Enhancing Search-Based QBF Solving by Dynamic Blocked Clause Elimination

Florian Lonsing[1] Fahiem Bacchus[2] Armin Biere[3]
Uwe Egly[1] Martina Seidl[3]

[1]Knowledge-Based Systems Group, Vienna University of Technology, Austria

[2]Department of Computer Science, University of Toronto, Canada

[3]Institute for Formal Models and Verification, JKU Linz, Austria

20th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, 24 - 28 November, 2015, Suva, Fiji

This work is supported by the Austrian Science Fund (FWF) under grants S11408-N23 and S11409-N23.

Introduction (1)

Quantified Boolean Formulas (QBF):

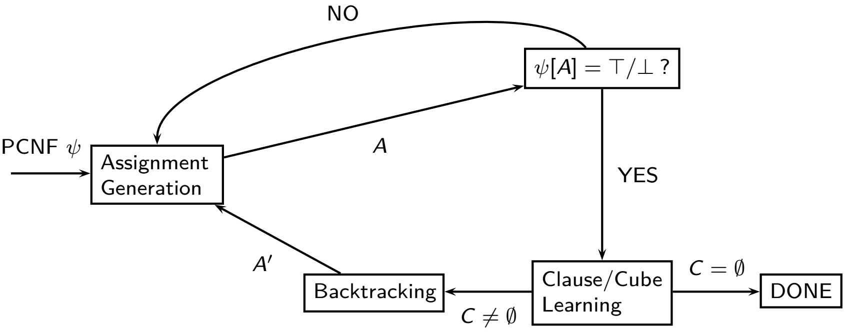
- Propositional logic with explicitly existentially/universally quantified variables.
- PSPACE-completeness: applications in AI, verification, synthesis,...

QBFs in Prenex CNF:

- $\psi := \hat{Q}.\phi$: quantifier prefix $\hat{Q}$ and propositional formula ϕ in CNF.
- $\hat{Q} := Q_1 v_1 \ldots Q_n v_n$ with $Q \in \{\forall, \exists\}$ and variables v_i, left-to-right ordering.

QBF Semantics:

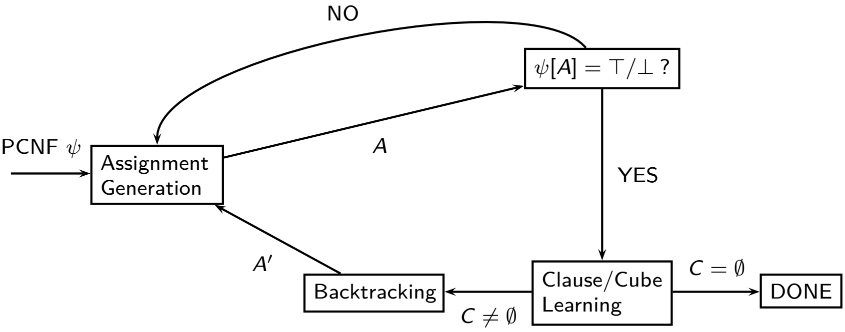
- Recursive instantiation of variables in prefix ordering.
- $\exists x \hat{Q}'.\phi$ is satisfiable iff $\hat{Q}'.\phi[x/\bot]$ or $\hat{Q}'.\phi[x/\top]$ is satisfiable.
- $\forall x \hat{Q}'.\phi$ is satisfiable iff $\hat{Q}'.\phi[x/\bot]$ and $\hat{Q}'.\phi[x/\top]$ is satisfiable.



Search-Based QBF Solving:

- QBF-specific variant of DPLL algorithm.
- Generation of variable assignments (implicit traversal of assignment tree).
- Case splitting and backtracking based on $\forall/\exists$ semantics.
- Given current assignment A, backtrack if PCNF $\psi[A] = \bot$ or $\psi[A] = \top$.
- Clause and cube learning: *QCDCL*.

Introduction (2)

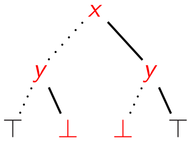


Conflicting assignment: $\psi[A] = \bot$, at least one clause falsified under A.

