Symbolic Model Checking for Embedded Systems: A Case Study *

Tao Pang and Zhenhua DuanI

ICTT and ISN Lab, Xidian University, Xi'an 710071, P.R. China
t _pang0126.com, zhhduan©mail.xidian.edu.cn

Abstract. This paper is concerned with symbolic model checking for embedded systems. To this end, Propositional Projection Temporal Logic (PPTL) and a symbolic model checking (SMC) algorithm for PPTL [8] are briefly introduced. As a case study, a railroad crossing control system (RCCS) is presented to illustrate how SMC for PPTL can be utilized in the specification and verification of embedded systems.

Keywords: Propositional Projection Temporal Logic, Symbolic Model Checking, Verification, Embedded Systems

1 Introduction

Model checking [1] is a widely used automatic formal verification approach. With this technique, the system is modeled as a state-transition structure while the specification is expressed in temporal logic. Since the system model mostly rely on explicit manipulation of state space, the model checking procedure suffers from the so called state space explosion problem. To conquer this problem, several approaches [5-8] have been proposed with success. In particular, in [8], authors put forward a symbolic model checking (SMC) algorithm for Propositional Projection Temporal Logic (PPTL) [2] which represents the transition relation of the system model with boolean functions, and then searching the system model for states satisfying the given PPTL formula based on Reduced Ordered Binary Decision Diagrams (ROBDDs) [4].

Embedded systems pervade in almost every aspect of our daily life. In embedded real-time systems, certain actions must accomplish within a limited time bounds or start after some point of time. For instance, the specification for a bus arbiter to be verified is: "a grant sinal is given between 15ns and 4Ons after the request signal" or "a bus never be occupied for more than lOns". Though numbers of temporal logics have been proposed to verify properties of concurrent systems, such as Computation Tree Logic (CTL) and Linear Temporal Logic (LTL) [1], they are not powerful enough to treat the above real-time properties. Fortunately, all these properties can be conveniently specified by PPTL

* This research is supported by the NSFC Grant No. 61003078, 61133001 and 60910004, 973 Program Grant No. 2010CB328102 and ISN Lab Grant No. ISN1102001. Corresponding author.

with chop and projection operators, i.e. request; len(15); len(25) A Ogrant and -,((len(10); true) A Elbus_isoccupied). It has been proved that PPTL has the expressive power of full regular [10]. In this paper, as a case study, we perform SMC for PPTL on the verification of several real-time properties for an embedded railroad crossing control system.

The rest of the paper is organized as follows. The following section briefly reviews the syntax and semantics of PPTL as well as its symbolic model checking algorithm. In section 3, a case of RCCS is studied by means of the SMC for PPTL. Finally, conclusions are drawn in section 4.

2 symbolic model checking for PPTL

2.1 Propositional Projection Temporal Logic

Our underlying logic is Propositional Projection Temporal Logic (PPTL) [2]. In the following, we briefly introduce its syntax and semantics.

Syntax: Let *Prop* be a countable set of propositions. PPTL formulas *0* are defined as follows:

P I 0 0 HO 100/ 02 I (01, • • • 0m) *Pri*

where *p E Prop, 01, , O_m* and *0* are all well-formed PPTL formulas. 0 (next) and *prj* (projection) are basic temporal operators.

Syntax: Follwoing the definition of Kripke structure [3], a state *s* is defined as a mapping from *Prop* to *B* = {true, false}. We will use *s[p]* to denote the valuation of *p* at the state *s*.

An interval *a* is a finite or infinite sequence of states. The length of *a, I a I ,* is the number of states minus 1 if *a* is finite, and w otherwise.We extend the set of non-negative integers NO to include w, i.e. *N„,* = N_o U {w}, and extend the relational operators, =, <, <, to N_W by considering *w* = w, and for all i E *No,* i < w. Furthermore, we define -< as < -{(w, w)}. For simplicity, we will denote *a* as < so, , s1,1 >, where s1,1 is undefined if *a* is infinite. Let *a* be an interval and *r_1, . , rh* be integers *(h* > 1) such that 0 < r1 < • • • < *rh 'al.* The projection of *a* onto *r1, . , rh* is the projected interval *a*j. , *rh)* =< *s_{t1}, st_2* , • • • , *st,* > where t_1, , *t1* is the longest strictly increasing subsequence obtained from r_1, , *rh* by deleting all duplicates. For instance, < s_o, s_i, s_2, s_3, s4 >,[. (0, 0, 2, 2, 2, 3) =< so, s2, s3 >

