Resolve and Expand

May 13, 2004

Armin Biere
Computer Systems Institute
Department of Computer Science
ETH Zürich, Switzerland

SAT’04
Seventh International Conference on
Theory and Applications of Satisfiability Testing

May 10–13, 2004, Vancouver, BC, Canada
Symbolic Transitive Closure

\[ T \quad \text{boolean formula encoding a (finite transition) relation} \]

\[ \left[ [T] \right] \subseteq \{0, 1\}^n \times \{0, 1\}^n \]

Transitive Closure

\[ T^* \equiv T^{2n} \]

Standard Linear Unfolding

\[ T^{i+1}(s,t) \equiv \exists m. T^i(s,m) \land T(m,t) \]

Iterative Squaring via Copying

\[ T^{2\cdot i}(s,t) \equiv \exists m. T^i(s,m) \land T^i(m,t) \]

Non Copying Iterative Squaring

\[ T^{2\cdot i}(s,t) \equiv \exists m. \forall c. \exists l, r. (c \rightarrow (l, r) = (s, m)) \land (\overline{c} \rightarrow (l, r) = (m, t)) \land T^i(l, r) \]
Why is QBF harder than SAT?

\[ \models \forall x. \exists y. (x \leftrightarrow y) \]

\[ \not\models \exists y. \forall x. (x \leftrightarrow y) \]

Decision Order Matters!
solve-sat(assignment)

boolean-constraint-propagation()
if contains-empty-clause() then return false
if no-clause-left() then return true
v := next-unassigned-variable()
return solve-sat(S \{v \mapsto false\}) \lor solve-sat(S \{v \mapsto true\})

solve-qbf(assignment)

boolean-constraint-propagation()
if contains-empty-clause() then return false
if no-clause-left() then return true
v := next-outermost-unassigned-variable()
@ := is-existential(v) \land \lor
return solve-sat(S \{v \mapsto false\}) \land \lor @ \land solve-sat(S \{v \mapsto true\})
• almost all implementations are QBF-enhanced DPLL:  [Cadoli...98] [Rintanen01]
  – recently learning was added  [Giunchiglia...01] [Letz01] [ZhangMalik02]
  – all deterministic solvers (except one) in QBF-Evaluation’03 were DPLL based
  – top-down: split on variables from the outside to the inside

• multiple quantifier elimination procedures:
  – enumeration  [PlaistedBiereZhu03] [McMillan02]
  – expansion  [Aziz-Abdulla...00] [WilliamsBiere...00] [AyariBasin02]
  – bottom-up: eliminate variables from the inside to the outside

• q-resolution  [Kleine-Büning...95]
Forall Reduction and Q-Resolution

- **collect** variables in scopes, **order** variables and scopes according to nesting depth:

  \[
  \exists a, b, c, d, \quad \forall x, y, z, \quad \exists r, s, t, \quad (c \lor d)(a \lor \bar{c} \lor \bar{x} \lor y)(\bar{a} \lor x \lor s)(t \lor \ldots) \ldots
  \]

  **attach** clauses to the scope of its innermost variables

- **remove** innermost universal literals in clauses attached to universal scopes:

  \[
  (a \lor \bar{c} \lor \bar{x} \lor y) \quad \text{simplifies to} \quad (a \lor \bar{c})
  \]

- q-resolution = resolution + forall reduction
• all clauses are forall reduced
  ⇒ innermost scope is always existential
  ⇒ no clauses attached to universal scopes

