*chrono*Event-B: A Rodin Plugin for Timed Constraints in Event-B

Amel Mammar[1], Cabrel Feukeng Momo[1,2], Marc Frappier[2], Paul Gibson[1]
[1] SAMOVAR, Institut Polytechnique de Paris, Télécom SudParis, France
first.last@telecom-sudparis.eu
[2]Université de Sherbrooke, 2500 Boulevard de l'Université, Sherbrooke(QC), Canada
first.last@usherbrooke.ca

Event-B [1] is a formal method particularly suitable for incremental modelling of complex systems through refinement. Although recognized for its rigor in formal specification and verification, this method has a significant limitation: the absence of language constructs to represent time, which reduces its relevance for systems where time is a critical factor. In the health domain, for instance, many requirements are timed [3]. For example, doctors need to express that *An X-Ray has a minimum and maximum exposure time (depending on a number of factors)*. This illustrates an action that has a duration and that some durations may be limited to a minimum and/or maximum value. To address this shortcoming, our work proposes an innovative extension of Event-B according to the following three contributions: (*i*) definition of a set of timed properties relevant to model timed requirements, (*ii*) formalisation in Event-B of the exhibited timed properties: invariants, particular events, etc. (*iii*) implementation of a Rodin-Plugin that permits defining new kinds of Event-B events to which different timing properties are associated.

1 Timed properties

In our work, we have considered timed properties that are common to several timed-critical domains. These properties can be divided into two classes. The first class includes mono-task constraints like *duration*, *separation*, *periodicity*, while the second class considers ordering constraints between two tasks like *followedBy* and *precededBy* relationships. Property *Duration* states the amount of time that a task takes to accomplish its execution. The time that elapses between the starts of two successive instances of a task is represented by the property of *periodicity*, while the property of *separation* denotes the time elapsed between the end of an instance and the beginning of the next one. Finally, time constraints can be attached to a task that should follow or precede another one. For instance, from the health domain, we have the requirement that the blood analysis must be carried out no later than two hours after the blood sample is taken.

2 Event-B modeling of timed constraints

In order to model timed constraints in Event-B, we have defined a new Event-B datatype, called *Scheduler*. A scheduler is associated with each task to keep track of its execution instants as an increasing sequence of integers. For instance, the scheduler [0, 3, 3, 8, 9], of a given task T, states that we have 3 instances of T: the first one started at 0 and finished at 3, the second started at 3 and finished at 8, the last one started at 9 but did not finish yet. Timed constraints are then modeled using the schedulers associated with different tasks. The property min/max duration $d_{min}(T)/d_{max}(T)$ of a task T is modeled using the following invariant, where *start* (resp. *end*) denotes the beginning (resp. ending) instant of an instance t of a task T ($\mathbb{T}$ denotes the set of instances of the task T:

$Duration(t) \mathrel{\widehat{=}} end(t)$ - $start(t)$, where $t \in \mathbb{T}$.
$maxDuration(T, d_{max}(T)) \mathrel{\widehat{=}} \forall\, t.\ t \in \mathbb{T} \Rightarrow Duration(t) \leq d_{max}(T)$
$minDuration(T, d_{min}(T)) \mathrel{\widehat{=}} \forall\, t.\ t \in \mathbb{T} \Rightarrow Duration(t) \geq d_{min}(T)$

A task T with a duration is modelled with two following Event-B events that correspond to its beginning and ending.

Event *start_T* $\widehat{=}$
where
grd1: $\exists\, k.\ (k \in \mathbb{N} \wedge \textbf{Size}(scheduler_T) = 2k)$
then
act1: $scheduler_T := \textbf{Append}(\text{scheduler}_T, CK)$
end

Event *end_T* $\widehat{=}$
where
grd1: $\exists\, k.\ (k \in \mathbb{N} \wedge \textbf{Size}(\text{scheduler}_T) = 2k + 1)$
grd2: $Ck \geq/\leq \textbf{Last}(\text{scheduler}_T) + d_{min}/d_{max}$
then
act1: $scheduler_T := \textbf{Append}(\text{scheduler}_T, CK)$
end

1. Event *start_T*: this event checks that there is no running instance of the task T (Guard grd1) by checking the the size is even, then, the scheduler $scheduler_T$, associated with the task T, is updated by inserting the start instant CK (Action act1) which denotes the current value of the master clock.

2. Event *end_T*: this event checks that there is a running instance of the task T (Guard grd1) and that the task duration has elapsed within the min/max bounds (Guard grd2). Then, the scheduler $scheduler_T$ is updated by inserting the end instant CK (Action act1) which denotes the current value of the master clock CK.

3 A Rodin plugin for timed constraints

To make timed constraints available under Rodin, we have developed a plugin that permits developers to build and verify time-dependent systems. The plugin includes an Event-B theory that defines the scheduler datatype and its associated operators and properties like **Size**, **Append**, etc. Invariants representing the properties, the guards, and the actions of events are modeled as operators that are implicitly generated when an event is created. Basically, when a task with duration is created, the events *start_A* and *end_A* are generated under Rodinbut without displaying their guards and actions that are hidden and implicit. This theory also defines a set of inference rules in order to help developers automatically discharge the proof obligations that ensure that the specified events preserve the timed constraints. For instance, we have defined and proved the following inference rule to discharge the proof obligation that ensures that the event *end_T* preserves the invariant *maxDuration*:

$$\frac{maxDuration(T, d_{max}(T)) \quad \exists k.(k \in \mathbb{N} \wedge \textbf{Size}(scheduler_T) = 2k) \quad Ck \leq \textbf{Last}(scheduler_T) + d_{max}}{maxDuration(\textbf{Append}(scheduler_T, CK), d_{max}(T))}$$

To create such specific events to represent tasks with timed constraints, the plugin extends the Rodin interface with buttons. A more detailed description of the plugin and the installation instructions are available in [2].

References

[1] J.R. Abrial. *Modeling in Event-B*. Cambridge University Press, 2010.

[2] A. Mammar and C. Feukeng Momo. chronoEventB. https://github.com/AmelMammar/chronoEventB.

[3] Annette Ten Teije, Mar Marcos, Michel Balser, Joyce van Croonenborg, Christoph Duelli, Frank van Harmelen, Peter Lucas, Silvia Miksch, Wolfgang Reif, Kitty Rosenbrand, et al. Improving medical protocols by formal methods. *Artificial intelligence in medicine*, 36(3):193–209, 2006.