Analysing Event-B Models with Multi-level Events

Christian ATTIOGBE

AeLoS / LINA – UMR CNRS 6241

Séminaire interne AeLoS - 20/02/2014

Plan

1. Introduction
2. Introduction Event-B
3. Reachability Analysis
4. Multi-level events model
5. Putting into practice
6. Generalisation

Plan

1 Introduction

2 Introduction Event-B

3 Reachability Analysis

4 Multi-level events model

5 Putting into practice

6 Generalisation

Context and motivations

Context

Modelling and analysis of complex systems :
- modelling the behaviour of systems,
- expressing and proving safety and liveness properties.

Motivation

To cover more easily the analysis of liveness (reachability, time) in Event-B.

Objectives

To elaborate a methodological approach to specify and prove reachability properties.
Extending B without changing it (Abrial, 1998)

Plan

1 Introduction

2 Introduction Event-B

3 Reachability Analysis

4 Multi-level events model

5 Putting into practice

6 Generalisation

Overview of Event-B

Event-B

Event-B [Abrial] is a modelling and development method where components are modelled as **abstract machines** which are composed and **refined into concrete machines**.

Machines

An *abstract machine* describes a mathematical model of a system behaviour : it is a discrete transition system.

Overview of Event-B

Dynamic and static parts

In the Event-B modelling process, *abstract machines* constitute the dynamic part whereas *Contexts* are used to describe the static part.
A *Context* is made of carrier **sets** and **constants**. It may contain properties (defined on the sets and constants), **axioms** and **theorems**.
A context is **seen** by abstract machines.

Engineering tools

Event-B is supported by **several (industrial) tools : AtelierB, ProB, Rodin**
Widely used in industries
Consistency checking
Refinement of machines into codes

Overview of Event-B

A machine is made of variables and invariant, together with several *event* descriptions.

MACHINE A
SEES *myContext*
SETS $\cdots$ // in addition to the context
VARIABLES $gv_x, \cdots$
INVARIANTS *Predicate* $\cdots$
EVENTS
initialisation $\widehat{=}$ $\cdots$
; $e_i \widehat{=}$ $\cdots$
; $e_j \widehat{=}$ $\cdots$
; $e_k \widehat{=}$ $\cdots$
; $e_l \widehat{=}$ $\cdots$
; $e_m \widehat{=}$ $\cdots$
END

event e_i
any lv_1 lv_2 $\cdots$
where
a guard(lv_i, gv_x)
then
$gv_x :=$ *Substitution*$(\cdots)$
end
; event e_j
when
a guard(gv_x)
then
$gv_x :=$ *Substitution*$(\cdots)$
end

Overview of Event-B

An event-based model *M* is described by an invariant (correct state space) and a set of events

Abstract machine :

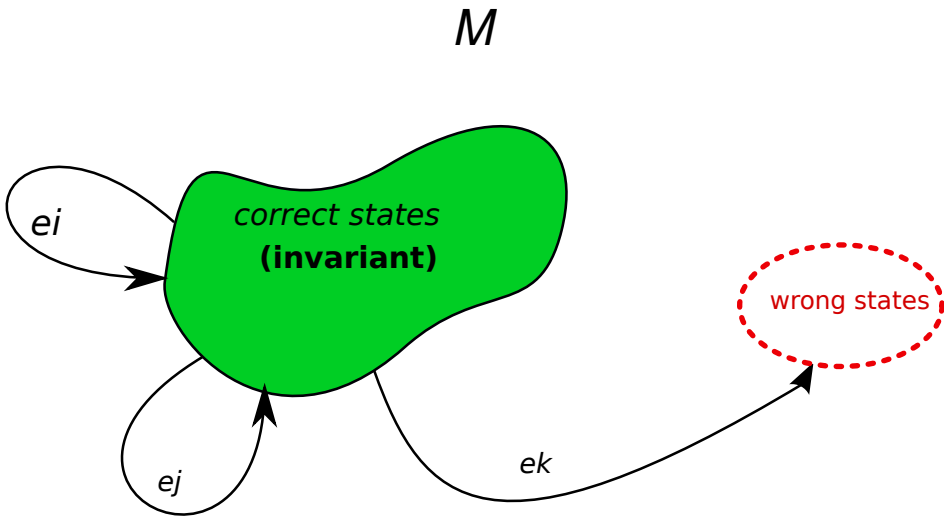


FIGURE: Invariant-based discrete-event transitions

Consistency checking by theorem proving

- The initialisation establishes the invariant.
- The events preserve the invariant.

