Evaluation of QBF: An Applications Perspective

Jussi Rintanen
Albert-Ludwigs-Universität Freiburg
Germany
Motivation for my research on QBF

SAT has many very strong applications:

- Satisfiability planning (Kautz & Selman, 1992, 1996) is a leading approach to AI planning.
- Bounded model-checking (BMC) with LTL (Biere et al. 1999) is a leading approach to formal verification (processors, ...)

Generalizations and more compact representations of these problems require quantifiers $\forall$ and $\exists$ and hence QBF.
Applications of QBF in AI planning

Satisfiability Planning (Kautz and Selman, 1992, 1996)

1. Deterministic planning for poly-length plans is NP-complete.

2. Deterministic planning in general is PSPACE-complete.

3. Nondeterministic planning (without observability) for poly-length plans is $\Sigma^P_2$-complete.

(Same problem outside planning: identifying reset/homing/synchronization sequences)
Satisfiability planning

Let $I$ be a formula describing the initial state (only one).
Let $G$ be a formula describing the goal states.

Plans of length $n$ are encoded as

$$I^0 \land R(P^0, P^1) \land R(P^1, P^2) \land \cdots \land R(P^{n-1}, P^n) \land G^n.$$ 

satisfying assignment = plan of length $n$
Logarithmic encoding for plan existence in QBF

A compact encoding for question: Is there a plan of length $2^n$?

\[
\begin{align*}
\text{reach}_0(S, S') & \overset{\text{def}}{=} \mathcal{R}(S, S') \\
\text{reach}_{i+1}(S, S') & \overset{\text{def}}{=} \exists T \forall b \exists T_1 \exists T_2 (\text{reach}_i(T_1, T_2) \\
& \quad \wedge (b \rightarrow (T_1 = S \land T_2 = T)) \\
& \quad \wedge (\neg b \rightarrow (T_1 = T \land T_2 = S')))
\end{align*}
\]

\[
\exists S^0 \exists S^1 (\text{reach}_n(S^0, S^1) \land I^0 \land G^1)
\]
Planning with nondeterminism in QBF

There is a sequence of operators so that every execution starting in an initial state reaches a goal state.

\[
\begin{align*}
\exists o_1 \cdots o_m \cdots o_1 & \cdots o_n \\
\forall p_1^0 \cdots p_n^0 a_1^{\sigma_1} & \cdots a_1^{n} \cdots a_{\sigma_k}^{t} \\
\exists p_1^1 \cdots p_n^1 & \cdots p_1^t \cdots p_n^t \\
(I^0 \rightarrow (R(P^0, P^1, A^0) & \wedge \cdots \wedge R(P^{t-1}, P^t, A^{t-1}) \wedge G^t))
\end{align*}
\]

Sets \( A^i \) of variables \( a \) are for nondeterminism.

assignment for \( o_1^1 \cdots o_m^1 \cdots o_1^t \cdots o_n^t = \text{plan of length } n \)
Applications of QBF in bounded model-checking

Bounded Model-Checking (Biere et al., 1999)

Assuming a polynomial bound on number of time points:

1. Checking LTL properties is NP-complete.

2. Checking CTL* properties is PSPACE-complete, and $\Sigma^p_i$-complete or $\Pi^p_i$-complete for $i$ alternations of path quantifiers.

$\implies$ No efficient (= PTIME) translations to SAT exist for (2).

$\implies$ Efficient translations to QBF do exist.
Why PSPACE-hard? Path quantifiers $A$ and $E$!

Proof idea: The problem of determining the truth-value of quantified Boolean formulae can be reduced in polynomial time to the validity of CTL* formulae with occurrences of the modal operators $A$, $E$ and $X$ only.

$\forall$ and $\exists$ in QBF represented as path quantifiers $A$ and $E$.

Consequence: There is no polynomial time translation from CTL* into the propositional logic assuming that NP $\neq$ PSPACE.
Translation of CTL* BMC into QBF

Given the transition relation as formulae $\Gamma_t$, $t \geq 0$, translation of CTL* BMC for formulae in NNF into QBF is as follows.

