

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-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
- Revisiting the motivating example
- Conclusion and future works

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

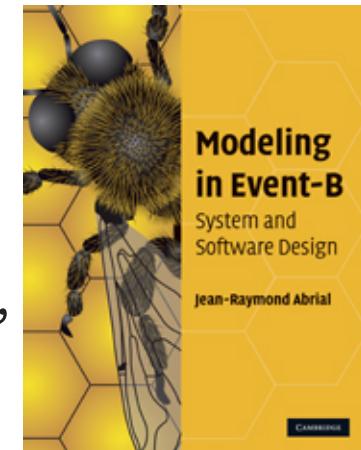
OUTLINE

- The context of the work
- The motivating example
- The proposed approach
- Revisiting the motivating example
- Conclusion and future works

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

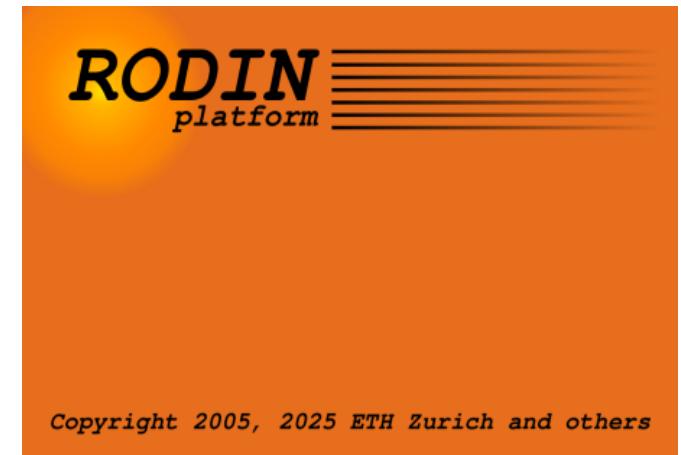
THE EVENT-B METHOD

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



THE EVENT-B METHOD

MODELS AND PROOF OBLIGATIONS

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 $\hat{=}$
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.

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

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

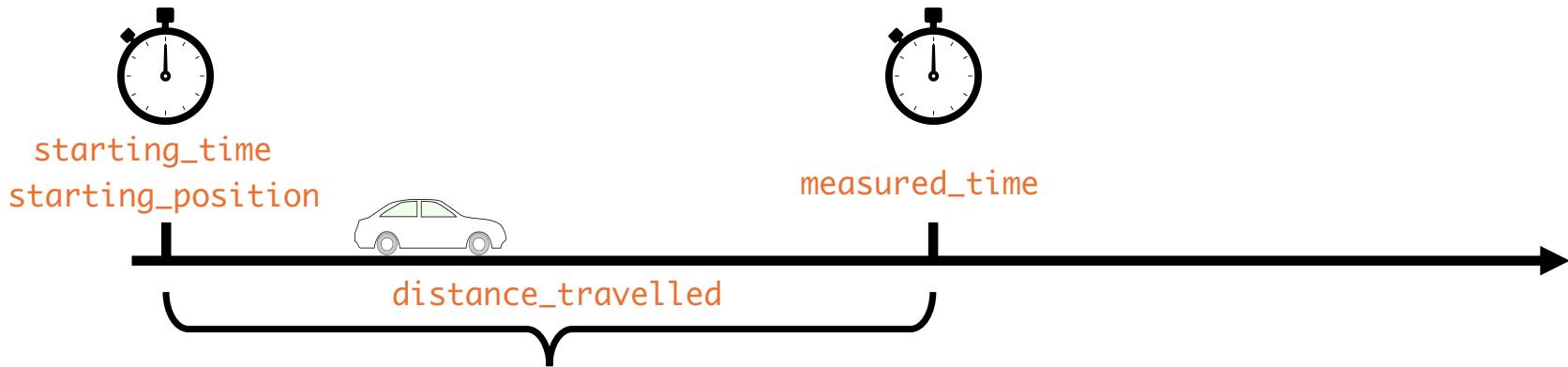
OUTLINE

- The context of the work
- The motivating example
- The proposed approach
- Revisiting the motivating example
- Conclusion and future works

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

A SIMPLE EXAMPLE

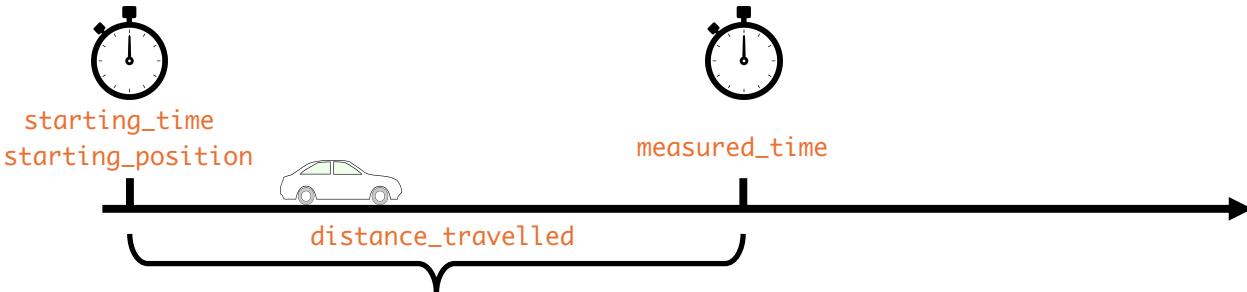
System that continuously calculates a moving object's speed



Analysing two functional properties:

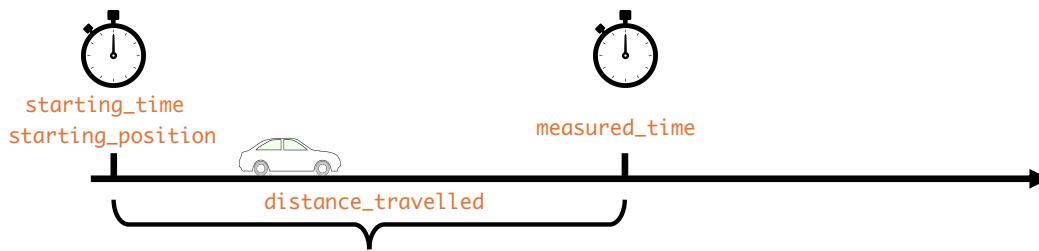
- PROP-1 : the velocity of the moving object is equal to the `distance_travelled` divided by the `measured_time` ($v = d/t$).
- PROP-2 : when the `distance_travelled` is strictly positive, the `speed` of the moving object must also be strictly positive.
 - the object moves when its `speed` is different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

THE EVENT-B MODEL



```
MACHINE mch_integer_version
...
INVARIANTS
@inv1: distance_travelled ∈ ℑ           // km
@inv2: measured_time ∈ ℑ₁                // s
@inv3: speed ∈ ℑ                         // km/h
@inv4: starting_position ∈ ℑ
@inv5: starting_time ∈ ℑ
@inv6: speed = distance_travelled ÷ measured_time // PROP-1
@inv7: distance_travelled > 0 ⇒ speed > 0 // PROP-2
```

