Journal First: Building Specifications in the Event-B Institution

Marie Farrell, Rosemary Monahan and James F. Power

ABZ 2023
May 31, 2023

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

marie.farrell@manchester.ac.uk

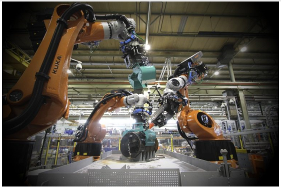
Disclaimer:

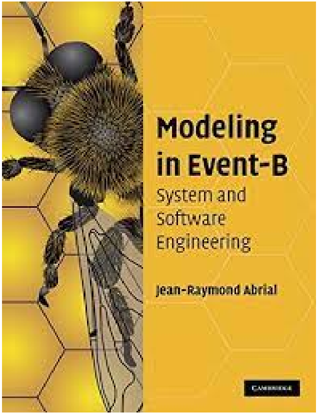
There will be equations and commutative diagrams on these slides but I will only superficially explain them. All of the details and proofs are in the paper.

Formal Methods for Critical Systems

What if I told you?

I modelled and verified critical systems using a language with **no formal semantics**. Further, there is **no native support to make the code modular** in this language and **translations** to other languages are **not systematic**.




Modeling
in Event-B
System and
Software
Engineering
Jean-Raymond Abrial

Think About It...

Formal Semantics

- Proof obligations give a list of properties to prove for a given model.
- Not a semantics for the language itself.

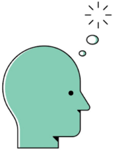
Modularisation

- Lots of plugins but no direct language support.

Interoperability

- Lots of plugins but no way of checking that the semantics is preserved.

Forget everything that you know about Event-B!



Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Event-B?

Event-B Formal Specification Language

```
CONTEXT ctx
   EXTENDS ctx0
   SETS S
   CONSTANTS c
   AXIOMS
      A(s,c)
```

```
MACHINE m  REFINES m0
SEES ctx
  VARIABLES x
  INVARIANTS I(x)
  VARIANT n(x)
  EVENTS
    INITIALISATION, e1,...,en
```

```
Event ei ≙  status
   any p
   when G(x,p)
   with W(x,p)
   then BA(x,p,x')
end
```

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

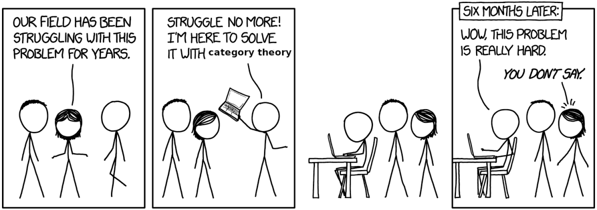
BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Institution?

Institutions: Some Maths



An **institution** $\mathcal{INS}$ for a given formalism

Vocabulary: a category **Sign** whose objects are called signatures and whose arrows are called signature morphisms.

An **institution** $\mathcal{INS}$ for a given formalism

Vocabulary: a category **Sign** whose objects are called signatures and whose arrows are called signature morphisms.

Syntax: a functor **Sen** : **Sign** $\rightarrow$ **Set** giving a set **Sen**(Σ) of Σ-*sentences* for each signature Σ and a function **Sen**(σ) : **Sen**$(\Sigma) \rightarrow$ **Sen**(Σ') for each signature morphism $\sigma : \Sigma \rightarrow \Sigma'$.

An **institution** $\mathcal{INS}$ for a given formalism

Vocabulary: a category **Sign** whose objects are called signatures and whose arrows are called signature morphisms.

Syntax: a functor **Sen** : **Sign** $\rightarrow$ **Set** giving a set **Sen**(Σ) of Σ-*sentences* for each signature Σ and a function **Sen**(σ) : **Sen**$(\Sigma) \rightarrow$ **Sen**(Σ') for each signature morphism $\sigma : \Sigma \rightarrow \Sigma'$.

Semantics: a functor **Mod** : **Sign**^{op} $\rightarrow$ **Cat** giving a category **Mod**(Σ) of Σ-*models* for each signature Σ and a functor **Mod**(σ) : **Mod**$(\Sigma') \rightarrow$ **Mod**(Σ) for each signature morphism $\sigma : \Sigma \rightarrow \Sigma'$.

An **institution** $\mathcal{INS}$ for a given formalism

Vocabulary: a category **Sign** whose objects are called signatures and whose arrows are called signature morphisms.

Syntax: a functor **Sen** : **Sign** $\rightarrow$ **Set** giving a set **Sen**(Σ) of Σ-*sentences* for each signature Σ and a function **Sen**(σ) : **Sen**$(\Sigma) \rightarrow$ **Sen**(Σ') for each signature morphism $\sigma : \Sigma \rightarrow \Sigma'$.

Semantics: a functor **Mod** : **Sign**$^{op} \rightarrow$ **Cat** giving a category **Mod**(Σ) of Σ-*models* for each signature Σ and a functor **Mod**(σ) : **Mod**$(\Sigma') \rightarrow$ **Mod**(Σ) for each signature morphism $\sigma : \Sigma \rightarrow \Sigma'$.

Satisfaction: for every signature Σ, a satisfaction relation $\models_{\mathcal{INS},\Sigma}$ between Σ-models and Σ-sentences.
An institution must uphold the **satisfaction condition**: for any signature morphism $\sigma : \Sigma \rightarrow \Sigma'$ and translations **Mod**(σ) of models and **Sen**(σ) of sentences we have for any $\phi \in$ **Sen**(Σ) and $M' \in |$ **Mod**(Σ') $|$.

