Various reasonings with Event-B [1]

Yamine Aït Ameur*
** **Joint work with G. Dupont, I. Mendil, M.Pantel, P. Rivière and N. Singh**

* IRIT/Toulouse INP-CNRS, University of Toulouse, France
** Joint work with **G. Dupont, I. Mendil, M.Pantel, P. Rivière** and **N. Singh**

July 1^{st}-2^{nd}, 2025

IFIP WG 1.9 Meeting

Paris - France

[1] This work is supported by ANR (Agence Nationale de la Recherche) grant ANR-19-CE25-0010-01 (EBRP)

Outline

Context

State based formal methods

Capability of formal state-based methods

- to **model** complex systems
- to **reason** and **establish properties** reflecting the modelled requirements.

In particular,

- ensuring system safety through the verification of invariant properties
- each reachable state of the modelled system fulfils these invariants, i.e. the system state is always in a safe region and never leaves it
- verification is based on an induction principle over traces of transition systems

State variables are modified by **actions** relying on

- the generalised assignment operation based on the *"becomes such that"* BAP
- defining a state transition
- ASM rules, substitutions or events in B and Event-B, Hoare triples, Guarded Commands (GCL), operations in RSL and VDM , actions in TLA$^+$, schemas in Z, ...

Context

Domain knowledge/Domain modelling

Modelling of complex systems in system engineering relies on domain knowledge

- **shared** and **reused** in system models
- **definitions** as well as domain-specific **properties**
- formalised as **theories** with **data types, operators, axioms**
- and **theorems** proved using the associated proof system, *independently* of the designed system models.

Necessity to model explicitly domain, context, environment, ...

$$D, S \vdash R$$

Context

Two different types of Domains

- **Once and for all** formalised domains, **stable** and **reusable**
  - mathematics: diff. eq., control theory, probabilities, etc.
  - physics, flight dynamics, units, etc.
  - more generally, external theories related to designed systems
- **System dependent** formalised domains
  - describe system concepts
  - "instantiations", "specialisations" of above theories with additional specific constraints
  - Examples: valves, user interfaces, tanks, cars, protocols, etc.

Domain specific theories

Borrowing domain specific theories in system design formal models brings

- Hypotheses
- Theorems and proof rules

This idea is not new !

Context

Expressivity of Event-B

Event-B is based on set theory and First order logic with very few typing capabilities

Which extensions for Event-B ?

- Event-B can be extended to handle domain theories using its Algebraic data Types *Theory component*.
  We use it to formalise **"invited semantics"**
  - *Transfer to and Reuse* in, the system design models, the proofs achieved on the theory side
  - Keep using the Event-B invariant preservation mechanism (induction) while **referring (annotation)** to externally defined (data-types)
  - Separation of concerns

Means

- Partial definitions and
- **Well-definedness conditions**
- Correctness by construction

Transitions are seen as partially defined operations limited to *"acceptable"* states

New reasoning capabilities with Event-B

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **END**

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **END**

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **EVENTS** INITIALISATION $\widehat{=}$ . . . evt^A $\widehat{=}$ **END** **END**

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **EVENTS** INITIALISATION $\widehat{=}$ . . . $evt^A \widehat{=}$ **ANY** α^A **WHERE** $G^A(x^A, \alpha^A)$ **THEN** $x^A :\mid BAP^A($ $\alpha^A, x^A, x^{A\prime})$ **END** **END**

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **INVARIANTS** $I^A(x^A)$ **THEOREMS** $T_{mch}(x^A)$ **VARIANT** $V(x^A)$ **EVENTS** INITIALISATION $\widehat{=}$ . . . $evt^A \widehat{=}$ **ANY** α^A **WHERE** $G^A(x^A, \alpha^A)$ **THEN** $x^A :\mid BAP^A($ $\alpha^A, x^A, x^{A\prime})$ **END** **END**

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **INVARIANTS** $I^A(x^A)$ **THEOREMS** $T_{mch}(x^A)$ **VARIANT** $V(x^A)$ **EVENTS** INITIALISATION $\widehat{=}$ $\ldots$ $evt^A \widehat{=}$ **ANY** α^A **WHERE** $G^A(x^A, \alpha^A)$ **THEN** $x^A :\mid BAP^A(\alpha^A, x^A, x^{A\prime})$ **END** **END**

Automatic generation of POs.

Rodin $\Longrightarrow$ Editor + POs generator + automatic/interactive provers

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **INVARIANTS** $I^A(x^A)$ **THEOREMS** $T_{mch}(x^A)$ **VARIANT** $V(x^A)$ **EVENTS** INITIALISATION $\widehat{=}$ $\ldots$ $evt^A \widehat{=}$ **ANY** α^A **WHERE** $G^A(x^A, \alpha^A)$ **THEN** $x^A :\mid BAP^A(\alpha^A, x^A, x^{A\prime})$ **END** **END**

Theorems Context (*THM_ctx*)	$A \Rightarrow T_{ctx}$
Theorems machine (*THM_mch*)	$A \wedge I^A(x^A) \Rightarrow T_{mch}(x^A)$
Initialisation (INIT)	$A \wedge AP^A(\alpha^A, x^{A\prime}) \Rightarrow I^A(x^{A\prime})$
Invariant preservation (INV)	$A \wedge I_A(x^A) \wedge G_A(x^A, \alpha^A)$ $\wedge BAP^A(x^A, \alpha^A, x^{A\prime}) \Rightarrow I^A(x^{A\prime})$
Event feasibility (FIS)	$A \wedge I_A(x^A) \wedge G^A(x^A, \alpha^A)$ $\Rightarrow \exists x^{A\prime} \cdot BAP^A(x^A, \alpha^A, x^{A\prime})$
Variant progress (VAR)	$A \wedge I^A(x^A) \wedge G^A(x^A, \alpha^A)$ $\wedge BAP^A(x^A, \alpha^A, x^{A\prime})$ $\Rightarrow V(x^{A\prime}) < V(x^A)$

Automatic generation of POs.

Rodin $\Longrightarrow$ Editor + POs generator + automatic/interactive provers

Event-B

Contexts (definitions/axioms/theorems) + Machines (state and events/inductive invariants)

Context	Machine
CONTEXT Ctx **SETS** s **CONSTANTS** c **AXIOMS** A **THEOREMS** T_{ctx} **END**	**MACHINE** M^A **SEES** Ctx **VARIABLES** x^A **INVARIANTS** $I^A(x^A)$ **THEOREMS** $T_{mch}(x^A)$ **VARIANT** $V(x^A)$ **EVENTS** INITIALISATION $\widehat{=}$ $\ldots$ $evt^A \widehat{=}$ **ANY** α^A **WHERE** $G^A(x^A, \alpha^A)$ **THEN** $x^A :\mid BAP^A(\alpha^A, x^A, x^{A\prime})$ **END** **END**

Theorems Context (*THM_ctx*)	$A \Rightarrow T_{ctx}$
Theorems machine (*THM_mch*)	$A \wedge I^A(x^A) \Rightarrow T_{mch}(x^A)$
Initialisation (INIT)	$A \wedge AP^A(\alpha^A, x^{A\prime}) \Rightarrow I^A(x^{A\prime})$
Invariant preservation (INV)	$A \wedge I_A(x^A) \wedge G_A(x^A, \alpha^A)$ $\wedge BAP^A(x^A, \alpha^A, x^{A\prime}) \Rightarrow I^A(x^{A\prime})$
Event feasibility (FIS)	$A \wedge I_A(x^A) \wedge G^A(x^A, \alpha^A)$ $\Rightarrow \exists x^{A\prime} \cdot BAP^A(x^A, \alpha^A, x^{A\prime})$
Variant progress (VAR)	$A \wedge I^A(x^A) \wedge G^A(x^A, \alpha^A)$ $\wedge BAP^A(x^A, \alpha^A, x^{A\prime})$ $\Rightarrow V(x^{A\prime}) < V(x^A)$
Deadlock freeness $\cdots$	$\cdots$
Reachability $\cdots$	$\cdots$
$\cdots$	$\cdots$

Automatic generation of POs.

Rodin $\Longrightarrow$ Editor + POs generator + automatic/interactive provers

Other properties ?

Event-B Structure and Proofs

Event-B

- a state based formal method with proof and refinement

Context	Machine
CONTEXT Ctx **SETS** s	**MACHINE** M^A **SEES** Ctx
CONSTANTS c	**VARIABLES** x^A
AXIOMS A	**INVARIANTS** $I^A(x^A)$
THEOREMS T_{ctx}	**THEOREMS** $T_{mch}(x^A)$
END	**VARIANT** $V(x^A)$
	EVENTS **EVENT** evt^A **ANY** α^A **WHERE** $G^A(x^A, \alpha^A)$ **THEN** $x^A :\mid BAP^A($ $\alpha^A, x^A, x^{A'})$ **END** ... **END**

- set theory, basic types (integers, booleans) and their associated operators
- first order logic
- explict state formalised as a set of state variables
- initialisation event and guarded events to record state changes based on BAP (Before-After Predicates)
- inductive reasoning on event traces
- invariant preservation and variant decreasing for reachability
- Rodin open source IDE

Event-B theories

Core Event-B is not equipped with

- rich data-types and associated operators
- in particular, there is no
  - reals NOR continuous features
  - capability to introduce new data types
  - possibility to generate new proof obligations for **invited semantics**

Event-B theories as a support for Event-B extensions

- introduced in 2010's by JR. Abrial, M. Butler, I. Maamria, ...
- support for defining new data types
- constructive or axiomatic
- tool supported as a PlugIn of the Rodin platform

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theories as extensions for Event-B basic language

Theory
THEORY Th
IMPORT Th1, ...

END

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory
THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
Type2(E, ...)
constructors
cstr1($p_1 : T_1$, ...)

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$
Constructive definitions

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory
THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
Type2(E, ...)
constructors
cstr1($p_1 : T_1$, ...)

