

Overview

Formality, Flexibility  
or Both

Testing

Horizontal + Vertical

Horizontal (E,X,O)

Vertical

Final

Event B

Observations

Summary

Protocol stacks

Comparisons

Subset Morphism

Divergence

Z+B

Example

# Meta B

## A flexible semantics for Event B

S. Reeves    D. Streader

Department of Computer Science  
University of Waikato

Newcastle 2009

Overview

Formality, Flexibility  
or Both

Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

Final

Event B  
Observations  
Summary  
Protocol stacks

Comparisons

Subset Morphism  
Divergence  
Z+B

Example

# Outline of Talk

## 1 Overview

Formality, Flexibility or Both

## 2 Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## 3 Final

Event B  
Observations  
Summary  
Protocol stacks

## 4 Comparisons

Subset Morphism  
Divergence  
Z+B

## 5 Example

## Overview

- Refinement formalises a step in the stepwise development of a specification into code.
- Semantics aid our informal interpretation of formal statements
- There are many styles of semantics
  - Operational - Relational or LTS
  - Denotational
  - Axiomatic
  - Testing
- The testing semantics try to directly formalise how entities interact and are observed
- Use more than one equivalent semantics

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

## Background

- The semantics of an Event B machine (Rodin D7) is a set of named relations
- With this semantics both individual operations and machines can be interpreted in many ways. This flexibility is important
- Thus Event B semantics formalises part of the semantic story; the other informal part is open to interpretation
- The balance between flexibility and formality is fixed
- Selecting the balance point should be a design step

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

## Proposal

- The Event B tool kit generates proof obligations that need to be discharged to establish that one machine is a forward simulation (and therefore a refinement) of another
- To increase flexibility, allow formal development steps other than forward simulation
  - Stepwise increase in formality
  - Be flexible and formal about types of events used
  - Stepwise increase in features considered: first specify only the correct behaviour and later add the error behaviour (example to follow)

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

## Correct Machine

MACHINE  
VARIABLES

*Set\_Machine*

$xx$

INVARIANT

$xx \subseteq \text{NAT}$

INITIALISATION

$xx := \emptyset$

OPERATIONS

$\text{AddElem}(new) \triangleq$

IF  $new \in \text{NAT}$  THEN  $xx := xx \cup \{new\}$  END;

END

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

# Machine with Error behaviour

REFINEMENT	$Set\_Machine\_R$
REFINES	$Set\_Machine$
VARIABLES	$xx\_seq$
INVARIANT	$xx\_seq \in iseq(NAT) \wedge$ $xx = ran(xx\_seq) \wedge size(xx\_seq) \leq 10$
<b>FRAME</b>	<b><math>resp = TRUE</math></b>
INITIALISATION	$xx\_seq := \langle \rangle$
OPERATIONS	$resp \leftarrow AddElem(new) \triangleq$ IF $new \in NAT$ THEN IF $size(xx\_seq) < 10$ THEN IF $new \notin ran(xx\_seq)$ THEN $xx\_seq := xx\_seq \frown [new] \parallel resp := TRUE$ ELSE $resp := TRUE$ END ELSE $resp := FALSE$ END END; END

Meta B  
A flexible  
semantics for  
Event B

Reeves,  
Streader

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Machine with Error behaviour

Meta B  
A flexible  
semantics for  
Event B

Reeves,  
Streader

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

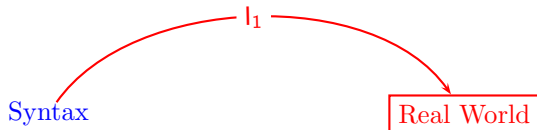
## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

```
REFINEMENT      Set_Machine_R
REFINES         Set_Machine
VARIABLES      xx_seq
INVARIANT       $xx\_seq \in \text{iseq}(\text{NAT}) \wedge$ 
                 $xx = \text{ran}(xx\_seq) \wedge \text{size}(xx\_seq) \leq 10$ 
GALOIS      F1 F2
INITIALISATION  $xx\_seq := \langle \rangle$ 
OPERATIONS
   $resp \leftarrow \text{AddElem}(new) \triangleq$ 
    IF  $new \in \text{NAT}$  THEN
      IF  $\text{size}(xx\_seq) < 10$  THEN
        IF  $new \notin \text{ran}(xx\_seq)$  THEN
           $xx\_seq := xx\_seq \frown [new] \parallel resp := \text{TRUE}$ 
        ELSE  $resp := \text{TRUE}$  END
      ELSE  $resp := \text{FALSE}$  END
    END;
END
```

# From formal Specification to informal world



- To be of **use** a formal statement must be **interpreted**
- Any set **Act** and **binary operator** is a valid interpretation (in the range of  $I_1$ )
- Semantics are designed to be closer to the real world

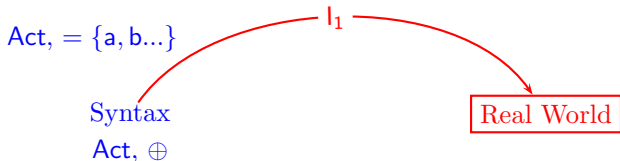
$$[a \oplus b] \triangleq [a][\oplus][b] \quad [\oplus] \triangleq \cup \quad \forall a \in Act. [a] \subseteq S \times S$$

- The meaning given to a term is part **formal** part **informal**
- Semantics restricts the interpretations of terms:

