Reasoning About Real-Time Systems in Event-B Models with Fairness Assumptions

Chenyang Zhu
Computer Science and Artificial Intelligence
Changzhou University
Changzhou, China
zcy@cczu.edu.cn

Michael Butler
Electronics and Computer Science
University of Southampton
Southampton, UK
mjb@ecs.soton.ac.uk

Corina Cirstea
Electronics and Computer Science
University of Southampton
Southampton, UK
cc2@ecs.soton.ac.uk

Thai Son Hoang
Electronics and Computer Science
University of Southampton
Southampton, UK
T.S.Hoang@soton.ac.uk

***Abstract*—Stepwise-based development supported by the Event-B formalism has been used in the domain of system design and verification. This refinement approach guarantees that safety properties are preserved, while additional reasoning is required to prove liveness properties. Our previous work proposes to use real-time trigger-response properties to reason about liveness properties and timed properties in real-time systems. Conditions such as weak fairness assumptions, relative deadlock freedom, and conditional convergence are explored to eliminate Zeno behavior when modeling real-time systems. In this reasoning framework, the response events are required not to be disabled by other events. This paper extends our previous results by using strong fairness assumptions to relax the constraints on response events. Refinement rules and strategies are also developed to refine real-time systems in the form of trace semantics and LTL operators with fairness assumptions. A simplified bounded re-transmission protocol case study is used to illustrate the approach.**

***Index Terms*—Event-B, Formal Specification, Real-time Systems, Fairness Assumptions, Refinement**

I. Introduction

In recent years, there has been growing recognition of the vital links between Cyber-Physical Systems (CPS) and time-dependent functionality. Timing properties should be specified when developing the system to guarantee that CPS is interacting with the environment correctly. Some safety-critical systems must satisfy certain real-time constraints in order to function as intended. For example, an artificial pacemaker should deliver therapy according to the misbehavior of the heart at fixed time intervals. Missing the stimulus deadlines could be catastrophic. Similarly, the flight control software must be able to distinguish between different flying environments and respond to different situations within a specific timeframe in order to control the aircraft safely. Thus CPS must fulfill its functional role in a time-regulated manner in order to avoid unsafe situations.

With the thriving growth of software technology, much attention has been concentrated on simplifying the design of software systems while maintaining their usability and dependability. However, the number of system errors increases in proportion to the scale and complexity of the whole system. Formal methods, which include mathematical techniques for specifying and verifying systems, help the developers to construct reliable software systems despite the design complexity [1]. As one of the formal methods, the Event-B formalism offers a stepwise development to manage complexity in system design and verification.

Safety and liveness are two key properties of formal models: a safety property states that dangerous situations will not arise while a liveness property ensures that something good will eventually happen [2]. In real-time systems, time should progress regardless of what happens in its environment [3]. Thus liveness properties should be reasoned together with the real-time systems. However, Event-B models are not concerned with fairness or scheduling specifications [4]. Our previous work in [5] proposed a reasoning framework to treat the timed properties and liveness properties formally in Event-B models. We explored sufficient conditions under which all the traces of an Event-B model satisfy the real-time trigger-response properties, in the form of Event-B proof obligations and weak fairness assumptions. In this reasoning framework, the response events are required not to be disabled by all the other events. Then weak fairness assumptions and conditional convergent intermediate events would lead to the eventually occurrence of response events. This paper extends the previous results by allowing response events to be disabled by some other events. Strong fairness assumptions on response events could still lead to the eventually occurrence of response events. Our previous work in [6], [7] also explored weak fairness assumptions and other conditions to refine the discrete timed models. In this paper, we extend the work by including the fairness assumptions in the refinement step. Refinement rules and strategies are also developed to refine real-time systems in the form of trace semantics and LTL operators with fairness assumptions. A simplified bounded re-transmission protocol (BRP) case study is used to depict the approach.

This paper is structured as follows. Section II describes the related work that manage timed properties, LTL properties in Event-B refinement. Section III presents the definition of trace semantics as well as LTL operators to be used in this paper. Section IV delivers rules that enforcing real-time trigger-response properties in Event-B models with fairness assumptions. The abstract level of the BRP case study is used to exemplify the rules. Section V provides rules that refine real-time systems in the form of trace semantics and LTL operators with fairness assumptions. The concrete level of the BRP case study is used to depict the approach. Section VI summarizes the work and describes future work.