$$M' \models_{\mathcal{INS},\Sigma'} \mathbf{Sen}(\sigma)(\phi) \quad \Leftrightarrow \quad \mathbf{Mod}(\sigma)(M') \models_{\mathcal{INS},\Sigma} \phi$$

$$\begin{array}{c} \Sigma_1 \\ \downarrow \sigma \\ \Sigma_2 \end{array}$$

$$\begin{array}{ccc} \mathbf{Sen}(\Sigma_1) & \xrightarrow{\rho^{Sen}_{\Sigma_1}} & \mathbf{Sen}'(\rho^{Sign}(\Sigma_1)) \\ {\scriptstyle \mathbf{Sen}(\sigma)}\downarrow & & \downarrow{\scriptstyle \mathbf{Sen}'(\rho^{Sign}(\sigma))} \\ \mathbf{Sen}(\Sigma_2) & \xrightarrow[\rho^{Sen}_{\Sigma_2}]{} & \mathbf{Sen}'(\rho^{Sign}(\Sigma_2)) \end{array} \qquad \begin{array}{ccc} \mathbf{Mod}'(\rho^{Sign}(\Sigma_2)) & \xrightarrow{\rho^{Mod}_{\Sigma_2}} & \mathbf{Mod}(\Sigma_2) \\ {\scriptstyle \mathbf{Mod}'(\rho^{Sign}(\sigma))}\downarrow & & \downarrow{\scriptstyle \mathbf{Mod}(\sigma)} \\ \mathbf{Mod}'(\rho^{Sign}(\Sigma_2)) & \xrightarrow[\rho^{Mod}_{\Sigma_1}]{} & \mathbf{Mod}(\Sigma_1) \end{array}$$

"truth is invariant under change of notation"

First-Order Predicate Logic with Equality ($\mathcal{FOPEQ}$)

Signatures: $\Sigma_{\mathcal{FOPEQ}} = \langle S, \Omega, \Pi \rangle$

- S is a set of sort names
- Ω is a set of operation names
- Π is a set of predicate names indexed by arity.

First-Order Predicate Logic with Equality ($\mathcal{FOPEQ}$)

Signatures: $\Sigma_{\mathcal{FOPEQ}} = \langle S, \Omega, \Pi \rangle$

- S is a set of sort names
- Ω is a set of operation names
- Π is a set of predicate names indexed by arity.

Sentences: closed first-order formulae using $\wedge, \vee, \neg\ , \Rightarrow, \Leftrightarrow, \exists, \forall$ and the logical constants `true` and `false`.

First-Order Predicate Logic with Equality ($\mathcal{FOPEQ}$)

Signatures: $\Sigma_{\mathcal{FOPEQ}} = \langle S, \Omega, \Pi \rangle$

- S is a set of sort names
- Ω is a set of operation names
- Π is a set of predicate names indexed by arity.

Sentences: closed first-order formulae using $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \exists, \forall$ and the logical constants `true` and `false`.

Models: consist of a carrier set $|A|_s$ for each sort name $s \in S$, a function $f_A : |A|_{s_1} \times \cdots \times |A|_{s_n} \rightarrow |A|_s$ for each operation name $f \in \Omega_{s_1 \ldots s_n, s}$ and a relation $p_A \subseteq |A|_{s_1} \times \cdots \times |A|_{s_n}$ for each predicate name $p \in \Pi_{s_1 \cdots s_n}$.

First-Order Predicate Logic with Equality ($\mathcal{FOPEQ}$)

Signatures: $\Sigma_{\mathcal{FOPEQ}} = \langle S, \Omega, \Pi \rangle$

- S is a set of sort names
- Ω is a set of operation names
- Π is a set of predicate names indexed by arity.

Sentences: closed first-order formulae using $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow, \exists, \forall$ and the logical constants `true` and `false`.

Models: consist of a carrier set $|A|_s$ for each sort name $s \in S$, a function $f_A : |A|_{s_1} \times \cdots \times |A|_{s_n} \rightarrow |A|_s$ for each operation name $f \in \Omega_{s_1 \ldots s_n, s}$ and a relation $p_A \subseteq |A|_{s_1} \times \cdots \times |A|_{s_n}$ for each predicate name $p \in \Pi_{s_1 \ldots s_n}$.

Satisfaction Relation: usual satisfaction of first-order sentences by first-order structures.

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Event-B Institution?

The Three-Layer Model

Event-B Superstructure	refines, sees	$\mathcal{EVT}$ specification-building operators
Event-B Infrastructure	variables, invariants, variants, events	$\mathcal{EVT}$-sentences
Mathematical Language	carrier sets, constants, axioms, extends	$\mathcal{FOPEQ}$-sentences and specification-building operators

The Three-Layer Model

Event-B Superstructure	refines, sees	$\mathcal{EVT}$ specification-building operators
	↑	
Event-B Infrastructure	variables, invariants, variants, events	$\mathcal{EVT}$-sentences
	↑	
Mathematical Language	carrier sets, constants, axioms, extends	$\mathcal{FOPEQ}$-sentences and specification-building operators

The $\mathcal{FOPEQ}$ Interface

$\mathcal{FOPEQ}$ **Operations**

- `F.and` : Σ-*formula** $\rightarrow$ Σ-*formula*
- `F.lt` : Σ-*term* $\times$ Σ-*term* $\rightarrow$ Σ-*formula*
- `F.leq` : Σ-*term* $\times$ Σ-*term* $\rightarrow$ Σ-*formula*
- `F.exists` : *VarName** $\times$ Σ-*formula* $\rightarrow$ Σ-*formula*
- `F.`ι : *VarName** $\rightarrow$ Σ-*formula* $\rightarrow$ Σ-*formula*

$\mathcal{FOPEQ}$ **Functions**

- $\mathbb{P}_\Sigma$: *LabelledPred* $\rightarrow$ Σ-*formula*
- $\mathbb{T}_\Sigma$: *Expression* $\rightarrow$ Σ-*term*
- $\mathbb{M}$: *SetName** $\times$ *ConstName** $\times$ *LabelledPred** $\rightarrow$ $|\mathbf{Sign}_{\mathcal{FOPEQ}}|$

The Three-Layer Model

Event-B Superstructure	refines, sees	$\mathcal{EVT}$ specification-building operators
Event-B Infrastructure	variables, invariants, variants, events	$\mathcal{EVT}$-sentences
Mathematical Language	carrier sets, constants, axioms, extends	$\mathcal{FOPEQ}$-sentences and specification-building operators

What is $\mathcal{EVT}$?