$I_2[t]$  is a **subset** of  $I_1 t$

$$[a \oplus a] = [a]$$

# From formal Specification to informal world



- To be of **use** a formal statement must be **interpreted**
- Any set **Act** and **binary operator** is a valid interpretation (in the range of  $I_1$ )
- Semantics are designed to be closer to the real world

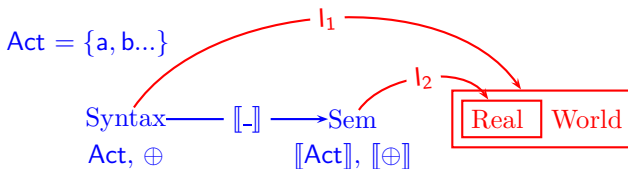
$$[a \oplus b] \triangleq [a][\oplus][b] \quad [\oplus] \triangleq \cup \quad \forall a \in Act. [a] \subseteq S \times S$$

- The meaning given to a term is part **formal** part **informal**
- Semantics restricts the interpretations of terms:

$I_2[t]$  is a **subset** of  $I_1 t$

$$[a \oplus a] = [a]$$

# From formal Specification to informal world



- To be of **use** a formal statement must be **interpreted**
- Any set **Act** and **binary operator** is a valid interpretation (in the range of  $I_1$ )

- Semantics are designed to be closer to the real world

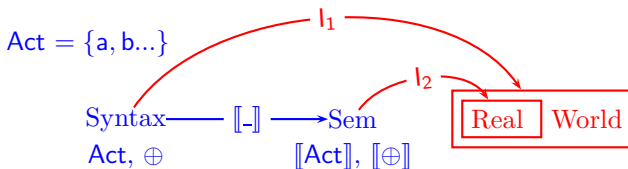
$$[[a \oplus b]] \triangleq [[a]][\oplus][[b]] \quad [[\oplus]] \triangleq \cup \quad \forall a \in \text{Act}. [[a]] \subseteq \mathcal{S} \times \mathcal{S}$$

- The meaning given to a term is part **formal** part **informal**
- Semantics restricts the interpretations of terms:

$I_2[[t]]$  is a **subset** of  $I_1 t$

$$[[a \oplus a]] = [[a]]$$

# From formal Specification to informal world



- To be of **use** a formal statement must be **interpreted**
- Any set **Act** and **binary operator** is a valid interpretation (in the range of  $I_1$ )

- Semantics are designed to be closer to the real world

$$[[a \oplus b]] \triangleq [[a]][\oplus][[b]] \quad [[\oplus]] \triangleq \cup \quad \forall a \in Act. [[a]] \subseteq S \times S$$

- The meaning given to a term is part **formal** part **informal**
- Semantics restricts the interpretations of terms:

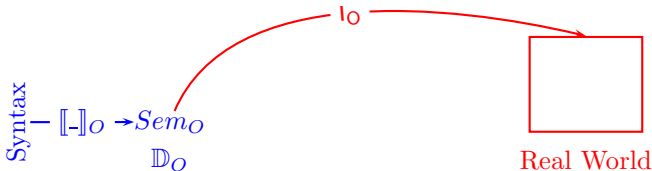
$I_2[[t]]$  is a **subset** of  $I_1 t$

$$[[a \oplus a]] = [[a]]$$

## Stepwise development

- Step One define the Syntax
- Step Two define a Semantics,  $\mathbb{D}_O$  and mapping  $\llbracket - \rrbracket_O$ 
  - $\llbracket - \rrbracket_O$  defines only part of the semantic story
  - Closer to our intuitions
  - Syntax is open to fewer interpretations
- Step Three repeat step Two
- Examples Z & B theory + practice

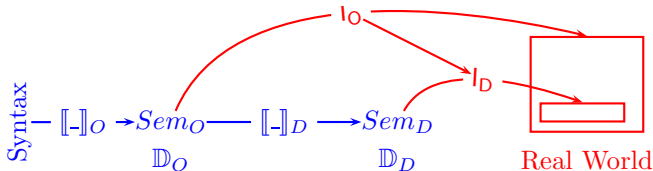
$\mathbb{D}_O$	partial relations	Correct
$\mathbb{D}_D$	lifted total relations	Correct + Error



## Stepwise development

- Step One define the Syntax
- Step Two define a Semantics,  $\mathbb{D}_O$  and mapping  $[\_ ]_O$ 
  - $[\_ ]_O$  defines only part of the semantic story
  - Closer to our intuitions
  - Syntax is open to fewer interpretations
- Step Three repeat step Two
- Examples Z & B theory + practice

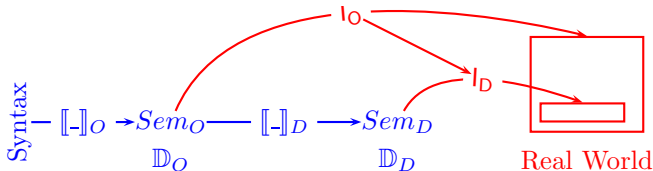
$\mathbb{D}_O$	partial relations	Correct
$\mathbb{D}_D$	lifted total relations	Correct + Error



## Stepwise development

- Step One define the Syntax
- Step Two define a Semantics,  $\mathbb{D}_O$  and mapping  $[-]_O$ 
  - $[-]_O$  defines only part of the semantic story
  - Closer to our intuitions
  - Syntax is open to fewer interpretations
- Step Three repeat step Two
- Examples Z & B theory + practice

$\mathbb{D}_O$	partial relations	Correct
$\mathbb{D}_D$	lifted total relations	Correct + Error