• normalized structure of quantified CNF:

\[ \Omega(S_1) \, S_1 \cdot \Omega(S_2) \, S_2 \cdot \ldots \cdot \forall S_{m-1} \cdot \exists S_m \cdot f \land g \quad m \geq 2 \]

\[
\begin{align*}
f & \equiv \text{clauses of scope } S_m \\
g & \equiv \text{clauses of outer scopes } S_i, \quad i < m - 1 \\
S_\exists & \equiv S_m \\
S_\forall & \equiv S_{m-1}
\end{align*}
\]
Algorithm

resolve-and-expand()

forever

simplify()

if contains-empty-clause() then return false

if no-clause-left() then return true

if is-propositional() then return sat-solve(∅)

v := schedule-cheapest-to-eliminate-variable()

if is-existential(ν) then resolve(ν)

if is-universal(ν) then expand(ν)
original clauses in which \( v \) or \( \neg v \) occurs:

\[
\neg r \lor v \\
\neg v \lor r \\
s \lor v \\
x \lor y \lor v \\
x \lor y \lor r
\]

add forall reduced non-trivial resolvents:

\[
(s \lor r), \quad (x \lor y \lor r), \quad \text{and} \quad (s \lor \neg x \lor \neg y \lor r)
\]

remove original clauses
one-to-one mapping of variables:  \( u \in S_\exists \)  mapped to  \( u' \in S'_\exists \)

before expansion:

\[
\begin{align*}
\Omega(S_1)S_1 & . \quad \Omega(S_2)S_2 & . \quad \ldots \quad \forall S_\forall & . \quad \exists S_\exists & . \quad f \land g
\end{align*}
\]

after expansion:

\[
\begin{align*}
\Omega(S_1)S_1 & . \quad \Omega(S_2)S_2 & . \quad \ldots \quad \forall (S_\forall - \{v\}) & . \quad \exists (S_\exists \cup S'_\exists) & . \quad f[v/0] \land f'[v/1] & \land g
\end{align*}
\]
Schedule

- elimination cost: \textbf{number of expected added literals}

\[ o(l) \equiv \text{number of clauses with literal } l \]
\[ s(l) \equiv \text{sum of lengths of clauses with literal } l \]
\[ s(S) \equiv \text{sum lengths of clauses with scope } S \]

- expansion cost: \[ s(S_{\exists}) = \left( s(v) + s(\neg v) + o(v) + o(\neg v) \right) \]

- resolution cost: \[ o(\neg v) \cdot \left( s(v) - o(v) \right) + o(v) \cdot \left( s(\neg v) - o(\neg v) \right) - \left( s(v) + s(\neg v) \right) \]
## Benchmarking Structured Instances of SAT’03 QBF Evaluation

<table>
<thead>
<tr>
<th>benchmark family</th>
<th>#inst</th>
<th>decide</th>
<th>qube</th>
<th>semprop</th>
<th>expand</th>
<th>quantor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 adder*</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2 Adder2*</td>
<td>14</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3 C[0-9]*</td>
<td>27</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4 CHAIN*</td>
<td>11</td>
<td>10</td>
<td>7</td>
<td>11</td>
<td>4</td>
<td>11</td>
</tr>
<tr>
<td>5 comp*</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6 flip*</td>
<td>7</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7 impl*</td>
<td>16</td>
<td>12</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>8 k*</td>
<td>171</td>
<td>77</td>
<td>91</td>
<td>97</td>
<td>60</td>
<td>108</td>
</tr>
<tr>
<td>9 mutex*</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>10 robots*</td>
<td>48</td>
<td>0</td>
<td>36</td>
<td>36</td>
<td>15</td>
<td>24</td>
</tr>
<tr>
<td>11 term1*</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>12 toilet*</td>
<td>260</td>
<td>187</td>
<td>260</td>
<td>260</td>
<td>259</td>
<td>259</td>
</tr>
<tr>
<td>13 TOILET*</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>14 tree*</td>
<td>12</td>
<td>10</td>
<td>12</td>
<td>12</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

(families with no difference and two actually random families removed)
• resolve quadratic in number of occurrences, expand may double the size
  ⇒ simplify CNF as much as possible before elimination

• standard simplification: **unit propagation, pure literal rule, forall reduction**

