Solving QBF with Free Variables

Will Klieber, Mikoláš Janota,
Joao Marques-Silva, Edmund Clarke

Sep 17, 2013

Quantified Boolean Formulas (QBF)

- Extension of propositional logic. Grammar:

$$
\begin{array}{rcl}
x & ::= & \text{boolean variable} \\
Q & ::= & \exists \mid \forall \\
\Phi & ::= & x \mid Qx.\ \Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi \mid \text{True} \mid \text{False}
\end{array}
$$

Quantified Boolean Formulas (QBF)

- Extension of propositional logic. Grammar:

$$
\begin{array}{rcl}
x & ::= & \text{boolean variable} \\
Q & ::= & \exists \mid \forall \\
\Phi & ::= & x \mid Qx.\ \Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi \mid \text{True} \mid \text{False}
\end{array}
$$

- No variable may be quantified more than once.
- No variable may occur both free and bound.

Quantified Boolean Formulas (QBF)

- Extension of propositional logic. Grammar:

$$
\begin{array}{rcl}
x & ::= & \text{boolean variable} \\
Q & ::= & \exists \mid \forall \\
\Phi & ::= & x \mid Qx.\ \Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \neg\Phi \mid \mathsf{True} \mid \mathsf{False}
\end{array}
$$

- No variable may be quantified more than once.
- No variable may occur both free and bound.
- Semantics:
  - $\forall x.\, \Phi \;=\; \Phi|_{x=\mathsf{True}} \wedge \Phi|_{x=\mathsf{False}}$
  - $\exists x.\, \Phi \;=\; \Phi|_{x=\mathsf{True}} \vee \Phi|_{x=\mathsf{False}}$

Open QBF

- Closed QBF: All variables quantified; answer is True or False.
- Open QBF: Contains free (unquantified) variables.
- Goal: Find equivalent propositional formula.
- E.g., given $\exists x.\, x \wedge (y \vee z)$, return $y \vee z$.

Open QBF

- Closed QBF: All variables quantified; answer is True or False.
- Open QBF: Contains free (unquantified) variables.
- Goal: Find equivalent propositional formula.
- E.g., given $\exists x.\, x \wedge (y \vee z)$, return $y \vee z$.
- Applications: symbolic model checking,
  synthesis from formal spec, etc.

Outline

- Naïve Algorithm
- Introduce *sequents* that generalize clauses for open QBF (without ghost variables)
- Experimental results
- *Ghost variables* (for non-CNF): see paper.

Naïve Algorithm

- Notation: "$\mathrm{ite}(x, \phi_1, \phi_2)$" is a formula with an *if-then-else*:

$$\mathrm{ite}(x, \phi_1, \phi_2) \;=\; (x \wedge \phi_1) \vee (\neg x \wedge \phi_2)$$

Naïve Algorithm

- Notation: "$\mathrm{ite}(x, \phi_1, \phi_2)$" is a formula with an *if-then-else*:

$$\mathrm{ite}(x, \phi_1, \phi_2) \;=\; (x \wedge \phi_1) \vee (\neg x \wedge \phi_2)$$

- Recursively Shannon-expand on free variables:

$$\Phi \;=\; \mathrm{ite}(x,\; \Phi|_{x=\mathsf{True}},\; \Phi|_{x=\mathsf{False}})$$

Naïve Algorithm

- Notation: "$\mathrm{ite}(x, \phi_1, \phi_2)$" is a formula with an *if-then-else*:

$$\mathrm{ite}(x, \phi_1, \phi_2) \;=\; (x \wedge \phi_1) \vee (\neg x \wedge \phi_2)$$

- Recursively Shannon-expand on free variables:

$$\Phi \;=\; \mathrm{ite}(x,\; \Phi|_{x=\mathsf{True}},\; \Phi|_{x=\mathsf{False}})$$

- Base case (no more free variables): Give to closed-QBF solver.

