depqbf: QBFEVAL'10 Solver Description

Florian Lonsing

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

1 Background

Many QBF solvers take as input formulae in prenex conjunctive normal form (PCNF). When converting an arbitrary QBF into PCNF structural properties of the original formula can be lost. In particular the linear quantifier prefix of PCNFs has to be respected by QBF solvers, which imposes a total ordering on the quantifier sets. This ordering results in strong variable dependencies which can influence solver performance negatively.

Dependency schemes [10] provide a general formalism for expressing variable dependencies in QBFs in terms of binary relations over the set of variables. In [10] schemes such as the *standard dependency scheme* D^{std} have been introduced which are (potentially) less restrictive than the total variable ordering in PCNFs: variables are not totally but partially ordered. This can grant QBF solvers more freedom in the solution process.

We suggested a compact representation for D^{std} as a graph over equivalence classes of variables in [7][8]. We argue that such a dependency graph can be used within a QBF solver to overcome the limitations of linear quantifier prefixes in PCNFs. For similar approaches, we refer to the references given in [7].

Our solver depqbf [1] is based on the DPLL algorithm for QBF (see e.g. [1]) such as e.g. QuBE or Quaffle and takes PCNFs as input. It integrates compact dependency graphs as presented in [7] to be used instead of the linear quantifier prefix. The core features are as follows:

- dependency graph (DG) for D^{std}: before solving starts DG is extracted from the input formula given in PCNF.
- re-initialization of DG: whenever the solver identifies new fixed (top-level) assignments DG is re-computed after the formula has been simplified under the set of all top-level assignments. This amounts to some sort of "semi-dynamic" treatment of dependencies.
- dependency manager (DM): DM uses DG to maintain dependency information which is forwarded to the solver. Before decision making the solver queries DM to retrieve *decision candidates* – the set of variables which can be assigned as decisions – under the solver's current partial assignment. Care must be taken not to violate D^{std}. This is done by incrementally keeping

[1] working title

track of assigned variables within DM and DG. DM is notified after assignments in the solver (decisions, unit or pure literals) and updates the set of decision candidates according to DG.

Further depqbf integrates techniques common to many search-based QBF solvers such as

- clause and cube learning [4][6][11]
- two-literal watching to detect unit literals [2]
- watched constraints to detect pure literals [2][3]
- restarts and phase saving [5][9]

depqbf is still work in progress and is considered an experimental prototype to study the effects of using D^{std} in practice.

2 Submission to QBFEVAL'10

We submitted two versions of depqbf to QBFEVAL'10, a basic version and another one which uses a new implementation of quantor for preprocessing.

References

1. Cadoli, M., Giovanardi, A., Schaerf, M.: An Algorithm to Evaluate Quantified Boolean Formulae. In: AAAI/IAAI. pp. 262–267 (1998)
2. Gent, I.P., Giunchiglia, E., Narizzano, M., Rowley, A.G.D., Tacchella, A.: Watched Data Structures for QBF Solvers. In: Giunchiglia, E., Tacchella, A. (eds.) SAT. LNCS, vol. 2919, pp. 25–36. Springer (2003)
3. Giunchiglia, E., Narizzano, M., Tacchella, A.: Monotone Literals and Learning in QBF Reasoning. In: Wallace, M. (ed.) CP. LNCS, vol. 3258, pp. 260–273. Springer (2004)
4. Giunchiglia, E., Narizzano, M., Tacchella, A.: Clause/Term Resolution and Learning in the Evaluation of Quantified Boolean Formulas. J. Artif. Intell. Res. (JAIR) 26, 371–416 (2006)
5. Gomes, C.P., Selman, B., Kautz, H.A.: Boosting Combinatorial Search Through Randomization. In: AAAI/IAAI. pp. 431–437 (1998)
6. Letz, R.: Lemma and Model Caching in Decision Procedures for Quantified Boolean Formulas. In: Egly, U., Fermüller, C.G. (eds.) TABLEAUX. LNCS, vol. 2381, pp. 160–175. Springer (2002)
7. Lonsing, F., Biere, A.: A Compact Representation for Syntactic Dependencies in QBFs. In: Kullmann, O. (ed.) SAT. LNCS, vol. 5584, pp. 398–411. Springer (2009)
8. Lonsing, F., Biere, A.: Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers. ENTCS 251, 83 – 95 (2009)
9. Pipatsrisawat, K., Darwiche, A.: A Lightweight Component Caching Scheme for Satisfiability Solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT. Lecture Notes in Computer Science, vol. 4501, pp. 294–299. Springer (2007)
10. Samer, M., Szeider, S.: Backdoor Sets of Quantified Boolean Formulas. Journal of Automated Reasoning (JAR) 42(1), 77–97 (2009)
11. Zhang, L., Malik, S.: Conflict driven learning in a quantified Boolean Satisfiability solver. In: Pileggi, L.T., Kuehlmann, A. (eds.) ICCAD. pp. 442–449. ACM (2002)