## Formality, Flexibility or Both

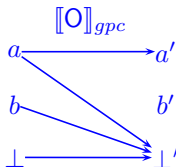
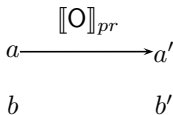
- Formal semantics fixes the meaning given to terms
- More formality; Less flexibility
- Example partial relations have many valid interpretations

$$\begin{array}{ccc} & \llbracket O \rrbracket_{pr} & \\ a & \longrightarrow & a' \\ b & & b' \end{array}$$

- From state  $a$   $O$  terminates and terminates in  $a'$
- From state  $a$  + undefined outside of precondition  $O$  terminates and terminates in  $a'$

## Formality, Flexibility or Both

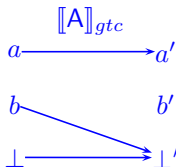
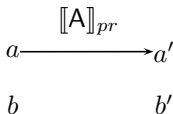
- Formal semantics fixes the meaning given to terms
- More formality; Less flexibility
- Example partial relations have many valid interpretations



- From state  $a$  If  $O$  terminates it terminates in  $a'$
- From state  $a$  + undefined outside of precondition  $O$  terminates and terminates in  $a'$

## Formality, Flexibility or Both

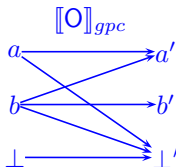
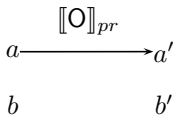
- Formal semantics fixes the meaning given to terms
- More formality; Less flexibility
- Example partial relations have many valid interpretations



- From state  $a$   $\circ$  terminates and terminates in  $a'$
- From state  $a$  + undefined outside of precondition  $\circ$  terminates and terminates in  $a'$

## Formality, Flexibility or Both

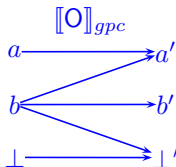
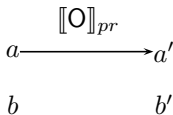
- Formal semantics fixes the meaning given to terms
- More formality; Less flexibility
- Example partial relations have many valid interpretations



- From state  $a$   $O$  terminates and terminates in  $a'$
- From state  $a$  + undefined outside of precondition  
If  $O$  terminates it terminates in  $a'$

## Formality, Flexibility or Both

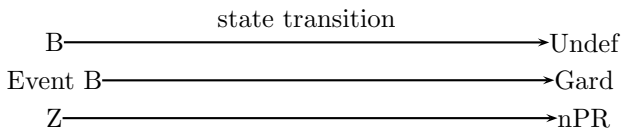
- Formal semantics fixes the meaning given to terms
- More formality; Less flexibility
- Example partial relations have many valid interpretations



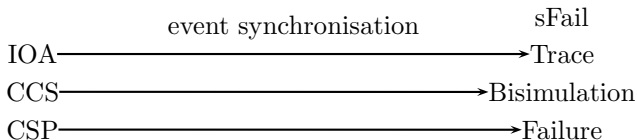
- From state  $a$   $O$  terminates and terminates in  $a'$
- From state  $a$  + undefined outside of precondition  $O$  terminates and terminates in  $a'$

# A view of Formal Models

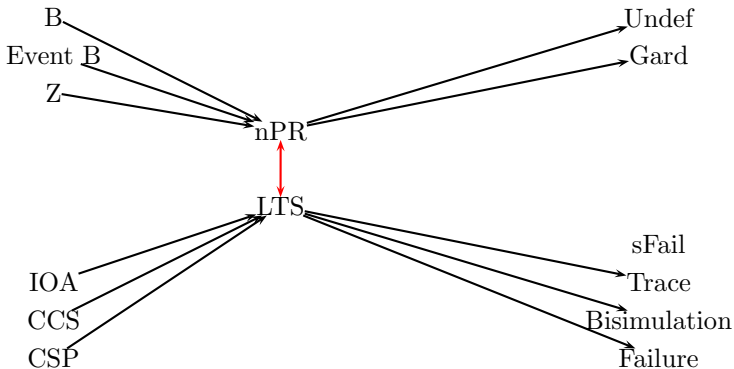
+ Operation



Each special theory has fixed interpretation!



## Common Semantics



### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

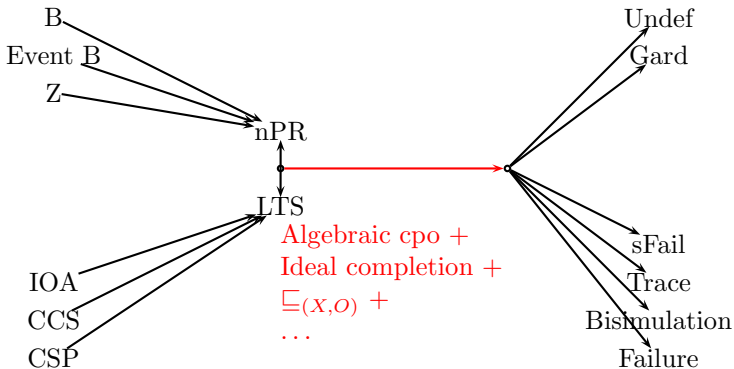
Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

# Common Semantics



## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical

Horizontal (E,X,O)

