DepQBF: A Dependency-Aware QBF Solver (System Description)

Florian Lonsing and Armin Biere

Institute for Formal Models and Verification (FMV)
Johannes Kepler University, Linz, Austria
http://fmv.jku.at

Pragmatics of SAT (POS) Workshop
July 10, 2010
Edinburgh, Scotland, United Kingdom
### Solver Scores

<table>
<thead>
<tr>
<th>Solver</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DepQBF</td>
<td>2896.68</td>
</tr>
<tr>
<td>DepQBF-pre</td>
<td>2508.96</td>
</tr>
<tr>
<td>aqme-10</td>
<td>2467.96</td>
</tr>
<tr>
<td>qmaiga</td>
<td>2117.55</td>
</tr>
<tr>
<td>AIGSolve</td>
<td>2037.22</td>
</tr>
<tr>
<td>quantor-3.1</td>
<td>1235.14</td>
</tr>
<tr>
<td>struqs-10</td>
<td>947.83</td>
</tr>
<tr>
<td>nenofex-qbfeval10</td>
<td>829.11</td>
</tr>
</tbody>
</table>

http://www.qbflib.org/index_eval.php

**This Talk:**
- DepQBF 0.1 system overview.
- Selected features: restarts, removal of learnt constraints.
- Experimental evaluation.
DepQBF:
- Input: QBFs in Prenex-CNF (PCNF).
- QDPLL with conflict-driven clause and solution-driven cube learning.
- Analysis of variable dependencies.

Variable Dependencies in QBFs:
- PCNF $Q_1 Q_2 \ldots Q_n. \phi$: linearly ordered sets of quantified variables.
- Left-to-right prefix order: strong dependencies.
- DepQBF: relaxing prefix order by dependency schemes.

Example
Quantifier ordering matters:
- $\forall x \exists y. (x = y)$ is satisfiable: value of $y$ depends on value of $x$.
- $\exists y \forall x. (x = y)$ is unsatisfiable: value of $y$ is fixed for all values of $x$. 
Dependency Schemes: $D \subseteq (V_\exists \times V_\forall) \cup (V_\forall \times V_\exists)$. [SS09, LB09, LB10, Ben05]

- $(x, y) \notin D$: $y$ independent from $x$.
- $(x, y) \in D$: conservatively regard $y$ as depending on $x$.

DepQBF: *standard dependency scheme* $D^{\text{std}} \subseteq D^{\text{triv}}$.

- Previous work: $D^{\text{std}}$ as dependency-DAG over equivalence classes.
- Efficient integration.

**Example:** $\exists a, b \forall x, y \exists c, d. (a \lor x \lor c) \land (a \lor b) \land (b \lor d) \land (y \lor d)$.

![Dependency Diagram](image)

Standard dependency scheme $D^{\text{std}}$, quantifier prefix $D^{\text{triv}}$. 

Florian Lonsing and Armin Biere

DepQBF: A Dependency-Aware QBF Solver (System Description)
**Boolean Constraint Propagation (BCP):**

- Propagation of unit and pure literals.
- Watched data-structures for efficient detection.
**Initialize Dependency-DAG:**

- Top-most decision level 0.
- All assignments at top-level are permanent.
- Permanent simplifications (satisfied clauses).
- Potential reduction of dependencies.

---

Figure: DepQBF workflow.
Retrieve Decision Candidates (DC):

- Get possible decision variables (candidates) from dependency-DAG.
- Candidate: all “preconditions” (predecessors in DAG) assigned.
- Candidate set is maintained incrementally and lazily.

Figure: DepQBF workflow.
**Decision Making:**

- Select decision variable from candidate set.
- Activity-based priority queue of variables (VSIDS, like MiniSAT 2).
- Assignment caching.
**Constraint Learning (Result Analysis):**

- Conflict/solution: generate *asserting* learnt clause/cube.
- Augmented CNF: $\phi := \phi_{OCL} \land (\phi_{LCL} \lor \phi_{LCU})$.
- Learnt clauses $\phi_{LCL}$ and cubes $\phi_{LCU}$.
- Q-resolution/consensus to derive learnt clauses/cubes.
- See also our SAT’10 paper.
Learnt Constraint Removal and Restarts:

- Check each time when adding a new learnt constraint.
- Capacity exhausted: remove half of learnt constraints.
- Heuristically try to keep “useful” constraints, increase capacity.
- Inner-outer restart schedule (like PicoSAT).
**Backtracking:**

- General (frequent) case: backtrack to asserting level of learnt constraint.
- Special case: backtrack to restart level.
**Learnt Constraint Removal**

**Learnt Constraints:** [GNT02, Let02, ZM02, GNT06, BKF95, GS08, ES03, GN02]
- Clauses $\phi_{LCL}$ and cubes $\phi_{LCU}$, stored in doubly-linked lists.
- Initial capacities depend on formula size: [2500, 10000].

