An Introduction to QBF Solving

Florian Lonsing

Knowledge-Based Systems Group, Vienna University of Technology, Austria
`http://www.kr.tuwien.ac.at/staff/lonsing/`

Second Indian SAT+SMT School
December 6-8 2017, Infosys Mysore Campus, Karnataka, India

This work is supported by the Austrian Science Fund (FWF) under grant S11409-N23.

Introduction (1)

Propositional Logic:

- Formula ϕ over propositional variables, Boolean domain $\mathcal{B} = \{\top, \bot\}$.
- Satisfiability problem (SAT): is ϕ satisfiable?
- NP-completeness of SAT.
- Modelling NP-complete problems in formal verification, AI, ...
- A SAT solver returns a model of ϕ or a proof that ϕ has no model.

Introduction (2)

Success Story of SAT Solving:

- Origins: backtracking algorithms in 1960s [DP60, DLL62].
- Clause learning (CDCL): [SS96, SS99].
- Efficient data structures and heuristics: [MMZ$^+$01].
- SAT solver exploit structure of formulas.
- Despite intractability: many (industrial) applications.

Introduction (3)

Commun. ACM, August 2017, Vol. 60, No. 8 [HK17]

review articles

Problem Solving using SAT:

- Problem encodings.
- Preprocessing (simplification).
- Solving.
- Result checking (proofs).
- Recent prominent example: *Boolean Pythagorean Triples Problem* [HKM16, HK17].

DOI:10.1145/3107239

Mathematics solves problems by pen and paper. CS helps us to go far beyond that.

BY MARIJN J.H. HEULE AND OLIVER KULLMANN

The Science of Brute Force

RECENT PROGRESS IN automated reasoning and supercomputing gives rise to a new era of brute force. The game changer is "SAT," a disruptive, brute-reasoning technology in industry and science. We illustrate its strength and potential via the proof of the Boolean Pythagorean Triples Problem, a long-standing open problem in Ramsey Theory. This 200TB proof has been constructed completely automatically—paradoxically, in an ingenious way. We welcome these bold new proofs emerging on the horizon, beyond human understanding—both mathematics and industry need them.

Many relevant search problems, from artificial intelligence to combinatorics, explore large search spaces to determine the presence or absence of a certain object. These problems are hard due to combinatorial explosion, and have traditionally been called infeasible. The brute-force method, which at least implicitly explores all possibilities, is a general approach to systematically search through such spaces.

Brute force has long been regarded as suitable only for simple problems. This has changed in the last two decades, due to the progress in Satisfiability (SAT) solving, which by adding brute reason renders brute force into a powerful approach to deal with many problems easily and automatically. Search spaces with far more possibilities than the number of particles in the universe may be completely explored.

SAT solving determines whether a formula in propositional logic has a solution, and its brute reasoning acts in a blind and uninformed way—as a feature, not a bug. We focus on applying SAT to mathematics, as a systematic development of the traditional method of proof by exhaustion.

Can we trust the result of running complicated algorithms on many machines for a long time? The strongest solution is to provide a proof, which is also needed to show correctness of highly complex systems, which are everywhere, from finance to health care to aviation.

» key insights

- **Long-standing open problems in mathematics can now be solved completely automatically resulting in clever though potentially gigantic proofs.**
- **Our time requires answers to hard questions regarding safety and security. In these cases knowledge is more important than understanding as long as we can trust the answers.**
- **Powerful SAT-solving heuristics facilitate linear speedups even when using thousands of cores. Combined with the ever-increasing capabilities of high-performance computing clusters they enable solving challenging problems.**

ILLUSTRATION BY PETER CROWTHER ASSOCIATES

Introduction (4)

Quantified Boolean Formulas (QBF):

- Propositional logic extended by existential ($\exists$) / universal ($\forall$) quantification of propositional variables.
- Checking QBF satisfiability: PSPACE-complete.
- Propositional satisfiability (SAT): NP-complete.
- QBF encodings: potentially more succinct than propositional logic.

Example

- QBF $\psi := \hat{Q}.\phi$ in *prenex conjunctive normal form (PCNF)*.
- $\psi = \underbrace{\forall u \exists x.}_{\text{quantifier prefix } \hat{Q}} \underbrace{(\bar{u} \vee x) \wedge (u \vee \bar{x})}_{\text{propositional CNF } \phi}.$

Introduction (5)

Quantifier Alternations in PCNFs:

- A PCNF $Q_1 B_1 Q_2 B_2 \ldots Q_n B_n.\ \phi$ has $n \geq 1$ *quantifier blocks* $Q_i B_i$.
- $Q_i B_i$: sets B_i of variables, quantifiers $Q_i \in \{\forall, \exists\}$ with $Q_i \neq Q_{i+1}$.
- A PCNFs with n quantifier blocks has $n - 1$ *quantifier alternations*.

Example

- PCNF $\psi = \exists x_1, x_2 \forall u_1, u_2 \exists x_3.\phi$.
- ψ has two quantifier alternations.
- Quantifier blocks $\exists B_1$, $\forall B_2$, $\exists B_3$.
- $B_1 : \{x_1, x_2\}$, $B_2 : \{u_1, u_2\}$, $B_3 : \{x_3\}$.

Introduction (5)

Polynomial Hierarchy (PH): cf. [MS72, Sto76, Wra76]

- Framework to describe the complexity of problems beyond NP.
- Satisfiability problem of a given PCNF is located in PH.

Proposition (cf. [BB09, MS72, Sto76, Wra76])

- *Let* $\psi := Q_1 B_1 \dots Q_n B_n.\ \phi$ *be a PCNF with* $k \geq 0$ *alternations.*
- $Q_1 = \exists$*: satisfiability problem of* ψ *is* Σ^P_{k+1}*-complete.*
- $Q_1 = \forall$*: satisfiability problem of* ψ *is* Π^P_{k+1}*-complete.*

Introduction (6): Encoding Problems as QBFs

Class	*Prefix Pattern*	*Problems (e.g.)*
$\Sigma_1^P = NP$	$\exists B_1.\phi$	Checking prop. logic satisfiability
$\Pi_1^P = co\text{-}NP$	$\forall B_1.\phi$	Checking prop. logic validity
Σ_2^P	$\exists B_1 \forall B_2.\phi$	MUS membership testing [JS11, Lib05], encodings of conformant planning [Rin07], ASP-related problems [FR05], abstract argumentation [CDG+15]
Π_2^P	$\forall B_1 \exists B_2.\phi$	
$\vdots$		
PSPACE	$Q_1 B_1 \ldots Q_n B_n.\phi$ (*n* depending on problem instance)	LTL model checking [SC85], NFA language inclusion, games [Sch78]

Example (Bounded Model Checking (BMC) [BCCZ99])

- System S, states of S as a state graph, invariant P.
- Goal: search for a counterexample to P of bounded length k.
- Counterexample: path to reachable state s_k where P violated.

Initial Bad

$$s_0 \longrightarrow s_1 \longrightarrow \dots \longrightarrow s_{k-1} \longrightarrow s_k$$

Example (Bounded Model Checking (BMC) [BCCZ99])

- System S, states of S as a state graph, invariant P.
- Goal: search for a counterexample to P of bounded length k.
- Counterexample: path to reachable state s_k where P violated.

$I(s_0)$ $B(s_k)$

$$s_0 \xrightarrow[T(s_0,s_1)]{} s_1 \xrightarrow[T(\ldots)]{} \ldots \xrightarrow[T(\ldots)]{} s_{k-1} \xrightarrow[T(s_{k-1},s_k)]{} s_k$$

SAT Encoding:

- Initial state predicate $I(s)$, transition relation $T(s, s')$.
- "Bad state" predicate $B(s)$: s is a state where P is violated.
- Error trace of length k: $I(s_0) \wedge T(s_0, s_1) \wedge \ldots \wedge T(s_{k-1}, s_k) \wedge B(s_k)$.

Introduction (7): Compact QBF Encodings

Example (Bounded Model Checking (BMC) [BCCZ99])

- System S, states of S as a state graph, invariant P.
- Goal: search for a counterexample to P of bounded length k.
- Counterexample: path to reachable state s_k where P violated.

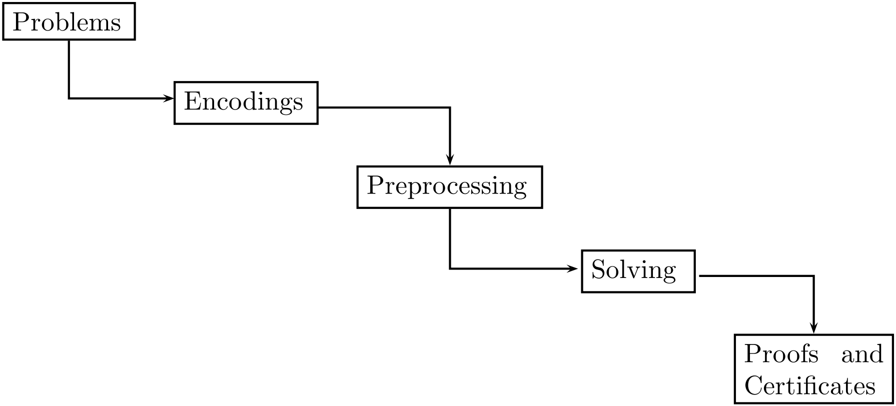
$$I(s_0) \qquad\qquad\qquad\qquad\qquad\qquad B(s_k)$$

$$s_0 \xrightarrow[T(s_0,s_1)]{} s_1 \xrightarrow[T(\ldots)]{} \ldots \xrightarrow[T(\ldots)]{} s_{k-1} \xrightarrow[T(s_{k-1},s_k)]{} s_k$$

QBF Encoding: [BM08, JB07]

- $\exists s_0, \ldots, s_k \forall x, x'.$
  $I(s_0) \wedge B(s_k) \wedge \Big[\big[\bigvee_{i=0}^{k-1}((x = s_i) \wedge (x' = s_{i+1}))\big] \rightarrow T(x, x')\Big].$
- Only one copy of T in contrast to k copies in SAT encoding.

Introduction (8): Typical QBF Workflow



Introduction (9): Progress in QBF Research

The Beginning of QBF Solving:

- 1998: backtracking DPLL for QBF [CGS98].
- 2002: clause learning for QBF (proofs) [GNT02, Let02, ZM02a].
- 2002: expansion (elimination) of variables [AB02].

$\Rightarrow$ compared to SAT (1960s), QBF still is a young field of research!

Introduction (9): Progress in QBF Research

Maturity of QBF Technology:

- QBF *not* yet widely applied at large scale.
- Higher complexity (PSPACE) comes at a cost.

Increased Interest in QBF:

- QBF proof systems: theoretical frameworks of solving techniques.
- CDCL (clause learning) and expansion: orthogonal solving approaches.
- QBF solving by counterexample guided abstraction refinement (CEGAR) [CGJ$^+$03, JM15b, JKMSC16, RT15].

QBF Research Community:

- QBFLIB: `http://www.qbflib.org/index.php`
- QBFEVAL'17: `http://www.qbflib.org/qbfeval17.php`

Introduction (10): Motivating QBF Applications

Synthesis and Realizability of Distributed Systems:

[GT14] Adria Gascón, Ashish Tiwari: A Synthesized Algorithm for Interactive Consistency. NASA Formal Methods 2014: 270-284.

[FT15] Bernd Finkbeiner, Leander Tentrup: Detecting Unrealizability of Distributed Fault-tolerant Systems. Logical Methods in Computer Science 11(3) (2015).

[FFRT17] Peter Faymonville, Bernd Finkbeiner, Markus N. Rabe, Leander Tentrup: Encodings of Bounded Synthesis. TACAS (1) 2017: 354-370.

Introduction (10): Motivating QBF Applications

Solving Dependency Quantified Boolean Formulas (NEXPTIME):

[FT14] Bernd Finkbeiner, Leander Tentrup: Fast DQBF Refutation. SAT 2014: 243-251.

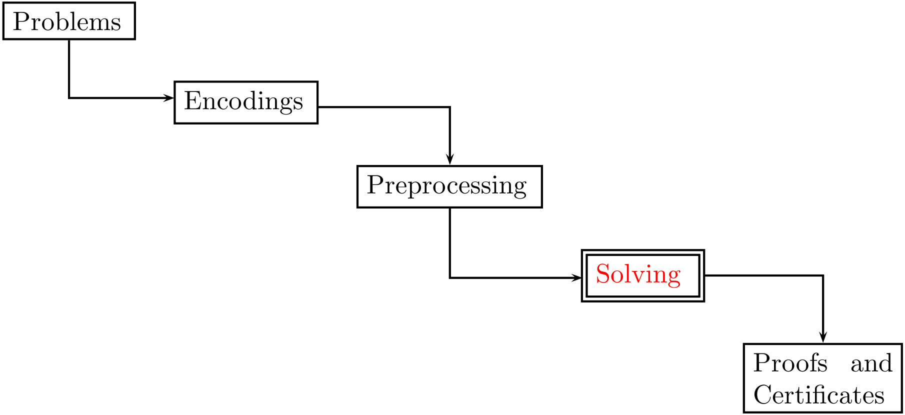
Introduction (10): Motivating QBF Applications

Formal Verification and Synthesis:

[HSM+14] Tamir Heyman, Dan Smith, Yogesh Mahajan, Lance Leong, Husam Abu-Haimed: Dominant Controllability Check Using QBF-Solver and Netlist Optimizer. SAT 2014: 227-242.

[CHR16] Chih-Hong Cheng, Yassine Hamza, Harald Ruess: Structural Synthesis for GXW Specifications. CAV 2016.

Introduction (11): Focus of Tutorial



Our Focus: Search-Based QBF Solving.

Outline of Tutorial

- Preliminaries:
  - Brief recapitulation: propositional logic.
  - QBF syntax and semantics.
- From backtracking search to modern search based QBF solving:
  - Basic backtracking approach.
  - Better assignment generation.
  - Backjumping.
  - Clause learning and Q-resolution.
  - Cube learning.
- QBF proofs and certificates.
- Preprocessing: blocked clause elimination (hands-on session).
- Expansion-based QBF solving.
- Experiments.
- Summary and conclusion.

Propositional Logic (1)

Definition (Basic Definitions)

- Boolean domain $\mathcal{B} = \{\top, \bot\}$: truth values "true" and "false".
- Boolean variables $Vars = \{x, y, \ldots\}$ (arbitrarily many but finite).
- Assignment $A : Vars \to \mathcal{B}$

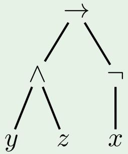
Propositional Logic (1)

Definition (Propositional Formulas (PF))

- $\top$ and $\bot$ are PFs.
- For propositional variables *Vars*, (x) where $x \in Vars$ is a PF.
- If ψ is a PF then $\neg(\psi)$ is a PF.
- To save space in notation, we also write $\bar{x}$ instead of $\neg x$.
- If ψ_1 and ψ_2 are PFs then $(\psi_1 \circ \psi_2)$ is a PF, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

Example

$\psi := (y \wedge z) \rightarrow \neg(x)$



Propositional Logic (1)

Definition (Conjunctive Normal Form (CNF))

- A literal l is a variable x or its negation $\bar{x}$.
- A clause $C = (l_1 \vee \ldots \vee l_m)$ is a disjunction over literals.
- A formula is in *CNF* if it consists of a conjunction of clauses.

Propositional Logic (2)

Definition (CNF Semantics)

- Given a CNF ϕ and an assignment A to the variables in ϕ.
- $\phi[A]$: replace variables x in ϕ by $\top$ ($\bot$) if $A(x) = \top$ ($A(x) = \bot$).
- We write $A := \{x\}$ if $A(x) = \top$ and $A := \{\bar{x}\}$ if $A(x) = \bot$.
- CNF ϕ is satisfiable iff there exists A such that $\phi[A] = \top$. Otherwise, ϕ is unsatisfiable.

Example

- $\phi := (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.
- Models M and M' of ϕ:
  - $M := \{x, y\}$ where $M(x) = M(y) = \top$.
  - $M' := \{\bar{x}, \bar{y}\}$ where $M'(x) = M'(y) = \bot$.

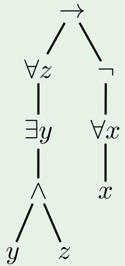
Syntax (1)

QBFs as Quantified Circuits:

- $\top$ and $\bot$ are QBFs.
- For propositional variables *Vars*, (x) where $x \in \mathit{Vars}$ is a QBF.
- If ψ is a QBF then $\neg(\psi)$ is a QBF.
- If ψ_1 and ψ_2 are QBFs then $(\psi_1 \circ \psi_2)$ is a QBF, $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.
- If ψ is a QBF and $x \in \mathit{Vars}(\psi)$, then $\forall x.(\psi)$ and $\exists x.(\psi)$ are QBFs.

Example

$$\psi := (\forall z.(\exists y.(y \wedge z))) \rightarrow \neg(\forall x.(x))$$



Syntax (1)

QBFs in Prenex CNF: $\psi := \hat{Q}.\phi$

- Quantifier prefix $\hat{Q} = Q_1 B_1 \ldots Q_n B_n$, $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_j$, $B_i \subseteq Vars$, $(B_i \cap B_j) = \emptyset$.
- Linear ordering of variables: $x_i < x_j$ iff $x_i \in B_i$, $x_j \in B_j$, and $i < j$.
- Quantifier-free CNF ϕ over propositional variables x_i.
- Assume: ϕ does not contain free variables, all x_i in $\hat{Q}$ appear in ϕ.

Example

- PCNF $\psi = \forall u \exists x.(\bar{u} \vee x) \wedge (u \vee \bar{x})$.
- Linear ordering: $u < x$.

Syntax (2)

Example (QDIMACS Format)

$\exists x_1, x_3, x_4 \forall y_5 \exists x_2.$
$(\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

- Extension of DIMACS format used in SAT solving.
- Literals of variables encoded as signed integers.
- One quantifier block per line, terminated by zero.
- "a" labels $\forall$, "e" labels $\exists$.
- One clause per line, terminated by zero.

```
p cnf 5 4
e 1 3 4 0
a 5 0
e 2 0
-1 2 0
3 5 -2 0
4 -5 -2 0
-3 -4 0
```

QDIMACS format: `http://www.qbflib.org/qdimacs.html`

Semantics (1)

Recursive Definition:

- Assume that a QBF does not contain free variables.
- The QBF $\perp$ is unsatisfiable, the QBF $\top$ is satisfiable.
- The QBF $\neg(\psi)$ is satisfiable iff the QBF ψ is unsatisfiable.
- The QBF $\psi_1 \wedge \psi_2$ is satisfiable iff ψ_1 and ψ_2 are satisfiable.
- The QBF $\psi_1 \vee \psi_2$ is satisfiable iff ψ_1 or ψ_2 is satisfiable.
- The QBF $\forall x.(\psi)$ is satisfiable iff $\psi[\neg x]$ and $\psi[x]$ are satisfiable.
  The QBF $\psi[\neg x]$ ($\psi[x]$) results from ψ by replacing x in ψ by $\perp$ ($\top$).
- The QBF $\exists x.(\psi)$ is satisfiable iff $\psi[\neg x]$ or $\psi[x]$ is satisfiable.

Definition

The QBFs ψ and ψ' are *satisfiability-equivalent* ($\psi \equiv_{sat} \psi'$) iff ψ is satisfiable whenever ψ' is satisfiable.

Semantics (1)

Example

Observe: recursive evaluation assigns variables in prefix ordering.

The PCNF $\psi = \forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$ is satisfiable if

(1) $\psi[x] = \exists y.(y)$ and

(2) $\psi[\bar{x}] = \exists y.(\bar{y})$ are satisfiable.

(1) $\psi[x] = \exists y.(y)$ is satisfiable since $\psi[x, y] = \top$ is satisfiable.
(2) $\psi[\bar{x}] = \exists y.(\bar{y})$ is satisfiable since $\psi[\bar{x}, \bar{y}] = \top$ is satisfiable.