II. Related Work

Event-B is a formal method that is usually used for system-level modelling and analysis with refinement and reasoning on the model [8]. However, it lacks explicit support for expressing and verifying timing constraints [9]. Cansell et al. developed an action-reaction pattern to model the causal order between events. Time constraints are imposed between action and reaction events to model real-time properties [10]. Butler and Falampin proposed an approach to modeling and refining timing properties in classical B, which adds a clock variable representing the current time and an operation which advances the clock [11]. Based on this approach, Sarshogh and Sulskus added explicit support for trigger-response properties with deadline, delay, expiry and interval timing properties [9], [12]. However these developments have failed to incorporate a proper treatment of critical issues in timed systems, namely, the divergence of intermediate events and infeasible responses caused by a lack of progress or conflicting timing constraints. Liveness properties should be specified and verified together with real-time properties to guarantee the global clock should always progress.

Work has been done to incorporate liveness properties in Event-B models. Hoang and Abrial proposed to use the notion of convergence, divergence and deadlock freedom to encode some liveness properties such as lead to property, existence property, progress property and persistence property [13]. Schneider et al. proposed to integrate Event-B and LTL together with several conditions, such as the eventually convergent new events introduced in the refinement step, the final machine in the refinement chain must be deadlock-free, to guarantee that liveness properties are preserved through the refinement chains [14]. Hoang et al. extended work in [14] by covering liveness in the context of convergent events and relaxing constraints between adjacent refinement levels by using strong deadlock freedom with new events. However, the proof rules in [13]–[15] have not yet considered fairness assumptions in the verification step and the refinement step. There are also work that integrate Event-B with other formal methods for the verification of liveness properties. Hudon and Hoang proposed Unit-B that integrate UNITY [16] and Event-B to verify both safety and liveness properties through stepwise refinement [17]. They illustrated four approaches for refining events with scheduling information and preserve the liveness properties. Abadi and Lamport defined the semantics of Temporal Logic of Actions (TLA), which uses fairness properties to handle the requirements of liveness for the digital clock in real-time systems [18]. Based on this, Méry and Poppleton proposed to integrate Event-B and TLA to include fairness assumptions in the refinement proofs to preserve liveness properties [4].

III. Trace Semantics and LTL Operators

An Event-B model that uses mathematical theory to describe a discrete transition system usually contains two kinds of components: $contexts$ and $machines$. $Contexts$ specify the static part of a model whereas $machines$ specify behavioral properties of Event-B models [19]. Our previous work abstracted away from the concrete syntax of Event-B and treat a machine as a form of labeled transition system [5]. In this paper, we extend the work by introducing fairness assumptions for the events in the machine. The definition of a machine is formalized in Definition 1. In the definition, the machine consists of state set S and initial states $init \subseteq S$. We use E to denote a set of event labels of the machine M. The transition relation K is used as a function from event label set E to the relation between states. As shown in Equation (1a), an event labelled e is enabled in state s if s is in the domain of event e's transition relation K_e. We use Equation (1b) to show that set of events is enabled when some event within this set is enabled.

Definition 1 (Machine [5])**.** *A machine M is a tuple $< S, init, E, K >$ consisting of*

- *S: a set of states, each state s is a mapping of variables of M to their values;*
- *$init$: a set of initial states $init \subseteq S \wedge init \neq \varnothing$, which correspond to initial configurations;*
- *E: a set of event labels of the machine M;*
- *K: a transition relation $K \in E \rightarrow (S \leftrightarrow S)$ that relates pairs of states;*

$$en(e, s) \triangleq s \in dom(K_e) \tag{1a}$$

$$en(A, s) \triangleq \exists e \cdot e \in A \wedge en(e, s) \tag{1b}$$

Work in [6] also defined $traces(M)$ of machine M to describe the infinite behavior of the system in terms of the set of pairs of state trace and event trace (u_s, u_e), which could be presented as an infinite sequences of alternating states and events of the form $< (u_s(0), u_e(0)), (u_s(1), u_e(1)), ... >$ in Equation (2).

$$\begin{aligned} traces(M) \triangleq \{(u_s, u_e) \mid u_s \in \mathbb{N} \rightarrow S \wedge u_e \in \mathbb{N} \rightarrow E \wedge u_s(0) \\ \in init \wedge \forall i \cdot i \geq 0 \Rightarrow u_s(i) \mapsto u_s(i+1) \in K(u_e(i))\} \end{aligned} \tag{2}$$

