SUPORT/ suplimente pentru Cursul 1 (37 sl)

- Vom <u>începe</u> cu:

Alte exemple de probleme (semi)rezolvate

(I)<u>Suma a trei numere întregi/ naturale consecutive și impare</u> =

Suma a trei numere întregi impare este 57. Găsiți întregii.

- Această problemă este *similară* cu una anterioară (*suma a două* ...; *diferențe*: sunt implicate 3 numere în loc de două, iar numerele nu sunt chiar oarecare, ci impare

2

- Să reprezentăm/ numim primul întreg impar implicat prin *x*; atunci următorul întreg (consecutiv dar impar) va fi (reprezentat prin) $x + 2$, iar cel de-al 3-lea (evident) prin $x + 4$ ($= (x + 2) + 2$)
- Relația dintre datele/ informațiile/ cantitățile cunoscute și/ sau necunoscute este:

$$x + (x + 2) + (x + 4) = 57$$

- Voi: *Finalizați rezolvarea și verificați (corectitudinea) rezultatul(ui)*

(II)Împărțirea comorii =

După ce au cules din pădure 770 de castane, 3 fetițe au decis să le împartă între ele astfel încât *cantitățile care revin fiecăreia să fie direct proporționale cu vârstele lor. Astfel, de fiecare dată când Mary își însușea 4 castane, Nellie lua 3. Apoi, de fiecare dată când Mary primea 6 castane, Susie lua 7. Câte castane a avut în final fiecare fetiță, după împărțirea descrisă ?*

<u>Rezolvare</u>:

- Reformulează/ prezintă problema (eventual)
- Se observă imediat că Mary este „legată” atât de Nellie cât și de Susie; dacă notăm cu c numărul de castane (total) obținut (la finalul împărțelii) de către Mary, atunci Nellie va primi $3 \times c/4$ castane, iar Susie - $7 \times c/6$

- Se știe că fetițele au culese în total 770 de castane, astfel încât avem ecuația:

  $c + 3 \times c/4 + 7 \times c/6 = 770$

- Voi:*Terminați rezolvarea și verificați corectitudinea ei*
- Continuăm cu:

Alte exemple de probleme (nerezolvate)

(III)Mere și pere = *Tom are în hambarul său trei cutii/ coșuri (închise ... „poză” ...) cu fructe: o cutie cu mere, o cutie cu pere și o cutie conținând atât mere cât și pere. Deși fiecare cutie are o etichetă pe care este înscris conținutul, se cunoaște faptul că niciuna dintre etichete nu este atașată cutiei* *corespunzătoare. Să presupunem că lui Tom i se dă doar un singur fruct, dintr-o cutie pe care, de asemenea, o cunoaște. Cum poate Tom să determine conținutul exact al fiecărei cutii ?*

(IV)<u>Un mincinos ciudat</u> =

Richard este un mincinos ciudat. Mai exact, el minte (total) în 6 dintre cele 7 zile ale unei săptămâni, dar în cea de-a 7-a zi el spune mereu adevărul. De curând, el a făcut următoarele afirmații, în trei zile consecutive:

<u>Ziua 1</u>: „Eu mint lunea și marțea”.

<u>Ziua 2</u>: „Astăzi este joi, sâmbătă sau duminică”.

<u>Ziua 3</u>: „Eu mint miercurea și vinerea”.

În ce zi a săptămânii spune Richard adevărul ?

(V)<u>O masă rotundă</u> (familii tradiționale) =

Într-o zi, Helen și soțul ei decid să-și invite vecinii, adică două familii (fiecare formând un cuplu), la masa de seară. Cei șase s-au așezat la o masă rotundă. După o masă foarte reușită, Helen a mărturisit următoarele:

-Victor a stat în stânga femeii care a stat în stânga bărbatului care a stat în stînga lui Anna;

-Esther a stat în stânga bărbatului care a stat în stânga femeii care a stat în stânga bărbatului care a stat în stânga femeii care a stat în stânga bărbatului meu;

-Jim a stat în stânga femeii care a stat în stânga lui Roger;

-Eu n-am stat lângă bărbatul meu.

Care este numele bărbatului lui Helen ?

(VI)<u>Fermierul, gâsca, grâul și vulpea</u> =

Gândiți-vă că sunteți un fermier/ țăran și că toate *bunurile voastre, în urma unei inundații sunt constituite doar dintr-o gâscă, o vulpe și o anumită cantitate de cereale/ grâu într-o sacoșă. Cineva v-a împrumutat o barcă, pentru a trece bunurile peste apă. Din păcate, nu puteți duce în barcă mai multe bunuri simultan, ci doar unul câte unul la fircare transport. Pe oricare mal, nu puteți însă lăsa gâsca împreună cu grâul, nici vulpea cu gâsca (pentru că se …). Cum puteți trece în siguranță toate bunurile dvs. pe partea cealaltă a apei ?*

(VII)<u>Misionarii și canibalii</u> =

Pe malul unui râu se află trei (călugări) misionari și trei canibali, care trebuie să traverseze pe celălalt mal cu ajutorul unei bărci. În barcă încap maxim două persoane la un drum. Dacă, la un moment dat, pe vreun mal, există/ rămân mai mulți (strict) canibali decât misionari, aceștia din urmă vor fi mâncați. Cum se poate face astfel încât toate cele 6 persoane să ajungă pe celălalt mal, rămânând tot timpul în siguranță ?

(VIII)<u>Jane, Jean și Joan</u> =

Joan și Jane sunt surori. Jean este fiica lui Joan și este cu 12 ani mai tânără decât mătușa sa. Joan este de două ori mai în vârstă decât Jean. Acum 4 *ani, Joan avea aceeași vârstă pe care o are acum Jane, iar Jane era de două ori mai în vârstă decâ nepoata sa. Câți ani are Jean ?*

(IX)<u>**O gospodărie în armonie**</u> =

Cineva care trăiește în casă a furat de la mătușa Agatha. În casă trăiesc Agatha, majordomul ei și James; nimeni altcineva. Hoțul își antipatizează victima și, în plus, nu a fost și nu va fi niciodată mai bogat decât aceasta. James nu antipatizează pe nimeni dintre cei pe care îi antipatizează Agatha. Agatha antipatizează toate persoanele, exceptându-l pe majordom. Majordomul, la rândul său, antipatizează pe oricine nu este mai bogat decât Agatha, precum și toate persoanele pe care aceasta le antipatizează. Nicio persoană din casă nu *antipatizează pe toată lumea. Agatha nu este majordomul. De aici, rezultă că Agatha a furat de la ea însăși ? Și, de asemenea, că nici James, nici majordomul, nu au furat de la Agatha ?*

(X)<u>O problemă de algebră abstractă</u> =

Fie G = $\langle$A, $\circ\rangle$ *o mulțime nevidă* (A)*, având cel puțin* 3 *elemente distincte* (*a*, *b* și *c*)*, dotată cu o operație asociativă* ($\circ$) *și având un element neutru* (*notat e*)*.* Înafară de legile știute*, adică:*

i)<u>Asociativitatea</u>: $x \circ (y \circ z) = (x \circ y) \circ z$*, pentru fiecare* x, y, $z \in$ A *și* ii)<u>Elementul neutru (la stânga și la dreapta)</u>:

$x \circ e = e \circ x = x$, pentru fiecare $x \in$ A,

mai sunt adevărate

iii)*$x \circ x = e$, pentru fiecare $x \in$ A, și*

iv)$a \circ b = c$.

Arătați că $b \circ a = c$.

- Presupunem acum că ne-am familiarizat cu noțiunile de problemă, rezolvare, algoritm, limbaj așa cum au fost introduse și conceptualizate până în prezent
- <u>Continuăm</u> cu „complemente” pentru definițiile structurale („gen” **N1** și **LP**)

- Astfel, putem defini constructiv (construi algoritmic) orice mulțime **M**, *numărabilă*
- 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 („grupate” în mulțimea **M’**)
- În **Pasul constructiv**, se repetă (*de câte ori este posibil*) anumite procedee de creare de elemente *noi* în **M**, folosindu-ne de elementele *vechi* (deja existente)
- **M’** (nevidă) este cunoscută/ construită dinainte, ca de altfel și mulțimea **O**, de „algoritmi”/ procedee
- Fiecare *operator* ***o*** $\in$ **O** este privit în sens *determinist*, *funcțional*: aplicat „intrării” $\langle m_1, m_2, \ldots, m_k \rangle$, va furniza (unica) „ieșire” *m*

Definiția structurală a unei mulțimi 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

$m_1, m_2, \ldots, m_k \in$ **M** și pentru fiecare $\boldsymbol{o} \in$ **O** (*operator de aritate k*), avem $\boldsymbol{o}(\langle m_1, m_2, \ldots, m_k\rangle) = m \in$ **M**.

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

- <u>Traducerea algoritmică imperativă</u>; [*n*] este notația *ordinală* a mulțimii {1, 2, ..., *n*}; *pairing functions*; în **N1**, *n* denotă **s(s(**...**(s(**0**))**...**)**

- Fie acum **M** orice mulțime definită structural ca mai sus (cu ajutorul lui **M’** și **O**) și o afirmație generală de tipul **Q** = $(\forall m)\mathbf{(P}(m)\mathbf{)}$, adică proprietatea **P** „privește” întreaga mulțime **M**
- Fie și afirmația **Q’**, *corespunzătoare definiției structurale a lui* **M**, dată prin **Q’** =

$(\forall a \in \mathbf{M'})\mathbf{(P}(a)\mathbf{)} \wedge$

$(\forall k \in \mathbf{N^*})(\forall m_1, m_2, \ldots, m_k \in \mathbf{M})(\forall \boldsymbol{o} \in \mathbf{O})$

$\mathbf{(P}(m_1) \wedge \mathbf{P}(m_2) \wedge \ldots \wedge \mathbf{P}(m_k) \rightarrow \mathbf{P}(m)\mathbf{)}$,

unde $m = \boldsymbol{o}(\langle m_1, m_2, \ldots, m_k \rangle)$

Principiul general al inducţiei structurale

- **Q** este adevărată ddacă putem arăta **Q**', adică:

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

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

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

Ideea - similară *cu inducția matematică*: *în loc să arătăm* **Q**, *vom arăta* **Q'** (*principiu* vs *teoremă*)

- <u>În continuare, revenim la conceptul de logică/ logici</u>
- Ele pot fi văzute și ca ***limbaje de programare***, și ca limbaje „naturale, dar exacte”, adică având o ***sintaxă*** (și o ***semantică***) ***formală***
- În acest fel, (o) ***formulă*** = (un) ***program***
- Semantică generală se bazează pe ideea de ***valoare de adevăr***
- Semnificația/ ***semantica*** unui program este dată (în sens *imperativ/ operațional*) de **execuțiile sale**: pentru intrarea *x*, se obține ieșirea *y*, prin efectuarea operațiilor indicate în textul programului (în ordinea precizată) asupra valorilor introduse din exterior sau obținute pe parcurs
- Semnificația unei formule va fi dată, similar, de valorile de adevăr (finale), obținute în urma *aplicării* operatorilor (logici) din formulă (ordine fixată) valorilor de adevăr ; <u>detalii</u> ...

