Validating QBF Invalidity in HOL4

Tjark Weber

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Interactive Theorem Proving (ITP)
14 July, 2010

Quantified Boolean Formulae

QBF = propositional logic + quantifiers over Boolean variables

Example (QBF)

$$\exists x\, \forall y\, \exists z.\; x \wedge (y \vee z) \wedge (y \vee \neg z)$$

- Applications in formal verification, adversarial planning, etc.
- QBF is the canonical PSPACE-complete problem.

Motivation

HOL4 is a popular interactive theorem prover. Interactive theorem proving benefits from automation.

QBF solvers are complex software tools. We need a way to validate their results.

Motivation

HOL4 is a popular interactive theorem prover. Interactive theorem proving benefits from automation.

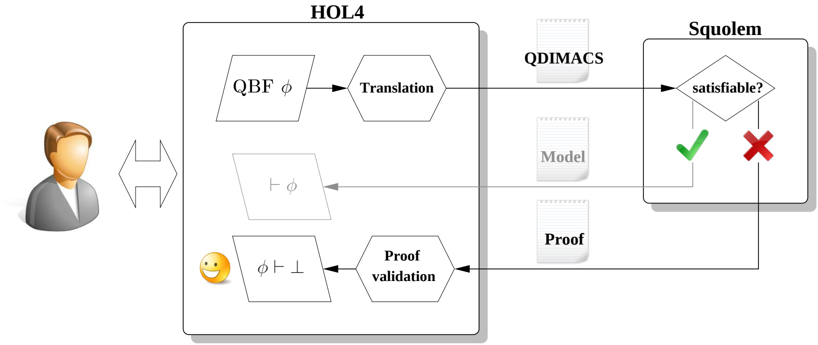
↓

Integrate a QBF solver with HOL4. Check its results, LCF-style.

↑

QBF solvers are complex software tools. We need a way to validate their results.

System Overview



Related Work

Integration of automated provers with ITPs

- SAT, SMT, FOL, HOL, . . .

Certificates for QBF solvers

- Squolem, sKizzo, yQuaffle, EBDDRES, . . .

Propositional Logic

- Boolean variables: x, y, z, . . .
- A literal is a possibly negated variable.
- A clause is a disjunction of literals.
- A propositional formula is in CNF iff it is a conjunction of clauses.

Example (CNF)

$$x \wedge (y \vee z) \wedge (y \vee \neg z)$$

Quantified Boolean Formulae

Definition (Quantified Boolean Formula)

A Quantified Boolean Formula (QBF) is of the form

$$Q_1 x_1 \ldots Q_n x_n.\ \phi,$$

where $n \geq 0$, each x_i is a Boolean variable, each Q_i is either $\forall$ or $\exists$, and ϕ is a propositional formula in CNF.

Example (QBF)

$$\exists x\, \forall y\, \exists z.\ x \wedge (y \vee z) \wedge (y \vee \neg z)$$

Quantified Boolean Formulae: Semantics

QBF semantics:

- $[\![\forall x.\ \phi]\!] = [\![\phi[x \mapsto \top] \wedge \phi[x \mapsto \bot]]\!]$
- $[\![\exists x.\ \phi]\!] = [\![\phi[x \mapsto \top] \vee \phi[x \mapsto \bot]]\!]$

Infeasible for QBF of interest!

Squolem establishes invalidity of QBF using an inference rule known as Q-resolution.

Q-Resolution

Propositional resolution:

$$\frac{\phi \vee x \qquad \psi \vee \neg x}{\phi \vee \psi}$$

Forall-reduction:

$$\frac{\forall x.\ \phi \vee (\neg)x}{\phi}\ x \notin \phi$$

Definition (Q-resolution)

Let ϕ and ψ be two clauses of a QBF that can be resolved. Their resolvent's forall-reduct is called the Q-resolvent of ϕ and ψ.

