

Lower Bound Techniques for QBF Proof Systems

Meena Mahajan



The Institute of Mathematical Sciences, HBNI, Chennai.

Proof Complexity Workshop, FLoC 2018, Oxford, UK.
7,8 July, 2018

My work on QBF proof complexity –

- partially supported by the EU Marie Curie IRSES grant CORCON.
- joint work with
 - Olaf Beyersdorff Univ of Leeds, UK
 - Leroy Chew Univ of Leeds, UK
 - Anil Shukla formerly at IMSc, Chennai
 now at IIT Ropar

QBF Proof systems

- What are they?
- Why do we study them?

Satisfiability

- SAT: Satisfiability.
eg. Is there an assignment to x, y, z satisfying all the clauses
 $(x \vee y \vee z), (x \vee \neg y \vee \neg z), (\neg x \vee y \vee \neg z), (\neg x \vee \neg y \vee z)$?
- Quintessential NP-complete problem.
- Very hard – in theory.

Satisfiability

- SAT: Satisfiability.
eg. Is there an assignment to x, y, z satisfying all the clauses
 $(x \vee y \vee z), (x \vee \neg y \vee \neg z), (\neg x \vee y \vee \neg z), (\neg x \vee \neg y \vee z)$?
- Quintessential NP-complete problem.
- Very hard – in theory.
In practice – a solved problem! Many good SAT solvers around.

Satisfiability

- SAT: Satisfiability.
eg. Is there an assignment to x, y, z satisfying all the clauses $(x \vee y \vee z), (x \vee \neg y \vee \neg z), (\neg x \vee y \vee \neg z), (\neg x \vee \neg y \vee z)$?
- Quintessential NP-complete problem.
- Very hard – in theory.
In practice – a solved problem! Many good SAT solvers around.
- Ambitious programs to design good solvers for problems harder than SAT.

- QBF: Quantified Boolean Formula

Subsumes SAT. eg. Is this QBF true?

$$\exists x \exists y \exists z (x \vee y \vee z), (x \vee \neg y \vee \neg z), (\neg x \vee y \vee \neg z), (\neg x \vee \neg y \vee z)$$

- PSPACE-complete, so much more expressive than SAT.

eg. Is this formula true?

$$\exists e \forall u \exists c \exists d (\neg e \vee c)(e \vee d)(\neg u \vee c)(u \vee d)(\neg c \vee \neg d)$$

- Quite a few QBF solvers developed in the last couple of decades.

Improving solvers

- How to improve the performance of a solver?
- Understand where it flounders.

Improving solvers

- How to improve the performance of a solver?
- Understand where it flounders.
- Underlying solver heuristics are formal proof systems: Runs of SAT/QBF solver provide proofs of unsatisfiability/falsity.
- Lower bounds in formal proof system
(no short proof of unsat/falsity)
 ⇓
no short runs.

Improving solvers

- How to improve the performance of a solver?
- Understand where it flounders.
- Underlying solver heuristics are formal proof systems: Runs of SAT/QBF solver provide proofs of unsatisfiability/falsity.
- Lower bounds in formal proof system
(no short proof of unsat/falsity)
 \Downarrow
 no short runs.
- Proving lower bounds – **back to theory!**

The Resolution Proof System for UNSAT

\mathcal{C} : bag of clauses. \tilde{a} : assignment to the variables.

If \tilde{a} satisfies

$$\mathcal{C} = \begin{array}{c} \vdots \\ A \vee x \\ B \vee \neg x \\ \vdots \end{array}$$

Then \tilde{a} satisfies

$$\mathcal{C}' = \begin{array}{c} \vdots \\ A \vee x \\ B \vee \neg x \\ \vdots \\ A \vee B \end{array}$$

The Resolution Proof System for UNSAT

\mathcal{C} : bag of clauses. \tilde{a} : assignment to the variables.

If \tilde{a} satisfies

$$\mathcal{C} = \begin{array}{c} \vdots \\ A \vee x \\ B \vee \neg x \\ \vdots \end{array}$$

Then \tilde{a} satisfies

$$\mathcal{C}' = \begin{array}{c} \vdots \\ A \vee x \\ B \vee \neg x \\ \vdots \\ A \vee B \end{array}$$

$$\mathcal{C}_0 \in \text{SAT} \implies \mathcal{C}_1 \in \text{SAT} \implies \dots \implies \mathcal{C}_{t-1} \in \text{SAT} \implies \mathcal{C}_t \in \text{SAT}$$

$$\mathcal{C}_0 \notin \text{SAT} \Leftarrow \dots \Leftarrow \mathcal{C}_i \notin \text{SAT} \Leftarrow \dots \Leftarrow \mathcal{C}_t \notin \text{SAT} \Leftarrow \square \in \mathcal{C}_t$$

Extending Resolution to QBFs

QBFs: Quantified Boolean Formulas

- W.l.o.g., QBF in prenex CNF: $Q\vec{x} \cdot F(\vec{x})$; F a set of clauses.
- Resolution is **sound**: If $Q\vec{x} \cdot F(x)$ is true, and we add a clause C to F through resolution to get F' , then $Q\vec{x} \cdot F'(x)$ is also true.

Extending Resolution to QBFs

QBFs: Quantified Boolean Formulas

- W.l.o.g., QBF in prenex CNF: $Q\vec{x} \cdot F(\vec{x})$; F a set of clauses.
- Resolution is **sound**: If $Q\vec{x} \cdot F(x)$ is true, and we add a clause C to F through resolution to get F' , then $Q\vec{x} \cdot F'(x)$ is also true.
- But Resolution alone is not enough. Consider

$$\exists x \forall u \quad (x \vee \neg u) \quad (\neg x \vee u).$$

Resolution can add $(x \vee \neg x)$ or $(u \vee \neg u)$. Useless.

- Universal variable u has to be handled differently.
- Two ways to proceed, modelling
 - CDCL-based solvers
 - expansion-based solvers

The Evaluation Game on QBFs

- QBF $Q\vec{x} \cdot F(x)$
- Two players, **Red** and **Blue**, step through quantifier prefix left-to-right. **Red** picks values for \exists variables, **Blue** for \forall variables.

Assignment constructed: \tilde{a} .

Red wins a run of the game if $F(\tilde{a})$ true. Otherwise **Blue** wins.

The Evaluation Game on QBFs

- QBF $Q\vec{x} \cdot F(x)$
- Two players, **Red** and **Blue**, step through quantifier prefix left-to-right. **Red** picks values for \exists variables, **Blue** for \forall variables.

Assignment constructed: \tilde{a} .

Red wins a run of the game if $F(\tilde{a})$ true. Otherwise **Blue** wins.

