TSPASS - a Fair Monodic Temporal Logic Prover

Michel Ludwig Ullrich Hustadt

Department of Computer Science, University of Liverpool, Liverpool, UK
{Michel.Ludwig, U.Hustadt}@liverpool.ac.uk

1 Introduction

First-Order Temporal Logic, FOTL, is an extension of classical first-order logic by temporal operators for a discrete linear model of time (isomorphic to $\mathbb{N}$). The set of valid formulae of this logic is not recursively enumerable. However, the set of valid *monodic* formulae is known to be finitely axiomatisable.

A first resolution-based calculus for monodic first-order temporal logic was introduced in Degtyarev et al. (2003). Then, a more machine-oriented version, the fine-grained first-order temporal resolution calculus, was described in Konev et al. (2005). A refinement of fine-grained temporal resolution, the ordered fine-grained temporal resolution with selection calculus, is presented in Hustadt et al. (2005). However, while these calculi represent important steps towards fully automated reasoning in the monodic fragment, they still all have one major drawback: they contain inference rules, reflecting the inductive nature of reasoning in this logic, whose applicability is only semi-decidable as they depend on first-order side conditions which in general are not decidable. This poses a problem for the development of refutation-complete theorem provers based on these calculi.

In more detail, resolution-based calculi for monodic first-order temporal logic require that a given set of monodic temporal formulae is transformed in a satisfiability equivalence preserving way into a clausal form consisting of four types of temporal clauses, namely *initial*, *universal*, *step* and *eventuality* clauses. These clauses are then used in inferences by the rules of the monodic fine-grained temporal resolution calculus. The majority of the rules, the so-called step resolution rules, are based on standard (ordered) first-order resolution between different types of temporal clauses. The remaining inference rules, the ground and the non-ground *eventuality resolution* rule, reflect the induction principle that holds for monodic temporal logic over a flow of time isomorphic to the natural numbers. We present the non-ground eventuality resolution rule; the ground version is similar:

$$\frac{\forall x(\mathcal{A}_1(x) \Rightarrow \bigcirc\mathcal{B}_1(x)) \quad \dots \quad \forall x(\mathcal{A}_n(x) \Rightarrow \bigcirc\mathcal{B}_n(x)) \qquad \Diamond L(x)}{\forall x \bigwedge_{i=1}^{n} \neg\mathcal{A}_i(x)} \ (\Diamond^{\mathcal{U}}_{res}),$$

where $\forall x(\mathcal{A}_i(x) \Rightarrow \bigcirc \mathcal{B}_i(x))$ are complex combinations of step clauses such that for all $i \in \{1, \dots, n\}$, the *loop* side conditions $\forall x(\mathcal{U} \wedge \mathcal{B}_i(x) \Rightarrow \neg L(x))$ and $\forall x(\mathcal{U} \wedge \mathcal{B}_i(x) \Rightarrow \bigvee_{j=1}^{n}(\mathcal{A}_j(x)))$, with $\mathcal{U}$ being the current set of all universal clauses, are both valid. The formula $\bigvee_{j=1}^{n} \mathcal{A}_j(x)$ is called a *loop formula* (for $\Diamond L(x)$).

In the realisation of the eventuality resolution rules a special resolution-based algorithm, called loop search algorithm, is used to find $\forall x(\mathcal{A}_i(x) \Rightarrow \bigcirc \mathcal{B}_i(x))$ for an eventuality $\Diamond L(x)$ satisfying the loop side conditions of the eventuality resolution rule. To do so, the loop search algorithm constructs a sequence of sets containing universal and step clauses, which are then saturated under a subset of the rules of fine-grained step resolution. For each attempt to apply a eventuality resolution rule an instance of the loop search algorithm needs to be executed. As a consequence of the undecidability of the validity of the loop side conditions, executions of the loop search algorithm cannot be guaranteed to terminate. It is therefore possible, for example, that a partial loop formula, which is essential for a refutation, has been discovered by the algorithm but, due to an infinite saturation process, this loop formula is never used in the computation of a consequence of an application of the eventuality resolution rule, and, therefore, is not available to construct the refutation. Moreover, if we try to solve the problem by delaying the application of the eventuality resolution rules until the set of temporal clauses is first saturated under inferences with all other rules, then it might happen that the loop search algorithm will not be executed at all as the saturation under these other inference rules is also not guaranteed to terminate.

