Validation of Formal Models Transformation through Animation

Alexandra Halchin, Neeraj Singh, Yamine Ait-Ameur , Abderrahmane Feliachi & Julien Ordioni

KMOTS, Guilin China, 1st August 2019

1. Introduction

RATP, a national public service company

1949

1959

1976

1992

ÉPIC RATP (Paris)
Historic state-owned part

RATP Group
in 14 countries

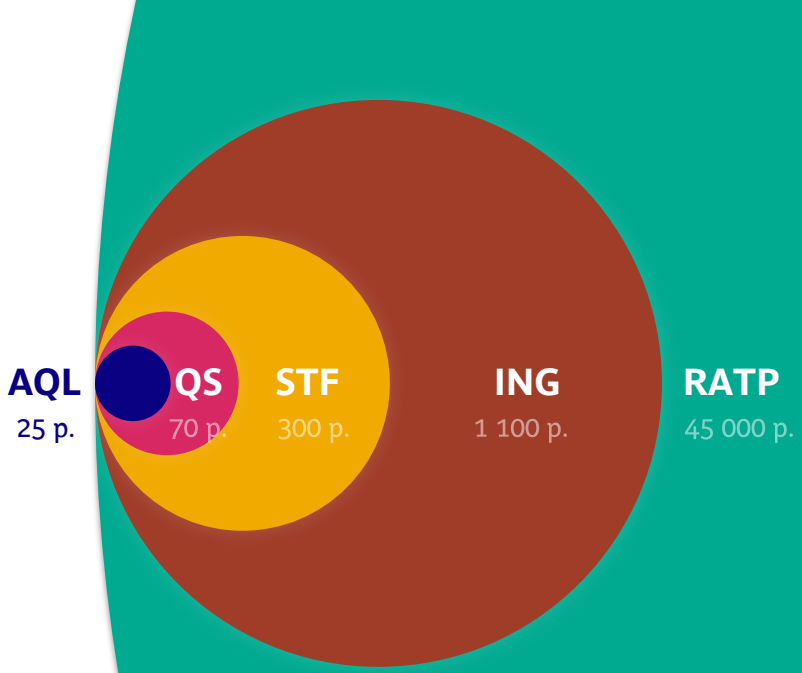
16 million journeys
every day worldwide

63.000 collaborators
(RATP Group)

One of the
5 leading players

1. Introduction

Who are we?



Engineering Department

Dealing with **transportation systems**

Involving **passenger safety**

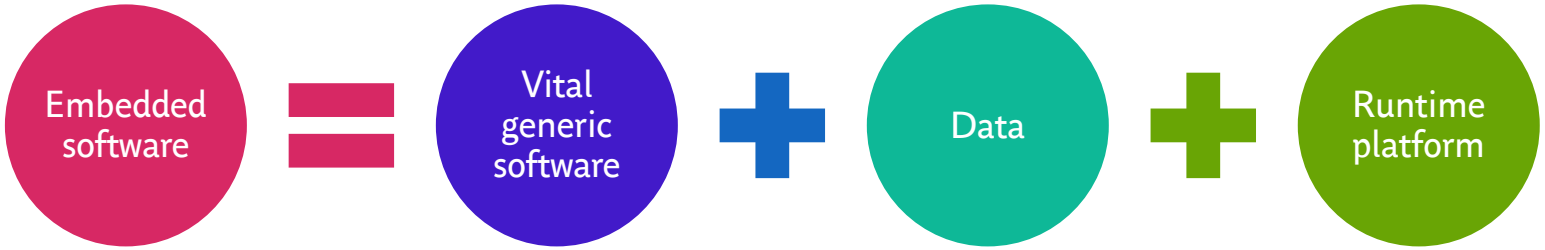
About **control/command software**

AQL, *Safety critical assessment software lab*, since end of 80s (SACEM, RER A)

1. Introduction

AQL, Safety critical software assessment lab

Assess that the running embedded software, including data, has a safe behavior regarding the traveler safety



For:

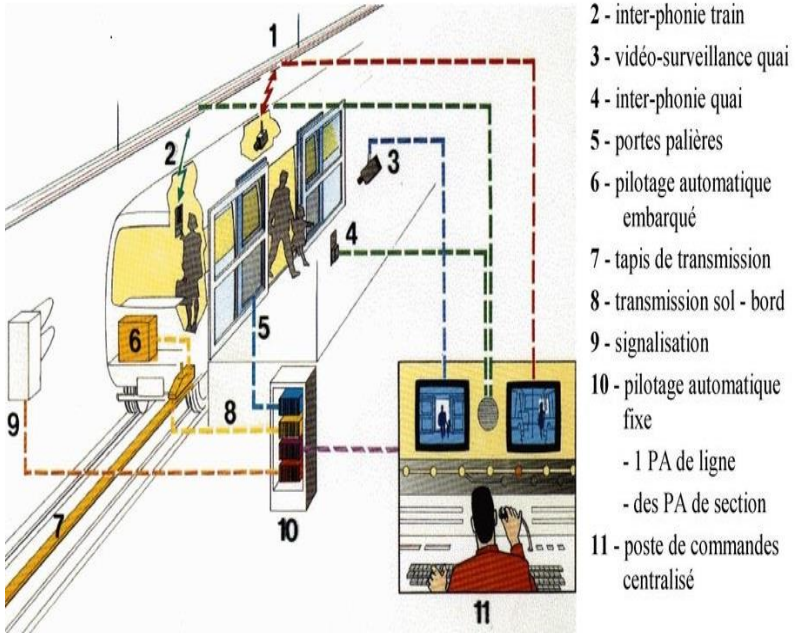
- Train control/command systems(CBTC)
- Computerized-based or hybrid interlocking systems(PMI, PHPI)
- Other safety critical software(PSD, DIL, DOF)

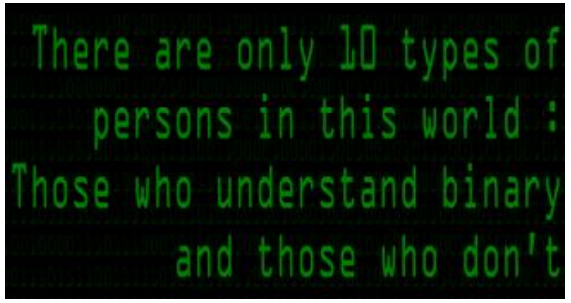
With:

- State of the art tools and methods
- Dedicated tools: HIL testing environment , proof servers (1 TB RAM)

1. Introduction

Starting Point
How to verify a safety critical software?