An interpretation is a triple I = *(a, k, j),* where *a* is an interval, *k* is an integer, and *j* an integer or w such that *k j < 101.* The notation *(a, k, j)* means that formula 0 is interpreted and satisfied over the subinterval *a(k...$_3$)* with the current state being *sk.* The satisfaction relation (=) is inductively defined as follows:

1. I = *p* •<=> *s* *k[p]* = *true,* for any *p* E *Prop*
2. / = - 0 <=> Z V 0
3.z=fi1V02 <=> .01 or / 02
4.Z 00 <=> *k* <*j* and *(o¯, k + 1, j)*
5. *¹ H (⁶1h,* • • •*, Om) prj* there exist integers 7-0 < r1 < • • • < rm <
such that (a, ro, ri) 01, (a, rt) = 0t,^{1} < *1* < *m,* and (a', 0, 1^{0}1)
0, for one of the following *a':*
(a) r_m <*j* and *a'* = a (ro, • • • , rm) • u(r_{m+},...i) or
(b) r_m =*j* and *a'* = *o¯* 4, (ro, , rh) for some 0 < *h* < *m*

2.2 Symbolic Model Checking for PPTL

Similar to that of CTL, SMC for PPTL checks whether a Kripke structure *M = (8, I, R, L)* satisfies a PPTL formula cb by calculating the subset of *S* where *0* holds, denoted by *Sat(q).* This idea is formalized in the following function checkPPTL, where *0* denotes the PPTL formula to be checked and checkPPTL returns an ROBDD representing *Sat(q).*

```
function checkPPTL (0 : PPTL) : ROBDD
begin
  Sat(0) = false : ROBDD
  for i=1 to n Sat(0_i) = false; end for
  ONF = NF (0); /* NF(0) = fiiL 1 Cbi where Oi can either be
  the terminating part (06 A r) or the future part 0d A Q0fi */
  for i=1 to 71
  case
          Odi A e : Sat(0i) = Sat(O_ei);
            (66 A 00fi (Op is not marked) : mark Of
             Sat(4) = Sat(0_ri) fl preStates(checkPPTL(Ofi : PPTL))
          Oci A 00fi (Op is marked):
              Sat(4) = Sat(4_i) fl preStates(f ixpoint(r(Sar(Ofi))));
  end case
  end for
  Sat(0) = Sat(0_1) U Sat(0_2) U . . . U Sat(O_n) ;
  return Sat(q) : ROBDD;
end
```

Fig. 1. Symbolic model checking of PPTL formulas

With checkPPTL, the model checking procedure can be performed in the following way: firstly, invoking checkPPTL to calculate *Sat(-0).* Secondly, if *Sat(-'0) f1*I equals to false, namely there is no states *s* E I in which -icb holds, we have *M 0.* Further, if *Sat(-s)* f1I false, then starting from any state in *Sat(i0)* i11¯, we can always find a path HcE of *M* over which -0 is satisfied. The detail of checkPPTL can be found in [8].

3 A Case Study: Railroad Crossing Control System

As a case study, we are concerned with how the following railroad crossing control system [11] can be verified by means of SMC for PPTL.

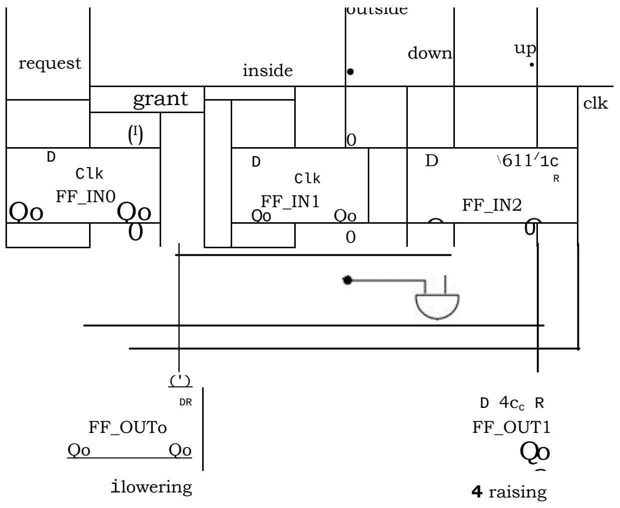


Fig. 2. Railroad Crossing Control System

In the RCCS system, a train tries to pass through a railroad crossing in such a way that the gate must in the "down" state when a train is entering the crossing. The hardware circuits of RCCS system is shown in Fig. 2.

