Verification of real-time properties based on Event-B models

Zhao Jinfu[1, a], Zhang Hong[2, b], Wang Xuejing[3,c]

[1,2,3]Science and Technology on Reliability and Environmental Engineering Laboratory

School of Reliability and Systems Engineering, Beihang University, Beijing, 100191, China

[a]email: buaazjf@qq.com, [b]email:zh@buaa.edu.cn, [c]email:crystal@qq.com

Keywords: real-time, verification, formal method, Event-B, UPPAAL

Abstract. A large number of dependable embedded systems have stringent real-time requirements, which cause real-time verification in modeling stage is desired. Event-B is a formal method used for system-level modeling and analysis, which is suitable for modeling high dependable software. Modeling real-time system in Event-B can ensure the reliability of system, while Event-B lacks real-time properties verification approaches at modeling stage. In this paper, one real-time property verification approaches for Event-B models assisted with UPPAAL is presented. Firstly, the Event-B models are transferred to UPPAAL models, and then UPPAAL checker is used to verify whether the UPPAAL models satisfied these terms.

Introduction

Real-time system are often used in safety-critical system, which decides it needs more dependability than other systems. Event-B[1] is a formal method used for system-level modeling and analysis and it is more rigorous and much more suitable for modeling high dependable system.

At this stage, Event-B can be used to model real-time system, while Event-B has defections in verifying real-time properties due to its inner mechanism. Hence, one approach is desired to verify these real-time properties in model stage as early as possible.

In this paper, we presented one approach that use one real-time validation tool UPPAAL to verify the real-time properties modeling in Event-B. The Event-B model of real-time properties are transferred to UPPAAL model, and then the UPPAAL model checker [3] is used to verify real-time properties of the models. Two basic time constraints (deadline and delay) are used to show our approach. At last, landing gear is taken as one simple example to prove the feasibility of the approach.

Modeling in Event-B

Event-B is an extension of B method[4] , which provides one formal approach for development of highly dependable software. Rodin platform provides automated tools for modeling and verification in Event-B.

Event-B model is formed by two parts: context and machine. Each context describe the static attributes of model, context consists set(s), constant(c) and axiom p(s, c). Sets is used to define new data type, constants declare the particular elements, axioms show limits to sets and constants. Machine describes the dynamic behaviors of model and consists variable(v), invariant I(v) and event(E). Variables change with the change of events, while variables must meet the limit of invariants, events consists two parts； G (s,v,c), S(v,v’). G (s,v,c) declares the guards of events and S(v,v’) describes relations between current and next states.

While defining events, we adopt the following syntax: $E \hat{=} \text{any } v \text{ where } G(s, v, c) \text{ then } S(v, v')$, where v is local event variable, the G(s,v,c) is the guard of event and S(v,v’) presents assignments to the state variables. The occurrence of events represents the observable behavior of the system. The guard defines the conditions under which the action can be executed.

Modeling real-time properties

In this section，two kinds of basic time constraints: deadline constraint and delay constraint are taken as examples to show our approach. In this part, we will give out the definition these time constraints and their Event-B models.

A. Deadline constraint

Deadline constraint means something must occur before a particular time. As is shown in Fig.1, S0,S1 and S2 represent states, A and B are events, tS1 and tS2 represent respectively the occurrence time of event A and event B and t is the deadline time. We use “deadline(S1,S2,A,B,t)” to show that event B must occur in t time after event A. In Fig.2, we give out its Event-B model.

