Efficient Extraction of Skolem Functions from QRAT Proofs

Marijn J.H. Heule

Joint work with
Martina Seidl and Armin Biere

FMCAD, October 23, 2014
Introduction and Challenges

From Clausal Proofs to Skolem Functions

Running Example

Validating Skolem Functions

Experimental Results

Conclusions
Introduction to QBF

A quantified Boolean formula (QBF) is a propositional formula where variables are existentially (∃) or universally (∀) quantified.

Consider the formula $\forall a \exists b, c. (a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c)$

A model is:

Consider the formula $\exists b \forall a \exists c. (a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c)$

A counter-model is:
Introduction to Skolem functions for QBF

A Skolem function \( f_x(U_x) \) for a QBF formula \( \pi.\psi \) defines the truth value of an existential variable \( x \) based on the set \( U_x \) of universal variables that occur earlier in the prefix than \( x \).

Consider the formula \( \forall a \exists b, c.(a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c) \)

A model is:

![Model Diagram]

The set of Skolem functions \( F \) (defining all existentials) is

\[
F = \{ f_b(a) = \neg a, f_c(a) = a \}
\]

The set of Skolem functions can be much smaller than a model.
Challenges for Quantified Boolean Formulas (QBF)

Preprocessing is crucial to solve most QBF instances efficiently. Proofs are useful for applications and to validate solver output. Main challenges regarding QBF and preprocessing [Janota’13]:

1. produce proofs that can be validated in polynomial time;
2. develop methods to validate all QBF preprocessing; and
3. narrow the performance gap between solving with and without proof generation.

In our IJCAR’14 paper [1], we meet all three challenges!


Here we show how to make Skolem functions out of the proofs.
From Clausal Proofs to Skolem Functions
Clausal Proof System

Learn: add a clause
* Preserve satisfiability

Satisfiable
* Forget last clause

Unsatisfiable
* Learn empty clause

Forget: remove a clause
* Preserve unsatisfiability

init → \( \pi, \psi \)
Informal definitions of the redundancy concepts in the QRAT proof system. They can be computed in polynomial time.

Definition (Asymmetric Tautologies (AT))
An asymmetric tautology is a clause that becomes a tautology after adding “hidden literals”. ATs are logically implied by a formula.

Definition (Quantified Resolution AT (QRAT))
A quantified resolution AT is a clause that contains a literal for which all “outer resolvents” are ATs.

Definition (Extended Universal Reduction (EUR))
A universal literal is redundant if assigning it to false cannot influence the value of universal literals.
## Rules of the QRAT Proof System

<table>
<thead>
<tr>
<th>Rule</th>
<th>Preconditions</th>
<th>Postconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N1)</td>
<td>$\pi.\psi$</td>
<td>$C$ is an asymmetric tautology</td>
</tr>
<tr>
<td></td>
<td>$\pi.\psi \setminus {C}$</td>
<td></td>
</tr>
<tr>
<td>(N2)</td>
<td>$\pi.\psi$</td>
<td>$C$ is an asymmetric tautology</td>
</tr>
<tr>
<td></td>
<td>$\pi'.\psi \cup {C}$</td>
<td></td>
</tr>
<tr>
<td>(E1)</td>
<td>$\pi.\psi$</td>
<td>$C \in \psi$, $Q(\pi, l) = \exists$</td>
</tr>
<tr>
<td></td>
<td>$\pi.\psi \setminus {C}$</td>
<td>$C$ has QRAT on $l$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(E2)</td>
<td>$\pi.\psi$</td>
<td>$C \not\in \psi$, $Q(\pi, l) = \exists$</td>
</tr>
<tr>
<td></td>
<td>$\pi'.\psi \cup {C}$</td>
<td>$C$ has QRAT on $l$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(U1)</td>
<td>$\pi.\psi \cup {C}$</td>
<td>$l \in C$, $Q(\pi, l) = \forall$, $\neg l \not\in C$, $C$ has QRAT on $l$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(U2)</td>
<td>$\pi.\psi \cup {C}$</td>
<td>$l \in C$, $Q(\pi, l) = \forall$, $\neg l \not\in C$, $C$ has EUR on $l$ w.r.t. $\psi$</td>
</tr>
</tbody>
</table>
## Rules of the QRAT Proof System