Q-Resolution: Example

Theorem (BKF95)

Q-resolution is sound and refutation-complete for QBF in prenex form.

Example

$$\exists x\, \forall y\, \exists z.\, x \wedge (y \vee z) \wedge (y \vee \neg z)$$

Q-Resolution: Example

Theorem (BKF95)

Q-resolution is sound and refutation-complete for QBF in prenex form.

Example

$$\frac{\exists x\, \forall y\, \exists z.\ x \wedge (y \vee z) \wedge (y \vee \neg z)}{\exists x\, \forall y\, \exists z.\ y}$$

Q-Resolution: Example

Theorem (BKF95)

Q-resolution is sound and refutation-complete for QBF in prenex form.

Example

$$\frac{\exists x\, \forall y\, \exists z.\; x \wedge (y \vee z) \wedge (y \vee \neg z)}{\exists x\, \forall y.\; y}$$

Q-Resolution: Example

Theorem (BKF95)

Q-resolution is sound and refutation-complete for QBF in prenex form.

Example

$$\dfrac{\dfrac{\exists x\, \forall y\, \exists z.\ x \wedge (y \vee z) \wedge (y \vee \neg z)}{\exists x\, \forall y.\ y}}{\exists x.\ \bot}$$

Q-Resolution: Example

Theorem (BKF95)

Q-resolution is sound and refutation-complete for QBF in prenex form.

Example

$$\cfrac{\cfrac{\exists x\, \forall y\, \exists z.\ x \wedge (y \vee z) \wedge (y \vee \neg z)}{\exists x\, \forall y.\ y}}{\bot}$$

LCF-style Theorem Proving

Theorems are implemented as an abstract data type.

There is a fixed number of constructor functions—one for each axiom schema/inference rule of HOL.

More complicated proof procedures must be implemented by composing these functions.

LCF-style Theorem Proving

Theorems are implemented as an abstract data type.

There is a fixed number of constructor functions—one for each axiom schema/inference rule of HOL.

More complicated proof procedures must be implemented by composing these functions.

The trusted code base consists only of the theorem ADT.

Selected HOL4 Inference Rules

$$\frac{}{\{\phi\} \vdash \phi}\ \textsc{Assume} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\ \textsc{Conj1} \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\ \textsc{Conj2}$$

$$\frac{\Gamma \vdash \phi \vee \psi \qquad \Delta_1 \cup \{\phi\} \vdash \theta \qquad \Delta_2 \cup \{\psi\} \vdash \theta}{\Gamma \cup \Delta_1 \cup \Delta_2 \vdash \theta}\ \textsc{DisjCases}$$

$$\frac{\Gamma \vdash \phi \implies \bot}{\Gamma \vdash \neg\phi}\ \textsc{NotIntro} \qquad \frac{\Gamma \vdash \neg\phi}{\Gamma \vdash \phi \implies \bot}\ \textsc{NotElim}$$

$$\frac{\Gamma \vdash \psi}{\Gamma \setminus \{\phi\} \vdash \phi \implies \psi}\ \textsc{Disch} \qquad \frac{\Gamma \vdash \phi \implies \psi \qquad \Delta \vdash \phi}{\Gamma \cup \Delta \vdash \psi}\ \textsc{MP}$$

$$\frac{\Gamma \vdash \phi}{\Gamma\theta \vdash \phi\theta}\ \textsc{Inst}_\theta \qquad \frac{\Gamma \vdash \forall x.\, \phi}{\Gamma \vdash \phi[x \mapsto t]}\ \textsc{Spec}_t$$

$$\frac{\Gamma \vdash \exists x.\, \phi \qquad \Delta \cup \{\phi[x \mapsto v]\} \vdash \psi}{\Gamma \cup \Delta \vdash \psi}\ \textsc{Choose}_v\ (v \text{ not free in } \Gamma,\ \Delta \text{ or } \psi)$$

Preliminaries

$$\{\phi\} \qquad \vdash \quad \phi$$

Preliminaries

$$Q_1 x_1 \ldots Q_n x_n.\ \phi$$

$$\{\phi\} \qquad \vdash \qquad \phi$$

$$\bot$$

Clear separation of propositional and quantifier reasoning!

