

Formal methods

with out the maths!

David Streader

`dstr@cs.waikato.ac.nz`

Department of Computer Science

University of Waikato

Hamilton, New Zealand

Overview of Formal methods

- Why bother?

Overview of Formal methods

- Why bother?
- What is it?

Overview of Formal methods

- Why bother?
- What is it?
- What are we doing that is so clever?

Overview of Formal methods

- Why bother?
- What is it?
- What are we doing that is so clever?
- Conclusion

Why bother?

- Some FM industry success!

Why bother?

- Some FM industry success!
- NASA, AirBus, Military, Transport, Electronic Money

Why bother?

- Some FM industry success!
- NASA, AirBus, Military, Transport, Electronic Money
- UK Military - For reliable systems FM is cost effective!

Why bother?

- Some FM industry success!
- NASSA, AirBus, Military, Transport, Electronic Money
- UK Military - For reliable systems FM is cost effective!
- People want reliable systems

Why bother?

- Some FM industry success!
- NASSA, AirBus, Military, Transport, Electronic Money
- UK Military - For reliable systems FM is cost effective!
- People want reliable systems
- Lawyers want laws

Why bother?

- Some FM industry success!
- NASSA, AirBus, Military, Transport, Electronic Money
- UK Military - For reliable systems FM is cost effective!
- People want reliable systems
- Lawers want laws
- But When!

What is it?

● Engineering ← FM → Maths

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Conceptual pressure is applied to a FM from both sides

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Conceptual pressure is applied to a FM from both sides
- Ideally the maths should give way to the engineering

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Conceptual pressure is applied to a FM from both sides
- Ideally the maths should give way to the engineering
- FM gives a rigorous definition of what you **want** and what you **have**

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Conceptual pressure is applied to a FM from both sides
- Ideally the maths should give way to the engineering
- FM gives a rigorous definition of what you **want** and what you **have**
- AND the ability to either:
 1. **verify** that what you have is what you want
 2. to **construct** what you get from what you wanted

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Conceptual pressure is applied to a FM from both sides
- Ideally the maths should give way to the engineering
- FM gives a rigorous definition of what you **want** and what you **have**
- AND the ability to either:
 1. **verify** that what you have is what you want
 2. to **construct** what you get from what you wanted
- FM: **Specification** \rightarrow **Implementation**

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Conceptual pressure is applied to a FM from both sides
- Ideally the maths should give way to the engineering
- FM gives a rigorous definition of what you **want** and what you **have**
- AND the ability to either:
 1. **verify** that what you have is what you want
 2. to **construct** what you get from what you wanted
- FM: **Specification** \rightarrow **Implementation**
- Key is Modularity and Abstraction (information hiding)! ●