- example:

$$\exists x \forall u \quad (x \vee \neg u) \quad (\neg x \vee u).$$

The Evaluation Game on QBFs

- QBF $Q\vec{x} \cdot F(x)$
- Two players, **Red** and **Blue**, step through quantifier prefix left-to-right. **Red** picks values for \exists variables, **Blue** for \forall variables.

Assignment constructed: \tilde{a} .

Red wins a run of the game if $F(\tilde{a})$ true. Otherwise **Blue** wins.

- example:

$$\exists x \forall u \quad (x \vee \neg u) \quad (\neg x \vee u).$$

Red: $x = 1$, **Blue**: $u = 1$: **Red** wins

Red: $x = 1$, **Blue**: $u = 0$: **Blue** wins

Red: $x = 0$, **Blue**: $u = 1$: **Blue** wins

The Evaluation Game on QBFs

- QBF $Q\vec{x} \cdot F(x)$
- Two players, **Red** and **Blue**, step through quantifier prefix left-to-right. **Red** picks values for \exists variables, **Blue** for \forall variables.

Assignment constructed: \tilde{a} .

Red wins a run of the game if $F(\tilde{a})$ true. Otherwise **Blue** wins.

- example:

$$\exists x \forall u \quad (x \vee \neg u) \quad (\neg x \vee u).$$

Red: $x = 1$, **Blue**: $u = 1$: **Red** wins

Red: $x = 1$, **Blue**: $u = 0$: **Blue** wins

Red: $x = 0$, **Blue**: $u = 1$: **Blue** wins

Blue can always win: set $u \neq x$.

The Evaluation Game on QBFs

- QBF $Q\vec{x} \cdot F(x)$
- Two players, **Red** and **Blue**, step through quantifier prefix left-to-right. **Red** picks values for \exists variables, **Blue** for \forall variables.

Assignment constructed: \tilde{a} .

Red wins a run of the game if $F(\tilde{a})$ true. Otherwise **Blue** wins.

- example:

$$\exists x \forall u \quad (x \vee \neg u) \quad (\neg x \vee u).$$

Red: $x = 1$, **Blue**: $u = 1$: **Red** wins

Red: $x = 1$, **Blue**: $u = 0$: **Blue** wins

Red: $x = 0$, **Blue**: $u = 1$: **Blue** wins

Blue can always win: set $u \neq x$.

- $Q\vec{x} \cdot F(x)$ false if and only if **Blue** has a winning strategy.
- Use this to extend Resolution.

The \forall reduction rule

Consider this scenario:

- $Q\vec{x} \cdot F(x)$ is true. So **Red** has a winning strategy.
- $F(x)$ has a clause C in which the **rightmost** variable (as per $Q\vec{x}$) is a universal variable u .
i.e. $C = A \vee \ell$; $\ell \in \{u, \neg u\}$; all variables in A are left of u .

The \forall reduction rule

Consider this scenario:

- $Q\vec{x} \cdot F(x)$ is true. So **Red** has a winning strategy.
- $F(x)$ has a clause C in which the **rightmost** variable (as per $Q\vec{x}$) is a universal variable u .
i.e. $C = A \vee \ell$; $\ell \in \{u, \neg u\}$; all variables in A are left of u .

Then, by the time **Blue** has to fix u , **Red**'s strategy must ensure that sub-clause A is already satisfied.

That is, **Red** has a winning strategy on $Q\vec{x} \cdot [F(x) \wedge A]$.
So $Q\vec{x} \cdot [F(x) \wedge A]$ is also true.

The proof system $QU\text{-Res} = \text{Res} + \forall\text{Red}$

$$Q\vec{x} \cdot \mathcal{C}$$

Grow the bag of clauses \mathcal{C} using

- Resolution: If $A \vee x$ and $B \vee \neg x$ are in the bag, can add $A \vee B$ (provided not a tautology),
- \forall -Reduction: If $A \vee \ell(u)$ in the bag, and all variables in A left of u , can add A ,

until the empty clause \square is added.

The proof system QU-Res (cont'd)

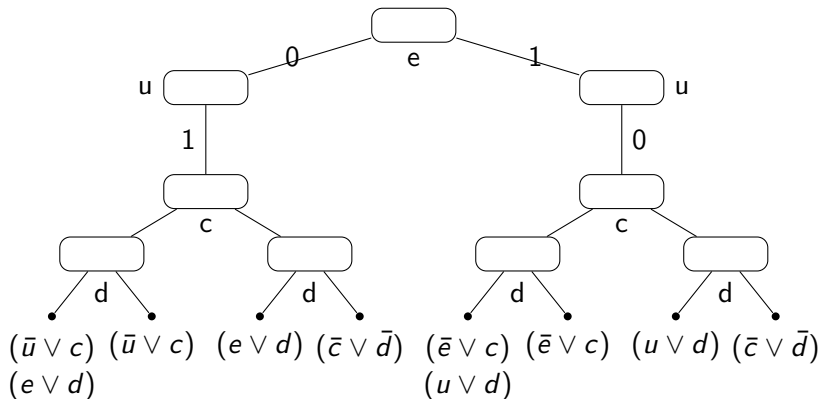
- **Sound:** A derivation of \square reveals a winning strategy for **Blue**.
[vanGelder 2012]
- **Complete:** Use a winning strategy of **Blue** to decide which clauses to derive.

The proof system QU-Res (cont'd)

- **Sound:** A derivation of \square reveals a winning strategy for **Blue**.
[vanGelder 2012]
- **Complete:** Use a winning strategy of **Blue** to decide which clauses to derive.
 - Suffices to resolve with existential pivots only (Q-Res, [KleineBüningKarpinskiFlögel 1995])
 - Suffices to eliminate variables in right-to-left order of quantification blocks (Level-ordered Q-Res)