Vertical

## Final

Event B

Observations

Summary

Protocol stacks

## Comparisons

Subset Morphism

Divergence

Z+B

## Example

# Horizontal + Vertical

## 1 General refinement and known examples (2003)

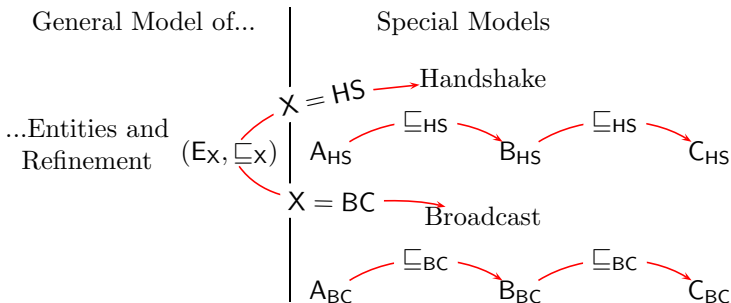
- Refinement is reduction of non-determinism
- Implementer is free to build any refinement of specification

## 2 Vertical refinement (2007)

- More powerful than horizontal refinement
- Reinterprets specification
- Formalises **client's design decision**

## Horizontal + Vertical

- 1 Horizontal refinement and known examples (2003)
- 2 Vertical refinement (2007)



## Refinement in General

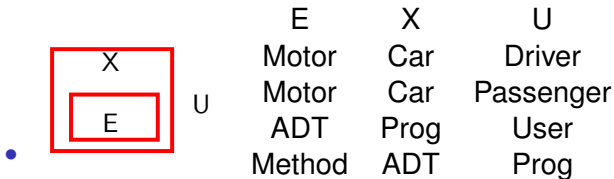
- The concrete entity C is a refinement of an abstract entity A when no user of A could observe if they were given C in place of A.
- To formalise this we use:
  - Entities A and C from some set  $\mathbb{E}$
  - Contexts  $x \in \Xi$  in which we place the entities  $[A]_x \in \mathbb{E}$
  - A user formalised by an observation function  $O : \mathbb{E} \rightarrow \mathbb{O}$
  - **Two interfaces**

E	X	U
Motor	Car	Driver
Motor	Car	Passenger
ADT	Prog	User
Method	ADT	Prog

- $\mathbb{O}$  is the set of complete traces

## Refinement in General

- The concrete entity C is a refinement of an abstract entity A when no user of A could observe if they were given C in place of A.
- To formalise this we use:
  - Entities A and C from some set  $\mathbb{E}$
  - Contexts  $x \in \Xi$  in which we place the entities  $[A]_x \in \mathbb{E}$
  - A user formalised by an observation function  $O : \mathbb{E} \rightarrow \mathbb{O}$
  - Two interfaces**



- $\mathbb{O}$  is the set of complete traces

## Refinement in General



$$A \sqsubseteq_{\Xi, O} C \triangleq \forall x \in \Xi. O([C]_x) \subseteq O([A]_x)$$

- The denotational semantics of entities is a relation:

$$[[A]]_{\Xi, O} \triangleq \{(x, o) \mid x \in \Xi, o \in O([A]_x)\}$$

- Refinement is subset or implication in a logical theory:  
A theory T is pair  $(E_T, \sqsubseteq_T)$
- For D a deterministic entity  $[[D]]_{\Xi, O}$  is a function
- General Refinement is made concrete by fixing:

$\mathbb{E}, \Xi$  and  $O$

## Refinement in General



$$A \sqsubseteq_{\Xi, O} C \triangleq \forall x \in \Xi. O([C]_x) \subseteq O([A]_x)$$

- The denotational semantics of entities is a relation:

$$[[A]]_{\Xi, O} \triangleq \{(x, o) \mid x \in \Xi, o \in O([A]_x)\}$$

- Refinement is subset or implication in a logical theory:  
A theory T is pair  $(\mathbb{E}_T, \sqsubseteq_T)$
- For D a deterministic entity  $[[D]]_{\Xi, O}$  is a function
- General Refinement is made concrete by fixing:

$$\mathbb{E}, \Xi \text{ and } O$$

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical

Horizontal (E,X,O)

Vertical

### Final

Event B

Observations

Summary

Protocol stacks

### Comparisons

Subset Morphism

Divergence

Z+B

### Example

## Interface types

- **Transactional interface**
  - Observation occurs at initialisation and at finalisation if termination is successful.
  - Examples - a method of an object and passenger in car
- **Interactive interface**
  - Observation can occur at many points throughout the execution
  - Observations can be made prior to termination and even prior to non-termination
  - Examples - a coffee machine and the driver of a car

# Horizontal + Vertical

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical

Horizontal (E,X,O)

Vertical

## Final

Event B

Observations

Summary

Protocol stacks

## Comparisons

Subset Morphism

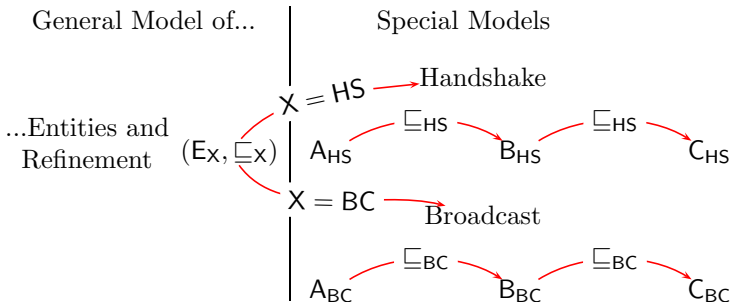
Divergence

Z+B

## Example

### 1 Horizontal refinement (2003)

### 2 Vertical refinement (2007)



# Horizontal + Vertical

- 1 Horizontal refinement (2003)
- 2 **Vertical refinement (2007)**

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical

Horizontal (E,X,O)