Semantics (1)

Example

Observe: recursive evaluation assigns variables in prefix ordering.

The PCNF $\psi = \exists y \forall x.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$ is unsatisfiable because neither

(1) $\psi[y] = \forall x.(x)$ nor

(2) $\psi[\bar{y}] = \forall x.(\bar{x})$ is satisfiable.

(1) $\psi[y] = \forall x.(x)$ is unsatisfiable since $\psi[y, \bar{x}]$ is unsatisfiable.
(2) $\psi[\bar{y}] = \forall x.(\bar{x})$ is unsatisfiable since $\psi[\bar{y}, x]$ is unsatisfiable.

Semantics (2)

Game-Based View:

- Player $P_\exists$ ($P_\forall$) assigns existential (universal) variables.
- Goal: $P_\exists$ ($P_\forall$) wants to satisfy (falsify) the formula.
- Players pick variables from left to right wrt. quantifier ordering.
- QBF ψ is satisfiable (unsatisfiable) iff $P_\exists$ ($P_\forall$) has a winning strategy.
- Winning strategy: $P_\exists$ ($P_\forall$) can satisfy (falsify) the formula regardless of opponent's choice of assignments.
- Close relation between winning strategies and QBF certificates.

Example

$\psi = \forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$.

- $P_\exists$ wins by setting y to the same value as x.

Backtracking Search

- DPLL algorithm [DLL62] for QBF: QDPLL [CGS98, CSGG02].
- Chronological backtracking (QBF semantics), nonrecursive in practice.

```
bool qdpll (PCNF Q{x}ψ, Assignment A)
   /* 1. Simplify under given assignment. */
       ψ' := simplify(Q{x}ψ[A]);
   /* 2. Check base cases. */
       if (ψ' == ⊥)
         return false;
       if (ψ' == ⊤)
         return true;
   /* 3. Decision making, backtracking. */
       if (Q == ∃)
         return qdpll (ψ', A ∪ {¬x}) ||
                qdpll (ψ', A ∪ {x});
       if (Q == ∀)
         return qdpll (ψ', A ∪ {¬x}) &&
                qdpll (ψ', A ∪ {x});
```

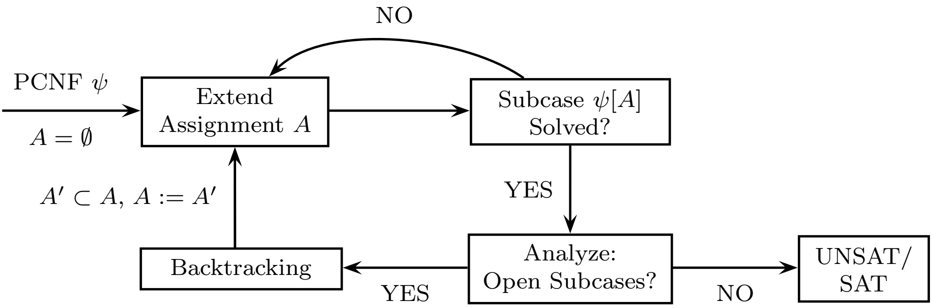
Backtracking Search and Recursive Semantics

Example (continued)

The PCNF $\psi = \forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$ is satisfiable:

- Assign x: $\psi[x] = \exists y.(y)$
  - Assign $\bar{y}$: $\psi[x, \bar{y}] = \bot$ unsatisfiable.
  - Backtrack, assign y: $\psi[x, y] = \top$ satisfiable.
- One subcase of $\forall x$ completed.
- Assign $\bar{x}$: $\psi[\bar{x}] = \exists y.(\bar{y})$
  - Assign y: $\psi[\bar{x}, y] = \bot$ unsatisfiable.
  - Backtrack, assign $\bar{y}$: $\psi[\bar{x}, \bar{y}] = \top$ satisfiable.

Backtracking Search: Abstract Workflow



- Assignment A extended tentatively (decision making, splitting).
- Termination: no open subcases left, depending on quantifier type.
- Backtracking: flipping of assignments depending on subcase.

$\Rightarrow$ refine workflow step by step.

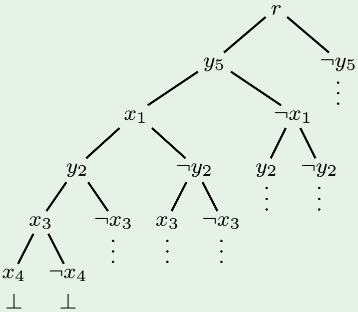
The Need for Better Assignment Generation

Example

$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

- Worst case: 2^5 branches to be explored by backtracking search.
- However: with better assignment generation, exploring a single branch is sufficient!
- **Goal:** make assignments that do not have to be flipped.

Search tree:



Boolean Constraint Propagation for QBF (1/5)

Definition (Unit Literal Detection [CGS98])

- Given a QBF ψ, a clause $C \in \psi$ is *unit* iff $C = (l)$ and $q(l) = \exists$.
- The existential literal l in C is called a *unit literal*.
- *Unit literal detection* $UL(C) := \{l\}$ collects the assignment $\{l\}$ from the unit clause $C = (l)$.
- Unit literal detection on a QBF ψ: $UL(\psi) := \bigcup_{C \in \psi} UL(C)$.

Example

$\psi := \forall y \exists x_1, x_2.(x_2) \wedge (\neg y \vee \neg x_2) \wedge (\neg y \vee x_1)$.
Clause (x_2) is unit: $UL(\psi) = \{x_2\}$.

Boolean Constraint Propagation for QBF (2/5)

Definition (Pure Literal Detection [CGS98])

- A literal l is *pure* in a QBF ψ if there are clauses which contain l but no clauses which contain $\neg l$.
- *Pure literal detection* $PL(\psi) := \bigcup\{l'\}$ collects the assignment $\{l'\}$ such that l is pure and $l' := l$ if $q(l) = \exists$ and $l' := \neg l$ if $q(l) = \forall$.
- The variable of an existential (universal) pure literal is assigned so that clauses are satisfied (not satisfied) by that assignment.

Example (continued)

$\psi := \forall y \exists x_1, x_2.(x_2) \wedge (\neg y \vee \neg x_2) \wedge (\neg y \vee x_1)$.
The universal literal $\neg y$ is pure: $PL(\psi) = \{y\}$.
$\psi[y] := \exists x_1, x_2.(x_2) \wedge (\neg x_2) \wedge (x_1)$.

Boolean Constraint Propagation for QBF (3/5)

Definition (Universal Reduction [BKF95])

Given a clause C, *universal reduction (UR)* of C produces the clause

$$UR(C) := C \setminus \{l \in C \mid q(l) = \forall, \forall l' \in C \text{ with } q(l') = \exists : var(l') < var(l)\}$$

where $<$ is the linear variable ordering given by the quantifier prefix.

- UR deletes locally "trailing" universal literals, i.e., shortens clauses.

Example (continued)

$\psi := \forall y \exists x_1, x_2.(x_2) \wedge (\neg y \vee \neg x_2) \wedge (\neg y \vee x_1)$.
By UL: $\psi[x_2] := \forall y \exists x_1.(\neg y) \wedge (\neg y \vee x_1)$.
In $\psi[x_2]$: $UR((\neg y)) = \emptyset$.

Boolean Constraint Propagation for QBF (4/5)

Definition

Boolean Constraint Propagation for QBF (QBCP):

- Given a PCNF ψ and the empty assignment $A = \{\}$, i.e. $\psi[A] = \psi$.
  1. Apply universal reduction (UR) to $\psi[A]$.
  2. Apply unit literal detection (UL) to $\psi[A]$ to get new assignments.
  3. Apply pure literal detection (PL) to $\psi[A]$ to find new assignments.
- Add assignments found by UL and PL to A, repeat steps 1-3.
- Stop if A does not change anymore or if $\psi[A] = \top$ or $\psi[A] = \bot$.

Boolean Constraint Propagation for QBF (5/5)

Properties of QBCP:

- QBCP takes a PCNF ψ and an assignment A and produces an extended assignment A' and a PCNF $\psi' = \psi[A']$ by UL, PL, and UR.
- Soundness: $\psi \equiv_{sat} \psi'$ (satisfiability-equivalence).
- No prefix ordering restriction: QBCP potentially assigns any variables.

QBCP in Practice:

- Combine decision making and QBCP.
- Successively apply QBCP after assigning some x as decision.
- Backtracking: no need to flip assignments made in QBCP.

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot$.
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \bot$.
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \bot \equiv_{sat} \psi$.

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot$.
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \bot$.
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \bot \equiv_{sat} \psi$.

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3).$
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3).$
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot.$
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \bot.$
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \bot \equiv_{sat} \psi.$

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3).$
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3).$
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot.$
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \bot.$
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \bot \equiv_{sat} \psi.$

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$ $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$ $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \perp$.
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \perp$.
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \perp \equiv_{sat} \psi$.

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot$.
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \bot$.
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \bot \equiv_{sat} \psi$.

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3).$
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3).$
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \perp.$
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \perp.$
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \perp \equiv_{sat} \psi.$

QBCP Example

Example

- $\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
  $(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- No simplifications of ψ by QBCP possible, make decision: $A = \{y_5\}$.
- $\psi[y_5] =$
  $\exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3)$.
- By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3)$.
- By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3)$.
- By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot$.
- By QBCP, we have shown: $\psi[y_5] \equiv_{sat} \psi[y_5, x_4, y_2, x_1, x_3] \equiv_{sat} \bot$.
- Since y_5 is universal: $\psi[y_5] \equiv_{sat} \bot \equiv_{sat} \psi$.

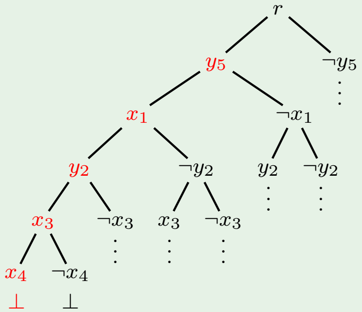
Benefits of QBCP

Example

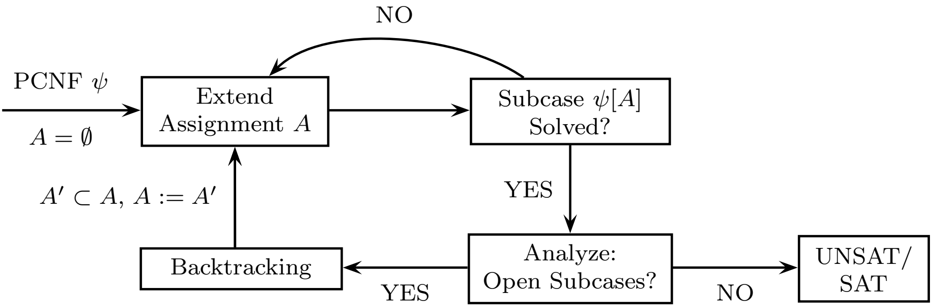
$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

- Worst case: 2^5 branches to be explored by backtracking search.
- Only one branch explored.
- One decision + QBCP.
- **Goal:** integrate QBCP in workflow for better assignment generation.

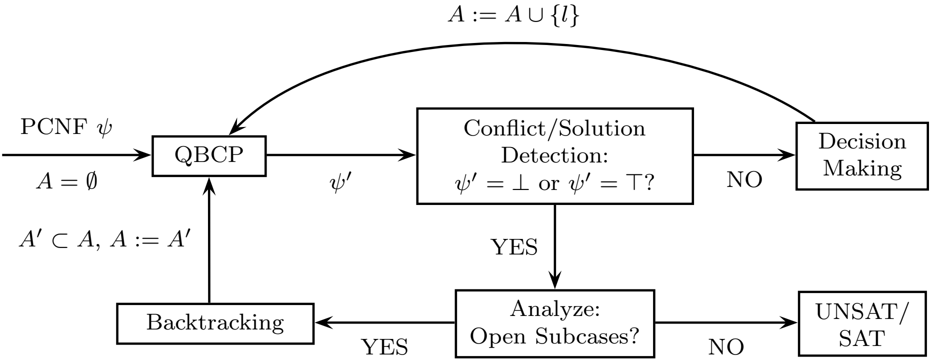
Search tree:



Backtracking Search: Previous Abstract Workflow



Backtracking Search: Refined Abstract Workflow



- QBCP influences assignment generation and detecting solved subcases.
- In the following, we focus on conflicts, i.e., unsatisfiable subcases.
- Need better ways of analyzing open subcases.

Implication Graphs (1/2)

Definition (Implication Graph (IG))

- Let ψ be the original QBF.
- Vertices: literals (assignments) in A made as decisions or by UL.
- Special vertex $\emptyset$ denoting a clause $C \in \psi$ such that $C[A] = \perp$ by UR.
- For assignments $\{l\}$ by UL from a unit clause $C[A]$: the clause $ante(l) := C$ with $C \in \psi$ is the *antecedent clause* of assignment $\{l\}$.
- Define $ante(\emptyset) = C$, for a clause $C \in \psi$ such that $C[A] = \perp$.
- Edges: $(x, y) \in E$ if y assigned by UL and literal $\neg x \in ante(y)$.

Implication Graphs (2/2)

- Antecedent clauses in the original PCNF ψ are recorded.
- Implication graphs are constructed on the fly during QBCP.
- On the fly construction requires efficient data structures [GGN$^+$04].
- *Conflict*: assignment A such that QBCP on $\psi[A]$ produces empty clause $\emptyset$.
- *Conflict graph*: implication graph containing empty clause $\emptyset$.

Constructing IGs On The Fly: Example

Example (formula from above)

$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

Make decision: $A = \{y_5\}$.
$\psi[y_5] = \exists x_1 \forall y_2 \exists x_3, x_4.(x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

Implication Graph:

y_5

Antecedents:

Constructing IGs On The Fly: Example

Example (formula from above)

$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

By UL: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1 \vee y_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$
By UR: $\psi[y_5, x_4] = \exists x_1 \forall y_2 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg y_2 \vee \neg x_3).$

Implication Graph:

$y_5 \longrightarrow x_4$

Antecedents:

$ante(x_4): \quad (\neg y_5 \vee x_4)$

Constructing IGs On The Fly: Example

Example (formula from above)

$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

By PL: $\psi[y_5, x_4, y_2] = \exists x_1 \exists x_3.(x_1) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_3).$
By UL: $\psi[y_5, x_4, y_2, x_1] = \exists x_3.(x_3) \wedge (\neg x_3).$

Implication Graph:

$$y_5 \longrightarrow x_4 \longrightarrow x_1$$

Antecedents:

$ante(x_4)$: $(\neg y_5 \vee x_4)$
$ante(x_1)$: $(x_1 \vee y_2 \vee \neg x_4)$

Constructing IGs On The Fly: Example

Example (formula from above)

$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \vee x_4) \wedge (y_5 \vee \neg x_4) \wedge (x_1 \vee y_2 \vee \neg x_4) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (\neg y_2 \vee \neg x_3).$

By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \bot$.

Implication Graph:

$$\begin{tikzcd} y_5 \arrow[r] & x_4 \arrow[r] \arrow[rr, bend right] & x_1 \arrow[r] & x_3 \end{tikzcd}$$

Antecedents:

$ante(x_4)$: $(\neg y_5 \vee x_4)$
$ante(x_1)$: $(x_1 \vee y_2 \vee \neg x_4)$
$ante(x_3)$: $(\neg x_1 \vee x_3 \vee \neg x_4)$

Constructing IGs On The Fly: Example

Example (formula from above)

$\psi = \forall y_5 \exists x_1 \forall y_2 \exists x_3, x_4.$
$(\neg y_5 \lor x_4) \land (y_5 \lor \neg x_4) \land (x_1 \lor y_2 \lor \neg x_4) \land (\neg x_1 \lor x_3 \lor \neg x_4) \land (\neg y_2 \lor \neg x_3).$

By UL: $\psi[y_5, x_4, y_2, x_1, x_3] = \perp$.

Implication Graph:

$$\begin{tikzcd} y_5 \arrow[r] & x_4 \arrow[r] \arrow[rr, bend right] & x_1 \arrow[r] & x_3 \arrow[r] & \emptyset \end{tikzcd}$$

Antecedents:

$$\begin{aligned} ante(x_4) &: \quad (\neg y_5 \lor x_4) \\ ante(x_1) &: \quad (x_1 \lor y_2 \lor \neg x_4) \\ ante(x_3) &: \quad (\neg x_1 \lor x_3 \lor \neg x_4) \\ ante(\emptyset) &: \quad (\neg y_2 \lor \neg x_3) \end{aligned}$$

Decisions: Levelized Implication Graphs

- Initially, the solver is at *decision level* L_0.
- Every decision increases the current level L_i by one to get L_{i+1}.
- Assignments by QBCP are added to the current level L_i.

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- No unit clauses present, level L_0 empty.

$L_0 :$

Decisions: Levelized Implication Graphs

- Initially, the solver is at *decision level* L_0.
- Every decision increases the current level L_i by one to get L_{i+1}.
- Assignments by QBCP are added to the current level L_i.

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi$.

- Decisions on x_1.
- QBCP has no effect.

$L_0 :$

$L_1 : x_1$

Decisions: Levelized Implication Graphs

- Initially, the solver is at *decision level* L_0.
- Every decision increases the current level L_i by one to get L_{i+1}.
- Assignments by QBCP are added to the current level L_i.

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Decisions on x_1, x_2.
- QBCP has no effect.

$L_0 :$

$L_1 : x_1$

$L_2 : x_2$

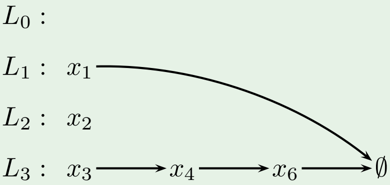
Decisions: Levelized Implication Graphs

- Initially, the solver is at *decision level* L_0.
- Every decision increases the current level L_i by one to get L_{i+1}.
- Assignments by QBCP are added to the current level L_i.

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Decisions on x_1, x_2, x_3: $A = \{x_1, x_2, x_3\}$.
- By QBCP (UL,UR): conflict $A = \{x_1, x_2, x_3, x_4, x_6\}$ at level L_3.



Analyzing Open Subcases (1/2)

Assignments:

- Represented as sequence $A = \{l_1, l_2, \ldots, l_n\}$ of literals.
- Assignments due to decision making and QBCP (UL, PL).
- Literals $l_i \in A$ are ordered chronologically as they were assigned.
- *Conflict*: assignment A such that $\psi[A] = \bot$ under QBCP.
- *Solution*: assignment A such that $\psi[A] = \top$ under QBCP.

$\Rightarrow$ we focus on conflicts and unsatisfiable QBFs.

Analyzing Open Subcases (2/2)

Chronological Backtracking:

- Given a conflict $A = \{\dots, d, \dots, l_n\}$, let d be the most-recent *unflipped* existential decision.
- No such d in A: formula solved.
- Retract decision d and all later assignments: $A' = A \setminus \{d, \dots, l_n\}$.
- Set the variable of d to the opposite value (flip): $A' = A' \cup \{\neg d\}$.
- Continue with $A = A'$.

$\Rightarrow$ similar approach for solutions and satisfiable QBFs.

Chronological Backtracking: Example (1/2)

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Assume that ϕ contains further clauses.

Chronological Backtracking: Example (1/2)

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Decisions on x_1, x_2, x_3: $A = \{x_1, x_2, x_3\}$.
- $\psi[x_1, x_2, x_3] = \exists x_4 \forall y_5 \exists x_6 . (x_4) \wedge (\neg x_4 \vee x_6) \wedge (y_5 \vee \neg x_6) \wedge \phi$.
- By QBCP (UL,UR): conflict $A_1 = \{x_1, x_2, {\color{red}x_3}, {\color{red}x_4}, {\color{red}x_6}\}$.

