă*** *P* ***are complexitatea*** $T_K(n) = \boldsymbol{\Omega}(f(n))$
- Mai mult, vom spune că ***un algoritm*** *K* pentru ***rezolvarea problemei*** *P* este ***optimal*** (relativ la timp) dacă *P* ***are complexitatea*** $\boldsymbol{\Omega}(T_K(n))$
- A dovedi că un algoritm dat este optimal pentru o problemă este o sarcină foarte dificilă, existând destul de puține rezultate generale și realizări practice în acest sens; de aceea ne limităm de obicei la considerarea marginilor superioare (sau inferioare, dar mai rar), adică ne vom raporta la clasa $\boldsymbol{O}(f)$
- Să facem câteva precizări legate de ***nedeterminism***, ***clasele formale de complexitate ale problemelor algoritmice***, ***calculabilitate și decidabilitate pentru probleme /algoritmi***, ***tratabilitatea algoritmilor***

8-16 (280) Algoritmi 9

- Un ***algoritm*** *K* având proprietatea că $T_K(n) = \boldsymbol{O}(f(n))$, unde *f* este un polinom de grad oarecare, se va numi ***polinomial*** (va avea complexitate timp polinomială)
- Variante (depinzând de aspectul funcției *f*): complexitate ***logaritmică*** (nu se ia în considerare „memorarea” intrării), ***liniară*** (e vorba de polinoame de grad 1), ***exponențială*** (clar), etc.
- Exceptând problemele care nu admit deloc rezolvări algoritmice (vezi în continuare), s-ar părea că pentru a rezolva o problemă este suficient să-i atașăm un algoritm corespunzător
- Nu este chiar așa, deoarece complexitatea poate crește atât de rapid (cu dimensiunea intrării) încât timpul destinat rezolvării unei instanțe de dimensiune „mare” poate fi prohibitiv pentru om (indiferent de capacitatea de calcul a unui computer concret)

8-17 (281) Algoritmi 10

- „Forma” funcției *f* contează în mod esențial, deși am putea argumenta că „10^n este mai mic decât $n^{10.000}$ în *destule* cazuri” (acest lucru se întâmplă însă pentru valori *mici* ale lui *n*, mai exact pentru un număr *finit* de numere naturale *n*)
- De aceea se justifică împărțirea clasei problemelor algoritmice nu numai în ***rezolvabile*** (există măcar un algoritm care o rezolvă) și ***nerezolvabile***, ci și a clasei problemelor rezovabile în ***tratabile*** (***tractable***) și ***netratabile*** (***intractable***)
- **Paradigmă**. O problemă pentru care nu se cunosc algoritmi polinomiali (***determiniști*** !) se consideră a fi intratabilă (netratabilă).

8-18 (282) Exemple complexitate 1

- **Exemplul 1**. Să presupunem că orice pas (operație elementară) al oricărui algoritm implementat necesită 10^{-6} secunde, adică

  $\boldsymbol{O}(1) = 10^{-6}$
- Atunci:

-Un algoritm cu funcția de complexitate dată de f(n) = n va „lucra" 0.00002 secunde pentru n = 20 și 0.00004 secunde pentru n = 40

-Un algoritm cu funcția de complexitate dată de f(n) = n^5 va „lucra" 3.2 secunde pentru n = 20 și 1.7 minute pentru n = 40

-Un algoritm cu funcția de complexitate dată de f(n) = 2^n va „lucra" 1.0 secunde pentru n = 20 și 12.7 zile pentru n = 40

-Un algoritm cu funcția de complexitate dată de f(n) = n^n va „lucra" $3 \cdot 10^{10}$ secole pentru n = 20 și ... *cât*, pentru n = 40 ?

- **Exemplul 2**. Fie P problema găsirii (calculării) lui a^m, unde $a \in \mathbf{N}$, $a \geq 2$, este dat; deci P este funcția (notată la fel) având atât domeniul cât și codomeniul egal cu **N** și dată prin $P(m) = a^m$

8-19 (283) Exemple complexitate 2

- Conform celor spuse anterior (ne reamintim de codificarea binară a informației), dimensiunea problemei (care depinde de fiecare instanță *m*, dar și de *a* în acest moment) va fi $g_m(P) = \lceil \log_2 a \rceil + \lceil \log_2 a \rceil$ (e vorba de funcția „parte întreagă superioară”, de la **R** la **N**)
- Ca o observație, deoarece *a* este fixat, pentru *m* suficient de mare valoarea $\lceil \log_2 a \rceil$ practic nu mai contează și dimensiunea poate fi aproximată la $n = \lceil \log_2 m \rceil$
- Fiind vorba de „partea întreagă superioară” am putea „pune aproximativ” și $2^n = 2^{\lceil \log 2m \rceil} = \mathrm{m}$ (vom proceda și în viitor în acest mod)
- Chiar fără o demonstrație formală, putem spune că cel mai simplu (în toate sensurile !), trivial și ***determinist***, algoritm (să-l notăm **A1**) care rezolvă problema este:

```
begin
        alam := 1;
        for i = 1 to m do alam := alam * a
end.
```

8-20 (284) Exemple complexitate 3

- Numărul de operații executat de algoritm pentru instanța *m* este, conform observației anterioare, 2^n, adică $\boldsymbol{O}(m) = \boldsymbol{O}(2^n)$
- Prin urmare, „**cel mai simplu algoritm**” al nostru este **exponențial**
- Intuitiv, algoritmii deterministi satisfac proprietatea că după execuția oricărui pas, pasul care urmează este unic determinat (de rezultatul execuției precedente)
- ***Nedeterminismul*** (definiția se obține desigur negând afirmația precedentă) pare o proprietate lipsită de temei, mai ales d.p.d.v. al practicii (cine își dorește un calculator despre care să nu putem fi siguri ce operație execută la un moment dat ?!)
- Însă, valoarea teoretică a acestui concept este inestimabilă (a se vedea mai jos influiența sa asupra claselor de complexitate)
- În plus, situații nedeterministe chiar apar în practică (să ne gândim doar la ***execuția simultană***, ***concurentă***, a mai multor programe/procese secvențiale, executate pe un același calculator, dar pe procesoare diferite, având „viteze” diferite de efectuare a operațiilor elementare)

8-21 (285) Exemple complexitate 4

- Să considerăm acum un algoritm *recursiv* („echivalent” cu cel anterior, în sensul calculării aceleiași funcții), bazat pe următoarea proprietate (tot trivială) a *funcției exponențiale cu baza a*:

$$a^m = \begin{cases} 1, & \text{dacă } m = 0 \\ (a^2)^{m\,\mathbf{div}\,2}, & \text{dacă } m \text{ este număr par} \\ a \cdot a^{m-1}, & \text{dacă } m \text{ este număr impar} \end{cases}$$

- Algoritmul (**A2**), rezultat prin *derecursivarea* proprietății anterioare, va fi:

begin *alam* := 1;
    while *m* > 0 **do**
        if odd(*m*) **then**
            begin *alam* := *alam* • *a*; *m* := *m* - 1 **end**
        else
            begin *a* := *a* • *a*; *m* := *m***div**2 **end**
end.