$\mathcal{EVT}$ - The Institution for Event-B (Vocabulary)

Signatures: $\Sigma_{\mathcal{EVT}} = \langle S, \Omega, \Pi, E, V \rangle$

- S, Ω, Π from $\mathcal{FOPEQ}$
- E is a function from event names to their status.
- V is a set of sort-indexed variable names.

Signature Extraction

```
CONTEXT cd
  CONSTANTS
    d
  AXIOMS
    axm1: d ∈ ℕ
    axm2: d > 0
END
```

```
MACHINE m0
  SEES cd
  VARIABLES
    n
  INVARIANTS
    inv1: n ∈ ℕ
    inv2: n ≤ d
  EVENTS
    Initialisation
      then
        act1: n := 0
    Event ML_out ≙ ordinary
      when
        grd1: n < d
      then
        act1: n := n + 1
    Event ML_in ≙ ordinary
      when
        grd1: n > 0
      then
        act1: n := n − 1
END
```

Signature

$$\Sigma_{\mathtt{m1}} = \langle\ S,\ \Omega,\ \Pi,\ E,\ V\ \rangle$$

where

$$\begin{aligned}
S &= \{\mathbb{N}\},\\
\Omega &= \{0 : \mathbb{N}, d : \mathbb{N}\},\\
\Pi &= \{> : \mathbb{N} \times \mathbb{N}\},\\
E &= \{(\mathtt{Init} \mapsto \mathtt{ordinary}), (\mathtt{ML_in} \mapsto \mathtt{ordinary}), (\mathtt{ML_out} \mapsto \mathtt{ordinary})\},\\
V &= \{\mathtt{n} : \mathbb{N}\}
\end{aligned}$$

$\mathcal{EVT}$ - The Institution for Event-B (Syntax)

Sentences:

```
MACHINE m REFINES a SEES ctx
  VARIABLES x̄
  INVARIANTS I(x̄)
  VARIANT n(x̄)
  EVENTS
  Initialisation ordinary
    then  act-name: BA(x̄′)
    ⋮
  Event eᵢ ≙  convergent
    any p̄
    when  guard-name: G(x̄, p̄)
    with  witness-name: W(x̄, p̄)
    then  act-name: BA(x̄, p̄, x̄′)
    ⋮
END
```

$$\{\langle e, \ I(\overline{x}) \ \wedge \ I(\overline{x}\prime)\rangle \ \mid \ e \ \in \ dom(\Sigma.E)\}$$

$$\langle \texttt{Init}, \ BA(\overline{x}')\rangle$$

$$\langle \mathrm{e}_i, \ n(\overline{x}\prime) \ < \ n(\overline{x})\rangle$$

$$\langle \mathrm{e}, \ \exists\, \overline{p} \ \cdot \ G(\overline{x}, \overline{p}) \ \wedge \ W(\overline{x}, \overline{p}) \ \wedge \ BA(\overline{x}, \overline{p}, \overline{x}')\rangle$$

$\mathcal{EVT}$ - The Institution for Event-B (Semantics)

Models: $\langle A, L, R \rangle$

- A is a $\Sigma_{\mathcal{FOPEQ}}$-model.
- $L \subseteq State_A$ provides the states after the `Init` event.
- $R.e \subseteq State_A \times State_A$.

```
Event e ≙
  when grd1: x < 2
  then act1: x := x + 1
       act2: y := false
```

$$R_e = \left\{ \begin{array}{llll} \{x \mapsto 0, & y \mapsto \mathit{false}, & x' \mapsto 1, & y' \mapsto \mathit{false}\}, \\ \{x \mapsto 0, & y \mapsto \mathit{true}, & x' \mapsto 1, & y' \mapsto \mathit{false}\}, \\ \{x \mapsto 1, & y \mapsto \mathit{false}, & x' \mapsto 2, & y' \mapsto \mathit{false}\}, \\ \{x \mapsto 1, & y \mapsto \mathit{true}, & x' \mapsto 2, & y' \mapsto \mathit{false}\} \end{array} \right\}$$

$\mathcal{EVT}$ - The Institution for Event-B (Satisfaction)

Satisfaction:

1. For any $\mathcal{EVT}$-model $\langle A, L, R \rangle$ and $\mathcal{EVT}$-sentence $\langle e, \phi(\overline{x}, \overline{x\prime}) \rangle$, where $e \neq \texttt{Init}$:

$$\langle A, L, R \rangle \models_{\Sigma} \langle e, \phi(\overline{x}, \overline{x}') \rangle \quad \Leftrightarrow \quad \forall \langle s, s' \rangle \in R.e \cdot A^{(s,s\prime)} \models_{\Sigma^{(V,V')}_{\mathcal{FOPEQ}}} \phi(\overline{x}, \overline{x}')$$

2. For $\mathcal{EVT}$-sentences of the form $\langle \texttt{Init}, \phi(\overline{x}') \rangle$:

$$\langle A, L, R \rangle \models_{\Sigma} \langle \texttt{Init}, \phi(\overline{x}') \rangle \quad \Leftrightarrow \quad \forall\, s' \in L \cdot A^{(s\prime)} \models_{\Sigma^{(V')}_{\mathcal{FOPEQ}}} \phi(\overline{x}')$$

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

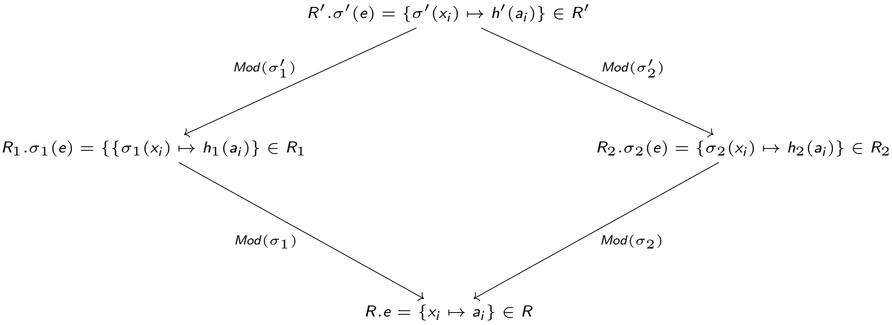
Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Building Specifications?

