

Journal First: Building Specifications in the Event-B Institution

Marie Farrell, Rosemary Monahan and James F. Power

CPS Seminar @ University of Southampton
June 29, 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:

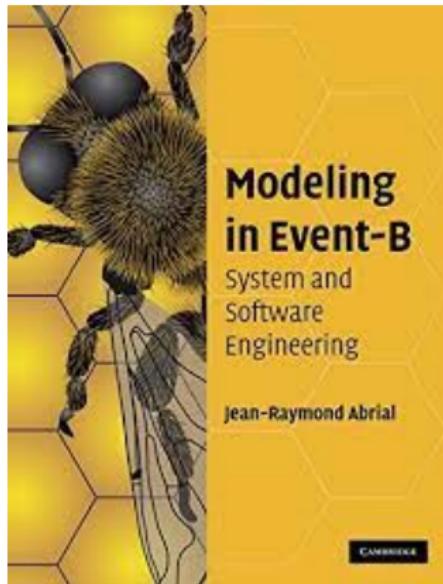
There will be equations and commutative diagrams on these slides but I will only superficially explain them. All of the details and proofs are in the paper.

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.

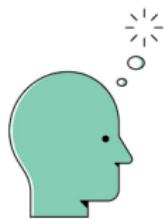
Modularisation

- 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

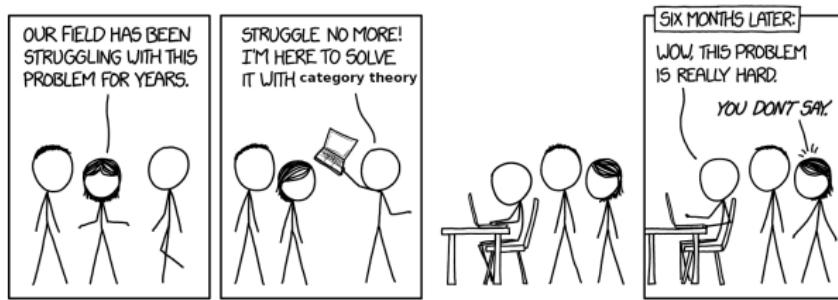
BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

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** \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 $\sigma : \Sigma \rightarrow \Sigma'$.

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

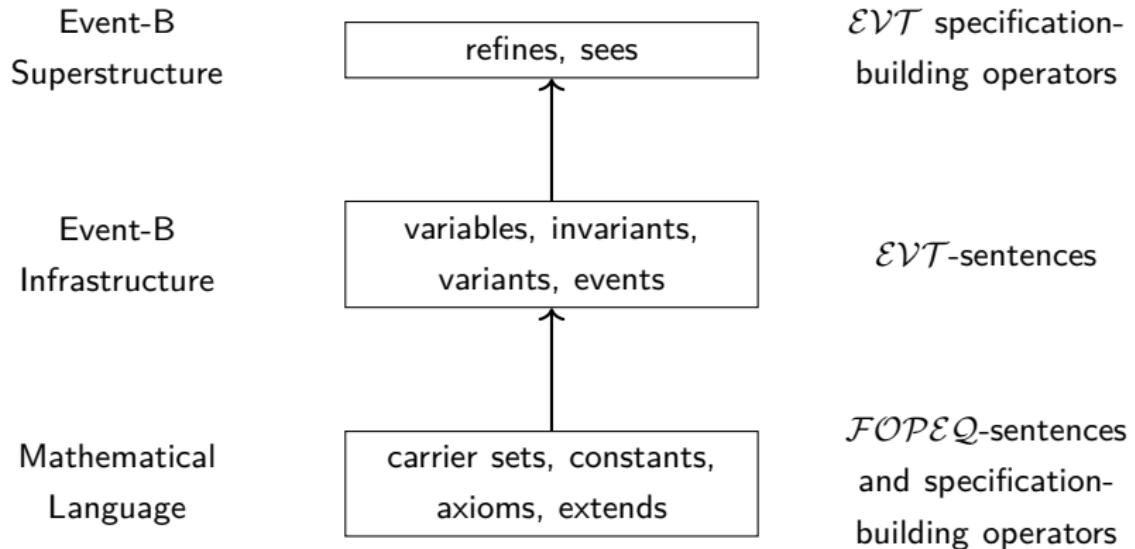
BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