**Move-To-Front (MTF) Strategy:** approximating clause activities.
- Want to keep “used” (i.e. important?) constraints: units, learning.
- Move used constraints $C_i$ to head of list:

\[
\{ C_1, \ldots, C_{i-1}, C_i, C_{i+1}, \ldots, C_n \} \xrightarrow{MTF(C_i)} \{ C_i, C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n \}
\]

most-recently used  least-recently used  \( \leftarrow \) deletion order \( \leftarrow \)

**Deletion:**
- Capacity exhausted: remove half of constraints, starting at tail of list.
- Least-recently used ones are deleted (hopefully: least-important ones).
- Increase capacity by constant 500.
Inner-Outer Restart Schedule: when to restart?

- Inspired by PicoSAT: separate inner/outer restarts.
- Inner restart after \( i \) backtracks, outer restart after \( o \) inner restarts.
- Initially \( i := 100, \ o := 10 \).
- Before \( i \)th ordinary backtrack: jump to restart level instead, \( i := i + 10 \).
- After \( o \) inner restarts: \( i := 100, \ o := o + 5 \) (outer restart).
Restart Level: where to jump to?
- Normally, DepQBF always jumps to asserting level.
- Restart: possibly jump *most-recent universal decision level* instead.
  - Always the longer jump is taken.
- Related to ideas from unrestricted backtracking [BLdSMS05].

Example:
- Assignment stack, in order of decision levels.
- Conflict/solution at level 4.
- Restart is scheduled, where to jump to?
**Restart Level:** where to jump to?

- Normally, DepQBF always jumps to asserting level.
- Restart: possibly jump *most-recent universal decision level* instead.
  - Always the longer jump is taken.
- Related to ideas from unrestricted backtracking [BLdSMS05].

**Example:**

- Current learnt constraint asserting at level 3.
- Last universal decision at level 2.
**Restart Level:** where to jump to?
- Normally, DepQBF always jumps to asserting level.
- Restart: possibly jump *most-recent universal decision level* instead.
  - Always the longer jump is taken.
- Related to ideas from unrestricted backtracking [BLdSMS05].

**Example:**
- Current learnt constraint asserting at level 3.
- Last universal decision at level 2.
- **Restart:** take the longer jump.
Restart Level: where to jump to?

- Normally, DepQBF always jumps to asserting level.
- Restart: possibly jump *most-recent universal decision level* instead.
  - Always the longer jump is taken.
- Related to ideas from unrestricted backtracking [BLdSMS05].

Example:

- Current learnt constraint asserting at level 1.
- Last universal decision at level 2.
**Restart Level:** where to jump to?

- Normally, DepQBF always jumps to asserting level.
- Restart: possibly jump *most-recent universal decision level* instead.
  - Always the longer jump is taken.
- Related to ideas from unrestricted backtracking [BLdSMS05].

**Example:**
- Current learnt constraint asserting at level 1.
- Last universal decision at level 2.
- **Restart:** take the longer jump.
<table>
<thead>
<tr>
<th></th>
<th>solved</th>
<th>avg.time</th>
<th>solved</th>
<th>avg.time</th>
<th>solved</th>
<th>avg.time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>All</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Solved SAT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Solved UNSAT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>without preprocessing</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DepQBF</td>
<td>370</td>
<td>337.10</td>
<td>165</td>
<td>54.58</td>
<td>205</td>
<td>20.82</td>
</tr>
<tr>
<td>DepQBF-nr</td>
<td>360</td>
<td>352.33</td>
<td>154</td>
<td>51.36</td>
<td>206</td>
<td>24.35</td>
</tr>
<tr>
<td>DepQBF-nc</td>
<td>350</td>
<td>384.66</td>
<td>157</td>
<td>107.48</td>
<td>193</td>
<td>28.05</td>
</tr>
<tr>
<td>DepQBF-np</td>
<td>345</td>
<td>398.12</td>
<td>141</td>
<td>114.72</td>
<td>204</td>
<td>45.37</td>
</tr>
<tr>
<td>DepQBF-ncnr</td>
<td>340</td>
<td>400.24</td>
<td>147</td>
<td>124.10</td>
<td>193</td>
<td>20.19</td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>332</td>
<td>425.44</td>
<td>135</td>
<td>147.71</td>
<td>197</td>
<td>47.27</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>301</td>
<td>468.51</td>
<td>113</td>
<td>136.48</td>
<td>188</td>
<td>55.27</td>
</tr>
</tbody>
</table>