Vertical

## Final

Event B

Observations

Summary

Protocol stacks

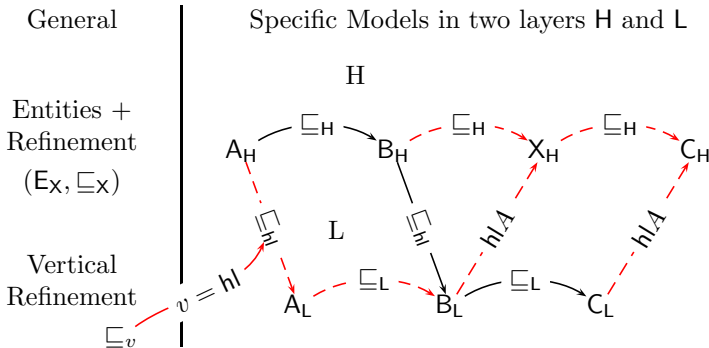
## Comparisons

Subset Morphism

Divergence

Z+B

## Example



## Theory morphism

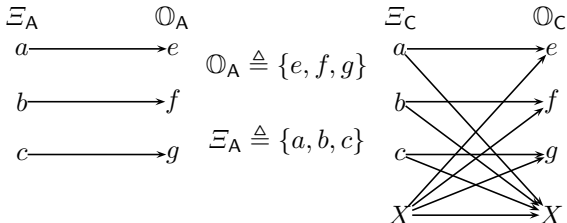
- Different kinds of entities have different theories
- Galois connections interpret one theory in another and vice versa
- A Galois connection is a pair of functions:  $\llbracket \_ \rrbracket_V^{HL}$  one from the high- to low-level theory the other  $\nu A^{HL} \_$  from the low- to high-level theory

$$\llbracket X_H \rrbracket_V^{HL} \sqsubseteq_{\Xi_L, O_L} Y_L \Leftrightarrow X_H \sqsubseteq_{\Xi_H, O_H} \nu A^{HL}(Y_L)$$

- We refer to our Galois connections as vertical refinements as they provide a **guarantee** of the behaviour of the high-level entities in term of the behaviour of the low-level entity and vice versa

## Subset morphism

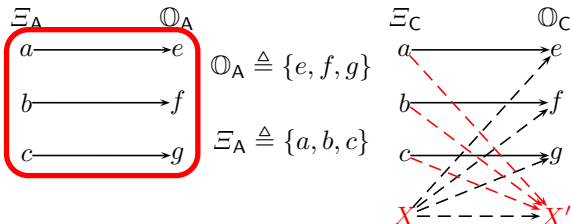
- Subset theories are related by a Galois connection
- If  $\Xi_A \times \mathbb{O}_A \subseteq \Xi_C \times \mathbb{O}_C$  then the embedding and projection functions are a Galois connection
- Example  $A \sqsubseteq_{sub} C$



- Silent outside of frame
- $X$  is an unconsidered context and  $X'$  an unconsidered observation

## Subset morphism

- Subset theories are related by a Galois connection
- If  $\Xi_A \times \mathbb{O}_A \subseteq \Xi_C \times \mathbb{O}_C$  then the embedding and projection functions are a Galois connection
- Example  $A \sqsubseteq_{sub} C$



- Silent outside of **frame**
- **X** is an unconsidered context and **X'** an unconsidered observation

## Event B

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

#### Event B

Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

- Relational semantics is open to many interpretations but formalises only part of the meaning
- Discharging proof obligations for Galois connections validates a vertical refinement
- A vertical refinement step can exchange flexibility for formality
- Backward simulation must be customised for particular interpretation
- Gaining confidence in a formal general theory that specialises to known theories is best achieved by machine checking the general theory
- Do not “Trust me I’m a Doctor”

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B

## Observations

Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Observations

- Singleton failure semantics is different from data refinement because each uses a different style of context/user interface
- Divergence only masks what you want to see when the context/user interface is transactional, not when it is interactive
- Dropping the stimulus-response nature of event synchronisation changes how determinism is modelled

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
**Summary**  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Summary

- General refinement factors out a common theory
- Special concrete examples include
  - Handshake processes
  - Broadcast processes
  - ADT
  - Operations
- Vertical refinement allows
  - layered development
  - reinterpretation
  - silent outside of frame
  - some retrenchments

## Protocol stacks

- Each layer in a protocol stack is implemented on the layer beneath
- Both valid entities and contexts are defined
- A broadcast layer has unblockable send events
- This can be formalised in two ways
  - $BC_{CSP}$  with processes that always listen and CSP style parallel composition
  - $BC_V$  with a definition of parallel composition to formalise non-blocking send
- There is a Galois connection between the two
- but not between  $BC_{CSP}$  and CSP style processes

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

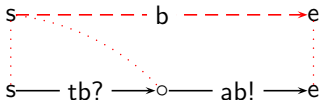
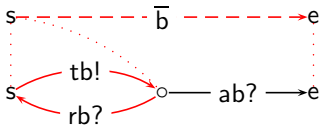
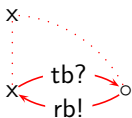
Subset Morphism  
Divergence  
Z+B

### Example

## On a Broadcast Layer

- Can we implement processes with handshake style events?
- Using broadcast send  $b!$  and broadcast listen  $b?$

$b \notin Ready(x)$



### Overview

Formality, Flexibility or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

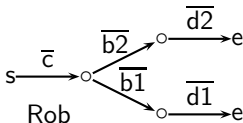
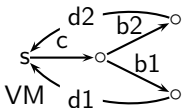
### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

## Active actions and determinism

- Are active actions the same as passive actions?