MARIE FARRELL , ROSEMARY MONAHAN , AND JAMES F. POWER 

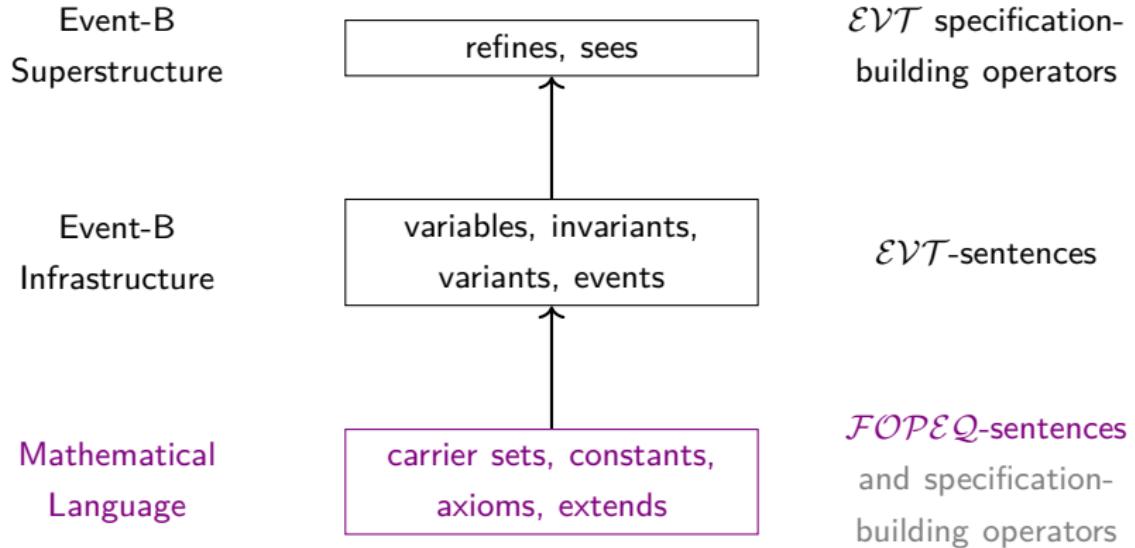
Department of Computer Science and Hamilton Institute, Maynooth University, Ireland
e-mail address: marie.farrell@mu.ie

Event-B Institution?

The Three-Layer Model



The Three-Layer Model



The \mathcal{FOPEQ} Interface

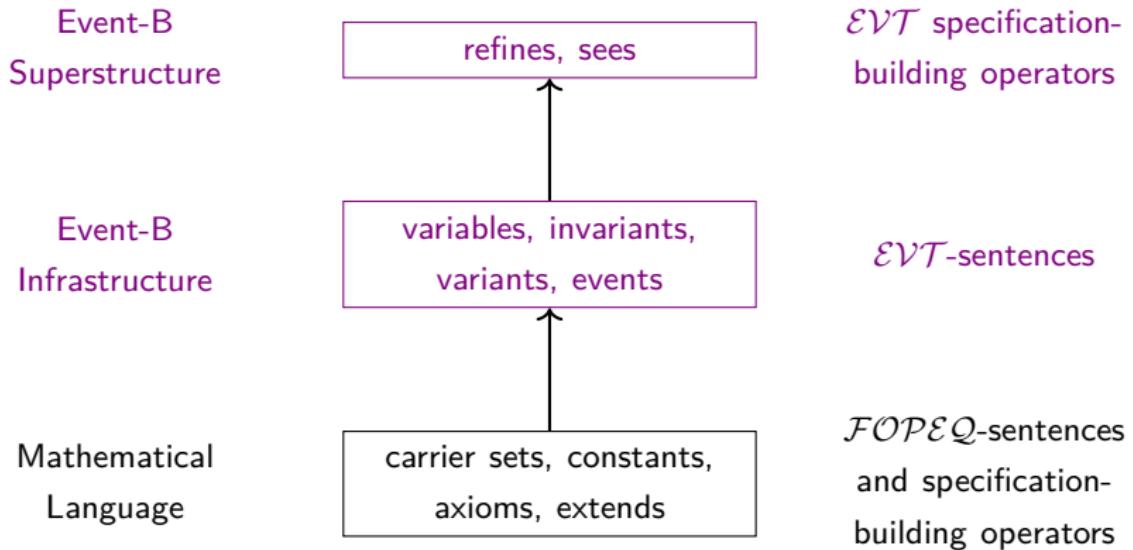
\mathcal{FOPEQ} Operations

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

\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

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

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

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

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

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

$\mathcal{EV}\mathcal{T}$ - 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  $e_i \doteq$  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