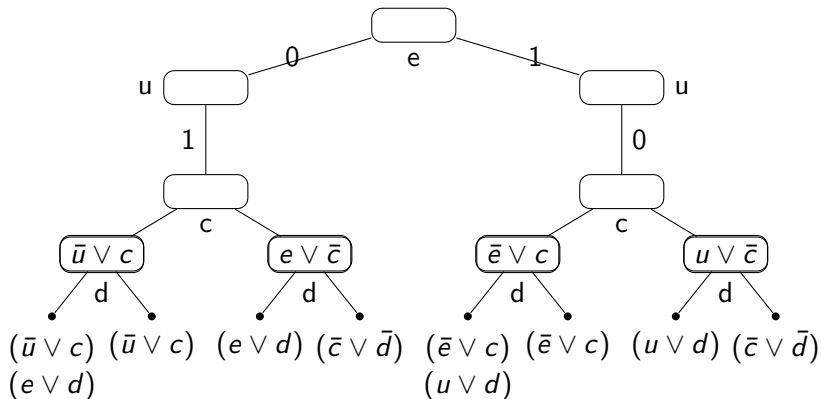
A derivation in Q-Res

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



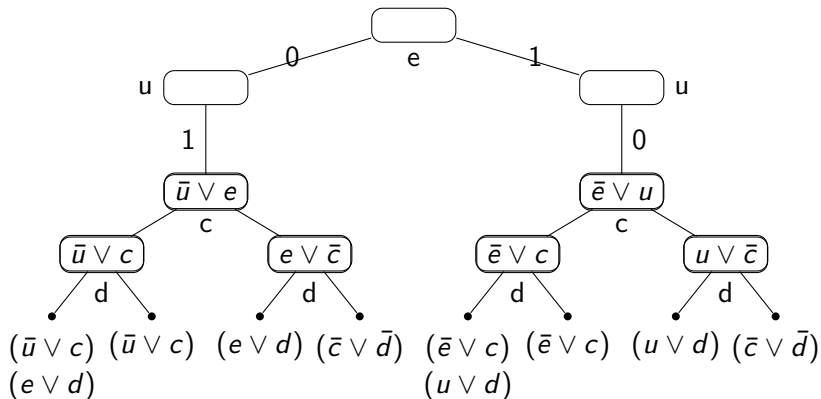
A derivation in Q-Res

$$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$$



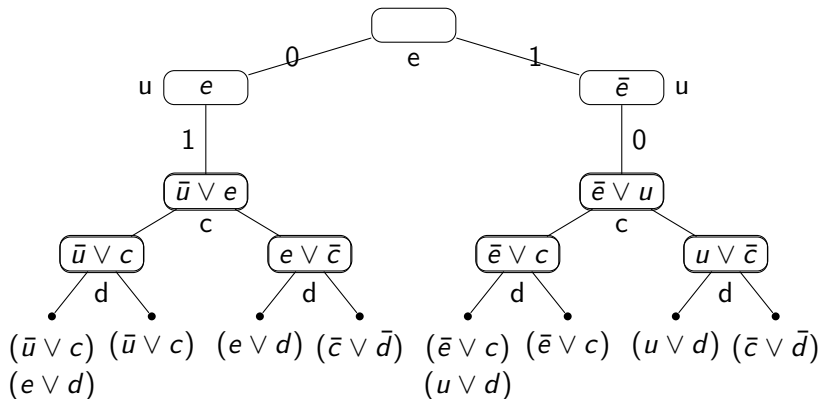
A derivation in Q-Res

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



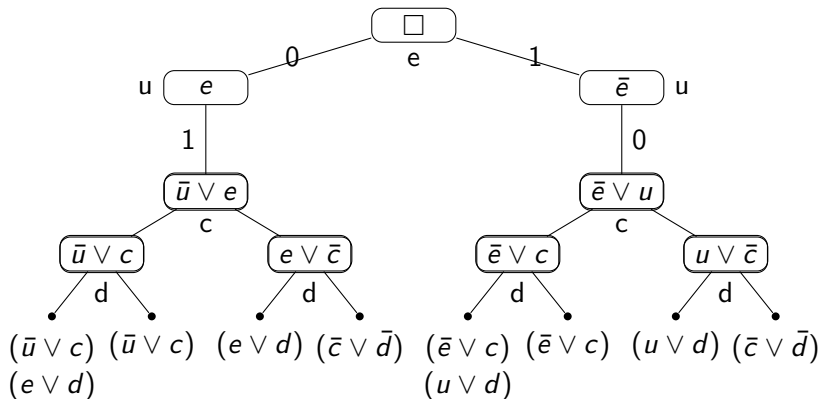
A derivation in Q-Res

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



A derivation in Q-Res

$$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$$



Adding the \forall -reduction rule

[Beyersdorff, Bonacina, Chew ITCS 2016]

P: Any sound and complete **line-based** proof system for UNSAT
eg Cutting Planes, Polynomial Calculus, Frege,
restrictions of Frege (AC^0 -Frege, $AC^0[p]$ -Frege, TC^0 -Frege ...)



$P+\forall$ Red: a sound and complete proof system for QBF

The CP+ \forall Red proof system

- CP+ \forall Red: Cutting Planes + \forall Reduction.

[Beyersdorff, Chew, M, Shukla FSTTCS 2016]

- Cutting Planes: Encode clauses as integer inequalities.

$$x \vee y \vee z \quad \rightarrow \quad x + y + z \geq 1$$

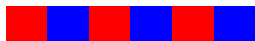
$$x \vee \neg y \vee z \quad \rightarrow \quad x + (1 - y) + z \geq 1$$
$$(x - y + z \geq 0)$$

$$x \vee \neg y \vee \neg z \quad \rightarrow \quad x + (1 - y) + (1 - z) \geq 1$$
$$(x - y - z \geq -1)$$

- Bags of inequalities, not clauses.
- Evaluation game: **Red** tries to satisfy all inequalities.
Blue tries to falsify some inequality.

The rules in Cutting Planes

If Red (\exists) can win



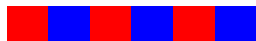
$$\begin{array}{rcl} \dots & & \\ \langle a \cdot x \rangle & \geq & A \\ \langle b \cdot x \rangle & \geq & B \\ \langle kc \cdot x \rangle & \geq & C \\ \dots & & \end{array}$$

(for $k \in \mathbb{Z}^{>0}$)

($\langle a \cdot x \rangle$ means $a_1x_1 + a_2x_2 + \dots + a_nx_n$.)

The rules in Cutting Planes

If Red (\exists) can win



$$\begin{array}{r} \dots \\ \langle a \cdot x \rangle \geq A \\ \langle b \cdot x \rangle \geq B \\ \langle kc \cdot x \rangle \geq C \\ \dots \end{array}$$

(for $k \in \mathbb{Z}^{>0}$)

Then Red can win



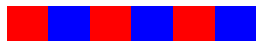
$$\begin{array}{r} \dots \\ \langle a \cdot x \rangle \geq A \\ \langle b \cdot x \rangle \geq B \\ \langle kc \cdot x \rangle \geq C \\ \dots \end{array}$$

$$\langle (a + b) \cdot x \rangle \geq A + B$$

the + rule

The rules in Cutting Planes

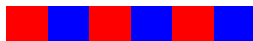
If Red (\exists) can win



$$\begin{array}{rcl} \dots & & \\ \langle a \cdot x \rangle & \geq & A \\ \langle b \cdot x \rangle & \geq & B \\ \langle kc \cdot x \rangle & \geq & C \\ \dots & & \end{array}$$

(for $k \in \mathbb{Z}^{>0}$)

