OBDD proofs are NP-hard to automate

MFCS 2022, Vienna

Dmitry Itsykson, Artur Riazanov

PDMI RAS

August 23, 2022

Solving SAT

- $\mathcal{S}$: algorithm finding solutions for SAT.
- If ϕ is satisfiable $\mathcal{S}(\phi)$ returns a satisfying assignment.
- If ϕ is unsatisfiable the execution log of $\mathcal{S}(\phi)$ is a **certificate** of $\phi \in$ UNSAT.

Solving SAT

- $\mathcal{S}$: algorithm finding solutions for SAT.
- If ϕ is satisfiable $\mathcal{S}(\phi)$ returns a satisfying assignment.
- If ϕ is unsatisfiable the execution log of $\mathcal{S}(\phi)$ is a **certificate** of $\phi \in \textsc{Unsat}$.

Going backwards

Resolution [Robinson, 1965]

ϕ: CNF formula. Resolution proof: a sequence $C_1, \ldots, C_m = \bot$ of clauses such that $C_i \in \phi$ or C_i is obtained by the resolution rule from the previous clauses.

$$\frac{A \vee x \qquad B \vee \neg x}{A \vee B}$$

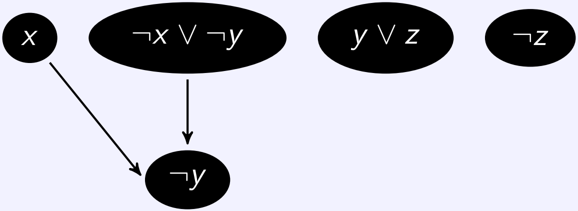
Example

$$\phi = x \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge \neg z.$$

Going backwards

Example

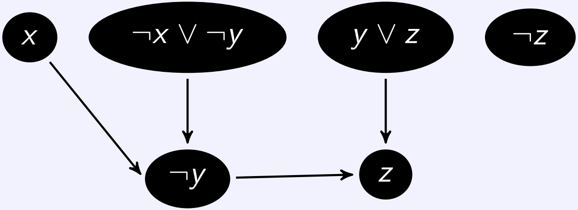
$$\phi = x \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge \neg z.$$



Going backwards

Example

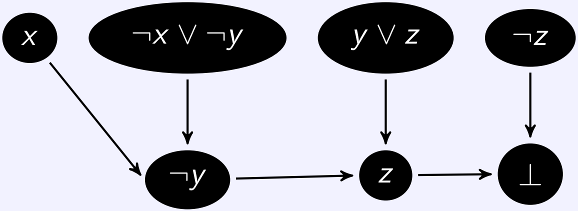
$$\phi = x \land (\neg x \lor \neg y) \land (y \lor z) \land \neg z.$$



Going backwards

Example

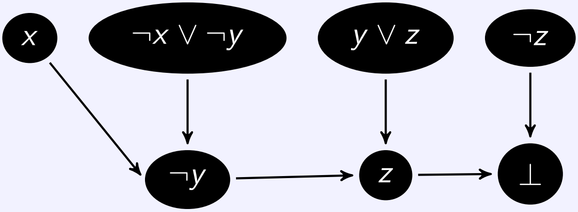
$$\phi = x \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge \neg z.$$



Going backwards

Example

$$\phi = x \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge \neg z.$$



Proposition

If there exists a resolution refutation of ϕ with clauses of width at most w, then it can be found in $n^{O(w)}$.

Automatability

Definition

A proof system Π for UNSAT is automatable is there exists an algorithm which can produce Π-refutations of a given CNF ϕ in poly time in ϕ and the length of the shortest Π-refutation of ϕ.

Theorem [Atserias, Muller, 2019]

Suppose Resolution is automatable. Then $\mathbf{P} = \mathbf{NP}$.

- NS, PC [GNPRSdR21];
- Res(k) [Gar20];
- Cutting Planes [GKMP20].

Automatability

Definition

A proof system Π for UNSAT is automatable is there exists an algorithm which can produce Π-refutations of a given CNF ϕ in poly time in ϕ and the length of the shortest Π-refutation of ϕ.

Theorem [Atserias, Muller, 2019]

Suppose Resolution is automatable. Then $\mathbf{P} = \mathbf{NP}$.

- NS, PC [GNPRSdR21];
- Res(k) [Gar20];
- Cutting Planes [GKMP20].

Automatability

Definition

A proof system Π for UNSAT is automatable is there exists an algorithm which can produce Π-refutations of a given CNF ϕ in poly time in ϕ and the length of the shortest Π-refutation of ϕ.