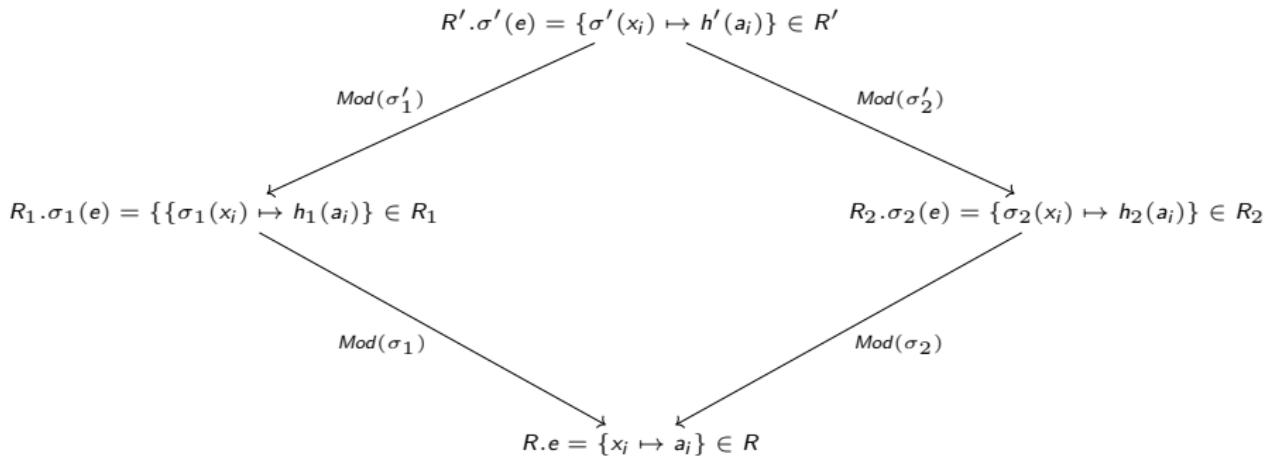
Operation	Format	Description
Translation	$SP_1 \text{ with } \sigma$	<p>Renames the signature components of SP_1 using the signature morphism $\sigma : \Sigma_{SP_1} \rightarrow \Sigma'$.</p> $\text{Sig}[SP_1 \text{ with } \sigma] = \Sigma'$ $\text{Mod}[SP_1 \text{ with } \sigma] = \{ M' \in \text{Mod}(\Sigma') \mid M' _\sigma \in \text{Mod}[SP_1] \}.$
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> $\text{Sig}[SP_1 \text{ hide via } \sigma] = \Sigma$ $\text{Mod}[SP_1 \text{ hide via } \sigma] = \{ M _\sigma \mid M \in \text{Mod}[SP_1] \}.$

We Need Pushouts for Specification Building

Given two \mathcal{EVT} -signature morphisms $\sigma_1 : \Sigma \rightarrow \Sigma_1$ and $\sigma_2 : \Sigma \rightarrow \Sigma_2$ a pushout is a triple $(\Sigma', \sigma'_1, \sigma'_2)$ that satisfies the universal property: for all triples $(\Sigma'', \sigma''_1, \sigma''_2)$ there exists a unique morphism $u : \Sigma' \rightarrow \Sigma''$ such that the diagram below commutes.

$$\begin{array}{ccc} & \Sigma & \\ \sigma_1 \swarrow & & \searrow \sigma_2 \\ \Sigma_1 & \sigma'_1 & \Sigma_2 \\ & \sigma''_1 & \sigma''_2 \\ \sigma''_1 \searrow & \Sigma' & \nearrow \sigma''_2 \\ & u & \end{array}$$

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

$\mathbb{R}_\Sigma : Event \rightarrow EventName \rightarrow Env \rightarrow | Spec_{\mathcal{EVT}} | \quad \#Extract\ specification\ from\ one\ refined\ event$

let

$\Sigma_a = \xi[\![a]\!], \quad \#Signature\ of\ abstract\ machine$
 $\# Use \Sigma_h, \sigma_h \text{ to select only refined events:}$
 $\Sigma_h = \langle \Sigma_a.S, \Sigma_a.\Omega, \Sigma_a.\Pi,$
 $\{\!\![e_1]\!\], \dots, \{\!\![e_n]\!\] \} \triangleleft \Sigma_a.E, \Sigma_a.V \rangle,$

$\sigma_h : \Sigma_h \hookrightarrow \Sigma_a,$
 $\# Use \sigma_m \text{ to reassign refined event sentences to } e_c:$
 $\sigma_m : \Sigma_h \rightarrow \Sigma$
 $\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 \{\!\![e_c]\!\] \triangleleft \Sigma.E, \\ \Sigma_h.V \hookrightarrow \Sigma.V \end{array} \right\rangle$

in

$(\{\!\![a]\!\] \text{ hide via } \sigma_h) \text{ with } \sigma_m.$

$\mathbb{I}_\Sigma : LabelledPred \rightarrow Sen_{\mathcal{EVT}}(\Sigma) \quad \#Invariant\ sentences$

$\mathbb{I}_\Sigma \quad [\![i]\!] = \{ \langle [\![e]\!], F.and(\mathbb{P}_\Sigma[\![i]\!], F.\iota(\Sigma.V)(\mathbb{P}_\Sigma[\![i]\!])) \mid e \in \text{dom}(\Sigma.E) \}$

$\mathbb{V}_\Sigma : Expression \rightarrow Sen_{\mathcal{EVT}}(\Sigma) \quad \#Variant\ can't\ increase\ for\ non-ord.\ events$

