Proving Valid Quantified Boolean Formulas in HOL Light

Ondřej Kunčar

Charles University in Prague
Automated Reasoning Group
http://arg.mff.cuni.cz

ITP, Nijmegen, August 25, 2011

Quantified Boolean Formula (QBF)

QBF informally

QBF = a propositional formula + quantifiers over Boolean variables

Example of a QBF

$\forall x_1 \, \forall x_2 \, \exists x_3 . \, x_3 \Leftrightarrow ((x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2))$

Applications:

- every finite two-player game can be encoded as a QBF
- in model checking
- in planning
- a natural framework for multiagent settings

Valid Quantified Boolean Formulas

QBF vs. SAT

- QBF can be seen as generalization of SAT problem
- transform the question "$\phi(x_1, x_2, \ldots, x_n)$ is satisfiable?" to
- the question "does $\exists x_1\, \exists x_2 \ldots \exists x_n.\, \phi(x_1, x_2, \ldots, x_n)$ evaluate to true?"

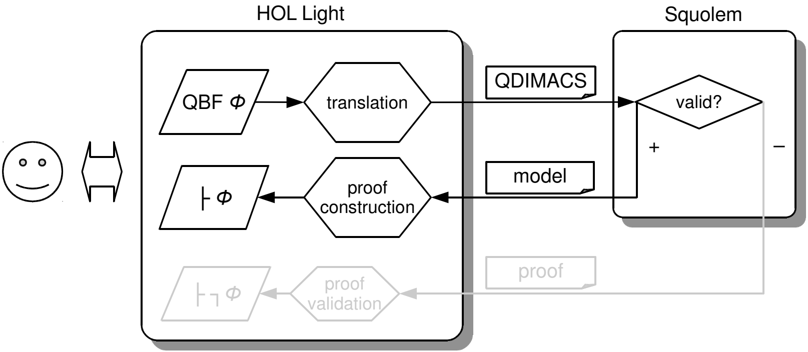
importance of QBF

- "is the given QBF true (=valid)?"
  - the canonical PSPACE-complete problem
- captures many problems in a natural and compact way

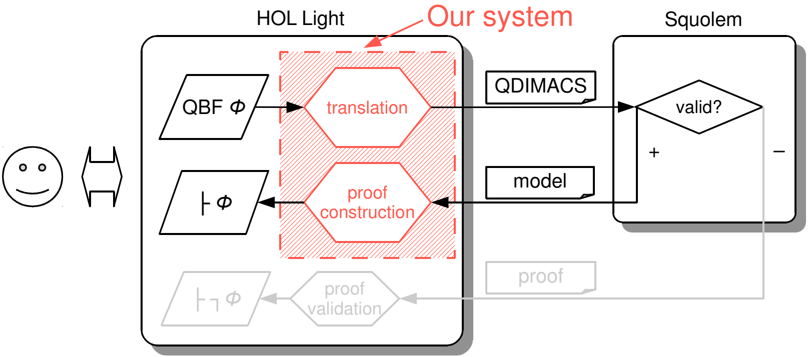
QBF Solvers

- automatically decide validity of QBF
- some of them can generate a certificate
  - which witnesses the output
  - Squolem, sKizzo, yQuaffle, ...
- Squolem – a state-of-the-art QBF solver
  - simple certificates
  - competitive performance

Big Picture


HOL Light
Squolem
QBF Φ
translation
QDIMACS
valid?
+
−
⊢ Φ
proof construction
model
⊢ ¬ Φ
proof validation
proof

Big Picture


HOL Light
Our system
Squolem
QBF Φ
translation
QDIMACS
valid?
+
−
⊢ Φ
proof construction
model
proof
⊢ ¬ Φ
proof validation

Motivation

1. To increase the amount of automation of interactive theorem provers (ITPs).
   - we have to construct a proof
     - lengthy
     - requires a considerable human effort.
2. An independent check of correctness of QBF solvers.
   - QBF solvers are complex tools
   - HOL Light can serve as another independent check
     - the LCF-style kernel provides very high assurance

A Bug in Squolem

- We really found a bug in Squolem.
  - If an input contains tautological clauses:
  - Squolem 1.0 gives an incorrect answer (i.e., invalid)
  - Squolem 2.0 gives a correct answer
    - but still an incorrect certificate
- The bug was resolved in Squolem 2.01.
  - after we pointed out the problem to Ch. Wintersteiger

Related Work

- T. Weber, 2010: Integration of Squolem into HOL4 for invalid formulas
  - based on Q-resolution
- R. Kumar, T. Webber, 2011: Integration of Squolem into HOL4 for valid formulas
  - ITP 2011: in couple of minutes :)
- other integrations
  - Ramana and Tjark are going to tell you more

Squolem's Certificate of Validity: Example

A QBF model = a set of witness functions.

