Beyond SAT - Proofs for QBF, and more

Meena Mahajan

The Institute of Mathematical Sciences (HBNI), Chennai, India.

18 April 2023

Talk Plan

Quanified Boolean Formulas (QBFs) and Formal Proofs

- A Proof Complexity perspective
- A QBF-Solving perspective
- A Computational Complexity perspective

Talk Plan

Quanified Boolean Formulas (QBFs) and Formal Proofs

- QBF basics
- A Proof Complexity perspective
- A QBF-Solving perspective
- A Computational Complexity perspective

Talk Plan

Quanified Boolean Formulas (QBFs) and Formal Proofs

- QBF basics
- A Proof Complexity perspective
- A QBF-Solving perspective
- A Computational Complexity perspective
- Some Questions / Directions / Speculations ...

Quantified Boolean Formulas

- Propositional satisfiability:
  Is $\varphi(x_1, x_2, \ldots, x_n)$ satisfiable?

  Restated as QBF:
  Is $\exists x_1 \exists x_2 \ldots \exists x_n \varphi(x_1, x_2, \ldots, x_n)$ true?
- Generalise: allow $\forall$ quantifiers as well. For $Q_i \in \{\exists, \forall\}$,
  Is $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi(x_1, x_2, \ldots, x_n)$ true?
- Same expressiveness as SAT, but more succinct.
- Deciding True/False: PSPACE-complete.

Quantified Boolean Formulas

- Propositional satisfiability:
  Is $\varphi(x_1, x_2, \ldots, x_n)$ satisfiable?

  Restated as QBF:
  Is $\exists x_1 \exists x_2 \ldots \exists x_n \varphi(x_1, x_2, \ldots, x_n)$ true?
- Generalise: allow $\forall$ quantifiers as well. For $Q_i \in \{\exists, \forall\}$,
  Is $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi(x_1, x_2, \ldots, x_n)$ true?
- Same expressiveness as SAT, but more succinct.
- Deciding True/False: PSPACE-complete.
- We consider QBFs that are
  - totally quantified (no unbound variables),
    (each such QBF either true or false)
  - in prenex form,
  - with inner propositional formula in CNF.

The two-player evaluation game

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

  Assignment constructed on a run: $\tilde{a}$.

  $P_\exists$ wins a run of the game if $F(\tilde{a})$ true. Otherwise $P_\forall$ wins.
- $Q\vec{x} \cdot F(x)$ true if and only if $P_\exists$ has a winning strategy.
  (model, Skolem function)
- $Q\vec{x} \cdot F(x)$ false if and only if $P_\forall$ has a winning strategy.
  (countermodel, Herbrand function)

Talk Plan

Quanified Boolean Formulas QBFs and Formal Proofs

✓ QBF basics

- A Proof Complexity perspective
- A QBF-Solving perspective
- A Computational Complexity perspective
- Some Questions / Directions / Speculations ...

Proof System

What we expect from a proof system:

- Proofs should be short.
- Proofs should be efficiently verifiable.
- Soundness – no proofs of false statements.
- Completeness – proofs of all true statements.

Proof Systems for refuting QBFs

- Propositional Proof Systems handle special case of QBFs.
  They prove $\exists \cdot \texttt{CNF}$ sentences false.
- Augment to handle full prenex false QBFs.

Proof Systems for refuting QBFs

- Propositional Proof Systems handle special case of QBFs.
  They prove $\exists \cdot \texttt{CNF}$ sentences false.
- Augment to handle full prenex false QBFs.
- Ensuring soundness: (augmented) rules allow extraction of a $P_\forall$ winning strategy (Herbrand function) from a proof.