Specification-Building Operators

Operation	Format	Description
Translation	SP_1 `with` σ	Renames the signature components of SP_1 using the signature morphism $\sigma : \Sigma_{SP_1} \to \Sigma'$. $Sig[SP_1 \texttt{ with } \sigma] = \Sigma'$ $Mod[SP_1 \texttt{ with } \sigma] = \{M' \in \lvert\mathbf{Mod}(\Sigma')\rvert \mid M'\vert_\sigma \in Mod[SP_1]\}$.
Sum	SP_1 `and` SP_2	Combines the specifications SP_1 and SP_2. $SP_1 \texttt{ and } SP_2 = (SP_1 \texttt{ with } \iota) \cup (SP_2 \texttt{ with } \iota')$ where $Sig[SP_1] = \Sigma$, $Sig[SP_2] = \Sigma'$, $\iota : \Sigma \hookrightarrow \Sigma \cup \Sigma'$, $\iota' : \Sigma' \hookrightarrow \Sigma \cup \Sigma'$
Enrichment	SP_1 `then` ...	Extends the specification SP_1 by adding new sentences after the `then` specification-building operator. This operator can be used to represent **superposition refinement** of Event-B specifications.
Hiding	SP_1 `hide via` σ	Interprets a specification, SP_1, as one restricted to the signature components of another specified by the signature morphism $\sigma : \Sigma \to \Sigma_{SP_1}$. $Sig[SP_1 \texttt{ hide via } \sigma] = \Sigma$ $Mod[SP_1 \texttt{ hide via } \sigma] = \{ M\vert_\sigma \mid M \in Mod[SP_1]\}$.

Institutions Must Preserve Amalgamation for Specification Building



...proofs are in the paper

Building Specifications in the Event-B Institution

$\mathbb{B}: \quad \mathit{Machine} \to \mathit{Env} \to |\mathbf{Spec}_{\mathcal{EVT}}|$ *#Build an $\mathcal{EVT}$ structured specification for one machine*

$$\mathbb{B}\left[\!\!\left[\begin{array}{l}\texttt{machine}\ m\\ \texttt{refines}\ a\\ \texttt{sees}\ ctx_1,\ldots,ctx_n\\ mbody\\ \texttt{end}\end{array}\right]\!\!\right]\xi = \left\langle \Sigma, \left[\begin{array}{l}\texttt{spec}\ [\![m]\!]\ \texttt{over}\ \mathcal{EVT} =\\ \quad \textit{\# Include contexts using the comorphism } \rho\textit{:}\\ \quad ([\![ctx_1]\!]\ \texttt{and}\ \ldots\ \texttt{and}\ [\![ctx_n]\!])\ \texttt{with}\ \rho\\ \quad \textit{\#Sentences from the refined machine (if any):}\\ \quad (\texttt{and}\ \mathbb{A}_\Sigma[\![mbody]\!][\![a]\!]\xi)\\ \texttt{then}\\ \quad \mathbb{S}_\Sigma[\![mbody]\!]\end{array}\right]\right\rangle$$

where $\Sigma = \xi[\![m]\!]$.

$\mathbb{A}_\Sigma: \quad \mathit{MachineBody} \to \mathit{EventName} \to \mathit{Env} \to |\mathbf{Spec}_{\mathcal{EVT}}|$

#Extract any relevant specification from the refined (abstract) machine

$$\mathbb{A}_\Sigma\left[\!\!\left[\begin{array}{l}\texttt{variables}\ v_1,\ldots,v_n\\ \texttt{invariants}\ i_1,\ldots,i_n\\ \texttt{theorems}\ t_1,\ldots,t_n\\ \texttt{variant}\ n\\ \texttt{events}\ e_{init},\ e_1,\ldots,e_n\end{array}\right]\!\!\right][\![a]\!]\xi = \begin{array}{l}\mathbb{I}_\Sigma[\![i_1]\!]\ \texttt{and}\ \ldots\ \texttt{and}\ \mathbb{I}_\Sigma[\![i_n]\!]\\ \texttt{and}\ \mathbb{R}_\Sigma[\![e_1]\!][\![a]\!]\xi\ \texttt{and}\ \ldots\ \texttt{and}\ \mathbb{R}_\Sigma[\![e_n]\!][\![a]\!]\xi\end{array}$$

#Conjoin sentences from each event definition

Building Specifications in the Event-B Institution

For an Event-B specification SP, we form an environment $\xi = \mathbb{D}[SP]\xi_0$ where ξ_0 is the empty environment.

- $Env = (MachineName \cup ContextName) \to |\mathbf{Sign}|$ *# An environment maps names to signatures*

- $\mathbb{D} : Specification \to Env \to Env$ *# Process a list of machine/context definitions*
  $\mathbb{D}\ \llbracket \langle\,\rangle \rrbracket\ \xi = \xi$
  $\mathbb{D}\ \llbracket hd :: tl \rrbracket\ \xi = \mathbb{D}\ (\llbracket tl \rrbracket)\ (\mathbb{D}\ \llbracket hd \rrbracket\ \xi)$

- $\mathbb{D} : Machine \to Env \to Env$ *# Extract and store the signature for one machine*
  $$\mathbb{D} \left\llbracket \begin{array}{l} \texttt{machine } m \\ \texttt{refines } a \\ \texttt{sees } ctx_1, \ldots, ctx_n \\ mbody \\ \texttt{end} \end{array} \right\rrbracket \xi = \xi \cup \{\llbracket m \rrbracket \mapsto (\langle S, \Omega, \Pi, E, V \rangle \cup r(\xi\llbracket a \rrbracket))\}$$
  where
  $\langle S, \Omega, \Pi \rangle = \{(\xi\ \llbracket ctx_1 \rrbracket) \cup \ldots \cup (\xi\ \llbracket ctx_n \rrbracket)\}$ *# Include signatures from 'seen' contexts*
  $\langle E, V, RA \rangle = \mathbb{D}\llbracket mbody \rrbracket$ *# Collect names from machine body*
  and
  $r : |\mathbf{Sign}_{\mathcal{EVT}}| \to |\mathbf{Sign}_{\mathcal{EVT}}|$ *# Signature for the abstract machine (less the refined events)*
  $r(\xi\llbracket a \rrbracket) = \texttt{let } \Sigma_a = \xi\llbracket a \rrbracket \texttt{ in } \langle \Sigma_a.S, \Sigma_a.\Omega, \Sigma_a.\Pi, RA \lessdot \Sigma_a.E, \Sigma_a.V \rangle$

