Implementing Invariant Search via Temporal Resolution ⋆

Alexei Lisitsa[1] and James Brotherston[2]

[1] Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, U.K.
A.Lisitsa@csc.liv.ac.uk
[2] Division of Informatics, University of Edinburgh, Edinburgh EH1 1HN, U.K.
jjb@dai.ed.ac.uk

Abstract. In this paper, we describe an implementation of a method for a searching for invariants in the theorem proving for first-order temporal logic. The method is based on clausal temporal resolution technique [FDP01] and implemented in the proof planning system λ*Clam* using the advantages of the higher-order logical programming language λ*Prolog*.

1 Introduction

First-order temporal logic over natural numbers is a very expressive formalism, allowing specification at natural level of abstraction, of various computational phenomena, algorithms and protocols. However, the reasoning in FOLTL is very complex. The logic itself, and even very small fragments, are non-recursively axiomatizable. That is why developing the proof methods, possibly based on heuristics, for fragments of FOLTL is the important task.

The key part of the reasoning in FOLTL is essentially inductive. For example, the following *inductive* proof rule can be used to prove 'always' conclusions.

$$\frac{\Gamma \to \Delta, I; \qquad I \to \bigcirc I; \qquad I \to \psi}{\Gamma \to \Delta, \Box\psi} \ (\to \Box)$$

However, the finding the *invariant* formula I during the proof search is a complex task and, in general, there is no complete procedure to achieve this.

In the paper [BDFL02] a method of finding the invariants in a class of first-order *monodic* [HWZ00] temporal formulae has been described and its completeness has been established. The method is based on a clausal temporal resolution technique [Fis91, FDP01].

Here we decribe an algorithm, based on this method, but extended to wider class of formulae, where it serves as, in general, *incomplete*, but useful procedure for the proving 'always'-formulae (i.e. of the form $\Box\psi$).

This algorithm has been implemented in the higher-order logic programming language λ*Prolog* [Lpro] with the general control provided by the proof planning system λ*Clam* [RSG98]. We here outline this implementation and discuss some of its features.

⋆ Work supported by EPSRC grants GR/M46624, GR/M46631 and GR/R45367.

2 Problem and Procedure

2.1 Temporal specifications and ground induction problem

We assume that the language $\mathcal{TL}$ of the first-order temporal logic is constructed in the standard way (see i.e. [HWZ00]) from a classical (non-temporal) first-order language $\mathcal{L}$ and a set of future-time temporal operators '$\Diamond$' (*sometime*), '$\Box$' (*always*), and '$\bigcirc$' (*in the next moment*).

Here, $\mathcal{L}$ does not contain equality or functional symbols. Formulae of $\mathcal{TL}$ are interpreted in *first-order temporal structures* with constant domain. We assume the *rigid* semantics for the variables and constants. We assume also the standard notions of *satisfiable* and *valid* formulae (evaluated in initial worlds of temporal structures).

A first-order temporal specification is a triple $< \mathcal{U}, \mathcal{S}, \mathcal{T} >$ where $\mathcal{U}$ and $\mathcal{S}$ are the *universal part* and the *initial part*, respectively, given by finite sets (conjunctions) of (nontemporal) first-order formulae, and $\mathcal{T}$ is the *temporal part* given by a finite set (conjunction) of *temporal step formulae*. A temporal step formula has the following form:

$$\forall \bar{x}\, (\varphi(\bar{x}) \rightarrow \bigcirc \chi(\bar{x}))$$

where φ and χ are first-order formulae with the (possibly empty) sequence of free variables $\bar{x}$. A temporal specification $< \mathcal{U}, \mathcal{S}, \mathcal{T} >$ is just a syntactical form of representation of the temporal formula $\Box \mathcal{U} \wedge \mathcal{S} \wedge \Box \mathcal{T}$.

If any temporal step formula of the specification *SP* has one of the following forms: $\forall x\ (P(x) \Rightarrow \bigcirc R(x))$, or $p \Rightarrow \bigcirc r$, where P and p are unary (i.e. one-place) predicate symbol and propositional symbol, respectively, $R(x)$ and r are boolean expressions composed from one-place predicates and propositional symbols, respectively, then the specification is called monodic [HWZ00, BDFL02].

Given a first-order temporal specification *SP* and a ground first-order formula ψ, *a ground induction problem* is that where the validity of the formula $SP \rightarrow \Box \psi$ must be established

2.2 Procedure

In the paper [BDFL02] a method for solving the ground induction problem was proposed and its completeness for the case of monodic specifications was established. We decsribe here a procedure, implementing this method, applicable to all the temporal specifications defined above.

The problem.