Two Worlds two semantics

Objects
ADT
State – based

Relational
Semantics

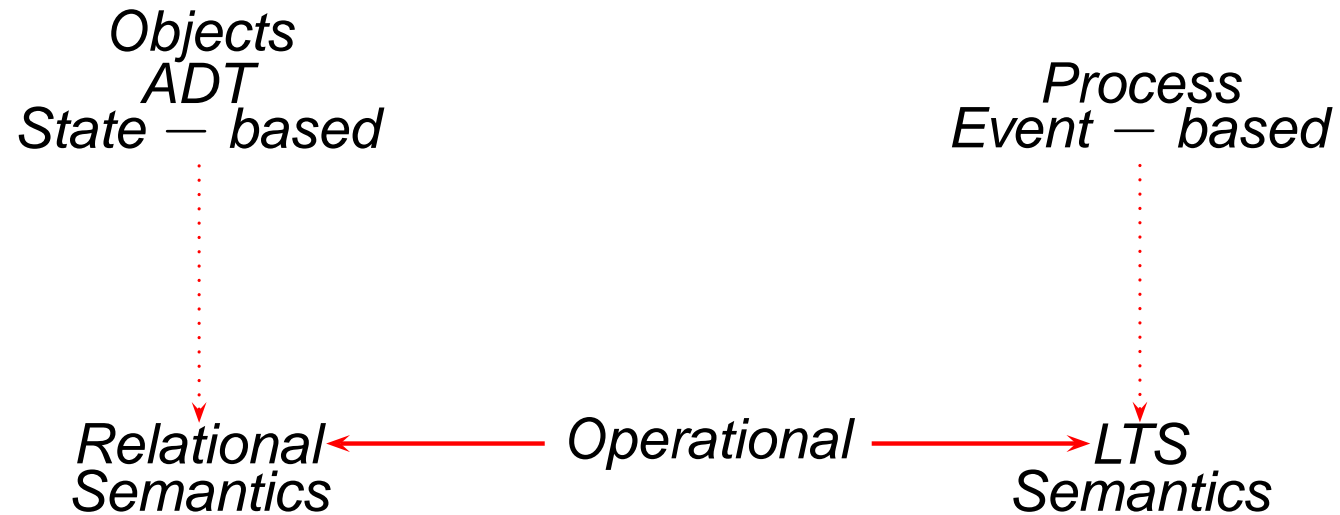


Process
Event – based

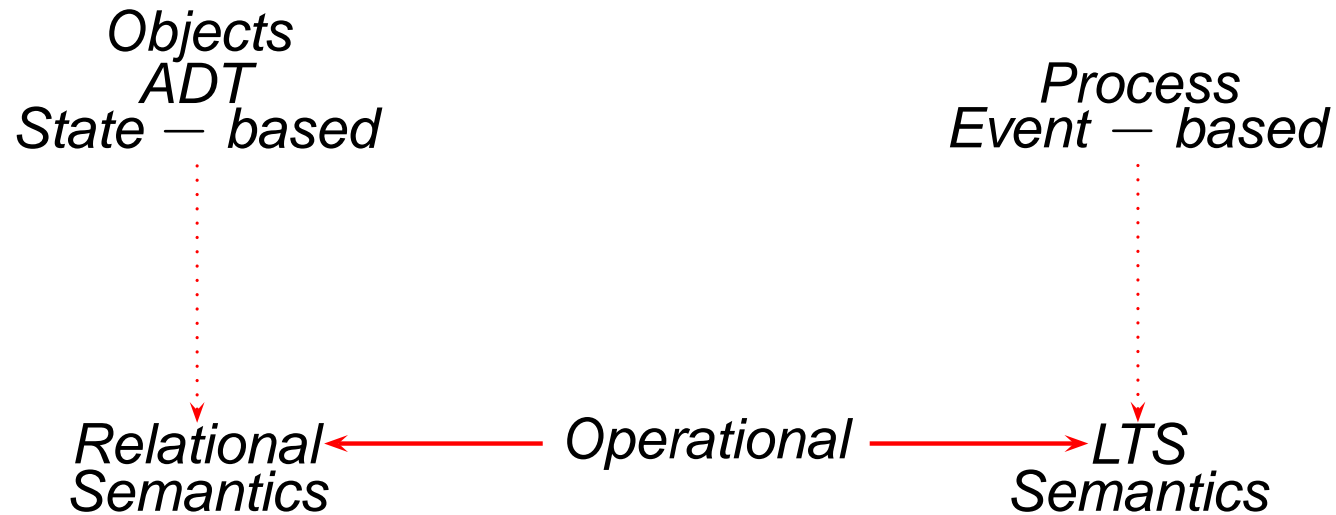
LTS
Semantics



Two Worlds two semantics



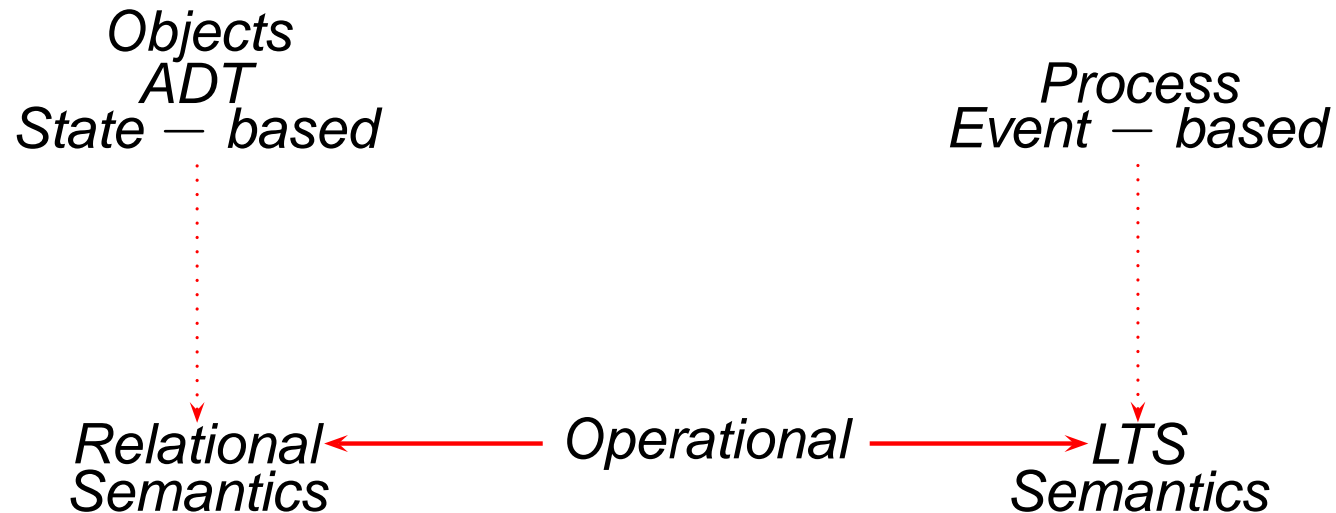
Two Worlds two semantics



Each World has its own

- Things to talk (overlapping)