Temporal logic is a formalism introduced by Pnueli to express formal requirements and Linear Temporal Logic (LTL) is one of the most used temporal logics that model time as a sequence of states [20]. In this paper, we extend the usage of LTL operators presented in [21] to show the properties of the

pairs of state trace and event trace (u_s, u_e). The property φ is a Boolean expression over the trace of (u_s, u_e), which is shown in Equation (3). A machine satisfies the property φ if $\forall (u_s, u_e) \in traces(M) \Rightarrow (u_s, u_e) \models \varphi$. We use Definition 2 to show $(u_s, u_e) \models \varphi$ by induction over φ.

$$\varphi ::= true \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \tag{3}$$

Definition 2. *Given the pair of state trace and event trace $(u_s, u_e) =< (u_s(0), u_e(0)), (u_s(1), u_e(1)), ... >$, the position $i \geq 0$, and the property φ, $(u_s, u_e) \models \varphi$, meaning that the trace (u_s, u_e) satisfies the trace property φ at position i, is defined inductively by the following rules:*

$$(u_s, u_e) \models \varphi \Leftrightarrow (u_s, u_e, 0) \models \varphi$$
$$(u_s, u_e, i) \models \varphi \Leftrightarrow (u_s(i), u_e(i)) \models \varphi$$
$$(u_s, u_e, i) \models \bigcirc \varphi \Leftrightarrow (u_s(i), u_e(i), i+1) \models \varphi$$
$$(u_s, u_e, i) \models \neg\varphi \Leftrightarrow (u_s, u_e, i) \not\models \varphi$$
$$(u_s, u_e, i) \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (u_s, u_e, i) \models \varphi_1 \wedge (u_s, u_e, i) \models \varphi_2$$
$$(u_s, u_e, i) \models \varphi_1 \vee \varphi_2 \Leftrightarrow (u_s, u_e, i) \models \varphi_1 \vee (u_s, u_e, i) \models \varphi_2$$

Here we use $(u_s, u_e)_i$ to denote the suffix of (u_s, u_e) without the first i elements of the trace of pairs. For example, $(u_s, u_e) =< (u_s(0), u_e(0)), ..., (u_s(n-1), u_e(n-1)) >^\frown (u_s, u_e)^n$. And we use $(u_s, u_e)^j$ to denote the prefix of (u_s, u_e) of the firth j elements of the trace. For example, $(u_s, u_e)^j =< (u_s(0), u_e(0)), ..., (u_s(j-1), u_e(j-1))$. Based on Definition 2, we present the temporal operators $always$, denoted $\Box$, $eventually$, denoted $\Diamond$ and $until$, denoted $\mathcal{U}$ as follows:

$$(u_s, u_e) \models \Box\varphi \Leftrightarrow \forall i \geq 0 \cdot (u_s, u_e)_i \models \varphi$$
$$(u_s, u_e) \models \Diamond\varphi \Leftrightarrow \exists i \geq 0 \cdot (u_s, u_e)_i \models \varphi$$
$$(u_s, u_e) \models \varphi_1 \mathcal{U} \varphi_2 \Leftrightarrow \exists k \geq 0 \cdot \forall i < k \cdot (u_s, u_e)_i \models \varphi_1 \wedge (u_s, u_e)_k \models \varphi_2$$

The temporal operator $\Box$ expresses the requirement that property φ should always be true on the trace. The $\Diamond$ operator denotes that the property φ would be held at some point on the trace. The $\mathcal{U}$ operator takes two arguments φ_1 and φ_2, so that there exists a position k such that $(u_s, u_e)_k$ satisfies φ_2 and the sub-trace from 0 to k satisfies φ_1.

Fairness assumptions could be imposed on machines to restrict the behavior traces. We use $M \wedge Fair$ to depict the machine M with fairness assumptions $Fair$ where $Fair$ is a conjunction of strong fairness assumptions (SF) and weak fairness assumptions (WF) shown in Equation 4. Definition 3 depicts the traces properties of machine M restricted by fairness assumptions.

$$Fair = SF(L_1) \wedge WF(L_2) \quad \text{where } L_1 \subseteq E \wedge L_1 \subseteq E \tag{4}$$

Definition 3 (Strong and Weak Fairness Assumptions)**.** *Given machine M with strongly fair events L_1 and weakly fair events L_2, the traces of pair of state trace and event trace satisfy the following properties:*

$$SF(L_1) \Leftrightarrow (u_s, u_e) \models \Box\Diamond en(L_1, u_s) \Rightarrow \Box\Diamond u_e \in L_1$$
$$\equiv (u_s, u_e) \models \Diamond\Box u_e \notin L_1 \Rightarrow \Diamond\Box \neg en(L_1, u_s)$$
$$WF(L_2) \Leftrightarrow (u_s, u_e) \models \Diamond\Box en(L_2, u_s) \Rightarrow \Box\Diamond u_e \in L_2$$
$$\equiv (u_s, u_e) \models \Box\Diamond \neg en(L_2, u_s) \vee \Box\Diamond u_e \in L_2$$

The Event-B notion of convergence requires convergent events to become disabled, eventually without any fairness assumptions. Our work in [6] extended the notion of convergence to conditional convergence to define a set of events are converging under certain condition $Q(v)$ in Definition 4.

