Evaluating Quantified Boolean Formulas
by means of Skizzo

Marco Benedetti
benedetti@itc.it
Outline of the talk

1. Introduction to QBF Evaluation
   - Motivation, and examples
   - Existing approaches

2. Our approach (and sKizzo)
   - Quantifier Tree Reconstruction
   - Symbolic Skolemization
   - Hybrid Inference Engine
   - Certification and Strategies

3. Results
   - Benchmarks and Experiments
   - Direct and indirect feedback

4. Final Remarks
Quantified Boolean Formulas (QBFs) are propositional formulas in which variables are existentially/universally quantified. It is a decidable logic. PSPACE-complete.

A simple example (in prenex conjunctive form):

\[ \forall a \exists b \forall c \exists d. (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor b) \]
Quantified Boolean Formulas (QBFs) are propositional formulas in which variables are existentially/universally quantified. It is a decidable logic. PSPACE-complete.

Another simple example:

Open problem: compute the smallest integer $N=|M|$ such that:

$$\exists A \forall M \left[ \text{sudoku}(A) \land \text{cellSet}(M) \land \forall B \left[ \text{sudoku}(B) \land A =_M B \rightarrow A = B \right] \right]$$
Quantified Boolean Formulas (QBFs) are propositional formulas in which variables are existentially/universally quantified. It is a decidable logic. PSPACE-complete.

Relevant to applications:
- Two-player games
- Planning
- Formal verification
- Hand-crafted
-...

Most problems generate parametrically scalable families of instances. Up to 1 million clauses/variables, tens of quantifier alternations.

More than 70 families and 5000 instances. Pointers to about 15 solvers. Conventional form for instances: Prenex Conjunctive Normal Form

$$\forall a \exists b \forall c \exists d. (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor b)$$
Quantified Boolean Formulas (QBFs) are propositional formulas in which variables are existentially/universally quantified. It is a decidable logic. PSPACE-complete.

Relevant to applications:

- Two-player games
- Planning
- Formal verification
- Hand-crafted

Shared repository

- QuBe
- Quaffle
- Semprop
- Quantor
- ZQSAT

Applications

- Model Checking [Biere, Dershowitz, Hanna, Katz]
- Reachability for HW circuits [Katz]
- Protocol verification [Ayari]
- Web Service composition [Pistore]
- Convergence checking problems [Lahiri, Seshia]
- Sequential depth computation [Mneimneh, Sakallah]
- Equivalence of partial implementation [Scholl, Becker]
Quantified Boolean Formulas (QBFs) are propositional formulas in which variables are existentially/universally quantified. It is a decidable logic. PSPACE-complete.

Relevant to applications:

- Two-player games
- Planning
- Formal verification
- Hand-crafted

Applications
- QuBe
- Quaffle
- Semprop
- Quantor
- ZQSAT

Shared repository

Model Checking [Biere, Dershowitz, Hanna, Katz]

\[ \exists S_0 \exists S_1 \ldots \exists S_n. I(S_0) \land F(S_n) \land \]
\[ \forall V \forall W \left[ (\lor_{i=0}^n V \leftrightarrow S_i \land W \leftrightarrow S_{i+1}) \rightarrow TR(V, W) \right] \]

...with \( I \): initial states - \( F \): bad states - \( TR \): Transition Relation

The CNF version of which has \( \exists \forall \exists \) alternation: \( \exists S \forall V \forall W \exists A. \text{CNF}(S, V, W, A) \)
Previous work

**Standard approach:** extend the DPLL search procedure to QBFs
(originating in [Cadoli, Giovanardi & Schaerf, 1998, journal version in JAR:28(2), 2002])

Explore the **semantic evaluation tree** of the formula via a *depth-first visit* following the ordering of variables given in the prefix.

**Alternative approaches:**
- Rather than search, *solve* QBFs by means of q-resolution/expansion [Biere, SAT2004]
- Use BDDs/ZDDs to support resolution [Pan & Vardi, CP2004]
- Use ZDDs to support search [Gashem Zadeh & al., Tech. rep., 2004]
- Exploit SAT solvers [Ayari & Basin, FMCAD 2002] [Samulowitz & Bacchus, CP05]
State of the Art

- QBF solvers are no longer proof-of-concept implementations:
  - The first “efficient” QBF solver is 8 years old [Cadoli, Giovanardi, Schaefer, 1997]
  - More than 10 solvers take part in the yearly “QBF evaluation” event;

- But:
  - Techniques borrowed from the purely-existential case (SAT) do not suffice.
  - Many “real-world” instances from applications are hard (never solved).
  - QBF is challenging: new decision procedures needed to make it applicable.