Proof Systems for refuting QBFs

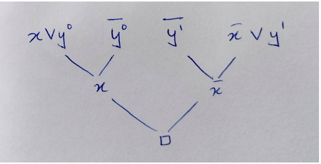
- Propositional Proof Systems handle special case of QBFs. They prove $\exists \cdot \texttt{CNF}$ sentences false.
- Augment to handle full prenex false QBFs.
- Ensuring soundness: (augmented) rules allow extraction of a $P_\forall$ winning strategy (Herbrand function) from a proof.
- Ensuring Completeness: different paradigms.
  - Expansion ($\forall \to \wedge$) – obvious semantics of universal variables
  - Universal reduction – preserves $P_\exists$ winning strategy if one exists
  - Literal Merging – implicitly remember $P_\forall$ winning strategy may be complex
  - Explicitly building up $P_\forall$ winning strategy

Proof Systems for refuting QBFs (by example)

$$\exists x \; \forall u \; \exists y \quad (x \vee u \vee y)(\bar{x} \vee \bar{u} \vee y)(\bar{y})$$

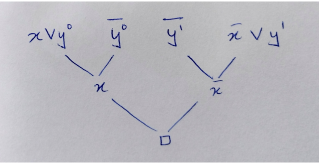
Proof Systems for refuting QBFs (by example)

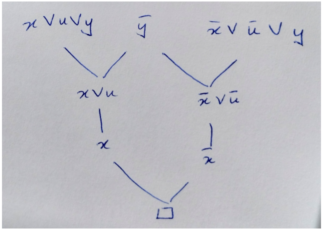
$$\exists x \ \forall u \ \exists y \quad (x \vee u \vee y)(\bar{x} \vee \bar{u} \vee y)(\bar{y})$$



Proof Systems for refuting QBFs (by example)

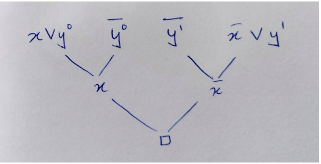
$$\exists x \; \forall u \; \exists y \quad (x \vee u \vee y)(\bar{x} \vee \bar{u} \vee y)(\bar{y})$$

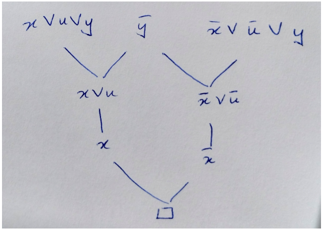


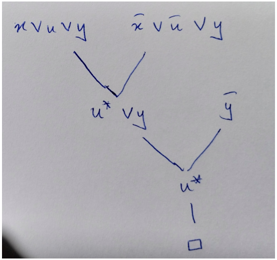


Proof Systems for refuting QBFs (by example)

$$\exists x \ \forall u \ \exists y \quad (x \vee u \vee y)(\bar{x} \vee \bar{u} \vee y)(\bar{y})$$

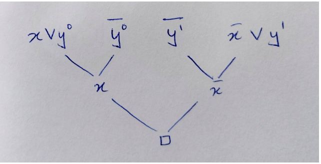


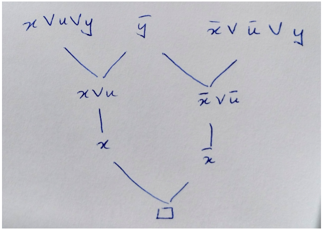


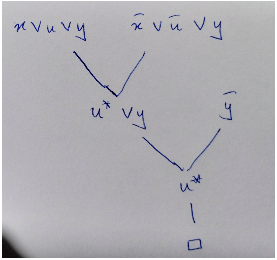


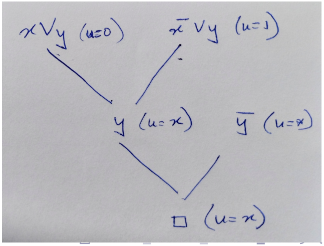
Proof Systems for refuting QBFs (by example)

$$\exists x\ \forall u\ \exists y \quad (x \vee u \vee y)(\bar{x} \vee \bar{u} \vee y)(\bar{y})$$

