Proof Complexity and Its Relations to SAT-Solving

Albert Atserias
Universitat Politècnica de Catalunya
Centre de Recerca Matemàtica
Barcelona, Catalonia, Spain

Overview of the talk

PART I: PROOF COMPLEXITY AND SAT

1. Propositional Logic
2. SAT-Solvers
3. Frege Systems
4. Cut-Free and Cut-Only Proofs

PART II: COMPLEXITY OF PROOF SEARCH

1. Proof Search and Automatability
2. Proof of NP-hardness for Resolution
3. An Open Problem

Part I

PROOF COMPLEXITY AND SAT

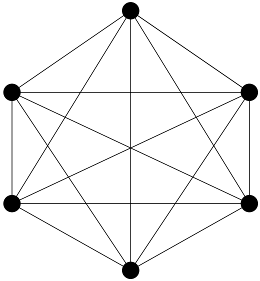
Satisfiability

Example 1: 15 variables and $40 = 20 + 20$ clauses

$x_1 \vee x_2 \vee x_6$	$x_1 \vee x_3 \vee x_7$	$x_1 \vee x_4 \vee x_8$	$x_1 \vee x_5 \vee x_9$
$x_2 \vee x_3 \vee x_{10}$	$x_2 \vee x_4 \vee x_{11}$	$x_2 \vee x_5 \vee x_{12}$	$x_3 \vee x_4 \vee x_{13}$
$x_3 \vee x_5 \vee x_{14}$	$x_4 \vee x_5 \vee x_{15}$	$x_6 \vee x_7 \vee x_{10}$	$x_6 \vee x_8 \vee x_{11}$
$x_6 \vee x_9 \vee x_{12}$	$x_7 \vee x_8 \vee x_{13}$	$x_7 \vee x_9 \vee x_{14}$	$x_8 \vee x_9 \vee x_{15}$
$x_{10} \vee x_{11} \vee x_{13}$	$x_{10} \vee x_{12} \vee x_{14}$	$x_{11} \vee x_{12} \vee x_{15}$	$x_{13} \vee x_{14} \vee x_{15}$
$\overline{x_1} \vee \overline{x_2} \vee \overline{x_6}$	$\overline{x_1} \vee \overline{x_3} \vee \overline{x_7}$	$\overline{x_1} \vee \overline{x_4} \vee \overline{x_8}$	$\overline{x_1} \vee \overline{x_5} \vee \overline{x_9}$
$\overline{x_2} \vee \overline{x_3} \vee \overline{x_{10}}$	$\overline{x_2} \vee \overline{x_4} \vee \overline{x_{11}}$	$\overline{x_2} \vee \overline{x_5} \vee \overline{x_{12}}$	$\overline{x_3} \vee \overline{x_4} \vee \overline{x_{13}}$
$\overline{x_3} \vee \overline{x_5} \vee \overline{x_{14}}$	$\overline{x_4} \vee \overline{x_5} \vee \overline{x_{15}}$	$\overline{x_6} \vee \overline{x_7} \vee \overline{x_{10}}$	$\overline{x_6} \vee \overline{x_8} \vee \overline{x_{11}}$
$\overline{x_6} \vee \overline{x_9} \vee \overline{x_{12}}$	$\overline{x_7} \vee \overline{x_8} \vee \overline{x_{13}}$	$\overline{x_7} \vee \overline{x_9} \vee \overline{x_{14}}$	$\overline{x_8} \vee \overline{x_9} \vee \overline{x_{15}}$
$\overline{x_{10}} \vee \overline{x_{11}} \vee \overline{x_{13}}$	$\overline{x_{10}} \vee \overline{x_{12}} \vee \overline{x_{14}}$	$\overline{x_{11}} \vee \overline{x_{12}} \vee \overline{x_{15}}$	$\overline{x_{13}} \vee \overline{x_{14}} \vee \overline{x_{15}}$

Diagonal Ramsey Numbers $R(k, k)$

$$R(3, 3) \le 6$$



In every party of six,
either three of them are mutual friends,
or three of them are mutual strangers.

Ramsey Numbers, Erdős, and the Aliens

Erdős asks us to imagine an alien force, vastly more powerful than us, landing on Earth and demanding the value of $R(5,5)$ or they will destroy our planet. In that case, he claims, we should marshal all our computers and all our mathematicians and attempt to find the value. But suppose, instead, that they ask for $R(6,6)$. In that case, he believes, we should attempt to destroy the aliens.

Joel Spencer, Ten Lectures on the Probabilistic Method, 1994.

Encodings Can Be Subtle

Different encoding: n^k vs $k^2 n^2$.

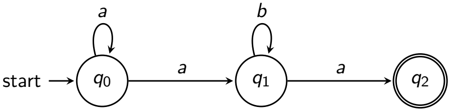
$b_{u,v}$: "the pair $\{u, v\}$ is colored blue (else red)"
$x_{i,u}$: "u is the i-th vertex of a blue k-clique"
$y_{i,v}$: "v is the i-th vertex of a red k-clique"

$$
\begin{array}{ll}
x_{i,1} \vee \cdots \vee x_{i,n} & \text{for all } i, \\
\overline{x_{i,u}} \vee \overline{x_{j,u}} & \text{for all } i \neq j \text{ and all } u, \\
\overline{x_{i,u}} \vee \overline{x_{j,v}} \vee b_{u,v} & \text{for all } i \neq j \text{ and all } u \neq v,
\end{array}
$$

$$
\begin{array}{ll}
y_{i,1} \vee \cdots \vee y_{i,n} & \text{for all } i, \\
\overline{y_{i,u}} \vee \overline{y_{j,u}} & \text{for all } i \neq j \text{ and all } u, \\
\overline{y_{i,u}} \vee \overline{y_{j,v}} \vee \overline{b_{u,v}} & \text{for all } i \neq j \text{ and all } u \neq v,
\end{array}
$$

More satisfiability

Example 2: Automaton accepts some n-symbol word.



x_i : "the i-th symbol in word is a (else b)"
$s_{t,q}$: "after reading t symbols the state is q"

s_{0,q_0}
$\overline{s_{t,q_0}} \vee \overline{x_t} \vee s_{t+1,q_0} \vee s_{t+1,q_1}$ for $t = 0, 1, \ldots, n$
$\overline{s_{t,q_0}} \vee \overline{x_t} \vee \overline{s_{t+1,q_2}}$ for $t = 0, 1, \ldots, n$
$\overline{s_{t,q_0}} \vee x_t \vee \overline{s_{t+1,q_0}}$ for $t = 0, 1, \ldots, n$
$\overline{s_{t,q_0}} \vee x_t \vee \overline{s_{t+1,q_1}}$ for $t = 0, 1, \ldots, n$
$\overline{s_{t,q_0}} \vee x_t \vee \overline{s_{t+1,q_2}}$ for $t = 0, 1, \ldots, n$
...
s_{n,q_2}

Cook-Levin and Fagin Theorems

Theorem [Cook-Levin 1971] SAT is NP-complete.

A is in NP
iff
A can be reduced to SAT
by polynomial-time reductions.