- Interesting quotes:
  - “QBF solvers are still lagging behind special purpose SAT and modal-K solvers” [Le Berre, Simon, Tacchella, 2003]
  - “[Given these evidences] it is clear that QBF based model checking still needs a long way to go” [Biere, 2004]
  - “We found that modern [...] QBF solvers are still unable to handle real-life instances of BMC problems...” [Dershowitz, Hanna, Katz, 2005]
Outline of the talk

1. Introduction to QBF Evaluation
   - Motivation, and examples
   - Existing approaches

2. Our approach (and sKizzo)
   - Quantifier Tree Reconstruction
   - Symbolic Skolemization
   - Hybrid Inference Engine
   - Certification and Strategies

3. Results
   - Benchmarks and Experiments
   - Direct and indirect feedback

4. Final Remarks
The **prenex conjunctive normal form** for QBFs enables us to

1. decouple the generation of instances from the solution of those instances;
2. exercise different QBF solvers against the same encodings;
3. profit from the shared *knowledge-base* of the QBFLIB community;

To mitigate its disadvantages (...) we rebuild **ex-post** part of the hidden internal structure of instances: a polynomial-time algorithm extracts **quantifier trees**.

\[
\forall a \exists b \exists c \forall d \exists e \exists f \exists g \exists h. \\
(a \lor \neg c) \land (a \lor h) \land (c \lor \neg d \lor g) \land (\neg a \lor b \lor f) \land \\
(c \lor e \lor \neg h) \land (\neg b \lor \neg f) \land (a \lor c \lor \neg g)
\]

Quantifier trees can be exploited in three ways:

1. **Simpler** skolem functions
   - relevant to skolemization-based solvers
2. **Divide-et-impera** behaviour
   - relevant to search-based solvers
3. Better **ordering heuristics** for resolution
   - relevant to resolution-based solvers
We map QBFs onto different (SAT-equivalent) instances. Three-step process:

1. Original QBF instance: A prefix followed by a set of clauses

2. Purely universal FOL conjunctive normal form

3. Outer skolemization is applied:
   - Skolem terms are substituted for existential variables.
   - Satisfiability is preserved.
   - No longer a QBF instance.

Example:

\[ \forall a \exists b \forall c \exists d. (-a \lor c \lor d) \land (-b \lor \neg d) \land (a \lor b \lor \neg d) \land (-a \lor \neg b) \]

...becomes:

\[ [\exists b \exists d] \forall a \forall c. (-a \lor c \lor d(a, b)) \land (-b(a) \lor \neg d(a, b)) \land (a \lor b(a) \lor \neg d(a, b)) \land (-a \lor \neg b(a)) \]
Our approach

We map QBFs onto different (SAT-equivalent) instances. Three-step process:

1. Original QBF instance: A prefix followed by a set of clauses

2. Purely universal FOL conjunctive normal form

3. Purely existential CNF propositional instance

The consistency of the skolem terms is encoded into propositional logic.

Inductive translation:

1. Propositional CNF formulas to represent the functions that interpret skolem terms;
2. Such propositional expansions are substituted for skolem terms;

- Satisfiability is preserved, and
- We have a SAT instance, but
- The resulting instance is (possibly) exponentially larger than the originating QBF.
We map QBFs onto different (SAT-equivalent) instances. Three-step process:

1. **Original QBF instance:** A prefix followed by a set of clauses
2. **Purely universal FOL conjunctive normal form**
3. **Purely existential CNF propositional instance**
4. **Symbolic QBF:** Set of *symbolic clauses* (plus a quantifier tree)

**ROBDDs** are leveraged to obtain an exponentially more succinct representation for C-instances.
We map QBFs onto different (SAT-equivalent) instances. Three-step process:

1. Original QBF instance: A prefix followed by a set of clauses

2. Purely universal FOL conjunctive normal form

3. Purely existential CNF propositional instance

Symbolic QBF: Set of symbolic clauses (plus a quantifier tree)

Note: We need to avoid managing C-instances ($10^X, X>100$ clauses!):
- The three steps are merged into a single A $\Rightarrow$ D translation.
Our approach

We map QBFs onto different (SAT-equivalent) instances. Three-step process:

1. Original QBF instance: A prefix followed by a set of clauses
2. Purely universal FOL conjunctive normal form
3. Purely existential CNF propositional instance
4. Symbolic QBF: Set of symbolic clauses (plus a quantifier tree)

We evaluate QBFs by symbolically reasoning on D-represented instances. Two styles:

- Incomplete Reasoning (normalization):
  - Pure literal elimination, unit clause propagation, subsumption, ...
  - Binary hyper-resolution, equivalency reasoning, ...

- Complete Reasoning:
  - Symbolic directional resolution;
  - Symbolic branching procedure;
  - Backward D → C translation followed by SAT-based reasoning.
Propositional Skolemization

(Interpretation of) Skolem functions for QBFs are *special cases*: \( \{0, 1\}^n \rightarrow \{0, 1\} \)

*Example:* skolemize \( s \) in \( \forall a \forall b \forall c \exists s. M(a, b, c, s) \)

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( s(a, b, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( S_{000} )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( S_{001} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( S_{010} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( S_{011} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( S_{100} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( S_{101} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( S_{110} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( S_{111} )</td>
</tr>
</tbody>
</table>

Any mapping \( s : \{0, 1\}^3 \rightarrow \{0, 1\} \) is completely specified by \( 2^3 = 8 \) boolean parameters.
(Interpretation of) Skolem functions for QBFs are special cases: \( \{0, 1\}^n \rightarrow \{0, 1\} \)