Safety Analysis

General statement :

$$M \models P?$$

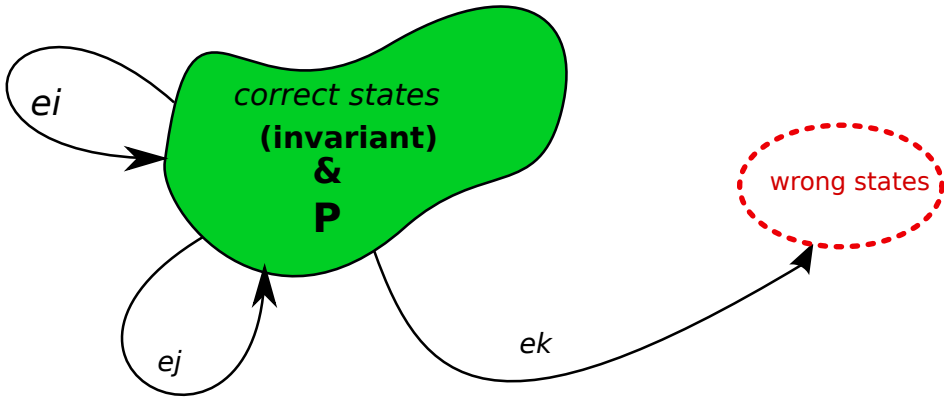


FIGURE: Invariant-based discrete-event transitions

Liveness properties (reachability)

- Several works on reachability
  - Mussat, Abrial, *Introducing Dynamic Constraints in B*, 1998
    (*leadsto*, decreasing a variant)
  - Abrial, Cansell, Mery, *Refinement and Reachability in Event_B*, 2005
  - Bert, Stouls, Barradas, et al. *Propriétés de vivacité dans les systèmes B*, 2005, 2007
    (*leadsto*+*ensures* à la Unity)
  - Diagne, Frappier, Mammar, *Reachability ... substitution refinement*, 2011-2013
    (CTL, substitution refinement, B, AtelierB)
- ProB (model-checking approach)
  supports an extended version of LTL (properties on states and transitions)
  $G([op] => X\{i > 0\})$
- Deploy Project

☞Still challenging

LTL/ProB - very pragmatic - integrated tools

f, g	formula/predicate (from the B model)
e(Op)	is the operation/event Op enabled ?
{predicate}	check any predicate

Temporal operators (future)	**Temporal operators (past)**
G f : globally	H f : history (dual to G)
F f : finally	O f : once (dual to F)
X f : next	Y f : yesterday (dual to X)
f U g : until	f S g : since (dual to until)
f W g : weak until	
f R g : release	f T g : trigger (dual to release)

[op]	if the next operation is op
$G([op] => X\{i > 0\})$	globally, if the next operation is op then $i > 0$ in the reached state

