

Bounded Model Checking of Traffic Light Control System

Bin Yu^{1,2} Zhenhua Duan*, Cong Tian³

*Institute of Computing Theory and Technology, and ISN Lab
Xidian University
Xi'an, P.R. China*

Abstract

Traffic Light Control System (TLCS) is widely used in our daily life. It is of great importance to ensure the correctness of TLCS. In this paper, bounded model checking (BMC) is chosen to verify a simple but practical TLCS. To this end, Propositional Projection Temporal Logic (PPTL) used as the property specification language and the process of BMC for PPTL are briefly introduced. Then, a TLCS is described and its corresponding Kripke structure is given. Finally, two related properties specified by PPTL formulas are verified for the system using the BMC approach. The verification result using our bounded model checker, BMC4PPTL, shows that the behavior of TLCS is consistent with the specification.

Keywords: Bounded Model Checking, PPTL, TLCS, Verification

1 Introduction

Techniques for automatic formal verification of finite state transition systems have been studied in recent years. The most widely used approach is called Model Checking [4,6]. As a trusted, strong and automatic verification technique, model checking has been widely used in many fields such as verification of hardware, software and communication protocols. With model checking, the system to be verified is modeled as a finite state machine and the specification is formalized in terms of temporal logic formulas. In practice, linear-time temporal logic (LTL) [16] and branching-time temporal logic (CTL) [4] are popular.

¹ This research is supported by the National Program on the Key Basic Research Project of China (973 Program) under Grant No. 2010CB328102, and the National Natural Science Foundation of China under Grant Nos. 61133001, 61272117, 61272118, 61202038, 91218301, 61322202 and 61373043. * Corresponding author.

² Email: yubin9011@126.com

³ Email: zhhdian,ctian@mail.xidian.edu.cn

SPIN [14] based on LTL and SMV [15] depended on CTL are two well-known model checkers. However, as known, automata-based model checking algorithms can easily lead to state space explosion when the number of states in the system is large. To fight this problem, several approaches, such as Symbolic Model Checking (SMC) [2], Abstract Model Checking (AMC) [5], and Compositional Model Checking [7], have been proposed with success. The combination of SMC with BDDs [15,8] pushed the barrier to systems with 10^{20} states and more [2]. But the bottleneck of SMC is the amount of memory that is required for storing and manipulating BDDs. The boolean functions required to represent the set of states can grow exponentially. Bounded model checking (BMC) is an important progress in formalized verification after symbolic model checking [1]. The basic idea of BMC is to search for a counterexample in executions whose length is bounded by some integer k . If the property is not satisfied, an error is found. Otherwise, we cannot tell whether the system satisfies the property or not. In this case, we can consider increasing k , and perform the process of BMC again. The BMC problem can be efficiently reduced to a propositional satisfiability problem, and can therefore be solved by SAT solvers rather than BDDs. Modern SAT solvers can handle propositional satisfiability problems with hundreds of thousands of variables.

With model checking and bounded model checking, the mostly used temporal logics are LTL, CTL and their variations. However, the expressiveness of LTL and CTL is not powerful enough, actually, not full regular. There are at least two types of properties in practice which cannot be specified by LTL and CTL: (1) some time duration related properties such as a property P holds after 100^{th} time unit and before 200^{th} time unit; (2) some periodically repeated properties P . Propositional Projection Temporal Logic (PPTL) [9,11] is a useful formalism for specification and verification of concurrent systems. The expressiveness of PPTL is full regular [17] which allows us to verify full regular properties and time duration related properties of systems in a convenient way.

To combine the advantages of BMC and PPTL, the bounded semantics of PPTL formulas and the process of BMC for PPTL have been presented in [12]. The bounded model checker for PPTL named BMC4PPTL has been developed so that automatical verification can be conducted. With BMC4PPTL, we describe the model by the input language used in NuSMV [3] and specify the property by a PPTL formula. When a PPTL formula R is a chop construct in the form of $R \equiv Q_1; Q_2$, R is false if Q_1 only has infinite models and we don't consider this case in this paper.

In our daily life, TLCS plays an important role to make the traffic be safe and efficient. So it is of great importance to ensure the correctness of TLCS. In this article, first we describe a TLCS by a Kripke structure M according to the requirement specification. Then one safety property and one periodically repeated property to be verified are specified by PPTL formulas. After that, the BMC approach is employed to find a counterexample. The verification is done automatically by BMC4PPTL and the results show that the system is consistent with the specification.