**Example:** skolemize \( s \) in \( \forall a \forall b \forall c \exists s. M(a, b, c, s) \)

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( s(a, b, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( s_{000} )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( s_{001} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( s_{010} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( s_{011} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( s_{100} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( s_{101} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( s_{110} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( s_{111} )</td>
</tr>
</tbody>
</table>

Plain 8-clause CNF representation for the resulting function: \( \text{Prop}(s(a, b, c)) \)
(Interpretation of) Skolem functions for QBFs are *special cases:*

\[ \{0, 1\}^n \rightarrow \{0, 1\} \]

*Example: skolemize* \( s \) in \( \forall a \forall b \forall c \exists s. M(a, b, c, s) \)

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
<th>( s(a, b, c) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( S_{000} )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( S_{001} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( S_{010} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( S_{011} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( S_{100} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( S_{101} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( S_{110} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( S_{111} )</td>
</tr>
</tbody>
</table>

\[ (a \lor b \lor c \lor S_{000}) \land \\
(a \lor b \lor c \lor S_{001}) \land \\
(a \lor \neg b \lor c \lor S_{010}) \land \\
(a \lor \neg b \lor c \lor S_{011}) \land \\
(\neg a \lor b \lor c \lor S_{100}) \land \\
(\neg a \lor b \lor c \lor S_{101}) \land \\
(\neg a \lor \neg b \lor c \lor S_{110}) \land \\
(\neg a \lor \neg b \lor c \lor S_{111}) \land \]

*Example: the value of* \( s(a, b, c) \) *on* \( \langle 1, 0, 1 \rangle \in \{0, 1\}^3 \) *is:*

\[ Prop(s(a, b, c))[a=\text{TRUE},b=\text{FALSE},c=\text{TRUE}] \]
Let us compute the propositional skolemization of:

$$\forall a \exists b \forall c \exists d. (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor \neg b)$$

Substitution of $\text{Prop}(d(a,b))$ for $d(a,b)$:

$$\exists b \exists d \forall a \forall c. (\neg a \lor c \lor d(a,c)) \land (\neg b(a) \lor \neg d(a,c)) \land (a \lor b(a) \lor \neg d(a,c)) \land (\neg a \lor \neg b(a))$$

Distribution:

$$\neg a \lor c \lor [ (a \lor c \lor d_{00}) \land (a \lor \neg c \lor d_{01}) \land (\neg a \lor c \lor d_{10}) \land (\neg a \lor \neg c \lor d_{11}) ]$$

For all-reduction:

$$\neg a \lor c \lor a \lor c \lor d_{00} \land \neg a \lor c \lor a \lor \neg c \lor d_{10} \land \neg a \lor c \lor \neg a \lor c \lor d_{10} \land \neg a \lor c \lor \neg a \lor c \lor d_{11}$$

Final result:

$$(d_{10})$$
Let us compute the propositional skolemization of:

\[
\forall a \exists b \forall c \exists d. \, (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor \neg b)
\]

\[
[\exists b \exists d] \forall a \forall c. \, (\neg a \lor c \lor d(a, b)) \land (\neg b(a) \lor \neg d(a, b)) \land (a \lor b(a) \lor \neg d(a, b)) \land (\neg a \lor \neg b(a))
\]

\[
(d_{10}) \land (\neg b_0 \lor \neg d_{00}) \land (\neg b_0 \lor \neg d_{01}) \land (\neg b_1 \lor \neg d_{10}) \land (\neg b_1 \lor \neg d_{11}) \land (b_0 \lor \neg d_{00}) \land (b_0 \lor \neg d_{01}) \land (\neg b_1)
\]

\[
2^{2-2} = 1 \quad 2^{2-0} = 4 \quad 2^{2-1} = 2 \quad 2^{1-1} = 1
\]

We obtain \(2^{\delta(C) - u(C)}\) clauses out of each QBF clause \(C\), where:

- \(\delta(C)\) is the number of universal variables dominating the deepest literal in \(C\).
- \(u(C)\) is the number of universal literals in the clause \(C\).
Let us compute the propositional skolemization of:

$$\forall a \exists b \forall c \exists d. (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor \neg b)$$

$$\exists b \exists d \forall a \forall c. (\neg a \lor c \lor d(a, b)) \land (\neg b(a) \lor \neg d(a, b)) \land (a \lor b(a) \lor \neg d(a, b)) \land (\neg a \lor \neg b(a))$$

$$\neg b_0 \lor \neg d_{00} \land (\neg b_0 \lor \neg d_{01}) \land (\neg b_1 \lor \neg d_{10}) \land (\neg b_1 \lor \neg d_{11}) \land (b_0 \lor \neg d_{00}) \land (b_0 \lor \neg d_{01}) \land (\neg b_1)$$

Key property:

$$\forall a \exists b \forall c \exists d. (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor \neg b)$$

TRUE iff SAT
Let us compute the propositional skolemization of:

\[ \forall a \exists b \forall c \exists d. \ (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor \neg b) \]

Problem:
- Exponential number of clauses/variables; unaffordable for real-world problems.

