Nenofex: QBFEVAL'10 Solver Description

Florian Lonsing

Institute for Formal Models and Verification
Johannes Kepler University, Linz, Austria
http://fmv.jku.at

1 Background

Nenofex (**Ne**gation **No**rmal **F**orm **Ex**pansion) is an expansion-based QBF solver which uses *negation normal form (NNF)* as internal formula representation [5]. The overall algorithm is inspired by Quantor [3] which in contrast operates on CNF and applies resolution and expansion for variable elimination. The motivation for using expansion on NNF instead of resolution on CNF is that resolution produces a quadratic number of resolvents whereas with expansion the formula grows linearly.

An NNF is implemented as a tree, i.e. there is no structural sharing of parts of the formula like in *and-inverter graphs (AIGs)*. The tree structure allows many core algorithms to be implemented simpler, yet on the other hand increases memory consumption.

Nenofex takes as input QBFs in *prenex conjunctive normal form (PCNF)* and successively eliminates universally and existentially quantified variables from the two innermost quantifier sets by Shannon expansion. The expansion schedule takes into account the amount of formula size increase, called expansion costs, that is caused by expanding a particular variable. An expansion requires to add a copy of the relevant subformula and assign the expanded variable accordingly. The relevant subformula is determined by the *least common ancestor (LCA)* of literal nodes in the NNF. When expanding universal variables, dependencies have to be captured by the relevant subformula as well.

A greedy strategy is pursued where variables with minimal costs are expanded first. In general, expansions of universal variables not from the innermost quantifier set are deferred until the formula size increase in recent expansions exceeds a certain threshold. This threshold is increased with every universal expansion.

In order to improve formula size after expansions two approaches from circuit minimization are applied. *Global flow (GF)* [4] analysis allows to transform a circuit based on identified implications of values at certain gates. *Redundancy removal (RR)* based on *automatic test pattern generation (ATPG)* [1] assumes faults at circuit gates and checks whether those faults modify the circuit's function. If not then the gate is redundant and can be removed.

Our implementation of RR runs in polynomial-time but is incomplete in the sense that not all redundancies can be detected. Both GF and RR are applied in cyclic fashion on a small part of the NNF until a heuristic stop criterion is

met. Apart from these two approaches detection of *unit* and *pure literals* (also called *unates* or *monotone literals*) allows to further simplify the formula.

If at any time the formula contains either only existential or only universal variables then the NNF is converted to a propositional CNF using a structural encoding [6][7][8]. This CNF is then forwarded to a SAT solver.

Fig. 1 shows the individual phases of the algorithm from reading the input formula (INIT), elimination of detected unit and pure literals (UNITS, UNATES), redundancy elimination (GF, RR), expansion (EXP) and sat solving (SAT). Either phase can determine the result of the formula.

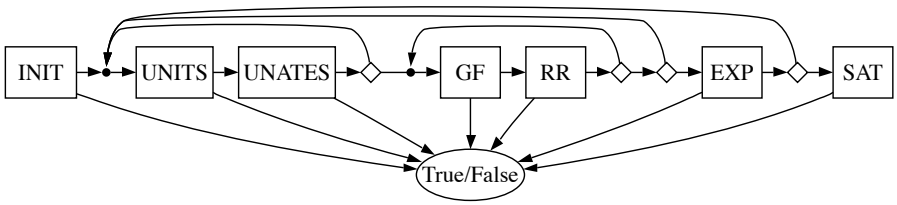

Fig. 1. Workflow

2 Submission to QBFEVAL'10

The version submitted to QBFEVAL'10 is essentially the same as the one which participated in QBFEVAL'08. The major differences are some tuned parameters and a more recent version of Picosat [2] as SAT solver back end.

References

1. Agrawal, V., Bushnell, M.: Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Kluwer (2000)
2. Biere, A.: Picosat essentials. JSAT 4(2-4), 75–97 (2008)
3. Biere, A.: Resolve and expand. In: Proc. SAT'04
4. Kunz, W., Stoffel, D.: Reasoning in Boolean Networks: Logic Synthesis and Verification Using Testing Techniques. Kluwer (1997)
5. Lonsing, F., Biere, A.: Nenofex: Expanding NNF for QBF Solving. In: Proc. SAT'08
6. Plaisted, D., Greenbaum, S.: A structure-preserving clause form translation. Symb. Comput. 2(3) (1986)
7. de la Tour, T.B.: An optimality result for clause form translation. Symb. Comput. 14(4) (1992)
8. Tseitin, G.: On the complexity of derivation in propositional calculus. Studies in Constructive Mathematics and Mathematical Logic 2 (1968)