- $\mathbb{D} : MachineBody \to (\mathcal{P}(EventName \times Stat) \times \mathcal{P}(VarName) \times \mathcal{P}(EventName))$ *# Extract signature elements from machine-body*
  $$\mathbb{D} \left\llbracket \begin{array}{l} \texttt{variables } v_1, \ldots, v_n \\ \texttt{invariants } i_1, \ldots, i_n \\ \texttt{theorems } t_1, \ldots, t_n \\ \texttt{variant } n \\ \texttt{events } e_{init}\ e_1, \ldots, e_n \end{array} \right\rrbracket = \langle E, V, RA \rangle$$
  where
  $E = \{def\llbracket e_{init} \rrbracket, def\llbracket e_1 \rrbracket, \ldots, def\llbracket e_n \rrbracket\}$ *# Names of events defined here*
  $V = \langle \llbracket v_1 \rrbracket, \ldots, \llbracket v_n \rrbracket \rangle$ *# Names of variables declared here*
  $RA = ref\llbracket e_{init} \rrbracket \cup ref\llbracket e_1 \rrbracket \cup \ldots \cup ref\llbracket e_n \rrbracket$ *# Names of (abstract) events refined here*

- $def : Event \to (EventName \times Stat)$ *# Extract event name & status from an event definition*
  $def\ \llbracket \texttt{event } e, \texttt{status } s, \texttt{refines } e_1, \ldots, e_n, \cdots \texttt{end} \rrbracket = (e \mapsto s)$
  $def\ \llbracket e_{init} \rrbracket = \langle \texttt{Init}, \texttt{ordinary} \rangle$

- $ref : Event \to \mathcal{P}(EventName)$ *# Extract names of refined events from an event definition*
  $ref\ \llbracket \texttt{event } e, \texttt{status } s, \texttt{refines } e_1, \ldots, e_n, \cdots \texttt{end} \rrbracket = \{\llbracket e_1 \rrbracket, \ldots, \llbracket e_n \rrbracket\}$
  $ref\ \llbracket e_{init} \rrbracket = \{\llbracket e_{init} \rrbracket\}$

- $\mathbb{D} : Context \to Env \to Env$ *# Extract and store the signature for one context*
  $$\mathbb{D} \left\llbracket \begin{array}{l} \texttt{context } ctx \\ \texttt{extends } ctx_1, \ldots, ctx_n \\ cbody \\ \texttt{end} \end{array} \right\rrbracket \xi = \xi \cup \{\llbracket ctx \rrbracket \mapsto (\mathbb{D}\llbracket cbody \rrbracket \cup \xi\llbracket ctx_1 \rrbracket \cup \ldots \cup \xi\llbracket ctx_n \rrbracket)\}$$

- $\mathbb{D} : ContextBody \to |\mathbf{Sign}_{\mathcal{FOPEQ}}|$ *# Extract the $\mathcal{FOPEQ}$ signature from a context body*
  $$\mathbb{D} \left\llbracket \begin{array}{l} \texttt{sets } s_1, \ldots, s_n \\ \texttt{constants } c_1, \ldots, c_n \\ \texttt{axioms } a_1, \ldots, a_n \\ \texttt{theorems } t_1, \ldots, t_n \end{array} \right\rrbracket = \langle S, \Omega, \Pi \rangle$$ *# Sorts, operations, predicates*
  where
  $\langle S, \Omega, \Pi \rangle = \mathbb{M}((\llbracket s_1 \rrbracket, \ldots, \llbracket s_n \rrbracket), (\llbracket c_1 \rrbracket, \ldots, \llbracket c_n \rrbracket), (\llbracket a_1 \rrbracket, \ldots, \llbracket a_m \rrbracket))$

- $\mathbb{S}_\Sigma : MachineBody \to Sen_{\mathcal{EVT}}(\Sigma)$ *# Build sentences from a machine body*
  $$\mathbb{S}_\Sigma \left\llbracket \begin{array}{l} \texttt{variables } v_1, \ldots, v_n \\ \texttt{invariants } i_1, \ldots, i_n \\ \texttt{theorems } t_1, \ldots, t_n \\ \texttt{variant } n \\ \texttt{events } e_{init}, e_1, \ldots, e_n \end{array} \right\rrbracket = \left(\begin{array}{ll} & \mathbb{I}_\Sigma\llbracket i_1 \rrbracket \cup \ldots \cup \mathbb{I}_\Sigma\llbracket i_n \rrbracket \\ \cup & \mathbb{V}_\Sigma\llbracket n \rrbracket \\ \cup & \mathbb{E}_\Sigma\llbracket e_{init} \rrbracket \\ \cup & \mathbb{E}_\Sigma\llbracket e_1 \rrbracket \cup \ldots \cup \mathbb{E}_\Sigma\llbracket e_n \rrbracket \end{array} \right)$$

- $\mathbb{I}_\Sigma : LabelledPred \to Sen_{\mathcal{EVT}}(\Sigma)$ *# Invariant sentences*
  $\mathbb{I}_\Sigma\ \llbracket i \rrbracket = \{\langle \llbracket e \rrbracket, \texttt{F.and}(\mathbb{P}_\Sigma\llbracket i \rrbracket, \texttt{F.}\iota(\Sigma.V)(\mathbb{P}_\Sigma\llbracket i \rrbracket)) \rangle \mid e \in dom(\Sigma.E)\}$