Theorem [Atserias, Muller, 2019]

Suppose Resolution is automatable. Then $\mathbf{P} = \mathbf{NP}$.

- NS, PC [GNPRSdR21];
- Res(k) [Gar20];
- Cutting Planes [GKMP20].

Automatability

Definition

A proof system Π for UNSAT is automatable is there exists an algorithm which can produce Π-refutations of a given CNF ϕ in poly time in ϕ and the length of the shortest Π-refutation of ϕ.

Theorem [Atserias, Muller, 2019]

Suppose Resolution is automatable. Then $\mathbf{P} = \mathbf{NP}$.

- NS, PC [GNPRSdR21];
- Res(k) [Gar20];
- Cutting Planes [GKMP20].

Automatability

Definition

A proof system Π for UNSAT is automatable is there exists an algorithm which can produce Π-refutations of a given CNF ϕ in poly time in ϕ and the length of the shortest Π-refutation of ϕ.

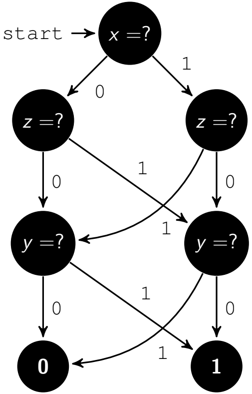
Theorem [Atserias, Muller, 2019]

Suppose Resolution is automatable. Then $\mathbf{P} = \mathbf{NP}$.

- NS, PC [GNPRSdR21];
- Res(k) [Gar20];
- Cutting Planes [GKMP20].

OBDD

- π-OBDD represents a Boolean function f;
- If D_1 and D_2 are OBDDs in the same order, can construct $D_1 \circ D_2$ for any Boolean $\circ$ in time $\mathcal{O}(|D_1||D_2|)$;
- If D_1 and D_2 are OBDDs in the same order, can check if $D_1 \equiv D_2$ in time $\mathcal{O}(|D_1||D_2|)$.



$f(x,y,z) = x \oplus y \oplus z$, order π is (x,z,y)

OBDD proof system

OBDD refutations

OBDD refutation of an unsat CNF F is a sequence of OBDDs **with the same order of variables** terminating with an OBDD computing identical False, such that each of the OBDDs either:

- encodes a clause of F;
- **semantically follows** from two preceding OBDDs in the sequence.

Theorem

OBDD proof system is not automatable unless $\mathbf{P} = \mathbf{NP}$.

OBDD proof system

OBDD refutations

OBDD refutation of an unsat CNF F is a sequence of OBDDs **with the same order of variables** terminating with an OBDD computing identical False, such that each of the OBDDs either:

- encodes a clause of F;
- **semantically follows** from two preceding OBDDs in the sequence.

Theorem

OBDD proof system is not automatable unless $\mathbf{P} = \mathbf{NP}$.

Atserias and Muller's proof

Resolution is NP-hard to automate:

- $\mathcal{A}$ poly-automates Resolution.
- Let's solve 3-SAT.
- Construct an algorithm $\mathcal{T}$, that given a satisfiable CNF constructs unsatisfiable CNF that is easy to refute;
- And given an unsatisfiable CNF constructs unsatisfiable CNF that is hard to refute;
- Now to solve 3-SAT simply run $\mathcal{A}(\mathcal{T}(\phi))$ for $n^{O(1)}$ steps and accept iff it terminates.

Atserias and Muller's proof

Resolution is NP-hard to automate:

- $\mathcal{A}$ poly-automates Resolution.
- Let's solve 3-SAT.
- Construct an algorithm $\mathcal{T}$, that given a satisfiable CNF constructs unsatisfiable CNF that is easy to refute;
- And given an unsatisfiable CNF constructs unsatisfiable CNF that is hard to refute;
- Now to solve 3-SAT simply run $\mathcal{A}(\mathcal{T}(\phi))$ for $n^{O(1)}$ steps and accept iff it terminates.

Atserias and Muller's proof

Resolution is NP-hard to automate:

- $\mathcal{A}$ poly-automates Resolution.
- Let's solve 3-SAT.
- Construct an algorithm $\mathcal{T}$, that given a satisfiable CNF constructs unsatisfiable CNF that is easy to refute;
- And given an unsatisfiable CNF constructs unsatisfiable CNF that is hard to refute;
- Now to solve 3-SAT simply run $\mathcal{A}(\mathcal{T}(\phi))$ for $n^{O(1)}$ steps and accept iff it terminates.

Adapting for Cutting Planes

Magic transformation

$\mathcal{L}$ is a CNF-to-CNF mapping such that $\mathcal{L}(\phi)$ is hard for a proof system Π iff ϕ is hard for resolution.

