Theoretical Computer Science 609 (2016) 639–657

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A complete axiom system for propositional projection temporal logic with cylinder computation model ☆

Nan Zhang, Zhenhua Duan *, Cong Tian

Institute of Computing Theory and Technology, and ISN Laboratory, Xidian University, Xi'an 710071, PR China

ARTICLE INFO

Article history:
Received 29 October 2014
Received in revised form 9 April 2015
Accepted 6 May 2015
Available online 13 May 2015

Keywords:
Axiom system
Multi-core
Model
Specification
Verification

ABSTRACT

To specify and verify multi-core parallel programs in a uniform framework, this paper proposes an axiom system for CCM–PPTL which extends that of PPTL by including transformation rules for sequence expressions and axioms as well as inference rules on the CCM construct. Further, the soundness and completeness of the extended axiom system are proved.

1. Introduction

With the rapid development of integrated circuits technology and the demand for higher performance, on-chip multi-core processors (CMP) have been brought into being. The reality of multi-core processor has made parallel programs pervasive. Creating a correct parallel program is not a straightforward process even for a considerable small program, because programmers are forced to consider that the program will always yield to a correct result no matter in what order the instructions are executed. To improve the reliability of parallel programs, formal verification is an important viable approach. Modeling multi-core parallel programs is a crucial step for formal verification of correctness and reliability of multi-core parallel programs.

Model checking [5,18] and theorem proving [3] are two key verification methods. With model checking, the system is often modeled as a finite transition system or automaton M, and the property is specified using a temporal logic formula P. Then a model checking procedure is employed to check whether or not $M \models P$ is valid. If so, the property is verified otherwise a counterexample can be found. The advantage of model checking is that the verification can be done automatically. However, model checking suffers from the state explosion problem [14]. Further, most of web applications are data-intensive which are not suitable to be verified by means of model checking since the treatment of the data usually leads to a huge, even infinite state space. Some successful model checking tools are SPIN [13], SMV [14] and so on. By contrast, theorem proving can handle many complex structures abstractly without suffering from state space explosion problem. However, theorem proving requires more human interventions and is often time-consuming. With theorem proving, both the system behavior and the desired property are specified as formulas, say S and P, in some appropriate logic. To prove that the

☆ This research is supported by the NSFC under Grant Nos. 61133001, 61202038, 61322202, 61420106004 and 91418201.
* Corresponding author.
E-mail addresses: nanzhang@xidian.edu.cn (N. Zhang), zhhduan@mail.xidian.edu.cn (Z. Duan), c.tian@mail.xidian.edu.cn (C. Tian).

http://dx.doi.org/10.1016/j.tcs.2015.05.007
0304-3975/

system satisfies the property amounts to proving that $\vdash S \rightarrow P$ is a theorem within the proof system of the logic. Some famous theorem provers are PVS [16], ACL2 [4], Coq [2], Isabella [17], HOL [11] and so on.

Verification of multi-core parallel programs raises a great challenge for theorem proving since it requires that the logic for modeling multi-core systems and specifying the expected properties has a powerful expressiveness. However, the widely used Propositional Linear Temporal Logic (PLTL) and Computational Tree Logic (CTL) are not powerful enough. In fact, they are not full regular. Although Quantified Linear time Temporal Logic (QLTL) [19], Extended Temporal Logic (ETL) [22] and linear μ-calculus [21] have full regular expressiveness, these logics are not intuitive to use in practice. Propositional Projection Temporal Logic (PPTL) [6] allows us to specify ω full regular properties [20]. Further, a decision procedure [9,8] and a complete proof system for PPTL [10] have been established. A model checker based on SPIN [7] and a theorem prover based on PVS have also been developed. Cylinder Computation Model (CCM) [23] is a concurrent semantic model which is defined based on PPTL and has been implemented in the interpreter of MSVL (Modeling, Simulation and Verification Language) [6], which is an executable subset of Projection Temporal Logic. CCM can be employed to model multi-core parallel programs since the sequence expressions in it have the nature of regular expressions. With CCM, the autonomy and parallelism of the processes occupying different cores on one chip can be described neatly and concisely. In [10], we have proposed an axiom system for PPTL, and proved its soundness and completeness. To specify and verify multi-core parallel programs in a uniform framework, this paper proposes an axiom system for CCM–PPTL which extends that of PPTL by including transformation rules for sequence expressions and axioms as well as inference rules on the CCM construct. Furthermore, the soundness and completeness of the extended axiom system are also proved.

