Model Checking of Real-time System Based on Timed CSP

Jing Guo¹, Zhong-wei Xu¹, Meng Mei¹

¹School of Electronics & Information Engineering, Tongji University, Shanghai, China

Abstract. With the continuous increase in the size and complexity of a real-time computer system, the use of formal verification methods in software development is also on the rise. The traditional formal verification method is not fully applicable to the development of actual system life cycle. Therefore, this paper presents a new real-time system verification method. It takes the deadlock timed Büchi automata as the medium, and translates the timed temporal logic into timed communicating sequential process language. The task event is also joined, which can be directly used for the detection of refinement tool FDR. The method verifies the situation of deadlock. To establish the link between the conventional model checking and refinement model checking can well combine the advantages of both and improve system security and reliability.

Keywords: refinement relation; formal verification; real-time system; safety

1 Introduction

Recent years, model checking has been a powerful automatic formal verification technique for establishing correctness of hardware and software systems. The method based on temporal logic is to model system as a labeled transition system (LTS) and specification as temporal logic formula. Then we decide whether the formula holds in the LTS. Another approach is based on the notion of refinement and is frequently used by verification such as Communicating Sequential Processes (CSP). In this conception, the system and the property are modeled by same formalisms, the aim is to verify the latter is refinement of the former. The two approaches have merits and demerits and there is connection between them. To study the connection, a significant number of techniques have been proposed.

Temporal logic can also used for describing system in the form of temporal logic of actions[1], which is verified by the notion of refinement and development. Therefore, the idea allows simulation of a single high level step by several lower level steps. Spatio-temporal logic[2] is somewhat alike to temporal logic, but different in the operators. It extends the meaning of syntax, where arise in the development of mobile systems, but the decidability of the model checking problem is negligible. The relationship between refinement-oriented specification and specification using a temporal logic is considered, and further the conversion of temporal logic into process
algebra[3]. Researchers verified equivalence of computation tree logic and failure trace, in which the former is expressed in the form of test sequence[4]. Despite validity of verification, the infinite states are born in the test case. The temporal logics LTL, CTL and the \( \mu \)-calculus convert to the formal language \( Z \) in the idea of refinement, which also used for language B, VDM based on state. Every conversion rule does not illustrate the main distinction[5].

In the development of system, system construction is stepwise process. The traditional verification method is unfavourable for changes of systems, while the refinement applies to verification about life cycle. Temporal logic is more convenient and common. Therefore, rise both union, temporal logic express the property and verify the property based on refinement. Real time factors are of great importance, so these must be taken into account. In this paper, we study the possibility of doing TLTL(timed linear temporal logic) model checking on TCSP specification in the context of refinement. Firstly, we model property as TLTL and define timed Büchi automata. Secondly, through the medium of timed Büchi automata TLTL is converted into TCSP, the verification is discussed on two cases in the refinement framework.

2 Refinement Verification based on TLTL

Translation of TLTL into TCSP is of first importance. Timed Büchi automata tests emptiness, and events being tested are visible. During execution doing nothing but empty events, process goes into deadlock state. We start with definition of deadlock timed Büchi automata.

\[
(\text{Deadlock Timed Büchi Automata}) \quad A_d = (C, Q, B, q_0, E, I_c, F, D)
\]

1) \( C \) refers to a set of finite clock.
2) \( Q \) refers to a set of discrete states, the initial state is \( q_0 \).
3) \( B \) refers to a set of finite channel. Every channel has two visible action: input action \( a^? \) indicates that through channel \( a \); output \( a^1 \) indicates that through channel \( a \).
4) \( E \) refers to a set of transition(\( e = (q, B, h, G(c), R, q') \in E \)), \( q \) and \( q' \) are separately source and destination state. \( B_n \) contains input and output actions. \( g \in G(c) \) is connection of clock constraint(\( g := c \gg n, g \land g; \text{true} \), \( n \in N_p \), \( \gg \in \{<,\leq,=,>,\geq\} \)), \( R_c \subseteq C \) is a set of clock sets which should be reset. Transition is a form of \( q \to q' \).
5) \( I_c : Q \to G(c) \) is a set of invariants mapping state to guard.
6) \( F \subseteq Q \) is a set of accepting states.
7) \( D \subseteq Q \) is a set of deadlock states.