Naïve Algorithm

```
1.   function solve(Φ) {
2.       if (Φ has no free variables)
3.           return closed_qbf_solve(Φ);




7.   }
```

Naïve Algorithm

```
1.   function solve(Φ) {
2.       if (Φ has no free variables)
3.           return closed_qbf_solve(Φ);
4.       x := (a free variable in Φ);
5.       return ite(x, solve( Φ|x=True ),
6.                     solve( Φ|x=False ));
7.   }
```

Naïve Algorithm

```
1.   function solve(Φ) {
2.       if (Φ has no free variables)
3.           return closed_qbf_solve(Φ);
4.       x := (a free variable in Φ);
5.       return ite(x, solve( Φ|x=True ),
6.                     solve( Φ|x=False ));
7.   }
```

Builds OBDD if:

1. same branch order,
2. formula construction is memoized, and
3. $\mathrm{ite}(x,\ \phi,\ \phi)$ is simplified to ϕ.

Naïve Algorithm

- Naïve Algorithm:
  - Similar to DPLL in terms of branching.
  - But lacks many optimizations that make DPLL fast:
    - Non-chronological backtracking
    - Clause learning
- Our open-QBF technique:
  - Extend existing closed-QBF algorithm to allow free variables.

Preliminaries

- **Prenex Form:** $Q_1 x_1 ... Q_n x_n . \phi$ where ϕ has no quantifiers.

Preliminaries

- **Prenex Form:** $Q_1 x_1 ... Q_n x_n . \phi$ where ϕ has no quantifiers.
- In $\forall x . \exists y . \phi$, we say that y is **downstream** of x.
  - $\exists y$ occurs inside scope of $\forall x$.

Preliminaries

- **Prenex Form:** $Q_1 x_1 ... Q_n x_n . \phi$ where ϕ has no quantifiers.
- In $\forall x . \exists y . \phi$, we say that y is **downstream** of x.
  - $\exists y$ occurs inside scope of $\forall x$.
- Free variables are upstream of all quantified variables.

Preliminaries

- **Prenex Form:** $Q_1 x_1 ... Q_n x_n . \phi$ where ϕ has no quantifiers.
- In $\forall x . \exists y . \phi$, we say that y is **downstream** of x.
  - $\exists y$ occurs inside scope of $\forall x$.
- Free variables are upstream of all quantified variables.
- **Outermost:** Not downstream of any unassigned variables.
  - E.g.: $\exists e_1 . \forall u_2 . \phi$ and assignment $\{(e_1, \text{True})\}$: u_2 is outermost.

Preliminaries

- **Prenex Form:** $Q_1 x_1 ... Q_n x_n . \phi$ where ϕ has no quantifiers.
- In $\forall x . \exists y . \phi$, we say that y is **downstream** of x.
  - $\exists y$ occurs inside scope of $\forall x$.
- Free variables are upstream of all quantified variables.
- **Outermost:** Not downstream of any unassigned variables.
  - E.g.: $\exists e_1 . \forall u_2 . \phi$ and assignment $\{(e_1, \text{True})\}$: u_2 is outermost.
- **Substitution:** $\Phi|_\pi$ where π is a partial assignment.

Closed QBF as a Game

- Existential variables are **owned** by Player $\exists$.
- Universal variables are **owned** by Player $\forall$.
- Players assign variables in quantification order.
- The **goal** of Player $\exists$ is to make Φ be true.
- The **goal** of Player $\forall$ is to make Φ be false.

Properties of Clauses and Cubes

- Motivate definition of sequents.
- Existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(\underbrace{e_1 \vee \ldots \vee e_n \vee u_1 \vee \ldots \vee u_m}_{\text{all false (under } \pi)})$ in CNF Φ_{in}

Properties of Clauses and Cubes

- Motivate definition of sequents.
- Existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(\underbrace{e_1 \vee \ldots \vee e_n \vee u_1 \vee \ldots \vee u_m}_{\text{all false (under } \pi)})$ in $\underbrace{\text{CNF } \Phi_{in}}_{\text{false}}$

Properties of Clauses and Cubes

- Motivate definition of sequents.
- Existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(\underbrace{e_1 \vee \ldots \vee e_n}_{\text{all false}} \vee \underbrace{u_1 \vee \ldots \vee u_m}_{\text{none true}})$ in $\underbrace{\text{CNF } \Phi_{in}}_{\Rightarrow \text{ false}}$