- Is Rob deterministic?

## Robot Problems

- We have a problem with branching active actions

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

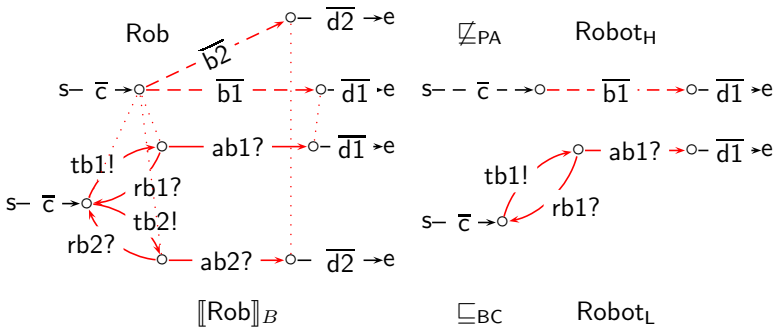
### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

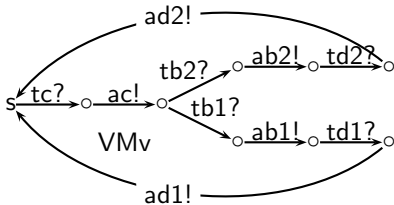
Subset Morphism  
Divergence  
Z+B

### Example



## Mixing action types

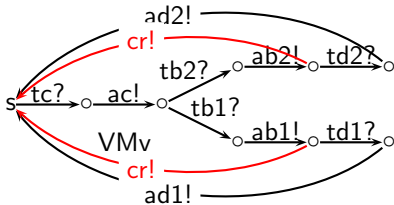
- Vending machine on broadcast layer



- Error action **coin return** can not be blocked!
- Simplify
- Refiine for two cups

## Mixing action types

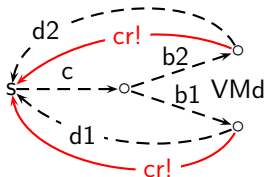
- Vending machine on broadcast layer



- Error action **coin return** can not be blocked!
- Simplify
- Refiine for two cups

## Mixing action types

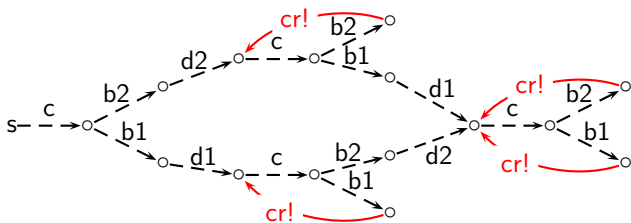
- Vending machine on broadcast layer



- Error action **coin return** can not be blocked!
- Simplify
- Refiine for two cups

## Mixing action types

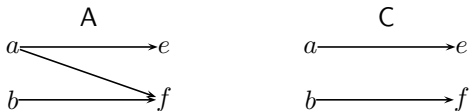
- Vending machine on broadcast layer



- Error action **coin return** can not be blocked!
- Simplify
- Refiine for two cups

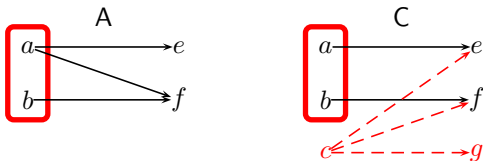
# What is Refinement?

- A **strict** view of reduction of non-determinism
- Undefined outside of **domain**  
The abstract specification does not consider behaviour outside of the domain **c**
- Silent outside of **frame**  
In addition the abstract specification does not consider when new observations **g** can be made
- Note the introduction of non-determinism



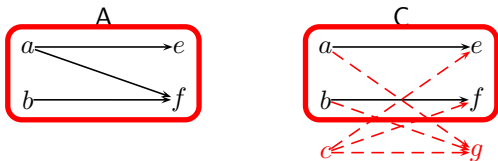
# What is Refinement?

- A **strict** view of reduction of non-determinism
- Undefined outside of **domain**  
The abstract specification does not consider behaviour outside of the domain **c**
- Silent outside of **frame**  
In addition the abstract specification does not consider when new observations **g** can be made
- Note the introduction of non-determinism



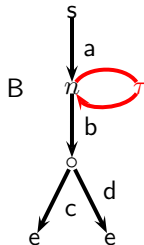
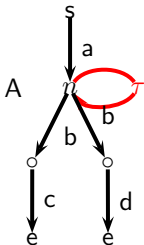
# What is Refinement?

- A **strict** view of reduction of non-determinism
- Undefined outside of **domain**  
The abstract specification does not consider behaviour outside of the domain **c**
- Silent outside of **frame**  
In addition the abstract specification does not consider when new observations **g** can be made
- Note the introduction of non-determinism



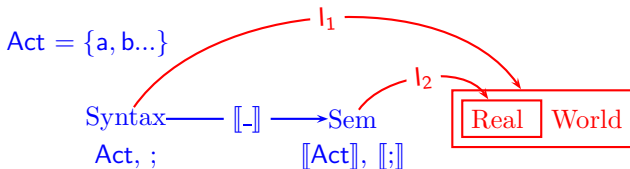
## Divergence

- The Chaotic interpretation of divergence  
“typically masks much L behaviour we should really  
want to see” (Roscoe)



- I-I** tests are a less masking semantics than **I-T** tests
- $A \sqsubseteq_{II} B$  and  $B \not\sqsubseteq_{II} A$  but  $A =_{IT} B$
- Consider the test **abc**

# B = Z + Formal methodology.



- Both Z and (Event) B use similar semantics But!