- $\mathbb{V}_\Sigma : Expression \to Sen_{\mathcal{EVT}}(\Sigma)$ *# Variant can't increase for non-ord. events*
  $\mathbb{V}_\Sigma\ \llbracket n \rrbracket = \{\langle \llbracket e \rrbracket, \texttt{F.lt}(\texttt{F.}\iota(\Sigma.V)(\mathbb{T}_\Sigma\llbracket n \rrbracket), \mathbb{T}_\Sigma\llbracket n \rrbracket) \rangle \mid (e \mapsto \texttt{convergent}) \in \Sigma.E\}$
  $\cup\ \{\langle \llbracket e \rrbracket, \texttt{F.leq}(\texttt{F.}\iota(\Sigma.V)(\mathbb{T}_\Sigma\llbracket n \rrbracket), \mathbb{T}_\Sigma\llbracket n \rrbracket) \rangle \mid (e \mapsto \texttt{anticipated}) \in \Sigma.E\}$

- $\mathbb{E}_\Sigma : InitEvent \to Sen_{\mathcal{EVT}}(\Sigma)$ *# Initial event: get sentences from actions*
  $$\mathbb{E}_\Sigma \left\llbracket \begin{array}{l} \texttt{event Initialisation} \\ \texttt{status ordinary} \\ \texttt{then } act_1, \ldots, act_n \\ \texttt{end} \end{array} \right\rrbracket = \{\langle \texttt{Init}, BA \rangle\}$$
  where
  $BA = \texttt{F.and}(\mathbb{P}_\Sigma\llbracket act_1 \rrbracket, \ldots, \mathbb{P}_\Sigma\llbracket act_n \rrbracket)$

- $\mathbb{E}_\Sigma : Event \to Sen_{\mathcal{EVT}}(\Sigma)$ *# Non-initial event: get sentences from event body*
  $$\mathbb{E}_\Sigma \left\llbracket \begin{array}{l} \texttt{event } e \\ \texttt{status } s \\ \texttt{refines } e_1, \ldots, e_n \\ ebody \\ \texttt{end} \end{array} \right\rrbracket = \{\langle \llbracket e \rrbracket, \mathbb{F}_\Sigma\llbracket ebody \rrbracket \rangle\}$$

- $\mathbb{F}_\Sigma : EventBody \to \Sigma\text{-}formula$ *# Build a $\mathcal{FOPEQ}$ formula for an event definition*
  $$\mathbb{F}_\Sigma \left\llbracket \begin{array}{l} \texttt{any } p_1, \ldots, p_n \\ \texttt{where } grd_1, \ldots, grd_n \\ \texttt{with } w_1, \ldots, w_n \\ \texttt{then } act_1, \ldots, act_n \end{array} \right\rrbracket = \texttt{F.exists}(p, \texttt{F.and}(G, W, BA))$$
  # Formula is existentially quantified over event parameters p
  where
  $p = \langle \llbracket p_1 \rrbracket, \ldots, \llbracket p_n \rrbracket \rangle$ *# List of parameters*
  $G = \texttt{F.and}(\mathbb{P}_\Sigma\llbracket grd_1 \rrbracket, \ldots, \mathbb{P}_\Sigma\llbracket grd_n \rrbracket)$ *# Guards*
  $W = \texttt{F.and}(\mathbb{P}_\Sigma\llbracket w_1 \rrbracket, \ldots, \mathbb{P}_\Sigma\llbracket w_n \rrbracket)$ *# Witnesses*
  $BA = \texttt{F.and}(\mathbb{P}_\Sigma\llbracket act_1 \rrbracket, \ldots, \mathbb{P}_\Sigma\llbracket act_n \rrbracket)$ *# Actions*

- $\mathbb{S}_\Sigma : ContextBody \to Sen_{\mathcal{FOPEQ}}(\Sigma)$ *# Context: get $\mathcal{FOPEQ}$ sentences from axioms*
  $$\mathbb{S}_\Sigma \left\llbracket \begin{array}{l} \texttt{sets } s_1, \ldots, s_n \\ \texttt{constants } c_1, \ldots, c_n \\ \texttt{axioms } a_1, \ldots, a_n \\ \texttt{theorems } t_1, \ldots, t_n \end{array} \right\rrbracket = \{\mathbb{P}_\Sigma\llbracket a_1 \rrbracket, \ldots, \mathbb{P}_\Sigma\llbracket a_n \rrbracket\}$$

The semantics of an Event-B specification SP are given by $\mathbb{B}[SP]\xi$, where $\xi = \mathbb{D}[SP]\xi_0$ is the environment defined by Figure 9.

- $\mathbb{B} : Specification \to Env \to |\mathbf{Spec}|^*$ *# Process specifications in an environment, build a list of structured specifications*
  $\mathbb{B}\ \llbracket \langle\,\rangle \rrbracket\ \xi = \langle\,\rangle$
  $\mathbb{B}\ \llbracket hd :: tl \rrbracket\ \xi = (\mathbb{B}\ \llbracket hd \rrbracket\ \xi) :: (\mathbb{B}\ \llbracket tl \rrbracket\ \xi)$

- $\mathbb{B} : Machine \to Env \to |\mathbf{Spec}_{\mathcal{EVT}}|$ *# Build an $\mathcal{EVT}$ structured specification for one machine*
  $$\mathbb{B} \left\llbracket \begin{array}{l} \texttt{machine } m \\ \texttt{refines } a \\ \texttt{sees } ctx_1, \ldots, ctx_n \\ mbody \\ \texttt{end} \end{array} \right\rrbracket \xi = \left\langle \Sigma, \left[\begin{array}{l} \texttt{spec } \llbracket m \rrbracket \texttt{ over } \mathcal{EVT} = \\ \quad \text{\# Include contexts using the comorphism } \rho\text{:} \\ \quad (\llbracket ctx_1 \rrbracket \texttt{ and } \ldots \texttt{ and } \llbracket ctx_n \rrbracket) \texttt{ with } \rho \\ \quad \text{\# Sentences from the refined machine (if any):} \\ \quad (\texttt{and } \mathbb{A}_\Sigma\llbracket mbody \rrbracket\llbracket a \rrbracket\xi) \\ \texttt{then} \\ \quad \mathbb{S}_\Sigma\llbracket mbody \rrbracket \end{array} \right] \right\rangle$$
  where $\Sigma = \xi\llbracket m \rrbracket$.