Theorem [Fagin 1974] NP $=$ ESO.

A is in NP
iff
A is a satisfiability problem itself, i.e.,
iff
A is the set of finite models of
a formula of the existential fragment
of second-order logic $\exists \overline{R}\ \forall \overline{x}\ \exists \overline{y}\ \mathrm{qf}$

An algorithm which:

Given a set of clauses F, finds:

either a satisfying assignment
or a proof of unsatisfiability

Caution:

For formulas with 1000 variables,
the search space is ridiculously HUGE!

“200 TB maths proof is largest ever” [Nature 2016]

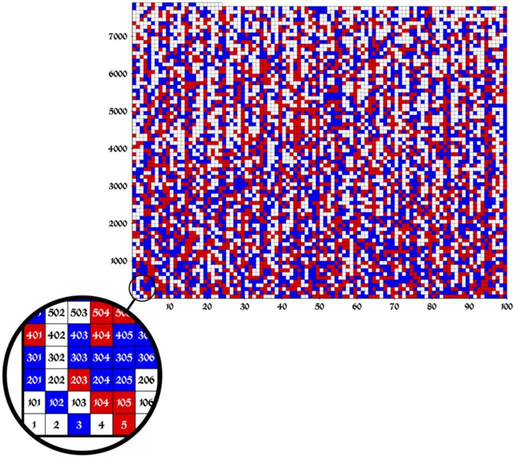
Theorem [Heule-Kullmann-Marek 2016]
The numbers $1, \ldots, 7825$ cannot be partitioned into two parts each without Pythagorean triples.

But the numbers $1, \ldots, 7824$, can.

$$a^2 + b^2 = c^2$$
$$a^2 + b^2 = c^2$$

The Coloring of $1, \ldots, 7824$

$a^2 + b^2 \neq c^2$
$a^2 + b^2 \neq c^2$



Source of image: Wikipedia

Certificates

Recall:

Given a set of clauses F, algorithm finds:

either a satisfying assignment
or a proof of unsatisfiability

An annoying asymetry:

Satisfying assignments are always small.
Proofs of unsatisfiability tend to be exponentially bigger.

This, among other reasons, motivates the study of propositional proof complexity.

Frege Systems, aka Hilbert-style Proof Systems

Language:

$$\to, \neg$$

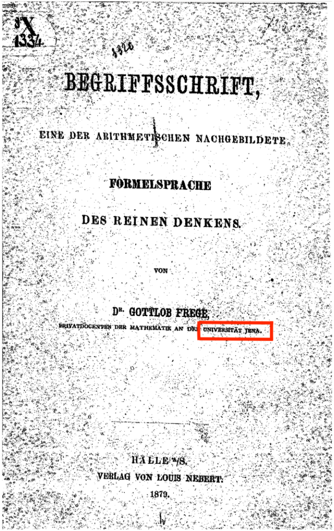
Modus ponens:

$$\frac{A \qquad A \to B}{B}$$

Axioms:

$$A \to (B \to A)$$
$$(C \to (B \to A)) \to ((C \to B) \to (C \to A))$$
$$(D \to (B \to A)) \to (B \to (D \to A))$$
$$(B \to A) \to (\neg A \to \neg B)$$
$$\neg\neg A \to A$$
$$A \to \neg\neg A$$

Gottlob Frege, Begriffsschrift, Universität Jena, 1879



BEGRIFFSSCHRIFT,

EINE DER ARITHMETISCHEN NACHGEBILDETE

FORMELSPRACHE

DES REINEN DENKENS.

VON

Dr. GOTTLOB FREGE,

PRIVATDOCENTEN DER MATHEMATIK AN DER UNIVERSITÄT JENA.

HALLE a/S.

VERLAG VON LOUIS NEBERT.

1879.

Source: Wikipedia
Guus Hoekman

Cook-Reckhow Theorem: Birth of Proof Complexity

Theorem [Cook-Reckhow'1979]
Any two Frege systems polynomially simulate each other.

Notes:

- Polynomial simulation $\equiv$ polynomial time translations exist.
- Also for "Extended Frege Systems": abbreviations allowed.
- Mild conditions apply: soundness, implicational completeness, complete basis of connectives.

Tait Style Systems

Language: $\wedge$, $\vee$, x_i, $\overline{x_i}$ (Negation Normal Form: A and $\overline{A}$)

Rules: Axiom, Weakening, Conjunction, Cut

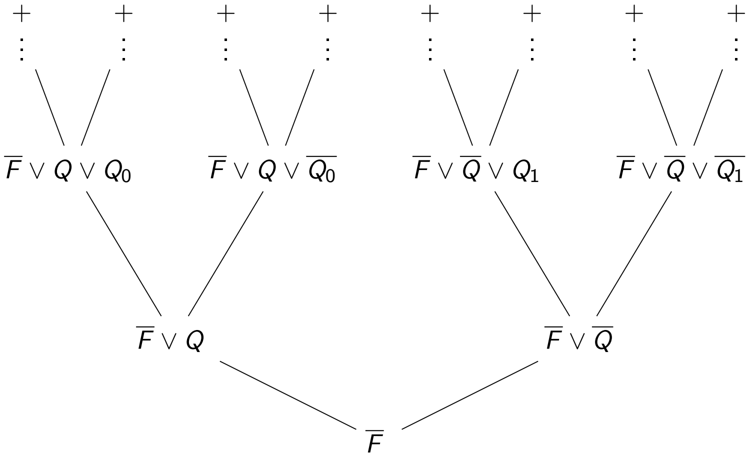
$$\frac{}{A \vee \overline{A}} \qquad \frac{A}{A \vee B} \qquad \frac{A \vee C \qquad B \vee D}{A \vee B \vee (C \wedge D)} \qquad \frac{A \vee C \qquad B \vee \overline{C}}{A \vee B}$$

Soundness: Obvious
Completeness: Also almost obvious; even cut-free!
Quantitative completeness:

$$2^{\#\mathrm{vars}(F)} \cdot \#\mathrm{gates}(F).$$

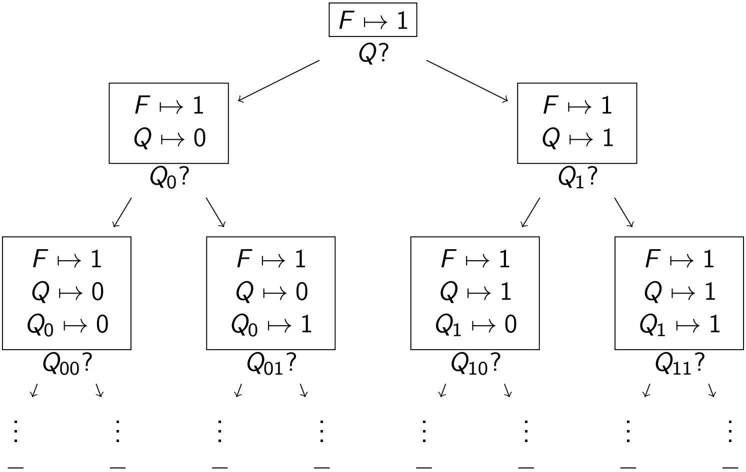
Resolution $\stackrel{\mathrm{def}}{\equiv}$ cut-only proofs from clauses to clauses.

