

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:

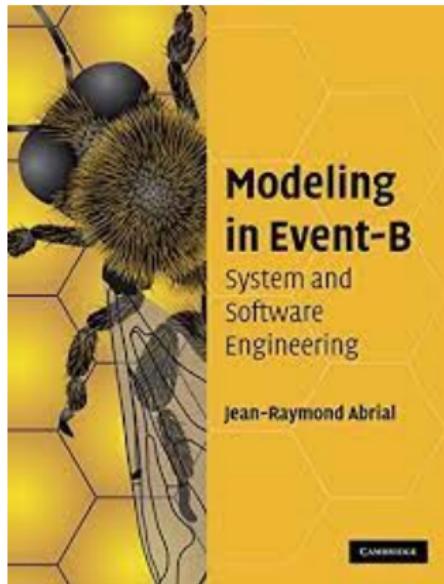
There will be equations and commutative diagrams on these slides but I will only superficially explain them. All of the details and proofs are in the paper.

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**.





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.

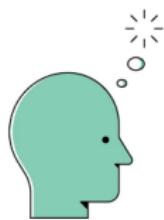
Modularisation

- 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

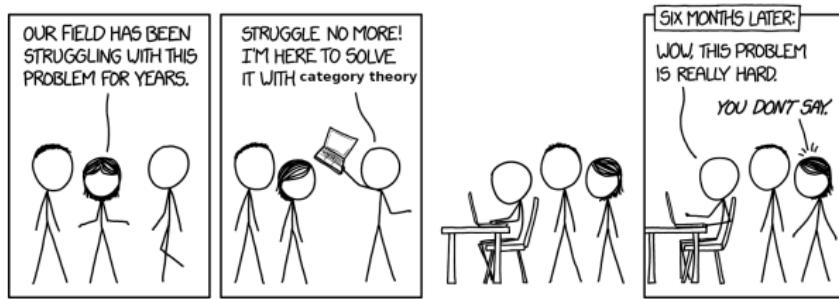
BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

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** → **Set** giving a set **Sen**(Σ) of Σ -sentences for each signature Σ and a function **Sen**(σ) : **Sen**(Σ) → **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**(Σ) \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**(Σ') \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**(Σ) \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**(Σ') \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 $\phi \in \mathbf{Sen}(\Sigma)$ and $M' \in \mathbf{Mod}(\Sigma')$.

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

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

$$\begin{array}{ccc} \mathbf{Sen}(\Sigma_1) & \xrightarrow{\rho_{\Sigma_1}^{Sen}} & \mathbf{Sen}'(\rho^{Sign}(\Sigma_1)) \\ \mathbf{Sen}(\sigma) \downarrow & & \downarrow \mathbf{Sen}'(\rho^{Sign}(\sigma)) \\ \mathbf{Sen}(\Sigma_2) & \xrightarrow{\rho_{\Sigma_2}^{Sen}} & \mathbf{Sen}'(\rho^{Sign}(\Sigma_2)) \end{array} \quad \begin{array}{ccc} \mathbf{Mod}'(\rho^{Sign}(\Sigma_2)) & \xrightarrow{\rho_{\Sigma_2}^{Mod}} & \mathbf{Mod}(\Sigma_2) \\ \mathbf{Mod}'(\rho^{Sign}(\sigma)) \downarrow & & \downarrow \mathbf{Mod}(\sigma) \\ \mathbf{Mod}'(\rho^{Sign}(\Sigma_2)) & \xrightarrow{\rho_{\Sigma_1}^{Mod}} & \mathbf{Mod}(\Sigma_1) \end{array}$$

“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 $f \in \Omega_{s_1 \dots s_n, s}$ and a relation $p_A \subseteq |A|_{s_1} \times \cdots \times |A|_{s_n}$ for each predicate name $p \in \Pi_{s_1 \dots s_n}$.

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 $f \in \Omega_{s_1 \dots s_n, s}$ and a relation $p_A \subseteq |A|_{s_1} \times \cdots \times |A|_{s_n}$ for each predicate name $p \in \Pi_{s_1 \dots s_n}$.

