TRP++ 2.0: A temporal resolution prover[⋆]

Ullrich Hustadt and Boris Konev[⋆⋆]

Department of Computer Science, University of Liverpool, UK
{U.Hustadt, 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 [14], databases [15], hardware verification [8] and agent-based systems [12].

In this paper we present **TRP**++ Version 2.0[1], a theorem prover for *propositional linear time logic* PLTL [4]. **TRP**++ is based on the resolution method for PLTL developed by Fisher [5] (see also [2, 3, 6]) which involves the translation of PLTL-formulae to separated normal form and the application of the inference rules of the *temporal resolution calculus*.

Arbitrary PLTL-formulae can be transformed into *separated normal form* (SNF) in a satisfiability equivalence preserving way using a renaming technique replacing non-atomic subformulae with new propositions and removing all occurrences of the $\mathcal{U}$ ('until') and $\mathcal{W}$ ('unless') operator [5, 6]. The result is a set of *SNF clauses* of the following form (where $\bigcirc$, $\Box$, and $\Diamond$ denote 'next', 'always', and 'eventually', respectively).

$$\begin{aligned}
&\textstyle\bigvee_{i=1}^{n} L_i && \text{(initial clause)}\\
&\Box(\textstyle\bigvee_{j=1}^{m} K_j \vee \bigvee_{i=1}^{n} \bigcirc L_i) && \text{(global clause)}\\
&\Box(\textstyle\bigvee_{j=1}^{m} K_j \vee \Diamond L) && \text{(eventuality clause)}
\end{aligned}$$

Here, K_j, L_i, and L (with $1 \leq j \leq m$, $0 \leq m$, and $1 \leq i \leq n$, $0 \leq n$) denote propositional literals. If $C = L_1 \vee \ldots \vee L_n$ we use $\bigcirc C$ to denote $\bigcirc L_1 \vee \ldots \vee \bigcirc L_n$.

Figure 1 shows all the inference rules of the *temporal resolution calculus*. To simplify the presentation, we have represented the conclusion of the eventuality resolution rule as a PLTL-formula (which would have to be transformed into a set of SNF clauses). In our implementation, we directly produce the corresponding SNF clauses without reverting to the transformation procedure. Note that finding a set of SNF clauses which satisfies the side conditions of the eventuality resolution rule is a non-trivial problem [2].

These inference rules provide a sound and complete calculus for deciding the satisfiability of a set of SNF clauses. Furthermore, under the assumption that

[⋆] Work supported by EPSRC grant GR/L87491.

[⋆⋆] On leave from Steklov Institute of Mathematics at St.Petersburg

[1] **TRP**++ can be downloaded from `http://www.csc.liv.ac.uk/~konev/trp++/`.

an inference step is performed only once for the same set of premises (or that we stop as soon as no new SNF clauses can be derived), any derivation from a set of SNF clauses will always terminate. The combination of the transformation procedure and the temporal resolution calculus is a EXPTIME decision procedure for PLTL (while the satisfiability problem of PLTL is PSPACE-complete). To our knowledge, there is no implementation of a complexity-optimal decision procedure for PLTL.

2 Details of implementation

TRP++ takes as input a set N of SNF clauses and tries to refute N using the temporal resolution calculus. The main procedure of our implementation of this calculus consists of a loop where at each iteration (i) the set of SNF clauses is saturated under application of the initial and step resolution rules, and (ii) then for every eventuality clause in the SNF clause set, an attempt is made to find a set of premises for an application of the eventuality resolution rule. If we find such a set, the set of SNF clauses representing the conclusion of the application is

Initial resolution rules:

$$\frac{C_1 \lor L \qquad \neg L \lor C_2}{C_1 \lor C_2} \qquad\qquad \frac{C_1 \lor L \qquad \Box(\neg L \lor D_2)}{C_1 \lor D_2}$$

where $C_1 \lor L$ and $\neg L \lor C_2$ are initial clauses, $\Box(\neg L \lor C_2)$ is a global clause and $\neg L \lor D_2$ is a propositional clause.

Step resolution rules:

$$\frac{\Box(C_1 \lor L) \qquad \Box(\neg L \lor C_2)}{\Box(C_1 \lor C_2)} \qquad\qquad \frac{\Box(C_1 \lor L) \qquad \Box(\bigcirc\neg L \lor D_2)}{\Box(\bigcirc C_1 \lor D_2)}$$

$$\frac{\Box(D_1 \lor \bigcirc L) \qquad \Box(\bigcirc\neg L \lor D_2)}{\Box(D_1 \lor D_2)}$$

where $\Box(C_1 \lor L)$ and $\Box(\neg L \lor C_2)$ are global clauses and $C_1 \lor L$ and $\neg L \lor C_2$ are propositional clauses, and $\Box(\bigcirc\neg L \lor D_2)$ and $\Box(D_1 \lor \bigcirc L)$ are global clauses. (The side conditions ensure that no clauses with nested occurrences of the $\bigcirc$-operator can be derived.)