- Acum nu ne interesează *limbajul concret de descriere a unui algoritm*, sau demonstrarea formală a faptului că acesta *se termină* și *este corect din punctul de vedere al specificațiilor*
- Presupunem, de asemenea, că intrările și ieșirile sunt gestionate separat

8-22 (286) Exemple complexitate 5

- După cum am precizat, ne ocupăm de cazul cel mai nefavorabil și găsim că $T_{\mathbf{A2}}(m)$ (numărul operațiilor elementare efectuate de **A2** pentru rezolvarea instanței *m*, problema pentru această instanță având dimensiunea structurală convenită deja de n = $\lceil \log_2 m \rceil$) este *de ordinul* $2 \bullet n$
- Aceasta pentru că dacă *m* chiar coincide cu 2^n (altfel spus,

  $m - 1 = 2^n - 1 = 1 + 2^1 + 2^2 + \ldots + 2^{n-1}$, conform dezvoltării unei diferențe a^n - b^n), numărul de împarțiri executate în bucla **while** va fi de „aproape” *n*, iar numărul de operații va fi $\boldsymbol{O}(2 \bullet n)$, deci „cam” $2 \bullet n$ (nu uităm nici de inițializarea lui *alam*, deși nesemnificativă, care reprezintă și ea 1 operație), ceea ce reprezintă $T_{\mathbf{A2}}(n)$
- Prin urmare, problema noastră *P* va avea complexitatea

  $2 \bullet \mathrm{n} = \mathrm{T}_{\mathbf{A2}}(n)) = \mathrm{g}(n)$, iar $\mathrm{g} \in \boldsymbol{O}(f(n))$, unde $f(n) = \mathrm{n}$
- **Aceasta înseamnă că pentru rezolvarea lui *P* am găsit un algoritm de complexitate liniară** (!!), ceea ce este evident o imposibilitate
- Analiza este prin urmare greșită
- **Unde este greșeala** ?

8-23 (287) Exemple complexitate 6

- Ea provine din faptul că am considerat că operațiile aritmetice se fac între **numere de lungime (binară) fixă**
- Dar ordinul de mărime al valorilor variabilelor implicate (*alam* și *a*) va crește odată cu creșterea valorii lui *m* (nu uităm că utilizarea calculatorului și a conceptelor de față sunt necesare doar în cazul valorilor mari)
- Astfel că, în realitate, numărul de operații elementare necesare înmulțirii unui număr întreg având o reprezentare binară de lungime „minimă” *p* (folosind *p* biți) cu altul de lungime *q*, cu algoritmul uzual de înmulțire binară, este $\boldsymbol{O}(p \cdot (p + q))$

8-24 (288) Exemple complexitate 7

- În algoritmul anterior va fi necesară executarea de operații (înmulțiri) pentru aflarea succesivă a valorilor

  $a^2 = a \bullet a$, $a^4 = a^2 \bullet a^2$, ..., $a^{2 \text{ la } n} = a^{2 \text{ la } (n-1)} \bullet a^{2 \text{ la } (n-1)}$, ..., precum și pentru a calcula $a^3 = a \bullet a^2$, $a^7 = a^3 \bullet a^4$, $a^{15} = a^7 \bullet a^8$, ...
- Dacă vom considera drept operație elementară înmulțirea a două numere (binare) de lungime t ($=\lceil \log_2 a \rceil$), atunci în precedentul prim șir de înmulțiri se efectuează întâi 1 înmulțire (de tip $t \bullet t$), ceea ce „ia” timp $\boldsymbol{O}(1)$; apoi o înmulțire (de tip $2t \bullet 2t$), care necesită $4 \bullet \boldsymbol{O}(1)$ operații, ..., o înmulțire „$(2^{n-1} \bullet \mathrm{t}) \bullet (2^{n-1} \bullet \mathrm{t})$” necesitând $2^{2 \bullet (n-1)} \bullet \boldsymbol{O}(1)$ operații ș.a.m.d.
- Avem prin urmare un număr total de $\boldsymbol{O}(2^{2 \bullet n})$ operații (elementare), pentru prima secvență
- Procedăm similar și cu a doua secvență, deducând la final că (și) **A2** are de fapt complexitate exponențială

8-25 (289) Definiții formale ale noțiunii de algoritm 1

- Pentru a putea grupa formal *problemele* (*funcțiile*) *algoritmice* (rezolvabile !) în ***clase de complexitate***, este nevoie de o ***definiție formală a noțiunii de algoritm*** (și ***semialgoritm***)
- Acesta poate fi introdusă cu ajutorul unor concepte ca: ***mulțimi*** și ***funcții recursive*** (calculabile prin algoritmi) ***și recursiv enumerabile*** (semicalculabile prin (semi)algoritmi); ***mașini Türing***; ***mașini cu acces aleator*** (**RAM** – **R***andom* **A***ccess* **M***achines* /**M***emory*), ș.a.
- Fără a insista, să spunem că într-un asemenea cadru se poate defini formal și (***ne***)***determinismul***
- În „mare”, orice „obiect” determinist este și nedeterminist, nu și reciproc
- Vom putea preciza formal și ce înseamnă ***calcul***, ***accesibilitate***, ***nedeterminism angelic*** (de tip „există”) sau ***demonic*** (de tip „orice”), ***terminare/oprire***, ***acceptare***, ***pre-*** și ***post-condiții***, ***invarianți***, ***specificații***, ***corectitudine***, etc.; se poate „fixa”, de asemenea, legătura între aceste concepte sau legătura dintre ele și calculatoarele reale

8-26 (290) Definiții formale ale noțiunii de algoritm 2

- Folosind, în particular, noțiunea de mașină Türing, putem vorbi despre: *bandă de lucru*, *intrare*, *ieșire* (acestea au lungime /dimensiune), *configurație*, *pas* de calcul, *calcul cu succes*, *limbaj acceptat*, *algoritm atașat* (*funcție calculată*), *complexitate* (*timp*) pentru o intrare x (cuvânt peste un alfabet Σ), ***complexitate*** (***timp***) ***pentru o mașină*** **MT**, etc.
- Similar cu ceea ce am descris înainte, vom nota această ultimă complexitate cu $T_{\mathbf{MT}}$ și ea va fi o funcție $T_{\mathbf{MT}} : \mathbf{N} \to \mathbf{N}$, dată prin:
- $T_{\mathbf{MT}}(n) = \sup_{x \in \Sigma^*, \, |x| = n}\{k \mid k$ este lungimea unui calcul de oprire al lui **MT** pe intrarea $x\}$, dacă mașina este deterministă, sau
- $T_{\mathbf{MT}}(n) = \sup_{x \in \mathbf{L}(\mathbf{MT}), \, |x| = n}$ $(\min\{k \mid k$ este lungimea unui calcul de acceptare al lui **MT** pe intrarea $x\})$, dacă mașina este nedeterministă

8-27 (291) Definiții formale ale noțiunii de algoritm 3

- Mai precis, dacă $\sum$ este un *alfabet* (mulțime finită și nevidă) și **MT** este o mașină Türing oarecare, deterministă sau nu (având ca intrări cuvinte peste $\sum$), ***limbajul acceptat*** de **MT** este:
- **L**(**MT**) ={*x* **|** *x* $\in \sum^*$ astfel încât există un calcul de acceptare al lui **MT** pentru intrarea *x*}
- Calculele de acceptare sunt calcule de oprire care satisfac (eventual, în plus) o condiție specifică
- Dacă *h* este o funcție pe $\sum^*$, spunem că *h* ***este calculabilă de mașina Türing deterministă*** **MT** dacă pentru fiecare intrare $x \in \sum^*$ calculul (mașina) se oprește având ieșirea *h*(*x*)
- În cazul *nedeterminist* (care este de tip angelic) putem vorbi de ***calculabilitatea funcțiilor parțiale***
- Dacă $T_{\mathbf{MT}} \in \boldsymbol{O}(f)$ și *f* este un polinom *p*, peste **N** (ceea ce, reamintim, se mai scrie $T_{\mathbf{MT}}(n) = \boldsymbol{O}(p(n))$), se spune că funcția *h* calculată de **MT** este ***polinomial calculabilă*** (în cazul problemelor de decizie cuvintele calculabil/rezolvabil pot fi înlocuite de ***decidabil***