Preliminaries: Sequent Clause Form

1. Eliminate conjunctions:

$$\{\phi\} \vdash \phi$$

2. Eliminate disjunctions:

$$\{\phi\} \vdash C_i$$

3. Dictionary: $i \mapsto (\{\phi, \neg l^i_1, \ldots, \neg l^i_{m_i}\} \vdash \bot,\ Q_1 x_1 \ldots Q_n x_n)$

Preliminaries: Sequent Clause Form

1. Eliminate conjunctions:

$$\{\phi\} \vdash C_1 \wedge \cdots \wedge C_k$$

2. Eliminate disjunctions:

$$\{\phi\} \vdash C_i$$

3. Dictionary: $i \mapsto (\{\phi, \neg l_1^i, \ldots, \neg l_{m_i}^i\} \vdash \bot,\ Q_1 x_1 \ldots Q_n x_n)$

Preliminaries: Sequent Clause Form

1. Eliminate conjunctions:

$$\{\phi\} \vdash C_1 \wedge \cdots \wedge C_k$$

$$\Downarrow$$

$$\{\phi\} \vdash C_1, \quad \ldots, \quad \{\phi\} \vdash C_k$$

2. Eliminate disjunctions:

$$\{\phi\} \vdash C_i$$

3. Dictionary: $i \mapsto (\{\phi, \neg l^i_1, \ldots, \neg l^i_{m_i}\} \vdash \bot,\ Q_1 x_1 \ldots Q_n x_n)$

Preliminaries: Sequent Clause Form

1. Eliminate conjunctions:

$$\{\phi\} \vdash C_1 \wedge \cdots \wedge C_k$$

$$\Downarrow$$

$$\{\phi\} \vdash C_1, \quad \ldots, \quad \{\phi\} \vdash C_k$$

2. Eliminate disjunctions:

$$\{\phi\} \vdash l_1^i \vee \cdots \vee l_{m_i}^i$$

3. Dictionary: $i \mapsto (\{\phi, \neg l_1^i, \ldots, \neg l_{m_i}^i\} \vdash \bot, \; Q_1 x_1 \ldots Q_n x_n)$

Preliminaries: Sequent Clause Form

1. Eliminate conjunctions:

$$\{\phi\} \vdash C_1 \wedge \cdots \wedge C_k$$

$$\Downarrow$$

$$\{\phi\} \vdash C_1, \quad \ldots, \quad \{\phi\} \vdash C_k$$

2. Eliminate disjunctions:

$$\{\phi\} \vdash l_1^i \vee \cdots \vee l_{m_i}^i$$

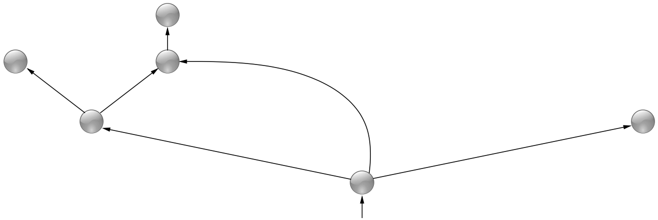
$$\Downarrow$$

$$\{\phi, \neg l_1^i, \ldots, \neg l_{m_i}^i\} \vdash \bot$$

3. Dictionary: $i \mapsto (\{\phi, \neg l_1^i, \ldots, \neg l_{m_i}^i\} \vdash \bot, \; Q_1 x_1 \ldots Q_n x_n)$

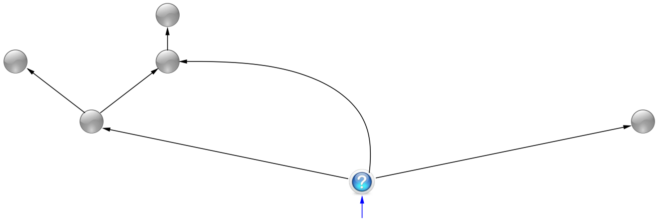
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