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

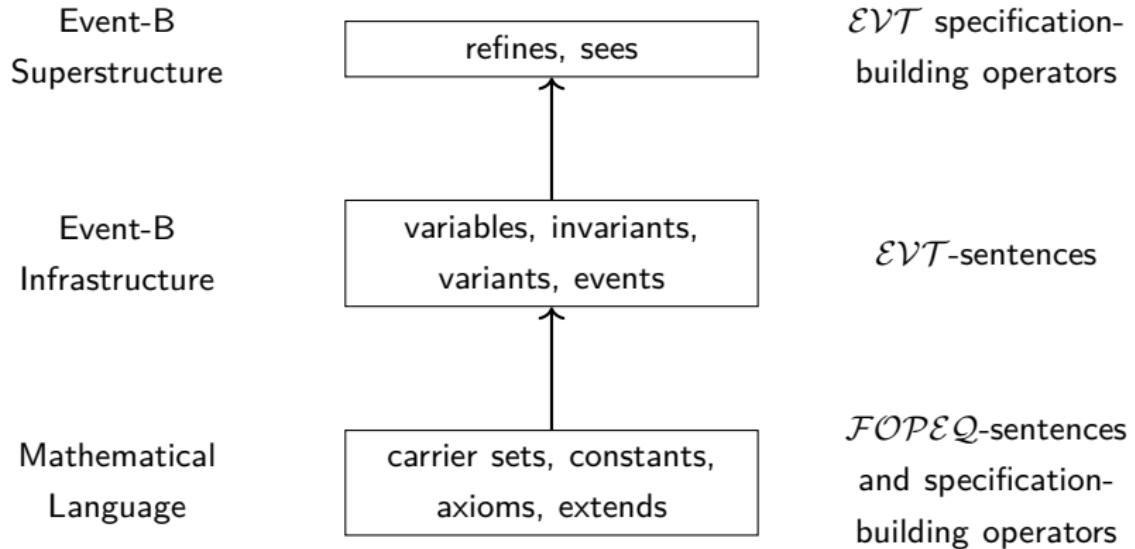
BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL , ROSEMARY MONAHAN , AND JAMES F. POWER 

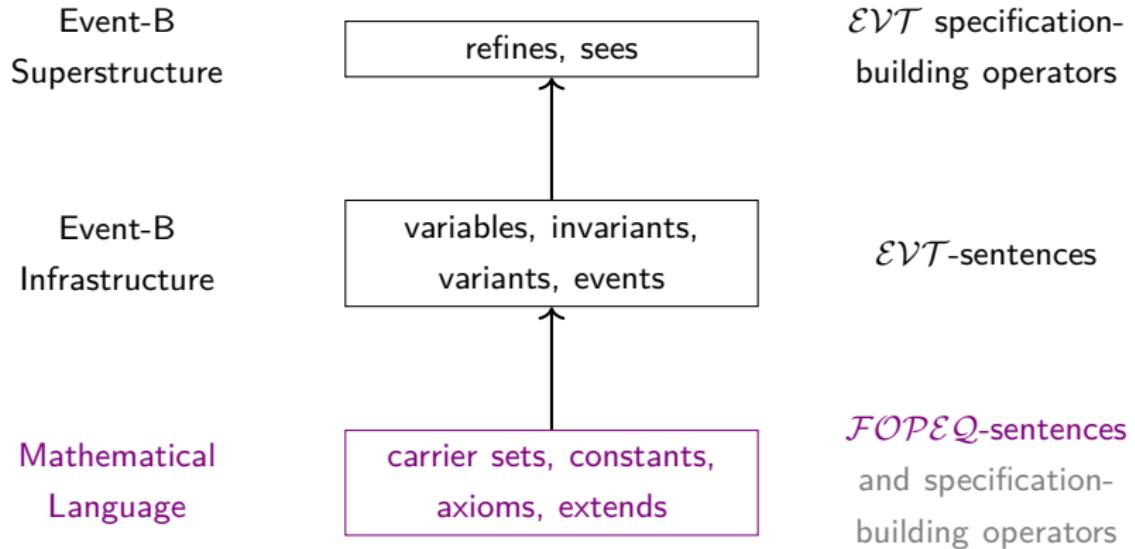
Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Event-B Institution?

