Compararea teoretică și experimentală a algoritmilor pentru rezolvarea problemei satisfiabilității în logica propozițională

Borbiro Paul-Marian
Universitatea de Vest din Timișoara
Email: paul.borbiro99@e-uvt.ro
GitHub: https://github.com/PBorbiro99/MPI_2025

12 mai 2025

Rezumat

Această lucrare prezintă o analiză comparativă a principalelor metode pentru rezolvarea problemei satisfiabilității în logica propozițională (SAT). Studiul se concentrează pe trei algoritmi fundamentali: Rezoluția, algoritmul Davis-Putnam (DP) și algoritmul Davis-Putnam-Logemann-Loveland (DPLL). Pentru fiecare metodă, analizăm proprietățile teoretice, implementarea și performanțele experimentale. Investigăm diverse euristici pentru ghidarea căutării și evaluăm eficiența acestora pe benchmark-uri recunoscute din SATLIB. Rezultatele demonstrează că DPLL cu euristici moderne de tip CDCL (Conflict-Driven Clause Learning) este superior în majoritatea cazurilor, dar celelalte metode pot fi competitive pentru anumite clase specializate de probleme. Evaluarea experimentală utilizează implementări open source de referință ale algoritmilor, iar toate datele și scripturile de experimente sunt disponibile public pentru reproducerea rezultatelor.

Cuvinte cheie: satisfiabilitate, SAT, rezoluție, Davis-Putnam, DPLL, CDCL, euristici.

1 Introducere

Problema satisfiabilității în logica propozițională (SAT) constă în determinarea dacă există o atribuire de valori de adevăr pentru variabilele unei formule logice care face formula adevărată. SAT are o importanță deosebită atât teoretic, fiind prima problemă demonstrată ca fiind NP-completă [3], cât și practic, cu numeroase aplicații în verificarea formală, planificare automată, inteligență artificială și multe altele [2].

Progresele recente în algoritmi și implementări SAT au condus la dezvoltarea unor rezolvatoare extrem de eficiente, capabile să soluționeze instanțe industriale cu milioane de variabile și clauze. Aceste avansuri au făcut ca SAT să devină o tehnologie crucială în diferite domenii, inclusiv verificarea hardware, sinteză de programe și inteligență artificială [9].

În această lucrare, realizăm un studiu comparativ al principalelor metode pentru rezolvarea SAT, cu accent pe trei algoritmi fundamentali:

- **Rezoluția** - o metodă bazată pe inferență logică, utilizând regula de rezoluție pentru a deriva noi clauze.
- **Algoritmul Davis-Putnam (DP)** - o metodă bazată pe eliminarea variabilelor prin aplicarea rezoluției.
- **Algoritmul Davis-Putnam-Logemann-Loveland (DPLL)** - o evoluție a algoritmului DP, utilizând căutarea cu backtracking și propagarea restricțiilor.

Pentru fiecare metodă, analizăm fundamentele teoretice, strategii pentru alegerea pasului următor, detalii de implementare și performanțele pe diverse clase de probleme SAT. Evaluăm de asemenea impactul tehnicilor moderne precum învățarea clauzelor condusă de conflicte (CDCL) asupra eficienței algoritmului DPLL.

2 Fundamente teoretice

2.1 Problema satisfiabilității

Problema SAT constă în determinarea existenței unei interpretări care satisface o formulă logică dată. Formal, având o formulă booleană F definită peste un set de variabile $V = \{x_1, x_2, \ldots, x_n\}$, problema este de a determina dacă există o atribuire de valori $\{0, 1\}$ pentru variabilele din V astfel încât F să fie evaluată la valoarea 1 (adevărat).

2.2 Forme normale conjunctive

În practică, formulele SAT sunt reprezentate în forma normală conjunctivă (CNF): o conjuncție de clauze, unde fiecare clauză este o disjuncție de literali, iar un literal este o variabilă sau negația acesteia.

Formal, o formulă F în CNF are forma:

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_m \tag{1}$$

unde fiecare clauză C_i are forma:

$$C_i = l_{i,1} \vee l_{i,2} \vee \ldots \vee l_{i,k_i} \tag{2}$$

și fiecare literal $l_{i,j}$ este fie o variabilă x_k, fie negația acesteia $\neg x_k$.