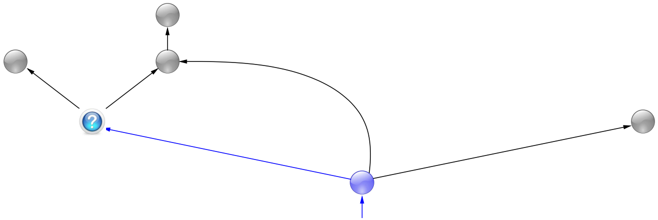
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



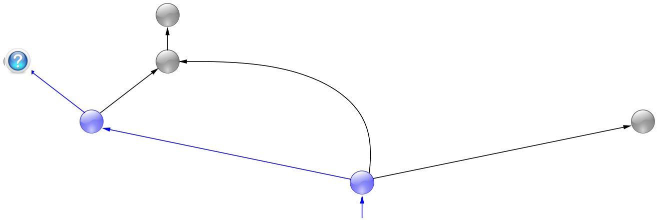
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



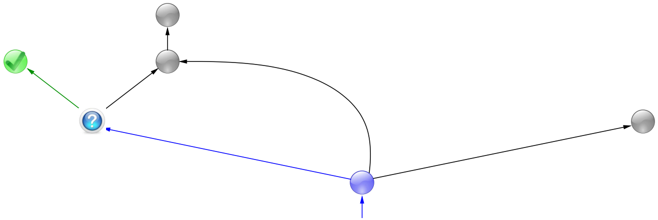
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



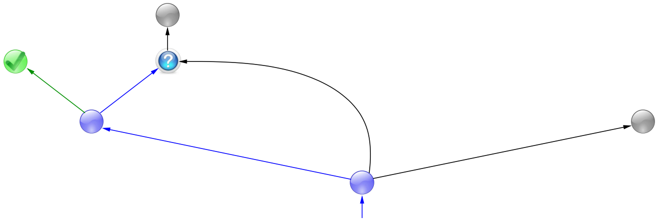
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



General Proof Structure

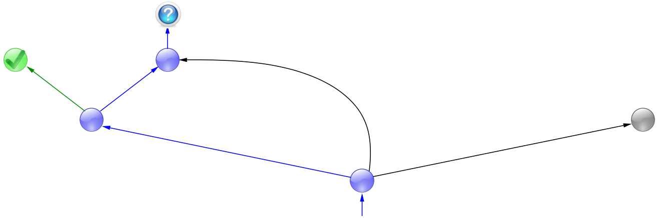
Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



General Proof Structure

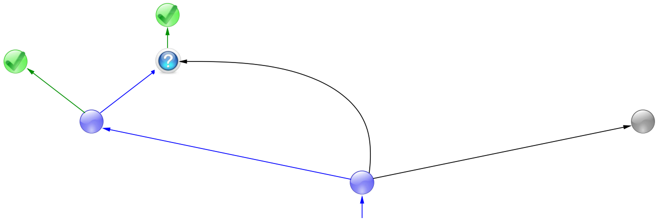
Squolem's certificates of invalidity encode a directed acyclic graph.

We perform a depth-first post-order traversal of this graph.



