Logică computațională
Curs 6

Lector dr. Pop Andreea-Diana

Metoda rezoluţiei (Robinson, 1965)

- metodă de demonstrare automată *sintactică*, prin *respingere*
- este o metodă corectă şi completă de demonstrare automată
- verificarea *consistenţei/inconsistenţei* unei mulţimi de clauze (scop)

Sistem formal (axiomatic) asociat Rezoluției propoziționale

- Res = $(\Sigma_{\text{Res}}, F_{\text{Res}}, A_{\text{Res}}, R_{\text{Res}})$
  - $\Sigma_{\text{Res}} = \Sigma_{\text{P}} \setminus \{\wedge, \rightarrow, \leftrightarrow\}$ – **alfabetul**
  - $F_{\text{Res}} \cup \{\square\}$ – **mulţimea formulelor bine-formate**
    - F_{Res} mulţimea tuturor clauzelor ce se pot forma folosind alfabetul Σ_{Res}
    - $\square$ **- clauza vidă** care nu conţine nici un literal, simbolizează inconsistenţa
  - $A_{\text{Res}} = \emptyset$ – **mulțimea axiomelor**
  - $R_{\text{Res}} = \{res\}$ **mulţimea regulilor de inferenţă** care conţine doar
    - **regula rezoluţiei**: $A \vee l, B \vee \neg l \vdash_{res} A \vee B$, unde l este un literal, iar $A, B \in F_{\text{Res}}$

Terminologie

- clauzele $C_1 = A \vee l$, $C_2 = B \vee \neg l$ **rezolvă** deoarece conţin doi literali opuşi (complementari)
- **Notație:** $C_3 = \mathrm{Res}_l\,(C_1, C_2)$
- C_3 **rezolventul** clauzelor C_1 și C_2
- clauzele C_1, C_2 **clauze părinte**
- caz particular: $C_1 = l$, $C_2 = \neg l$, $\mathrm{Res}_l\,(C_1, C_2) = \square$ - inconsistentă

Observație:

- Rezoluţia ca şi regulă de inferenţă este o generalizare a regulilor *modus ponens, modus tollens* şi a *silogismului.*

Algoritmul rezoluției propoziționale:

Date de intrare: S – o mulţime de clauze

Date de ieșire: S consistentă sau inconsistentă

$S_0 = S$

$i = 0$

Repetă

@ se aleg două clauze $C_1, C_2 \in S$ care rezolvă

$C_3 = \text{Res}(C_1, C_2)$

$S_{i+1} = S_i \cup \{C_3\}$

Dacă $C_3 = \square$

Atunci Scrie "S este inconsistentă"; **STOP**

Altfel $i = i + 1$

Sfârșit_dacă

Până când $S_i = S_{i-1}$ *//nu se mai pot deriva clauze noi*

Scrie "S este consistentă"

Sfârșit algoritm

Notație:

- $S \vdash_{Res} \square$ ”din mulțimea S de clauze s-a derivat clauza vidă prin aplicarea algoritmului rezoluției propoziținale”

Teorema de corectitudine și completitudine

- Teorema de corectitudine

  Dacă $S \vdash_{\text{Res}} \square$ atunci S este inconsistentă.

- Teorema de completitudine

  Dacă S este inconsistentă atunci $S \vdash_{\text{Res}} \square$.

- Teorema de corectitudine și completitudine

  Mulțimea S este inconsistentă dacă și numai dacă $S \vdash_{\text{Res}} \square$.

Teoreme

- U este tautologie dacă și numai dacă $FNC(\neg U) \vdash_{\text{Res}} \square$
- $U_1, U_2, \ldots, U_n \models V$ dacă și numai dacă
  $U_1, U_2, \ldots, U_n \vdash V$ dacă și numai dacă
  $FNC(U_1 \wedge U_2 \wedge \ldots \wedge U_n \wedge \neg V) \vdash_{\text{Res}} \square$