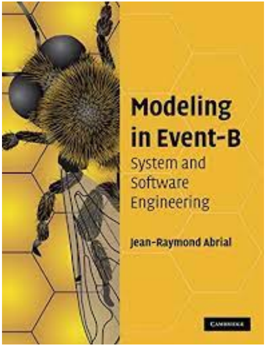
- $\mathbb{A}_\Sigma : MachineBody \to EventName \to Env \to |\mathbf{Spec}_{\mathcal{EVT}}|$ *# Extract any relevant specification from the refined (abstract) machine*
  $$\mathbb{A}_\Sigma \left\llbracket \begin{array}{l} \texttt{variables } v_1, \ldots, v_n \\ \texttt{invariants } i_1, \ldots, i_n \\ \texttt{theorems } t_1, \ldots, t_n \\ \texttt{variant } n \\ \texttt{events } e_{init}, e_1, \ldots, e_n \end{array} \right\rrbracket \llbracket a \rrbracket\ \xi = \begin{array}{l} \mathbb{I}_\Sigma\llbracket i_1 \rrbracket \texttt{ and } \ldots \texttt{ and } \mathbb{I}_\Sigma\llbracket i_n \rrbracket \\ \texttt{and } \mathbb{R}_\Sigma\llbracket e_1 \rrbracket\llbracket a \rrbracket\xi \texttt{ and } \ldots \texttt{ and } \mathbb{R}_\Sigma\llbracket e_n \rrbracket\llbracket a \rrbracket\xi \end{array}$$
  # Conjoin sentences from each event definition

- $\mathbb{R}_\Sigma : Event \to EventName \to Env \to |\mathbf{Spec}_{\mathcal{EVT}}|$ *# Extract specification from one refined event*
  $$\mathbb{R}_\Sigma \left\llbracket \begin{array}{l} \texttt{event } e_c \\ \texttt{status } s \\ \texttt{refines } e_1, \ldots, e_n \\ ebody \\ \texttt{end} \end{array} \right\rrbracket \llbracket a \rrbracket\ \xi = \begin{array}{l} \texttt{let} \\ \quad \Sigma_a = \xi\llbracket a \rrbracket, \quad \text{\# Signature of abstract machine} \\ \quad \text{\# Use } \Sigma_h, \sigma_h \text{ to select only refined events:} \\ \quad \Sigma_h = \langle \Sigma_a.S, \Sigma_a.\Omega, \Sigma_a.\Pi, \{\llbracket e_1 \rrbracket, \ldots, \llbracket e_n \rrbracket\} \lhd \Sigma_a.E, \Sigma_a.V \rangle, \\ \quad \sigma_h : \Sigma_h \hookrightarrow \Sigma_a, \\ \quad \text{\# Use } \sigma_m \text{ to reassign refined event sentences to } e_c\text{:} \\ \quad \sigma_m : \Sigma_h \to \Sigma \\ \quad \sigma_m = \left\langle \begin{array}{l} \Sigma_h.S \hookrightarrow \Sigma.S,\ \Sigma_h.\Omega \hookrightarrow \Sigma.\Omega, \Sigma_h.\Pi \hookrightarrow \Sigma.\Pi, \\ \Sigma_h.E \mapsto \{\llbracket e_c \rrbracket\} \lhd \Sigma.E, \\ \Sigma_h.V \hookrightarrow \Sigma.V \end{array} \right\rangle \\ \texttt{in} \\ \quad (\llbracket a \rrbracket \texttt{ hide via } \sigma_h) \texttt{ with } \sigma_m. \end{array}$$

- $\mathbb{B} : Context \to Env \to |\mathbf{Spec}_{\mathcal{FOPEQ}}|$ *# Build a $\mathcal{FOPEQ}$ structured specification for one context*
  $$\mathbb{B} \left\llbracket \begin{array}{l} \texttt{context } ctx \\ \texttt{extends } ctx_1, \ldots, ctx_n \\ cbody \\ \texttt{end} \end{array} \right\rrbracket \xi = \left\langle \Sigma, \left[\begin{array}{l} \texttt{spec } \llbracket ctx \rrbracket \texttt{ over } \mathcal{FOPEQ} = \\ \quad \llbracket ctx_1 \rrbracket \texttt{ and } \ldots \texttt{ and } \llbracket ctx_n \rrbracket \\ \texttt{then} \\ \quad \mathbb{S}_\Sigma\llbracket cbody \rrbracket \end{array} \right] \right\rangle$$
  where $\Sigma = \xi\llbracket ctx \rrbracket$.

…it's all in the paper.

An Example: Cars On A Bridge



An Example: Cars On A Bridge

```
CONTEXT cd
  CONSTANTS  d
  AXIOMS
     axm1: d ∈ ℕ
     axm2: d > 0
END

MACHINE m0
  SEES cd
  VARIABLES n
  INVARIANTS
     inv1: n ∈ ℕ
     inv2: n  ≤  d
  EVENTS
    Initialisation
      then  act1: n  :=  0
     Event ML_out ≙ ordinary
      when  grd1: n < d
      then  act1: n  :=  n + 1
    Event ML_in ≙ ordinary
      when  grd1: n > 0
      then  act1: n  :=  n − 1
END
```

```
spec  CD =
  sort ℕ
  ops  d:ℕ
  .  d >  0
end

spec  M0 =
  CD
  then
    ops  n:ℕ
    .  n ≤ d
     EVENTS
      Initialisation
        thenAct  n := 0
      Event ML_out ≙ ordinary
        when n < d
        thenAct  n := n+1
      Event ML_in ≙ ordinary
        when n > 0
        thenAct  n := n-1
end
```