This paper is organized as follows. The next section presents the preliminaries, including the Kripke structure used for the description of a model and the property

specification language PPTL. In addition, the process of BMC for PPTL is introduced. In section 3, a simple but practical TLCS is described and its corresponding Kripke structure is given. In section 4, two related properties are verified using the BMC approach. The verification results using BMC4PPTL for checking these properties are shown. Finally, the conclusion is drawn in section 5.

2 Preliminaries

2.1 The Kripke Structure

A Kripke structure M is a quadruple $M = (S, I, T, L)$ where S is the set of states which are defined by valuations to a set of boolean variables (atomic propositions) A , $I \subseteq S$ is the set of initial states, $T \subseteq S \times S$ is the transition relation and $L : S \rightarrow 2^{(A)}$ is the labeling function. Labeling is a way to attach observations to the system: for a state $s \in S$ the set $L(s)$ contains exactly those atomic propositions that hold in s . The set of initial states I and the transition relation T are given as functions in terms of A .

2.2 Propositional Projection Temporal Logic

Let $Prop$ be a countable set of atomic propositions. The formula P of PPTL is given by the following grammar:

$$P ::= p \mid \bigcirc P \mid \neg P \mid P_1 \vee P_2 \mid (P_1, \dots, P_m) \text{ prj } P$$

where $p \in Prop$, P_1, \dots, P_m and P are all well-formed PPTL formulas. \bigcirc (*next*) and *prj* (*projection*) are basic temporal operators. The semantics have been defined in [10].

Normal Form (NF) [13] is a useful notation in our method. We assume Q_p is the set of atomic propositions appearing in the PPTL formula Q . Then, the NF can be defined as follows:

$$NF(Q) \equiv \bigvee_{j=1}^{n_0} (Q_{ej} \wedge \text{empty}) \vee \bigvee_{i=1}^n (Q_{ci} \wedge \bigcirc Q'_i)$$

where Q_{ej} and Q_{ci} are conjunctions of atomic propositions (or their negations) in Q_p , Q'_i is an arbitrary PPTL formula.

Any PPTL formula Q can be rewritten into its normal form [13].

Example 2.1 Given a PPTL formula f , $f \equiv \neg(\text{true}; \neg(\bigcirc q)) \wedge \neg(p; q)$, we can get

$$NF(f) \equiv p \wedge \neg q \wedge \bigcirc(q \wedge \square \bigcirc q \wedge \square \neg q) \vee \neg p \wedge \bigcirc(q \wedge \square \bigcirc q)$$

2.3 The Process of Bounded Model Checking for PPTL

The bounded semantics of PPTL formulas and the process of BMC for PPTL have been defined in [12]. In this subsection, we briefly introduce the process of BMC for PPTL which is mainly used in this paper.

Given a finite state transition system M (a Kripke structure), the property of the system in terms of a PPTL formula f , and a bound k , the procedure of BMC can be described as a process for constructing a proposition formula $[M, f]_k$. Let

(s_0, \dots, s_k) be a finite sequence of states on a path π . The definition of formula $[M, f]_k$ consists of two components: M_k , the first component, is a propositional formula that constrains (s_0, \dots, s_k) to be a valid path starting from an initial state; X_k , the second component, a propositional formula that constrains π to satisfy f with bound k . To define the second component, we first give the definition of *loop condition*, which is a propositional formula that is evaluated to true only if the path π contains a loop.

For a Kripke structure M , and $k \geq 0$,

$$M_k \stackrel{\text{def}}{=} I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$$

Loop condition:

$$L_{(k,l)} \stackrel{\text{def}}{=} T(s_k, s_l) \quad (0 \leq l \leq k)$$

$$L_k \stackrel{\text{def}}{=} \bigvee_{l=0}^k L_{(k,l)}$$

General translation (BMC to SAT):

$$X_k \stackrel{\text{def}}{=} (\neg L_k \wedge f_{(0,k)}) \vee \bigvee_{l=0}^k (L_{(k,l)} \wedge f_{(0,k,l)})$$

$$[M, f]_k \stackrel{\text{def}}{=} M_k \wedge X_k$$

In the definition of X_k , the translation rule of $f_{(i,k,l)}(i, l \leq k)$ for a PPTL formula f over an infinite path π with a (k, l) – loop and $f_{(i,k)}(i \leq k)$ for a PPTL formula f over a finite path π with no loop have been defined in [12]. Putting everything together we can get the following boolean formula:

$$[M, f]_k \stackrel{\text{def}}{=} I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge [(\neg L_k \wedge f_{(0,k)}) \vee \bigvee_{l=0}^k (L_{(k,l)} \wedge f_{(0,k,l)})]$$

As we can see, the right side of the definition can be divided into two parts: the first part $I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge (\neg L_k \wedge f_{(0,k)})$ indicates a path with no loop and the translation without loop is used; the second part $I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge (\bigvee_{l=0}^k (L_{(k,l)} \wedge f_{(0,k,l)}))$ represents that a path with a (k, l) – loop and all possible starting points l of a loop and the translation for a (k, l) – loop is conjoined with the corresponding loop condition $L_{(k,l)}$.