Then Red can win



$$\begin{array}{rcl} \dots & & \\ \langle a \cdot x \rangle & \geq & A \\ \langle b \cdot x \rangle & \geq & B \\ \langle kc \cdot x \rangle & \geq & C \\ \dots & & \end{array}$$

$$\langle (a + b) \cdot x \rangle \geq A + B$$

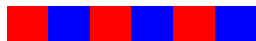
the + rule

$$\langle ka \cdot x \rangle \geq kA$$

the \times rule

The rules in Cutting Planes

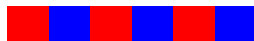
If Red (\exists) can win



$$\begin{array}{r} \dots \\ \langle a \cdot x \rangle \geq A \\ \langle b \cdot x \rangle \geq B \\ \langle kc \cdot x \rangle \geq C \\ \dots \end{array}$$

(for $k \in \mathbb{Z}^{>0}$)

Then Red can win



$$\begin{array}{r} \dots \\ \langle a \cdot x \rangle \geq A \\ \langle b \cdot x \rangle \geq B \\ \langle kc \cdot x \rangle \geq C \\ \dots \end{array}$$

$$\langle (a + b) \cdot x \rangle \geq A + B$$

the + rule

$$\langle ka \cdot x \rangle \geq kA$$

the \times rule

$$\langle c \cdot x \rangle \geq \left\lceil \frac{C}{k} \right\rceil$$

the \div rule

\forall reduction in $CP + \forall Red$

- If **Red** can win with \mathcal{I} containing $\langle a \cdot x \rangle \geq A$ where the rightmost non-zero coefficient in a is **blue**,
 $a = a' \mathbf{b} 00\dots 0$, (ie a universal variable, u)
- then **Red** can win with
 $\mathcal{I} \cup \{ \langle a' \mathbf{0}00\dots 00 \rangle \cdot x \geq A - b \} \cup \{ \langle a' \mathbf{0}00\dots 0 \rangle \cdot x \geq A \} .$
 \uparrow \uparrow
(Set $u = 1$) (Set $u = 0$)
- This **Blue**-elimination is the \forall -Reduction rule.

Proofs in the $CP+\forall$ Red proof system

Keep using the $+$, \times , \div and \forall Reduction rules.

Red can win with $\mathcal{I} = \mathcal{I}_0$



Red can win with \mathcal{I}_1



Red can win with \mathcal{I}_2



\vdots



Red can win with \mathcal{I}_t .

If \mathcal{I}_t contains $\mathbf{0} \geq \mathbf{1}$,

then Red can't win with \mathcal{I}_t , and so Red can't win with \mathcal{I} .

Expansion-Based Systems

$\forall u Q \vec{x} \cdot F(u, \vec{x})$ is true

\Updownarrow

$[Q \vec{x} \cdot F(0, \vec{x})] \wedge [Q \vec{x} \cdot F(1, \vec{x})]$ is true

\Updownarrow

$Q \vec{x}^{u/0} Q \vec{x}^{u/1} \cdot [F(0, \vec{x}^{u/0}) \wedge F(1, \vec{x}^{u/1})]$ is true

Expansion-Based Systems

$\forall u Q\vec{x} \cdot F(u, \vec{x})$ is true

\Updownarrow

$[Q\vec{x} \cdot F(0, \vec{x})] \wedge [Q\vec{x} \cdot F(1, \vec{x})]$ is true

\Updownarrow

$Q\vec{x}^{u/0} Q\vec{x}^{u/1} \cdot [F(0, \vec{x}^{u/0}) \wedge F(1, \vec{x}^{u/1})]$ is true

- Expand the initial formula judiciously, on the fly.

Then use standard resolution.

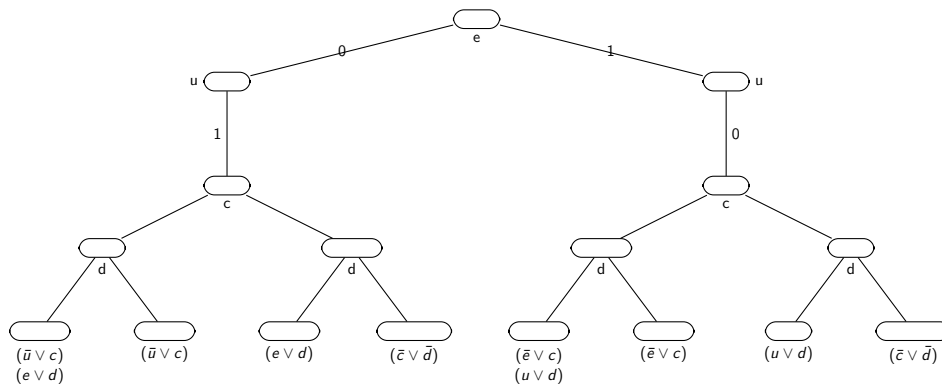
- Expansion-based systems:

$\forall\text{Exp}+\text{Res}$ [Janota, Marques-Silva 2015],

IR [Beyersdorff, Chew, Janota 2014].

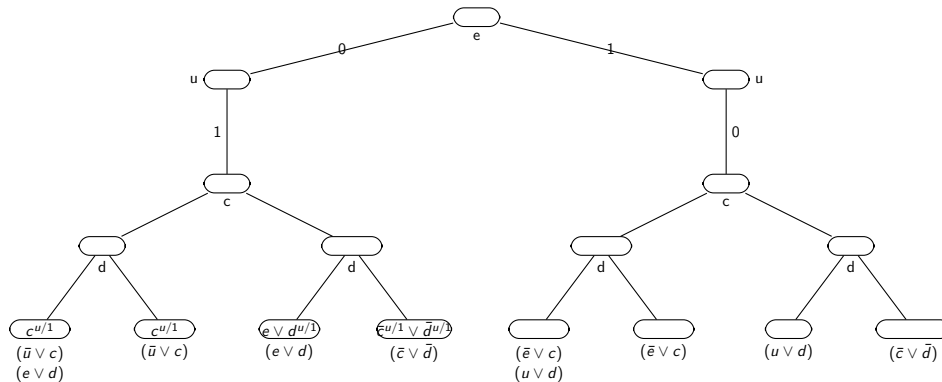
Part of a derivation in $\forall\text{Exp}+\text{Res}$

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



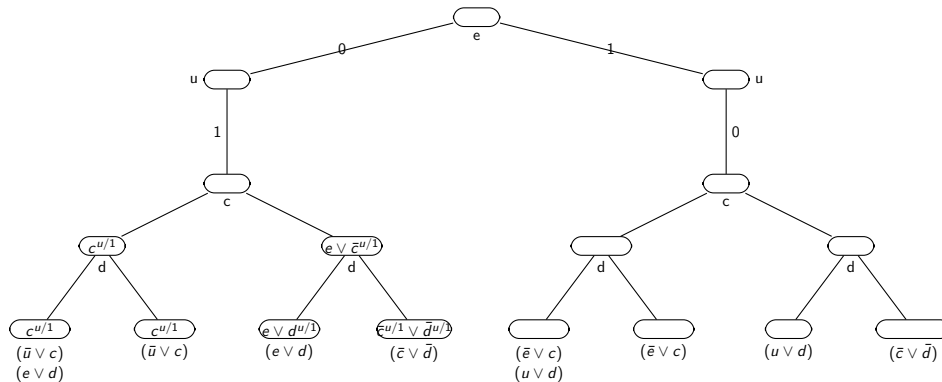
Part of a derivation in $\forall\text{Exp}+\text{Res}$