• **equivalence reasoning:** extract bi-implications and substitute variables
  \[
  \forall x . \exists y . (x \lor y)(x \rightarrow y)(y \rightarrow x) \equiv \forall x . \exists y . (x \lor y)(x = y) \equiv \forall x . \exists y . (x \lor x) \equiv 0
  \]

• **subsumption:** remove subsumed clauses
  – backward subsumption is checked on-the-fly whenever a clause is added
  – forward subsumption is expensive and only checked before expensive operations
### Solved Hard Instances of SAT’03 QBF Evaluation: QUANTOR

<table>
<thead>
<tr>
<th>hard instance</th>
<th>time</th>
<th>space</th>
<th>$\forall$</th>
<th>$\exists$</th>
<th>units</th>
<th>pure</th>
<th>subsu.</th>
<th>subst.</th>
<th>$\forall$red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Adder2-6-s</td>
<td>29.6</td>
<td>19.7</td>
<td>90</td>
<td>13732</td>
<td>126</td>
<td>13282</td>
<td>174081</td>
<td>0</td>
<td>37268</td>
</tr>
<tr>
<td>2 adder-4-sat</td>
<td>0.2</td>
<td>2.8</td>
<td>42</td>
<td>1618</td>
<td>0</td>
<td>884</td>
<td>6487</td>
<td>0</td>
<td>960</td>
</tr>
<tr>
<td>3 adder-6-sat</td>
<td>36.6</td>
<td>22.7</td>
<td>90</td>
<td>13926</td>
<td>0</td>
<td>7290</td>
<td>197091</td>
<td>0</td>
<td>54174</td>
</tr>
<tr>
<td>4 C49<em>1.</em>_0_0*</td>
<td>27.9</td>
<td>13.3</td>
<td>1</td>
<td>579</td>
<td>0</td>
<td>0</td>
<td>48</td>
<td>84</td>
<td>0</td>
</tr>
<tr>
<td>5 C5<em>1.</em>_0_0*</td>
<td>56.2</td>
<td>16.0</td>
<td>2</td>
<td>2288</td>
<td>10</td>
<td>0</td>
<td>4552</td>
<td>2494</td>
<td>0</td>
</tr>
<tr>
<td>6 k_path_n-15</td>
<td>0.1</td>
<td>0.8</td>
<td>32</td>
<td>977</td>
<td>66</td>
<td>82</td>
<td>2369</td>
<td>2</td>
<td>547</td>
</tr>
<tr>
<td>7 k_path_n-16</td>
<td>0.1</td>
<td>0.8</td>
<td>34</td>
<td>1042</td>
<td>69</td>
<td>85</td>
<td>2567</td>
<td>2</td>
<td>597</td>
</tr>
<tr>
<td>8 k_path_n-17</td>
<td>0.1</td>
<td>0.9</td>
<td>36</td>
<td>1087</td>
<td>72</td>
<td>100</td>
<td>3020</td>
<td>2</td>
<td>639</td>
</tr>
<tr>
<td>9 k_path_n-18</td>
<td>0.1</td>
<td>0.9</td>
<td>36</td>
<td>1146</td>
<td>76</td>
<td>106</td>
<td>3242</td>
<td>2</td>
<td>725</td>
</tr>
<tr>
<td>10 k_path_n-20</td>
<td>0.1</td>
<td>0.9</td>
<td>38</td>
<td>1240</td>
<td>84</td>
<td>149</td>
<td>3967</td>
<td>2</td>
<td>855</td>
</tr>
<tr>
<td>11 k_path_n-21</td>
<td>0.1</td>
<td>1.0</td>
<td>40</td>
<td>1318</td>
<td>84</td>
<td>130</td>
<td>4470</td>
<td>2</td>
<td>909</td>
</tr>
<tr>
<td>12 k_t4p_n-7</td>
<td>15.5</td>
<td>105.8</td>
<td>43</td>
<td>88145</td>
<td>138</td>
<td>58674</td>
<td>760844</td>
<td>8</td>
<td>215</td>
</tr>
<tr>
<td>13 k_t4p_p-8</td>
<td>5.8</td>
<td>178.6</td>
<td>29</td>
<td>12798</td>
<td>206</td>
<td>5012</td>
<td>85911</td>
<td>4</td>
<td>138</td>
</tr>
<tr>
<td>14 k_t4p_p-9</td>
<td>0.3</td>
<td>4.5</td>
<td>32</td>
<td>4179</td>
<td>137</td>
<td>1389</td>
<td>23344</td>
<td>10</td>
<td>142</td>
</tr>
<tr>
<td>15 k_t4p_p-10</td>
<td>27.9</td>
<td>152.9</td>
<td>35</td>
<td>130136</td>
<td>193</td>
<td>63876</td>
<td>938973</td>
<td>4</td>
<td>137</td>
</tr>
<tr>
<td>16 k_t4p_p-11</td>
<td>86.0</td>
<td>471.5</td>
<td>38</td>
<td>196785</td>
<td>204</td>
<td>79547</td>
<td>1499430</td>
<td>4</td>
<td>140</td>
</tr>
<tr>
<td>17 k_t4p_p-15</td>
<td>84.6</td>
<td>354.7</td>
<td>50</td>
<td>240892</td>
<td>169</td>
<td>181676</td>
<td>1336774</td>
<td>9</td>
<td>226</td>
</tr>
<tr>
<td>18 k_t4p_p-20</td>
<td>3.6</td>
<td>16.1</td>
<td>65</td>
<td>27388</td>
<td>182</td>
<td>21306</td>
<td>197273</td>
<td>11</td>
<td>325</td>
</tr>
</tbody>
</table>