The Three-Layer Model



The Three-Layer Model



The \mathcal{FOPEQ} Interface

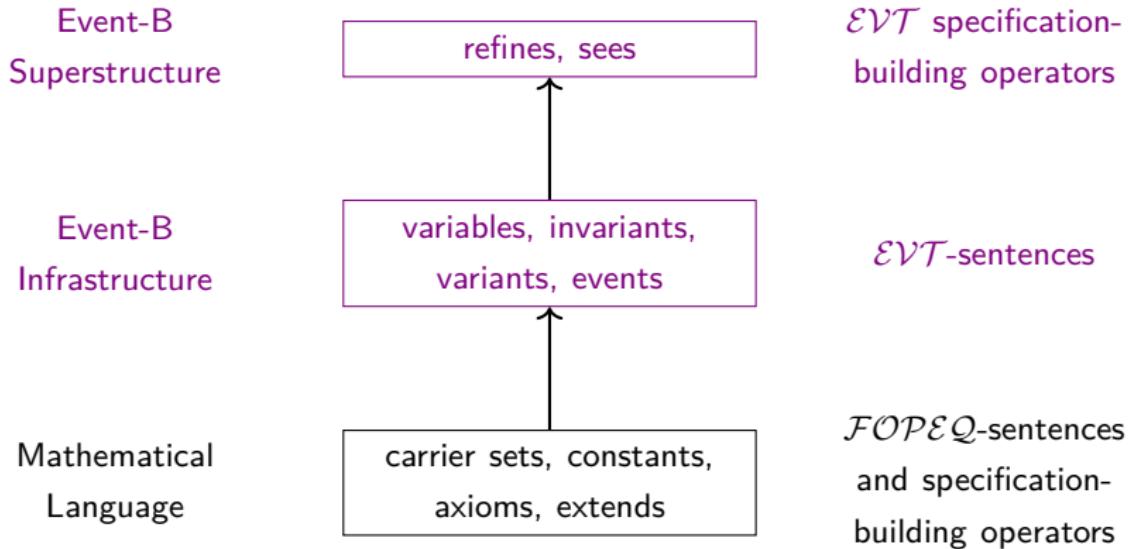
\mathcal{FOPEQ} Operations

- $F.\text{and} : \Sigma\text{-formula}^* \rightarrow \Sigma\text{-formula}$
- $F.\text{lt} : \Sigma\text{-term} \times \Sigma\text{-term} \rightarrow \Sigma\text{-formula}$
- $F.\text{leq} : \Sigma\text{-term} \times \Sigma\text{-term} \rightarrow \Sigma\text{-formula}$
- $F.\text{exists} : \text{VarName}^* \times \Sigma\text{-formula} \rightarrow \Sigma\text{-formula}$
- $F.\iota : \text{VarName}^* \rightarrow \Sigma\text{-formula} \rightarrow \Sigma\text{-formula}$

\mathcal{FOPEQ} Functions

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

The Three-Layer Model



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

```
1 CONTEXT cd
2 CONSTANTS
3   d
4 AXIOMS
5   axm1: d ∈ N
6   axm2: d > 0
7 END

1 MACHINE m0
2 SEES cd
3 VARIABLES
4   n
5 INVARIANTS
6   inv1: n ∈ N
7   inv2: n ≤ d
8 EVENTS
9   Initialisation
10  then
11    act1: n := 0
12 Event ML_out ≡ ordinary
13 when
14   grd1: n < d
15 then
16   act1: n := n + 1
17 Event ML_in ≡ ordinary
18 when
19   grd1: n > 0
20 then
21   act1: n := n - 1
22 END
```

Signature

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

where

$$S = \{N\},$$

$$\Omega = \{0 : N, d : N\},$$

$$\Pi = \{> : N \times N\},$$

$$E = \{(Init \mapsto \text{ordinary}), (ML_in \mapsto \text{ordinary}), (ML_out \mapsto \text{ordinary})\},$$

$$V = \{n : N\}$$

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

Sentences:

```
1 MACHINE m REFINES a SEES ctx
2 VARIABLES  $\bar{x}$ 
3 INVARIANTS  $I(\bar{x})$ 
4 VARIANT  $n(\bar{x})$ 
5 EVENTS
6 Initialisation ordinary
7 then act-name: BA( $\bar{x}'$ )
8 :
9 Event  $e_i \hat{=} \text{convergent}$ 
10 any  $\bar{p}$ 
11 when guard-name:  $G(\bar{x}, \bar{p})$ 
12 with witness-name:  $W(\bar{x}, \bar{p})$ 
13 then act-name: BA( $\bar{x}, \bar{p}, \bar{x}'$ )
14 :
15 END
```

$\{\langle e, I(\bar{x}) \wedge I(\bar{x}') \rangle \mid e \in \text{dom}(\Sigma.E)\}$

$\langle \text{Init}, BA(\bar{x}') \rangle$

$\langle e_i, n(\bar{x}') < n(\bar{x}) \rangle$

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

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

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

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

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

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

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

Satisfaction:

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

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

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

$$\langle A, L, R \rangle \models_{\Sigma} \langle \text{Init}, \phi(\bar{x}') \rangle \Leftrightarrow \forall s' \in L \cdot A^{(s')} \models_{\Sigma_{\mathcal{FOPEQ}}^{(V')}} \phi(\bar{x}')$$