$\mathbb{V}_\Sigma \quad [\![n]\!] = \begin{aligned} & \{ \langle [\![e]\!], F.lt(F.\iota(\Sigma.V)(\mathbb{T}_\Sigma[\![n]\!]), \mathbb{T}_\Sigma[\![n]\!]) \rangle \mid (e \mapsto \text{convergent}) \in \Sigma.E \} \\ & \cup \{ \langle [\![e]\!], F.leq(F.\iota(\Sigma.V)(\mathbb{T}_\Sigma[\![n]\!]), \mathbb{T}_\Sigma[\![n]\!]) \rangle \mid (e \mapsto \text{anticipated}) \in \Sigma.E \} \end{aligned}$

$\mathbb{E}_\Sigma : InitEvent \rightarrow Sen_{\mathcal{EVT}}(\Sigma) \quad \#Initial\ event:\ get\ sentences\ from\ actions$

$\mathbb{E}_\Sigma \quad \left[\begin{array}{l} \text{event Initialisation} \\ \text{status ordinary} \\ \text{then } act_1, \dots, act_n \\ \text{end} \end{array} \right] = \{ \langle \text{Init}, BA \rangle \}$

where

$BA = F.and(\mathbb{P}_\Sigma[\![act_1]\!], \dots, \mathbb{P}_\Sigma[\![act_n]\!])$

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 $D: \boxed{[\cdot] \in \xi \subseteq \xi}$
- $D: \boxed{[hd : t] \in \xi \subseteq \mathbb{D}[[\cdot]] \cup [\cdot]} \quad \xi = \xi \cup \{[\cdot] \mapsto ([S, O, I] \cup E, V) \cup r(\xi)[\cdot]\}$

- \bullet $D: \boxed{\text{Machine} \rightarrow \text{Env}} \quad \xi \text{ Extract and store the signature for one machine}$
- \bullet $\begin{cases} \text{machines } m \\ \text{refinements } a \\ \text{sees } c_1, \dots, c_m \\ \text{refines } r \\ \text{and } \\ \text{where } \\ [S, O, I] = \{[\cdot] \mid [\cdot] \in \xi \cup \{[\cdot] \mapsto ([S, O, I], E, V) \cup r(\xi)[\cdot]\}\} \end{cases} \quad \xi = \xi \cup \{[\cdot] \mapsto ([S, O, I], E, V) \cup r(\xi)[\cdot]\}$

- \bullet $D: \boxed{\text{MachineBody} \rightarrow \text{Env}} \quad \xi \text{ Extract the signature for one machine body}$
- \bullet $\begin{cases} \text{variables } v_1, \dots, v_n \\ \text{invariants } i_1, \dots, i_n \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_1, \dots, e_n \\ \text{where } \\ [S, O, I] = \{[\cdot] \mid [\cdot] \in \xi \cup \{[\cdot] \mapsto ([S, O, I], E, V) \cup r(\xi)[\cdot]\}\} \end{cases} \quad \xi = \xi \cup \{[\cdot] \mapsto ([S, O, I], E, V) \cup r(\xi)[\cdot]\}$

- \bullet $D: \boxed{\text{MachineBody} \rightarrow (\text{EventName} \times \text{Stat}) \times \text{PVName} \times \text{P}(\text{EventName})} \quad \xi \text{ Extract signature elements from machine-body}$

- \bullet $D: \boxed{\begin{cases} \text{variables } v_1, \dots, v_n \\ \text{invariants } i_1, \dots, i_n \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_1, \dots, e_n \\ \text{where } \\ E = \{def[v_1], def[v_2], \dots, def[v_n]\} \\ V = \{def[i_1], \dots, def[i_n]\} \\ RA = refP([v_1] \cup \dots \cup refP[e_1]) \end{cases} \quad \xi = \{E, V, RA\}}$

- \bullet $D: \boxed{\text{Event} \rightarrow (\text{EventName} \times \text{Stat})} \quad \xi \text{ Extract event name } E \text{ / status from an event definition}$
- \bullet $\begin{cases} \text{def } [event] = \{status\} \\ \text{def } [event \wedge status] = \{refine \text{ } [event] \} \\ \text{def } [event \wedge status \wedge refinements] = \{refine \text{ } [event] \} \end{cases} \quad \xi = \{E \mapsto \{status\}, ordinary\}$

- \bullet $\text{ref: Event} \rightarrow \mathbb{P}(\text{EventName}) \quad \xi \text{ Extract names of refined events from an event definition}$
- $\text{ref: } [event \wedge status \wedge refinements] \rightarrow \{refine \text{ } [event] \} \quad \xi = \{E \mapsto \{status\}\}$