Goal: Prove F is unsatisfiable. **Means:** Build up $\overline{F}$ from axioms.



Decision Trees

Goal: Prove F is unsatisfiable. **Means:** Reduce F to $\overline{\text{axioms}}$



Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

- 1 is direct because trees are very regular: turn upside down.

Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

- 1 is direct because trees are very regular: turn upside down.
- 1 gives proofs whose lines are disjunctions of (co-)queries.

Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

- 1 is direct because trees are very regular: turn upside down.
- 1 gives proofs whose lines are disjunctions of (co-)queries.
- 1 remains true if lines are clauses and queries are literals.

Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

- 1 is direct because trees are very regular: turn upside down.
- 1 gives proofs whose lines are disjunctions of (co-)queries.
- 1 remains true if lines are clauses and queries are literals.
- 2 is not direct because proofs can be very irregular; indeed:

Quantitative Equivalence (With a But)

Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

- 1 is direct because trees are very regular: turn upside down.
- 1 gives proofs whose lines are disjunctions of (co-)queries.
- 1 remains true if lines are clauses and queries are literals.
- 2 is not direct because proofs can be very irregular; indeed:
- 2 provably not true if lines are clauses and queries are literals.

Quantitative Equivalence (With a But)

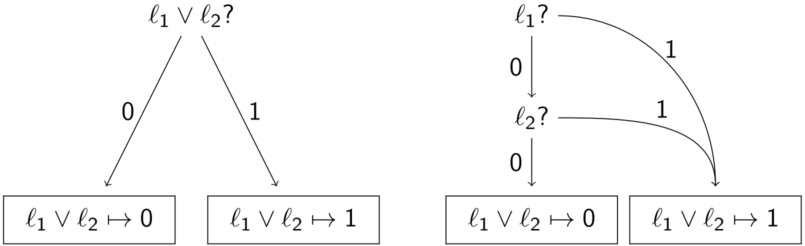
Theorem [Buss-Pudlak'1995]

1. If there is a decision tree proof of $\overline{F}$ with L nodes,
   then there is a proof of $\overline{F}$ with $\mathrm{poly}(L)$ lines.
2. If there is a proof of $\overline{F}$ with L lines,
   then there is a decision tree proof of $\overline{F}$ with $\mathrm{poly}(L)$ nodes.

Notes:

- 1 is direct because trees are very regular: turn upside down.
- 1 gives proofs whose lines are disjunctions of (co-)queries.
- 1 remains true if lines are clauses and queries are literals.
- 2 is not direct because proofs can be very irregular; indeed:
- 2 provably not true if lines are clauses and queries are literals.
  Separation: Pebbling Formulas [Ben-Sasson-Wigderson'99].

Solution: Decision DAGs



Resolution

Definition Given $F = C_1 \wedge \cdots \wedge C_m$ with each C_i a clause, a Resolution refutation of F is a cut-only proof

$$C_1, \ldots, C_m, D_1, D_2, \ldots, D_L = \emptyset$$

of the $\emptyset$ from the C_i.

Proposition
Up to multiplicative constants, the following are the same:

1. Decision trees with clause-queries and L nodes.
2. Decision dags with literal-queries and L nodes.
3. Tree-like DNF-proofs of length L.
4. Dag-like clause-proofs of length L.
5. Resolution refutations of length L.

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level
5. Frequent restarts keeping (some of the) learned clauses

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level
5. Frequent restarts keeping (some of the) learned clauses
6. Deletions

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level
5. Frequent restarts keeping (some of the) learned clauses
6. Deletions
7. Ultrafast unit clause propagation (2 watched literal rule)

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level
5. Frequent restarts keeping (some of the) learned clauses
6. Deletions
7. Ultrafast unit clause propagation (2 watched literal rule)
8. Ultrafast decision heuristic based on activity (VSIDS)

Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level
5. Frequent restarts keeping (some of the) learned clauses
6. Deletions
7. Ultrafast unit clause propagation (2 watched literal rule)
8. Ultrafast decision heuristic based on activity (VSIDS)
9. Preprocessing and inprocessing

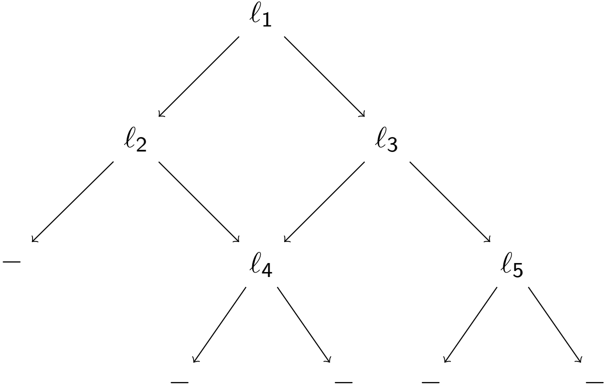
Back to SAT-Solvers

DPLL: Searches for tree-like Resolution proofs
CDCL: Searches for dag-like Resolution proofs

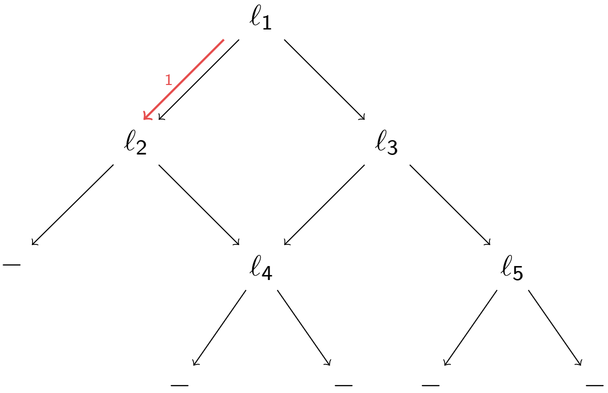
Some of the Key Ideas:

1. Backtracking search
2. Greedy unit clause propagation
3. Memoization following conflict analysis: aka clause learning
4. Backjumping possibly more than one level
5. Frequent restarts keeping (some of the) learned clauses
6. Deletions
7. Ultrafast unit clause propagation (2 watched literal rule)
8. Ultrafast decision heuristic based on activity (VSIDS)
9. Preprocessing and inprocessing
10. Symmetry breaking
11. ...