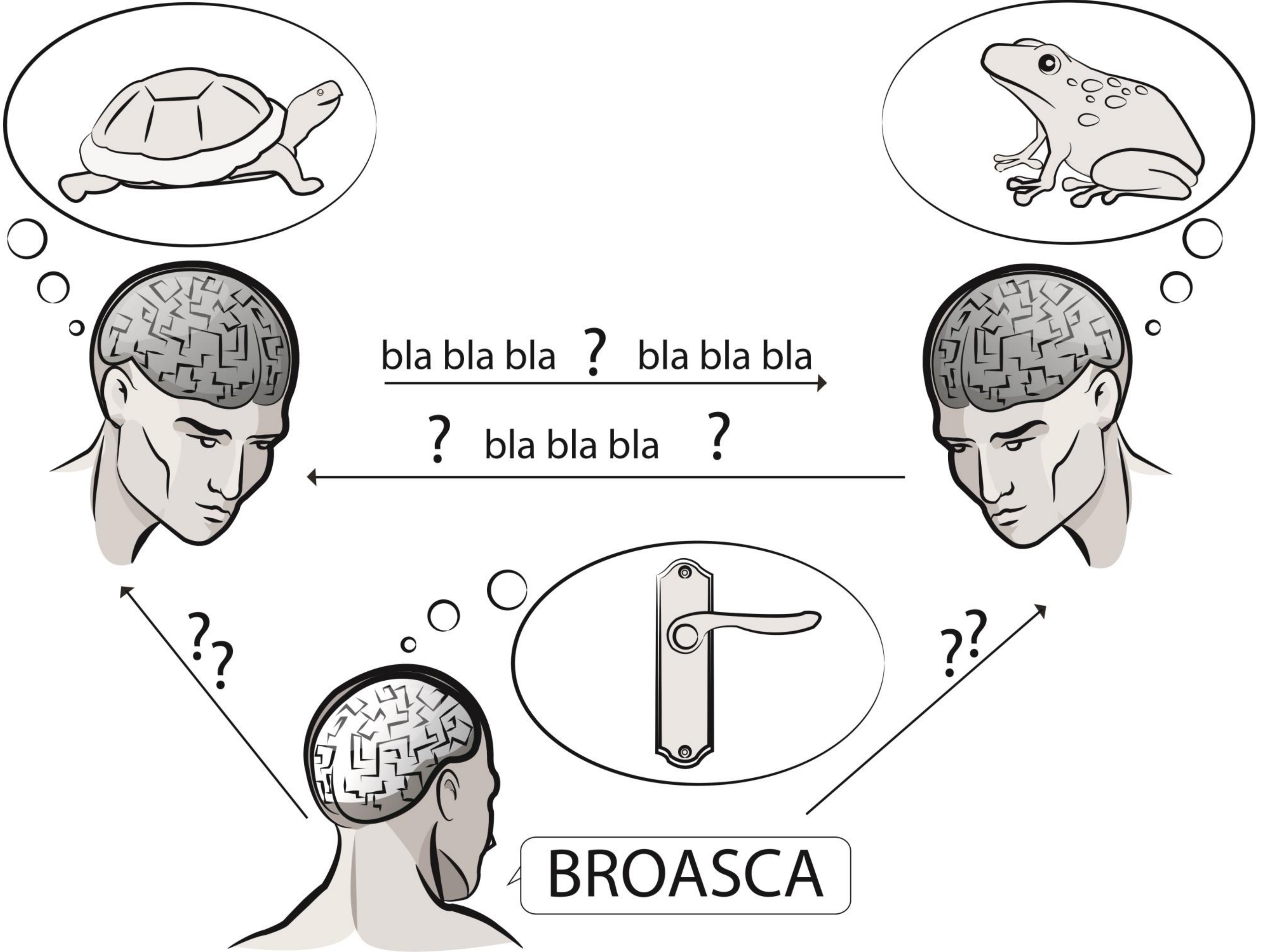
Logica este o știință în sine:

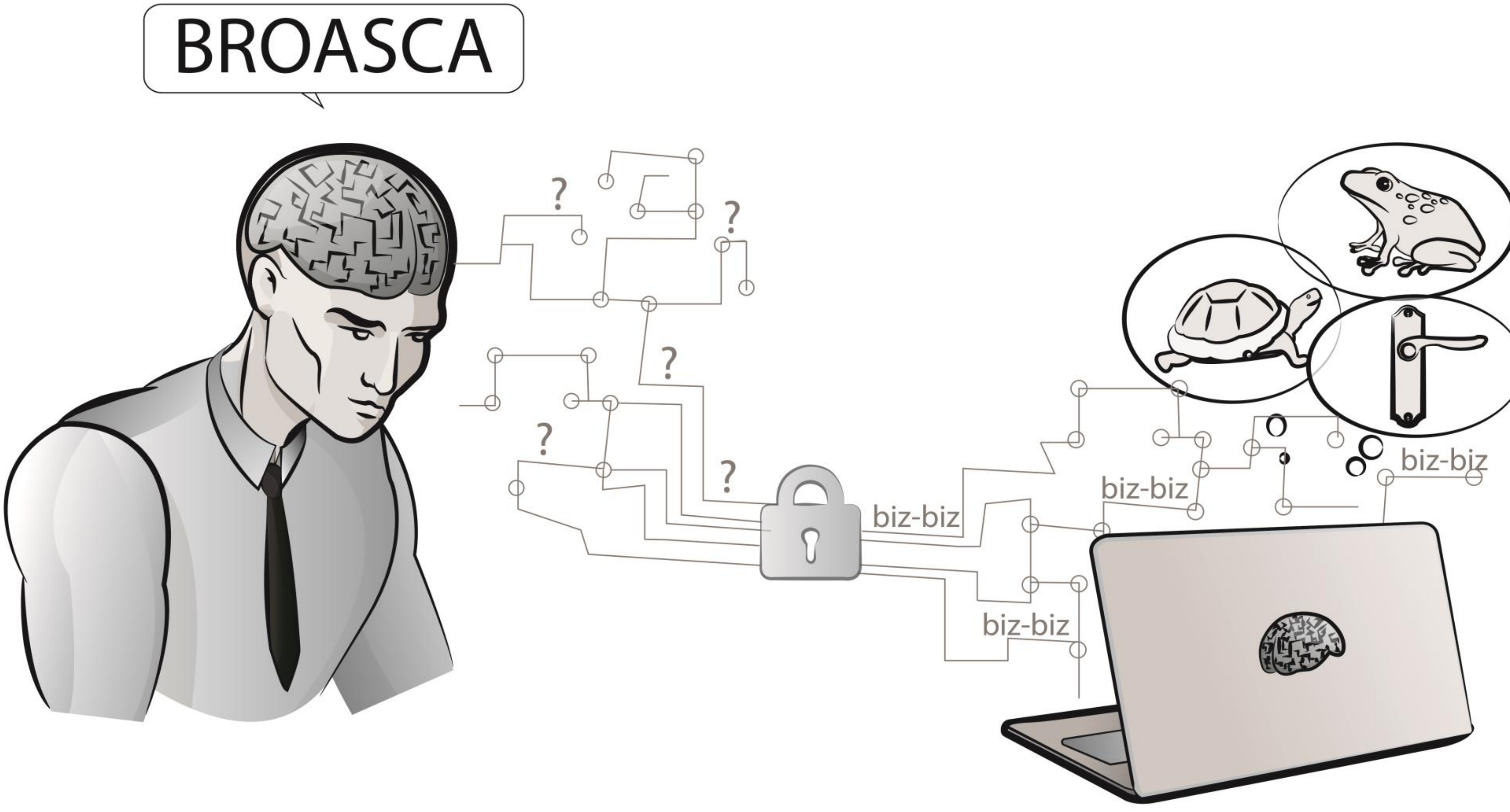
- *Știinţa regulilor generale ale gândirii, cu accent pe aspectele exacte şi de natură structurală ale acesteia* sau, ***Știință a demonstrației, al cărei obiect este stabilirea condițiilor corectitudinii gândirii, a formelor și a legilor generale ale raționării corecte*** (**DEX**online)
- **Realitatea/ 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 (în creier) prin <u>afirmaţii</u> (*judecăţi*), acestea sunt reformulate în limbaj <u>natural/ de discurs</u> (română, engleză ...)

- 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*** (sau ***atom***; nu mai poate fi „descompusă” în alte afirmații) sau ***compusă***
- Însă „adevărul” (care, în general, 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***

20

- Sintaxa (**DEX**online): *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/ înțelese* pentru a putea fi folosite în comunicare ...


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


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

- Semantica (**DEX**online): *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 (intonație/ accent !):

Textul 1 (Pitia)

- *Nu vei muri* (*...*)

Textul 2

- *Țara mea este mama mea* (*...*)

- 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)
- Să nu uităm nici de definiția 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 (*analfabeți funcțional*)
- 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*

Textul 3

- 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*
- 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 folosite mai sus sunt de bun simț și cunoscute (?!) din raționamentele matematice uzuale (+ „Logica” de liceu ... mai jos ...)

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

- *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* : *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 (onomatopeele ...); poate fi *confuz*; *paradoxuri* ...

- <u>Robert Swartz</u> (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***
- Ar trebui să stăpâniți deja noțiunile („Logica de liceu”): *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*)

Logica – parte a matematicii

- *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*
- 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.; există logici ***extensionale*** și *intensionale ...*

Logica – parte a informaticii

- „Proiectarea” logicii matematice în **Informatică** (cea mai nouă, 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***
- Concepte mai profunde: *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* (vezi Cursul 8, suplimentar ...)

- Câteva argumente în favoarea 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*, *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., dar ...
- Acestea din urmă nu sunt, în multe cazuri, suficiente (*dacă* sunt posibile); ba sunt chiar nesigure uneori, având nevoie la rândul lor de o verificare formală prealabilă

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

- 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 $10^{(10 \text{ la puterea } 90)}$ 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 ...

- Încheiem aceste suplimente cu câteva nume de referință pentru domeniul nostru (să-l numim ***Logică și demonstrare automată***), precum și alte referințe bibliografice, incluzând site-uri web

Persoane importante și contribuții

-**Alan Robinson**: „părintele” *rezoluției* (1961)

-**Gérard Huét**: „părintele” *sistemelor de rescriere* (1976)

-**Alan Bundy**: expert *problem solving* (carte: CLAM Theorem Prover)

-**Hillary Putnam**: carte, Davis Putnam Prover (1957)

-**Larry Wos**: Otter Prover (1980); un limbaj succesor este Prover 9

-**Robert Kowalski**: introduce conceptul de *connection graph* (1976); carte: Theory of Logic Programmimng (1979)

-**Donald Knuth**: expert în *teoria algoritmilor*; de menționat *algoritmul Knuth-Bendix* (1981); creatorul *editorului de texte* Teχ (succesor - LaTeχ)

-**Martin Davis**: *Davis-Putnam Algorithm* (1957)

-**Roger S. Boyer, Jay S. Moore**: Boyer-Moore Prover; introduc conceptul de *structure sharing in theorem-proving programs*

-**Donald Loveland**: introduce conceptul de *model elimination* (1957)

-**Norbert Eisinger**: dezvoltatorul *connection graph theory* (1986)

-**Alain Colmérauer**: primul *sistem* PROLOG funcțional

-**David Plaisted**: introduce conceptul de *hyper-linking*

-**Reinhold Letz**: Setheo, free variable tableau proving (1990)

-**Mark Stickel**: PTTP theorem prover (1982)

-**Woody Bledsoe**: UT prover (bazat pe deducția naturală)

-**Ian Horrocks**: *description logic theorem prover*

-**Larry Paulson**: creatorul **ATP**-ului general **Isabelle**

-**Andrei Voronkov**: Vampire theorem prover

Referințe suplimentare și website-uri

- **Alan Bundy** – *The Computer Modelling of Mathematical Reasoning* (academic press)
- **Chin-Liang Chang, Richard Char-Tung Lee** – *Symbolic Logic and Mechanical Theorem Proving* (academic press)
- **Mel Fitting** – *First Order Logic and Automated Theorem Proving* (Springer Verlag, Ed.)
- **Alan Robinson** – *Logic: Form and Function* (Edinburgh Press)
- **Larry Wos** – *Automated reasoning: Introduction and Applications* (McGrawHill Press)
- **John-Arnold Kalman** – *Automated Reasoning with Otter* (Rinton Press)

- **Karl-Hans Blasius, Hans-Jürgen Burkert** – *Deduction Systems in Artificial Intelligence* (Ellis Horwood Press)
- **Jean-Louis Lassez, Gordon Plotkin** (eds.) – *Computational Logic* (M.I.T. Press)
- **Antonis-C. Kakas, Fariba Sadri** (eds.) – *Computational Logic: Logic Programming and Beyond* (Springer Verlag, Ed.)
- **Robert Veroff** – *Automated Reasoning and its Applications* (M.I.T. Press)
- **Jean-H. Gallier** – *Logic for Computer Science: Foundations of Automated Theorem Proving* (Harper Row Press)

SUPORT/ suplimente pentru Cursul 2 (5 sl)

- Ne ocupăm în plus doar de **SAT**, pentru o clasă *mai restrânsă* de formule: clasa *formulelor Horn*
- **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 $\mathbf{A}$, nu din $\bar{\mathbf{A}}$
- 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:

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

2

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

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

- Din motive tehnice, admitem și $C \triangleq \square$ (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.