Orice formulă booleană poate fi transformată într-o formulă CNF echivalentă din punct de vedere al satisfiabilității folosind transformarea Tseitin [14], care introduce variabile auxiliare pentru a evita explozia exponențială a dimensiunii formulei.

3 Rezoluția

3.1 Principiile metodei de rezoluție

Rezoluția este o regulă de inferență propusă de Robinson [13], care stă la baza multor sisteme de demonstrare automată a teoremelor. Regula de rezoluție se aplică pe două clauze care conțin literali complementari (un literal și negația sa) și produce o nouă clauză, numită rezolventa.

Dacă avem două clauze:

$$C_1 = (l_1 \vee l_2 \vee \ldots \vee l_m \vee x) \tag{3}$$

$$C_2 = (l_{m+1} \vee l_{m+2} \vee \ldots \vee l_n \vee \neg x) \tag{4}$$

atunci rezolventa lor este:

$$C = (l_1 \vee l_2 \vee \ldots \vee l_m \vee l_{m+1} \vee l_{m+2} \vee \ldots \vee l_n) \tag{5}$$

3.2 Algoritmul de rezoluție pentru SAT

Algoritmul de rezoluție pentru SAT funcționează prin aplicarea sistematică a regulii de rezoluție până când se obține clauza vidă (indicând nesatisfiabilitatea) sau până când nu se mai pot genera noi rezolvente (indicând satisfiabilitatea).

Algorithm 1 Rezoluție pentru SAT

```
Input: Formula F în CNF
Output: SAT sau UNSAT
S ← mulțimea clauzelor din F
while clauza vidă □ nu este în S și există clauze care pot fi rezolvate do
    Selectează clauze C1, C2 ∈ S care conțin literali complementari
    Calculează rezolventa C a clauzelor C1 și C2
    if C nu este în S și C nu este o tautologie then
        Adaugă C la S
    end if
end while
if clauza vidă □ este în S then return UNSAT
elsereturn SAT
end if
```

3.3 Strategii pentru rezoluție

Eficiența rezoluției depinde în mare măsură de strategiile utilizate pentru selectarea clauzelor care urmează să fie rezolvate:

- **Rezoluția unitară**: Selectează pentru rezolvare doar perechi în care cel puțin una este o clauză unitară.
- **Rezoluția liniară**: Fiecare rezolventă nou generată este imediat utilizată pentru următoarea aplicare a rezoluției.

- **Rezoluția ordonată**: Aplică rezoluția doar pe perechile de clauze care respectă o anumită ordine a variabilelor.
- **Set-of-support**: Începe cu un subset de clauze "suport" și aplică rezoluția doar când cel puțin una din clauze provine din acest set.

Studii recente demonstrează că, deși în cel mai rău caz rezoluția are complexitate exponențială, anumite variante optimizate ale rezoluției pot fi eficiente pentru clase specifice de probleme, în special probleme structurate din verificarea formală [16].

4 Algoritmul Davis-Putnam (DP)

4.1 Principiile algoritmului Davis-Putnam

Algoritmul Davis-Putnam (DP), propus în 1960 [4], este o metodă de eliminare a variabilelor pentru rezolvarea SAT. Pentru fiecare variabilă x, algoritmul identifică toate clauzele care conțin x și toate clauzele care conțin $\neg x$. Pentru fiecare pereche de clauze (C_1, C_2), unde C_1 conține x și C_2 conține $\neg x$, se calculează rezolventa și se adaugă la formulă. După aceea, toate clauzele care conțin x sau $\neg x$ sunt eliminate.

4.2 Algoritmul DP pentru SAT

Algorithm 2 Algoritmul Davis-Putnam

Input: Formula F în CNF
Output: SAT sau UNSAT
while F conține variabile **do**
    Aplică propagarea unităților și eliminarea literalilor puri
    if F conține clauza vidă **then return** UNSAT
    end if
    if F este vidă **then return** SAT
    end if
    $x \leftarrow$ Selectează Variabilă(F)
    $P_x \leftarrow$ clauzele din F care conțin literalul x
    $N_x \leftarrow$ clauzele din F care conțin literalul $\neg x$
    $R \leftarrow \emptyset$
    for fiecare $C_1 \in P_x$ și $C_2 \in N_x$ **do**
        $C \leftarrow$ rezolventa lui C_1 și C_2 pe variabila x
        if C nu este o tautologie **then**
            $R \leftarrow R \cup \{C\}$
        end if
    end for
    $F \leftarrow (F \setminus (P_x \cup N_x)) \cup R$
end while
if F conține clauza vidă **then return** UNSAT
elsereturn SAT
end if