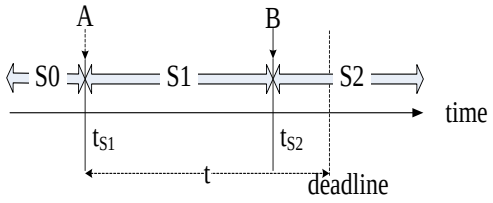


Fig. 1 Deadline constraint

variable S0,S1,S2time,tS1 **Invariants** @inv1 S0, S1, S2∈BOOL @inv2 time ∈ ℕ @inv3 S1 = true ⇒ time < t	**Initialisation** ≙ **BEGIN** @act1 time := 0 @act2 S0 :=true END	**tick_lock** ≙ WHEN THEN @act1 time ≔ time + 1 END
event A ≙ WHEN @grd1 S0=true THEN @act1 S0:=False @act2 time:≔ 0 @act3 S1:≔true END		**event** B ≙ WHEN @grd1 S1=true @grd2 time <= t THEN @act1 S1:≔False @act2 S2:≔true END

Fig. 2 Event-B model of deadline constraint

In Fig.2, time is the clock variable, Invariants shows the limits to variables. In event A, “S0 is true” is the guard, event A make system leave S0 turn into S1 and the occurrence time is assigned the value of 0. In event B, "S1 is true" and " time<=t" are the guards. Event B makes system leave S1 turn into S2 and event tick_lock is added to model to show the time pass.

B. Delay constraint

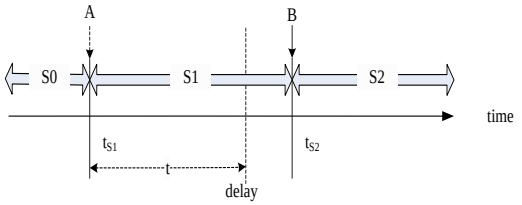


Fig. 3 Delay constraint

variable S0,S1,S2,time **Invariants** @inv1 time ∈ ℕ @inv2 S0,S1,S2 ∈BOOL	**Initialisation** ≙ BEGIN @act1 S0:=true @act2 time := 0 END	**tick_tock** ≙ WHEN THEN @act1 time:≔time+1 END
event A≙ WHEN @grd1 S0=true THEN @act1 S0:=false @act2 S1:=true @act3 time ≔ 0 END	**event B** ≙ WHEN @grd1 S1=true @grd2 time ≥t @grd3 S2=true⇒time>t THEN @act1 S1:=false @act2 S2:= true END	

Fig. 4 Event-B model of delay constraint

Delay constraint means something must occur after a particular time. As is shown in Fig. 3, we use Delay(S1, S2，A, B, t) to show that event B must occur after t time of event A and its Event-B model is given in Fig. 4.

In this section, we present three kinds of time constraints and give out their Event-B models. In the rest paper, one approach will be presented how to verify the correction of these time constraints models.

Verification of real-time properties

In this section, we will show how to use UPPAAL to verify the real-time properties of Event-B model. At first, Event-B model will be transferred to UPPAAL model and then some prove terms will be used to verify whether the Event-B models satisfy these time constraints.

UPPAAL is one real-time properties modeling and verification tool based on timed automata. Timed automata is a process model with time constraints, It is based on finite automaton charactering the system behavior via a number of states and transitions, and clocks variables are introduced to limits the behavior of system. UPPAAL can verify whether the time constraints on state and transitions are satisfied via finite states search.

A. Transferred rules

Event-B models are composed by a set of events, and events can be defined as E(G (s,v,c),

S(v,v')). UPPAAL model can be defined as $L\ (t,v) \xrightarrow{r,\phi,u} L'(t',v')$, L and L' are before-after states, t is clock variable, v is common variable, r is time constraints on transition, ϕ is the set of clock variables that need to reset and u is the actions in this transition. When Event-B model is transferred to UPPAAL model, we use M indicate Event-B model and N indicate UPPAAL model, so we have following rules:

(1) Sets, constants and variables in M must be declared in N, the integer type of time in M transferred to clock type in N and the others keep the type unchanged.

(2) Time constraints in G(s,v,c) should be transferred to r in N and states constraints in G(s,v,c) should be transferred to the constraints on L.

(3) Actions of resetting time variables in S(v,v') should be transferred to ϕ in N, and other actions in S(v,v') should be transferred to u.