- \bullet $D: \boxed{\text{Context} \rightarrow \text{Env}} \quad \xi \text{ Extract and store the signature for one context}$
- \bullet $\begin{cases} \text{context } ct \\ \text{extends } ct_1, \dots, ct_n \\ \text{actions } a_1, \dots, a_n \\ \text{theorems } t_1, \dots, t_n \\ \text{sets } s_1, \dots, s_n \\ \text{constants } c_1, \dots, c_n \\ \text{axioms } a_1, \dots, a_n \\ \text{where } \\ [S, O, I] = M[[s_1, \dots, s_n], \{c_1, \dots, c_n\}, \{a_1, \dots, a_n\}] \end{cases} \quad \xi = \xi \cup \{[\cdot] \mapsto ([S, O, I] \cup \{[\cdot] \mapsto ([S, O, I], E, V) \cup r(\xi)[\cdot]\})\}$

- \bullet $D: \boxed{\text{ContextBody} \rightarrow [\text{Sign}_{\text{purereg}}]} \quad \xi \text{ Extract the FOPQ/Q signature from a context body}$
- \bullet $\begin{cases} \text{constants } c_1, \dots, c_n \\ \text{actions } a_1, \dots, a_n \\ \text{theorems } t_1, \dots, t_n \\ \text{sets } s_1, \dots, s_n \\ \text{where } \\ [S, O, I] = M[[s_1, \dots, s_n], \{c_1, \dots, c_n\}, \{a_1, \dots, a_n\}] \end{cases} \quad \xi = \{S, O, I\}$

- \bullet $D: \boxed{\text{ContextBody} \rightarrow [\text{Sign}_{\text{purereg}}]} \quad \xi \text{ Extract the FOPQ/Q signature from a context body}$
- \bullet $\begin{cases} \text{constants } c_1, \dots, c_n \\ \text{actions } a_1, \dots, a_n \\ \text{theorems } t_1, \dots, t_n \\ \text{sets } s_1, \dots, s_n \\ \text{where } \\ [S, O, I] = M[[s_1, \dots, s_n], \{c_1, \dots, c_n\}, \{a_1, \dots, a_n\}] \end{cases} \quad \xi = \{S, O, I\}$

- \bullet $R_E: \boxed{\text{Env} \rightarrow \text{Succinct}(\Sigma)} \quad \xi \text{ Build sentences from a machine body}$

$$R_E: \boxed{\begin{cases} \text{variables } v_1, \dots, v_n \\ \text{invariants } i_1, \dots, i_n \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_1, \dots, e_n \end{cases} \quad \xi = \left\{ \begin{array}{l} I_1[e_1] \cup I_2[e_2] \cup \dots \cup I_n[e_n] \\ \cup \\ V_1[e_1] \cup V_2[e_2] \cup \dots \cup V_n[e_n] \\ \cup \\ R_1[e_1] \cup R_2[e_2] \cup \dots \cup R_n[e_n] \end{array} \right\}}$$

- \bullet $I_1: \boxed{\text{Loc}(\text{BodyPred} \rightarrow \text{Succinct}(\Sigma))} \quad \xi = \{I_1 = \{[\cdot], F_1(\Sigma), P_1(\Sigma)\} \mid [\cdot] \in \text{dom}(\Sigma)\}$

- \bullet $V_1: \boxed{\text{Expression} \rightarrow \text{Succinct}(\Sigma)}$
- $V_1: \boxed{[v] = \{([\cdot], F_1 \circ g^*(\Sigma, V), P_1(\Sigma), T_1[\cdot]) \mid (\text{e} \leftrightarrow \text{convergent}) \in \Sigma, E\}}$

- \bullet $R_1: \boxed{\text{InitEvent} \rightarrow \text{Succinct}(\Sigma)} \quad \xi \text{ Build event: get sentences from actions}$

$$R_1: \boxed{\begin{cases} \text{event } \text{initialization} \\ \text{status } \text{ordinary} \\ \text{then } a_1, \dots, a_n \\ \text{end } \\ \text{body } \\ \text{where } \\ BA = F_1\text{and}(P_1[e_1], \dots, P_1[e_n]) \end{cases} \quad \xi = \left\{ \{Init, BA\} \right\}}$$

- \bullet $R_2: \boxed{\text{Event} \rightarrow \text{Succinct}(\Sigma)} \quad \xi \text{ Non-initial event: get sentences from event body}$

$$R_2: \boxed{\begin{cases} \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } a_1, \dots, a_n \\ \text{end } \\ \text{body } \\ \text{where } \\ BA = F_2\text{and}(P_2[e_1], \dots, P_2[e_n]) \end{cases} \quad \xi = \{e\}}$$

- \bullet $R_3: \boxed{\text{EventBody} \rightarrow \Sigma\text{-formulas}} \quad \xi \text{ Build a FOPQ/Q formula for an event definition}$

$$R_3: \boxed{\begin{cases} \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } a_1, \dots, a_n \\ \text{with } w_1, \dots, w_n \\ \text{end } \\ \text{body } \\ \text{where } \\ F = \{[\cdot]\}, \dots, \{[\cdot]\} \\ G = F \text{and}(P_3[e_1], \dots, P_3[e_n]) \\ H = F \text{and}(P_3[e_1], \dots, P_3[e_n]) \\ BA = F_3\text{and}(P_3[e_1], \dots, P_3[e_n]) \end{cases} \quad \xi = \{F, \text{event } e, F \circ G, W, BA\}}$$