AXIOMATIC DEFINITIONS
TYPES A_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory

THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
  Type2(E, ...)
  constructors
    cstr1(p_1:T_1, ...)
OPERATORS
  Op1<nature> (p_1:T_1, ...)

    direct definition D_1

AXIOMATIC DEFINITIONS
  TYPES A_1, ...
  OPERATORS
   AOp2<nature> (p_1:T_1, ...):T_r

  AXIOMS A_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory
THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
Type2(E, ...)
constructors
cstr1(p_1:T_1, ...)
OPERATORS
Op1<nature> (p_1:T_1, ...)
well—definedness $WD(p_1, \ldots)$
direct definition D_1

AXIOMATIC DEFINITIONS
TYPES A_1, ...
OPERATORS
AOp2<nature> (p_1:T_1, ...):T_r
well—definedness $WD(p_1, \ldots)$
AXIOMS A_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Each operator may be associated with Well-Definedness (WD) conditions (partial definitions).

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory

THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
  Type2(E, ...)
  constructors
    cstr1(p_1:T_1, ...)
OPERATORS
  Op1<nature> (p_1:T_1, ...)
    well—definedness $WD(p_1, \ldots)$
    direct definition D_1

AXIOMATIC DEFINITIONS
  TYPES A_1, ...
  OPERATORS
    AOp2<nature> (p_1:T_1, ...):T_r
    well—definedness $WD(p_1, \ldots)$
  AXIOMS A_1, ...

THEOREMS T_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Each operator may be associated with Well-Definedness (WD) conditions (partial definitions).

Relevant proved theorems.

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory

THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
  Type2(E, ...)
  constructors
    cstr1(p_1:T_1, ...)
OPERATORS
  Op1<nature> (p_1:T_1, ...)
    well—definedness $WD(p_1, \ldots)$
    direct definition D_1

AXIOMATIC DEFINITIONS
  TYPES A_1, ...
  OPERATORS
    AOp2<nature> (p_1:T_1, ...):T_r
    well—definedness $WD(p_1, \ldots)$
  AXIOMS A_1, ...

THEOREMS T_1, ...

PROOF RULES T_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Each operator may be associated with Well-Definedness (WD) conditions (partial definitions).

Relevant proved theorems.

Proof rules (inference/rewrite) can be included in the Rodin proof tactics

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory

THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
  Type2(E, ...)
  constructors
    cstr1(p_1:T_1, ...)
OPERATORS
  Op1<nature> (p_1:T_1, ...)
    well—definedness $WD(p_1, \ldots)$
    direct definition D_1

AXIOMATIC DEFINITIONS
  TYPES A_1, ...
  OPERATORS
    AOp2<nature> (p_1:T_1, ...):T_r
    well—definedness $WD(p_1, \ldots)$
  AXIOMS A_1, ...

THEOREMS T_1, ...

PROOF RULES T_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Each operator may be associated with Well-Definedness (WD) conditions (partial definitions).

Relevant proved theorems.

Proof rules (inference/rewrite) can be included in the Rodin proof tactics

Tool-support in Rodin environment.

Proof tactics in Rodin environment.

Event-B Theories

Formalisation of new data-types for Event-B $\Longrightarrow$ Theories

Theory

THEORY Th
IMPORT Th1, ...
TYPE PARAMETERS E, F, ...

DATATYPES
  Type2(E, ...)
  constructors
    cstr1(p_1:T_1, ...)
OPERATORS
  Op1<nature> (p_1:T_1, ...)
    well—definedness $WD(p_1, \ldots)$
    direct definition D_1

AXIOMATIC DEFINITIONS
  TYPES A_1, ...
  OPERATORS
    AOp2<nature> (p_1:T_1, ...):T_r
    well—definedness $WD(p_1, \ldots)$
  AXIOMS A_1, ...

THEOREMS T_1, ...

PROOF RULES T_1, ...

END

Theories as extensions for Event-B basic language

Algebraic definition for data-types $\Longrightarrow$ Constructive definitions and/or axiomatic definitions.

Each operator may be associated with Well-Definedness (WD) conditions (partial definitions).

Relevant proved theorems.

Proof rules (inference/rewrite) can be included in the Rodin proof tactics

Tool-support in Rodin environment.

Proof tactics in Rodin environment.

Library for mathematical and domain-specific theories (i.e., Reals, differential equations etc.)

Event-B theories: an example

Extend Event-B with mathematical structures or domain knowledge
$\Rightarrow$ *reals*, *differential equations*, *kinematics*, *ontologies*...

Event-B theories: an example

Extend Event-B with mathematical structures or domain knowledge
$\Rightarrow$ *reals, differential equations, kinematics, ontologies...*

THEORY Reals
IMPORT Relations , Rings
AXIOMATIC DEFINITIONS
TYPES $\mathbb{R}$
OPERATORS
$Rzero : \mathbb{R}$,
$Rone : \mathbb{R}$,
$plus : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$,
$times : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$,
$leq : \mathbb{R} \leftrightarrow \mathbb{R}$,
...
AXIOMS
order : order(leq) $\wedge$ total(leq)
reals : Field($plus$, $times$, $Rzero$, $Rone$)$\wedge$
integral($times$, $Rzero$)$\wedge$
ringCompatible($plus$, $times$, $Rzero$, $Rone$, leq)
...

Event-B theories: an example

Extend Event-B with mathematical structures or domain knowledge
$\Rightarrow$ *reals, differential equations, kinematics, ontologies...*

THEORY Reals
IMPORT Relations , Rings
AXIOMATIC DEFINITIONS
TYPES $\mathbb{R}$
OPERATORS
$Rzero : \mathbb{R}$,
$Rone : \mathbb{R}$,
$plus : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$,
$times : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$,
$leq : \mathbb{R} \leftrightarrow \mathbb{R}$,
...
AXIOMS
order : order(leq) $\wedge$ total(leq)
reals : Field($plus$, $times$, $Rzero$, $Rone$)$\wedge$
integral($times$, $Rzero$)$\wedge$
ringCompatible($plus$, $times$, $Rzero$, $Rone$, leq)
...

MACHINE M
VARIABLES x , y ,
INVARIANTS
inv1 : $x, y \in \mathbb{R}$
inv2 : $Rzero \ \ leq \ \ x \wedge x \neq Rzero$
inv3 : $y = Rzero \vee y = Rone$
EVENTS
...
Evt
WHERE
grd1 : $x \ \ times \ \ yleqRzero$
grd2 : $y = Rzero$
THEN $y := y \ \ plus \ \ Rone$
END

Well-Definedness (WD)

Event-B: Well-definedness Proof obligations

- Barringer, H., Cheng, J.H., Jones, C.B. *A logic covering undefinedness in program proofs.* Acta Informatica 21, 251–269 (1984)
- According to J.R. Abrial, Well-Definedness describes the

> *circumstances under which it is possible to introduce new term symbols by means of conditional definitions in a formal theory as if the definitions in question were unconditional, . . .* It avoids describing ill-defined operators, formulas, axioms, theorems, and invariants.

- Avoidance of ill-defined operators, formulas, axioms, theorems, and invariants.
- Each formula is associated to well-definedness POs that ensure that the formula is well-defined and that *two-valued logic can be used* (M. Leuschell - IFM'2020).

- An inductively defined WD predicate $WD(f)$ is associated with each formula f

- **Example.** Let a and b be two integers, $f \in D \pfun R$, then
  - $WD(a \div b) \equiv WD(a) \land WD(b) \land b \neq 0$
  - $WD(f(a)) \equiv WD(a) \land f \in D \pfun R \land a \in \mathrm{dom}(f)$

Well-Definedness (WD)

Event-B Theories: Well-definedness Proof obligations

- Each defined operator is associated to a (WD) condition ensuring its correct definition.
- When it is applied (in the theory or in an Event-B machine or context), this WD condition generates a PO requiring to establish that this condition holds
- The theory developer defines these WD conditions for the partially defined operators.
- They are then added to the native Event-B WD POs
- Once the WD POs are proved, they are added as hypotheses in the proofs of the other POs

New proof obligations

- Use of the WD mechanism
- When an operator is defined/applied, its WD PO is automatically generated

Principle

- WD Proof obligations to exclude non desired behaviours
- Under a *WD* condition a predicate operator *OP* entails a property *prop*

$$WD \wedge OP \Longrightarrow Prop$$

The Rodin Platform

Rodin is Eclipse based

- IDE for Event-B models development
- Automatic and interactive provers
- SMT solvers, predicate provers,
- Model checking
- Model animation
- Model visualisation
- Model transformation
- Code generators

Several plugIns are available

Maintenance is currently supported by the EBRP project `https://www.irit.fr/EBRP/`

Event-B Turned to a Domain specific modelling language

Application of the previous approach

Three extensions of Event-B

Hybrid systems [G. Dupont]

- To enrich Event-B with continuous behaviours

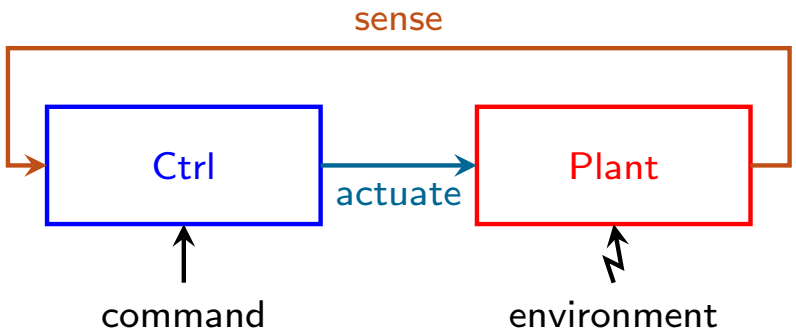
Ontologies and domain modelling [I. Mendil]

- To formalise standards in the area of Critical Interactive System

Reflexive Event-B [P. Rivière]

- To manipulate Event-B concepts and define new POs

Modelling Hybrid Systems in Event-B



Hybrid system = discrete + continuous behaviour

- Discrete behaviour: OK, supported by Event-B
- Continuous behaviour? ⇒ requires more work...