Properties of Clauses and Cubes

- Motivate definition of sequents.
- Existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(\underbrace{e_1 \vee \ldots \vee e_n}_{\text{all false}} \vee \underbrace{u_1 \vee \ldots \vee u_m}_{\text{none true}})$ in $\underbrace{\text{CNF } \Phi_{in}}_{\Rightarrow \text{ false}}$
- Cube $(\underbrace{u_1 \wedge \ldots \wedge u_n}_{\text{all true}} \wedge \underbrace{e_1 \wedge \ldots \wedge e_m}_{\text{none false}})$ in $\underbrace{\text{DNF } \Phi_{in}}_{\Rightarrow \text{ true}}$

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.
- **Definition.** We say that $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ **matches** assignment π iff:
  1. every literal in L^{now} evaluates to True under π, and
  2. no literal in L^{fut} evaluates to False under π.

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.
- **Definition.** We say that $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ **matches** assignment π iff:
  1. every literal in L^{now} evaluates to True under π, and
  2. no literal in L^{fut} evaluates to False under π.
- E.g., $\langle \{e\}, \{u\} \rangle$ matches $\{(e, \text{True})\}$ and $\{(e, \text{True}), (u, \text{True})\}$,

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.
- **Definition.** We say that $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ **matches** assignment π iff:
  1. every literal in L^{now} evaluates to True under π, and
  2. no literal in L^{fut} evaluates to False under π.
- E.g., $\langle \{e\}, \{u\} \rangle$ matches $\{(e, \text{True})\}$ and $\{(e, \text{True}), (u, \text{True})\}$, but does not match $\{\}$ or $\{(e, \text{True}), (u, \text{False})\}$.

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.
- **Definition.** We say that $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ **matches** assignment π iff:
  1. every literal in L^{now} evaluates to True under π, and
  2. no literal in L^{fut} evaluates to False under π.
- E.g., $\langle \{e\}, \{u\} \rangle$ matches $\{(e, \text{True})\}$ and $\{(e, \text{True}), (u, \text{True})\}$, but does not match $\{\}$ or $\{(e, \text{True}), (u, \text{False})\}$.
- $\langle L^{\text{now}}, \{\ell, \neg\ell\} \rangle$ matches π only if π doesn't assign ℓ.

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.
- **Definition.** We say that $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ **matches** assignment π iff:
  1. every literal in L^{now} evaluates to True under π, and
  2. no literal in L^{fut} evaluates to False under π.
- **Definition.** "$\langle L^{\text{now}}, L^{\text{fut}} \rangle \models (\Phi \Leftrightarrow \psi)$" means "for all assignments π that match $\langle L^{\text{now}}, L^{\text{fut}} \rangle$, $\Phi|_\pi$ is logically equivalent to $\psi|_\pi$ unless π is a **don't-care** assignment".

$\langle L^{\text{now}}, L^{\text{fut}} \rangle$ Sequents

- **Definition.** A **game-state specifier** is a pair $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ consisting of two sets of literals, L^{now} and L^{fut}.
- **Definition.** We say that $\langle L^{\text{now}}, L^{\text{fut}} \rangle$ **matches** assignment π iff:
  1. every literal in L^{now} evaluates to True under π, and
  2. no literal in L^{fut} evaluates to False under π.
- **Definition.** "$\langle L^{\text{now}}, L^{\text{fut}} \rangle \models (\Phi \Leftrightarrow \psi)$" means "for all assignments π that match $\langle L^{\text{now}}, L^{\text{fut}} \rangle$, $\Phi|_\pi$ is logically equivalent to $\psi|_\pi$ unless π is a **don't-care** assignment".
- Without ghost variables: No assignments are don't-care.
- With ghost variables: See paper for details.

Correspondence of Sequents to Clauses and Cubes

- Consider a QBF with existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(e_1 \vee \ldots \vee e_n \vee u_1 \vee \ldots \vee u_m)$ in CNF Φ_{in} corresponds to sequent $\langle \{\neg e_1, \ldots, \neg e_n\}, \{\neg u_1, \ldots, \neg u_m\} \rangle \models (\Phi_{in} \Leftrightarrow \text{False})$.