3
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, ... , 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 $A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \to B$, cu $A_1, A_2, A_3, \ldots, A_k$ *marcaţi* şi B *nemarcat*, sau de forma

$A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \to \mathbf{0}$, 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** ($C = A_1 \wedge A_2 \wedge A_3 \wedge \ldots \wedge A_k \to B$)

atunci

Pasul 5. Marchează B peste tot în F

altfel

Pasul 6. i := 1

sf_Dacă

sf_Cât_timp

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 (**d** și **e** ...; începem cu **1** $\rightarrow$ B ... alte **c**)

SUPORT/ suplimente pentru Cursul 3 (30)

- Scopul principal al acestor suplimente este de a ***demonstra*** că **Algoritmul DPLL** este ***corect și complet față de* SAT**
- Pentru a simplifica înțelegerea demonstrației, vom începe prin a face câteva observații și comentarii asupra algoritmului și procedurii recursive pe care acesta o conține
- Sunt propuse și niște exerciții noi, care ar putea fi rezolvate de către cititor simultan cu parcurgerea textului

2

- Orice atom din *prop*(F) odată ales, va fi (re)asignat (față de valoarea inițială) în noua structură curentă și apoi scos din *prop*(F); odată scoși din *prop*(F), atomii nu vor mai fi reasignați
- Astfel, viditatea lui *prop*(F) poate constitui criteriu de terminare, alături de cele menționate (Ø/ {□})
- Mai precis, dacă atomul A va fi vizat în algoritm de vreo operație din ***O***, *este posibil* ca structura curentă **S** să se modifice și ea:

-dacă se efectuează **PURE** „asupra” literalului L, acesta „devine adevărat în **S**”; practic, dacă L = A, atunci **S**(A) = **1** (dacă L = ⌉A, atunci *s*-ar pune **S**(⌉A) = **1**, adică **S**(A) = **0**, ceea ce nu modifică de fapt valoarea inițială, **0**)

-dacă se efectuează **UNIT** asupra faptului {A}, atunci procedăm ca mai sus: **S**(A) = **1**; dacă operăm asupra faptului {⌉A}, nu modificăm nimic de fapt (**S**(⌉A) = **1** …)

3

-apoi, știm deja că **SPLIT** „cumulează” efectul unui „dublu” **UNIT**, deci după efectuarea unei asemenea operații asupra atomului A, în „stânga” punem **S**(A) = **1**, iar în „dreapta” **S**($\rceil$A) = **1**, adică **S**(A) = **0** (rămâne așa)

- În caz de satisfiabilitate a lui F (există măcar un drum de la F la o frunză cu „**DA**”), alegerea unor drumuri diferite poate genera structuri finale diferite (oricare dintre ele fiind însă model pentru F)
- Ca o simplă constatare, printre structurile finale există și cea ***minimală***, adică cea care „conține cele mai puține elemente de **1**” necesare pentru a fi model pentru F (în caz că există așa ceva)
- Dacă nu ne propunem găsirea tuturor modelelor pentru F în cazul satisfiabilității (sau construirea arborelui complet), structura minimală poate fi unica cerută a fi construită de algoritm

- În literatură există mai multe structuri de date pentru a memora **S**-urile: ca o listă de atomi (presupunând că cei trecuți în listă sunt „adevărați”; pentru rest, valoarea de adevăr nu e importantă, sau e implicit „fals”); ca o listă de literali (atât atomii din listă cât și negațiile lor sunt „adevărați”, pentru rest, valoarea nu este esențială), ca ***o funcție*** (vector - listă a valorilor din codomeniu), etc.
- Ca o observație simplă, dacă pentru un **SPLIT** (din cursul execuției) ambele ramuri ale branșării conduc în final la frunze de tip „**NU**”, atunci valorile (eventual diferite) pentru **S**-urile curente nu mai contează, F fiind nesatisfiabilă
- Recomandăm (pentru înțelegerea mai profundă la nivel intuitiv a situațiilor care pot apare), să găsiți singuri arborele complet (de execuție, aplicând **DP**(**LL**)) pentru formulele din exemplele care urmează (**Exemplul 1**, „făcut” complet, este la final), deducând (ne)satisfiabilitatea acestora și construind model(e) acolo unde este posibil:

Exemplul 2. F = {{A, B}, {$\rceil$A, $\rceil$B}, {$\rceil$B, C}, {$\rceil$C, A}, {C, D}} + {$\rceil$A, B}.

Exemplul 3. F = {{A, B}}.

- Pseudocodul în care este redactat **Algoritmul DP**(**LL**) este, sperăm, ușor de înțeles și traductibil imediat în orice limbaj comercial care permite exprimarea directă a recursiei
- O prezentare imperativă ar fi fost mai dificil de redat datorită operației **SPLIT**; de asemenea, terminarea – mai greu de demonstrat ...
- Să presupunem acum că ***p*** = <**S**, ***c*** > denotă perechea/ ***configurația*** (structură „model”, respectiv formulă/ mulțime de clauze) curentă (inițial, ***c*** este reprezentarea deja fixată a lui F, iar **S** este „pusă pe **0**”), adică procesată pe parcursul execuției **DP**(**LL**), iar ***p’*** = <**S’**, ***c ’***> este perechea obținută în urma execuției efective a unui „pas” din algoritm (operație din ***O***, în general)

6

- Deci ***p'*** se obține din ***p***, în sensul că lui ***c*** i se aplică una dintre operațiile ***o*** $\in$ ***O*** (pentru a se obține ***c'***); în anumite situații, și dacă se dorește, și **S'** se obține din **S** în sensul că este posibil să fie „schimbată" (în **S'** față de **S**) <u>o singură</u> valoare, și anume cea aferentă literalului „ales" A
- Presupunem și că ***p'*** nu este o ***configurație finală***, adică ***c'*** nu este nici vidă, nici {□} (***c*** nu conține 2 clauze de forma {A} și {$\rceil$A}, cu A $\in$ *prop*(F))
- În urma aplicării unei operații **SUBS** rezultatul este previzibil (*supraclauzele nu influențează satisfiabilitatea*) și nu se selectează niciun literal (**S'** este chiar **S)**; apoi, avem ***c*** := ***c'*** $\cup$ {D}, unde clauza D $\supseteq$ C, cu C $\in$ ***c'***
- Este posibil (vezi **SPLIT**) să nu putem trata pe A / $\rceil$A „în oglindă"/ „inversate" și simultan; atunci, alături de ***p'***, este nevoie, pentru claritatea exprimării rezultatului, și de un ***p''*** = <**S''**, ***c''*** > (simultan)

7

- Operațiile care rămân în atenție sunt astfel: **PURE**, executată după alegerea unui L $\in$ C ($\in$ ***c***), indiferent dacă L = A sau dacă L = $\rceil$A (operație pentru care ***nu*** se generează o branșare în arborele de execuție); **UNIT**, în care este implicată alegerea unei clauze C = {$\rceil$A} (sau C = {A}), rolurile atomului L = A și a lui L = $\rceil$A ca literali complementari fiind „inversate" în fiecare caz (deși iar ***nu*** se generează o branșare, am putea vorbi totuși de o anumită simultaneitate și de folosirea atât a lui ***p'*** cât și a lui ***p''***); și, desigur, **SPLIT**, care se execută asupra unei clauze (neunitare) C $\in$ ***c*** și asupra unui atom A $\in$ C, caz în care se va obține o branșare (cu „inversare" **și** simultaneitate, ***p'***, ***p''*** etc.)
- În fiecare caz, literalul implicat nu a mai fost însă asignat în **S**-ul curent (exceptând pasul inițial, dinainte de începerea execuției lui **DP**(**LL**))

- Desigur că în cazul în care se dorește a ști doar dacă F este sau nu satisfiabilă, referirile la **S** și/ sau A / $\rceil$A se pot omite din toate considerațiile (trecute sau viitoare)

Teorema 1. Date ***p*** și ***p'/ p''*** ca mai înainte, unde ***p'***/ ***p''*** sunt obținute printr-una dintre operațiile **PURE** sau **UNIT** (deci fără branșare), atunci ***c*** este satisfiabilă ddacă ***c'/ c''*** este satisfiabilă. *În particular*: **S** ⊨ ***c*** ddacă **S'** ⊨ ***c'*** (sau **S''** ⊨ ***c''***).

Teorema 2. Dată configurația ***p*** = <**S**, ***c*** > și configurațiile (obținute prin branșare, aplicând **SPLIT**), $\boldsymbol{p_1}$ = < $\mathbf{S_1}$, $\boldsymbol{c_1}$> și $\boldsymbol{p_2}$ = < $\mathbf{S_2}$, $\boldsymbol{c_2}$> ($\boldsymbol{p_1}$ joacă rolul lui ***p'*** pentru „stânga”, iar $\boldsymbol{p_2}$ joacă rolul lui ***p''*** pentru „dreapta”), avem: ***c*** este satisfiabilă ddacă fie $\boldsymbol{c_1}$ este satisfiabilă, fie $\boldsymbol{c_2}$ este satisfiabilă (neexclusiv). Altfel spus, ***c*** este nesatisfiabilă ddacă atât $\boldsymbol{c_1}$ cât și $\boldsymbol{c_2}$ sunt nesatisfiabile. *În particular*, avem și: **S** ⊨ ***c*** ddacă fie $\mathbf{S_1} \vDash \boldsymbol{c_1}$, fie $\mathbf{S_2} \vDash \boldsymbol{c_2}$ (neexclusiv).

- Considerăm necesar ca, înainte de a prezenta demonstrațiile formale ale teoremelor (care, practic, stabilesc corectitudinea **DP**(**LL**)), să facem alte câteva considerații care să fie utile pentru înțelegerea acestora
- Acceptând că terminarea este deja demonstrată și lăsând pe moment deoparte structura **S**, ideea principală este ca fiecare operație din ***O*** să „păstreze”/ să aibă ca ***invariant*** satisfiabilitatea (respectiv nesatisfiabilitatea) „formulei” căreia i se aplică
- Mai exact, pornind, să zicem, cu c_1 (inițial, c_1 reprezintă acum chiar formula F din **LP**), să presupunem mai întâi că aplicăm de câteva ori operații de tipul **PURE** sau **UNIT**, „ajungând” într-un c_k, diferit (deocamdată) de Ø sau {□} (cu așa ceva începe de fapt procedura **DPLL**)
- **Teorema 1** „ne spune” că dacă c_1 (adică F) este satisfiabilă, atunci și c_k este satisfiabilă; echivalent, putem spune că dacă c_1 este nesatisfiabilă, atunci și c_k este nesatisfiabilă

10

- Pentru ultima afirmație, am folosit faptul că relația „≡” este tranzitivă; în plus, echivalențele $p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$ și $p \to q \equiv \rceil q \to \rceil p$, sunt adevărate
- Dacă vreun $\boldsymbol{c_i}$ ($2 \leq i \leq k$) este Ø sau {□} atunci ne referim la $\boldsymbol{c_k} = \boldsymbol{c_{i-1}}$
- Să presupunem acum că îl prelucrăm pe $\boldsymbol{c_k}$ și aplicăm operația **SPLIT**, rezultând $\boldsymbol{c_{k1}}$ pentru „stânga” și $\boldsymbol{c_{k2}}$ pentru „dreapta”
- **Teorema 2** „ne spune” că $\boldsymbol{c_k}$ este satisfiabilă ddacă ($\boldsymbol{c_{k1}}$ este satisfiabilă <u>sau</u> $\boldsymbol{c_{k2}}$ este satisfiabilă); echivalent, înseamnă și că: $\boldsymbol{c_k}$ este nesatisfiabilă ddacă ($\boldsymbol{c_{k1}}$ este nesatisfiabilă <u>și</u> $\boldsymbol{c_{k2}}$ este nesatisfiabilă)
- Aici ne-am folosit din nou de echivalențele de mai sus și de faptul că relația „≡” este tranzitivă; dar și de legea lui deMorgan: $\rceil(p \vee q) \equiv \rceil p \wedge \rceil q$
- Concluzia ar fi acum că, pornind cu F, care poate fi satisfiabilă sau (exclusiv !) nesatisfiabilă și aplicând succesiv („formulei” inițiale și „formulelor” intermediare G)

operațiile posibile (elemente ale lui ***O***), toate acestea vor avea ***același caracter*** cu F, adică ***satisfiabil*** sau ***nesatisfiabil***; și este vorba inclusiv de „formula finală” (Ø = {} = satisfiabilitate, sau {□} = nesatisfiabilitate)