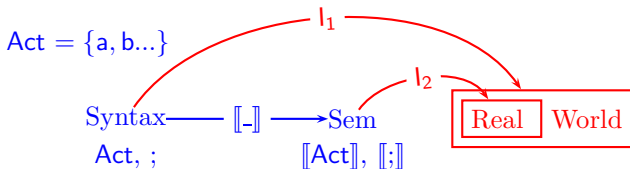
$$[a;b] \triangleq [a][;][b] \quad [;] \triangleq ; \quad \forall a \in Act. [a] \subseteq S \times S$$

- Theoretical computer science can keep to **formality**
- Engineers must take account of the **informal**

$l_1(a;b)$  is equal to  $l_1(a)l_1(; )l_1(b)$   
 $l_1(; )$  is the sequential composition of operations

- Z Spivey **care needed** with sequential composition.

# B = Z + Formal methodology.



- Both Z and (Event) B use similar semantics But!

$$[a;b] \triangleq [a][;][b] \quad [;] \triangleq ; \quad \forall a \in Act. [a] \subseteq S \times S$$

- Theoretical computer science can keep to **formality**
- Engineers must take account of the **informal**

$I_1(a;b)$  is equal to  $I_1(a)I_1(;)I_1(b)$   
 $I_1(;)$  is the sequential composition of operations

- Z Spivey **care needed** with sequential composition.

## First specify correct behaviour later specify errors

$$\text{Set}_C = \text{Set}_A + \text{errors}$$

### Correct Behaviour

$\text{Set}_A$

$$\text{State}_A = s : \mathbb{PN}$$

$$\text{Put}_A(n?)$$

If  $n? \notin s$  then

$$s' = s \cup \{n?\}$$

$$\text{Get}_A(n!)$$

If  $n! \in s$  then

$$s' = s \setminus \{n!\}$$

$$\text{State}_C = t : \mathbb{PN} \cup \{X\}$$

$$\text{Put}_C(n?)$$

If  $t \neq X \wedge \#t < 3 \wedge n? \notin s$  then

$$s' = s \cup \{n?\} \text{ elseif}$$

$t \neq X \wedge n? \notin s \wedge \#t = 3$  then

$$t = X$$

$$\text{Get}_C(n!)$$

If  $t \neq X \wedge n! \in s$  then

$$s' = s \setminus \{n!\}$$

$$\text{Reset}_C$$

If  $t = X$  then  $t' = \emptyset$

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

First specify correct behaviour later specify errors

$$\text{Set}_C = \text{Set}_A + \text{errors}$$

## Correct Behaviour

$\text{Set}_A$

$$\text{State}_A = s : \mathbb{PN}$$

$$\text{Put}_A(n?)$$

If  $n? \notin s$  then

$$s' = s \cup \{n?\}$$

$$\text{Get}_A(n!)$$

If  $n! \in s$  then

$$s' = s \setminus \{n!\}$$

$$\text{State}_C = t : \mathbb{PN} \cup \{X\}$$

$$\text{Put}_C(n?)$$

If  $t \neq X \wedge \#t < 3 \wedge n? \notin s$  then

$$s' = s \cup \{n?\} \text{ elseif}$$

$t \neq X \wedge n? \notin s \wedge \#t = 3$  then

$$t = X$$

$$\text{Get}_C(n!)$$

If  $t \neq X \wedge n! \in s$  then

$$s' = s \setminus \{n!\}$$

**Reset<sub>C</sub>**

If  $t = X$  then  $t' = \emptyset$

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

First specify correct behaviour later **specify errors**

$$\text{Set}_C = \text{Set}_A + \text{errors}$$

## Correct Behaviour

$\text{Set}_A$

$$\text{State}_A = s : \mathbb{PN}$$

$$\text{Put}_A(n?)$$

If  $n? \notin s$  then

$$s' = s \cup \{n?\}$$

$$\text{Get}_A(n!)$$

If  $n! \in s$  then

$$s' = s \setminus \{n!\}$$

$$\text{State}_C = t : \mathbb{PN} \cup \{X\}$$

$$\text{Put}_C(n?)$$

If  $t \neq X \wedge \#t < 3 \wedge n? \notin s$  then

$s' = s \cup \{n?\}$  **elseif**

$t \neq X \wedge n? \notin s \wedge \#t = 3$  then

$t = X$

$$\text{Get}_C(n!)$$

If  $t \neq X \wedge n! \in s$  then

$$s' = s \setminus \{n!\}$$

**Reset<sub>C</sub>**

If  $t = X$  then  $t' = \emptyset$

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Is $\text{Set}_A \sqsubseteq \text{Set}_C$ ?

- In most formalisms NO!
  - In most formalisms  $\text{Set}_A$ , not  $\text{Set}_C$ , allows  $\text{Put}^\infty$
- If we want to develop software like this how are we interpreting the specification?
- What is  $\text{Set}_A$  formally specifying?
  - We interpret  $\text{Set}_A$  as only guaranteeing behaviour when no errors occur. Thus when:
    1. state remains in  $\text{State}_A$  and
    2. only operations in  $\text{Set}_A$  are called/observed.
- Flexible refinement will formalise such development steps

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Is $\text{Set}_A \sqsubseteq \text{Set}_C$ ?