Correspondence of Sequents to Clauses and Cubes

- Consider a QBF with existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(e_1 \vee \ldots \vee e_n \vee u_1 \vee \ldots \vee u_m)$ in CNF Φ_{in} corresponds to sequent $\langle \{\neg e_1, \ldots, \neg e_n\}, \{\neg u_1, \ldots, \neg u_m\} \rangle \models (\Phi_{in} \Leftrightarrow \text{False})$.
- Cube $(u_1 \wedge \ldots \wedge u_m \wedge e_1 \wedge \ldots \wedge e_n)$ in DNF Φ_{in} corresponds to sequent $\langle \{u_1, \ldots, u_m\}, \{e_1, \ldots, e_n\} \rangle \models (\Phi_{in} \Leftrightarrow \text{True})$.

Correspondence of Sequents to Clauses and Cubes

- Consider a QBF with existential literals $e_1 \ldots e_n$ and universal literals $u_1 \ldots u_m$.
- Clause $(e_1 \vee \ldots \vee e_n \vee u_1 \vee \ldots \vee u_m)$ in CNF Φ_{in} corresponds to sequent $\langle \{\neg e_1, \ldots, \neg e_n\}, \{\neg u_1, \ldots, \neg u_m\} \rangle \models (\Phi_{in} \Leftrightarrow \text{False})$.
- Cube $(u_1 \wedge \ldots \wedge u_m \wedge e_1 \wedge \ldots \wedge e_n)$ in DNF Φ_{in} corresponds to sequent $\langle \{u_1, \ldots, u_m\}, \{e_1, \ldots, e_n\} \rangle \models (\Phi_{in} \Leftrightarrow \text{True})$.
- Sequents generalize clauses/cubes because $\langle L^{\text{now}}, L^{\text{fut}} \rangle \models (\Phi \Leftrightarrow \psi)$ can have ψ be a formula in terms of free variables.

Inference rule for free variable

Literal r is free

$$\langle L_1^{\mathsf{now}} \cup \{r\},\, L_1^{\mathsf{fut}} \rangle \models (\Phi_{in} \Leftrightarrow \psi_1)$$

$$\langle L_2^{\mathsf{now}} \cup \{\neg r\},\, L_2^{\mathsf{fut}} \rangle \models (\Phi_{in} \Leftrightarrow \psi_2)$$

$$\langle L_1^{\mathsf{now}} \cup L_2^{\mathsf{now}},\, L_1^{\mathsf{fut}} \cup L_2^{\mathsf{fut}} \cup \{r, \neg r\} \rangle \models (\Phi_{in} \Leftrightarrow \mathrm{ite}(r, \psi_1, \psi_2))$$

Top-level algorithm (based on DPLL)

```
1.  initialize_sequent_database();
2.  π_cur := ∅;  Propagate();
3.  while (true) {

12. }
```

Top-level algorithm (based on DPLL)

```
1.  initialize_sequent_database();
2.  π_cur := ∅;  Propagate();
3.  while (true) {
4.    while (π_cur doesn't match any database sequent) {
5.      DecideLit();
6.      Propagate();
7.    }
12. }
```

Top-level algorithm (based on DPLL)

```
1.  initialize_sequent_database();
2.  π_cur := ∅;  Propagate();
3.  while (true) {
4.    while (π_cur doesn't match any database sequent) {
5.      DecideLit();
6.      Propagate();
7.    }
8.    Learn();
9.    if (learned seq has form ⟨∅, L^fut⟩ ⊨ (Φ_in ⇔ ψ)) return ψ;
10.   Backtrack();
11.   Propagate();
12. }
```

Propagation

- Let *seq* be a sequent $\langle L^{\text{now}}, L^{\text{fut}} \rangle \models (\Phi_{in} \Leftrightarrow \psi)$ in database.
- If there is a literal $\ell \in L^{\text{now}}$ such that
  1. $\pi_{cur} \cup \{\ell\}$ matches *seq*, and
  2. ℓ is not downstream of any unassigned literals in L^{fut},

  then $\neg\ell$ is *forced*; it is added to the current assignment π_{cur}.