Definition 4 (Conditional Convergence [6])**.** *A group of events A is defined to be conditional convergent under the condition Q as $cov(A, Q)$ when*

- *$V(v)$ is an integer expression.*
- *When the group of events is enabled, the variant $V(v)$ is a natural number.*

$$I(v) \wedge G_A(v) \Rightarrow V(v) \in \mathbb{N}$$

- *An execution of each event $e \in A$ decreases $V(v)$ provided $Q(v)$ holds.*

$$I(v) \wedge G_e(v) \wedge S_e(v, v') \wedge Q(v) \Rightarrow V(v') < V(v)$$

Based on Definition 4, we develop Lemma 1 to show the trace properties restricted by conditional convergent events. There are two cases for the trace $(u_s, u_e) \in traces(M)$: 1) $Q(v)$ is infinitely often disabled, and 2) $Q(v)$ is eventually continuously enabled. Observe that in the first case, the event set A is not required to be convergent. Moreover, there are two cases under the condition that $Q(v)$ is eventually continuously enabled: 3) event $e \in A$ infinitely often occurs on the trace, and 4) event $e \in A$ never starts from some point on the trace. Based on Definition 4, A will eventually continuously be disabled when $Q(v)$ is enabled in case 3. Based on weak fairness on A, A should be infinitely often disabled in case 4. Otherwise, A would occur at some point on the trace.

Lemma 1. *Given that machine M is weakly fair towards A, trace $(u_s, u_e) \in traces(M)$ satisfies the conditional convergent property denoted as $(u_s, u_e) \models cov(A, Q)$, formally:*

$$(u_s, u_e) \models cov(A, Q) \Leftrightarrow \Box\Diamond(\neg en(A) \vee \neg Q)$$

Proof.

$$(u_s, u_e) \models \Box\Diamond\neg Q \vee \Diamond\Box Q$$
$$\equiv \Box\Diamond\neg Q \vee (\Diamond\Box Q \wedge (\Box\Diamond(u_e \in A) \vee \Diamond\Box(u_e \notin A)))$$
$$\equiv \Box\Diamond\neg Q \vee \begin{cases} \Diamond\Box Q \wedge \Box\Diamond(u_e \in A) \\ \Diamond\Box Q \wedge \Diamond\Box(u_e \notin A) \end{cases}$$
$$\Rightarrow \{\text{Convergent event A under condition Q}\}$$
$$\Box\Diamond\neg Q \vee \begin{cases} \Diamond\Box Q \wedge \Box\Diamond(u_e \in A \mathcal{U} \neg en(A, u_s)) \\ \Diamond\Box Q \wedge \Diamond\Box(u_e \notin A) \end{cases}$$
$$\Rightarrow \{\text{Weak Fairness on } A\}$$
$$\Box\Diamond\neg Q \vee \begin{cases} \Box\Diamond(u_e \in A \mathcal{U} \neg en(A, u_s)) \\ \Box\Diamond \neg en(A, u_s) \end{cases}$$
$$\equiv \Box\Diamond(\neg en(A) \vee \neg Q)$$

□

IV. Enforcing Real-time Trigger-Response Properties with Fairness Assumptions

To model discrete timing properties in Event-B models, our previous work in [5] provided the syntax of real-time trigger-response properties as Definition 5 based on a trigger-response pair (T, R), where $T \subseteq E$ are trigger events, $R \subseteq E$ are response events, and $T \cap R = \varnothing$. We use $Resp(T, R, w, d)$ to denote the timing property between trigger events T and response events R, where w is the delay constraint and d is the deadline constraint.