4.3 Strategii pentru algoritmul DP

Ordinea în care variabilele sunt eliminate afectează dramatic numărul de clauze generate. Cercetări recente în domeniul SAT [7] au propus strategii eficiente pentru selecția variabilelor:

- **Minimul produsului**: Selectează variabila care minimizează produsul numărului de clauze pozitive și negative, reducând astfel numărul maxim de rezolvente generate.
- **Minimul sumei**: Selectează variabila care apare în cel mai mic număr total de clauze.
- **Minimul max-ului**: Selectează variabila care minimizează maximul dintre numărul de clauze pozitive și negative.

Deși algoritmul DP original nu este competitiv pentru probleme generale SAT, tehnicile de eliminare a variabilelor bazate pe principiile DP sunt extrem de eficiente în faza de preprocesare a rezolvatoarelor SAT moderne [7].

5 Algoritmul Davis-Putnam-Logemann-Loveland (DPLL)

5.1 Principiile algoritmului DPLL

Algoritmul DPLL [5] utilizează o abordare bazată pe căutare cu backtracking, combinată cu propagarea constrângerilor. Ideea fundamentală este de a atribui valori variabilelor una câte una, propagând efectele acestor atribuiri și verificând dacă se ajunge la un conflict.

DPLL utilizează două reguli de simplificare importante:

- **Propagarea unităților**: Dacă o clauză conține un singur literal, acea variabilă trebuie să primească valoarea care face literalul adevărat.
- **Eliminarea literalilor puri**: Dacă o variabilă apare doar cu o singură polaritate, aceasta poate fi setată la valoarea care face literalii corespunzători adevărați.

5.2 Algoritmul DPLL pentru SAT

Algorithm 3 Algoritmul DPLL

function DPLL(F)
  $F \leftarrow$ PropagăUnitățiȘiLiteraliPuri(F)
  if F conține clauza vidă **then return** UNSAT
  end if
  if F este vidă **then return** SAT
  end if
  $x \leftarrow$ Selectează Variabilă(F)
  if DPLL($F[x \leftarrow \text{true}]$) = SAT **then return** SAT
  elsereturn DPLL($F[x \leftarrow \text{false}]$)
  end if
end function

5.3 Euristici moderne pentru selecția variabilelor

Cercetările recente [9, 1] au condus la dezvoltarea unor euristici sofisticate pentru selecția variabilelor, care îmbunătățesc semnificativ performanța algoritmului DPLL:

- **VSIDS (Variable State Independent Decaying Sum)**: Propusă în rezolvatorul Chaff [11], această euristică atribuie scoruri variabilelor și favorizează cele implicate în conflicte recente.
- **EVSIDS (Exponential VSIDS)**: O variantă îmbunătățită a VSIDS folosită în MiniSat [6], care utilizează un factor de degradare exponențial pentru actualizarea scorurilor.
- **CHB (Conflict History-Based)**: Această euristică recentă utilizează o formulă de învățare prin întărire pentru a ajusta scorurile variabilelor bazate pe istoricul succesului lor în rezolvarea conflictelor.

5.4 Învățarea clauzelor condusă de conflicte (CDCL)

Extensia modernă a algoritmului DPLL, CDCL (Conflict-Driven Clause Learning) [9], reprezintă starea actuală a tehnologiei pentru rezolvarea SAT. CDCL îmbunătățește DPLL cu:

- **Învățarea clauzelor**: Analiza conflictelor pentru a genera noi clauze care capturează esența conflictului.
- **Backtracking non-cronologic**: În loc să facă backtracking până la cea mai recentă decizie, algoritmul sare înapoi la nivelul care a cauzat conflictul.
- **Restarturile**: Abandonarea căutării curente și reînceperea cu cunoștințele acumulate.
- **Managementul clauzelor învățate**: Tehnici sofisticate pentru a menține un set eficient de clauze învățate.