- Set of assumptions
- Methodology

Two Worlds two semantics



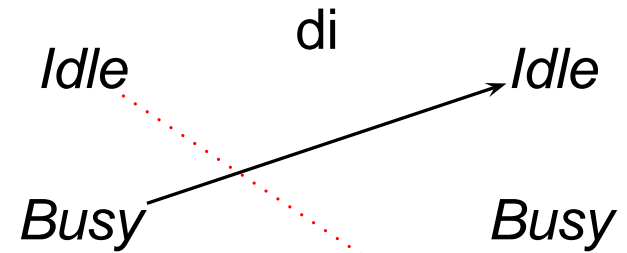
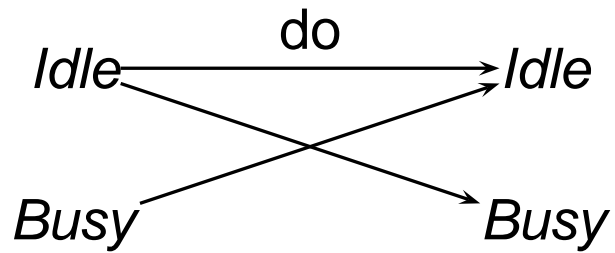
Each World has its own

- Things to talk (overlapping)
- Set of assumptions
- Methodology

These may be implicit.

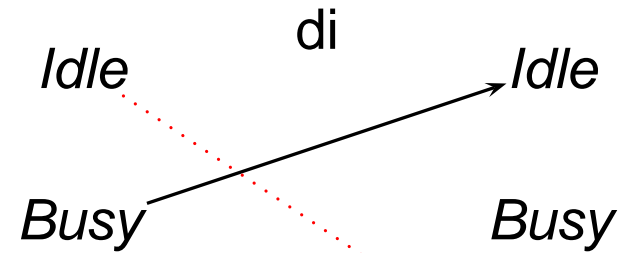
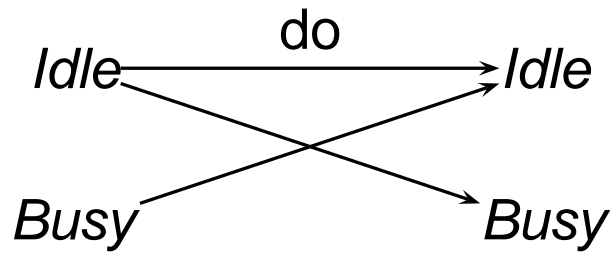
Example