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 

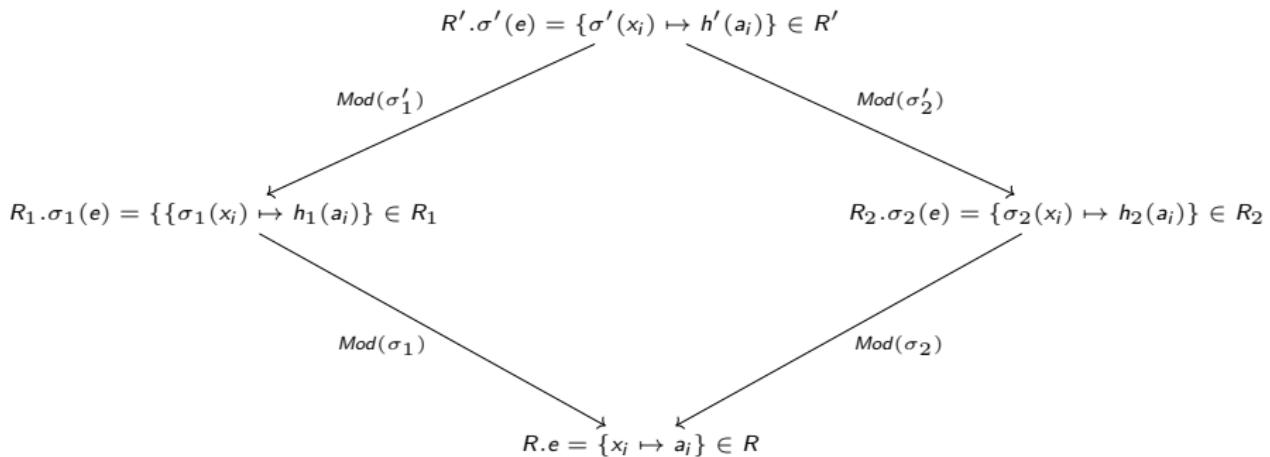
Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Building Specifications?

Specification-Building Operators

Operation	Format	Description
Translation	$SP_1 \text{ with } \sigma$	<p>Renames the signature components of SP_1 using the signature morphism $\sigma : \Sigma_{SP_1} \rightarrow \Sigma'$.</p> $\text{Sig}[SP_1 \text{ with } \sigma] = \Sigma'$ $\text{Mod}[SP_1 \text{ with } \sigma] = \{ M' \in \text{Mod}(\Sigma') \mid M' _\sigma \in \text{Mod}[SP_1] \}.$
Sum	$SP_1 \text{ and } SP_2$	<p>Combines the specifications SP_1 and SP_2.</p> $SP_1 \text{ and } SP_2 = (SP_1 \text{ with } \iota) \cup (SP_2 \text{ with } \iota')$ <p>where $\text{Sig}[SP_1] = \Sigma$, $\text{Sig}[SP_2] = \Sigma'$, $\iota : \Sigma \hookrightarrow \Sigma \cup \Sigma'$, $\iota' : \Sigma' \hookrightarrow \Sigma \cup \Sigma'$</p>
Enrichment	$SP_1 \text{ then } \dots$	<p>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.</p>
Hiding	$SP_1 \text{ hide via } \sigma$	<p>Interprets a specification, SP_1, as one restricted to the signature components of another specified by the signature morphism $\sigma : \Sigma \rightarrow \Sigma_{SP_1}$.</p> $\text{Sig}[SP_1 \text{ hide via } \sigma] = \Sigma$ $\text{Mod}[SP_1 \text{ hide via } \sigma] = \{ M _\sigma \mid M \in \text{Mod}[SP_1] \}.$

Institutions Must Preserve Amalgamation for Specification Building



...proofs are in the paper

Building Specifications in the Event-B Institution

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

$$\mathbb{B} \left[\begin{array}{l} \text{machine } m \\ \text{refines } a \\ \text{sees } ctx_1, \dots, ctx_n \\ mbody \\ \text{end} \end{array} \right] \xi = \left\langle \Sigma, \left[\begin{array}{l} \text{spec } \llbracket m \rrbracket \text{ over } \mathcal{EVT} = \\ \quad \# \text{Include contexts using the comorphism } \rho: \\ \quad (\llbracket ctx_1 \rrbracket \text{ and } \dots \text{ and } \llbracket ctx_n \rrbracket) \text{ with } \rho \\ \quad \# \text{Sentences from the refined machine (if any):} \\ \quad (\text{and } A_\Sigma \llbracket mbody \rrbracket \llbracket a \rrbracket \xi) \\ \quad \text{then} \\ \quad S_\Sigma \llbracket mbody \rrbracket \end{array} \right] \right\rangle$$

where $\Sigma = \xi \llbracket m \rrbracket$.

