The Specification and Verification of Real-time System Based on the Temporal Logic of Action

Tang Zheng-yi
College of Computer Science and Information
Guizhou university
Guiyang, China
william_falcon@126.com

Peng Chang-gen
College of Science
Guizhou university
Guiyang, China
peng_stud@163.com

Li Jun-tao
College of Computer Science and Information
Guizhou university
Guiyang, China
jtxq@21cn.com

Li Xiang
College of Computer Science and Information
Guizhou university
Guiyang, China
lixiang@gzu.edu.cn

***Abstract*—Real-Time system is the safety-critical system. Its safety should be ensured by the formal method. The different formal method has different advantages and deficiencies, the suitable combination of them can obtain a better effect. This paper introduces the timed automata and the temporal logic of action(TLA). The former is the modeling tool of Real-Time system, the oher is a logic which has a strong ability of system description and attribute specification. On this basis, we research the method of using TLA to describe the Real-Time system which is expressed by timed automata and verify it by an actual example.**

Keywords-Timed Automata; TLA; Real-Time System; Formal Method; Model Checking

I. INTRODUCTION

With the growing of software system's complexity, the problem of ensuring the accuracy and reliability in the Software development process must be solved. Especially in the field of safety-critical. Formal methods is one of the most promising technologies which can solve this problem. It is based on a rigorous mathematical foundation and can make the system have higher accuracy and credibility. Different formal method has different features, if we can make them combine, there will be better results.

Timed automata[1] is a formal theoretical model for the automated analysis and verification of the real-time system. Although the time-automata-based automatic analysis and verification(TLA) tools are in the continuous development, they all have defects. The temporal logic of action is a logic which can describes the system and specify the properties of system. The model checking tools based on the temporal logic of action can alleviate the problem of state-space explosion effectively. This paper describes the temporal logic of action and timed automata briefly and studies the method of expressing the real-time systems by TLA and timed automata. And we verify it by actual example, it gets good effect.

II. REAL-TIME SYSTEM AND TIMED AUTOMATA

Real-time system is a computing system with time constraints. The computer control system of reactor, flight control and railway scheduling and so on, are all belong to the real-time system[3]. The finish of these systems' behaviors are time-related. It means that they must satisfy the time constraints. For example, some behaviors must finish within one second. It is very important to ensure the accuracy and reliability of these systems. Because the operation of real-time systems are often associated with the external environment, it has many uncertainty. Many computer scientists believe that the formal method is the fundamental method which can solve the problem of real-time system's security.

Timed automata is a formal description method, Alur and Dill creat it by adding the time constraint mechanism to the traditional finite automata. Since the the finite automata has the concept of time, timed automata is always a standard model of describing the time system. Timed automata have a limited number of control positions and the clock with real value. There may be conversion between the two control locations, it can happen only when the clock constraints associated with it are all met. The occurrence changes the control position which timed automata remains and reset the corresponding clock.

The syntax of timed automata is defined as follows:

Definition 1 Clock and the set of clock constraints[4]: X is a finite set of automata, the set of clock constraints C(X) is the set of the formulas which generate by the syntax: $\Phi ::= (x \sim c) \mid \Phi_1 \wedge \Phi_2 \mid \text{true}$, where $x \in X$, $\sim \in \{<, \leq, \geq, >\}$, $c \in N^+$。

Definition 2 assignment of clock[4]: μ: $X \vert\!\!\longrightarrow R^+$, $\mu + t$ means $\forall x \in X, \mu(x + t) = \mu(x) + t, t \in R^+$. The μ meets the time constraint Φ (denoted as $\mu \models \Phi$), iff Φ is true in the assignment μ. $\mu[Y := 0]$ means $\forall x \in Y (Y \subseteq X)$ assigns 0, the others are unchanged.

Definition 3 Timed automata[1]: Timed automata is a sextuple on X: $TA = <Q, I, \Sigma, X, L, E>$, where:

- Q is the finite set of states, $I \in Q$ is the set of initial states.
- Mapping L: $Q \mid\!\!\longrightarrow$ C(X) assigns a time constraint to every node q.
- Σ is the finite set of transfer label.
- $E \subseteq Q \times C(X) \times \Sigma \times 2^X \times Q$ is the set of transfer. A transfer (q, Φ, σ, Y, q') means that state q can transfer to state q' by the transfer $\sigma \in \Sigma$ when the constraint is satisfied and the the clocks which belong to $Y(Y \subseteq X)$ are assigns 0.