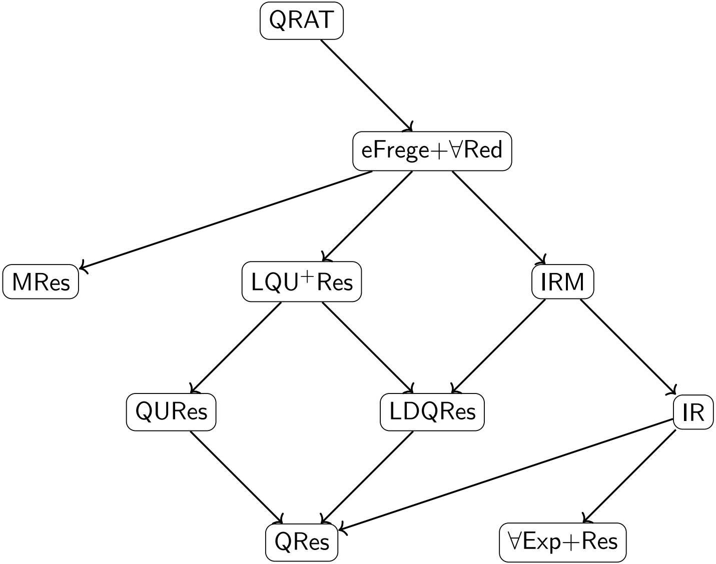








Some QBF Proof Systems – The simulation order



(Accident of nomenclature: What is truly Resolution for QBFs?)

Augmenting PPS to QBF proof systems

- Expansion works for **any** PPS.
- Universal reduction, $P + \forall\texttt{red}$, works for **most** line-based PPS.
- Literal merging: seems specific to Resolution, and not yet fully understood.
- Explicitly building up $P_\forall$ strategies: seems specific to Resolution, but not fully understood.

Are there other undiscovered paradigms?

Techniques for Lower Bounds

- Transfer propositional hardness.
- Transfer computational hardness.
- Identify semantic hardness.

Lower Bounds: Transferring Propositional Hardness

- Inside every reasonable QBF proof system P,
  there is an easily-described embedded PPS Q.
- In a reasonable QBF proof system P, with underlying PPS Q,
  for every UNSAT formula F,
  refuting $\exists.F$ in P no easier than proving unsatisfiability of F in Q.

Lower Bounds: Transferring Propositional Hardness

- Inside every reasonable QBF proof system P,
  there is an easily-described embedded PPS Q.
- In a reasonable QBF proof system P, with underlying PPS Q,
  for every UNSAT formula F,
  refuting $\exists.F$ in P no easier than proving unsatisfiability of F in Q.

  So we already have lower bounds.

Lower Bounds: Transferring Propositional Hardness

- Inside every reasonable QBF proof system P,
  there is an easily-described embedded PPS Q.
- In a reasonable QBF proof system P, with underlying PPS Q,
  for every UNSAT formula F,
  refuting $\exists.F$ in P no easier than proving unsatisfiability of F in Q.

  So we already have lower bounds.
- Not "genuine QBF hardness".

Lower Bounds: Transferring Propositional Hardness

- Inside every reasonable QBF proof system P, there is an easily-described embedded PPS Q.
- In a reasonable QBF proof system P, with underlying PPS Q, for every UNSAT formula F, refuting $\exists.F$ in P no easier than proving unsatisfiability of F in Q.

  So we already have lower bounds.
- Not "genuine QBF hardness".
- Feasible Interpolation gives lower bounds in many QBF systems. Again, not "genuine QBF hardness".
- Prover-Delayer game-based arguments give lower bounds in treelike QRes. Again, not "genuine QBF hardness".

What is "Genuine QBF Hardness"?

Genuine QBF hardness –
not hardness stemming merely from underlying propositional hardness.
Formalising genuineness –

- in expansion systems, seems natural.
- in reduction systems: the NP-oracle.
  Discount deduction steps that employ reasoning checkable by reduction to SAT.

  Effectively, count only reduction steps.
- in systems using merging: Discount deduction steps that employ reasoning without affecting partial information about $P_\forall$ winning strategy.

What is "Genuine QBF Hardness"?

Genuine QBF hardness –
not hardness stemming merely from underlying propositional hardness.
Formalising genuineness –

- in expansion systems, seems natural.
- in reduction systems: the NP-oracle.
  Discount deduction steps that employ reasoning checkable by reduction to SAT.

  Effectively, count only reduction steps.
- in systems using merging: Discount deduction steps that employ reasoning without affecting partial information about $P_\forall$ winning strategy.
- Why stop at NP-oracle? Other oracles – hierarchy....

In many QBF systems, computational hardness can be transferred: Efficient Strategy Extraction.

- Key idea: Proofs contains information about $P_\forall$ winning strategies.

Lower Bounds: Transferring Computational Hardness – 1

In many QBF systems, computational hardness can be transferred: Efficient Strategy Extraction.