- RATP safety policy

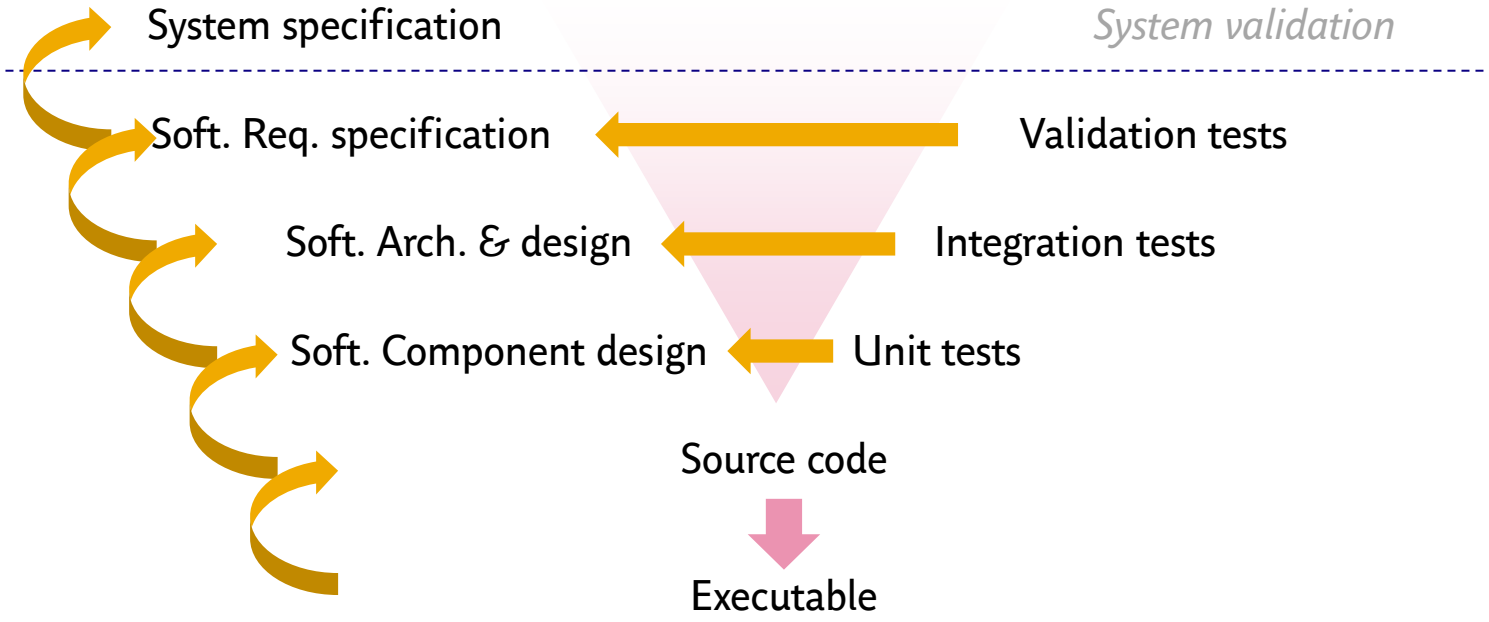
Provide an internal and independent assessment of safety critical systems before commissioning

- Challenges:
  - **Adapt** to suppliers process (different methods and languages)
  - **Non-intrusive** verification (source code shouldn't be modified)

1. Introduction

Typical assessment activities

- Validation of each steps of the refinements
  - System ⇨ software specification
  - Software specification ⇨ source code
  - Code source ⇨ executable
- Manual code review
- Tests validation and test coverage validation



