Quantified Boolean Formulas: Solving and Proofs

Solving

Dr. Joshua Blinkhorn

Friedrich-Schiller-Universität Jena
https://github.com/JoshuaBlinkhorn/QBF

1

Overview

Solving technologies

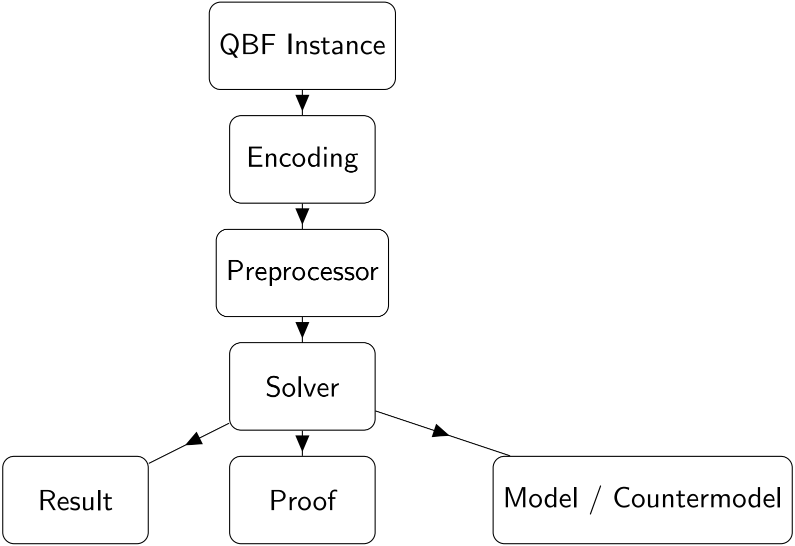
SAT	NP	established efficient technology
QBF	PSPACE	happening now
DQBF	NEXP	in its infancy

Leading Solvers

- In SAT, QCDCL is the dominant solving paradigm
- In QBF, there are various competitive paradigms

Solver	**Paradigm**	**Proof System**
RAReQS	CEGAR	$\forall$Exp+Res (with NP oracle)
CAQE	Clausal Abstraction	Level-ordered Q-Res
Dep-QBF	QCDCL	LD-Q-Res
Dep-QBF	Dependency awareness	Q($\mathcal{D}$)-Res
Qute	Dependency learning	LD-Q-Res

QBF Solving Workflow



The DIMACS CNF Encoding

- machine readable encoding
- variables are natural numbers: $x_1 \mapsto 1$, $x_2 \mapsto 2$ etc.
- negation represented by minus: $\overline{x_1} \mapsto -1$, $\overline{x_2} \mapsto -2$ etc.

$$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_3}) \wedge (\overline{x_1} \vee x_3)$$

```
p cnf 3 4
1 2 3 0
-1 -2 0
-3 0
-1 3 0
```

The QDIMACS Prenex QCNF Encoding

- extends DIMACS
- existential quantifier represented by ‘e’
- universal quantifier represented by ‘a’

$\exists x_1 \exists x_2 \forall x_3 \exists x_4 \cdot (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_2}) \wedge (\overline{x_3} \vee x_4)$

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

2

Preprocessing

Why Preprocess?

- Preprocessors attempt to simplify a QBF while preserving its truth value
- Notion: easier to solve after preprocessing
- Usually, this means reducing the number of variables and the number of clauses
- There are a wide variety of preprocessing techniques
- The proof system QRAT was introduced to cover all of them
- Leading QBF preprocessors: bloqqer and HQS-Pre

Purely Propositional Techniques

- Propositional preprocessing techniques that are logically correct still work for QBFs
- Subsumption:
  $$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge D \wedge E \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i) \wedge D$$
  provided D is a subclause of E
- Strengthening:
  $$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee a) \wedge (E \vee \overline{a}) \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee \overline{a}) \wedge E$$
  provided D is a subclause of E

Pure Literal Elimination

- Pure literal elimination is not propositionally logically correct; it only preserves satisfiability
- Works differently for existentials and universals
- Existential version:

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge \bigwedge_j (D_j \vee a) \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i)$$

  provided a is existential, $\overline{a}$ doesn't appear in $(\bigwedge_i C_i) \wedge (\bigwedge_j D_j)$

- Universal version

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge \bigwedge_j (D_j \vee a) \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i) \wedge \bigwedge_j (D_j)$$

  provided a is universal, $\overline{a}$ doesn't appear in $(\bigwedge_i C_i) \wedge (\bigwedge_j D_j)$