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



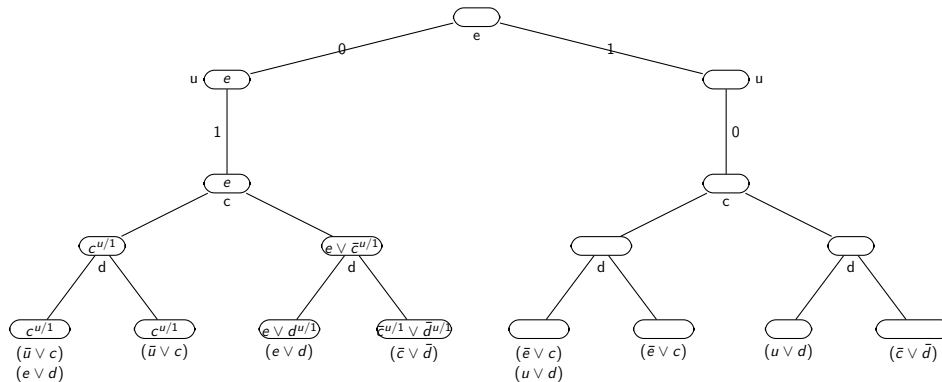
Part of a derivation in $\forall\text{Exp}+\text{Res}$

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



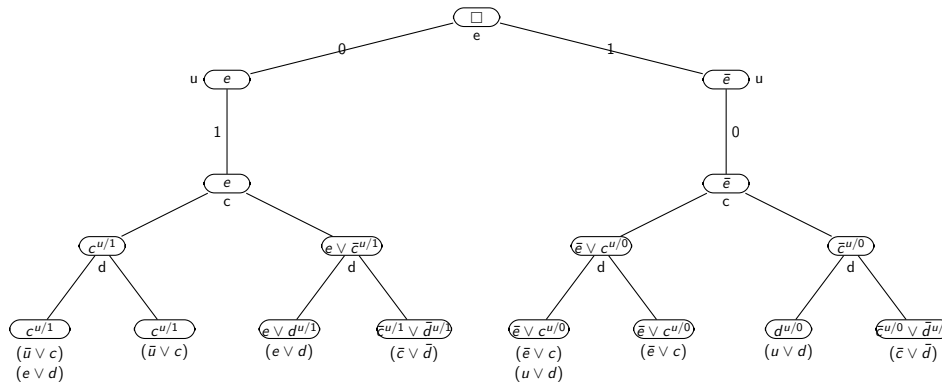
Part of a derivation in $\forall\text{Exp}+\text{Res}$

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



Part of a derivation in $\forall\text{Exp}+\text{Res}$

$\exists e \forall u \exists c \exists d [(\neg e \vee c), (\neg u \vee c), (e \vee d), (u \vee d), (\neg c \vee \neg d)]$



Merging complementary literals

- Consider $\exists x \forall u (x \vee \neg u)(\neg x \vee u)$.

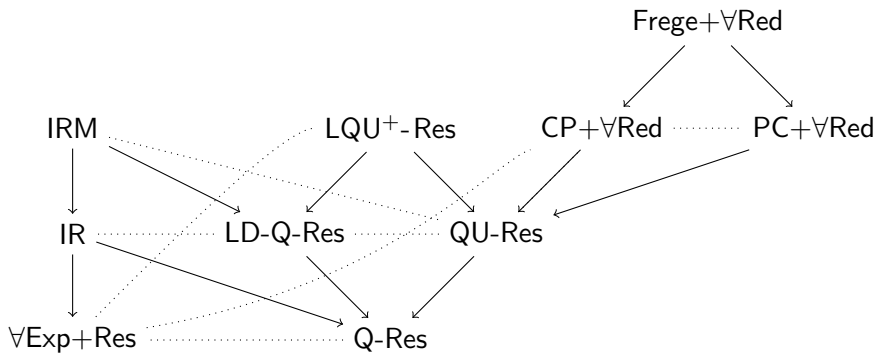
Merging complementary literals

- Consider $\exists x \forall u (x \vee \neg u)(\neg x \vee u)$.
- Resolve on x ; instead of tautology $u \vee \neg u$, merge u and $\neg u$ into u^* .
Intended meaning: Blue's winning strategy for u is not dictated by this clause, but will be decided by the setting to x .

Merging complementary literals

- Consider $\exists x \forall u (x \vee \neg u)(\neg x \vee u)$.
- Resolve on x ; instead of tautology $u \vee \neg u$, merge u and $\neg u$ into u^* .
Intended meaning: Blue's winning strategy for u is not dictated by this clause, but will be decided by the setting to x .
- Proof Systems that use merging:
LD-Q-Res (Long-Distance QRes), [Balabanov, Jiang 2012]
LQU⁺-Res, [Balabanov, Widl, Jiang 2014]
IRM (Instantiation, Resolution, Merge) [Beyersdorff, Chew, Janota 2014].

The relative power of some QBF proof systems:



Lower Bounds for QBF proof systems

- from propositional hardness.
not useful for understanding QBF solvers

Lower Bounds for QBF proof systems

- from propositional hardness.
not useful for understanding QBF solvers
- by adapting techniques for propositional hardness.
let's review

The Size-Width Relation in Resolution

In Resolution, **Short proofs are narrow.**

(Size of proof: number of steps. Width of proof: max width of clause in proof.)

Theorem ([Ben-Sasson, Wigderson 2001])

For all unsatisfiable CNFs F in n variables:

$$S(\frac{\vdash}{\text{Res}_T} F) \geq 2^{w(\frac{\vdash}{\text{Res}} F) - w(F)} . \quad (\text{tree-like proofs; no reusing clauses})$$

$$S(\frac{\vdash}{\text{Res}} F) = \exp \left(\Omega \left(\frac{(w(\frac{\vdash}{\text{Res}} F) - w(F))^2}{n} \right) \right) .$$

The Size-Width relation in Q-Res

In Q-Res, this fails completely!

[Beyersdorff,Chew,M,Shukla STACS 2016, ACM Trans. Comp. Logic 2018]

The Size-Width relation in Q-Res

In Q-Res, this fails completely!

