Algorithms for Guiding Clausal Temporal Resolution

M.Carmen Fernández-Gago,
Department of Computer Science,
University of Liverpool
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 [6], the implementation of temporal query languages [2], and temporal logic programming [1]. Consequently, different proof methods have been developed, implemented and applied.
Here we use a proof method for temporal logics based upon the use of classical resolution [4]. The resolution procedure is characterised by the translation to a normal form, the application of a classical style resolution between formulae that occur at the same moment in time (*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 occur only after all step resolution inferences have been carried out before that. This means that cases where a large amount of step resolution can occur the method may be very expensive.
Temporal resolution also requires the detection of a set of clauses known as a *loop*, that is, formulae that together imply $\Box l$ (*l holds now and at all moments in the future*) for resolution with $\Diamond \neg l$.
As the search for the candidates for a loop is certain to be required but, it is the most expensive part of the resolution operations our intention is to try to avoid all unnecessary step resolution operations, and apply temporal resolution earlier. In this sense we propose an algorithm based on the outputs of a refined temporal operation [5].

2 An Algorithm based on a Refined Rule

The temporal resolution rule is as follows:

$$\frac{\begin{array}{ccc} \phi_1 & \Rightarrow & \bigcirc\psi_1 \\ \vdots & \vdots & \vdots \\ \phi_n & \Rightarrow & \bigcirc\psi_n \\ \chi & \Rightarrow & \Diamond\neg l \end{array}}{\chi \Rightarrow (\neg \bigvee_{i=1}^{n} \phi_i)\mathcal{W}\neg l}$$

where we know $\bigwedge_{i=1}^{n}(\phi_i \Rightarrow \bigcirc(l \wedge \bigvee_{i=1}^{n} \phi_j))$ (note that $\bigcirc$ means *in the next moment in time* and $\phi\mathcal{W}\psi$ means *ϕ holds unless ψ holds*). This condition ensures that the set of $\phi_i \Rightarrow \bigcirc\psi_i$ rules together imply $\bigvee_{i=1}^{n} \phi_i \Rightarrow \bigcirc\Box l$.
We adopt the new version of the temporal resolution operation described in [5]. The basic idea behind the new operation is that, rather than insisting we already have a set of clauses that characterise a loop, we derive a more complex resolvent that allows for the possibility that such a loop does not exist. If the

set of clauses chosen do comprise a loop, then the new resolvent turns out to be the resolvent from the original temporal resolution rule. More importantly, if the clauses chosen do not comprise a loop, we can use the resolvents produced to guide further step resolution in such a way that allows us to detect a loop where possible. This new temporal resolution rule is:

$$\frac{\begin{array}{ccc} \phi_1 & \Rightarrow & \bigcirc\psi_1 \\ \phi_2 & \Rightarrow & \bigcirc\psi_2 \\ \vdots & \vdots & \vdots \\ \phi_n & \Rightarrow & \bigcirc\psi_n \\ \chi & \Rightarrow & \Diamond\neg l \end{array}}{\chi\Rightarrow\left[(\Diamond\neg\bigwedge_{i=1}^{n}(\phi_i \Rightarrow \bigcirc(l\wedge\bigvee_{j=1}^{n}\phi_j))) \Rightarrow (\neg\bigvee_{i=1}^{n}\phi_i)\mathcal{W}\neg l\right]}$$

Notice that now there is no side-condition. This has been incorporated into the resolvent as:

$$\Box(\bigwedge_{i=1}^{n}(\phi_i \Rightarrow \bigcirc(l\wedge\bigvee_{j=1}^{n}\phi_j)))$$

As the choice of ϕ_i seems to be crucial, our main aim is to guide the search for a loop using this new temporal resolution rule. More specifically, we will use the output from applying the above revised temporal resolution rule to a "guessed" loop to guide the subsequent guesses. We proceed as follows. We guess a formula for the ϕ_i such that if our guess is correct, that is, $\bigvee_{i=1}^{n}\phi_i \Rightarrow \bigcirc\Box l$, then the loop search phase terminates. Otherwise we use output from subsequent resolution steps to make a new guess based on the previous one. This process terminates with the detection of a loop if one exists.

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 [7, 3], 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 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.

[4] 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.

[5] M. Fisher and C. Dixon. Guiding Clausal Temporal Resolution. In *Advances in Temporal Logic*, volume 16 of *Applied Logic Series*, pages 167–184. Kluwer, 2000. Proceedings The Second International Conference on Temporal Logic (ICTL). ISBN 0-7923-6149-0.

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

[7] 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.