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

QBFEVAL’10: Overall Score-Based Ranking

Solver	*Score*
DepQBF	**2896.68**
DepQBF-pre	2508.96
aqme-10	2467.96
qmaiga	2117.55
AIGSolve	2037.22
quantor-3.1	1235.14
struqs-10	947.83
nenofex-qbfeval10	829.11

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

This Talk:

- DepQBF 0.1 system overview.
- Selected features: restarts, removal of learnt constraints.
- Experimental evaluation.

Overview

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:

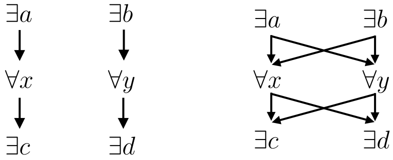
- $\forall x \exists y. (x = y)$ is satisfiable: value of y *depends* on value of x.
- $\exists y \forall x. (x = y)$ is unsatisfiable: value of y is fixed for all values of x.

Dependency Schemes

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^{std} as dependency-DAG over equivalence classes.
  - Efficient integration.

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



D^{std} D^{triv}

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

System Overview (1/7)

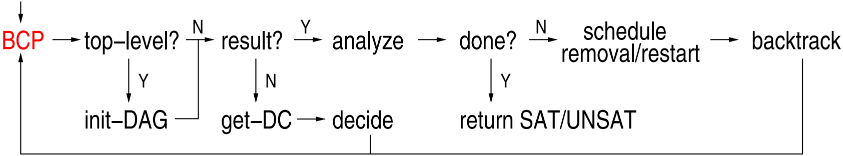


Figure: DepQBF workflow.

Boolean Constraint Propagation (BCP):

- Propagation of unit and pure literals.
- Watched data-structures for efficient detection.

System Overview (2/7)

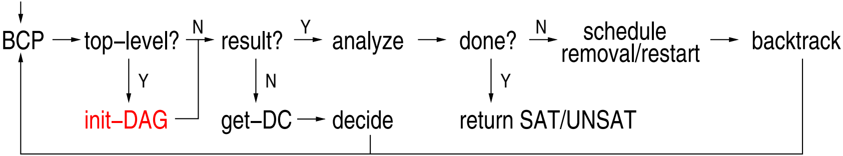


Figure: DepQBF workflow.

Initialize Dependency-DAG:

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

System Overview (3/7)

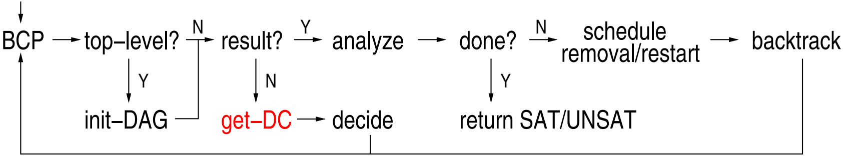


Figure: DepQBF workflow.

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.

System Overview (4/7)

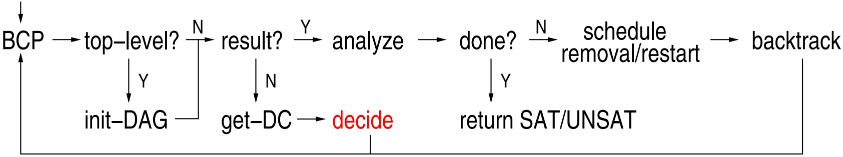


Figure: DepQBF workflow.

Decision Making:

- Select decision variable from candidate set.
- Activity-based priority queue of variables (VSIDS, like MiniSAT 2).
- Assignment caching.

System Overview (5/7)

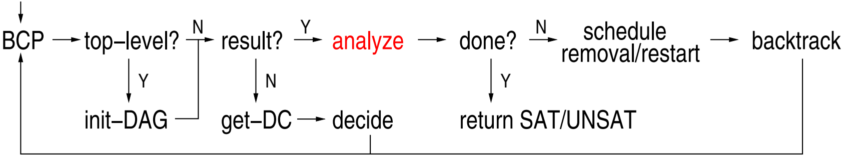


Figure: DepQBF workflow.