3 Modeling of TLCS

Traffic light control system (TLCS) is common in our daily life. When cars and pedestrians cross an intersection, traffic lights make them pass or stop. A poorly designed TLCS will cause traffic chaos. The normal function of traffic lights requires sophisticated control and coordination to ensure that cars move as smoothly and safely as possible and that pedestrians are protected when they cross the roads. A variety of different control systems are used to accomplish this, ranging from simple clockwork mechanisms to sophisticated computerized control and coordination systems that self-adjust to minimize delay to people using the road. An intuitive model of a crossing is shown in figure 1.

As we all know, the duration of the green lights for the main road should be longer than that of the red lights in the rush hours because there are more cars

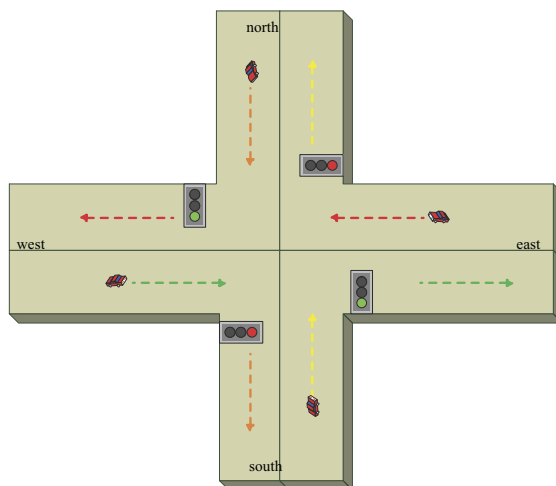


Fig. 1. An intuitive model of a crossing

and pedestrians in the main road while the duration is equal at other time. In fact, an east-west traffic light and a west-east traffic light change simultaneously, so we use one light to represent both for clarity in our model. Similarly, we use the south-north traffic light to represent the north-south and south-north traffic lights. Now a simple rule is made for the light control system. We assume there are two modes in the system. Mode 1 represents the rush hours and mode 0 represents the other time. When the current time is between 7 o'clock and 9 o'clock or between 17 o'clock and 19 o'clock, the system is in mode 1. The system can translate from mode 0 to mode 1 and from mode 1 to mode 0 according to time o'clock.

The procedure of the TLCS system can be stated as follows:

- (1) The system starts at 0 o'clock and the next step is (2);
- (2) The mode of the system is set as 0. The green light of the east-west direction and the red light of the south-north direction are on. This state lasts 25 seconds and the next step is (3);
- (3) The yellow light of the east-west direction flashes and the red light of the south-north direction is on. This state lasts 5 seconds and the next step is (4);
- (4) The red light of the east-west direction and the green light of the south-north direction are on. This state lasts 25 seconds and the next step is (5);
- (5) The red light of the east-west direction is on and the yellow light of the south-north direction flashes. This state lasts 5 seconds. According to the current time, the mode is set for the next state. If the current time is between 7 o'clock and 9 o'clock or between 17 o'clock and 19 o'clock, the next step is (6), otherwise the next step is (2);
- (6) The mode of the system is set as 1. The green light of the east-west direction and the red light of the south-north direction are on. This state lasts 30 seconds and the next step is (7);
- (7) The yellow light of the east-west direction flashes and the red light of the

south-north direction is on. This state lasts 5 seconds and the next step is (8);

(8) The red light of the east-west direction and the green light of the south-north direction are on. This state lasts 20 seconds and the next step is (9);

(9) The red light of the east-west direction is on and the yellow light of the south-north direction flashes. This state lasts 5 seconds. According to the current time, the mode is set for the next state. If the current time is between 7 o'clock and 9 o'clock or between 17 o'clock and 19 o'clock, the next step is (6), otherwise the next step is (2).

