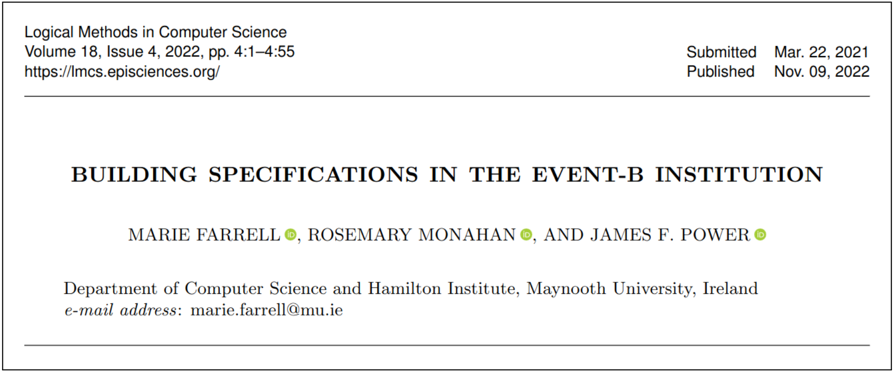
Building Specifications in the Event-B Institution

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`

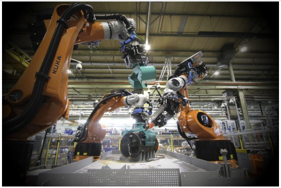
Disclaimer:

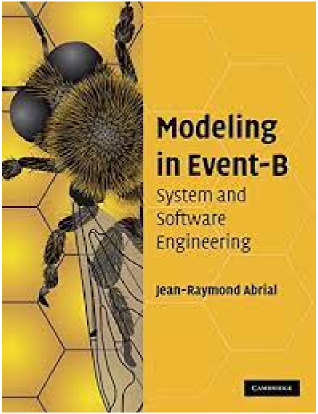
There will be equations and commutative diagrams on these slides but I will only superficially explain them. All of the details and proofs are in the paper.

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**.




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.

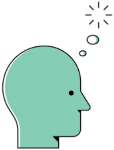
Modularisation

- 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

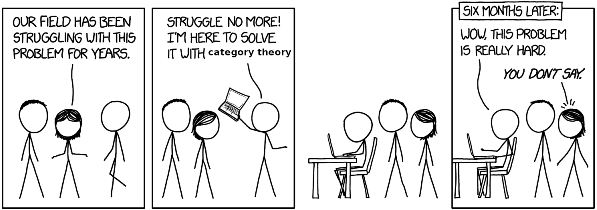
BUILDING SPECIFICATIONS IN THE EVENT-B INSTITUTION

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 $M' \in \mid$ **Mod**$(\Sigma') \mid$.

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

$$
\begin{tikzcd}
\Sigma_1 \arrow[d, "\sigma"] \\
\Sigma_2
\end{tikzcd}
$$

$$
\begin{tikzcd}
\mathbf{Sen}(\Sigma_1) \arrow[r, "\rho^{Sen}_{\Sigma_1}"] \arrow[d, "\mathbf{Sen}(\sigma)"'] & \mathbf{Sen}'(\rho^{Sign}(\Sigma_1)) \arrow[d, "\mathbf{Sen}'(\rho^{Sign}(\sigma))"] \\
\mathbf{Sen}(\Sigma_2) \arrow[r, "\rho^{Sen}_{\Sigma_2}"'] & \mathbf{Sen}'(\rho^{Sign}(\Sigma_2))
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathbf{Mod}'(\rho^{Sign}(\Sigma_2)) \arrow[r, "\rho^{Mod}_{\Sigma_2}"] \arrow[d, "\mathbf{Mod}'(\rho^{Sign}(\sigma))"'] & \mathbf{Mod}(\Sigma_2) \arrow[d, "\mathbf{Mod}(\sigma)"] \\
\mathbf{Mod}'(\rho^{Sign}(\Sigma_2)) \arrow[r, "\rho^{Mod}_{\Sigma_1}"'] & \mathbf{Mod}(\Sigma_1)
\end{tikzcd}
$$

“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 $f_A : |A|_{s_1} \times \cdots \times |A|_{s_n} \to |A|_s$ 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 $f_A : |A|_{s_1} \times \cdots \times |A|_{s_n} \to |A|_s$ 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 $p \in \Pi_{s_1 \cdots s_n}$.