Given: A temporal specification $SP =< \mathcal{U}, \mathcal{S}, \mathcal{T} >$
with $\mathcal{T} = \{\forall \bar{x}(\varphi_1(\bar{x}) \rightarrow \chi_1(\bar{x})), \ldots, \varphi_k((\bar{x}) \rightarrow \chi_k(\bar{x})\})$ and a first-order formula ψ
Decide: Is $SP \rightarrow \Box \psi$ valid ?

The procedure.

Stage 1. Saturation of the universal part

$\mathcal{U}_1 := \mathcal{U}$; $\mathcal{U}_2 := \emptyset$;
while $(\mathcal{U}_1 \neq \mathcal{U}_2)$ do
{
$\mathcal{U}_2 := \mathcal{U}_1$;
for all subsets $\{i_1, \ldots i_l\} \subseteq \{1, 2, \ldots k\}$
do
{
if $\mathcal{U}_1 \cup \{\forall \bar{x}(\chi_{i_1}(\bar{x}) \vee \ldots \vee \chi_{i_1}(\bar{x}))\} \vdash \bot$
then $\mathcal{U}_1 := \mathcal{U}_1 \cup \{\neg \forall \bar{x}(\varphi_{i_1}(\bar{x}) \vee \ldots \vee \varphi_{i_1}(\bar{x}))\}$;
if $\mathcal{U}_1 \cup \{\exists \bar{x}(\chi_{i_1}(\bar{x}) \wedge \ldots \wedge \chi_{i_1}(\bar{x}))\} \vdash \bot$
then $\mathcal{U}_1 := \mathcal{U}_1 \cup \{\neg \exists \bar{x}(\varphi_{i_1}(\bar{x}) \wedge \ldots \wedge \varphi_{i_1}(\bar{x}))\}$;
}
}

Stage 2. Checking the start conditions

if $\mathcal{S} \wedge \mathcal{U}_1 \nvdash \psi$ then STOP with answer "No" ;
else Continue;

Stage 3. Generation of merged rules

$M := \emptyset$
for all subsets $\{i_1, \ldots i_l\} \subseteq \{1, 2, \ldots k\}$
do
{
if $(\mathcal{U}_1 \cup \{\exists \bar{x}((\chi_{i_1}(\bar{x}) \wedge \ldots \wedge \chi_{i_1}(\bar{x}))\} \vdash \psi)$
then $M := M \cup \{\exists \bar{x}(\varphi_{i_1}(\bar{x}) \wedge \ldots \wedge \varphi_{i_l}(\bar{x})) \Rightarrow \bigcirc(\exists \bar{x}(\chi_{i_1}(\bar{x}) \wedge \ldots \wedge \chi_{i_l}(\bar{x})))\}$;
if $(\mathcal{U}_1 \cup \{\forall \bar{x}((\chi_{i_1}(\bar{x}) \vee \ldots \vee \chi_{i_1}(\bar{x}))\} \vdash \psi)$
then $M := M \cup \{\forall \bar{x}(\varphi_{i_1}(\bar{x}) \vee \ldots \vee \varphi_{i_l}(\bar{x})) \Rightarrow \bigcirc(\forall \bar{x}(\chi_{i_1}(\bar{x}) \vee \ldots \vee \chi_{i_l}(\bar{x})))\}$;
}

Stage 4. Checking the loop conditions

for all subsets $\{Q_{i_1}\bar{x}\varphi_{i_1} \to \bigcirc Q_{i_1}\bar{x}\chi_{i_1}, \ldots, Q_{i_l}\bar{x}\varphi_{i_l} \to \bigcirc Q_{i_l}\bar{x}\chi_{i_l}(\bar{x})\} \subseteq M$ $(Q_{i_k} \in \{\forall, \exists\})$
do
{
if $\mathcal{S} \wedge \mathcal{U} \vdash \exists \bar{x}(\bigvee_{j=1}^{l} \varphi_{i_j}(\bar{x}))$ and $\forall j \in \{1, \ldots l\}\, \mathcal{U} \vdash (\forall \bar{x} \exists \bar{z}(\chi_{i_j}(\bar{x}) \to \bigvee_{k=1}^{l} \varphi_{i_k}(\bar{z})))$
then STOP with answer "Yes"
}
STOP with answer "I don't know".

The procedure is correct in the following sense: if it stops and produces the answer "Yes" then, indeed, $SP \to \Box\psi$ is valid (by straightforward check, see [BDFL02]). However, in general it is incomplete and may produce an "I don't know" answer for a valid formula. In the case of monodic specifications it is complete *modulo first-order derivability*, i.e. if one assumes the existense of oracle resolving the tasks of the kind

$\rho_1 \vdash \rho_2$ for the first-order formulae ρ_1 and ρ_2. To obtain the complete procedure in the case of real semi-decision (proof search) procedures for first-order logic, one needs to apply special strategies, ensuring the (uniform) progress in the proof search for *all* first-order formulae generated.