**Table:** QBFEVAL’10 main track (568 formulae). Ranking by number of solved formulae.

**Setup:**
- Ubuntu 9.04, Intel® Q9550@2.83 GHz, 3 GB/900 sec.
- DepQBF: version 0.1 which participated in QBFEVAL’10.
### Table: QBFEVAL’10 main track (568 formulae). Ranking by number of solved formulae.

**Important:**
- Reversals (disabled in DepQBF-nr).

<table>
<thead>
<tr>
<th>System</th>
<th>All solved</th>
<th>All avg.time</th>
<th>Solved SAT solved</th>
<th>Solved SAT avg.time</th>
<th>Solved UNSAT solved</th>
<th>Solved UNSAT avg.time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DepQBF</td>
<td>370</td>
<td>337.10</td>
<td>165</td>
<td>54.58</td>
<td>205</td>
<td>20.82</td>
</tr>
<tr>
<td>DepQBF-nr</td>
<td>360</td>
<td>352.33</td>
<td>154</td>
<td>51.36</td>
<td>206</td>
<td>24.35</td>
</tr>
<tr>
<td>DepQBF-nc</td>
<td>350</td>
<td>384.66</td>
<td>157</td>
<td>107.48</td>
<td>193</td>
<td>28.05</td>
</tr>
<tr>
<td>DepQBF-np</td>
<td>345</td>
<td>398.12</td>
<td>141</td>
<td>114.72</td>
<td>204</td>
<td>45.37</td>
</tr>
<tr>
<td>DepQBF-ncnr</td>
<td>340</td>
<td>400.24</td>
<td>147</td>
<td>124.10</td>
<td>193</td>
<td>20.19</td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>332</td>
<td>425.44</td>
<td>135</td>
<td>147.71</td>
<td>197</td>
<td>47.27</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>301</td>
<td>468.51</td>
<td>113</td>
<td>136.48</td>
<td>188</td>
<td>55.27</td>
</tr>
</tbody>
</table>

**Important:**
- Reversals (disabled in DepQBF-nr).
### Experiments: QBFEVAL’10 Main Track

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Solved SAT</th>
<th>Solved UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>solved</td>
<td>avg.time</td>
<td>solved</td>
</tr>
<tr>
<td>DepQBF</td>
<td>370</td>
<td>337.10</td>
<td>165</td>
</tr>
<tr>
<td>DepQBF-nr</td>
<td>360</td>
<td>352.33</td>
<td>154</td>
</tr>
<tr>
<td><strong>DepQBF-nc</strong></td>
<td><strong>350</strong></td>
<td><strong>384.66</strong></td>
<td><strong>157</strong></td>
</tr>
<tr>
<td>DepQBF-np</td>
<td>345</td>
<td>398.12</td>
<td>141</td>
</tr>
<tr>
<td>DepQBF-ncnr</td>
<td>340</td>
<td>400.24</td>
<td>147</td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>332</td>
<td>425.44</td>
<td>135</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>301</td>
<td>468.51</td>
<td>113</td>
</tr>
</tbody>
</table>

**Table:** QBFEVAL’10 main track (568 formulae). Ranking by number of solved formulae.

**Important:**

- Restarts.
- Assignment caching (disabled in DepQBF-nc).
### Table: QBFEVAL’10 main track (568 formulae). Ranking by number of solved formulae.