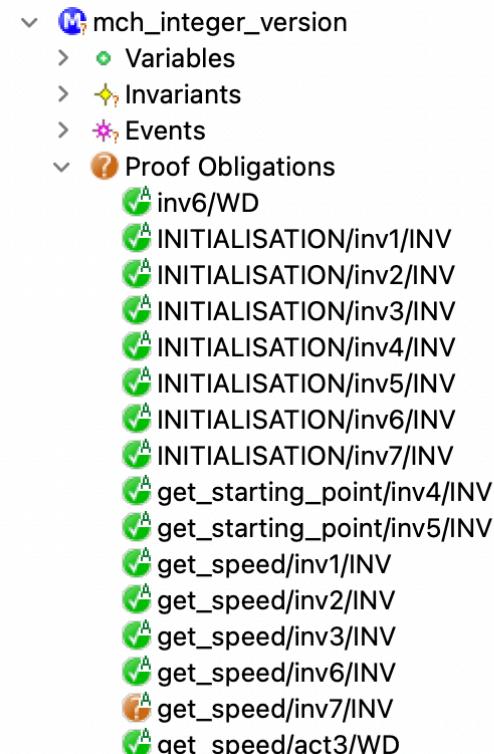
THE EVENT-B MODEL



```
MACHINE mch_integer_version
...
EVENTS
...
get_speed ≡
  any p t
  where
    @grd1: p ∈ ℑ_1 ∧ p > starting_position
    @grd2: t ∈ ℜ_1 ∧ t > starting_time
  then
    @act1: distance_travelled := p - starting_position
    @act2: measured_time := t - starting_time
    @act3: speed := (p - starting_position) ÷ (t - starting_time)
  end
END
```

GENERATED AND PROVEN POS

- All POs are green except the one for maintaining the `@inv7` invariant by the `get_speed` event.
- PROP 2 → $distance_travelled \neq 0$ when $speed \neq 0$.
 - the value of $distance_travelled$ can be $<$ the value of $measured_time$.
 - the value of $speed$ can be $= 0$ ($distance_travelled \div measured_time$) while $distance_travelled \neq 0$
- No possibility to check the consistency of formulas annotated with measurement units.
 - Example: is the unit of $speed$ (km/h) the same with the unit of the expression $distance_travelled \div measured_time$ (km ÷ s) ?



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 **measurements unit annotations** for such variables and using **integer** variables is not sufficient to handle **small values** ($0 < v < 1$).
 - converting from the smallest point of view to the most significant ones
 - from **Milli** to **Kilo**, for example
- Formal verification of CPS systems requires a physical measurement **model**, e.g. **the International System of Units (SI)**.
- Using **explicit units** improves the **CPS validation process** by ensuring **unit compatibility** in arithmetic expressions.

THE OBJECTIVES

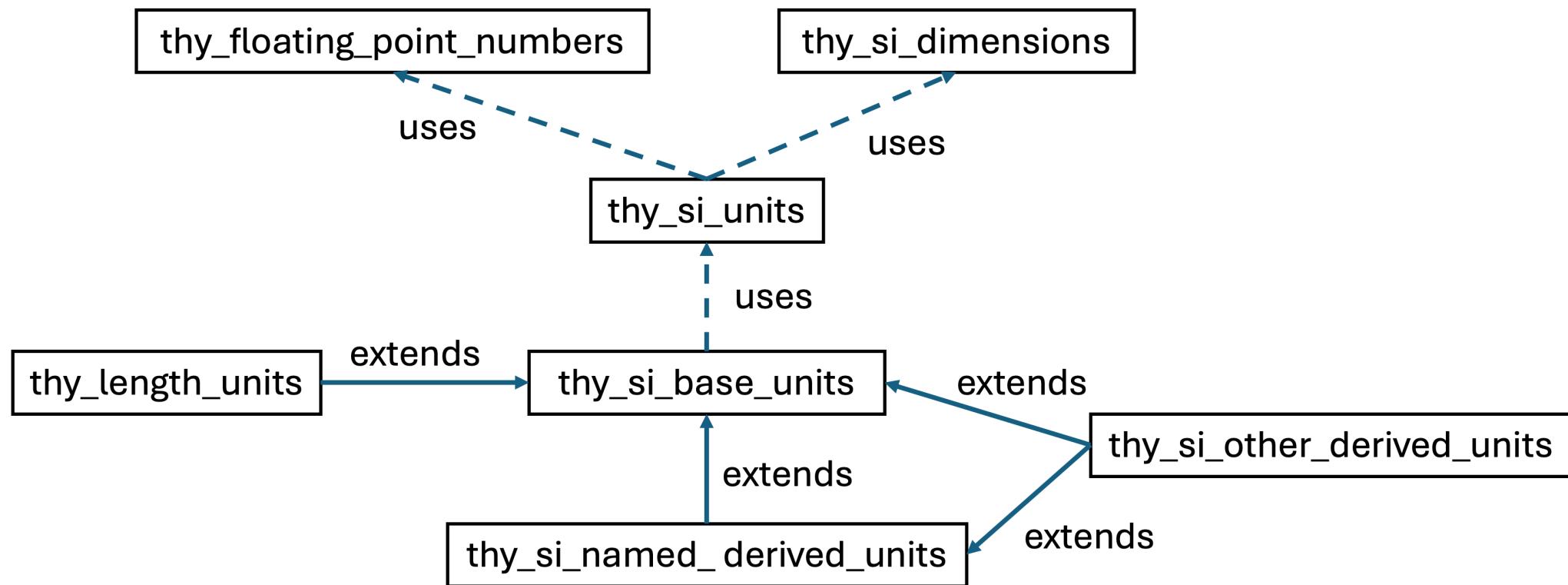
- New syntax to formally annotate Event-B variables with measurement units.
- New generic arithmetic operators for the annotated variables.
- New Well-Defined Proof Obligations (WD POs) to ensure unit consistency.
- Automatic checking of correct unit usage in arithmetic expressions.
- Example: $d = v/2 a$
→ must ensure that the unit of d matches that of $v/2 a$.