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**

- $\texttt{F.and}$: Σ-*formula** $\rightarrow$ Σ-*formula*
- $\texttt{F.lt}$: Σ-*term* $\times$ Σ-*term* $\rightarrow$ Σ-*formula*
- $\texttt{F.leq}$: Σ-*term* $\times$ Σ-*term* $\rightarrow$ Σ-*formula*
- $\texttt{F.exists}$: *VarName** $\times$ Σ-*formula* $\rightarrow$ Σ-*formula*
- $\texttt{F.}\iota$: *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

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

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

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

$E = \{(\mathtt{Init} \mapsto \mathtt{ordinary}), (\mathtt{ML\_in} \mapsto \mathtt{ordinary}), (\mathtt{ML\_out} \mapsto \mathtt{ordinary})\},$

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

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
```

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

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

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

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

$\mathcal{EVT}$ - 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 $\sigma : \Sigma_{SP_1} \to \Sigma'$. $Sig[SP_1 \texttt{ with } \sigma] = \Sigma'$ $Mod[SP_1 \texttt{ with } \sigma] = \{M' \in \lvert\mathbf{Mod}(\Sigma')\rvert \mid M'\rvert_\sigma \in Mod[SP_1]\}$.
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 $\sigma : \Sigma \to \Sigma_{SP_1}$. $Sig[SP_1 \texttt{ hide via } \sigma] = \Sigma$ $Mod[SP_1 \texttt{ hide via } \sigma] = \{ M\rvert_\sigma \mid M \in Mod[SP_1]\}$.

We Need Pushouts for Specification Building

Given two $\mathcal{EVT}$-signature morphisms $\sigma_1 : \Sigma \to \Sigma_1$ and $\sigma_2 : \Sigma \to \Sigma_2$ 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 $u : \Sigma' \to \Sigma''$ such that the diagram below commutes.

$$
\begin{tikzcd}
 & \Sigma \arrow[dl, "\sigma_1"'] \arrow[dr, "\sigma_2"] & \\
\Sigma_1 \arrow[dr, "\sigma_1'"] \arrow[ddr, "\sigma_1''"'] & & \Sigma_2 \arrow[dl, "\sigma_2'"'] \arrow[ddl, "\sigma_2''"] \\
 & \Sigma' \arrow[d, dashed, "u"] & \\
 & \Sigma'' &
\end{tikzcd}
$$

Institutions Must Preserve Amalgamation for Specification Building

$$\begin{tikzcd}[column sep=small]
 & R'.\sigma'(e) = \{\sigma'(x_i) \mapsto h'(a_i)\} \in R' \arrow[dl, "Mod(\sigma'_1)"'] \arrow[dr, "Mod(\sigma'_2)"] & \\
R_1.\sigma_1(e) = \{\{\sigma_1(x_i) \mapsto h_1(a_i)\} \in R_1 \arrow[dr, "Mod(\sigma_1)"'] & & R_2.\sigma_2(e) = \{\sigma_2(x_i) \mapsto h_2(a_i)\} \in R_2 \arrow[dl, "Mod(\sigma_2)"] \\
 & R.e = \{x_i \mapsto a_i\} \in R &
\end{tikzcd}$$

...proofs are in the paper

Building Specifications in the Event-B Institution

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

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

$$\textit{where } \Sigma = \xi[\![m]\!].$$

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

#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

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

$$\mathbb{R}_\Sigma \left[\!\!\left[\begin{array}{l}\texttt{event } e_c\\ \texttt{status } s\\ \texttt{refines } e_1, \ldots, e_n\\ \textit{ebody}\\ \texttt{end}\end{array}\right]\!\!\right] [\![a]\!]\ \xi \ = \begin{array}{l}\texttt{let}\\ \quad \Sigma_a = \xi[\![a]\!], \qquad \textit{\#Signature of abstract machine}\\ \quad \textit{\# Use } \Sigma_h,\ \sigma_h \textit{ to select only refined events:}\\ \quad \Sigma_h = \langle \Sigma_a.S,\ \Sigma_a.\Omega,\ \Sigma_a.\Pi,\\ \qquad\qquad \{[\![e_1]\!], \ldots, [\![e_n]\!]\} \lhd \Sigma_a.E,\ \Sigma_a.V\rangle,\\ \quad \sigma_h : \Sigma_h \hookrightarrow \Sigma_a,\\ \quad \textit{\# Use } \sigma_m \textit{ to reassign refined event sentences to } e_c\textit{:}\\ \quad \sigma_m : \Sigma_h \rightarrow \Sigma\\ \quad \sigma_m = \left\langle \begin{array}{l}\Sigma_h.S \hookrightarrow \Sigma.S,\ \Sigma_h.\Omega \hookrightarrow \Sigma.\Omega, \Sigma_h.\Pi \hookrightarrow \Sigma.\Pi,\\ \Sigma_h.E \mapsto \{[\![e_c]\!]\} \lhd \Sigma.E,\\ \Sigma_h.V \hookrightarrow \Sigma.V\end{array}\right\rangle\\ \texttt{in}\\ \quad ([\![a]\!]\ \texttt{hide via}\ \sigma_h)\ \texttt{with}\ \sigma_m.\end{array}$$

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

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

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

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

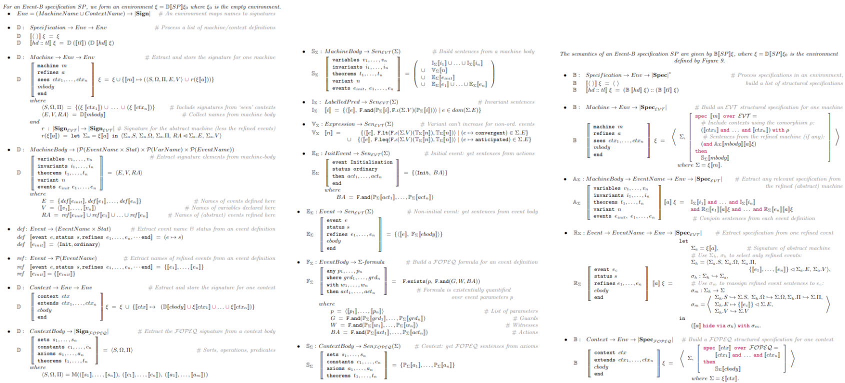
$\mathbb{E}_\Sigma$: *InitEvent* $\rightarrow$ $Sen_{\mathcal{EVT}}(\Sigma)$ #Initial event: get sentences from actions

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

where

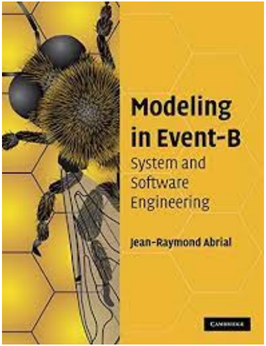
$$BA \ = \ \texttt{F.and}(\mathbb{P}_\Sigma[\![act_1]\!], \ldots, \mathbb{P}_\Sigma[\![act_n]\!])$$

Building Specifications in the Event-B Institution



...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

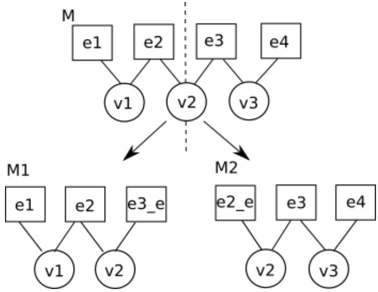
```
spec  DATAM0 =
  CD then
    ops n:ℕ
    . n ≤ d
      EVENTS
      Initialisation
        thenAct n := 0
end

