Validating QBF Validity in HOL4

Ramana Kumar and Tjark Weber

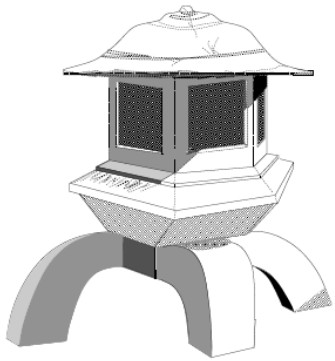
ARG Lunch

March 15, 2011

Last Year's Future Work

Future Work

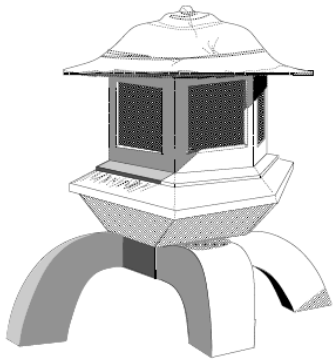
- Applications, case studies
- QBF validity
- Other ITPs/QBF solvers
- Different approaches (e.g., reflection)



Last Year's Future Work

Future Work

- Applications, case studies
- QBF validity
- Other ITPs/QBF solvers
- Different approaches (e.g., reflection)



Quantified Boolean Formulae

QBF = propositional logic + quantifiers over Boolean variables

Example (QBF)

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

- 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.

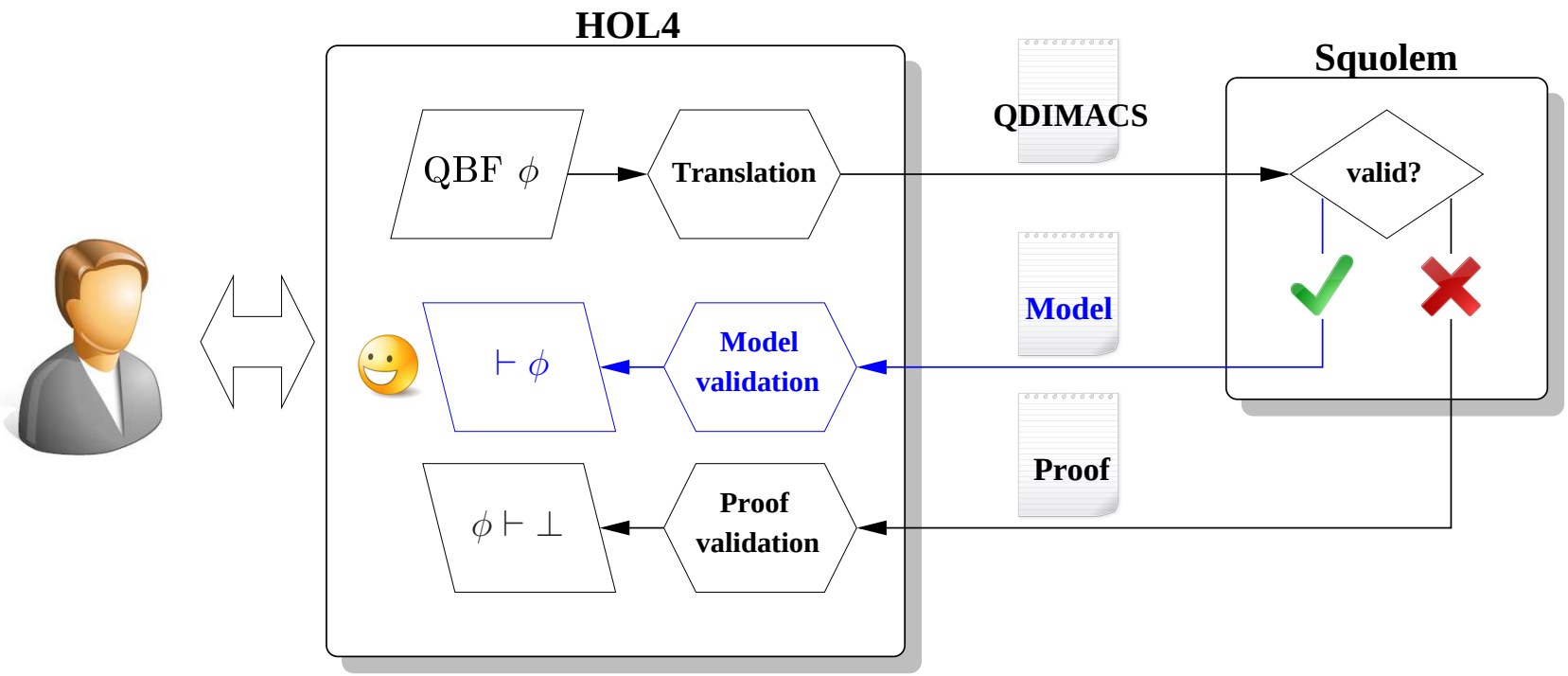
Motivation

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

Validation of QBF invalidity *(ITP 2010)*

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)

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

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)

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

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 validity of QBF by providing witness functions for existential variables.