Implication Graph of conflict A_1:

$$
\begin{tikzcd}
L_0 : & & & & \\
L_1 : & x_1 \arrow[rrrdd, bend left] & & & \\
L_2 : & x_2 & & & \\
L_3 : & {\color{red}x_3} \arrow[r] & {\color{red}x_4} \arrow[r] & {\color{red}x_6} \arrow[r] & \emptyset
\end{tikzcd}
$$

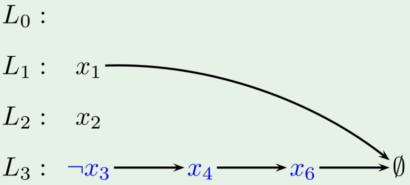
Chronological Backtracking: Example (1/2)

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi$.

- Flip most recent unflipped decision x_3: $A = \{x_1, x_2, \neg x_3\}$.
- $\psi[x_1, x_2, \neg x_3] = \exists x_4 \forall y_5 \exists x_6 . (x_4) \wedge (\neg x_4 \vee x_6) \wedge (y_5 \vee \neg x_6) \wedge \phi$.
- Conflict $A_2 = \{x_1, x_2, \neg x_3, x_4, x_6\}$, by UL,UR.

Implication Graph of conflict A_2:



Chronological Backtracking: Example (1/2)

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6. (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Flip most recent unflipped x_2, decision on x_3: $A = \{x_1, \neg x_2, x_3\}$.
- $\psi[x_1, \neg x_2, x_3] = \exists x_4 \forall y_5 \exists x_6. (x_4) \wedge (\neg x_4 \vee x_6) \wedge (y_5 \vee \neg x_6) \wedge \phi$.
- Conflict $A_3 = \{x_1, \neg x_2, x_3, x_4, x_6\}$ by UL,UR.

Implication Graph of conflict A_3:

$$\begin{tikzcd}
L_0 : & & & & \\
L_1 : & x_1 \arrow[rrrdd, bend left] & & & \\
L_2 : & \neg x_2 & & & \\
L_3 : & x_3 \arrow[r] & x_4 \arrow[r] & x_6 \arrow[r] & \emptyset
\end{tikzcd}$$

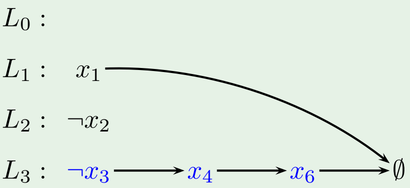
Chronological Backtracking: Example (1/2)

Example

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6.(\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Flip most recent unflipped decision x_3: $A = \{x_1, \neg x_2, \neg x_3\}$.
- $\psi[x_1, \neg x_2, \neg x_3] = \exists x_4 \forall y_5 \exists x_6.(x_4) \wedge (\neg x_4 \vee x_6) \wedge (y_5 \vee \neg x_6) \wedge \phi$.
- Conflict $A_4 = \{x_1, \neg x_2, \neg x_3, x_4, x_6\}$ by UL,UR.

Implication Graph of conflict A_4:



- Repeated assignments $\{x_3, x_4, x_6\}$, $\{\neg x_3, x_4, x_6\}$ in A_1, A_3 and A_2, A_4.

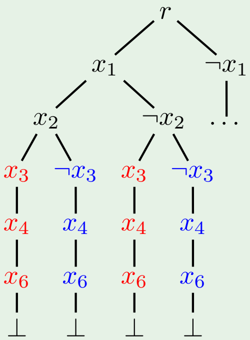
Chronological Backtracking: Example (2/2)

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6.(\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

Conflicts generated:

- $A_1 = \{x_1, x_2, x_3, x_4, x_6\}$.
- $A_2 = \{x_1, x_2, \neg x_3, x_4, x_6\}$.
- $A_3 = \{x_1, \neg x_2, x_3, x_4, x_6\}$.
- $A_4 = \{x_1, \neg x_2, \neg x_3, x_4, x_6\}$.
- Same conflicting subtrees after flipping x_2.
- Decision x_2 is irrelevant in this context.



Drawbacks of Chronological Backtracking:

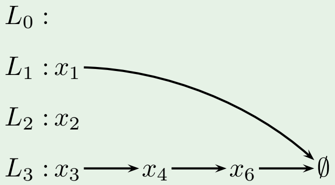
- Flipping variables which are irrelevant for the current conflict.
- Repeating subassignments of previous conflicts: needless branching.

- Given: conflict $A = \{l_1, l_2, \ldots, l_n\}$ and its implication graph (IG).
- Start at node $\emptyset$ and traverse IG backwards towards decision nodes.
- Compute *conflict set (CS)*: collect all decisions d_i reachable from $\emptyset$.
- $CS := \{d_1, \ldots, d_{i-1}, d_i, \ldots, d_k\}$ where $CS \subseteq A$.

Example (continued)

$\psi = \exists x_1,x_2,x_3,x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi$.
Consider conflict $A_1 = \{x_1, x_2, x_3, x_4, x_6\}$ with decisions x_1,x_2,x_3.

- $CS := \{x_1, x_3\}$.



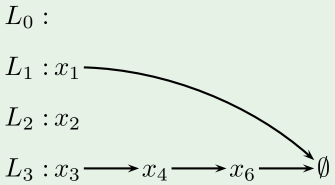
Non-Chronological Backtracking: Backjumping (1/2)

- Let $d_i \in CS$ be the *most recent unflipped existential decision*.
- No such d_i: formula solved (i.e., unsatisfiable).
- Decision $d_{i-1} \in CS$ was assigned *before* d_i most recently in CS.

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$
Consider conflict $A_1 = \{x_1, x_2, x_3, x_4, x_6\}$ with decisions x_1, x_2, x_3.

- $CS := \{x_1, x_3\}$.
- $d_{i-1} = x_1$, $d_i = x_3$.



Non-Chronological Backtracking: Backjumping (1/2)

- Update A by retracting *all assignments* made *after* the level of d_{i-1}.
- Flip value of d_i by making a new decision: $A := A \cup \{\neg d_i\}$.
- Backjumping relies on a more fine-grained analysis of the IG.
- To emulate chron. backtracking, let CS contain all decisions made.

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi$.
Consider conflict $A_1 = \{x_1, x_2, x_3, x_4, x_6\}$ with decisions x_1, x_2, x_3.

- $d_{i-1} = x_1$, $d_i = x_3$.
- Retract $\{x_2, x_3, x_4, x_6\}$ from A.
- Flip x_3, $A \cup \{\neg x_3\} = \{x_1, \neg x_3\}$.

$L_0 :$

$L_1 : x_1$

$L_2 : \neg x_3$

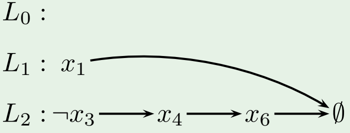
Non-Chronological Backtracking: Backjumping (1/2)

- Update A by retracting *all assignments* made *after* the level of d_{i-1}.
- Flip value of d_i by making a new decision: $A := A \cup \{\neg d_i\}$.
- Backjumping relies on a more fine-grained analysis of the IG.
- To emulate chron. backtracking, let CS contain all decisions made.

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi$.
Consider conflict $A_1 = \{x_1, x_2, x_3, x_4, x_6\}$ with decisions x_1, x_2, x_3.

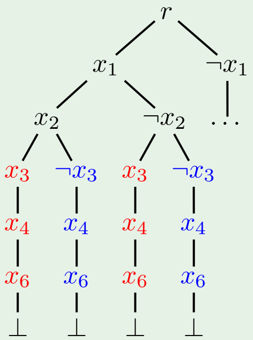
- New conflict $A = \{x_1, \neg x_3, x_4, x_6\}$.
- $CS := \{x_1, \neg x_3\}$, $\neg x_3$ flipped already.
- $d_i = x_1$, retract entire A (no d_{i-1}).
- Flip x_1, $A \cup \{\neg x_1\} = \{\neg x_1\}$.



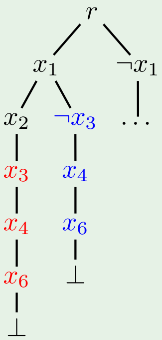
Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6 . (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

Chronological backtracking:



Non-chronological backtracking:



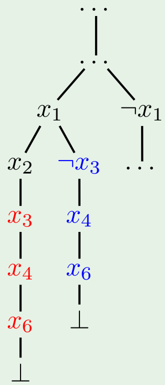
- Backjumping potentially avoids irrelevant branches.
- Similar approaches for satisfiable QBFs.

Drawback of Backjumping

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6.(\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- Assume that the assignment tree on the right is a subtree of a bigger tree.
- Observation: every assignment A with $\{x_1, x_4\} \subseteq A$ is a conflict (under QBCP).
- UL extends $\{x_1, x_4\}$ to $\{x_1, x_4, x_6\}$.
- $C := (\neg x_1 \vee y_5 \vee \neg x_6)$ is empty under $A := \{x_1, x_4, x_6\}$ and QBCP.
- Repeating $\{x_1, x_4\} \subset A$ in other branches falsifies the same clause C under QBCP.
- Backjumping cannot avoid this problem.



Towards Conflict Driven Clause Learning (QCDCL)

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6. (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

- $C := (\neg x_1 \vee y_5 \vee \neg x_6)$ is empty under $A := \{x_1, x_4, x_6\}$ and QBCP.
- Repeating $\{x_1, x_4\} \subset A$ falsifies the same clause C under QBCP.
- Adding the new clause $C_L := (\neg x_1 \vee \neg x_4)$ to the given formula ψ prevents repetition of subassignment $\{x_1, x_4\}$.
- Assigning x_1 (x_4) to *true* triggers assignment of $\neg x_4$ ($\neg x_1$) by UL.

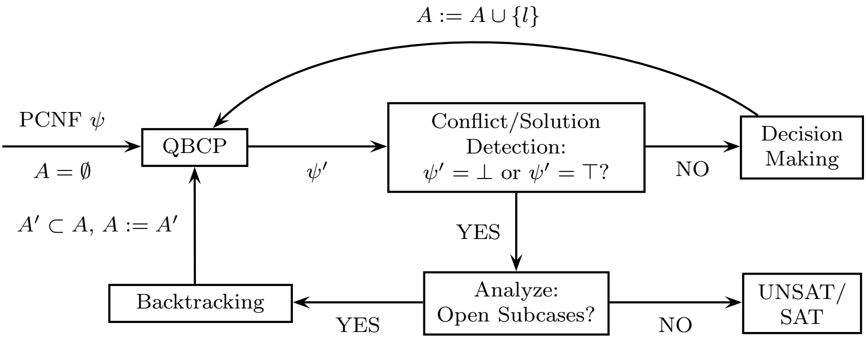
Towards Conflict Driven Clause Learning (QCDCL)

Clause Learning:

- Adding new clauses C_L to given PCNF by analyzing a conflict.
- Learned clause prevents subassignments.
- Related to CDCL for SAT solving.
- CDCL: pioneered by solvers like GRASP or Chaff [SS99, MMZ$^+$01].
- Correctness requirement: $\hat{Q}.\phi \equiv_{sat} \hat{Q}.(\phi \wedge C_L)$

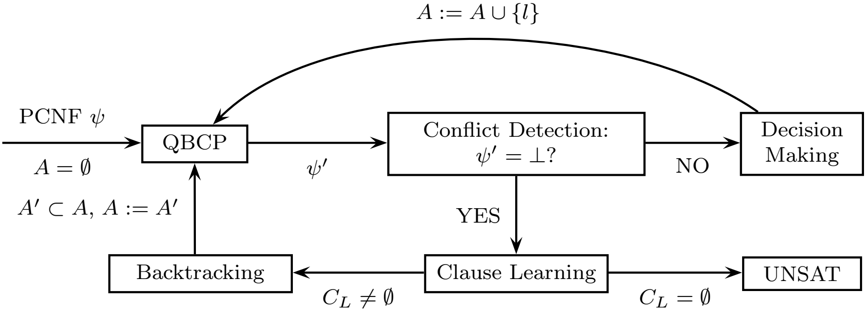
$\Rightarrow$ deriving learned clauses by the *Q-resolution calculus (QRES)*.

Abstract Workflow: Adding Clause Learning



- Chronological backtracking and backjumping: suboptimal analysis of open subcases.
- Clause learning in QCDCL: stronger than backtracking/-jumping.

Abstract Workflow: Adding Clause Learning



- *For now, we focus on unsatisfiable PCNFs.*
- Learned clause C_L derived by *QRES* based on implication graphs.
- Formal foundation of clause learning: proof system *QRES*.
- Termination and backtracking controlled by properties of C_L.

Q-Resolution (1/2)

Definition (Q-Resolution Calculus QRES, c.f. [BKF95])

Let $\psi = \hat{Q}.\phi$ be a PCNF and C, C_1, C_2 clauses.

$$\frac{}{C} \quad \text{for all } x \in \hat{Q}\text{: } \{x, \bar{x}\} \not\subseteq C \text{ and } C \in \phi \qquad (init)$$

$$\frac{C \cup \{l\}}{C} \quad \begin{array}{l} \text{for all } x \in \hat{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C \cup \{l\}),\ q(l) = \forall, \text{ and} \\ l' < l \text{ for all } l' \in C \text{ with } q(l') = \exists \end{array} \qquad (red)$$

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \begin{array}{l} \text{for all } x \in \hat{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2), \\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and } q(p) = \exists \end{array} \qquad (res)$$

- Axiom *init*, universal reduction *red*, resolution *res*.
- PCNF ψ is unsatisfiable iff empty clause $\emptyset$ can be derived by QRES.

Q-Resolution (2/2)

Example (continued)

$\psi = \exists x_1, x_2, x_3, x_4 \forall y_5 \exists x_6. (\neg x_3 \vee x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge (\neg x_1 \vee y_5 \vee \neg x_6) \wedge \phi.$

Applying QRES:

- Axiom *init* selects initial clauses.
- Resolution on clauses by *res* using existential pivots.
- Reduction of trailing universal literals from clauses by *red*.

$$\begin{array}{ccc} (\neg x_4 \vee x_6) & & (\neg x_1 \vee y_5 \vee \neg x_6) \\ & (\neg x_1 \vee \neg x_4 \vee y_5) & \\ & (\neg x_1 \vee \neg x_4) & \end{array}$$

- For clauses C_L derived from PCNF $\hat{Q}.\phi$ by *QRES*: $\hat{Q}.\phi \equiv_{sat} \hat{Q}.(\phi \wedge C_L)$.
- *QRES* for clause learning: driven by conflicts and implication graphs.
- Stronger, more flexible variants of *QRES* exist.

QCDCL: Basic Idea (1/3)

Example (Clause Learning)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2 . (\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

QCDCL: Basic Idea (1/3)

Example (Clause Learning)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2.(\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

- Make decision $A = \{x_1\}$:
  $\psi[\{x_1\}] = \exists x_3, x_4 \forall y_5 \exists x_2.(x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

QCDCL: Basic Idea (1/3)

Example (Clause Learning)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2. (\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

- Make decision $A = \{x_1\}$:
  $\psi[\{x_1\}] = \exists x_3, x_4 \forall y_5 \exists x_2. (x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$
- By UL: $\psi[\{x_1, x_2\}] = \exists x_3, x_4 \forall y_5. (x_3 \vee y_5) \wedge (x_4 \vee \bar{y}_5) \wedge (\bar{x}_3 \vee \bar{x}_4)$.

QCDCL: Basic Idea (1/3)

Example (Clause Learning)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2.(\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

- Make decision $A = \{x_1\}$:
  $\psi[\{x_1\}] = \exists x_3, x_4 \forall y_5 \exists x_2.(x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$
- By UL: $\psi[\{x_1, x_2\}] = \exists x_3, x_4 \forall y_5.(x_3 \vee y_5) \wedge (x_4 \vee \bar{y}_5) \wedge (\bar{x}_3 \vee \bar{x}_4)$.
- By UR: $\psi[\{x_1, x_2\}] = \exists x_3, x_4.(x_3) \wedge (x_4) \wedge (\bar{x}_3 \vee \bar{x}_4)$

QCDCL: Basic Idea (1/3)

Example (Clause Learning)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2. (\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

- Make decision $A = \{x_1\}$:
  $\psi[\{x_1\}] = \exists x_3, x_4 \forall y_5 \exists x_2. (x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$
- By UL: $\psi[\{x_1, x_2\}] = \exists x_3, x_4 \forall y_5. (x_3 \vee y_5) \wedge (x_4 \vee \bar{y}_5) \wedge (\bar{x}_3 \vee \bar{x}_4)$.
- By UR: $\psi[\{x_1, x_2\}] = \exists x_3, x_4. (x_3) \wedge (x_4) \wedge (\bar{x}_3 \vee \bar{x}_4)$
- By UL: $\psi[\{x_1, x_2, x_3, x_4\}] = \bot$, clause $(\bar{x}_3 \vee \bar{x}_4)$ conflicting.

QCDCL: Basic Idea (1/3)

Example (Clause Learning)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2 . (\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

- Make decision $A = \{x_1\}$:
  $\psi[\{x_1\}] = \exists x_3, x_4 \forall y_5 \exists x_2 . (x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$
- By UL: $\psi[\{x_1, x_2\}] = \exists x_3, x_4 \forall y_5 . (x_3 \vee y_5) \wedge (x_4 \vee \bar{y}_5) \wedge (\bar{x}_3 \vee \bar{x}_4)$.
- By UR: $\psi[\{x_1, x_2\}] = \exists x_3, x_4 . (x_3) \wedge (x_4) \wedge (\bar{x}_3 \vee \bar{x}_4)$
- By UL: $\psi[\{x_1, x_2, x_3, x_4\}] = \bot$, clause $(\bar{x}_3 \vee \bar{x}_4)$ conflicting.

Implication graph G:

$$L_i : \begin{tikzcd} x_1 \arrow[r] & x_2 \arrow[r] \arrow[dr] & x_3 \arrow[r] & \emptyset \\ & & x_4 \arrow[ur] & \end{tikzcd}$$

Antecedent clauses:

$$\begin{aligned} ante(x_2) &: \quad (\bar{x}_1 \vee x_2) \\ ante(x_3) &: \quad (x_3 \vee y_5 \vee \bar{x}_2) \\ ante(x_4) &: \quad (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \\ ante(\emptyset) &: \quad (\bar{x}_3 \vee \bar{x}_4) \end{aligned}$$

QCDCL: Basic Idea (2/3)

Example (Clause Learning, continued)

Prefix: $\exists x_1, x_3, x_4 \forall y_5 \exists x_2$
Assignment $A = \{x_1, x_2, x_3, x_4\}$
Implication graph G:

$$L_i: \begin{tikzcd} x_1 \arrow[r] & x_2 \arrow[r] \arrow[dr] & x_3 \arrow[r] & \emptyset \\ & & x_4 \arrow[ur] & \end{tikzcd}$$

Antecedent clauses:

$ante(x_2)$: $(\bar{x}_1 \lor x_2)$
$ante(x_3)$: $(x_3 \lor y_5 \lor \bar{x}_2)$
$ante(x_4)$: $(x_4 \lor \bar{y}_5 \lor \bar{x}_2)$
$ante(\emptyset)$: $(\bar{x}_3 \lor \bar{x}_4)$

- Start at $\emptyset$, select pivots in reverse assignment ordering: resolve antecedents of x_4, x_3.
- Q-resolution [BKF95] disallows tautologies like $(\bar{y}_5 \lor y_5 \lor \bar{x}_2)$!
- Pivot selection more complex than in CDCL for SAT solving.

$$\dfrac{\dfrac{(\bar{x}_3 \lor \bar{x}_4) \qquad (x_4 \lor \bar{y}_5 \lor \bar{x}_2)}{(\bar{x}_3 \lor \bar{y}_5 \lor \bar{x}_2)} \qquad (x_3 \lor y_5 \lor \bar{x}_2)}{\cancel{(\bar{y}_5 \lor y_5 \lor \bar{x}_2)}}$$

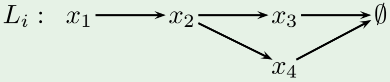
QCDCL: Basic Idea (3/3)—Avoiding Tautologies

Example (Clause Learning, continued)

Prefix: $\exists x_1, x_3, x_4 \forall y_5 \exists x_2$

Assignment $A = \{x_1, x_2, x_3, x_4\}$

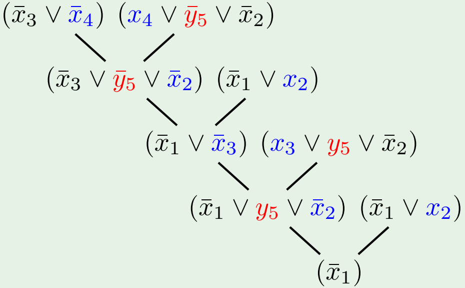
Implication graph G:



Antecedent clauses:

$$ante(x_2) : \quad (\bar{x}_1 \vee x_2)$$
$$ante(x_3) : \quad (x_3 \vee y_5 \vee \bar{x}_2)$$
$$ante(x_4) : \quad (x_4 \vee \bar{y}_5 \vee \bar{x}_2)$$
$$ante(\emptyset) : \quad (\bar{x}_3 \vee \bar{x}_4)$$

- To avoid tautologies, resolve on UR-blocking existentials.
- Select pivots: x_4, x_2, x_3, x_2.
- Potentially resolve on variables more than once to derive learned clause $C_L := (\neg x_1)$.



