Advances in QBF Solving

Mikoláš Janota

IST/INESC-ID,
University of Lisbon, Portugal

Oaxaca, 29 August 2018

Outline

SAT and QBF

- SAT — for a Boolean formula, determine if it is satisfiable
- Example: $(x \vee y) \wedge (x \vee \neg y)$
  $x \triangleq 1$, $y \triangleq 0$
- QBF — for a *Quantified* Boolean formula, determine if it is true
- Example: $\forall x \exists y.\ (x \leftrightarrow y)$
- Quantifications as shorthands for connectives
  ($\forall = \wedge$, $\exists = \vee$)

  Example:

  (1) $\forall x \exists y.\ (x \leftrightarrow y)$
  (2) $\forall x.\ (x \leftrightarrow 0) \vee (x \leftrightarrow 1)$
  (3) $((0 \leftrightarrow 0) \vee (0 \leftrightarrow 1)) \wedge ((1 \leftrightarrow 0) \vee (1 \leftrightarrow 1))$
  (4) 1 (True)

QBF is a strict subset of BernaysfiSchönfinkel (EPR)

- Consider the QBF:

$$(\forall u \exists e)(u \leftrightarrow e)$$

1. Introduce a predicate for truth,
2. each existential variable replace by a predicate,
3. universal variables wrapped by the truth predicate:

$$\text{is-true}(t) \wedge \neg\text{is-true}(f) \wedge$$
$$(\forall u)(\text{is-true}(u) \leftrightarrow p_e(u))$$

- Alternatively, use equality:

$$t \neq f \wedge (\forall u)((u = t) \leftrightarrow p_e(u))$$

Relation to Complexity Theory

$$
\begin{tikzcd}
\vdots & \vdots & & \vdots & \vdots \\
\forall Z \exists Y \forall X.\, \phi & \Pi_3^P & & \Sigma_3^P & \exists Z \forall Y \exists X.\, \phi \\
\forall Y \exists X.\, \phi & \Pi_2^P \arrow[u] & & \Sigma_2^P \arrow[u] & \exists Y \forall X.\, \phi \\
\forall X.\, \phi & \Pi_1^P = \text{co-NP} \arrow[u] & & \Sigma_1^P = \text{NP} \arrow[u] & \exists X.\, \phi \\
 & & \text{P} \arrow[ul] \arrow[ur] & &
\end{tikzcd}
$$

- Deciding QBF is PSPACE complete

Relation to Two-player Games

- In this talk we consider prenex form: *Quantifier-prefix*. *Matrix*
  Example $\forall y_1 y_2 \exists x_1 x_2 .\, (\neg y_1 \vee x_1) \wedge (y_2 \vee \neg x_2)$
- A QBF represents a two-player games between $\forall$ and $\exists$.
- $\forall$ wins a game if the matrix becomes false.
- $\exists$ wins a game if the matrix becomes true.
- A QBF is false iff there exists a winning strategy for $\forall$.
- A QBF is true iff there exists a winning strategy for $\exists$.
  Example

$$\forall u \exists e .\, (u \leftrightarrow e)$$

$\exists$-player wins by playing $e \triangleq u$.

Why Quantified Boolean Formulas?

- “Funamental problem”: PSPACE, 2-player games (fin. space)
- Direct applications
  - model checking (subproblems)
  - (circuit) synthesis
  - non-monotonic reasoning
  - conformant planning
  - ...
- In other reasoners?
  - SMT (e.g. Quantified bit vectors)
  - optimization with quantification (“MaxQBF”)
  - ...

Example: Smallest MUS

Given a CNF ϕ:

- $\vec{s} = \{s_C \mid C \in \phi\}$ are fresh variables
- $\vec{x}$ are the original variables of ϕ
- $k \in \mathbb{N}$
- construct the following QBF:

$$(\exists \vec{s} \forall \vec{x}) \left(\bigvee_{C \in \phi} (s_C \wedge \neg C) \right) \wedge |\vec{s}| \leq k$$

[Ignatiev et al., 2015]

Framework à la CDCL

- Conceptually backtracking algorithm with
- ... unit propagation
- ... clause learning
- ... order heuristics within the same quantifier block

- run propagation in parallel on ϕ and $\neg\phi$
- ϕ false if $\perp$ derived from ϕ
- ϕ true if $\perp$ derived from $\neg\phi$

[Zhang and Malik, 2002, Klieber et al., 2010, Goultiaeva et al., 2013].