Control theory in FOL + set theory ⇒ use Event-B theories

Handling such system models require

- continuous building blocks (functions, diff. equations)
- **continuous behaviour** specification ("continuous events")
- **discrete/continuous** integration

Continuous Variables, Assignment and past preservation

continuous state variables

- *functions of time*
- $var \in \mathbb{R}^+ \pfun S$

continuous Before-After Predicate CBAP

$$
\begin{aligned}
&\mathrm{CBAP}(t, t', x_p, x_p', \mathcal{P}, H) \equiv \\
&\quad x_p{:}|_{t \to t'} \mathcal{P}(x_p, x_p') \& H \equiv \\
&\qquad [0, t[\lhd x_p' = [0, t[\lhd x_p && (\textit{Past Preservation}) \\
&\qquad \land\ \mathcal{P}([0, t] \lhd x_p, [t, t'] \lhd x_p') && (\textit{Predicate}) \\
&\qquad \land\ \forall t^* \in [t, t'], x_p(t^*) \in H && (\textit{Evolution Dom.})
\end{aligned}
$$

- $\mathcal{P}$ is the BAP for continuous variable x_p on the time interval $[t, t']$
- H a *safety envelope* the value $x_p(t^*)$ belongs to

Note: *shorthand for differential equations:*

$$x_p{:}\sim_{t \to t'} \mathcal{E} \& H \equiv x_p :|_{t \to t'} \mathrm{solutionOf}([t, t'], \mathcal{E}, x_p') \,\&\, H$$

- *solutionOf* a specific predicate for $\mathcal{P}$ from the theory of differential equations
- associated to Well-Definedness conditions

Continuous BAP – Properties

CBAP associated to particular proved **meta-theorems**:

Well-Definedness: assignment is well-defined iff

1- Time progression $t < t'$

2- Type/domain coherence $\forall u, v \cdot \mathcal{P}(u, v) \Rightarrow u \in \mathbb{R}^+ \pfun S \land v \in \mathbb{R}^+ \pfun S \land [0, t[\subseteq \mathrm{dom}(u) \land [t, t'] \subseteq \mathrm{dom}(v)$

Feasibility: there exists $x_p^0 \in \mathbb{R}^+ \pfun S$ with $[t, t'] \subseteq \mathrm{dom}(x_p^0)$ s.t.:

1- Predicate holds $\mathcal{P}([0, t] \lhd x_p, x_p^0)$

2- Evolution domain holds $\forall t^* \in [t, t'], x_p^0(t^*) \in H$

$\Rightarrow$ *reachability of next state* t'

Invariant preservation : for establishing invariant $\mathcal{I} \subseteq S$ on $[0, t']$, it is sufficient that:

Continuous induction

1- $\forall t^* \in [0, t[, x_p(t^*) \in \mathcal{I}$
2- $\mathrm{CBAP}(t, t', x_p, x_p', \mathcal{P}, H \cap \mathcal{I})$

$\Rightarrow$ instantiated *to discharge POs for continuous events*

Relevant theories

Externally defined theories for

- Relations, Rings, Reals, Functions,
- ...

are imported to handle continuous behaviours of hybrid systems

THEORY *Reals* **IMPORT** *Relations* , *Rings*
AXIOMATIC DEFINITIONS
TYPES $\mathbb{R}$
OPERATORS
$Rzero : \mathbb{R}$,
$Rone : \mathbb{R}$,
$plus : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$,
$times : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$,
$leq : \mathbb{R} \leftrightarrow \mathbb{R}$,
...
AXIOMS
order : $order(leq) \land total(leq)$
reals : $Field(plus, times, Rzero, Rone) \land$
...

Continuous behaviours: continuous BAP

THEORY *DiffEq* **IMPORT** *Functions*
TYPE PARAMETERS E , F
DATATYPES
$\mathrm{DE}(F) \equiv \mathrm{ode}(f, \eta_0, t_0) \mid \ldots$
OPERATORS
solutionOf **predicate** $(D : \mathbb{P}(\mathbb{R}), \eta : \mathbb{R} \nrightarrow F, \mathcal{E} : \mathrm{DE}(F))$

Solvable **predicate** $(D : \mathbb{P}(\mathbb{R})\,,\ \mathcal{E} : \mathrm{DE}(F))$
CBAP **predicate** $(t, t' : \mathbb{R}^+, x_p, x_p' : \mathbb{R} \nrightarrow F,$
$\mathcal{P} : \mathbb{P}((\mathbb{R} \nrightarrow F) \times (\mathbb{R} \nrightarrow F)), H : \mathbb{P}(F))$
well−definedness condition
$[0, t] \subseteq \mathrm{dom}(x_p) \wedge [0, t'] \subseteq \mathrm{dom}(x_p')$
direct definition ...

$:\sim$ **predicate** $(t, t' : \mathbb{R}^+\,,\ x_p, x_p' : \mathbb{R} \nrightarrow F\,,$
$\mathcal{E} : \mathrm{DE}(F)\,,\ H : \mathbb{P}(F))$
well−definedness condition
$[0, t] \subseteq \mathrm{dom}(x_p) \wedge [0, t'] \subseteq \mathrm{dom}(x_p') \wedge \mathrm{Solvable}([t, t'], \mathcal{E})$
direct definition $\mathrm{solutionOf}([t, t'], x_p', \mathcal{E}) \wedge \ldots$
AXIOMS
CauchyLipschitz: $\forall \mathcal{E}, D, D_F \cdot \mathcal{E} \in \mathrm{DE}(F) \wedge \ldots$
$\Rightarrow \mathrm{Solvable}(D, \mathcal{E})$
THEOREMS
CBAPINV: $\forall t, t', \eta, \eta', \mathcal{P}, H, \mathcal{I} \cdot \ldots$
$\eta(t) \in H \wedge (\forall t^* \cdot t^* \in [0, t] \Rightarrow \eta(t^*) \in \mathcal{I}) \wedge$
$\mathrm{CBAP}(t, t', \eta, \eta', \mathcal{P}, \mathcal{I} \cap H)$
$\Rightarrow (\forall t^* \cdot t^* \in [0, t'] \Rightarrow \eta'(t^*) \in \mathcal{I})$

An axiomatic theory for differential equations

- *ode*
- *solutionOf* solution of a Diff Eq
- *Solvable* a Diff Eq has a solution
- *CBAP* continuous BAP
- $:\sim$ Shorthand with a Diff Eq $\mathcal{E}$
- Condition for solvability: Cauchy-Lipshitz theorem
- Invariant preservation theorem *CBAPINV*

Generic Model

MACHINE Generic
VARIABLES t, x_s, x_p
INVARIANTS
inv1: $t \in \mathbb{R}^+$
inv2: $x_s \in STATES$
inv3: $x_p \in \mathbb{R} \pfun S$
inv4: $[0, t] \subseteq \mathrm{dom}(x_p)$

- Explicit dense time t (read-only)
- Discrete state x_s (or *mode*)
- Continuous state x_p as function of time, at least defined on $[0, t]$

Generic Model (Cont'd)

Event parameters for genericity

Sensing with guard on continuous state and discrete state (grd3)

Sense
ANY s, p
WHERE
grd1: $s \in \mathbb{P}1(\text{STATES})$
grd2: $p \in \mathbb{P}(\text{STATES} \times \mathbb{R} \times S)$
grd3: $(x_s \mapsto t \mapsto x_p(t)) \in p$
THEN
act1: $x_s :\in s$
END

Actuate
ANY $\mathcal{P}$, s, H, t'
WHERE
grd0: $t' > t$
grd1: $\mathcal{P} \in (\mathbb{R}^+ \nrightarrow S) \times (\mathbb{R}^+ \nrightarrow S)$
grd2: Feasible$([t, t'], x_p, \mathcal{P}, H)$
grd3: $s \subseteq \text{STATES}$
grd4: $x_s \in s$
grd5: $H \subseteq S$
grd6: $x_p(t) \in H$
THEN
act1: $x_p :|_{t \to t'} \mathcal{P}(x_p, x_p') \,\&\, H$
END

Model plant's behaviour

Continuous event based on CBAP
$\Rightarrow$ *generic continuous behaviour* $\mathcal{P}$

Feasibility: **Feasible** guard

Associated discrete state

Constrained by evolution domain

Example: Stopping Car

MACHINE Car **REFINES** Generic
VARIABLES t, x_s, p, v
INVARIANTS
inv31—32: $p \in \mathbb{R} \nrightarrow S$, $v \in \mathbb{R} \nrightarrow S$
inv41—42: $[0, t] \subseteq \mathrm{dom}(p), [0, t] \subseteq \mathrm{dom}(v)$
inv5: $x_p = [\, v \; p \,]^\top$
inv6: $\forall t^* \cdot t^a st \in [0, t] \Rightarrow p(t) \leq SP$

sense_close **REFINES** **Sense**
WHERE **grd1**: $x_s = l_0$
grd2: $p(t) + v(t)^2/2 \geq SP$
WITH $s : s = \{l_1\}$
$p : p = \{p^*, v^* \mid p^* + v^{*2}/2 \geq SP\}$
THEN **act1**: $x_s := l_1$
END

actuate_move **REFINES** **Actuate**
ANY t'
WHERE **grd0**: $t' > t$
grd1: $x_s = l_0$
grd2: $p(t) + v(t)^2/2 < SP$
WITH $eq : eq = \mathrm{ode}(f_{move}, [\, v(t) \; p(t) \,]^\top, t)$
$s : s = \{l_0\}$
$x'_p : x'_p = [\, v' \; p' \,]^\top$
$H : H = \{v^*, p^* \mid p^* + v^{*2}/2 \geq SP\}$
THEN **act1**: $v, p :\sim_{t \to t'}$
$\mathrm{ode}(f_{move}, [\, v(t) \; p(t) \,]^\top, t)$
$\&\{v^*, p^* \mid p^* + v^{*2}/2 \geq SP\}$

- **Instantiation = refinement**

  $\Rightarrow$ **witnesses** *(WITH) and* **gluing invariant** *(inv5) provided*

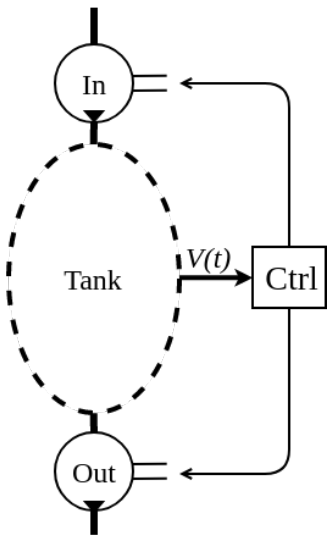
- Continuous behaviour = **ODE**

  $\Rightarrow$ *ODE solvability* **required by WD of** $:\sim_{t \to t'}$ $\Rightarrow$ *by GS:*

  $$\textit{solvability} \Rightarrow \text{Feasible}$$

  Several architecture patterns as instances of this generic model have been developed/published.