Witness Functions

A model of the QBF $Q_1 x_1 \ldots Q_n x_n.\ \phi$ is a map from existential variables x_k to witness functions $f_k : \mathbb{B}^{k-1} \longrightarrow \mathbb{B}$.

Squolem's certificates of validity encode a model. Each witness function is given by a propositional formula.

Theorem

A QBF is valid if (and only if) there is a model such that the propositional formula obtained by replacing existential variables with their witness functions is valid.

Witness Functions: Example

QBF

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

Model

$$y \mapsto f_y, \quad f_y(x) = \bot \qquad\qquad z \mapsto f_z, \quad f_z(x, y) = x$$

Witness Functions: Example

QBF

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

Model

$$y \mapsto f_y, \quad f_y(x) = \bot \qquad\qquad z \mapsto f_z, \quad f_z(x, y) = x$$

Propositional Tautology

$$(x \vee \bot \vee \neg x) \wedge (x \vee \top \vee x) \wedge (\neg x \vee \bot \vee x) \wedge (\neg x \vee \top \vee \neg x) \wedge (\top \vee x)$$

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

$$\frac{}{\{\phi\} \vdash \phi}\ \textsc{Assume}_{\phi} \qquad \frac{\Gamma \vdash \phi}{\Gamma\,\theta \vdash \phi\,\theta}\ \textsc{Inst}_{\theta} \qquad \frac{}{\vdash t = t}\ \textsc{Refl}_{t}$$

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

$$\frac{\Gamma \vdash \phi}{\Gamma \vdash \forall x.\, \phi}\ \textsc{Gen}_{x}\ (x \text{ not free in } \Gamma) \qquad \frac{\Gamma \vdash \phi[t]}{\Gamma \vdash \exists x.\, \phi[x]}\ \textsc{Exists}_{(\exists x.\, \phi[x], t)}$$

Extension Variables and Witnesses

Given the QBF

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

Squolem's certificate contains definitions of extension variables

$$v_i \Leftrightarrow t_i$$

and witnesses for all existential variables

$$x_{e_k} \Leftrightarrow v_{e_k}.$$

Extension Variables and Witnesses

Given the QBF

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

Squolem's certificate contains definitions of extension variables

$$v_i \Leftrightarrow t_i$$

and witnesses for all existential variables

$$x_{e_k} \Leftrightarrow v_{e_k}.$$

We construct the propositional formula

$$(v_1 \Leftrightarrow t_1) \implies \ldots \implies (x_{e_1} \Leftrightarrow v_{e_1}) \implies \ldots \implies \phi.$$

Propositional Reasoning

We prove the (purely propositional) theorem

$$\vdash (v_1 \Leftrightarrow t_1) \implies \ldots \implies (x_{e_1} \Leftrightarrow v_{e_1}) \implies \ldots \implies \phi$$

by calling MiniSat *(Weber/Amjad, JAL 2009)*.

Propositional Reasoning

We prove the (purely propositional) theorem

$$\vdash (v_1 \Leftrightarrow t_1) \implies \ldots \implies (x_{e_1} \Leftrightarrow v_{e_1}) \implies \ldots \implies \phi$$

by calling MiniSat *(Weber/Amjad, JAL 2009)*.

We then turn antecedents into hypotheses to obtain

$$\{v_1 \Leftrightarrow t_1, \ldots, x_{e_1} \Leftrightarrow v_{e_1}\} \vdash \phi.$$

Our final task: eliminate hypotheses and reintroduce quantifiers.

Hypothesis Elimination

To eliminate a hypothesis of the form $v \Leftrightarrow t$, we instantiate v to t and use REFL.

To introduce quantifiers, we simply use GEN and EXISTS.