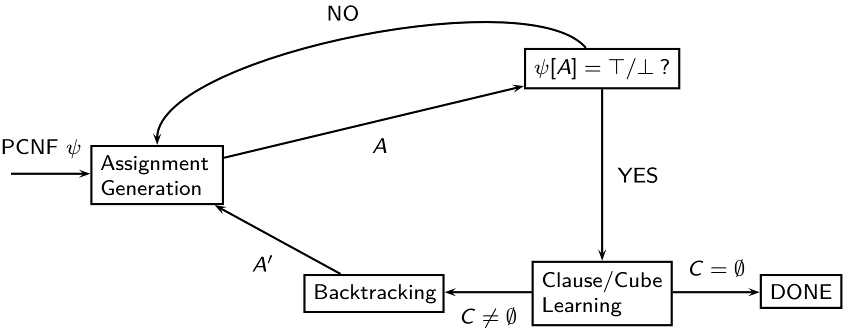
Example: $\psi := \forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$

Left branch: $\psi[\{x \mapsto \bot, y \mapsto \top\}] = \bot$
Right branch: $\psi[\{x \mapsto \top, y \mapsto \bot\}] = \bot$

Assignment tree related to ψ:



Introduction (2)

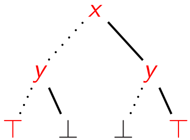


Satisfying assignment: $\psi[A] = \top$, all clauses of CNF are satisfied under A.

Example: $\psi := \forall x \exists y.(x \vee \bar{y}) \wedge (\bar{x} \vee y)$

Assignment tree related to ψ:

Left branch: $\psi[\{x \mapsto \bot, y \mapsto \bot\}] = \top$
Right branch: $\psi[\{x \mapsto \top, y \mapsto \top\}] = \top$



Introduction (3)

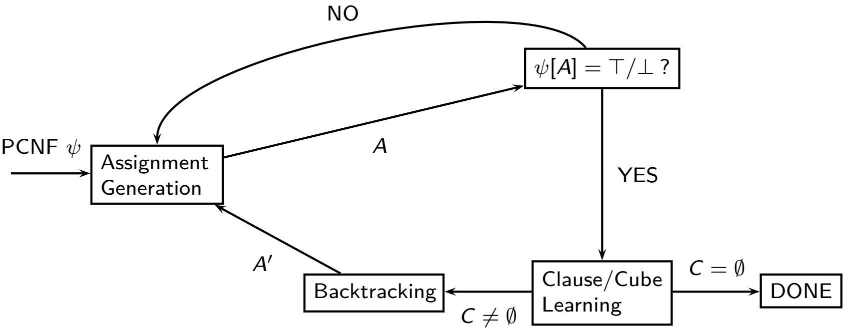
Observation:

- Uniformity of CNF representation allows for efficient data structures.
- Problem: CNF introduces a bias towards detecting conflicting assignments.
- In general, detecting satisfying assignments involves assigning more variables.

Idea:

- Detect satisfying assignments earlier to backtrack earlier.
- Generalization: "assignment A is satisfying iff PCNF $\psi[A]$ is satisfiable".
- $\psi[A] = \top$ no longer required, i.e. some clauses may not be satisfied under A.
- Problem: how to *efficiently* detect whether $\psi[A]$ is satisfiable?

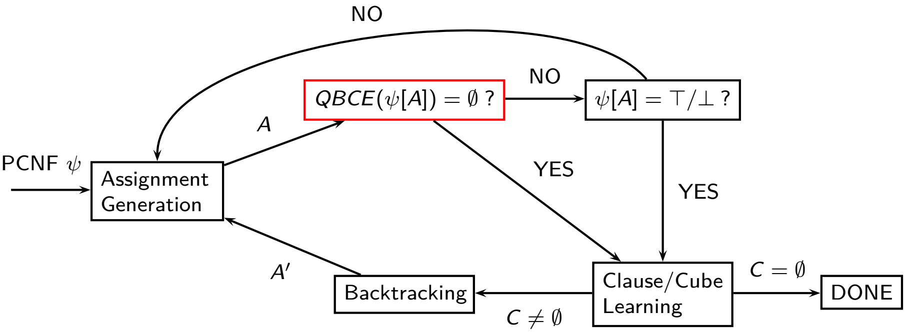
Contributions



Blocked Clause Elimination for QBF (QBCE)

- Satisfiability-preserving elimination of certain clauses from a PCNF.
- So far, QBCE has been applied for PCNF preprocessing only.