Two-step solution:
- **Factorization**: group together clauses originating from the same QBF clause.
- **Symbolic representation**: use BDDs to compactly represent factorized clauses.
Let us compute the propositional skolemization of:

\[ \forall a \exists b \forall c \exists d. (\neg a \lor c \lor d) \land (\neg b \lor \neg d) \land (a \lor b \lor \neg d) \land (\neg a \lor \neg b) \]

We obtain a set of **factorized clauses**. Each factorized clause \( \Gamma_I \) is made up of:

- A set \( \Gamma \) of literals: **shared** by all the originating clauses, associated to skolem terms.
- A set of indexes \( I \subseteq \{0, 1\}^{\delta(\Gamma)} \): each element in the set identifies one clause.
Propositional skolemization

Our clauses becomes **symbolic clauses** as soon as we represent index sets via BDDs:

\[ \forall a \exists b \forall c \exists d. (-a \lor c \lor d) \land (-b \lor -d) \land (a \lor b \lor -d) \land (-a \lor -b) \]

\[ [d]_{10} \land [-b, -d]_{00, 01, 10, 11} \land [b, -d]_{00, 01} \land [-b]_{1} \]

- **Support set** labeled by **universal variables**
- **Forest of ROBDDs** (*Reduced Ordered Binary Decision Diagrams*) with complemented arcs
- **Sharing** of structural information
- Overall size **linear** in the size of the original QBF
- A **symbolic** formula is just a **compact representation** for the propositional skolemization, whose **semantics** is fully inherited.
**Key point:** symbolic formulas enable to perform **symbolic inferences**.

- **Symbolic inferences** are directly performed on the symbolic representation **without** ever expanding its (possibly exponentially larger) propositional meaning.
- Each symbolic inference corresponds to the execution of many (possibly exponentially many) **propositional inferences**. Example: *Symbolic resolution*, \( e \in \Gamma, \neg e \in \Lambda, \delta(e) = \delta(\Gamma) = \delta(\Lambda) \)

\[
\frac{\Gamma, \{e\} \cup \Lambda, \{\neg e\}}{(\Gamma \setminus \{e\}) \cap \Lambda, \{\neg e\}) \cap J, \quad \Gamma_{\cap J}, \quad \Lambda_{\delta e J}
\]

- The "existential" part of the representation is managed like in the standard propositional case.
- The computational burden is on "universal" reasoning: Intersection, complementation, merge of index sets are performed via BDDs primitives.
On top of symbolic **subsumption**, **resolution** and **substitution**, we designed:

- **SUCP** (Symbolic Unit Clause Propagation): Assign until fixpoint every $[e]_I \in F$

- **SPLE** (Symbolic Pure Literal Elimination): Assign every symbolic literal $[e]_{I^+ \cap \neg F}$ and $[-e]_{I^- \cap \neg F}$ where $I^+ = \bigcup_{e \in \Gamma, \Gamma_I} I$ and $I^- = \bigcup_{-e \in \Gamma, \Gamma_I} I$

- **SSUB** (Symbolic Subsumption): remove every clause $\Gamma_I$ such that $\exists \Lambda \in F, \Lambda \subseteq \Gamma, I \subseteq J$

- **SHBR** (Symbolic Hyper Binary Resolution): Assign every failed symbolic literal $[-a]_{\cap \Lambda I}$ derived by a chain of binary resolution steps $[a] \xrightarrow{I_1} [a_1] \xrightarrow{I_2} \cdots \xrightarrow{I_n} [-a]$

- **SER** (Symbolic Equivalence Reasoning): Extract non-empty strongly connected components $\{a, a_1, \ldots, a_n\}_I$ out of the binary symbolic implication graph for the formula and apply the resulting symbolic substitution $\{[a/a_1]_I, \ldots, [a/a_n]_I\}$

- **SDR** (Symbolic Directional Resolution): Replace the set of symbolic clauses containing $e$ with the set of all their symbolic resolvents over $e$. 
Each strategy (resolution, SAT-based, branching, ...) has strengths and weaknesses;

Attempt to exploit each one at its best:

- Real mix of inference styles (not simply a sequential/parallel application);
- The emerging inference policy tunes automatically on the instance at hand.

Three representation spaces for QBFs

Interconnected by two satisfiability-preserving transformations
Each strategy (resolution, SAT-based, branching, ...) has strengths and weaknesses;

Attempt to exploit each one at its best:

- Real mix of inference styles (not simply a sequential/parallel application);
- The emerging inference policy tunes automatically on the instance at hand.

**Ground QBF reasoning** as a pre-processing step

DPLL-like search-based evaluation (over compressed clause sets)

SAT-based reasoning is leveraged if (and as soon as) the instance becomes affordable

BDD-based symbolic solution reasoning in the skolem space

**Hybrid Evaluation Engine**

Ground QBF reasoning

symbolic skolemisation

Incomplete Symbolic Reasoning

Complete Symbolic Reasoning

Branching Reasoning

SAT-based CNF Reasoning

**Q**: Ground QBF Reasoning

**S**: Incomplete Symbolic Reasoning

**R**: Complete Symbolic Reasoning

**B**: Branching Reasoning

**G**: SAT-based CNF Reasoning

- QuBOS, S-QBF

- QuBe, quaffle... (ZQSAT)

- unique to sKizzo

- real mix of inference styles (not simply a sequential/parallel application)

- the emerging inference policy tunes automatically on the instance at hand
Each strategy (resolution, SAT-based, branching, ...) has strengths and weaknesses;

Attempt to exploit each one at its best:

- Real mix of inference styles (not simply a sequential/parallel application);
- The emerging inference policy tunes automatically on the instance at hand.