- În cazul lipsei vreunei branșări, afirmația precedentă este aproape imediat evidentă, deși, formal, mai este nevoie și de o demonstrație prin inducție matematică (mai ales pentru a „prinde” și cazul în care F este o mulțime numărabilă de clauze – cf. **Teoremei de compactitate**, care va fi enunțată în cursul urnmător)
- În cazul unei branșări, dacă G este satisfiabilă, atunci măcar una dintre succesoarele sale G1/ G2 este satisfiabilă (și știm că $\mathbf{0} \vee \mathbf{1} = \mathbf{1} \vee \mathbf{0} = \mathbf{1} \vee \mathbf{1} = \mathbf{1}$); deci, oricum „rămânem pe **1**” până la final
- Dacă, la o branșare, „formula” G în cauză este nesatisfiabilă, atunci succesoarele G1/ G2 rămân ambele nesatisfiabile și, din nou ($\mathbf{0} \wedge \mathbf{0} = \mathbf{0}$), „valoarea” (**0**) este „purtată” până la final

- În demonstrații, nu vom „scăpa” nici în acest caz de o inducție, iar ambele teoreme vor fi mai întâi demonstrate fără a lua în discuție structura **S**
- *Cazurile particulare* (a se vedea sfârșitul enunțurilor), care implică și structura, vor fi „prinse” astfel în afirmația, denotată **(FAPT)**: structura **S *construită la final de algoritm este într-adevăr model pentru intrarea*** F (în cazul în care aceasta este formulă satisfiabilă)

Demonstrația Teoremei 1. Fie configurațiile notate

p = <**S**, ***c***>, ***p’*** = <**S’**, ***c’***> și ***p’’*** = <**S’’**, ***c’’***>, pentru a respecta notațiile folosite în procedură.

-Să presupunem mai întâi că ***p’*** este obținut din ***p*** în urma execuției unei operații **PURE**, adică a unui **Pas 11** din procedura **DPLL**.

a)Fie C o clauză din ***c*** care conține atomul A, care este pur, adică ⌉A nu mai apare în nicio (altă) clauză din ***c*** (adică A $\in$ *prop*(F)). În acest caz, ***c'*** se obține din ***c*** prin eliminarea tuturor clauzelor care conțin A (inclusiv C). Dacă ***c*** este satisfiabilă, fie ***I*** un model oarecare al său; atunci toate clauzele din ***c*** sunt adevărate în ***I***, prin urmare așa sunt și toate clauzele rămase în ***c'***. Astfel, avem și ***I*** ⊨ ***c'***, adică ***c'*** este satisfiabilă. Invers, să presupunem că ***c'*** este satisfiabilă. Atunci există ***I***, cu ***I*** ⊨ ***c'***. Cum ***c'*** conține doar clauze din ***c*** (care nu conțin A) pentru a arăta că și ***c*** este satisfiabilă va trebui să extindem pe ***I*** la un ***I'***, astfel încât ***I'*** să fie definit și pentru A (și, în același timp, rămânând model pentru ***c***). Este evident că vom obține ceea ce vrem, dacă vom pune ***I'***(A) = **1**. În acest caz, clauzele din ***c*** care-l conțin pe A vor fi satisfiabile (advărate în ***I'***), iar cele care nu-l conțin pe A (și nici pe $\rceil$A, de altfel !) se regăsesc și în ***c'***, deci rămân și ele adevărate (și) în ***I'***.

b)În acest caz, trebuie să „inversăm” pe A cu $\rceil$A. Raționamentul este însă „în oglindă”: există C $\in$ ***c*** , C conține pe $\rceil$A, care este pur (A $\in$ *prop*(F)); ***c'*** se obține din ***c*** prin eliminarea tuturor clauzelor care conțin $\rceil$A (inclusiv clauza C); … ; se ia ***I'***(A) = **0** … etc. Singura diferență „de scos în evidență” este aceea că vom pune (la momentul corespunzător) ***I'***(A) = **0**.

c)Dacă ne ocupăm de **(FAPT)**, este imediat că, dacă luăm **S** = ***I***, atunci structura **S'** construită de procedură va fi exact ***I'*** definită mai sus, prin urmare teorema este complet demonstrată pentru cazul operației **PURE**. Să remarcăm și că, în ambele situații, A se „scoate” din *prop*(F) și se „produce” (prima și ultima, exceptând-o pe cea inițială) sa (re)asignare.

-<u>Să presupunem acum că ***p'/ p''*** sunt obținute din ***p*** în urma execuției unei operații **UNIT**</u>, adică a unui **Pas 17/ Pas 22** din procedura **DPLL**.

a)Am ales deci A, A $\in$ *prop*(F), cu {A} $\in$ ***c*** ({A} este o clauză unitară pozitivă, adică un fapt pozitiv, din „formula" curentă). Conform **Pasul**ui **17**, găsim noua configurație ***p'*** = <**S'**, ***c'***>, unde: **S'**(A) = **1** (în rest, **S'** coincide cu **S**), A se asignează pentru ultima oară (în structura finală, prima oară fiind la inițializare), scoțându-se apoi din *prop*(F), și:

c' := {C $\in$ ***c*** | A $\notin$ C și ($\rceil$A) $\notin$ C} $\cup$ {C \ {$\rceil$A} | C $\in$ ***c***}.

Raționând ca în cazul precedent, trebuie să arătăm întâi: dacă ***c*** este satisfiabilă atunci ***c'*** este satisfiabilă. Fie ***I***, cu ***I*** $\models$ ***c***. Dacă ***I***(A) = **1**, atunci ***I***($\rceil$A) = **0**. Astfel, scoaterea lui $\rceil$A din clauzele C (rezultând niște C') ale lui ***c*** nu afectează valoarea de adevăr a acestora (și aveam ***I***(C) = **1**); avem atunci și ***I***(C') = **1** (pentru toate asemenea clauze C'). Mai mult, clauzele din ***c*** care conțineau pe A au fost eliminate în/ din ***c’***. Rezultă imediat că

I ⊨ ***c’***. Invers, să presupunem că există ***I***, model pentru ***c’***, și să arătăm că, în acest caz, putem construi și un model pentru ***c*** . Este însă imediat faptul că ***I*** extins la ***I’*** (***c’*** nu conține nici pe A, nici pe ⌉A), cu ***I’***(A) = **1**, va satisface condiția ***I’*** ⊨ ***c***: clauzele „nou apărute” fie vor conține pe A (deci vor fi adevărate în ***I’***), fie sunt cele din ***c’***, care erau adevărate în ***I***, dar conțin în plus pe ⌉A (le notăm temporar cu D). Este drept că ***I’***(⌉A) = **0**, dar pentru restul atomilor, ***I’*** coincide cu ***I*** (să observăm și că asemenea clauze D nu pot fi unitare, D = {⌉A}; atunci am fi avut, de la început în ***c*** , atât pe {A} cât și pe {⌉A} și am fi terminat deja procedura și algoritmul).

b)Să tratăm acum situația din **Pasul 22**, prin alegerea unui A, A ∈ *prop*(F), cu {⌉A} ∈ ***c*** , {⌉A} fiind o clauză unitară negativă, adică un fapt negativ din „formula” curentă. Conform **Pasul**ui **22**, găsim noua configurație

p’’ = <**S’’**, ***c’’***>, unde: **S’’**(A) = **0** și (în rest, **S’’** coincide cu **S**), A se asignează pentru ultima oară (în structura finală, prima oară fiind la inițializare), scoțându-se apoi din *prop*(F), iar

c’’ := {C $\in$ ***c*** | A $\notin$ C și (⌉ A) $\notin$ C} $\cup$ {C \ {A} | C $\in$***c***}.

Demonstrația este aproape identică cu cea de la punctul a) anterior: se înlocuiește A cu ⌉ A și reciproc (simultan), **0** cu **1** (tot reciproc și simultan), ***I*’** cu ***I*’’**, **S’** cu **S’’**, etc.

c)Tratăm cazul particular, **(FAPT)**. Ca și în cazul operației **PURE**, dacă luăm **S** = ***I***, atunci structurile **S’/ S’’** construite de procedură vor fi exact structurile ***I’/ I’’*** definite înainte, prin urmare teorema este complet demonstrată și pentru operația **UNIT**. Să remarcăm și faptul că referirea la „prima și ultima (exceptând-o pe cea inițială) (re)asignare a unui atom ales A”, va conduce la acceptarea adevărului observației că **S** final va ***putea*** conține „cel mai mic număr posibil de valori egale cu **1**” (depinde însă și de alegerile făcute pe parcurs.

q.e.d.

Demonstrația Teoremei 2. Fie configurația curentă $\boldsymbol{p}$ = <**S**, $\boldsymbol{c}$>, neterminală, și noile configurații (obținute prin branșare, aplicând **SPLIT**, adică executând pașii **27-33** din **procedure DPLL**(**S**, $\boldsymbol{c}$), procedură care furnizează în cazul de față un rezultat doar de forma **0**/ **1**), folosindu-se pentru (re)apelare configurațiile $\boldsymbol{p_1}$ = <$\mathbf{S_1}$, $\boldsymbol{c_1}$> ($\boldsymbol{p_1}$ joacă rolul lui $\boldsymbol{p'}$ pentru „stânga") și, în rolul lui $\boldsymbol{p''}$ pentru „dreapta", $\boldsymbol{p_2}$ = <$\mathbf{S_2}$, $\boldsymbol{c_2}$>. Mai exact:

$\boldsymbol{c'} := \{C \in \boldsymbol{c} \mid A \notin C$ și $(\rceil A) \notin C\} \cup \{C \setminus \{\rceil A\} \mid C \in \boldsymbol{c}\}$,

$\boldsymbol{c''} := \{C \in \boldsymbol{c} \mid A \notin C$ și $(\rceil A) \notin C\} \cup \{C \setminus \{A\} \mid C \in \boldsymbol{c}\}$,

S'(A) = **1** (în rest, **S'** coincide cu **S**) și **S''**(A) = **0** și (în rest, **S''** coincide cu **S**). Mai mult (**Pasul 33**), înainte de orice apel ulterior, A se scoate din *prop*(F).

a)Să presupunem că ***c*** este satisfiabilă și să arătăm că: fie ***c'*** este satisfiabilă, fie ***c''*** este satisfiabilă. Fie atunci ***I*** o structură care este model pentru ***c***. Știm că atât A, cât și $\neg$A nu se mai regăsesc nici în clauzele rămase în ***c'***, nici în cele rămase în ***c''***. Ca urmare, orice structură care ar putea fi model pentru ***c'*** sau ***c''*** (construită sau nu pornind de la ***I***), nu trebuie să fie definită pentru A. Nu putem avea însă decât ***I***(A) = **1** sau ***I***(A) = **0**. Să rezumăm întreaga situație în care ne aflăm:

-A este într-o clauză C din ***c***, neunitară;

-Există cu siguranță (măcar) o altă clauză (D) în ***c***, care conține pe $\neg$A; altfel, A ar fi literal pur și s-ar elimina din ***c***, prin aplicarea unui **PURE**; mai mult, D nu poate fi nici ea o clauză unitară (negativă), deoarece atunci s-ar putea aplica acum o operație **UNIT**;

-A și $\neg$A nu se află într-o aceeași clauză.