QBF

$\forall x_1\, \forall x_2\, \exists x_3 .\, (x_3 \vee x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee \neg x_1) \wedge (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_3 \vee \neg x_1 \vee \neg x_2)$

Squolem's Certificate/Model

$q_1 \Leftrightarrow x_1 \wedge \neg x_2$

$q_2 \Leftrightarrow \neg x_1 \wedge x_2$

$q_3 \Leftrightarrow (q_1 \wedge q_1) \vee (\neg q_1 \wedge q_2) \quad (= \textit{if } q_1 \textit{ then } q_1 \textit{ else } q_2)$

$x_3 \Leftrightarrow q_3$

Model term

We make a model term from Squolem's certificate C:

Squolem's Certificate/Model

$q_1 \Leftrightarrow x_1 \wedge \neg x_2$

$q_2 \Leftrightarrow \neg x_1 \wedge x_2$

$q_3 \Leftrightarrow (q_1 \wedge q_1) \vee (\neg q_1 \wedge q_2) \quad (= \mathit{if}\ q_1\ \mathit{then}\ q_1\ \mathit{else}\ q_2)$

$x_3 \Leftrightarrow q_3$

Model term

$$\mathfrak{M}_C = (q_1 \Leftrightarrow x_1 \wedge \neg x_2) \wedge (q_2 \Leftrightarrow \neg x_1 \wedge x_2) \wedge$$
$$(q_3 \Leftrightarrow (q_1 \wedge q_1) \vee (\neg q_1 \wedge q_2)) \wedge x_3 \Leftrightarrow q_3$$

$\mathfrak{M}_C$ is called a model term.

Validating Squolem's certificate

Let us have a valid QBF

$$\Phi = Q_1 x_1 \ldots Q_n x_n . \phi \quad ,$$

Squolem's certificate C of Φ and the corresponding model term $\mathfrak{M}_C$.

Observation

The propositional formula $\mathfrak{M}_C \Rightarrow \phi$ is a tautology if and only if C is a model of Φ.

We use already done integration of SAT solvers Minisat and zChaff in HOL Light (T. Weber, H. Amjad, 2009).

Outline of Proof Construction

1. validate that $\mathfrak{M}_C$ is a model of Φ (using SAT solver):

$$\vdash \mathfrak{M}_C \Rightarrow \phi$$

Outline of Proof Construction

1. validate that $\mathfrak{M}_C$ is a model of Φ (using SAT solver):

$$\vdash \mathfrak{M}_C \Rightarrow \phi$$

2. add quantifiers (see the paper):

$$\vdash \mathbf{Q_e}\mathfrak{M}_C \Rightarrow \mathbf{Q}\phi$$

Outline of Proof Construction

1. validate that $\mathfrak{M}_C$ is a model of Φ (using SAT solver):

$$\vdash \mathfrak{M}_C \Rightarrow \phi$$

2. add quantifiers (see the paper):

$$\vdash \mathbf{Q_e}\mathfrak{M}_C \Rightarrow \mathbf{Q}\phi$$

3. prove $\mathbf{Q_e}\mathfrak{M}_C$ by our tactic LIFT (see the paper):

$$\vdash \mathbf{Q_e}\mathfrak{M}_C$$

Outline of Proof Construction

1. validate that $\mathfrak{M}_C$ is a model of Φ (using SAT solver):

$$\vdash \mathfrak{M}_C \Rightarrow \phi$$

2. add quantifiers (see the paper):

$$\vdash \mathbf{Q_e}\mathfrak{M}_C \Rightarrow \mathbf{Q}\phi$$

3. prove $\mathbf{Q_e}\mathfrak{M}_C$ by our tactic LIFT (see the paper):

$$\vdash \mathbf{Q_e}\mathfrak{M}_C$$

4. use modus ponens and derive:

$$\vdash \mathbf{Q}\phi$$

HOL Light is slow

After we implemented optimizations, performance was still poor. Thus we have done some profiling:

- The system spent **99.4 %** of the run-time in HOL Light's kernel function alphaorder!!
- alphaorder implements the order of HOL Light's terms
  - with the property that alpha-equivalent terms are equal
- used for the test that two terms are alpha-equivalent
  - common test: e.g. in modus ponens (MP)

Alphaorder: Implementation

`alphaorder t1 t2`

- go simultaneously through (up to bottom) the structure of `t1` and `t2` and compare recursively smaller parts
- maintain a list of pairs of alpha-equivalent bound variables
- if `t1` is $\lambda x.\, s_1$ and `t2` is $\lambda y.\, s_2$, add the new pair of alpha-equivalent variables (x, y)
- if you need compare two variables, check the list of alpha-equivalent variables first
    - in linear time
- ineffective for formulas with many abstractions
    - for the whole formula in quadratic time
    - our QBFs have thousands of variables $\Longrightarrow$ thousands abstractions

Alphaorder: Optimization

Observation

Alpha-equivalence of two identical terms is even quadratic because the pair (x, y) is added to the list even if x and y are identical variables.

Optimization: don't do that!

- It allows the pointer-EQ shortcut inside terms with abstractions.
- accepted to HOL Light's code in the revision `r83`.
- We measured a speed-up factor of **321.0** due to alpha-equivalence optimization.
  - measured on problems with the time limit 13 seconds

Evaluation

We used the standard *2005 fixed instance* and *2006 preliminary QBF-Eval* data sets – 445 QBF instances.

- Squolem solved 100 instances (valid) = our evaluation data set.

Run-times

time limit (s)	succ. rate (%)	average time (s)	quantifier blocks	variables	clauses
5	33	0.9	41	286	649
60	53	12	53	1378	5458
600	81	73	133	3015	17752
3000	94	248	133	11570	19663

Conclusion: Example

impl04: a QBF instance, only 18 variables.

Without our system:

```
# MESON [] impl04;;
CPU time (user): 1516.384475
```

With our system:

```
# PROVE_QBF impl04;;
CPU time (user): 0.32195
```

Extended Quantifier Prefix

Each extension defines a new fresh variable. We need to incorporate these variables into the quantifier prefix of Φ

Quantifier Prefix

$\mathbf{Q} = \forall x_1 \, \forall x_2 \, \exists x_3$.

$q_1 \Leftrightarrow x_1 \wedge \neg x_2 \quad \rightsquigarrow \quad q_1$ depends on x_1 and x_2

$q_2 \Leftrightarrow \neg x_1 \wedge x_2 \quad \rightsquigarrow \quad q_2$ depends on x_1 and x_2

$q_3 \Leftrightarrow (q_1 \wedge q_1) \vee (\neg q_1 \wedge q_2) \quad \rightsquigarrow \quad q_3$ depends on q_1 and q_2

$x_3 \Leftrightarrow q_3 \quad \rightsquigarrow \quad x_3$ depends on q_3

Extended Quantifier Prefix

$\mathbf{Q_e} = \forall x_1 \, \forall x_2 \, \exists q_2 \, \exists q_1 \, \exists q_3 \, \exists x_3$

How to prove $\mathbf{Q_e}\mathfrak{M}_C$?

Prove $\mathbf{Q}_j E_j$ for each extension and witness assignment:

Example

$q_1 \Leftrightarrow x_1 \wedge \neg x_2 \quad \rightsquigarrow$

$\vdash \forall x_1\, \forall x_2\, \exists q_1\, .\, q_1 \Leftrightarrow x_1 \wedge \neg x_2 \quad (\mathbf{Q}_1 E_1)$

$\vdots \qquad\qquad \vdots$

Use our rule LIFT and do "lifting":

$$\frac{\vdash (\mathbf{Q}_1 E_1) \wedge \cdots \wedge (\mathbf{Q}_N E_N)}{\vdash \mathbf{Q_e}(E_1 \wedge \cdots \wedge E_N) \quad (= \mathbf{Q_e}\mathfrak{M}_C)}\ N-1 \text{ calls of LIFT}$$

LIFT is a key part of our system. See the paper for more details.