<table>
<thead>
<tr>
<th>Rule</th>
<th>Preconditions</th>
<th>Postconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N1)</td>
<td>$\pi.\psi$</td>
<td>$\pi.\psi \setminus {C}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C$ is an asymmetric tautology</td>
</tr>
<tr>
<td>(N2)</td>
<td>$\pi.\psi$</td>
<td>$\pi' . \psi \cup {C}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C$ is an asymmetric tautology</td>
</tr>
<tr>
<td>(E1)</td>
<td>$\pi . \psi$</td>
<td>$\pi.\psi \setminus {C}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C \in \psi$, $Q(\pi)$, $C$ has QRAT on $\pi$ w.r.t. $\psi$</td>
</tr>
<tr>
<td>(E2)</td>
<td>$\pi . \psi$</td>
<td>$\pi' . \psi \cup {C}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$C \not\in \psi$, $C$ has $\psi$</td>
</tr>
<tr>
<td>(U1)</td>
<td>$\pi . \psi \cup {C}$</td>
<td>$I \in C$, $Q(\pi)$, $C$ has $\psi$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I \in C$, $Q(\pi)$, $C$ has $\psi$</td>
</tr>
<tr>
<td>(U2)</td>
<td>$\pi . \psi \cup {C}$</td>
<td>$I \in C$, $Q(\pi)$, $C$ has $\psi$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$I \in C$, $Q(\pi)$, $C$ has $\psi$</td>
</tr>
</tbody>
</table>
ComputeSkolem (prefix π, QRAT proof P)

1. let ψ be an empty formula
2. foreach existential variable e do $f_e(U) := \ast$  
   // initialize $F$
3. while (P is not empty) do
4.   $\langle \text{rule } R, \text{ clause } C, \text{ literal } l \rangle := P.pop()$
5.   if ($R = E1$) then
6.     let $e$ be $\text{var}(l)$
7.     $f_e(U) := \text{IfThenElse}(F(\text{OF}(\pi, \psi, l))), \text{polarity}(l), f_e(U))$
8.   if ($R = E1$ or $R = N1$) then  
   // Forget rules
9.     $\psi := \psi \cup \{C\}$
10.  if ($R = E2$ or $R = N2$) then  
11.    $\psi := \psi \setminus \{C\}$

// Learn rules
Adding a Skolem Function

The outer clause of $D$ w.r.t. a literal $l$ under prefix $\pi$ is:

$$OC(\pi, D, l) := \{ k | k \in D, \pi(k) \leq \pi(l), \text{ and } k \neq l \}$$

The outer formula of $\psi$ w.r.t. a literal $l$ under prefix $\pi$ is:

$$OF(\pi, \psi, l) := \{ OC(\pi, D, \neg l) | D \in \psi, \neg l \in D \}$$

How to understand

$$f_e(U) := \text{IfThenElse}(F(OF(\pi, \psi, l))), \text{polarity}(l), f_e(U))$$

If a clause $C$ has QRAT on literal $l \in C$ w.r.t. $\psi$, then

- any assignment that falsifies $OF(\pi, \psi, l)$ satisfies $C$
- if $OF(\pi, \psi, l)$ is satisfied, we can safely assign $l$ to true
Running Example
Running Example

Consider again \( \pi.\psi := \forall a \exists b, c.(a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c) \)

QRAT proof \( P \) using the rules E1 (Forget) and E2 (Learn):

\[ E2(\neg a \lor \neg b), \ E1(\neg a \lor c), \ E1(\neg b \lor \neg c), \ E1(\neg a \lor \neg b), \ E1(a \lor b) \]

<table>
<thead>
<tr>
<th>Rule</th>
<th>( \psi )</th>
<th>( O\mathcal{F}(\pi, \psi, I) )</th>
<th>Skolem set ( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>( \emptyset )</td>
<td>( n \setminus a )</td>
<td>( f_b(a) = *, \ f_c(a) = * )</td>
</tr>
</tbody>
</table>
Running Example

Consider again \( \pi \cdot \psi := \forall a \exists b, c. (a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c) \)

QRAT proof \( P \) using the rules E1 (Forget) and E2 (Learn):

\[
\begin{align*}
E2(\neg a \lor \neg b), & \quad E1(\neg a \lor c), \quad E1(\neg b \lor \neg c), \quad E1(\neg a \lor \neg b), \quad E1(a \lor b) \\
\text{Rule} & \quad \psi & \quad OF(\pi, \psi, l) & \quad \text{Skolem set } F \\
\hline
init & \emptyset & n \setminus a & f_b(a) = *, \ f_c(a) = * \\
E1(a \lor b) & \emptyset & \emptyset & f_b(a) = \top, \ f_c(a) = *
\end{align*}
\]
Running Example