Propagation in QCNF

- remove false existential literal (as in SAT)
- remove universal literal that is the innermost w.r.t. prefix in the clause

Example: $\boxed{e_1}$

$$(\exists e_1 e_2 \forall u \exists e_3)$$

$\neg e_1$	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	e_3

Propagation in QCNF

- remove false existential literal (as in SAT)
- remove universal literal that is the innermost w.r.t. prefix in the clause

Example: e_1

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	e_3

Propagation in QCNF

- remove false existential literal (as in SAT)
- remove universal literal that is the innermost w.r.t. prefix in the clause

Example: $\boxed{e_1}$

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	e_3

Propagation in QCNF

- remove false existential literal (as in SAT)
- remove universal literal that is the innermost w.r.t. prefix in the clause

Example: $\boxed{e_1}$

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	

Propagation in QCNF

- remove false existential literal (as in SAT)
- remove universal literal that is the innermost w.r.t. prefix in the clause

Example: e_1

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$		$\vee$	

Propagation in QCNF

- remove false existential literal (as in SAT)
- remove universal literal that is the innermost w.r.t. prefix in the clause

Example: e_1

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$		$\vee$	

Propagation and Learning in QCNF

Example: propagate $\boxed{e_1}$

$(\exists e_1 e_2 \forall u \exists e_3)$

$\neg e_1$	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	e_3
e_2	$\vee$	$\neg u$	$\vee$	e_3

Propagation and Learning in QCNF

Example: propagate e_1

$(\exists e_1 e_2 \forall u \exists e_3)$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	e_3
e_2	$\vee$	$\neg u$	$\vee$	e_3

Example: propagate $\boxed{e_1}$

$(\exists e_1 e_2 \forall u \exists e_3)$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	e_3
e_2	$\vee$	$\neg u$	$\vee$	e_3

Propagation and Learning in QCNF

Example: propagate $\boxed{e_1}$

$(\exists e_1 e_2 \forall u \exists e_3)$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$	u	$\vee$	
e_2	$\vee$	$\neg u$	$\vee$	e_3

Propagation and Learning in QCNF

Example: propagate e_1

$(\exists e_1 e_2 \forall u \exists e_3)$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$		$\vee$	
e_2	$\vee$	$\neg u$	$\vee$	e_3

Propagation and Learning in QCNF

Example: propagate e_1

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$		$\vee$	
e_2	$\vee$	$\neg u$	$\vee$	e_3

Propagation and Learning in QCNF

Example: propagate e_1

$(\exists e_1 e_2 \forall u \exists e_3)$

	$\lor$		$\lor$	$\neg e_3$
$\neg e_2$	$\lor$		$\lor$	
e_2	$\lor$	$\neg u$	$\lor$	

Propagation and Learning in QCNF

Example: propagate $\boxed{e_1}$

$$(\exists e_1 e_2 \forall u \exists e_3)$$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$		$\vee$	
e_2	$\vee$		$\vee$	

Propagation and Learning in QCNF

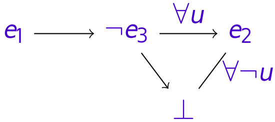
Example: propagate e_1

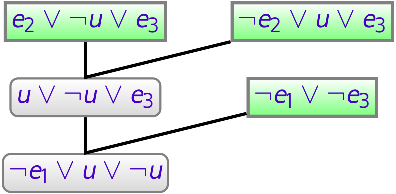
$(\exists e_1 e_2 \forall u \exists e_3)$

	$\vee$		$\vee$	$\neg e_3$
$\neg e_2$	$\vee$		$\vee$	
	$\vee$		$\vee$	

Propagation and Learning in QCNF Contd.