QCDCL by Traditional Q-Resolution [BKF95]:

- Avoid tautologies by appropriate pivot selection [GNT06].
- Problem: derivation of a learned clause may be exponential [VG12].
- Annotate nodes in conflict graph with intermediate resolvents, resulting in *tree-like* (instead of linear) Q-resolution derivations of learned clauses [LEG13].

QCDCL: Pivot Selection—Long Distance Q-Resolution

QCDCL by Long Distance (LD) Q-Resolution [ZM02a, BJ12]:

- Key property: allow tautological resolvents of a certain kind.
- First implementation in QCDCL solver quaffle: `https://www.princeton.edu/~chaff/quaffle.html`.
- LDQ-resolution calculus is exponentially stronger than QRES.
- Practice: always select pivots in strict reverse assignment ordering.
  - Every resolution step is a valid LDQ-resolution step [ZM02a, ELW13].

LDQ-Resolution Definition

Example (continued)

$\psi = \exists x_1, x_3, x_4 \forall y_5 \exists x_2.(\bar{x}_1 \vee x_2) \wedge (x_3 \vee y_5 \vee \bar{x}_2) \wedge (x_4 \vee \bar{y}_5 \vee \bar{x}_2) \wedge (\bar{x}_3 \vee \bar{x}_4)$

$$(\bar{x}_3 \vee \bar{x}_4) \quad (x_4 \vee \bar{y}_5 \vee \bar{x}_2)$$

$$(\bar{x}_3 \vee \bar{y}_5 \vee \bar{x}_2) \quad (x_3 \vee y_5 \vee \bar{x}_2)$$

$$(\bar{y}_5 \vee y_5 \vee \bar{x}_2)$$

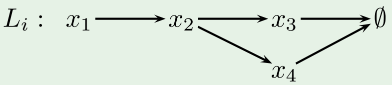
Long-Distance Q-Resolution: [ZM02a, BJ12]

- Generation of tautologies must respect prefix ordering of pivots.
- Tautological resolvent C with $\{x, \bar{x}\} \subseteq C$:
  - $q(x) = \forall$
  - Existential pivot p: $p < x$ in prefix ordering.

LDQ-Resolution Example

Example (Clause Learning, continued)

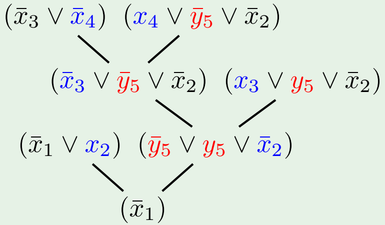
Prefix: $\exists x_1, x_3, x_4 \forall y_5 \exists x_2$
Assignment $A = \{x_1, x_2, x_3, x_4\}$
Implication graph G:



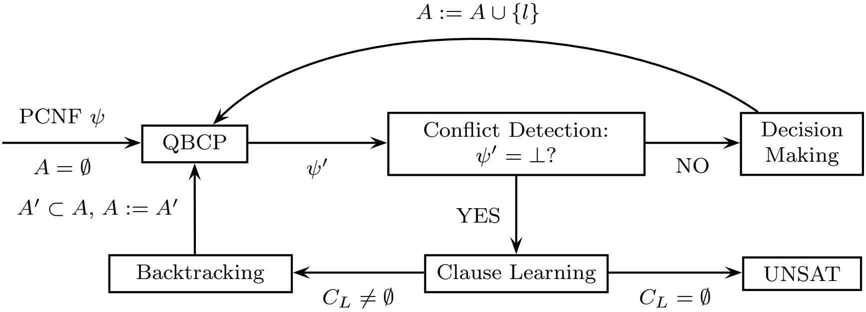
Antecedent clauses:

$ante(x_2)$: $(\bar{x}_1 \vee x_2)$
$ante(x_3)$: $(x_3 \vee y_5 \vee \bar{x}_2)$
$ante(x_4)$: $(x_4 \vee \bar{y}_5 \vee \bar{x}_2)$
$ante(\emptyset)$: $(\bar{x}_3 \vee \bar{x}_4)$

- Start at $\emptyset$, *always* select pivots in reverse assignment ordering: Resolve antecedents of x_4, x_3, x_2.
- Pivots obey order restriction of LDQ-resolution: $x_3 < y_5$
- To derive $C_L := (\neg x_1)$, resolve at most once on a variable.

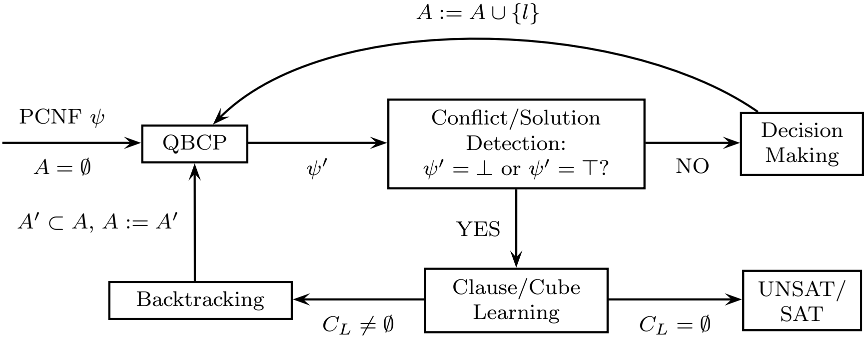


Abstract Workflow: Adding Cube Learning



- So far, we have focused on unsatisfiable QBFs.
- Clause learning: generation of *QRES* proofs of *unsatisfiability*.

Abstract Workflow: Adding Cube Learning



- Cube learning: solving satisfiable QBFs, similar to clause learning.
- Cube: *conjunction* of literals.
- QCDCL: clause and cube learning, driven by implication graphs.
- Derivation of cubes from a given PCNF ψ: variant of QRES.
- Termination and backtracking controlled by learned clause/cube C_L.

Cube Learning: Variant of QRES (1/2)

Definition (Model Generation, cf. [GNT06, Let02, ZM02b])

Let $\psi = \hat{Q}.\phi$ be a PCNF.

$$\frac{}{\mathrm{C}} \quad \begin{array}{l} C = (\bigwedge_{l \in A}) \text{ is a cube where } \{x, \bar{x}\} \not\subseteq C \text{ and } A \text{ is an as-} \\ \text{signment with } \psi[A] = \top \text{, i.e. every clause of } \psi \text{ satisfied.} \end{array} \quad (\textit{cu-init})$$

Cube Learning as a Proof System:

- Cube C by model generation: $v \in C$ ($\bar{v} \in C$) if v assigned to $\top$ ($\bot$).
- C (also called *cover set*): implicant of CNF ϕ, i.e. $C \Rightarrow \phi$.
- Model generation: a new axiom added to QRES.
- QRES for cubes: Q-resolution and *existential reduction* on cubes.
- PCNF ψ is satisfiable iff the empty cube can be derived from ψ.

Cube Learning: Variant of QRES (2/2)

Definition (Model Generation, cf. [GNT06, Let02, ZM02b])

Let $\psi = \hat{Q}.\phi$ be a PCNF.

$$\frac{}{C} \quad \begin{array}{l} C = (\bigwedge_{l \in A}) \text{ is a cube where } \{x, \bar{x}\} \not\subseteq C \text{ and } A \text{ is an as-} \\ \text{signment with } \psi[A] = \top \text{, i.e. every clause of } \psi \text{ satisfied.} \end{array} \quad (\textit{cu-init})$$

Example

$\psi = \exists x \forall u \exists y.(\bar{x} \vee u \vee \bar{y}) \wedge (\bar{x} \vee \bar{u} \vee y) \wedge (x \vee u \vee y) \wedge (x \vee \bar{u} \vee \bar{y})$

$(\bar{x} \wedge u \wedge \bar{y}) \quad (\bar{x} \wedge \bar{u} \wedge y)$

- By model generation: derive cubes $(\bar{x} \wedge u \wedge \bar{y})$ and $(\bar{x} \wedge \bar{u} \wedge y)$.

Cube Learning: Variant of QRES (2/2)

Definition (Existential Reduction, cf. [GNT06, Let02, ZM02b])

Let C be a cube.

$$\frac{C \cup \{l\}}{C} \quad \text{for all } x \in \hat{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C \cup \{l\}),\ q(l) = \exists, \text{ and } l' < l \text{ for all } l' \in C \text{ with } q(l') = \forall \qquad (\textit{cu-red})$$

Example

$\psi = \exists x \forall u \exists y.(\bar{x} \vee u \vee \bar{y}) \wedge (\bar{x} \vee \bar{u} \vee y) \wedge (x \vee u \vee y) \wedge (x \vee \bar{u} \vee \bar{y})$

$$\begin{array}{cc} (\bar{x} \wedge u \wedge \bar{y}) & (\bar{x} \wedge \bar{u} \wedge y) \\ | & | \\ (\bar{x} \wedge u) & (\bar{x} \wedge \bar{u}) \end{array}$$

- By model generation: derive cubes $(\bar{x} \wedge u \wedge \bar{y})$ and $(\bar{x} \wedge \bar{u} \wedge y)$.
- By existential reduction: reduce trailing $\bar{y}$ from $(\bar{x} \wedge u \wedge \bar{y})$, y from $(\bar{x} \wedge \bar{u} \wedge y)$.

Cube Learning: Variant of QRES (2/2)

Definition (Cube Resolution, cf. [GNT06, Let02, ZM02b])

Let C_1, C_2 be cubes.

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \begin{array}{l}\text{for all } x \in \hat{Q}\text{: } \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2), \\ \bar{p} \notin C_1,\ p \notin C_2, \text{ and } q(p) = \forall\end{array} \qquad (\textit{cu-res})$$

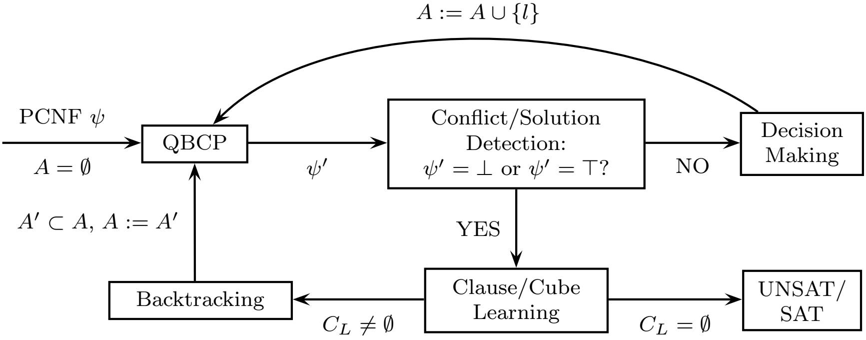
Example

$\psi = \exists x \forall u \exists y. (\bar{x} \vee u \vee \bar{y}) \wedge (\bar{x} \vee \bar{u} \vee y) \wedge (x \vee u \vee y) \wedge (x \vee \bar{u} \vee \bar{y})$

$$\begin{tikzcd}[column sep=small]
(\bar{x} \wedge u \wedge \bar{y}) \arrow[d, dash] & & (\bar{x} \wedge \bar{u} \wedge y) \arrow[d, dash] \\
(\bar{x} \wedge u) \arrow[dr, dash] & & (\bar{x} \wedge \bar{u}) \arrow[dl, dash] \\
& (\bar{x}) \arrow[d, dash] & \\
& \emptyset &
\end{tikzcd}$$

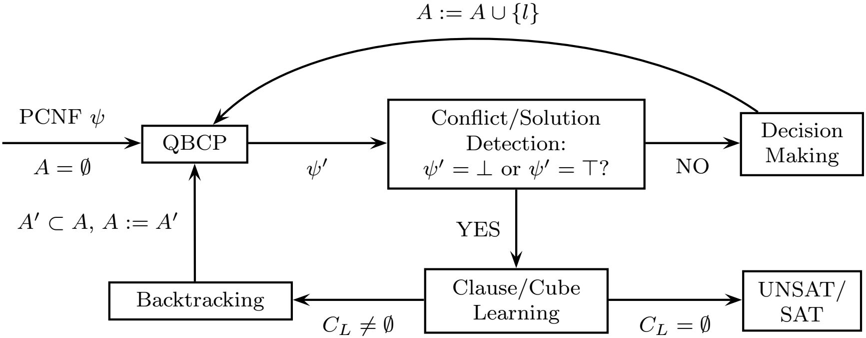
- By model generation: derive cubes $(\bar{x} \wedge u \wedge \bar{y})$ and $(\bar{x} \wedge \bar{u} \wedge y)$.
- By existential reduction: reduce trailing $\bar{y}$ from $(\bar{x} \wedge u \wedge \bar{y})$, y from $(\bar{x} \wedge \bar{u} \wedge y)$.
- Resolve $(\bar{x} \wedge \bar{u})$ and $(\bar{x} \wedge u)$ on universal u.
- Reduce $(\bar{x})$ to derive $\emptyset$.

Abstract Workflow: Final QCDCL View



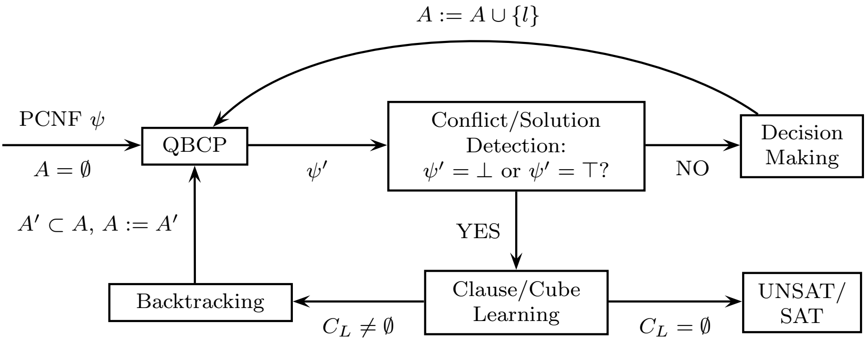
- Generate assignments A by decision making and (unit) propagation.
- Simplify ψ under A to obtain ψ'.
- Conflict: $\psi' = \bot$: ψ' contains a falsified clause.
- Solution: $\psi' = \top$: all clauses in ψ' satisfied (i.e., empty CNF).

Abstract Workflow: Final QCDCL View



- Generate learned clause (cube) C_L by Q-resolution, added to ψ.
- Empty clause (cube) $C_L = \emptyset$: formula proved UNSAT (SAT).
- Q-resolution proofs of (un)satisfiability by QRES.

Abstract Workflow: Final QCDCL View

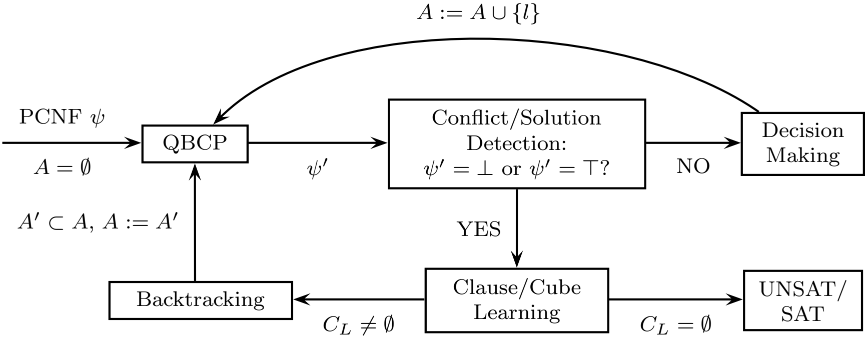


- Conflict detected: select clauses for Q-resolution.

Definition (Clause Axiom of QRES)

$$\overline{C} \qquad \text{Given a PCNF } \psi = \hat{Q}.\phi,\ C \in \phi \text{ is a clause.}$$

Abstract Workflow: Final QCDCL View



- Solution detected: select cubes for Q-resolution.

Definition (Cube Axiom of QRES)

$$\overline{\quad C \quad}$$

Given a PCNF $\psi = \hat{Q}.\phi$ and an assignment A with $\psi[A] = \top$, $C = (\bigwedge_{l \in A})$ is a cube.

QCDCL in Practice

Clause and Cube Learning:

- PCNF $\psi := \hat{Q}.\,\phi$ with quantifier prefix $\hat{Q}$ and CNF ϕ.
- CNFs of learned clauses ϕ_{CL} and DNF of cubes ϕ_{CU}.
- Properties: $\hat{Q}.\,\phi \equiv_{sat} \hat{Q}.\,(\phi \wedge \phi_{CL})$ and $\hat{Q}.\,\phi \equiv_{sat} \hat{Q}.\,(\phi \vee \phi_{CU})$.

Interplay Between Clauses and Cubes:

- QBCP applied to ϕ, ϕ_{CL}, and ϕ_{CU}.
- Assignments by unit clauses can trigger unit cubes and vice versa.
- Antecedent clauses and *antecedent cubes* are recorded as usual.

Applying the Q-Resolution Calculus:

- Similar to clause learning, cube rules are driven by implication graph.
- In a derivation, applications of clause and cube rules are never mixed.

Asserting Learned Clauses and Cubes

Search Space Exploration in QCDCL:

- No explicit flipping of variables in decision making.
- Fundamental difference to traditional backtracking algorithms.
- Backjumping: *asserting* clauses (cubes) become unit by QBCP.
- Asserting clauses (cubes) cause flipping of variables.

Asserting Learned Clauses and Cubes

Asserting Criteria Applied During Learning:

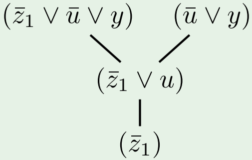
- Start at empty clause $\emptyset$ or the cube derived by model-gen. at level k.
- Let C be the current clause/cube derived by $QRES$.
- C asserting if C becomes unit in QBCP at some level $j < k$.
- If C asserting, then stop derivation, learn C, and backjump to level j.
- Otherwise, continue applying $QRES$ rules.

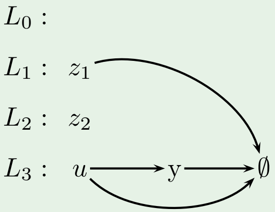
Clause and Cube Learning Example (1/3)

Example

$\exists z_1, z_2 \forall u \exists y. (u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z_1 \vee u \vee \bar{y}) \wedge (z_2 \vee \bar{u} \vee y) \wedge (\bar{z}_1 \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}_2 \vee u \vee y)$

- Level 0 is empty, no unit clauses present.
- Levels 1, 2: decisions z_1 and z_2.
- Level 3: decision u, implies y by QBCP, $ante(y) := (\bar{u} \vee y)$.
- Conflict: $ante(\emptyset) := (\bar{z}_1 \vee \bar{u} \vee \bar{y})$.





