The Event-B Modelling Notation

J.-R. Abrial

October 2007

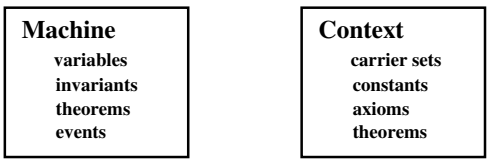
Version 1.5

The Event-B Modelling Notation

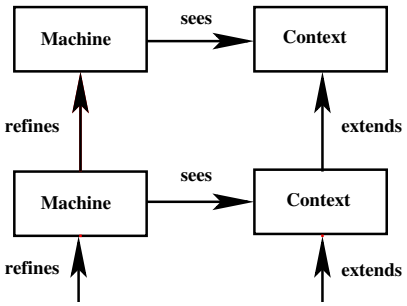
Contents

1 Machines and Contexts

The primary concept in doing formal developments in Event-B is that of a *model*. A model contains the complete mathematical development of a *Discrete Transition System*. It is made of several components of two kinds: *machines* and *contexts*. Machines contain the variables, invariants, theorems, and events (section 2) of a model, whereas contexts contain carrier sets, constants, axioms, and theorems of a model:



Machines and contexts have various relationships: a machine can be "refined" by another one, and a context can be "extended" by another one (no cycles are allowed in both these relationships). Moreover, a machine can "see" one or several contexts. An example of machine and context relationship is as follows:



The following shows a more formal description of machines and contexts:

machine
$< machine\_identifier >$
refines $\star$
$< machine\_identifier >$
sees $\star$
$< context\_identifier >$
$\ldots$
variables
$< variable\_identifier >$
$\ldots$
invariants
$< label >: < predicate >$
$\ldots$
theorems $\star$
$< label >: < predicate >$
$\ldots$
events
$< event >$
$\ldots$
variant $\star$
$< variant >$
end

context
$< context\_identifier >$
extends $\star$
$< context\_identifier >$
$\ldots$
sets $\star$
$< set\_identifier >$
$\ldots$
constants $\star$
$< constant\_identifier >$
$\ldots$
axioms $\star$
$< label >: < predicate >$
$\ldots$
theorems $\star$
$< label >: < predicate >$
$\ldots$
end

In these descriptions, section names (that is, **axioms** , **extends**, etc.) followed by a star $\star$ are optional. Notice that all machine and context identifiers must be distinct in the same model

A model can only contain contexts, or only machines, or both. In the first case, the model represents a pure mathematical structure. In the third case, the machines of the model are parameterized by the contexts. Finally, the second case represents a machine which is not parameterized.

Notice that invariants, axioms and theorems denote predicates which are labelled: such labels must be distinct within a given section. Predicates are defined by means of the Event-B mathematical language (section 6).

The **variant** section is explained in section 3.

The previous representation of machines or contexts as well as the usage of keywords do not form a syntax as there is no such syntax in the Event-B notation. The only part of the Event-B notation which depends on a well defined syntax is the Event-B mathematical notation (section 6).

2 Events

A machine event represents a transition. It is essentially made of guards and actions (section 4). The guards together denote the necessary condition for the event to be enabled, whereas the actions together represent the way the variables of the corresponding machine are modified. Events can have no guards, they can be also simple and guarded (keyword **where**) or parameterized and guarded (keywords **any** and **where**). When an event lies in a machine which refines another one then the event may specify (if any) the abstract event(s) it refines (keyword **refines**). When a refining event refines an abstract event which is parameterized, one may provide some witnesses (keyword **with**), which are presented in section 5. All this is denoted as follows:

$< event\_identifier > \widehat{=}$
status
$\{normal, convergent, anticipated\}$
refines $\star$
$< event\_identifier >$
...
any $\star$
$< parameter\_identifier >$
...
where $\star$
$< label >: < predicate >$
...
with $\star$
$< label >: < witness >$
...
then $\star$
$< label >: < action >$
...
end

Notice that guards, witnesses, and actions are labelled. As said previously, such labels must be distinct within a given section.