$$S_i \overset{\text{not.}}{=\!=} FNC(U_i),\ i = \overline{1,n}$$

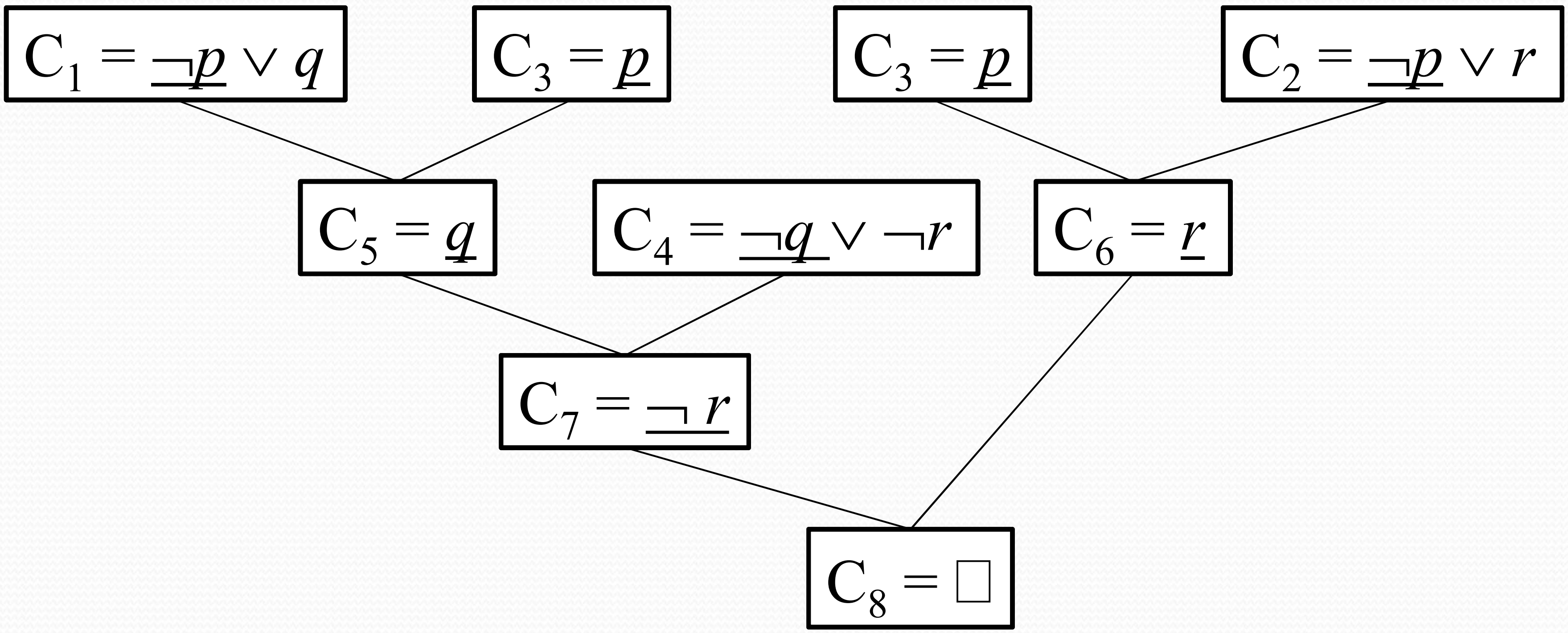
$$S_{n+1} \overset{\text{not.}}{=\!=} FNC(\neg V)$$

$$S_1 \cup S_2 \cup \ldots \cup S_n \cup S_{n+1} \vdash_{\text{Res}} \square$$

Exemplu

$$S = \{\neg p \vee q, \neg p \vee r, p, \neg q \vee \neg r\}$$

$$C_1 \overset{\text{not.}}{=\!=} \neg p \vee q, \quad C_2 \overset{\text{not.}}{=\!=} \neg p \vee r, \quad C_3 \overset{\text{not.}}{=\!=} p, \quad C_4 \overset{\text{not.}}{=\!=} \neg q \vee \neg r$$



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 □

Strategia eliminării

- inspirată din procedura Davis-Putman
- O mulțime S de clauze poate fi simplificată, păstrând consistența/inconsistența ei prin aplicarea următoarelor transformări:
  - **Eliminarea clauzelor tautologice** (nu pot contribui la derivarea clauzei vide): $\cancel{\underline{\neg p} \vee q \vee \underline{p} \vee \neg r}$
  - **Eliminarea clauzelor subsumate de alte clauze din *S***: clauza C_1 este subsumată de C_2 dacă există o clauză C_3 astfel încât $C_1 = C_2 \vee C_3$:

    $\cancel{\underline{\neg p \vee q} \vee r}$ este subsumată de $\neg p \vee q$
  - **Eliminarea clauzelor care conțin literali puri în *S***: Un literal este pur dacă negația sa nu apare în nici o clauză din S:

    $\{\neg p \vee q, \neg \cancel{p \vee \underline{r}}, p, \neg \cancel{q \vee \underline{r}}\}$

- Dacă C=l este o **clauză unitate** din S**, se șterg toate clauzele *care-l conțin pe l* și *¬l din clauzele rămase***.

$\{\cancel{\underline{\neg p}} \vee q, \cancel{\underline{\neg p}} \vee r, \cancel{\underline{p}}, \neg q \vee \neg r, \cancel{p \vee \neg r}\} \Rightarrow \{q, r, \neg q \vee \neg r\}$

$\varnothing$ **(consistentă)** sau $\{\square\}$ **(inconsistentă)**

Strategia saturării pe nivele (algoritmul)

Date de intrare: S – o mulţime de clauze

Date de ieșire: S consistentă sau inconsistentă

//Se generează mulțimile de clauze S^0 , S^1, … S^k ce reprezintă nivelele

$S^0 = S$

$k= 0$

Repetă

$k = k + 1$

$S^k = \{\text{Res}\ (C_1, C_2) \mid C_1 \in S^0 \cup S^1 \cup \ldots \cup S^{k-1} , C_2 \in S^{k-1} \}$

$S^k = S^k \setminus (S^0 \cup S^1 \cup \ldots \cup S^{k-1})$

Până când $\square \in S^k$ **sau** $S^k = \emptyset$

Dacă $\square \in S^k$

Atunci Scrie ”S este inconsistentă”;

Altfel Scrie ”S este consistentă”

Sfârșit_dacă

Sfârșit algoritm

Strategia mulţimii suport

- se *evită* aplicarea regulii de rezoluţie asupra unor clauze dintr-o *submulţime consistentă* a mulţimii iniţiale de clauze, deoarece rezolvenţii obţinuţi sunt *irelevanţi* în procesul de derivare a □
- Această strategie a fost inspirată din faptul următor: în general mulţimea *premizelor* (faptelor) unei deducţii este *consistentă*, deci rezolvarea unor clauze din această mulţime consistentă nu poate duce la derivarea clauzei vide (inconsistenţa)

- **Definiţie:** Fie S o mulţime de clauze. O submulţime Y a lui S se numeşte *mulţime suport* a lui S, dacă $S \setminus Y$ este consistentă. **Rezoluţia mulţimii suport** este rezoluţia a două clauze care nu aparţin ambele mulţimii $S \setminus Y$.