TRP++: Implementation of Clausal Resolution for Propositional Linear-Time Temporal Logic

Boris Konev*

Logic and Computation Group, Department of Computer Science
University of Liverpool, Liverpool L69 7ZF, UK
B.Konev@csc.liv.ac.uk

1 Introduction

Temporal logics are extensions of classical logic, with operators that deal with time. They have been used in a wide variety of areas within Computer Science and Artificial Intelligence, for example robotics [18], databases [19], hardware verification [11] and agent based systems [17]. In particular, propositional temporal logics have already made significant impact within Computer Science, having been applied to:

- the specification and verification of reactive (e.g. distributed or concurrent) systems [15];
- the synthesis of programs from temporal specifications [16, 14];
- the semantics of executable temporal logic [8, 9];
- algorithmic verification via model-checking [10, 2]; and
- knowledge representation and reasoning [4, 1, 20].

In developing these techniques, temporal proof is often required, and we base our work on practical proof techniques on the clausal resolution approach to temporal logic.

The clausal resolution method for propositional linear-time temporal logics provides the basis for a number of temporal provers. The method is based on an intuitive clausal form, called SNF, comprising 3 main clause types and a small number of resolution rules [5]. While the approach has been shown to be competitive [12, 13], we aim at an even more efficient implementation.

As usual, the temporal resolution approach aims at deduction of a contradiction from a given set of formulas.

2 Separated Normal Form

A *propositional temporal problem*, whose satisfiability we are interested in, consists of four parts[1]:

1. a universal part $\mathcal{U}$, given by a set of propositional clauses;
2. an initial part $\mathcal{I}$ with the same form as the universal part;
3. a step part $\mathcal{S}$, given by a set of propositional step temporal clauses of the form:

$$L_1 \wedge L_2 \wedge \ldots \wedge L_k \Rightarrow \bigcirc(M_1 \vee M_2 \vee \ldots \vee M_l) \qquad (\textit{step } \text{clause}),$$

   where L_i, M_j are literals;
4. an eventuality part $\mathcal{E}$, given by a set of propositional eventuality temporal clauses of the form:

$$L_1 \wedge L_2 \wedge \ldots \wedge L_k \Rightarrow \Diamond(M),$$

   where L_i, M are literals.

We say that a problem is satisfiable if, and only if, the formula $\mathcal{I} \wedge \Box\mathcal{U} \wedge \Box\mathcal{S} \wedge \Box\mathcal{E}$ is satisfiable.

It is known [5] that a PLTL-formula is satisfiable if, and only if, a set of temporal clauses is satisfiable. To translate a temporal formula into the SNF form (see [5] for full details), we essentially apply a set of the transformation rules based upon the renaming of complex expressions by new propositions and upon the substitution of temporal operators by their fixpoint definitions.

* On leave from Steklov Institute of Mathematics at St.Petersburg

[1] The **TRP++** system uses slightly different normal form than described in [5] in that we allow universal clauses; each universal clause C can be represented by a pair $\{\mathbf{start} \Rightarrow C, \mathbf{true} \Rightarrow \bigcirc(C)\}$ in terms of [5].

3 Temporal resolution

The deduction system for temporal problems essentially consist of the step resolution rule (that could be thought of as an ordinary resolution rule acting on "arithmetization" of the initial, universal, and step parts) and the eventuality resolution rule that involves the detection of a set of formulas that together imply $\Box\neg l$ (known as a *loop in* $\neg l$) for resolution with $\Diamond l$. Note that for arithmetization, we represent temporal clauses as (a very restrictive form of) first order clauses, namely, an initial clause C will be represented as $C(0)$, a universal clause D as $D(x)$, and a step clause $(\bigwedge L_i \Rightarrow \bigcirc(\bigvee M_j))$ as $(\bigvee \neg(L_i(x)) \vee \bigvee(M_j(f(x)))$.

Then, any first-order resolution system would allow us to perform step resolution steps; however, due to very restrictive nature of the formulas, **TRP++** uses it's own "near propositional" approach to deal with them. Further, search for loops (that is, the most costly part of the method) can be implemented by similar techniques to those described in [3].

4 Acknowledgment

The author would like to thank A. Degtyarev, C. Dixon, U. Hustadt, and M. Fisher for fruitful discussions about the subject. The author would like to thank the both EPSRC (via grant GR/L87491) and the University of Liverpool (via Research Development Funding) for their support in this work.

References

1. A. Artale and E. Franconi. Temporal description logics. In *Handbook of Time and Temporal Reasoning in Artificial Intelligence*. To appear.
2. E. Clarke, O. Grumberg and D. A. Peled. *Model Checking.* MIT Press, 2000.
3. C. Dixon Temporal Resolution using a Breadth-First Search Algorithm. in *Annals of Mathematics and Artificial Intelligence.* volume 22, pages 87–115, special issue on Temporal Representation and Reasoning 1998. Baltzer Science Publishers. ISSN 1012-2443.
4. R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge.* MIT Press, 1996.
5. M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computation Logic*, 2(1), January 2001.
6. M. Fisher. A resolution method for temporal logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91).* Morgan Kaufman, 1991.
7. M. Fisher. A normal form for first-order temporal formulae. In Proc.of 11th International Conference on Automated Deduction(CADE'11), volume 607 of *Lecture Notes in Computer Science*. Springer Verlag, 1992.
8. M. Fisher. An introduction to executable temporal logics. *Knowledge Engineering Review*, 11:43–56, 1996.
9. M. Fisher. A temporal semantics for concurrent MetateM. *J. Symbolic Computation*, 22(5/6):627–648, 1996.
10. G.J. Holzmann. *Design and Validation of Computer Protocols.* Prentice Hall, New Jersey, 1991.
11. G. J. Holzmann. The Model Checker Spin. *IEEE Trans. on Software Engineering*, 23(5):279–295, May 1997. Special Issue on Formal Methods in Software Practice.
12. U. Hustadt and R. Schmidt. Scientific Benchmarking with Temporal Logic Decision Procedures *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, 2002 (to appear).
13. U. Hustadt and R. Schmidt. Formulae which Highlight Differences between Temporal Logic and Dynamic Logic Provers In *Issues in the Design and Experimental Evaluation of Systems for Modal and Temporal Logics.* pp.68–76, Siena, Italy, 2001.
14. O. Kupferman and M. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic.* pp.109-128. Kluwer, 2000.
15. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer, 1992.
16. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. Princip. Prog. Lang.*, pp.179–190, 1989.
17. A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–342, 1998.
18. M. P. Shanahan. *Solving the Frame Problem.* MIT Press, 1997.
19. A. Tansel, editor. *Temporal Databases: theory, design, and implementation.* Benjamin/Cummings, 1993.
20. F. Wolter and M. Zakharyaschev. Temporalizing description logics. In *Frontiers of Combining Systems II*, pp.379–401. Research Studies Press LTD, England, 2000.