Fairness in Event-B

By
Xuedong Li
Supervisor: Kai Engelhardt
Assessor: Carroll Morgan

A thesis submitted for the degree of
FIXME

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY · AUSTRALIA

Computer Science and Engineering,
UNSW, Australia.

May 2014

Contents

0.1 Acknowledgement

Testing LTL: $\Diamond\Box\phi \Rightarrow \Box\Diamond\phi$.

Testing citations [Eng13, MP91, HA11, MP92, MP88].

0.2 Introduction

Testing LTL: $\Diamond\Box\phi \Rightarrow \Box\Diamond\phi$.

Testing citations [Eng13, MP91, HA11, MP92, MP88].

0.3 Background

0.3.1 Event-B

Event-B is a formal method for system-level modelling and analysis. Event-B uses set theory as a modelling notation and stepwise refinement to represent systems at different abstraction levels. Between different refinement levels, Event-B uses mathematical proof to justify consistency.

We use machines to describe the dynamic behavior of a model. For a machine M, the variables V are declared in the Variable section. The type of the variables and also the additional constraints I for the variables are declared in the Invariants section.

The possible state changes for the machine is defined by events. For an event E, additional external variables t are introduced in the Parameter section. The preconditions G which enable an events execution are given by guards. The actions A implement the detailed changing between pre state and after state. Proofs are required to ensure the after state satisfy the invariants.

Every machine in Event-B must have an event INITIALIZATION (E_0) to set all the variables as the model start. E_0 does not have any guards nor parameters and the actions must cover the setup for all the variables.

However, in Event-B, only one event can happen at a time, which means original Event-B does not support concurrent execution.

In the rest parts of the thesis report, we will use following notions:

M : Machine

v : Variables

J : Invariant

For all event i $\in$ I, I is the set of all events

E_i : The event i, where E_0 is the INITIALIZATION event

g_i: The guard of Event i

a_i: The action of Event i

0.3.2 Temporal Logic

Temporal logic is a system of rules and symbolism for representing, and reasoning about, propositions qualified in terms of time. With modalities referring to time, linear temporal logic or linear-time temporal logic (LTL) is a modal temporal logic. In LTL, we can encode formulas about the future of paths, one example of this is "one condition will be always true after the system satisfy another condition". Some LTL notations we will use

in the rest parts of this thesis report are:
$\Box\phi$ Globally: ϕ always happen (hold on the entire subsequent path)
$\Diamond\phi$ Finally: ϕ eventually happen (hold somewhere on the subsequent path)
$\bigcirc\phi$ Next: ϕ will happen next (hold at next state)
The LTL for a system is defined under the assumption of infinite execution. It does not make any sense for using LTL in a finite execution system (a system will automatic halt after limited executions).

0.4 Fairness-for-Event-B

0.4.1 Introduction

Fairness is a property of unbounded nondeterminism, which ensure the system fairly take the transactions when they are enabled, to avoid the situation that some transactions are always ignored. For example, a randomize number generator must be able to eventually generate every value within its range. Fairness can't be used on systems with finite executions.

Weak Fairness We say that an event i is weakly fairness (WF) iff it infinitely often disabled or infinitely often taken.

$$\text{WF(i)} \iff \Box\Diamond\neg g_i \vee \Box\Diamond taken_i$$

Strong Fairness We say that an event i is strong fairness (SF) iff it eventually disabled forever or infinitely often taken.

$$\text{SF(i)} \iff \Diamond\Box\neg g_i \vee \Box\Diamond taken_i$$

However, by working on the definitions for weak fairness and strong fairness, we can easily find that weak fairness imply strong fairness. Weak fairness is more detailed that strong fairness.