3 Implementation

The method described above has been implemented as a part of a prototype prover for temporal specifications in the λ*Clam* envinronment [RSG98]. λ*Clam* is a proof planning [Bun88] system, implemented in the Teyjus implementation of λProlog [Tey, Lpro], a higher-order typed logic programming language Ȧ proof plan is a representation of a proof at some level of abstraction (usually above the level of basic inference rules, but not necessarily so). In λ*Clam* a proof plan is generated from a goal by the application of planning operators called *proof methods*. Atomic methods are suitable for the implementation of basic proof rules, or automated proof procedures, while compound methods are used to build more complex proof strategies (or heuristics) from atomic methods.

The features of the system are as follows:

- Our system works with arithmetical translations of temporal formulae, i.e. with translation into two-sorted first-order formulae, with one sort being the natural numbers. This is done for better integration with a body of proof plans and methods already existing in λ*Clam* for arithmetic.
- For first-order (non-temporal) proving called within the prover an atomic method *proof_tableau* re-implementing the simple, but convenient LeanTap tableaux prover [BP95] in λProlog, is used. Higher-order unification and λ-terms as the data structure in λ*Prolog* allows us to re-implement LeanTap in more direct and declarative way, without using the "non-logical" *copy_term* predicate and with λ-terms presenting the quantifier's bindings. Efficiency of such λLeanTap proves to be comparable with the original LeanTap, running on Sicstus Prolog [Lis02].
- We have inplemented, so far, the simplest strategy concerning the depth of search within first-order tableau prover – it is invoked with the pre-defined constant depth bound. We consider the possibility to use the proof planning methodology to control the depth of search. Simple heuristics based on the number of quantifiers (or quantifier depth) may be used here.
- The kernel of the system is an atomic method called *mutual_induction*, implementing the above procedure. It uses the higher-order unification of λ*Prolog* to separate a given a set of formulae into the sets of step rules and the universal and start parts, and also to produce subformulae, which are used later to construct the goals for first-order theorem proving (conditions to check in the Stages *1–4*.) After that, *mutual_induction* invokes four sub-methods (λ-Prolog predicates) implementing the *Stages 1–4* of the procedure.
- The sets of formulae are presented as Prolog lists and a simple procedure for generating all its subsets is used. Simple optimization, based on the fact that any superset of an inconsistent set of formulae is itself inconsistent, is applied at Stage 1. It

remains to be explored whether the use of higher-order features of λ*Prolog* may optimize the search for *minimal* inconsistent subsets.

Initial experiments have indicated the viability of our approach. The system is able to prove all the examples of temporal specifications given in [BDFL02].

Our further work is to develop and implement optimizations for the proof search procedure together with imporved strategies/heuristics.

References

[BDFL02] J. Brotherston, A. Degtyarev, M. Fisher and A.Lisitsa, Searching for Invariants using Temporal Resolution, Proceedings of LPAR 2002, (to appear)

[BP95] B. Beckert and J. Posegga. leanT^{A}P: Lean, Tableau-based Deduction. *Journal of Automated Reasoning*, Vol. 15, No. 3, pages 339-358, 1995.

[Bun88] A. Bundy. The use of explicit plans to guide inductive proofs. In *Proc. of 9th International Conference on Automated Deduction* Springer-Verlag, 1988.

[DF01] A. Degtyarev and M. Fisher. Towards first-order temporal resolution. In *Proccedings of KI-2001*, volume 2174 of *LNAI*, 2001.

[FDP01] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computation Logic*, 2(1), January 2001.

[Fis91] M. Fisher. A resolution method for temporal logic. In *Proc. Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufman, 1991.

[HWZ00] I. Hodkinson, F. Wolter, and M. Zakharyaschev. Fragments of first-order temporal logics. *Annals of Pure and Applied logic*, 106:85–134, 2000.

[Lis02] A. Lisitsa, λ*LeanTap*: lean deduction in λ*Prolog* (forthcoming)

[Lpro] λ*Prolog* home page, http://www.cse.psu.edu/ dale/lProlog/

[RSG98] J. Richardson, A. Smaill, and I. Green. System description: proof planning in higher-order logic with lambdaclam. In *Proc. of CADE'15*, volume 1421 of LNAI, 1998.

[Tey] Teyjus web page, http://teyjus.cs.umn.edu/