Study on Fairness Property and Liveness Property of Temporal Logic of Action

Bai Jinshan[1] , Liu Chunying[1] , Li Xiang[1]

[1] Institute of Computer Software and Theory
Guizhou University
Guiyang 550025, China
E-mail:qingqingtao1986@163.com

Abstract— The paper introduces about temporal logic of action, it discuss the relation among closure, security, fairness property and liveness property in TLA, action and point out and weak fairness and strong fairness as prerequisites, base on this, it discrible in detail how to establish system model with machine-closed through cases, in the end we gain correct conclusion.

Keywords— The Temporal Logic of Action; fairness property;Liveness property; Machine-closed

1. Introduction

In the processing of putting up Model Checking using temporal logic of action TLA, we must introduce fairness in Model Checking for ensuring the checking can put up in the needed route. Therefore, it is an important question that which kinds of fairness will introduce and what condition will be satisfied on the fairness that was introduced. In this paper, we will study in two aspects of theory and practice.

2. Temporal logic of action Semantics

Lamport established the temporal logic of action TLA based on temporal logic for describing efficiency transfer relation of the states. TLA introduces primed variable operator, if it contains variable x in currently state, let x' represent its value in thereafter state.

An action of TLA is a Boolean_variabled expression formed from variables and primed variables.

Assume s and t is state, V is the sets of state variable, v' is the subsequence sets of primed variable relative to v, A is an action:

A(s[[v]]/v) denotes the values obtained from action A by replacing each unprimed variable v by the analog of state s.

A(t[[v]]/v') denotes the values obtained from action A by replacing each primed variable v' by the analog of state t.

[[A]](s,t) $\equiv$ A(s[[v]]/v, t[[v]]/v') equals true iff executing an action A in state s can produce state t, where s and t is a step. Denote for (s, A, t).

$[A]_v \equiv A \vee (v' = v)$;

$\langle A \rangle_v \equiv A \wedge (v' \neq v)$.

Definition 1: temporal proposition logic of action module[1] is a Kripke framework $< S , I , \Sigma , R , V >$:

S : S is a finite set of states.

V: Atomic(LT)$\times$S$\rightarrow${0,1} is a function that labels each state with the set of atomic propositions true in that state.

I: I is the set of initial states.

Σ: $\Sigma$is the set of actions.

R: R is a transition R $\subseteq$S$\times\Sigma\times$S.

An action A$\in\Sigma$ is enabled at state s$\in$S iff (s, A, t)$\in$R for some t$\in$S. The predicate Enabled A for a state is defined by

[[Enabled A]](s) $\equiv$ $\exists$t$\in$S: [[A]](s,t)

The set Σ of actions contains a special "suttering" action τ with (q, τ, q')$\in$R iff q'= q.

A behavior of σ is an infinite sequence

$$\sigma = s_0 \xrightarrow{A_0} s_1 \xrightarrow{A_1} \cdots$$

where $s_0 \in I$, $(s_i, A_i , s_{i+1}) \in R$, $i \in N$. This formula can denote for $\sigma = s_0, s_1, \ldots$.

The meaning [[π]] is a Boolean_Valued function on behaviors. If π is a state function is a nonboolean expression containing unprimed variable, then [[π]](σ) is true iff it is true in the first state of the behavior σ. If π is a action containing primed variables, then [[π]](σ) is true iff the first two states is a π step. Boolean operator on formulas are defined in the obvious way, for example $[[\pi \wedge \Phi]] = [[\pi]] \wedge [[\Phi]]$.

$\Box\pi$ is the formula that is satified by a behavior σ iff

978-0-7695-3635-4/09 $25.00 © 2009 IEEE
DOI 10.1109/ICNDS.2009.37

every suffix of σ satisfies π. If π is an action , $\Box\pi(\sigma)$ is true of a behavior σ iff π is true in every state of the behavior σ.

$\Diamond\pi$ is the formula that is satified by a behavior σ iff some suffix of σ satisfies π.

The canonical form of a TLA formula is $Init\wedge\Box[N]v$, where Init is a state predicate, N an action , v a state function.

3. Liveness and Fairness

Fairness conditions specify that some action must happen, provided they are "sufficiently often" enabled. They place additional "global" constraints on runs of transition systems. The Fairness conditions can be expressed in term of weak fairness and strong fairness.

Weak fairness[3]: a run of state sequence is weakly fair on an action A iff if A is enabled at all states beyond m then An =A for some n≥m. The formula is expressed as follows.