OUTLINE

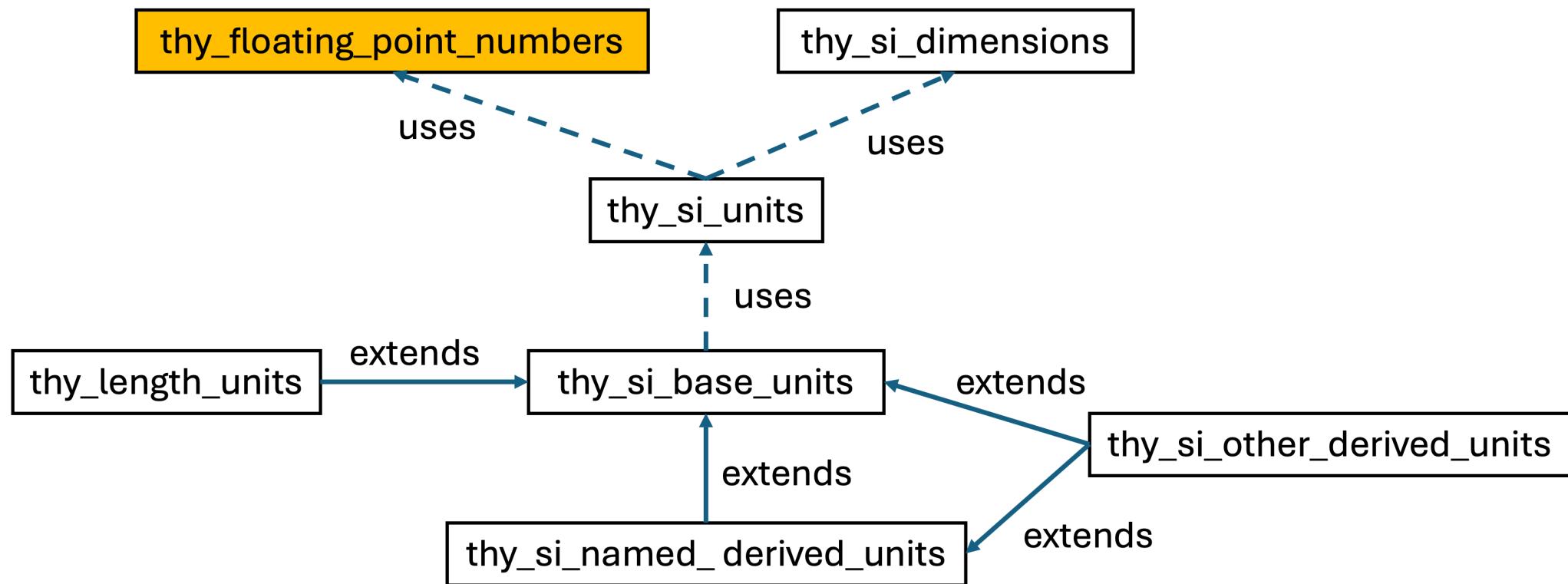
- The context of the work
- The motivating example
- The proposed approach
- Revisiting the motivating example
- Conclusion and future works

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

PROPOSED APPROACH



PROPOSED APPROACH



FLOATING-POINT NUMBERS

$$x = 3.14159265359 = \underbrace{314159265359}_{\text{significand}} \times \underbrace{10}_{\text{base}}^{-11}^{\text{exponent}}$$

We have chosen that the base always equals ten in our models.

$$x = s(x) \times 10^{e(x)}$$

- The proposed theory **does not model limited precision**.
- The **operators** defined in the theory involve **no precision loss**.

THE FLOATING-POINT NUMBERS THEORY

```
THEORY thy_floating_point_numbers
DATATYPES
  FLOAT_Type ≡ NEW_FLOAT(s ∈ ℤ, e ∈ ℤ) //  $x = s(x) \times 10^{e(x)}$ 
OPERATORS
  F0 ≡ NEW_FLOAT(0,0) // 0
  F1 ≡ NEW_FLOAT(1,0) //  $10^0 = 1$ 
  ...
  MILLI ≡ NEW_FLOAT(1,-3) //  $10^{-3}$ 
  CENTI ≡ NEW_FLOAT(1,-2) //  $10^{-2}$ 
  DECI ≡ NEW_FLOAT(1,-1) //  $10^{-1}$ 
  DECA ≡ NEW_FLOAT(1,1) //  $10^1$ 
  HECTO ≡ NEW_FLOAT(1,2) //  $10^2$ 
  KILO ≡ NEW_FLOAT(1,3) //  $10^3$ 
  ...
  eq(x ∈ FLOAT_Type, y ∈ FLOAT_Type) INFIX ≡ ...
  gt(x ∈ FLOAT_Type, y ∈ FLOAT_Type) INFIX ≡ ...
  ...
  plus(x ∈ FLOAT_Type, y ∈ FLOAT_Type) INFIX ≡ ...
  mult(x ∈ FLOAT_Type, y ∈ FLOAT_Type) INFIX ≡ ...
  ...
END
```

THE FLOATING-POINT NUMBERS THEORY

THEORY thy_floating_point_numbers

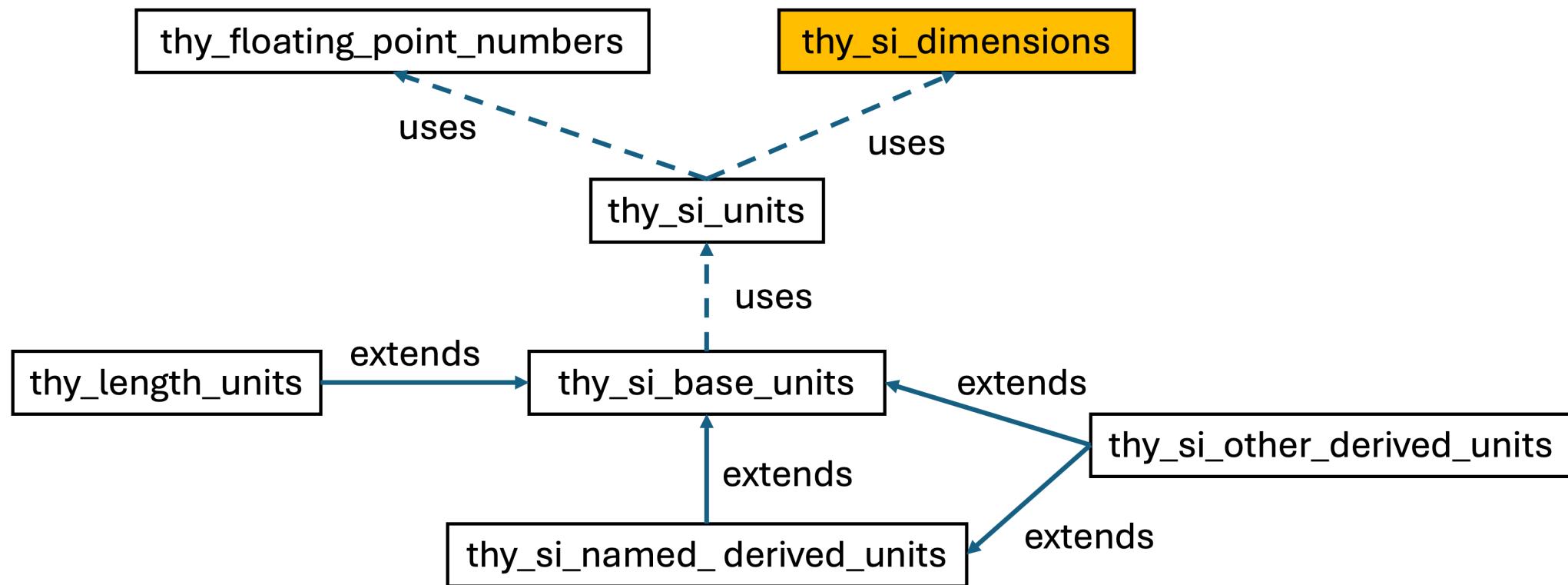
...