[Beyersdorff, Chew, M, Shukla STACS 2016, ACM Trans. Comp. Logic 2018]

$$\forall u_1 u_2 \dots u_n \exists e_0 e_1 \dots e_n \left[\begin{array}{l} (e_0) \\ (\neg e_{i-1} \vee u_i \vee e_i) \quad \text{for } i \in [n] \\ (\neg e_n) \end{array} \right]$$

- Using Resolution, derive $u_1 \vee \dots \vee u_n$. ($n + 1$ steps)
- Then using \forall Red, derive \square . (n steps)
- So proof of size $O(n)$. Even tree-like.
- We show: Any proof **must** derive $u_1 \vee \dots \vee u_n$.
- So width of any proof $\Omega(n)$.

Re-defining Width For QBFs

Problem: accumulation of universal variables.

Possible fix: Redefine Width_{\exists} . Count only existential variables.

Now does an analogue of the short-proofs-are-narrow hold?

Re-defining Width For QBFs

Problem: accumulation of universal variables.

Possible fix: Redefine Width_{\exists} . Count only existential variables.

Now does an analogue of the short-proofs-are-narrow hold?

No!

Completion Principle: clausal encoding of

$$\exists X \in \{0, 1\}^{n \times n} \forall z \ (z \vee \exists \text{ all-1s row}) \wedge (\neg z \vee \exists \text{ all-0s column})$$

Re-defining Width For QBFs

Problem: accumulation of universal variables.

Possible fix: Redefine Width_{\exists} . Count only existential variables.

Now does an analogue of the short-proofs-are-narrow hold?

No!

Completion Principle: clausal encoding of

$$\exists X \in \{0, 1\}^{n \times n} \forall z (z \vee \exists \text{ all-1s row}) \wedge (\neg z \vee \exists \text{ all-0s column})$$

Under appropriate clausal encoding, proof of size $O(n^2)$.

Even tree-like proof : no reusing derived clauses.

We show: Any proof must have $\text{width}_{\exists} \Omega(n)$.

Size-Width \exists relation for non-tree-like proofs

$$\exists e_1 \forall u_1 \exists c_1 \exists d_1 \quad \exists e_2 \forall u_2 \exists c_2 \exists d_2 \quad \dots \quad \exists e_n \forall u_n \exists c_n \exists d_n$$

for $i \in [n]$,

$$\begin{array}{ll} (\neg e_i \vee c_i) & (e_i \vee d_i) \\ (\neg u_i \vee c_i) & (u_i \vee d_i) \end{array}$$
$$\neg c_1 \vee \neg d_1 \vee \neg c_2 \vee \neg d_2 \vee \dots \vee \neg c_n \vee \neg d_n$$

Winning strategy for universal player: $u_i = \neg e_i$.

Size-Width \exists relation for non-tree-like proofs

$$\exists e_1 \forall u_1 \exists c_1 \exists d_1 \quad \exists e_2 \forall u_2 \exists c_2 \exists d_2 \quad \dots \quad \exists e_n \forall u_n \exists c_n \exists d_n$$

for $i \in [n]$,

$$\begin{array}{ll} (\neg e_i \vee c_i) & (e_i \vee d_i) \\ (\neg u_i \vee c_i) & (u_i \vee d_i) \end{array}$$
$$\neg c_1 \vee \neg d_1 \vee \neg c_2 \vee \neg d_2 \vee \dots \vee \neg c_n \vee \neg d_n$$

Winning strategy for universal player: $u_i = \neg e_i$.

Encode last clause with additional \exists variables as short clauses.

Size-Width \exists relation for non-tree-like proofs

$$\exists e_1 \forall u_1 \exists c_1 \exists d_1 \quad \exists e_2 \forall u_2 \exists c_2 \exists d_2 \quad \dots \quad \exists e_n \forall u_n \exists c_n \exists d_n$$

$$\begin{aligned} \text{for } i \in [n], \quad & (\neg e_i \vee c_i) \quad (e_i \vee d_i) \\ & (\neg u_i \vee c_i) \quad (u_i \vee d_i) \\ & \neg c_1 \vee \neg d_1 \vee \neg c_2 \vee \neg d_2 \vee \dots \vee \neg c_n \vee \neg d_n \end{aligned}$$

Winning strategy for universal player: $u_i = \neg e_i$.

Encode last clause with additional \exists variables as short clauses.

Short proofs in Q-Res, size $n^{O(1)}$.

We show: Width \exists of any Q-Res proof $\Omega(n)$.

Size-Width_∃ relation for non-tree-like proofs

$$\exists e_1 \forall u_1 \exists c_1 \exists d_1 \quad \exists e_2 \forall u_2 \exists c_2 \exists d_2 \quad \dots \quad \exists e_n \forall u_n \exists c_n \exists d_n$$

$$\begin{aligned} \text{for } i \in [n], \quad & (\neg e_i \vee c_i) \quad (e_i \vee d_i) \\ & (\neg u_i \vee c_i) \quad (u_i \vee d_i) \\ & \neg c_1 \vee \neg d_1 \vee \neg c_2 \vee \neg d_2 \vee \dots \vee \neg c_n \vee \neg d_n \end{aligned}$$

Winning strategy for universal player: $u_i = \neg e_i$.

Encode last clause with additional \exists variables as short clauses.

Short proofs in Q-Res, size $n^{O(1)}$.

We show: Width_∃ of any Q-Res proof $\Omega(n)$.

Large width requirement does not give size lower bound.

Lower Bounds for QBF proof systems

- from propositional hardness.
not useful for understanding QBF solvers
- by adapting techniques for propositional hardness.
let's review:
size-width fails for Q-Res

Lower Bounds for QBF proof systems

- from propositional hardness.
not useful for understanding QBF solvers
- by adapting techniques for propositional hardness.
let's review:
size-width fails for Q-Res
interpolation?

Feasible Interpolation – the propositional case

$$F = A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r}) \text{ in UNSAT}$$



for all assignments \vec{a} to \vec{p} , either $A(\vec{a}, \vec{q})$ or $B(\vec{a}, \vec{r})$ in UNSAT.

Feasible Interpolation – the propositional case

$$F = A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r}) \text{ in UNSAT}$$



for all assignments \vec{a} to \vec{p} , either $A(\vec{a}, \vec{q})$ or $B(\vec{a}, \vec{r})$ in UNSAT.

- Given \vec{a} , can we tell which is in UNSAT?

Feasible Interpolation – the propositional case

$$F = A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r}) \text{ in UNSAT}$$



for all assignments \vec{a} to \vec{p} , either $A(\vec{a}, \vec{q})$ or $B(\vec{a}, \vec{r})$ in UNSAT.