- Learn clause $C_{L,1} := (\bar{z}_1)$, asserting at L_0.

Clause and Cube Learning Example (2/3)

Example

$\exists z_1, z_2 \forall u \exists y. (u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z_1 \vee u \vee \bar{y}) \wedge (z_2 \vee \bar{u} \vee y) \wedge (\bar{z}_1 \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}_2 \vee u \vee y)$

- Backjump to L_0, $C_{L,1} = (\bar{z}_1)$ unit.
- Level 1: decision $\bar{z}_2$.
- Level 2: decision $\bar{u}$, implies $\bar{y}$ by QBCP.
- All clauses satisfied.

$$(\bar{z}_1 \wedge \bar{z}_2 \wedge \bar{u} \wedge \bar{y})$$
$$|$$
$$(\bar{z}_1 \wedge \bar{z}_2 \wedge \bar{u})$$

$L_0 : \ \bar{z}_1$

$L_1 : \ \bar{z}_2$

$L_2 : \ \bar{u} \longrightarrow \bar{y}$

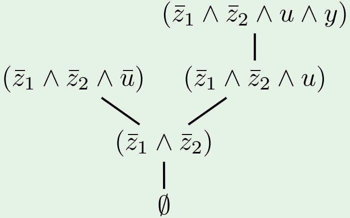
- Cube learning: model generation, existential reduction.
- Learn cube $C_{L,2} := (\bar{z}_1 \wedge \bar{z}_2 \wedge \bar{u})$, asserting at L_1.

Clause and Cube Learning Example (3/3)

Example

$\exists z_1, z_2 \forall u \exists y. (u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z_1 \vee u \vee \bar{y}) \wedge (z_2 \vee \bar{u} \vee y) \wedge (\bar{z}_1 \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}_2 \vee u \vee y)$

- Backjump to L_1, $C_{L,2} := (\bar{z}_1 \wedge \bar{z}_2 \wedge \bar{u})$ unit.
- Level 1: $ante(u) := C_{L,2}$, implies (y).
- All clauses satisfied.



$L_0: \ \bar{z}_1$

$L_1: \ \bar{z}_2 \longrightarrow u \longrightarrow y$

- Cube learning: derive empty cube, proving satisfiability.

Clause and Cube Learning: Remarks

QCDCL Properties (by Construction):

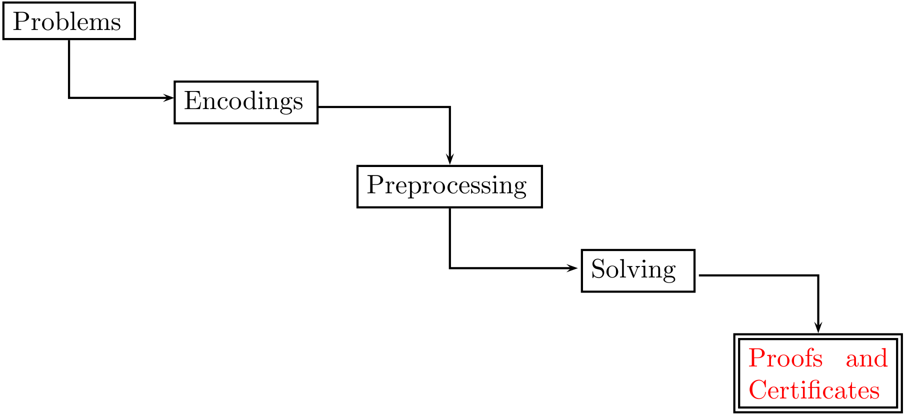
- Implication graph, i.e., assignment order, guides QRES rules.
- Graph may contain assignments from unit clauses *and* cubes.
- At conflict: only clauses are derived, but never cubes.
- At solution: only cubes are derived, but never clauses.
- Empty clause (cube) potentially derived at any level (termination).

Clause and Cube Learning: Remarks

Cube Learning Worst Case: [RBM97, Let02]

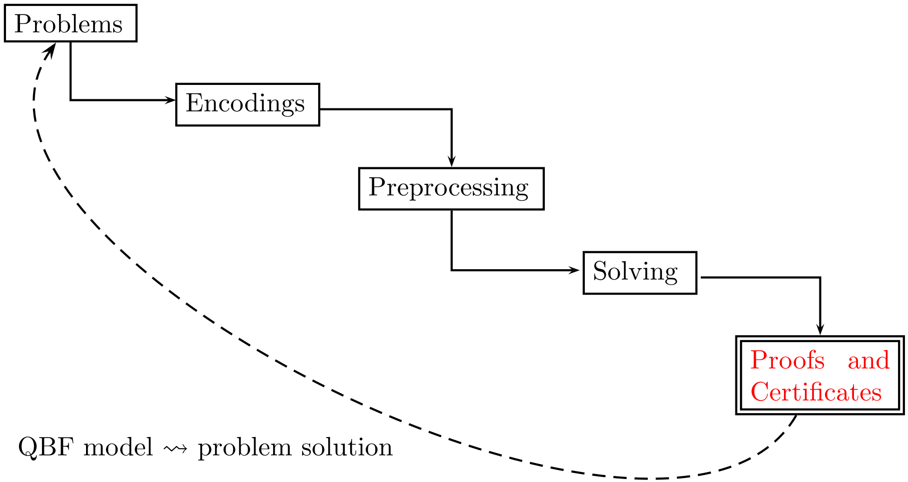
- $\psi = \forall u_1 \exists x_1 \ldots \forall u_n \exists x_n. \bigwedge_{i=1}^{n} [(u_i \vee \bar{x}_i) \wedge (\bar{u}_i \vee x_i)]$
- Easy satisfiable formula: as the value of x_i, always choose $f(u_i) := u_i$.
- However: all cube resolution proofs are exponential (worst case DNF).

Typical QBF Workflow: Generating Proofs and Certificates



Solver Correctness: How to verify the result?

Typical QBF Workflow: Generating Proofs and Certificates



(Counter-)Models: How to obtain solution to original problem?

Models of Satisfiable QBFs

Definition (Skolem Function)

Let ψ be a PCNF, y a existential variable.

- Let $D^{\psi}(v) := \{w \in \psi \mid q(v) \neq q(w) \text{ and } w < v\}$, $q(v) \in \{\forall, \exists\}$.
- Skolem function $f_y(x_1, \ldots, x_k)$ of y: $D^{\psi}(y) = \{x_1, \ldots, x_k\}$.
- Function f_y depends on all universal variables smaller than y.

Definition (Skolem Function Model)

A PCNF ψ with existential variables $y_1, \ldots, y_m$ is satisfiable iff $\psi[y_1/f_{y_1}(D^{\psi}(y_1)), \ldots, y_m/f_{y_m}(D^{\psi}(y_m))]$ is satisfiable.

Models of Satisfiable QBFs

Example (Skolem Function Model)

$\psi = \exists x \forall u \exists y.(\bar{x} \vee u \vee \bar{y}) \wedge (\bar{x} \vee \bar{u} \vee y) \wedge (x \vee u \vee y) \wedge (x \vee \bar{u} \vee \bar{y})$

- Skolem function $f_x = \bot$ of x with $D^{\psi}(x) = \emptyset$.
- Skolem function $f_y(u) = \bar{u}$ of y with $D^{\psi}(y) = \{u\}$.
- $\psi[x/f_x, y/f_y(u)] = \forall u.(\bot \vee u \vee \bar{u}) \wedge (\bot \vee \bar{u} \vee u)$
- Satisfiable: $\psi[x/f_x, y/f_y(u)] = \top$

Checking Skolem Function Models:

- Observe: $\psi[x/f_x, y/f_y(u)]$ contains only $\forall$-variables.
- Use a SAT solver to check whether $\neg(\psi[x/f_x, y/f_y(u)])$ is unsatisfiable.

Countermodels of Unsatisfiable QBFs

Definition (Herbrand Function)

Let ψ be a PCNF, x a universal variable.

- Let $D^{\psi}(v) := \{w \in \psi \mid q(v) \neq q(w) \text{ and } w < v\}$, $q(v) \in \{\forall, \exists\}$.
- Herbrand function $f_x(y_1, \ldots, y_k)$ of x: $D^{\psi}(x) = \{y_1, \ldots, y_k\}$.
- Function f_x depends on all existential variables smaller than x.

Definition (Herbrand Function Countermodel)

A PCNF ψ with universal variables $x_1, \ldots, x_m$ is unsatisfiable iff $\psi[x_1/f_{x_1}(D^{\psi}(x_1)), \ldots, x_m/f_{x_m}(D^{\psi}(x_m))]$ is unsatisfiable.

Countermodels of Unsatisfiable QBFs

Example (Herbrand Function Countermodel)

$\psi = \exists x \forall u \exists y . (x \vee u \vee y) \wedge (x \vee u \vee \bar{y}) \wedge (\bar{x} \vee \bar{u} \vee y) \wedge (\bar{x} \vee \bar{u} \vee \bar{y})$

- Herbrand function $f_u(x) = (x)$ of u with $D^{\psi}(u) = \{x\}$.
- $\psi[u/f_u(x)] = \exists x, y . (x \vee x \vee y) \wedge (x \vee x \vee \bar{y}) \wedge (\bar{x} \vee \bar{x} \vee y) \wedge (\bar{x} \vee \bar{x} \vee \bar{y})$
- Unsatisfiable: $\psi[u/f_u(x)] = \exists x, y . (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y})$

Checking Herbrand Function Countermodels:

- Observe: $\psi[x/f_x, y/f_y(u)]$ contains only $\exists$-variables.
- Use a SAT solver to check whether $\psi[x/f_x, y/f_y(u)]$ is unsatisfiable.

Generating (Counter)Models from Proofs

Q-Resolution Proofs:

- QCDCL solvers produce derivations P of the empty clause/cube.
- Proof P can be filtered out of derivations of all learned clauses/cubes.

Extracting Skolem/Herbrand Functions from Proofs:

- By inspection of P, run time linear in $|P|$ ($|P|$ can be exponential).
- Extraction from long-distance Q-resolution proofs [BJJW15].
- Approaches to compute winning strategies from P [GGB11, ELW13].

Generating (Counter)Models from Proofs

Definition (Extracting Herbrand functions [BJ11, BJ12])

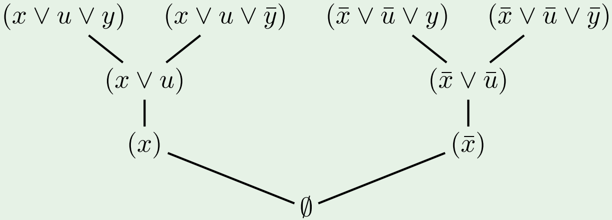
Let P be a proof (Q-resolution DAG) of the empty clause $\emptyset$.

- Visit clauses in P in topological ordering.
- Inspect universal reduction steps $C' = UR(C)$.
- Update Herbrand functions of variables u reduced from C by C'.

Generating Countermodels from Proofs: Example

Example (Extracting Herbrand Functions [BJ11, BJ12])

$\psi = \exists x \forall u \exists y.(x \vee u \vee y) \wedge (x \vee u \vee \bar{y}) \wedge (\bar{x} \vee \bar{u} \vee y) \wedge (\bar{x} \vee \bar{u} \vee \bar{y})$



- Literal u reduced from $(x \vee u)$, update: $f_u(x) := (x)$.
- Literal $\bar{u}$ reduced from $(\bar{x} \vee \bar{u})$, update: $f_u(x) := f_u(x) \wedge \neg(\bar{x}) = (x)$.
- Unsatisfiable: $\psi[u/f_u(x)] = \exists x, y.(x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y})$

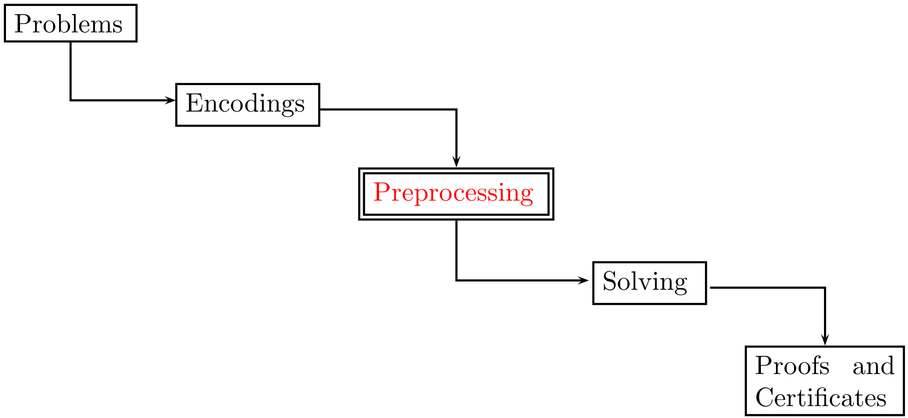
(Counter)Models: Special Cases

Example

Let $\psi := \exists X \forall Y.\ \phi$ and $\psi' := \forall Y \exists X.\ \phi$ be one-alternation QBFs.

- If ψ satisfiable: all Skolem functions are constant.
- If ψ' unsatisfiable: all Herbrand functions are constant.
- No need to produce derivations of the empty clause/cube.
- QBF solvers can directly output values of Skolem/Herbrand functions.
- Useful for modelling and solving problems in Σ_2^P and Π_2^P.
- QDIMACS output format specification.

Typical QBF Workflow: Preprocessing



Blocked Clause Elimination (QBCE) (1/2)

Definition (Blocking Literal, Blocked Clause [Kul99, BLS11, HJL+15])

Let $\psi = \hat{Q}.\phi$ be a PCNF and $C \in \phi$ a clause.

- *blocking literal l*: $l \in C$ with $q(l) = \exists$ such that for all $C' \in \phi$ with $\bar{l} \in C'$, there exists x with $x \leq l$ such that $\{x, \bar{x}\} \subseteq (C \cup (C' \setminus \{\bar{l}\}))$.
- A clause C is *blocked* if it contains a blocking literal.
- Removing blocked clauses preserves satisfiability.

$$\underbrace{(\ldots \vee \bar{x}_1 \vee \ldots \vee \bar{l} \ldots)}_{C'_1} \quad \ldots \quad \underbrace{(\ldots \vee \bar{x}_i \vee \ldots \vee \bar{l} \ldots)}_{C'_i} \quad \ldots \quad \underbrace{(\ldots \vee \bar{x}_n \vee \ldots \vee \bar{l} \ldots)}_{C'_n}$$

$$\{x_1, \bar{x}_1\} \subseteq C \cup (C'_1 \setminus \{\bar{l}\}) \qquad x_1 \leq l$$

$$C = (x_1 \vee \ldots \vee x_i \vee \ldots \vee x_n \vee \ldots \vee l \vee \ldots)$$

Blocked Clause Elimination (QBCE) (1/2)

Definition (Blocking Literal, Blocked Clause [Kul99, BLS11, HJL+15])

Let $\psi = \hat{Q}.\phi$ be a PCNF and $C \in \phi$ a clause.

- *blocking literal* l: $l \in C$ with $q(l) = \exists$ such that for all $C' \in \phi$ with $\bar{l} \in C'$, there exists x with $x \leq l$ such that $\{x, \bar{x}\} \subseteq (C \cup (C' \setminus \{\bar{l}\}))$.
- A clause C is *blocked* if it contains a blocking literal.
- Removing blocked clauses preserves satisfiability.

$$\underbrace{(\ldots \vee \bar{x}_1 \vee \ldots \vee \bar{l} \ldots)}_{C'_1} \quad \ldots \quad \underbrace{(\ldots \vee \bar{x}_i \vee \ldots \vee \bar{l} \ldots)}_{C'_i} \quad \ldots \quad \underbrace{(\ldots \vee \bar{x}_n \vee \ldots \vee \bar{l} \ldots)}_{C'_n}$$

$$\{x_i, \bar{x}_i\} \subseteq C \cup (C'_i \setminus \{\bar{l}\}) \qquad x_i \leq l$$

$$C = (x_1 \vee \ldots \vee x_i \vee \ldots \vee x_n \vee \ldots \vee l \vee \ldots)$$

Blocked Clause Elimination (QBCE) (1/2)

Definition (Blocking Literal, Blocked Clause [Kul99, BLS11, HJL+15])

Let $\psi = \hat{Q}.\phi$ be a PCNF and $C \in \phi$ a clause.

- *blocking literal l*: $l \in C$ with $q(l) = \exists$ such that for all $C' \in \phi$ with $\bar{l} \in C'$, there exists x with $x \leq l$ such that $\{x, \bar{x}\} \subseteq (C \cup (C' \setminus \{\bar{l}\}))$.
- A clause C is *blocked* if it contains a blocking literal.
- Removing blocked clauses preserves satisfiability.

$$\underbrace{(\ldots \vee \bar{x}_1 \vee \ldots \vee \bar{l} \ldots)}_{C'_1} \quad \ldots \quad \underbrace{(\ldots \vee \bar{x}_i \vee \ldots \vee \bar{l} \ldots)}_{C'_i} \quad \ldots \quad \underbrace{(\ldots \vee \bar{x}_n \vee \ldots \vee \bar{l} \ldots)}_{C'_n}$$

$$\{x_n, \bar{x}_n\} \subseteq C \cup (C'_n \setminus \{\bar{l}\}) \qquad x_n \leq l$$

$$C = (x_1 \vee \ldots \vee x_i \vee \ldots \vee x_n \vee \ldots \vee l \vee \ldots)$$

Blocked Clause Elimination (QBCE) (2/2)

Important Facts:

- Blocking literal l: existentially quantified.

Example

$\psi := \exists y \forall x \exists z.(y \vee \bar{x} \vee z) \wedge (\bar{y} \vee x \vee z) \wedge (y) \wedge (\bar{z})$

- ψ is unsatisfiable.
- Universal x *cannot* be a blocking literal.
- Otherwise, first two clauses would *erroneously* be blocked.
- Unsoundness: ψ becomes satisfiable.

Blocked Clause Elimination (QBCE) (2/2)

Important Facts:

- Blocking literal l: existentially quantified.
- Tautology-producing variable x: $\leq l$ in prefix ordering.

Example

$\psi := \exists y \forall x.(y \vee \bar{x}) \wedge (\bar{y} \vee x)$.

- ψ is unsatisfiable.
- Prefix ordering matters.
- Literals of y are not blocking literals since $y \leq x$.
- Erroneous removal of any clause makes formula satisfiable.

Blocked Clause Elimination (QBCE) (2/2)

Important Facts:

- Blocking literal l: existentially quantified.
- Tautology-producing variable x: $\leq l$ in prefix ordering.
- Check all potential resolution candidates on l.
  - Pure $\exists$-literals: vacuously blocking.

Example

$\psi = \exists y \forall x \exists z. \; (\bar{y} \vee z) \wedge (\bar{y} \vee \bar{z}) \wedge (\bar{x} \vee z) \wedge (x \vee \bar{z})$.

- $\exists$-literal $\bar{y}$ is pure.
- No resolution candidates on clauses containing y.
- Condition of blocking literal is vacuously satisfied.
- Clauses containing $\bar{y}$ can be removed.

Expansion (1/4)

$$\psi_0 \rightsquigarrow \psi_1 \rightsquigarrow \psi_2 \rightsquigarrow \ldots \rightsquigarrow \psi_n = \bot/\top$$

- Successively eliminate variables from a given PCNF ψ_0.
- Elimination produces satisfiability-equivalent PCNFs $\psi_i \equiv_{sat} \psi_{i+1}$.
- Worst case exponential space procedure.
- Redundancy elimination on ψ_i (depending on formula representation).
- Stop if ψ_i reduces to truth constant $\top$ or $\bot$.
- Call a SAT solver if ψ_i contains only $\exists$-variables.
- Lazy expansion by counter example guided abstraction refinement (CEGAR) [CGJ+03, JM15b, JKMSC16, RT15].