<table>
<thead>
<tr>
<th>Solver</th>
<th>All solved</th>
<th>All avg.time</th>
<th>Solved SAT solved</th>
<th>Solved SAT avg.time</th>
<th>Solved UNSAT solved</th>
<th>Solved UNSAT avg.time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DepQBF</td>
<td>370</td>
<td>337.10</td>
<td>165</td>
<td>54.58</td>
<td>205</td>
<td>20.82</td>
</tr>
<tr>
<td>DepQBF-nr</td>
<td>360</td>
<td>352.33</td>
<td>154</td>
<td>51.36</td>
<td>206</td>
<td>24.35</td>
</tr>
<tr>
<td>DepQBF-nc</td>
<td>350</td>
<td>384.66</td>
<td>157</td>
<td>107.48</td>
<td>193</td>
<td>28.05</td>
</tr>
<tr>
<td><strong>DepQBF-np</strong></td>
<td><strong>345</strong></td>
<td><strong>398.12</strong></td>
<td><strong>141</strong></td>
<td><strong>114.72</strong></td>
<td><strong>204</strong></td>
<td><strong>45.37</strong></td>
</tr>
<tr>
<td>DepQBF-ncnr</td>
<td>340</td>
<td>400.24</td>
<td>147</td>
<td>124.10</td>
<td>193</td>
<td>20.19</td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>332</td>
<td>425.44</td>
<td>135</td>
<td>147.71</td>
<td>197</td>
<td>47.27</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>301</td>
<td>468.51</td>
<td>113</td>
<td>136.48</td>
<td>188</td>
<td>55.27</td>
</tr>
</tbody>
</table>

**Important:**
- Restarts.
- Assignment caching.
- Pure literal detection (disabled in DepQBF-np).
## Experiments: QBFEVAL’10 Main Track

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Solved SAT</th>
<th>Solved UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>solved</td>
<td>avg.time</td>
<td>solved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>without preprocessing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DepQBF</td>
<td>370</td>
<td>337.10</td>
<td>165</td>
</tr>
<tr>
<td>DepQBF-nr</td>
<td>360</td>
<td>352.33</td>
<td>154</td>
</tr>
<tr>
<td>DepQBF-nc</td>
<td>350</td>
<td>384.66</td>
<td>157</td>
</tr>
<tr>
<td>DepQBF-np</td>
<td>345</td>
<td>398.12</td>
<td>141</td>
</tr>
<tr>
<td><strong>DepQBF-ncnr</strong></td>
<td><strong>340</strong></td>
<td><strong>400.24</strong></td>
<td><strong>147</strong></td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>332</td>
<td>425.44</td>
<td>135</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>301</td>
<td>468.51</td>
<td>113</td>
</tr>
</tbody>
</table>

### Table: QBFEVAL’10 main track (568 formulae). Ranking by number of solved formulae.

### Important:

- Restarts.
- Assignment caching.
- Pure literal detection.
- Combining restarts with assignment caching (disabled in DepQBF-ncnr).
<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Solved SAT</th>
<th>Solved UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>solved</td>
<td>avg.time</td>
<td>solved</td>
</tr>
<tr>
<td>QuBE7.0-pre⇒DepQBF</td>
<td>424</td>
<td>254.23</td>
<td>197</td>
</tr>
<tr>
<td>QuBE7</td>
<td>414</td>
<td>310.29</td>
<td>187</td>
</tr>
<tr>
<td>QuBE6.6</td>
<td>387</td>
<td>341.91</td>
<td>168</td>
</tr>
</tbody>
</table>

without preprocessing

<table>
<thead>
<tr>
<th></th>
<th>All</th>
<th>Solved SAT</th>
<th>Solved UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>solved</td>
<td>avg.time</td>
<td>solved</td>
</tr>
<tr>
<td>DepQBF</td>
<td>370</td>
<td>337.10</td>
<td>165</td>
</tr>
<tr>
<td>DepQBF-nr</td>
<td>360</td>
<td>352.33</td>
<td>154</td>
</tr>
<tr>
<td>DepQBF-nc</td>
<td>350</td>
<td>384.66</td>
<td>157</td>
</tr>
<tr>
<td>DepQBF-np</td>
<td>345</td>
<td>398.12</td>
<td>141</td>
</tr>
<tr>
<td>DepQBF-ncnr</td>
<td>340</td>
<td>400.24</td>
<td>147</td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>332</td>
<td>425.44</td>
<td>135</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>301</td>
<td>468.51</td>
<td>113</td>
</tr>
</tbody>
</table>

Table: QBFEVAL’10 main track (568 formulae). Ranking by number of solved formulae.