S2M, Centralised control (Example)

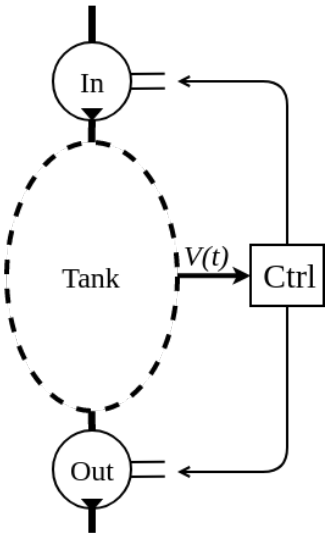


Abstract tank, volume $V(t)$

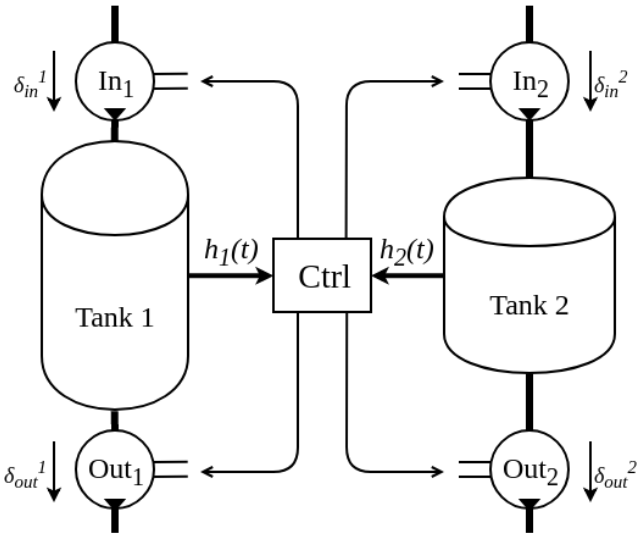
Controller state x_s
(Filling, Emptying, ...)
$\Rightarrow$ control In/Out pumps

Safety: $V_{low} \leq V \leq V_{high}$

S2M, Centralised control (Example)



$\Rightarrow$

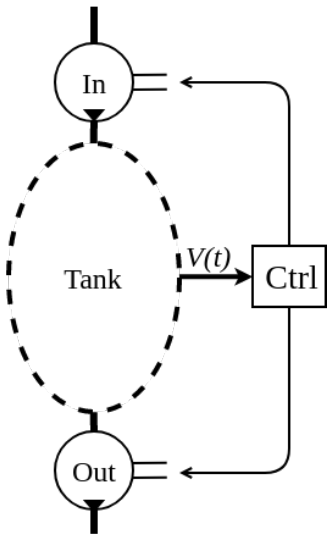


Abstract tank, volume $V(t)$

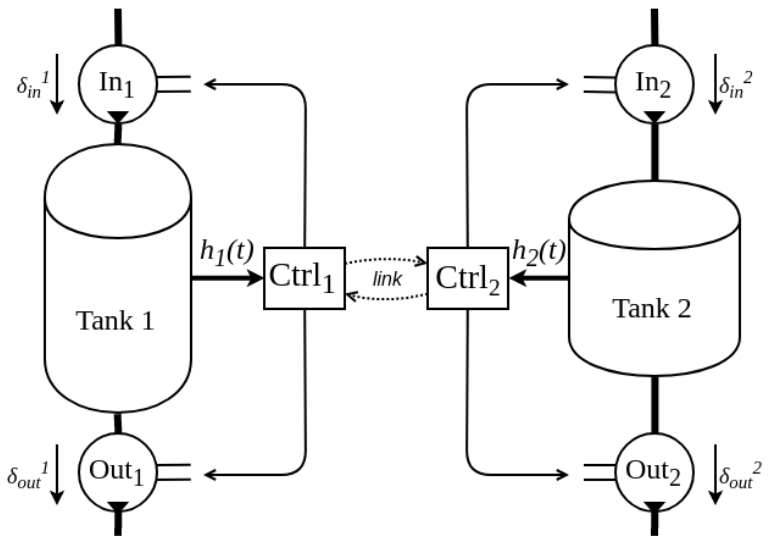
Controller state x_s
(Filling, Emptying, ...)
$\Rightarrow$ control In/Out pumps

Safety: $V_{low} \leq V \leq V_{high}$

M2M, Distributed HS (Example)



$\Rightarrow$



Abstract tank, volume $V(t)$

Controller state x_s
(Filling, Emptying, ...)
$\Rightarrow$ control In/Out pumps

Safety: $V_{low} \leq V \leq V_{high}$

Assessment

- Application to train controller (Davis Diff. Eq)
- Reachability analysis with JuliaREACh - Collab. with Uni. Newcaste
- Several case studies
- Generation of Simulink models
- Approximate refinement - Linearisation

Points of attention

- Alignment with Simulink models
- Complexity of the proofs
- Discretisation and Floating point numbers

Domain models as ontologies

Design of Critical Interactive Systems

- Descriptive standards with rules and regulations for CIS
- Designers must conform to the standards
- ARINC[2] 661 standard describes Cockpit Display System (CDS) standard for communication protocols between interface objects and aircraft systems
  - Widespread use in the industry ⇒ i.e., Airbus A380 and Boeing B787.
  - 800 pages of definitions and requirements for the CDS and its graphical objects
  - In Our work ⇒ we focus on the widget aspects (chapter 3.0).

  Several developed case studies: Weather Radar System (WXR), Traffic Collision Avoidance System (TCAS), ...

Modelling ARINC661 as an ontology

- Need to define ontologies in Event-B

  ⇒ Define an Event-B Theory
- CIS components
  - are instances of the ontology
  - manipulated by Event-B models

[2] Aeronautical Radio, Incorporated

Ontologies as Event-B theories

Ontologies Modelling Language (OML) - DATATYPE

THEORY *OntologiesTheory*
TYPE PARAMETERS
C ,
P ,
I
DATATYPES
Ontology(C, P, I) $\equiv$
consOntology(
classes : $\mathbb{P}(C)$,
properties : $\mathbb{P}(P)$,
instances : $\mathbb{P}(I)$,
classProperties : $\mathbb{P}(C \times P)$,
classInstances : $\mathbb{P}(C \times I)$,
classAssociations : $\mathbb{P}(C \times P \times C)$,
instanceAssociations : $\mathbb{P}(I \times P \times I)$
)

OPERATORS
. . .
END

- **Ontology(C, P, I)** : a generic data type for **classes**, **properties**, **instances**.
- Specifying class properties, class associations and classe intances
- Constrained instantiation: `instancePropertyValues` & `isWDInstancePropertyValues`.
- **37 operators**

Ontologies as Event-B theories

Ontologies Modelling Language (OML) - Operators

THEORY *OntologiesTheory*

...

OPERATORS

isWDInstancesAssociations **predicate** $(o : Ontology(C, P, I))$

well—definedness $isWDClassProperites(o) \land isWDClassInstances(o) \land isWDClassAssociations(o)$

direct definition

$instanceAssociations(o) \subseteq instances(o) \times properties(o) \times instances(o) \land$

$instanceAssociations(o) \subseteq \{i1 \mapsto p \mapsto i2 \mid i1 \in I \land p \in P \land i2 \in I \land$

$i1 \mapsto p \mapsto i2 \in instances(o) \times properties(o) \times instances(o) \land$

$(\exists c1, c2 \cdot c1 \in C \land c2 \in C \land \{c1, c2\} \subseteq getClasses(o)$

$\Rightarrow (c1 \mapsto p \mapsto c2 \in getClassAssociations(o) \land p \in getClassProperties(o)[\{c1\}] \land$

$i1 \in getClassInstances(o)[\{c1\}] \land i2 \in getClassInstances(o)[\{c2\}]))\}$

getInstanceAssociations **expression** $(o : Ontology(C, P, I))$

well—definedness $isWDInstancesAssociations(o)$

direct definition $instanceAssociations(o)$

isWDOntology **predicate** $(o : Ontology(C, P, I))$

direct definition

$isWDClassProperties(o) \land isWDClassInstances(o) \land$

$isWDClassAssociations(o) \land isWDInstancesAssociations(o)$

CheckOfSubsetOntologyInstances **predicate** $(o : Ontology(C, P, I), ipvs : \mathbb{P}(I \times P \times I))$

well—definedness $isWDOntology(o)$

direct definition

$ipvs \subseteq \{i1 \mapsto p \mapsto i2 \mid i1 \in I \land p \in P \land i2 \in I \land i1 \mapsto p \mapsto i2 \in instances(o) \times properties(o) \times$

$instances(o) \land ...\}$

isA **predicate** $(o : Ontology(C, P, I), c1 : C, c2 : C) \cdots$

...

THEOREMS

...

END

Ontologies as Event-B theories

Ontologies Modelling Language (OML) - Theorems

Useful theorems are proved.