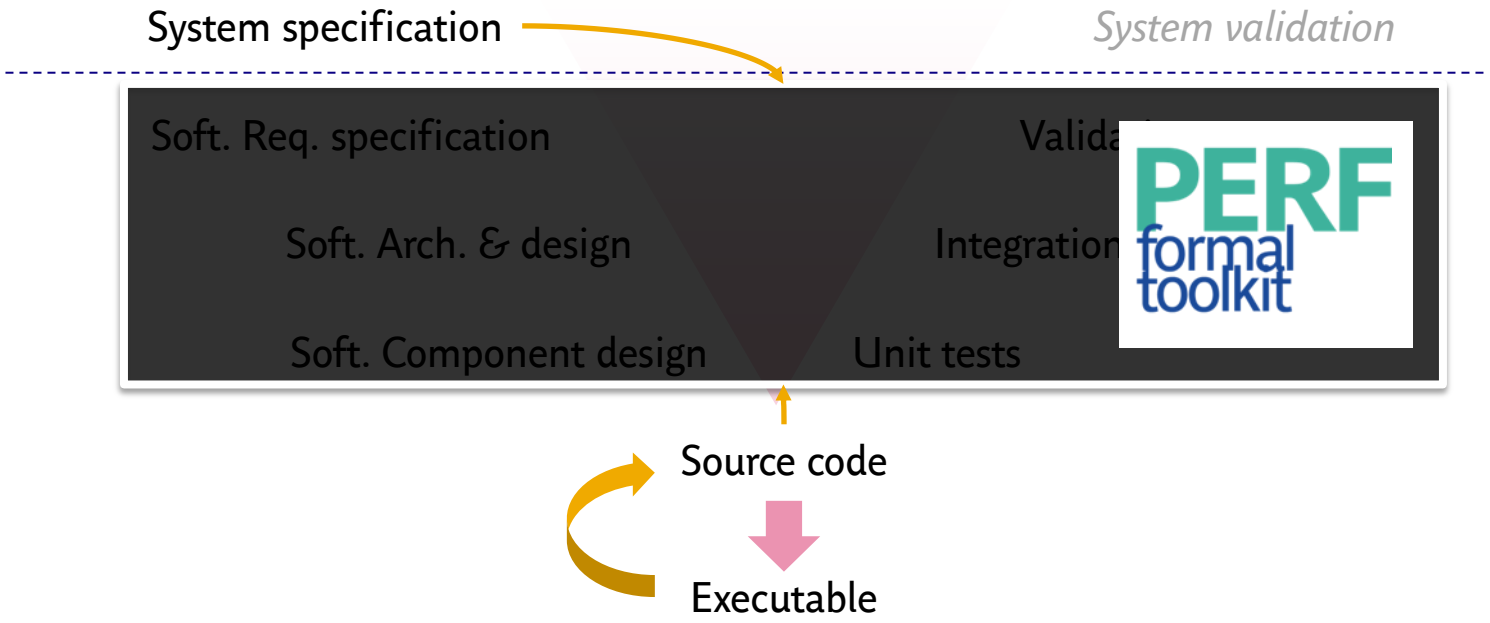
1. Introduction

Retro-modelling

Because it's efficient

PERF Method

- *Proof Executed over a Retro-engineered Formal model*
- A RATP *PERF formal toolkit* combined with third-party SAT proof engine
- Suitable for different projects and suppliers
- Independent of the software development cycle
- HLL Property level (component, soft. or system level) depends on the needs