THEOREMS

- @thm1: $\forall x, y \cdot (\dots \Rightarrow x = y \Leftrightarrow y = x)$
- @thm2: $\forall x \cdot (\dots \Rightarrow x \geq x \wedge x \leq x)$
- @thm3: $\forall x, y \cdot (\dots x \leq y \wedge y \leq x \Rightarrow x = y)$
- @thm4: $\forall x, y \cdot (\dots \Rightarrow x \leq y \vee y \leq x)$
- @thm5: $\forall x, y, z \cdot (\dots x \leq y \wedge y \leq z \Rightarrow x \leq z)$
- @thm6: $\forall x, y, z \cdot (\dots x \leq y \Rightarrow (x + z) \leq (y + z))$
- @thm7: $\forall x, y, z \cdot (\dots x \leq y \Rightarrow (x \cdot z) \leq (y \cdot z))$
- @thm8: $\forall x \cdot (\dots \Rightarrow x + F0 = x)$
- @thm9: $\forall x, y \cdot (\dots \Rightarrow x + y = y + x)$
- @thm10: $\forall x, y \cdot (\dots \Rightarrow x + \text{neg}(y) = y - x)$
- @thm11: $\forall x \cdot (\dots \Rightarrow x - F0 = x)$
- @thm12: $\forall x \cdot (\dots \Rightarrow x - x = F0)$
- @thm13: $\forall x \cdot (\dots \Rightarrow x \cdot F0 = F0)$
- @thm14: $\forall x \cdot (\dots \Rightarrow x \cdot F1 = x)$
- @thm15: $\forall x, y \cdot (\dots \Rightarrow x \cdot y = y \cdot x)$
- @thm16: $\forall x \cdot (\dots \Rightarrow \text{inv}(x) = F1 / x)$
- @thm17: $\forall x \cdot (\dots \Rightarrow x / F1 = x)$
- @thm18: $\forall x \cdot (\dots \Rightarrow x / x = F1)$
- @thm19: $\forall x \cdot (\dots \Rightarrow x \cdot \text{inv}(x) = F1)$

...

PROPOSED APPROACH



DIMENSIONS FORMALISATION

- **SI System** → a coherent system of measurement based on **seven base quantities**.
- **Base Quantities:**
Time (T), Length (L), Mass (M), Electric current (I), Thermodynamic temperature (Θ), Amount of substance (N), Luminous intensity (J).
- Each **base quantity** corresponds to **a base dimension**.
- **Physical quantities** are organized in a **system of dimensions**.
- **The dimension** of any **quantity** Q is expressed as:

$$\dim Q = T^\alpha L^\beta M^\gamma I^\delta \Theta^\varepsilon N^\zeta J^\eta$$

➡ the exponents $\alpha, \beta, \gamma, \delta, \varepsilon, \zeta$ and η are **the dimensional exponents**
(can be positive, negative, or zero).

DIMENSIONS FORMALISATION

DATATYPES

```
SI_DIMENSION_Type ≡ SI_DIMENSION(  
    exp_d1 ∈ ℤ, // length dimension  
    exp_d2 ∈ ℤ, // mass dimension  
    exp_d3 ∈ ℤ, // time dimension  
    exp_d4 ∈ ℤ, // electric current dimension  
    exp_d5 ∈ ℤ, // thermodynamic temperature dimension  
    exp_d6 ∈ ℤ, // amount of substance dimension  
    exp_d7 ∈ ℤ) // luminous intensity dimension
```

OPERATORS

```
L_DIM (exp_d ∈ ℤ) ≡ SI_DIMENSION(exp_d,0,0,0,0,0,0) // length quantity
```

```
M_DIM (exp_d ∈ ℤ) ≡ SI_DIMENSION(0,exp_d,0,0,0,0,0) // mass quantity
```

```
T_DIM (exp_d ∈ ℤ) ≡ SI_DIMENSION(0,0,exp_d,0,0,0,0) // time quantity
```

...

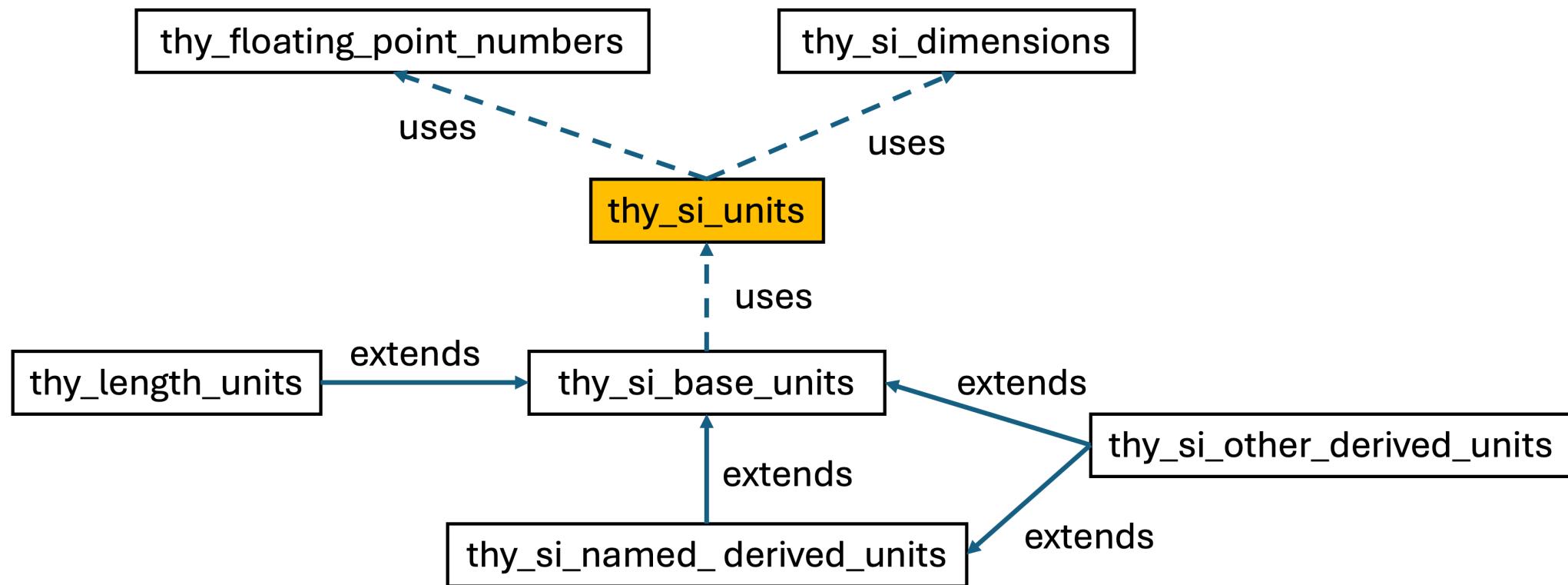
```
DIM_MULT(dim1 ∈ SI_DIMENSION_Type, dim2 ∈ SI_DIMENSION_Type) ≡  
    SI_DIMENSION(..., exp_di(dim1)+exp_di(dim2)), ...)
```

```
DIM_DIV(dim1 ∈ SI_DIMENSION_Type, dim2 ∈ SI_DIMENSION_Type) ≡  
    SI_DIMENSION(..., exp_di(dim1)-exp_di(dim2)), ...)
```

```
HAVE_SAME_EXP_DIMENSIONS(dim1 ∈ SI_DIMENSION_Type, dim2 ∈ SI_DIMENSION_Type) ≡  
    dim1=dim2
```

...