$A_\Sigma : MachineBody \rightarrow EventName \rightarrow Env \rightarrow |\mathbf{Spec}_{\mathcal{EVT}}|$ #Extract any relevant specification from the refined (abstract) machine

$$A_\Sigma \left[\begin{array}{l} \text{variables } v_1, \dots, v_n \\ \text{invariants } i_1, \dots, i_n \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_{init}, e_1, \dots, e_n \end{array} \right] \llbracket a \rrbracket \xi = I_\Sigma \llbracket i_1 \rrbracket \text{ and } \dots \text{ and } I_\Sigma \llbracket i_n \rrbracket \\ \text{and } R_\Sigma \llbracket e_1 \rrbracket \llbracket a \rrbracket \xi \text{ and } \dots \text{ and } R_\Sigma \llbracket e_n \rrbracket \llbracket a \rrbracket \xi$$

#Conjoin sentences from each event definition

Building Specifications in the Event-B Institution

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

- \bullet $\text{Env} = (\text{MachineName} \cup \text{ContextName}) \rightarrow [\text{Sign}]$ // An environment maps names to signatures
- \bullet $D: \boxed{\begin{array}{l} \text{machines } \xi \in \xi \\ \text{context } \eta \in \xi \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}} \subseteq \xi$ // Process a list of machine/context definitions

- \bullet $D: \boxed{\begin{array}{l} \text{Machine } \xi \rightarrow \text{Env} \\ \text{machines } \eta \in \xi \\ \text{refines } a \\ \text{uses } c_1, \dots, c_m \\ \text{refines } r \\ \text{and } \\ \text{where } \\ \{S, O, H\} = \{I \in \{[\sigma]\} \cup \dots \cup \{I \in [ta]\}\} \end{array}}$ // Extract and store the signature for one machine

$$D: \boxed{\begin{array}{l} \text{Machine } \xi \rightarrow \text{Env} \\ \text{machines } \eta \in \xi \\ \text{refines } a \\ \text{uses } c_1, \dots, c_m \\ \text{refines } r \\ \text{and } \\ \text{where } \\ \{S, O, H\} = \{I \in \{[\sigma]\} \cup \dots \cup \{I \in [ta]\}\} \end{array}} \quad \xi = \xi \cup \{I \in \{[\sigma]\} \rightarrow (S, O, H, I, V) \mid r(I) \in \xi\}$$

$$\text{and } r: \boxed{\begin{array}{l} [\text{Sign}_{\text{EVT}}] \rightarrow [\text{Sign}_{\text{EVT}}] \\ \text{Signature for the abstract machine } (\text{the refined events } r(I)) \end{array}} = \text{List } \Sigma = \{I \in \{[\sigma]\} \mid \{S, O, H, I, V\} \in \Sigma\}$$

- \bullet $D: \boxed{\begin{array}{l} \text{MachineBody } \rightarrow (\text{EventName} \times \text{Stat}) \times \text{PVName} \times \text{P}(\text{EventName}) \\ \text{variables } v_1, \dots, v_n \\ \text{actions } a_1, \dots, a_m \\ \text{status } s \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}}$ // Extract signature elements from machine-body

$$D: \boxed{\begin{array}{l} \text{MachineBody } \rightarrow (\text{EventName} \times \text{Stat}) \times \text{PVName} \times \text{P}(\text{EventName}) \\ \text{variables } v_1, \dots, v_n \\ \text{actions } a_1, \dots, a_m \\ \text{status } s \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}} \quad \xi = \{I, V, H, A\}$$

$$\text{variant } n \\ \text{events } e_1, \dots, e_n \\ \text{where } \\ E = \{def[e_1], def[e_2], \dots, def[e_n]\} \\ V = \{def[v_1], def[v_n]\} \\ RA = \{ref[e_1], ref[e_2], \dots, ref[e_n]\}$$

- \bullet $\text{def: Event } \rightarrow (\text{EventName} \times \text{Stat})$ // Extract event name / status from an event definition

$$\text{def: Event } \rightarrow (\text{EventName} \times \text{Stat}) \\ \text{def: event } \& \text{status } e_1, \dots, e_m \& \text{end} = \{e \mapsto e\}$$

- \bullet $\text{ref: Event } \rightarrow \mathbb{D}[\text{EventName}]$ // Extract names of refined events from an event definition

$$\text{ref: Event } \rightarrow \mathbb{D}[\text{EventName}] \\ \text{ref: event } \& \text{status } e_1, \dots, e_m \& \text{end} = \{[e]\}, \{[s]\}$$