Definition 4 Transfer relationship ($\to$)[4]:

- Delayed transfer: $(q, \mu) \xrightarrow{\delta} (q, \mu')$, if $\mu \models L(q)$ and $\mu' \models L(q)$, where $\mu' = \mu + \delta$, $\delta \in R^+$.
- Discrete transfer: $(q, \mu) \xrightarrow{\sigma} (q', \mu')$, if$(q, \Phi, \sigma, Y, q') \in E$, $\mu \models \Phi$ and $\mu' \models L(q')$, where $\mu' = \mu[Y := 0]$.

Delayed transfer expresses the time's transfer under a state. Discrete transfer expresses the transfer between states when time constraint is satisfied.

The following figure is a timed automata which describe the traffic light system:

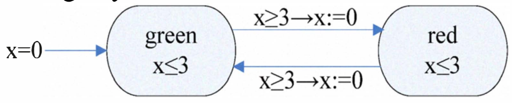


Figure 1. The timed automata of traffic light system

The state transition diagram in figure 1 expresses a simple traffic light system. In the first, the green light is on. After three time units, the red light is on and the green light is off. After three time units again, the green light is on and the red light is off. Over and over again.

III. THE TEMPORAL LOGIC OF ACTION AND THE TLA+ LANGUAGE

The temporal logic of action is a proposed by American scientist Leslie Lamport. It is a program logic which mixes temporal logic and action logic. It introduces the concepts of action and run, and makes the system and the properties of system to be expressed by the same logic. Microsoft, European Union and some companies and organizations have established the dedicated project team to make TLA to specify and verify system now. The main semanticses TLA are:

Definition 5 Primed Variable[2]: The next state value of variable. If we v use to express the value of variable v, then the primed variable v' expresses the value of variable v in next state.

Definition 6 Action[2]: It is made of variable, primed variable and the boolean expression which is expressed by constant. Variable is related to current state, primed variable is related to next state. It expresses the transfer relationship between states. A state pair (s, t) is called action A, iff $s[[A]]t \equiv A(\forall v: s[[v]] / v, t[[v]] / v')$ is true. "$s[[v]] / v$" means replacing the v in A by the value of v under the state s. "$t[[v]] / v'$" means replacing the v' in A by the value of v under the state t.

Definition 7 Stuttering Step[5]: Stuttering step is a special action. When it occurs, All of the variables in system will be unchanged.

Definition 8 Label Transfer System[6]: A label transfer system $LA = (Q, I, A, \delta)$, where:

- A valid assignment of all variables in the system is called state q, all of the states compose the state set(denoted as Q).
- Initial state $I \subseteq Q$.
- A is the set of all action.
- $\delta \subseteq Q \times A \times Q$.

Definition 9 Run [5]: The run of TLA is a sequence which is made of infinite states or finite states $\sigma = q_0 \xrightarrow{A_0} q_1 \xrightarrow{A_1} q_2 \cdots\cdots$, where $q_0 \in I$, $A_i \in A$, $(q_i, A_i, q_{i+1}) \in \delta$. Because TLA introduces stuttering step, finite sequence can be extended to infinite sequence by stuttering step. So the run in TLA is infinite.

Definition 10 Enabled [2]: A action $A \in A$ in the state $q_i \in Q$ is enabled, iff $\exists q_j \in Q$ and $(q_i, A_i, q_j) \in \delta$(denoted as *Enabled* A).

Definition 11 Weak Fairness[2]: For a run σ, if $\sigma \models \Box((\Diamond<A>_f) \vee (\Diamond\neg Enabled<A>_f))$, then the action A has weak fairness in the run σ(denoted as $\sigma \models WF_f(A)$).

Definition 12 Strong Fairness[2]: For a run σ, if $\sigma \models \Box((\Diamond<A>_f) \vee (\Diamond\Box\neg Enabled<A>_f))$, then the action A has strong fairness in the run σ(denoted as $\sigma \models SF_f(A)$).

TLA+ language constructs specification by form of module and combine multiple modules to form complicated specification by the methods of extension, instantiation or others. It is a language which is decribed by temporal logic. TLA+ supports many forms of specification, the standard form is $Init \wedge \Box[N]_v \wedge L$. Where $Init$ is initial state predicate, it describes all the legitimate initial state. N is next state relationship, it stipulates all the possible action in the system. v is tuple the which is made of all the variable, it is state function and include stuttering step. L is a temporal formula, it stipulates the liveness of system by conjunction of every action's liveness.