PROPOSED APPROACH



UNIT OF A QUANTITY

- A **unit** is formalised using a product of a **multiplier** with **dimension** shifted by an **offset**:

$$unit = multiplier \times dimension + offset$$

- **Multiplier**
 - represents **prefixes** applied to base units.
 - **examples:** milli, centi, deci, deca, kilo, etc.
 - used to express **multiples** or **submultiples** of a **base unit** (e.g., $1\text{km} = 1000\text{m}$).
- **Offset**
 - defines a **shift** relative to a **base unit**.
 - **example:** the **degree Celsius** is offset by 273.15 from the **Kelvin (K)** unit.
 - useful for units that are not directly proportional to their base unit.

UNIT OF A QUANTITY

DATATYPES

$\text{SI_UNIT_Type} \hat{=} \text{SI_UNIT}(\text{multiplier} \in \text{FLOAT_Type}, \text{dimension} \in \text{SI_DIMENSION_Type}, \text{offset} \in \text{FLOAT_Type})$
 $\text{MEASURE_Type} \hat{=} \text{MEASURE}(\text{value} \in \text{FLOAT_Type}, \text{unit} \in \text{SI_UNIT_Type})$

OPERATORS

$\text{UNIT_MULT}(u_1 \in \text{SI_UNIT_Type}, u_2 \in \text{SI_UNIT_Type}) \hat{=}$
 $\text{SI_UNIT}(\text{multiplier}(u_1) \text{ mult } \text{multiplier}(u_2), \text{DIM_MULT}(\text{dimension}(u_1), \text{dimension}(u_2)), \text{F0})$
 $\text{UNIT_DIV}(u_1 \in \text{SI_UNIT_Type}, u_2 \in \text{SI_UNIT_Type}) \hat{=}$
 $\text{SI_UNIT}(\text{multiplier}(u_1) \text{ div } \text{multiplier}(u_2), \text{DIM_DIV}(\text{dimension}(u_1), \text{dimension}(u_2)), \text{F0})$

...

$\text{SI_MEASURE_Type}(t \in \text{SI_UNIT_Type}) \hat{=} \{x : x \in \text{MEASURE_Type} \wedge \text{unit}(x) = t \mid x\}$

$\text{HAVE_THE_SAME_UNIT}(m_1 \in \text{MEASURE_Type}, m_2 \in \text{MEASURE_Type}) \hat{=} \text{unit}(m_1) = \text{unit}(m_2)$

$\text{SI_EQ}(m_1 \in \text{MEASURE_Type}, m_2 \in \text{MEASURE_Type}) \hat{=}$

wd : HAVE_THE_SAME_UNIT(m1, m2)
def : value(m1) eq value(m2)

...

$\text{SI_PLUS}(m_1 \in \text{MEASURE_Type}, m_2 \in \text{MEASURE_Type}) \hat{=}$

wd : HAVE_THE_SAME_UNIT(m1, m2)
def : MEASURE(value(m1) plus value(m2), unit(m1))

...

$\text{SI_MULT}(m_1 \in \text{MEASURE_Type}, m_2 \in \text{MEASURE_Type}) \hat{=}$

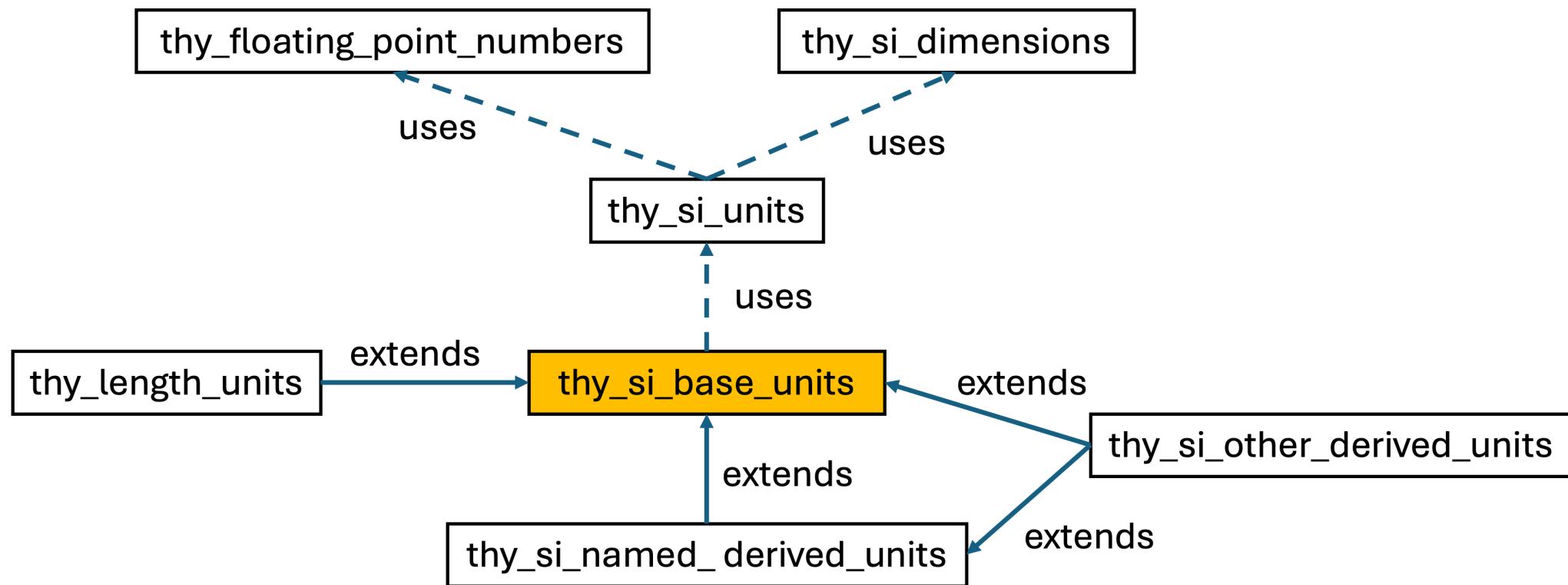
MEASURE(value(m1) mult value(m2), UNIT_MULT(unit(m1), unit(m2)))

...