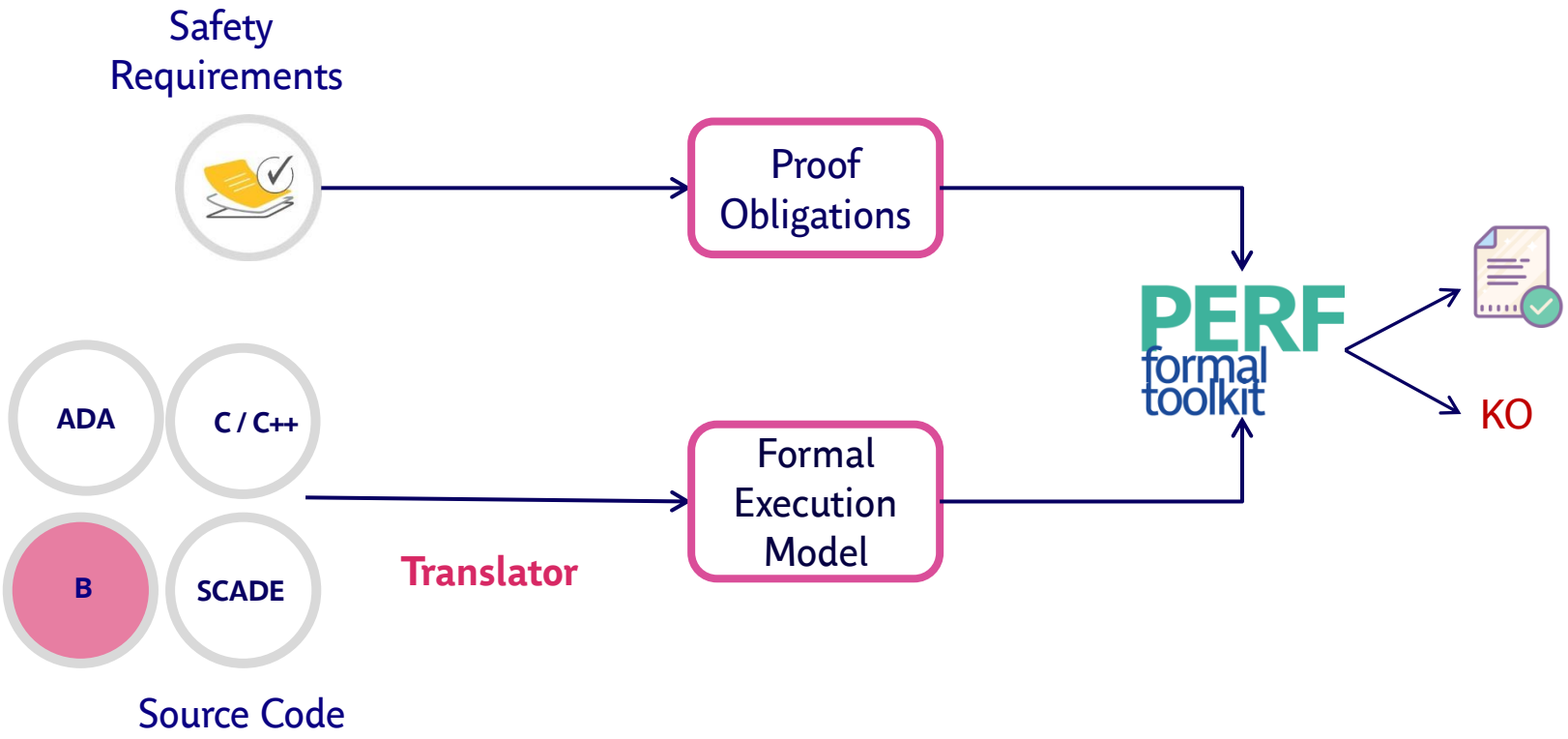


Performed on projects: 1 3 4 5 6 8 9 11 12 13

1. Introduction

PERF, An Integration and Verification Framework

Motivation



- ***P**roof **E**xecuted over a **R**etro-engineered **F**ormal model*

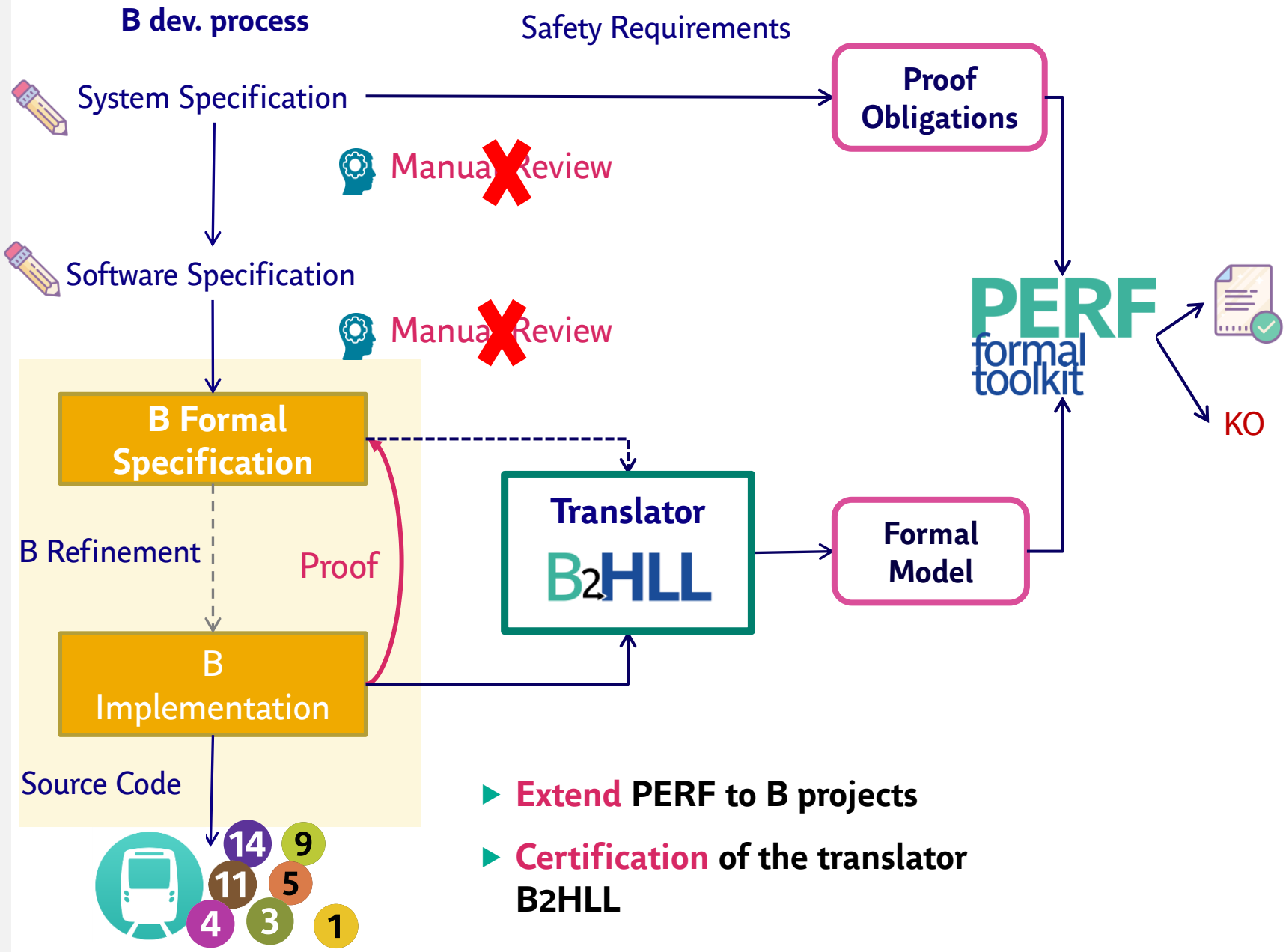
Challenges

- 50% of embedded software is developed using the B Method
- Certification of the translator

1. Introduction

PERF, An Integration and Verification Framework

Motivation



- **Extend PERF to B projects**
- **Certification of the translator B2HLL**

Outline

1. Introduction
2. B-PERFect
3. B2HLL Tool Validation
4. Conclusion

2. B-PERFect

B method & HLL

B Method

- Successful industrial examples: CBTC on board component line 3, CBTC wayside component line 5 and line 9, Driverless metro line 1 and line 14
- Based on first order logic and set theory
- B0 - subset of B language with programming-like constructs
- Dynamic changes of B components
  - *State:* set of variables constrained by an invariant
  - *Operations:* modify the state while maintaining the invariant (Before After Predicate)

2. B-PERFect

B method & HLL

HLL, the core of PERF

- HLL is the pivot language of the PERF methodology
- A variable represents an infinite stream of data -> *Data-flow language*
- Models reactive systems -> *Synchronous language*
- The focus is on the input/output relationship rather than on control structure -> *Declarative langugage*
- Unique implicit clock -> logical time
- Temporal operators: *X(e), Pre(e)*
- This is the language family of Scade, Simulink, LabView, etc.

2. B-PERFect

HLL, Dataflow Equations

Two possible formulations:

- `v := e, f;`
- `I(v) := e;`
  `X(v) := f;`

e	e_0	e_1	e_2	e_3	e_4	...
f	f_0	f_1	f_2	f_3	f_4	...
v	e_0	f_0	f_1	f_2	f_3	...

Cyclic references must be broken by a latch:

- `X(e)`
- `Pre(e)`
- `Pre(e, i)`

e	e_0	e_1	e_2	e_3	e_4	...
X(e)	e_1	e_2	e_3	e_4	e_5	...
Pre(e)	nil	e_0	e_1	e_2	e_3	...
i	i_0	i_1	i_2	i_3	i_4	...
Pre(e, i)	i_0	e_0	e_1	e_2	e_3	...

Equivalent formulation:

