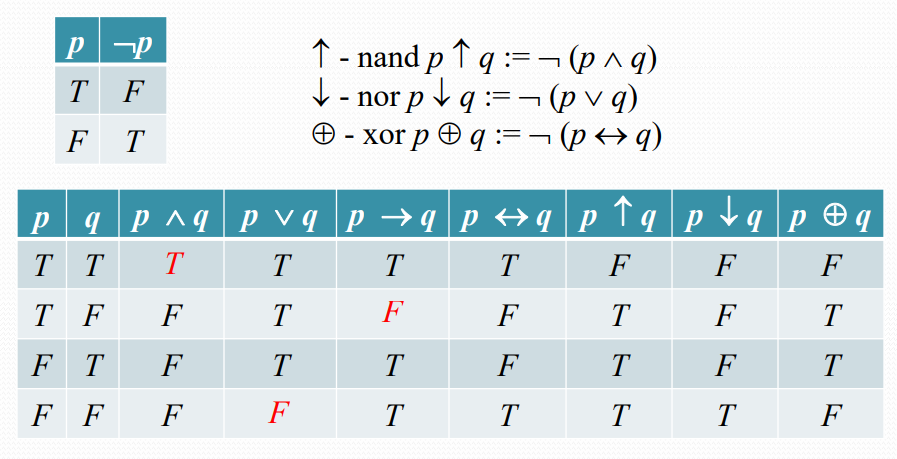
1. Semantica

Propozițiile logice sunt modele ale afirmațiilor propoziționale care sunt fie *adevărate*, fie *false*.  
**Domeniul semantic**: {F(fals), T(true, adevărat} astfel încât ¬F = T și ¬T = F  
**Semantica conectivelor**:



**Interpretare**: O interpretare a formulei U(p1, p2, …, pn) ∈ FP este o funcție i : { p1, p2, …, pn} ⟶ {T, F} care asociează valori de adevăr variabilelor propoziționale și poate fi extinsă la o funcție i : FP ⟶ {T, F} cu proprietatea că funcția i trece peste conective.

**Model**: O interpretare i : {p1, p2, …, pn} ⟶ {T, F} care evaluează formula U ca fiind adevărată, i(U)=T, se numește model al formulei.

**Anti-model**: O interpretare i : { p1, p2, …, pn} ⟶ {T, F} care evaluează formula U ca fiind falsă, i(U)=F, se numeste anti-s al formulei.

**Tautologie**: Formula U se numeste validă (tautologie), notație |=, ⬄ U este evaluată ca fiind adevărată în orice interpretare, adică: ∀ i : { p1, p2, …, pn} ⟶ {T, F}, i(U)=T.

**Formulă consistentă**: Formula U se numește consistentă (realizabilă) ⬄ are cel puțin un model, deci poate fi evaluată ca fiind adevărată: ∃ i : { p1, p2, …, pn} ⟶ {T, F}, astfel încât i(U)=T.

**Formulă inconsistentă**: Formula U se numeste inconsistentă (nerealizabilă) ⬄ U nu are niciun model, adica U este interpretata totdeauna ca falsă: ∀ i : { p1, p2, …, pn} ⟶ {T, F}, i(U)=F.

**Formula contingentă**: Formula U se numește contingentă ⬄ este consistentă, dar nu este validă.

**Consecință logică**: Formula V este consecință logică a formulei U, notație U |= V, ⬄ ∀ i : FP ⟶ {T, F} astfel încât dacă i(U)=T are loc i(V)=T.

**Echivalența logică**. Formulele U, V(p1, p2, …, pn) ∈ FP sunt logic echivalente, notație U ≡ V ⬄ tablele lor de adevăr sunt identice, adică ∀ i : FP ⟶ {T, F}, i(U) = i(V).

**Concepte semantice pentru mulțimi de formule**.  
 **Mulțime consistentă**. O mulțime {U1, U2, …, Un} de formule se numește consistentă (realizabilă) ⬄ formula U1 ∧ U2 ∧ … ∧ Un este consistentă, interpretarea i se va numi model al mulțimii.  
 **Mulțime inconsistentă**. O mulțime {U1, U2, …, Un} de formule se numeste inconsistentă (nerealizabilă, contradictorie) ⬄ formula U1 ∧ U2 ∧ … ∧ Un este inconsistentă, interpretarea i se va numi anti-model al mulțimii.  
 **Consecință logică**. Formula V se numește consecință logică a mulțimii de formule {U1, U2, …, Un} ⬄ U1 ∧ U2 ∧ … ∧ Un |= V. Formulele U1, U2, …, Un se numesc premize, ipoteze, fapte, iar V se numește concluzie.