Expansion (2/4)

Example

$\psi = \exists x \forall u \exists y.\ (\bar{x} \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (u \vee \bar{y})$

- Eliminate rightmost y:
  $\psi = \exists x \forall u.\ \underbrace{[(\bar{x}) \wedge (\bar{u})]}_{y \text{ replaced by } \bot} \vee \underbrace{[(x) \wedge (u)]}_{y \text{ replaced by } \top}$
- Convert back to PCNF (distributivity):
  $\psi = \exists x \forall u.\ (\bar{x} \vee x) \wedge (\bar{x} \vee u) \wedge (x \vee \bar{u}) \wedge (u \vee \bar{u})$

Expansion of $\exists$-Variables: cf. [AB02, Bie04]

- Eliminate *rightmost* variables by Shannon expansion [Sha49].
- Replace $\hat{Q}\exists x.\phi$ by $\hat{Q}.(\phi[x/\bot] \vee \phi[x/\top])$.
- Based on CNF, NNF, and-inverter graphs [AB02, LB08, PS09].

Expansion (3/4)

Example (continued)

- Eliminate rightmost y:
  $\psi = \exists x \forall u.\ \underbrace{[(\bar{x}) \wedge (\bar{u})]}_{y \text{ replaced by } \bot} \vee \underbrace{[(x) \wedge (u)]}_{y \text{ replaced by } \top}$
- Convert to back PCNF:
  $\psi = \exists x \forall u.\ (\bar{x} \vee x) \wedge (\bar{x} \vee u) \wedge (x \vee \bar{u}) \wedge (u \vee \bar{u})$
- Simplify and reduce u: $\psi = \exists x.\ (\bar{x}) \wedge (x)$

Special Case – ψ in PCNF:

- Eliminate leftmost $\forall$-variables by universal reduction.
- Implemented in early expansion-based solvers, cf. [AB02, Bie04].

Expansion (4/4)

Example (continued)

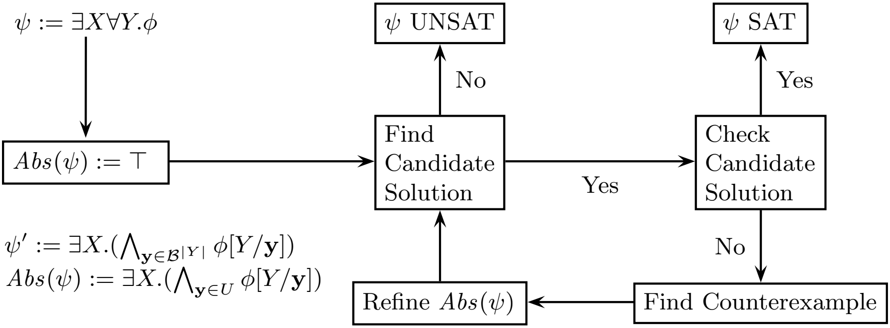
$\psi = \exists x \forall u \exists y.\ (\bar{x} \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (u \vee \bar{y})$

- Expand u: copy CNF and replace y by fresh y_d in copy of CNF.
- $\psi' = \exists x, y, y_d.\ \underbrace{(\bar{x} \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{y})}_{u \text{ replaced by } \bot} \wedge \underbrace{(\bar{x} \vee y_d) \wedge (x \vee \bar{y}_d) \wedge (y_d)}_{u \text{ replaced by } \top,\ y \text{ replaced by } y_d}$
- Obtain $(\bar{x})$ from $(\bar{x} \vee y)$ and $(\bar{y})$, (x) from $(x \vee \bar{y}_d)$ and (y_d).

Expansion of $\forall$-Variables: cf. [AB02, Bie04]

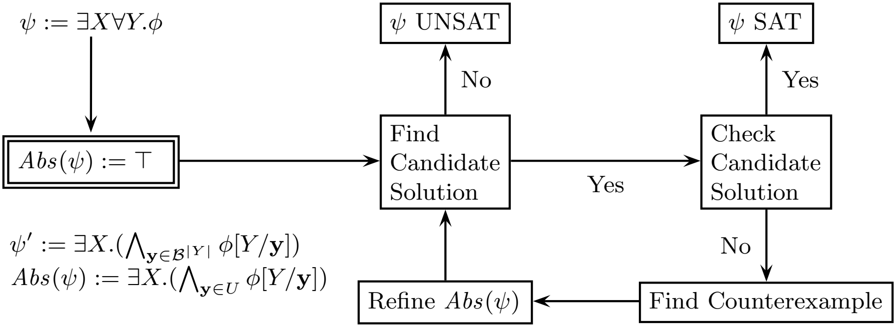
- Eliminate all universal variables by Shannon expansion.
- Finally, apply SAT solving.
- If x innermost: replace $\hat{Q}\forall x.\phi$ by $\hat{Q}.(\phi[x/\bot] \wedge \phi[x/\top])$.
- Otherwise, duplicate existential variables inner to x [Bie04, BK07].

Lazy Expansion by CEGAR



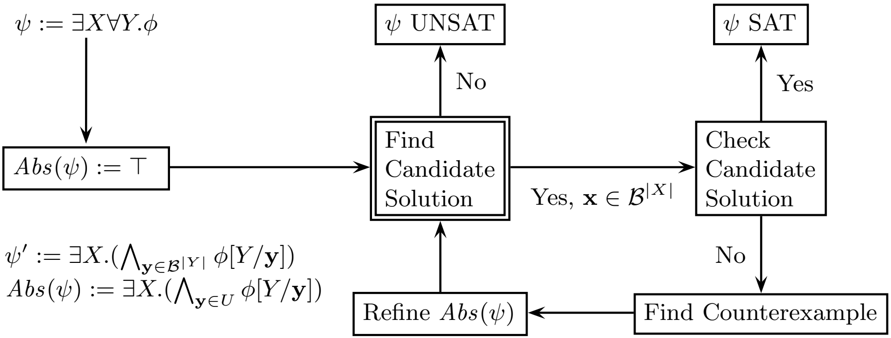
- Let $\psi := \exists X \forall Y . \ \phi$ be a one-alternation QBF, ϕ a non-CNF formula.
- ψ is satisfiable iff $\psi' := \exists X . (\bigwedge_{\mathbf{y} \in \mathcal{B}^{|Y|}} \phi[Y/\mathbf{y}])$ is satisfiable.
- Full expansion ψ' of $\forall Y$ by set $\mathcal{B}^{|Y|}$ of all possible assignments $\mathbf{y}$ of Y.
- Idea: consider a *partial expansion* of $\forall Y$ as an *abstraction* of ψ'.

Lazy Expansion by CEGAR



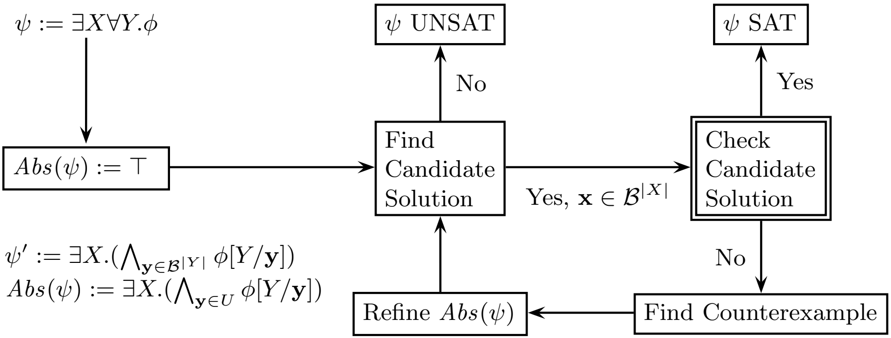
- Subset $U \subseteq \mathcal{B}^{|Y|}$ of set $\mathcal{B}^{|Y|}$ of all possible assignments $\mathbf{y}$ of Y.
- Partial expansion: given U, define $Abs(\psi) := \exists X . (\bigwedge_{\mathbf{y} \in U} \phi[Y/\mathbf{y}])$.
- Abstraction $Abs(\psi)$: if $Abs(\psi)$ unsatisfiable, then also ψ unsatisfiable.
- Initially, set $U := \emptyset$ and $Abs(\psi) := \top$.

Lazy Expansion by CEGAR



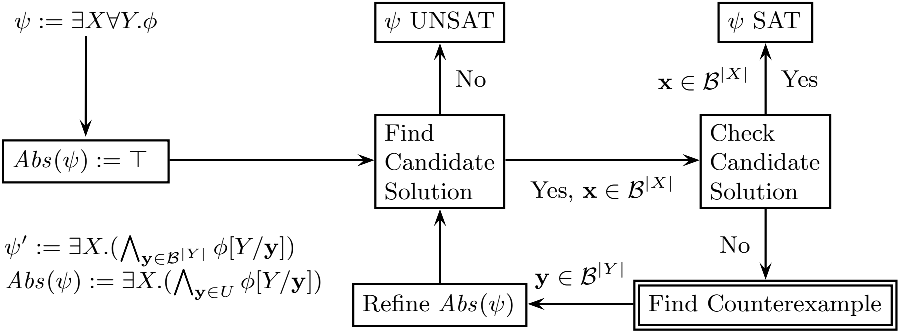
- Check satisfiability of $Abs(\psi)$ using a SAT solver.
- If $Abs(\psi)$ unsatisfiable: also ψ unsatisfiable, terminate.
- If $Abs(\psi)$ satisfiable: let $\mathbf{x} \in \mathcal{B}^{|X|}$ be a model of $Abs(\psi)$.
- $\mathbf{x} \in \mathcal{B}^{|X|}$: candidate solution of full exp. $\psi' := \exists X . (\bigwedge_{\mathbf{y} \in \mathcal{B}^{|Y|}} \phi[Y/\mathbf{y}])$.

Lazy Expansion by CEGAR



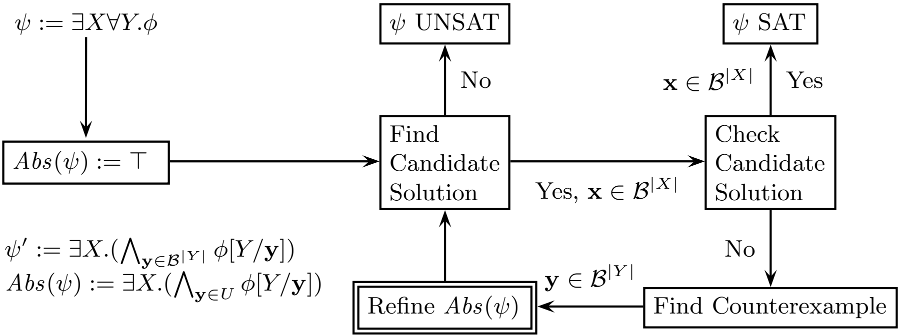
- If $\mathbf{x}$ is also a model of the full expansion ψ', then ψ is satisfiable.
- $\mathbf{x}$ is a model of full expansion ψ' iff $\forall Y.\phi[X/\mathbf{x}]$ is satisfiable.
- $\forall Y.\phi[X/\mathbf{x}]$ is satisfiable iff $\exists Y.\neg\phi[X/\mathbf{x}]$ is unsatisfiable.
- Check satisfiability of $\exists Y.\neg\phi[X/\mathbf{x}]$ using a SAT solver.

Lazy Expansion by CEGAR



- If $\exists Y . \neg\phi[X/\mathbf{x}]$ unsatisfiable: ψ is satisfiable, return $\mathbf{x}$ and terminate.
- If $\exists Y . \neg\phi[X/\mathbf{x}]$ satisfiable: let $\mathbf{y} \in \mathcal{B}^{|Y|}$ be a model of $\exists Y . \neg\phi[X/\mathbf{x}]$.
- Note: $\mathbf{y}$ is an assignment to $\forall$-variables in ψ.
- $\mathbf{y}$ is a counterexample to candidate solution $\mathbf{x}$ of full expansion ψ'.

Lazy Expansion by CEGAR



- Refine abstraction $Abs(\psi)$ by counterexample $\mathbf{y}$.
- Let $U := U \cup \{\mathbf{y}\}$ and $Abs(\psi) := \exists X . (\bigwedge_{\mathbf{y} \in U} \phi[Y/\mathbf{y}])$.
- Adding $\mathbf{y}$ to $Abs(\psi)$ prevents repetition of candidate solution $\mathbf{x}$.
- E.g. for 2QBF [RTM04, BJS$^+$16], RAReQS (recursive) [JKMSC16].

Experiments (1/6)

Benchmark Set from QBFEVAL'16:

- 825 prenex CNF instances, 1800 seconds, 7 GB memory limits.

QBF Solvers:

- Top ranked solvers from QBFEVAL'16.
- Five different solving paradigms.
- Some solvers are based on *orthogonal proof systems*.
- Theory: exponential gap in solving capabilities.

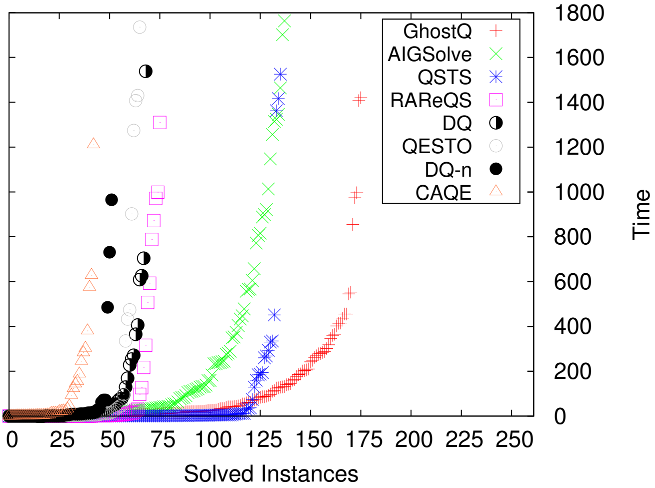
Experiments (1/6)

Alternation Bias in QBFEVAL'16 Benchmarks: cf. [LE17]

- 56% of the benchmarks have no more than two quantifier alternations.
- Theory: numbers of alternations $\approx$ levels in polynomial hierarchy.
- Focus: 402 instances not solved by preprocessing using Bloqqer [BLS11].
- Analysis wrt. instances having few/many alternations.

Experiments (2/6): 402 Filtered Instances

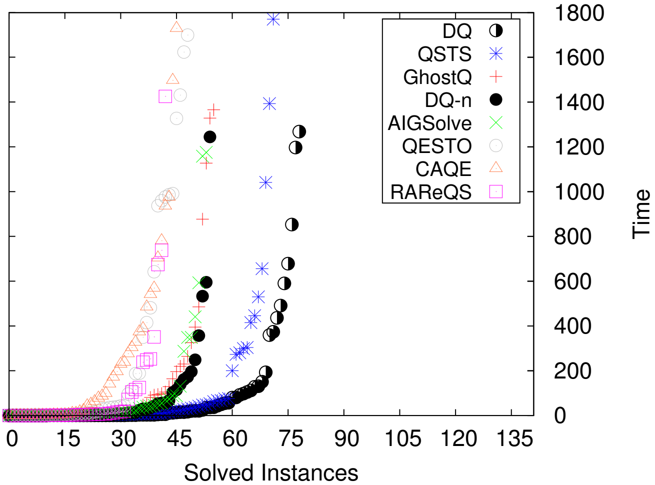
	Solved
GhostQ	176
AIGSolve	138
QSTS	136
RAReQS	76
DQ	69
QESTO	66
DQ-n	52
CAQE	43



- 261 instances (65%), $\leq$ 2 alternations, filtered but *not* preprocessed.

Experiments (2/6): 402 Filtered Instances

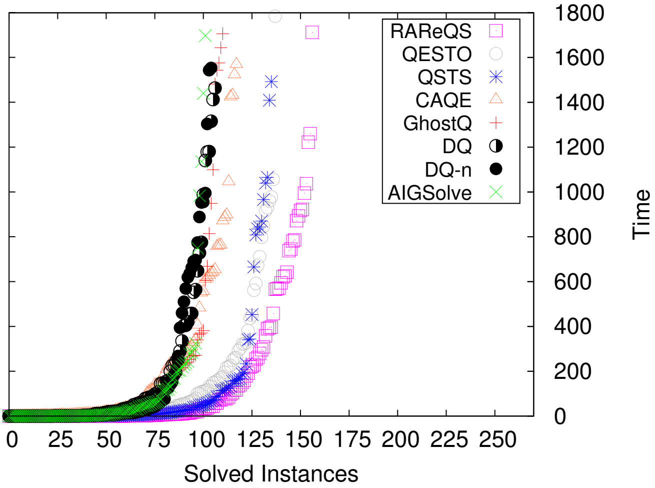
	Solved
DQ	79
QSTS	72
GhostQ	56
DQ-n	55
AIGSolve	54
QESTO	49
CAQE	46
RAReQS	43



- 141 instances (35%), ≥ 3 alternations, filtered but *not* preprocessed.
- QCDCL, e.g. DepQBF (DQ), performs better on many alternations.

Experiments (3/6): 402 Filtered Instances

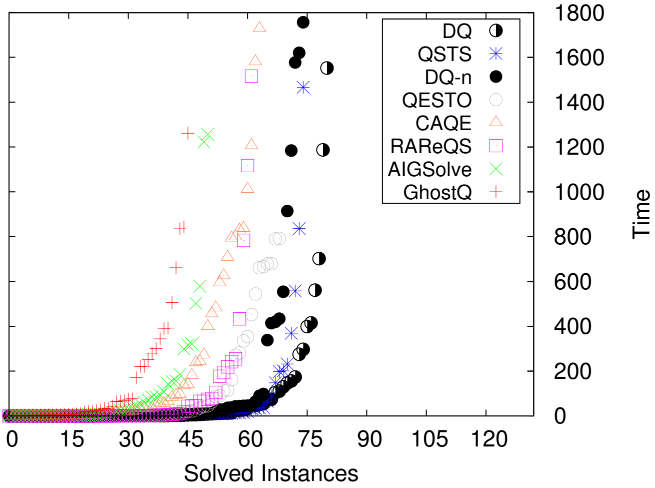
	Solved
RAReQS	157
QESTO	138
QSTS	136
CAQE	118
GhostQ	111
DQ	107
DQ-n	105
AIGSolve	102



- 270 instances (67%), $\leq$ 2 alternations, filtered *and* preprocessed.

Experiments (3/6): 402 Filtered Instances

	Solved
DQ	81
QSTS	75
DQ-n	75
QESTO	69
CAQE	64
RAReQS	62
AIGSolve	51
GhostQ	46



- 132 instances (33%), $\geq$ 3 alternations, filtered *and* preprocessed.
- QCDCL, e.g. DepQBF (DQ), performs better on many alternations.

Experiments (4/6): QBFEVAL'17

Table: Solved instances (S), solved unsatisfiable ($\perp$) and satisfiable ones ($\top$), and total wall clock time including time outs on *437 filtered instances* from QBFEVAL'17 without (a) and with preprocessing by Bloqqer (b).

Solver	S	$\perp$	$\top$	*Time*
AIGSolve	177	121	56	489K
Rev-Qfun	174	106	68	497K
GhostQ	145	79	66	547K
RAReQS	126	94	32	577K
CAQE	126	87	39	578K
Heretic	122	95	27	580K
DepQBF-opt	115	78	37	603K
Ijtihad	110	88	22	599K
QSTS-d	103	75	28	618K
Qute-random	77	47	30	658K
QESTO	76	56	20	661K
DynQBF	47	27	20	714K

(a) Not preprocessed.

Solver	S	$\perp$	$\top$	*Time*
RAReQS	175	127	48	499K
CAQE	169	114	55	514K
Heretic	164	119	45	513K
AIGSolve	138	98	40	555K
Ijtihad	136	103	33	555K
Rev-Qfun	135	92	43	563K
QSTS-d	127	98	29	576K
QESTO	115	84	31	601K
DepQBF-opt	102	64	38	624K
GhostQ	82	47	35	661K
Qute-random	73	56	17	672K
DynQBF	65	37	28	684K

(b) Preprocessed by Bloqqer.