Conchidem că: în ***c*** se află clauze neunitare (notate cu C) care-l conțin pe A (dar nu-l conțin pe $\rceil$A), clauze neunitare (notate cu D) care-l nu-l conțin pe A (dar care-l conțin pe $\rceil$A) și, eventual, alte clauze (notate cu E), dar tot neunitare și care nu conțin nici A, nici $\rceil$A (dacă ar mai exista clauze unitare, bazate pe alt literal, din nou am aplica mai întâi o operație **PURE** sau **UNIT**); mai mult, toate aceste clauze sunt adevărate în ***I***: ***I*** $\models$ C, ***I*** $\models$ D, ***I*** $\models$ E. Fie atunci structura ***J***, care coincide cu ***I***, peste tot (doar că nu este definită pe A). Dacă ***I***(A) = **1**, atunci ***I***($\rceil$A) = **0**. Cum în ***c'*** nu avem clauze de tip C, ci doar de tipul E (nemodificate), care satisfac imediat ***J*** $\models$ E și, să zicem, D' (provenite dintr-un D, fără însă a-l mai conține pe $\rceil$A), avem și ***J*** $\models$ D' (***I***($\rceil$A) = **0**, dar ***I***(D) = **1**). Prin urmare, <u>dacă ***I***(A) = **1**, atunci ***J*** $\models$ ***c'***</u>. Raționând într-un mod similar („inversând" pe A cu $\rceil$A etc.), se observă că <u>dacă ***I***(A) = **0**, atunci ***J*** $\models$ ***c''***</u>.

b)<u>Invers</u>, să presupunem că fie ***c'*** este satisfiabilă, fie ***c''*** este satisfiabilă, și să arătăm că ***c*** este satisfiabilă. Să presupunem, de exemplu (păstrând toate notațiile de mai înainte), că <u>***c'*** este satisfiabilă</u> și fie J_1 o structură care este model pentru ***c'***, $J_1 \vDash c'$. Extindem, mai întâi, pe J_1 la J_2, astfel încât J_2 să fie definit și pentru toți atomii noi, care apar (eventual) în plus în ***c''*** (nu ne interează valoarea de adevăr a lui ***c''*** în J_2). Putem găsi acum o structură corectă pentru ***c*** (s-o notăm cu ***I***) pornind de la J_2. Vom pune întâi ***I***(B) = J_2(B), pentru fiecare B $\neq$ A. Singurul lucru pe care trebuie să-l mai facem, este să definim ***I***(A) (A nu era prezent nici în ***c'***, nici în ***c''***). Evident că vom pune ***I***(A) = **1**, pentru a avea ***I***(***c***) = **1**. Într-adevăr, pentru clauzele de tip C din ***c*** avem ***I***(C) = **1**, deoarece ***I***(A) = **1**. Clauzele de tip E'/ E, se „transportă" fidel (din ***c'*** în ***c***) și deci avem și pentru ele ***I***(E) = **1**. În sfârșit, și pentru clauzele de tip D'/ D avem că ***I***(D) = **1**, deși avem ***I***($\rceil$ A) = **0** (pentru că J_2(D') = **1**, iar D, față de D', nu-l conține în plus decât pe $\rceil$ A). Concluzionăm că ***c*** este satisfiabilă. Un raționament similar se aplică dacă plecăm cu ipoteza că <u>***c''*** este satisfiabilă</u> (vom pune ***I***(A) = **0**).

c)Pentru a demonstra și cazul particular **(FAPT)** din **Teorema 2**, este clar că dacă $\mathbf{S_1}$/ $\mathbf{S_2}$ vor „juca rolul" acelui „generic" $\boldsymbol{J_2}$ din cele discutate anterior, atunci **S** va „juca rolul" unei structuri ***I*** de mai înainte. Este evident că va trebui să punem **S**(A) = **1** în cazul în care ***c'*** este satisfiabilă și **S**(A) = **0** în cazul în care ***c''*** este satisfiabilă.

q.e.d.

- Să repetăm faptul că, exceptând pre-inițializarea cu **0**, odată ce un literal L este selectat de o operație (**PURE**, **UNIT**, **SPLIT**), el va fi (re)inițializat (valoarea sa în noua structură **S** construită rămâne **0**, sau se modifică în **1**); apoi, acesta (dacă este atom) sau/ și complementarul său, se elimină din *prop*(F), și oricum din noua „formula" curentă „de prelucrat"; noile structuri construite vor afecta apoi unii dintre literalii rămași; atomii neselectați pe parcursul execuției algoritmului, rămân asignați pe **0** (în structura finală, folositoare sau nu), dar li se poate „da" și valoarea **1**

- Am mai spus că tructurile pot fi reprezentate și ca liste; atunci algoritmul „pleacă” cu lista vidă; orice literal nou selectat de o operație se va „trece” în capul listei, fie că este atom, fie că este negația unui atom; dacă la terminarea algoritmului vom avea răspunsul „**DA**”, adică F inițială este satisfiabilă, atunci valoarea sa (***true***) trebuie să se obțină considerând că orice literal din listă are valoarea ***true*** (fie că este pozitiv, fie negativ); atomii care nu apar în listă pot avea asignată orice valoare de adevăr (de unde și existența posibilă a mai multor modele)
- Să remarcăm și faptul (evident acum) că, pe parcursul execuției, „formula curentă rămasă de prelucrat” (***c***), ca de altfel și *prop*(F), devin din ce în ce mai „scurte”, în timp ce structura, ca listă, devine din ce în ce mai „lungă”; doar operațiile **SUBS** nu afectează o asemenea listă (totuși, aceste operații nu pot fi scoase din corpul procedurii și efectuate o singură dată, similar cu, de exemplu, „operația” de eliminare a tautologiilor)

- Putem demonstra astfel chiar ceva mai precis:

Propoziție. Presupunând că structura **S** este reprezentată ca o listă, la fiecare apel al procedurii, **DPLL**(**S**, ***c***)/ **DPLL**($\mathbf{S_1}$, $\boldsymbol{c_1}$)/ **DPLL**($\mathbf{S_2}$, $\boldsymbol{c_2}$), dacă un literal L apare în **S**/ $\mathbf{S_1}$ /$\mathbf{S_2}$, atunci nici el, nici complementarul său, nu apar în ***c***/$\boldsymbol{c_1}$ /$\boldsymbol{c_2}$.

Demonstrație. Este imediată, urmărind cum se construiesc noile structuri și noile „formule” curente în urma execuției oricărui pas al procedurii.

q.e.d.

- Afirmația din propoziția anterioară devine un invariant suplimentar pentru fiecare pas executat de **DP**(**LL**); deoarece pornim cu **S** ca fiind lista vidă, la final ea va conține toți literalii ***aleși*** pentru a fi prelucrați (și va trebui să fie model pentru F, în cazul satisfiabilității acesteia și în sensul precizat; dacă privim **S** ca o funcție (inițializată cu **0**), ne putem gândi, așa după cum am mai remarcat, la un mod de a face alegerile a.î. **S** să „conțină un număr minim de **1**”

25

- În acest moment, putem enunța teorema finală; demonstrația formală completă este puțin mai complicată decât pare, chiar folosind rezultatele deja obținute; acest lucru se datorează în special faptului că, după cum am amintit, în arborele total de execuție a algoritmului, orice „ramificare” poate fi de două tipuri distincte: ***branșare*** și simplă ***alegere*** nedeterministă; în acest mod, invarianții nu pot fi pur și simplu „purtați” de la rădăcină la frunze (care sunt și ele de două tipuri: „de tip” **0** și „de tip” **1**)

Teorema 3 (de ***corectitudine*** și ***completitudine*** pentru **Algoritmul DP**(**LL**) față de **SAT**). Fie F $\in$ **LP**, o formulă oarecare reprezentată așa cum este **Intrarea** în **Algoritmul DP**(**LL**). Atunci:

a) (**Completitudine**) **DP**(**LL**) se termină.

b) (**Corectitudine**) F este nesatisfiabilă ddacă **DPLL**(**S**, ***c***) = **0**. Echivalent, F este satisfiabilă ddacă **DPLL**(**S**, ***c***) = **1**. În plus, dacă F este satisfiabilă, avem și **S** $\models$ F.

Demonstrație.

a)Terminarea am demonstrat-o deja, chiar de mai multe ori. Simplificând, pe orice fel de „drum” am „apuca-o” (de la rădăcină la orice frunză, în arborele complet de execuție a algoritmului), numărul de atomi și/ sau numărul de clauze (din „formula” curentă) se micșorează odată cu „trecerea” la un nou nod.

b)Demonstrăm, prin inducție după numărul inițial, *k*, de elemente din *prop*(F) (numărul de variabile propoziționale peste care este construită intrarea F) că afirmația următoare este adevărată:

(AF) „(F este nesatisfiabilă ddacă **DPLL**(**S**, ***c***) = **0**) **și** (Structura **S** furnizată ca ieșire de algoritm este model pentru F – asta doar pentru cazul în care F este satisfiabilă, adică **DPLL**(**S**, ***c***) = **1**)”.

k* = 0**. Acest caz putea fi omis, intrarea în algoritm (de fapt, ***c-ul inițial) fiind presupusă a fi un șir nevid de caractere. Adică, F nu este construită peste niciun atom (ar putea fi – prin extensie – direct constantele booleene **0**/ **1**). Oricum, și în acest caz special, este clar că **(AF)** este *adevărată prin lipsă/ true by default*.

k* = 1**. Prin urmare, execuția algoritmului începe cu ***c (nu uităm, aceasta este o mulțime de mulțimi de literali) construită peste un singur atom (să zicem, *prop*(F) = {A}). După „perierile" de rigoare (ceea ce ne-ar putea aduce totuși în situația anterioară, datorită eliminării tautologiilor), s-ar putea să avem ***c*** = {{A}}, ***c*** = {{⌉A}} sau ***c*** = {{A}, {⌉A}}. Primele două cazuri se tratează similar (este vorba despre un literal pur și se aplică **Pasul 11** din procedura **DPLL**), răspunsul algoritmului fiind „**1**" („F este satisfiabilă" și **S**(A) = **1**/ **S**(⌉A) = **1**). În ultimul caz, se aplică **Pasul 3** din procedură și răspunsul algoritmului este „**0**" („F este nesatisfiabilă", fără a ne interesa **S**). În toate situațiile, răspunsurile sunt ***corecte***.

<u>*k* > 1</u>. Fie configurația curentă ***p*** = <**S**, ***c***>, în care ***c*** este construită peste *k* variabile propoziționale și să presupunem că executăm un pas din **Algoritmul DP**(**LL**), adică facem apelul de procedură **DPLL**(**S**, ***c***). Dacă suntem în cazurile tratate în **Pasul 1**, **Pasul 3** sau **Pasul 7** din procedură, fie că algoritmul se va opri (**Pasul 1**, **Pasul 3**), fie că va continua (**Pasul 7**; aici este vorba despre eliminarea unei clauze subsumate, adică execuția unei operații **SUBS**), posibil chiar cu același număr de (*k*) de atomi, este clar că fie răspunsul (final) este corect, fie continuarea va implica (la un moment ulterior) micșorarea lui *k*. Altfel, se va executa unul dintre **Paș**ii **11**/ **17**/ **22**/ **33** din procedură (corespunzători execuției operațiilor: **PURE**, **UNIT** pentru un atom A sau pentru negația sa ⌉A, respectiv **SPLIT**). Pentru primele 2-3 cazuri va rezulta noua configurație curentă ***p'*** = <**S'**, ***c'***>, iar în ultimul caz (**SPLIT**) trebuie luate în considerare configurațiile $\boldsymbol{p_1}$ = < $\mathbf{S_1}$, $\boldsymbol{c_1}$> și $\boldsymbol{p_2}$ = < $\mathbf{S_2}$, $\boldsymbol{c_2}$>. În fiecare situație însă, toate noile „formule” curente (***c'***/ $\boldsymbol{c_1}$/ $\boldsymbol{c_2}$) vor fi construite peste (cel mult) *k – 1* atomi. Conform ipotezei inductive, **(AF)** este adevărată. Acum se aplică (pentru fiecare situație în parte, în mod corespunzător) una dintre **Teoremele 2/ 3** și obținem că **(AF)** este adevărată și pentru *k*.

q.e.d.

- Exemplul care urmează (denotat **Exemplul 4**) poate fi prezentat la Seminar
- Recomandăm însă ca acesta să fie rezolvat în paralel cu percurgerea demonstrației și complet, în „spiritul” celor deja menționate: construirea arborelui total de execuție, folosind reprezentarea lui **S** ca listă etc.

- **Exemplul 4**. Fie $F \in$ **LP**, considerată deja în forma dorită (nevidă, reprezentată în **FNC**, ca mulțime de mulțimi de literali etc.), adică $\boldsymbol{c} = \{C_1, C_2, C_3, C_4\}$, unde $C_1 = \{A, B\}$, $C_2 = \{D, \rceil B, \rceil C\}$, $C_3 = \{\rceil A, C\}$, $C_4 = \{\rceil D\}$. Să se arate că F este satisfiabilă folosind **Algoritmul DP**(**LL**) și găsind <u>toate</u> structurile care sunt model pentru F.

