

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

Amel Mammam¹, Cabrel Feukeng Momo^{1,2}, **Marc Frappier**², Paul Gibson¹

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

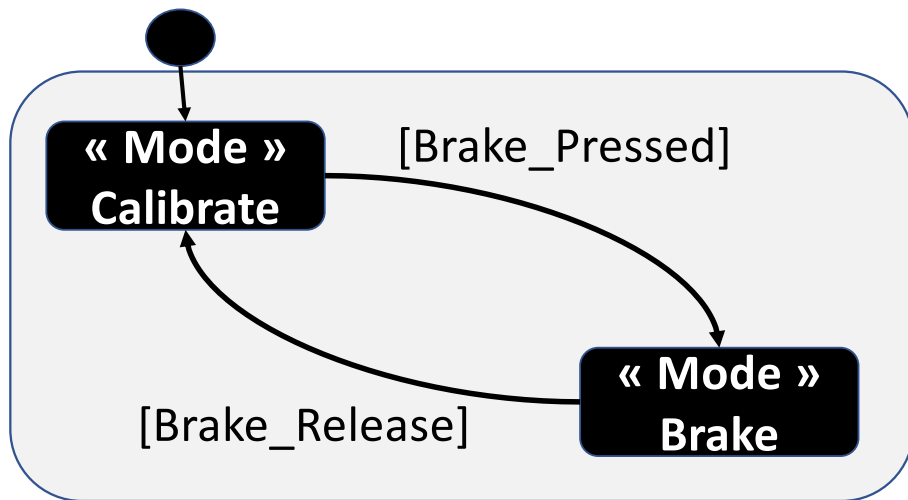
1 - Context

- Timing constraints are often considered too late in the design process
- Unsatisfactory for critical systems
 - temporal properties are essential for safety
 - must be integrated from the early modeling stages to prevent catastrophic failures
- Time is not a native concept in Event-B
- Managing timing constraints introduces a lot of details that are cumbersome to deal with
- This plugin hide some of the details

2 - Case Study

Anti-lock Braking System : ABS

The ABS is a control unit that ensures a car's stability while driving and in extreme braking situations. It operates in two modes: Calibrate mode, which is the default mode, and Brake mode.



□ Calibration Mode

- ❖ SenseSpeed
- ❖ Calibrate

□ Braking Mode

- ❖ SenseBrake
- ❖ BrakeControl
- ❖ BrakeWheel

2 - Case Study

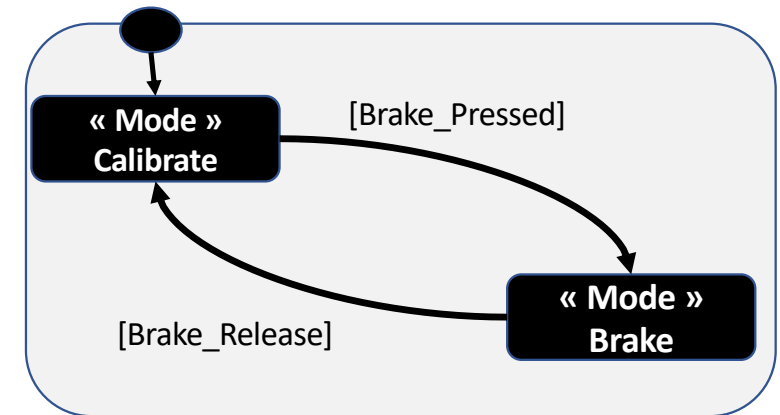
Anti-lock Braking System requirements

Calibrate Mode (Calibration)

ABS1: *SenseSpeed* is always followed by Calibrate *Scheduling

ABS2: *SenseSpeed* is periodic with a period of 100 ms *Periodicity

ABS3: Calibrate is performed within 10 ms following *SenseSpeed* *Duration



Brake Mode (Braking)