THEORY *OntologiesTheory*
TYPE PARAMETERS
C ,
P ,
I
DATATYPES
$Ontology(C, P, I)$
. . .
OPERATORS
. . .
THEOREMS

thm1 : $\forall o, c1, c2, c3 \cdot o \in Ontology(C, P, I) \land isWDOntology(o) \land c1 \in C \land c2 \in C \land c3 \in C \land ontologyContainsClasses(o, \{c1, c2, c3\}) \Rightarrow (\mathbf{isA}(o, c1, c2) \land \mathbf{isA}(o, c2, c3) \Rightarrow \mathbf{isA}(o, c1, c3))$

. . .
END

ARINC 661 Standard - ARINC661Theory

- Types of widgets $\Longrightarrow$ classes, i.e. RadioBox
- Widget's attributes $\Longrightarrow$ properties, i.e. hasChildrenForRadioBox
- Widgets $\Longrightarrow$ instances of the ontology, i.e. LabelInstances
- consA661Ontology $\Longrightarrow$ correct instantiation of ARINC 661 ontology
- operators and axioms $\Longrightarrow$ actions and constrains i.e. isWDRadioBox.
- Theorems of ARCINC 661 ontology (ARINC661Theory) are proved once and for all
- **54 operators** and **17 axioms** were needed for chapter 3 of the ARINC 661 standard

THEORY ARINC661Theory IMPORT OntologiesTheory
Types A661C, A661P, A661I
OPERATORS
RadioBox expression () : A661C
hasChildrenForRadioBox expression () : A661P
LabelInstances expression () : $\mathbb{P}$(A661I)
consA661Ontology expression (ii, cii, ipvs)
. . .
isWDRadioBox predicate (o)
well-definedness isWDOntology(o)
. . .
isWDChangeModeSelection predicate (o: Ontology(...), ui, mode):
changeModeSelection expression (...) : $\mathbb{P}$(...)
well-definedness isWDChangeModeSelection(o, ui, mode)

isWDChangeTiltAngle predicate (o: Ontology(...), ui, mode):
changeTiltAngle expression (...) : $\mathbb{P}$(...)
well-definedness isWDChangeTiltAngle(o, ui, mode)

AXIOMS
A661CDef: partition(A661C, {Label}, {RadioBox}, {CheckButton}, . . .)
consA661OntoDef: ... $\Rightarrow$ consARINC661Ontology(...) = consOntology(...)
isWDRadioBoxDef:$\forall$o· o $\in$ A661Ontology(...) $\Rightarrow$ (isWDRadioBox(o) $\Leftrightarrow$ ($\forall$ ...)

ARINC 661 Standard - ARINC661Theory

- Types of widgets $\Longrightarrow$ classes, i.e. RadioBox
- Widget's attributes $\Longrightarrow$ properties, i.e. hasChildrenForRadioBox
- Widgets $\Longrightarrow$ instances of the ontology, i.e. LabelInstances
- consA661Ontology $\Longrightarrow$ correct instantiation of ARINC 661 ontology
- operators and axioms $\Longrightarrow$ actions and constrains i.e. isWDRadioBox.
- Theorems of ARCINC 661 ontology (ARINC661Theory) are proved once and for all
- **54 operators** and **17 axioms** were needed for chapter 3 of the ARINC 661 standard

THEORY ARINC661Theory IMPORT OntologiesTheory
Types A661C, A661P, A661I
OPERATORS
RadioBox expression () : A661C
hasChildrenForRadioBox expression () : A661P
LabelInstances expression () : $\mathbb{P}$(A661I)
consA661Ontology expression (ii, cii, ipvs)
. . .
isWDRadioBox predicate (o)
well-definedness isWDOntology(o)
. . .
isWDChangeModeSelection predicate (o: Ontology(...), ui, mode):
changeModeSelection expression (...) : $\mathbb{P}$(...)
well-definedness isWDChangeModeSelection(o, ui, mode)

isWDChangeTiltAngle predicate (o: Ontology(...), ui, mode):
changeTiltAngle expression (...) : $\mathbb{P}$(...)
well-definedness isWDChangeTiltAngle(o, ui, mode)

AXIOMS
A661CDef: partition(A661C, {Label}, {RadioBox}, {CheckButton}, . . .)
consA661OntoDef: ... $\Rightarrow$ consARINC661Ontology(...) = consOntology(...)
isWDRadioBoxDef:$\forall$o· o $\in$ A661Ontology(...) $\Rightarrow$ (isWDRadioBox(o) $\Leftrightarrow$ ($\forall$ ...)

THEOREMS
RadioButtonsExlusiveThm
$\forall$rb, b1, b2· rb $\in$ RadioBoxInstances $\wedge$
b1, b2 $\in$ CheckButtonInstances $\wedge$
rb $\mapsto$ hasChildrenForRadioBox $\mapsto$ b1 $\in$ ui $\wedge$
rb $\mapsto$ hasChildrenForRadioBox $\mapsto$ b2 $\in$ ui $\wedge$
b1 $\mapsto$hasCheckButtonState$\mapsto$SELECTED$\in$ui
b2 $\mapsto$hasCheckButtonState$\mapsto$SELECTED$\in$ui
$\Rightarrow$ b1=b2
. . .

Case study - Weather Radar System - Multi-Purpose Interactive Application WXR-MPIA

Requirements

The pilot interacts with this application (Mode selection, angle selection, etc.).

- The tilt angle must be within a specific range in [-15°, +15°]
- The selection of the Radio box button shall be exclusive

- Widgets ⇒ formalised as instances
- Action on the widgets ⇒ theory operators
- Properties of the application ⇒ Proved as theorems

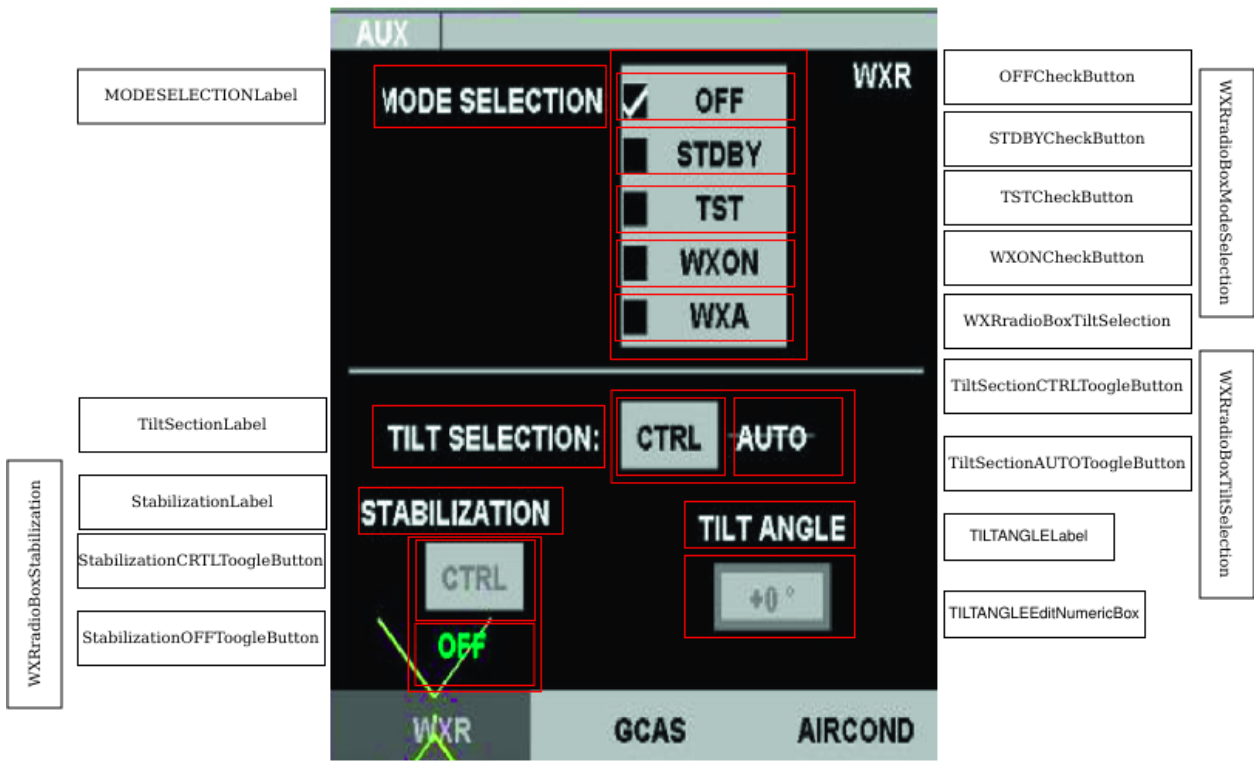


Figure: Dassault Falcon 6X cockpit

Case study - Weather Radar System - Multi-Purpose Interactive Application WXR-MPIA

Annotation of MPIA model $\Rightarrow$ typing the state variable MPIA

Only data types and operators of `ARINC661Theory` are used $\Rightarrow$ theorem transferring

THEORY ARINC661Theory
IMPORT OntologiesTheory
...
THEOREMS
RadioButtonsExlusiveThm
$\forall o, ui \cdot ui \in ... \Rightarrow ($
$\forall rb, b1, b2 \cdot rb \in RadioBoxInstances \land$
$b1, b2 \in CheckButtonInstances \land$
$rb \mapsto hasChildrenForRadioBox \mapsto b1 \in ui \land$
$rb \mapsto hasChildrenForRadioBox \mapsto b2 \in ui \land$
$b1 \mapsto hasCheckButtonState \mapsto SELECTED \in ui$
$b2 \mapsto hasCheckButtonState \mapsto SELECTED \in ui$
$\Rightarrow b1 = b2)$
...

Case study - Weather Radar System - Multi-Purpose Interactive Application WXR-MPIA

Annotation of MPIA model ⇒ typing the state variable MPIA
Only data types and operators of `ARINC661Theory` are used ⇒ theorem transferring

THEORY ARINC661Theory
IMPORT OntologiesTheory
...
THEORIES
RadioButtonsExlusiveThm
∀o, ui· ui ∈ ... ⇒ (
∀rb, b1, b2· rb ∈ RadioBoxInstances ∧
b1, b2 ∈ CheckButtonInstances ∧
rb ↦ hasChildrenForRadioBox ↦ b1 ∈ ui ∧
rb ↦ hasChildrenForRadioBox ↦ b2 ∈ ui ∧
b1 ↦hasCheckButtonState↦SELECTED∈ui
b2 ↦hasCheckButtonState↦SELECTED∈ui
⇒ b1=b2)
...