IV. DESCRIBING THE REAL-TIME SYSTEM BY THE TEMPORAL LOGIC OF ACTION

Although there have been several model checking tools based on the timed automata, these tools describe system and specify system property by using different language[7]. If describing real-time system and specifing property can use the same language, there will be many conveniences in many cases. For example, it is conducive to the stepwise refinement of real-time system and verify the consistency between the real-time system and specification. So, if we can use TLA+ language to decribe the real-time system which is described by timed automata, then system and property can be described by the same logic. Besides, the model checking tool based on the temporal logic of action has high efficiency when deal with the problem of state-space explosion.

There have beem some successful examples of using TLA+ language to decribe concurrent system[5, 6, 8], they all get good effect. In fact, the real-time system is the concurrent system with time constraints. In the basis of decribing concurrent system, it only need time constraint to decribe real-time system.

The state of real-time system is made of the assignments of all variables, we divide them into two parts:

- The set of time variables *X*: all of the clock variables.
- The set of action variables *S*: all of the other variables.

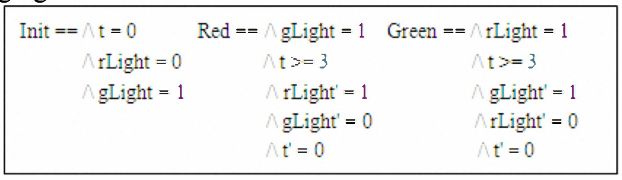
The set of time variable and the set of action variables constitute the set of variables $V \equiv X \cup S$.

The difficulty of using TLA+ to describe real-time system is exact description of action. So we divide the actions of real-time system into three parts:

Definition 13 $A \equiv BEnabled{<}A{>} \wedge TEnabled{<}A{>} \wedge$ MDY(V), where:

- *BEnabled<A>*: Action Enabled, an action A is Action Enabled in $q_i \in Q$, iff $\exists q_j \in Q$, $A(\forall v \in S$: q_i $[[v]] / v$, q_j $[[v]] / v')$ is true.
- *TEnabled<A>*: Time Enabled, an action A is Time Enabled in $q_i \in Q$, iff $\exists q_j \in Q$, $A(\forall v \in X$: q_i $[[v]] / v$, q_j $[[v]] / v')$ is true.
- MDY(V): $(\forall v \in S$: $v' = v \vee v' =$ NewValue$) \wedge$ $(\forall v \in X$: $v' = v \vee v' = 0)$。

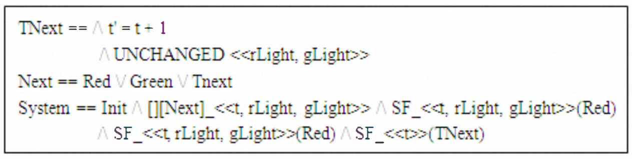
As an example, the traffic light system in the picture 1 uses variable t as time variable. When the value of variable gLight is 1, the green light is on. When the value of variable rLight is 1, the red light is on. At the same time, the system requires each state to sustain three time units. So it must stay on the "Red" state for three time units before operating. Summary, the enabled condition of action green is "rLight = 1", the enabled condition of action red is "gLight = 1", the time enabled condition is "t ≥ 3", MDY(V) is "rLight' = 0 ∧ gLight' = 1 ∧ t' = 0".



```
Init == /\ t = 0         Red == /\ gLight = 1    Green == /\ rLight = 1
        /\ rLight = 0          /\ t >= 3                /\ t >= 3
        /\ gLight = 1          /\ rLight' = 1           /\ gLight' = 1
                               /\ gLight' = 0           /\ rLight' = 0
                               /\ t' = 0                /\ t' = 0
```

Figure 2. The initial state and system action

Besides, it still needs a action Timer to stipulate the increasing speed of time variable. Here, we assume that the increasing speed of time variable is one time unit. Because the operation of each action is instantaneous, the other variables are unchanged when the action Timer is operatting. The next state relationship of system is the disjunction of the actions above. The liveness request is strong fairness because the enabled state of the actions above are infinite but not successive. As shown in Figure 3.



```
TNext == /\ t' = t + 1
         /\ UNCHANGED <<rLight, gLight>>
Next == Red \/ Green \/ Tnext
System == Init /\ [][Next]_<<t, rLight, gLight>> /\ SF_<<t, rLight, gLight>>(Red)
         /\ SF_<<t, rLight, gLight>>(Red) /\ SF_<<t>>(TNext)
```