Execution Trace of CDCL-Based SAT-Solvers



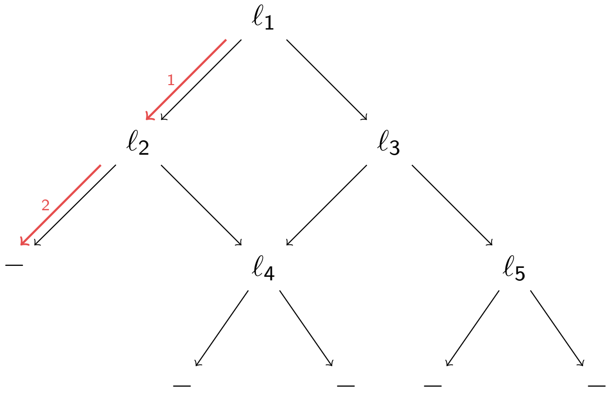
Execution Trace of CDCL-Based SAT-Solvers



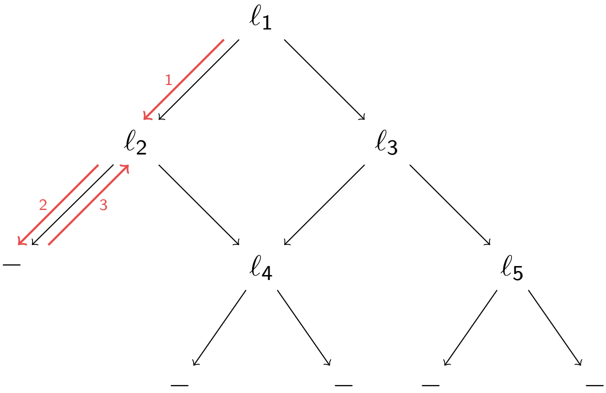
1. decide

Execution Trace of CDCL-Based SAT-Solvers

1. decide
2. decide

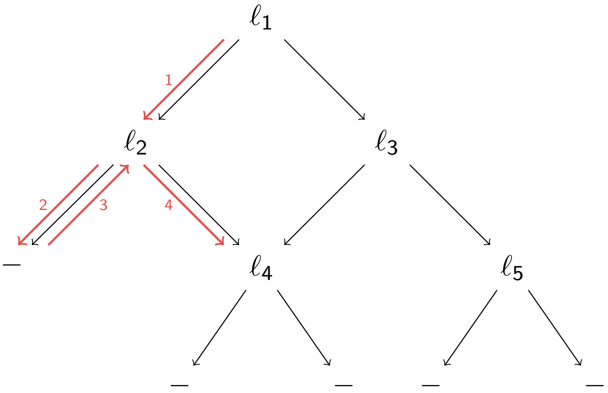


Execution Trace of CDCL-Based SAT-Solvers



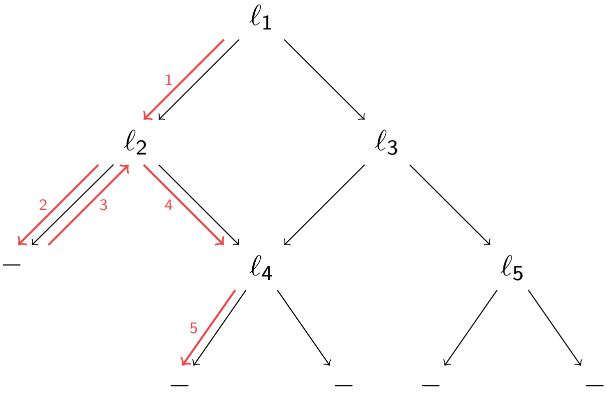
1. decide
2. decide
3. conflict / learn

Execution Trace of CDCL-Based SAT-Solvers



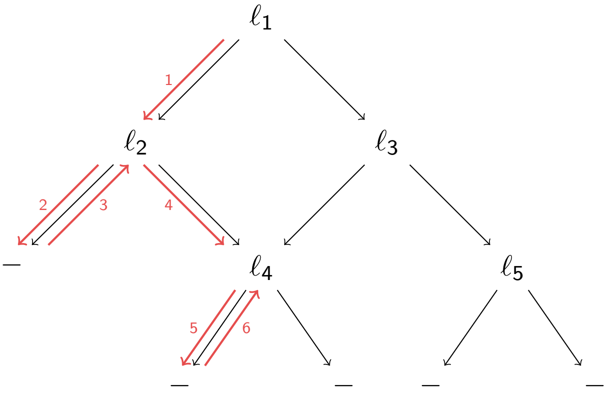
1. decide
2. decide
3. conflict / learn
4. propagate

Execution Trace of CDCL-Based SAT-Solvers



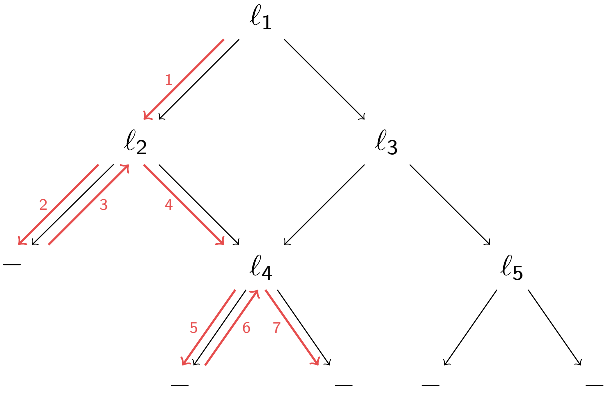
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide

Execution Trace of CDCL-Based SAT-Solvers



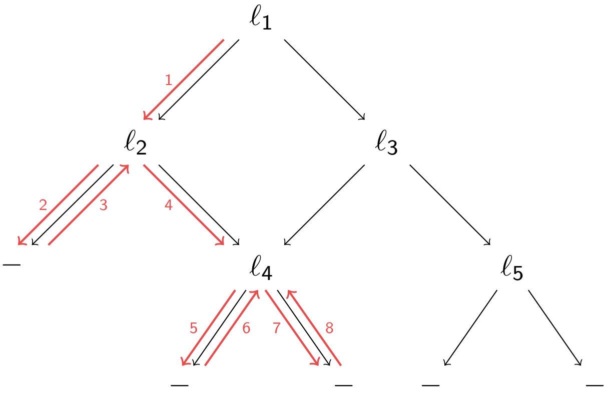
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn

Execution Trace of CDCL-Based SAT-Solvers



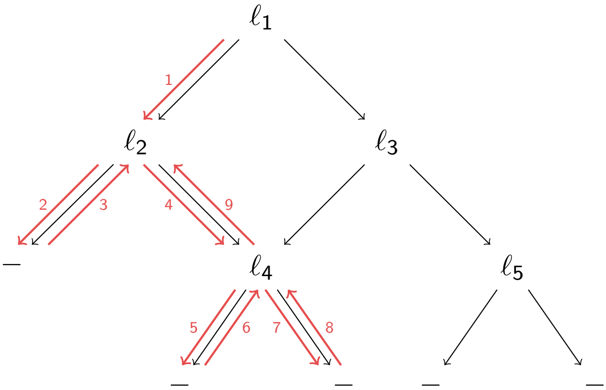
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate

Execution Trace of CDCL-Based SAT-Solvers



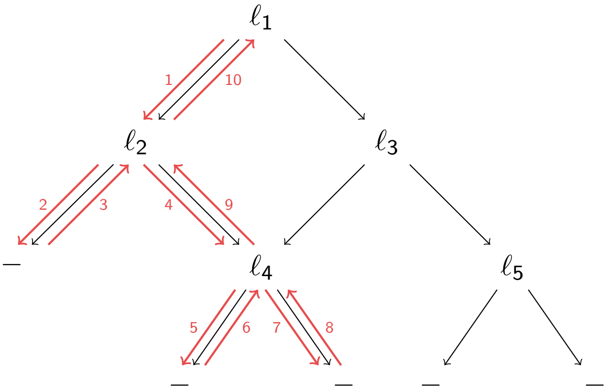
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis

Execution Trace of CDCL-Based SAT-Solvers



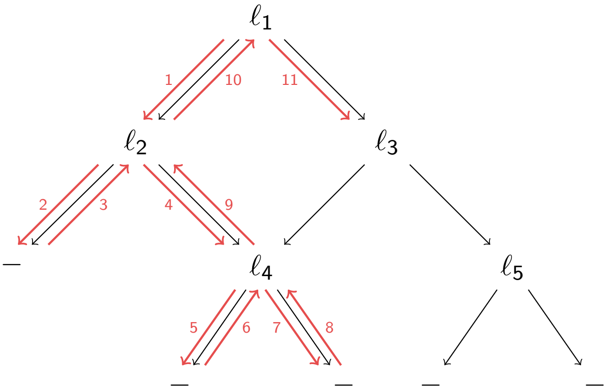
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis

Execution Trace of CDCL-Based SAT-Solvers



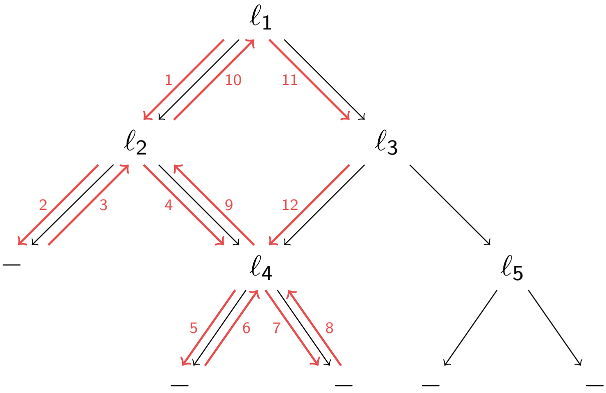
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn

Execution Trace of CDCL-Based SAT-Solvers



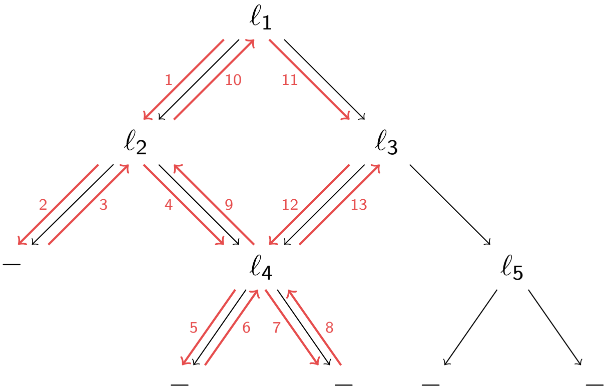
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate

Execution Trace of CDCL-Based SAT-Solvers



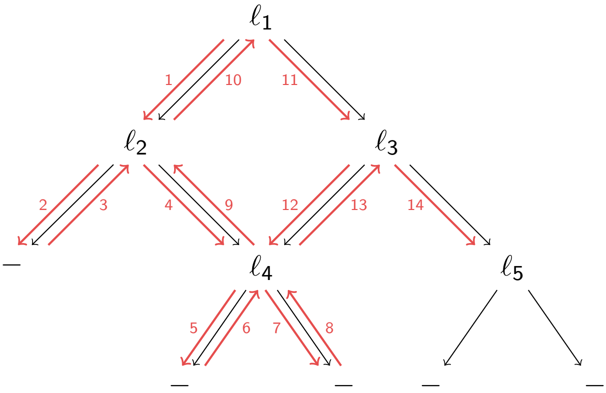
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide

Execution Trace of CDCL-Based SAT-Solvers



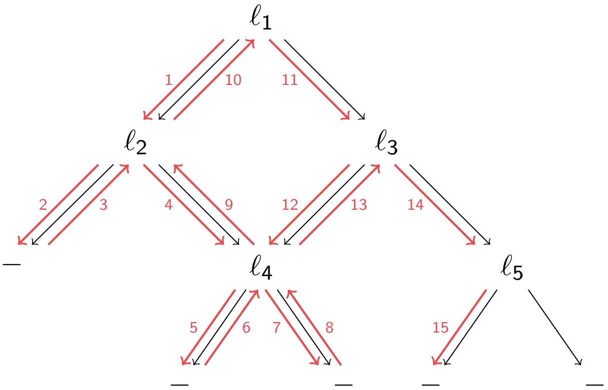
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn

Execution Trace of CDCL-Based SAT-Solvers



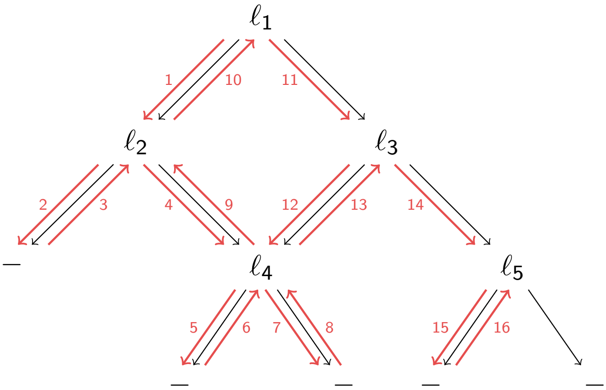
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate

Execution Trace of CDCL-Based SAT-Solvers



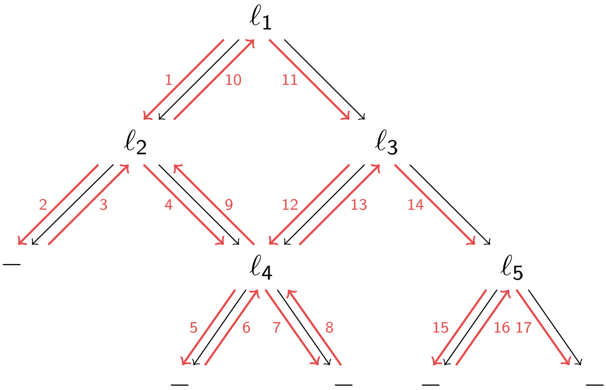
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate
15. decide

Execution Trace of CDCL-Based SAT-Solvers



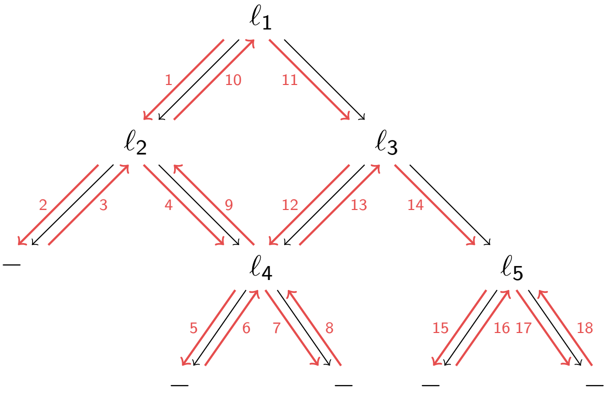
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate
15. decide
16. conflict / learn

