Theoretical Computer Science 609 (2016) 544–560

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A canonical form based decision procedure and model checking approach for propositional projection temporal logic ☆

Zhenhua Duan, Cong Tian *, Nan Zhang

ICTT and ISN Laboratory, Xidian University, Xi'an, 710071, PR China

ARTICLE INFO

Article history:
Received 31 October 2014
Received in revised form 27 June 2015
Accepted 24 August 2015
Available online 10 September 2015

Keywords:
Decision procedure
Propositional projection temporal logic
Büchi automata
Model checking

ABSTRACT

This paper proposes a Canonical Form (CF) for chop formulas of Propositional Projection Temporal Logic (PPTL). Based on CF, an improved algorithm for constructing Labeled Normal Form Graph (LNFG) of a PPTL formula is presented. This improvement leads to a better decision procedure for PPTL with infinite models. In addition, a transformation from LNFGs to Generalized Büchi Automata (GBA) and then Büchi Automata (BA) is formalized. Thus, a SPIN based model checking approach is generalized for PPTL. To illustrate how these algorithms work, several examples are given.

1. Introduction

Model checking is a useful approach for verifying programs [1,2]. The usually used temporal logics for defining properties are Linear Temporal Logic (LTL) and Computing Tree Logic (CTL) [3,4]. However, as is well known, the expressive power of both LTL and CTL is limited such that some necessary properties of programs cannot be verified. In fact, the expressive power of these two logics is less than full regular expressions. Therefore, a stronger temporal logic with full regular expressive power is desired in program verification.

Propositional Projection Temporal Logic (PPTL) [7,10,5], whose expressive power is equal to full regular expressions [11], subsumes Propositional Interval Temporal Logic (PITL) [9]. It is a useful logic in the specification and verification of concurrent systems [17,18]. In the past years, a decision procedure based on Labeled Normal Form Graph (LNFG) was given for PPTL formulas [25] and is improved in [6,13]. Specifically, finiteness of LNFGs is proved in the former and a practical constructing algorithm of LNFGs is formalized in the latter. The decision algorithm is actually to find out a model in the LNFG of a formula. Given a PPTL formula P, to check whether or not it is satisfiable, we first try to construct its LNFG, then look for a model in the LNFG of the formula. If a model is found out for P in its LNFG, P is satisfiable, otherwise, it is unsatisfiable. Based on the decision procedure, a model checking approach based on SPIN [15] for PPTL is proposed in [8]. Actually, to verify a property of a system with model checking, the property can be specified by a PPTL formula Q, while the system is modeled with a transition system such as Kripke structure or an automaton M. In the next step, we first transform $\neg Q$ to an LNFG G, then we check whether or not $M \cap G$ is empty.

☆ This research is supported by National Natural Science Foundation of China under Grant Nos. 61133001, 61322202, 61420106004, 61272117, and 91418201.

* Corresponding author.
E-mail address: ctian@mail.xidian.edu.cn (C. Tian).

http://dx.doi.org/10.1016/j.tcs.2015.08.039
0304-3975/

In this paper, we further improve the existing constructing algorithm of LNFGs for a PPTL formula in the following two aspects (see Section 2 for details of notations such as $len(k)$, $fin(l_k)$ etc.): (1) for a simple chop formula, $P; Q$, if $P \equiv P' \wedge len(k)$, $k \in N_0$, we do not need to add a *fin* label to the formula since P is a terminal formula; (2) for a canonical form of chop formulas, $P \equiv (P_1; P_2) \wedge Z$, where P_1 and P_2 are also canonical chop formulas, and Z is not a chop formula, unlike in [6], we need only to add a *fin* label into the formula $P_1 \wedge fin(l_k); P_2$ without further concerning chop constructs inside the formulas P_1. In this way, less *fin* labels are required during the construction process of an LNFG. Thus, the constructing algorithm of LNFGs for PPTL formulas can be simplified, leading to a simpler decision procedure. In addition, an LNFG can be transformed to a Generalized Büchi Automaton (GBA) [14], and a GBA can further be transformed to a Büchi Automaton (BA) [14]. Therefore, when a system is modeled by an automaton M and a property of the system is specified by a PPTL formula P, the LNFG G of the formula $\neg P$ is constructed, then G is transformed into a GBA A and further transformed into a BA A'. To check whether or not $M \models P$ amounts to checking if $M \cap A' = \emptyset$. This can be done by utilizing model checker SPIN [15]. Thus, we can obtain a SPIN based model check approach for PPTL.

The paper is organized as follows. The next section briefly introduces PPTL, including its syntax and semantics. Section 3 discusses chop formulas. In particular, a canonical form of chop formulas is presented. In Section 4, an improved constructing algorithm of LNFGs for PPTL formulas is formalized. Some examples are given to show how the algorithm works. Section 5 focuses on SPIN based model checking approach for PPTL formulas. In particular, the transformation algorithms from an LNFG to a GBA and then to a BA are presented in details. Finally, conclusion and future research are drawn in Section 6.