ϕ $\xrightarrow{\text{SAT}}$ $\mathrm{Res}(\mathcal{T}(\phi)) = n^{O(1)}$

ϕ $\xrightarrow{\text{UNSAT}}$ $\mathrm{Res}(\mathcal{T}(\phi)) = 2^{n^{\Omega(1)}}$

Adapting for Cutting Planes

Magic transformation

$\mathcal{L}$ is a CNF-to-CNF mapping such that $\mathcal{L}(\phi)$ is hard for a proof system Π iff ϕ is hard for resolution.

$$\begin{tikzcd} & \boxed{\mathrm{Res}(\mathcal{T}(\phi)) = n^{O(1)}} \\ \phi \arrow[ur, dashed, "\text{SAT}"] \arrow[dr, dashed, "\text{UNSAT}"'] & \\ & \boxed{\mathrm{Res}(\mathcal{T}(\phi)) = 2^{n^{\Omega(1)}}} \end{tikzcd}$$

Adapting for Cutting Planes

Magic transformation

$\mathcal{L}$ is a CNF-to-CNF mapping such that $\mathcal{L}(\phi)$ is hard for a proof system Π iff ϕ is hard for resolution.

$$\begin{tikzcd}
 & \mathrm{Res}(\mathcal{T}(\phi)) = n^{O(1)} \arrow[r, "\mathcal{L}"] & \Pi(\mathcal{L}(\mathcal{T}(\phi))) = n^{O(1)} \\
\phi \arrow[ur, dashed, "\text{SAT}"] \arrow[dr, dashed, "\text{UNSAT}"'] & & \\
 & \mathrm{Res}(\mathcal{T}(\phi)) = 2^{n^{\Omega(1)}} \arrow[r, "\mathcal{L}"] & \Pi(\mathcal{L}(\mathcal{T}(\phi))) = 2^{n^{\Omega(1)}}
\end{tikzcd}$$

Adapting for Cutting Planes

Theorem [GGKS18]

If a CNF F over n variables requires a resolution width w, then $F \circ g$ requires cutting planes size $n^{\Omega(w)}$.

$$\begin{tikzcd}
 & w(\mathcal{T}(\phi)) = n^{\Omega(1)} \arrow[r, "\circ g"] & \mathrm{CP}(\mathcal{T}(\phi) \circ g) = 2^{n^{\Omega(1)}} \\
\phi \arrow[ur, dashed, "\text{SAT}"] \arrow[dr, dashed, "\text{UNSAT}"'] & & \\
 & w(\mathcal{T}(\phi)) = n^{\Omega(1)} \arrow[r, "\circ g"] & \mathrm{CP}(\mathcal{T}(\phi) \circ g) = 2^{n^{\Omega(1)}}
\end{tikzcd}$$

Adapting for Cutting Planes

Theorem [GGKS18]

If a CNF F over n variables requires a resolution width w, then $F \circ g$ requires cutting planes size $n^{\Omega(w)}$.

$\phi \xrightarrow{\text{SAT}} w(\mathcal{T}(\phi)) = n^{\Omega(1)} \xrightarrow{\circ g} \mathrm{CP}(\mathcal{T}(\phi) \circ g) = 2^{n^{\Omega(1)}}$

$\phi \xrightarrow{\text{UNSAT}} w(\mathcal{T}(\phi)) = n^{\Omega(1)} \xrightarrow{\circ g} \mathrm{CP}(\mathcal{T}(\phi) \circ g) = 2^{n^{\Omega(1)}}$

Adapting for Cutting Planes

Theorem [GGKS18]

If a CNF F over n variables requires a resolution width w, then $F \circ g$ requires cutting planes size $n^{\Omega(w)}$.

- Blockwidth: consider a partition of variables, then blockwidth of a clause is the number of blocks in the partition mentioned in it.
- [AM19] reduction transforms sat instances into $O(1)$-blockwidth, unsat into $n^{\Omega(1)}$-blockwidth.
- [GKMP20] give a lifting theorem from blockwidth to cutting planes size.

Adapting for Cutting Planes

Theorem [GGKS18]

If a CNF F over n variables requires a resolution width w, then $F \circ g$ requires cutting planes size $n^{\Omega(w)}$.

- Blockwidth: consider a partition of variables, then blockwidth of a clause is the number of blocks in the partition mentioned in it.
- [AM19] reduction transforms sat instances into $O(1)$-blockwidth, unsat into $n^{\Omega(1)}$-blockwidth.
- [GKMP20] give a lifting theorem from blockwidth to cutting planes size.

Adapting for Cutting Planes

Theorem [GGKS18]

If a CNF F over n variables requires a resolution width w, then $F \circ g$ requires cutting planes size $n^{\Omega(w)}$.