Definition 5 (Real-Time Trigger-Response Property [5])**.** *A real-time trigger-response property $Resp(T, R, w, d)$ of a machine M with event labels E consists of:*

- *trigger events $T \subseteq E \wedge T \neq \varnothing$;*
- *response events $R \subseteq E \wedge R \neq \varnothing \wedge T \cap R = \varnothing$;*
- *a delay $w \in \mathbb{N}$ and a deadline $d \in \mathbb{N}$.*

Figure 1 shows the Event-B syntax that encodes real-time trigger-response properties. The response event $response$ must occur within time d of trigger event $trigger$ occurring and can only occur if the delay period has passed. We use tT to refer to the time that a trigger event happens, and we use tR to refer to the time that a response event happens. $Ga(v)$, Act_a and $Gb(v)$, Act_b are the guards and actions of untimed trigger and response events respectively. Invariant $@inv1$ and $@inv2$ specify the delay and deadline timing property between $trigger$ and $response$ respectively. Guard $@grd3$ of the $response$ event guarantees that the response is disabled when the global clock has not passed the delay period thus preserving $@inv1$. Guard $@grd1$ of the $Tick$ event constrains the global clock not to tick when the response event is missing its deadline thus preserving $@inv2$. $@inv3$ is needed to prove invariant $@inv2$. As shown in Figure 1, the trace (u_s, u_e) that encodes with real-time trigger-response properties satisfies two properties:1) the response event could remove the constraints on $Tick$ event and re-enable the $Tick$ event; 2) The enabledness of $Tick$ event is preserved all the other events in E, formally presented in Equation 5.

$$(u_s, u_e) \models \Box(u_e \in R \wedge \bigcirc en(Tick, u_s)) \\ \wedge \Box(u_e \in E \setminus \{Tick\} \wedge en(Tick, u_s) \wedge \bigcirc en(Tick, u_s)) \tag{5}$$

To model real-time systems, we introduce a special event $Tick$ in event traces to represent the progress of time. In a real-time system, one essential property requires that time should always progress. The event traces of real-time systems should always be *infinite traces* with *infinitely many* $Tick$ events. A trace (u_s, u_e) is defined to satisfy $Resp(T, R, w, d)$ provided:

$$(u_s, u_e) \models Resp(T, R, w, d) \Leftrightarrow (u_s, u_e) \models \Box \Diamond u_e = Tick \\ \wedge (u_s, u_e) \models \Box(u_e \in T \Rightarrow \Diamond u_e \in R) \\ \wedge (u_s, u_e) \models \Box(u_e \in T \Rightarrow \neg en(T, u_s) \mathcal{U} u_e \in R)$$

In the formalization, the first property is the timed property, which guarantees an infinite number of $Tick$ events occurring in the trace. Moreover, only a finite number of $non - Tick$ events could occur between any two $Tick$ events. The second property requires that a response event always follows a trigger event. It is also required that the trigger events are disabled between the trigger-response pair to avoid the recurring of trigger events. Our previous work in [5] provided conditions and proofs for the machine to satisfy $Resp(T, R, w, d)$ in Theorem 1 under fairness assumptions on intermediate events, response events and $Tick$ event. In this paper we use temporal operator to present the conditions needed for the trace to satisfy $Resp(T, R, w, d)$.

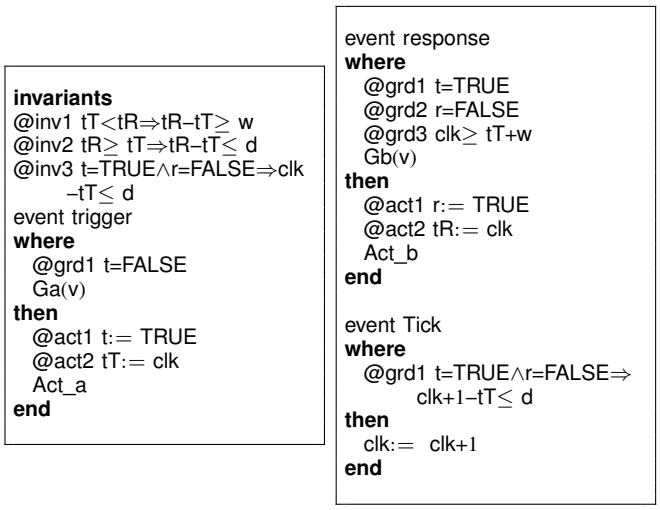

Fig. 1. Model Timing Properties of Trigger-Response Events with Delay and Deadline

Theorem 1. *Let M be an Event-B machine, let $Resp(T, R, w, d)$ be a real-time trigger-response property and let $H \subset E$ such that $T \cap H = \varnothing$ and $R \cap H = \varnothing$. If trace $(u_s, u_e) \in traces(M)$ satisfies the conditions 1) to 6), then $(u_s, u_e) \models Resp(T, R, w, d)$ when $M \wedge WF(H) \wedge WF(R) \wedge WF(Tick)$:*