Eventuality resolution rule:

$$\frac{\begin{array}{lcll} \Box(C^1_1 \lor \bigvee_{l=1}^{k^1_1} \bigcirc D^1_{1,l}) & & \Box(C^n_1 \lor \bigvee_{l=1}^{k^n_1} \bigcirc D^n_{1,l}) & \\ \vdots & & \vdots & \\ \Box(C^1_{m_1} \lor \bigvee_{l=1}^{k^1_{m_1}} \bigcirc D^1_{m_1,l}) & \cdots & \Box(C^n_{m_n} \lor \bigvee_{l=1}^{k^n_{m_n}} \bigcirc D^n_{m_n,l}) & \Box(C \lor \Diamond L) \end{array}}{\Box(C \lor (\neg(\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} C^i_j)\ \mathcal{W}\ L))}$$

where for all i, $1 \leq i \leq n$, $(\bigwedge_{j=1}^{m_i} \bigvee_{l=1}^{k^i_j} D^i_{j,l}) \rightarrow \neg L$ and $(\bigwedge_{j=1}^{m_i} \bigvee_{l=1}^{k^i_j} D^i_{j,l}) \rightarrow (\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} C^i_j)$ are provable.

Fig. 1. The temporal resolution calculus

added to the current set of SNF clauses. The main loop terminates if the empty clause is derived, indicating that the initial set N of SNF clauses is unsatisfiable, or if no new clauses have been derived during the last iteration of the main loop, which in the absence of the empty clause indicates that the initial set N of SNF clauses is satisfiable. Since the number of SNF clauses which can be formed over the finite set of propositional variables contained in the initial set of SNF clauses is itself finite, we can guarantee termination of the main procedure.

Inference rules. It is easy to check that under the natural arithmetic translation of initial and global clauses into first-order logic (an initial clause $(\neg)q_1 \vee \ldots \vee (\neg)q_n$ is represented by the first-order clause $(\neg)q_1(0) \vee \ldots \vee (\neg)q_n(0)$ where 0 represents the natural number 0; a global clause $(\neg)p_1 \vee \ldots \vee (\neg)p_m \vee \bigcirc(\neg)q_1 \vee \ldots \vee \bigcirc(\neg)q_n$ as $\forall x\, ((\neg)p_1(x) \vee \ldots \vee (\neg)p_m(x) \vee (\neg)q_1(s(x)) \vee \ldots \vee (\neg)q_n(s(x)))$ where s represents the successor function on the natural numbers), initial and step resolution exactly correspond to standard first-order *ordered* resolution with respect to an atom ordering $\prec$ where $p(x) \prec q(s(x)) \prec r(s(s(x)))$ for arbitrary predicate symbols p, q, and r. The eventuality resolution rule (whose application is the computationally most costly part of the method) can be implemented by a search algorithm which is again based on step resolution [3]. Hence, performance of the step resolution inference engine is critical for the system.

Using the arithmetic translation, any state-of-the-art first-order resolution system could perform step resolution and initial resolution. However, our formulae have a very restrictive nature, and **TRP**++ uses its own “near propositional” approach to deal with them.

Data representation. We represent SNF clauses as propositional clauses and supply each literal with an “attribute”—one of initial, global_now, and global_next with obvious meaning (eventuality clauses are kept and processed separately). In addition, we define a total ordering $<$ on attributed literals which satisfies the constraint that for every initial literal K, global_now literal L, and global_next literal M we have $K < L < M$.

The ordering is then used to restrict resolution inference steps to the maximal literals in a clause. This ensures, for example, that in a clause $C \vee L$ where L is a global_now literal but C contains some global_next literals, a resolution step on L is impossible. (Note that this behaviour is in accordance with the inference rules of the temporal resolution calculus.)

To simulate the effect of first-order unification on the arithmetical translation of SNF clauses, unification of literals in our “near propositional” representation has to take their attributes into account. This is implemented by means of *attribute transformers*—objects that can change the attribute of a literal. For further details see [9].

Saturation by step resolution. We implement an OTTER-like saturation method where the set of all SNF clauses is split into an *active* and a *passive* clause set, and all inferences are performed between an SNF clause, *selected* from the passive clause set, and the active clause set (for a detailed description see e.g. [13]). Generated SNF clauses are simplified by subsumption and forward subsumption resolution.

Indexing. In order to speed-up resolution, we group active clauses according to their maximal literal. For the current implementation, the (multi-literal) subsumption indexing algorithm does not distinguish literals with different attributes, thus providing us only with an imperfect filter whose result is re-checked afterwards. Global clauses are split into the *now* and *next* parts that are inserted into the index separately. Every propositional clause is represented as an *ordered* string of literals; no literal occurs more than once into the string. A set of strings (clauses) is kept in a *digital search trie* [11]. We formulate then both *forward subsumption* (to test if a given query clause is subsumed by an indexed clause) and *backward subsumption* (to find all indexed clauses subsumed by a given query clause) as string matching with wild-card characters similar to [1].