- Given \vec{a} , can we tell which is in UNSAT?
- We want an interpolant circuit C in \vec{p} variables:

$$C(\vec{a}) = 0 \implies A(\vec{a}, \vec{q}) \text{ is in UNSAT, and}$$

$$C(\vec{a}) = 1 \implies B(\vec{a}, \vec{r}) \text{ is in UNSAT.}$$

Feasible Interpolation – the propositional case (cont'd)

Theorem ([Krajíček 1997],[Pudlák 1997])

- *Resolution proofs of size s give Boolean circuits of size $s^{O(1)}$ computing interpolants.*
- *Cutting Planes proofs of size s give real arithmetic circuits of size $s^{O(1)}$ computing interpolants.*

Feasible Interpolation – the propositional case (cont'd)

Theorem ([Krajčček 1997],[Pudlák 1997])

- *Resolution proofs of size s give Boolean circuits of size $s^{O(1)}$ computing interpolants.*
- *Cutting Planes proofs of size s give real arithmetic circuits of size $s^{O(1)}$ computing interpolants.*
- *If \vec{p} variables appears only positively in $A(\vec{p}, \vec{q})$ or only negatively in $B(\vec{p}, \vec{r})$, then interpolant circuit is (real-) monotone.*

Feasible Interpolation – the propositional case (cont'd)

Theorem ([Krajčček 1997],[Pudlák 1997])

- *Resolution proofs of size s give Boolean circuits of size $s^{O(1)}$ computing interpolants.*
- *Cutting Planes proofs of size s give real arithmetic circuits of size $s^{O(1)}$ computing interpolants.*
- *If \vec{p} variables appears only positively in $A(\vec{p}, \vec{q})$ or only negatively in $B(\vec{p}, \vec{r})$, then interpolant circuit is (real-) monotone.*
- *All resolution / cutting-plane proofs of the clique-colour formulas are of exponential size.*

(Clique-colour formulas: CNF encodings of

“ \exists a graph that is $(k - 1)$ -colourable and has a k -clique.”)

Feasible Interpolation for QBFs

$\exists \vec{p} \ Q \vec{q} \ Q \vec{r} \ [A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})]$ is false

\Downarrow

$\exists \vec{p} [Q \vec{q} A(\vec{p}, \vec{q}) \wedge Q \vec{r} B(\vec{p}, \vec{r})]$ is false

\Downarrow

for all assignments \vec{a} to \vec{p} , either $Q \vec{q} A(\vec{a}, \vec{q})$ or $Q \vec{r} B(\vec{a}, \vec{r})$ is false.

Interpolant circuit:

$C(\vec{a}) = 0 \implies Q \vec{q} A(\vec{a}, \vec{q})$ is false, and

$C(\vec{a}) = 1 \implies Q \vec{r} B(\vec{a}, \vec{r})$ is false.

Feasible Interpolation works for many QBF proof systems

The Clique-coClique formulas: CNF encodings of
“ \exists an n -vertex graph G , $\forall u$, u implies G has a k -clique, ”
“ $\forall u$, u implies G has a k -clique, ”
“ $\exists u$, u implies G has a k -clique, ”
“ $\forall u$, $\neg u$ implies G has no k -clique.”

(Note: To express **no clique**, universal quantifiers used.
Not succinctly expressible as UNSAT instance.)

Theorem ([Beyersdorff,Chew,M,Shukla ICALP15, LMCS17, FSTTCS16])

All the resolution-based QBF proof systems

Q-Res, QU-Res, LD-Q-Res, LQU⁺-Res, \forall Exp+Res, IR, IRM

as well as the proof system CP+ \forall Red, admit feasible monotone interpolation.

All Clique-coClique formulas need exponential-sized proofs in all these proof systems.

Lower Bounds for QBF proof systems

- from propositional hardness.
not useful for understanding QBF solvers
- by adapting techniques for propositional hardness. let's review:
 - Size lower bounds from Width lower bounds does not work with the simplest extension of Resolution, Q-Res.
 - Feasible Interpolation works for all Resolution based systems and for CP+ \forall Red.

Lower Bounds for QBF proof systems

- from propositional hardness.
not useful for understanding QBF solvers
- by adapting techniques for propositional hardness. let's review:
 - Size lower bounds from Width lower bounds does not work with the simplest extension of Resolution, Q-Res.
 - Feasible Interpolation works for all Resolution based systems and for CP+ \forall Red.
- from strategy extraction.
all-new; specific to QBFs

The winning strategy of the universal player in the evaluation game leads to new lower bound techniques.

Strategy Extraction

- Main idea: A proof reveals information about a winning strategy.
- Examine a proof.
- Construct a circuit of a special type for computing the winning strategy.
- Circuit type: depends on the proof system
Circuit size: depends on the proof size
- If the winning strategy is hard to compute in the relevant circuit model, then all proofs in the proof system must be large.

From Proof to Decision List for Winning Strategy

Blue has to choose the value of a variable u .

Blue knows values of all variables left of u ; partial assignment \vec{a} .

Proof lines L_1, L_2, \dots, L_m .

\forall Red on u at $(1 <) i_1 < i_2 < \dots < i_k (\leq m)$.

L_{i_r} : eliminate u from $L_{j_r}, j_r < i_r$.

From Proof to Decision List for Winning Strategy

Blue has to choose the value of a variable u .

Blue knows values of all variables left of u ; partial assignment \vec{a} .

Proof lines L_1, L_2, \dots, L_m .

\forall Red on u at $(1 <) i_1 < i_2 < \dots < i_k (\leq m)$.

L_{i_r} : eliminate u from $L_{j_r}, j_r < i_r$.

```
if       $L_{i_1}(\vec{a})$  false then set  $u$  to make  $L_{j_1}(\vec{a})$  false
elseif  $L_{i_2}(\vec{a})$  false then set  $u$  to make  $L_{j_2}(\vec{a})$  false
:
:
elseif  $L_{i_k}(\vec{a})$  false then set  $u$  to make  $L_{j_k}(\vec{a})$  false
else                                     set  $u = 0$ .
```

From Proof to Decision List for Winning Strategy

Blue has to choose the value of a variable u .

Blue knows values of all variables left of u ; partial assignment \vec{a} .

Proof lines L_1, L_2, \dots, L_m .

\forall Red on u at $(1 <) i_1 < i_2 < \dots < i_k (\leq m)$.

L_{i_r} : eliminate u from $L_{j_r}, j_r < i_r$.

if $L_{i_1}(\vec{a})$ false **then** set u to make $L_{j_1}(\vec{a})$ false
elseif $L_{i_2}(\vec{a})$ false **then** set u to make $L_{j_2}(\vec{a})$ false
 \vdots \vdots
elseif $L_{i_k}(\vec{a})$ false **then** set u to make $L_{j_k}(\vec{a})$ false
else set $u = 0$.

[Beyersdorff, Chew, Janota 2015], [Beyersdorff, Bonacina, Chew 2016]:

This strategy is a winning strategy for Blue.