1) $(u_s, u_e) \models \Box((en(H, u_s) \vee en(R, u_s)) \Rightarrow \neg en(T, u_s))$;
2) $(u_s, u_e) \models \Box(u_e \in T \wedge \bigcirc(\neg en(T, u_s) \wedge (en(H, u_s) \vee en(R, u_s))))$;
3) $(u_s, u_e) \models \Box(u_e \in E \setminus (T \cup R) \wedge en(H, u_s) \bigcirc(en(H, u_s) \vee en(R, u_s)))$;
4) $(u_s, u_e) \models \Box(u_e \in E \setminus (T \cup R) \wedge en(R, u_s) \bigcirc en(R, u_s))$;
5) $(u_s, u_e) \models cov(H, \neg en(R))$;
6) $w < d$.

In this setting, condition 2 guarantees that the trigger event enables $H \cup R$ and disables itself, and $H \cup R$ would be kept enabled by condition 3. Condition 1 is used to avoid the recurring of trigger events between trigger-response pair. Condition 4 requires that if a response event is enabled, it cannot be disabled by any event other than a response event to avoid the scenario that intermediate events and response events are enabled alternatively but never get executed. In this paper, we relax this condition by allowing strong fairness assumptions on response events. Rules and proofs are provided

in Theorem 2.

Theorem 2. *Let M be an Event-B machine, let $Resp(T, R, w, d)$ be a real-time trigger-response property and let $H \subset E$ such that $T \cap H = \varnothing$ and $R \cap H = \varnothing$. If the following conditions 1)-5) are true, then $traces(M)$ satisfies $Resp(T, R, w, d)$ when $M \wedge WF(H) \wedge SF(R) \wedge WF(Tick)$:*

1) $(u_s, u_e) \models \Box((en(H, u_s) \vee en(R, u_s)) \Rightarrow \neg en(T, u_s))$;
2) $(u_s, u_e) \models \Box(u_e \in T \wedge \bigcirc(\neg en(T, u_s) \wedge (en(H, u_s) \vee en(R, u_s))))$;
3) $(u_s, u_e) \models \Box(u_e \in E \setminus (T \cup R) \bigcirc (en(H, u_s) \vee en(R, u_s)))$;
4) $(u_s, u_e) \models cov(H, \neg en(R))$;
5) $w < d$.

Proof. We first prove $(u_s, u_e) \models \Box(u_e \in T \Rightarrow \Diamond u_e \in R)$ by contradiction. Let $u \in traces(M)$ and assume $u_e(i) \in T$ and $(u_s, u_e)_i \models \Box\, u_e \notin R$. Based on condition 3, $(u_s, u_e)_i \models \Box(en(H, u_s) \vee en(R, u_s))$.

$$
\begin{aligned}
&(u_s, u_e)_i \models \Box\, u_e \notin R \wedge \Box(en(H, u_s) \vee en(R, u_s)) \\
\Rightarrow&\{SF(R)\} \\
&\Diamond\Box\, \neg en(R, u_s) \wedge \Box(en(H, u_s) \vee en(R, u_s)) \\
\equiv&\Box(\neg en(R, u_s) \wedge en(H, u_s))
\end{aligned}
$$

Thus $(u_s, u_e)_i \models \Box(\neg en(R, u_s) \wedge en(H, u_s))$. Based on condition 4, $(u_s, u_e) \models cov(H, \neg en(R))$. And Lemma 1 shows that $(u_s, u_e) \models \Box\Diamond(\neg en(H) \vee en(R))$, which could be used to derive a contradiction.

Then we prove $(u_s, u_e) \models \Box(u_e \in T \Rightarrow \neg en(T, u_s)\, \mathcal{U}\, u_e \in R)$. Based on the previous proof, let $u_e(i) \in T$ and $u_e(j) \in R$. Then:

$$
\begin{aligned}
&(u_s, u_e) \models \Box(u_e \in T \Rightarrow \neg en(T, u_s)\, \mathcal{U}\, u_e \in R) \\
\equiv&\{u_e(i) \in T \wedge (u_s, u_e)_i \models \Box(\neg en(R, u_s) \wedge en(H, u_s))\} \\
&(u_s, u_e)_i^j \models \Box(\neg en(T, u_s))
\end{aligned}
$$

Based on condition 1 and condition 2, the trigger event would be kept disabled.