Machine *MPIAModel*
VARIABLES *MPIAUI*
INVARIANTS
typing: MPIAUI ∈ ℙ(A661I × A661P × A661I) ∧
UsedOps: ∃*uiArg* · ... ∨
∃*m* · (*isWDChangeModeSelection*(*A661Ontology*, *uiArg*, *m*) ∨
∃*a* · (*isWDChangeTiltAngle*(*A661Ontology*, *uiArg*, *a*) ...
THEOREMS
RadioButtonsExlusiveThmInst
∀rb, b1, b2· rb ∈ RadioBoxInstances ∧ b1, b2 ∈ CheckButtonInstances ∧
rb ↦ hasChildrenForRadioBox ↦ b1 ∈ MPIAUI ∧
rb ↦ hasChildrenForRadioBox ↦ b2 ∈ MPIAUI ∧
b1 ↦hasCheckButtonState↦SELECTED∈MPIAUI
b2 ↦hasCheckButtonState↦SELECTED∈MPIAUI
⇒ b1=b2)
...
EVENTS
userClicksOnMode **ANY** *NewMode*
WHERE *isWDChangeModeSelection*(*A661Ontology*, *MPIAUI*, *NewMode*)
THEN *MPIAUI* := *changeModeSelection* (*A661Ontology*, *MPIAUI*, *NewMode*)
END

UserChangeTitlAngle **ANY** *newAngle*
WHERE *isWDChangeTitlAngle*(*A661Ontology*, *MPIAUI*, *newAngle*)
THEN *MPIAUI* := *changeTitleAngle* (*A661Ontology*, *MPIAUI*, *mode*)
END
...
END

Assessment

Use of domain knowledge theories

- Proof effort is transferred on the theory side
- Domain theories theorems are proved once and for all
- Definition of several ontologies
- Study of other domains: embedded systems
- First steps towards standard conformance by construction
- Current studies for railway systems

Points of attention

- Development of such domain theories may be resource consuming
- Impact of domain theories evolution
- Consistence of axioms

Reflexive Event-B

Reasoning on Event-B itself

- Manipulation of Event-B concepts (machines, guards, events, states, ...)
- Exchange format for Event-B models
- Definition of other analyses of Event-B models

A theory for Event-B Machines

EB4EB Meta theory Principles

- A **Machine** data type is parameterised by two TYPE parameters
  - **STATE** for machine states (state variables)
  - **EVENT** for machine Events
- **Machine** data type components
  - A single **Init** event and a set **Progress** of events (transitions)
  - **AP**(After Predicate - initialisation) and **BAP** Before-After Predicate - events. Defined as relations on states and Events
  - **Grd** as a set of states
  - **Inv** a set of safe states

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS
DATATYPES
Machine( ) $\triangleq$
$\triangleright$ *Cons_machine*(
. . .)
OPERATORS
. . .

Machine

MACHINE M **SEES** . . .
VARIABLES x
INVARIANTS $I(x)$
. . .
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
. . .)
OPERATORS
. . .

STATE: machine states

EVENT: machine events

Machine

MACHINE M **SEES** . . .
VARIABLES x
INVARIANTS $I(x)$
. . .
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
...)
OPERATORS
...

STATE: machine states

EVENT: machine events

Init: initialisation event

Progress: progress events

Machine

MACHINE *M* **SEES** ...
VARIABLES *x*
INVARIANTS $I(x)$
...
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
...)
OPERATORS
...

STATE: machine states

EVENT: machine events

Init: initialisation event

Progress: progress events

AP (After Predicate): initialisation event action

BAP (Before-After Predicate): actions of progress events

Machine

MACHINE *M* **SEES** ...
VARIABLES *x*
INVARIANTS *I*(*x*)
...
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),

...)
OPERATORS
...

STATE: machine states

EVENT: machine events

Init: initialisation event

Progress: progress events

AP (After Predicate): initialisation event action

BAP (Before-After Predicate): actions of progress events

Grd: progress events guards

Machine

MACHINE *M* **SEES** ...
VARIABLES x
INVARIANTS $I(x)$
...
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
...)
OPERATORS
...

STATE: machine states

EVENT: machine events

Init: initialisation event

Progress: progress events

AP (After Predicate): initialisation event action

BAP (Before-After Predicate): actions of progress events

Grd: progress events guards

Inv: machine invariants

Machine

MACHINE *M* **SEES** ...
VARIABLES x
INVARIANTS $I(x)$
...
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
...)
OPERATORS
...
check_Machine_consistency(m)...

STATE: machine states

EVENT: machine events

Init: initialisation event

Progress: progress events

AP (After Predicate): initialisation event action

BAP (Before-After Predicate): actions of progress events

Grd: progress events guards

Inv: machine invariants

Check_Machine_Consistency: core Event-B POs

Machine

MACHINE *M* **SEES** ...
VARIABLES *x*
INVARIANTS $I(x)$
...
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

The EB4EB Meta Theory

- Explicit manipulation of Event-B features
  $\Longrightarrow$ a constructive data-type for Event-B machines.

EventBTheory

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
...)
OPERATORS
...
check_Machine_consistency(m)...

STATE: machine states

EVENT: machine events

Init: initialisation event

Progress: progress events

AP (After Predicate): initialisation event action

BAP (Before-After Predicate): actions of progress events

Grd: progress events guards

Inv: machine invariants

Check_Machine_Consistency: core Event-B POs

Machine

MACHINE *M* **SEES** ...
VARIABLES x
INVARIANTS $I(x)$
...
EVENTS
INITIALISATION $\triangleq$
THEN
$x :| AP(\alpha, x)$
END
$EVT_i \triangleq$
ANY α
WHERE $G_i(x, \alpha)$
THEN
$x :| BAP_i(\alpha, x, x)$
END
...

Check_Machine_Consistency(m) $\equiv$ PO_INV(m) $\wedge$ PO_THM(m) $\wedge$ PO_FIS(m) $\wedge$ PO_VAR(m) $\wedge$ PO_NAT(m)

Instantiation of the EB4EB Meta Theory

Event-B Machines as a FOL expression

- An Event-B machine is seen as a FOL formula (sets, constants, and axioms)

CONTEXT Deep
SETS St Ev
CONSTANTS m, ...
AXIOMS
axm1: $partition(Ev, \ldots)$
axm2: $m \in Machine(St, Ev)$
axm3: $Event(m) = Ev$
axm4: $State(m) = \ldots$
axm5: $Init(m) = \ldots$
axm6: $Progress(m) = \{\ldots\}$
axm7: $Thm(m) = \{\ldots\}$
axm8: $Inv(m) = \{\ldots\}$
axm9: $AP(m) = \{\ldots\}$
axm10: $Grd(m) = \{\ldots\}$
axm11: $BAP(m) = \{\ldots\}$
axm12: $Convergent(m) = \{\ldots\}$
axm13: $Ordinary(m) = \{\ldots\}$
axm14: $Variant(m) = \{\ldots\}$
THEOREMS
thm1: $\boldsymbol{check\_Machine\_consistency}(m)$
END

Machine consistency is ensured by discharging the *THM_ctx* PO generated for the thm1 theorem

$$axm1 \wedge \ldots \wedge axm14 \Rightarrow Check\_Machine\_consistency(m)$$

A theory for Event-B Machines. Proof Obligations

State-transitions semantics defined on Events Grd and BAP

Each PO is described as an operator of the theory with a WD condition

Invariant PO (INV)

The **Invariant** Proof Obligation rule

$$\frac{M \vdash AP(\alpha, x') \Rightarrow I \qquad M \vdash \forall i \cdot G_i(\alpha, x) \wedge BAP_i(\alpha, x, x') \wedge I(x) \Rightarrow I(x')}{M \vdash INV}$$

Associated operators **Predicate** Operator

Invariant for Initialisation

Mch_INV_Init predicate $(m : Machine(STATE, EVENT))$
direct definition $AP(m) \subseteq Inv(m)$

Invariant for a single Event (Inductive case)

Mch_INV_OneEvt predicate $(m : Machine(STATE, EVENT), e : \mathbb{P}(EVENT))$
well—definedness $e \in Progress(m)$
direct definition $BAP(m)[\{e\}][Inv(m) \cap Grd(m)[\{e\}]] \subseteq Inv(m)$

Invariant Proof Obligation (proved by induction on sets)

Mch_INV predicate $(m : Machine(STATE, EVENT))$
direct definition $Mch\_INV\_Init(m) \wedge (\forall \boldsymbol{e} \cdot e \in Progress(m) \Rightarrow Mch\_INV\_OneEvt(m, e))$

Correctness of the proposed formalisation

Semantics & Traces

A trace of Machine M is a sequence of states $tr = s_0 \mapsto s_1 \mapsto \ldots \mapsto s_n \mapsto \ldots$ such that

- initial state s_0 satisfies the after predicate (AP) of the initialisation event
- consecutive states s_i, s_{i+1} corresponds to the activation of an event e of M, i.e.
  - the guard of e holds for s_i
  - the BAP of e holds for $s_i \mapsto s_{i+1}$

  if tr is *finite*, then its final state must correspond to a deadlock (i.e., no progress anymore)

Inductive reasoning on traces.

THEORY *EvtBTraces* IMPORT *EventBTheory*
TYPE PARAMETERS *STATE*,*EVENT*
OPERATORS

IsANextState predicate $(m : Machine(STATE, EVENT), s : STATE, sp : STATE)$
direct definition
$\exists e \cdot e \in Progress(m) \wedge s \in Grd(m)[\{e\}] \wedge s \mapsto sp \in BAP(m)[\{e\}]$

IsATrace predicate $(m : Machine(STATE, EVENT), tr : \mathbb{P}(\mathbb{N} \times STATE))$
direct definition
$(tr \in \mathbb{N} \rightarrow STATE \vee$
$(\exists n \cdot n \in \mathbb{N} \wedge tr \in 0..n \rightarrow STATE \wedge tr(n) \notin Grd(m)[Progress(m)])$ // finite tr
$) \wedge$
$tr(0) \in AP(m) \wedge$
$(\forall i, j \cdot i \in dom(tr) \wedge j \in dom(tr) \wedge j = i + 1 \Rightarrow IsANextState(m, tr(i), tr(j)))$

THEOREMS
...
END

Correctness of the proposed formalisation

Soundness is expressed on each defined operator

- A template for proving soundness

THEORY Theo4[PO] Correctness IMPORT EvtBTraces, Theo4[PO]
TYPE PARAMETERS *STATE*, *EVENT*
THEOREMS
thm_of_Correctness_of_[PO] : $\forall m, tr \cdot m \in Machine(STATE, Ev) \wedge$
$check\_Machine\_Consistency(m) \wedge IsATrace(tr, m) \wedge$ **[PO](m,args)**
$\Rightarrow$ ***PO_Spec_On_Traces(...)***

- For each Proof Obligation defined predicate operator $\Longrightarrow$ a proven soundness correctness theorem.