To solve this problem we have developed a calculus which only contains inference rules whose applicability is decidable. The basic idea underlying the calculus is that with each clause set which occurs in the sequence of clause sets constructed by an invocation of the loop search algorithm we can associate a unique marker literal which is added to every clause occurring in such a set. In the loop search algorithm we can then work with just one clause set in which clauses are separated by those marker literals, instead of constructing a sequence of clause sets. Furthermore, we do not even have to use different instances of the loop search algorithm for each application of an eventuality resolution rule, but can instead use one global clause set incorporating all of them. This then allows us to fairly perform step resolution inferences

Problem	Clauses Generated		Time		Result
	TeMP	TSPASS	TeMP	TSPASS	
0	19611	5707	0.481s	0.386s	Satisfiable
1	21812	833	0.519s	0.075s	Unsatisfiable
2	-	4827	-	0.372s	Unsatisfiable
12	689	793	0.029s	0.073s	Unsatisfiable
18	32395	5262	0.975s	0.389s	Unsatisfiable

Table 1: Results obtained for the robot specification examples

which essentially drive forward all computations in a parallel way, whereas they were executed sequentially in the original approach. This fair inference architecture has been presented in Ludwig and Hustadt (2008a) and in more detail in Ludwig and Hustadt (2008b).

2 Implementation and Experimental Results

The fair inference architecture has been implemented in the theorem prover TSPASS[1], which is based on the first-order resolution prover SPASS 3.0.

The practical performance of TSPASS has been analysed on different temporal problems and we have found it to be competitive with TeMP. In the monodic first-order temporal logic prover TeMP (Hustadt et al., 2004), which is based on the sequential reasoning approach described above, the first-order prover Vampire is used as a black box which saturates sets containing the first-order translations of temporal clauses.

The experimental setting was as follows: the experiments were run on a PC equipped with an Intel Core 2 6400 CPU and 3 GB of main memory. The execution timeout on each problem was set to 12 minutes. For TeMP the input problems were first transformed into its clausal input form and then TeMP was started on this clausal input without any additional settings. TSPASS was instructed to perform subsumption-based loop search testing.

Table 1 shows the satisfiability status, the number of clauses generated and the median CPU time in seconds over three different runs of TeMP and TSPASS for five representative examples (out of 39) based on the specifications of simple foraging robots and some associated properties. The robot specification results from a novel application of monodic first-order temporal logic in the verification of the behaviour of robot swarms. Further details can be found in (Behdenna et al.). The specification of the robot transition system was given as problem 0, and the remaining problems verify some properties of the transition system. Each of these problems contains at least seven eventualities. TeMP and TSPASS both terminate on the satisfiable problem 0, but TeMP cannot solve the unsatisfiable problem 2 within the given time limit. Additionally, on average TeMP derives more clauses and requires more execution time than TSPASS, except for problem 12. We attribute this observation to the subsumption-based loop search test in TSPASS and to the fact that inferences in TSPASS which have been computed once for a loop search instance do not have to be computed again for further loop search saturations. Further details and more examples can be found in (Ludwig and Hustadt, 2008b).

References

A. Behdenna, C. Dixon, and M. Fisher. Deductive verification of simple foraging robotic behaviours. To appear.

A. Degtyarev, M. Fisher, and B. Konev. Monodic temporal resolution. In *Proc. CADE-19*, volume 2741 of *Lecture Notes in Computer Science*, pages 397–411. Springer, 2003.

U. Hustadt, B. Konev, A. Riazanov, and A. Voronkov. TeMP: A temporal monodic prover. In *In Proc. IJCAR-04*, volume 3097 of *LNAI*, pages 326–330. Springer, 2004.

U. Hustadt, B. Konev, and R. A. Schmidt. Deciding monodic fragments by temporal resolution. In *Proc. CADE-20*, volume 3632 of *LNAI*, pages 204–218. Springer, 2005.

B. Konev, A. Degtyarev, C. Dixon, M. Fisher, and U. Hustadt. Mechanising first-order temporal resolution. *Information and Computation*, 199(1-2):55–86, 2005.

M. Ludwig and U. Hustadt. Fair monodic temporal reasoning. In *Proc. ARW'08*, 2008a.

M. Ludwig and U. Hustadt. Implementing a fair monodic temporal logic prover. 2008b. Submitted.

[1]http://www.csc.liv.ac.uk/˜michel/software/tspass/