Propagation

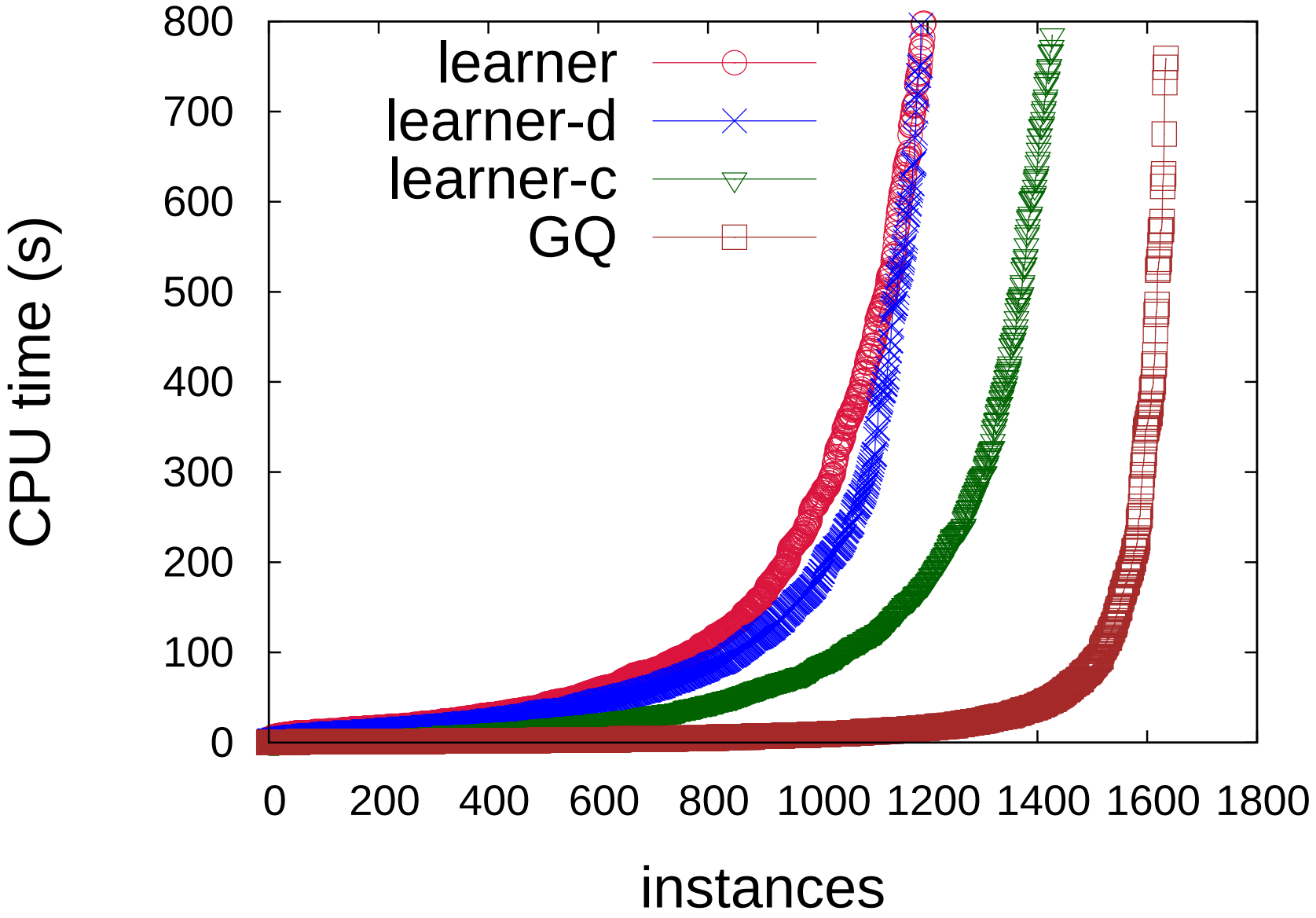
- Let *seq* be a sequent $\langle L^{\text{now}}, L^{\text{fut}} \rangle \models (\Phi_{in} \Leftrightarrow \psi)$ in database.
- If there is a literal $\ell \in L^{\text{now}}$ such that
  1. $\pi_{cur} \cup \{\ell\}$ matches *seq*, and
  2. ℓ is not downstream of any unassigned literals in L^{fut},

  then $\neg\ell$ is *forced*; it is added to the current assignment π_{cur}.
- Propagation ensures that the solver never re-explores areas of the search space for which it already knows the answer.