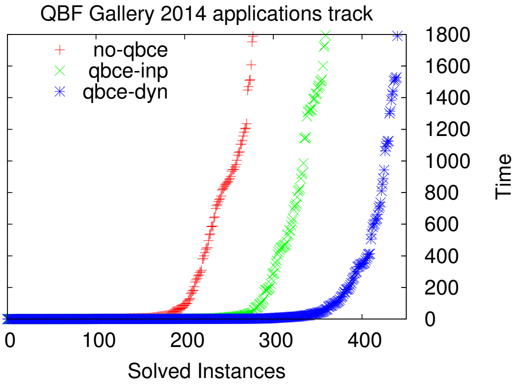
Contributions



Search-Based QBF Solving with Dynamic QBCE:

- QBCE interleaved with assignment generation.
- Incomplete polynomial-time detection of generalized satisfying assignments.
- If QBCE eliminates all clauses of $\psi[A]$, then $\psi[A]$ is satisfiable.

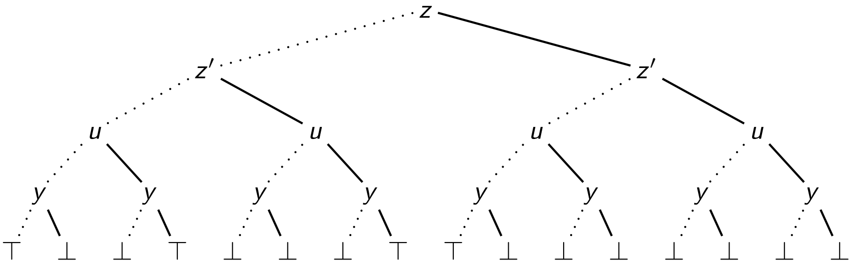
Contributions



Experimental Results:

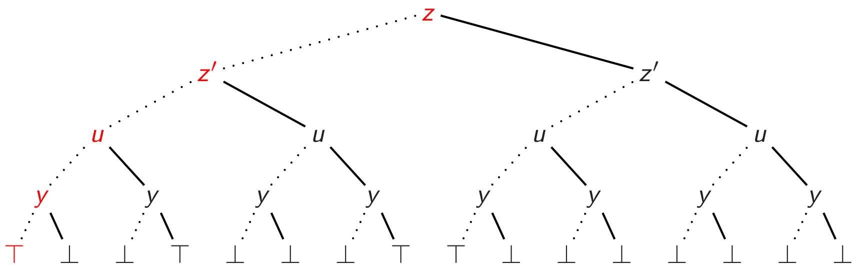
- Dynamic QBCE in search-based QBF solver DepQBF (version 5.0).
- 58% more application instances solved with dynamic QBCE.
- Full preprocessing may affect our approach negatively.

Example (1)



$\exists z, z' \forall u \exists y.$
$(u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z \vee u \vee \bar{y}) \wedge (z' \vee \bar{u} \vee y) \wedge (\bar{z} \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}' \vee u \vee y)$

Example (2)



$\exists z, z' \forall u \exists y.$
$(u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z \vee u \vee \bar{y}) \wedge (z' \vee \bar{u} \vee y) \wedge (\bar{z} \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}' \vee u \vee y)$

- Consider satisfying assignment $A = \{z \mapsto \bot, z' \mapsto \bot, u \mapsto \bot, y \mapsto \bot\}$.
- Derive a cube (conjunction of literals) from A and ψ.
- After backtracking, cube helps to prevent repeating (subset of) A.
- *Solution driven cube learning (SDCL)* in search-based QBF solving.

Cube Learning as a Proof System (1)

Let $\psi = \hat{Q}.\phi$ be a PCNF.

Axiom (Model Generation):

$$\frac{}{\mathrm{C}} \quad \begin{array}{l} C = (\bigwedge_{l \in A}) \text{ is a cube where } \{x, \bar{x}\} \not\subseteq C \text{ and } A \text{ is an assignment} \\ \text{with } \psi[A] = \top \end{array} \qquad (init)$$