Constraint Learning (Result Analysis):

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

System Overview (6/7)

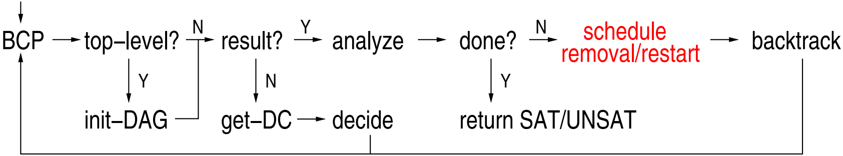


Figure: DepQBF workflow.

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).

System Overview (7/7)

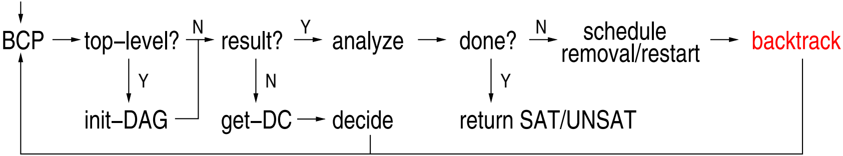


Figure: DepQBF workflow.

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 ϕ_{LCL} and cubes ϕ_{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:

$$\{C_1, \dots, C_{i-1}, C_i, C_{i+1}, \dots, C_n\} \overset{MTF(C_i)}{\Longrightarrow} \{C_i, C_1, \dots, C_{i-1}, C_{i+1}, \dots, C_n\}$$

most-recently used least-recently used ⟵ deletion order ↩

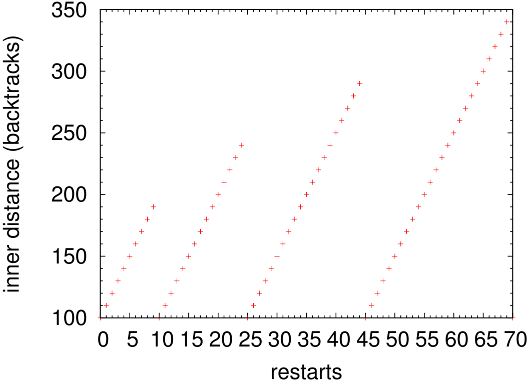
Deletion:

- 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.

Restarts (1/2)

Inner-Outer Restart Schedule: when to restart? [Bie08]

- 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).



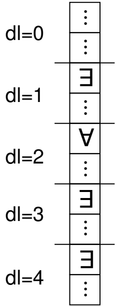
Restarts (2/2)

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:

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



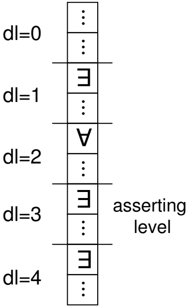
Restarts (2/2)

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.



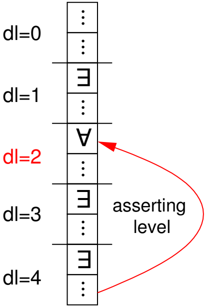
Restarts (2/2)

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.



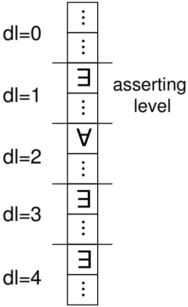
Restarts (2/2)

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.



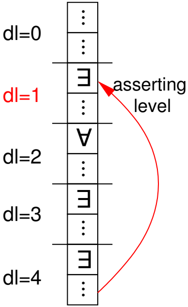
Restarts (2/2)

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.



Experiments: QBFEVAL'10 Main Track

	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
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-np	345	398.12	141	114.72	204	45.37
DepQBF-ncnr	340	400.24	147	124.10	193	20.19
QuBE7.0-nopp	332	425.44	135	147.71	197	47.27
QuBE6.6-nopp	301	468.51	113	136.48	188	55.27

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.

Experiments: QBFEVAL'10 Main Track

	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
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-np	345	398.12	141	114.72	204	45.37
DepQBF-ncnr	340	400.24	147	124.10	193	20.19
QuBE7.0-nopp	332	425.44	135	147.71	197	47.27
QuBE6.6-nopp	301	468.51	113	136.48	188	55.27

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

Important:

- Restarts (disabled in DepQBF-nr).