ABS4: *SenseBrake* is always followed by BrakeControl *Scheduling

ABS5: BrakeControl is always followed by BrakeWheel *Scheduling

ABS6: *SenseBrake* is periodic with a period of 10 ms *Periodicity

ABS7: BrakeWheel is performed within 1 ms following *SenseBrake* *Duration

3 - State of Art

	Contributions	Limitations
H. Peng, X. Zhang, G. Cao, Z. Liu, Y. Jing, and L. Rao. <i>A time refinement framework based on iUML-B state machine. Scientific Programming</i>	Modeling delays using temporal variables and deadlines.	Limited handling of simultaneous events.
M. R. Sarshogh and M. J. Butler. <i>Specification and refinement of discrete timing properties in Event-B.</i>	Addition of Deadline, Delay, and Expiry properties	Approach specific to trigger-response models
C. Fu and K. Zheng. <i>Patterns for modeling task-level timing constraints with Event-B.</i>	Task synchronization (priority, coincidence)	No tool support available
R. Banach, M. J. Butler, S. Qin, N. Verma, and H. Zhu. <i>Core hybrid event-b I : Single hybrid Event-B machines.</i>	Hybrid extension with continuous behavior	Lack of a functional tool
A. Iliasov, A. B. Romanovsky, L. Laibinis, E. Troubitsyna, and T. Latvala. <i>Augmenting Event-B modeling with real-time verification</i> & J. Berthing, P. Boström, K. Sere, L. Tsiopoulos, and J. Vain. <i>Refinement-based development of timed systems</i>	Verification of temporal properties using timed automata (UPPAAL)	Multiplicity of formalisms and the need for advanced knowledge to effectively use them

4 - Definitions

- A **Task** is a unit of work or an operation that needs to be executed. It represents a specific activity within a system or application.

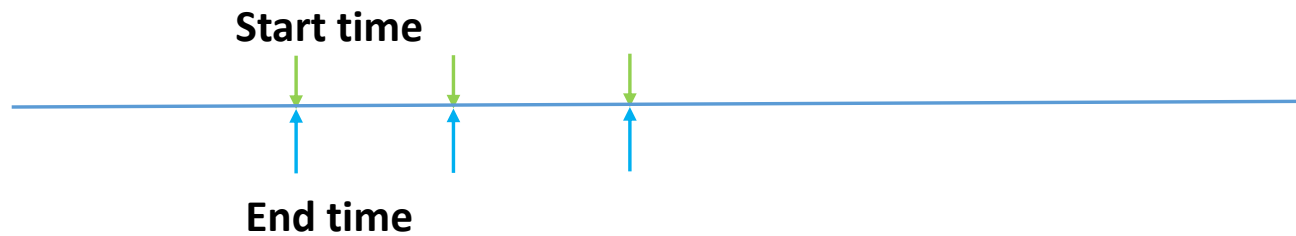
eg : processing a request

- A **Scheduler** is the component responsible for planning and organizing task execution. It determines *when* and *in what order* tasks should run.

eg : operating system scheduler

5 – Timed Properties : Timed constraints

□ Atomic Task



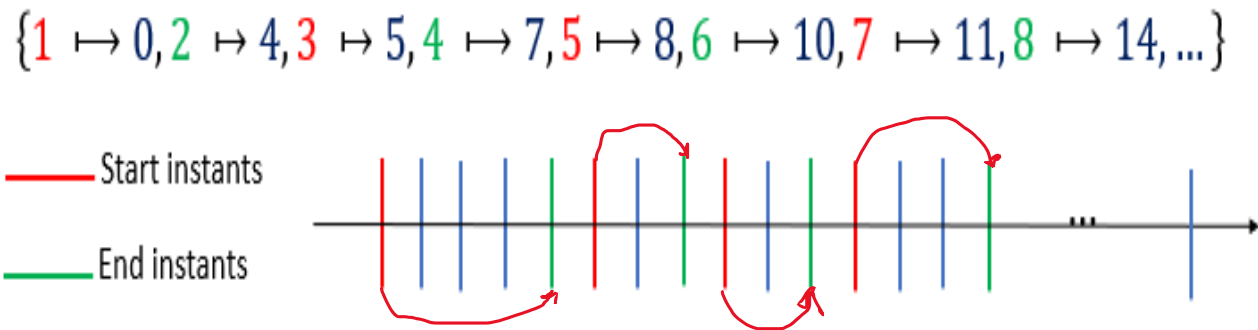
□ Non-Atomic Task



5 – Modeling : TaskScheduler Type

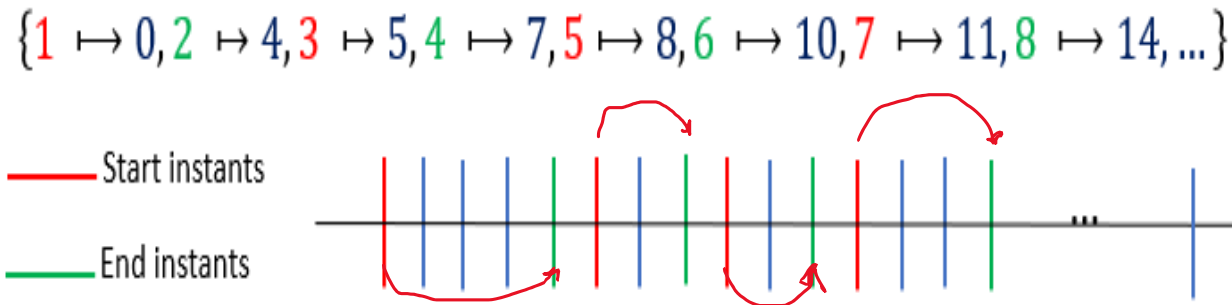
The set of all non-decreasing functions f from $1..n$ to \mathbb{N} such that values at odd positions separated by two indices are different.