- Key idea: Proofs contains information about $P_\forall$ winning strategies.
- For a proof system P, find the correct circuit model M.
  Refutations in P yield circuits in M for $P_\forall$ winning strategies.

Lower Bounds: Transferring Computational Hardness – 1

In many QBF systems, computational hardness can be transferred: Efficient Strategy Extraction.

- Key idea: Proofs contains information about $P_\forall$ winning strategies.
- For a proof system P, find the correct circuit model M. Refutations in P yield circuits in M for $P_\forall$ winning strategies.
- Find function f in P/poly hard in M.
- Using P/poly circuit description, construct false Σ_3 formula where winning strategy must compute f.

Lower Bounds: Transferring Computational Hardness – 2

- In $\mathcal{C}$-Frege+$\forall$red systems, only two sources of hardness:
  - propositional hardness of a related formula, or
  - $\mathcal{C}$ lower bounds.
- From a proof in $\mathcal{C}$-Frege+$\forall$red, efficiently extract
  - a set of witnessing circuits in $\mathcal{C}$, and
  - a propositional proof that the circuits compute a $P_\forall$ winning strategy (witness validation).
- No short proofs for QBFs if every countermodel is either computationally hard, or hard to validate, or both.

  (Thus, lower bounds even for $AC^0[p]$-Frege+$\forall$red.)

Lower Bounds: Transferring Computational Hardness – 3

- Hardness via Size-Width relation: doesn't work for QRes.
- A modified adaptation works for QURes; gives lower bounds for bounded alternation formulas.
  Key idea: Circuit characterisation of QURes proofs.
- Fits the template of transferring computational hardness.

Lower Bounds: Identify semantic hardness

- In systems between QRes and EFrege+$\forall$`red`, a seemingly third source of hardness.
- Formulas with no underlying propositional hardness, and with trivial winning strategies, can be hard.
  - size (of proof), cost (of formula), capacity (of proof system)
  - strategy size, strategy weight
  - formula gauge

Lower Bounds: Identify semantic hardness

- In systems between QRes and EFrege+$\forall$`red`, a seemingly third source of hardness.
- Formulas with no underlying propositional hardness, and with trivial winning strategies, can be hard.
  - size (of proof), cost (of formula), capacity (of proof system)
  - strategy size, strategy weight
  - formula gauge
- Is this really a third source, or is it just that we haven't identified the right circuit model?
- eg The Equality Formulas: cost, weight, gauge, high.
  But winning strategies trivial, projections.
  Still, hard in a multi-output decision-list model – explains QURes hardness.

Questions

- Exploit game semantics better to design new proof systems.
- Harness the power of algebraic reasoning.
  (QBF analogues of static pps?)
- Identify more candidate hard formulas.
  - Exploit succinctness of QBF as opposed to CNF-SAT instance.
  - Mathematical principles? (PHP, Tseitin, mutilated chessboard, ...)
  - Based on computation?
- Formalise the "random formula" model.
- Characterise more proof systems via appropriate circuit classes.
- Understand the sources of hardness.

Talk Plan

Quanified Boolean Formulas QBFs and Formal Proofs

- ✓ QBF basics
- ✓ A Proof Complexity perspective
- A QBF-Solving perspective
- A Computational Complexity perspective
- Some Questions / Directions / Speculations ...

Proof Systems and QBF solving

- CDCL: a nondeterministic template for an algorithm for (UN)SAT.
- CDCL $\equiv$ Resolution.

  Analog in QBF world?

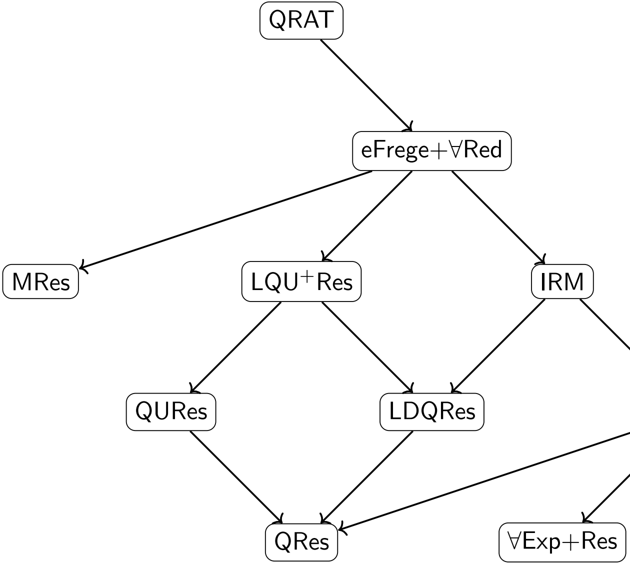
  Which of the QBF Resolution proof systems reflects Q-CDCL?
- Lifting CDCL to QBF: potentially many ways.