1. $[p]_k^i := P_{p,i}$  $[-p]_k^i := \neg P_{p,i}$
2. $[f \lor g]_k^i := [f]_k^i \lor [g]_k^i$
3. $[f \land g]_k^i := [f]_k^i \land [g]_k^i$
4. $[X\phi]_k^i := \text{if } i < k \text{ then } [\phi]_{k}^{i+1} \text{ else } \bot$
5. $[A\phi]_k^i := \forall X \exists Y (\bigwedge_{j=i+1}^{n} \Gamma_j \land [\phi]_k^i)$ where $n = \min(i + \text{depth}(\phi), k)$, $X = \bigcup_{j=i+1}^{n} U_j$, and $Y = \bigcup_{j=i+1}^{n} D_j$.
6. $[E\phi]_k^i := \exists X \exists Y (\bigwedge_{j=i+1}^{n} \Gamma_j \land [\phi]_k^i)$ where $n = \min(i + \text{depth}(\phi), k)$, $X = \bigcup_{j=i+1}^{n} U_j$, and $Y = \bigcup_{j=i+1}^{n} D_j$. 
7. \[ [G\phi]^i_k := \bot \]
8. \[ [F\phi]^i_k := \bigvee_{j=i}^{k} [\phi]^j_k \]
9. \[ [\phi U\psi]^i_k := \bigvee_{j=1}^{k} ([\psi]^j_k \land \bigwedge_{n=i}^{j-1} [\phi]^n_k) \]
10. \[ [\phi R\psi]^i_k := \bigvee_{j=1}^{k} ([\phi]^j_k \land \bigwedge_{n=i}^{j} [\psi]^n_k) \]

Formula’s depth is the number of time points outside the scope of any path quantifier. **Example:** \( \text{depth}(q \lor A p) = 1 \) and \( \text{depth}(XX(p \land A(q \lor Gr))) = 3. \)

1. \( \text{depth}(p) = \text{depth}(\neg p) = 1 \)
2. \( \text{depth}(f \otimes g) = \max(\text{depth}(f), \text{depth}(g)) \)
3. \( \text{depth}(X\phi) = 1 + \text{depth}(\phi) \)
4. \( \text{depth}(A\phi) = \text{depth}(E\phi) = 0 \) and \( \infty \) for others
**Q-resolution (Kleine Büning et al. 1995)**

Let $\alpha_1$ be a clause with $\exists$-literal $y_l$ and let $\alpha_2$ be a clause with $\overline{y_l}$. Then the Q-resolvent $\alpha$ of $\alpha_1$ and $\alpha_2$ is obtained as follows.

1. Remove all occurrences of $y_l$ and $\overline{y_l}$ in $\alpha_1 \lor \alpha_2$.

2. If the resulting clause contains complementary literals then no resolvent exists. Otherwise the resolvent is the clause without occurrences of $\forall$-literals not preceding a $\exists$-literal.

An $\exists$-unit clause is a clause with exactly one $\exists$-literal.
Stronger algorithm for inferring unit clauses

Let $Y_1, \ldots, Y_N$ be the sets of universal variables in the prefix.

1. Let $C$ be our clauses (the body of the QBF).
2. Let $C' = \emptyset$ be an auxiliary clause set.
3. Choose a valuation $V : Y_1, \ldots, Y_N \mapsto \{T, F\}$.
4. Remove from $C'$ all clauses for which some universal literal is made true by $V$.
5. Perform unit Q-resolution with $C \cup C'$, adding the new clauses to $C'$.
6. Go back to 3.
PROCEDURE unit2(⟨Y₁, X₁, Y₂, X₂, . . . , Yₙ, Xₙ⟩, V, C, C')
IF n = 0 THEN RETURN C'∪Q-unit-resolvents(C' ∪ C);
REPEAT
  gotnew := false;
  FOR EACH valuation V' (as set of literals) of Y₁ DO
    C' := clauses in C' not made true by V ∪ V';
    C'' := unit2(⟨Y₂, X₂, . . . , Yₙ, Xₙ⟩, V ∪ V', C, C');
    IF there is clause in C''\C' without occurrences of vars in Y₁
      THEN gotnew := true;
    C' := C'' without clauses with occurrences of vars in Y₁;
  END
  UNTIL gotnew = false;
RETURN C';
## Improvement on structured QBF

<table>
<thead>
<tr>
<th>problem</th>
<th>steps</th>
<th>prefix</th>
<th>vars</th>
<th>clauses</th>
<th>runtime in seconds</th>
<th>DP/QBF</th>
<th>+LPAR’01</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks4</td>
<td>2</td>
<td>∃∀∃</td>
<td>149</td>
<td>1183</td>
<td>&gt; 1h</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>blocks4</td>
<td>4</td>
<td>∃∀∃</td>
<td>210</td>
<td>1505</td>
<td>&gt; 1h</td>
<td>3.20</td>
<td></td>
</tr>
<tr>
<td>blocks4</td>
<td>8</td>
<td>∃∀∃</td>
<td>271</td>
<td>1827</td>
<td>&gt; 1h</td>
<td>4.22</td>
<td></td>
</tr>
<tr>
<td>hanoi3</td>
<td>2</td>
<td>∃∀∃</td>
<td>709</td>
<td>16599</td>
<td>&gt; 1h</td>
<td>0.54</td>
<td></td>
</tr>
<tr>
<td>hanoi3</td>
<td>4</td>
<td>∃∀∃</td>
<td>962</td>
<td>17553</td>
<td>&gt; 1h</td>
<td>23.08</td>
<td></td>
</tr>
<tr>
<td>hanoi3</td>
<td>8</td>
<td>∃∀∃</td>
<td>1215</td>
<td>18507</td>
<td>&gt; 1h</td>
<td>&gt; 1h</td>
<td></td>
</tr>
<tr>
<td>bw-l.a</td>
<td>2</td>
<td>∃∀∃</td>
<td>1099</td>
<td>62916</td>
<td>&gt; 1h</td>
<td>1.13</td>
<td></td>
</tr>
<tr>
<td>bw-l.a</td>
<td>4</td>
<td>∃∀∃</td>
<td>1370</td>
<td>65688</td>
<td>&gt; 1h</td>
<td>256.75</td>
<td></td>
</tr>
<tr>
<td>bw-l.b</td>
<td>2</td>
<td>∃∀∃</td>
<td>1871</td>
<td>178890</td>
<td>&gt; 1h</td>
<td>2.38</td>
<td></td>
</tr>
<tr>
<td>bw-l.b</td>
<td>4</td>
<td>∃∀∃</td>
<td>2268</td>
<td>183741</td>
<td>&gt; 1h</td>
<td>15.65</td>
<td></td>
</tr>
<tr>
<td>bw-l.b</td>
<td>8</td>
<td>∃∀∃</td>
<td>2665</td>
<td>188592</td>
<td>&gt; 1h</td>
<td>&gt; 1h</td>
<td></td>
</tr>
</tbody>
</table>
## Improvement on random QBF