**Teoremă**.  
- |= U ⬄ ¬U inconsistentă  
- U |= V ⬄ |= U → V ⬄ {U, ¬V} inconsistentă  
- U ≡ V ⬄ |= U ↔ V  
- {U1, U2, …, Un} |= V ⬄ |= U1 ∧ U2 ∧ … ∧ Un → V ⬄ {U1, U2, …, Un, ¬V} inconsistentă

**1.1. Echivalențe logice în logica propozițională**.

* **Legile lui DeMorgan**: ¬(U ∧ V) ≡ ¬U ∨ ¬V ; ¬(U ∨ V) ≡ ¬U ∧ ¬V
* Legile de absorbție: U ∧ (U ∨ V) ≡ U ; U ∨ (U ∧ V) ≡ U
* Legile de comutativitate: U ∧ V ≡ V ∧ U ; U ∨ V ≡ V ∨ U
* Legile de asociativitate: U ∧ (V ∧ Z) ≡ (U ∧ V) ∧ Z ; U ∨ (V ∨ Z) ≡ (U ∨ V) ∨ Z
* **Legile de distributivitate**: U ∧ (V ∨ Z) ≡ (U ∧ V) ∨ (U ∧ Z)  
   U ∨ (V ∧ Z) ≡ (U ∨ V) ∧ (U ∨ Z)
* Legile de idempotență: U ∨ U ≡ U ; U ∧ U ≡ U

**Definirea conectivelor derivate**

* **U → V ≡ ¬U ∨ V**
* U ↔ V ≡ (U → V) ∧ (V → U)
* U ↑ V ≡ ¬(U ∧ V)
* U ↓ V ≡ ¬(U ∨ V)
* U ⊕ V ≡ ¬(U ↔ V)

**1.2. Principiul dualității**.

Pentru orice echivalență logică U ≡ V care conține doar conectivele ¬, ∧, ∨, ↑, ↓ există o altă echivalență logică U’ ≡ V’, unde U’ și V’ sunt formule obținute din U și V prin interschimbarea conectivelor logice duale: (∧, ∨), (↑, ↓) și a valorilor de adevăr duale: (T, F). Concepte duale: tautologie, formula inconsistentă.

**1.3. Forme normale în logica propozițiilor**.

**Literal**: Un literal este o variabilă propozițională sau negația sa. (p, ¬p)

**Clauză**: O clauză este disjuncția unui număr finit de literali. (p ∨ ¬q ∨ r)

**Cub**: Un cub este conjuncția unui număr finit de literali. (p ∧ ¬q ∧ ¬r)

**Clauza vidă**: (simbolizată prin ☐) este clauza fără literali, fiind singura inconsistentă.

**Forma normală disjunctivă**: O formulă este în FND dacă aceasta se scrie ca o disjuncție de cuburi.

**Forma normala conjunctivă**: O formulă este în FNC dacă aceasta se scrie ca o conjuncție de clauze.

**Algoritmul de normalizare**.   
**Pas1**. Înlocuirea formelor U ↔ V cu formele echivalente [(U → V) ∧ (V → U)]. Înlocuirea formelor U → V cu formele echivalente [¬U ∨ V].  
**Pas2**. Aplicarea legilor lui DeMorgan.  
**Pas3**. Aplicarea legilor distributivității.  
**Pas4**. Simplificarea formei obținute folosind alte echivalențe logice.

**Teoremă**.  
- O formulă în FNC este tautologie ⬄ toate clauzele sunt valide.  
- O formulă în FND este inconsistentă ⬄ toate cuburile sunt inconsistente.

**2. Sintaxa**.

**Axiome**. AP = {A1, A2, A3} scheme axiomatice  
- A1: U → (V → U)  
- A2: (U → (V → Z)) → ((U → V) → (U → Z))  
- A3: (U → V) → (¬V → ¬U)

**Reguli de inferență**. RP = {mP} o singură regulă de inferență “*modus ponens*”  
- mP: U, U → V |- V : “din faptele U și U → V se deduce V”

**2.1. Deducția**.

Fie formulele U1, U2, …, Un numite ipoteze și formula V formulă propozițională. Spunem că V e detuctibilă din U1, U2, …, Un și notăm U1, U2, …, Un |- V dacă există o secvență de formule (f1, f2, ..., fm) astfel încât fm = V și ∀ i ∈ {1, 2, …, m} avem:  
- fi ∈ AP- fi ∈ {U1, U2, …, Un}  
- fi, fk fi , j < i și k < i  
 Secvența (f1, f2, ..., fm) se numește deducția lui V din U1, U2, ..., Un .