- \bullet $D: \boxed{\begin{array}{l} \text{Context } \xi \rightarrow \text{Env} \\ \text{contexts } \eta \in \xi \\ \text{extends } c_1, \dots, c_m \\ \text{actions } a_1, \dots, a_m \\ \text{status } s \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}}$ // Extract and store the signature for one context

$$D: \boxed{\begin{array}{l} \text{Context } \xi \rightarrow \text{Env} \\ \text{contexts } \eta \in \xi \\ \text{extends } c_1, \dots, c_m \\ \text{actions } a_1, \dots, a_m \\ \text{status } s \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}} \quad \xi = \xi \cup \{I \in \{[\sigma]\} \rightarrow (\mathbb{D}[\text{hd}], \{I \in \{[\sigma]\} \cup \dots \cup \{I \in [ta]\}\})\}$$

- \bullet $D: \boxed{\begin{array}{l} \text{ContextBody } \rightarrow [\text{Sign}_{\text{EVT}}] \\ \text{sets } s_1, \dots, s_n \\ \text{constants } c_1, \dots, c_m \\ \text{actions } a_1, \dots, a_n \\ \text{status } s \\ \text{theorems } t_1, \dots, t_n \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}}$ // Extract the FOPQ/Q signature from a context body

$$D: \boxed{\begin{array}{l} \text{ContextBody } \rightarrow [\text{Sign}_{\text{EVT}}] \\ \text{sets } s_1, \dots, s_n \\ \text{constants } c_1, \dots, c_m \\ \text{actions } a_1, \dots, a_n \\ \text{status } s \\ \text{theorems } t_1, \dots, t_n \\ \text{hd } \eta \in \mathbb{D} \{ \text{[hd]} \} \end{array}} \quad \xi = (S, O, H)$$

$$\text{where } (S, O, H) = \mathbb{M}[\{s_1, \dots, s_n\}, \{c_1, \dots, c_m\}, \{a_1, \dots, a_n\}, \{t_1, \dots, t_n\}]$$

- \bullet $R_E: \boxed{\begin{array}{l} \text{MachineBody } \rightarrow \text{Sentences } (\Sigma) \\ \text{variables } v_1, \dots, v_n \\ \text{actions } a_1, \dots, a_m \\ \text{status } s \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_1, \dots, e_n \end{array}}$ // Build sentences from a machine body

$$R_E: \boxed{\begin{array}{l} \text{MachineBody } \rightarrow \text{Sentences } (\Sigma) \\ \text{variables } v_1, \dots, v_n \\ \text{actions } a_1, \dots, a_m \\ \text{status } s \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_1, \dots, e_n \end{array}} = \left(\begin{array}{l} I_1[I_1] \cup \dots \cup I_n[I_n] \\ \cup \\ I_1[V_1] \cup \dots \cup I_n[V_n] \\ \cup \\ I_1[R_1] \cup \dots \cup I_n[R_n] \\ \cup \\ I_1[e_1] \cup \dots \cup I_n[e_n] \end{array} \right)$$

- \bullet $I_1: \boxed{\begin{array}{l} \text{Loc} \\ \text{hd } \eta = \{I\} \\ \text{I} = \{([v], F_1, (\Sigma, V), P_1, [q]) \mid v \in \text{dom}(\Sigma, V)\} \end{array}}$ // Invariant sentences

$$I_1: \boxed{\begin{array}{l} \text{Loc} \\ \text{hd } \eta = \{I\} \\ \text{I} = \{([v], F_1, (\Sigma, V), P_1, [q]) \mid v \in \text{dom}(\Sigma, V)\} \end{array}} \quad \xi = \{I_1\}$$

- \bullet $V_1: \boxed{\begin{array}{l} \text{Expression} \\ \text{V}_1 = \{([v], F_1 \wedge g((\Sigma, V), P_1, [q])) \mid v \in \text{dom}(\Sigma, V)\} \end{array}}$ // Variant can't increase for variant events

$$V_1: \boxed{\begin{array}{l} \text{Expression} \\ \text{V}_1 = \{([v], F_1 \wedge g((\Sigma, V), P_1, [q])) \mid v \in \text{dom}(\Sigma, V)\} \end{array}} \quad \xi = \{V_1\}$$

- \bullet $R_1: \boxed{\begin{array}{l} \text{InitEvent} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } \text{initialization} \\ \text{status } \text{ordinary} \\ \text{then } a_1, \dots, a_m \\ \text{end} \\ \text{body } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Initial event: get sentences from actions

$$R_1: \boxed{\begin{array}{l} \text{InitEvent} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } \text{initialization} \\ \text{status } \text{ordinary} \\ \text{then } a_1, \dots, a_m \\ \text{end} \\ \text{body } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{([a], BA)\}$$