8-28 (292) Definiții formale ale noțiunii de algoritm 4

- Oricum, peste fiecare alfabet dat $\sum$, putem considera două clase importante de limbaje:

P = {L $\subseteq \sum^*$**|** există o **MT** deterministă și un polinom *p* peste **N** astfel încât L = **L**(**MT**) și $T_{\mathbf{MT}}(n) \leq p(n)$, pentru fiecare *n*} și

NP = {L $\subseteq \sum^*$**|** există o **MT** nedeterministă și un polinom *p* peste **N** astfel încât L = **L**(**MT**) și $T_{\mathbf{MT}}(n) \leq p(n)$, pentru fiecare *n*}

- O mașină Türing deterministă este și nedeterministă, prin definiție
- În plus, definiția timpului de lucru, $T_{\mathbf{MT}}(n)$, al unei mașini deterministe este mai restrictiv decât al unei mașini nedeterministe, de unde rezultă **P** $\subseteq$ **NP**
- Încă nu se cunoaște dacă incluziunea precedentă este strictă, problema fiind una deschisă și cu implicații covârșitoare asupra teoriei generale a complexității
- Ceea ce putem remarca acum este faptul că ***orice intrare*** *x* pentru o problemă algoritmică (pentru un algoritm, pentru o mașină Türing, etc.), poate fi presupusă (dacă nu, cazul este neinteresant d.p.d.v. al prelucrărilor electronice !) ca fiind codificată ca un cuvânt dintr-un $\sum^*$ (sau, chiar din **N**, cele două mulțimi având același „număr de elemente”, adică, de fapt, același număr cardinal)

8-29 (293) Definiții formale ale noțiunii de algoritm 5

- Atunci, o problemă de decizie *P* poate fi privită ca *o întrebare cu răspuns binar*, de exemplu $P(x) = 0$ sau $P(x) = 1$
- Cum atât **P** cât și **NP** au fost definite drept clase de *limbaje*, fiecărei asemenea probleme (până la urmă, oricărei probleme algoritmice, deoarece din punctul de vedere al resurselor folosite, nu ne interesează cu exactitate ieșirea $P(x)$, ci doar dacă ea există), i se poate *atașa* limbajul $\mathbf{L} = \{x \mid x \in \sum^*, P(x) = 1\}$
- ***Funcția caracteristică*** a acestui limbaj va fi chiar *P*, iar rezolvarea lui *P* va însemna „același lucru” cu a *testa apartenența* unui element *x* la limbajul **L** (the ***membership problem*** pentru mulțimea **L**)
- Dacă $\mathbf{L} \in \mathbf{P}$ acest lucru va însemna că există un algoritm (privit ca o mașină Türing deterministă; acest lucru nu este esențial, mașina Türing fiind acceptată drept un ***model universal pentru orice calculator***), care este „scurt în ceea ce privește timpul necesar”, și care rezolvă *P*

8-30 (294) Definiții formale ale noțiunii de algoritm 6

- Dacă **L** $\in$ **NP**, algorimul polinomial care există este „nedeterminist”, ceea ce este echivalent cu a spune că „putem rezolva repede/polinomial în **|**x**|** problema *P* ” dacă $P(x) = 1$
- Dar dacă cumva $P(x) = 0$, atunci algoritmul poate să nu se termine (altfel spus, problema *P* descrie, în cazul general, o funcție parțială și atunci avem de-a face cu un ***semialgoritm***)
- Continuând, date două probleme de decizie

  $P_1 : I_1 \rightarrow \{0, 1\}$ și $P_2 : I_2 \rightarrow \{0, 1\}$, vom spune că P_1 ***se reduce polinomial la*** P_2 (notat $P_1 \angle P_2$), dacă există o funcție polinomial calculabilă $\varphi : I_1 \rightarrow I_2$ (nu uităm că atât I_1 cât și I_2 pot fi codificate în **N**, sau într-un același $\sum$*) astfel încât să avem: $P_1(x) = P_2(\varphi(x))$, pentru fiecare $x \in I_1$
- O problemă de decizie *P* ***este* NP*-completă dacă*** $P \in$ **NP** și pentru fiecare $P' \in$ **NP** avem $P \angle P'$

8-31 (295) Definiții formale ale noțiunii de algoritm 7

- Clasa problemelor **NP**-complete este nevidă:
- **Teoremă (S. A. Cook)**. Problema **SAT**, a satisfiabilității formulelor booleene (vezi și capitolele ulterioare), este **NP**-completă.
- Importanța clasei **NP** este evidentă: dacă $P \in$ **NP** și dacă pentru ea am găsi (și) un algoritm polinomial (determinist) care să o rezolve (adică avem și $P \in$ **P**) atunci orice altă problemă *P'* din **NP** se se va putea rezolva (și) în timp polinomial (prin transformarea polinomială – conform definiției - a oricărei instanțe a lui *P'* într-o instanță a lui *P*, care poate fi rezolvată polinomial)
- **Ceea ce ar însemna că P = NP**
- Enumerăm câteva alte concepte importante privind calculabilitatea care ar trebui cunoscute: ***complexitatea spațiu*** (inclusiv clasele **PSPACE**, **NPSPACE**, completitudine și reducere legate de această resursă)
- În privința complexității (ca și schimbarea modului de analiză generală a algoritmilor, vezi *comportarea în medie*), se poate vorbi complet separat despre ***complexitatea algoritmilor paraleli/concurenți/distribuiți***

8-32 (296) Numere cardinale și numere ordinale 1

- Începem cu câteva noțiuni suplimentare legate de mulțimile parțial ordonate
- Fie *M* o mulţime oarecare (nevidă) şi „≤” o relaţie binară pe *M* care este *reflexivă*, *antisimetrică* şi *tranzitivă*
- Cuplul <*M*, ≤ > se numeşte ***mulţime parţial ordonată*** (***poset***), iar „≤” este o ***relaţie de ordine*** (***parţială***)
- Dacă pentru fiecare a, b *M* avem fie a ≤ b fie b ≤ a, atunci <*M*, ≤ > este ***total ordonată*** **(*lanț*)**
- Un lanț *M* care nu conține egalități formează o ***ordine totală strictă/ierarhie***, notată și < *M*, < >
- Mai putem spune că o ordine „≤” este ***strictă*** (notat: „<”) dacă, în caz că a ≤ b atunci a $\neq$ b

8-33 (297) Numere cardinale și numere ordinale 2

- Considerând A $\subseteq$ *M*, vom spune că a $\in$ *M* este ***majorant*** pentru A dacă b ≤ a, pentru fiecare b $\in$ A
- Un element a *M* este ***cel mai mic majorant*** (*cea mai mică margine superioară; l.u.b.;* $\sqcup$) pentru A dacă este majorant pentru A şi pentru orice alt majorant a’ al lui A avem a ≤ a’
- A $\subseteq$ *M* este ***majorată*** (***mărginită superior***) dacă există cel puţin un majorant al ei
- b $\in$ A este ***maximal*** dacă pentru nici un c $\in$ A – {b} nu avem b ≤ c
- Un element b $\in$ A este ***cel mai mare element*** (***al lui*** A) dacă c ≤ b pentru fiecare c $\in$ A \ {b}