1. QTree reconstruction
2. Unit clause propagation
3. Pure literal elimination
4. Forall reduction
5. Variable elimination by q-resolution
6. Symbolic UCP
7. Symbolic PLE
8. Subsumption check
9. Symb. hyper binary resolution
10. Symb. equivalence reasoning
11. Branching reasoning
12. Symb. Directional Resolution
13. Expansion-to-SAT, followed by SAT-based reasoning
Each strategy (resolution, SAT-based, branching, ...) has strengths and weaknesses;

Attempt to exploit each one at its best:

- Real **mix** of inference styles (not simply a sequential/parallel application);
- The emerging inference policy **tunes** automatically on the instance at hand.

The **efficiency** of each inference rule is **dinamically assessed** and computational resources are accordingly distributed.

**Directional Resolution** is applied in a **controlled environment**: memory consumption is monitored, transactional commit/rollback points are inserted, etc.

Branching reasoning uses symbolic and SAT-based reasoning as forms of **look-ahead**. The optimal B/G switch size is estimated.
Each strategy (resolution, SAT-based, branching, ...) has strengths and weaknesses;

Attempt to exploit each one at *its* best:
- Real **mix** of inference styles (not simply a sequential/parallel application);
- The emerging inference policy **tunes** automatically on the instance at hand.
**SAT-certificate** $C(F)$: piece of information whose “cheap” verification against $F$ provides **solver-independent** evidence of satisfiability.

- Natural candidate: a (representation of some) **model**

\[
\forall a \forall b \exists c \forall d \exists e \exists f \quad \neg b \lor e \lor f \land (a \lor c \lor f) \land (a \lor d \lor e) \land \neg a \lor d \lor \neg e \land (a \lor e \lor \neg f) \land (\neg a \lor b \lor \neg d \lor e) \land (\neg a \lor b \lor c) \land (\neg a \lor c \lor \neg f) \land (a \lor d \lor \neg e)
\]

We need to provide the truth value of:
- $c$ as a function of $a, b$
- $e$ as a function of $a, b, d$
- $f$ as a function of $a, b, d$

I.e.: we need **interpretations** for a set of skolem functions.

- **Solver-independent**
- **Deduction unnecessary**
- **Quite informative!**
- **Not easy** to represent
- **Not polynom.** to verify
- **Not built** by QBF solvers
SAT-certificate $C(F)$: piece of information whose “cheap” verification against $F$ provides solver-independent evidence of satisfiability.

Natural candidate: a (representation of some) model

\[ \forall a \forall b \forall c \forall d \forall e \forall f \ (\neg b \lor e \lor f) \land (a \lor c \lor f) \land (a \lor d \lor e) \land (\neg a \lor d \lor \neg e) \land (a \lor e \lor \neg f) \land (\neg a \lor b \lor \neg d \land e) \land (\neg a \land b \land \neg c \land e) \land (\neg a \lor c \lor \neg f) \land (a \lor d \lor \neg e) \]

Two roots for each existential variable to represent $\{T,F\}^n \rightarrow \{T,F,DC\}$ funct.

\[
c(a,b) = \begin{cases} 
    T & \text{if } c^+(a,b) = 1 \\
    F & \text{if } c^-(a,b) = 1 \\
    DC & \text{otherwise}
\end{cases}
\]

A certificate is a forest of ROBDDs

Universal variables as support set
Sharing of structural information
Efficient BDD-based operations
**SAT-certificate** $C(F)$: piece of information whose “cheap” verification against $F$ provides *solver-independent* evidence of satisfiability.

Natural candidate: a (representation of some) model

$$\forall a \forall b \exists c \forall d \exists e \exists f \ (\neg b \lor e \lor f) \land (a \lor c \lor f) \land (a \lor d \lor e) \land (\neg a \lor d \lor \neg e) \land (a \lor \neg e \lor \neg f) \land (a \lor d \lor \neg e) \land (\neg a \lor b \lor \neg d \lor e) \land (\neg a \lor b \lor \neg c) \land (\neg a \lor \neg c \lor \neg f) \land (a \lor \neg d \lor \neg e)$$

Certificate as forest of ROBDDs

BDD operator: **COFACTOR**

BDDs are **canonic**: **EQUALITY CHECK**

BDD operator: **OR**

\[ [c^-(a) + f^-(a, b, d)]_{a=1} \equiv 1 \]
Extracting certificates

**Static view**

\[ \forall a \exists b \ (a \lor b) \land \ldots \]

QBF solver → Op1 → QBF certifier → Op2 → Op3 → Op4 → QBF verifier

**Dynamic view**

Inference log

\[ \forall a \forall b \forall c \forall d \exists e \exists f : (-b \lor e \lor f) \land (a \lor c \lor f) \land (a \lor d \lor e) \land (-a \lor b \lor -e) \land (-a \lor b \lor -c) \land (-a \lor -c \lor e) \land (a \lor -d \lor e) \land (-a \lor d \lor e) \land (-a \lor d \lor -f) \land (a \lor -e \lor -f) \]

is SAT!

