

VECoS'25

A GENERIC EVENT-B THEORY FOR THE FORMALISATION OF THE INTERNATIONAL SYSTEM OF UNITS

This work was supported by a grant from the French national research agency [ANR ANR-19-CE25-0010 \(EBRP Project\)](#).

🎓 18th International Conference on Verification and Evaluation of Computer and Communication Systems
🏢 Centre d'intégration Nano-INNOV - CEA-LIST, Palaiseau, France 📅 5-6 November 2025



Idir AIT SADOUNE
idiraitsadoune@lmf.cnrs.fr

OUTLINE

- The context of the work
- The motivating example
- The proposed approach

[Back to the outline](#) - [Back to the begin](#)

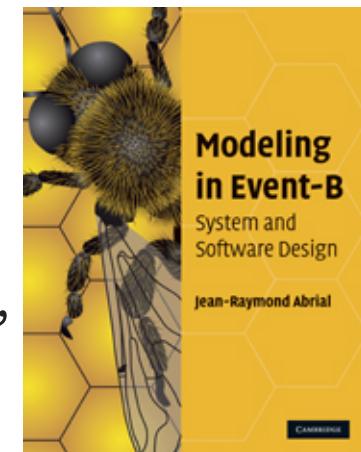
OUTLINE

- The context of the work
- The motivating example
- The proposed approach

[Back to the outline](#) - [Back to the begin](#)

THE EVENT-B METHOD

- The **Event-B method** is an evolution of the **classical B method**.
 - modeling a system by a **set of events** instead of **operations**.
- The **Event-B method** is a **formal method** based on **first-order logic** and **set theory**.
- The **Event-B method** is based on :
 - the notions of pre-conditions and post-conditions (**Hoare**),
 - the **weakest pre-condition** (**Dijkstra**),
 - and the **calculus of substitution** (**Abrial**).



USING EVENT-B METHOD

- The **Rodin** platform (an **Eclipse-based IDE**) is intended to support the construction and verification of **Event-B models**.
- The use of the **Event-B method** has continued to increase.
 - applied to various applications and domains.
 - railway, automotive, aeronautics, cybersecurity, nuclear-energy, ...
- The **Event-B method** is adapted to analyse **discrete systems**.
 - offers the possibility of modelling **discrete behaviors**.

THE EVENT-B METHOD

CONTEXT ctx_1
EXTENDS ctx_2

SETS s
CONSTANTS c
AXIOMS
 $A(s, c)$
THEOREMS
 $T(s, c)$
END

MACHINE mch_1
REFINES mch_2
SEES ctx_i

VARIABLES v
INVARIANTS
 $I(s, c, v)$
THEOREMS
 $T(s, c, v)$
EVENTS
 $[events_list]$
END

$event \triangleq$
any x
where
 $G(s, c, v, x)$
then
 $BA(s, c, v, x, v')$
end

$$\begin{aligned} A(s, c) &\vdash T(s, c) \\ A(s, c) \wedge I(s, c, v) &\vdash T(s, c, v) \\ A(s, c) \wedge I(s, c, v) \wedge G(s, c, v, x) &\vdash \exists v'. BA(s, c, v, x, v') \\ A(s, c) \wedge I(s, c, v) \wedge G(s, c, v, x) \wedge BA(s, c, v, x, v') &\vdash I(s, c, v') \\ \dots \end{aligned}$$

THE EVENT-B METHOD

STATIC TYPE CHECKING

- Event-B supports static type checking using tools such as Rodin or AtelierB.
- These tools generate proof obligations (POs) to check the correct use of arithmetic operations (Well-Defined proof obligations - WD POs).
- WD POs ensure that expressions (axioms, theorems, invariants, guards, actions, etc.) are mathematically well-defined.
- Example → for the expression $x \div y$, a WD PO ensures that $y \neq 0$.

THE EVENT-B METHOD

THE THEORY PLUGIN

- **Theory Plug-in** provides capabilities to [extend the Event-B mathematical language](#) and [the Rodin proving infrastructure](#).
- An **Event-B theory** can contain :
 - new datatype definitions,
 - new polymorphic operator definitions,
 - axiomatic definitions,
 - theorems,
 - associated rewrite and inference rules.

- Michael J. Butler and Issam Maamria.
Practical theory extension in Event-B. Theories of Programming and Formal Methods. 2013.
- Thai Son Hoang, Laurent Voisin, Asieh Salehi, Michael J. Butler, Toby Wilkinson, and Nicolas Beauger.
Theory plug-in for Rodin 3.x. CoRR, abs/1701.08625, 2017.