Unit Literal Elimination

- Unit literal elimination is also not propositionally logically correct; but it does preserve satisfiability
- It can only be applied on existential unit clauses:

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (a) \quad \Rightarrow \quad (\mathcal{Q} \cdot \bigwedge_i C_i)[\alpha]$$

  provided a is existential, and α is the smallest assignment satisfying a

- Any QBF containing a universal unit clause is false

Universal Reduction

- Universal reduction is logically correct in terms of QBF models
- So it preserves QBF truth value

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee a) \quad \Rightarrow \quad \mathcal{Q} \cdot \bigwedge_i (C_i) \wedge D$$

  provided a is universal, and var(a) is quantified after all existentials in D, and $(D \vee a)$ is not a tautology

- As a consequence: we can often assume that the final block of a QBF with a CNF matrix is existentially quantified

Blocked Clause Elimination

- Blocked clauses play a key role in SAT preprocessing
- It is an example of a redundancy property
- A redundancy property defines clauses that can be removed (or added) to a CNF while preserving satisfiability
- Propositionally, clause B is blocked w.r.t. a CNF F if B contains a literal for which all resolvents with F are tautologies
- The quantified version again requires a tweak:

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee a) \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i)$$

provided a is existential, and for all C_i containing $\overline{a}$, $C_i \otimes_{\overline{a}} D$ has complimentary literals in a variable left of var(a)

Blocked Literal Elimination

- This is the universal analogue of blocked clause elimination
- It allows a universal literal to be removed from a clause:

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee a) \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i) \wedge D$$

  provided a is universal, and for all C_i containing $\overline{a}$, $C_i \otimes_{\overline{a}} D$ has complimentary literals in a variable left of var(a)

- In contrast to universal reduction, the removed literal is not necessarily right of all existential in the clause

Covered Literal Addition

- Preoprocessors sometimes add literals to clauses
- This can actually be useful - for example, it may increase the set of models for a true QBF
- Covered literal addition

$$\mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee a) \quad \Rightarrow \quad \mathcal{Q} \cdot (\bigwedge_i C_i) \wedge (D \vee a \vee b)$$

  provided a is existential, var(b) is left of var(a), and for all C_i containing $\overline{a}$, either :
  - b is in C_i, or
  - $C_i \vee D$ has complimentary literals in a variable left of var(a)

Existential Variable Elimination

- A method of removing existential variables in the final block
- Based on DP Resolution (Davis-Putnam)
- Propositionally:
    1. take a CNF F
    2. choose a variable x
    3. add all resolvents over x to F
    4. remove all clauses containing x
- This process preserves satisfiability - so it forms a CNF decision procedure
- For QBF, it can be performed on existentials in the final block while preserving truth value
- Hence, it forms a decision procedure for QBF in combination with universal reduction

Universal Expansion

- Expansion of single universal variables preserves truth value
- Preprocessors may perform some universal expansions where it is considered beneficial
- This is a form of partial expansion (but it is not a partial expansion w.r.t. a subset of total universal assignments)
- Guided by heuristics

Ownership and Acknowledgement

- In many cases, the QBF is solved completely in preprocessing
- This raises the question of acknowledgement - for example, in competitions (QBFEVAL)
- Janota: "I used MiniSAT and the C compiler!"

3

Expansion-based Solving

Recap

- The expansion of a QBF $\Phi = \mathcal{Q} \cdot F$ is the CNF

$$\exp(\Phi) := \bigcup_{\alpha \in \langle \mathsf{vars}_{\forall}(\Phi) \rangle} F\left[\alpha \cup \left\{x \mapsto x_{\alpha \restriction L(x)} : x \in \mathsf{vars}_{\exists}(\Phi)\right\}\right]$$

- The partial expansion of a QBF $\Phi = \mathcal{Q} \cdot F$ w.r.t. a set of universal assignments $R \subseteq \langle \mathsf{vars}_{\forall}(\Phi) \rangle$ is the CNF

$$\exp(\Phi, R) := \bigcup_{\alpha \in R} F\left[\alpha \cup \left\{x \mapsto x_{\alpha \restriction L(x)} : x \in \mathsf{vars}_{\exists}(\Phi)\right\}\right]$$

- The partial expansion may be unsatisfiable even when $R \subset \langle \mathsf{vars}_{\forall}(\Phi) \rangle$ is a proper subset of universal assignments