☞ still restricted for reasoning on events and their ordering

Plan

1 Introduction

2 Introduction Event-B

3 Reachability Analysis

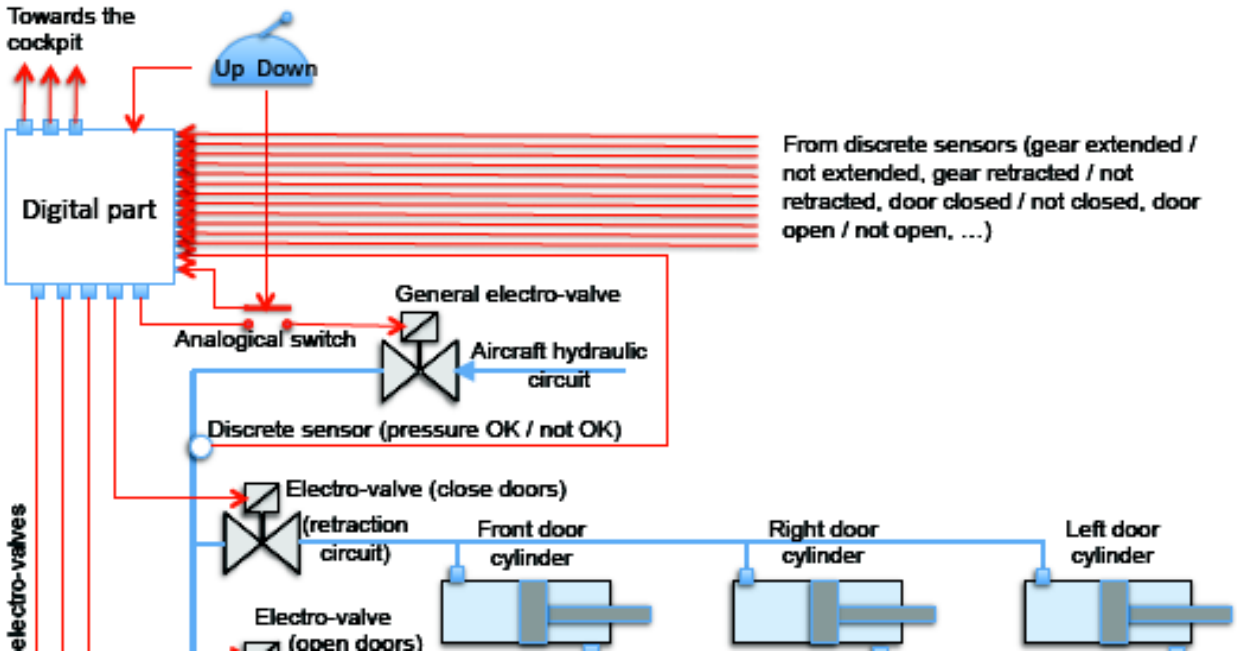
4 Multi-level events model

5 Putting into practice

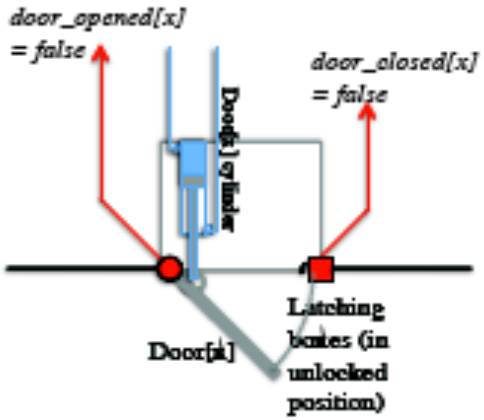
6 Generalisation

Reasoning on the **Landing Gear System** case study

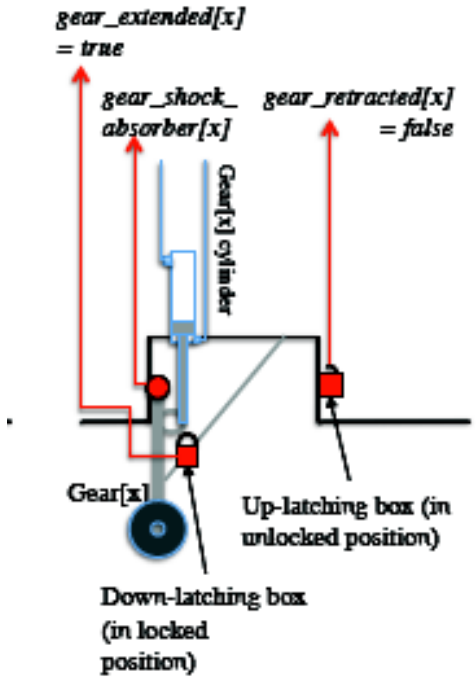
A case study submitted to the model-based FM communities for the ABZ'2014 Conf.



Requirements - reasoning on events ordering



- (b) door in the intermediate position



Requirements - examples of properties

Prop1

When the command line is working (normal mode), if the landing gear command handle has been pushed DOWN and stays DOWN, then eventually the gears will be locked down and the doors will be seen closed ;

Prop2

If one of the three doors is still seen locked in the closed position more than 7 seconds after stimulating the opening electro-valve, then the boolean output normal mode is set to false.

Modelling the properties

Modelling the properties= relating some specific events observed during the execution of the system.

- LAMPORT invented a simple mechanism by which the happened-before ordering can be captured numerically.
  A Lamport **logical clock** is an incrementing counter maintained in each process.
- For every **two events a** and **b** occurring in the same process, and C(x) being the timestamp for a certain event x, it is necessary that **C(a)** never equals **C(b)**.
- LAMPORT's Clock condition : **a occurs before b**, **C(a) before C(b)**

BUT these events are not (necessarily) the *behavioural events* of a B model ☹

Plan

1 Introduction

2 Introduction Event-B

3 Reachability Analysis

4 Multi-level events model

5 Putting into practice

6 Generalisation

Multi-level events : Behaviour vs observation

- Event-B model behaviour : set of traces of events of the model.
  $behEvents = \{e_i, e_j, e_k, \cdots\}$
- External observations of the behaviour of an Event-B model
  - An observation : something happens (an event). An observed event has a name.
  - Analysis of observations : a sequence of events of interest.
    They are observed ; they do not impact on the behaviour of the model.
    obsEvents the observed events

$$behEvents \cap obsEvents = \emptyset$$

Definition (Analytical observation)

An analytical observation is a set of observed events equipped with a happened-before ordering : a partial relation defined on the timestamps of the events $(obsEvents, <)$.