8-34 (298) Numere cardinale și numere ordinale 3

- În mod cu totul analog se definesc noţiunile de ***minorant***, ***cel mai mare minorant***, ***mărginire inferioară***, ***element minimal***, ***cel mai mic element***
- Sunt importante relaţiile de ordine *„**bune**”*, adică ***well ordered sets***, mulțimile ***bine-ordonate***
- Mai exact, o ierarhie $\langle M, < \rangle$ este bine-ordonată dacă orice submulţime nevidă a ei are un cel mai mic element (denumit și ***prim element***, și notat uzual, în cazul întregii mulțimi și dacă există, cu $\perp$)
- Alternativ, putem vorbi despre cel mai mare, sau ***ultim*** element (notat și cu $\top$)
- În cazul ierarhiilor, $\langle M, < \rangle$, poate este bine să precizăm că un element $x \in M$ se numește cel mai mic element (al lui M) dacă $(\forall y)(y \in M \wedge y \neq x \to x < y)$
- Similar, pentru fiecare X, X $\in M$, $\sqcap$X va nota ***cea mai mare margine inferioară*** (*g.l.b.*), dacă există

8-35 (299) Numere cardinale și numere ordinale 4

- <*M*, ≤ > este o ***latice*** dacă ⊔X şi ⊓X există pentru fiecare submulţime finită (nevidă) X a lui *M*
- <*M*, ≤ > este o ***latice completă*** dacă l.u.b. şi g.l.b. există pentru fiecare submulţime a lui *M*
- O funcţie între două mulțimi (parțial) ordonate, *f* : *M* → *M'*, este ***continuă*** dacă şi numai dacă este ***monotonă*** („*păstrează” ordinile*) şi „*păstrează” lub-ul oricărui lanţ*

8-36 (300) Numere cardinale și numere ordinale 5

- **Teoremă (de punct fix a lui Knaster-Tarski)**. Dată *M*, o latice completă și *f* o funcţie continuă $f : M \to M$, *f* ***are un cel mai mic punct fix p*** (*f****(p) = p***) care este dat de $\boldsymbol{p} = \sqcup_{n\geq 0} f^{(n}(\perp)$.
- Să revenim la subiectul legat de „mărimea/dimensiunea” mulțimilor, fie acestea finite sau nu
- Nu ne vom „lansa” în considerații filozofice (nici nu vom „intra” în formalizări excesive, gen teoria axiomatică a mulțimilor), dar vom admite că **infinitul este doar „une façon de parler”** (F. Gauss) și vom adopta **principiul constructivist** (enunțat deja) **de a nu lucra cu mulțimi infinite de obiecte „în întregime”, ci doar cu elementele acestora, definite /obținute structural**
- Putem spune că *echipotența* (ca relație de echivalență) este criteriul fundamental prin care sunt comparate *dimensional* (în sensul *numărului de elemente*) mulțimile
- Utilizînd ca „etalon de măsură” numerele naturale, putem clasifica mulțimile în ***finite***, adică echipotente cu numerele naturale (privite ca *submulțimi/fragmente inițiale* ale lui **N**), și ***infinite*** (mulțimi care nu sunt echipotente cu numerele naturale în sine)

8-37 (301) Numere cardinale și numere ordinale 6

- Dacă în privința mulțimilor finite vom putea vorbi chiar de numărul de elemente (dimensiune legată de numerele naturale, ca obiecte în sine), restul mulțimilor (infinite) vor putea fi clasificate exclusiv utilizînd relația de echipotență
- Prin urmare, vor exista mulțimi echipotente cu **N** (numite și ***numărabile***), precum și altele, neechipotente cu **N** (***nenumărabile***)
- Folosind o relație de ordine, mulțimile infinite vor putea fi ierarhizate la rândul lor în clase
- Se știe astfel că pot exista mulțimi „din ce în ce mai mari”, care pot avea (sau nu !) același „număr de elemente” cu cele din care sunt derivate
- Pentru început, completăm o definiție deja cunoscută

8-38 (302) Numere cardinale și numere ordinale 7

- **Definiție (echipotență, număr cardinal)**. Două mulțimi A, B, sunt ***echipotente*** dacă există o funcție (totală) bijectivă, $f : A \to B$ (notațional: $A \sim B$). Se mai spune că A, B au același (număr) ***cardinal*** (sau, aceeași ***putere /cardinalitate***). Se numește ***număr cardinal*** o clasă maximală (în raport cu incluziunea, privită ca relație de ordine) de mulțimi echipotente.
- Orice mulțime $A \sim n$, $n \in \mathbf{N}$, este o ***mulțime finită***
- În acest context, n va denota mulțimea $\{0, 1, \ldots, n - 1\}$ (*segment inițial* al lui N)
- Dacă $n = 0$, va fi vorba despre „clasa mulțimilor vide” (de obicei, toate elementele sale sunt reprezentate printr-un simbol unic, Ø)
- Câteodată, n va denota mulțimea $\{1, 2, \ldots, n\}$, care însă va fi mai precis denotată prin $[n]$ (avem și $[0] \triangleq Ø$)
- *Orice altă mulțime va fi* ***infinită***
- Cardinalul unei mulțimi A se denotă prin $|A|$, sau $card(A)$
- O mulțime echipotentă cu **N** (infinită) se numește ***numărabilă***, iar numărul cardinal corespunzător se notează cu $\aleph_0$ (*alef zero*; alef, ℵ, este prima literă a alfabetului ebraic)

8-39 (303) Numere cardinale și numere ordinale 8

- O mulțime este ***cel mult numărabilă*** dacă este finită sau numărabilă (sau Ø)
- Furnizăm aici câteva rezultate interesante (complicat de demonstrat în contextul teoriei mulțimilor)
- **Teoremă**. Niciun număr natural nu este echipotent cu o submulțime proprie a sa.
- **Teoremă**. Orice submulțime finită și nevidă a mulțimii **N** are un cel mai mare element (și un cel mai mic element).
- **Teoremă**. O mulțime $A \subseteq \mathbf{N}$ este infinită dacă și numai dacă, pentru fiecare $n \in \mathbf{N}$, există $a \in A$, astfel încât $n \leq a$.
- **Teoremă**. Dacă A este numărabilă, atunci 2^A (adică mulțimea părților mulțimii A, care este echipotentă cu mulțimea funcțiilor având domeniul A și codomeniul orice mulțime cu două elemente, fie ea $\{0, 1\}$) nu este numărabilă. Nici $\mathbf{N}^{\mathbf{N}}$ nu este numărabilă (dar este infinită).

8-40 (304) Numere cardinale și numere ordinale 9

- **Teoremă**. Orice submulțime infinită a unei mulțimi numărabile este numărabilă. Mulțimea **N** este total ordonată strict și bine-ordonată (relația de ordine și cea de ordine strictă sunt arhicunoscute).
- **Teoremă**. Dacă A și B sunt cel mult numărabile, atunci $A \cup B$ este cel mult numărabilă. Reuniunea unei familii F (mulțime *indexată* de mulțimi) finite de mulțimi cel mult numărabile este cel mult numărabilă (același lucru se întâmplă cu reuniunea dacă F este numărabilă și mulțimile sunt nevide). Dacă $\{A_i\}_{i \in \boldsymbol{I}}$ este o asemenea familie (iar $\boldsymbol{I} = [n]$), atunci $A_1 \times A_2 \times \ldots \times A_n$ este cel mult numărabilă.
- De asemenea, mulțimea tuturor secvențelor finite (*cuvintelor*) peste o mulțime nevidă, cel mult numărabilă, este numărabilă.

