EXPLOITING QBF DUALITY ON A CIRCUIT REPRESENTATION

Alexandra Gouliaeva and Fahiem Bacchus
Quantified Boolean Formulas (QBF)

- A QBF formula has the form $Q \phi$, where $Q$ is a sequence of universal/existential quantifier and $\phi$ is a propositional formula.
- Determining the truth of a QBF is a PSAPCE-Complete problem
  - A much wider range of problems can be covered
State-of-the-art QBF Solvers

- QBF solvers inherit from SAT solving technology
  - Using Skolemization (QBF $\rightarrow$ SAT)
  - Adapting DPLL search augmented with clause learning
  - + Cube learning
  - All methods use CNF in representing QBF

- CNF or non-clausal for SAT solvers?
  - Losing structure
  - Non-clausal representations can be effective
  - Using CNF can lead to high performance
  - Non-clausal representations could not outweigh the advantages of CNF in SAT solvers

The answer is CNF.
State-of-the-art QBF Solvers

- CNF or non-clausal for QBF solvers?
  - The search space explored by QBF is much larger than SAT solver
  - There is potential for savings from exploiting non-clausal representations

The answer is non-clausal.

- One of the most general and structure laden non-clausal representations is a circuit representation.

CirQit Solver
Circuit Representation

\[ \exists e_1 \forall u_1 \exists e_2 \forall u_2 . (e_1 \land \neg u_1) \lor (e_2 \land u_2) \]

\[ \exists e_1 \forall u_1 \exists x_1 \exists x_2 e_2 \forall u_2 \exists x_3 . (e_1 \land \neg u_1) \lor (e_2 \land u_2) \]
CirQit – A Circuit-based Solver

- CirQit utilizes a DPLL search procedure running on a circuit representation.
  - Propagation, clause and cube learning
- CirQit does backtracking search search

Variables are instantiated respecting the quantifiers ordering.
Solver has to verify both values for universal quantifier
\[ \exists e_1 \forall u_1 \exists e_2 \forall u_2 \cdot (e_1 \land \neg u_1) \lor (e_2 \land u_2) \]
Cube Learning in Circuit Solvers

- Cube learning is initiated once a sufficient number of input lines are set by backward search
  - Any subset of these inputs that suffices to propagate 1 to the output forms a cube.
    - Greedy approach
  - Backtracking to the most recently set universal variable in cube
  - Merging cubes
Limitations of Cube Learning in QBF Solvers

- No cubes at the beginning

- The learnt cubes tend to be very large
  - Cubes learnt for circuit are usually shorter than CNF

- Cubes from a circuit representation include only literals corresponding to set input lines
  - No auxiliary variables in cubes

Q: How to Address these Weaknesses?
A: Dual Propagation.
Dual Propagation

- Cubes and clauses are duals
  - Logically, if propagation forces an input literal $l$, then
    $\neg l$ must make $\neg Q.\phi$ false and $Q.\phi$ true.
  - $\neg Q.\phi$ is false iff $Q.\phi$ is true.
Dual Propagation Intuition

- \( \neg Q \cdot \phi \) is false iff \( Q \cdot \phi \) is true.
- Logically, if propagation forces an input literal \( I \), then \( \neg I \) must make \( \neg Q \cdot \phi \) false and \( Q \cdot \phi \) true.

- Unit propagation on \( \neg Q \cdot \phi \) allows us to steer the solver away from truthifying assignments.
  - Unit propagation on \( Q \cdot \phi \) allows us to steer the solver away from falsifying assignments.

- Conflicts in \( \neg Q \cdot \phi \) identify when \( Q \cdot \phi \) must be true.
- Clauses learnt from these conflicts allow us to detect the assignments that must truthify \( Q \cdot \phi \).
Implementation of Dual Propagation

The formula is TRUE.
Don’t-care Propagation

- DC-1: variables whose value is irrelevant assuming the 1-value of the output
- DC-0: variables whose value is irrelevant assuming the 0-value of the output.

- The solver need never branch on any variable that is don’t care on either channel
- Performance improvement
Experimental Results – CirQit2 vs CirQit

- The effect of dual propagation on performance
CirQit2 vs. Other Solvers

- Bottom-up solvers (Quantor) are better suited for “assertion”, “consistency” and “possibility”.
- CirQit2 is unable of solving a problem than depqbf can solve.
- It outperforms other solvers otherwise.
- CirQit2 is able to solve all the problems in less time than its CirQit took to solving only some.
CirQit2 vs. Other Solvers

This figure shows the number of problems solved by the solvers as time goes on.
Conclusion

- Circuit representation for QBF solvers
- CirQit performs unit propagation, clause and cube learning.
- CirQit2 improves clause learning effectiveness by reconciling clauses and cubes.
- Results show performance improvement.