\( A_d \) includes two acceptance conditions: traditional acceptance condition for Timed Büchi Automata and special acceptance condition end with deadlock state.
We should handle transition about deadlock state. Transition contains change of states and clock variables, and the accepting states including deadlock events are deadlock states. The deadlock transitions are removed from automata. The transition system of automata is of great importance for transformation of automata into TCSP, we define it as follows:

According to the above transition system, we establish the relationship between it and TCSP. We define the translation of process as follows:1) We map state \( q \) to a process \( P \in \Sigma \), name the initial state and take it as start point.2) For every non-accepting state, it is expressed as \( label(q) = a \rightarrow label(q) \). If clock is reset, \( d \) will be 0. If clock is increasing, \( d \) will be \( \delta \).3) For every accepting state, all destination states are \( q_1, q_2, \ldots q_n \) and actions are \( a_1, a_2, \ldots a_n \). We add one event accept, the choice of visible events are possible transitions. The process is \( label(q) = accept \rightarrow (a_1 \rightarrow label(q_1) \cdots a_n \rightarrow label(q_n)) \).4) For every deadlock states, all destination states are \( q_1, q_2, \ldots q_n \) and actions are \( a_1, a_2, \ldots a_n \). We add two events: deadlock and special. The process is \( label(q) = deadlock \rightarrow (a_1 \rightarrow special \rightarrow STOP \cdots \rightarrow a_n \rightarrow special \rightarrow STOP) \).

After the translation of Deadlock Timed \( \text{Buchi} \) automata into TCSP, can test as form of \( \left( P \sqcup T \right) \cup \left( \sum \cup \{v_i\} \right) \). \( S \) is description of system, \( T \) is the result of translation.

Not only do we consider emptiness, but also we must consider deadlock. So test has two conditions: the former accept is test event and deadlock, special should be hidden. We test whether the result has infinite accept, which is analogous to may test, test accept is refinement of \( \left( P \cup T \right) \cup \left( \sum \cup \{deadlock, special\} \right) \); the latter test whether \( deadlock \rightarrow STOP \) execute, which is analogous to must test, test \( deadlock \rightarrow STOP \) is refinement of \( \left( P \cup T \right) \cup \left( \sum \cup \{accept\} \right) \).
3 Verification Case: Railway Crossing

Fig. 1 The railway crossing system

The description of the crossing system is as follows:

Train = train.near → near.ind → enter.crossing g → leave.crossing g → out.ind → Train

Gate = down.command → down → confirm → Gate \
up.command → up → confirm → Gate

Controller = near.ind → down.command → confirm → Controller \
out.ind → up.command → confirm → Controller

Crossing g = Controller \n\nTCGSystem Train 

The events near.ind , out.ind model respectively tell the sensors that train has enter and leave. The events train.near , enter.crossing g , leave.crossing g model respectively the situations where the train is close to the crossing, the train enters the crossing, and the train leaves the crossing. down.command , up.command instruct the gate to go down and up respectively. C, G and T are set of events which describe controller, gate, and train respectively. If train enter the crossing, during 10 units of time up and down do not occur. Check the property, the form of TLTL is ¬◊ (enter.crossing g ⇒ x.t ≤ x + 10 ∧ ¬down ∧ ¬up ), the result TCSP of translation is as follows:

T = State1

State1 = enter.crossing g → State2 

[0,10] State2 = down → State3 

[0,10] State2 = up → State3

State3 = accept → ( (down → State3 ∪ (up → State3)) ) ,
\[
\text{State3} = \text{deadlock} \rightarrow ((\text{down} \rightarrow \text{special} \rightarrow \text{Stop}) \lor (\text{up} \rightarrow \text{special} \rightarrow \text{Stop}))
\]

The FDR tool support the test, take event represent one unit of time that establish the relationship between TCSP and CSP. Firstly, we test emptiness, test whether the result has infinite accept: \(Acept = accept \rightarrow \neg Acept\),

\[\text{assert Composition}1 \left[ T = Acept \right] \text{The test result is refinement fails to no infinite trace violates formula} \Rightarrow \text{OK, showing that it does not produce infinite accept.}\]

We test deadlock, test deadlock trace whether receive the negative of the property:

\[\text{Deadlock} = \text{deadlock} \rightarrow \neg \text{Stop}, \text{assert Composition}2 \left[ T = \text{Deadlock} \right] \text{The test result is refinement fails to no deadlock trace violates formula} \Rightarrow \text{OK, showing that there is not deadlock trace in the system.}\]

According to the two results, system meet the property.

5 Conclusions

This paper explored the relationship between TLTL and TCSP. We took deadlock timed \(Bu\chi\) automata as medium, realized the translation between them. TCSP provided a counter example if a refinement check fails and not. We highlighted a case, railway crossing, verified the property. The experiments showed that people can verify the TLTL property in the framework of FDR, so the result is to improve efficiency, and can be well used in the development of system.

References

4. Zhiyu ZHANG., Model Checking is Refinement-Computational Temporal Logic is Equivalent to Failure Trace Testing. Sherbrooke, Quebec, Canada: Bishop’s University,(2008)
5. John Derrick, Graeme Smith, Temporal-logic property preservation under Z refinement.