Important:
- Restarts.
- Assignment caching.
- Pure literal detection.
- Combining restarts with assignment caching.
- Preprocessing (not part of DepQBF 0.1, disabled in QuBE*-nopp).
Experiments: QBFEVAL’10 Main Track

Florian Lonsing and Armin Biere

DepQBF: A Dependency-Aware QBF Solver (System Description)
Experiments: QBF EVAL’10 Main Track

Florian Lonsing and Armin Biere

DepQBF: A Dependency-Aware QBF Solver (System Description)
Experiments: QBFEVAL'10 Main Track

Florian Lonsing and Armin Biere

DepQBF: A Dependency-Aware QBF Solver (System Description)
DepQBF:

- Search-based QBF solver with clause- and cube-learning.
- Relaxing prefix order by dependency-DAG for $D_{\text{std}}$.
- Approaches from SAT domain.
- Development:
  - Fuzz testing using QBFuzz: http://fmv.jku.at/qbfuzz/
  - Delta-debugging using QBFDD: http://fmv.jku.at/qbfdd/
  - Cross-checking against other solvers, mainly QuBE.

Performance:

- Top-ranked solver in QBFEVAL’10.
- DepQBF 0.1 does not include preprocessing.
- But: preprocessing is very important.

Future Work:

- Preprocessing, parameter tuning, decision heuristics, . . .

DepQBF 0.1 is open source: http://fmv.jku.at/depqbf/
Unit Clauses: Clause $C$ is unit iff [CGS98, GGN+03, MMZ+01, GNT07]

- no $l \in C$ is true.
- exactly one $l_e \in L_\exists(C)$ is unassigned.
- for all unassigned $l_u \in L_\forall(C)$: $l_u \not\prec l_e$, i.e. $\text{Var}(l_u), \text{Var}(l_e)$ independent.
- Dependency checking $\prec$ with respect to dependency scheme.
- Dual definition for cubes.

Two-Literal-Watching:

- Watch two unassigned literals $l_1, l_2 \in C$ such that
  1. either $q(l_1) = q(l_2) = \exists$, or
  2. $q(l_1) = \forall$, $q(l_2) = \exists$ and $l_1 \prec l_2$.

Watcher Update:

- Dependency checking needed only in case (2).
- Stop when finding satisfying literal.
- No work needed during backtracking.
Pure Literals (PL):

- Variable has only positive/negative literals left.
- Assigning $\forall$-PLs/$\exists$-PLs can trigger new units/further PLs.
- Drawback: expensive detection in $\phi_{OCL} \land (\phi_{LCL} \lor \phi_{LCU})$.

Spurious Pure Literals (SPL):

- Def.: Variable is pure (SPL) if it is pure in original clauses $\phi_{OCL}$ only.
- SPL-Detection neglects all learnt constraints in $(\phi_{LCL} \lor \phi_{LCU})$.
  - Advantage: more efficient detection.
- Variable might be pure in $\phi_{OCL}$ but not in $\phi_{OCL} \land (\phi_{LCL} \lor \phi_{LCU})$.
  - Drawback: must ignore such SPL-implications in $(\phi_{LCL} \lor \phi_{LCU})$.

Clause Watching:

- Positive/negative occurrences $C(x), C(\overline{x}) \subseteq \phi_{OCL}$.
- Watch two unsatisfied clauses $C_x \in C(x)$ and $C_{\overline{x}} \in C(\overline{x})$. 

[CGS98, GGN$^+$03, GNT04]
Clause Watcher Update:
- Assign $x/\overline{x}$: all clauses in $C(x)/C(\overline{x})$ will be satisfied.
- Update watchers of variables $y$ watching clauses in $C(x)/C(\overline{x})$.

Notification Lists:
- Goal: avoid searching for variables which need watcher update.
- Lists $NL_x/NL_{\overline{x}}$ of variables $y$ watching clauses in $C(x)/C(\overline{x})$.
- Assign $x/\overline{x}$:
  - exactly all variables in $NL_x/NL_{\overline{x}}$ must update their watcher.
  - update $NL_x/NL_{\overline{x}}$ of variables $x$ occurring in old and new watched clauses.
- No work needed during backtracking.
Activity-Based Variable Priority Queue: [MMZ⁺01, ES03]

- DepQBF: straight-forward generalization of idea from SAT domain.
- Maintain VSIDS score (activity) for each variable.
- Increase activity of variables encountered during learning.
- Periodically down-scale activities.
- Implementation follows MiniSAT 2.
- Decision making: select candidate with highest activity.
- Lazy priority queue maintenance (like in MiniSAT):
  - Discard assigned variables and non-candidates on the fly upon removal.
Also called: Phase Saving