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10

Certificate of satisfiability

Solver → Certifier

Marco Benedetti

Evaluating Quantified Boolean Formulas
Inference log

\[ \forall a \exists b (a \lor b) \land \ldots \]  
\[ \text{Op1} \]  
\[ \text{Op2} \]  
\[ \text{Op3} \]  
\[ \text{Op4} \]  
\[ \text{QBF} \text{ solver} \]  
\[ \text{QBF} \text{ certifier} \]  
\[ \text{QBF} \text{ verifier} \]  
\[ \text{yes/no} \]

\[ \forall a \forall b \exists c \exists d \exists e \exists f \ldots \text{is SAT!} \]

\[ (a \lor c \lor f) \land (a \lor d \lor e) \land \ldots \]

Dynamic view

Certificate of satisfiability

Static view

Extracting certificates
An example: TicTacToe as a QBF instance

Problem statement

Given the rules of “TicTacToe”, is it true that

- For all the possible moves of the player “X”,
- Exists a move of the player “O” such that
  - For all the possible moves of the player “X”,
    - ...
  - Exists a move of the player “O” such that
    - the player “X” does not win?

Alternation of for-all / exists quantifications.

Domain theory and desired property naturally stated in propositional logic

= QBF
An example: TicTacToe as a QBF instance

Problem statement

Given the rules of “TicTacToe”, is it true that

- For all the possible moves of the player “X”,
- Exists a move of the player “O” such that
  - For all the possible moves of the player “X”,
    - ...
  - Exists a move of the player “O” such that
    - the player “X” does not win?

To certify the validity of a “yes” answer we exhibit a strategy as a witness

This certificate is a non-loosing strategy for O: O’s moves as a function of X’s ones

Declarative specification + deductive approach: no game-tree search!
Overall Architecture and Usage

QBF encoding

∀\exists b
(a \lor b) \land ...

Problem:

\langle F, C(F) \rangle

true/false

symbols

1: O(1,1)
2: X(1,1)
3: O(2,1)
...

\langle F, C(F) \rangle

ok/no

QBF solver

QBM library

QBF certifier

QBF verifier

\log

certif.

\langle F, C(F) \rangle
Outline of the talk

1. Introduction to QBF Evaluation
   - Motivation, and examples
   - Existing approaches

2. Our approach (and sKizzo)
   - Quantifier Tree Reconstruction
   - Symbolic Skolemization
   - Hybrid Inference Engine
   - Certification and Strategies

3. Results
   - Benchmarks and Experiments
   - Direct and indirect feedback

4. Final Remarks
Experimental Results

Biere’s benchmarks

- Model checking problems
  - 64 instances / 4 families

Ayari’s benchmarks

- Protocol and circuit verification
  - 68 instances / 5 families
Experimental Results

QBF solver

QBF certifier

QBF verifier

\( \forall a \exists b (a \lor b) \land \ldots \)

Time

Size

vars/steps/nodes

secs

counter-re family

www.qbflib.org

\[
\begin{array}{c}
\text{cnt}_2re \\
\text{cnt}_3re \\
\text{cnt}_4re \\
\text{cnt}_5re \\
\text{cnt}_6re \\
\text{cnt}_7re \\
\text{cnt}_8re \\
\text{cnt}_9re
\end{array}
\]

\[
\begin{array}{c}
\text{#var} \\
\text{#steps} \\
\text{#nodes}
\end{array}
\]

\[
\begin{array}{c}
\text{Time} \\
\text{Size}
\end{array}
\]

\[
\begin{array}{c}
\text{vars/steps/nodes} \\
\text{secs}
\end{array}
\]

\[
\begin{array}{c}
\text{cnt}_2re \\
\text{cnt}_3re \\
\text{cnt}_4re \\
\text{cnt}_5re \\
\text{cnt}_6re \\
\text{cnt}_7re \\
\text{cnt}_8re \\
\text{cnt}_9re
\end{array}
\]

\[
\begin{array}{c}
\text{#var} \\
\text{#steps} \\
\text{#nodes}
\end{array}
\]