This must be done in the right order. We topologically sort variables according to these dependencies:

- $x_k \leftarrow x_{k+1}$
- $v_{e_k} \leftarrow x_{e_k}$
- $v \leftarrow v_i$, for each variable v in t_i

Example

QBF

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

Example

QBF

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

Model

$v_1 \Leftrightarrow \bot, \quad v_2 \Leftrightarrow x, \quad y \Leftrightarrow v_1, \quad z \Leftrightarrow v_2$

Example

QBF

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

Model

$v_1 \Leftrightarrow \bot, \quad v_2 \Leftrightarrow x, \quad y \Leftrightarrow v_1, \quad z \Leftrightarrow v_2$

1. MiniSat proves $\vdash (v_1 \Leftrightarrow \bot) \implies (v_2 \Leftrightarrow x) \implies (y \Leftrightarrow v_1) \implies (z \Leftrightarrow v_2) \implies \phi$

Example

QBF

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

Model

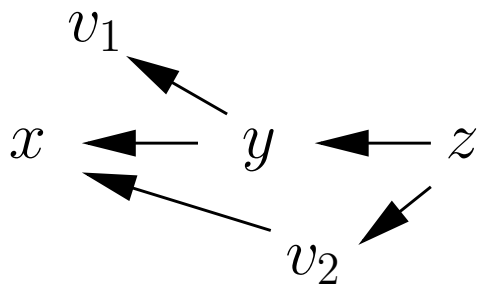
$v_1 \Leftrightarrow \bot, \quad v_2 \Leftrightarrow x, \quad y \Leftrightarrow v_1, \quad z \Leftrightarrow v_2$

1. MiniSat proves $\vdash (v_1 \Leftrightarrow \bot) \implies (v_2 \Leftrightarrow x) \implies (y \Leftrightarrow v_1) \implies (z \Leftrightarrow v_2) \implies \phi$
2. $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:

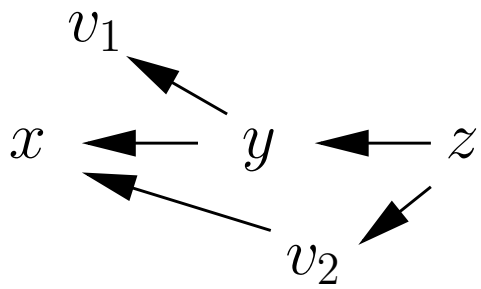


e.g., $z < v_2 < y < x < v_1$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

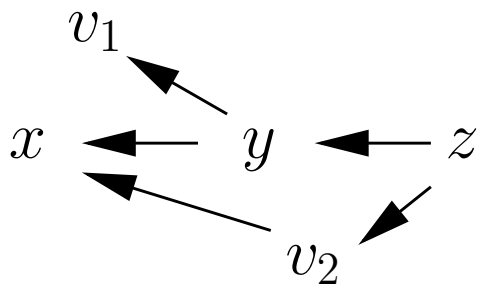
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

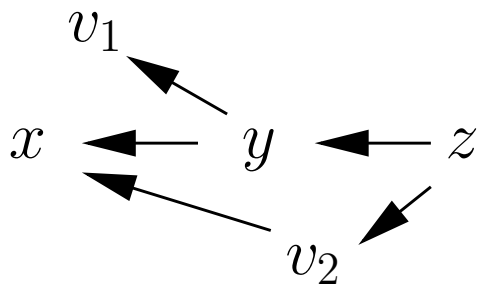
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

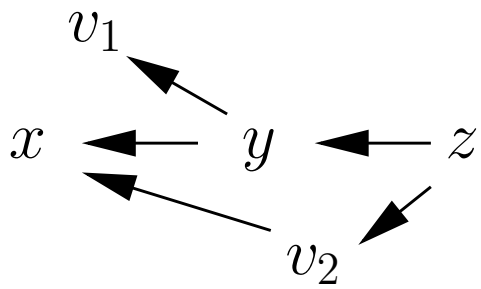
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, v_2 \Leftrightarrow v_2\} \vdash \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

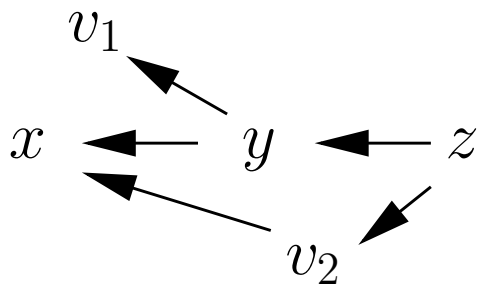
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1\} \vdash \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