$$\{n, f. n \in \mathbb{N} \wedge f \in 1..n \rightarrow \mathbb{N} \wedge (\forall x, y. x \in 1..n \wedge y \in 1..n \wedge x \leq y \Rightarrow f(x) \leq f(y)) \wedge (\forall k. k \in 1 \cdot \cdot (n-1) \div 2 \Rightarrow f(2k-1) \neq f(2k+1)) \mid f \}$$



5 – Modeling : TaskScheduler Type

Each task has its own task scheduler



TaskScheduler operations

- TaskSchedulerBefLast
- TaskSchedulerAppend
- TaskSchedulerSize
- TaskSchedulerFirst
- TaskSchedulerLast

5 – Modeling : Progress Event and global clock

To mark the progress of our global clock **CK** in steps of time unit, we define the *ProgressEvent* event with the step parameter.

```
ProgressEvent: not extended ordinary >  
ANY  
◦ step >  
WHERE  
◦ @progressGrd: step ∈ ℕ1 not theorem >  
THEN  
◦ @progressAct: CK = CK + step >  
END
```

5 – Modeling : Atomic Task

Let SH_A Be the taskScheduler of the task A

INV

$atomicTaskSchedule(SH_A) \triangleq$

$(\exists k. k \in \mathbb{N} \wedge taskSchedulerSize(SH_A) = 2k)$

\wedge

$(\forall k. k \in 1 \dots taskSchedulerSize(SH_A) \div 2 \Rightarrow SH_A(2k - 1) = SH_A(2k))$

Event Atomic_A \triangleq

then

act1: $SH_A := taskSchedulerAppend(taskSchedulerAppend(SH_A, CK), CK)$

end

5 – Modeling : Task with Duration

Let SH_A Be the taskScheduler of the task A

INV

taskSchedulerDuration(SH_A, d) \triangleq

$(\forall k. k \in 1 \dots \text{taskSchedulerSize}(SH_A) \div 2)$

\Rightarrow

$SH_A(2k) - SH_A(2k - 1) = d$

Event start_A $\hat{=}$

where

grd1: $\exists k.(k \in \mathbb{N} \wedge \text{size}(SH_A)=2k)$

then

act1: $SH_A := \text{taskSchedulerAppend}(SH_A, CK)$

end

Event end_A $\hat{=}$

where

grd1: $\exists k.k \in \mathbb{N} \wedge \text{taskSchedulerZise}(SH_A)=2k + 1$

grd2: $CK - \text{taskSchedulerLast}(SH_A) = d$

then

act1: $SH_A := \text{taskSchedulerAppend}(SH_A, CK)$

end

5 – Modeling : Task with period

The difference between the start time of an occurrence (index $2k+1$) and the start time of the preceding occurrence (index $2k-1$) is equal to p u.t.



INV *periodicalTaskScheduler*(SH_A, p) \triangleq

$$\forall k. k \in 1 \dots (\text{taskSchedulerSize}(SH_A) - 1) \div 2$$

\Rightarrow

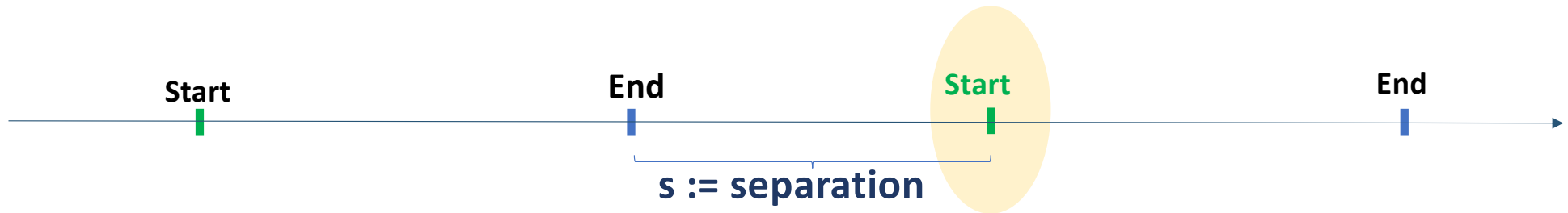
$$SH_A(2k + 1) - SH_A(2k - 1) = p$$

A guard `@grd1` is added to the event **Atomic_A** for an atomic task (respectively **start_A** for a non-atomic task).