Consider again $\pi.\psi := \forall a \exists b, c. (a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c)$

QRAT proof $P$ using the rules $E1$ (Forget) and $E2$ (Learn):

$E2(\neg a \lor \neg b), E1(\neg a \lor c), E1(\neg b \lor \neg c), E1(\neg a \lor \neg b), E1(a \lor b)$

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\psi$</th>
<th>$OF(\pi, \psi, l)$</th>
<th>Skolem set $F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>$\emptyset$</td>
<td>$n \setminus a$</td>
<td>$f_b(a) = \ast, f_c(a) = \ast$</td>
</tr>
<tr>
<td>$E1(a \lor b)$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$f_b(a) = \top, f_c(a) = \ast$</td>
</tr>
<tr>
<td>$E1(\neg a \lor \neg b)$</td>
<td>$(a \lor b)$</td>
<td>$(a)$</td>
<td>$f_b(a) = \neg a, f_c(a) = \ast$</td>
</tr>
</tbody>
</table>
Running Example

Consider again $\pi.\psi := \forall a \exists b, c.(a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c)$

QRAT proof $P$ using the rules $E1$ (Forget) and $E2$ (Learn):

$E2(\neg a \lor \neg b), E1(\neg a \lor c), E1(\neg b \lor \neg c), E1(\neg a \lor \neg b), E1(a \lor b)$

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\psi$</th>
<th>$\mathcal{OF}(\pi, \psi, I)$</th>
<th>Skolem set $F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>$\emptyset$</td>
<td>$n \setminus a$</td>
<td>$f_b(a) = \star$, $f_c(a) = \star$</td>
</tr>
<tr>
<td>$E1(a \lor b)$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$f_b(a) = \top$, $f_c(a) = \star$</td>
</tr>
<tr>
<td>$E1(\neg a \lor \neg b)$</td>
<td>$(a \lor b)$</td>
<td>$(a)$</td>
<td>$f_b(a) = \neg a$, $f_c(a) = \star$</td>
</tr>
<tr>
<td>$E1(\neg b \lor \neg c)$</td>
<td>$(a \lor b)$ $\land$ $(\neg a \lor \neg b)$</td>
<td>$\emptyset$</td>
<td>$f_b(a) = \neg a$, $f_c(a) = \bot$</td>
</tr>
</tbody>
</table>
Running Example

Consider again $\pi.\psi := \forall a \exists b, c. (a \lor b) \land (\neg a \lor c) \land (\neg b \lor \neg c)$

QRAT proof $P$ using the rules $E1$ (Forget) and $E2$ (Learn):

$E2(\neg a \lor \neg b), E1(\neg a \lor c), E1(\neg b \lor \neg c), E1(\neg a \lor \neg b), E1(a \lor b)$

<table>
<thead>
<tr>
<th>Rule</th>
<th>$\psi$</th>
<th>$OF(\pi, \psi, l)$</th>
<th>Skolem set $F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>$\emptyset$</td>
<td>$n \setminus a$</td>
<td>$f_b(a) = *, f_c(a) = *$</td>
</tr>
<tr>
<td>$E1(a \lor b)$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$f_b(a) = \top, f_c(a) = *$</td>
</tr>
<tr>
<td>$E1(\neg a \lor \neg b)$</td>
<td>$(a \lor b)$</td>
<td>$(a)$</td>
<td>$f_b(a) = \neg a, f_c(a) = *$</td>
</tr>
<tr>
<td>$E1(\neg b \lor \neg c)$</td>
<td>$(a \lor b) \land (\neg a \lor \neg b)$</td>
<td>$\emptyset$</td>
<td>$f_b(a) = \neg a, f_c(a) = \bot$</td>
</tr>
<tr>
<td>$E1(\neg a \lor c)$</td>
<td>$(a \lor b) \land (\neg a \lor \neg b) \land (\neg b)$</td>
<td>$f_b(a) = \neg a, f_c(a) = \neg f_b(a)$</td>
<td></td>
</tr>
</tbody>
</table>
Validating Skolem Functions
Checks to Validate Skolem Functions

Two tests are required to validate Skolem functions:

1. Can we falsify a clause in formula $\psi$ while satisfying the Skolem functions $F(U)$?

   \[
   \text{solve}(\neg \psi \land F(U)) = \text{UNSAT}?
   \]

2. Check that all Skolem functions depend only on universal variables that occur earlier in the prefix.

   Problem: our method could create a Skolem function

   \[
   f_x(U_x) := f_y(U_y) \text{ with } \pi(x) < \pi(y)
   \]

   Solution: convert Skolem functions to And-Inverter-Graphs (AIGs) and check for reachability.
Check Reachability in AIGs

Consider the formula $\pi.\psi$:

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

Skolem functions for $\pi.\psi$:

Our algorithm could have produced $f_b(a) := f_d(a, c)$, but that is not problematic because $f_d(a, c)$ does not depend on $c$.

How to simplify the circuit and preserve the dependencies?
Experimental Results
Experimental Results: Solving versus Extraction

We used the benchmarks of QBF Eval 2012 as the test set.

First, we compare the costs of solving true QBF formulas and the costs to extract Skolem functions from the proofs

- Extraction of Skolem functions includes proof validation

Summary of the results of the first experiment:

- Extraction costs of Skolem functions is comparable to solving time. The theoretical worst-case is polynomial.
- The size of the set of Skolem functions is linear in the solving time: a few megabyte (AAG format) per second.
- Validating the Skolem functions is comparable to the extraction time, but can be an order of magnitude slower.
## Experimental Results: Comparison with other Tools

<table>
<thead>
<tr>
<th>solver</th>
<th>sol-#</th>
<th>sol-t</th>
<th>ch-#</th>
<th>ch-t</th>
<th>cer-s</th>
</tr>
</thead>
<tbody>
<tr>
<td>bloqqer+QRAT</td>
<td>32</td>
<td>1</td>
<td>32</td>
<td>47</td>
<td>1851</td>
</tr>
<tr>
<td>bloqqer+RES</td>
<td>22</td>
<td>1</td>
<td>22</td>
<td>1</td>
<td>861</td>
</tr>
<tr>
<td>bloqqer+RES+depQBF</td>
<td>28</td>
<td>113</td>
<td>27</td>
<td>13</td>
<td>1040</td>
</tr>
<tr>
<td>depQBF</td>
<td>2</td>
<td>843</td>
<td>2</td>
<td>1</td>
<td>224</td>
</tr>
<tr>
<td>ebdd</td>
<td>15</td>
<td>491</td>
<td>7</td>
<td>118</td>
<td>409479</td>
</tr>
<tr>
<td>squolem</td>
<td>16</td>
<td>465</td>
<td>16</td>
<td>2</td>
<td>382</td>
</tr>
<tr>
<td>sKizzo</td>
<td>23</td>
<td>275</td>
<td>23</td>
<td>1</td>
<td>108750</td>
</tr>
</tbody>
</table>

- **sol-#**: # solved formulas
- **sol-t**: avg. solving time (s)
- **ch-#**: checked certificates
- **ch-t**: avg. checking time (s)
- **cer-s**: avg. certificate size (kilobyte)
Experimental Results: Size Comparison

Above the diagonal: Skolem functions from QRAT proofs are smaller
Conclusions
Conclusions

Compute Skolem functions out of QRAT proofs:
▶ All QBF preprocessing techniques can be stated in QRAT
▶ The proof size is polynomial in solving time (worst-case)
▶ We showed how to convert QRAT into Skolem functions
▶ The size of Skolem functions is relatively small: Linear in the size of proofs in practice, polynomial in worst-case

Directions for future work:
▶ How to state all QBF solving techniques in QRAT?
  ▶ That would allow Skolem functions for the full QBF tool chain
▶ Shrink Skolem functions using circuit simplification
  ▶ There are strong circuit simplification tools around, e.g. ABC
Conclusions

Compute Skolem functions out of QRAT proofs:
▶ All QBF preprocessing techniques can be stated in QRAT
▶ The proof size is polynomial in solving time (worst-case)
▶ We showed how to convert QRAT into Skolem functions
▶ The size of Skolem functions is relatively small: Linear in the size of proofs in practice, polynomial in worst-case

Directions for future work:
▶ How to state all QBF solving techniques in QRAT?
  ▶ That would allow Skolem functions for the full QBF tool chain
▶ Shrink Skolem functions using circuit simplification
  ▶ There are strong circuit simplification tools around, e.g. ABC

Thanks!