Experiments: QBFEVAL'10 Main Track

	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
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-np	345	398.12	141	114.72	204	45.37
DepQBF-ncnr	340	400.24	147	124.10	193	20.19
QuBE7.0-nopp	332	425.44	135	147.71	197	47.27
QuBE6.6-nopp	301	468.51	113	136.48	188	55.27

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

Important:

- Restarts.
- Assignment caching (disabled in DepQBF-nc).

Experiments: QBFEVAL'10 Main Track

	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
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-np	**345**	**398.12**	**141**	**114.72**	**204**	**45.37**
DepQBF-ncnr	340	400.24	147	124.10	193	20.19
QuBE7.0-nopp	332	425.44	135	147.71	197	47.27
QuBE6.6-nopp	301	468.51	113	136.48	188	55.27

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

Important:

- Restarts.
- Assignment caching.
- Pure literal detection (disabled in DepQBF-np).

Experiments: QBFEVAL'10 Main Track

	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
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-np	345	398.12	141	114.72	204	45.37
DepQBF-ncnr	**340**	**400.24**	**147**	**124.10**	**193**	**20.19**
QuBE7.0-nopp	332	425.44	135	147.71	197	47.27
QuBE6.6-nopp	301	468.51	113	136.48	188	55.27

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).

Experiments: QBFEVAL’10 Main Track

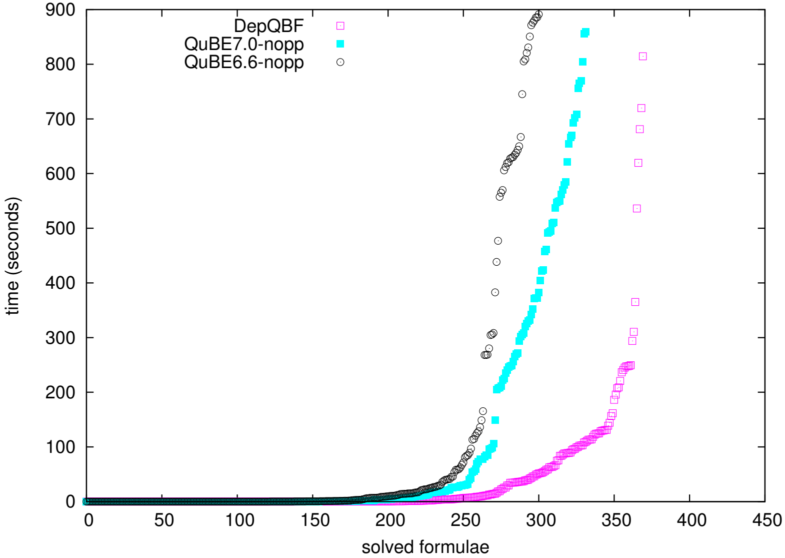
	All		*Solved SAT*		*Solved UNSAT*	
	solved	*avg.time*	*solved*	*avg.time*	*solved*	*avg.time*
QuBE7.0-pre⇒DepQBF	**424**	**254.23**	**197**	**48.17**	**227**	**23.42**
QuBE7	**414**	**310.29**	**187**	**130.52**	**227**	**58.33**
QuBE6.6	**387**	**341.91**	**168**	**98.97**	**219**	**67.03**
without 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-np	345	398.12	141	114.72	204	45.37
DepQBF-ncnr	340	400.24	147	124.10	193	20.19
QuBE7.0-nopp	**332**	**425.44**	**135**	**147.71**	**197**	**47.27**
QuBE6.6-nopp	**301**	**468.51**	**113**	**136.48**	**188**	**55.27**