SUPORT/ suplimente pentru Cursul 4 (7 sl)

- Este suficient să completăm acest curs cu ***strategii ale rezoluției***
- Strategiile nu restrâng, conceptual, spaţiul de căutare (*graful total*) dar folosesc anumite *informaţii suplimentare despre clauze*, astfel încât să crească şansele pentru selectarea rapidă a unei *respingeri*, adică a unui „cel mai scurt" drum pornind de la frunze (elementele lui F), către rădăcina, sperăm, □
- La modul ideal graful total nu se va construi deci în întregime, ci doar acele porţiuni din el (cât mai puţine şi cât mai „mici"), care, posibil, vor „conţine" (măcar) o respingere

2

- 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
- Deducem că ***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

- 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, de altfel, și ***restricția unitară***)

- 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.

5

- ***Rezoluția liniară*** *se bazează pe o* ***clauză inițială***
- Considerăm astfel $F \in$ **LP**, $F = \{C_1, C_2, ..., C_n\}$ (numite *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 ...*)

6

- 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*)

7

- Pentru a obține o **SLD**-rezoluție „clasică” (adică, folosind doar 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** și – eventual, dezvoltarea altor concepte – la Seminar (dacă este timp ...), folosind cartea mea (tipărită)

SUPORT/ suplimente pentru Cursul 5 (18 sl)

- Putem începe cu alte completări legate de funcțiile booleene
- De exemplu, folosind **Tabelul 1.1**, pag.30 (din cartea mea „scrisă”, **Capitol**ul **1**), prezentat și pe slide-ul următor, se pot rezolva anumite exerciții nu doar semantic (folosind tabelele de adevăr ale funcțiilor booleene implicate), ci și sintactic, „legile” acționând ca niște *reguli de inferență* (așa-numite) *ne-logice*
- De altfel, chiar legile din **Tabel** (notate, nu întâmplător, cu 6) - 13) (respectiv 6’) - 13’)) pot fi demonstrate sintactic pornind de la ***axiomele*** unei algebre booleene (1) - 5), respectiv 1’) - 5’))

2

6) $\overline{\overline{x}} = x$	6') $\overline{\overline{x}} = x$
7) $x \bullet \overline{x} = 0$	7') $x + \overline{x} = 1$
8) $x \bullet x = x$	8') $x + x = x$
9) $x \bullet 0 = 0$	9') $x + 1 = 1$
10) $x \bullet 1 = x$	10') $x + 0 = x$
11) $x_1 \bullet x_2 \bullet \ldots \bullet x_n = 0$ dacă şi numai dacă există $i \in [n]$ astfel încât $x_i = 0$ (oricare ar fi $n \geq 2$ şi oricare ar fi $x_1, x_2, ..., x_n \in B$)	11') $x_1 + x_2 + \ldots + x_n = 1$ dacă şi numai dacă există $i \in [n]$ astfel încât $x_i = 1$ (oricare ar fi $n \geq 2$ şi oricare ar fi $x_1, x_2, ..., x_n \in B$)
12) $x_1 \bullet x_2 \bullet \ldots \bullet x_n = 1$ dacă şi numai dacă pentru fiecare $i \in [n]$ avem $x_i = 1$ (oricare ar fi $n \geq 2$ şi oricare ar fi $x_1, x_2, ..., x_n \in B$)	12') $x_1 + x_2 + \ldots + x_n = 0$ dacă şi numai dacă pentru fiecare $i \in [n]$ avem $x_i = 0$ (oricare ar fi $n \geq 2$ şi oricare ar fi $x_1, x_2, ..., x_n \in B$)
13) $\overline{x_1 \cdot x_2 \cdot \ldots \cdot x_n} = \overline{x}_1 + \overline{x}_2 + \ldots + \overline{x}_n$ (oricare ar fi $n \geq 2$ şi oricare ar fi $x_1, x_2, ..., x_n \in B$)	13') $\overline{x_1 + x_2 + \ldots + x_n} = \overline{x}_1 \cdot \overline{x}_2 \cdot \ldots \cdot \overline{x}_n$ (oricare ar fi $n \geq 2$ şi oricare ar fi $x_1, x_2, ..., x_n \in B$)

3

- Apoi, să lăsăm la o parte, pentru moment, reprezentarea oricărei funcții booleene printr-o „tabelă de adevăr” (de fapt, acolo se explicitează valoarea funcției în fiecare punct, domeniul de definiție fiind finit)
- În acest caz, avem reprezentarea funcțiilor „ca text”, **FNC(P)** și **FND(P)** nefiind altceva decât niște ***reprezentări standard***, după anumite criterii
- Criteriul „numărul total de *apariţii* ale (numelui) argumentelor/ variabilelor în reprezentarea textuală a lui f ” (apariţia unei aceleiaşi variabile pe *poziţii* diferite numărându-se distinct), adică *lungimea* textului, ca expresie a lui f (notată n(f) mai jos), poate genera *alte tipuri de* forme normale

4

- Folosind această *măsură*, putem numi, de exemplu, *formă normală disjunctivă minimală* (**FNDM**) pentru o funcție f $\in$ **FB** orice **FND**, f', pentru f, astfel încât:

  n(f') = *min* {n(φ) | φ este **FND** pentru f}
- Dată astfel f $\in$ **FB**, se poate pune problema determinării ***tuturor*** **FNDM** pentru f, sau, din nou, măcar a uneia *standard* (a se vedea **Algoritmul lui Quine** ... <u>voi</u>)
- În cazul precedent, putem lua (sau nu) în calcul și operatorii/ conectorii logici care apar în expresia de difinire a funcției (oricum, fiecare variabilă și fiecare operator se consideră a avea lungime 1)

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

(I) Clasa *funcţiilor booleene* ***elementare*** (numite și ***proiecții***) este:

$\mathbf{E} = \{ i_p^n \mid n \in \mathbf{N^*}, 1 \leq p \leq n, i_p^n : \mathbf{B}^n \to \mathbf{B}$, dată prin $i_p^n(x_1, x_2, \ldots, x_p, \ldots, x_n) = x_p\}$

(II) Fie $n \in \mathbf{N^*}$, t un număr natural, f, $h_1, h_2, \ldots, h_t$ elemente din $\mathbf{FB}^{(n)}$ şi $g \in \mathbf{FB}^{(t)}$.

Spunem că $f \in \mathbf{FB}^{(n)}$ *se obţine din g, h_1, h_2, ... , h_t prin* ***superpoziţie*** dacă pentru fiecare $x = \langle x_1, x_2, ... , x_n\rangle$ avem: $f(x) = g(\langle h_1(x), h_2(x), ... , h_t(x)\rangle)$; notație:

$$f = \mathbf{SUP}(g, h_1, h_2, ..., h_t).$$

- Fie acum $Y \subseteq \mathbf{FB}$, oarecare. Se numeşte Y-*şir* orice *secvenţă (listă) finită* $f_0, f_1, ... , f_r$ de funcţii booleene în care fiecare f_i este fie din **E** U Y, fie se obţine prin superpoziţie din alte funcţii, „aflate în aceeaşi listă, dar *înaintea* lui f_i" (alte **c** ...)
- Mulțimea $\bar{Y}$, a funcțiilor care sunt prezente/ pot fi incluse în Y-șiruri, se numește ***închiderea*** lui Y
- **Definiție***.* O mulțime $Y \subseteq \mathbf{FB}$ se numește închisă dacă $Y = \bar{Y}$.

- Clase speciale (dar *banale*) de mulțimi închise: Ø, **E**, **FB**
- Exemple *nebanale* de mulțimi închise: $\mathbf{T_0}$, $\mathbf{T_1}$, **Aut**, **Mon**, **Lin** (explicații suplimentare ... a se vedea cartea mea „scrisă”, practic întregul **Capitol 1**)
- **Teoremă**. O mulţime M $\subseteq$ **FB** este închisă dacă şi numai dacă ea conţine funcţiile elementare şi orice superpoziţie de funcţii din M se află (tot) în M. (**d** - simplă)
- **Definiţie**. O mulţime de funcţii booleene este ***completă*** dacă închiderea sa coincide cu **FB**. O mulţime completă se numeşte ***bază*** dacă este ***minimală*** (în sensul că nici o submulţime proprie a sa nu mai este completă).

- Să furnizăm (fără demonstrații) câteva rezultate importante (împreună cu anumite **<u>c</u>**)
- **Teoremă (<u>E. L. Post</u>)**. O mulţime de funcţii booleene este completă dacă şi numai dacă nu este inclusă în nici una dintre mulţimile $\mathbf{T_0}$**, **$\mathbf{T_1}$**, Aut, Mon, Lin**.
- **Teoremă**. Orice bază conţine cel mult patru funcţii şi există baze compuse din una, două, trei şi patru funcţii.
- **Teoremă**. Orice mulţime închisă de funcţii booleene fie este inclusă cel puţin în una dintre mulţimile $\mathbf{T_0}$**, **$\mathbf{T_1}$**, Aut, Mon, Lin**, fie coincide cu **FB**.
- **Teoremă**. Mulţimile $\mathbf{T_0}$**, **$\mathbf{T_1}$**, Aut, Mon, Lin** sunt ***precomplete*** (o mulţime M de funcţii booleene este precompletă dacă pentru orice altă funcţie f $\notin$ M, mulţimea M U {f} este completă).
- Post a determinat chiar **toate** mulţimile închise de funcţii booleene încă din anul 1941, precum şi relaţiile de incluziune între ele şi anumite baze (**<u>e</u>**)

- Pentru <u>finalul</u> acestor suplimente, se pot mai spune anumite lucruri noi și despre diagramele de decizie binare (ordonate sau nu), care sunt – după cum știm deja – o altă modalitate (grafică/ graf/ arbore ...) de a reprezenta funcțiile booleene
- Putem, de fapt, profita în diverse moduri de această reprezentare: traducerea într-un limbaj „vizual”/ simplificat a unor algoritmi care privesc circuitele logice/ partea de hard a calculatoarelor; dezvoltarea unor noi algoritmi privind anumite clase de funcții booleene, privite ca obiecte complexe, netextuale, etc. (<u>vezi și **H/ R**</u>)

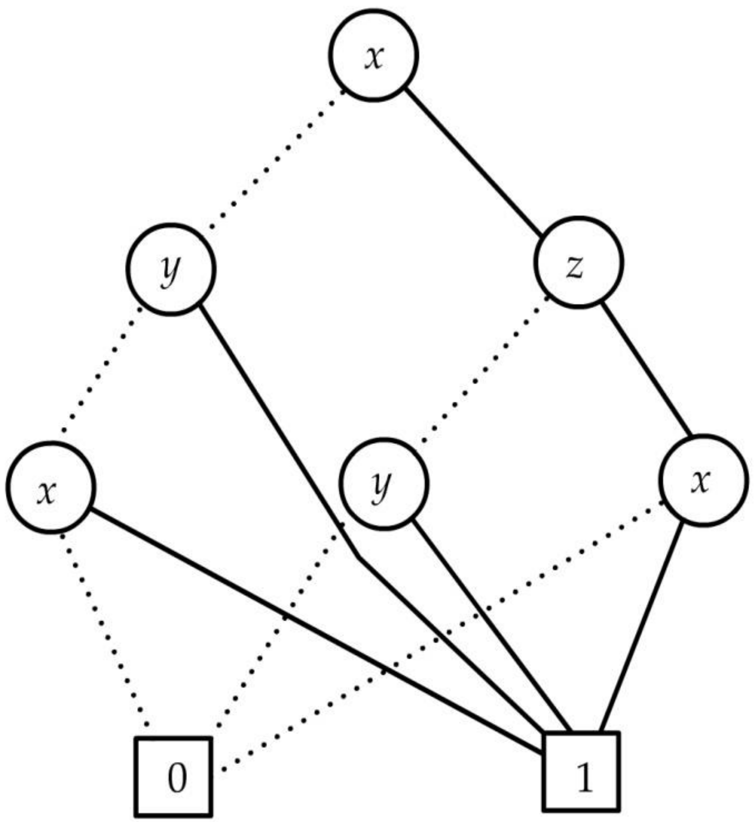
10

- **Definiţie.** Fie o **BDD**, D, peste mulţimea de variabile X, care calculează o funcţie f $\in$ **FB**. Se numeşte ***drum consistent pentru*** f ***în*** D, un drum (d) de la rădăcină la o frunză care satisface condiţia că pentru fiecare x $\in$ X, (d) conţine doar arce reprezentate ca linii punctate sau doar arce reprezentate ca linii continue, care ies din fiecare nod etichetat cu x (acest lucru echivalează cu a stipula că pentru a afla valoarea de adevăr a lui f într-o asignare dată, unei variabile x nu i se pot atribui simultan valorile **0** şi **1**; absolut normal). Atunci, vom spune că f este ***satisfiabilă*** dacă există o **BDD** D care o reprezintă, precum şi un drum consistent pentru ea în D astfel încât acesta „leagă” rădăcina de o frunză etichetată cu **1** (similar cu cazul **LP** se pot defini în acest context noţiunile de formulă ***validă***/ ***contradicţie***).