$\text{SI_CONVERT}(u \in \text{SI_UNIT_Type}, m \in \text{MEASURE_Type}) \hat{=}$

wd : HAVE_SAME_EXP_DIMENSIONS(dimension(unit(m)), dimension(u))
def : // v2 = (v1 - o1) × (m1 × d1)/(m2 × d2) + o2

PROPOSED APPROACH

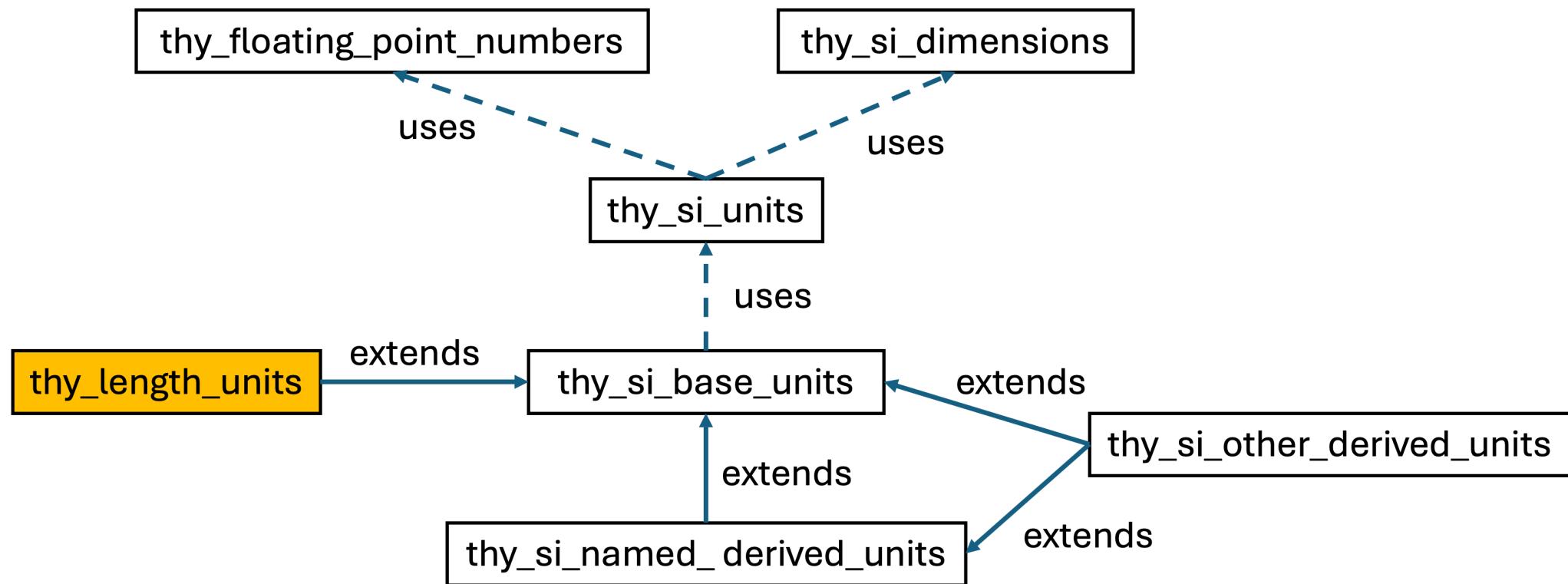


SI BASE UNITS FORMALISATION

OPERATORS

```
METRE_UNIT  $\hat{=}$  SI_UNIT(F1, L_DIM(1), F0) // m
KILO_GRAM_UNIT  $\hat{=}$  SI_UNIT(KILO, M_DIM(1), F0) // kg
SECOND_UNIT  $\hat{=}$  SI_UNIT(F1, T_DIM(1), F0) // s
AMPERE_UNIT  $\hat{=}$  SI_UNIT(F1, I_DIM(1), F0) // A
KELVIN_UNIT  $\hat{=}$  SI_UNIT(F1, O_DIM(1), F0) // K
MOLE_UNIT  $\hat{=}$  SI_UNIT(F1, N_DIM(1), F0) // mol
CANDELA_UNIT  $\hat{=}$  SI_UNIT(F1, J_DIM(1), F0) // cd
```

PROPOSED APPROACH

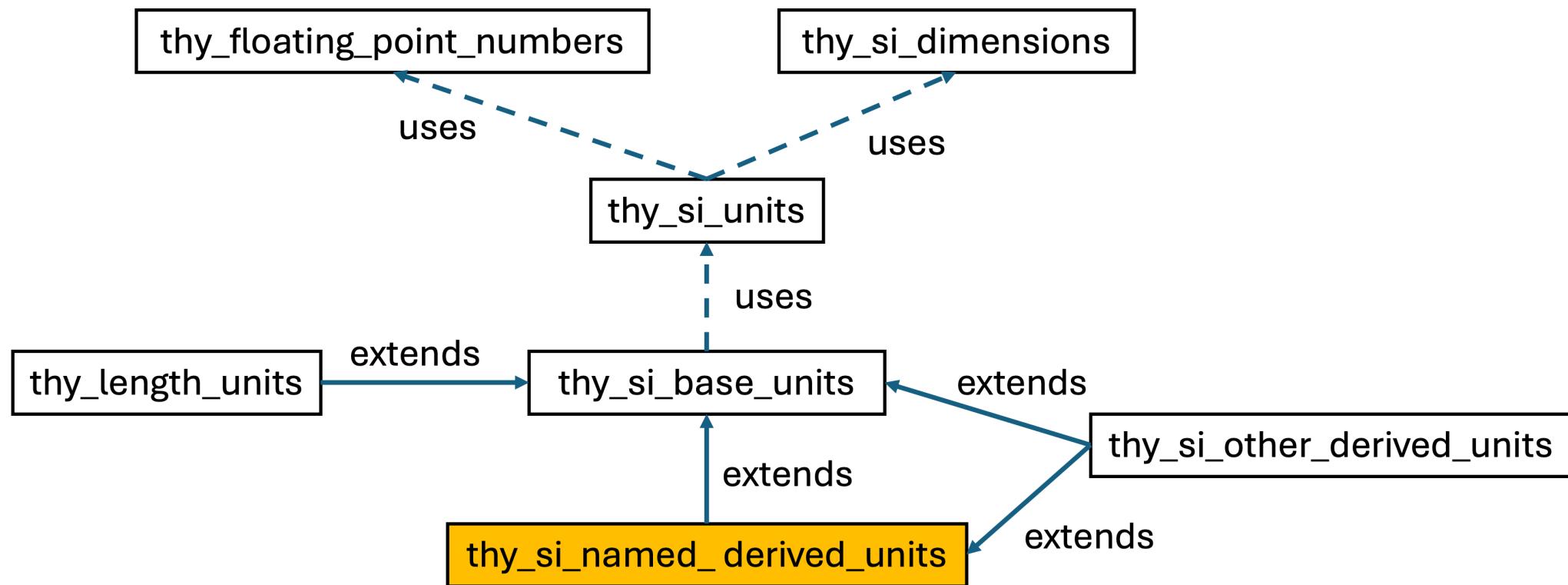


LENGTH UNITS FORMALISATION

OPERATORS

```
MILLI_METRE_UNIT  $\hat{=}$  SI_UNIT(MILLI, L_DIM(1), F0) // mm  
CENTI_METRE_UNIT  $\hat{=}$  SI_UNIT(CENTI, L_DIM(1), F0) //cm  
DECI_METRE_UNIT  $\hat{=}$  SI_UNIT(DECI, L_DIM(1), F0) //dm  
DECA_METRE_UNIT  $\hat{=}$  SI_UNIT(DECA, L_DIM(1), F0) //dam  
HECTO_METRE_UNIT  $\hat{=}$  SI_UNIT(HECTO, L_DIM(1), F0) //hm  
KILO_METRE_UNIT  $\hat{=}$  SI_UNIT(KILO, L_DIM(1), F0) //km  
...
```