Most of the time the **status** section states that the event is "normal". A "convergent" event must be a new event in a refined machine (one that does not appear in the abstraction). All convergent events in a

machine are concerned with the **variant** section of that machine (see section 3). An "anticipated" event is a new event which is not "convergent" yet but should become "convergent" in a subsequent refinement.

Notice that a new event might be "normal": it means that it is not concerned by the **variant** section. This is used quite often when using refinement to add more detail to a specification (without considering any liveness property of the system).

3 Variant

The **variant** section appears in a refined machine (section 1) containing some "convergent" events (section 2). This machine section contains either a natural number expression which must be decreased by each "convergent" event, or a finite set expression which must be made strictly smaller by each "convergent" event.

4 Actions

An action can be deterministic, in which case it is made of a list of distinct variable identifiers, followed by $:=$, followed by a list of expressions. The latter must be of the same length as the former. Variables which do not occur in the list are not modified. This is illustrated below:

$$< variable\_identifier\_list > := < expression\_list >$$

Here is an example of a deterministic action in an event situated in machine with variables x. y, and z:

$$x,\ y \ := \ x + z,\ y - x$$

Variables x and y are modified as indicated whereas variable z is not modified.

Alternatively, an action can be non-deterministic, in which case it is made of a list of distinct variable identifiers, followed by $:|$, followed by a before-after predicate. Variables which do not occur in the list are not modified. This is illustrated below:

$$< variable\_identifier\_list > :| < before\_after\_predicate >$$

The before-after predicate may contain all variables of the machine: they denote the corresponding values just before the action takes place. It can also contain some of the variable identifiers of the list: such identifiers are primed, they denote the corresponding values just after the action has taken place. Example: suppose we have three variables x. y and z. Here is a non-deterministic action:

$$x, y \quad :| \quad x' > y \ \wedge \ y' > x' + z$$

Variable x becomes greater than y and the variable y becomes greater than x' (the new value for x) added to z (a variable which is not modified).

A final option is to define a non-deterministic action as a variable identifier (not a list), followed by $:\in$, followed by a set expression. This is illustrated below:

$$< variable\_identifier > :\in < set\_expression >$$

This form is just a special case of the previous one. It can always be translated to a non-deterministic case as shown in the following example. Suppose a machine with variables A, x, and y. Here is an action:

$$x :\in A \cup \{y\} \qquad \text{which is the same as} \qquad x :| x' \in A \cup \{y\}$$

Variable x becomes a member of the set $A \cup \{y\}$, whereas variables A and y are not modified.

Finally, notice that all actions in the same event must be concerend with different variables. As an example, it is not possible to have the following actions in the same event:

$$x := 5$$
$$x :| x' > x$$

5 Witnesses

When a concrete event refines an abstract one which is parameterized, then all abstract parameters must receive a value in the concrete event. Such values are called the witnesses. Each witness is labelled with the concerned abstract parameter. The witness is defined by a predicate involving the abstract parameter. Most of the time, this predicate is a simple equality. Next is an example showing two witnesses. On the left hand side we have an abstract event named pass with two parameters. On the right hand side we have a concrete event named new_pass refining pass

pass $\widehat{=}$
any
p
l
where
$grd1: \ p \mapsto l \in aut$
$grd2: \ sit(p) \mapsto l \in com$
then
$act1: \ sit(p) := l$
end

new_pass $\widehat{=}$
refines
pass
any
d
where
$grd1: \ d \in ran(dap)$
with
$p: \ p = dap^{-1}(d)$
$l: \ l = dst(d)$
then
$act1: \ sit(dap^{-1}(d)) := dst(d)$
$act2: \ dap := dap \rhd \{d\}$
end

When the concrete event is also parameterized then an abstract parameter which is the same as a concrete need not be given an explicit witness: it is always the corresponding concrete parameter.

6 Syntax of the Event-B Mathematical Notation

It is defined in a separate document entitled "The Event-B Mathematical Notation"