Basic Expansion Decision Procedure

- Arguably the easiest way to solve a QBF Φ:
  1. Write $\exp(\Phi)$ in DIMACS
  2. Pass it to a SAT solver
- Benefit: easy implementation - all work done by SAT solver
- SAT solver employed as an NP oracle
- Drawback: expansion is expensive
- Just computing the expansion takes exponentional time if there are linearly many universal variables, even if the expansion is small

$$\exp(\forall u_1 \cdots \forall u_n \cdot \top) = \top$$

- It makes sense to work with partial expansions

Benders Decomposition

- A techinque for solving linear programming problems
- Exploits block structure of a problem (variable set can be partitioned)
- Divide-and-conquer approach:
  - Divide variables into two sets A and B
  - Solve the *master problem* over A
  - For each candidate solution to the master problem, solve a *subproblem* over B
  - If the subproblem is insoluble, generate a *cut* and add it to the master problem
  - The cut *rules out* the candidate: it will not be selected again
  - Resolve the master problem until no more cuts can be added

Basic Benders Decomposition Approach to QBF

- Consider a QBF $\Phi := \forall U \exists X \cdot F$
- A *winning move* for Φ is $\alpha \in \langle U \rangle$ such that $F[\alpha]$ is unsatisfiable
- Goal: find a winning move for Φ, if one exists
  1. Maintain a set of moves $A \subseteq \langle U \rangle$, initally empty
  2. Find a move $\alpha \in \langle U \rangle$ not in A
  3. Determine whether $F[\alpha]$ is satisfiable with a SAT solver
  4. If not, return α
  5. If so, add α to A
  6. If $A \neq \langle U \rangle$, goto line 2
- Drawback: if Φ is true, all assignments in $\langle U \rangle$ *will* be tested
- In other words: total universal expansion of Φ is constructed
- No information from subproblem passed to master problem

An Extreme Example

- Consider what would happen with this QBF

  $\forall u \forall v \exists x \cdot (u \vee v \vee x) \wedge (u \vee \overline{v} \vee x) \wedge (\overline{u} \vee v \vee x) \wedge (\overline{u} \vee \overline{v} \vee x)$

- Under *every* assignment to $\{u, v\}$, matrix satisfied by $x \mapsto 1$
- SAT solver outputs this in *each* of the four subproblems

- Satisfying assignment to a subproblem explains why a candidate move fails
- We also call this a *counterexample* for the candidate
- In this case, it happens to be the same counterexample for each candidate
- Idea: add counterexamples back into the master problem

Benders Decomposition Done Better

- Find a winning move for $\Phi := \forall U \exists X \cdot F$

  1. Maintain a set of CNFs A in the variables U, initally empty
  2. Find a candidate move $\alpha \in \langle U \rangle$ that falsifies all CNFs in A
  3. Determine whether $F[\alpha]$ is satisfiable with a SAT solver
  4. If not, return α
  5. If so, collect the satisfying assignment β, add $F[\beta]$ to A
  6. Goto line 2

- β is a counterexample to α
- β is also a counterexample to any α' satisfying $F[\beta]$
- Hence, in line 2, if no such move exists, then Φ is true, because every candidate has a counterexample
- The set A is called an *abstraction*

Extreme Example Revisited

- Consider again the QBF

  $\forall u \forall v \exists x \cdot (u \vee v \vee x) \wedge (u \vee \overline{v} \vee x) \wedge (\overline{u} \vee v \vee x) \wedge (\overline{u} \vee \overline{v} \vee x)$

- Regardless of which candidate in $\langle \{u, v\} \rangle$ is chosen first, the counterexample $x \mapsto 1$ is found, and $A = \{\top\}$
- Since $\top$ has no falsifying assignments, we deduce that the QBF is true

- In this case, we only needed to consider a single candidate
- We avoided constructing the total universal expansion
- Essentially, we constructed a partial expansion, whose counterexamples formed a satisfiable abstraction

Quantifiers Exchanged - the Σ_2 Version

- Consider a QBF $\Phi := \exists X \forall U \cdot F$
- A *winning move* for Φ is $\alpha \in \langle X \rangle$ such that $F[\alpha]$ is a tautology
- Goal: find a winning move for Φ, if one exists