THE EVENT-B METHOD

THE THEORY PLUGIN

THEORY thy_1
IMPORT thy_2

DATATYPES

DT_1, \dots, DT_n

OPERATORS

OP_{11}, \dots, OP_{1n}

AXIOMATIC DEFINITIONS

operators

OP_{21}, \dots, OP_{2n}

axioms

A

THEOREMS

T

PROOF RULES

PR

END

VECoS'25

CONTEXT ctx_1
EXTENDS ctx_2

SETS s

CONSTANTS c

AXIOMS

$A(s, c)$

THEOREMS

$T(s, c)$

END

MACHINE mch_1
REFINES mch_2
SEES ctx_i

VARIABLES v

INVARIANTS

$I(s, c, v)$

THEOREMS

$T(s, c, v)$

EVENTS

$[events_list]$

END

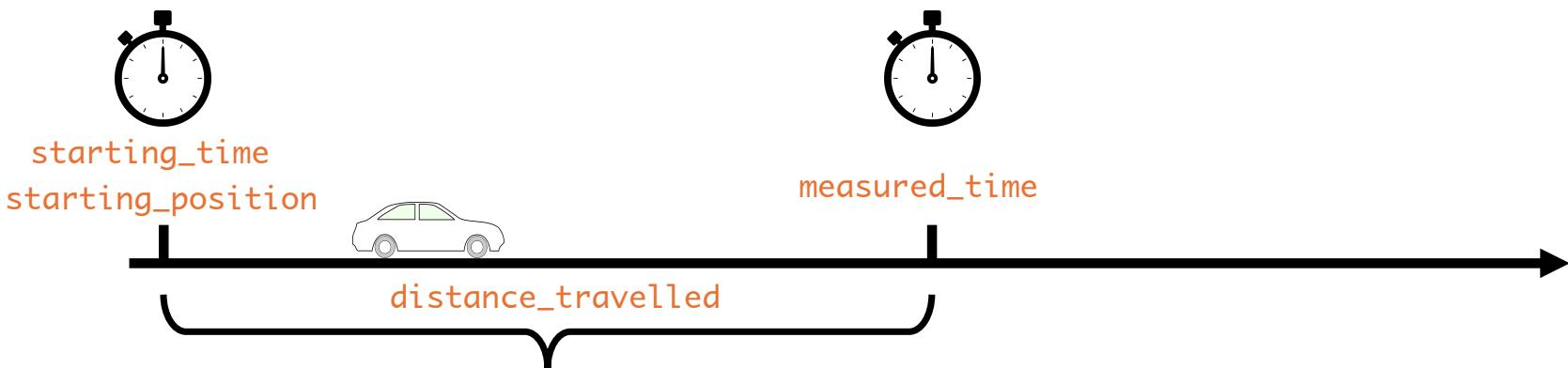
OUTLINE

- The context of the work
- The motivating example
- The proposed approach

[Back to the outline](#) - [Back to the begin](#)

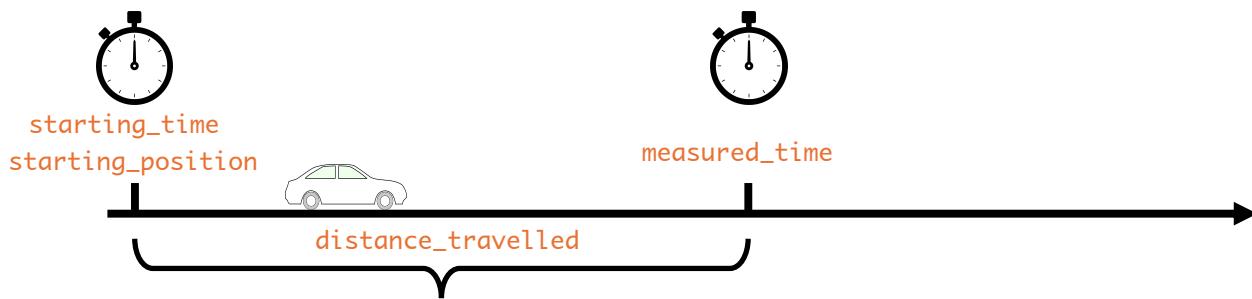
A SIMPLE EXAMPLE

System that continuously calculates **a moving object's speed**



- Analysing **two functional properties**:
 - PROP-1 : the speed of the moving object is equal to the *distance_travelled* divided by the *measured_time* ($v = d/t$).
 - PROP-2 : when the *distance_travelled* is strictly positive, the *speed* of the moving object must also be strictly positive.
 - the object moves when its *speed* is different from zero.

CHALLENGES IN MODELLING CPS SYSTEMS



- More generally, **Cyber-Physical Systems (CPS)** models often require **variables/expressions**, formalising **measurements/physics and mechanics laws**.
- **Event-B** does not support **physical unit annotations** for such variables.
- Formal verification of CPS systems requires a physical measurement model, e.g. **the International System of Units (SI)**.
- Using explicit units improves rigour by ensuring unit compatibility in **VECoS'25** computations.

PROPOSED APPROACH

- **Objective**
 - Formally annotate numerical variables with measurement units in Event-B.
 - Provide automatic checking of correct unit usage in arithmetic expressions.
 - Define Well-Defined Proof Obligations (WD POs) to ensure unit consistency.
 - Example: $b = v/2a_{max,brake}$
→ must ensure that the unit of b matches that of $v/2a_{max,brake}$.
- **Proposal**
 - Develop a measurement units theory using the Theory plugin.
 - Extend the Event-B type-checking system to handle reasoning about measurement units.
 - Introduce a formal method for annotating Event-B variables with their associated units of measurement.

OUTLINE

- The context of the work
- The motivating example
- The proposed approach

[Back to the outline](#) - [Back to the begin](#)

THANK YOU

[PDF version of the slides](#)

[Back to the begin](#) - [Back to the outline](#)