- State based view of ADT



Example

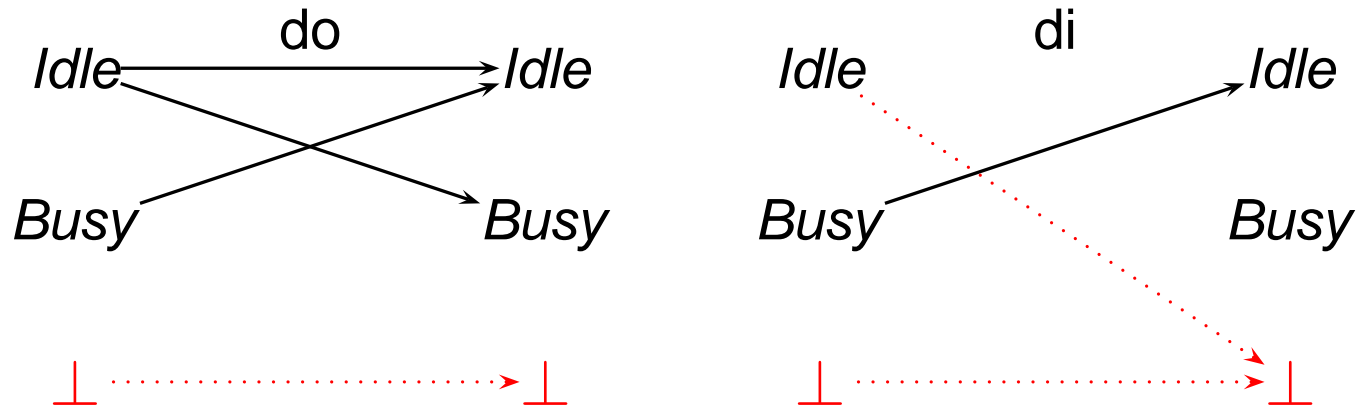
- State based view of ADT



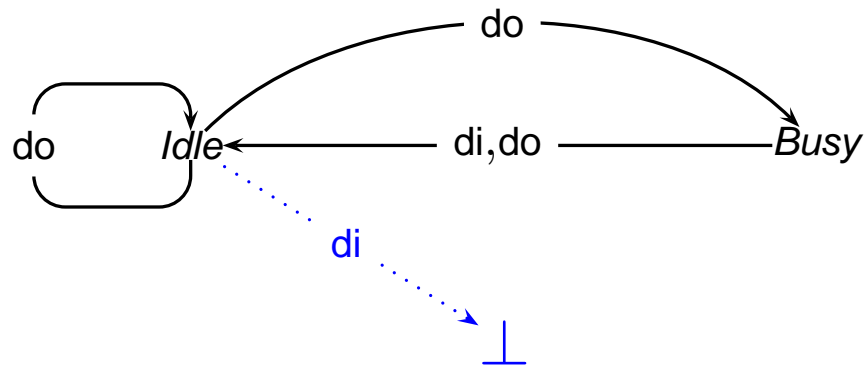
- What dose this mean?