8-41 (305) Numere cardinale și numere ordinale 10

- **Definiție (secvențe monotone)**. Dată o mulțime parțial ordonată cu ordinea strictă „**<**”, $<M, <>$, o submulțime/secvență a sa, $\{a_i\}_{i \in \mathbf{N}} \subseteq M$, se numește ***strict crescătoare*** (respectiv, ***strict descrescătoare***) dacă $a_i < a_{i+1}$ (respectiv $a_i > a_{i+1}$), pentru fiecare $i \in \mathbf{N}$.
- **Teoremă**. O mulțime total ordonată strict este bine-ordonată dacă și numai dacă nu conține secvențe infinite strict descrescătoare.
- Aceste lucruri fiind fixate, se poate trece la definirea (mai mult sau mai puțin axiomatică, mai mult sau mai puțin constructivă) mulțimilor „de numere” **Z**, **Q**, **R**, **C** (nu insistăm)
- **Teoremă**. $\mathbf{R} \sim (0, 1) \sim (0, 1) \times (0, 1) \sim 2^{\mathbf{N}}$. Mai mult, dacă din **R** îndepărtăm o mulțime numărabilă de elemente, atunci mulțimea rămasă rămâne echipotentă cu **R**.

8-42 (306) Numere cardinale și numere ordinale 11

- Prin urmare, există cu siguranță următoarele „numere cardinale” distincte: 0, 1, ..., *n*, ..., $\aleph_o$, $2^{\aleph_o} = \aleph_1$, primele fiind finite
- $\aleph_o$ este cardinalul („infinit” al) mulțimilor numărabile (al lui **N**), iar $\aleph_1$ este *primul* cardinal infinit nenumărabil (și este ***cardinalul lui*** **R**, notat și cu **c** și numit ***continuu*(*m*)**)
- „Procesul” sugerat se poate continua „în mod natural”: luând orice mulțime *A*, de cardinal $\aleph_1$, mulțimea 2^A va avea cardinal $2^{\aleph_1} = \aleph_2$, etc.
- Se obține astfel ***secvența infinită*** (de ***numere cardinale infinite***) $\aleph_0$, $\aleph_1$, $\aleph_2$, ... ($\aleph_0$ fiind cardinalul numărabil, al lui **N** și cel mai mic cardinal infinit), unde $\aleph_{i+1} = 2^{\aleph_i}$, pentru fiecare i $\in$ **N**

8-43 (307) Numere cardinale și numere ordinale 12

- Această secvență „de alef-uri” se poate adăuga „la coada” secvenței numerelor naturale, obținându-se ***secvența totală*** a mulțimii (de fapt, a ***clasei***, în sens axiomatic) numerelor cardinale
- Spunem „secvență”, deoarece ***clasa precedentă poate fi dotată cu o ordine totală strictă*** (notată tot cu „<”), care generalizează/extinde ordinea similară de pe **N**, folosind noțiunea de *funcție injectivă* (alături de cea de funcție bijectivă)
- Să precizăm că afirmația „$\aleph_{i+1} = 2^{\aleph_i}$, pentru fiecare $i \in$ **N**”, datorată lui G. Cantor (de fapt, doar pentru i = 0), este numită ***ipoteza continuului*** (**CH** – **C***ontinuum* **H***ypothesis*) și este *independentă* de axiomele **ZFC**
- Mai mult, dacă se adaugă acestora, se obține (tot) un *sistem deductiv consistent* (axiomatic)
- Dar ... la fel de bine am putea pune $2^{\aleph_0} = \aleph_2$!

8-44 (308) Numere cardinale și numere ordinale 13

- **Definiție (ordine strictă între numere cardinale)**. Date două numere cardinale diferite, $\mathbf{c_1}$ și $\mathbf{c_2}$ (nu există bijecție între niciun element din $\mathbf{c_1}$ și altul din $\mathbf{c_2}$), spunem că $\mathbf{c_1} < \mathbf{c_2}$ dacă luând mulțimile oarecare $A_1 \in \mathbf{c_1}$ și $A_2 \in \mathbf{c_1}$ există o funcție injectivă având domeniul A_1 și codomeniul A_2 (acest lucru se mai scrie $A_1 \prec A_2$).
- **Teoremă**. Avem $0 < 1 < ... < n < ... < \aleph_0 < \aleph_1 < \aleph_2 < ...$ și între oricare două numere cardinale din lanț nu mai există alt număr cardinal (diferit de cele deja menționate).
- Pentru că teorema anterioară (cel puțin) are nevoie de destule alte rezultate (precum și de alte concepte) pentru ca demonstrația ei să poată fi măcar schițată, trebuie menționate câteva lucruri legate de ***numerele ordinale***, prin care se detaliază și amplifică noțiunea de număr cardinal și problematica generală legată de aceasta

8-45 (309) Numere cardinale și numere ordinale 14

- Începem prin a furniza câteva alte teoreme, făcând parte din același context (în plus, privesc chestiuni legate de *combinatorică* sau *probleme de optimizare*)
- Am putea aminti și de *enumerarea lui G. Cantor pentru* **N**, *reprezentarea* p-*adică a numerelor naturale*, *funcții de împerechere* sau *aritmetica lui M. Presburger*)
- Presupunem în plus că sunteți familiarizați cu elementele de bază ale *teoriei grafurilor*

8-46 (310) Numere cardinale și numere ordinale 15

- Acceptăm însă, pentru moment, următoarea definiție a unui arbore: un ***arbore*** este o relație binară ρ care satisface proprietatea că există un „nod” $a_0 \in$ **dom**(ρ) numit rădăcină, astfel încât, pentru fiecare $a \in$ **dom**(ρ) ⋃ **ran**(ρ) există un *unic* ρ-lanț finit de la a_0 la a
- Fie acum ρ un arbore cu rădăcina a_0 (unică); ***secțiunea*** lui este familia de mulțimi $\{A_i \mid i \in \mathbf{N}\}$ definită recursiv/structural prin (A_i se numește ***nivel***):

Baza. $A_0 = \{a_0\}$.

Pas constructiv. $A_{i+1} = \rho(A_i)$ (pentru fiecare $i \in \mathbf{N}$).

- Este ușor de stabilit că dacă există un (cel mai mic) număr natural i astfel încât $A_{i+1} = \emptyset$, atunci avem $A_j \neq \emptyset$ și $A_k = \emptyset$, pentru fiecare $j \leq i$ și $k > i$
- Un arbore ρ este numit finit (respectiv, infinit) dacă relația ρ este finită (respectiv, infinită)
- Dacă ρ(a) (= $\{a' \mid a \, \rho \, a'\}$) este finită (pentru fiecare $a \in$ **dom**(ρ)), atunci arborele ρ se va numi ***finit ramificat***
- Se poate verifica (prin inducție structurală, care aici coincide cu inducția matematică obișnuită) că avem: dacă ρ este finit ramificat, atunci nivelul i formează o mulțime finită, pentru fiecare $i \in \mathbf{N}$