(4) Adjusting the transferred model according to the actual situation and ensure the consistence between M and N.

In the rest paper, two basic time constraints will be transferred UPPAAL model, then some prove terms are presented to verify whether these UPPAAL models satisfy the requirements.

B. Verification of Deadline constraint

Event-B model of deadline constraint is shown in Fig.2. Here, we transfer Event-B model of deadline to UPPAAL model, the variables and constant are declared in UPPAAL: clock time, const int t=10; S0, S1 and S2 are state and S0 is the initial state; event A and event B are the transition, their guards as the guards of transitions and their actions as transition actions. Hence, we get the UPPAAL model of deadline constraint, as is shown in Fig.5.

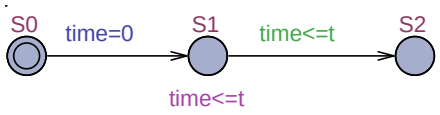


Fig. 5 UPPAAL model of deadline constraint

```
Deadline.S1 -->Deadline.S2 imply time<=t
Verification/kernel/elapsed time used: 0s / 0s / 0.01s.
Resident/virtual memory usage peaks: 7,380KB / 26,636KB.
Property is satisfied.
A<> time>=t imply not Deadline.S2
Verification/kernel/elapsed time used: 0s / 0s / 0.003s.
Resident/virtual memory usage peaks: 7,948KB / 27,664KB.
Property is satisfied.
```

Fig. 6 Verification results of Deadline constraint

In Fig.5, S0 is the initial state; event A change to the transition from S0 to S1 and time is assigned value 0 in this transition. "time<=t" means the time of system stay in S1 must less than t. Event B change to the transition from S1 to S2 and "time<=t" is its guard.

To verify deadline constraint, some terms must be proved: **a**）transition from S1 to S2 occurs just when time<=t, the prove term is: Deadline.S1 -->Deadline.S2 imply time<=t; **b)** for any time>t, system cannot in Deadline.S1, the prove language is: A<> time>t imply not Deadline.S1. Fig.6 shows that UPPAAL model satisfies deadline constraint.

C. Verification of Delay constraint

The Event-B model of delay constraint is shown in Fig.4 and its UPPAAL model is shown in Fig.7. Event A and event B are the transition, their guards as the guards of transitions and their actions as transition actions.

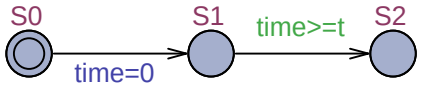


Fig. 7 UPPAAL model of delay constraint

```
Delay.S1--> Delay.S2 imply time>=t
Verification/kernel/elapsed time used: 0s / 0s / 0.002s.
Resident/virtual memory usage peaks: 7,812KB / 27,116KB.
Property is satisfied.
A<>( time<t &&Delay.S1) imply not Delay.S2
Verification/kernel/elapsed time used: 0s / 0s / 0.002s.
Resident/virtual memory usage peaks: 7,812KB / 27,116KB.
Property is satisfied.
```

Fig. 8 Verification results of delay constraint

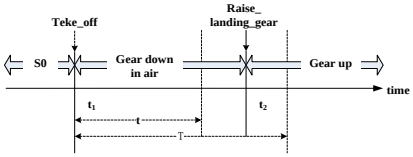
To verify delay constraint, some terms must be proved: **a**） when transition from S1 to S2 occurs, time>=t, the prove term is: Delay.S1--> Delay.S2 imply time>=t; **b)** when time<t and system in S2, no the transition from S1 to S2, the prove language is: A<>(time<t &&Delay.S1) imply not Delay.S2. Fig.8 shows the UPPAAL model satisfies delay constraint.

Case Study

In this section, we will use Event-B to model the real-time properties of landing gear, and then

use our approach to verify the corrections of modeled real-time properties.