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

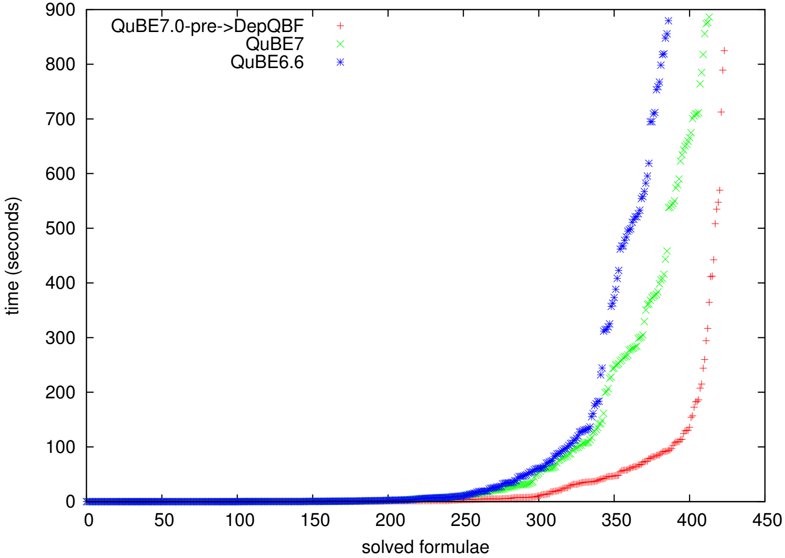
Important:

- Restarts.
- Assignment caching.
- Pure literal detection.
- Combining restarts with assignment caching.
- Preprocessing (**not** part of DepQBF 0.1, disabled in QuBE*-nopp).

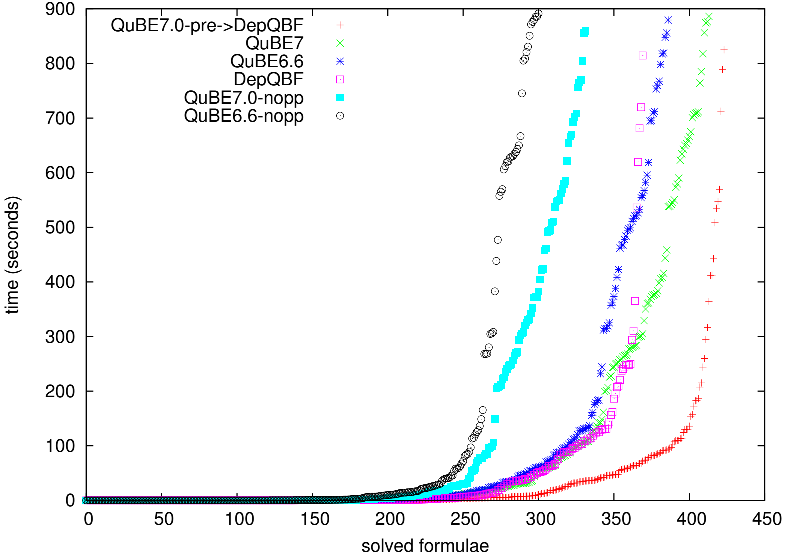
Experiments: QBFEVAL'10 Main Track



Experiments: QBFEVAL’10 Main Track



Experiments: QBFEVAL'10 Main Track



Conclusion

DepQBF: [BLB10]

- Search-based QBF solver with clause- and cube-learning.
- Relaxing prefix order by dependency-DAG for D^{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/`

[APPENDIX] – BCP: Unit Literals

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. $Var(l_u)$, $Var(l_e)$ 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.

[APPENDIX] – BCP: Pure Literals (1/2)

Pure Literals (PL): [CGS98, GGN$^+$03, GNT04]

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

Spurious Pure Literals (SPL):

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

Clause Watching:

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

[APPENDIX] – BCP: Pure Literals (2/2)

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.

[APPENDIX] – Variable Activities

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.

[APPENDIX] – Assignment Caching

Also called: Phase Saving [PD07]

- 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.

[APPENDIX] – Experiments: Suite *mqm*

Suite mqm (136 formulae)		
	solved	*avg.time*
DepQBF	136	39.83
QuBE7	117	306.43
QuBE7.0-nopp	115	304.82
QuBE6.6	100	393.93
QuBE6.6-nopp	97	399.55

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.

[APPENDIX] – Experiments: Pairwise Comparison

QBFEVAL'10: solved formulae only						
	∩		SAT-∩		UNSAT-∩	
solved	328		132		196	
avg.time	84.97	**21.87**	140.16	**32.43**	47.81	**14.75**
QBFEVAL'10: unique results						
	⇔		SAT-⇔		UNSAT-⇔	
solved	**86**	42	**55**	33	**31**	9

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

QBFEVAL'10: solved formulae only						
	∩		SAT-∩		UNSAT-∩	
solved	308		115		193	
avg.time	80.14	**17.49**	114.17	**23.23**	59.86	**14.07**
QBFEVAL'10: unique results						
	⇔		SAT-⇔		UNSAT-⇔	
solved	**79**	62	**53**	50	**26**	12

Table: QuBE6.6 (left columns) vs. DepQBF (right columns).