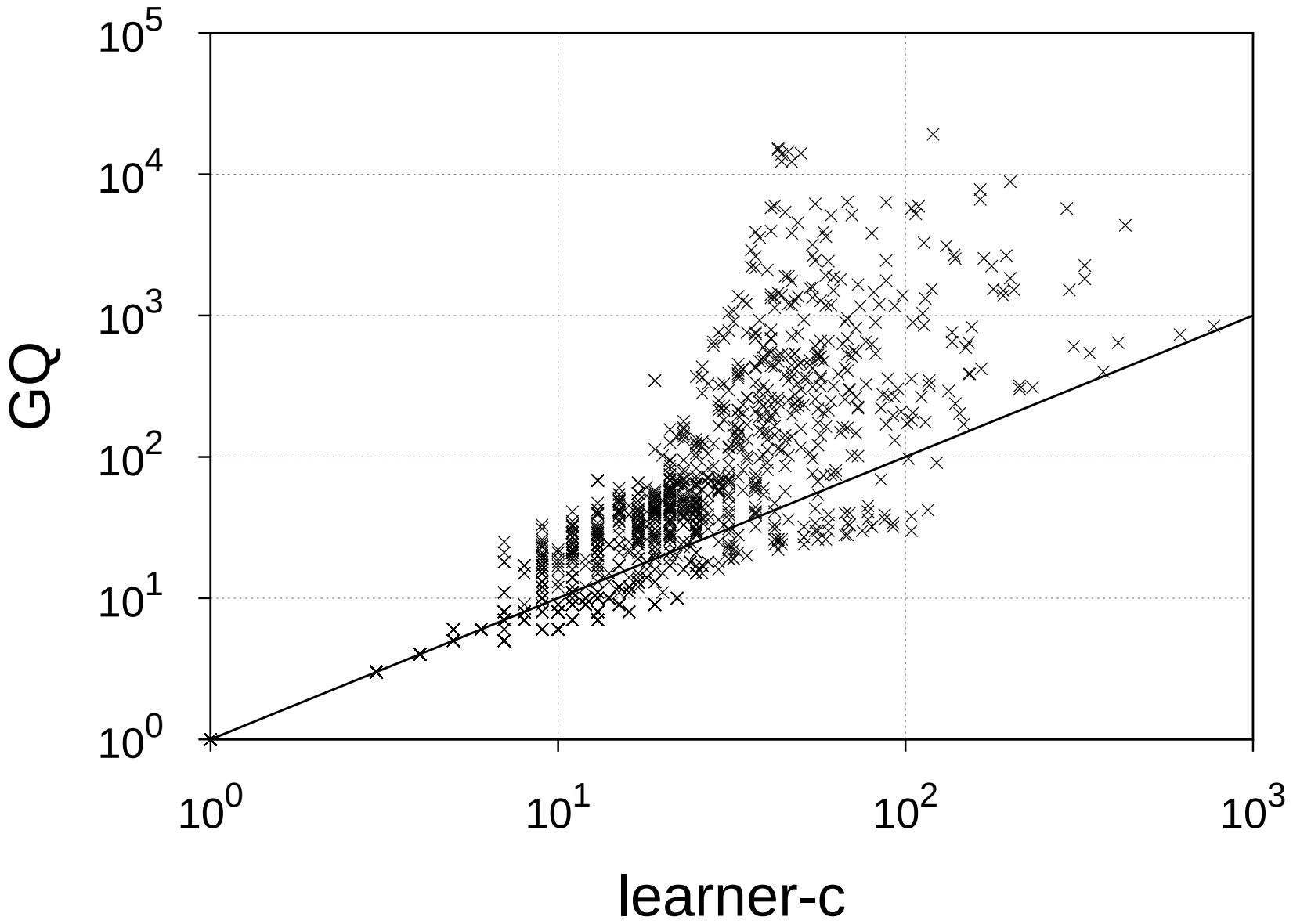
Experimental Comparison

- Our solver: GhostQ.
- Compared to computational-learning solver from:
  B. Becker, R. Ehlers, M. Lewis, and P. Marin,
  "ALLQBF solving by computational learning" (ATVA 2012).
- Benchmarks (from same paper): synthesis from formal specifications.
- HWMCC'10 Benchmarks: One-step forward reachability.

