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.

June 2014

Contents

0.1 Acknowledgement 2
0.2 Introduction 3
0.3 Background 4
0.3.1 Event-B 4
0.3.2 Temporal Logic 5
0.4 Fairness-for-Event-B 6
0.4.1 Introduction 6

Bibliography **10**

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 a notation resembling set theory for modeling systems and stepwise refinement to connect systems at different levels of abstraction. Mathematical proofs are required to justify consistency of system models as well as the preservation of safety properties proved at higher levels of abstraction down to lower levels.

In Event-B, machines are built to describe the static properties and track 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

Since the event INITIALIZATION is different to the others, and can be executed once only. We introduce set I_1 to store the events other than INITIALIZATION.

$$I = I_1 \cap \{0\}$$

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)

In Event-B, ϕ can be a state of the machine, or a status that an event is chosen to be executed.

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 for system which have infinite execution, this property ensure that transitions (events) can be "fairly" chosen to execute when their preconditions (guards) are satisfied. Without fairness constraints, it's possible that some transitions can be always ignored even if they are enabled in some states. It does not make sense to deal with fairness constraints on systems with finite execution.

Weak Fairness An transition set t is a weakly fair set if every transition in t is always eventually disabled or infinitely often taken.

$$\mathrm{WF}(i) \Leftrightarrow \Box\Diamond\neg g_i \vee \Box\Diamond\mathrm{taken}_i$$

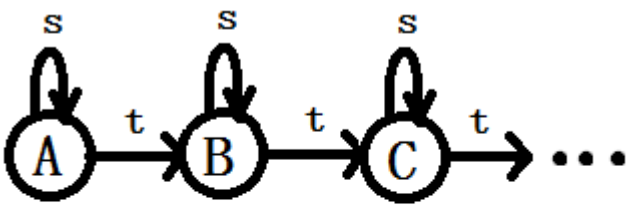
Strong Fairness An transition set t is a strongly fair set if every transition in t is eventually always disabled or infinitely often taken.

$$\mathrm{SF}(i) \Leftrightarrow \Diamond\Box\neg g_i \vee \Box\Diamond\mathrm{taken}_i$$

However, by working on the definitions for weak fairness and strong fairness, we can easily find that strong fairness imply weak fairness. Strong fairness is stronger than weak fairness.

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

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 arrives at a new place, he might stay there and spend several days for sightseeing. In this example, assume having an one-day sightseeing is modeled as the event E_{s}, and we specify the event E_{t} as the event of moving to a new place by either of the two offered means of transporation.



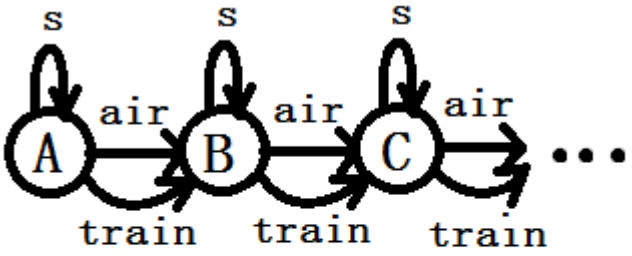
Event s: Sightseeing
Event t: Travel

The following constraint

$$\mathrm{WF}(t)$$

must be satisfied to ensure he does move from one place to another rather than spending the rst of his life sightseeing somewhere without ever moving again.

Assume Peter can only use train or aiplane to travel, we replace event E_{t} by event E_{air} and event E_{train}, where the event E_{air} is for moving on air and the event E_{train} is for moving by train.

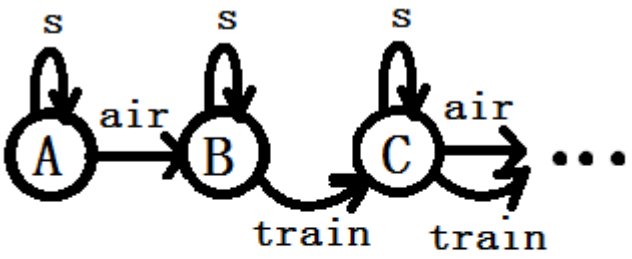


Event s: Sightseeing
Event train: By train
Event air: On air

If every place has an airport and a train station, or none of them, travel is fair (can be always eventually chosen) iff air is fair (can be always eventually chosen) or train is fair (can be always eventually chosen). Since if one of the transportations is fair to choose, he can always eventually move to other places.

$$\mathrm{I} \wedge g_{air} = \mathrm{I} \wedge g_{train} \Rightarrow (\mathrm{WF}(travel) \Leftrightarrow \mathrm{WF}(air) \vee \mathrm{WF}(train))$$

However, if there have at least one place that has only have airport, $\mathrm{WF}(train)$ can't ensure he can leave this place unless there have extra requirements force him occationally move on air (weak fairness for air when train is disabled). We also have similar problem for places only have train station.



Event s: Sightseeing
Event train: By train
Event air: On air

We have:

$$(\mathrm{WF}(train) \wedge \mathrm{WF}(air@g_{air} \wedge \neg g_{train})) \vee (\mathrm{WF}(air) \wedge \mathrm{WF}(train@g_{train} \wedge \neg g_{air})) \Leftrightarrow \mathrm{WF}(t)$$

For this type of constraints, we introduce the following rule for merging events: For event E_i and event E_j, where $i, j \in \mathrm{I}$, the merged event $E_{i \vee j}$ is given as:

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

The merged event $E_{i \vee j}$ can happen whenever either of its constituent events (E_i and E_j) could fire, and if $E_{i \vee j}$ fires, the effect is one of the enabled constituent events.
Some properties for fairness constraints of merged events are:
Assume $i, j \in \mathrm{I}$, event $E_{i \vee j}$ is the result of merging events E_i and E_j.

$$(\mathrm{WF}(i) \wedge \mathrm{WF}(j@g_j \wedge \neg g_i)) \vee (\mathrm{WF}(j) \wedge \mathrm{WF}(i@g_i \wedge \neg g_j)) \Leftrightarrow \mathrm{WF}(i \vee j)$$
$$(\mathrm{SF}(i) \wedge \mathrm{SF}(j@g_j \wedge \neg g_i)) \vee (\mathrm{SF}(j) \wedge \mathrm{SF}(i@g_i \wedge \neg g_j)) \Leftrightarrow \mathrm{SF}(i \vee j)$$

Event Splitting In an opposite way, we can also split an event into two mutually exclusive sub-events, by refining them with different mutually exclusive strengthened guards which imply the original guard.

For event h where $h \in I$, event i and event j are *split* events of h iff:

$$g_i \vee g_j \Leftrightarrow g_h$$
$$\neg(g_i \wedge g_j)$$
$$a_i = a_j = a_h$$

Note that the fairness constraints of the original event are not inherited to 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, since the sub-events have same guard as the original event.

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

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 for proving response properties.

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

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.