Table: Instances solved in *437 filtered instances not preprocessed* by Bloqqer with respect to classes by number of quantifier blocks (#q) and number of formulas in each class (#f).

#q	#f	*AIGSolve*	*Rev-Qfun*	*GhostQ*	*RAReQS*	*CAQE*	*Heretic*	*DepQBF-opt*	*Ijtihad*	*QSTS-d*	*Qute-random*	*QESTO*	*DynQBF*
2	63	**33**	17	32	2	5	2	6	2	8	2	4	18
3	215	83	**101**	89	62	56	50	47	49	43	36	35	19
4–6	36	**27**	16	3	16	20	16	6	16	14	6	2	0
7–9	27	**19**	9	1	17	5	18	8	16	7	4	4	4
10–15	15	0	2	0	3	2	4	**10**	0	0	2	1	0
16–20	21	2	4	3	3	8	7	**10**	4	8	5	7	1
21–	60	13	25	17	23	**30**	25	28	23	23	22	23	5
2–3	278	116	118	**121**	64	61	52	53	51	51	38	39	37
4–	159	61	56	24	62	65	**70**	62	59	52	39	37	10

Experiments (6/6): QBFEVAL'17

Table: Instances solved in *437 filtered instances preprocessed* by Bloqqer with respect to classes by number of quantifier blocks (#q) and number of formulas in each class (#f).

#q	#f	*RAReQS*	*CAQE*	*Heretic*	*AIGSolve*	*Ijtihad*	*Rev-Qfun*	*QSTS-d*	*QESTO*	*DepQBF-opt*	*GhostQ*	*Qute-random*	*DynQBF*
2	65	16	15	13	12	10	6	11	14	7	3	4	**24**
3	218	80	**81**	65	65	59	65	46	50	34	53	23	18
4–6	32	18	20	17	**24**	17	19	17	6	5	2	7	8
7–9	27	19	6	**21**	19	18	10	8	7	9	3	3	5
10–15	25	13	9	14	3	8	10	8	7	**15**	6	11	3
16–20	28	12	16	**17**	4	9	12	16	11	15	3	10	1
21–	42	17	**22**	17	11	15	13	21	20	17	12	15	6
2–3	283	96	**96**	78	77	69	71	57	64	41	56	27	42
4–	154	79	73	**86**	61	67	64	70	51	61	26	46	23

Outlook and Future Work

Outlook and Future Work (1/2)

QBF in Practice:

- QBF tools are not (yet) a push-button technology.
- Pitfalls: Tseitin encodings, premature preprocessing.
- Goal: integrated workflow without the need for manual intervention.

Challenges:

- Extracting proofs and certificates in workflows including preprocessing [HSB14a, HSB14b] and incremental solving [MMLB12, LE14].
- Integrating *dependency schemes* [SS09, LB10, VG11, PSS16, PSS17] in workflows to relax the linear quantifier ordering.
- Implementations of QCDCL do not harness the full power of Q-resolution [Jan16].
- Combining strengths of orthogonal solving approaches.

Outlook and Future Work (2/2)

- QBF is still an emerging field with plenty of applications.
- Assuming that NP $\neq$ PSPACE, QBF is more difficult than SAT...
- ...but allows for exponentially more succinct encodings than SAT.
- Recent theoretical progress: QBF proof systems.
- Computational hardness motivates exploring alternative approaches: e.g. CEGAR-based expansion, computing Skolem functions [RS16].
- Expert and/or domain knowledge may be necessary for tuning.
- Please document and publish your tools and benchmarks!

Appendix

[Appendix] Expansion and Instantiation

Definition (∀Exp+RES [JM13, BCJ14, JM15a])

- Axiom: $\overline{C}$ for all $x \in \hat{Q}$: $\{x, \bar{x}\} \not\subseteq C$ and $C \in \phi$
- Instantiation: $\dfrac{C}{\{l^{A_l} \mid l \in C, q(l) = \exists\}}$

  Complete assignment A to universal variables s.t. literals in C falsified, $A_l \subseteq A$ restricted to universal variables u with $u < l$.
- Resolution: $\dfrac{C_1 \cup \{p^A\} \qquad C_2 \cup \{\bar{p}^A\}}{C_1 \cup C_2}$ for all $x \in \hat{Q}$: $\{x, \bar{x}\} \not\subseteq (C_1 \cup C_2)$

- First, instantiate (i.e. replace) all universal variables by constants.
- Existential literals in a clause are annotated by partial assignments.
- Finally, resolve on existential literals with matching annotations.
- Instantiation and annotation mimics universal expansion.

[Appendix] Expansion and Instantiation

Example (continued)

$\psi = \exists x \forall u \exists y. \; (\bar{x} \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (u \vee \bar{y})$

- Complete assignments: $A = \{\bar{u}\}$ and $A' = \{u\}$.
- Instantiate: $(\bar{x} \vee y^{\bar{u}}) \wedge (x \vee \bar{y}^{u}) \wedge (y^{u}) \wedge (\bar{y}^{\bar{u}})$
- Note: cannot resolve (y^{u}) and $(\bar{y}^{\bar{u}})$ due to mismatching annotations.
- Obtain (x) from $(x \vee \bar{y}^{u})$ and (y^{u}), $(\bar{x})$ from $(\bar{x} \vee y^{\bar{u}})$ and $(\bar{y}^{\bar{u}})$.

Different Power of QBF Proof Systems:

- Q-resolution and expansion/instantiation are incomparable [BCJ15].
- Interpreting QBFs as first-order logic formulas [SLB12, Egl16].

[Appendix] QBFs as First-Order Logic Formulas

Definition (QBF to FOL Translation [SLB12])

Mapping $[\![\cdot]\!] : QBF \rightarrow FOL$ with respect to unary FOL predicate p:

$$[\![\exists x.\phi]\!] = \exists x.[\![\phi]\!] \qquad [\![\forall x.\phi]\!] = \forall x.[\![\phi]\!]$$

$$[\![\phi \vee \psi]\!] = [\![\phi]\!] \vee [\![\psi]\!] \qquad [\![\phi \wedge \psi]\!] = [\![\phi]\!] \wedge [\![\psi]\!]$$

$$[\![x]\!] = p(x) \qquad [\![\neg\psi]\!] = \neg[\![\psi]\!]$$

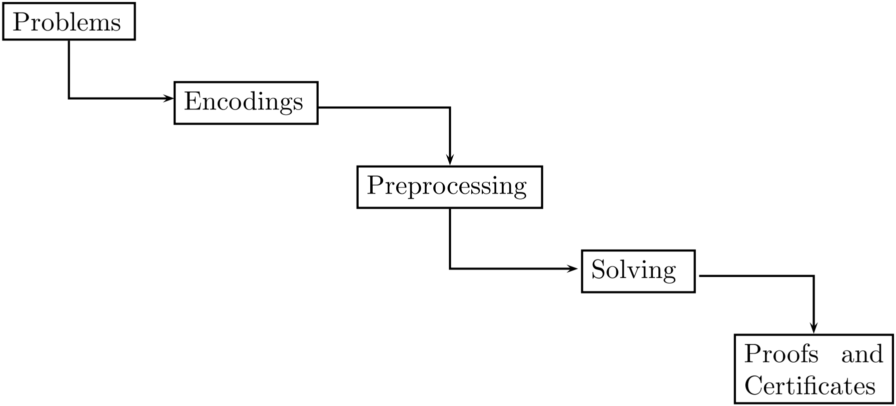
$$[\![\top]\!] = p(\mathit{true}) \qquad [\![\bot]\!] = p(\mathit{false})$$

It holds that $p(\mathit{true})$ ($p(\mathit{false})$) is true (false) in every FOL interpretation.

Proposition ([SLB12])

The QBF ψ is satisfiable iff $[\![\psi]\!] \wedge p(\mathit{true}) \wedge \neg p(\mathit{false})$ is satisfiable.

[Appendix] Typical QBF Workflow



QCIR: Quantified CIRcuit

- Format for QBFs in non-prenex non-CNF.
- Conversion tools, e.g., part of GhostQ solver [Gho16, KSGC10].

2 Format Specification

2.1 Syntax

The following BNF grammar specifies the structure of a formula represented in QCIR (Quantified CIRcuit).

qcir-file	::=	*format-id qblock-stmt output-stmt (gate-stmt nl)**
format-id	::=	`#QCIR-G14` [*integer*] *nl*
qblock-stmt	::=	[`free` (*var-list*) *nl*] *qblock-quant**
qblock-quant	::=	*quant* (*var-list*) *nl*
var-list	::=	(*var* ,)* *var*
lit-list	::=	(*lit* ,)* *lit* | ϵ
output-stmt	::=	`output` (*lit*) *nl*
gate-stmt	::=	*gvar* = *ngate_type* (*lit-list*)
	|	*gvar* = `xor` (*lit* , *lit*)
	|	*gvar* = `ite` (*lit* , *lit* , *lit*)
	|	*gvar* = *quant* (*var-list* ; *lit*)
quant	::=	`exists` | `forall`
var	::=	(A string of ASCII letters, digits, and underscores)
gvar	::=	(A string of ASCII letters, digits, and underscores)
nl	::=	newline
lit	::=	*var* | -*var* | *gvar* | -*gvar*
ngate_type	::=	`and` | `or`

3.2 Formula in Non-Prenex Form

A formula in non-prenex form looks as follows:

```
#QCIR-G14

forall(z)

output(g3)

g1 = and(x1, x2, z)

g2 = exists(x1, x2; g1)

g3 = or(z, g2)
```

$$\forall z.\ \overbrace{z \lor \underbrace{\exists x_1.\exists x_2.\ \underbrace{(x_1 \land x_2 \land z)}_{g_1}}_{g_2}}^{g_3}$$

From [QCI14]: `http://qbf.satisfiability.org/gallery/qcir-gallery14.pdf`

[Appendix] Encodings (2)

Definition (Prenexing, cf. [AB02, Egl94, EST$^+$03, ETW02, GNT07])

$(Qx.\ \phi) \circ \psi \equiv Qx.\ (\phi \circ \psi)$, ψ a QBF, $Q \in \{\forall, \exists\}, \circ \in \{\wedge, \vee\}$, $x \notin Var(\psi)$.

Definition (CNF transformation, cf. [Tse68, NW01, PG86])

- Given a prenex QBF $\psi := \hat{Q}.\phi$, subformulas ψ_i of ψ.
- $\psi_i = (\psi_{i,l} \circ \psi_{i,r})$, $\circ \in \{\vee, \wedge, \rightarrow, \leftrightarrow, \otimes\}$.
- Add equivalences $t_i \leftrightarrow (\psi_{i,l} \circ \psi_{i,r})$, fresh variable t_i.
- Convert each $t_i \leftrightarrow (\psi_{i,l} \circ \psi_{i,r})$ to CNF depending on $\circ$.
- Resulting PCNF ψ': satisfiability-equivalent to ψ, size linear in $|\psi|$.
- Safe: quantify each t_i innermost [GMN09]: $\psi := \hat{Q}\exists t_i.\phi$.

[Appendix] Encodings (3)

Definition (QBF Extension Rule, cf. [Tse68, JBS+07, BCJ16])

- Let $\psi := Q_1 x_1 \dots Q_i x_i \dots Q_j x_j \dots Q_n x_n.\phi$ be a PCNF.
- Consider variables x_i, x_j with $x_i \leq x_j$ in ψ, fresh existential variable v.
- Add definition $v \leftrightarrow (\bar{x}_i \vee \bar{x}_j)$ in CNF: $(\bar{v} \vee \bar{x}_i \vee \bar{x}_j) \wedge (v \vee x_i) \wedge (v \vee x_j)$.
- Strong variant: quantify v after x_j, $Q_1 x_1 \dots Q_i x_i \dots Q_j x_j \exists v \dots Q_n x_n$.
- Weak variant: quantify v innermost, $Q_1 x_1 \dots Q_i x_i \dots Q_j x_j \dots Q_n x_n \exists v$.

Proposition (cf. [JBS+07, BCJ16])

Q-resolution with the strong extension rule is exponentially more powerful than with the weak extension rule with respect to lengths of refutations.

$\Rightarrow$ "bad" placement of Tseitin variables in encoding phase may have negative impact on solving in a later stage.

[Appendix] Encodings (4): QParity

Definition (QParity Function [BCJ15])

$QParity_n := \exists x_1, \ldots, x_n \forall y.\ XOR(XOR(\ldots XOR(x_1, x_2), \ldots, x_n), y)$.

CNF ϕ of $QParity_n$ by Tseitin translation:

$$
\begin{array}{c}
(t_1 \leftrightarrow XOR(x_1, x_2)) \wedge \\
\bigwedge_{1<i<n} (t_i \leftrightarrow XOR(t_{i-1}, x_{i+1})) \wedge \\
(t_n \leftrightarrow XOR(t_{n-1}, y)) \wedge (t_n)
\end{array}
$$

Prefix by weak extension rule : $\hat{Q}_W := \exists x_1, \ldots, x_n \forall y \exists t_1, \ldots, t_n$

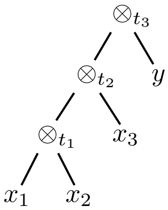
Prefix by strong extension rule: $\hat{Q}_S := \exists x_1, \ldots, x_n \exists t_1, \ldots, t_{n-1} \forall y \exists t_n$

Proposition ([BCJ15, BCJ16])

- *The PCNF $\hat{Q}_W.\phi$ has only exponential Q-resolution refutations.*
- *The PCNF $\hat{Q}_S.\phi$ has polynomial Q-resolution refutations.*

[Appendix] Encodings (5): QParity

$$\hat{Q}_W.\phi := \exists x_1, x_2, x_3 \forall y \quad . \; XOR_3(XOR_2(XOR_1(x_1, x_2), x_3), y)$$

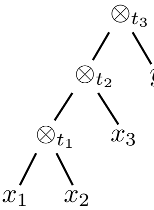


$$\begin{aligned} t_1 &\leftrightarrow XOR(x_1, x_2) \\ t_2 &\leftrightarrow XOR(t_1, x_3) \\ t_3 &\leftrightarrow XOR(t_2, y) \end{aligned}$$

$$\begin{array}{ll}
t_1: & (\bar{t}_1 \vee x_1 \vee x_2) \wedge \\
 & (\bar{t}_1 \vee \bar{x}_1 \vee \bar{x}_2) \wedge \\
 & (t_1 \vee \bar{x}_1 \vee x_2) \wedge \\
 & (t_1 \vee x_1 \vee \bar{x}_2) \wedge \\
\hline
t_2: & (\bar{t}_2 \vee t_1 \vee x_3) \wedge \\
 & (\bar{t}_2 \vee \bar{t}_1 \vee \bar{x}_3) \wedge \\
 & (t_2 \vee \bar{t}_1 \vee x_3) \wedge \\
 & (t_2 \vee t_1 \vee \bar{x}_3) \wedge \\
\hline
t_3: & (\bar{t}_3 \vee t_2 \vee y) \wedge \\
 & (\bar{t}_3 \vee \bar{t}_2 \vee \bar{y}) \wedge \\
 & (t_3 \vee \bar{t}_2 \vee y) \wedge \\
 & (t_3 \vee t_2 \vee \bar{y}) \wedge \\
\hline
out: & (t_3)
\end{array}$$

[Appendix] Encodings (5): QParity

$$\hat{Q}_W.\phi := \exists x_1, x_2, x_3 \forall y \exists t_1, t_2, t_3.\ XOR_3(XOR_2(XOR_1(x_1, x_2), x_3), y)$$



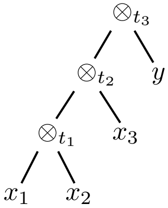
$t_1 \leftrightarrow XOR(x_1, x_2)$

$t_2 \leftrightarrow XOR(t_1, x_3)$

$t_3 \leftrightarrow XOR(t_2, y)$

$$\begin{array}{ll}
t_1: & (\bar{t}_1 \vee x_1 \vee x_2) \wedge \\
 & (\bar{t}_1 \vee \bar{x}_1 \vee \bar{x}_2) \wedge \\
 & (t_1 \vee \bar{x}_1 \vee x_2) \wedge \\
 & (t_1 \vee x_1 \vee \bar{x}_2) \wedge \\
\hline
t_2: & (\bar{t}_2 \vee t_1 \vee x_3) \wedge \\
 & (\bar{t}_2 \vee \bar{t}_1 \vee \bar{x}_3) \wedge \\
 & (t_2 \vee \bar{t}_1 \vee x_3) \wedge \\
 & (t_2 \vee t_1 \vee \bar{x}_3) \wedge \\
\hline
t_3: & (\bar{t}_3 \vee t_2 \vee y) \wedge \\
 & (\bar{t}_3 \vee \bar{t}_2 \vee \bar{y}) \wedge \\
 & (t_3 \vee \bar{t}_2 \vee y) \wedge \\
 & (t_3 \vee t_2 \vee \bar{y}) \wedge \\
\hline
out: & (t_3)
\end{array}$$

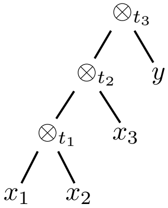
$$\hat{Q}_S.\phi := \exists x_1, x_2, x_3 \qquad \forall y \qquad . \; XOR_3(XOR_2(XOR_1(x_1, x_2), x_3), y)$$



$$t_1 \leftrightarrow XOR(x_1, x_2)$$
$$t_2 \leftrightarrow XOR(t_1, x_3)$$
$$t_3 \leftrightarrow XOR(t_2, y)$$

$$\begin{array}{ll}
t_1 : & (\bar{t}_1 \vee x_1 \vee x_2) \wedge \\
 & (\bar{t}_1 \vee \bar{x}_1 \vee \bar{x}_2) \wedge \\
 & (t_1 \vee \bar{x}_1 \vee x_2) \wedge \\
 & (t_1 \vee x_1 \vee \bar{x}_2) \wedge \\
\hline
t_2 : & (\bar{t}_2 \vee t_1 \vee x_3) \wedge \\
 & (\bar{t}_2 \vee \bar{t}_1 \vee \bar{x}_3) \wedge \\
 & (t_2 \vee \bar{t}_1 \vee x_3) \wedge \\
 & (t_2 \vee t_1 \vee \bar{x}_3) \wedge \\
\hline
t_3 : & (\bar{t}_3 \vee t_2 \vee y) \wedge \\
 & (\bar{t}_3 \vee \bar{t}_2 \vee \bar{y}) \wedge \\
 & (t_3 \vee \bar{t}_2 \vee y) \wedge \\
 & (t_3 \vee t_2 \vee \bar{y}) \wedge \\
\hline
out : & (t_3)
\end{array}$$

$\hat{Q}_S.\phi := \exists x_1, x_2, x_3, t_1, t_2 \forall y \exists t_3.\ XOR_3(XOR_2(XOR_1(x_1, x_2), x_3), y)$



$t_1 \leftrightarrow XOR(x_1, x_2)$
$t_2 \leftrightarrow XOR(t_1, x_3)$
$t_3 \leftrightarrow XOR(t_2, y)$

$$
\begin{array}{ll}
t_1: & (\bar{t}_1 \lor x_1 \lor x_2) \land \\
 & (\bar{t}_1 \lor \bar{x}_1 \lor \bar{x}_2) \land \\
 & (t_1 \lor \bar{x}_1 \lor x_2) \land \\
 & (t_1 \lor x_1 \lor \bar{x}_2) \land \\
\hline
t_2: & (\bar{t}_2 \lor t_1 \lor x_3) \land \\
 & (\bar{t}_2 \lor \bar{t}_1 \lor \bar{x}_3) \land \\
 & (t_2 \lor \bar{t}_1 \lor x_3) \land \\
 & (t_2 \lor t_1 \lor \bar{x}_3) \land \\
\hline
t_3: & (\bar{t}_3 \lor t_2 \lor y) \land \\
 & (\bar{t}_3 \lor \bar{t}_2 \lor \bar{y}) \land \\
 & (t_3 \lor \bar{t}_2 \lor y) \land \\
 & (t_3 \lor t_2 \lor \bar{y}) \land \\
\hline
out: & (t_3)
\end{array}
$$

[Appendix] QBF Solving by Clause Selection

Example (Clause Selection and Clausal Abstraction [JM15b, RT15])

Let $\psi := \forall X \exists Y.\ \phi$ be a one-alternation QBF, ϕ a CNF.