The implementation of the system is shown as a Kripke structure in Fig.2. There are eight states in the Kripke structure. Each state s is represented by four state variables $s[3]$, $s[2]$, $s[1]$ and $s[0]$. $s[3] = f(t)$ is used for representing the mode at the current time t of the state s . The current time is measured in seconds. The function f returns 1 when the traffic is in rush hours otherwise returns 0. For $0 \leq t < 24 \times 3600$, the definition of f is shown as follows:

$$f(t) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } 7 \times 3600 \leq t < 9 \times 3600 \text{ or } 17 \times 3600 \leq t < 19 \times 3600 \\ 0 & \text{otherwise} \end{cases}$$

Because the red light of either the south-north direction or the east-west direction is on at every state, we can use $s[2]$ to represent the red light of the south-north direction is on and $\neg s[2]$ to represent the red light of the east-west direction is on. As the yellow light of the south-north or east-west direction and the green light of the south-north or east-west direction, only one kind of lights works at one state for a time. So we can use $s[1] \wedge s[0]$ to represent the green light of the south-north direction is on, $s[1] \wedge \neg s[0]$ to represent the green light of the east-west direction is on, $\neg s[1] \wedge s[0]$ to represent the yellow light of the south-north direction flashes and $\neg s[1] \wedge \neg s[0]$ to represent the yellow light of the east-west direction flashes.

The initial starting time of the system is at 0 o'clock and it takes 60 seconds to execute a loop each time. From the definition of $f(t)$, we know that $t = 7 \times 3600$ is the switching time when the system translates from mode 0 to mode 1. When the current time is between 0 o'clock and 7 o'clock, the system runs on the left circle. The mode of each state on the left circle is 0 according to the evaluation of $f(t)$. Because 7×3600 is divisible by 60, the system may translate from mode 0 to mode 1 only when the last state in the loop has been executed. When t reaches 7×3600 , the function $f(t)$ returns 1 and the system goes to the right circle representing the mode for the rush hours. During the execution time, the mode changes and the system runs with the corresponding circle. From the description above, we can see that the procedure represented by the Kripke structure is consistent with the steps of the TLCS system.

When system reaches at s_4 , the loop of the left or right side must have been executed n times in total where n is an integer. The mode of s_4 is 1 and the current time t is between 7×3600 and 9×3600 or between 17×3600 and 19×3600 . We can use $60 \times n$ to represent t because it takes 60 seconds to execute a loop each time. Now we assume t is between 7×3600 and 9×3600 . Then we can get the following inequality:

$$7 \times 3600 \leq 60 \times n < 9 \times 3600$$

$$\Rightarrow 420 \leq n < 540$$

$$\Rightarrow 420 \leq n \leq 539$$

$$\Rightarrow 420 + 1/2 \leq n + 1/2 \leq 539 + 1/2$$

$$\Rightarrow 420 < n + 1/2 < 540$$

$$\Rightarrow 7 \times 3600 < 60 \times n + 30 < 9 \times 3600$$

In the same way, we can get the following inequalities:

$$7 \times 3600 < 60 \times n + 35 < 9 \times 3600$$

$$7 \times 3600 < 60 \times n + 55 < 9 \times 3600$$

$$7 \times 3600 < 60 \times n + 60 \leq 9 \times 3600$$

From the range of $60 \times n + 30$, $60 \times n + 35$ and $60 \times n + 55$, we can know that the mode of s_5, s_6, s_7 is 1 which is the same as that of their last state. When s_7 has been executed, the mode of its next state can be 0 or 1 because $f(t)$ returns 0 or 1 for $7 \times 3600 < t \leq 9 \times 3600$. So in the Kripke structure, there are two output arcs from the state s_7 . When t is between 17×3600 and 19×3600 , we can obtain the same result.

In a similar way, we can also prove that there is one arc from s_0, s_1, s_2 and there are two output arcs from the state s_3 on the left side.

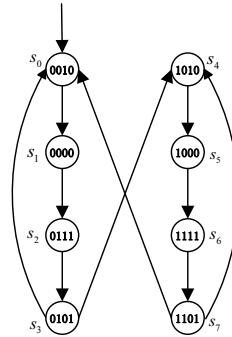


Fig. 2. The Kripke structure of TLCS M

The set of initial states is represented as follows:

$$I(s) := \neg s[3] \wedge \neg s[2] \wedge s[1] \wedge \neg s[0]$$

The transition relation is represented as follows:

$$\begin{aligned} T(s, s') := & (\neg s[3] \wedge \neg s[2] \wedge s[1] \wedge \neg s[0] \wedge \neg s'[3] \wedge \neg s'[2] \wedge \neg s'[1] \wedge \neg s'[0]) \\ & \vee (\neg s[3] \wedge \neg s[2] \wedge \neg s[1] \wedge \neg s[0] \wedge \neg s'[3] \wedge s'[2] \wedge s'[1] \wedge s'[0]) \\ & \vee (\neg s[3] \wedge s[2] \wedge s[1] \wedge s[0] \wedge \neg s'[3] \wedge s'[2] \wedge \neg s'[1] \wedge s'[0]) \\ & \vee (\neg s[3] \wedge s[2] \wedge \neg s[1] \wedge s[0] \wedge \neg s'[3] \wedge \neg s'[2] \wedge s'[1] \wedge \neg s'[0]) \\ & \vee (\neg s[3] \wedge s[2] \wedge \neg s[1] \wedge s[0] \wedge s'[3] \wedge \neg s'[2] \wedge s'[1] \wedge \neg s'[0]) \\ & \vee (s[3] \wedge \neg s[2] \wedge s[1] \wedge \neg s[0] \wedge s'[3] \wedge \neg s'[2] \wedge \neg s'[1] \wedge \neg s'[0]) \\ & \vee (s[3] \wedge \neg s[2] \wedge \neg s[1] \wedge \neg s[0] \wedge s'[3] \wedge s'[2] \wedge s'[1] \wedge s'[0]) \\ & \vee (s[3] \wedge s[2] \wedge s[1] \wedge s[0] \wedge s'[3] \wedge s'[2] \wedge \neg s'[1] \wedge s'[0]) \\ & \vee (s[3] \wedge s[2] \wedge \neg s[1] \wedge s[0] \wedge s'[3] \wedge \neg s'[2] \wedge s'[1] \wedge \neg s'[0]) \end{aligned}$$

$$\vee (s[3] \wedge s[2] \wedge \neg s[1] \wedge s[0] \wedge \neg s'[3] \wedge \neg s'[2] \wedge s'[1] \wedge \neg s'[0])$$

The structure can be described by the input language for NuSMV which is used in our bounded model checker, BMC4PPTL, as shown in figure 3.

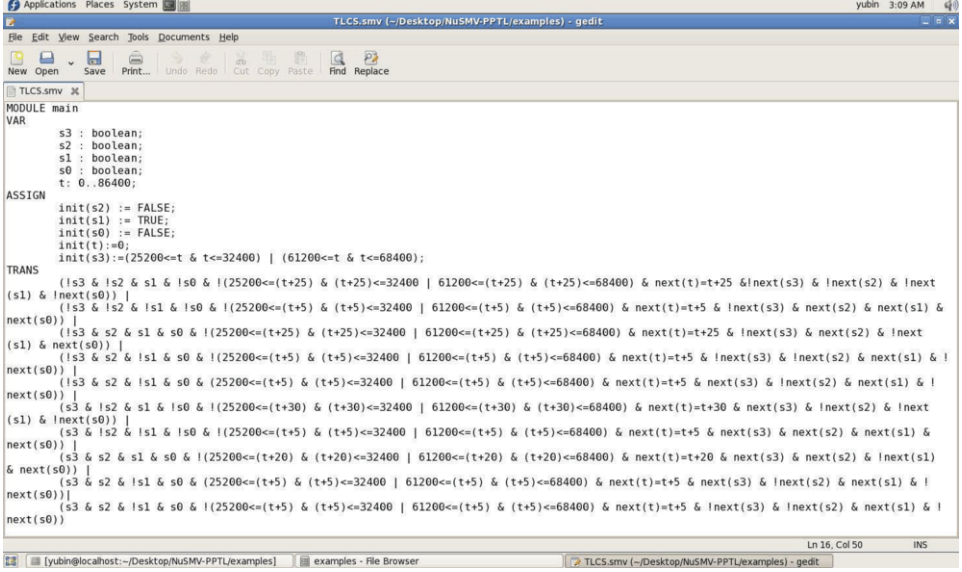


Fig. 3. The program in the NuSMV language for TLCS

4 Bounded Model Checking for TLCS

First we consider a safety property. At any state, the yellow or green light of the south-north direction works when the red light of the east-west direction is on, and the yellow or green light of the east-west direction works when the red light of the south-north direction is on. We can specify this safety property by a PPTL formula as follows:

$$\begin{aligned} F1 &\equiv \Box((s[2] \wedge \neg s[1] \wedge s[0]) \vee (s[2] \wedge s[1] \wedge s[0]) \vee (\neg s[2] \wedge \neg s[1] \wedge \neg s[0]) \\ &\quad \vee (\neg s[2] \wedge s[1] \wedge \neg s[0])) \\ &\equiv \Box((s[2] \wedge s[0]) \vee (\neg s[2] \wedge \neg s[0])) \\ &\equiv \Box(s[2] \leftrightarrow s[0]) \end{aligned}$$

The property $F1$ can be represented as $\Box p$, where p is $s[2] \leftrightarrow s[0]$.