- `v := pre(f,e);`

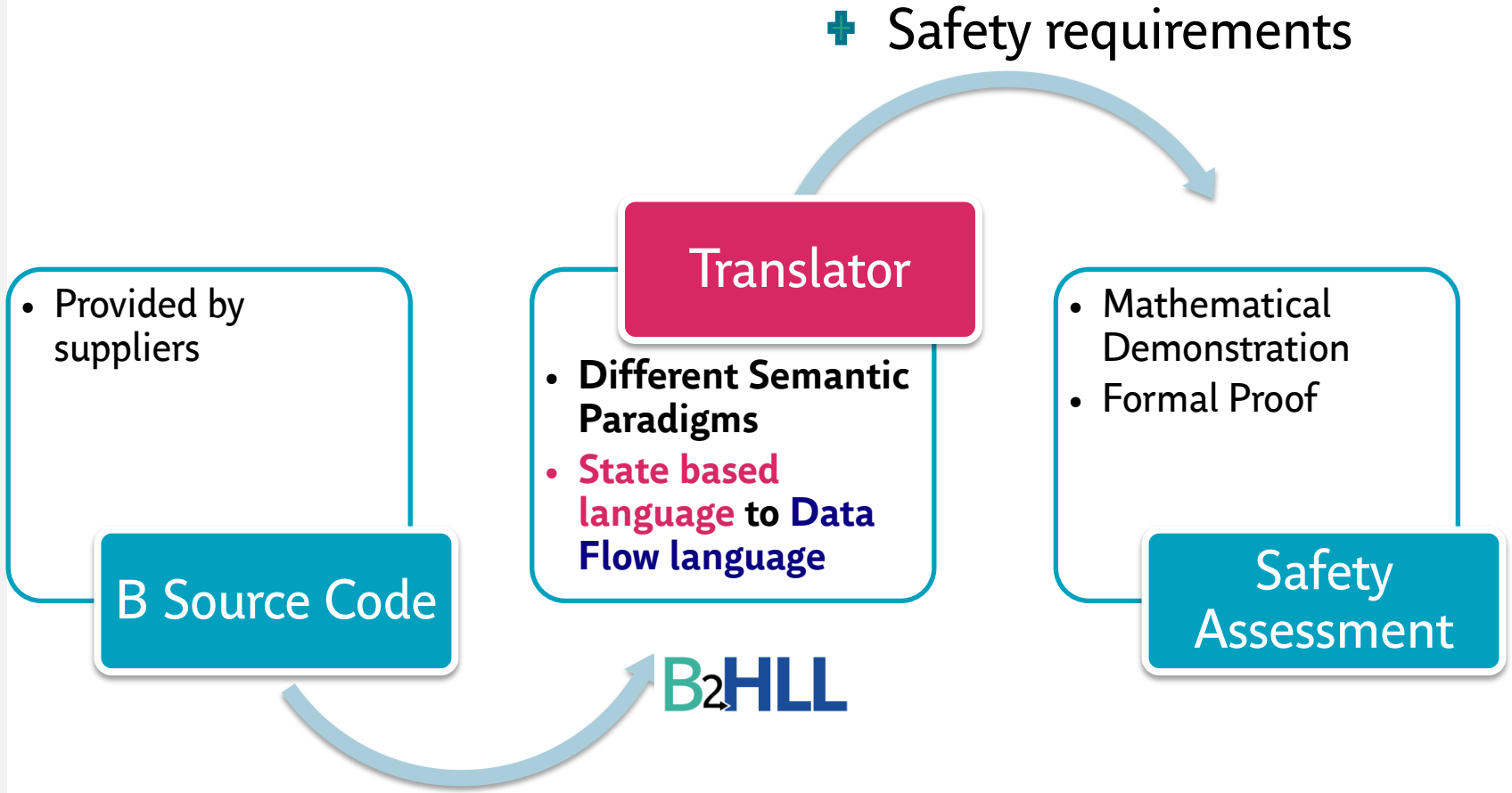
2. B-PERFect

B & HLL

Characteristic	B0	HLL
Paradigms	Imperative (HOW?)	Declarative (WHAT?)
Modularity	Machines	Absent
State management	Stateful	Stateless
Execution order	Important	Absent
Primary control flow	Sequence, loops, conditionals, operations and operations call	Conditionals, lambda expressions

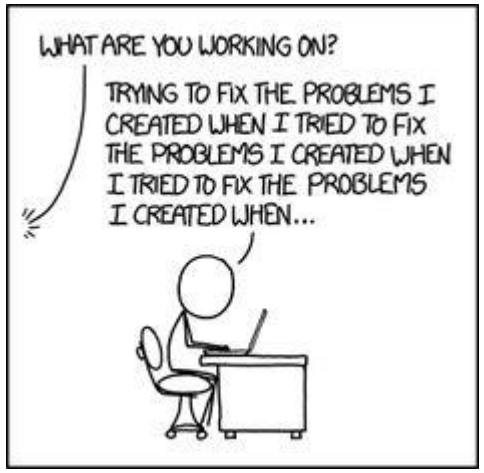
2. B-PERFect

B2HLL



2. B-PERFect

Objectives



Define the translation principles

Formalize languages semantics and assert semantic preservation of translation rules

B2HLL

Tool Validation with respect to formalized translation rules via **Model Animation**

2. B-PERFect

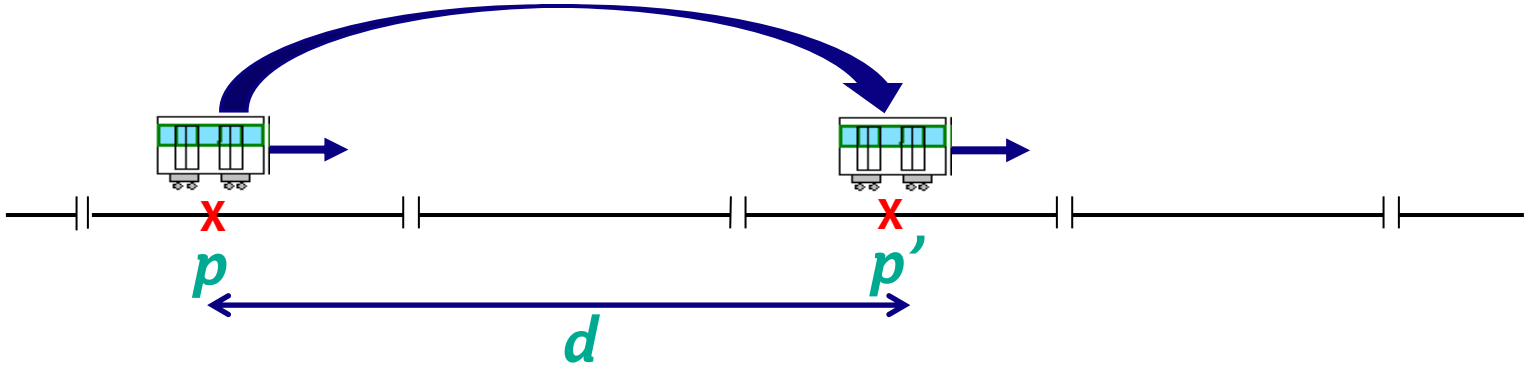
Train Localization in a CBTC system

Case Study Description

Description

Given a topology of *n* line segments, a travelled distance estimation *d* and a train position *p* corresponding to a segment identifier, an abscissa and an orientation. The new train position *p'* shall be computed according to the given distance *d* .