Correctness of the proposed formalisation

Soundness is expressed on each defined operator

- A template for proving soundness

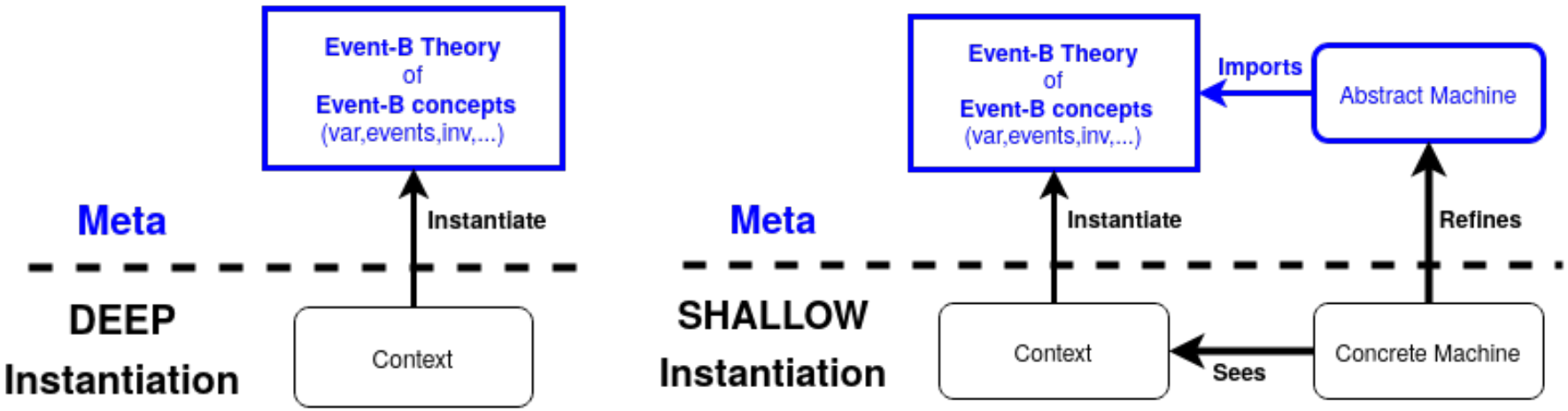
THEORY Theo4[PO] Correctness IMPORT EvtBTraces, Theo4[PO]
TYPE PARAMETERS $STATE, EVENT$
THEOREMS
thm_of_Correctness_of_[PO] : $\forall m, tr \cdot m \in Machine(STATE, Ev) \wedge$
$check\_Machine\_Consistency(m) \wedge IsATrace(tr, m) \wedge$ **[PO](m,args)**
$\Rightarrow$ ***PO_Spec_On_Traces(...)***

- For each Proof Obligation defined predicate operator $\Longrightarrow$ a proven soundness correctness theorem.
- Case of soundness of the Invariant proof obligation definition.

THEOREMS
thm_of_Correctness_of_Invariant : $\forall m, tr.$
$m \in Machine(STATE, EVENT) \wedge IsATrace(m, tr) \wedge$ **MCH_INV**$(m) \Rightarrow$
$(tr(0) \in Inv(m) \wedge$
$(\forall i \cdot i \in dom(tr).tr(i) \in Inv(m) \wedge i + 1 \in dom(tr) \Rightarrow tr(j) \in Inv(m))$
...

Theory Instantiation Mechanisms

Instantiate a Machine by: **SHALLOW** modelling



- **Deep modelling.** Machine instances as an Event-B **context** by instantiating generic type parameters **STATE** and **EVENT** of the meta theory with machine state variables and events.
  - Proof of Theorems of the obtained context
  - An Event-B model is seen as a FOL formula
- **Shallow modelling.** Machine instances as an Event-B **context** + a **refinement** of the abstract state variables and events (Init + Progress) of a Event-B generic machine *ShallowGen*.
  - Event-B induction is set up using ***Mch_INV_Init*** and ***Mch_INV_One_Ev*** EB4EB operators
  - No need to use the ***Mch_Inv*** operator or **all** Events

Deep Modelling. A Machine as Context

A context to instantiate EB4EB meta-Theory

CONTEXT Deep
SETS Ev
CONSTANTS m, ...
AXIOMS
axm1: $partition(Ev, \ldots)$
axm2: $m \in Machine(\ldots, Ev)$
axm3: $Event(m) = Ev$
axm4: $State(m) = \ldots$
axm5: $Init(m) = \ldots$
axm6: $Progress(m) = \{\ldots\}$
axm7: $Thm(m) = \{\ldots\}$
axm8: $Inv(m) = \{\ldots\}$
axm9: $AP(m) = \{\ldots\}$
axm10: $Grd(m) = \{\ldots\}$
axm11: $BAP(m) = \{\ldots\}$
axm12: $Convergent(m) = \{\ldots\}$
axm13: $Ordinary(m) = \{\ldots\}$
axm14: $Variant(m) = \{\ldots\}$
THEOREMS
thm1: check_Machine_Consistency(m)
END

- Each Machine data type component is axiomatised
- Predicates are given as set comprehension
- POs are generated by the Event-B THM-Theorem PO

Deep Modelling. A Machine as Context

A context to instantiate EB4EB meta-Theory

CONTEXT Deep
SETS Ev
CONSTANTS m, ...
AXIOMS
axm1: $partition(Ev, \ldots)$
axm2: $m \in Machine(\ldots, Ev)$
axm3: $Event(m) = Ev$
axm4: $State(m) = \ldots$
axm5: $Init(m) = \ldots$
axm6: $Progress(m) = \{\ldots\}$
axm7: $Thm(m) = \{\ldots\}$
axm8: $Inv(m) = \{\ldots\}$
axm9: $AP(m) = \{\ldots\}$
axm10: $Grd(m) = \{\ldots\}$
axm11: $BAP(m) = \{\ldots\}$
axm12: $Convergent(m) = \{\ldots\}$
axm13: $Ordinary(m) = \{\ldots\}$
axm14: $Variant(m) = \{\ldots\}$
THEOREMS
thm1: check_Machine_Consistency(m)
END

Non intrusive analyses

- Additional semantic features (Temporal Logic)
- Define new proof obligations as theorems

e.g. Every **input** event shall be eventually followed by a **confirmation** event

Shallow modelling. The *ShallowGen* Machine.

```
CONTEXT ShallowGen
SETS Ev, S // Instance of Event and abstract State
CONSTANTS machine
AXIOMS
  axm1: machine ∈ Machine(S, Ev)
END
```

```
MACHINE ShallowGenMac SEES ShallowGen
VARIABLES s, InitDone // abstract state
INVARIANTS // variable s satisfy the invariant
  inv1: s ∈ S ∧ InitDone ∈ BOOL ∧ (InitDone = TRUE ⇒ s ∈ Inv(machine))
EVENTS
  INITIALISATION
  THEN
    act1: s, InitDone :| s' ∈ S ∧ InitDone' = FALSE
  END
  Do_Init // do a correct initialisation
  WHERE
    grd1: InitDone = FALSE ∧ Mch_FIS_Init(machine) ∧ Mch_INV_Init(machine)
  THEN
    act1: s, InitDone :| s' ∈ AP(machine) ∧ InitDone' := TRUE
  END
  Do_Progress ANY e // do a correct progress event in the proof obligation in the guard
  WHERE
    grd1: InitDone = TRUE ∧ s ∈ Grd(machine)[{e}] ∧ e ∈ Progress(machine)
    grd2: Mch_FIS_One_Ev(machine, e) ∧ Mch_INV_One_Ev(machine, e)
  THEN
    act1: s :| s' ∈ BAP(machine)[{e}][{s}]
  END
  ...
END
```

Behavioural properties

Annotation of Events

- Introduction of states annotations
- Possible using data-types

Is it possible to extend annotation to events/transitions ?

Properties of interest

Every **input** event shall be eventually followed by a **confirmation** or **abortion** events

Of course, this property can be formalised in ad-hoc manner

Annotation of Events

- No possibility to manipulate events
- Extend the Meta-theory of Event-B and define a new proof obligation

Generation of Proof Obligations

How is a Proof Obligation generated?

Principle

- Well-Definedness (WD) PO
- Theorem (Thm) PO

check_Machine_Consistency **predicate** operator for PO generation

- At instantiation, the predicate shall be **proved as a theorem** for each operator
- Modularisation of the Proof by decomposing operators definitions
- Proof of correctness (Theory of traces)
- Non-intrusive model analysis
  - New PO $\Longrightarrow$ New Predicate Operator
  - PO Generation $\Longrightarrow$ Native THM PO

check_Machine_Consistency<**predicate**>
$(m : Machine(STATE, EVENT))$
well—definedness Machine_WellCons(m)
direct definition $Mch\_THM(m) \wedge$
$Mch\_INV(m) \wedge$
$Mch\_FIS(m) \wedge$
$Mch\_VARIANT(m) \wedge$
$Mch\_NAT(m)$

How to introduce a *"New"* Proof Obligation ?

- Extend the `EventBTheory`

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
. . .)
OPERATORS
. . .
check_Machine_consistency(m). . .

CONTEXT Deep
SETS *St* *Ev*
CONSTANTS m, . . .
AXIOMS
axm1: *partition*(*Ev*, . . .)
axm2: $m \in Machine(St, Ev)$
axm3: $Event(m) = Ev$
axm4: $State(m) = \ldots$
axm5: $Init(m) = \ldots$
axm6: $Progress(m) = \{\ldots\}$
axm7: $Thm(m) = \{\ldots\}$
axm8: $Inv(m) = \{\ldots\}$
axm9: $AP(m) = \{\ldots\}$
axm10: $Grd(m) = \{\ldots\}$
axm11: $BAP(m) = \{\ldots\}$
axm12: $Convergent(m) = \{\ldots\}$
axm13: $Ordinary(m) = \{\ldots\}$
axm14: $Variant(m) = \{\ldots\}$
THEOREMS
thm1: ***check_Machine_consistency***(m)
END

How to introduce a "*New*" Proof Obligation ?

- Extend the `EventBTheory`
- New POs are formalised as a Predicate operator

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
. . .)
OPERATORS
. . .
check_Machine_consistency(m). . .