Figure 3. The description of system

V. THE SPECIFICATION AND VERIFICATION OF PROPERTY

A. *Mutually Exclusive*

As for the system, the most important property is the mutually exclusive. The values of the variables gLight and rLight must not be 1 at the same time:

Mutex == ∨ ∧ rLight = 0
          ∧ gLight = 1
        ∨ ∧ rLight = 1
          ∧ gLight = 0

We let it as a invariant and declare it in the configuration file "cfg" of TLC. And then, if there will be arbitrary state which violates the invariant, the TLC will give the error track.

B. *state persistence*

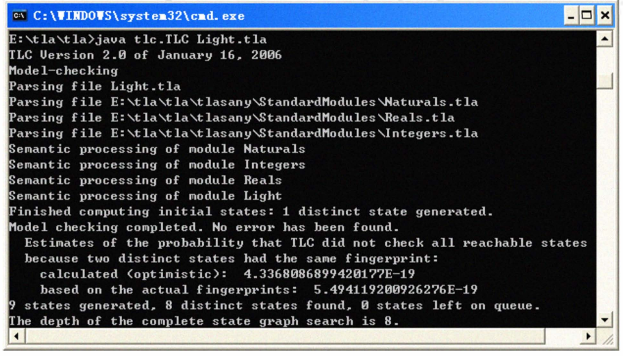
The system requires that every state must sustain three time units. When the value of rLight is 1, it means that the system is on the state "red". So when its value is 1, The timer will increase to 3 eventually:

RedLast == (rLight = 1) => (t = 3)

Similarly, there is the same property as the "green" state:

GreenLast == (gLight = 1) => (t = 3)

We use the model checking TLC which is based on the TLA to verify the property above and there is not erro. It means that the system satisfies the property above.



```
E:\tla\tla>java tlc.TLC Light.tla
TLC Version 2.0 of January 16, 2006
Model-checking
Parsing file Light.tla
Parsing file E:\tla\tla\tlasany\StandardModules\Naturals.tla
Parsing file E:\tla\tla\tlasany\StandardModules\Reals.tla
Parsing file E:\tla\tla\tlasany\StandardModules\Integers.tla
Semantic processing of module Naturals
Semantic processing of module Integers
Semantic processing of module Reals
Semantic processing of module Light
Finished computing initial states: 1 distinct state generated.
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
    calculated (optimistic):  4.336808689942017E-19
    based on the actual fingerprints:  5.494119200926276E-19
9 states generated, 8 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 8.
```

Figure 4. The checking result

VI. CONCLUSION

Real-time system is a class of safety-critical system, any error may bring significant economic losses, environmental damage and even a threat to life. So formal methods must be used to ensure its security fundamentally. But different formal method has its advantages and disadvantages. We will get better Analysis and verification results if we can combine the appropriate methods.

ACKNOWLEDGMENT

This research work was financial supported by the National Natural Science Foundation of China (Grant No. 60963023).

REFERENCES

[1] Alur R, Dill DL. A theory of timed automata[J]. Theoretical Computer Science. 1994, 126(2):183-235.

[2] Leslie Lamport. The Temporal Logic of Action[J]. ACM Transactions on Programming Languages and Systems. 1994, 16(3):872-923.

[3] Li Guang-yuan. LTLC: A Continuous-time Temporal Logic for Real-time and Hybrid Systems[D]. Doctoral Dissertation of the Chinese Academy of Sciences. 2001, 5:35-67.

[4] Yan Rong-jie, Li Guang-yuan, Xu Yu-bo, Liu Chun-ming, Tang Zhi-song. Reachabilit Checking of Finite Precision Timed Automata[J]. Journal of Software. 2006, 17(1):1-10.

[5] Leslie Lamport. Specifying Systems[M]. Addi-son-Wesley Longman Publishing Co., Inc. 2002.

[6] Wan Liang. The Research of System, Rules, and Checking of Protocol Based on TLA[D]. Doctoral Dissertation of the Guizhou University. 2009, 5:7-24.

[7] Fang Min, Zhang Ya-shun, Li Hui. Formal Verification of Hybrid Systems[J]. Journal of System Simulation. 2006, 18(10):2921-2924.

[8] Leslie Lamport. Hybrid Systems in TLA+ [J]. Lecture Notes in Computer Science. 1993, 736:77-102.