11

- **Definiţie.** Fie $X = \{x_1, x_2, \ldots, x_n\}$ o mulţime de variabile considerată a fi deja total (strict) ordonată (X este de fapt lista $\langle x_1, x_2, \ldots, x_n\rangle$) şi o **BDD**, D, peste X. Spunem că D ***are ordinea*** $\langle x_1, x_2, \ldots, x_n\rangle$ dacă şi numai dacă pentru fiecare drum (d) în D de la rădăcină la orice frunză şi fiecare apariţie a etichetelor distincte x_i şi x_j pe noduri din (d), dacă x_i apare înaintea lui x_j atunci $i < j$. O **BDD** D se numeşte ***ordonată*** (pe scurt, **OBDD**), dacă există o listă de variabile X (inclusiv lista vidă sau cea care conţine un unic element) astfel încât D are ordinea X.
- Deşi esenţiale pentru testarea satisfiabilităţii unei formule date (a cărei semantică este dată de funcţia din **FB** reprezentată ca **(O)BDD**) sunt doar variabilele care apar într-o **BDD** care o calculează, să notăm că nu am cerut în mod explicit ca lista să conţină *exact* etichetele care apar într-o **OBDD**

- Astfel, dacă un **OBDD** are ordinea $\langle$x, y, z$\rangle$ atunci ea are şi ordinea $\langle$u, x, y, v, z, w$\rangle$ etc. Deoarece am presupus că ordinea este totală şi strictă, relaţia respectivă „$<$” nu este reflexivă, dar este antisimetrică (în sensul că dacă x $<$ y atunci nu putem avea şi y $<$ x) şi tranzitivă. Datorită acestor proprietăţi, într-o **OBDD** nu există apariţii multiple ale unei variabile pe un drum şi este clar că există măcar o **BDD** care nu este şi o **OBDD**:



13

- Cu toate aceste/ unele (posibile) avantaje, se pare totuşi că reprezentarea cu ajutorul **(O)BDD** a funcţiilor booleene nu este încă „convingătoare”
- Nu sunt astfel „vizibili” algoritmi simpli pentru a testa echivalenţa semantică a două **(O)BDD** deja reduse dar diferite (ca în cazul tabelelor de adevăr) şi nici pentru a le „compune” uşor (ca în cazul formelor normale din **LP**)
- În **H/ R** sunt prezentate câteva exemple de **OBDD**-uri nereduse cât şi una maximal redusă, suficient de „sofisticate”
- **Definiţie.** Fie **o1** şi **o2** două ordini (totale, stricte) peste mulţimile de variabile X şi respectiv Y (notăm Z = X U Y). **o1** şi **o2** se numesc ***compatibile*** dacă nu există două variabilele distincte x, y $\in$ Z astfel încât x precede y în **o1** și y precede x în **o2**.

- Restrângând clasa ordinilor posibile la ordinile compatibile, obţinem (aproape imediat) adevărul următoarei afirmaţii
- **Teoremă**. Fie f $\in$ **FB** orice funcţie booleană. Atunci **OBDD**-ul maximal redus care o reprezintă este ***unic via ordinile compatibile***. Mai exact, fie D şi D' două **OBDD**-uri maximal reduse care reprezintă (aceleași) f (***semantic echivalente***). Atunci D şi D' coincid sintactic (ca grafuri).
- Din teorema precedentă rezultă că *verificarea echivalenţei semantice devine banală* în acest context (într-o anumită implementare ar trebui să *verificăm* pur şi simplu *egalitatea a doi pointeri*)
- Tot de aici rezultă şi faptul că *indiferent de ordinea în care aplicăm reducerile* vom obţine *aceeaşi diagramă maximal redusă*

- **Definiţie**. Fie orice n $\in$ **N**, orice f $\in$ **FB**$^{(n)}$ şi orice „mulţime” de variabile/ nume, total şi strict ordonată, notată $X = \langle x_1, x_2, \dots , x_n\rangle$. Fie D o **OBDD** peste X, care are ordinea **o** (exprimată chiar prin scrierea lui X), este maximal redusă şi, în plus, reprezintă f. Atunci D poate fi numită (**OBDD**-)***forma canonică pentru/ a lui*** f.
- Am putea trage concluzia că dimensiunea unei **OBDD** este independentă de ordinea aleasă
- Din păcate acest lucru nu este valabil în general şi dependenţa dimensiunii unei **OBDD** de ordinea aleasă este preţul pe care îl plătim pentru avantajele pe care **OBDD**-rile le au faţă de **BDD**-uri (şi chiar asupra altor tipuri de reprezentări)

16

- În concluzie, chiar dacă nu trebuie să supraestimăm importanţa (**O**)**BDD**-urilor şi a existenţei reprezentărilor canonice (similare cu cele numite „normale”) pentru funcţiile booleene „text”, putem enumera următoarele avantaje ale utilizării acestora:

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

-Formele canonice se pot construi efectiv şi în mod unic pornind de la alte reprezentări (şi reciproc !)

-Nu conţin apariţii nenecesare de variabile/ nume; dacă valoarea lui $f \in \mathbf{FB}^{(n)}$ în $\langle x_1, x_2, \ldots, x_n \rangle$ nu depinde de un x_i atunci forma canonică care reprezintă f nu va conţine nici un x_i-nod (nod etichetat cu x_i)

-Dacă două funcţii f, g $\in$ **FB**$^{(n)}$ sunt reprezentate de **Df** respectiv **Dg**, **OBDD**-uri canonice cu ordini compatibile, atunci se poate decide simplu echivalenţa semantică a lui **Df** şi **Dg** adică, de fapt, egalitatea f = g

-Putem testa dacă o funcţie este satisfiabilă „lucrând” pe forma sa canonică, adică: forma canonică ***nu*** este **D0**; validitatea/ contradicţia este la fel de simplu de testat, adică: forma canonică a funcţiei ***coincide*** cu **D1** (sau cu **D0**)

-Putem testa dacă f „implică” g (f, g $\in$ **FB**$^{(n)}$), adică dacă pentru fiecare $\langle a_1, a_2, \ldots, a_n \rangle \in \mathbf{B}^n$, din $f(a_1, a_2, \ldots, a_n) = \mathbf{1}$ rezultă $g(a_1, a_2, \ldots, a_n) = \mathbf{1}$. Asta înseamnă să calculăm forma canonică pentru $f \cdot \overline{g}$ şi aceasta trebuie să coincidă cu **D0** în cazul în care implicaţia este adevărată

-Şi în cazul acestei reprezentări se pot defini, prin algoritmi eficienţi, anumite operaţii asupra **OBDD**-urilor (formelor canonice) prin care putem „construi” întreaga clasă a funcţiilor booleene (şi nu numai), pornind cu anumite funcţii de bază (elementare)

SUPORT/ suplimente pentru Cursul 6/7 (20 sl)

- Vom începe cu animite completări legate de teorii logice, sisteme deductive, teoreme de completitudine și corectitudine etc.
- 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
- Folosirea metodelor sintactice, deși la fel de „complexe” au avantajul că pot fi totuşi *automatizate* (măcar parţial)

- În acest context, se poate pune problema ***axiomatizării teoriilor logice, cu ajutorul sistemelor de demonstraţie***
- Acest lucru înseamnă că având dată o teorie logică **TE** $\subseteq$ **FORM** (de exemplu, mulţimea $\mathcal{Val}$(**LP**), a formulelor valide din **LP**), 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' =** $\langle \mathcal{A}', \mathcal{R} \rangle$**)** astfel încât **TE =** $\mathcal{Th}$**(SD')**

3

- În acest caz, se impune de obicei ca $\mathcal{A}'$ să fie măcar o mulţime satisfiabilă (adică 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 una de ***structură***)

4

- 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 $<\mathcal{A}',\mathcal{R}>$)

5

- Putem considera şi situaţia inversă, în care avem dat un sistem **SD'** = $\langle \mathcal{A}', \mathcal{R} \rangle$ şi dorim să ne convingem că $\mathcal{Th}$(**SD'**) este într-adevăr o teorie logică; mai mult, să știm dacă $\mathcal{Cs}(\mathcal{A}')$ **=** $\mathcal{Th}$(**SD'**)
- **Definiţie**. Un *sistem de demonstraţie* **SD'** = $\langle \mathcal{A}', \mathcal{R} \rangle$ se numeşte ***corect şi complet*** *pentru o teorie* **TE** dacă **TE** = $\mathcal{Th}$(**SD'**) = $\mathcal{Cs}(\mathcal{A}')$ şi $\mathcal{A}' \subseteq$ **TE**. O *teorie* **TE** este *axiomatizabilă* dacă există un sistem deductiv

  SD' = $\langle \mathcal{A}', \mathcal{R} \rangle$ corect şi complet pentru ea, adică satisfăcând condiţiile anterioare.
- Dacă **SD'** este finit (numărul de scheme ...) specificabil/ axiomatizabil, atunci teoria corespunzătoare se numeşte ***finit axiomatizabilă***

- 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 formal noţiunea de *adevăr*), un sistem deductiv **SD’** = $\langle \mathcal{A}', \mathcal{R} \rangle$ în **FORM**, cu $\mathcal{A}' = \mathcal{A} \cup \mathcal{I}$ ($\mathcal{A}$ 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:

$$\mathcal{T}h(\mathbf{SD'}) = \mathcal{C}s(\mathcal{A}').$$

- **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 **TE** să fie (eventual, chiar finit) axiomatizabilă

8

- În cazul în care este vorba de o teorie formată doar din formule valide (lipsind $\mathcal{I}$), 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** = $\langle \mathcal{A}, \mathcal{R} \rangle$, 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**).

- 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 ddacă $\mathcal{J} \vdash_{\mathbf{SD3}}$ F.
- **Teoremă**. Fie orice F $\in$ **LP1**. Atunci: $\vdash_{\mathbf{SD1}}$ F dacă şi numai dacă $\models$ F (aici mai sunt necesare nişte precizări ...).