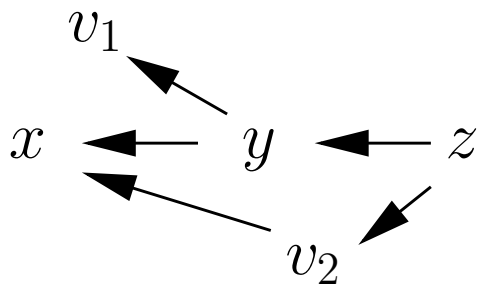
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, x \Leftrightarrow x, y \Leftrightarrow v_1\} \vdash \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

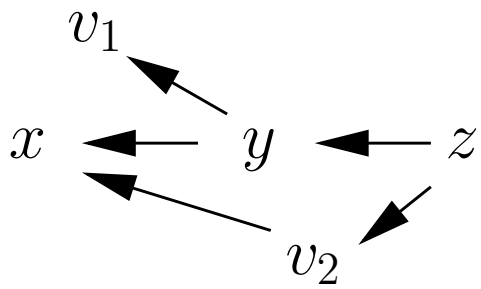
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, y \Leftrightarrow v_1\} \vdash \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

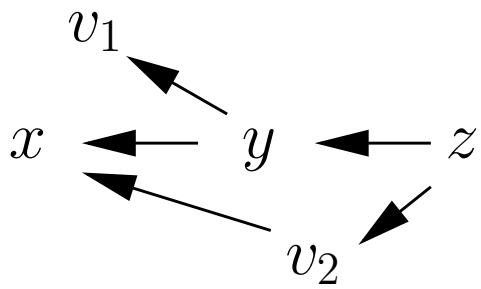
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, y \Leftrightarrow v_1\} \vdash \exists y\, \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

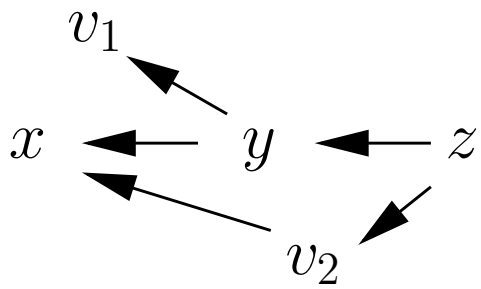
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot, v_1 \Leftrightarrow v_1\} \vdash \exists y\, \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

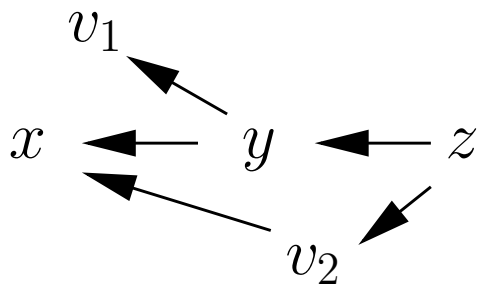
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot\} \vdash \exists y\, \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

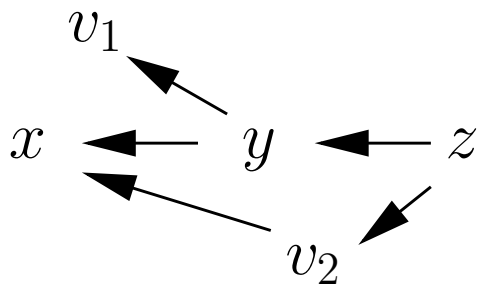
4. Eliminate hypotheses, introduce quantifiers:

$$\{v_1 \Leftrightarrow \bot\} \vdash \forall x\, \exists y\, \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

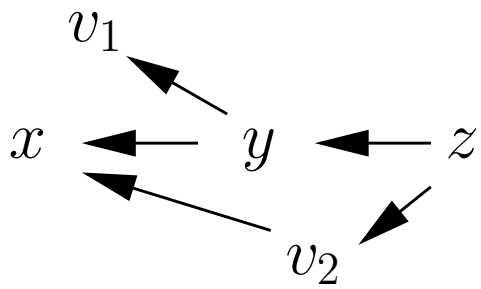
4. Eliminate hypotheses, introduce quantifiers:

$$\{\bot \Leftrightarrow \bot\} \vdash \forall x\, \exists y\, \exists z.\ \phi$$

Example (cont.)