Strategy description: A Decision List for each universal variable.

Strategy Extraction from $P+\forall$ Red proofs

Proof with s \forall Reduction steps



Winning strategy can be computed by a Decision List with s steps.

- QU-Res: Each condition is a **clause**.
(is $a_1 \vee a_2 \vee \dots \vee a_n$ true?)
- $CP+\forall$ Red: Each condition is a **linear threshold function**.
(is $c_1a_1 + c_2a_2 + \dots + c_na_n \geq b$?)

Genuine QBF bounds for $P+\forall$ Red proofs

Proof with s \forall Reduction steps



Winning strategy can be computed by a Decision List with s steps.

Contrapositive gives lower bounds on number of \forall Reduction steps, not just on proof size.

Genuine QBF bounds for $P+\forall$ Red proofs

Proof with s \forall Reduction steps



Winning strategy can be computed by a Decision List with s steps.

Contrapositive gives lower bounds on number of \forall Reduction steps, not just on proof size.

i.e. Proving the QBF false will require large size **even with a SAT oracle** (appropriately formalised).

Winning Strategies compute desired functions

Fix Boolean function f . Fix small circuit C computing f (size m).

Define QBF $Q_{f,C}$:

$$\exists x_1 x_2 \dots x_n \forall w \exists z_1 z_2 \dots z_m \left[\begin{array}{l} (w \neq z_m) \\ (z_i = \text{value of } i\text{th gate of } C(x)) : i \in [m] \end{array} \right]$$

(z_i clauses enforce $z_m = f(x)$.)

Winning Strategies compute desired functions

Fix Boolean function f . Fix small circuit C computing f (size m).

Define QBF $Q_{f,C}$:

$$\exists x_1 x_2 \dots x_n \forall w \exists z_1 z_2 \dots z_m \left[\begin{array}{l} (w \neq z_m) \\ (z_i = \text{value of } i\text{th gate of } C(x)) : i \in [m] \end{array} \right]$$

(z_i clauses enforce $z_m = f(x)$.)

- Blue can choose $w = f(x)$ and force a win.
- No other way for Blue to win.
- $f(x)$ not hard to compute – it has a small circuit.
If no small decision list, then no small proof.

Winning Strategies Hard for Decision Lists

- The PARITY function has an $O(n)$ size circuit.
The PARITY function requires exponentially long decision lists of clauses. ([Håstad]: $\oplus \notin AC^0$.) Hence

Theorem ([Beyersdorff, Chew, Janota 2015])

Any Q-Res or QU-Res proof for Q-PARITY must be of exponential size.

Winning Strategies Hard for Decision Lists

- The PARITY function has an $O(n)$ size circuit.
The PARITY function requires exponentially long decision lists of clauses. ([Håstad]: $\oplus \notin AC^0$.) Hence

Theorem ([Beyersdorff,Chew,Janota 2015])

Any Q-Res or QU-Res proof for Q-PARITY must be of exponential size.

- The Inner Product function has an $O(n)$ size circuit.
 $IP(x, y) \triangleq \langle x \cdot y \rangle \pmod 2$.
The IP function needs $> 2^{n/2} - 1$ steps in a decision list of linear threshold functions. ([Turán,Vatan 1997]) Hence

Theorem ([Beyersdorff,Chew,M,Shukla 2016])

Any CP+ \forall Red proof for Q-IP must be of exponential size.

Other Sources of Hardness

- [Beyersdorff, Pich LICS 2016]
Every lower bound in Frege+ \forall Red stems from
 - either propositional hardness,
 - or a circuit lower bound.No other source of hardness.

Other Sources of Hardness

- [Beyersdorff, Pich LICS 2016]
Every lower bound in $\text{Frege} + \forall\text{Red}$ stems from
 - either propositional hardness,
 - or a circuit lower bound.No other source of hardness.
- Not true for weaker systems.

Other Sources of Hardness (cont'd)

$$\begin{aligned} & \exists x_1 \cdots x_n \forall u_1 \cdots u_n \exists t_1 \cdots t_n \\ & \quad (x_i \vee u_i \vee t_i) && i \in [n] \\ & \quad (\neg x_i \vee \neg u_i \vee t_i) && i \in [n] \\ & \quad (\neg t_1 \vee \cdots \vee \neg t_n) \end{aligned}$$

- Blue has a trivial winning strategy: $u_i = x_i$.
- But the formula is still hard to prove false in QU-Res. Why?

Winning Strategies needing Varied Responses (cont'd)

- Many responses needed in winning strategy.
 2^n possible values for $u_1 \cdots u_n$, all necessary.
cost of formula high.
- Each line can contribute only so much: capacity small.
- Hence proof size must be large.

Theorem ([Beyersdorff,Blinkhorn,Hinde ITCS2018])

Size-Cost-Capacity Theorem:

For any proof π of a QBF ϕ in a system $P+\forall Red$,

$$Size(\pi) \times Capacity(\pi) \geq Cost(\phi)$$

Winning Strategies needing Varied Responses (cont'd)

- Many responses needed in winning strategy.
 2^n possible values for $u_1 \cdots u_n$, all necessary.
cost of formula high.
- Each line can contribute only so much: capacity small.
- Hence proof size must be large.

Theorem ([Beyersdorff,Blinkhorn,Hinde ITCS2018])

Size-Cost-Capacity Theorem:

For any proof π of a QBF ϕ in a system $P+\forall Red$,

$$Size(\pi) \times Capacity(\pi) \geq Cost(\phi)$$

Similar result for expansion-based systems; [Beyersdorff,Blinkhorn STACS 18].

To conclude ...

QBF Proof systems

- What are they?

Formal systems for proving false QBFs false.

- Why do we study them?

Lower bounds can help guide development of better solvers.

To conclude ...

QBF Proof systems

- What are they?

Formal systems for proving false QBFs false.

- Why do we study them?

Lower bounds can help guide development of better solvers.

- What do we know?

Extracting strategies from proofs leads to lower bounds.

To conclude ...

QBF Proof systems

- What are they?

Formal systems for proving false QBFs false.

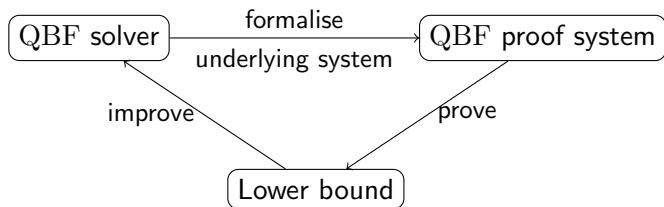
- Why do we study them?

Lower bounds can help guide development of better solvers.

- What do we know?

Extracting strategies from proofs leads to lower bounds.

- What next? Continue the cycle



Thank you