$(\exists e_1 e_2 \forall u \exists e_3)$





Long-distance Q-resolution — Remarks

- The clause $\neg e_1 \vee u \vee \neg u$ immediately propagates $\neg e_1$
- resolution with complementary universal literals: long-distance resolution
- as a proof system, long-distance resolution requires a side condition
- ... always sound when obtained in propagation
- for semantics see [Suda and Gleiss, 2018]

Solving by CEGAR Expansion

$$(\exists \vec{E} \forall \vec{U})\, \phi \equiv (\exists \vec{E}) \bigwedge_{\mu \in 2^{\vec{U}}} \phi[\mu]$$

Can be solved by $\mathtt{SAT}\left(\bigwedge_{\mu \in 2^{\vec{U}}} \phi[\mu]\right)$. Impractical!

Observe:

$$(\exists \vec{E}) \left(\bigwedge_{\mu \in 2^{\vec{U}}} \phi[\mu]\right) \Rightarrow (\exists \vec{E}) \bigwedge_{\mu \in \omega} \phi[\mu]$$

for any $\omega \subseteq 2^{\vec{U}}$

Solving by CEGAR Expansion Contd.

$$(\exists \vec{E} \forall \vec{U})\ \phi \equiv (\exists \vec{E})\ \bigwedge_{\mu \in 2^{\vec{U}}} \phi[\mu]$$

Expand gradually instead: [J. and Marques-Silva, 2011]

- Pick τ_0 arbitrary assignment to $\vec{E}$
- $\mathtt{SAT}(\neg\phi[\tau_0]) = \mu_0$ assignment to $\vec{U}$
- $\mathtt{SAT}(\phi[\mu_0]) = \tau_1$ assignment to $\vec{E}$
- $\mathtt{SAT}(\neg\phi[\tau_1]) = \mu_2$ assignment to $\vec{U}$
- $\mathtt{SAT}(\phi[\mu_0] \wedge \phi[\mu_1]) = \tau_2$ assignment to $\vec{E}$
- After n iterations

$$(\exists \vec{E}\,)\ \bigwedge_{i \in 1..n} \phi[\tau_i]$$

Abstraction-Based Algorithm for a Winning Move

Algorithm for $\exists\forall$ [J. and Marques-Silva, 2011]

Function $\texttt{Solve}((\exists\vec{E}\,\forall\vec{U})\,\phi)$

$\alpha \leftarrow \mathsf{true}$ // start with an empty abstraction

while true **do**

  $\tau \leftarrow \texttt{SAT}(\alpha)$ // find a candidate

  if $\tau = \bot$ **then return** $\bot$

  $\mu \leftarrow \texttt{SAT}(\neg\phi[\vec{E} \leftarrow \tau])$ // find a countermove

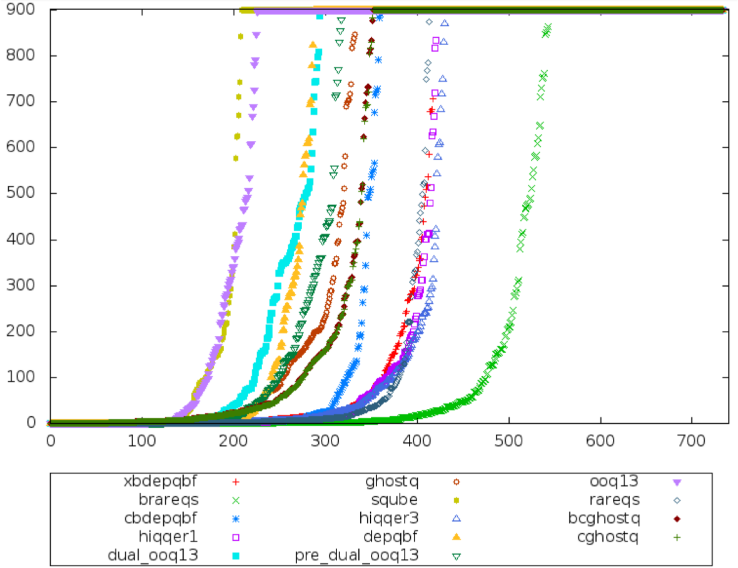
  if $\mu = \bot$ **then return** τ

  $\alpha \leftarrow \alpha \wedge \phi[\vec{U} \leftarrow \mu]$ // refine abstraction

Expansion Continued

- The algorithm is non-CNF
- The algorithm can be generalized
- . . . to arbitrary number of levels by recursion [J. et al., 2012]
- . . . non-prenex [J. et al., 2016].