Example

- State based view of ADT

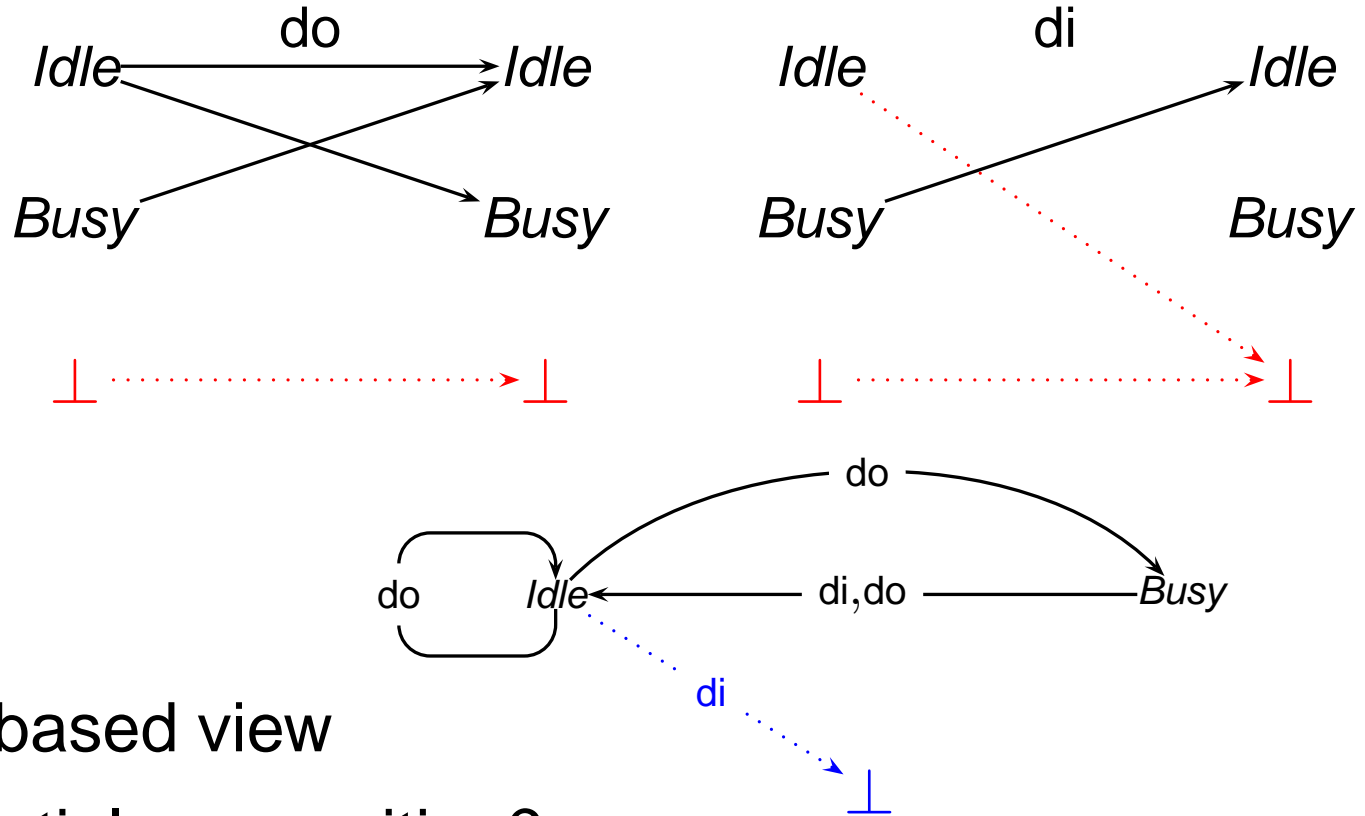


- Event based view



Example

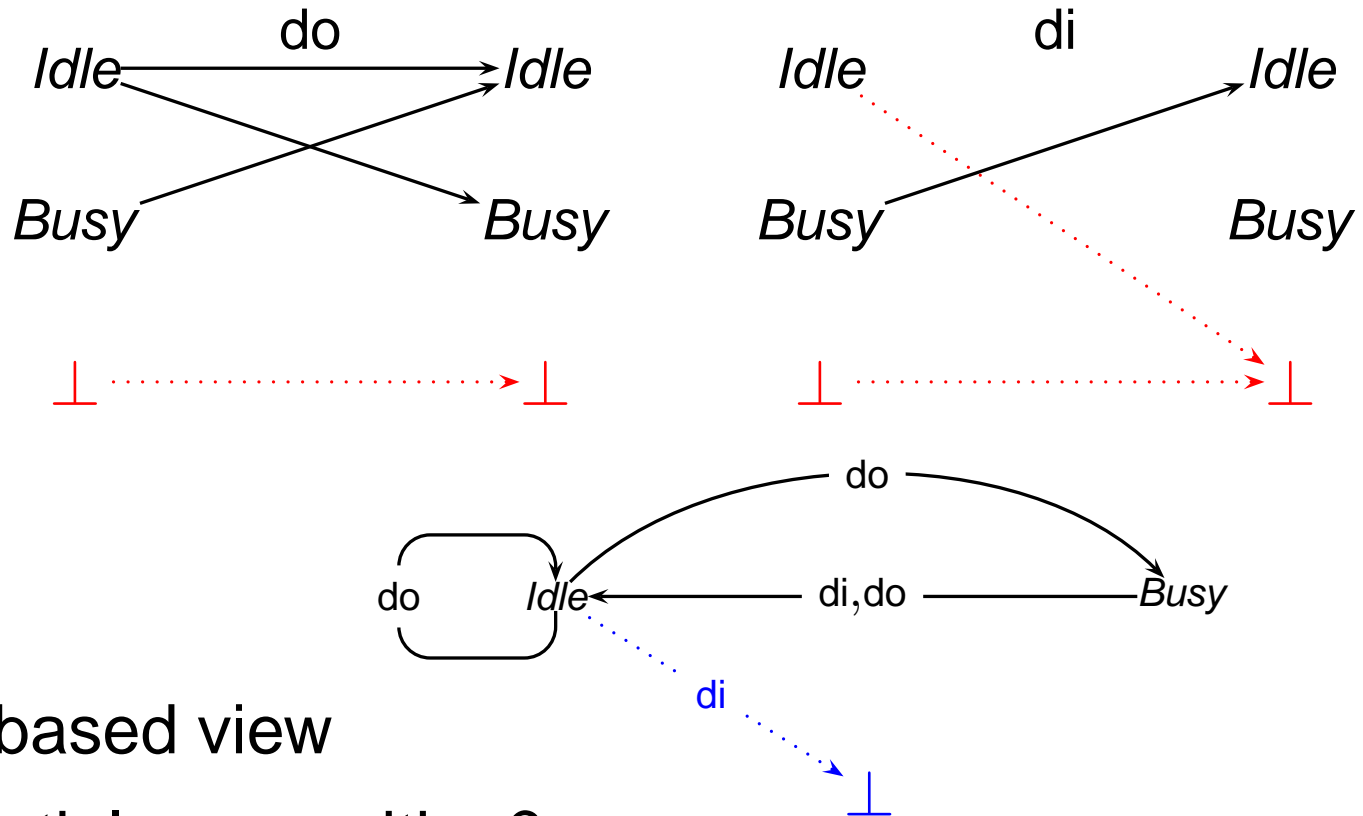
- State based view of ADT



- Event based view
- Sequential composition?

Example

- State based view of ADT



- Event based view
- Sequential composition?
- Are implicit methodological solutions OK?

Our Bridge

- Problem is that semantic models are ambiguous!
Have conceptual holes in them?

Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

Our Bridge

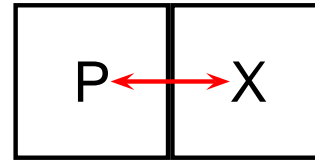
