Lecture 6: Algorithms!

Sep 22

How hard is it (algorithmically) to actually find short resolution proofs?

Unless P = NP, there's no proof system that both

- has short proofs always, and
- we can efficiently find short proofs in the system

Fact: All of the best practical SAT solvers generate resolution* proofs when their inputs are unsatisfiable.

How do we formalize the first question?

Search: Given an unsat CNF F, output the shortest res. proof of F

Optimization: Given unsat F, approximate the length of the shortest resolution proof.

Not really known: Are there good search-to-decision reductions for this problem?

Not reasonable to ask if Search has a polynomial time algorithm because of potentially large output.

Should say: algorithm is polynomial in the output length.

Defn (Automatizability / Automatability)

Given an unsat CNF formula F, output a resolution proof of F in time $\mathrm{poly}(|F| + S_{Res}(F))$.

What's known? Sep 22

[Beame-Pitassi 96] Tree-like resolution is automatizable in quasi-polynomial time.

Formally: there is an automatizing alg. in time

$$2^{\log^{O(1)}(|F| + S^{T}_{Res}(F))}$$

For DAG-like resolution:

$$w_{Res}(F) = O\left(\sqrt{n \log S_{Res}(F)}\right) \quad (*)$$

Assume F is a 3-CNF, then just derive all clauses of width $w = 3, 4, \cdots$ until you derive $\bot$.

By width-size relation, stop when $w \leq (*)$, $n^{O(w)}$ time.

$\therefore$ Alg. in time $n^{O(\sqrt{n \log S_{Res}(F)})}$. $n^{\sqrt{n}} = 2^{\sqrt{n}\log n} \ll 2^n$

On the other hand, there is a number of complexity-theoretic results.

[ABMP 98, 01] (P: Moran) Optimization is NP-Hard to approximate even within a multiplicative factor

$$2^{\log^{1-o(1)} n}.$$

In other words: NP-Hard to decide if the smallest resolution refutation of F has

Frege, Extended-Frege, and others

$$S_{Res}(F) \leq s \quad \text{or} \quad S_{Res}(F) \geq 2^{\log^{1-o(1)} n} s.$$

[Alekhnovich–Razborov 01, 08]

$FPT \neq W[P]$

Assuming (some reasonable conjecture in parametrized complexity theory) then

(Tree-)Resolution is <u>not</u> automatizable

[Atserias–Müller 19, FOCS Best paper]

It is NP-Hard to automatize resolution.

<u>Pf Idea</u> "Reduction" from 3-SAT

Given 3-CNF F, create polynomial time algorithm A that outputs another 3-CNF formula $A(F)$ s.t.

If F is SAT, then $A(F)$ has res. ref. of length $\leq s$

If F is UNSAT, then $S_{Res}(A(F)) \gg \text{poly}(s)$.

$\square$

<u>Next</u> How do SAT algs actually work?

Use a top-down definition of Resolution

<u>Prover–Delayer Games</u>

Let F be an unsat CNF formula. Describe a 2-player game

Players: Prover, Delayer

State: Set S of literals from F

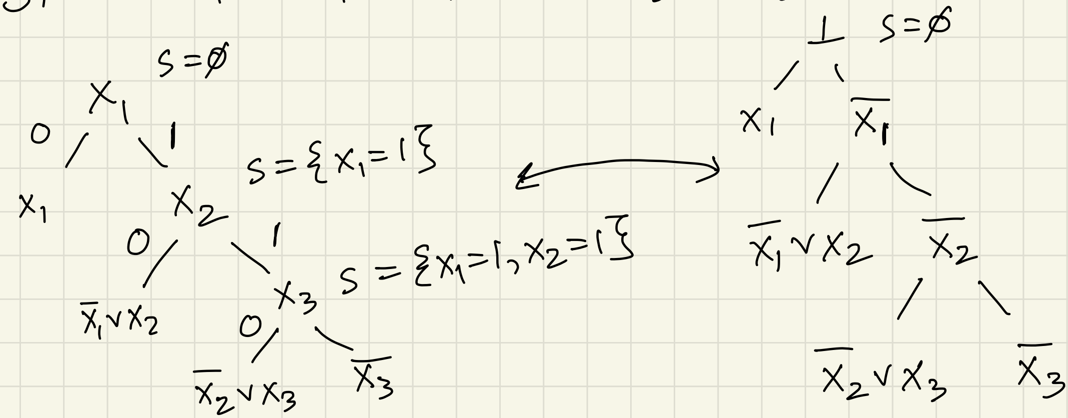
Provers goal: Find an assignment that falsifies a clause from F.

Delayers goal: Stop the Prover.

Game played in rounds. In each round:

- Prover picks a variable x that is not in S
- Delayer picks $b \in \{0,1\}$, then literal x^b $(x=b)$ is added to S.
- The Prover can select $I \subseteq [n]$ and delete all corresponding literals from the state.

e.g.) $F = x_1 \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge \bar{x}_3$



Clear bijection between Prover strategies and Resolution proofs!

PD Game for PHP^{n+1}_n

Sep 22

Prover consider each pigeon $i=1,\dots,n+1$ in sequence.
Prover remembers a partial matching between pigeons and holes.

For pigeon i:

Prover queries $x_{i1}, x_{i2}, \dots, x_{in}$ in sequence.

If Delayer always answers 0, then pigeon axiom is falsified.

If Delayer matches pigeon i to a hole j that is already occupied, hole$_j$ axiom is falsified.

Else, prover remembers the hole that Delayer matches pigeon i with and continues on to pigeon $i+1$.

Strategies for the Prover $\equiv$ Upper bounds (i.e. proofs) in Resolution

Strategies for the Delayer $\equiv$ Lower bounds for Resolution

Next time:

- Complexity measures for PD games
- Nice method for lower bounds on $S^T_{Res}(F)$
- SAT algorithms.