3 Current advances

TRP++ 1.0 was already presented in [9] where we demonstrated that the system is competitive to other known provers for PLTL. Also, it was noticed that the overall behaviour of the system relies on its ability to perform the step resolution inferences efficiently and we compared our step resolution engine with state of the art first-order resolution provers SPASS 2.0[2] and Vampire[3]. **TRP**++ was slower than these provers on the 'easier' problems (indicated by the median CPU time used), but competed quite well with Vampire 2.0 and SPASS 2.0 on the whole (indicated by the total CPU time used). One of the reasons of **TRP**++'s failure was the lack of a sophisticated *clause choice* function. As on a typical run of saturation by step resolution, the given set of clauses is satisfiable (since the set of clauses is originating from the search algorithm needed for the eventuality resolution rule), we believed that special clause selection and clause preference techniques would not influence the system performance. However, it turned out that a good clause choice function may help to establish satisfiability of a set of clauses faster. Apart from code improvements, **TRP**++ 2.0 differs from **TRP**++ 1.0 by a new clause choice function based on the length of clauses. Shorter clauses are selected first; clauses with the same length are sorted according to the maximal literal.

Figure 2 depicts the comparison of **TRP**++ 2.0 with **TRP**++ 1.0 and Vampire 5.0 (the winner of CASC-18 in the MIX and FOF divisions). Vampire 2.0 and SPASS cannot compete with **TRP**++ 2.0 and are omitted from Figure 2. For the system comparison we have selected series of randomly generated propositional formulae from the SATLIB benchmark problem library[4] (satisfiable uf20-91, uf50-218, and flat30-60; and unsatisfiable uuf50-218, each consisting of 100 problems) and two first-order problems, ring3 and ring5, the arithmetic translation of two problems describing the behaviour of an algorithm [7] that orients rings with 3 and 5 nodes, respectively. The tests have been performed on a PC

[2] `http://spass.mpi-sb.mpg.de/`

[3] `http://www.math.miami.edu/~tptp/CASC/`

[4] `http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html`

	TRP++ 1.0		Vampire 5.0		**TRP**++ 2.0	
	median	total	median	total	median	total
uf20-91	0.02	2.14	0.01	1.42	0.01	1.46
flat30-60	11.55	2605.34	1.80	360.90	5.38	1129.35
uf50-218	197.01	27909.14	1.65	211.70	4.15	556.69
uuf50-218	109.11	19153.86	1.30	143.70	1.46	199.77
ring3		0.93		4.89		0.78
ring5		7191.63		5631.53		324.23

Fig. 2. TRP++ 2.0 performance

with a 1.3GHz AMD Athlon processor, 512MB main memory running RedHat Linux 7.1.

As the data shows, **TRP**++ 2.0 possesses a fast step resolution engine, and the use of a new choice function provides a significant speed-up compared to **TRP**++ 1.0. One can expect that the use of more sophisticated atom orderings and literal selection functions will further reduce the number of inference steps that need to be performed, and may in some cases even eliminate a fundamental problem that PLTL decision procedures based on temporal resolution have [10].

We thank the anonymous referees for their helpful comments and suggestions.

References

1. J. Bentley and R. Sedgewick. Fast algorithms for sorting and searching strings. In *SODA: ACM-SIAM Symposium on Discrete Algorithms*, 1997.
2. C. Dixon. Search strategies for resolution in temporal logics. In *Proc. CADE-13*, volume 1104 of *LNAI*, pages 673–687. Springer, 1996.
3. C. Dixon. Using Otter for temporal resolution. In *Advances in Temporal Logic*, pages 149–166. Kluwer, 2000.
4. E. A. Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science*, chapter 16, pages 997–1072. Elsevier, 1990.
5. M. Fisher. A resolution method for temporal logic. In *Proc. IJCAI'91*, pages 99–104. Morgan Kaufman, 1991.
6. M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic*, 2(1):12–56, 2001.
7. J.-H. Hoepman. Uniform deterministic self-stabilizing ring-orientation on odd-length rings. In *WDAG '94*, volume 857 of *LNCS*, pages 265–279. Springer, 1994.
8. G. J. Holzmann. The model checker Spin. *IEEE Trans. on Software Engineering*, 23(5):279–295, 1997.
9. U. Hustadt and B. Konev. **TRP**++: A temporal resolution prover. In *Proc. WIL'02*. Available as `http://www.lsi.upc.es/~roberto/wilproceedings.html`.
10. U. Hustadt and R. A. Schmidt. Scientific benchmarking with temporal logic decision procedures. In *Proc. KR2002*, pages 533–544. Morgan Kaufmann, 2002.
11. D. E. Knuth. *The Art of Computer Programming. Volume III: Sorting and Searching*. Addison-Wesley, 1973.
12. A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–343, June 1998.
13. A. Riazanov and A. Voronkov. Limited resource strategy in resolution theorem proving. *Journal of Symbolic Computation*, to appear.
14. M. P. Shanahan. *Solving the Frame Problem.* MIT Press, 1997.
15. A. Tansel, editor. *Temporal Databases: theory, design, and implementation.* Benjamin/Cummings, 1993.