www.qbflib.org

Evaluating Quantified Boolean Formulas
Experimental Results

QBF solver

QBF certifier

QBF verifier

∀a∃b (a∨b)∧…

"k-d4-n" family
www.qbflib.org

Time

vars/steps/nodes

Size

secs

#var

#steps

#nodes

kd4_n6  kd4_n8  kd4_n9  kd4_n12  kd4_n13  kd4_n14  kd4_n15  kd4_n16

vars/steps/nodes

Time

1000

100

10

1

0,1

0,01

0,001

kd4_n6  kd4_n8  kd4_n9  kd4_n12  kd4_n13  kd4_n14  kd4_n15  kd4_n16

www.qbflib.org
### Experimental Results

#### Third QBF evaluation (2005)

**Solver-wise views: fixed struct. (514)**

<table>
<thead>
<tr>
<th>Solver</th>
<th>#</th>
<th>True</th>
<th>False</th>
<th>Sum</th>
<th>Min</th>
<th>Max</th>
<th>Median</th>
<th>IQ-Rng</th>
</tr>
</thead>
<tbody>
<tr>
<td>sKizzo_v0.5</td>
<td>304</td>
<td>132</td>
<td>172</td>
<td>4257.40</td>
<td>0.00</td>
<td>567.47</td>
<td>0.43</td>
<td>0.12</td>
</tr>
<tr>
<td>quantor</td>
<td>287</td>
<td>132</td>
<td>155</td>
<td>1192.18</td>
<td>0.00</td>
<td>222.42</td>
<td>0.09</td>
<td>0.03</td>
</tr>
<tr>
<td>sKizzo_v0.4</td>
<td>281</td>
<td>118</td>
<td>163</td>
<td>3031.03</td>
<td>0.00</td>
<td>499.10</td>
<td>0.32</td>
<td>0.09</td>
</tr>
<tr>
<td>run-semprop</td>
<td>260</td>
<td>113</td>
<td>147</td>
<td>6660.30</td>
<td>0.00</td>
<td>546.89</td>
<td>0.37</td>
<td>0.11</td>
</tr>
<tr>
<td>QChaffLearn</td>
<td>234</td>
<td>90</td>
<td>144</td>
<td>4481.90</td>
<td>0.00</td>
<td>532.46</td>
<td>0.39</td>
<td>0.13</td>
</tr>
<tr>
<td>yquaffle</td>
<td>220</td>
<td>83</td>
<td>137</td>
<td>3613.92</td>
<td>0.01</td>
<td>371.89</td>
<td>0.38</td>
<td>0.16</td>
</tr>
<tr>
<td>run-ssolve</td>
<td>219</td>
<td>89</td>
<td>130</td>
<td>3579.60</td>
<td>0.00</td>
<td>565.40</td>
<td>0.32</td>
<td>0.09</td>
</tr>
<tr>
<td>GRL</td>
<td>217</td>
<td>84</td>
<td>133</td>
<td>5515.57</td>
<td>0.00</td>
<td>528.69</td>
<td>0.49</td>
<td>0.20</td>
</tr>
<tr>
<td>QMRes</td>
<td>201</td>
<td>107</td>
<td>94</td>
<td>3405.89</td>
<td>0.00</td>
<td>340.01</td>
<td>0.98</td>
<td>0.41</td>
</tr>
<tr>
<td>qbfIHR</td>
<td>191</td>
<td>83</td>
<td>108</td>
<td>4871.80</td>
<td>0.00</td>
<td>507.54</td>
<td>0.12</td>
<td>0.05</td>
</tr>
<tr>
<td>run-openqbf</td>
<td>184</td>
<td>72</td>
<td>112</td>
<td>10156.39</td>
<td>0.15</td>
<td>594.29</td>
<td>3.40</td>
<td>0.92</td>
</tr>
<tr>
<td>WalkQSAT</td>
<td>169</td>
<td>86</td>
<td>83</td>
<td>3035.61</td>
<td>0.00</td>
<td>521.08</td>
<td>0.27</td>
<td>0.11</td>
</tr>
<tr>
<td>qbfbdd</td>
<td>91</td>
<td>42</td>
<td>49</td>
<td>3223.87</td>
<td>0.00</td>
<td>478.23</td>
<td>0.52</td>
<td>0.16</td>
</tr>
</tbody>
</table>
“...Those are very impressive results!! No other general QBF solver can come close. I'm very impressed with how far you pushed the solver.”

“I don’t know what skizzo is doing, but it is doing that thing really fast! By the way, could you fix this silly bug? :)

“on my own QBF problem instances, skizzo runs much faster than quantor, ssolve, semprop, and so forth... thanks a lot for the great QBF solver!”

“[on our instances] semprop is consistently worse than skizzo. skizzo is very good at find solutions when the instances are SAT [...] quantor was curiously better than skizzo on some UNSAT instances. But these instances are really complex. Some planners really have problem solving them!”

Bart Selman
[Cornell University]
Instances: two-player games

Stefan Woltran
[Vienna Univ. Of Technology]
Instances: hand crafted

Hiroaki Yoshida
[Tokyo University]
Instances: logic synthesis

Hector V. Palacios
(joint work with H. Geffner)
[UPF, Barcelona]
Instances: conformant planning
## Experimental Results