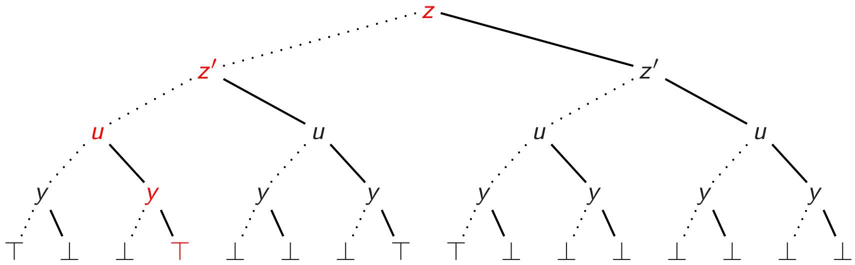
- Every clause of ψ is satisfied under A: $\psi[A] = \top$.
- Cube C is constructed from A such that $v \in C$ if $v \mapsto \top$ and $\bar{v} \in C$ if $v \mapsto \bot$.
- C is a propositional implicant of the CNF part ϕ: $C \Rightarrow \phi$.
- Several heuristics applicable when constructing C.

E. Giunchiglia, M. Narizzano, A. Tacchella: Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. JAIR 2006.

R. Letz: Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. TABLEAUX 2002.

L. Zhang, S. Malik: Towards a Symmetric Treatment of Satisfaction and Conflicts in Quantified Boolean Formula Evaluation. CP 2002.

Example (3)



$\exists z, z' \forall u \exists y.$
$(u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z \vee u \vee \bar{y}) \wedge (z' \vee \bar{u} \vee y) \wedge (\bar{z} \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}' \vee u \vee y)$

- Derive the cube $C = (\bar{z} \wedge \bar{z}' \wedge \bar{u} \wedge \bar{y})$ by model generation, backtrack and obtain the satisfying assignment $A' = \{z \mapsto \bot, z' \mapsto \bot, u \mapsto \top, y \mapsto \top\}$.
- Derive another cube $C' = (\bar{z} \wedge \bar{z}' \wedge u \wedge y)$ by model generation.
- Observe: both subcases of the universal variable u are satisfiable.

Cube Learning as a Proof System (2)

Let $\psi = \hat{Q}.\phi$ be a PCNF.

Resolution:

$$\frac{C_1 \cup \{p\} \qquad C_2 \cup \{\bar{p}\}}{C_1 \cup C_2} \quad \begin{array}{l} C_1, C_2 \text{ are cubes, quant}(p) = \forall, \\ \{x, \bar{x}\} \not\subseteq (C_1 \cup C_2),\ \bar{p} \notin C_1,\ p \notin C_2 \end{array} \qquad (res)$$

Reduction:

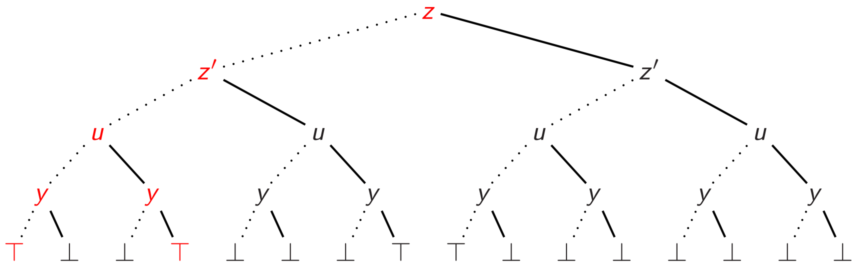
$$\frac{C \cup \{l\}}{C} \quad \begin{array}{c} C \text{ is a cube, quant}(l) = \exists,\ \{x, \bar{x}\} \not\subseteq (C \cup \{l\}), \\ l' <_{\hat{Q}'} l \text{ for all } l' \in C \text{ with quant}(l') = \forall \end{array} \qquad (red)$$

- A PCNF ψ is satisfiable iff the empty cube is derivable by rules *init*, *res*, *red*.

Example (continued):
$\exists z, z' \forall u \exists y. \phi$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(\bar{z} \wedge \bar{z}' \wedge \bar{u} \wedge \bar{y})}}{(\bar{z} \wedge \bar{z}' \wedge \bar{u})} \qquad \dfrac{\dfrac{}{(\bar{z} \wedge \bar{z}' \wedge u \wedge y)}}{(\bar{z} \wedge \bar{z}' \wedge u)}}{\bar{z} \wedge \bar{z}'}}{\emptyset}$$

Example (4)



$\exists z, z' \forall u \exists y.$
$(u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z \vee u \vee \bar{y}) \wedge (z' \vee \bar{u} \vee y) \wedge (\bar{z} \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}' \vee u \vee y)$

- Learned cubes represent paths in search tree.
- Cubes are derived starting from leaves in bottom up fashion.
- Consider cube resolvent $(\bar{z} \wedge \bar{z}')$: $\psi[z \mapsto \bot, z' \mapsto \bot]$ is satisfiable.
- A posteriori analysis: no need to inspect subtree rooted at branch $\bar{z}, \bar{z}'$.