- Blockwidth: consider a partition of variables, then blockwidth of a clause is the number of blocks in the partition mentioned in it.
- [AM19] reduction transforms sat instances into $O(1)$-blockwidth, unsat into $n^{\Omega(1)}$-blockwidth.
- [GKMP20] give a lifting theorem from blockwidth to cutting planes size.

Plan of the proof

Theorem

OBDD proof system is not automatable unless $\mathbf{P} = \mathbf{NP}$.

- Start with [AM19];
- Modify lifting theorem from [GKMP20] for it to work for OBDDs;
- Apply (massaged) Segerlind's [Seg08] transformation to factor off the problem of variable ordering.

$$\begin{tikzcd} & \phi \arrow[dl, "\textsc{Sat}"'] \arrow[dr, "\textsc{Unsat}"] & \\ bw(\mathcal{T}(\phi)) = O(1) & & bw(\mathcal{T}(\phi)) = n^{\Omega(1)} \end{tikzcd}$$

Plan of the proof

- Start with [AM19];
- Modify lifting theorem from [GKMP20] for it to work for OBDDs;
- Apply (massaged) Segerlind's [Seg08] transformation to factor off the problem of variable ordering.

ϕ

SAT: $bw(\mathcal{T}(\phi)) = O(1)$ $\xrightarrow{\circ g}$ $OBDD_\pi(\mathcal{T}(\phi) \circ g) = n^{O(1)}$

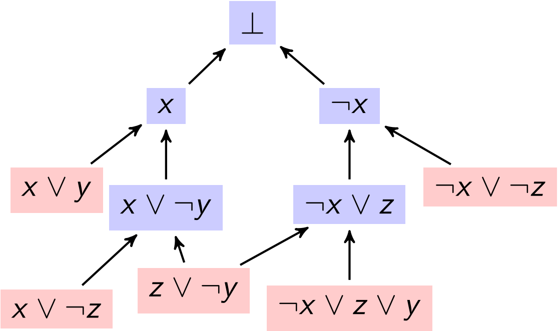
UNSAT: $bw(\mathcal{T}(\phi)) = n^{\Omega(1)}$ $\xrightarrow{\circ g}$ $OBDD_\pi(\mathcal{T}(\phi) \circ g) = 2^{n^{\Omega(1)}}$

Plan of the proof

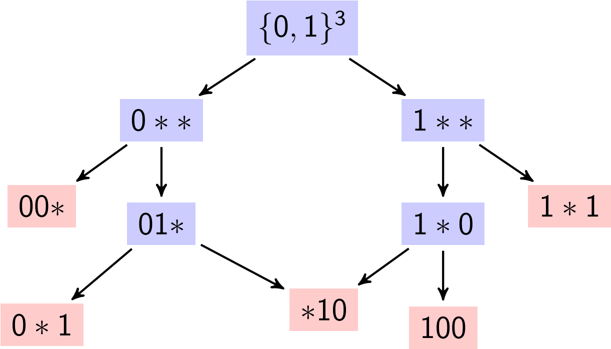
- Start with [AM19];
- Modify lifting theorem from [GKMP20] for it to work for OBDDs;
- Apply (massaged) Segerlind's [Seg08] transformation to factor off the problem of variable ordering.

$$
\begin{tikzcd}
 & \phi \arrow[dl, "\textsc{Sat}"'] \arrow[dr, "\textsc{Unsat}"] & \\
bw(\mathcal{T}(\phi)) = O(1) \arrow[d, "\circ g"] & & bw(\mathcal{T}(\phi)) = n^{\Omega(1)} \arrow[d, "\circ g"] \\
OBDD_{\pi}(\mathcal{T}(\phi) \circ g) = n^{O(1)} \arrow[d, "\mathcal{S}"] & & OBDD_{\pi}(\mathcal{T}(\phi) \circ g) = 2^{n^{\Omega(1)}} \arrow[d, "\mathcal{S}"] \\
OBDD(\mathcal{S}(\mathcal{T}(\phi) \circ g)) = n^{O(1)} & & OBDD(\mathcal{S}(\mathcal{T}(\phi) \circ g)) = 2^{n^{\Omega(1)}}
\end{tikzcd}
$$