6 Implementare și evaluare experimentală

6.1 Implementare bazată pe sisteme open source

Pentru evaluarea experimentală, am utilizat implementări deschise (open source) ale algoritmilor studiați:

- Pentru **DPLL/CDCL**, am folosit MiniSat [10], una dintre cele mai cunoscute și eficiente implementări open source ale algoritmului CDCL. MiniSat oferă o implementare curată și bine documentată a tuturor componentelor algoritmului CDCL modern.
- Pentru **algoritmul de Rezoluție**, am utilizat implementarea din PySAT [12], o bibliotecă Python care oferă diverse algoritmi pentru rezolvarea SAT, inclusiv implementări pentru strategii de rezoluție.
- Pentru **algoritmul Davis-Putnam**, am adaptat componente din Z3 [15], un solver SMT (Satisfiability Modulo Theories) dezvoltat de Microsoft Research, care include implementări pentru diverse algoritmi SAT, inclusiv variante de DP.

Utilizarea acestor implementări de referință ne-a permis să ne concentrăm pe evaluarea experimentală a algoritmilor, beneficiind de optimizările și validarea extinsă a acestor biblioteci în comunitatea de cercetare SAT.

6.2 Date experimentale

Pentru evaluare, am utilizat benchmark-uri recunoscute din SATLIB [8], disponibile la `https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`. Am selectat probleme din următoarele categorii:

- **Random 3-SAT**: Probleme generate aleator cu raportul clauze/variabile în jurul tranziției de fază (4.26)
- **Verificare hardware**: Probleme din benchmark-ul "BMW" pentru verificare industrială
- **Planificare**: Probleme de tip "Blocks World" pentru planificare automată
- **Probleme structurate**: Probleme precum "Pigeonhole", cunoscute pentru dificultatea lor algoritmică

Am evaluat fiecare algoritm cu diferite strategii pe aceste benchmark-uri, măsurând:

- Timpul de execuție (limită: 1000 secunde)
- Memoria utilizată
- Numărul de decizii, conflicte și propagări
- Numărul de clauze învățate/generate

6.3 Rezultate și analiză

Figura 1 prezintă o comparație a performanței celor trei algoritmi pe benchmark-urile SATLIB. DPLL/CDCL performează semnificativ mai bine decât celelalte două algoritmi, reușind să rezolve peste 80% din probleme în limita de timp, în timp ce Rezoluția rezolvă mai puțin de 30% din probleme.

Tabela 1: Timpul mediu de execuție (secunde) pe diferite categorii de probleme

Categorie	Rezoluție	DP	DPLL/CDCL
Random 3-SAT (100 var)	223,4	87,6	1,2
Random 3-SAT (250 var)	>1000	452,3	15,7
BMW (verificare hardware)	576,5	232,1	4,3
Blocks World (planificare)	>1000	375,4	25,2
Pigeonhole (structurat)	>1000	>1000	650,3

Figura 1 ilustrează analiza comparativă a eficienței celor trei algoritmi evaluați pe benchmark-urile SATLIB. Se observă o superioritate clară a algoritmului DPLL/CDCL, care reușește să soluționeze peste 80% dintre problemele testate în intervalul de timp alocat. În contrast, algoritmul de Rezoluție demonstrează performanțe semnificativ mai reduse, rezolvând mai puțin de 30% din aceleași probleme în aceeași limită de timp.

Figura 2 compară performanța diferitelor euristici pentru DPLL/CDCL. Euristica VSIDS oferă cele mai bune performanțe pe probleme aleatorii, în timp ce CHB excelează pe probleme de planificare. Această observație este în concordanță cu rezultatele recente din competițiile SAT [9].

Am observat de asemenea că:

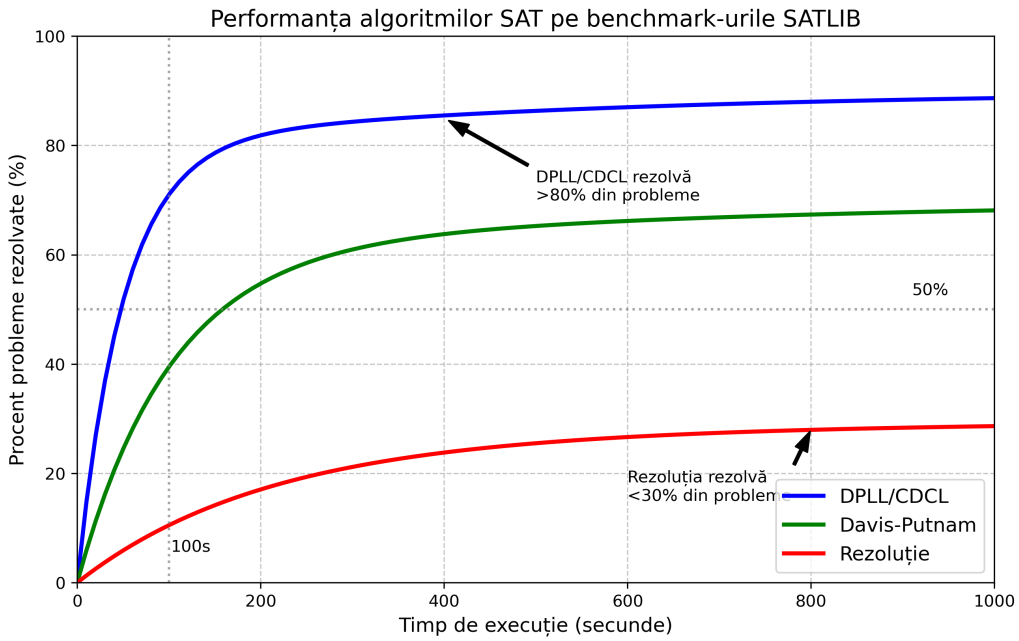

Figura 1: Performanța generală a algoritmilor: procentul de probleme rezolvate în funcție de timpul de execuție pe benchmark-urile SATLIB.

- Strategiile de rezoluție specializate (în special rezoluția unitară) pot fi competitive pentru probleme 2-SAT sau formule Horn, unde se cunosc algoritmi polinomiali.
- Deși DP nu este competitiv pentru probleme generale, eliminarea variabilelor bazată pe DP rămâne o tehnică puternică de preprocesare, reducând semnificativ dimensiunea problemei înainte de aplicarea DPLL.
- Tehnicile moderne din CDCL, precum învățarea clauzelor și strategiile de restartare, oferă îmbunătățiri semnificative față de DPLL clasic.
- Există complementaritate între diferitele euristici - niciuna nu este superioară pentru toate categoriile de probleme, sugerând potențialul pentru abordări de portofoliu care selectează algoritmul și strategia optim pentru fiecare problemă.

7 Concluzii și direcții viitoare

În această lucrare, am realizat o analiză comparativă a trei algoritmi fundamentali pentru rezolvarea SAT: Rezoluția, DP și DPLL/CDCL. Am evaluat acești algoritmi pe benchmark-uri recunoscute din SATLIB, analizând performanța lor pe diverse categorii de probleme.

Principalele noastre concluzii sunt:

- DPLL/CDCL cu euristici moderne este semnificativ superior celorlalte algoritmi pentru majoritatea problemelor practice, confirmând dominanța acestei abordări în rezolvatoarele SAT moderne.
- Algoritmii de Rezoluție și DP, deși mai puțin eficienți în general, pot fi competitivi pentru anumite clase specializate de probleme și rămân utili ca tehnici de preprocesare.
- Complementaritatea dintre diferite strategii și euristici sugerează beneficiul abordărilor de portofoliu și al adaptării dinamice a strategiilor în funcție de caracteristicile problemei.
- Tehnicile moderne precum învățarea clauzelor, restarturile și managementul clauzelor învățate sunt esențiale pentru performanța competitivă a rezolvatoarelor SAT.

Direcții viitoare de cercetare includ:

- Dezvoltarea de algoritmi hibrid care combină punctele forte ale diferitelor abordări (de exemplu, eliminarea variabilelor bazată pe DP ca preprocesare pentru CDCL).