- \bullet $R_4: \boxed{\text{Event} \rightarrow \Sigma\text{-formulas}} \quad \xi \text{ Build a FOPQ/Q formula for an event definition}$

$$R_4: \boxed{\begin{cases} \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } a_1, \dots, a_n \\ \text{body } \\ \text{where } \\ F = \{[\cdot]\}, \dots, \{[\cdot]\} \\ G = F \text{and}(P_4[e_1], \dots, P_4[e_n]) \\ H = F \text{and}(P_4[e_1], \dots, P_4[e_n]) \\ BA = F_4\text{and}(P_4[e_1], \dots, P_4[e_n]) \end{cases} \quad \xi = \{F, \text{event } e, F \circ G, W, BA\}}$$

- \bullet $R_5: \boxed{\text{EventBody} \rightarrow \text{Succinct}(\Sigma)} \quad \xi \text{ Build FOPQ/Q sentences from events}$

$$R_5: \boxed{\begin{cases} \text{sets } s_1, \dots, s_n \\ \text{constants } c_1, \dots, c_n \\ \text{actions } a_1, \dots, a_n \\ \text{theorems } t_1, \dots, t_n \\ \text{where } \\ [S, O, I] = M[[s_1, \dots, s_n], \{c_1, \dots, c_n\}, \{a_1, \dots, a_n\}] \end{cases} \quad \xi = \{P_5[s_1], \dots, P_5[s_n]\}}$$

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

- \bullet $R_S: \boxed{\text{Specification} \rightarrow \text{Env} \rightarrow [\text{Spec}]} \quad \xi \text{ Process specifications in an environment, build a list of structured specifications}$

- \bullet $R_M: \boxed{\text{Machine} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{PVT}}]} \quad \xi \text{ Build an EVT structured specification for one machine}$

$$R_M: \boxed{\begin{cases} \text{machine } m \\ \text{refinements } a \\ \text{sees } c_1, \dots, c_n \\ \text{body } \\ \text{where } \\ \text{spec } [m] \text{ includes } \Sigma \\ \text{includes } c_1 \dots \dots c_n \text{ with } \rho \\ \text{in refinements from the refined machine } (f \text{ arg: } f \text{ and } Ac[f] \text{ body: } f \text{ and } Ac[fbody][f][\rho]) \\ \text{then } \\ \Sigma = \{m\} \\ \text{where } \Sigma = \{m\} \end{cases} \quad \xi = \left\{ \Sigma \right\}}$$

- \bullet $R_E: \boxed{\text{MachineBody} \rightarrow \text{EventName} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{PVT}}]} \quad \xi \text{ Extract any relevant specification from the refined (abstract) machine}$

$$R_E: \boxed{\begin{cases} \text{variables } v_1, \dots, v_n \\ \text{invariants } i_1, \dots, i_n \\ \text{theorems } t_1, \dots, t_n \\ \text{variant } n \\ \text{events } e_1, \dots, e_n \\ \text{body } \\ \text{where } \\ BA = F_4\text{and}(P_4[e_1], \dots, P_4[e_n]) \end{cases} \quad \xi = \{BA = \{e_1\} \cup \dots \cup \{e_n\} \text{ and } BA = \{v_1, \dots, v_n\} \cup \dots \cup \{v_n\}\}}$$

- \bullet $R_E: \boxed{\text{Event} \rightarrow \text{EventName} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{PVT}}]} \quad \xi \text{ Extract specification from one refined event}$

$$R_E: \boxed{\begin{cases} \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } a_1, \dots, a_n \\ \text{end } \\ \text{body } \\ \text{where } \\ \Sigma = \{e\} \\ \text{Signature of abstract machine} \\ \text{if } \Sigma \subseteq \Sigma_e \text{ to refine only refined events:} \\ \Sigma_e = (\Sigma, S \rightarrow \Sigma_e, O \rightarrow \Sigma_e, \{e\} \rightarrow \{e\}, \{e\}_1 \rightarrow \{e\}_1, \dots, \{e\}_n \rightarrow \{e\}_n, E \rightarrow \Sigma_e, V \rightarrow \Sigma_e) \\ a_1 : \Sigma_e \rightarrow \Sigma_e \\ \dots \\ a_n : \Sigma_e \rightarrow \Sigma_e \\ \text{if } \Sigma \subseteq \Sigma_e \text{ to merge refined event sentences to } e: \\ \Sigma_e = (\Sigma, S \rightarrow \Sigma, O \rightarrow \Sigma, E \rightarrow \Sigma, V \rightarrow \Sigma) \\ \sigma_n = \left\{ \begin{array}{l} \Sigma_e, S \rightarrow \Sigma, O \rightarrow \Sigma, E \rightarrow \Sigma, V \rightarrow \Sigma \\ \Sigma_e \rightarrow \Sigma, S \rightarrow \Sigma, O \rightarrow \Sigma, E \rightarrow \Sigma, V \rightarrow \Sigma \end{array} \right\} \\ \{e\} \text{ side via } e \text{ with } \sigma_n. \end{cases} \quad \xi = \{e\}}$$