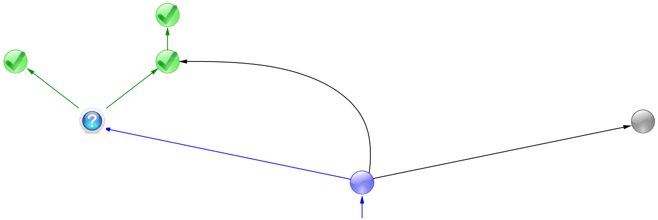
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



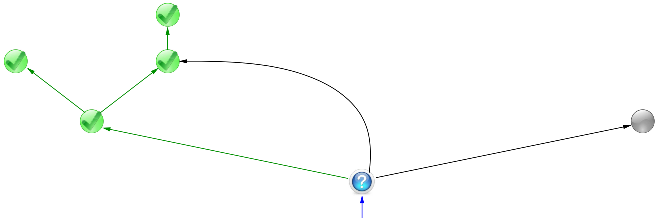
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



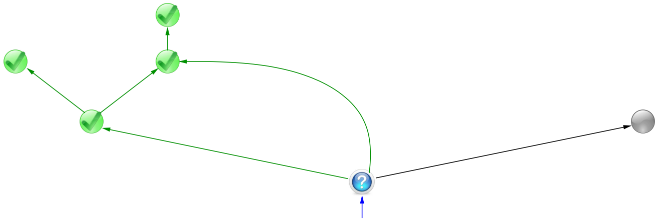
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



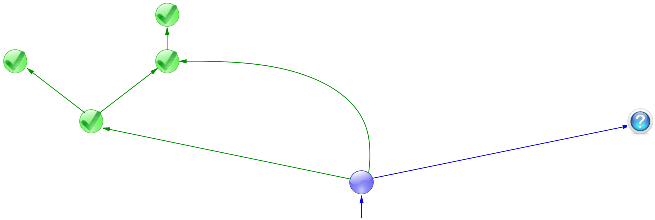
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



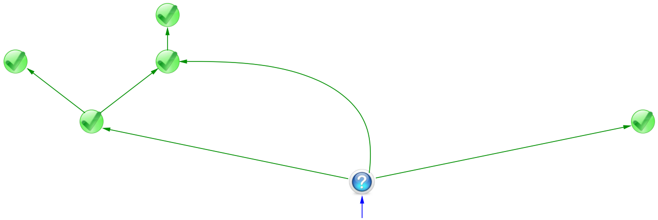
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



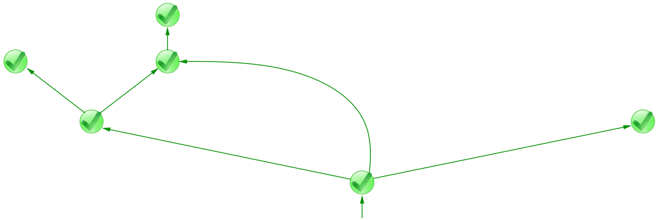
General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph. We perform a depth-first post-order traversal of this graph.



General Proof Structure

Squolem's certificates of invalidity encode a directed acyclic graph.
We perform a depth-first post-order traversal of this graph.



Q-Resolution: Example

Example (QBF)

$$\exists x\, \forall y\, \exists z.\ \phi, \text{ where } \phi = x \wedge (y \vee z) \wedge (y \vee \neg z)$$

1. Assume ϕ to obtain $\{\phi\} \vdash \phi$.

Q-Resolution: Example

Example (QBF)

$\exists x\, \forall y\, \exists z.\ \phi$, where $\phi = x \wedge (y \vee z) \wedge (y \vee \neg z)$

1. Assume ϕ to obtain $\{\phi\} \vdash \phi$.
2. Separate clause theorems:
   1. $\{\phi\} \vdash x$ 2. $\{\phi\} \vdash y \vee z$ 3. $\{\phi\} \vdash y \vee \neg z$

Q-Resolution: Example

Example (QBF)

$\exists x\, \forall y\, \exists z.\ \phi$, where $\phi = x \wedge (y \vee z) \wedge (y \vee \neg z)$

1. Assume ϕ to obtain $\{\phi\} \vdash \phi$.
2. Separate clause theorems:
   1. $\{\phi\} \vdash x$ 2. $\{\phi\} \vdash y \vee z$ 3. $\{\phi\} \vdash y \vee \neg z$
3. Sequent form:
   1. $\{\phi, \neg x\} \vdash \bot$ 2. $\{\phi, \neg y, \neg z\} \vdash \bot$ 3. $\{\phi, \neg y, z\} \vdash \bot$.
   The missing quantifier prefix for each theorem is $\exists x\, \forall y\, \exists z$.