In the last step, we prove that $(u_s, u_e) \models \Box\Diamond u_e = Tick$.

$$
\begin{aligned}
&(u_s, u_e) \models \Box(u_e \in T \Rightarrow \Diamond\, u_e \in R) \\
\Rightarrow&\{\text{Equation (5)}\} \\
&(u_s, u_e) \models \Box(u_e \in T \Rightarrow \Diamond\Box\, en(Tick, u_s)) \\
\Rightarrow&\{WF(Tick)\} \\
&(u_s, u_e) \models \Box(u_e \in T \Rightarrow \Box\Diamond\, u_e = Tick)
\end{aligned}
$$

In the cases that trigger event never occurs, then $Tick$ event is kept enabled. Based on weak fairness assumption, the $Tick$ would infinitely often occurs on the trace.

□

V. Refinement of Real-Time Properties with Fairness Assumptions

Abstraction and refinement are usually essential to manage the complexity of modeling and reasoning about a system. Refinement of a system usually involves changing the variables of the system [22]. Data refinement is used to add more details to the data structure in the model, either by replacing existing variables or adding new variables to the model. In Event-B machines, gluing invariants are used to link the variables in the refined model to the variables in the abstract model.

We use $M =< S, S_0, E, K >$ to denote the abstract machine, which is data-refined to the concrete machine $M' =< S', S_0', F, K' >$. Syntactically the abstract and concrete state spaces are represented by the possible values of the variables v and w respectively. S_0' is defined by a predicate $L(w)$. The transition relation K' of an event $e \in F$ is defined by its guard $H_e(w)$ and action predicate $R_e(w)$. In general, gluing invariants define a relational mapping between concrete and abstract states. We assume $J \in S \leftrightarrow S'$ as a gluing relation that relates the states of M and M'. Syntactically, J is represented by a predicate $J(v, w)$. Event mapping function $g \in F \rightarrow E \cup \{skip\}$ is a total function from refined event labels to abstract event labels and $skip$. The $skip$ events are mapped from concrete new events in M'. We define machine M' refines machine M in Equation (6) by showing that $M \sqsubseteq M'$ *iff* for any concrete trace $(v_s, v_e) \in traces(M')$, there exists an abstract trace $(u_s, u_e) \in traces(M)$, which satisfies that $v_s \in J[u_s]$ and $u_e = g(v_e) \setminus \{skip\}$.

$$
\begin{aligned}
M \sqsubseteq M' \equiv \forall(v_s, v_e) \in traces(M') \Rightarrow \exists(u_s, u_e) \in traces(M) \\
\wedge\, (u_s, u_e) \models (v_s \in J[u_s] \wedge u_e = g(v_e) \setminus \{skip\})
\end{aligned}
\tag{6}
$$

Work in [7] developed rules to prove $M \sqsubseteq M'$ in Theorem 3. Based on the theorem, we designed a two-step refinement strategy to refine the models. In the first step we introduce intermediate events to the trigger-response pair while preserving the timing properties. Then in the second step we refine the timing properties into sub-timing properties. Conditions such as forward simulation, deadlock freedom and conditional convergence are imposed in the first step to guarantee the introduction of intermediate still preserves the real-time trigger-response properties. However, Theorem 3 alone does not guarantee that real-time trigger-response properties are preserved in the refinement step. For example, given that $(u_s, u_e) \in traces(M)$ and $(u_s, u_e) \models Resp(T, R, w, d)$. M' introduces new events G' and H' where G' disables R' but H' re-enables R'. In this case the weak fairness assumptions on R' is not sufficient to prove the eventually occurrence of response events. And the refined timing property $Resp(T', R', w, d)$ could not be satisfied by M'.

Theorem 3. *Given M with transition relation K and M' with transition relation K'. Let F be the set of event labels in M and $\mathcal{N}$ be the introduced new events in M'. $M \sqsubseteq M'$ provided the following conditions hold:*

- *M forward simulated by M'.*
- *M' is deadlock free relative to M: $J[dom(K)] \subseteq dom(K')$.*
- *M' is weakly $(F \setminus \mathcal{N})$-fair.*
- *Events $\mathcal{N}$ in machine M' are conditional convergent under the condition that events $F \setminus \mathcal{N}$ are disabled;*

VI. Conclusion and Future Work

References

[1] E. M. Clarke and J. M. Wing, "Formal methods: State of the art and future directions," *CSUR*, vol. 28, no. 4, pp. 626–643, Dec. 1996. [Online]. Available: https://doi.org/10.1145/242223.242257