Execution Trace of CDCL-Based SAT-Solvers



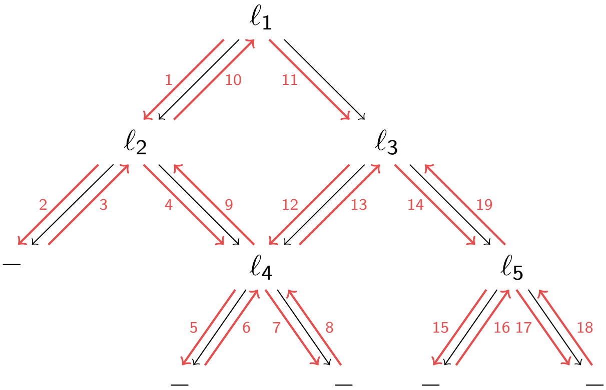
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate
15. decide
16. conflict / learn
17. propagate

Execution Trace of CDCL-Based SAT-Solvers



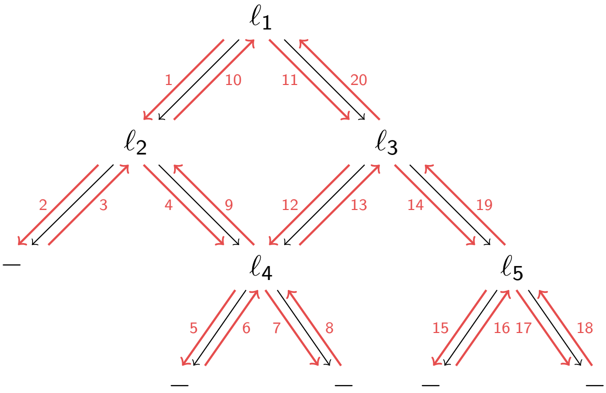
1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate
15. decide
16. conflict / learn
17. propagate
18. conflict / analysis

Execution Trace of CDCL-Based SAT-Solvers



1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate
15. decide
16. conflict / learn
17. propagate
18. conflict / analysis
19. conflict / analysis

Execution Trace of CDCL-Based SAT-Solvers



1. decide
2. decide
3. conflict / learn
4. propagate
5. decide
6. conflict / learn
7. propagate
8. conflict / analysis
9. conflict / analysis
10. conflict / learn
11. propagate
12. decide
13. conflict with learned / learn
14. propagate
15. decide
16. conflict / learn
17. propagate
18. conflict / analysis
19. conflict / analysis
20. conflict / learn empty clause

Comparison with Resolution

Theorem [Beame-Kautz-Sabharwal'2004]
If a CNF F with n variables has a Resolution refutation of length L, then there is a sequence of non-deterministic ideal choices for CDCL with restarts, rebranching, and any non-redundant learning scheme that learns the empty clause in $O(nL)$ steps.

Comparison with Resolution

Theorem [Beame-Kautz-Sabharwal'2004]
If a CNF F with n variables has a Resolution refutation of length L, then there is a sequence of non-deterministic ideal choices for CDCL with restarts, rebranching, and any non-redundant learning scheme that learns the empty clause in $O(nL)$ steps.

Notes

Comparison with Resolution

Theorem [Beame-Kautz-Sabharwal'2004]
If a CNF F with n variables has a Resolution refutation of length L, then there is a sequence of non-deterministic ideal choices for CDCL with restarts, rebranching, and any non-redundant learning scheme that learns the empty clause in $O(nL)$ steps.

Notes

- The "rebranching" is never done in real solvers

Comparison with Resolution

Theorem [Beame-Kautz-Sabharwal'2004]
If a CNF F with n variables has a Resolution refutation of length L, then there is a sequence of non-deterministic ideal choices for CDCL with restarts, rebranching, and any non-redundant learning scheme that learns the empty clause in $O(nL)$ steps.

Notes

- The "rebranching" is never done in real solvers
- Later removed at cost $O(n^4 L)$ [Pipatsrisawat-Darwiche'09]

Comparison with Resolution

Theorem [Beame-Kautz-Sabharwal'2004]
If a CNF F with n variables has a Resolution refutation of length L, then there is a sequence of non-deterministic ideal choices for CDCL with restarts, rebranching, and any non-redundant learning scheme that learns the empty clause in $O(nL)$ steps.

Notes

- The "rebranching" is never done in real solvers
- Later removed at cost $O(n^4 L)$ [Pipatsrisawat-Darwiche'09]
- Is non-determinism essential? ... now answered (next lecture)

Comparison with Resolution

Theorem [Beame-Kautz-Sabharwal'2004]
If a CNF F with n variables has a Resolution refutation of length L, then there is a sequence of non-deterministic ideal choices for CDCL with restarts, rebranching, and any non-redundant learning scheme that learns the empty clause in $O(nL)$ steps.

Notes

- The "rebranching" is never done in real solvers
- Later removed at cost $O(n^4 L)$ [Pipatsrisawat-Darwiche'09]
- Is non-determinism essential? ... now answered (next lecture)
- For bounded width Resolution (e.g., 2-SAT, bounded tree-width), randomness suffices to ensure $n^{O(\text{width})}$ steps [Atserias-Fichte-Thurley'09]

Theorem [Haken'1986]
Every Resolution refutation of the Pigeonhole Principle formulas PHP_n^{n+1} must have length $2^{\Omega(n)}$.

Pigeonhole Principle Formulas PHP_n^{n+1}:

$p_{u,j}$: "pigeon $u \in \{1, \ldots, n+1\}$ flies to hole $j \in \{1, \ldots, n\}$"

$p_{u,1} \vee \cdots \vee p_{u,n}$ for all u
$\overline{p_{u,j}} \vee \overline{p_{v,j}}$ for all $u \neq v$ and all j

Random Restriction Method in Three Steps: I

STEP I: Choose a suitable collection H of partial assignments α, so that the restricted formula $\mathrm{PHP}_n^{n+1}|_\alpha$ is isomorphic to a smaller instance PHP_m^{m+1} of itself.

Here:

Let H be the set of partial assignments α that describe partial matchings of $n-m$ pigeons to $n-m$ holes.

$$\begin{array}{ll}
\alpha(p_{u,j}) = 1 & \text{if } u \text{ is matched to } j \\
\alpha(p_{u,j}) = 0 & \text{if } u \text{ is matched to } j' \neq j \\
\alpha(p_{u,j}) = 0 & \text{if } u \text{ is not matched and } j \text{ is matched} \\
\alpha(p_{u,j}) = p_{u,j} & \text{if } u \text{ is not matched and } j \text{ is not matched.}
\end{array}$$

We will choose $m = n/2$.

Random Restriction Method in Three Steps: II

STEP II: Define a suitable notion of weak clause that is very likely true under a random partial assignment from H.