PROPOSED APPROACH



THE NAMED DERIVED UNIT FORMALISATION

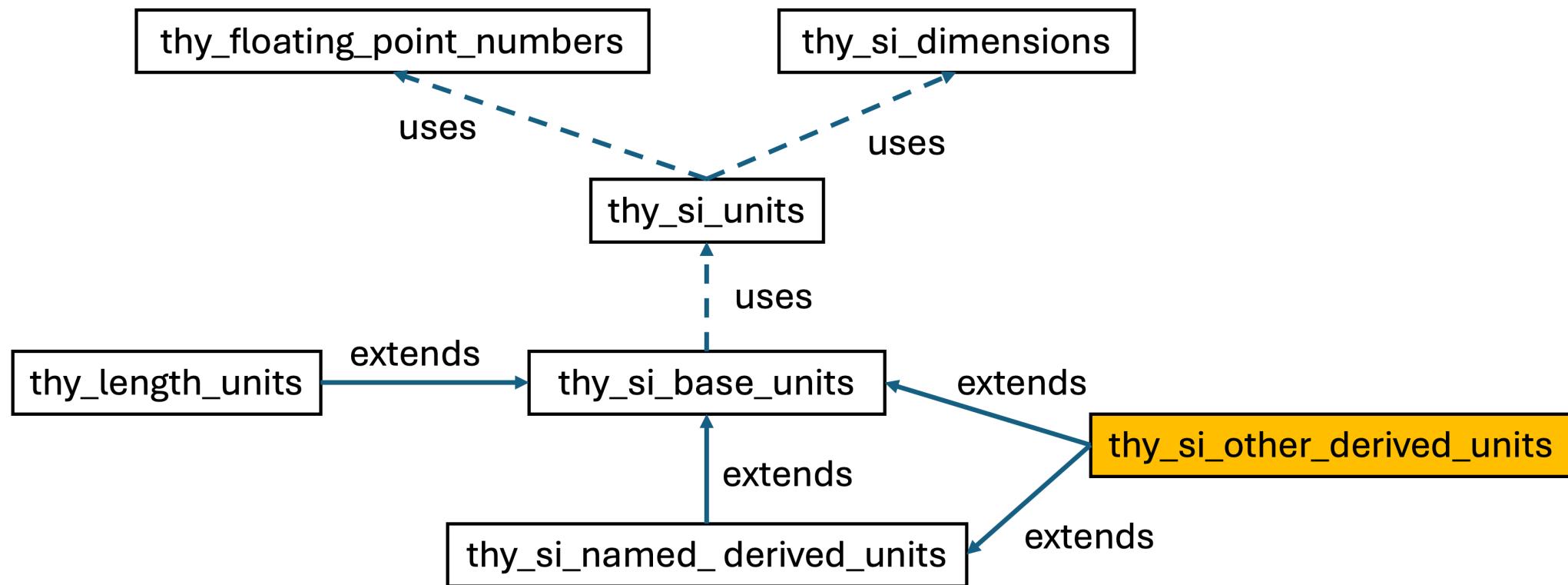
- **Derived units** → defined as products of powers of base units (dimensions).
- **Coherent derived units** → occur when the numerical factor in the product is one.
- **Special coherent derived units** → 22 units in the SI have **special names**, e.g. **radian, hertz, coulomb, degree Celsius, etc.**
- These 22 named units are defined by **combining the seven base units**.
- These 22 coherent derived units + 7 base units form the core of the **International System of Units (SI)**.

OPERATORS

```
HERTZ_UNIT ≡ // 1/s
    UNIT_INV(SECOND_UNIT)
COULOMB_UNIT ≡ // s A
    UNIT_MULT(SECOND_UNIT, AMPERE_UNIT)
NEWTON_UNIT ≡ // kg m / s^2
    UNIT_MULT(KILO_GRAM_UNIT, UNIT_DIV(METRE_UNIT, UNIT_MULT(SECOND_UNIT, SECOND_UNIT)))
...

```

PROPOSED APPROACH



THE OTHER DERIVED UNIT FORMALISATION

The **seven base units** and **twenty-two units with special names** may be combined to express the units of other derived physical quantities.

OPERATORS

```
SQUARE_METRE_UNIT  $\hat{=}$  //area m2
    UNIT_MULT(METRE_UNIT, METRE_UNIT)
CUBIC_METRE_UNIT  $\hat{=}$  // volume m3
    UNIT_MULT(SQUARE_METRE_UNIT, METRE_UNIT)
METRE_PER_SECOND_UNIT  $\hat{=}$  // speed, velocity m/s
    UNIT_DIV(METRE_UNIT, SECOND_UNIT)
METRE_PER_SECOND_SQUARED_UNIT  $\hat{=}$  // acceleration m/s2
    UNIT_DIV(METRE_UNIT, UNIT_MULT(SECOND_UNIT, SECOND_UNIT))
...
COULOMB_PER_CUBIC_METRE_UNIT  $\hat{=}$  // electric charge density
    UNIT_DIV(COULOMB_UNIT, CUBIC_METRE_UNIT) // coulomb/m3 = s.A/m3
...
```

NON-SI UNITS FORMALISATION

The most used **Non-SI units** that accepted for use with the SI Units and that we can find in [the official SI Brochure](#), can be formalised as a **SI_UNIT_Type** datatype

OPERATORS

$$\text{NONSI_UNIT}(v \in \text{FLOAT_Type}, u \in \text{SI_UNITE_Type}) \hat{=} \\ \text{SI_UNIT}(v \text{ mult } \text{multiplier}(u), \text{dimension}(u), \text{offset}(u))$$

$$\text{MINUTE_UNIT} \hat{=} \text{NONSI_UNIT}(\text{FLOAT}(60), \text{SECOND_UNIT})$$

$$\text{HOUR_UNIT} \hat{=} \text{NONSI_UNIT}(\text{FLOAT}(3600), \text{SECOND_UNIT})$$

$$\text{HECTARE_UNIT} \hat{=} \text{NONSI_UNIT}(\text{FLOAT}(10000), \text{SQUARE_METRE_UNIT})$$

$$\text{LITRE_UNIT} \hat{=} \text{NONSI_UNIT}(\text{NEW_FLOAT}(1, -3), \text{CUBIC_METRE_UNIT})$$

...

OUTLINE

- The context of the work
- The motivating example
- The proposed approach
- Revisiting the motivating example
- Conclusion and future works

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

REFINEMENT BASED APPROACH

We have used the **Event-B refinement** to deal separately with the problem of **using small values** and the problem of **correctly using measurement units**.

- **Refinement** is an excellent solution to decompose a complex proof.