`@grd1: $SH_A \neq \emptyset \Rightarrow CK - \text{taskSchedulerBefLast}(SH_A) = p$`

5 – Modeling : Task with Separation

The difference between the start time of an occurrence (index $2k+1$) of A and the end time of the preceding occurrence (index $2k$) is equal to s u.t



INV $sepDuration(SH_A, s) \triangleq$

$$\forall k. k \in 1 \dots (taskSchedulerSize(SH_A) - 1) \div 2$$

\Rightarrow

$$SH_A(2k + 1) - SH_A(2k) = s$$

A guard `@grd4` is added to the event **Atomic_A** for an atomic task (respectively **start_A** for a non-atomic task).

`@grd4: taskSchedulerSize(SH_A) ≥ 2 ⇒ CK - taskSchedulerLast(SH_A) = s`

5 – Modeling : Precedence relationship

Given two tasks A and B, this dependency requires that every execution of an occurrence of task A be preceded by the execution of an occurrence of task B before the next occurrence of A.

INV

$preceded(SH_A, SH_B) \triangleq$

$$\forall k. k \in 1..(taskSchedulerSize(SH_A) + 1) \div 2$$

\Rightarrow

$$\exists k'. k' \in 1..taskSchedulerSize(SH_B) \wedge$$

$$SH_B(2k') \leq SH_A(2k - 1) \wedge$$

$$2k - 2 \in dom(SH_A) \Rightarrow SH_B(2k' - 1) \geq SH_A(2k - 2)$$

PRECEDED_BY(A,B)



BABAB...



BBBABBABBA



A...

5 – Modeling : Preceded relationship

The guard `@grdpcd` is added to the event `Atomic_A` if task A is atomic (respectively `start_A` if task A is non-atomic).

`@grdpcd:` $(SH_A = \emptyset \Rightarrow \exists k. 2k \in \text{dom}(SH_B)) \wedge$

$(SH_A \neq \emptyset \Rightarrow \exists k. 2k + 1 \in \text{dom}(SH_B \triangleright \text{taskSchedulerLast}(SH_A)..CK) \wedge 2k + 2 \in \text{dom}(SH_B \triangleright \text{taskSchedulerLast}(SH_A)..CK))$

PRECEDED_BY(A,B)



BABAB...



BBBABBABBA



A...

5 – Modeling : followed relationship

Given two tasks A and B, this constraint requires that the execution of an occurrence of task A be followed by the execution of an occurrence of task B, before the next occurrence of A.

INV

$$followed(SH_A, SH_B) \triangleq \forall k.$$

$$k \in 1..(taskSchedulerSize(SH_A) - 1) \div 2$$

⇒

$$\exists k'. (2k' - 1 \in dom(SH_B \triangleright SH_A(2k)..SH_A(2k + 1))) \wedge$$

$$2k' \in dom(SH_B \triangleright SH_A(2k)..SH_A(2k + 1))$$

FOLLOWED_BY(A,B)

- ✓ ABABAB...
- ✗ BBABBBABBA
- ✗ BABAABAA...
- ✗ AA...

PRECEDED_BY(A,B)

- ✓ BABAB...
- ✓ BBBABBBABBA
- ✗ A...

5 – Modeling : followed relationship

The guard `@grdfwd` is added to the event `Atomic_A` if A is an atomic task (respectively `start_A` if A is a non-atomic task).

`@grdfwd`: $SH_A \neq \emptyset \Rightarrow \exists k. (2k+1 \in \text{dom}(SH_B \triangleright \text{taskSchedulerLast}(SH_A) .. CK) \wedge 2k+2 \in \text{dom}(SH_B \triangleright \text{taskSchedulerLast}(SH_A) .. CK))$

FOLLOWED_BY(A,B)



ABABAB...



BBABBBABBA



BABAABAA...