CONTEXT Deep
SETS *St* *Ev*
CONSTANTS m, . . .
AXIOMS
axm1: *partition*(*Ev*, . . .)
axm2: $m \in Machine(St, Ev)$
axm3: $Event(m) = Ev$
axm4: $State(m) = \ldots$
axm5: $Init(m) = \ldots$
axm6: $Progress(m) = \{\ldots\}$
axm7: $Thm(m) = \{\ldots\}$
axm8: $Inv(m) = \{\ldots\}$
axm9: $AP(m) = \{\ldots\}$
axm10: $Grd(m) = \{\ldots\}$
axm11: $BAP(m) = \{\ldots\}$
axm12: $Convergent(m) = \{\ldots\}$
axm13: $Ordinary(m) = \{\ldots\}$
axm14: $Variant(m) = \{\ldots\}$
THEOREMS
thm1: ***check_Machine_consistency***(m)
END

THEORY Theo4[PO]
IMPORTS EventBTheory
. . .
OPERATORS
. . .
[PO_definition](m)

How to introduce a "*New*" Proof Obligation ?

- Extend the `EventBTheory`
- New POs are formalised as a Predicate operator
- New POs are automatically generated using the `Theorem` clause

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$*Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
. . .)
OPERATORS
. . .
check_Machine_consistency(m). . .

CONTEXT Deep
SETS *St* *Ev*
CONSTANTS m, . . .
AXIOMS
axm1: *partition*(*Ev*, . . .)
axm2: *m* $\in$ *Machine*(*St*, *Ev*)
axm3: *Event*(*m*) = *Ev*
axm4: *State*(*m*) = . . .
axm5: *Init*(*m*) = . . .
axm6: *Progress*(*m*) = {. . .}
axm7: *Thm*(*m*) = {. . .}
axm8: *Inv*(*m*) = {. . .}
axm9: *AP*(*m*) = {. . .}
axm10: *Grd*(*m*) = {. . .}
axm11: *BAP*(*m*) = {. . .}
axm12: *Convergent*(*m*) = {. . .}
axm13: *Ordinary*(*m*) = {. . .}
axm14: *Variant*(*m*) = {. . .}
THEOREMS
thm1: ***check_Machine_consistency***(*m*)
END

THEORY Theo4[PO]
IMPORTS EventBTheory
. . .
OPERATORS
. . .
[PO_definition](m)

CONTEXT Deep[PO] EXTENDS Deep
. . .
THEOREMS
thm[PO]: **[PO_definition]**(m)

How to introduce a "*New*" Proof Obligation ?

- Extend the `EventBTheory`
- New POs are formalised as a Predicate operator
- New POs are automatically generated using the `Theorem` clause

THEORY EventBTheory
TYPES PARAMETERS STATE,EVENT
DATATYPES
Machine(STATE,EVENT) $\triangleq$
$\triangleright$ *Cons_machine*(
Init:*EVENT*,
Progress:$\mathbb{P}$(*EVENT*),
AP:$\mathbb{P}$(*STATE*),
BAP:
$\mathbb{P}$(*EVENT* $\times$ (*STATE* $\times$ *STATE*)),
Grd:
$\mathbb{P}$(*EVENT* $\times$ *STATE*),
Inv:$\mathbb{P}$(*STATE*),
. . .)
OPERATORS
. . .
check_Machine_consistency(m). . .

CONTEXT Deep
SETS *St* *Ev*
CONSTANTS m, . . .
AXIOMS
axm1: *partition*(*Ev*, . . .)
axm2: *m* $\in$ *Machine*(*St*, *Ev*)
axm3: *Event*(*m*) = *Ev*
axm4: *State*(*m*) = . . .
axm5: *Init*(*m*) = . . .
axm6: *Progress*(*m*) = {. . .}
axm7: *Thm*(*m*) = {. . .}
axm8: *Inv*(*m*) = {. . .}
axm9: *AP*(*m*) = {. . .}
axm10: *Grd*(*m*) = {. . .}
axm11: *BAP*(*m*) = {. . .}
axm12: *Convergent*(*m*) = {. . .}
axm13: *Ordinary*(*m*) = {. . .}
axm14: *Variant*(*m*) = {. . .}
THEOREMS
thm1: ***check_Machine_consistency***(*m*)
END

THEORY Theo4[PO]
IMPORTS EventBTheory
. . .
OPERATORS
. . .
[PO_definition](m)

CONTEXT Deep[PO] EXTENDS Deep
. . .
THEOREMS
thm[PO]: **[PO_definition]**(m)

- Several extensions have been defined
- Temporal logic properties with predicate operators (e.g. `IsReachable` operator)

Behavioural domain specific property formalisation and associated PO

Every **input** event shall be eventually followed by a **confirmation** or **abortion** event

- Need to **manipulate** events $\Longrightarrow$ Use the `EventBTheory` theory

Behavioural domain specific property formalisation and associated PO

Every **input** event shall be eventually followed by a **confirmation** or **abortion** event

- Need to **manipulate** events $\Longrightarrow$ Use the `EventBTheory` theory
- Need to **define** event tags $\Longrightarrow$ An ontology of the `OntologiesTheory` theory

THEORY *OntologiesTheory*
TYPE PARAMETERS
C , P , I
DATATYPES
Ontology$(C, P, I) \equiv$ consOntology(
classes : $\mathbb{P}(C)$,
properties : $\mathbb{P}(P)$,
instances : $\mathbb{P}(I)$,
classProperties : $\mathbb{P}(C \times P)$,
classInstances : $\mathbb{P}(C \times I)$,
. . .)
. . .

Behavioural domain specific property formalisation and associated PO

Every **input** event shall be eventually followed by a **confirmation** or **abortion** event

- Need to **manipulate** events $\Longrightarrow$ Use the EventBTheory theory
- Need to **define** event tags $\Longrightarrow$ An ontology of the OntologiesTheory theory
- Need to **annotate** events $\Longrightarrow$ events are ontology instances

THEORY *DomainDynamicPropertiesTheory*
IMPORT THEORY *EventBTheory*, *OntologiesTheory*
TYPE PARAMETERS *St*, *Ev*, *Tg*, *Prop*
OPERATORS
isNecFollowedByWD <predicate> (...)
isNecFollowedBy <predicate>
(*m* : *Mach*(*St*, *Ev*),
eo : *Ontology*(*Tg*, *Prop*, *Ev*),
startTag : *Tg*
middleTag : $\mathbb{P}$(*Tg*),
endTag : $\mathbb{P}$(*Tg*),
v : $\mathbb{P}$(*Ev* $\times$ $\mathbb{P}$(*St* $\times$ $\mathbb{Z}$)))
well−definedness ***isNecFollowedByWD***(...)
direct definition
$\forall$ ***EvtInst*** · ***EvtInst*** $\in$ ***insOfC***(***eo***, ***startTag***) $\Rightarrow$
Is_Reachable(*m*,
EvtInst,
insOfC(*eo*, *endTag*),
insOfC(*eo*, *middleTag*),
v(*EvtInst*))
END

THEORY *OntologiesTheory*
TYPE PARAMETERS
C, *P*, *I*
DATATYPES
Ontology(*C*, *P*, *I*) $\equiv$ consOntology(
classes : $\mathbb{P}$(*C*),
properties : $\mathbb{P}$(*P*),
instances : $\mathbb{P}$(*I*),
classProperties : $\mathbb{P}$(*C* $\times$ *P*),
classInstances : $\mathbb{P}$(*C* $\times$ *I*),
...)
...

- isNecFollowedBy predicate operator formalises the associated property schema
- Is_Reachable Predicate operator borrowed from EventBTheory
- *v*(*EvtInst*) introduces the required variant

Assessment

The Event-B Meta-theory

– Event-B features are manipulated, in particular events
- Intrusive v.s. Non Intrusive analyses
- Refinement is defined as a binary relation
- Manipulation Event-B models at any level of the refinement chain is possible
- An exchange format for Event-B models

Points of attention

- Impact of domain theories evolution
- Consistence of axioms

Conclusion

Event-B + Theories and Invited semantics

- A framework integrating both
  - Event-B method modelling style with built-in induction + Abstract Data Types
- Enrich Event-B modelling language with data-types
- Well-Definedness is useful for PO generation
- Outsourcing: complex proofs at the theory level, once and for all
- Reuse of theorems in formal Event-B models and reduction of proof efforts
- Model annotation through $x_s \in T$

To Do

- More formalised domain theories
- More model annotation through refinement $x_s \in T$ glued with $x_s \mapsto x_a \in T \longleftrightarrow T_a$
- Formalising other refinement operations
- Certification of PlugIns and model transformations
- Build bridges for Event-B Theories with other proof assistants
- An engineering process: what is the level of granularity for axiomatic theories ?
- Reasoning on assurance cases
- ...

Thank You

yamine@n7.fr

Traces and soundenss

```
THEORY NotEmptyList
TYPE PARAMETERS T
DATA TYPES
  NotEmptyList(T) ≡
    base_case(base : T)
    | cons(el : T, next : NotEmptyList(T))
OPERATORS
  AllAre predicate (l : NotEmptyList(T), predicate : ℙ(T))
    recursive definition
      case l :
        base_case(n) ⇒
          n ∈ predicate
        cons(t, q) ⇒
          t ∈ predicate ∧ AllAre(q, predicate)
END
```

```
THEORY EvtBTraces IMPORT EvtBStruc, NotEmptyList
TYPE PARAMETERS STATE, EVENT
OPERATORS
  IsATrace predicate (tr : NotEmptyList(STATE),
        m : Machine(STATE, EVENT))
    recursive definition
      case tr :
        base_case(s) ⇒ s ∈ AP(m)
        cons(sp, q) ⇒
              IsANextState(sp, first(q), m) ∧ IsATrace(q, m)
  IsANextState predicate (sp : STATE, s : STATE,
        m : Machine(STATE, EVENT))
    direct definition
      ∃e · e ∈ Progress(m)∧
          s ∈ Grd(m)[{e}] ∧ s ↦ sp ∈ BAP(m)[{e}])
```

```
THEORY EvtBCorrectness
IMPORT EvtBTraces, EvtBPO
TYPE PARAMETERS STATE, EVENT
THEOREMS
  thm1 :
    ∀m, tr · m ∈ Machine(STATE, EVENT)∧
      Machine_WellCons(m) ∧ IsATrace(tr, m) ∧ Mch_INV(m)
          ⇒ AllAre(tr, Inv(m))
END
```