Description of the computation function: *p' = findLoc (p, d)*



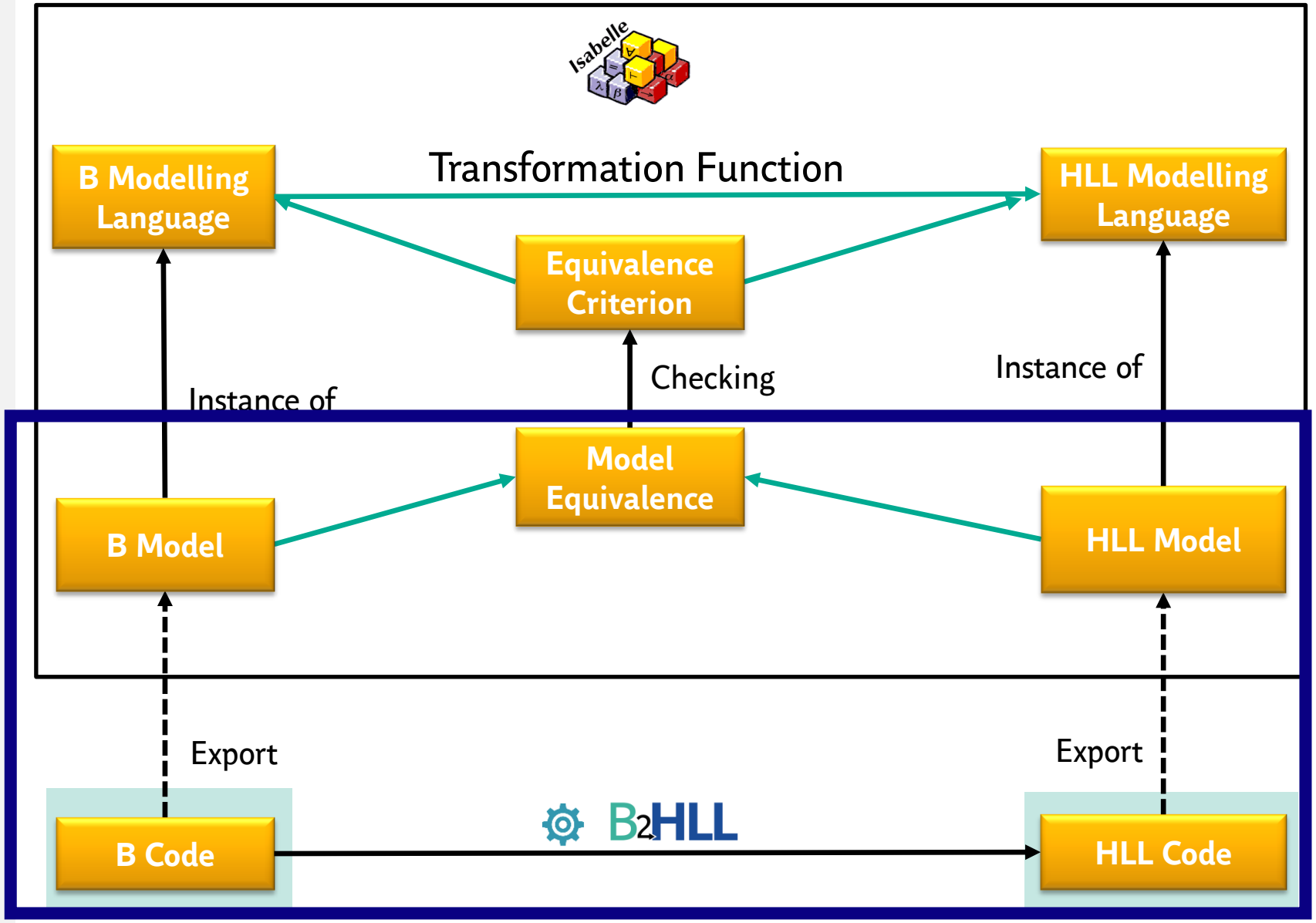
Execution values:

- p (segment identifier : 1, abscissa : 10);
- n = 10:
- segment_length = 30;
- d = 30;

3. B2HLL Tool Validation

A Formal Framework of Certified Translator

1. Introduction
2. **B-PERFect**
3. B2HLL Tool Validation
4. Conclusion



3. B2HLL Tool Validation

Model Animation

Not a « silver bullet »

- « Beware of bugs in the above code; I have only proved it correct, not tried it » Donald E. Knuth

Does the output of the B2HLL tool correspond to the defined formal specification?

We apply animation

- To identify the right formalisation of our definition, lemmas

- To validate the translation rules implemented in the B2HLL tool

3. B2HLL Tool Validation

Our Approach

Tool Validation via Instantiation and Model Animation in Isabelle/HOL

Specific models encoding (export)

- Developed B model using the formalized syntax
- HLL model produced by the B2HLL tool using the formalized syntax

Models Instantiation

- Initial values

Models execution

- Observation of B & HLL states evolution

3. B2HLL Tool Validation

B Encoding in Isabelle/HOL

B language

- Arithmetic expressions
- Boolean expressions
- Statements
- **Semantics** with meaning function

$$meaning\_instruction \in instruction \rightarrow env \rightarrow env$$

```
fun meaning_instruction :: "instruction ⟹ env ⟹ env" where
    "meaning_instruction (SKIP) σ = σ

  | "meaning_instruction (Bl list) σ =
       (case list of [] ⟹ σ
                   | e#l ⟹ meaning_instruction (Bl l) (meaning_instruction e σ))"

  | "meaning_instruction (Assign (vn, Tval)(Bexp exp)) σ =
       σ((vn, Tval.Bool)  := B (meaning_b exp σ))"

  | "meaning_instruction (If c b1 b2) σ =
       (if meaning_b c σ then meaning_instruction b1 σ
        else meaning_instruction b2 σ)"
```