Results, QBF-Gallery '14, Application Track



Careful Expansion: Good Example

$(\exists x \ldots \forall y \ldots)(\phi \wedge y)$
Setting countermove $y \leftarrow 0$ yields false. Stop.
$(\exists x \ldots \forall y \ldots)(x \vee \phi)$
Setting candidate $x \leftarrow 1$ yields true (impossible to falsify). Stop.

Careful Expansion: Bad Example

$(\exists x \forall y)(x \leftrightarrow y)$

1. $x \leftarrow 1$ candidate
2. $\texttt{SAT}(\neg(1 \leftrightarrow y)) \ldots y \leftarrow 0$ countermove
3. $\texttt{SAT}(x \leftrightarrow 0) \ldots x \leftarrow 0$ candidate
4. $\texttt{SAT}(\neg(0 \leftrightarrow y)) \ldots y \leftarrow 1$ countermove
5. $\texttt{SAT}(x \leftrightarrow 0 \wedge x \leftrightarrow 1) \ldots$ UNSAT Stop

Careful Expansion: Ugly Example

$$(\exists x_1 x_2 \forall y_1 y_2)\left((x_1 \leftrightarrow y_1) \lor (x_2 \leftrightarrow y_2)\right)$$

1. $x_1, x_2 \leftarrow 0, 0$
2. $\mathtt{SAT}(\neg(0 \leftrightarrow y_1 \lor \neg 0 \leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 1$
3. $\mathtt{SAT}(x_1 \leftrightarrow 1 \lor x_2 \leftrightarrow 1) \ldots x_1, x_2 \leftarrow 0, 1$
4. $\mathtt{SAT}(\neg(0 \leftrightarrow y_1 \lor 1 \leftrightarrow y_2)) \ldots y_1 \leftarrow 1, y_2 \leftarrow 0$
5. $\mathtt{SAT}\left((x_1 \leftrightarrow 1 \lor x_2 \leftrightarrow 1) \land (x_1 \leftrightarrow 1 \lor x_2 \leftrightarrow 0)\right) \ldots$
6. $\ldots$

Issue

- CEGAR requires 2^n SAT calls for the formula

$$(\exists x_1 \ldots x_n \forall y_1 \ldots y_n) \bigvee_{i \in 1..n} x_i \leftrightarrow y_i$$

- BUT: We know that the formula is immediately false if we set $y_i \leftarrow \neg x_i$.

$$\left(\exists x_1 \ldots x_n \forall y_1 \ldots y_n. \bigvee_{i \in 1..n} x_i \leftrightarrow \neg x_i\right) \equiv \left(\exists x_1 \ldots x_n. \, 0\right)$$

- Idea: instead of plugging in constants, plug in functions.
- **Where do we get the functions?**

Use Machine Learning

[J., 2018]

1. Enumerate some number of candidate–countermove pairs.
2. Run a machine learning algorithm to learn a Boolean function for each variable in the inner quantifier.
3. Strengthen abstraction with the functions.
4. Repeat.

Machine Learning Example

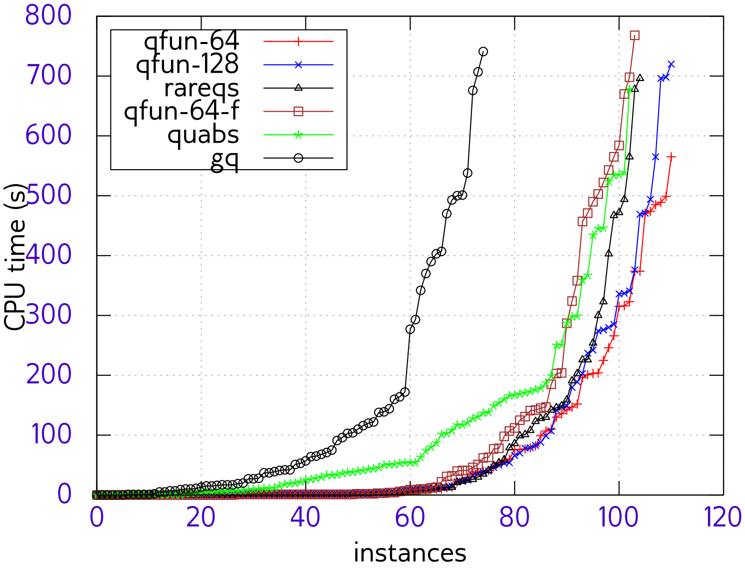
x_1	x_2	$\dots$	x_n	y_1	y_2	$\dots$	y_n
0	0	$\dots$	0	1	1	$\dots$	1
1	0	$\dots$	0	0	1	$\dots$	1
0	0	$\dots$	1	1	1	$\dots$	0
0	1	$\dots$	1	1	0	$\dots$	0