| time in seconds, space in MB |
## Solved Hard Instances of SAT’03 QBF Evaluation: **EXPAND ONLY**

<table>
<thead>
<tr>
<th>hard instance</th>
<th>time</th>
<th>space</th>
<th>( \forall )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Adder2-6-s</td>
<td>(12.2)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>2 adder-4-sat</td>
<td>(12.1)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>3 adder-6-sat</td>
<td>(13.0)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>4 C49<em>1.</em>.0.0*</td>
<td>98.3</td>
<td>40.8</td>
<td>1</td>
</tr>
<tr>
<td>5 C5<em>1.</em>.0.0*</td>
<td>357.0</td>
<td>45.6</td>
<td>2</td>
</tr>
<tr>
<td>6 k_path_n-15</td>
<td>(16.5)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>7 k_path_n-16</td>
<td>(16.6)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>8 k_path_n-17</td>
<td>(16.2)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>9 k_path_n-18</td>
<td>(16.8)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>10 k_path_n-20</td>
<td>(21.4)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>11 k_path_n-21</td>
<td>(21.0)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>12 k_t4p_n-7</td>
<td>(16.8)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>13 k_t4p_p-8</td>
<td>(21.4)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>14 k_t4p_p-9</td>
<td>(21.2)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>15 k_t4p_p-10</td>
<td>(17.3)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>16 k_t4p_p-11</td>
<td>(17.3)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>17 k_t4p_p-15</td>
<td>(21.3)</td>
<td>m.o.</td>
<td>–</td>
</tr>
<tr>
<td>18 k_t4p_p-20</td>
<td>(20.9)</td>
<td>m.o.</td>
<td>–</td>
</tr>
</tbody>
</table>

Time in seconds, space in MB, m.o. = memory out (\( > 1 \) GB)
Conclusion

• novel resolution and expansion based QBF decision procedure:
  – simplifications: fast subsumption algorithm, ...
  – resource-driven scheduler
  – efficient implementation: QUANTOR

• future directions:
  – combine bottom-up with top-down
  – QBF more complex than SAT:
    ⇒ many and more expensive optimizations possible
  – compact data structures: BDDs or ZBDDs