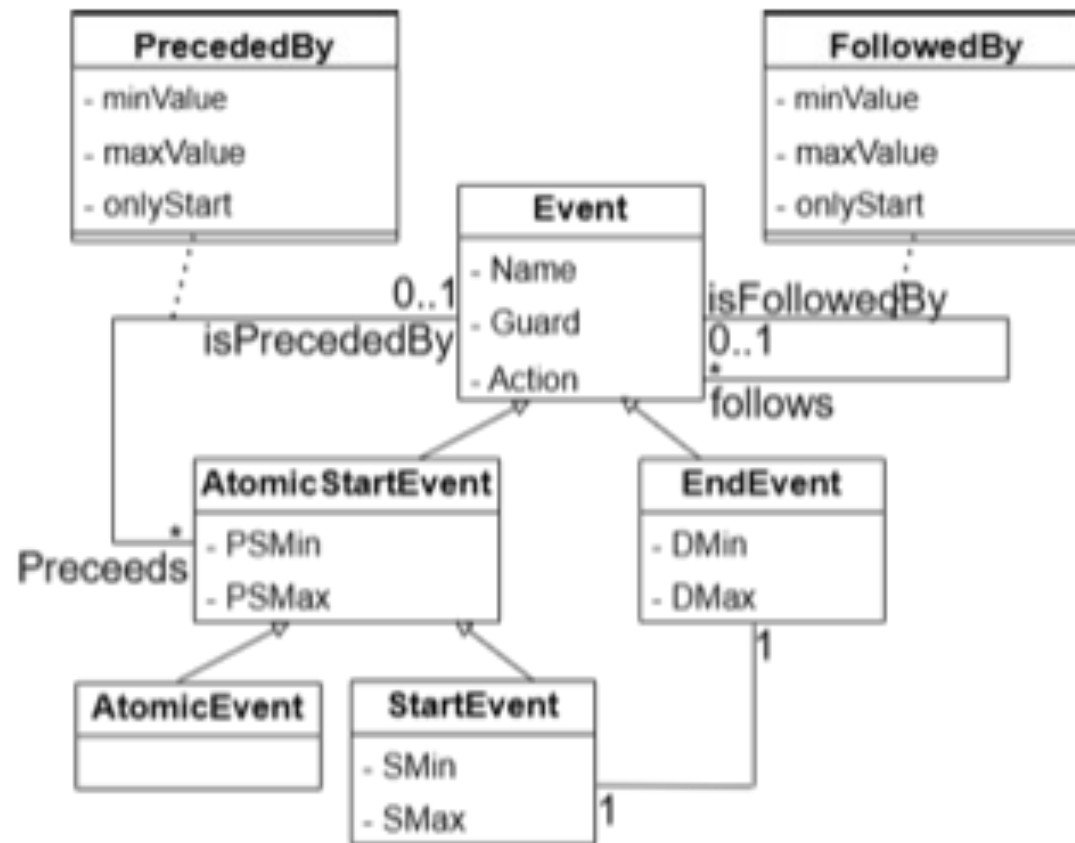
AA...

6- Plugin Features

- Creation of atomic and non-atomic tasks
- Creation of dependency constraints between tasks
 - PrecededBy (min, max)
 - FollowedBy (min,max)
- Adding temporal constraints to an existing task
 - Period and Separation
- Automatic generation of guards and invariants
- Proof rules provided in theory

