# Integrating Dependency Schemes in Search-Based QBF Solvers

Florian Lonsing and Armin Biere

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

*Presented at the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT), 2010, Edinburgh, Scotland, UK.*

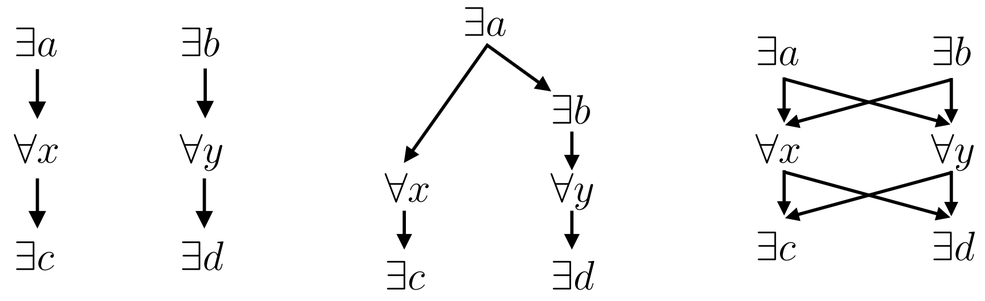
## Search-based QBF Solving for Prenex-CNFs

- QDPLL with clause/cube learning.
- $Q_1 Q_2 \ldots Q_n.\ \phi$: linearly ordered quantifier prefix.
- Quantifier ordering matters:
  - $\forall x \exists y.\ (x = y)$ satisfiable: value of $y$ *depends* on value of $x$.
  - $\exists y \forall x.\ (x = y)$ unsatisfiable: $y$ fixed for all values of $x$.
- Drawback: QDPLL with strict left-to-right quantifier prefix.
- **Goal:** finding independent variables to relax prefix ordering.

## Dependency Schemes

- Def.: Relation $D \subseteq (V_\exists \times V_\forall) \cup (V_\forall \times V_\exists)$ such that reordering quantifier prefix with respect to $D$ preserves equivalence.
- If $(x, y) \notin D$ then $y$ does not depend on $x$, otherwise regard $y$ as depending on $x$ (conservative over-approximation).
- Tractable syntactic approaches for constructing $D$:
  - *Trivial dependency scheme* $D^{\text{triv}}$: given prefix.
  - *Quantifier trees* $D^{\text{tree}}$: non-deterministic mini-scoping.
  - *Standard dependency scheme* $D^{\text{std}}$: variable connections.

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




Dependencies: $D^{\text{std}}$ (left) $\subseteq$ $D^{\text{tree}}$ (middle) $\subseteq$ $D^{\text{triv}}$ (right).

- Expecting more freedom from $D^{\text{std}}$ than from $D^{\text{tree}}$ and $D^{\text{triv}}$.
- **Goal:** generalizing QDPLL to arbitrary dependency schemes.
  - Unit literal detection, q-resolution and learning.
  - Assigning decision variables out of prefix ordering.

## QDPLL with $D^{\text{std}}$: DepQBF

- Integrating $D^{\text{std}}$ in QDPLL as compact dependency-DAG.
- DAG over classes of variables to keep overhead low.
- $(x, y) \in D^{\text{std}}$: checking successor-relation in DAG.
- Approaches from SAT domain:
  - Restarts, phase saving,...