$$\text{WF(i)} \Rightarrow \text{SF(i)}$$

Event Merging For some situations we only have the fairness constraints for a set of events instead of fairness constraints for each of them. For example, Peter wants to travel all over the world, once he arrive a new place, he might stay there and spent several days for sightseeing. In this example, assume having an one-day sightseeing is event $E_{sightseeing}$, moving on air is event E_{air} and moving by train is event E_{train}. By illustrating an event E_{travel}, as a merged event for moving to a new place by whatever mean. The following constraint

$$\text{WF}(travel)$$

must be satisfied to ensure he can move from one place to the others rather than spending infinite days sightseeing somewhere without moving.
Since he does have any concern for choosing transportation, it's not necessary to ensure

$$\text{WF}(air) \text{ or } \text{WF}(train)$$

which force him always eventually choose airplane or train.
For this type of constraints, we introduce following rule for merging event:
For event E_i and event E_j, where i,j $\in$ I, the merged event $E_{i\vee j}$ satisfy:

$$\begin{array}{lcl} g_{i\vee j} & = & g_i \vee g_j \\ a_{i\vee j} & | & a'_{i\vee j}\wedge \\ & & (((g_i \wedge \neg g_j) \Rightarrow (a'_{i\vee j} = a_i)) \\ & & \vee((\neg g_i \wedge g_j) \Rightarrow (a'_{i\vee j} = a_j)) \\ & & \vee((g_i \wedge g_j) \Rightarrow (a'_{i\vee j} = a_j \vee a_j))) \end{array}$$

The merged event E_j have a disjoint guard of event E_i and event E_j, and $a_{i\vee j}$ is the corresponding available action from of event E_i and event E_j, if multiple guards satisfied, the action will be the disjoint of a_j and a_j.
Some properties for fairness constraints of merged events are:
Assume i,j $\in$ I, event $E_{i\vee j}$ is the merged event from E_i and E_j

$$\text{WF(i)} \vee \text{WF(j)} \Rightarrow \text{WF}(i \vee j)$$
$$\text{SF(i)} \vee \text{SF(j)} \Rightarrow \text{SF}(i \vee j)$$

Event Splitting In an opposite way, we can also split an event into several mutually exclusive sub-events, by refining them with different mutually exclusive strengthen guards which fullfill the original guard.
For event h where h $\in$ I, event i and event j are splitted events of h iff:

$$g_i \vee g_j = g_h$$
$$g_i \wedge g_j = \varnothing$$
$$a_i = a_j = a_h$$

Note that the fairness constraints the original event hold can't be inherited by any one of its sub-events, for the same reason explained in event merging part. However, similar to the outcome of the event merging process, we can say a event is fair if we can ensure that one of its sub-events is fair.

$$\text{WF(i)} \vee \text{WF(j)} \Rightarrow \text{WF(h)}$$
$$\text{SF(i)} \vee \text{SF(j)} \Rightarrow \text{SF(h)}$$

Where event i and j are sub-events of event h.

Currently Event-B does not have suppport for LTL, most of properties described by LTL (including fairness) can't be assumed or proved by Event-B. In the rest parts of this report, we will extend the original Event-B method and show the usage of fairness

assumptions in proving response, a basic property of unbounded nondeterminism.

$${}_{1}^{2}\prod_{3a}^{4^b} \bigvee_{i=1}^{n} E_i$$

References

[Eng13] Kai Engelhardt. Proving response properties of event-b machines. Lecture notes for COMP2111, accessed at 2013/11/12, `http://www.cse.unsw.edu.au/~cs2111/13s1/lec/note09.pdf`, May 2013.

[HA11] Thai Son Hoang and Jean-Raymond Abrial. Reasoning about liveness properties in Event-B. In Shengchao Qin and Zongyan Qiu, editors, *Formal Methods and Software Engineering - 13th International Conference on Formal Engineering Methods, ICFEM 2011, Durham, UK, October 26-28, 2011. Proceedings*, volume 6991 of *LNCS*, pages 456–471. Springer-Verlag, 2011.

[MP88] Zohar Manna and Amir Pnueli. The anchored version of the temporal framework. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 201–284. Springer-Verlag, 1988.

[MP91] Zohar Manna and Amir Pnueli. Tools and rules for the practicing verifier. In Richard F. Rashid, editor, *Computer Science: A 25th Anniversary Commemorative*, pages 121–156. ACM Press and Addison-Wesley Publishing Co., 1991.

[MP92] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1992.