- ψ unsatisfiable iff, for some $\mathbf{x} \in \mathcal{B}^{|X|}$, $\exists Y.\ \phi[X/\mathbf{x}]$ unsatisfiable.
- Think of $\mathbf{x} \in \mathcal{B}^{|X|}$ as a selection $\phi_S^{\mathbf{x}} \subseteq \phi$ of clauses.
- Clause $C \in \phi_S^{\mathbf{x}}$ iff C not satisfied by $\mathbf{x}$, i.e. $C[X/\mathbf{x}] \neq \top$.
- If $\exists Y.\ \phi_S^{\mathbf{x}}[X/\mathbf{x}]$ unsatisfiable then $\exists Y.\ \phi[X/\mathbf{x}]$ and ψ unsatisfiable.
- Otherwise, consider model $\mathbf{y} \in \mathcal{B}^{|Y|}$ of $\exists Y.\ \phi_S^{\mathbf{x}}[X/\mathbf{x}]$.
- Find new $\mathbf{x}' \in \mathcal{B}^{|X|}$ such that there exists $C \in \phi_S^{\mathbf{x}'}$ with $C[Y/\mathbf{y}] \neq \top$.
- If no such $\mathbf{x}'$ exists then ψ is satisfiable.
- CEGAR: find candidate solutions $\mathbf{x}$ and counterexamples $\mathbf{y}$ by SAT solving, refinement step blocks unsuccessful selections $\phi_S^{\mathbf{x}}$.

References

References I

Please note: since the duration of this talk is limited, the list of references below is incomplete and does not reflect the history and state of the art in QBF research in full accuracy.

[AB02] Abdelwaheb Ayari and David A. Basin.
QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers.
In *FMCAD*, volume 2517 of *LNCS*, pages 187–201. Springer, 2002.

[BB09] Hans Kleine Büning and Uwe Bubeck.
Theory of Quantified Boolean Formulas.
In *Handbook of Satisfiability*, volume 185 of *FAIA*, pages 735–760. IOS Press, 2009.

[BCCZ99] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu.
Symbolic Model Checking without BDDs.
In *TACAS*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.

[BCJ14] Olaf Beyersdorff, Leroy Chew, and Mikolas Janota.
On unification of QBF resolution-based calculi.
In *MFCS*, volume 8635 of *LNCS*, pages 81–93. Springer, 2014.

References II

[BCJ15] Olaf Beyersdorff, Leroy Chew, and Mikolás Janota.
Proof Complexity of Resolution-based QBF Calculi.
In *STACS*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76–89. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[BCJ16] Olaf Beyersdorff, Leroy Chew, and Mikolas Janota.
Extension Variables in QBF Resolution.
Electronic Colloquium on Computational Complexity (ECCC), 23:5, 2016.
Beyond NP Workshop 2016 at AAAI-16.

[Bie04] Armin Biere.
Resolve and Expand.
In *SAT*, volume 3542 of *LNCS*, pages 59–70. Springer, 2004.

[BJ11] Valeriy Balabanov and Jie-Hong R. Jiang.
Resolution Proofs and Skolem Functions in QBF Evaluation and Applications.
In *CAV*, volume 6806 of *LNCS*, pages 149–164. Springer, 2011.

[BJ12] Valeriy Balabanov and Jie-Hong R. Jiang.
Unified QBF certification and its applications.
Formal Methods in System Design, 41(1):45–65, 2012.

References III

[BJJW15] Valeriy Balabanov, Jie-Hong Roland Jiang, Mikolas Janota, and Magdalena Widl.
Efficient Extraction of QBF (Counter)models from Long-Distance Resolution Proofs.
In *AAAI*, pages 3694–3701. AAAI Press, 2015.

[BJS+16] Valeriy Balabanov, Jie-Hong Roland Jiang, Christoph Scholl, Alan Mishchenko, and Robert K. Brayton.
2QBF: Challenges and Solutions.
In *SAT*, volume 9710 of *LNCS*, pages 453–469. Springer, 2016.

[BK07] Uwe Bubeck and Hans Kleine Büning.
Bounded Universal Expansion for Preprocessing QBF.
In *SAT*, volume 4501 of *LNCS*, pages 244–257. Springer, 2007.

[BKF95] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel.
Resolution for Quantified Boolean Formulas.
Inf. Comput., 117(1):12–18, 1995.

[BLS11] Armin Biere, Florian Lonsing, and Martina Seidl.
Blocked Clause Elimination for QBF.
In *CADE*, volume 6803 of *LNCS*, pages 101–115. Springer, 2011.

References IV

[BM08] Marco Benedetti and Hratch Mangassarian.
QBF-Based Formal Verification: Experience and Perspectives.
JSAT, 5(1-4):133–191, 2008.

[CDG+15] Günther Charwat, Wolfgang Dvorák, Sarah Alice Gaggl, Johannes Peter Wallner, and Stefan Woltran.
Methods for solving reasoning problems in abstract argumentation - A survey.
Artif. Intell., 220:28–63, 2015.

[CGJ+03] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith.
Counterexample-guided abstraction refinement for symbolic model checking.
J. ACM, 50(5):752–794, 2003.

[CGS98] Marco Cadoli, Andrea Giovanardi, and Marco Schaerf.
An Algorithm to Evaluate Quantified Boolean Formulae.
In *AAAI*, pages 262–267. AAAI Press / The MIT Press, 1998.

[CHR16] Chih-Hong Cheng, Yassine Hamza, and Harald Ruess.
Structural Synthesis for GXW Specifications.
In *CAV*, volume 9779 of *LNCS*, pages 95–117. Springer, 2016.

References V

[CSGG02] Marco Cadoli, Marco Schaerf, Andrea Giovanardi, and Massimo Giovanardi.
An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation.
JAIR, 28(2):101–142, 2002.

[DLL62] Martin Davis, George Logemann, and Donald W. Loveland.
A Machine Program for Theorem-Proving.
Commun. ACM, 5(7):394–397, 1962.

[DP60] Martin Davis and Hilary Putnam.
A Computing Procedure for Quantification Theory.
J. ACM, 7(3):201–215, 1960.

[Egl94] Uwe Egly.
On the Value of Antiprenexing.
In *LPAR*, volume 822 of *LNCS*, pages 69–83. Springer, 1994.

[Egl16] Uwe Egly.
On Stronger Calculi for QBFs.
In *SAT*, volume 9710 of *LNCS*, pages 419–434. Springer, 2016.

References VI

[ELW13] Uwe Egly, Florian Lonsing, and Magdalena Widl.
Long-Distance Resolution: Proof Generation and Strategy Extraction in Search-Based QBF Solving.
In *LPAR*, volume 8312 of *LNCS*, pages 291–308. Springer, 2013.

[EST+03] Uwe Egly, Martina Seidl, Hans Tompits, Stefan Woltran, and Michael Zolda.
Comparing Different Prenexing Strategies for Quantified Boolean Formulas.
In *SAT*, volume 2919 of *LNCS*, pages 214–228. Springer, 2003.

[ETW02] Uwe Egly, Hans Tompits, and Stefan Woltran.
On Quantifier Shifting for Quantified Boolean Formulas.
In *In Proceedings of the SAT-02 Workshop on Theory and Applications of Quantified Boolean Formulas (QBF-02*, pages 48–61, 2002.

[FFRT17] Peter Faymonville, Bernd Finkbeiner, Markus N. Rabe, and Leander Tentrup.
Encodings of Bounded Synthesis.
In *TACAS*, volume 10205 of *LNCS*, pages 354–370. Springer, 2017.

[FR05] Wolfgang Faber and Francesco Ricca.
Solving Hard ASP Programs Efficiently.
In *LPNMR*, volume 3662 of *LNCS*, pages 240–252. Springer, 2005.

References VII

[FT14] Bernd Finkbeiner and Leander Tentrup.
Fast DQBF Refutation.
In *SAT*, volume 8561 of *LNCS*, pages 243–251. Springer, 2014.

[FT15] Bernd Finkbeiner and Leander Tentrup.
Detecting Unrealizability of Distributed Fault-tolerant Systems.
Logical Methods in Computer Science, 11(3), 2015.

[GGB11] Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus.
A Uniform Approach for Generating Proofs and Strategies for Both True and False QBF Formulas.
In *IJCAI*, pages 546–553. IJCAI/AAAI, 2011.

[GGN$^+$04] Ian P. Gent, Enrico Giunchiglia, Massimo Narizzano, Andrew G. D. Rowley, and Armando Tacchella.
Watched Data Structures for QBF Solvers.
In *SAT*, volume 2919 of *LNCS*, pages 25–36. Springer, 2004.

[Gho16] GhostQ: A QBF Solver, 2010–2016.
`http://www.cs.cmu.edu/~wklieber/ghostq/`.

References VIII

[GMN09] Enrico Giunchiglia, Paolo Marin, and Massimo Narizzano.
Reasoning with Quantified Boolean Formulas.
In *Handbook of Satisfiability*, volume 185 of *FAIA*, pages 761–780. IOS Press, 2009.

[GNT02] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella.
Learning for Quantified Boolean Logic Satisfiability.
In *AAAI*, pages 649–654. AAAI Press / The MIT Press, 2002.

[GNT06] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella.
Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas.
JAIR, 26:371–416, 2006.

[GNT07] E. Giunchiglia, M. Narizzano, and A. Tacchella.
Quantifier Structure in Search-Based Procedures for QBFs.
TCAD, 26(3):497–507, 2007.

[GT14] Adria Gascón and Ashish Tiwari.
A Synthesized Algorithm for Interactive Consistency.
In *NASA Formal Methods*, volume 8430 of *LNCS*, pages 270–284. Springer, 2014.

References IX

[HJL+15] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere.
Clause Elimination for SAT and QSAT.
JAIR, 53:127–168, 2015.

[HK17] Marijn J. H. Heule and Oliver Kullmann.
The science of brute force.
Commun. ACM, 60(8):70–79, 2017.

[HKM16] Marijn J. H. Heule, Oliver Kullmann, and Victor W. Marek.
Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer.
In *SAT*, volume 9710 of *LNCS*, pages 228–245. Springer, 2016.

[HSB14a] Marijn Heule, Martina Seidl, and Armin Biere.
A Unified Proof System for QBF Preprocessing.
In *IJCAR*, volume 8562 of *LNCS*, pages 91–106. Springer, 2014.

[HSB14b] Marijn Heule, Martina Seidl, and Armin Biere.
Efficient extraction of Skolem functions from QRAT proofs.
In *FMCAD*, pages 107–114. IEEE, 2014.

References X

[HSM+14] Tamir Heyman, Dan Smith, Yogesh Mahajan, Lance Leong, and Husam Abu-Haimed.
Dominant Controllability Check Using QBF-Solver and Netlist Optimizer.
In *SAT*, volume 8561 of *LNCS*, pages 227–242. Springer, 2014.

[Jan16] Mikolás Janota.
On Q-Resolution and CDCL QBF Solving.
In *SAT*, volume 9710 of *LNCS*, pages 402–418. Springer, 2016.

[JB07] Toni Jussila and Armin Biere.
Compressing BMC Encodings with QBF.
Electr. Notes Theor. Comput. Sci., 174(3):45–56, 2007.

[JBS+07] Toni Jussila, Armin Biere, Carsten Sinz, Daniel Kröning, and Christoph M. Wintersteiger.
A First Step Towards a Unified Proof Checker for QBF.
In *SAT*, volume 4501 of *LNCS*, pages 201–214. Springer, 2007.

[JKMSC16] Mikoláš Janota, William Klieber, Joao Marques-Silva, and Edmund Clarke.
Solving QBF with counterexample guided refinement.
Artificial Intelligence, 234:1–25, 2016.

References XI

[JM13] Mikolás Janota and João Marques-Silva.
On Propositional QBF Expansions and Q-Resolution.
In *SAT*, volume 7962 of *LNCS*, pages 67–82. Springer, 2013.

[JM15a] Mikolás Janota and Joao Marques-Silva.
Expansion-based QBF solving versus Q-resolution.
Theor. Comput. Sci., 577:25–42, 2015.

[JM15b] Mikolás Janota and Joao Marques-Silva.
Solving QBF by Clause Selection.
In *IJCAI*, pages 325–331. AAAI Press, 2015.

[JS11] Mikolás Janota and João P. Marques Silva.
On Deciding MUS Membership with QBF.
In *CP*, volume 6876 of *LNCS*, pages 414–428. Springer, 2011.

[KSGC10] William Klieber, Samir Sapra, Sicun Gao, and Edmund M. Clarke.
A Non-prenex, Non-clausal QBF Solver with Game-State Learning.
In *SAT*, volume 6175 of *LNCS*, pages 128–142. Springer, 2010.

[Kul99] Oliver Kullmann.
On a Generalization of Extended Resolution.
Discrete Applied Mathematics, 96-97:149–176, 1999.

References XII

[LB08] Florian Lonsing and Armin Biere.
Nenofex: Expanding NNF for QBF Solving.
In *SAT*, volume 4996 of *LNCS*, pages 196–210. Springer, 2008.

[LB10] Florian Lonsing and Armin Biere.
Integrating Dependency Schemes in Search-Based QBF Solvers.
In *SAT*, volume 6175 of *LNCS*, pages 158–171. Springer, 2010.

[LE14] Florian Lonsing and Uwe Egly.
Incremental QBF Solving.
In *CP*, volume 8656 of *LNCS*, pages 514–530. Springer, 2014.

[LE17] Florian Lonsing and Uwe Egly.
Evaluating QBF Solvers: Quantifier Alternations Matter.
CoRR, abs/1701.06612, 2017.
technical report.

[LEG13] Florian Lonsing, Uwe Egly, and Allen Van Gelder.
Efficient clause learning for quantified boolean formulas via QBF pseudo unit propagation.
In *SAT*, volume 7962 of *LNCS*, pages 100–115. Springer, 2013.

References XIII

[Let02] Reinhold Letz.
Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas.
In *TABLEAUX*, volume 2381 of *LNCS*, pages 160–175. Springer, 2002.

[Lib05] Paolo Liberatore.
Redundancy in logic I: CNF propositional formulae.
Artif. Intell., 163(2):203–232, 2005.

[MMLB12] Paolo Marin, Christian Miller, Matthew D. T. Lewis, and Bernd Becker.
Verification of partial designs using incremental QBF solving.
In *DATE*, pages 623–628. IEEE, 2012.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik.
Chaff: Engineering an Efficient SAT Solver.
In *DAC*, pages 530–535. ACM, 2001.

[MS72] Albert R. Meyer and Larry J. Stockmeyer.
The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space.
In *13th Annual Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society, 1972.

References XIV

[NW01] Andreas Nonnengart and Christoph Weidenbach.
Computing Small Clause Normal Forms.
In *Handbook of Automated Reasoning*, pages 335–367. Elsevier and MIT Press, 2001.

[PG86] David A. Plaisted and Steven Greenbaum.
A Structure-Preserving Clause Form Translation.
J. Symb. Comput., 2(3):293–304, 1986.

[PS09] Florian Pigorsch and Christoph Scholl.
Exploiting structure in an AIG based QBF solver.
In *DATE*, pages 1596–1601. IEEE, 2009.

[PSS16] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider.
Long Distance Q-Resolution with Dependency Schemes.
In *SAT*, volume 9710 of *LNCS*, pages 500–518. Springer, 2016.

[PSS17] Tomás Peitl, Friedrich Slivovsky, and Stefan Szeider.
Dependency Learning for QBF.
In *SAT*, volume 10491 of *LNCS*, pages 298–313. Springer, 2017.

[QCI14] QCIR-G14: A Non-Prenex Non-CNF Format for Quantified Boolean Formulas, 2014.
`http://qbf.satisfiability.org/gallery/qcir-gallery14.pdf`.

References XV

[RBM97] Anavai Ramesh, George Becker, and Neil V. Murray.
CNF and DNF Considered Harmful for Computing Prime Implicants/Implicates.
JAIR, 18(3):337–356, 1997.

[Rin07] Jussi Rintanen.
Asymptotically Optimal Encodings of Conformant Planning in QBF.
In *AAAI*, pages 1045–1050. AAAI Press, 2007.

[RS16] Markus N. Rabe and Sanjit A. Seshia.
Incremental Determinization.
In *SAT*, volume 9710 of *LNCS*, pages 375–392. Springer, 2016.

[RT15] Markus N. Rabe and Leander Tentrup.
CAQE: A Certifying QBF Solver.
In *FMCAD*, pages 136–143. IEEE, 2015.

[RTM04] Darsh P. Ranjan, Daijue Tang, and Sharad Malik.
A Comparative Study of 2QBF Algorithms.
In *SAT*, 2004.

[SC85] A. Prasad Sistla and Edmund M. Clarke.
The Complexity of Propositional Linear Temporal Logics.
J. ACM, 32(3):733–749, 1985.

References XVI

[Sch78] Thomas J Schaefer.
On the Complexity of Some Two-Person Perfect-Information Games.
Journal of Computer and System Sciences, 16(2):185–225, 1978.

[Sha49] Claude Elwood Shannon.
The Synthesis of Two-Terminal Switching Circuits.
Bell System Technical Journal, 28(1):59–98, 1949.

[SLB12] Martina Seidl, Florian Lonsing, and Armin Biere.
qbf2epr: A Tool for Generating EPR Formulas from QBF.
In *PAAR Workshop*, volume 21 of *EPiC Series*, pages 139–148. EasyChair, 2012.

[SS96] João P. Marques Silva and Karem A. Sakallah.
GRASP - a new search algorithm for satisfiability.
In *ICCAD*, pages 220–227, 1996.

[SS99] João P. Marques Silva and Karem A. Sakallah.
GRASP: A Search Algorithm for Propositional Satisfiability.
IEEE Trans. Computers, 48(5):506–521, 1999.

[SS09] Marko Samer and Stefan Szeider.
Backdoor Sets of Quantified Boolean Formulas.
JAR, 42(1):77–97, 2009.

References XVII

[Sto76] Larry J. Stockmeyer.
The Polynomial-Time Hierarchy.
Theor. Comput. Sci., 3(1):1–22, 1976.

[Tse68] G. S. Tseitin.
On the Complexity of Derivation in Propositional Calculus.
Studies in Constructive Mathematics and Mathematical Logic, 1968.

[VG11] Allen Van Gelder.
Variable Independence and Resolution Paths for Quantified Boolean Formulas.
In *CP*, volume 6876 of *LNCS*, pages 789–803. Springer, 2011.

[VG12] Allen Van Gelder.
Contributions to the Theory of Practical Quantified Boolean Formula Solving.
In *CP*, volume 7514 of *LNCS*, pages 647–663. Springer, 2012.

[Wra76] Celia Wrathall.
Complete Sets and the Polynomial-Time Hierarchy.
Theor. Comput. Sci., 3(1):23–33, 1976.

[ZM02a] Lintao Zhang and Sharad Malik.
Conflict Driven Learning in a Quantified Boolean Satisfiability Solver.
In *ICCAD*, pages 442–449. ACM / IEEE Computer Society, 2002.

References XVIII

[ZM02b] Lintao Zhang and Sharad Malik.
Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation.
In *CP*, volume 2470 of *LNCS*, pages 200–215. Springer, 2002.