With BMC for PPTL, we are trying to find a counterexample of the property $F1$, which means we are looking for a witness for $\neg F1$. The existence of such a witness indicates that the property $F1$ is false. If, on the other hand, no such witnesses can be found, it means that this property holds up to the given bound.

Assume the bound $k = 3$. Unfolding the transition relation, we can get the following formula:

$$M_3 \equiv I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3)$$

The loop condition is:

$$L_3 \equiv \bigvee_{l=0}^3 L(3, l) \equiv \bigvee_{l=0}^3 T(s_3, s_l)$$

By means of normal form of PPTL formulas, we can work out $NF(\neg F1)$:

$$NF(\neg F1) \equiv NF(\neg \Box p) \equiv (\neg p \wedge \varepsilon) \vee (\neg p \wedge \text{more}) \vee (\bigcirc \Diamond \neg p)$$

According to the translation rules for PPTL formulas, we can get the translation of $\neg F1$ for the path with a $(3, 0)$ – loop as follows:

$$\begin{aligned} & (\neg F1)_{(0,3,0)} \\ & \equiv (\neg \Box p)_{(0,3,0)} \\ & \equiv (\Diamond \neg p)_{(0,3,0)} \\ & \equiv ((\neg p \wedge \varepsilon) \vee (\neg p \wedge \text{more}) \vee \bigcirc(\Diamond \neg p))_{(0,3,0)} \\ & \equiv \neg p(s_0) \vee (\Diamond \neg p)_{(1,3,0)} \\ & \equiv \neg p(s_0) \vee \neg p(s_1) \vee (\Diamond \neg p)_{(2,3,0)} \\ & \equiv \neg p(s_0) \vee \neg p(s_1) \vee \neg p(s_2) \vee (\Diamond \neg p)_{(3,3,0)} \\ & \equiv \neg p(s_0) \vee \neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3) \vee (\Diamond \neg p)_{(0,3,0)} \end{aligned}$$

In the equation above, $(\Diamond \neg p)_{(0,3,0)}$ occurs again when it is decomposed. By using the fixpoint theory, we can get the following equation:

$$(\neg F1)_{(0,3,0)} \equiv \neg p(s_0) \vee \neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3)$$

Putting everything together we can get the following boolean formula:

$$\begin{aligned} & [M, \neg F1]_3 \\ & \equiv I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge (\neg L_3 \wedge (\neg F1)_{(0,3)} \vee \bigvee_{l=0}^3 (L(3, l) \\ & \quad \wedge (\neg F1)_{(0,3,l)})) \end{aligned}$$

As shown in figure 2, there exists one path whose length is 3. Because this path has a $(3, 0)$ – loop, it results in the following formula:

$$[M, \neg F1]_3 \equiv I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge (L(3, 0) \wedge (\neg F1)_{(0,3,0)})$$

We find that there is no assignment satisfying the formula $[M, \neg F1]_3$, which means that there is no counterexample of the property $F1$ when the bound is 3.

By our bounded model checker for PPTL, we can get the result for $F1$ when the bound is 3 as shown in figure 4.

```

yubin@localhost:/home/yubin/Desktop/NuSMV-PPTL/examples
File Edit View Terminal Tabs Help
[root@localhost examples]# ./BMC4PPTL -bmc -bmc_length 3 TLCs.smv
-- The PPTL property is: [](s2<->s0)
-- no counterexample found with bound 0
-- no counterexample found with bound 1
-- no counterexample found with bound 2
-- no counterexample found with bound 3

```

Fig. 4. The result of BMC4PPTL for $F1$ when the bound is 3

As expected, the system satisfies this safety property in 3 time steps. If we wish, we can increase the bound and perform the process of BMC again.

Now we check a periodically repeated property which shows the advantages of PPTL. In the system, there are two modes. As we can see in figure 2, the states 0010 and 0111 alternately appear in the left path. The state 0010 lasts 25 seconds and the state 0111 lasts 25 seconds because this path dose not represent the rush hours. The green light of the east-west direction and the red light of the south-north direction are on in state 0010. The red light of the east-west direction and the green light of the south-north direction are on in state 0111. So in mode 0, a property which

means that every other state in which the green light of the east-west direction and the red light of the south-north direction are on appears can not be satisfied. This is a typical example for showing the limitation of the expressive power of LTL. This property cannot be described by an LTL formula since the property is full regular. Certainly, we can specify this property by a PPTL formula as follows:

$$F2 \equiv (q \wedge \varepsilon); (\bigcirc^2(q \wedge \varepsilon))^*$$

In the formula above, q represents $\neg s[3] \rightarrow (\neg s[2] \wedge s[1] \wedge \neg s[0])$ where $\neg s[3]$ means that the system runs in mode 0 and $\neg s[2] \wedge s[1] \wedge \neg s[0]$ stands for the state in which the green light of the east-west direction and the red light of the south-north direction are on.