- After 2 steps: $y_1 \leftarrow \neg x_1$, $y_i \leftarrow 1$ for $i \in 2..n$.
- $SAT(x_1 \leftrightarrow \neg x_1 \vee \bigvee_{i \in 2..n} x_i \leftrightarrow 1)$
- After 4 steps: $y_1 \leftarrow \neg x_1$ $y_2 \leftarrow \neg x_2 \dots$
- Eventually we learn the right functions.

Current Implementation

- Use CEGAR as before.
- Recursion to generalize to multiple levels as before.
- Refinement as before.
- Every K refinements, learn new functions from last K samples. Refine with them.
- Learning using decision trees by ID3 algorithm.
- Additional heuristic: If a learned function still works, keep it. "Don't fix what ain't broke."

Current Implementation: Experiments



Challenges

- CNF input harmful because we need to reason about the negation as well
  [Ansótegui et al., 2005, J. and Marques-Silva, 2017]
- but CNF preprocessing useful [Biere et al., 2011]

- Formulas with sure strategies can be hard to solve
- Approach: Machine learning strategies [J., 2018]
- Approach: incremental determinization
  [Rabe and Seshia, 2016]

- Since QBF is a subset of FOL, relation to FOL solvers?

Summary

- QBF natural representation for PSPACE or problems at n^{th} level of polynomial hierarchy
- SAT's CDCL can be lifted to QBF
- An alternative approach: gradually expand quantifiers and then call a SAT solver
- Experiments show that expansion tends to be better on small number of quantifier levels and the other way around.
- An important challenge: find good winning strategies
- One way of tackling: machine learning
- Other approaches?

Thank You for Your Attention!

Questions?

Ansótegui, C., Gomes, C. P., and Selman, B. (2005).
The Achilles' heel of QBF.
In *National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI)*, pages 275–281.

Biere, A., Lonsing, F., and Seidl, M. (2011).
Blocked clause elimination for QBF.
In *The 23rd International Conference on Automated Deduction CADE*.

Goultiaeva, A., Seidl, M., and Biere, A. (2013).
Bridging the gap between dual propagation and CNF-based QBF solving.
In *DATE*, pages 811–814.

Ignatiev, A., Janota, M., and Marques-Silva, J. (2015).
Quantified maximum satisfiability.
Constraints, pages 1–26.

J. (2018).

Towards generalization in QBF solving via machine learning.
In *AAAI Conference on Artificial Intelligence*.

J., Klieber, W., Marques-Silva, J., and Clarke, E. (2016).
Solving QBF with counterexample guided refinement.
Artificial Intelligence, 234:1–25.

J., Klieber, W., Marques-Silva, J., and Clarke, E. M. (2012).
Solving QBF with counterexample guided refinement.
In *SAT*, pages 114–128.

J. and Marques-Silva, J. (2011).
Abstraction-based algorithm for 2QBF.
In *SAT*.

J. and Marques-Silva, J. (2017).
An Achilles' heel of term-resolution.
In *Conference on Artificial Intelligence (EPIA)*, pages 670–680.

Klieber, W., Sapra, S., Gao, S., and Clarke, E. M. (2010).
A non-prenex, non-clausal QBF solver with game-state learning.

In *SAT*.

Rabe, M. N. and Seshia, S. A. (2016).
Incremental determinization.
In *Theory and Applications of Satisfiability Testing (SAT)*, pages 375–392.

Suda, M. and Gleiss, B. (2018).
Local soundness for QBF calculi.
In *Theory and Applications of Satisfiability Testing - SAT*, pages 217–234.

Zhang, L. and Malik, S. (2002).
Conflict driven learning in a quantified Boolean satisfiability solver.
In *ICCAD*.