Generalized Model Generation

Let $\psi = \hat{Q}.\phi$ be a PCNF.

Axiom (Generalized Model Generation):

$$\frac{}{\mathrm{C}} \quad \begin{array}{l} C = (\bigwedge_{l \in A}) \text{ is a cube where } \{x, \bar{x}\} \not\subseteq C \text{ and } A \text{ is an assignment} \\ \text{such that } \psi[A] \text{ is satisfiable} \end{array} \qquad (ginit)$$

- Caution: assignment A must have certain properties.
- Some clauses of ψ may not be satisfied under A: $\psi[A] \neq \top$.
- C is not a propositional implicant of the CNF part ϕ: $C \not\Rightarrow \phi$.
- Generalized model generation potentially derives shorter cubes.
- How to *efficiently* check whether $\psi[A]$ is satisfiable? PSPACE-completeness!

Example (continued):
$\exists z, z' \forall u \exists y.\phi[\{z \mapsto \bot, z' \mapsto \bot\}] =$
$\forall u \exists y.(u \vee \bar{y}) \wedge (\bar{u} \vee y)$ satisfiable

$$\frac{\overline{\bar{z} \wedge \bar{z}'}}{\emptyset}$$

Dynamic Blocked Clause Elimination for QBF

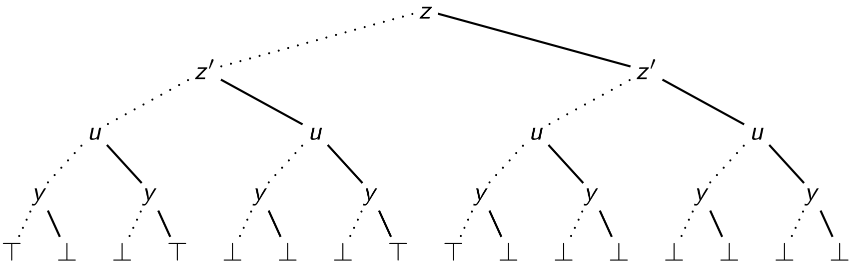
Blocked Clause Elimination for QBF (QBCE):

- A clause C is blocked if it contains an existential *blocking literal* l.
- Finding blocking literals l: inspect all clauses C' with $\neg l \in C'$.
- QBCE can be carried out in polynomial time wrt. formula size.
- QBCE preserves satisfiability.

Dynamic QBCE:

- Interleave QBCE with search process.
- Incremental application based on extended assignments: $\psi[A]$, $\psi[A \cup A']$, $\ldots$
- If QBCE eliminates all clauses of $\psi[A]$ then $\psi[A]$ is satisfiable.

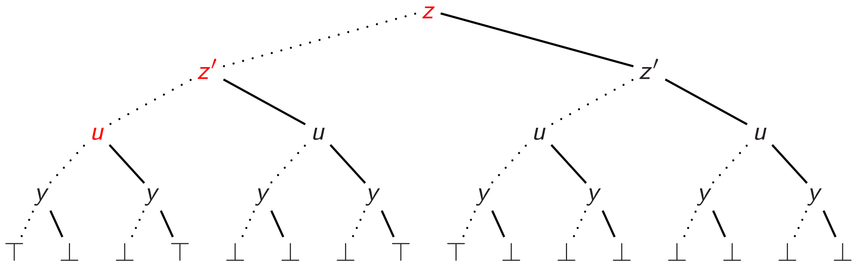
Dynamic QBCE Example



$\exists z, z' \forall u \exists y.$
$(u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (z \vee u \vee \bar{y}) \wedge (z' \vee \bar{u} \vee y) \wedge (\bar{z} \vee \bar{u} \vee \bar{y}) \wedge (\bar{z}' \vee u \vee y)$

- Consider initial assignment $A = \emptyset$ and $\psi[A]$.
- No clause blocked in $\psi[A]$.

