Certificate Extraction from Variable-Elimination QBF Preprocessors

Allen Van Gelder
Computer Science Dept.
Univ. of California
Santa Cruz, CA, USA

http://www.cse.ucsc.edu/~avg/

http://www.cse.ucsc.edu/~avg/ProofChecker/
These slides are bloqqer-cert-trans.pdf

http://www.cse.ucsc.edu/~avg/ProofChecker/
Software directory.
Variable Elimination for Solving Quantified Boolean Formulas (QBFs)

**Variable-Elimination Resolution (VER):** Eliminates one existential variable, $e$:
- Do all resolutions on $(e, \overline{e})$; add good ones.
- **Delete all clauses with** $(e, \overline{e})$.
  Davis and Putnam (JACM 1960) for propositional formulas.
  Include **universal reduction** for QBF.

**Universal-Variable Expansion (UVE):** Eliminates one universal variable, $u$:
- For each existential variable $f$ inner to $u$, create two offsprings, $(f^0_u, f^1_u)$.
  $f$ is called the **parent variable**.
- Create two new copies of all clauses containing a parent variable,
  call these sets $G^0_u$ and $G^1_u$.
  call the parent set of clauses $G$.
- Replace $f$ by $f^0_u$ throughout $G^0_u$; replace $f$ by $f^1_u$ throughout $G^1_u$.
- Apply restriction $u = 0$ to $G^0_u$; apply restriction $u = 1$ to $G^1_u$.
- **Delete all clauses in** $G$.

**The Problem:**

What information needs to be recorded to permit a certificate to be created for the original formula (matrix $\mathcal{F}$)?
Adequate Audit Trail, or Trace

Each clause has a unique positive integer identifier

Each variable has a unique positive integer identifier

Each *round* eliminates one variable.

Number rounds from 1 to $m$.

For a preprocessor, a nontrivial formula may remain after round $m$.

Each clause has added fields to record how it came into being.

*By resolution:* Clause1 ID, Clause2 ID, clashing literal (in Clause2).

*By UVE:* parent-clause ID, eliminated universal variable, 0 or 1.

Each created variable has added fields to record how it came into being.

*By UVE:* parent variable, eliminated universal variable, 0 or 1.
Technical Problem: Pushing a Hitting Set Back Through Rounds

**Hitting Set for round** $r$: partial assignment that satisfies each clause in $F_r$.

**Main Idea:**
Given a hitting set for $F_{r+1}$, use the records for new variables in round $r+1$ to decide values for their parent variables in round $r$.

See workshop paper for details.
Toward a Comprehensive Language for QBF Solving

**Scope:** Express formulas, derived constraints, and certificates consistently.

**Design Goals:**
- (Mostly) Compatible with Qdimacs.
- Original formula and new material intermixed.
- Human understandable and easy to parse
- Extensible, so new constructs can be added.

**Motivation:** New work wants to introduce cubes (presented in this conference).
- Extension variables (EBDDRES)
- Universal-Variable Expansion introduces new variables.
- Incremental methods introduce new “original” clauses.

**Current practice:**
- Make up your own format.
- Ignore formats anyone else has made up before.
Proposal for QBF Standard Format (qsf)

Initially: c through EOL is a comment. p through EOL is problem declaration.

Migration: # through EOL is a comment. ! through EOL is problem declaration.

Otherwise: EOL is white-space but not otherwise significant.

Single-Character Token Principle:
Except for comment and problem declaration, any nonnumeric token is a single character, mostly lowercase letters.

An integer is a token (positive, negative, or zero).

White-space, including EOL, separates integers, and is optional to separate single-character tokens.

A constraint statement is a sequence of integers, terminated by zero, without an embedded zero.

Other statements are introduced by a single-character token and terminated by z if they may contain embedded constraints, or by zero.

a, e introduce quantified variables, terminated by zero.
New Features in qsf

( introduces a label or id for a derived or otherwise introduced constraint.
   (Matching ) optional.)

[ introduces a reference variable or constraint.
   (Matching ] optional. See examples.)

Other new tokens:
  r: begin derived clause statement, terminated by z.
  s: begin derived cube statement, terminated by z.
  g: begin derived guard clause statement, terminated by z.
  i: begin QIR derived clause statement, terminated by z.

       Next token would be r, s, or g.
       QIR is chained input resolution, based somewhat on tracecheck.
Quick Examples

\[ r(101) -3 -2 0 100 7 \ z \text{ replaces } QRP 101 -3 -2 0 100 7 0 \]

\[ s(101) -3 -2 0 100 7 \ z \text{ replaces } QRP 101 -3 -2 0 100 7 0 \]

\[ s(101) -3 -2 0 \ z \text{ could mean this is an initial cube, found by magic, just believe it.} \]

\[ ir(101) -3 -2 0 \ t \ 98 \ r \ -23 9 \ r \ -22 8 \ r \ 21 7 \ z \text{ 3-step QIR derivation.} \]

More mathematically, \( C_{98} \otimes_{23} C_{9} \otimes_{22} C_{8} \otimes_{21} C_{7} \)

\[ e \ [42] \ 2001 \ 2002 \ 0 \text{ could mean new variables 2001 and 2002 are in the same quantifier block as 42, likely 42 is the parent and UVE is involved.} \]

\[ u(203) -2001 -2 0 \ [15] -12 \ z \text{ could mean clause 203 came from UVE on 12 with parent clause 15 and 12 := false. Clause 15 may have been } (-42 -2) \]

**Conclusion:** This seems to let us express a large variety of current solving operations for certificate purposes. Some programs can parse and skip statements they do not wish to process.