2. Propositional project temporal logic

Propositional Projection Temporal Logic (PPTL) [10,7] is an extension of Propositional ITL (PITL) [9] with infinite models and a new projection construct. Let *Prop* be a countable set of atomic propositions and $B = \{true, false\}$ the boolean domain. Usually, we use lowercase letters like p, q and r, possibly with subscripts, to denote atomic propositions and capital letters like P, Q and R, possibly with subscripts, to represent general PPTL formulas. Formulas of PPTL are defined by the following grammar:

$$P ::= p \mid \neg P \mid P_1 \wedge P_2 \mid \bigcirc P \mid (P_1, \ldots, P_m)\ \mathrm{prj}\ P \mid P^+$$

where $p \in Prop$, $\bigcirc$ (next), $+$ (chop-plus) and prj (projection) are temporal operators. $\neg$ and $\wedge$ are similar as that in the classical propositional logic.

We define a *state* s over *Prop* to be a mapping from *Prop* to B, $s : Prop \to B$. We write $s[p]$ to denote the valuation of p at state s. An *interval* $\sigma =< s_0, s_1, \cdots >$ is a non-empty sequence of states, which can be finite or infinite. The length of σ, $|\sigma|$, is the number of states in σ minus one if σ is finite; otherwise it is ω. Let N_0 denote the set of non-negative integers. To have a uniform notation for both finite and infinite intervals, we will use *extended integers* as indices, that is $N_\omega = N_0 \cup \{\omega\}$, and extend the comparison operators, $=, <, \leq$, to N_ω by considering $\omega = \omega$, and for all $i \in N_0, i < \omega$. Moreover, we write $\preceq$ as $\leq -\{(\omega, \omega)\}$. To simplify definitions, we will denote σ by $< s_0, \cdots, s_{|\sigma|} >$, where $s_{|\sigma|}$ is undefined if σ is infinite. With such a notation, $\sigma_{(i..j)} (0 \leq i \preceq j \leq |\sigma|)$ denotes the sub-interval $< s_i, \cdots, s_j >$.

To formalize the semantics of the projection construct, we need an auxiliary operator $\downarrow$. Let $\sigma =< s_0, s_1, \ldots >$ be an interval and $r_1, \ldots, r_h$ be integers $(h \geq 1)$ such that $0 \leq r_1 \leq \ldots \leq r_h \preceq |\sigma|$. The projection of σ onto $r_1, \ldots, r_h$ is the *projected interval*, $\sigma \downarrow (r_1, \ldots, r_h) \stackrel{\text{def}}{=} < s_{t_1}, s_{t_2}, \ldots, s_{t_l} >$, where $t_1, \ldots, t_l$ are attained from $r_1, \ldots, r_h$ by deleting all duplicates. In other words, $t_1, \ldots, t_l$ is the longest strictly increasing subsequence of $r_1, \ldots, r_h$. For instance, $< s_0, s_1, s_2, s_3 > \downarrow (0, 2, 2, 2, 3) =< s_0, s_2, s_3 >$. The concatenation $(\cdot)$ of a finite interval $\sigma =< s_0, s_1, \ldots, s_{|\sigma|} >$ with another interval $\sigma' =< s'_0, s'_1, \ldots, s'_{|\sigma'|} >$ is represented by $\sigma \cdot \sigma' =< s_0, s_1, \ldots, s_{|\sigma|}, s'_0, s'_1, \ldots, s'_{|\sigma'|} >$ (not sharing any states).

An *interpretation* is a tuple $\mathcal{I} = (\sigma, k, j)$, where $\sigma =< s_0, s_1, \ldots >$ is an interval, k is a non-negative integer, and j is an integer or ω such that $0 \leq k \preceq j \leq |\sigma|$. We write (σ, k, j) to mean that a formula is interpreted over a subinterval $\sigma_{(k..j)}$ with the current state being s_k. We utilize I^k_{prop} to stand for the state interpretation at state s_k. The satisfaction relation $\models$ for formulas is given as follows:

$\mathcal{I} \models p$	iff $s_k[p] = I^k_{prop}[p] = \text{true}$
$\mathcal{I} \models \neg P$	iff $\mathcal{I} \not\models P$
$\mathcal{I} \models P_1 \wedge P_2$	iff $\mathcal{I} \models P_1$ and $\mathcal{I} \models P_2$
$\mathcal{I} \models \bigcirc P$	iff $k < j$ and $(\sigma, k+1, j) \models P$
$\mathcal{I} \models (P_1, \ldots, P_m)\ \mathrm{prj}\ P$	iff there exist integers $r_0, \ldots, r_m$, and $k = r_0 \leq \ldots \leq r_{m-1} \preceq r_m \leq j$ such that $(\sigma, r_{l-1}, r_l) \models P_l$ for all $1 \leq l \leq m$ and $(\sigma', 0, \lvert\sigma'\rvert) \models P$ for σ' given by: (1) $r_m < j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_m) \cdot \sigma_{(r_m+1, \ldots, j)}$ (2) $r_m = j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_h)$ for some $0 \leq h \leq m$
$\mathcal{I} \models P^+$	iff there are finitely many integers $r_0, \ldots, r_n$ and $k = r_0 \leq r_1 \leq \ldots \leq r_{n-1} \preceq r_n = j$ $(n \geq 1)$ such that $(\sigma, r_{l-1}, r_l) \models P$ for all $1 \leq l \leq n$; or $j = \omega$ and there are infinitely many integers $k = r_0 \leq r_1 \leq r_2 \leq \ldots$ such that $\lim_{i \to \infty} r_i = \omega$ and $(\sigma, r_{l-1}, r_l) \models P$ for all $l \geq 1$.