**Teoremă**. O formulă U ∈ FP, astfel încât Ø |- U ( |- U ) se numește **teoremă**. Teoremele sunt formule deductibile doar din axiome și folosind regula modus ponens.

**Teorema de deducție**. Dacă U1, U2, ..., Un |- V, atunci U1, U2, …, Un-1 |- Un → V

**Inversa teoremei de deducție**. Dacă U1, U2, ..., Un-1 |- Un → V, atunci U1, U2, ..., Un |- V

**Consecințele teoremei de deducție**.  
○ |- U → ((U → V) → V)  
○ |- (U → V) → ((V → Z) → (U → Z)) – legea silogismului  
○ |- (U → (V → Z)) → (V → (U → Z)) – legea permutării premizelor  
○ |- (U → (V → Z)) → (U ∧ V → Z) – legea reuniunii premizelor  
○ |- (U ∧ V → Z) → (U → (V → Z)) – legea separării premizelor

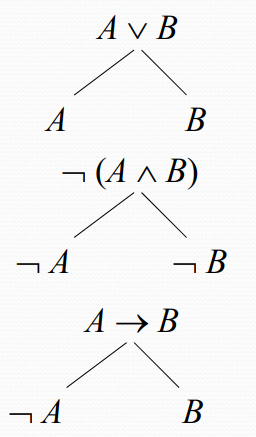
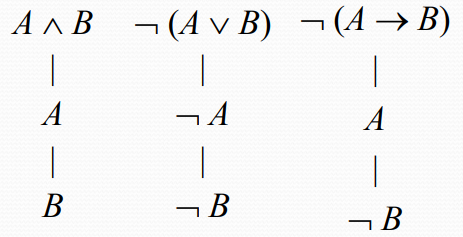
**2.2. Proprietățile logicii propozițiilor**.  
 **Teorema de corectitudine**. Dacă |- U, atunci |= U. (Validitate sintactică implică validitate semantică).  
 **Teorema de completitudine**. Dacă |= U, atunci |- U. (Validitate semantică implică validitate sintactică).  
 **Teorema de corectitudine și de completitudine**. |- U dacă și numai dacă |= U.

**Consecințe**.  
1) Logica propozițiilor este **necontradictorie**. (nu pot avea loc simultan |- U și |- **¬**U).  
2) Logica propozițiilor este **necoerentă**. (nu orice formulă propozițională este teoremă).  
3) Logica propozițiilor este **decidabilă**. (se poate decide dacă o formulă propozițională este teoremă sau nu).

**3. Metoda tabelelor semantice**.

\*idee: descompunerea formulei inițiale în subformule până la nivel de literali

**Clase de formule**.  
- clasa α – formule de tip conjunctiv  
 A ∧ B | ¬(A ∨ B) | ¬(A → B)  
- clasa β – formulă de tip disjunctiv  
 A ∨ B | ¬(A ∧ B) | A → B

 **Reguli de descompunere a formulelor**.  
 regula α regula β

**Arborele binar de descompunere a unei formule**.  
 Având o formulă U, ei ii se poate asocia o tabelă semantică, care este de fapt un arbore binar ce conține în nodurile sale formule și se construiește astfel:  
 - rădăcina arborelui este etichetată cu formula lui U  
 - fiecare ramură a arborelui care conține o formulă va fi extinsă cu subarborele corespunzător regulii de descompunere care se aplică formulei  
 - extinderea unei ramuri se încheie în două situații:  
 a) dacă pe o ramură apare formula și negația sa  
 b) dacă au fost descompuse toate formulele pe acea ramură

**Tipuri de ramuri**.  
 **Ramură închisă**. O ramură a tabelei este închisă, simbolizată prin ⊗, dacă ea conține o formulă și negația ei.  
 **Ramură deschisă**. O ramură a tabelei este deschisă, simbolizată prin ⨀, dacă ea nu este închisă.  
 **Ramură completă**. O ramură a tabelei este completă dacă ea este fie închisă, fie toate formulele de pe acea ramură au fost descompuse.