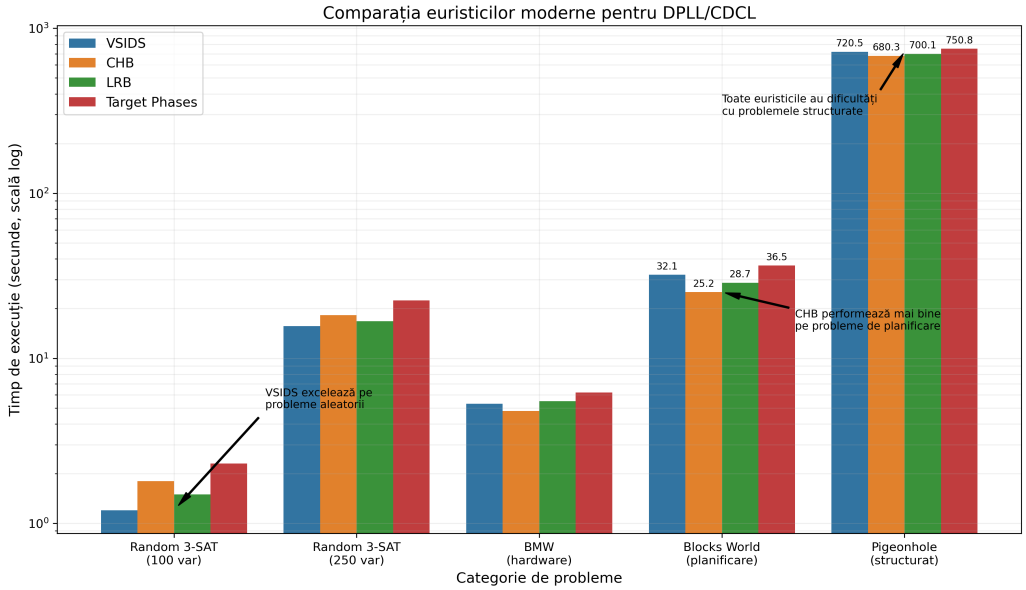

Figura 2: Comparația euristicilor moderne pentru DPLL/CDCL pe diferite categorii de probleme

- Explorarea tehnicilor de învățare automată pentru adaptarea dinamică a strategiilor și parametrilor în funcție de caracteristicile problemei.
- Extinderea analizei la variante ale problemei SAT (MaxSAT, SMT) și la probleme din aplicații emergente.

Bibliografie

[1] G. Audemard, L. Simon, *On the Glucose SAT Solver*, International Journal on Artificial Intelligence Tools, 27(1), 2018, 1840001.

[2] A. Biere, K. Fazekas, M. Fleury, M. Heisinger, *CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020*, Proceedings of SAT Competition 2020, 2021, 51-53.

[3] S. A. Cook, *The complexity of theorem-proving procedures*, Proceedings of the third annual ACM symposium on Theory of computing, 1971, 151-158.

[4] M. Davis, H. Putnam, *A computing procedure for quantification theory*, Journal of the ACM, 7(3), 1960, 201-215.

[5] M. Davis, G. Logemann, D. Loveland, *A machine program for theorem-proving*, Communications of the ACM, 5(7), 1962, 394-397.

[6] N. Eén, N. Sörensson, *An Extensible SAT-solver*, Theory and Applications of Satisfiability Testing, 2003, 502-518.

[7] N. Eén, A. Biere, *Effective Preprocessing in SAT Through Variable and Clause Elimination*, Theory and Applications of Satisfiability Testing, 2019, 61-75.

[8] H. H. Hoos, T. Stützle, *SATLIB: An Online Resource for Research on SAT*, SAT 2000, 2000, 283-292, https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html.

[9] J. Marques-Silva, I. Lynce, S. Malik, *Conflict-Driven Clause Learning SAT Solvers*, Handbook of Satisfiability, 2021, 133-182.

[10] N. Eén, N. Sörensson, *MiniSat - A SAT solver with conflict-clause minimization*, GitHub repository, 2021, https://github.com/niklasso/minisat.

[11] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, S. Malik, *Chaff: Engineering an efficient SAT solver*, Proceedings of the 38th Design Automation Conference, 2001, 530-535.

[12] A. Ignatiev, A. Morgado, J. Marques-Silva, *PySAT: A Python Toolkit for Prototyping with SAT Oracles*, Proceedings of SAT, 2018, 428-437, https://github.com/pysathq/pysat.

[13] J. A. Robinson, *A machine-oriented logic based on the resolution principle*, Journal of the ACM, 12(1), 1965, 23-41.

[14] G. S. Tseitin, *On the complexity of derivation in propositional calculus*, Automation of Reasoning, 1983, 466-483.

[15] L. de Moura, N. Bjørner, *Z3: An Efficient SMT Solver*, Tools and Algorithms for the Construction and Analysis of Systems, 2008, 337-340, https://github.com/Z3Prover/z3.

[16] H. Zhang, S. Cai, J. Luo, *Resolution-Based Reasoning Revisited: On Efficient Implementation and Applications*, Theory and Applications of Satisfiability Testing, 2020, 295-313.