3. B2HLL Tool Validation

Train Localization in a CBTC system

B Model in Isabelle/HOL


```
OPERATIONS
findLoc (i_seg, i_abs, i_dep) =
 VAR  l_x,  l_seg  IN
       l_x := i_abs + i_dep ;
       l_seg := i_seg ;
       WHILE  ( c_seg_length  <  l_x ) &  (l_seg < c_nb_segs)  DO
                    l_x  := l_x - c_seg_length  ;
                    l_seg  := l_seg + 1 ;
      INVARIANT  ( l_seg - i_seg ) * c_seg_length + l_x  = i_abs  +  i_dep
                     …
      VARIANT l_x
       END
…
```

```
definition "TrainPositioning_findLoc = b.Bl [ findLoc_locals_assign, findLoc_while]"
definition "findLoc_locals_assign =
b.Bl  [  (b.Assign  findLoc_l_x  (b.Plus  findLoc_i_abs  findLoc_i_dep)),
         (b.Assign   findLoc_l_seg (b.AVar findLoc_i_seg))]"
definition "findLoc_while_body =
b.Bl  [  (b.Assign  findLoc_l_x  (b.Minus  findLoc_l_x  TypingPos_c_seg_length)),
         (b.Assign  findLoc_l_seg (b.Plus  findLoc_l_seg  b.Value 1)))]"
definition "findLoc_while =  b.b_while_to_if  variant findLoc_while_cond
findLoc_while_body"
…
```

3. B2HLL Tool Validation

HLL Encoding in Isabelle/HOL

HLL language

- Arithmetic expressions
- Boolean expressions
- Statements
- **Semantics** with meaning function:

$$meaning\_instruction \in instruction \rightarrow env \rightarrow env$$

```
fun meaning_instruction :: "instruction ⟹ env ⟹ env" where
    "meaning_instruction (Bl list) σ =
        (case list of [] ⟹ σ
                    | e#l ⟹ meaning_instruction (Bl l) (meaning_instruction e σ))"

  | "meaning_instruction (Assign vn exp) σ = σ(vn := meaning_exp exp σ)"

  | "meaning_instruction (Assign' vn exp1 exp2) σ =
        (let v1 = meaning_exp exp1 σ in
         let v2 = meaning_exp exp2 σ in
         σ (vn := stream_comp v1 v2))"
```

3. B2HLL Tool Validation

Train Localization in a CBTC system

HLL Model in Isabelle/HOL


```
Namespaces: "findLoc_0"{
Definitions:
    //mapping input parameters
    i_dep_0 := l_xDep_0;
    i_abs_0 := v_absOnSeg_0;
    i_seg_0 := v_seg_0;
    //distance computation
    l_x_0  := i_abs_0 + i_dep_0;
    l_seg_0 := i_seg_0;

    //Begin While Iter 0
    l_x_1     :=  l_x_0 - c_seg_length;
    l_seg_1   :=  l_seg_0 + 1;
    l_x_2     :=   if c_seg_length  <  l_x_0 &
                     (l_seg_0 < c_nb_segs)
                   then l_x_1
                   else l_x_0;
    l_seg_2     := if c_seg_length < l_x_0  &
                     (l_seg_0 < c_nb_segs)
                   then l_seg_1
                   else l_seg_0;
    …}
```

```
definition "TrainPositioning_findLoc_0_Hll =
    hll.Bl [ findLoc_locals_assign_Hll, findLoc_while_Hll0, ...] "
definition "findLoc_locals_assign_Hll = hll.Bl [
    (hll.Assign findLoc_l_x0 (hll.Plus  findLoc_i_abs0 findLoc_i_dep0))),
    (hll.Assign findLoc_l_seg0 (hll.AVar findLoc_i_seg0))  ]"
definition "findLoc_while_body_Hll0 = hll.Bl [
    (hll.Assign findLoc_l_x1 (hll.Minus (findLoc_l_x0 TypingPos_c_seg_length0))),
    (hll.Assign findLoc_l_seg1 (hll.Plus (findLoc_l_seg0 hll.Value (£_.1))))]"
definition "findLoc_while_Hll0 = hll.Bl [
   hll.Assign findLoc_l_x2 (hll.exp.If findLoc_while_cond_Hll0 findLoc_l_x1 findLoc_l_x0),
   hll.Assign findLoc_l_seg2  (hll.exp.If findLoc_while_cond_Hll0 findLoc_l_seg1 findLoc_l_seg0)]"
```

3. B2HLL Tool Validation

Transformation Function

From B to HLL

- Defined inductively on the syntactic constructs of B language
- Mapping of B state variable to HLL data flows (streams)

$$Mapping = Bvname \twoheadrightarrow (Hllvname\ x\ Hllvname)$$

```
fun Transformation :: "b.instruction ⟹ mapping ⟹ (hll.instruction × mapping)" where
    "Transformation (b.Bl [ ]) m              = (hll.Bl [ ], m)"
  | "Transformation (b.Bl (a#list)) m         =
        (comp (Transformation a m) (Transformation (b.Bl list)))"
  | "Transformation b.SKIP m                  = (hll.Bl [ ], m)"
  | "Transformation (b.Assign vname exp) m =
        (let v = (createFreshHLLVariable vname m) in
            (hll.Assign v (Transformation_exp exp m), m (vname ↦ (v, v))))"
  | "Transformation (b.If bexp instruction1 instruction2) m =
    (let c' = (Transformation_exp( b.Bexp (bexp)) m);
        (* c' → Condition transformation, c1 → If block transformation and m1 → Resulting mapping *)
        (c1,m1) = Transformation instruction1 m;
        (c2,m2) = Transformation instruction2 (m ⊕ m1);
        (* vars → Modified vars in one of IF branches *)
        vars = {v. v : (dom m) ∧ ((m v ≠ m1 v) ∨ (m v ≠ m2 v))};
        (* List of assigns for modified vars *)
        assigns = Finite_Set.fold (Transformation_if_step_i m1 m2 inst c' st) {} vars
    in (Bl ([c1, c2]@(set_to_list assigns)), st))"
```

3. B2HLL Tool Validation

Train Localization in a CBTC system

Proof Debugging

Do we have the right formalization?