Dynamic QBCE Example



$\exists z, z' \forall u \exists y.$
$(u \vee \bar{y}) \wedge (\bar{u} \vee y) \wedge (\bot \vee u \vee \bar{y}) \wedge (\bot \vee \bar{u} \vee y) \wedge (\top \vee \bar{u} \vee \bar{y}) \wedge (\top \vee u \vee y)$

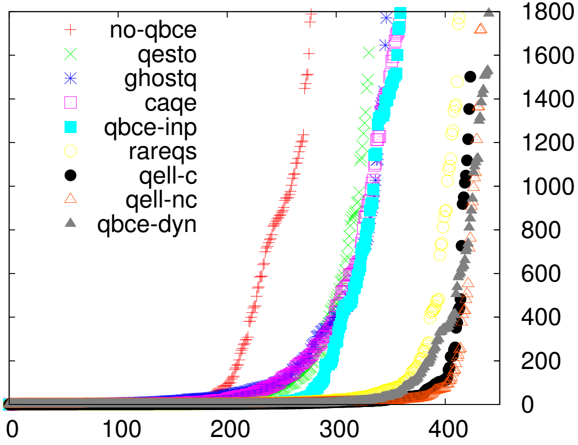
- Consider $A = \{z \mapsto \bot, z' \mapsto \bot\}$ and $\psi[A] = \forall u \exists y.(u \vee \bar{y}) \wedge (\bar{u} \vee y)$.
- All clauses are blocked in $\psi[A]$, hence $\psi[A]$ is satisfiable.
- By generalized model generation, learn cube $C = (\bar{z} \wedge \bar{z}')$ and finally $\emptyset$.
- Observe: we proved satisfiability without considering $\forall u$ during search.

Experiments (1)

Solver	Solved	Unsat	Sat	Time
qbce-dyn	441	222	219	573,142
qell-nc	434	302	132	563,989
qell-c	424	300	124	577,760
rareqs	414	272	142	611,742
qbce-inp	360	161	199	735,073
caqe	359	197	162	750,173
ghostq	347	166	181	752,950
qesto	331	188	143	767,757
no-qbce	278	128	150	880,485

- Implementation in search-based QBF solver DepQBF.
- Application benchmarks used in the QBF Gallery 2014 *without* preprocessing.
- no-qbce: plain DepQBF without dynamic QBCE.
- qbce-inp: DepQBF with restricted dynamic QBCE.
- qbce-dyn: DepQBF with fully dynamic QBCE.
- Comparison to recently published solvers: caqe, qell, qesto.

Experiments (1): Runtime



- Solved instances (x-axis) sorted by run times (y-axis).

Experiments (2)

Solver	Solved	Unsat	Sat	Time
rareqs	547	314	233	379,916
qell-nc	501	301	200	445,369
qell-c	495	299	196	452,034
qesto	463	248	215	558,703
qbce-dyn	405	201	204	624,719
caqe	395	191	204	647,227
no-qbce	390	205	185	651,909
qbce-inp	390	205	185	655,329
ghostq	350	176	174	739,294

- Application benchmarks *with* preprocessing.
- Full preprocessing by Bloqqer (none solved): `http://fmv.jku.at/bloqqer/`
- Preprocessing may blur formula structure and thus hinder dynamic QBCE.

Experiments (3)

Solver	Solved	Unsat	Sat	Time
qell-nc	483	306	177	480,736
qell-c	474	308	166	494,281
rareqs	471	272	199	509,489
qbce-dyn	463	243	220	533,829
caqe	435	226	209	585,618
qesto	401	212	189	662,695
no-qbce	400	221	179	651,739
qbce-inp	393	219	174	657,400
ghostq	306	148	158	823,312

- Application benchmarks *with partial* preprocessing.
- Only QBCE and expansion of universal variables.
- Moderate performance improvement of dynamic QBCE.
- Mostly detrimental to other solvers.

Conclusion

CNF-based QBF Solving:

- Falsifying assignments detected more easily than satisfying ones.
- Initially, long cubes are learned (propositional implicants of CNF).

Generalized Model Generation:

- Detect satisfiable subtrees early, learn shorter cubes (no implicants).
- Exponentially more powerful proof system.
- Dynamic QBCE: incomplete polynomial time satisfiability check.

Future Work:

- Combination of dynamic QBCE and preprocessing.

DepQBF version 5.0: `http://lonsing.github.io/depqbf/`