$\forall x\, \exists y\, \exists z.\ \phi$ $\qquad$ $\{v_1 \Leftrightarrow \bot, v_2 \Leftrightarrow x, y \Leftrightarrow v_1, z \Leftrightarrow v_2\} \vdash \phi$

3. Topologically sort all variables:



e.g., $z < v_2 < y < x < v_1$

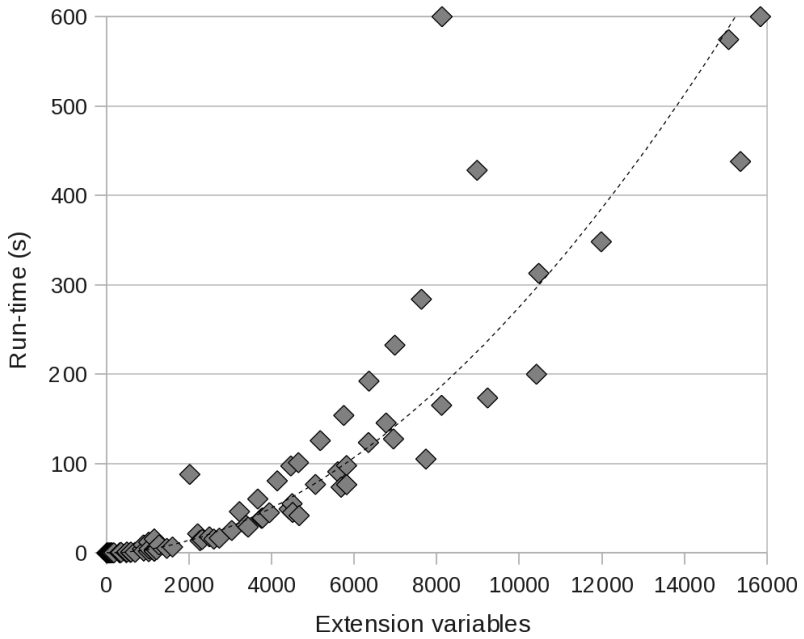
4. Eliminate hypotheses, introduce quantifiers:

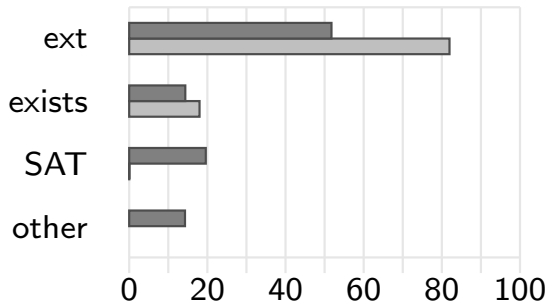
$$\emptyset \vdash \forall x\, \exists y\, \exists z.\ \phi$$

Run-Times and Profiling Results

Evaluation on 100 valid QBF problems from the *2005 fixed instance* and *2006 preliminary QBF-Eval* data sets

up to 133 alternating quantifiers, 11,570 variables, 131,072 clauses





Run-Times and Profiling Results

Evaluation on 100 valid QBF problems from the *2005 fixed instance* and *2006 preliminary QBF-Eval* data sets

up to 133 alternating quantifiers, 11,570 variables, 131,072 clauses

Success rate: 87%

- Average run-times: 134 s (de Bruijn), 163 s (name-carrying)
- Essentially quadratic in the number of extension variables
- 18 times slower than Squolem
- 16 times slower than non-LCF-style validation

Conclusions

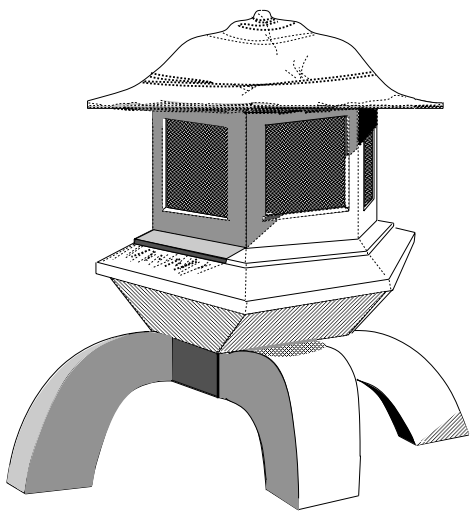
Integration of a QBF solver with HOL4

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

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

Future Work

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



Future Work

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

Thank You!