$WF_v(A) \equiv \Box\Diamond<A>_v\vee\Box\Diamond\neg Enabled<A>_v$

Strong fairness[3]: a run of state sequence is strongly fair on an action A iff if A is enabled at infinitely many states beyond m then An =A for some n≥m. The formula is expressed as follows.

$SF_v(A) \equiv \Box\Diamond<A>_v\vee\Diamond\Box\neg Enabled<A>_v$

String fairness implies weak fairness[1]. $SF_v(A)\Rightarrow WF_v(A)$.

Weak and strong fairness are equivalent for an action that, once enabled, remains enabled until it is executed. Strong fairness can be more difficult to implement than weak fairness. When strong fairness and weak fairness are equivalent, the fairness property should be written as weak fairness. A system specification is usually of the form $Init\wedge\Box[N]_v\wedge F$, where F is the conjunction of formulas of the form $SF_v(A)$ or $WF_v(A)$.

Liveness property: A property Φ is a liveness property iff any finite sequence of system can be extended to some infinite sequence satisfying Φ.

4. Closure and Safety

Let $\rho\subseteq\sigma$ mean that ρ is a finite prefix of the behavior σ. Let "." be concatenation of sequences. The Closure C(π) of a formula π is defined by

$[[C(\pi)]](\sigma)\equiv\forall\rho\subseteq\sigma:\exists\beta\in S^{\infty}:[[\pi]](\rho.\beta)$.

A behavior σ satisfies C(π) iff every finite prefix of σ can be extended to a behavior that satifies π.

A Safety formula is one that equals its closure. Thus, a safety formula π is satisfied by a behavior σ iff every prefix of σ can be extended to a behavior satisfying π. The set of runs of a transition system with a total transition relation, but without fairness conditions, is a safety property[3].

Let S^{∞} be the set of all behaviors, Let S* be the set of all finite behaviors.

Proposition 1[4] If Init is a state predicate, F is the conjunction of countably many formulas of the form $WF_v(A)$ and $SF_v(A)$, where each $<A>_v$ implies N, then

$C(Init\wedge\Box[N]_v\wedge F)\equiv Init\wedge\Box[N]_v$

W/SF denotes either WF or SF. Assume:

1. $|=<A>_v\Rightarrow N$
2. $\sigma\in S^{\infty}$

Prove: In the following proof, F consists of a single WF or SF formula.

Lemma 1 $(\forall\rho\subseteq\sigma:\exists\beta\in S^{\infty}:[[Init\wedge\Box[N]_v\wedge W/SF_v(A)]](\rho.\beta))\Rightarrow[[Init\wedge\Box[N]_v]](\sigma)$

Assume 1 implies that Init holds in the first state of σ and $[N]_v$ holds in every pair of successive states of σ, which implies $[[Init\wedge\Box[N]_v]](\sigma)$ by definition of □ and of [[B]] for an action B.

Lemma 2 $[[Init\wedge\Box[N]_v]](\sigma)\Rightarrow\forall\rho\subseteq\sigma:\exists\beta\in S^{\infty}:[[Init\wedge\Box[N]_v\wedge W/SF_v(A)]](\rho.\beta)$

2.1 Choose states $s_0, s_1, \ldots$ such that $\rho=s_0,\ldots,s_n$, $\forall i\geq n$:

$\wedge\ ([[Enabled<A>_v]](s_i)\Rightarrow[[<A>_v]](s_i, s_{i+1}))$

$\wedge\ (\neg[[Enabled<A>_v]](s_i)\Rightarrow(s_{i+1}=s_i))$

The existence of the si follows from the definitions of Enabled and "suttering" action. The proof in after is put up in this route.

2.2 $[[W/SF_v(A)]](s_0,s_1,\ldots)$

(1) $\Box\Diamond Enabled<A>_v$

$\Rightarrow\ \Box\Diamond([[Enabled<A>_v]](s_i))$

$\Rightarrow\ \Box\Diamond([[<A>_v]](s_i,s_{i+1}))$ (2.1)

$\Rightarrow\ (\Box\Diamond[[<A>_v]])(s_0,s_1,\ldots)$

$\Rightarrow\ [[W/SF_v(A)]](s_0,s_1,\ldots)$

(2) $\neg\Box\Diamond Enabled<A>_v$

$\Rightarrow\ \Diamond\Box\neg Enabled<A>_v$

$\Rightarrow\ [[SF_v(A)]](s_0,s_1,\ldots)$

$\Rightarrow\ [[WF_v(A)]](s_0,s_1,\ldots)$

By (1), (2): 2.2 holds.