Ref : <https://github.com/AmelMammar/chronoEventB>

Event Metamodel

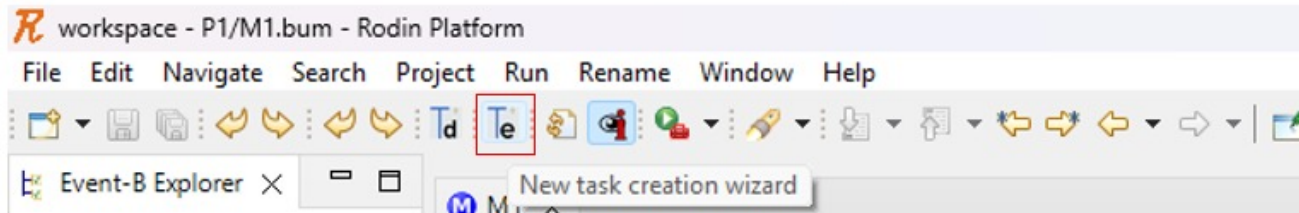


Conclusion

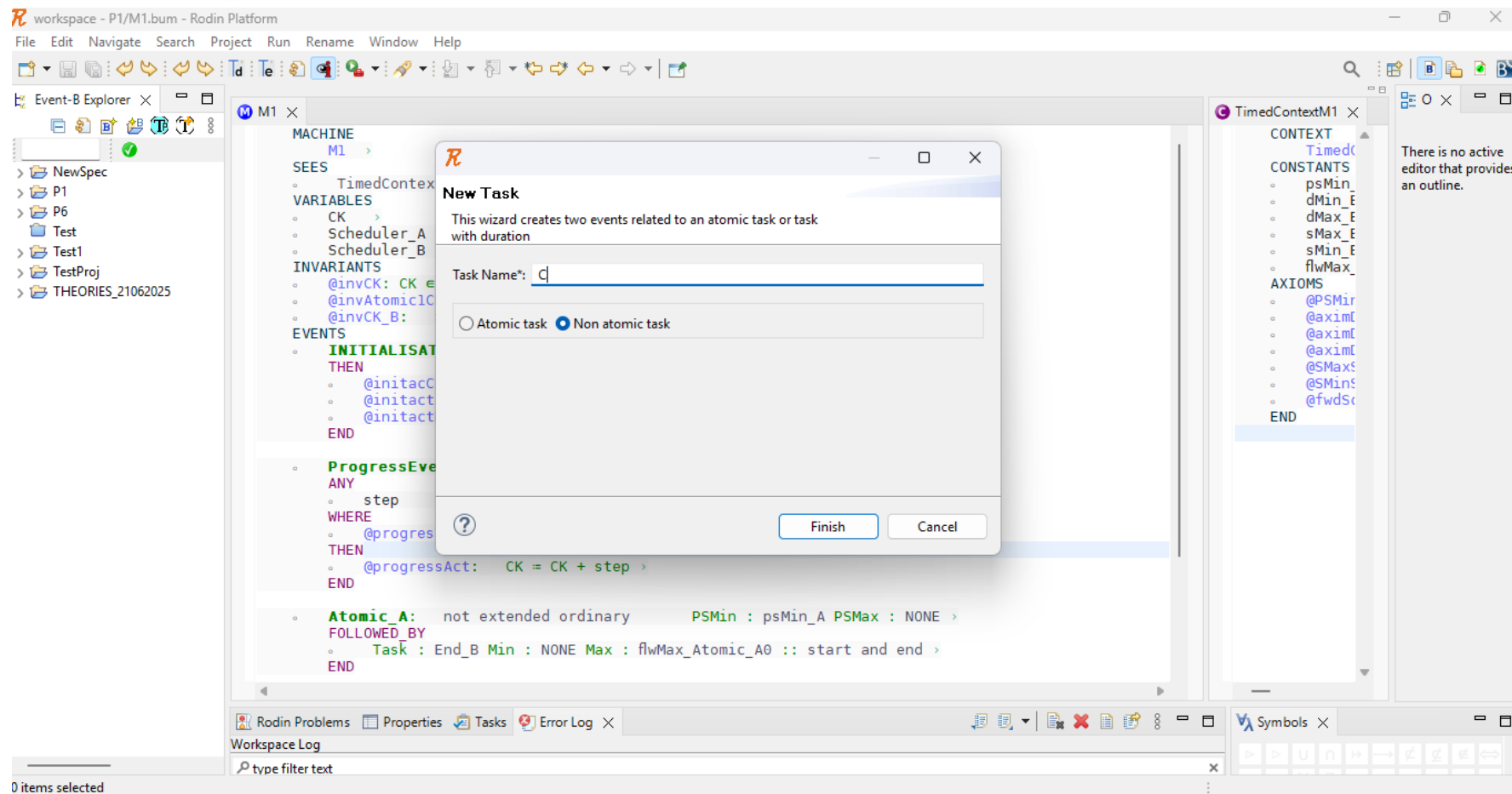
- A classification of temporal constraints for Event-B, providing a structured framework for their specification: a dual modeling of tasks
- A mathematical formalization of temporal dependencies, ensuring consistency between tasks.
- The implementation of a functional Rodin plugin that seamlessly integrates our theoretical advances into the development environment.

Perspectives

- Hierarchical task refinement
 - Develop a mechanism allowing an abstract task to be refined into a set of more precise subtasks,
 - while preserving temporal properties.
- Adaptive temporal dependencies
 - Introduce the ability to refine dependencies between tasks
 - Enable the specification of increasingly precise temporal constraints throughout the system refinement process.