Let us consider a case where the bound $k = 2$. Unfolding the transition relation, we can get the following formula:

$$M_2 \equiv I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2)$$

The loop condition is:

$$L_2 \equiv \bigvee_{l=0}^2 L(2, l) \equiv \bigvee_{l=0}^2 T(s_2, s_l)$$

According to the translation rules for PPTL formulas, we can get the translation of $\neg F2$ for the path without loops as following:

$$\begin{aligned} & (\neg F2)_{(0,2)} \\ & \equiv (\neg((q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*)))_{(0,2)} \\ & \equiv ((\neg(q \wedge \varepsilon) \vee (q \wedge \bigcirc \neg(\bigcirc(q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*))) \vee (\neg q \wedge \bigcirc true))_{(0,2)} \\ & \equiv q(s_0) \wedge (\bigcirc \neg(\bigcirc(q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*)))_{(0,2)} \vee \neg q(s_0) \\ & \equiv q(s_0) \wedge (\neg(\bigcirc(q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*)))_{(1,2)} \vee \neg q(s_0) \\ & \equiv q(s_0) \wedge (\varepsilon \vee (true \wedge \bigcirc \neg((q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*))))_{(1,2)} \vee \neg q(s_0) \\ & \equiv q(s_0) \wedge (\bigcirc \neg((q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*)))_{(1,2)} \vee \neg q(s_0) \\ & \equiv q(s_0) \wedge (\neg((q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*)))_{(2,2)} \vee \neg q(s_0) \\ & \equiv q(s_0) \wedge (\neg q(s_2) \vee ((q \wedge \bigcirc \neg(\bigcirc(q \wedge \varepsilon); ((\bigcirc \bigcirc (q \wedge \varepsilon))^*))))_{(2,2)} \vee \neg q(s_0) \\ & \equiv q(s_0) \wedge (\neg q(s_2) \vee q(s_2) \wedge false) \vee \neg q(s_0) \\ & \equiv \neg q(s_0) \vee \neg q(s_2) \end{aligned}$$

Putting everything together we can get the following boolean formula:

$$\begin{aligned} & [M, \neg F2]_2 \\ & \equiv I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg L_2 \wedge (\neg F2)_{(0,2)} \vee \bigvee_{l=0}^2 (L(2, l) \wedge (\neg F2)_{(0,2,l)})) \end{aligned}$$

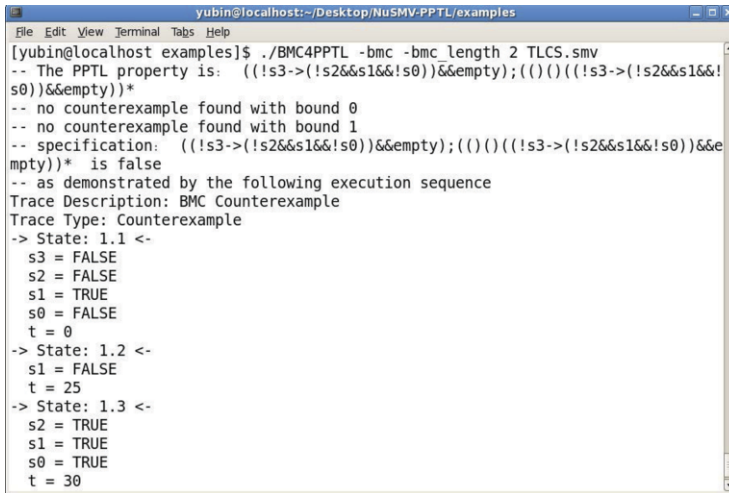
As shown in figure 2, there exists one path whose length is 2. Because this path has no loops, the formula above can be simplified as the follows:

$$[M, \neg F2]_2 \equiv I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg L_2 \wedge (\neg F2)_{(0,2)})$$

The assignment 0010, 0000, 0111 satisfies $[M, \neg F2]_2$. This assignment corresponds to a path from the initial state 0010 to the state 0111 that violates the periodically repeated property.

The verification result using BMC4PPTL for $F2$ with the bound 2 is shown in figure 5.