Landing gear is one very important part of plane for either take-off or landing. After the plane taking off, the landing gear can be packed up automatically, while it should be neither too early for emergency landing, nor too late for flight safety. Hence, we must control the gear packs up in one interval time, here we define the interval time is [t, T] (t<=T), and we call this time constraint: interval delay, just as is shown in Fig. 9.



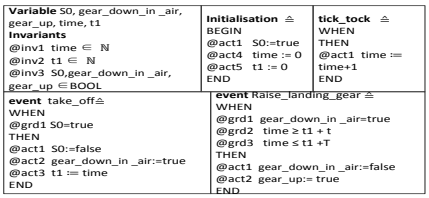
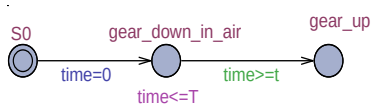


Fig. 9 Time constraints of raising landing gear Fig. 10 Event-B model of raising landing gear

Here S0 is the state of plane on the ground, “Gear down in air” and “Gear up” are two sates of landing gear, t1 is the time of the plane takes off, t2 is the time of raising the landing gear. We can use “Interval delay (Gear_down_in _air, Gear_up, Take_off, Raise_landing_gear, t, T)” to show the time constraint, which means event “Raise_landing_gear” must occur in the time[t,T] after event “take_off”. In Fig. 10, we give out the Event-B model of raising landing gear.

In fact, we can see “Interval delay (Gear_down_in_air, Gear_up, Take_off, Raise_landing_gear, t, T)” as the combination of “Delay(Gear_down_in_air, Gear_up, Take_off, Raise_landing_gear, t)” and “Deadline (Gear_ down_in_air, Gear_up, Take_off, Raise_landing_gear, T)”. And then we transfer this model to UPPAAL model and the transferred UPPAAL model is shown in Fig. 11.



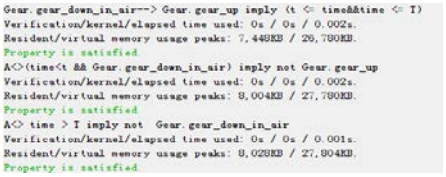


Fig. 11 UPPAAL model of landing gear Fig. 12 verification result of landing gear model

The prove terms is the combination of delay prove terms and deadline prove terms, and we get the interval delay prove terms: **a)** Gear.gear_down_in_air--> Gear.gear_up imply (t <= time&&time <= T); **b)** A<>(time<t && Gear.gear_down_in_air) imply not Gear.gear_up; **c**）A<> time > T imply not Gear.gear_down_in_air. The prove result is shown in Fig. 12.

Conclusion

This paper carries out one approach of using UPPAAL to verify real-time properties of Event-B models. For verifying the real-time properties, Event-B models are transferred to UPPAAL model and some terms are presented to verify these time constraints by using UPPAAL. At last, one simple example of landing gear is used to prove the feasible of our approach.

Reference

[1].C.Metayer, J.Abrial and L.Voisin. Rodin Deliverable D7:Event-B language. Project IST-511599, School of Computing Science, Newcastle University, 2005 Event-B [M].

[2].Rodin[M],2012:http://en.wikipedia.org/wiki/Rodin.

[3].Gerd Behrmann, Alexandre David, Kim G. Larsen. A Tutorial on Uppaal. Department of Computer Science, Aalborg University, Denmark, 2004.

[4].J.R.Abrial , The B-Book: Assigning Programs to Meanings. Cambridge University Press, 2005.

[5].J.-R.Abrial, Modeling in Event- B. Cambridge University Press, 2010.

[6].S.Mohammad Reza, B.Michael. Specification and refinement of discrete timing properties in Event-B[J]. Automated Verification of Critical Systems 2011. 2011. 46.

[7].Alexe Ilasov, Linas Labins. Supporting Reuse in Event B Development: Modularisation ApproachAbstract State Machines， Alloy， B and Z[J]. 2010. 5977: 174-188