Presented at **QCSP05** (ws at **CP05**)

courtesy of Palacios/Geffner

### Comparing with qbf solvers

<table>
<thead>
<tr>
<th>problem</th>
<th>$2^{TV}$</th>
<th>SAT</th>
<th>our</th>
<th>quizze 0.8</th>
<th>quantor</th>
<th>qe/prop</th>
<th>qbf vars</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube-11-par</td>
<td>13</td>
<td>unsat</td>
<td>181.5</td>
<td>864.9</td>
<td></td>
<td></td>
<td>1331</td>
<td>51076</td>
</tr>
<tr>
<td>cube-11-par</td>
<td>14</td>
<td>sat</td>
<td>&gt; 2 Gb</td>
<td>&gt; 2 Gb</td>
<td></td>
<td></td>
<td>1430</td>
<td>51626</td>
</tr>
<tr>
<td>sq-center-3-ser</td>
<td>19</td>
<td>unsat</td>
<td>207.4</td>
<td>958</td>
<td>6973</td>
<td></td>
<td>934</td>
<td>6023</td>
</tr>
<tr>
<td>sq-center-3-ser</td>
<td>20</td>
<td>sat</td>
<td>18.8</td>
<td>3</td>
<td>2632</td>
<td>&gt; 9000</td>
<td>962</td>
<td>6266</td>
</tr>
<tr>
<td>sq-center-4-par</td>
<td>21</td>
<td>unsat</td>
<td>1181</td>
<td>&gt; 9702</td>
<td></td>
<td></td>
<td>2058</td>
<td>19681</td>
</tr>
<tr>
<td>sq-center-4-par</td>
<td>22</td>
<td>sat</td>
<td>243</td>
<td>&gt; 8400</td>
<td></td>
<td></td>
<td>2152</td>
<td>20228</td>
</tr>
<tr>
<td>ring-6</td>
<td>16</td>
<td>unsat</td>
<td>0.39</td>
<td>&gt; 6000</td>
<td>5375</td>
<td></td>
<td>767</td>
<td>81826</td>
</tr>
<tr>
<td>ring-6</td>
<td>17</td>
<td>sat</td>
<td>0.8</td>
<td>&gt; 6000</td>
<td>&gt; 6240</td>
<td></td>
<td>813</td>
<td>82019</td>
</tr>
<tr>
<td>sort-6-ser</td>
<td>11</td>
<td>unsat</td>
<td>&gt; 7200</td>
<td>&gt; 60000</td>
<td>&gt; 60000</td>
<td></td>
<td>753</td>
<td>4462</td>
</tr>
<tr>
<td>sort-6-ser</td>
<td>12</td>
<td>sat</td>
<td>40</td>
<td>1.23</td>
<td>&gt; 60000</td>
<td>&gt; 60000</td>
<td>819</td>
<td>4745</td>
</tr>
<tr>
<td>sort-7-ser</td>
<td>15</td>
<td>unsat</td>
<td>&gt; 7200</td>
<td>&gt; 60000</td>
<td></td>
<td></td>
<td>1400</td>
<td>10430</td>
</tr>
<tr>
<td>sort-7-ser</td>
<td>16</td>
<td>sat</td>
<td>3035.6</td>
<td>12.5</td>
<td></td>
<td></td>
<td>1491</td>
<td>10886</td>
</tr>
<tr>
<td>sort-8-par</td>
<td>5</td>
<td>unsat</td>
<td>&gt; 7200</td>
<td>18.8</td>
<td></td>
<td></td>
<td>644</td>
<td>11688</td>
</tr>
<tr>
<td>sort-8-par</td>
<td>6</td>
<td>sat</td>
<td>4256</td>
<td>26</td>
<td></td>
<td></td>
<td>764</td>
<td>21181</td>
</tr>
</tbody>
</table>

Héctor Palacios, 09/2005

---

Evaluating Quantified Boolean Formulas
Outline of the talk

1. Introduction to QBF Evaluation
   - Motivation, and examples
   - Existing approaches

2. Our approach (and sKizzo)
   - Quantifier Tree Reconstruction
   - Symbolic Skolemization
   - Hybrid Inference Engine
   - Certification and Strategies

3. Results
   - Benchmarks and Experiments
   - Direct and indirect feedback

4. Final Remarks
Final remarks

Four principles, four keywords:

- **Extract and exploit structure:** Q-tree Reconstruction
- **Treat ∀ quantifiers properly:** Symbolic Skolemization
- **Be flexible while solving:** Hybrid Inference Engine
- **Construct evidences of truth:** Model Extraction

...showing that:

- The structure is really there, and is **expansible**
- Search-based QBF solver can be **defeated**!
- Hybrid approaches **pay**
- QBF certification is **not impractical**

What all these improvements aim to?

- To have QBF-based techniques **solving** real-world problems
Future work

- Start working on the encoding side of the problem (MC and more)
- **Major improvements** in the implementation due soon
- Theory on three **new inference strategies** under development

References

- “Evaluating QBFs via Symbolic Skolemization” [Proc. of LPAR04]
- “Extracting Certificates from QBFs” [Proc. of IJCAI05]
- “Quantifier Trees for QBFs” [Proc. of SAT05]
- “sKizzo: A Suite to Evaluate and Certify QBFs” [Proc. of CADE05]
- “Hybrid Evaluation Procedures for QBF” [Proc. of RCRA05]
- “Experimenting with QBF-based Formal verification” [Proc. of CFV05]

**Web site** with more info, experiments and downloadable tools: 
http://sra.itc.it/people/benedetti/sKizzo