<table>
<thead>
<tr>
<th>Improvement (%)</th>
<th>1.00</th>
<th>1.50</th>
<th>2.00</th>
<th>2.50</th>
<th>3.00</th>
<th>3.50</th>
<th>4.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0%</td>
<td>5.14</td>
<td>8.80</td>
<td>18.62</td>
<td>27.11</td>
<td>34.16</td>
<td>51.22</td>
<td>313.54</td>
</tr>
<tr>
<td>5.3%</td>
<td>5.19</td>
<td>9.43</td>
<td>152.52</td>
<td>1874.86</td>
<td>5679.83</td>
<td>4225.16</td>
<td>988.06</td>
</tr>
<tr>
<td>10.7%</td>
<td>5.29</td>
<td>10.97</td>
<td>847.85</td>
<td>6815.47</td>
<td>3922.47</td>
<td>1141.70</td>
<td>450.39</td>
</tr>
<tr>
<td>16.0%</td>
<td>5.60</td>
<td>16.67</td>
<td>1369.13</td>
<td>1985.74</td>
<td>869.00</td>
<td>402.73</td>
<td>232.92</td>
</tr>
<tr>
<td>21.3%</td>
<td>5.60</td>
<td>28.63</td>
<td>491.05</td>
<td>438.56</td>
<td>281.99</td>
<td>207.00</td>
<td>134.24</td>
</tr>
<tr>
<td>26.7%</td>
<td>5.84</td>
<td>43.45</td>
<td>149.33</td>
<td>140.81</td>
<td>120.16</td>
<td>98.20</td>
<td>86.07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Improvement (%)</th>
<th>1.00</th>
<th>1.50</th>
<th>2.00</th>
<th>2.50</th>
<th>3.00</th>
<th>3.50</th>
<th>4.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0%</td>
<td>5.20</td>
<td>9.06</td>
<td>19.20</td>
<td>28.35</td>
<td>36.26</td>
<td>54.76</td>
<td>392.60</td>
</tr>
<tr>
<td>5.3%</td>
<td>5.82</td>
<td>12.00</td>
<td>80.59</td>
<td>107.53</td>
<td>369.88</td>
<td>439.06</td>
<td>153.00</td>
</tr>
<tr>
<td>10.7%</td>
<td>5.97</td>
<td>14.26</td>
<td>80.64</td>
<td>382.74</td>
<td>240.94</td>
<td>122.04</td>
<td>84.91</td>
</tr>
<tr>
<td>16.0%</td>
<td>6.00</td>
<td>19.93</td>
<td>195.35</td>
<td>228.58</td>
<td>131.70</td>
<td>126.41</td>
<td>85.20</td>
</tr>
<tr>
<td>21.3%</td>
<td>6.86</td>
<td>34.37</td>
<td>181.10</td>
<td>173.56</td>
<td>140.23</td>
<td>95.50</td>
<td>88.40</td>
</tr>
<tr>
<td>26.7%</td>
<td>7.19</td>
<td>56.05</td>
<td>164.50</td>
<td>160.94</td>
<td>117.94</td>
<td>100.16</td>
<td>87.78</td>
</tr>
</tbody>
</table>
Conclusions, research directions

- Many good applications for QBF; algorithms do not scale up.

- Possibilities of using Q-resolution to obtain more powerful algorithms for QBF?
  - Some of the modern SAT algorithms with clause learning are essentially inferring new clauses by resolution, until the empty clause is inferred. ⇒ Forget the story about binary search trees as in the Davis-Putnam procedure!!
  - Can the same idea be efficiently applied to QBF as well?