10

- Ca o concluzie, d.p.d.v. practic avem nevoie de teorii care să conţină măcar „aritmetica”, „egalitatea” şi axiomele logice „primare” …; acestea, deși par simple” sunt însă ***incomplete*** (***nu tot ceea este adevărat este demonstrabil*** ...), din păcate
- Există și o teoremă de incompletitudine a lui K. Gödel pentru **LP1** (legată în primul rând de „apariția” explicită a simbolului „=” …)
- Continuăm suplimentele cu o altă formă de prezentare a sistemului **SD0** (diferită de deducția naturală)

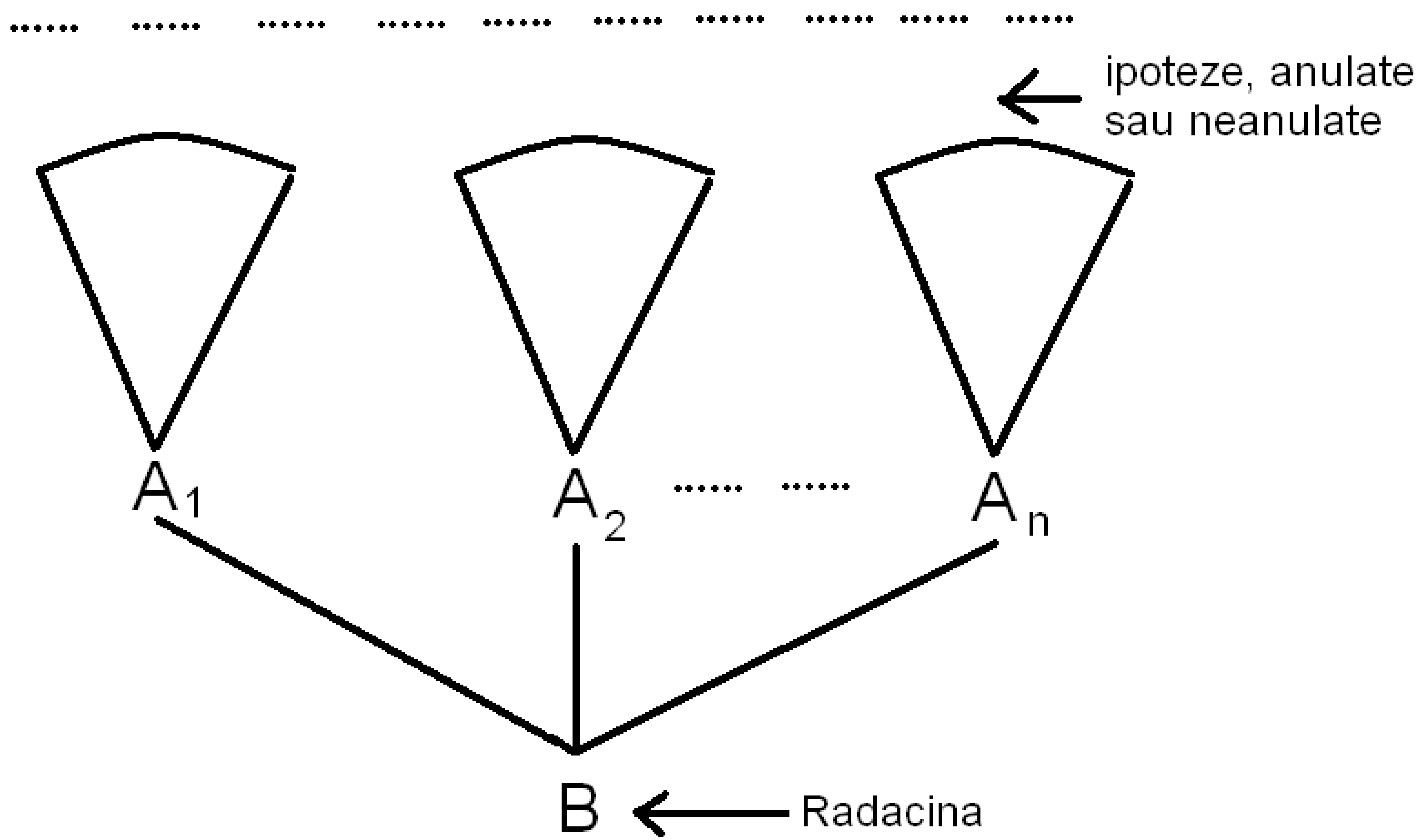
11

- Clasa **FORM** este, în cele ce urmează, **LP**
- Alfabetul peste care sunt construite formulele conține în acest caz doar conectorii $\rceil$ (notat uneori și prin $\neg$) și $\wedge$
- Regulile de inferență sunt „mai multe” decât axiomele (system de tip Gentzen-Jaskowski); **SD0 nu are nicio axiomă**
- O demonstrație (în „noua” deducție naturală) 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)

12

- 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)

13



14

- 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)

15

- Acest **SD0** este un *sistem predicativ*, *finit specificat* și *boolean complet* (**c**); dacă introducem direct și conectorii $\vee$ și $\rightarrow$ în alfabet, putem folosi și următoarele *reguli derivate*:
- 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*

- 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**)
- Pot fi făcute generalizările relative la **LP1** …

- **Definiție**. 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 că formula C este consecință semantică și sintactică din mulțimea

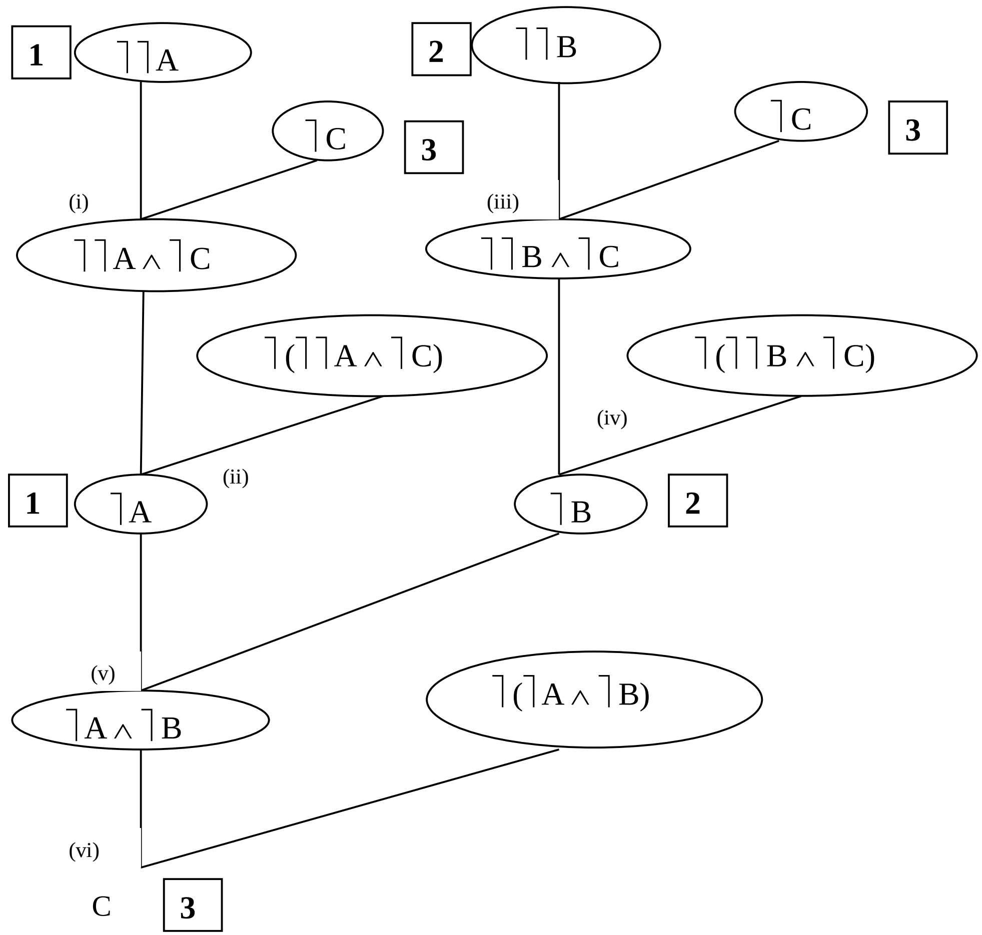
  $\mathcal{I} = \{\rceil(\rceil\rceil A \wedge \rceil C), \rceil(\rceil\rceil B \wedge \rceil C), \rceil(\rceil A \wedge \rceil B)\}$

18

- Desigur că mai sunt ipoteze (frunze) în arbore (și anume $\neg\neg A$, $\neg C$ (de două ori) și $\neg\neg B$), 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

- 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 $\rceil A$ obținut în pasul (ii) după aplicarea (unei instanțe a) regulii (**EN**) (în care B este $\rceil\rceil A \wedge \rceil C$, iar A este $\rceil A$, anulându-se astfel ipoteza $\rceil\rceil A$), dar și ipotezei care se anulează, $\rceil\rceil A$
- Tot ca precizare pentru exemplul care urmează, în pasul notat (i) s-a aplicat (o instanță a) regula (regulii) (**IC**), cu $\rceil\rceil A$ „pe post de A” și cu $\rceil C$ „pe post de B”
- Etc.

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



SUPORT/ suplimente pentru Cursul 7/6 (33 sl)

- Vom <u>începe</u> cu „complemente” legate de deducția naturală, așa cum a fost ea prezentată în curs (după **H/ R**)
- Multe dintre comentarii se referă la explicarea intuitivă a *alegerii* unor reguli, atât d.p.d.v. procedural cât și declarativ
- Când vorbim de „alegere”, ne referim atât la plasarea regulii în sistemul axiomatic, cât și la folosirea ei la un anumit pas al unei demonstrații
- Snt introduc și anumite concept/ rezultate noi
- Exemplele sunt numerotate „în continuare”

- **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

3

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

  1 p *premiză* ?

  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 p *presupunere*

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

- Iar așa ceva s-ar putea interpreta (tot „prin lipsă”) ca fiind ***demonstrația validității secvenței*** $\vdash p \to p$ ($\varnothing \vdash p \to p$), prin aceasta „argumentându-se” faptul că „adevărul” unei formule/ afirmații de genul $p \to p$ 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*)

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

  $\varphi = (q \rightarrow r) \rightarrow ((\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r))$

  box1...

1 $q \rightarrow r$ *presupunere*

  box2...

2 $\neg q \rightarrow \neg p$ *presupunere*

  box3...

3 p *presupunere*

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

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

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

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

  sfbox3...

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

  sfbox2...

9 $(\neg q \rightarrow \neg p) \rightarrow (p \rightarrow r)$ ($\rightarrow$**i**) 2-8

  sfbox1...

10 φ ($\rightarrow$**i**) 1-9

6

- **Observație**. Din exemplul precedent vedem cum o demonstrație pentru validitatea secvenței $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ se poate transforma într-o demonstrație a teoremei $\theta = \varphi_1 \to (\varphi_2 \to (\varphi_3 \to (\ldots (\varphi_n \to \psi)\ldots)))$; 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 $\varphi_n, \varphi_{n-1}, \ldots, \varphi_1$.
- 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)

- 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 **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)

8

- „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

9

- 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 ?

10

- 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

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

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

1	$p \vee q$	*premiză*
	box1..	
2	p	*presupunere*
3	$q \vee p$	($\vee$**i**$_2$) 2
	sfbox1...	
	box2..	
4	q	*presupunere*
5	$q \vee p$	($\vee$**i**$_1$) 4
	sfbox2...	
6	$q \vee p$	($\vee$**e**) 1, 2-3, 4-5

12

- 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

13

- 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”)

14

- 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

- 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 ?

16

- Ideea este desigur cunoscută: într-o secvență, după „⊢”, poate/ trebuie să apară orice afirmație care ar putea fi dedusă (presupunând că premizele dinainte de „⊢” 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”)

- 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

18

- În exemplul anterior am „pus alături” (grafic) dreptunghiurile /demonstrațiile care sunt ipoteze pentru regula ($\vee$**e**) (împreună cu premiza $\rceil p \vee q$); 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 ($\rceil$ **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”*

- **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

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

1 $p \rightarrow \neg p$ *premiză*

box1..

2 p *presupunere*

3 $\neg p$ ($\rightarrow$**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*

21

- Prezentăm mai întâi demonstrația validității secvenței $\varphi \to \psi, \neg\psi \vdash \neg\varphi$
- 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	$\neg\psi$	($\neg$**i**) 3-5

- **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

- Să trecem la (**PBC**): *dacă pornind cu* $\rceil\varphi$ *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 $\rceil\varphi$ și *jos* $\perp$” vom „scrie” (pentru simplitate) că am făcut demonstrația lui $\rceil\varphi \rightarrow \perp$ (de fapt, folosim (**→i**) în mod implicit)
- Astfel, avem demonstrația faptului că (**PBC**) este derivată din (**→i**), ($\rceil$**i**), (**→e**) și ($\rceil\rceil$**e**):

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

- 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 ...

- 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

27

- 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

28

- 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

- 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:

$$
\begin{array}{c}
\cdot \\
\text{premize} \\
\cdot \\
\varphi \to \psi
\end{array}
$$

acum va fi:

.

premize

.

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

φ *presupunere*

ψ

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

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

- **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 (**→e**) (în loc de (**→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 (**→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

- Să notăm că regula ($\urcorner$**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”)

- Regulile „aproape” neatinse (ca explicații **CUM/ CE**) sunt: ($\rightarrow$**e**), ($\neg\neg$**e**), ($\neg$**e**) și ($\perp$**e**)
- ($\rightarrow$**e**) este de fapt (**MP**) și nu mai comentăm
- ($\neg\neg$**e**) a fost comentată puțin, iar „duala” sa ($\neg\neg$**i**) a fost arătată a fi „derivată din ($\neg$**i**) și ($\neg$**e**)”
- Regulile ($\neg$**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*** ($\neg$**i**)