- M. Benedetti.
  Quantifier Trees for QBFs.
  In F. Bacchus and T. Walsh, editors, *SAT*, volume 3569 of *LNCS*, pages 378–385. Springer, 2005.

- A. Biere.
  PicoSAT Essentials.
  JSAT, 4(2-4):75–97, 2008.

- H. Kleine Büning, M. Karpinski, and A. Flögel.
  Resolution for Quantified Boolean Formulas.
  Inf. Comput., 117(1):12–18, 1995.

- 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.

- A. Bhalla, I. Lynce, J. T. de Sousa, and J. Marques-Silva.
  Heuristic-Based Backtracking Relaxation for Propositional Satisfiability.
  Journal of Automated Reasoning (JAR), 35(1-3):3–24, 2005.

- M. Cadoli, A. Giovanardi, and M. Schaerf.
  An Algorithm to Evaluate Quantified Boolean Formulae.
  In *AAAI/IAAI*, pages 262–267, 1998.

N. Eén and N. Sörensson.
An Extensible SAT-Solver.
In E. Giunchiglia and A. Tacchella, editors, *SAT*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.

I. P. Gent, E. Giunchiglia, M. Narizzano, A. G. D. Rowley, and A. Tacchella.
Watched Data Structures for QBF Solvers.
In E. Giunchiglia and A. Tacchella, editors, *SAT*, volume 2919 of *LNCS*, pages 25–36. Springer, 2003.

E. Goldberg and Y. Novikov.
BerkMin: A Fast and Robust SAT-Solver.
In *Proc. DATE*, pages 142–149. IEEE Computer Society, 2002.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Learning for Quantified Boolean Logic Satisfiability.
In *AAAI/IAAI*, pages 649–654, 2002.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Monotone Literals and Learning in QBF Reasoning.
In M. Wallace, editor, *CP*, volume 3258 of *LNCS*, pages 260–273. Springer, 2004.

E. Giunchiglia, M. Narizzano, and A. Tacchella.

Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas.
J. Artif. Intell. Res. (JAIR), 26:371–416, 2006.

E. Giunchiglia, M. Narizzano, and A. Tacchella.
Quantifier Structure in Search-Based Procedures for QBFs.
TCAD, 26(3):497–507, 2007.

R. Gershman and O. Strichman.
HaifaSat: A SAT Solver Based on an Abstraction/Refinement Model.
Journal on Satisfiability, Boolean Modeling and Computation, 6:33–51, 2008.

F. Lonsing and A. Biere.
A Compact Representation for Syntactic Dependencies in QBFs.
In O. Kullmann, editor, *SAT*, volume 5584 of *LNCS*, pages 398–411. Springer, 2009.

F. Lonsing and A. Biere.
Integrating Dependency Schemes in Search-Based QBF Solvers.
In O. Strichman and S. Szeider, editors, *SAT (accepted for publication)*, LNCS. Springer, 2010.

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

In U. Egly and C. G. Fermüller, editors, *TABLEAUX*, volume 2381 of *LNCS*, pages 160–175. Springer, 2002.

M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik.
Chaff: Engineering an Efficient SAT Solver.
In *DAC*, pages 530–535. ACM, 2001.

K. Pipatsrisawat and A. Darwiche.
A Lightweight Component Caching Scheme for Satisfiability Solvers.
In J. Marques-Silva and K. A. Sakallah, editors, *SAT*, volume 4501 of *LNCS*, pages 294–299. Springer, 2007.

M. Samer and S. Szeider.
Backdoor Sets of Quantified Boolean Formulas.
Journal of Automated Reasoning (JAR), 42(1):77–97, 2009.

L. Zhang and S. Malik.
Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation.
In P. Van Hentenryck, editor, *CP*, volume 2470 of *LNCS*, pages 200–215. Springer, 2002.