Formalization debugging

- **Transformation Function** execution
- Requires initial mapping **n** and initial states: σ_{0B} , σ_{0HLL}

"lemma findLoc_equivalence:

(codeHLL, m) = **Transformation** TrainPositioning_findLoc n ⟹

(b.meaning_instruction TrainPositioning_findLoc σ_B)

$\cong_m$

(hll.meaning_instruction codeHLL σ_{HLL})"

Inconsistencies are discovered during failed proof attempts

3. B2HLL Tool Validation

Train Localization in a CBTC system

Models execution

Does the output of the B2HLL tool correspond to the B code?

Provided mapping t

```
definition " t = Map.empty (...
                findLoc_l_x      |-> (findLoc_l_x20,  findLoc_l_x20),
                findLoc_l_seg    |-> (findLoc_l_seg20,  findLoc_l_seg20) ...)
```

Models Execution

```
lemma findLocEquivalence
        " (b.meaning_instruction TrainPositioning_findLoc σ_B)
                        ≅_t
  (hll.meaning_instruction TrainPositioning_findLoc_0_Hll σ_HLL)"

  apply (subst TrainPositioning_findLoc _def)
  apply (subst b.meaning_instruction.simps)
  apply (subst TrainPositioning_findLoc_0_Hll _def)
  apply (subst hll.meaning_instruction.simps)
  apply (subst meaning_equiv_def)
  apply (simp)
  done
```

3. B2HLL Tool Validation

Train Localization in a CBTC system

Models execution

B & HLL Models Execution

lemma findLocEquivalence

" (b.meaning_instruction TrainPositioning_findLoc σ_B)

$$\cong_t \begin{pmatrix} findLoc\_i\_abs & \rightarrow findLoc\_i\_abs\_0, \\ findLoc\_i\_seg & \rightarrow findLoc\_i\_seg\_0, \\ findLoc\_l\_x & \rightarrow findLoc\_l\_x\_20, \\ findLoc\_l\_seg & \rightarrow findLoc\_l\_seg\_20 \end{pmatrix}$$

(hll.meaning_instruction TrainPositioning_findLoc_0_Hll σ_{HLL})“

Evaluation of expressions using term substitutions

3. B2HLL Tool Validation

Train Localization in a CBTC system

B2HLL tool Certification


```
definition "TrainPositioning_findLoc = b.Bl [ findLoc_locals_assign, findLoc_while]"
definition "findLoc_locals_assign =
b.Bl [ (b.Assign findLoc_l_x (b.Plus findLoc_i_abs findLoc_i_dep)),
       (b.Assign  findLoc_l_seg (b.AVar findLoc_i_seg))]"
definition "findLoc_while_body =
b.Bl [ (b.Assign findLoc_l_x (b.Minus findLoc_l_x Typing
       (b.Assign findLoc_l_seg (b.Plus findLoc_l_seg b.Value 1)))]"
definition "findLoc_while = b.b_while_to_if variant findL
...
```

$\sigma_B(\ \lambda v.v \Rightarrow undefined\ )$

$\sigma_B(findLoc\_l\_x \Rightarrow 40,\ findLoc\_l\_seg \Rightarrow 1)$

$\sigma_B(findLoc\_l\_x \Rightarrow 10,\ findLoc\_l\_seg \Rightarrow 2)$

```
definition "TrainPositioning_findLoc_0_Hll =
   hll.Bl [ findLoc_locals_assign_Hll, findLoc_while_Hll0, ...] "
definition "findLoc_locals_assign_Hll = hll.Bl [
   (hll.Assign findLoc_l_x0 (hll.Plus findLoc_i_abs(
   (hll.Assign findLoc_l_seg0 (hll.AVar findLoc_i_se
definition "findLoc_while_body_Hll0 = hll.Bl [
   (hll.Assign findLoc_l_x1 (hll.Minus (findLoc_l_x(
   (hll.Assign findLoc_l_seg1 (hll.Plus (findLoc_l_s(
definition "findLoc_while_Hll0 = hll.Bl [
   hll.Assign findLoc_l_x2 (hll.exp.If findLoc_while_
   hll.Assign findLoc_l_seg2  (hll.exp.If findLoc_whi
```

$\sigma_{HLL}(\ \lambda v.v \Rightarrow \lambda i.\ undefined\ )$

$\sigma_{HLL}(findLoc\_l\_x\_0 \Rightarrow \lambda i.40,\ findLoc\_l\_seg\_0 \Rightarrow \lambda i.1)$

$\sigma_{HLL}(findLoc\_l\_x\_1 \Rightarrow \lambda i.10,\ findLoc\_l\_seg\_1 \Rightarrow \lambda i.2,\ findLoc\_l\_x\_2 \Rightarrow \lambda i.10,\ findLoc\_l\_seg\_2 \Rightarrow \lambda i.2)$

3. B2HLL Tool Validation

Train Localization in a CBTC system

B2HLL tool Certification

Are the states obtained after the execution of B and HLL models equivalent?

$$\sigma_B(findLoc\_l\_x \Rightarrow 10,\ \ findLoc\_l\_seg \Rightarrow 2)$$

$$\cong_t \begin{pmatrix} findLoc\_i\_abs \rightarrow findLoc\_i\_abs\_0, \\ findLoc\_i\_seg \rightarrow findLoc\_i\_seg\_0, \\ findLoc\_l\_x \rightarrow findLoc\_l\_x\_20, \\ findLoc\_l\_seg \rightarrow findLoc\_l\_seg\_20 \end{pmatrix}$$

$$\sigma_{HLL}(findLoc\_l\_x\_1 \Rightarrow \lambda\, i.\, 10,\ \ findLoc\_l\_seg\_1 \Rightarrow \lambda\, i.\, 2,$$
$$findLoc\_l\_x\_20 \Rightarrow \lambda\, i.\, 10,\ \ findLoc\_l\_seg\_20 \Rightarrow \lambda\, i.\, 2)$$

4. Conclusion

- Proof debugging relying on the animation of conjectures
- Guarantee the correctness of the translation from B to HLL
- Formally certified B2HLL tool translation principles
- Proof of concept on several case studies
- Ongoing B2HLL prototype industrialization in the PERF RATP framework

THANK YOU FOR YOUR ATTENTION!

RATP

Alexandra HALCHIN

+33 1 58 786 1 63
alexandra.halchin@ratp.fr
alexandra.halchin@enseeiht.fr