(1) The train notifies the controller at least 2 minutes before entering the crossing, and will exit after at most 5 minutes;

(2) After 1 minute, the controller will gradually drop the gate. The gate will be down within 1 minute;

(3) Within 1 minute after the train has exit, the controller will start lifting the gate. The gate will be up within 1-2 minutes;

(4) The train has 2 statuses: inside and outside while the gate has 4 *statuses: lowering, raising, up and down.*

The purpose of RCCS is to ensure that: (a) the gate is down before the train enters the crossing; (b) the gate is never down for more than 10 minutes. These properties can be expressed in PPTL as follows, where len(n); *(gate_status = down)* denotes that after n minutes *gate_status = down* holds:

(a) *train_request = true* (len(2); *(gate_status = down))*
(b) *0(gate_status = down);111(gate_status = down)* A $V_{n=}^{io}$ $_1$ len(n);
0(gate_status = up)

To present them in a standard way, atomic propositions *q, t, d, 1* and r are employed to denote *train_request = true, train_status = inside, gate_status = down, gate_status = lowering* and *gate_status = raising* respectively. Successively, we have:

(a) *q* —> (len(2); (b) O*d ;Ed* A v^{i}_{n} ' $_1$ len(n); 10(¬il A -,*c1* A -,r)

Moreover, we assume that and -,1A-idA respectively represent *train_status = outside* and *gate_status = up.* Then, we can model the RCCS system as a

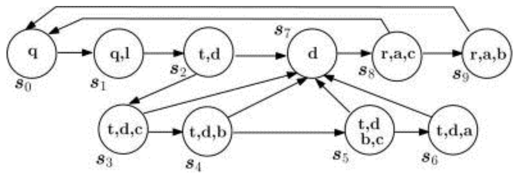


Fig. 3. Model of RCCS system

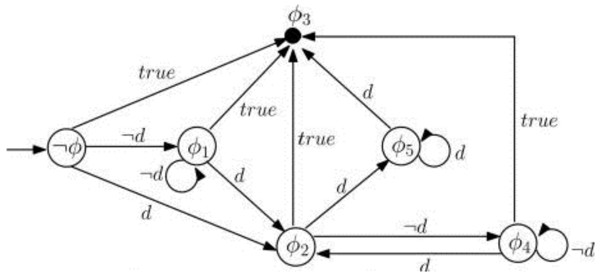
Kripke structure *M = (8, I, R, L)* defined on *AP = {q,t,d,l,r},* i.e. the set of atomic propositions, as show in Fig. 3.

Apparently, there are five states, i.e. s2 ti 86, labeled with *{t, d}.* To encoded the system with boolean functions, three additional atomic propositions a, *b* and *c* are added to *AP* for distinguishing s2 s6 between each other. Moreover, a, *b* and *c* can also be utilized to differentiate s_8 and s9. We assume a fixed order on atomic propositions in *AP as q < t < 1 < d < r < a < b < c,* and assign each atomic proposition a corresponding boolean variable xi (0 < i < 7), then each state *s* E *S* can be represented with the boolean vector (x0, xi, x2, x3, x4, x5, x6, *x7),* where xi equals to 1 if its corresponding atomic proposition holds in state *s* and x_i = 0 otherwise.