Here:

A pigeon u is n-weak in the clause if the clause has

- $n/2$ positive literals $p_{u,j_1}, \ldots, p_{u,j_{n/2}}$ of pigeon u, or
- a negative literal $\overline{p_{u,j}}$ of pigeon u.

A clause is n-weak if there are $n/2$ many n-weak pigeons in it.

Rough estimation of probability:

- Fix a weak clause C; choose $\alpha \in H$ at random.
- Roughly $(n-m)/2 = n/4$ of the matched pigeons are weak.
- Roughly $1/2$ of the positive ones satisfy C.
- Roughly $1 - 1/(n-m) \geq 1/2$ of the negative ones satisfy C.

$$\Pr_{\alpha \in H}[C|_\alpha \neq 1] \lesssim (1/2)^{n/4}$$

Random Restriction Method in Three Steps: III

STEP III: Show that every Resolution refutation of PHP^{m+1}_{m} must contain at least one n-weak clause.

Here:

- For contradiction, fix a refutation without n-weak clauses.
- By $m = n/2$, in all clauses, not all pigeons are n-weak.
- Walking up the dag from the empty clause to the axioms, do:
- Sustain a partial matching from m pigeons to m holes.
- The partial matching will falsify the current clause.
- And the unmatched pigeon will not be weak in current clause.
- Initially: any matching works since all falsify the empty clause.
- At an inference step resolving on $p_{u,j}$:
- Follow the falsified clause.
- If unmatched pigeon became weak, exchange with non-weak.
- Eventually we reach a clause of PHP^{m+1}_{m}.
- Contradiction: our partial matchings do not falsify those. QED

Part II

COMPLEXITY OF PROOF SEARCH

Definition [Bonet-Pitassi-Raz'1999]
A proof system P is automatable in time $T(s)$ if there is an algorithm that given a tautology F finds a P-proof of F in time $T(s^*)$, where s^* is the size of the smallest P-proof of F.

Definition [Bonet-Pitassi-Raz'1999]
A proof system P is automatable in time $T(s)$ if there is an algorithm that given a tautology F finds a P-proof of F in time $T(s^*)$, where s^* is the size of the smallest P-proof of F.

A Fundamental Question

Which proof systems are automatable in non-trivial time?

A Non-Trivial Notion

An Early Lower Bound:

Theorem [Krajicek-Pudlak'1994]
Extended Frege systems are not automatable in time $T(s)$,
unless n-bit RSA cryptosystem can be broken in time $T(\mathrm{poly}(n))$.

A Non-Trivial Notion

An Early Lower Bound:

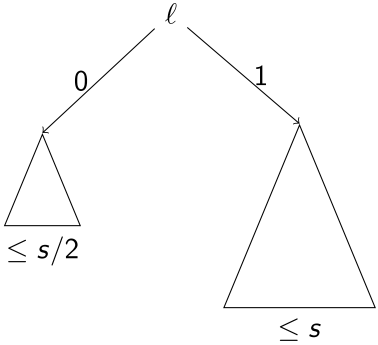
Theorem [Krajicek-Pudlak'1994]
Extended Frege systems are not automatable in time $T(s)$,
unless n-bit RSA cryptosystem can be broken in time $T(\mathrm{poly}(n))$.

An Early Upper Bound:

Theorem [Beame-Pitassi'1998]
Tree-like Resolution *is* automatable in time $T(s) = s^{O(\log s)}$.

Beame-Pitassi Algorithm

1. guess the root literal ℓ ($2n$ choices only)
2. recurse with parameter $s/2$ (abort the branch if it fails)
3. recurse with parameter s (it must succeed; subtle because the chosen ℓ need not be optimal).



$$T(n,s) \leq 2nT(n-1,s/2) + T(n-1,s)$$

$$T(n,s) = n^{O(\log s)} \leq s^{O(\log s)}.$$

Non-Automatability of Resolution

Theorem [Atserias-Müller'2019]
Resolution is not automatable in time $T(s)$, [poly-time]
unless n-variable SAT is solvable in time $T(\mathrm{poly}(n))$ [P = NP].

Non-Automatability of Resolution

Theorem [Atserias-Müller'2019]
Resolution is not automatable in time $T(s)$, [poly-time]
unless n-variable SAT is solvable in time $T(\mathrm{poly}(n))$ [P = NP].

Theorem [de Rezende'2021]
Tree-Like Resolution is not automatable in time $T(s) = s^{o(\log s)}$,
unless n-variable SAT is solvable in randomized time $2^{o(n)}$.

Non-Automatability of Resolution

Theorem [Atserias-Müller'2019]
Resolution is not automatable in time $T(s)$, [poly-time]
unless n-variable SAT is solvable in time $T(\mathrm{poly}(n))$ [P = NP].

Theorem [de Rezende'2021]
Tree-Like Resolution is not automatable in time $T(s) = s^{o(\log s)}$,
unless n-variable SAT is solvable in randomized time $2^{o(n)}$.

Notes:

- Compare with Beame-Pitassi algorithm!
- Improved earlier results of [Alekhnovich-Razborov'2001]
- Introduced a new method for proving non-automatability
- Correctness of the reduction involves proving a lower bound!

Proof Strategy for NP-Hardness

We want a polynomial-time reduction:

from	n-variable SAT
to	min proof-size approximation for Resolution (R).

$$F \xrightarrow{\mathrm{poly}(n)\ \text{time}} G_F$$

Requirements:

1. If F is satisfiable, then $\mathsf{SIZE}_{\mathsf{R}}(G_F) \leq \mathrm{poly}(n)$.
2. If F is unsatisfiable, then $\mathsf{SIZE}_{\mathsf{R}}(G_F) \not\leq \exp(\Omega(n))$.

Choice of the Formula G_F : the REF Formulas

$$\mathrm{REF}_{F,s} \quad = \quad \text{“the CNF formula } F \text{ has an R-refutation of length } s\text{”}$$

Variables:

$D_{u,i,b}$	:	“line u contains variable x_i with sign $b \in \{0,1\}$”
$I_{u,j}$	:	“line u is an initial assumption; the j-th clause of F”
$V_{u,i}$	:	“line u is derived by resolving on variable x_i”
$L_{u,v}$	:	“line u is derived using v as left assumption”
$R_{u,v}$	:	“line u is derived using v as right assumption”
A_u	:	“line u is active; i.e., actually used in the proof”

Clauses (a sample):

$$\begin{array}{lll}
\overline{A_u} \vee \overline{V_{u,i}} \vee \overline{L_{u,v}} \vee D_{v,i,1} & \overline{A_u} \vee \overline{V_{u,i}} \vee \overline{R_{u,v}} \vee D_{v,i,0} & \overline{D_{s,i,b}} \\
\overline{A_u} \vee \overline{V_{u,i}} \vee \overline{L_{u,v}} \vee A_v & \overline{A_u} \vee \overline{V_{u,i}} \vee \overline{R_{u,v}} \vee A_v & A_s \\
\ldots & &
\end{array}$$

Requirement 1 : The Upper Bound

If F is satisfiable, then $\mathsf{SIZE}_{\mathrm{R}}(\mathsf{REF}_{F,n^c}) \leq \mathrm{poly}(n)$.

