Algorithms for Guiding Clausal Temporal Resolution

M.Carmen Fernández Gago,
Logic and Computation Group
Department of Computer Science,
University of Liverpool,
Peach Stret, L69 7ZF, United Kingdom
Email: M.C.Gago@csc.liv.ac.uk

1 Introduction

Temporal logic is a variety of non-classical logic used in a range of areas within Computer Science and Artificial Intelligence, for example the verification of reactive systems [8], the implementation of temporal query languages [2], and temporal logic programming [1]. Consequently, different proof methods have been developed, implemented and applied.

In this work we use a proof method for temporal logics based upon resolution for classical logic [6]. Resolution based methods have the advantage that, as in the classical case, a wide range of strategies can potentially be used.

The temporal resolution procedure is characterised by the translation to a normal form, the application of a classical style resolution rule between formulae that occur at the same moment in time, termed *step resolution*, that is, where the clauses contain no eventualities ($\Diamond l$, i.e, *l holds now or at sometime in the future*), together with a novel temporal resolution rule, which derives contradictions over temporal sequences (*temporal resolution*). Although the clausal temporal resolution method has been defined, proved correct and implemented it sometimes generates an unneccesarily large set of formulas that may be irrelevant to the refutation. Not only that, but temporal resolution operations are applied only after all step resolution inferences have been carried out. This means that cases where a large amount of step resolution can occur the method may be very expensive.

The application of the temporal resolution rule involves searching for a set of clauses that together represent $X \Rightarrow \bigcirc \Box l$ for resolution with $\Diamond \neg l$. Rather than using an external procedure to do this, as happens in [7], we provide an algorithm to detect X systematically using only step resolution. If we need to search for a second loop we have developed an additional algorithm that can reuse the clauses generated during previous searches rather than recreating then as is necessary in [3].

2 The Search for Loops

For the application of the temporal resolution rule a loop must be detected, that is, a set of clauses $\phi_i \Rightarrow \bigcirc\psi_j$, to be resolved with $\Diamond\neg l$, such that $\bigwedge_{i=1}^{n}(\phi_i \Rightarrow \bigcirc(l \wedge \bigvee_{j=1}^{n} \phi_j))$.

Trying to find such a set of clauses we propose the following algorithm where we begin choosing a candidate for $G_i = \bigvee_{j=1}^{n} \phi_i$

1. Choose $G_{-1} \Leftrightarrow \mathbf{true}$
2. Given a guess G_i add the clause $s_i \Rightarrow \bigcirc(\neg G_i \vee \neg l)$ and apply Step Resolution.
3. For all clauses $\mathbf{true} \Rightarrow \bigcirc(\neg s_i \vee F_j)$ obtained during the proof, let $G_{i+1} \Leftrightarrow G_i \wedge (\bigvee_{j=1}^{m} \neg F_j)$.
4. Go to 2 until either

(a) $G_i \Leftrightarrow G_{i+1}$ (we terminate having found a loop).

(b) G_{i+1} is empty. (we terminate without having found a loop).

We can show

Theorem 1 (Soundness) *If the above algorithm outputs X when applied to $\Diamond\neg l$ then $X \Rightarrow \bigcirc\Box l$.*

Theorem 2 (Completeness) *If a set of clauses contains a loop X such that $X \Rightarrow \bigcirc\Box l$ then the algorithm detects X' such that $X \Rightarrow X'$.*

Proofs can be found in [5]. Essentially the proofs are based on [3].

Also in [5] we give an algorithm for searching for a second or subsequent loops that reuses clauses generated by the previous searches.

In the future we hope to apply these results to the development of strategies for temporal resolution that allows us to reduce the search space. In particular, we are interested in incorporating the set of support strategy [9, 4], to guide the search for a proof in temporal resolution, with the expectation that it will be success as in the classical case.

References

[1] M. Abadi and Z. Manna. Temporal Logic Programming. *Journal of Symbolic Computation*, 8: 277–295, 1989.

[2] J. Chomicki and D. Niwinski. On the Feasibility of Checking Temporal Integrity Constraints. *Journal of Computer and System Sciences*, 51(3):523–535, December 1995.

[3] C. Dixon. Temporal Resolution using a Breadth-First Search Algorithm. *Annals of Mathematics and Artificial Intelligence*, 22:87–115, 1998.

[4] C. Dixon and M. Fisher. The Set of Support Strategy in Temporal Resolution. In *Proceedings of TIME-98 the Fifth International Workshop on Temporal Representation and Reasoning*, Sanibel Island, Florida, May 1998. IEEE Computer Society Press.

[5] M. C. Fernández-Gago, M. Fisher, and C. Dixon. Algorithms for Guiding Clausal Temporal Resolution. Current Paper. In preparation.

[6] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 99–104, Sydney, Australia, August 1991. Morgan Kaufman.

[7] M. Fisher, C. Dixon, and M. Peim. Clausal Temporal Resolution. In *Transactions on Computational Logic 2(1)*, January 2001.

[8] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[9] L. Wos, G. Robinson, and D. Carson. Efficiency and Completeness of the Set of Support Strategy in Theorem Proving. *ACM Journal*, 12:536–541, October 1965.