[2] L. Lamport, "Proving the correctness of multiprocess programs," *IIEEE Trans. Software Eng.*, vol. SE-3, no. 2, pp. 125–143, Mar. 1977. [Online]. Available: https://doi.org/10.1109/tse.1977.229904

[3] J. S. Ostroff, "Composition and refinement of discrete real-time systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 8, no. 1, pp. 1–48, Jan. 1999. [Online]. Available: https://doi.org/10.1145/295558.295560

[4] D. Méry and M. Poppleton, "Towards an integrated formal method for verification of liveness properties in distributed systems: With application to population protocols," *Softw Syst Model*, vol. 16, no. 4, pp. 1083–1115, Dec. 2015. [Online]. Available: https://doi.org/10.1007/s10270-015-0504-y

[5] C. Zhu, M. Butler, and C. Cirstea, "Semantics of real-time trigger-response properties in Event-B," in *2018 International Symposium on Theoretical Aspects of Software Engineering (TASE)*. IEEE, Aug. 2018, pp. 150–155. [Online]. Available: https://doi.org/10.1109/tase.2018.00028

[6] C. Zhu, M. J. Butler, and C. Cirstea, "Towards refinement semantics of real-time trigger-response properties in Event-B," in *2019 International Symposium on Theoretical Aspects of Software Engineering, TASE 2019, Guilin, China, July 29-31, 2019*, D. Méry and S. Qin, Eds. IEEE, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/TASE.2019.00-26

[7] C. Zhu, M. Butler, and C. Cirstea, "Trace semantics and refinement patterns for real-time properties in Event-B models," *Science of Computer Programming*, p. 102513, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642320301210

[8] Event B.org, "Event-B and the Rodin Platform," 2019, available online at http://www.event-b.org [Accessed: 9 Mar 2019].

[9] G. Sulskus, M. Poppleton, and A. Rezazadeh, "Modelling complex timing requirements with refinement," in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, vol. 9392. IEEE, Jul. 2016, pp. 292–307. [Online]. Available: https://doi.org/10.1109/iri.2016.23

[10] D. Cansell, D. Méry, and J. Rehm, "Time constraint patterns for Event-B development," in *International Conference of B Users*. Springer, 2007, pp. 140–154.

[11] M. Butler and J. Falampin, "An approach to modelling and refining timing properties in b," in *Refinement of Critical Systems (RCS)*, Jan. 2002. [Online]. Available: https://eprints.soton.ac.uk/256235/

[12] M. R. Sarshogh and M. Butler, "Specification and refinement of discrete timing properties in Event-B," in *AVoCS 2011*, Sep. 2011, event Dates: September 2011. [Online]. Available: https://eprints.soton.ac.uk/272480/

[13] T. S. Hoang and J.-R. Abrial, "Reasoning about liveness properties in Event-B," in *International Conference on Formal Engineering Methods*. Springer, 2011, pp. 456–471.

[14] S. Schneider, H. Treharne, H. Wehrheim, and D. M. Williams, "Managing ltl properties in Event-B refinement," in *International Conference on Integrated Formal Methods*. Springer, 2014, pp. 221–237.

[15] T. S. Hoang, S. Schneider, H. Treharne, and D. M. Williams, "Foundations for using linear temporal logic in Event-B refinement," *Formal Aspects of Computing*, vol. 28, no. 6, pp. 909–935, 2016.

[16] C. C. KM and J. Misra, "Parallel program design: a foundation," 1988.

[17] S. Hudon and T. S. Hoang, "Systems design guided by progress concerns," in *International Conference on Integrated Formal Methods*. Springer, 2013, pp. 16–30.

[18] M. Abadi and L. Lamport, "An old-fashioned recipe for real time," *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 5, pp. 1543–1571, Sep. 1994. [Online]. Available: https://doi.org/10.1145/186025.186058

[19] T. S. Hoang, "How to interpret failed proofs in Event-B," *Technical report*, vol. 672, 2010.

[20] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, IEEE. IEEE, Sep. 1977, pp. 46–57. [Online]. Available: https://doi.org/10.1109/sfcs.1977.32

[21] D. Plagge and M. Leuschel, "Seven at one stroke: Ltl model checking for high-level specifications in b, z, csp, and more," *International journal on software tools for technology transfer*, vol. 12, no. 1, pp. 9–21, 2010.

[22] R.-J. Back, "Refinement calculus, part II: Parallel and reactive programs," in *Workshop/School/Symposium of the REX Project (Research and Education in Concurrent Systems)*. Springer, 1989, pp. 67–93.