Proof idea:

Use a satisfying assignment α of F to nail down the refutation!

Proof sketch:

- Prove that every active line contains a literal satisfied by α.
- Concretely, derive the clauses

$$T_u \quad := \quad \overline{A_u} \vee \bigvee_{i=1}^{n} D_{u,i,\alpha(i)} \qquad \text{for } u = 1, 2, \ldots, L.$$

- Produce empty clause by resolving T_s with A_s and the $\overline{D_{s,i,b}}$.

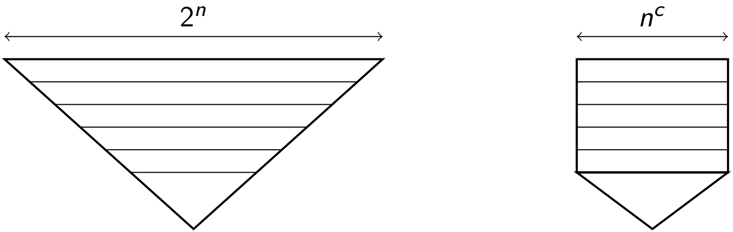
QED

Requirement 2 : The Lower Bound

If F is unsatisfiable, then $\mathrm{SIZE}_{\mathrm{R}}(\mathrm{REF}_{F,n^c}) \not\leq \exp(\Omega(n))$.

Proof idea:

Use a model β^* of $\mathrm{REF}_{F,2^n}$ to construct
a collection of "pseudo-models" β for REF_{F,n^c}.



$\beta^* \models \mathrm{REF}_{F,2^n}$ $\qquad$ $\beta \not\models \neg\mathrm{REF}_{F,n^c}$

The Lower Bound in Three Steps

- Identify a set H of α such that $\mathrm{REF}_{F,s}|_\alpha \cong \mathrm{REF}_{F,s/2}$.
- Here: let α set $1/2$ of all lines as inactive (but not the last).
- And let α also set all other variables of those lines.

- Identify a notion of weak clause made likely true by random α.
- Here: the clauses that mention more than $n/2$ lines.
- Calculation: $\Pr_{\alpha \in H}[C|_\alpha \neq 1] \leq (3/4)^{n/2}$.

- Prove that refutations of $\mathrm{REF}_{F,s/2}$ must contain weak clauses.
- Walk up the dag from empty clause to axioms, and do:
- Sustain a matching between active lines and the lines in β^*.
- The corresponding assignments are the "pseudo-models" β.

QED

Reminder

Theorem
Resolution is not automatable in time $T(s)$, [poly-time]
unless n-variable SAT is solvable in time $T(\mathrm{poly}(n))$ [P $=$ NP].

✓

Theorem
Tree-Like Resolution is not automatable in time $T(s) = s^{o(\log s)}$,
unless n-variable SAT is solvable in randomized time $2^{o(n)}$.

The Tree-Like Case

We want a reduction:

from n-variable SAT
to min proof-size approximation for tree-like R (called R^*).

$$F \xrightarrow{\exp(o(n)) \text{ time}} G_F$$

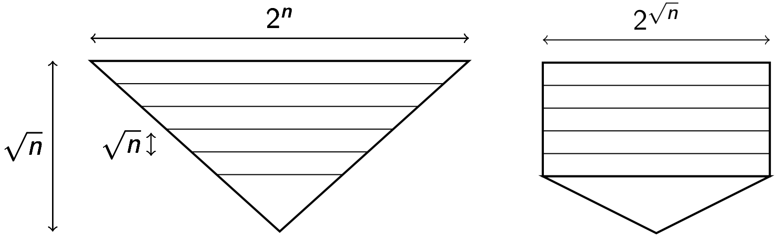
Requirements:

1. If F is satisfiable, then $\mathrm{SIZE}_{R^*}(G_F) \leq \exp(O(\sqrt{n}))$.
2. If F is unsatisfiable, then $\mathrm{SIZE}_{R^*}(G_F) \not\leq \exp(\Omega(n))$.

Modification of the Formula G_F : Shallow REF

Key Observation:

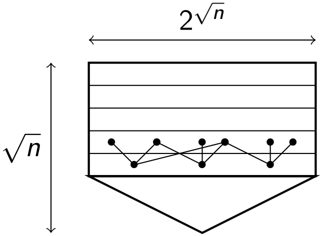
In the "F is unsatisfiable" case,
the model β^* of $\mathrm{REF}_{F,2^n}$ happens to be:
tree-like and layered, and have depth n.



$\beta^* \models \mathrm{REF}_{F,2^n}$ $\quad$ $\beta \not\models \neg\mathrm{REF}_{F,2^{\sqrt{n}}}$

Modification of the Formula in More Details

- Modify the formula G_F; now $\mathrm{REF}_{F,s}|_\gamma$ with $s = 2^{\sqrt{n}}$.
- The γ restricts A, D, I, V, L, R in a way compatible with β^*:
- Instead of arbitrary dag-depth, impose depth n.
- Instead of arbitrary structure, impose $\sqrt{n}$ layers of depth $\sqrt{n}$.
- Instead of $\mathrm{poly}(n)$-size layers, allow layers of size $2^{\sqrt{n}}$.
- Instead of full connectivity between layers, place expanders.
- Their bounded degree d ensures tree-like size $d^{\sqrt{n}} = 2^{O(\sqrt{n})}$.
- Their expansion property ensures matchability with β^*.



Reminder

Theorem
Resolution is not automatable in time $T(s)$, [poly-time]
unless n-variable SAT is solvable in time $T(\mathrm{poly}(n))$ [P $=$ NP]. ✓

Theorem
Tree-Like Resolution is not automatable in time $T(s) = s^{o(\log s)}$,
unless n-variable SAT is solvable in randomized time $2^{o(n)}$. ✓

Big Remaining Open Problem

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

Notes:

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

Notes:

- Automatability is about short vs. not short refutability.

Big Remaining Open Problem

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

Notes:

- Automatability is about short vs. not short refutability.
- Weak automatability is about short vs. impossible refutability.

Big Remaining Open Problem

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

Notes:

- Automatability is about short vs. not short refutability.
- Weak automatability is about short vs. impossible refutability.
- Therefore: it cannot be harder than NP $\cap$ co-NP.

Big Remaining Open Problem

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

Notes:

- Automatability is about short vs. not short refutability.
- Weak automatability is about short vs. impossible refutability.
- Therefore: it cannot be harder than NP $\cap$ co-NP.
- For Resolution, the problem is PARITY GAMES hard [BPT].

WEAK AUTOMATABILITY OF RESOLUTION?

For Resolution specifically:

Is it computationally feasible to
distinguish satisfiable formulas
from shortly refutable formulas?

Notes:

- Automatability is about short vs. not short refutability.
- Weak automatability is about short vs. impossible refutability.
- Therefore: it cannot be harder than NP $\cap$ co-NP.
- For Resolution, the problem is PARITY GAMES hard [BPT].
- For (Extended) Frege, the problem is RSA-hard [KP,BPR].

THE END