REVISITING OUR EXAMPLE I

```
MACHINE mch_floating_point_version
...
INVARIANTS
@inv1: distance_travelled ∈ PFL0AT_Type
@inv2: measured_time ∈ PFL0AT1_Type
@inv3: speed ∈ PFL0AT_Type
@inv4: starting_position ∈ PFL0AT_Type
@inv5: starting_time ∈ PFL0AT_Type
@inv6: div_WD(distance_travelled, measured_time)
@inv7: speed eq distance_travelled div measured_time
@inv8: distance_travelled gt F0 ⇒ speed gt F0
...
END
```

REVISITING OUR EXAMPLE II

```
MACHINE mch_floating_point_version
...
EVENTS
...
get_speed  $\hat{=}$ 
  any p t
  where
    @grd1: p  $\in$  PFLOAT_Type  $\wedge$  p gt starting_position
    @grd2: t  $\in$  PFLOAT_Type  $\wedge$  t gt starting_time
    @grd3: div_WD(p minus starting_position, t minus starting_time)
  then
    @act1: distance_travelled := p minus starting_position
    @act2: measured_time := t minus starting_time
    @act3: speed := (p minus starting_position) div (t minus starting_time)
  end
END
```

GENERATED AND PROVEN POs

- ✓ M mch_floating_point_speed
 - > ● Variables
 - > ✦ Invariants
 - > ✯ Events
 - ✓ Proof Obligations
 - ✓ inv6/WD
 - ✓ inv7/WD
 - ✓ INITIALISATION/inv1/INV
 - ✓ INITIALISATION/inv2/INV
 - ✓ INITIALISATION/inv3/INV
 - ✓ INITIALISATION/inv4/INV
 - ✓ INITIALISATION/inv5/INV
 - ✓ INITIALISATION/inv6/INV
 - ✓ INITIALISATION/inv7/INV
 - ✓^A INITIALISATION/inv8/INV
 - ✓^A get_starting_point/inv4/INV
 - ✓^A get_starting_point/inv5/INV
 - ✓ get_speed/grd5/WD
 - ✓ get_speed/inv1/INV
 - ✓ get_speed/inv2/INV
 - ✓ get_speed/inv3/INV
 - ✓^A get_speed/inv6/INV
 - ✓ get_speed/inv7/INV
 - ✓ get_speed/inv8/INV
 - ✓ get_speed/act3/WD

- All generated POs have been proven.
- The **get_speed/inv8/INV** PO becomes ✓.
 - ➡ thanks to handling small values ($[0..1]$),
 - ➡ and to the new arithmetic operators specifications.

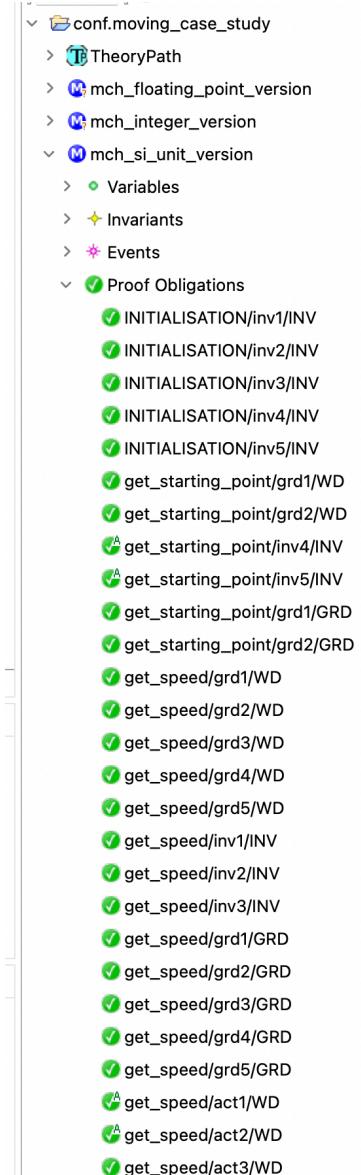
The floating-point numbers theory is more suitable than the basic integers of Event-B.

THE ANNOTATED MODEL

```
MACHINE mch_annotated_version REFINES mch_floating_point_version
...
INVARIANTS
@inv1: si_distance_travelled ∈ SI_MEASURE_Type(METRE_UNIT)
@inv2: si_measured_time ∈ SI_MEASURE_Type(SECOND_UNIT)
@inv3: si_speed ∈ SI_MEASURE_Type(METRE_PER_SECOND_UNIT)
@inv4: si_starting_position ∈ SI_MEASURE_Type(METRE_UNIT)
@inv5: si_starting_time ∈ SI_MEASURE_Type(SECOND_UNIT)
@glueing-1: value(si_distance_travelled) = distance_travelled
@glueing-2: value(si_measured_time) = measured_time
@glueing-3: value(si_speed) = speed
...
EVENTS
...
get_speed ^=
  any si_p si_t
  where
    @grd1: si_p ∈ SI_MEASURE_Type(METRE_UNIT) ∧ si_p SI_GT si_starting_position
    @grd2: si_t ∈ SI_MEASURE_Type(SECOND_UNIT) ∧ si_t SI_GT si_starting_time
    @grd3: div_WD(...)
  with
    value(si_p) = p ∧ value(si_t) = t
  then
    @act1: si_distance_travelled := si_p SI_MINUS si_starting_position
    @act2: si_measured_time := si_t SI_MINUS si_starting_time
    @act3: si_speed := (si_p SI_MINUS si_starting_position) SI_DIV (si_t SI_MINUS si_starting_time)
  end
END
```

THE ANNOTATED MODEL

```
MACHINE mch_annotated_version REFINES mch_floating_point_version
...
INVARIANTS
@inv1: si_distance_travelled ∈ SI_MEASURE_Type(METRE_UNIT)
@inv2: si_measured_time ∈ SI_MEASURE_Type(SECOND_UNIT)
@inv3: si_speed ∈ SI_MEASURE_Type(METRE_PER_SECOND_UNIT)
@inv4: si_starting_position ∈ SI_MEASURE_Type(METRE_UNIT)
@inv5: si_starting_time ∈ SI_MEASURE_Type(SECOND_UNIT)
@glueing-1: value(si_distance_travelled) = distance_travelled
@glueing-2: value(si_measured_time) = measured_time
@glueing-3: value(si_speed) = speed
...
EVENTS
...
get_speed ≡
  any si_p si_t
  where
    @grd1: si_p ∈ SI_MEASURE_Type(METRE_UNIT) ∧ si_p SI_GT si_starting_position
    @grd2: si_t ∈ SI_MEASURE_Type(SECOND_UNIT) ∧ si_t SI_GT si_starting_time
    @grd3: div_WD(...)
  with
    value(si_p) = p ∧ value(si_t) = t
  then
    @act1: si_distance_travelled := si_p SI_MINUS si_starting_position
    @act2: si_measured_time := si_t SI_MINUS si_starting_time
    @act3: si_speed := (si_p SI_MINUS si_starting_position) SI_DIV (si_t SI_MINUS si_starting_time)
  end
END
```



Rodin generates a large number of WD POs, verifying the correct use of measurement units associated with variables that appear in different arithmetic expressions.

OUTLINE

- The context of the work
- The motivating example
- The proposed approach
- Revisiting the motivating example
- Conclusion and future works

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

CONCLUSION AND FUTURE WORK

Proposed Approach

- Extension of the Event-B type-checking system using the Theory plugin
- Integration of standard units of measurement (SI units)
- A generic theory as a support for :
 - the seven base units
 - derived units (named or not)
 - arithmetic operators adapted for unit-based expressions

Future Work

- Application to a more complex case study (autonomous vehicles)
- Planned integration into our framework **OntoEventB**
 - for automatic generation of Event-B models from ontologies

THANK YOU

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