We focus on the formula 0 Od ; Ed A $_{\text{Vim}^\circ\ 1}$ len(n); OH A -*id* A -,r). Accordingly, is transformed to its LNFG [9] as depicted in Fig. 4. With checkPPTL, since the formula —1cbtrueAEV—idA00$_1$Vd()0$_2$, we have *S at(—icb) =*

Sat(true A*E)* U *(Sat(—d)* preStates *(Sat(01)))* U *(Sat(d)* npreStates(Sat(02)))

Consequently, this first step toward calculating *Sat(-iq5)* is to determine *Sat(q5$_1$)* and *Sat(0$_2$).* Similarly, we can infer that the computation of *Sat(q5$_1$)*



1: 1=1-0; Cid A 4$_{=1}$ len(n); OH A A -*j*r))

2: AL0 -,((Ed A len(n)); OH A A --,r)) A 04

3: 6

4: 0(EldA len(n);O(/ A

5: *02* A *d*

Fig. 4. LNFG of formula -,(0'd ; Lid A V$_n$$^{i\circ}$ $_1$ len(n); OH A -id A -,r))

and *Sat(cb2)* depends on *Sat(04)* and *Sat(cb5)•*

Intuitively, we have *Sat(d) = xo • xi • x2 • x3* • • *xi • x2 • x3 • x4 • x5 • xs •* + • • *x2 • x3 • x4,* namely the boolean encoding of set {s2, s3, s4, s5, s6, s7}. Successively, as the normal form of formula 05 is dAeVdA0q55, we have

$$Sat(0_5) = Sat(d \text{ A } 6) \text{ U } Sat(d) \cap \text{preStates}(\text{Sat}(0_5))$$

With the procedure fixpoint by initially assigning *B* as *Sat(d),* we can figure out that *Sat(0₅)* equals to false : ROBDD. Similarly, we can figure out that $Sat(0_4)$ = false. Accordingly, we calculate each $Sat(0_i)$ $(1 < i < 4)$ in a way backtracking the paths in the LNFG depicted in Fig. 4. As a consequence, we have $Sat(-0) = Sat(0_1) = Sat(0_2) = Sat(q5_4) = Sat(0_5)$ = false.

Furthermore, since *Sat(-0)* = false, *Sat(-iq)* n I = false. Hence, there is no state *s* E *S* in which —0 holds. On the other side, *cb* holds along all paths stemming from so. We can prove that *M* = *q* (len(2); *d)* in the same way.

Finally, by proving properties (a) and (b) with SMC for PPTL, we confirm the correctness of this RCCS system.

4 Conclusion

In this paper, we briefly introduce PPTL and its corresponding symbolic model checking algorithm. This enables us to specify and verify real-time properties of embedded systems with PPTL, and alleviate the state space explosion problems. Then, a case of railroad crossing control system is studied to show the correctness and feasibility of SMC for PPTL. However, it should be noted that this paradigm just considers simple real-time properties. In the future, we will further explore the specification and verification of quantitative properties for embedded systems with PPTL in a systematic fashion.

References

1. Clarke, E. M., Grumberg, J. 0., Peled, D. A.: Model Checking. MIT Press, 1999.
2. Duan, Z.: Temporal Logic and Temporal Logic Programming. Science Press, 2006.
3. Kripke, S. A.: Semantical analysis of modal logic I: normal propositional calculi. Z. Math. Logik Grund. Math. 9, pp. 67-96, 1963
4. Bryant, R. E.: Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers C-35,* 6(Aug), pp. 677-691, 1986.
5. Clarke, E. M., Long, D, E., McMillian, K. L.: Compositional model checking. In proceedings of the 4th Annual Symposium on Logic in Computer Science. IEEE Computer Society Press, Los Alamitos, Calif., pp. 46-51, 1989.
6. Burch, J. R., Clarke, E. M., McMillian, K. L.: Symbolic Model Checking: 10^{20} States and Beyond. Fifth Annual IEEE Symposium on Logic in Computer Science, pp. 428-439, 1990.
7. Biere, A., Cimatti, A., Clarke, E. M., Strichman, 0. and Zue. Y.: Bounded Model Checking, volume 58 of Advances in computers. Academic Press, 2003.
8. Pang, T., Duan, Z., Tian, C.: Symbolic Model Checking for Propositional Projection Temporal Logic. In: He, Z., Yin, B. TASE2012, 2012
9. Duan, Z., Tian, C.: An Improved Decision Procedure for Propositional Projection Temporal Logic. ICFEM 2010. pp. 90-105, 2010
10. Tian, C., Duan, Z.: Propositional Projection Temporal Logic, Buchi Automata and w-Regular Expressions. In proceedings of TAMC'08. pp. 47-48, Springer, 2008.
11. Moller, J. B.: Symbolic Model Checking of Real-Time Systems using Difference Decision Diagrams. PhD thesis, IT University of Copenhagen, April 2002