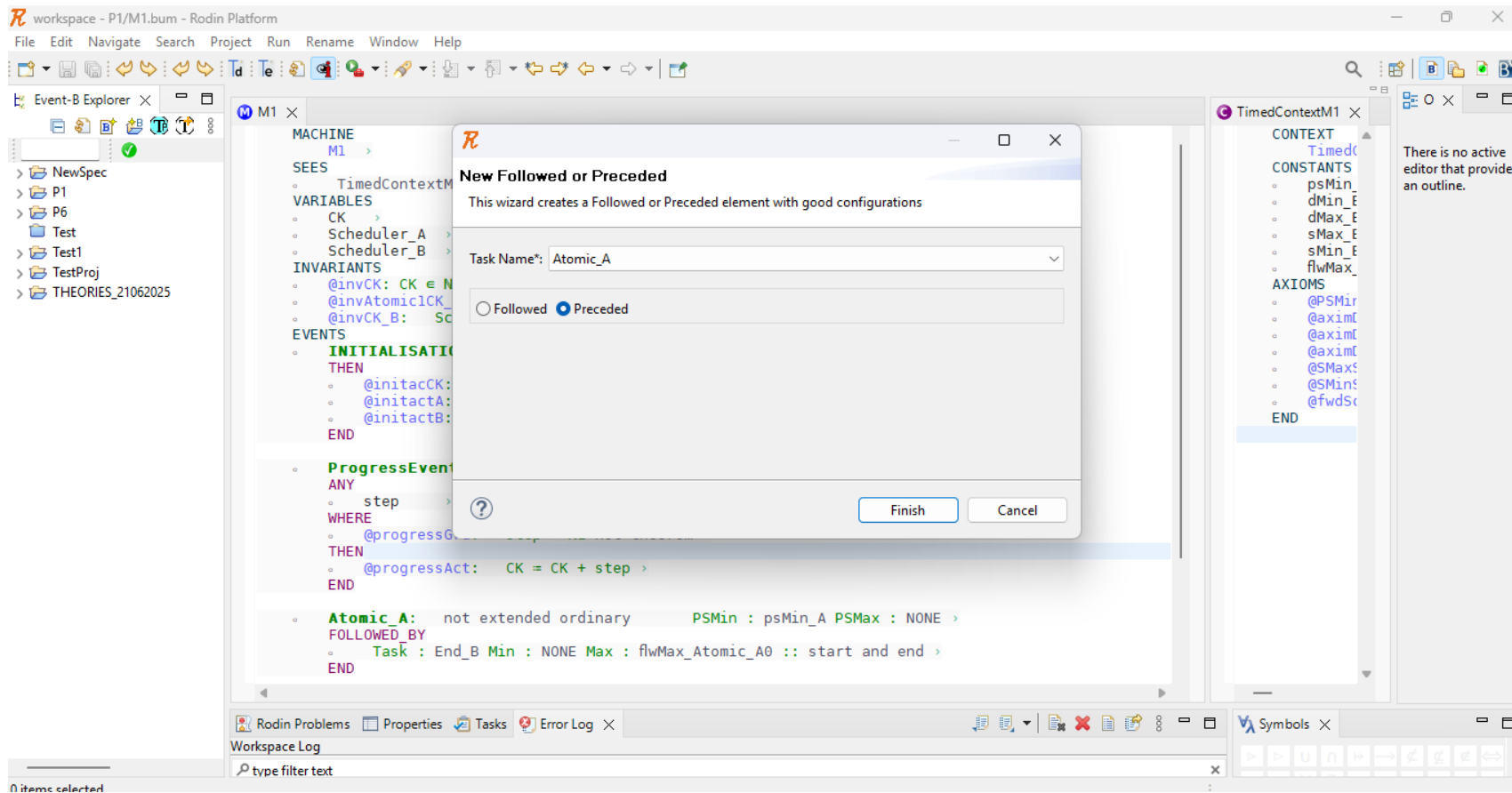


Creation of a new task





Creation of a new dependency



Illustration

- **Atomic_A:** not extended ordinary PMin : psMin_A PSMax : NONE >
FOLLOWED_BY
 - Task : End_B Min : NONE Max : flwMax_Atomic_A0 :: start and end >
 - Task : Atomic_D Min : NONE Max : flwMax_Atomic_A1 :: only start >PRECEDED_BY
 - Task : Start_B Min : pcdMin_Atomic_A0 Max : pcdMax_Atomic_A0 :: start and end >
 - Task : Atomic_T Min : NONE Max : pcdMax_Atomic_A1 :: only start >END
- **Start_B:** not extended ordinary PMin : NONE PMax : NONE SMin : sMin_B SMax : sMax_B >
FOLLOWED_BY
 - Task : Atomic_A Min : NONE Max : NONE :: only start >
 - Task : Atomic_T Min : NONE Max : NONE :: start and end >END
- **End_B:** not extended ordinary DMin : dMin_B DMax : dMax_B >
PRECEDED_BY
 - Task : Atomic_D Min : pcdMin_End_B0 Max : NONE :: only start >
 - Task : Atomic_T Min : NONE Max : pcdMax_End_B1 :: start and end >END
- **Start_B:** not extended ordinary PMin : NONE PMax : NONE SMin : sMin_B SMax : sMax_B >
END
- **End_B:** not extended ordinary DMin : dMin_B DMax : dMax_B >
END
- **Atomic_D:** not extended ordinary PSMin : NONE PSMax : NONE >
END

Proof obligations

The screenshot displays the Rodin Platform workspace for a project named 'workspace - P1/M1.bum'. The interface is divided into several panes:

- Event-B Explorer (Left):** A tree view showing the project structure. Under 'M1', there are sections for 'Variables', 'Invariants', 'Events', and 'Proof Obligations'. The 'Proof Obligations' section is expanded, listing numerous obligations such as '@precededScheduler_A_Scheduler_B/WD', '@DMinInvEnd_A/WD', and various 'INITIALISATION' and 'ProgressEvent' obligations. Some obligations are marked with a green checkmark, while others have a question mark.
- Code Editor (Center):** Displays the source code for 'TimedContextM1'. The code includes initialization blocks for '@initacCK', '@initactA', and '@initactB'. It defines a 'ProgressEvent' with a 'step' and a 'WHERE' clause involving '@progressGrd'. It also defines 'Start_A', 'End_A', 'Start_B', and 'End_B' events with their respective parameters and conditions.
- Outline (Right):** A pane that is currently empty, displaying the message: 'There is no active editor that provides an outline.'
- Bottom Panel:** Contains an 'Error Log' showing '22 errors, 0 warnings, 0 others'. Other tabs for 'Workspace Log', 'Rodin Problems', 'Properties', 'Tasks', and 'Symbols' are also visible.