**Tipuri de tabele semantice**.  
 **Tabelă închisă**. O tabelă este închisă dacă toate ramurile sale sunt închise.  
 **Tabelă deschisă**. O tabelă este deschisă daca cel puțin o ramură a sa este deschisă.  
 **Tabelă completă**. O tabelă este completă daca toate ramurile sale sunt complete.

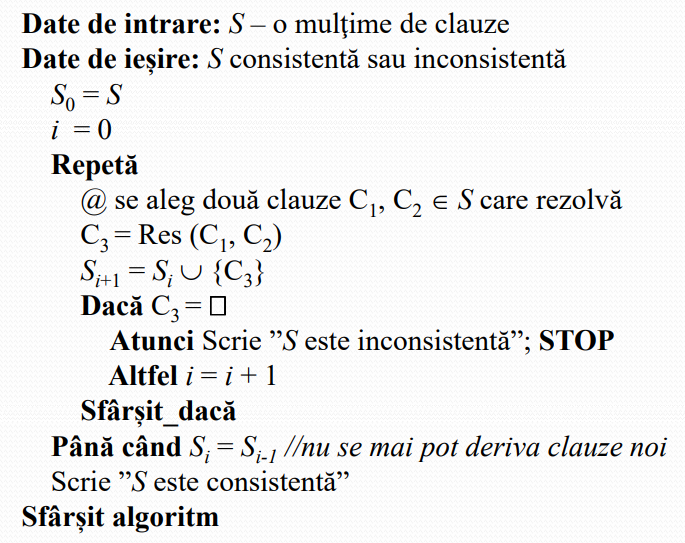
**Teorema de corectitudine și de completitudine a metodei tabelelor semantice**.  
 O formulă U este teoremă (tautologie) ⬄ există o tabelă semantică închisă pentru formula ¬U.

**Teoremă**. U1, U2, …, Un |- Y (echivalent cu U1, U2, …, Un |- Y) ⬄ există o tabelă semantică închisă pentru U1 ∧ U2 ∧ … ∧ Un ∧ ¬Y.

**4. Metoda rezoluției**.  
- demonstrare sintactică, prin respingere

**Terminologie**.  
- clauzele C1 = A ∨ l și C2 = B ∨ ¬l rezolvă deoarece conțin 2 literali opuși (literali complementari)  
- notație: C3 = Resl(C1, C2)  
- C3 se numeste **rezolventul** clauzelor C1 și C2  
- clauzele C1 și C2 se numesc **clauze părinte**  
- caz particular: C1 = l, C2 = ¬l, Resl(C1, C2) = ☐

**Algoritmul rezoluției propoziționale**:

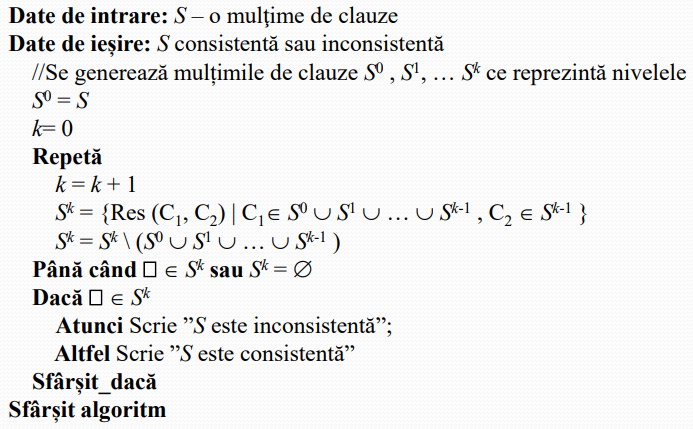
  
 **Notație**: S |-res ☐ “din mulțimea S de clauze s-a derivat clauza vidă în urma aplicării algoritmului rezoluției propoziționale.”

**Teorema de corectitudine și de completitudine**  
 Mulțimea S este inconsistentă ⬄ S |-res ☐.

**Teoreme**.  
1) U tautologie ⬄ FNC(¬U) |-res ☐  
2) U1, U2, ..., Un |= V ⬄ U1, U2, ..., Un |- V ⬄ FNC(U1 ∧ U2 ∧ … ∧ Un ∧ ¬V) |-res ☐.