- \bullet $E_1: \boxed{\begin{array}{l} \text{Event} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Non-initial event: get sentences from event body

$$E_1: \boxed{\begin{array}{l} \text{Event} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{([e], BA)\}$$

- \bullet $F_1: \boxed{\begin{array}{l} \text{EventBody } \rightarrow \Sigma\text{-formulas} \\ \text{var } p_1, \dots, p_n \\ \text{where } gr_1, \dots, gr_n \\ \text{with } w_1, \dots, w_n \\ \text{then } ne_1, \dots, ne_m \end{array}}$ // Build a FOPQ/Q formula for an event definition

$$F_1: \boxed{\begin{array}{l} \text{EventBody } \rightarrow \Sigma\text{-formulas} \\ \text{var } p_1, \dots, p_n \\ \text{where } gr_1, \dots, gr_n \\ \text{with } w_1, \dots, w_n \\ \text{then } ne_1, \dots, ne_m \end{array}} = \text{Fextitl}(p_1, G, W, BA)$$

$$\text{Fextitl}(p_1, G, W, BA) \quad \text{// Formula is existentially quantified over event parameters } p$$

- \bullet $P_1: \boxed{\begin{array}{l} \text{Set } \{p_1, \dots, p_n\} \\ \text{G} = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \\ W = \{P_1[e_1], \dots, P_1[e_n]\} \\ BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // List of parameters

$$P_1: \boxed{\begin{array}{l} \text{Set } \{p_1, \dots, p_n\} \\ \text{G} = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \\ W = \{P_1[e_1], \dots, P_1[e_n]\} \\ BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{P_1\}$$

- \bullet $R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Extract specification from one refined event

$$R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{([e], BA)\}$$

- \bullet $R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Signature of abstract machine

$$R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{E\}$$

- \bullet $R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Statuses from the refined machine (if any)

$$R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{S\}$$

- \bullet $R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Actions from the refined machine (if any)

$$R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{A\}$$

- \bullet $R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Context: get FOPQ/Q sentences from actions

$$R_E: \boxed{\begin{array}{l} \text{EventName} \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{P_1[s_1], \dots, P_1[s_n]\}$$

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

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Specification } \rightarrow \text{Env} \rightarrow [\text{Spec}]^* \\ \text{B } \{ \} = \{ \} \end{array}}$ // Process specifications in an environment, build a list of structured specifications

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Machine } \xi \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{EVT}}]^* \\ \text{B } \{ \} = \{ \} \end{array}}$ // Build an EVT structured specification for one machine

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Machine } \xi \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{EVT}}]^* \\ \text{B } \{ \} = \{ \} \end{array}} \quad \xi = \left\langle \Sigma, \begin{array}{l} \text{spec } [n] \\ \text{includes } \Sigma^* \\ \text{B } \{ \} \text{ contains any theorems } p: \\ ([p] \& \dots \& [p]) \text{ with } \rho \\ \text{in } \text{Theorems from the refined machine } (\text{if any}) \\ \text{then } \text{Ac}[\text{hdbody}][[p]] \\ \text{else } \Sigma_{\{p\}}[\text{body}] \end{array} \right\rangle$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{MachineBody } \rightarrow \text{EventName} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{EVT}}] \\ \text{B } \{ \} = \{ \} \end{array}}$ // Extract any relevant specification from the refined machine

$$\mathbb{B}: \boxed{\begin{array}{l} \text{MachineBody } \rightarrow \text{EventName} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{EVT}}] \\ \text{B } \{ \} = \{ \} \end{array}} \quad \xi = \{I_1\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Extract sentences from each event definition

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{([e], BA)\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Extract specification from one refined event

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{([e], BA)\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Signatures of abstract machine

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{E\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Statuses from the refined event sentences (if any)

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{S\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Actions from the refined event (if any)

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} \quad \xi = \{A\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Context: get FOPQ/Q sentences from actions

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{P_1[s_1], \dots, P_1[s_n]\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Context: get FOPQ/Q sentences from predicates

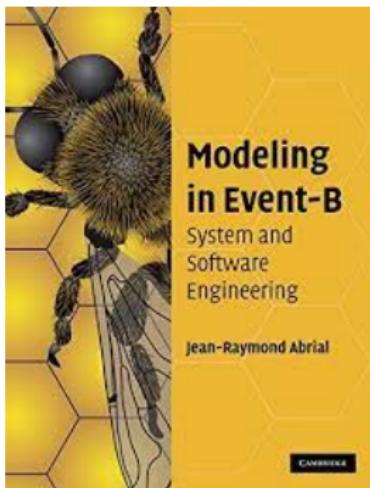
$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{P_1[p_1], \dots, P_1[p_n]\}$$

- \bullet $\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}}$ // Context: get FOPQ/Q sentences from theorems

$$\mathbb{B}: \boxed{\begin{array}{l} \text{Event } \xi \rightarrow \text{Env} \rightarrow \text{Sentences } (\Sigma) \\ \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } BA = \text{FAnd}(P_1[e_1], \dots, P_1[e_n]) \end{array}} = \{P_1[t_1], \dots, P_1[t_n]\}$$

...it's all in the paper.