Q-Resolution: Example (cont.)

1. $\{\phi, \neg x\} \vdash \bot$ 2. $\{\phi, \neg y, \neg z\} \vdash \bot$ 3. $\{\phi, \neg y, z\} \vdash \bot$ $(\exists x\, \forall y\, \exists z)$

4. Q-resolve theorems (2) and (3). Propositional resolution yields $\{\phi, \neg y\} \vdash \bot$. The resolvent's largest variable is y.

Q-Resolution: Example (cont.)

1. $\{\phi, \neg x\} \vdash \bot$ 2. $\{\phi, \neg y, \neg z\} \vdash \bot$ 3. $\{\phi, \neg y, z\} \vdash \bot$ $(\exists x\, \forall y\, \exists z)$

4. Q-resolve theorems (2) and (3). Propositional resolution yields $\{\phi, \neg y\} \vdash \bot$. The resolvent's largest variable is y.
5. Since y is universal, we perform forall-reduction. We introduce missing quantifiers $\exists z$ and $\forall y$, first deriving $\{\exists z.\, \phi,\ \neg y\} \vdash \bot$, and then $\{\forall y \exists z.\, \phi,\ \neg y\} \vdash \bot$.

Q-Resolution: Example (cont.)

1. $\{\phi, \neg x\} \vdash \bot$ 2. $\{\phi, \neg y, \neg z\} \vdash \bot$ 3. $\{\phi, \neg y, z\} \vdash \bot$ $(\exists x\, \forall y\, \exists z)$

4. Q-resolve theorems (2) and (3). Propositional resolution yields $\{\phi, \neg y\} \vdash \bot$. The resolvent's largest variable is y.
5. Since y is universal, we perform forall-reduction. We introduce missing quantifiers $\exists z$ and $\forall y$, first deriving $\{\exists z.\, \phi,\ \neg y\} \vdash \bot$, and then $\{\forall y \exists z.\, \phi,\ \neg y\} \vdash \bot$.
6. Now we eliminate y by instantiating it to $\bot$, thereby obtaining $\{\forall y \exists z.\, \phi,\ \neg\bot\} \vdash \bot$. Discharging $\neg\bot$ yields $\{\forall y \exists z.\, \phi\} \vdash \bot$.

Q-Resolution: Example (cont.)

1. $\{\phi, \neg x\} \vdash \bot$ 2. $\{\phi, \neg y, \neg z\} \vdash \bot$ 3. $\{\phi, \neg y, z\} \vdash \bot$ $(\exists x\, \forall y\, \exists z)$

4. Q-resolve theorems (2) and (3). Propositional resolution yields $\{\phi, \neg y\} \vdash \bot$. The resolvent's largest variable is y.
5. Since y is universal, we perform forall-reduction. We introduce missing quantifiers $\exists z$ and $\forall y$, first deriving $\{\exists z.\, \phi,\ \neg y\} \vdash \bot$, and then $\{\forall y \exists z.\, \phi,\ \neg y\} \vdash \bot$.
6. Now we eliminate y by instantiating it to $\bot$, thereby obtaining $\{\forall y \exists z.\, \phi,\ \neg \bot\} \vdash \bot$. Discharging $\neg\bot$ yields $\{\forall y \exists z.\, \phi\} \vdash \bot$.
7. The next missing quantifier is $\exists x$, and x does not occur in the clause (except in ϕ). We finally arrive at $\{\exists x \forall y \exists z.\, \phi\} \vdash \bot$.

Q-Resolution: Propositional Resolution

Q-resolution is propositional resolution followed by forall-reduction.