2.3 $[[\Box[N]_v]](s_0,s_1,\ldots)$

Assume i<n: We can see that $[[[N]_v]](s_i,s_{i+1})$ come into existence in according to Lemma 2's premise.

Assume i≥n:

(1) $[[Enabled<A>_v]](s_i)$

⇒ $[[<A>_v]](s_i, s_{i+1})$ (2.1)

⇒ $[[N]](s_i, s_{i+1})$ (Assume 1)

⇒ $[[[N]_v]](s_i, s_{i+1})$

(2) ¬ $[[Enabled<A>_v]](s_i)$

⇒ $s_{i+1}=s_i$ (2.1)

⇒ $[[[N]_v]](s_i, s_{i+1})$ ("suttering" action)

By (1), (2): 2.3 holds.

2.4 $[[Init]](\sigma) \Rightarrow [[Init]](\rho) \Rightarrow [[Init]](s_0,s_1,\ldots)$

Let $\beta= s_i, s_{i+1},\ldots$, by 2.2, 2.3, 2.4, lemma 2 holds.

By lemma 1, lemma 2, and the definition of Cosure, Proposition 1 holds.

Q.E.D.

So, any finite sequence of a System can be extended to some sequence satisfying a fairness property.

5. Machine closed

Let S be a safety property and L be any property. The pair (S, L) is machine closed iff C(S∧L)=S. If (S, L) is a system specification then it should be machine closed. An action A is a subaction of Next iff every A step is a Next step. Equivalently, A is a subaction of Next iff A implies Next. We should express liveness with fairness properties for subactions of Next action. Following take a simple operation of the railway system as an example, show how to set up approprite fairness propriety and establish the system model with machine closed.

In figure 1, train W run clockwise in the inner railway, train E run count-clockwise in out railway, the two railways have to go through a single-track railway bridge, on both sides of the bridge has been installed with 2 traffic lights, lamps in the initial state are set to red light. We want that the two trains will be able to go through the railway bridge unlimited and not a collision.

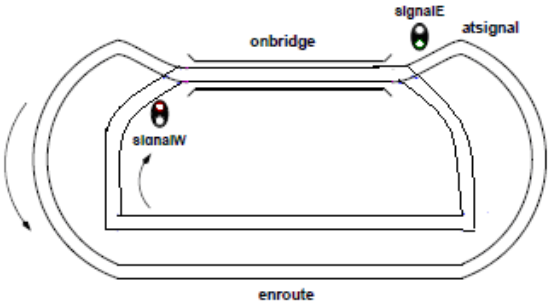

Fig1. railway system

Railways will be divided into three parts corresponding to each of the train of three actions to be able to show the motion location of train. "onbridge" denotes the train on the bridge, "enroute" denotes the train has gone through the bridge, "atsignal" denotes will be going to the bridge reach the location of the traffic lights.

The enabled condition of the action onbridge is that the run location of train is "atsignal", and opposite to the red light, the enabled condition is likely to lose lead to the action can not be executed by reason of the red light of opposite changed green light , so action onbridge should be set up strong fairness. When the action onbridge is executed, should be one's own set of green lights to prevent the train of opposite across the bridge.

The enabled condition of the action enroute is that the run location of train is "onbridge", the enabled condition will finish until the action enroute is executed, so the action onbridge should be set up weak fairness. By the same taken, the action atsignl should be set up weak fairness.

According to the above thinking, establishment of the TLA model checking of the railway system is as follows:

```
MODULE Traffic
VARIABLES TW, TE
Type≡/\ TW∈[pos:{"E","A","O"}, sig:{"R","G"}]
      /\ TE∈[pos:{"E","A","O"}, sig:{"R","G"}]
Init  ≡/\ Type /\ TW.pos ="E" /\ TW.sig ="R"
      /\ TE.pos ="E" /\ TE.sig ="R"
AtSignal(t,other)≡/\ t.pos ="E"
                  /\ t' =[t EXCEPT!.pos ="A"]
                  /\ UNCHANGED <other>
OnBridge (t,other) ≡ /\ t.pos ="A"/\other.sig ="R"
                     /\ t' =[pos→"O", sig→"G"]
```

$\wedge$ UNCHANGED <other>

Enroute (t,other) $\equiv$ $\wedge$ t.pos ="O"

$\wedge$ t' =[pos→"E", sig→"R"]