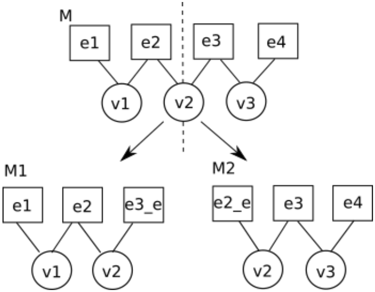
An Example: Cars On A Bridge

```
spec  M1 =
  M0 and CD
    then
    ops   a:ℕ
          b:ℕ
          c:ℕ
    .   n = a + b + c
        a = 0  ∨  c = 0
    variant 2 ∗ a + b
    EVENTS
      Initialisation
        thenAct   a := 0
                  b := 0
                  c := 0
      Event ML_out ≙ ordinary
        when a + b < d
              c = 0
        thenAct a := a+1
      Event IL_in ≙ convergent
        when   a > 0
        thenAct a := a-1
                b := b+1
      Event IL_out ≙ convergent
        when 0 < b
               a = 0
        thenAct b := b-1
                c := c+1
      Event ML_in ≙ ordinary
        when   c > 0
        thenAct  c := c-1
end
```

...more detail in the paper.

So What?

Modularisation via Specification Building: Shared Variable




```
spec  M1 =
  (M hide via  σ1)
    with {e3 ↦ e3_e}
end
  where σ1 = {v1 ↦ v1, v2 ↦ v2,
                e1 ↦ e1, e2 ↦ e2,
                e3 ↦ e3}

spec  M2 =
  (M hide via  σ2)
    with {e2 ↦ e2_e}
end
  where σ2 = {v2 ↦ v2, v3 ↦ v3,
                e2 ↦ e2, e3 ↦ e3,
                e4 ↦ e4}
```

...shared event and generic instantiation are covered in the paper.

What About Refinement?

What About Refinement?

... we can do that too!

Refinement

1. Signatures are the same:

$$SP_A \sqsubseteq SP_C \quad \Leftrightarrow \quad Mod(SP_C) \subseteq Mod(SP_A)$$

2. Signatures are different:

$$SP_A \sqsubseteq SP_C \quad \Leftrightarrow \quad Mod(\sigma)(SP_C) \subseteq Mod(SP_A)$$

```
refinement REF0 : M0 to M1 =
   ML_in ↦ ML_in, ML_out ↦ ML_out
end

refinement REF1A : M1 to M2 =
   ML_in  ↦ ML_in, ML_out ↦ ML_out1,  IL_in  ↦ IL_in, IL_out ↦ IL_out1
end

refinement REF1B : M1 to M2 =
   ML_in ↦ ML_in, ML_out ↦ ML_out2, IL_in ↦ IL_in, IL_out ↦ IL_out2
end
```

..other interesting refinement examples are in the paper.

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Our Contributions:

1. A formal (translational) **semantics** for Event-B using the EB2EVT tool.

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Our Contributions:

1. A formal (translational) **semantics** for Event-B using the EB2EVT tool.
2. A standard approach to **modularisation** using specification-building operators.

Building Specifications in the Event-B Institution

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Our Contributions:

1. A formal (translational) **semantics** for Event-B using the EB2EVT tool.
2. A standard approach to **modularisation** using specification-building operators.
3. An explication of Event-B **refinement** in the context of the EVT institution.

Conclusions and Future Work

- Provide access to stronger, more general modularisation for Event-B without the need to modify the formalism itself.

Conclusions and Future Work

- Provide access to stronger, more general modularisation for Event-B without the need to modify the formalism itself.
- Demonstrated how such modulrisation capabilities can be added to a formal specification language using the theory of institutions.

Conclusions and Future Work

- Provide access to stronger, more general modularisation for Event-B without the need to modify the formalism itself.
- Demonstrated how such modulrisation capabilities can be added to a formal specification language using the theory of institutions.
- Future work: examine how this approach can be applied to other similar formal languages.

Conclusions and Future Work

- Provide access to stronger, more general modularisation for Event-B without the need to modify the formalism itself.
- Demonstrated how such modulrisation capabilities can be added to a formal specification language using the theory of institutions.
- Future work: examine how this approach can be applied to other similar formal languages.
- Future Work: incorporate our semantics specification-building operators into Rodin using EB4EB and Theory Plugin.

Conclusions and Future Work

- Provide access to stronger, more general modularisation for Event-B without the need to modify the formalism itself.
- Demonstrated how such modulrisation capabilities can be added to a formal specification language using the theory of institutions.
- Future work: examine how this approach can be applied to other similar formal languages.
- Future Work: incorporate our semantics specification-building operators into Rodin using EB4EB and Theory Plugin.
- Future Work: define institution morphisms to enable interoperability with other formalisms.

Logical Methods in Computer Science
Volume 18, Issue 4, 2022, pp. 4:1–4:55
https://lmcs.episciences.org/

Submitted Mar. 22, 2021
Published Nov. 09, 2022

BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL, ROSEMARY MONAHAN, AND JAMES F. POWER

Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Questions?

marie.farrell@manchester.ac.uk

FM AS

Formal Methods for Autonomous Systems

5TH WORKSHOP ON

FORMAL METHODS FOR AUTONOMOUS SYSTEMS

Important Dates:

- Submission: *17th of August 2023* (AOE)
- Notification: *15th of September 2023*
- Final Version due: *20th of October 2023*
- Workshop: *15th and 16th of November 2023*, hybrid format, at iFM 2023

Submission Information:

- Vision Papers and Research Previews: *6 pgs EPTCS*
- Regular Papers and Experience Reports: *15 pgs EPTCS*

Topics of Interest include:

- Applicable, tool-supported Formal Methods that are suited to Autonomous Systems,
- Runtime Verification or other formal approaches to deal with the gap between models/simulations and the real world
- Verification against safety assurance arguments or standards documents,
- Case Studies that identify challenges when applying formal methods to autonomous systems

https://fmasworkshop.github.io/FMAS2023/