- \bullet $R_E: \boxed{\text{Event} \rightarrow \text{EventName} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{PVT}}]} \quad \xi \text{ Extract specification from one refined event}$

$$R_E: \boxed{\begin{cases} \text{event } e \\ \text{status } s \\ \text{status } e_1, \dots, e_n \\ \text{then } a_1, \dots, a_n \\ \text{body } \\ \text{where } \\ \Sigma = \{e\} \\ \text{Signature of abstract machine} \\ \text{if } \Sigma \subseteq \Sigma_e \text{ to refine only refined events:} \\ \Sigma_e = (\Sigma, S \rightarrow \Sigma_e, O \rightarrow \Sigma_e, \{e\} \rightarrow \{e\}, \{e\}_1 \rightarrow \{e\}_1, \dots, \{e\}_n \rightarrow \{e\}_n, E \rightarrow \Sigma_e, V \rightarrow \Sigma_e) \\ a_1 : \Sigma_e \rightarrow \Sigma_e \\ \dots \\ a_n : \Sigma_e \rightarrow \Sigma_e \\ \text{if } \Sigma \subseteq \Sigma_e \text{ to merge refined event sentences to } e: \\ \Sigma_e = (\Sigma, S \rightarrow \Sigma, O \rightarrow \Sigma, E \rightarrow \Sigma, V \rightarrow \Sigma) \\ \sigma_n = \left\{ \begin{array}{l} \Sigma_e, S \rightarrow \Sigma, O \rightarrow \Sigma, E \rightarrow \Sigma, V \rightarrow \Sigma \\ \Sigma_e \rightarrow \Sigma, S \rightarrow \Sigma, O \rightarrow \Sigma, E \rightarrow \Sigma, V \rightarrow \Sigma \end{array} \right\} \\ \{e\} \text{ side via } e \text{ with } \sigma_n. \end{cases} \quad \xi = \{e\}}$$

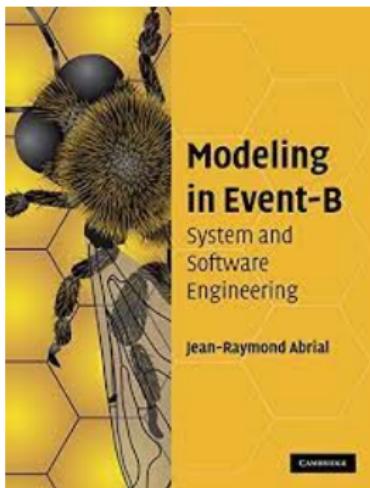
- \bullet $R_E: \boxed{\text{EventBody} \rightarrow \text{EventName} \rightarrow \text{Env} \rightarrow [\text{Spec}_{\text{PVT}}]} \quad \xi \text{ Build a FOPQ/Q structured specification for one event}$

$$R_E: \boxed{\begin{cases} \text{context } ct \\ \text{extends } ct_1, \dots, ct_n \\ \text{actions } a_1, \dots, a_n \\ \text{theorems } t_1, \dots, t_n \\ \text{sets } s_1, \dots, s_n \\ \text{constants } c_1, \dots, c_n \\ \text{axioms } a_1, \dots, a_n \\ \text{where } \\ BA = F_5\text{and}(P_5[e_1], \dots, P_5[e_n]) \end{cases} \quad \xi = \left\{ \Sigma \right\}}$$

where $\Sigma = \{m\}$.