Propositional resolution for clauses in sequent form [AW09]:

$$\dfrac{\dfrac{\Gamma \cup \{\neg v\} \vdash \bot}{\Gamma \vdash \neg v \implies \bot}\ \textsc{Disch} \qquad \dfrac{\dfrac{\Delta \cup \{v\} \vdash \bot}{\Delta \vdash v \implies \bot}\ \textsc{Disch}}{\Delta \vdash \neg v}\ \textsc{NotIntro}}{\Gamma \cup \Delta \vdash \bot}\ \text{MP}$$

Q-Resolution: Forall-Reduction (1)

Let x_i be the largest variable that occurs in $\{\phi, l_1, \ldots, l_m\} \vdash \bot$. We must perform forall-reduction if x_i is universal. Suppose the missing quantifier prefix is $Q_1 x_1 \ldots \forall x_i \ldots Q_j x_j$, with $j \geq i$.

Q-Resolution: Forall-Reduction (1)

Let x_i be the largest variable that occurs in $\{\phi, l_1, \ldots, l_m\} \vdash \bot$. We must perform forall-reduction if x_i is universal. Suppose the missing quantifier prefix is $Q_1 x_1 \ldots \forall x_i \ldots Q_j x_j$, with $j \geq i$.

- If $Q_j = \forall$, we derive

$$\dfrac{\dfrac{\{\phi, l_1, \ldots, l_m\} \vdash \bot}{\{l_1, \ldots, l_m\} \vdash \phi \implies \bot}\ \textsc{Disch} \qquad \dfrac{\dfrac{}{\{\forall x_j.\, \phi\} \vdash \forall x_j.\, \phi}\ \textsc{Assume}}{\{\forall x_j.\, \phi\} \vdash \phi}\ \textsc{Spec}_{x_j}}{\{\forall x_j.\, \phi,\ l_1, \ldots, l_m\} \vdash \bot}\ \mathrm{MP}$$

Q-Resolution: Forall-Reduction (1)

Let x_i be the largest variable that occurs in $\{\phi, l_1, \ldots, l_m\} \vdash \bot$. We must perform forall-reduction if x_i is universal. Suppose the missing quantifier prefix is $Q_1 x_1 \ \ldots \ \forall x_i \ \ldots \ Q_j x_j$, with $j \geq i$.

- If $Q_j = \forall$, we derive

$$\dfrac{\dfrac{\{\phi, l_1, \ldots, l_m\} \vdash \bot}{\{l_1, \ldots, l_m\} \vdash \phi \implies \bot}\ \textsc{Disch} \qquad \dfrac{\dfrac{}{\{\forall x_j.\, \phi\} \vdash \forall x_j.\, \phi}\ \textsc{Assume}}{\{\forall x_j.\, \phi\} \vdash \phi}\ \textsc{Spec}_{x_j}}{\{\forall x_j.\, \phi,\ l_1, \ldots, l_m\} \vdash \bot}\ \textrm{MP}$$

- If $Q_j = \exists$, then necessarily $j > i$, and we derive

$$\dfrac{\dfrac{}{\{\exists x_j.\, \phi\} \vdash \exists x_j.\, \phi}\ \textsc{Assume} \qquad \{\phi, l_1, \ldots, l_m\} \vdash \bot}{\{\exists x_j.\, \phi,\ l_1, \ldots, l_m\} \vdash \bot}\ \textsc{Choose}_{x_j}$$

Q-Resolution: Forall-Reduction (2)

- Repeating this to introduce all missing quantifiers up to $\forall x_i$, we arrive at $\{\forall x_i \ldots Q_j x_j.\, \phi,\ l_1, \ldots, l_m\} \vdash \bot$.
- Now x_i occurs free only in one of the literals $l_1, \ldots, l_m$. We instantiate x_i to $\neg\bot$ if it occurs positively, and to $\bot$ if it occurs negatively.
- In either case the literal becomes $\neg\bot$ and can be discharged.