- In most formalisms NO!
  - In most formalisms  $\text{Set}_A$ , not  $\text{Set}_C$ , allows  $\text{Put}^\infty$
- If we want to develop software like this how are we interpreting the specification?
- What is  $\text{Set}_A$  formally specifying?
  - We interpret  $\text{Set}_A$  as only guaranteeing behaviour when no errors occur. Thus when:
    1. state remains in  $\text{State}_A$  and
    2. only operations in  $\text{Set}_A$  are called/observed.
- Flexible refinement will formalise such development steps

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Is $\text{Set}_A \sqsubseteq \text{Set}_C$ ?

- In most formalisms NO!
  - In most formalisms  $\text{Set}_A$ , not  $\text{Set}_C$ , allows  $\text{Put}^\infty$
- If we want to develop software like this how are we interpreting the specification?
- What is  $\text{Set}_A$  formally specifying?
  - We interpret  $\text{Set}_A$  as only guaranteeing behaviour when no errors occur. Thus when:
    1. state remains in  $\text{State}_A$  and
    2. only operations in  $\text{Set}_A$  are called/observed.
- Flexible refinement will formalise such development steps

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# Is $\text{Set}_A \sqsubseteq \text{Set}_C$ ?

- In most formalisms NO!
  - In most formalisms  $\text{Set}_A$ , not  $\text{Set}_C$ , allows  $\text{Put}^\infty$
- If we want to develop software like this how are we interpreting the specification?
- What is  $\text{Set}_A$  formally specifying?
  - We interpret  $\text{Set}_A$  as only guaranteeing behaviour when no errors occur. Thus when:
    1. state remains in  $\text{State}_A$  and
    2. only operations in  $\text{Set}_A$  are called/observed.
- Flexible refinement will formalise such development steps

## Overview

Formality, Flexibility  
or Both

## Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

## Final

Event B  
Observations  
Summary  
Protocol stacks

## Comparisons

Subset Morphism  
Divergence  
Z+B

## Example

# From $\text{Set}_A$ to $\text{Set}_C$ in three steps

- Step one: state-based subset morphism  $\sqsubseteq_{sub}^{\{X\}}$
- Step two: general refinement
- Step three: event-based subset morphism  $\sqsubseteq_{sub}^{\{\text{Reset}\}}$

$$\text{Set}_A \sqsubseteq_{sub}^{\{X\}} \text{Set}_X \sqsubseteq \text{Set}_B \sqsubseteq_{sub}^{\{\text{Reset}\}} \text{Set}_C$$

Overview

Formality, Flexibility  
or Both

Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

Final

Event B  
Observations  
Summary  
Protocol stacks

Comparisons

Subset Morphism  
Divergence  
Z+B

Example

Step 1 from  $\text{Set}_A$  to  $\text{Set}_X$   $\{X\}$  is outside the frame

$\text{Set}_A$

$\text{Set}_X$

$\text{State}_A = s : \mathbb{PN}$

$\text{Put}_A(n?)$

If  $n? \notin s$  then

$s' = s \cup \{n?\}$

$\text{Get}_A(n!)$

If  $n! \in s$  then

$s' = s \setminus \{n!\}$

$\text{State}_X = s : \mathbb{PN} \cup \{X\}$

$\text{Put}_X(n?)$

If  $s \neq X \wedge n? \notin s$  then

$s' = s \cup \{n?\} \vee s' = X$

$\text{Get}_X(n!)$

If  $s \neq X \wedge n! \in s$  then

$s' = s \setminus \{n!\} \vee s' = X$

## Step 2 from $\text{Set}_X$ to $\text{Set}_B$ General Refinement $\text{Set}_B$

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example

$\text{Set}_X$

$\text{State}_X = s : \mathbb{PN} \cup \{X\}$

$\text{Put}_X(n?)$

If  $s \neq X \wedge n? \notin s$  then

$s' = s \cup \{n?\} \vee s' = X$

$\text{Get}_X(n!)$

If  $s \neq X \wedge n! \in s$  then

$s' = s \setminus \{n!\} \vee s' = X$

$\text{State}_B = t : \mathbb{PN} \cup \{X\}$

$\text{Put}_B(n?)$

If  $t \neq X \wedge n? \notin t \wedge \#t < 3$  then

$t' = t \cup \{n?\}$  **elseif**

$t \neq X \wedge n? \notin t \wedge \#t = 3$  then

$t' = X$

$\text{Get}_B(n!)$

If  $t \neq X \wedge n! \in t$  then

$t' = t \setminus \{n!\}$

## Step 3 from $\text{Set}_B$ to $\text{Set}_C$ **Reset is outside the frame**

$$\text{Set}_C = \text{Set}_B + \text{Reset}_C$$

$$\text{State}_C = t : \mathbb{PN} \cup \{X\}$$

$$\text{Put}_C(n?)$$

If  $t \neq X \wedge \#t < 3 \wedge n? \notin s$  then

$$s' = s \cup \{n?\} \text{ elseif}$$

$t \neq X \wedge n? \notin s \wedge \#t = 3$  then

$$t = X$$

$$\text{Get}_C(n!)$$

If  $t \neq X \wedge n! \in s$  then

$$s' = s \setminus \{n!\}$$

**Reset<sub>C</sub>**

If  $t = X$  then  $t' = \emptyset$

### Overview

Formality, Flexibility  
or Both

### Testing

Horizontal + Vertical  
Horizontal (E,X,O)  
Vertical

### Final

Event B  
Observations  
Summary  
Protocol stacks

### Comparisons

Subset Morphism  
Divergence  
Z+B

### Example