spec  DATAM1 =
  DATAM0 then
    ops a, b, c : ℕ
    . n = a + b + c
      a = 0 ∨ c = 0
    variant 2∗a+b
    EVENTS
      Initialisation
        thenAct   a := 0
                  b := 0
                  c := 0
end
```

```
spec  INOUT =
  ops v1, v2 : ℕ
  EVENTS
    Event out ≙ ordinary
      when v1 = 0
      thenAct v2 := v2 + 1
    Event in ≙ ordinary
      when v1 > 0
      thenAct v1 := v1 + 1
end

spec  M1 =
  DATAM1 and M0 and
  INOUT with {⟨out, ordinary⟩ ↦ ⟨ML_out, ordinary⟩,
               ⟨in, ordinary⟩ ↦ ⟨ML_in, ordinary⟩,
                v1 ↦ c, v2 ↦ a} and
  INOUT with {⟨out, ordinary⟩ ↦ ⟨IL_out, convergent⟩,
             ⟨in, ordinary⟩ ↦ ⟨IL_in, convergent⟩,
              v1 ↦ a, v2 ↦ c}
   then
     EVENTS
       Event ML_out ≙ ordinary
         when a + b < d
       Event IL_in ≙ convergent
         thenAct b:= b + 1
       Event IL_out ≙ convergent
         when 0 < b
         thenAct b := b + 1
end
```

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.

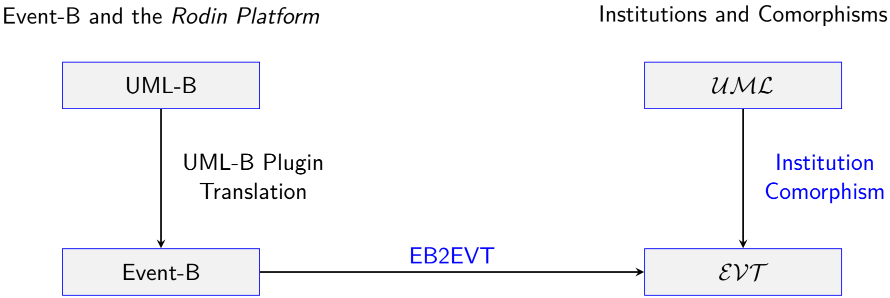
Conclusions and Future Work

- 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.

Not in the paper...



Connecting Institutions: Institution Comorphism

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

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

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

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

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/