An Example: Cars On A Bridge



An Example: Cars On A Bridge

```
1 CONTEXT cd
2 CONSTANTS d
3 AXIOMS
4   axm1: d ∈ N
5   axm2: d > 0
6 END

7 MACHINE m0
8 SEES cd
9 VARIABLES n
10 INVARIANTS
11   inv1: n ∈ N
12   inv2: n ≤ d
13 EVENTS
14   Initialisation
15     then act1: n := 0
16   Event ML_out ≡ ordinary
17     when grd1: n < d
18     then act1: n := n + 1
19   Event ML_in ≡ ordinary
20     when grd1: n > 0
21     then act1: n := n - 1
22 END

1 spec CD =
2 sort N
3 ops d:N
4 . d > 0
5 end

6 spec M0 =
7 CD
8 then
9 ops n:N
10 . n ≤ d
11 EVENTS
12   Initialisation
13     thenAct n := 0
14   Event ML_out ≡ ordinary
15     when n < d
16     thenAct n := n + 1
17   Event ML_in ≡ ordinary
18     when n > 0
19     thenAct n := n - 1
20 end
```

An Example: Cars On A Bridge

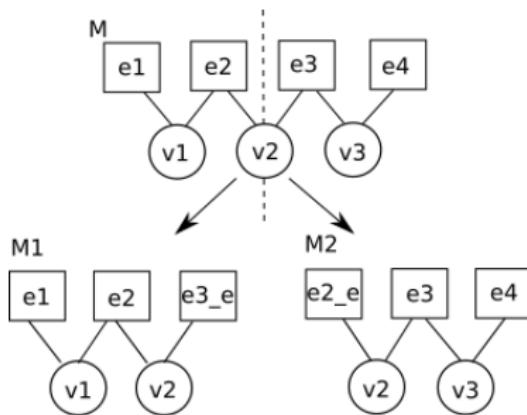
```
1 spec M1 =
2   M0 and CD
3   then
4     ops a:N
5       b:N
6       c:N
7     . n = a + b + c
8     a = 0 ∨ c = 0
9   variant 2 * a + b
10  EVENTS
11    Initialisation
12      thenAct a := 0
13      b := 0
14      c := 0
15    Event ML_out ≡ ordinary
16      when a + b < d
17      c = 0
18      thenAct a := a+1
19
20
21
22
23
24
25
26
27
28
29
30
31 end
```

Event IL_in $\hat{=}$ convergent
when $a > 0$
thenAct $a := a-1$
 $b := b+1$
Event IL_out $\hat{=}$ convergent
when $0 < b$
 $a = 0$
thenAct $b := b-1$
 $c := c+1$
Event ML_in $\hat{=}$ ordinary
when $c > 0$
thenAct $c := c-1$

...more detail in the paper.

So What?

Modularisation via Specification Building: Shared Variable



```
1 spec M1 =
2   (M hide via  $\sigma_1$ )
3   with {e3  $\mapsto$  e3_e}
4 end
5 where  $\sigma_1 = \{v1 \mapsto v1, v2 \mapsto v2,$ 
6            $e1 \mapsto e1, e2 \mapsto e2,$ 
7            $e3 \mapsto e3\}$ 

8 spec M2 =
9   (M hide via  $\sigma_2$ )
10  with {e2  $\mapsto$  e2_e}
11 end
12 where  $\sigma_2 = \{v2 \mapsto v2, v3 \mapsto v3,$ 
13            $e2 \mapsto e2, e3 \mapsto e3,$ 
14            $e4 \mapsto e4\}$ 
```

...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:

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

- ② Signatures are different:

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

```
1 refinement REFO : M0 to M1 =
2   ML_in ↪ ML_in, ML_out ↪ ML_out
3 end

4 refinement REF1A : M1 to M2 =
5   ML_in ↪ ML_in, ML_out ↪ ML_out1, IL_in ↪ IL_in, IL_out ↪ IL_out1
6 end

7 refinement REF1B : M1 to M2 =
8   ML_in ↪ ML_in, ML_out ↪ ML_out2, IL_in ↪ IL_in, IL_out ↪ IL_out2
9 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:

- ➊ 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:

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

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

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/>