  Which algorithm is the right lift? truly Q-CDCL?

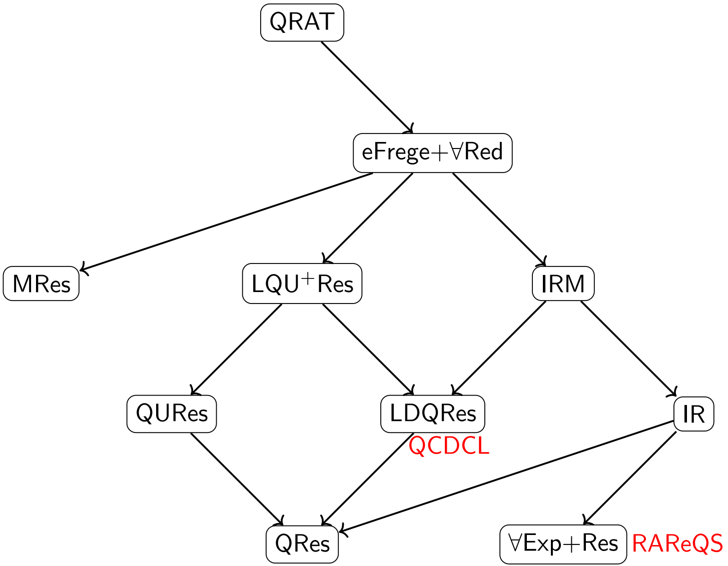
QBF solving

- Expansion-based solvers
- Extending CDCL:
  - decision order policy
  - reduction policy
  - propagation policy
  - conflict analysis
  - pre-processing
- Dependency Schemes
- Dependency Learning
- ...

QBF Proof Systems and Solvers



QBF Proof Systems and Solvers



Q-CDCL: Some surprises? distractions?

- Evaluating QBF as a 2-player game: inherently sequential. Hence Level-Order for decisions reasonable.

  But proof-theoretically, Any-Order is also sound.

Q-CDCL: Some surprises? distractions?

- Evaluating QBF as a 2-player game: inherently sequential. Hence Level-Order for decisions reasonable.

  But proof-theoretically, Any-Order is also sound.

- Solvers don't know a priori whether input is true or false. Treat every assignment as a conflict – either for $P_\exists$ or for $P_\forall$. Learn clauses or cubes. Use cubes too in trails.

  (Suggested Nomenclature: CDL – Conflict-Driven Learning. Conflict-Driven Clause Learning and Conflict-Driven Cube Learning.)

  For false(true) QBFs, learning clauses (cubes) suffices. But learning cubes (clauses) can shorten runs.

Q-CDCL: Some surprises? distractions?

- Evaluating QBF as a 2-player game: inherently sequential.
  Hence Level-Order for decisions reasonable.

  But proof-theoretically, Any-Order is also sound.

- Solvers don't know a priori whether input is true or false.
  Treat every assignment as a conflict – either for $P_\exists$ or for $P_\forall$.
  Learn clauses or cubes. Use cubes too in trails.

  (Suggested Nomenclature: CDL – Conflict-Driven Learning.
  Conflict-Driven Clause Learning and Conflict-Driven Cube Learning.)

  For false(true) QBFs, learning clauses (cubes) suffices.
  But learning cubes (clauses) can shorten runs.

- Dependency schemes never lengthen, and can shorten, proofs.
  But in the QCDCL proof system formalising runs of solvers (with level-ordered decisions) on false QBFs, not always so –
  Using / avoiding dependency schemes gives incomparable systems.