8-47 (311) Numere cardinale și numere ordinale 16

- **Teoremă (lema lui D. König)**. Fie ρ un arbore și $\{A_i \mid i \in \mathbf{N}\}$ secțiunea lui. Dacă ρ este infinit, dar finit ramificat, atunci există un ρ-lanț $\{a_i \mid i \in \mathbf{N}\}$, $a_i \, \rho \, a_{i+1}$, astfel încât $a_i \in A_i$, pentru fiecare $i \in \mathbf{N}$.
- Pentru un ***graf oarecare***, vom accepta că acesta este o mulțime G, de *muchii* (orice muchie este o pereche de *noduri*)
- O muchie se va nota cu $\{a, b\}$ (vom vorbi și despre *muchii orientate*, sau *arce*, care se vor nota cu $\langle a, b \rangle$), *a* și *b* fiind desigur noduri (*adiacente*, *vecine*)
- Mulțimea nodurilor care apar într-un graf G se notează cu **nod**(G)
- Dată o mulțime oarecare *A* (de noduri), ***graful complet peste*** *A*, notat $[A]_2$, este dat de mulțimea (de muchii): $[A]_2 = \{\{a, b\} \mid a, b \in A, a \neq b\}$
- Un graf oarecare G se numește ***complet***, dacă G = [**nod**(G)]
- ***Complementarul*** unui graf G este graful G' = [**nod**(G)] \ G
- Orice submulțime H $\subseteq$ G va constitui un ***subgraf*** al lui G

8-48 (312) Numere cardinale și numere ordinale 17

- **Teoremă (F. P. Ramsey)**. Dacă G este un graf infinit, atunci, sau G sau complementarul său G', va avea un subgraf infinit, care este și complet.
- Revenim la numerele ordinale, mai exact la mulțimile bine-ordonate
- Practic, **fiecare număr natural este format din exact toate numerele definite anterior** (se mai spune că numerele naturale, adică *mulțimile finite/numerele cardinale finite*, *sunt „gradate dimensional”*)
- Astfel, 0 este (reprezentat de) Ø; 1 este {0}; 2 este {0, 1}, etc.; mai mult,

  N = {0, 1, 2, … }
- Ceea ce face ca **N** să nu fie el însuși un număr natural, este faptul (demonstrabil) că nu este adevărat că „orice submulțime nevidă a lui **N** are cel mai mare element” (condiție necesară)
- Cum **N** este, totuși, și el format din „toate numerele naturale anterior construite”, nu ne împiedică nimeni să spunem că **N este un număr**, ***chiar dacă nu unul natural***

8-49 (313) Numere cardinale și numere ordinale 18

- Dacă acceptăm aceast lucru și notăm „numărul descris de **N**” prin **ω**, secvența anterioară de numere (o notăm ($\star$) mai jos) poate fi continuată:

  ($\star$) $0 \triangleq \varnothing$, $1 \triangleq \{0\}$, $2 \triangleq \{0, 1\}$, ..., $\boldsymbol{\omega} \triangleq \{0, 1, 2, \ldots\}$, apoi punem

  $s(\boldsymbol{\omega}) \triangleq \boldsymbol{\omega} + 1 \triangleq \boldsymbol{\omega} \cup \{\boldsymbol{\omega}\}$, $s(s(\boldsymbol{\omega})) \triangleq \boldsymbol{\omega} + 2 \triangleq s(\boldsymbol{\omega}) \cup \{s(\boldsymbol{\omega})\}$, …
- Practic aceste „noi numere”, **ω**, **ω** + 1, **ω** + 2, ... (ca și numerele naturale anterioare) vor fi cazuri particulare de numere ordinale, care vor fi folosite pentru *a grada dimensional mulțimile infinite/numerele cardinale infinite*
- O ***mulțime*** *A* se numește ***tranzitivă*** dacă elementele sale sunt mulțimi și este satisfăcută proprietatea: $(\forall x)(x \in A \to x \subseteq A)$
- Astfel, mulțimea **N** = $\{0, 1, 2, \ldots\} \triangleq \{\varnothing, \{0\}, \{0,1\}, \ldots\}$ este tranzitivă (d.p.d.v. formal, sunt implicate mai multe dintre axiomele **ZFC**)
- Există de fapt o adevărată ***aritmetică pe clasa numerelor***, cu operații gen adunare, înmulțire, etc.; dar și reuniune , produs cartezian, etc.; sau, aflarea supremumului unui șir, etc.

8-50 (314) Numere cardinale și numere ordinale 19

- **Definiție (număr ordinal)**. O mulțime se numește ***număr ordinal*** (sau, pe scurt, ***ordinal***) dacă este tranzitivă și bine-ordonată, relația de ordine fiind relația de apartenență/incluziune.
- **Teoremă**. Dată o mulțime *A*, cu o ordine bună $M = \langle A, < \rangle$, există un unic ordinal α astfel încât *M* și α sunt izomorfe (adică există o aplicație bijectivă, monotonă, de la α la *M*). Acest izomorfism este și el unic.
- Unicul ordinal izomorf cu *M*, a cărui existență este dată de teorema precedentă, se numește ***tipul de ordine al mulțimii*** *M* și se notează cu **||**M**||**
- **Nu va exista** însă o mulțime a tuturor ordinalilor și nici o mulțime a tuturor ordinilor bune izomorfe cu o ordine bună dată (ci doar clase)
- Se observă că toate elementele lui ($\star$) sunt numere ordinale

8-51 (315) Numere cardinale și numere ordinale 20

- La modul general, *teoria cumulată a ordinalilor și cardinalilor* (bazată, de exemplu pe teoria axiomatică **ZFC**, a mulțimilor) este foarte complexă
- Intuitiv, așa cum nu există o mulțime a tuturor mulțimilor (această ipoteză generând paradoxuri), ci doar clasa tuturor mulțimilor (ceea ce implică un studiu axiomatic, formal), nu va exista nici „mulțimea numerelor cardinale” și nici „mulțimea numerelor ordinale”
- Aceste clase se pot însă „organiza” într-un mod similar cu mulțimile de numere amintite, având, de exemplu, o *aritmetică proprie* (utilizând operații analoage cu adunarea sau înmulțirea, limite de secvențe infinite de ordinali, etc.), *relații specifice* (gen ordini), *metode specifice de demonstrație* (cum ar fi *inducția transfinită*, care este o generalizare a inducției matematice și a demonstrațiilor constructive), etc.
- Vom trece în revistă câteva definiții/rezultate importante, doar pentru a „simți” cât de cât domeniul și ***a puncta legătura calitativ*** **(*structură*) – *cantitativ* („*număr de elemente*”)** ***dintre ordinali și cardinali***

8-52 (316) Numere cardinale și numere ordinale 21

- **Definiție (ordine între ordinali)**. Fie ordinalii α și β. spunem că α ***este (strict) mai mic decât*** β, notat $\alpha < \beta$, dacă $\alpha \in \beta$. În mod uzual, definim $\alpha \leq \beta$, $\alpha > \beta$, $\alpha \geq \beta$.
- Relația de ordine strictă „<” pe o mulțime de ordinali este desigur tranzitivă și nereflexivă (conform definițiilor generale)
- **Teoremă**. Pentru orice ordinali α și β, este adevărată exact una dintre relațiile $\alpha < \beta$, $\alpha = \beta$, $\beta < \alpha$.
- **Teoremă**. Dacă α este un ordinal, atunci $s(\alpha)$ ($= \alpha \cup \{\alpha\}$) este ordinal. În plus, $\alpha < s(\alpha)$ și nu există niciun ordinal β astfel încât $\alpha < \beta < s(\alpha)$. Reciproc, dacă $s(\alpha)$ este ordinal, atunci α este ordinal.
- **Definiție (ordinal succesor și ordinal limită)**. Un ordinal α este numit ***ordinal succesor***, dacă există un ordinal β astfel încât $\alpha = s(\beta)$; altfel, α se numește ***ordinal limită***. În plus, un ordinal α este numit ***ordinal finit*** dacă $\alpha = 0$; sau, dacă α este ordinal succesor și orice ordinal β, cu $\beta < \alpha$, este (la rândul lui) sau 0, sau un (alt) ordinal succesor; în rest, α este numit ***ordinal infinit***.