**Automatizarea procesului rezolutiv**- prin intermediul unor strategii:  
 - asigură exploatarea tuturor modurilor posibile de derivare a clauzei vide  
 - evitarea deducerii unor clauze redundante sau irelevante pentru obținerea ☐

**4.1. Strategia eliminării**.  
 O mulțime S de clauze poate fi simplificată, păstrânt consistența / inconsistența ei prin aplicarea următoarelor transformări:  
 1) **Eliminarea clauzelor tautologie** (nu pot contribui la derivarea clauzei vide) (ex: ¬p ∨ q ∨ p ∨ ¬r).  
 2) **Eliminarea clauzelor subsumate de alte clauze din S**: clauza C1 este subsumată de C2 dacă există o clauză C3 astfel încât C1 = C2 ∨ C3 (ex: ¬p ∨ q ∨ r este subsumată de ¬p ∨ q).  
 3) **Eliminarea clauzelor care conțin literali puri în S**: Un literal este pur dacă negația sa nu apare în nicio clauză din S (ex: {¬p ∨ q, **¬p ∨ r**, p, **¬q ∨ r**)  
 4) Dacă C = l este o **clauză unitate** din S, **se șterg toate clauzele care-l conțin pe l și pe ¬l din clauzele rămase**.

**4.2. Strategia saturării pe nivele** (algoritm)  


**4.3. Strategia mulțimii suport** Se evită aplicarea regulii de rezoluție asupra unor clauze dintr-o submulțime consistentă a mulțimii inițiale de clauze, deoarece rezolvenții obținuți sunt irelevanți în procesul de derivare a ☐.  
 Fie S o mulțime de clauze. O submulțime Y a lui S se numește **mulțime suport** a lui S dacă S / Y este consistentă.  
 **Rezoluția mulțimii suport** este rezoluția a două clauze ce nu aparțin ambele lui S / Y.

**5. Rafinările rezoluției**.  
- impun restricții asupra clauzelor care rezolvă, pentru a eficientiza procesul rezolutiv  
**Notație**: S ☐ din mulțimea S de clauze s-a derivat clauza vidă prin aplicarea strategiei st a rezoluției propoziționale.

**5.1. Rezoluția blocării**.  
- fiecare apariție de literal din mulțimea de clauze este indexat arbitrar cu un întreg  
- restricția: literalii care rezolvă din clauzele părinți trebuie să aibă **cei mai mici indici** din aceste clauze  
- literalii din rezolvenți moștenesc indicii de la clauzele părinți, iar în cazul moștenirii a doi literali identici, se păstrează cel cu indicele mai mic  
- este foarte eficientă și ușor de implementat, se recomandă combinarea ei cu strategia saturării pe nivele  
 **Teorema de corectitudine și de completitudine**.  
**Corectitudine**: Fie S o mulțime de clauze în care fiecare literal este indexat în mod arbitrar cu un întreg. Dacă S este inconsistentă, atunci există o deducție din mulțimea S a clauzei vide prin rezoluția blocării.  
**Completitudine**: Fie S o mulțime de clauze în care fiecare literal este indexat în mod arbitrar cu un întreg. Dacă din S se deduce prin rezoluția blocării clauza vidă, atunci S este inconsistentă.

**5.2. Rezoluția liniară**.  
- procesul rezolutiv este liniar: la fiecare pas una dintre clauzele părinte este rezolventul obținut la pasul anterior  
 **Teorema de corectitudine și de completitudine**.  
Mulțimea S de clauze este inconsitentă ⬄ S ☐.  
- rezoluția liniară furnizează o strategie la nivel de implementare: căutarea cu revenire  
 - la fiecare iterație, pentru clauza centrale pot exista mai multe posibile clauze laterale  
 - după ce au fost utilizate toate posibilele clauze laterale, dar nu s-a obținut clauza vidă se  
 revine la iterația precedent  
 - consistența mulțimii de clauze este demonstrate după o căutare complete fără derivarea  
 clauzei vide

**Cazuri particulare ale rezoluției liniare**.  
1) **Rezoluția unitară** (*unit*): clauzele centrale au cel puțin o clauză părinte unitară (conține un singur literal)  
2) **Rezoluția de intrare** (*input*): clauzele laterale sunt clauze inițiale (de intrare)  
**Teorema de echivalență**. Fie mulțimea S de clauze. S ☐ ⬄ S | ☐.  
**Corectitudinea**. Dacă S | ☐, atunci S este inconsistentă.  
**Incompletitudinea**. Există mulțimi inconsistente de clauze din care nu se poate deriva clauza vidă folosind rezoluția input sau rezoluția unit.