Questions

- Can solvers based on general "QCDCL proof systems" actually be implemented?
- Can solvers based on other QBF proof systems actually be implemented?
- What proof systems characterise the heuristics in determinised QCDCL-style solvers?
- How can Dependency Learning be captured in a proof system?

Talk Plan

Quanified Boolean Formulas QBFs and Formal Proofs

- ✓ QBF basics
- ✓ A Proof Complexity perspective
- ✓ A QBF-Solving perspective
- A Computational Complexity perspective
- Some Questions / Directions / Speculations ...

Sliding the Cook-Reckhow framework

Sliding the Cook-Reckhow framework

Proof Size	Verification Complexity	Complexity Class	
poly	P	NP	known

Sliding the Cook-Reckhow framework

Proof Size	Verification Complexity	Complexity Class	
poly	P	NP	known
poly	P	coNP	$\Longrightarrow$ coNP=NP complementing nondeterministic time

Sliding the Cook-Reckhow framework

Proof Size	Verification Complexity	Complexity Class	
poly	P	NP	known
poly	P	coNP	$\implies$ coNP=NP complementing nondeterministic time
poly	P	PSPACE (NPSPACE,coNPSPACE)	$\implies$ (N)PSPACE=NP collapsing nondeterministic space to time

Sliding the Cook-Reckhow framework

Proof Size	Verification Complexity	Complexity Class	
poly	P	NP	known
poly	P	coNP	$\implies$ coNP=NP complementing nondeterministic time
poly	P	PSPACE (NPSPACE,coNPSPACE)	$\implies$ (N)PSPACE=NP collapsing nondeterministic space to time
poly	P	NEXP	do not exist time hierarchy
exp	P	NEXP	known

Sliding the Cook-Reckhow framework

Proof Size	Verification Complexity	Complexity Class	
poly	P	NP	known
poly	P	coNP	$\Longrightarrow$ coNP=NP complementing nondeterministic time
poly	P	PSPACE (NPSPACE,coNPSPACE)	$\Longrightarrow$ (N)PSPACE=NP collapsing nondeterministic space to time
poly	P	NEXP	do not exist time hierarchy
exp	P	NEXP	known
exp	P	coNEXP	$\Longrightarrow$ coNEXP=NEXP complementing nondeterministic time

Sliding the Cook-Reckhow framework

Proof Size	Verification Complexity	Complexity Class	
poly	P	NP	known
poly	P	coNP	$\Longrightarrow$ coNP=NP complementing nondeterministic time
poly	P	PSPACE (NPSPACE,coNPSPACE)	$\Longrightarrow$ (N)PSPACE=NP collapsing nondeterministic space to time
poly	P	NEXP	do not exist time hierarchy
exp	P	NEXP	known
exp	P	coNEXP	$\Longrightarrow$ coNEXP=NEXP complementing nondeterministic time
?	?	EXP	$\Longrightarrow$ EXP=PSPACE collapsing time to space; removing alternation in space

Multiparty games, DQBF, NEXP

- NP: one-player games; SAT
- PSPACE: (bounded) two-player games; QBF
- NEXP: multiplayer-games; DQBF (Dependency QBFs)
- $$\forall x_1 \forall x_2 \ldots \forall x_n \exists y_1(S_1) \exists y_2(S_2) \ldots y_m(S_m) \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$$
- Proof systems for DQBF: Augment&Lift QBF proof systems. How?
  Expansion-based systems work.
  For many reduction-based systems, either soundness or completeness breaks down.
- Are DQBF solvers for real?!

Questions

- Succinct proofs?
- Proof systems/Solvers for fragments of NEXP?
- QBF proof systems by restricting DQBF/NEXP-style systems rather than augmenting PPS?
- Appropriate formulations of proof-search?

Talk Plan

Quanified Boolean Formulas QBFs and Formal Proofs

✓ QBF basics

✓ A Proof Complexity perspective

✓ A QBF-Solving perspective

✓ A Computational Complexity perspective

- Some Questions / Directions / Speculations ...

Some Directions

- (More) Lower bounds for (more) QBF proof systems
- Understanding QBF solvers better
- "Uniformly" generating partial strategies - can proof complexity help?
- QBFs for optimisation - underlying proof systems
- DQBF solvers and proof systems