...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

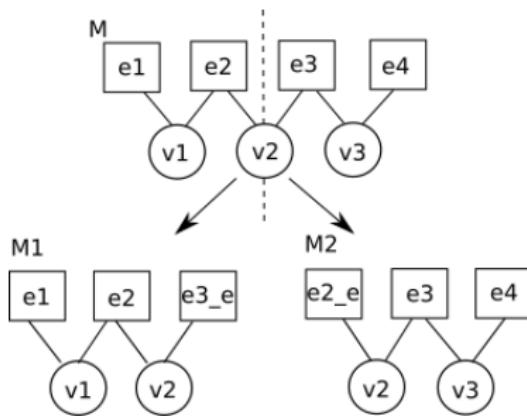
```
1 spec DATAM0 =
2   CD then
3     ops n:N
4       . n ≤ d
5       EVENTS
6         Initialisation
7           thenAct n := 0
8 end

9 spec DATAM1 =
10  DATAM0 then
11    ops a, b, c : N
12      . n = a + b + c
13      a = 0 ∨ c = 0
14      variant 2*a+b
15      EVENTS
16        Initialisation
17          thenAct a := 0
18          b := 0
19          c := 0
20 end

21 spec INOUT =
22   ops v1, v2 : N
23   EVENTS
24     Event out ≡ ordinary
25       when v1 = 0
26       thenAct v2 := v2 + 1
27     Event in ≡ ordinary
28       when v1 > 0
29       thenAct v1 := v1 + 1
30 end

31 spec M1 =
32   DATAM1 and M0 and
33   INOUT with {⟨out, ordinary⟩ ↦ ⟨ML_out, ordinary⟩,
34               ⟨in, ordinary⟩ ↦ ⟨ML_in, ordinary⟩,
35               v1 ↦ c, v2 ↦ a} and
36   INOUT with {⟨out, ordinary⟩ ↦ ⟨IL_out, convergent⟩,
37               ⟨in, ordinary⟩ ↦ ⟨IL_in, convergent⟩,
38               v1 ↦ a, v2 ↦ c}
39 then
40   EVENTS
41     Event ML_out ≡ ordinary
42       when a + b < d
43     Event IL_in ≡ convergent
44       thenAct b := b + 1
45     Event IL_out ≡ convergent
46       when 0 < b
47       thenAct b := b + 1
48 end
```

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.

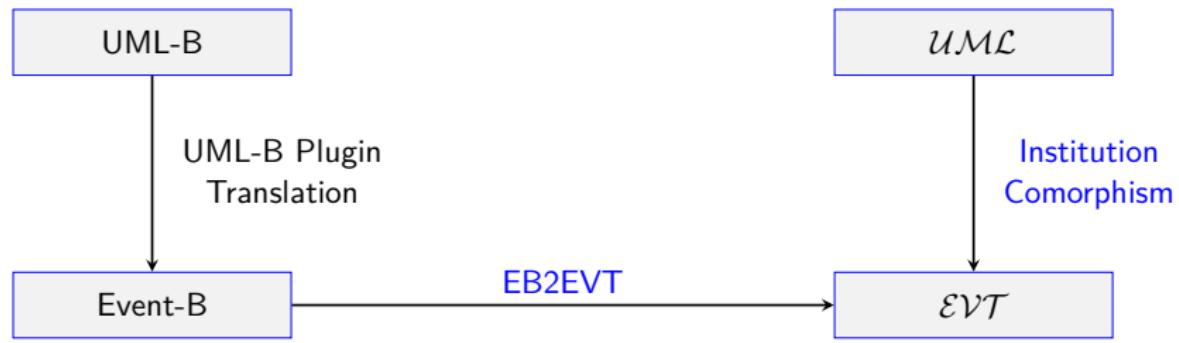
Conclusions and Future Work

- 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.

Not in the paper...

Event-B and the *Rodin Platform*

Institutions and Comorphisms



Connecting Institutions: Institution Comorphism

An institution comorphism $\rho : \mathbf{INS} \rightarrow \mathbf{INS}'$ is composed of:

- A functor $\rho^{\text{Sign}} : \mathbf{Sign} \rightarrow \mathbf{Sign}'$.
- A natural transformation $\rho^{\text{Sen}} : \mathbf{Sen} \rightarrow \rho^{\text{Sign}}$; \mathbf{Sen}' , that is, for each $\Sigma \in |\mathbf{Sign}|$, a function $\rho_{\Sigma}^{\text{Sen}} : \mathbf{Sen}(\Sigma) \rightarrow \mathbf{Sen}'(\rho^{\text{Sign}}(\Sigma))$.
- A natural transformation $\rho^{\text{Mod}} : (\rho^{\text{Sign}})^{\text{op}} ; \mathbf{Mod}' \rightarrow \mathbf{Mod}$, that is, for each $\Sigma \in |\mathbf{Sign}|$, a functor $\rho_{\Sigma}^{\text{Mod}} : \mathbf{Mod}'(\rho^{\text{Sign}}(\Sigma)) \rightarrow \mathbf{Mod}(\Sigma)$.

An institution comorphism must ensure that for any signature $\Sigma \in |\mathbf{Sign}|$, the translations $\rho_{\Sigma}^{\text{Sen}}$ of sentences and $\rho_{\Sigma}^{\text{Mod}}$ of models preserve the satisfaction relation, that is, for any $\psi \in \mathbf{Sen}(\Sigma)$ and $M' \in |\mathbf{Mod}(\rho^{\text{Sign}}(\Sigma))|$:

$$\rho_{\Sigma}^{\text{Mod}}(M') \models_{\Sigma} \psi \Leftrightarrow M' \models'_{\rho^{\text{Sign}}(\Sigma)} \rho_{\Sigma}^{\text{Sen}}(\psi)$$

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/>