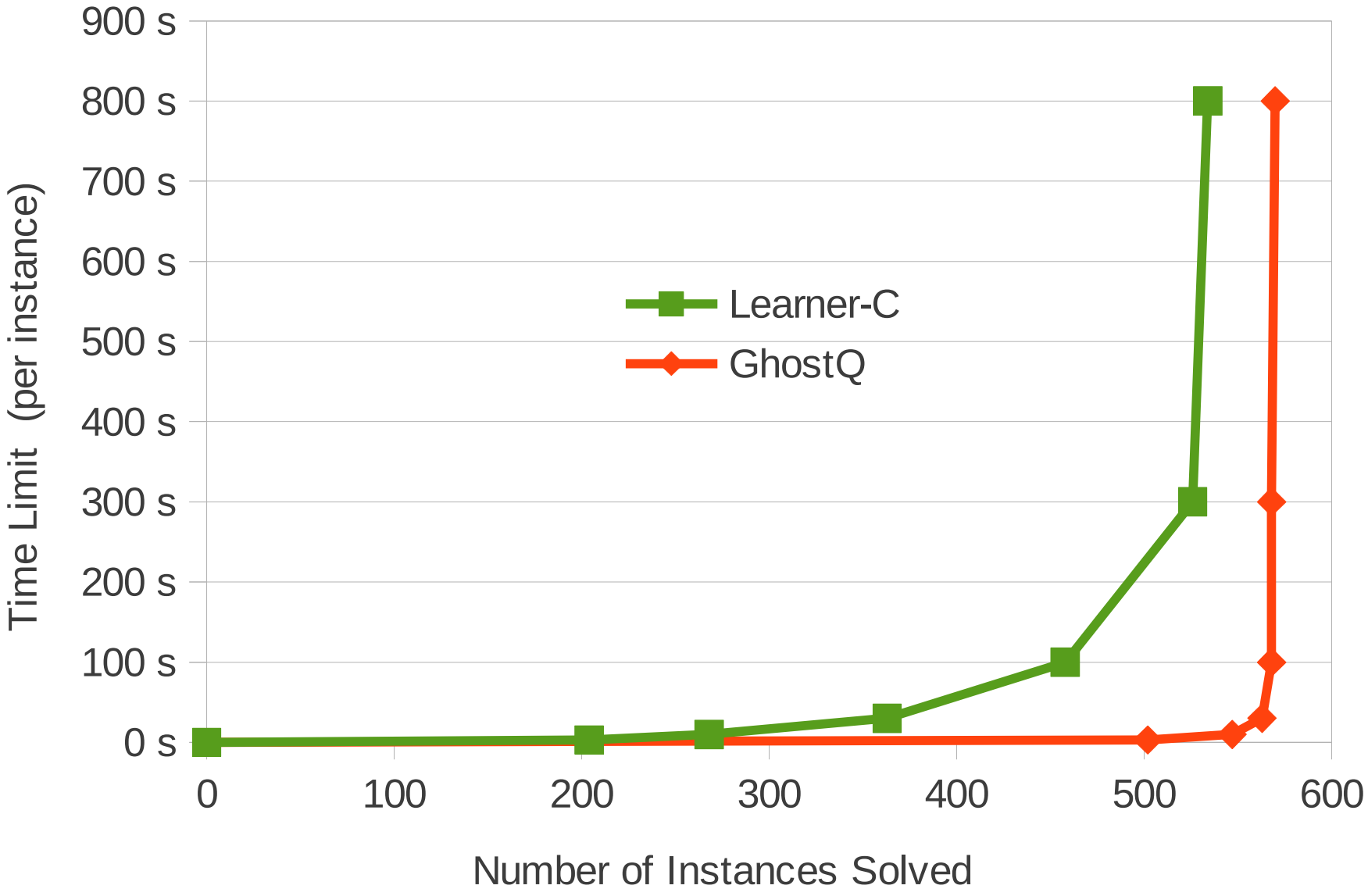
Cactus Plot


learner
learner-d
learner-c
GQ
CPU time (s)
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000
1200
1400
1600
1800
instances

Formula Size



HWMCC'10 Benchmarks: Cactus Plot



Conclusion

- DPLL-based solver for open QBF.
- Sequents generalize clauses and cubes.
- Generates proof certificates.
- Our solver produces **unordered** BDDs.
  - Unordered because of unit propagation.
  - In our experience, often larger than OBDDs.