The paper is organized as follows: in the next section, the underlying logic PPTL and CCM are reviewed. Based on PPTL, CCM–PPTL is introduced in Section 3, including the syntax, semantics and some logical laws regarding the CCM construct. In Section 4, an axiom system for CCM–PPTL is formalized. In particular, the axioms, inference rules and the proofs of the soundness and completeness are given in detail. In Section 5, an example is given to illustrate how to use CCM and its proof system to model and verify practical algorithms. Finally, conclusions are drawn in Section 6.

2. Preliminaries

Our underlying logic is Propositional Projection Temporal Logic (PPTL), which is an interval-based temporal logic with ω full regular expressiveness. For more details on PPTL, refer to [6,10].

2.1. Propositional projection temporal logic

2.1.1. Syntax

The formula P of PPTL can be defined by the following grammar,

$$P ::= p \mid \bigcirc P \mid \neg P \mid P_1 \vee P_2 \mid (P_1, \ldots, P_m)\ prj\ P$$
$$\mid (P_1, \ldots, (P_i, \ldots, P_l)^{\oplus}, \ldots, P_m)\ prj\ P$$

where $p \in Prop$, $P_1, \ldots, P_i, \ldots, P_l, \ldots, P_m$ $(1 \leq i \leq l \leq m,\ i, l, m \in N_0)$ and P are all well-formed PPTL formulas, and $\bigcirc$, prj and $prj\oplus$ (projection-plus) are primitive temporal operators. A formula is called a state formula if it contains no temporal operators, otherwise it is called a temporal formula.

2.1.2. Semantics

We define a state s over $Prop$ to be a mapping from $Prop$ to B.

$$s : Prop \rightarrow B$$

We use $s[p]$ to denote the valuation of p at state s.

An interval σ is a non-empty sequence of states, which can be finite or infinite. The length, $|\sigma|$, of σ is ω if σ is infinite, and the number of states minus 1 if σ is finite. We consider the set N_0 of non-negative integers and ω, $N_\omega = N_0 \cup \{\omega\}$ and extend the comparison operators, $=$, $<$, $\leq$, to N_ω by considering $\omega = \omega$, and for all $i \in N_0$, $i < \omega$. Furthermore, we define $\preceq$ as $\leq -\{(\omega, \omega)\}$. To simplify definitions, we will denote σ as $\langle s_0, \ldots, s_{|\sigma|}\rangle$, where $s_{|\sigma|}$ is undefined if σ is infinite. With such a notation, $\sigma_{(i\ldots j)} (0 \leq i \preceq j \leq |\sigma|)$ denotes the sub-interval $\langle s_i, \ldots, s_j\rangle$ and σ^i $(0 \leq i \leq |\sigma|)$ denotes the prefix interval $\langle s_0, \ldots, s_i\rangle$. The concatenation of a finite σ with another interval (or empty string) σ' is denoted by $\sigma \bullet \sigma'$ (not sharing any states). Let $\sigma = \langle s_0, s_1, \ldots, s_{|\sigma|}\rangle$ be an interval and $r_1, \ldots, r_h$ be integers $(h \geq 1)$ such that $0 \leq r_1 \leq r_2 \leq \ldots \leq r_h \preceq |\sigma|$. The projection of σ onto $r_1, \ldots, r_h$ is the interval (called projected interval)

$$\sigma \downarrow (r_1, \ldots, r_h) = \langle s_{t_1}, s_{t_2}, \ldots, s_{t_l}\rangle$$

where $t_1, \ldots, t_l$ are obtained from $r_1, \ldots, r_h$ by deleting all duplicates. That is, $t_1, \ldots, t_l$ is the longest strictly increasing subsequence of $r_1, \ldots, r_h$.

An interpretation is a triple $\mathcal{I} = (\sigma, k, j)$, where σ is an interval, k an integer, and j an integer or ω such that $0 \leq k \preceq j \leq |\sigma|$. We use the notation $(\sigma, k, j) \models P$ to indicate that some formula P is interpreted and satisfied over the subinterval $\langle s_k, \ldots, s_j\rangle$ of σ with the current state being s_k. The satisfaction relation $(\models)$ is inductively defined in Table 1.