8-53 (317) Numere cardinale și numere ordinale 22

- **Teoremă**. Orice ordinal α se poate scrie sub forma $\alpha = \beta + \gamma$, unde β este un ordinal limită iar γ este un ordinal finit.
- **Teoremă (a inducției transfinite)**. Fie P(*x*) o proprietate privind numerele ordinale *x*, astfel încât:
- **Baza**. P(0) este adevărată.
- **Pas inductiv**.

(i) Pentru fiecare ordinal α, avem adevărată afirmația „P(α) implică P(α + 1)”.

(ii) Pentru fiecare ordinal limită $\alpha \neq 0$, este adevărată afirmația

„Dacă P(β) este adevărată pentru fiecare $\beta < \alpha$, atunci și P(α) este adevărată”.

Atunci P(*x*) este adevărată pentru fiecare număr ordinal *x*.

8-54 (318) Numere cardinale și numere ordinale 23

- **Observație**. Există o ***legătură esențială*** între „structura” și „mărimea” unei mulțimi, adică între numerele cardinale și numerele ordinale
- Astfel, ***cardinalul unei mulțimi*** *A* ***poate fi definit*** (J. Von Neumann, 1928) ***ca fiind cel mai mic ordinal echipotent cu*** *A*
- Mai mult, ***se va numi număr cardinal*** **(*cardinal, pe scurt*)*, orice ordinal care este cardinal al*** **(*măcar*)** ***unei mulțimi***
- Acest lucru poate fi făcut într-un mod formal, de exemplu folosind axiomatica **ZFC**
- În sensul observației anterioare, vom trece în revistă câteva rezultate interesante, care pot fi chiar demonstrate în cadrul amintit
- **Teoremă (principiul bunei ordonări)**. Orice mulțime poate fi bine-ordonată.

8-55 (319) Numere cardinale și numere ordinale 24

- **Teoremă**. Axioma alegerii este echivalentă cu principiul bunei ordonări.
- **Teoremă (legea trihotomiei mulțimilor)**. Pentru oricare două mulțimi A, B, exact una dintre relațiile $A \prec B$, $A \sim B$, $A \succ B$, este satisfăcută.
- **Teoremă (lema lui F. Hartogs)**. Pentru fiecare mulțime A, există un cel mai mic ordinal α care nu este echipotent cu nici o submulțime a mulțimii A (inclusiv ea însăși).
- **Teoremă**. Principiul bunei ordonări este echivalent cu legea trihotomiei mulțimilor.

8-56 (320) Numere cardinale și numere ordinale 25

- **Teoremă**. Pentru fiecare mulțime A, există un *unic* ordinal α care nu este echipotent cu niciun (alt) ordinal β mai mic decât el.
- Astfel, **definiția unui cardinal (concept cantitativ) prin intermediul unui ordinal (concept calitativ, structural), exprimată în observația imediat anterioară**, devine coerentă și acceptabilă
- **Teoremă**. Fie α un ordinal. Atunci, următoarele afirmații sunt echivalente:

1. α este cardinal.
2. α nu este echipotent cu nici un ordinal $\beta < \alpha$.
3. α este cardinalul mulțimii α ($\alpha = |\alpha|$).

- **Teoremă**. Sunt adevărate afirmațiile:

-$|\alpha| \leq \alpha$, pentru fiecare ordinal α.

-$||A|| = |A|$, pentru fiecare mulțime A.

-$|n| = n$, pentru fiecare $n \in \mathbf{N}$, prin urmare orice număr natural este număr cardinal (ceea ce, intuitiv, enunțasem deja).

-ω este număr natural (din nou, afirmație deja enunțată).

8-57 (321) Numere cardinale și numere ordinale 26

- **Observație**. Ordinalii α care nu sunt echipotenți cu ordinali $\beta < \alpha$ se numesc ***ordinali inițiali***.
- **Teorema 5.3.23** afirmă că orice număr cardinal este ordinal inițial și reciproc
- Fără a fi introdus „cardinalul unei mulțimi”, ***putem*** astfel ***defini*** acum ***numerele cardinale ca fiind ordinalii inițiali***
- Atunci, nu numai că se justifică niște notații anterioare (de exemplu, $|n| = n$), dar se pot trage și niște concluzii de genul: $\boldsymbol{\omega}$ este număr cardinal, dar $\boldsymbol{\omega} + 1$ nu este, deoarece $\boldsymbol{\omega} \sim \boldsymbol{\omega} + 1$ și $\boldsymbol{\omega} < \boldsymbol{\omega} + 1$ (se arată că $\boldsymbol{\omega} + n$ nu este nici el număr cardinal, pentru niciun $n \in \mathbf{N}^*$)

8-58 (322) Numere cardinale și numere ordinale 27

- Pentru că numerele cardinale sunt cazuri particulare de numere ordinale, putem considera că relația „<” între ordinali poate fi considerată ca fiind și între cardinali (și aritmetica ordinalilor poate fi particularizată aici, deși se poate dezvolta direct o *aritmetică specifică pentru cardinali*, fie ei finiți sau infiniți; mai mult, așa cum nu există o mulțime a numerelor ordinale, nu va exista o mulțime a numerelor cardinale; ci doar clase)
- Echipotența mulțimilor are drept corespondent egalitatea cardinalilor, iar existența unei injecții între mulțimi va avea drept corespondent inegalitatea strictă
- **Teoremă**. Fie **a** și **b** numerele cardinale asociate mulțimilor A și B. Atunci $A \prec B$ ddacă $\mathbf{a} < \mathbf{b}$.

8-59 (323) Numere cardinale și numere ordinale 28

- Tragem concluzia că orice doi cardinali sunt comparabili (via „<” și derivatele sale) și clasa cardinalilor (**CN**) poate fi „prezentată” printr-o *secvență transfinită*, care continuă secvența finită cunoscută (a numerelor naturale); aceasta similar cu modul în care am procedat cu clasa numerelor ordinale (**ON**)
- Precizăm că **CN** $\subsetneq$ **ON** (lucru care rezultă imediat, deoarece $\boldsymbol{\omega}$ + 1 nu este număr cardinal) și astfel secvența tuturor cardinalilor va fi o subsecvență a tuturor ordinalilor
- Reluând raționamentul, cardinalii sunt ordinali, deci pot fi împărțiți în *cardinali finiți* și *cardinali infiniți* (așa cum am mai precizat, la nivel intuitiv)
- Cei finiți sunt sunt exact ordinalii finiți (adică numerele naturale, în abordarea „firească”) și există chiar o mulțime a acestora (notată $\boldsymbol{\omega}$)
- În plus, $\boldsymbol{\omega}$ este și primul (cel mai mic) cardinal infinit
- Dar, știm deja, nu orice ordinal infinit este (și) cardinal infinit (și nu există o mulțime a tuturor cardinalilor infiniți)