From the result in figure 5, we can see that a counterexample is found out when the bound is 2. The verification result of $F2$ is consistent with what we expect.



```

yubin@localhost:~/Desktop/NuSMV-PPTL/examples
File Edit View Terminal Tabs Help
[yubin@localhost examples]$ ./BMC4PPTL -bmc -bmc length 2 TLC5.smv
-- The PPTL property is: ((!s3->(!s2&s1&&!s0))&&empty);(!)()(!s3->(!s2&s1&&!s0))&&empty))*
-- no counterexample found with bound 0
-- no counterexample found with bound 1
-- specification: ((!s3->(!s2&s1&&!s0))&&empty);(!)()(!s3->(!s2&s1&&!s0))&&empty))* is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  s3 = FALSE
  s2 = FALSE
  s1 = TRUE
  s0 = FALSE
  t = 0
-> State: 1.2 <-
  s1 = FALSE
  t = 25
-> State: 1.3 <-
  s2 = TRUE
  s1 = TRUE
  s0 = TRUE
  t = 30

```

Fig. 5. The result of BMC4PPTL for $F2$

5 Conclusion

In this paper, we analyze a simple but practical traffic light control system. The Kripke structure is given according to the requirement specification. Then, we use the BMC for PPTL to verify two related properties. The verification result using our model checker, BMC4PPTL, shows that the system is consistent with the requirement specification. However, we have not investigated the complexity of the process of BMC. To examine our method, several case studies with larger examples are required. Besides, it is necessary for us to further improve our BMC tool BMC4PPTL in the near future.

References

- [1] Biere, A., A. Cimatti, E. M. Clarke, O. Strichman and Y. Zhu, *Bounded model checking*, Advances in computers **58** (2003), pp. 117–148.
- [2] Burch, J. R., E. M. Clarke, K. L. McMillan, D. L. Dill and L.-J. Hwang, *Symbolic model checking: 10²⁰ states and beyond*, Information and computation **98** (1992), pp. 142–170.
- [3] Cavda, R., A. Cimatti, C. A. Jochim, G. Keighren, E. Olivetti, M. Pistore, M. Roveri and A. Tchaltsev, *Nusmv 2.5 user manual* (2010).
- [4] Clarke, E. M. and E. Emerson, *Design and synthesis of synchronization skeletons using branching time temporal logic*, Logics of Programs (1982), pp. 52–71.
- [5] Clarke, E. M., O. Grumberg and D. E. Long, *Model checking and abstraction*, ACM Transactions on Programming Languages and Systems (TOPLAS) **16** (1994), pp. 1512–1542.
- [6] Clarke, E. M., O. Grumberg and D. A. Peled, “Model checking,” MIT press, 1999.
- [7] Clarke, E. M., D. E. Long and K. L. McMillan, *Compositional model checking*, in: *Proceedings of the 4th Annual Symposium on Logic in Computer Science*, IEEE, 1989, pp. 353–362.
- [8] Coudert, O. and J. C. Madre, *A unified framework for the formal verification of sequential circuits*, in: *Proceedings of the 1990 International Conference on Computer-Aided Design*, IEEE, 1990, pp. 126–129.
- [9] Duan, Z., “An extended interval temporal logic and a framing technique for temporal logic programming.” Ph.D. thesis, University of Newcastle upon Tyne (1996).

- [10] Duan, Z., “Temporal logic and temporal logic programming,” Science Press, 2005.
- [11] Duan, Z., M. Koutny and C. Holt, *Projection in temporal logic programming*, in: *Logic Programming and Automated Reasoning*, Springer, 1994, pp. 333–344, (A technical report No. 452, University of Newcastle Upon Tyne, 1993, is available).
- [12] Duan, Z., C. Tian, M. Yang and J. He, *Bounded model checking for propositional projection temporal logic*, in: *Computing and Combinatorics*, Springer, 2013, pp. 591–602.
- [13] Duan, Z., C. Tian and L. Zhang, *A decision procedure for propositional projection temporal logic with infinite models*, *Acta Informatica* **45** (2008), pp. 43–78.
- [14] Holtzman, G., *The spin model checker, primer and reference manual* (2003).
- [15] McMillan, K. L., “Symbolic model checking,” Springer, 1993.
- [16] Pnueli, A., *The temporal logic of programs*, in: *Proceedings of the 18th Annual Symposium on the Foundations of Computer Science*, IEEE, 1977, pp. 46–57.
- [17] Tian, C. and Z. Duan, *Propositional projection temporal logic, büchi automata and ω -regular expressions*, in: *Theory and Applications of Models of Computation*, Springer, 2008, pp. 47–58.