- DepQBF: straight-forward generalization of idea from SAT-domain.
- Each variable has a cached assignment (possibly undefined).
- All assignments (unit, pure literals, decisions) update cache.
- Decision variables: assign cached value, if any.
- No distinction between different quantifiers.
Suite mqm (136 formulae)

<table>
<thead>
<tr>
<th>Solver</th>
<th>solved</th>
<th>avg.time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DepQBF</td>
<td>136</td>
<td>39.83</td>
</tr>
<tr>
<td>QuBE7</td>
<td>117</td>
<td>306.43</td>
</tr>
<tr>
<td>QuBE7.0-nopp</td>
<td>115</td>
<td>304.82</td>
</tr>
<tr>
<td>QuBE6.6</td>
<td>100</td>
<td>393.93</td>
</tr>
<tr>
<td>QuBE6.6-nopp</td>
<td>97</td>
<td>399.55</td>
</tr>
</tbody>
</table>

Table: Solvers sorted by number of solved formulae.

Benchmark Suite *mqm*:

- Minimal Query Inseparability Module Extraction in DL-Lite.
- Newly submitted to QBFEVAL’10 by Roman Kontchakov.
- As the only solver, DepQBF solved entire suite in QBFEVAL’10.
### QBFEVAL’10: solved formulae only

<table>
<thead>
<tr>
<th></th>
<th>∩</th>
<th>SAT-∩</th>
<th>UNSAT-∩</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>328</td>
<td>132</td>
<td>196</td>
</tr>
<tr>
<td>avg.time</td>
<td>84.97</td>
<td>21.87</td>
<td>140.16</td>
</tr>
</tbody>
</table>

### QBFEVAL’10: unique results

<table>
<thead>
<tr>
<th></th>
<th>∩</th>
<th>SAT-⇔</th>
<th>UNSAT-⇔</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>86</td>
<td>42</td>
<td>55</td>
</tr>
</tbody>
</table>

Table: QuBE7 (left columns) vs. DepQBF (right columns).

### QBFEVAL’10: solved formulae only

<table>
<thead>
<tr>
<th></th>
<th>∩</th>
<th>SAT-∩</th>
<th>UNSAT-∩</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>308</td>
<td>115</td>
<td>193</td>
</tr>
<tr>
<td>avg.time</td>
<td>80.14</td>
<td>17.49</td>
<td>114.17</td>
</tr>
</tbody>
</table>

### QBFEVAL’10: unique results

<table>
<thead>
<tr>
<th></th>
<th>∩</th>
<th>SAT-⇔</th>
<th>UNSAT-⇔</th>
</tr>
</thead>
<tbody>
<tr>
<td>solved</td>
<td>79</td>
<td>62</td>
<td>53</td>
</tr>
</tbody>
</table>

Table: QuBE6.6 (left columns) vs. DepQBF (right columns).
M. Benedetti.  
Quantifier Trees for QBFs.  

A. Biere.  
PicoSAT Essentials.  

H. Kleine Büning, M. Karpinski, and A. Flögel.  
Resolution for Quantified Boolean Formulas.  

R. Brummayer, F. Lonsing, and A. Biere.  
Automated Testing and Debugging of SAT and QBF Solvers.  
In O. Strichman and S. Szeider, editors, SAT (accepted for publication), LNCS. Springer, 2010.

Heuristic-Based Backtracking Relaxation for Propositional Satisfiability.  

M. Cadoli, A. Giovanardi, and M. Schaerf.  
An Algorithm to Evaluate Quantified Boolean Formulae.  
N. Eén and N. Sörensson.
An Extensible SAT-Solver.

Watched Data Structures for QBF Solvers.

E. Goldberg and Y. Novikov.
BerkMin: A Fast and Robust SAT-Solver.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Learning for Quantified Boolean Logic Satisfiability.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Monotone Literals and Learning in QBF Reasoning.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Quantifier Structure in Search-Based Procedures for QBFs.

R. Gershman and O. Strichman.
HaifaSat: A SAT Solver Based on an Abstraction/Refinement Model.

F. Lonsing and A. Biere.
A Compact Representation for Syntactic Dependencies in QBFs.

F. Lonsing and A. Biere.
Integrating Dependency Schemes in Search-Based QBF Solvers.

R. Letz.
Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas.