8-60 (324) Numere cardinale și numere ordinale 29

- Repetăm, secvența 0, 1, 2, ... va reprezenta mulțimea numerelor naturale și va coincide cu mulțimea numerelor ordinale finite și cu mulțimea numerelor cardinale finite
- Apoi, secvența $\boldsymbol{\omega}$, $\boldsymbol{\omega} + 1$, ..., $\boldsymbol{\omega} + \boldsymbol{\omega}$, ... este secvența numerelor ordinale infinite, care va include secvența numerelor cardinale infinite
- Pe scurt, o asemenea subsecvență se definește „constructiv, transfinit, pe porțiuni”
- Pentru o „porțiune”, ideea este de a „începe” (**Baza**) cu cardinalul **a** și apoi de a continua (în **Pasul constructiv**) mai întâi cu succesorii lui **a** (adică cu $\mathbf{a}^+$, „pe post” *de ordinal succesor*, aici fiind vorba despre *succesorul imediat* al lui **a**; urmează $(\mathbf{a}^+)^+$, etc.)
- Totul „se închie” prin a considera numărul care este *supremumul* (în sensul lui „<”) numerelor precedente (adică cu $\mathbf{b_a} = \sup\{\mathbf{a}, \mathbf{a}^+, (\mathbf{a}^+)^+, \ldots\}$, acesta fiind „pe post” *de ordinal limită*)
- **Prima porțiune** din secvența transfinită ar fi $\boldsymbol{\omega}$, $\boldsymbol{\omega}^+$, $(\boldsymbol{\omega}^+)^+$, ..., $\mathbf{b}_{\boldsymbol{\omega}}$

  $(= \sup\{\boldsymbol{\omega}, \boldsymbol{\omega}^+, (\boldsymbol{\omega}^+)^+, \ldots\}$; se va continua cu porțiunea (sub-subsecvența) $\mathbf{b}_{\boldsymbol{\omega}}$ (desigur că în secvența totală nu-l vom lua de două ori), $\mathbf{b}_{\boldsymbol{\omega}}^+$, $(\mathbf{b}_{\boldsymbol{\omega}}^+)^+$, ..., $\sup\{\mathbf{b}_{\boldsymbol{\omega}}, \mathbf{b}_{\boldsymbol{\omega}}^+, (\mathbf{b}_{\boldsymbol{\omega}}^+)^+, \ldots\} = \mathbf{b_c}$, unde $\mathbf{c} = \mathbf{b}_{\boldsymbol{\omega}}$

8-61 (325) Numere cardinale și numere ordinale 30

- Chiar și înainte de prima porțiune transfinită se procedează, practic, în mod similar: primii cardinali sunt 0, 1, 2, ... (finiți) și sup{**0**, 1, 2, …} = $\boldsymbol{\omega}$, care poate fi notat cu $\mathbf{b_0}$, etc.
- Singurele lucruri de „făcut” sunt atunci acelea de a stabili **care este succesorul imediat al unui cardinal infinit, a** (adică $\mathbf{a}^+$) și **care este supremumul** (gen $\mathbf{b_a}$, al) **unei mulțimi** (de fapt, **secvențe crescătoare**) **de cardinali infiniți**
- Fiecare dintre aceste (noi, să zicem) numere cardinale, făcând parte din porțiuni (sub-subsecvențe), vor fi denotate (după cum am sugerat înainte) cu ajutorul literei $\aleph$
- Mai precis, vom apela întâi la lema lui Hartogs de mai sus pentru:
- **Definiție (numere Hartogs)**. Fie *A* o mulțime. ***Numărul Hartogs*** asociat mulțimii *A*, notat *h*(*A*), este cel mai mic ordinal α care nu este echipotent cu nici o submulțime a lui *A*.

8-62 (326) Numere cardinale și numere ordinale 30

- Conform definiției, pentru orice mulțime *A*, avem desigur **|***A***|** < **|***h*(*A*)**|**
- **Teoremă**. Pentru orice mulțime *A*, *h*(*A*) este cardinal.
- **Teoremă**. Pentru fiecare cardinal **a**, *h*(**a**) este cel mai mic cardinal mai mare decât **a**, și el este succesorul imediat al lui **a** (notat $\mathbf{a}^+$).
- Mai mult, se știe că supremumul oricărei mulțimi de ordinali *X* este un ordinal ce depășește strict orice ordinal din *X* (în ipoteza că *X* nu conține un cel mai mare ordinal)
- Un rezultat similar are loc și în cazul în care *X* este o mulțime de cardinali

8-63 (327) Numere cardinale și numere ordinale 31

- **Teoremă**. Fie *X* o mulțime nevidă de cardinali. Atunci:

-$\bigcap X$ este cardinal, fiind și cel mai mic cardinal din mulțimea *X*.

-$\bigcup X$ este cardinal și avem: Dacă *X* admite un cel mai mare element, **a**, atunci $\bigcup X$ = **a**; în caz contrar, **a** < $\bigcup X$ (pentru fiecare **a** $\in X$).

-$h(\bigcup X) \notin X$.

- La fel ca în cazul general (și la fel ca în cazul ordinalilor), vom pune $\bigcap X \triangleq \inf(X)$ și $\bigcup X \triangleq \sup(X)$
- Ei vor fi, desigur, infimumul, respectiv supremumul mulțimii de cardinali *X*
- Utilizarea „alefilor” ($\aleph$) se face practic cu ajutorul unei *funcții*, notată cu același simbol $\aleph$ (și având domeniul **ON** și codomeniul **CN** – nu uităm că acestea *nu sunt mulțimi*, ci *clase*)

8-64 (328) Numere cardinale și numere ordinale 32

- $\aleph$ se definește structural, transfinit, prin (notăm $\aleph_\alpha$ în loc de $\aleph(\alpha)$):

Baza. $\aleph_0 = \boldsymbol{\omega}$.

Pas constructiv.

- $\aleph_{\alpha+1} = h(\aleph_\alpha)$, pentru fiecare ordinal α.
- $\aleph_\alpha = \sup\{\aleph_\beta \mid \beta < \alpha\}$, pentru fiecare ordinal limită α, diferit de 0.
- Prin urmare, valorile succesive (de mai sus) ale funcției $\aleph$ ne va furniza o secvență (crescătoare) transfinită, indexată după clasa ordinalilor
- Cum primul element al secvenței ($\boldsymbol{\omega}$) este un cardinal infinit, se deduce imediat (prin inducție transfinită și folosind rezultatele anterioare) că pentru fiecare ordinal α, $\aleph_\alpha$ este un cardinal infinit și în plus $\aleph_{\alpha+1}$ este succesorul imediat al lui $\aleph_\alpha$

8-65 (329) Numere cardinale și numere ordinale 33

- **Teoremă**. Avem:

-Un cardinal **a** este infinit ddacă există un ordinal α astfel încât **a** = $\aleph_\alpha$.

-Pentru oricare doi ordinali α și β, avem $\alpha < \beta$ ddacă $\aleph_\alpha < \aleph_\beta$.

-Pentru orice ordinal α, $\aleph_\alpha$ este (și) ordinal limită.

- În urma rezultatelor anterioare putem lua în considerare

  așa-numita ***ipoteză generalizată a continuului***

  (**GCH** – **G**_eneralized_ **C**_ontinuum_ **H**_ypothesis_; F. Hausdorff), care poate fi exprimată pe scurt prin: $(\forall\alpha\in$ **ON**$)(2^{\aleph_\alpha} = \aleph_{\alpha+1})$,

  sau: $2^{\mathbf{a}} = \mathbf{a}^+$, pentru fiecare cardinal infinit **a**

- Ca și **CH**, formula anterioară este _naturală_, adică este consistentă cu celelalte axiome ale sistemului **ZFC** și poate fi adăugată ca axiomă suplimentară (nefiind consecință din celelalte)
- Să remarcăm și faptul (legat de notații) că simbolurile **ω** și $\aleph$ sunt folosite exclusiv pentru a „discuta” despre ordinalitate, respectiv cardinalitate