Q-Resolution: Forall-Reduction (2)

- Repeating this to introduce all missing quantifiers up to $\forall x_i$, we arrive at $\{\forall x_i \ldots Q_j x_j . \phi, l_1, \ldots, l_m\} \vdash \bot$.
- Now x_i occurs free only in one of the literals $l_1, \ldots, l_m$. We instantiate x_i to $\neg\bot$ if it occurs positively, and to $\bot$ if it occurs negatively.
- In either case the literal becomes $\neg\bot$ and can be discharged.

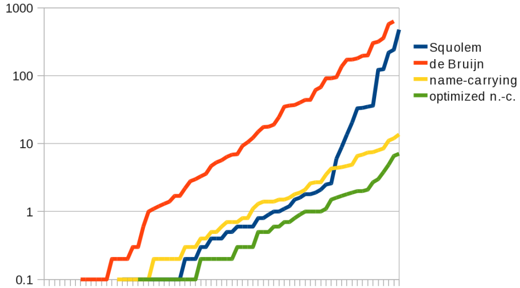
That's all! We have . . .

- introduced quantifiers for variables that don't occur,
- eliminated the universal variable x_i.

Run-Times

Evaluation on 69 invalid QBF problems from the *2005 fixed instance* and *2006 preliminary QBF-Eval* data sets

up to 131 alternating quantifiers, 24,562 variables, 35,189 clauses



Run-Times

Evaluation on 69 invalid QBF problems from the *2005 fixed instance* and *2006 preliminary QBF-Eval* data sets

up to 131 alternating quantifiers, 24,562 variables, 35,189 clauses

All problems are checked successfully!

- Average run-times: 60 s (de Bruijn), 2 s (name-carrying), 0.8 s (optimized name-carrying)
- 25 times faster (after opt.) than proof search with Squolem
- 1-2 orders of magnitude slower than stand-alone checking

Variable Binding and Substitution

$\forall x.\,\phi$ is syntactic sugar for $\forall(\lambda x.\,\phi)$ (likewise for $\exists x.\,\phi$).

de Bruijn: $(\lambda x.\,\phi)\,x \to_\beta \phi[0 \mapsto x]$ name-carrying: $(\lambda x.\,\phi)\,x \to_\beta \phi$

HOL's name-carrying kernel is 29 times faster for QBF validation than the kernel that uses de Bruijn indices internally.

Variable Binding and Substitution

$\forall x.\, \phi$ is syntactic sugar for $\forall(\lambda x.\, \phi)$ (likewise for $\exists x.\, \phi$).

de Bruijn: $(\lambda x.\, \phi)\, x \to_\beta \phi[0 \mapsto x]$ name-carrying: $(\lambda x.\, \phi)\, x \to_\beta \phi$

HOL's name-carrying kernel is 29 times faster for QBF validation than the kernel that uses de Bruijn indices internally.

Capture-avoiding substitution may have to rename bound variables away from the free variables in the body of a λ-abstraction.

We achieved a further speed-up of 3 by improving HOL4's implementation of capture-avoiding substitution to collect free variables only when they are actually needed for renaming.

Conclusions

Integration of a QBF solver with HOL4

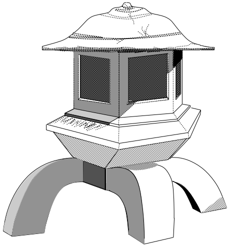
- Improved automation for QBF in HOL4
- High correctness assurances for Squolem's results

- LCF-style proof checking for QBF invalidity is feasible.
- HOL4: `http://hol.sourceforge.net/`

Future Work

- Applications, case studies
- QBF validity

- Other ITPs/QBF solvers
- Different approaches (e.g., reflection)



Future Work

- Applications, case studies
- QBF validity

- Other ITPs/QBF solvers
- Different approaches (e.g., reflection)

Thank You!