$\wedge$ UNCHANGED <other>

Next $\equiv$ $\vee$ AtSingnal(TW,TE) $\vee$ OnBridge(TW,TE)

$\vee$ Enroute(TW,TE) $\vee$ AtSingnal(TE,TW)

$\vee$ OnBridge (TE, TW) $\vee$ Enroute (TE, TW)

Vars $\equiv$ <TW, TE>

property1$\equiv$[]<>(TW.pos ="O")/\[]<>(TE.pos ="O")

* trains can go through the bridge infinitely many *

property2$\equiv$ [](¬((TW.pos ="O") $\wedge$ (TE.pos ="O")))

* the two train can not collision *

WF $\equiv$ $\wedge$ WF_{Vars}(Enroute (TE,TW))

$\wedge$ WF_{Vars}(Enroute(TW,TE))

$\wedge$ WF_{Vars}(AtSingnal(TE,TW))

$\wedge$ WF_{Vars} (AtSingnal(TW,TE))

SF $\equiv$ $\wedge$ SF_{Vars}(OnBridge(TE,TW))

$\wedge$ SF_{Vars}(OnBridge (TW,TE))

Traffic $\equiv$ Init $\wedge$ [][Next]$_{Vars}$ $\wedge$ WF $\wedge$ SF

The results of checking this system model use of TLC shown in Figure 2 to be the right conclusion. TLC produced 8 different states, Figure 3 describes system transfer of statechart diagram by consisting of 8 states.

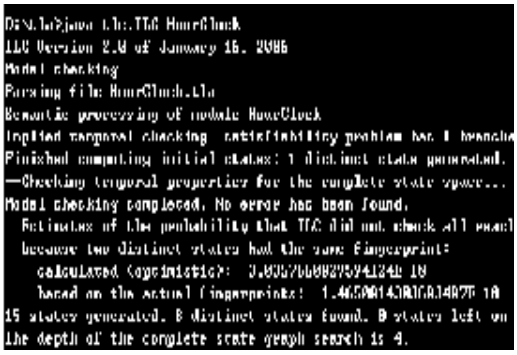

Fig2. checking of the railway system

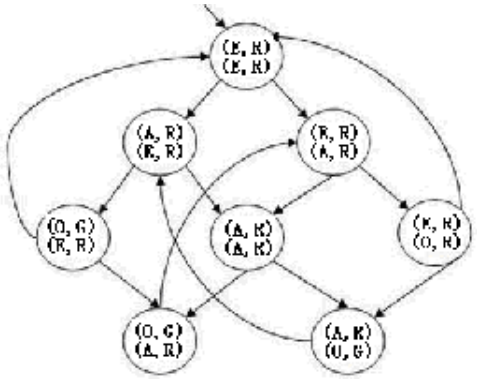

Fig3. statechart diagram of railway system

6. Conclusion

In the processing of model checking for the system, we need limiting the fairness of system in order to avoid the behavior happening on the not related paths and make ensure the system have the kind of activity. The system with the fairness must be a machine closed, make sure the action with fairness contain transfer action of system. We have to choose appropriate limitation of fairness for corresponding action, in order to establish efficiency system model.

REFERENCES

[1]Leslie Lamport. The Temporal Logic of Actions[M]. ACM Transactions on Programming Languages and Systems.1994.5,16(3):872-923

[2]Leslie Lamport. Specifying Systems[M]. Addison-Wesley Longman Publishing Co., Inc. 2002.

[3] Stephan Eerz. Modeling and Developing Systems Using TLA+[M]. Escuela de Verano,2005

[4] Leslie Lamport. Proving possibility properties. Theoretical Computer Science,206(1−2):341−352, October 1998.

[5] Alpern, B., and F.B. Schneider. Defining liveness [J]. Information Processing Letters,1985,

21(4):181-185.

[6] Alpern, B., and Schneider, F.B. Verifying temporal properties without using temporal logic[M]. Dept. of Computer Science,Cornell Univ. 1985.

[7] Jeng M D, Diceare F. A review of synthesis techniques for Petri nets with application to automated manufacturing systems[J]. IEEE Trans. on Systems, Man, and Cybernetics, 1993, 23(1): 301-312.

[8]Jeng M D. A Petri net synthesis theory for modeling flexiblemanufacturing systems[J]. IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, 1997, 27(2): 169-183.

[9] Aybar A, Ifar A. Overlapping decompositions and expansions of Petri nets IEEE Trans[J]. on Automatic Con~ol, 2002, 47(3): 5l1～515.