Multi-level events

obsEvents = {a, b, c, d}

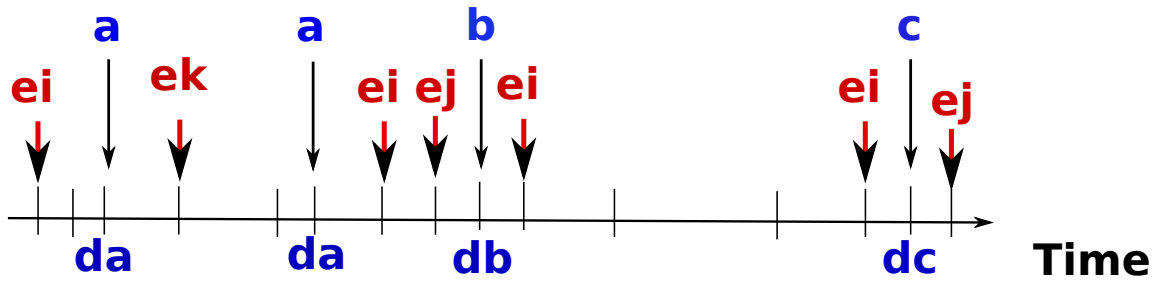


FIGURE: Observed events and timestamps

☞ Increase the Event-B model with the set obsEvents and a logical clock llc.

Observation events

Given a set *obsEvents* of events and a logical clock modelled as a natural number, the occurrences of the events can be ordered by the timestamps given by the clock.
Two events cannot happen at the same time.
A partial function to record the timestamps of the events.

$$ldate \in obsEvents \pfun \mathbb{N}$$

We can compare and reason on the timestamps of the events which occur during a sequence or specifically within a **finite** period of time.

☞ Increase the Event-B model with the function ldate and a variable standing for the logical clock value llc.

Plan

1 Introduction

2 Introduction Event-B

3 Reachability Analysis

4 Multi-level events model

5 Putting into practice

6 Generalisation

Modelling properties and formal analysis

Build an event-model equipped with both sets of events

The property Prop1 of the Landing Gear System

$$\begin{array}{l}\forall dj.(((dj \in \boldsymbol{N}) \wedge (dcge \in dom(ldate)) \wedge (dj = ldate(dcge)) \\ \qquad \wedge (endCycle = TRUE) \wedge dj < llc) \Rightarrow \\ \quad \exists di.((di \in \boldsymbol{N}) \wedge (downH \in dom(ldate)) \wedge (di = ldate(downH)) \wedge (di < dj) \wedge \\ \qquad \forall ii.(ii \in \boldsymbol{N} \wedge di \leq ii \wedge ii < dj \Rightarrow ldate \sim [\{ii\}] \neq \{upH\})))\end{array}$$

✌ Benefit : easy to prove with the B theorem-prover

Reasoning on the observations

Practically, the properties are to be included in the invariant.

The method is

- compliant with the refinement.
- compliant with the composition/decomposition.
  The events of an abstract machine result from the composition of events of different machines.

  Example of Producer/Consumer : P.a, C.b, ...
- Events of distributed processes P_i may be indexed and compared.

Reasoning on the observations

Clock condition : Impact of non-determinism of the Event-B Model

- Scheduling and synchronisation of events in case of controlled system ✔
- Priority of observations on behavioural events ✔
- Tedious/Untractable in case of (uncontrolled) highly distributed system ✘

Plan

1 Introduction

2 Introduction Event-B

3 Reachability Analysis

4 Multi-level events model

5 Putting into practice

6 Generalisation

Generalisation : forthcoming works

- Generalisation to the behavioural events ; can be synthetised/simulated.
- Mixing behaviour and observation events
- Analysis of set of observations (after several runs)
- Generalisation to delay, (hard) time constraints ? see Prop2 (Mery, Cansell, Rehm)
- High-level analysis
- Observation events with high priority

Conclusion

- Lightweight method to enable liveness reasoning in Event-B
- Based on LAMPORT's logical clock and multi-level events (Behaviour and observations)
- Several ideas to be studied, on the basis of existing results (causality, time).

References

- Lamport, *Time, Clocks, and the Ordering of Events in Distributed Systems*, ACM 1978
- Diagne, Mammar, Frappier *Reachablity with refinement...*
- Cansell, Méry, Rehm, *Time Constraint Patterns for Event B Development*, 2007
- JR Abrial
- Illiasov et al.(Turku), *Event B, Real-time Constraints*, Deploy,