  1. Maintain a set of CNFs A in the variables X, initally empty
  2. Find a candidate move $\alpha \in \langle X \rangle$ that satisfies all CNFs in A
  3. Determine whether $F[\alpha]$ is a tautology with a SAT solver
  4. If so, return α
  5. If not, collect the falsifying assignment β, add $F[\beta]$ to A
  6. Goto line 2

- β is a counterexample to α and any α' falsifying $F[\beta]$
- Hence, in line 2, if no such move exists, then Φ is false, because every candidate has a counterexample

Connections to Countermodels

- For a false Σ_2 QBF $\Phi := \exists X \forall U \cdot F$, the set of counterexamples forms the range of a countermodel
  - Why? every candidate has a counterexample *amongst those encoutered*
  - Hence, for each $\alpha \in \langle X \rangle$ a counterexample $\beta \in \langle U \rangle$ was encountered such that $\alpha \cup \beta$ falsifies F
  - In Σ_2, a countermodel is exactly such a mapping
- Hence we must encounter at least $\sigma(\Phi)$ counterexamples, where $\sigma(\Phi)$ is the minimum range of a countermodel for Φ
- Therefore $\sigma(\Phi)$ is a lower bound on the running time of the algorithm
- The final abstraction is essentially the partial expansion of Φ with respect to the set of counterexamples discovered

CEGAR Solving

- CEGAR: Counterexample-guided Abstraction Refinement
- A form of Benders decomposition for solving QBF
- Block structure from quantifier prefix: $\forall U_1 \exists X_1 \cdots \forall U_n \exists X_n$
- A leading CEGAR solver: RAReQs by Janota

Multi-Games

- Merely convenient notation for the pseudocode
- Definition: A multi-game is an expression of the form $QZ \cdot \{\Phi_1, \ldots, \Phi_n\}$ where
  - Q is a quantifier and Z is a block of variables
  - the Φ_i are prenex QBFs whose only free variables are from Z
  - the Φ_i all have the same prefix $\mathcal{Q}$
  - the first quantifier of $\mathcal{Q}$ (if it is not the empty prefix) is opposite to Q
  - the variables of $\mathcal{Q}$ are disjoint from Z
- A winning move for a multigame is an assignment $\alpha \in \langle Z \rangle$ such that
  - if $Q = \exists$, all $\Phi_i[\alpha]$ are true
  - if $Q = \forall$, all $\Phi_i[\alpha]$ are false
- Without loss of generality: assume final block is existential

RAReQs Pseudocode

Function: RAReQs($QZ \cdot \{\Phi_1, \ldots, \Phi_n\}$)
Output: A winning move for Q, or NULL if none exist

if Φ_i have no quantifiers **then return** `SAT`($\bigwedge_i \Phi_i$)
$A \leftarrow \emptyset$
$\Psi \leftarrow QZ \cdot A$ `// form initial empty abstraction`
while `true` **do**
    $\alpha' =$ RAReQs(Ψ) `// seek a winning move for the abstraction`
    if $\alpha' =$ NULL **then return** NULL
    $\alpha \rightarrow \alpha' \restriction_Z$ `// filter a move for` Z
    for $i \in [n]$ do $\mu_i \leftarrow$ RAReQS($\Phi_i[\tau]$) `// look for a counterexample`
    if $\mu_i =$ NULL for all $i \in [n]$ **return** τ
    let $i \in [n]$ such that $\mu_i \neq$ NULL
    Remove QZ from the prefix of Φ_i
    $A \leftarrow A \cup \{\Phi_i[\mu_i]\}$ `// refine the abstraction`
end

The Key to RAReQS' Success

- According to the author, RAReQS is based on $\forall$Exp+Res
- A formal proof that an $\forall$Exp+Res refutation can be extracted from the solver trace on a false QBF has not been given
- RAReQs is arguably most successful expansion-based solver
- Key to success: abstraction limits the amount of expansion
- Building the abstraction and solving it is a serious overhead
- Trade-off against the benefit of partial expansion appears favourable

RAReQS and Countermodels

- Consider RAReQS on a false QBF Φ
- Imagine the winning moves found for each universal block, concatenated with those from the recursive calls
- This generates a set S of total universal assignments
- S is the range of a countermodel
- So $\exp(\Phi, S)$ is unsatisfiable
- Suggestion: RAReQS based on $\forall$Exp+Res with an NP oracle
- Hence minimal countermodel range $\sigma(\Phi)$ is a lower bound for the algorithm running time
- Corollary: equality formulas should be hard for RAReQs