ProB Model Checking

The screenshot displays the ProB Model Checker interface with the following components:

- Events Panel:** A list of events with their parameters. The event `Atomic_SpeedSense` is highlighted in blue.

Event	Parameter(s)
▶ Brake_Released	
◻ Brake_Pressed	
▶ ProgressEvent (×3)	1
◻ Atomic_SpeedSense	
◻ Start_Calibrate	
◻ End_Calibrate	
▶ Atomic_SenseBrake	
◻ Atomic_BrakeControl	
◻ Start_BrakeWheel	
◻ End_BrakeWheel	
- State Panel:** A table showing the current state of variables.

Name	Value	Previous value
TimedContextM1		
psMin_SenseBrake	4	4
psMax_SenseBrake	4	4
pcdMax_End_BrakeWheel0	0	0
psMax_SpeedSense	5	5
pcdMax_End_Calibrate0	0	0
pcdMax_Atomic_BrakeCon	0	0
dMax_BrakeWheel	2	2
psMin_SpeedSense	5	5
dMax_Calibrate	4	4
M0		
mode	Brake	Brake
M1		
CK	9	8
Scheduler_BrakeControl	{{(1→5),(2→5)}}	{{(1→5),(2→5)}}
Scheduler_BrakeWheel	{{(1→5),(2→6)}}	{{(1→5),(2→6)}}
Scheduler_Calibrate	{{(1→0),(2→3)}}	{{(1→0),(2→3)}}
Scheduler_SenseBrake	{{(1→5),(2→5)}}	{{(1→5),(2→5)}}
Scheduler_SpeedSense	{{(1→0),(2→0)}}	{{(1→0),(2→0)}}
Formulas		
variables		
constants		
sets		
invariants	T	T
axioms	T	T
event guards		
theory operators		
- History Panel:** A table showing the sequence of events for models M1 and M0.

M1	M0
ProgressEvent(1)	
ProgressEvent(2)	
End_BrakeWheel	
ProgressEvent(1)	
Start_BrakeWheel	
Atomic_BrakeControl	
Atomic_SenseBrake	
Brake_Pressed	Brake_Pressed
ProgressEvent(1)	
ProgressEvent(1)	
End_Calibrate	
ProgressEvent(3)	
Start_Calibrate	
Atomic_SpeedSense	
INITIALISATION	INITIALISATION
SETUP_CONTEXT	
(uninitialised state)	
- Bottom Status Bar:**
 - Left: `Invariants ok` (green background)
 - Right: `No event errors detected` (green background)
 - Right: `Timeout occurred!` (red background)

ABS requirements

N° Contrainte	Modélisation en <i>Event-B</i>
<i>ABS2</i>	<i>periodicalMinMaxTaskScheduler</i> (SenseSpeed, 10 ⁵ , 10 ⁵)
<i>ABS6</i>	<i>periodicalMinMaxTaskScheduler</i> (SenseBrake, 10 ⁴ , 10 ⁴)
<i>ABS3</i>	<i>taskSchedulerMaxDuration</i> (Calibrate, 10 ⁴)
<i>ABS7</i>	<i>taskSchedulerMaxDuration</i> (BrakeWheel, 10 ³)
<i>ABS1</i>	<i>followed</i> (SenseSpeed, Calibrate)
<i>ABS4</i>	<i>followed</i> (SenseBrake, BrakeControl)
<i>ABS5</i>	<i>followed</i> (BrakeControl, BrakeWheel)
<i>ABS3</i>	<i>precededMax</i> (Calibrate, SenseSpeed, 0)
<i>ABS7 et ABS4</i>	<i>precededMax</i> (BrakeControl, SenseBrake, 0)
<i>ABS7 et ABS5</i>	<i>precededMax</i> (BrakeWheel, BrakeControl, 0)
<i>ABS7</i>	<i>preceded</i> (BrakeWheel, SenseBrake)

Thank you

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

Amel Mammam¹, Cabrel Feukeng Momo^{1,2}, **Marc Frappier**², Paul Gibson¹

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