Proofs as Protocols



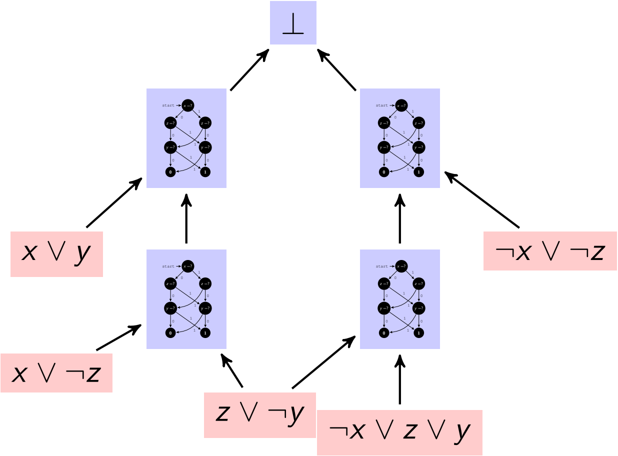
Proofs as Protocols



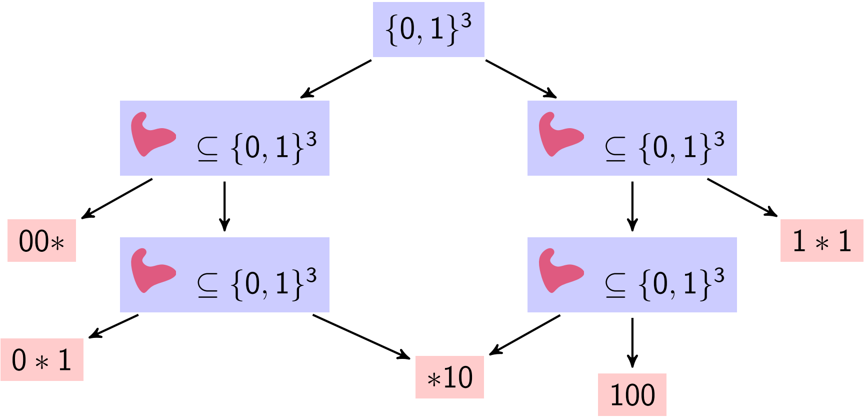
Example: $0** = \{000, 001, 010, 011\}$.

Set in a node is **covered** by the sets in its predecessors.

Proofs as Protocols



Proofs as Protocols



Summary

- Resolution can be converted to a protocol of Boolean **subcubes** or sets of small **query complexity**.
- OBDD refutation is a protocol of sets of small **communication complexity** (essentially [Kra06], explicitly [BIKS18]).

Proposition

OBDD refutation is a protocol of sets of small $o(n)$-party **number in hand** communication complexity.

Summary

- Resolution can be converted to a protocol of Boolean **subcubes** or sets of small **query complexity**.
- OBDD refutation is a protocol of sets of small **communication complexity** (essentially [Kra06], explicitly [BIKS18]).

Proposition

OBDD refutation is a protocol of sets of small $o(n)$-party **number in hand** communication complexity.

Lifting

Query complexity

$f\colon \{0,1\}^n \to \{0,1\}$. How many bits one needs to probe to learn the value of f.

Communication complexity

$f\colon X \times Y \to Z$. Alice knows $x \in X$, Bob knows $y \in Y$, how many bits they need to exchange to learn $f(x,y)$?

Theorem [Raz, McKenzie 1997], [GPW15]

If query complexity of f is d, then communication complexity of $f \circ g$ is at least d for some appropriate g.

Dag-like version

Theorem [GGKS18]

If a CNF F over n variables requires a **resolution width** w, then $F \circ g$ requires a dag-like communication protocol of size $n^{\Omega(w)}$.

Lifting from blockwidth

Theorem [GKMP20]

If a CNF F over n variables requires a **resolution blockwidth** w, then $F \circ g$ requires a $(n+1)$-party dag-like communication protocol of size $n^{\Omega(w)}$.

Theorem (this work)

If a CNF F over n variables requires a **resolution blockwidth** w, then $F \circ g$ requires a $o(n)$-party dag-like communication protocol of size $n^{\Omega(w)}$.

Lifting from blockwidth

Theorem [GKMP20]

If a CNF F over n variables requires a **resolution blockwidth** w, then $F \circ g$ requires a $(n+1)$-party dag-like communication protocol of size $n^{\Omega(w)}$.

Theorem (this work)

If a CNF F over n variables requires a **resolution blockwidth** w, then $F \circ g$ requires a $o(n)$-party dag-like communication protocol of size $n^{\Omega(w)}$.

Proof highlights

- Coarser input partition then in [GKMP20];
- Reduce to 2-party lifting in the key richness lemma.
- Small changes in Segerlind's transformation to make in work for essentially arbitrary CNFs.

Open questions

- Is OBDD($\wedge$) automatable? It does not simulate resolution, so the upper bound fails.
- Can we randomize number-in-hand multiparty dag-like lifting?