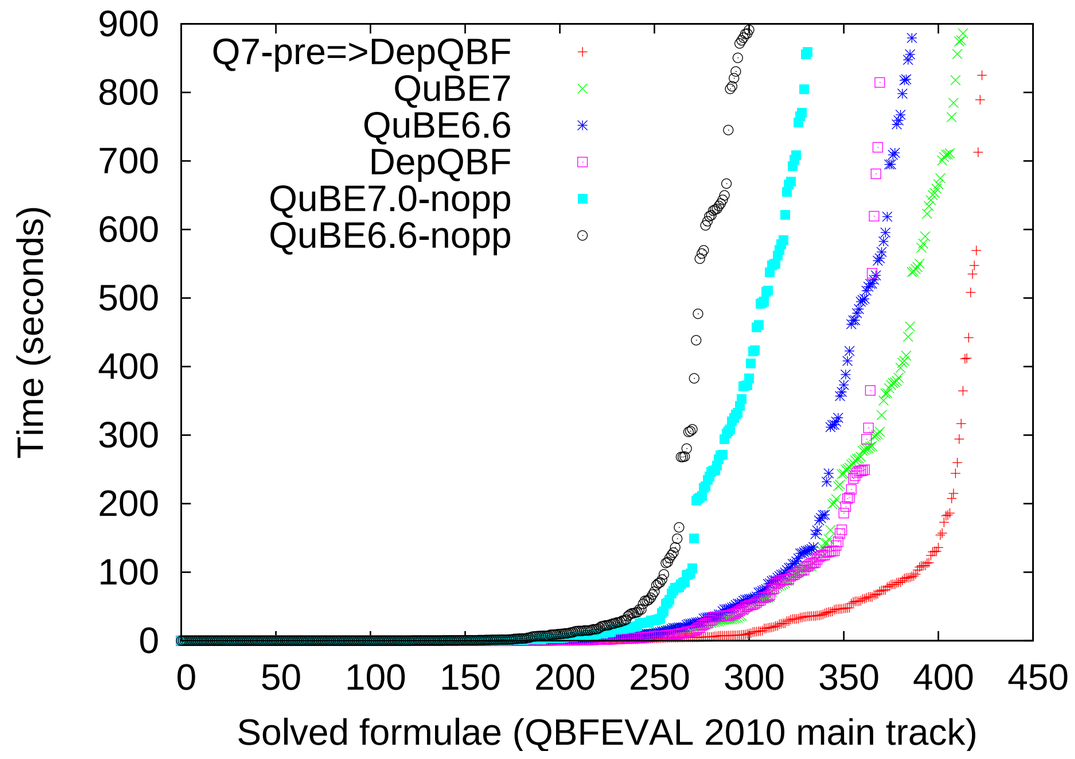
## Performance

| *QBFEVAL 2008 (3326 formulae)* | | | | | |
|---|---|---|---|---|---|
| | $D^{\text{triv}}$ | $D^{\text{tree}}$ | $D^{\text{std}}$ | QuBE6.6-nopp | QuBE6.6 |
| *solved* | 1223 | 1221 | **1252** | 1106 | **2277** |
| *avg. time* | 579.94 | 580.64 | **572.31** | 608.97 | **302.49** |

DepQBF (no preprocessing) with $D^{\text{triv}}$, $D^{\text{tree}}$, $D^{\text{std}}$ and QuBE6.6 with(out) preprocessing (-nopp).

| *QBFEVAL 2010 MAIN TRACK (568 formulae)* | | | | | | |
|---|---|---|---|---|---|---|
| | *All* | | *Solved SAT* | | *Solved UNSAT* | |
| | *solved* | *avg.time* | *solved* | *avg.time* | *solved* | *avg.time* |
| **Q7-pre⇒DepQBF** | **424** | **254.23** | **197** | **48.17** | **227** | **23.42** |
| QuBE7 | 414 | 310.29 | 187 | 130.52 | 227 | 58.33 |
| *without preprocessing* | | | | | | |
| **DepQBF** | **370** | **337.10** | **165** | **54.58** | **205** | **20.82** |
| DepQBF-nr | 360 | 352.33 | 154 | 51.36 | 206 | 24.35 |
| DepQBF-nc | 350 | 384.66 | 157 | 107.48 | 193 | 28.05 |
| DepQBF-ncnr | 340 | 400.24 | 147 | 124.10 | 193 | 20.19 |
| QuBE7.0-nopp | 332 | 425.44 | 135 | 147.71 | 197 | 47.27 |

DepQBF 0.1 using QuBE7.0 for preprocessing (Q7-pre), no restarts (-nr), no phase saving (-nc) and neither (-ncnr), and QuBE7.0 with(out) preprocessing (-nopp).




Source code: http://fmv.jku.at/depqbf/

A comprehensive list of references may be found in [1, 2].

## References

[1] F. Lonsing and A. Biere. DepQBF: A Dependency-Aware QBF Solver. In *Pragmatics of SAT (POS) Workshop*, 2010.

[2] F. Lonsing and A. Biere. Integrating Dependency Schemes in Search-Based QBF Solvers. In *In Proc. SAT*, 2010.