- The meaning of a specification is given by the set of implementations that satisfy it.

$[P]_x$

Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

$[P]_x$

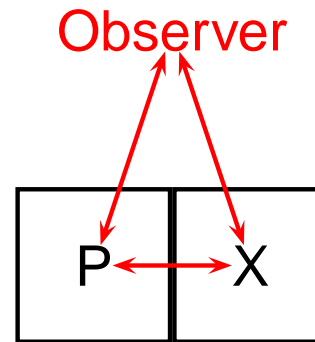


Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

$Obs([P]_x)$

$[P]_x$

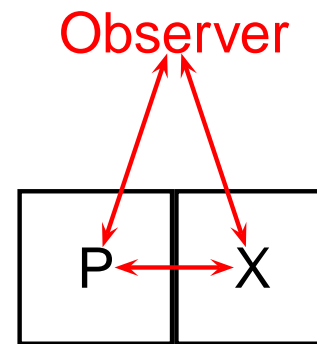


Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

$Obs([P]_x)$

$[P]_x$



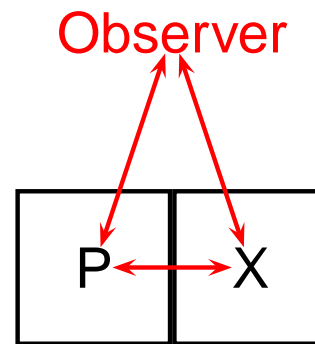
$Spec \sqsubseteq Imp$ Hence \sqsubseteq defines the meaning of a $Spec$.

Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

$Obs([P]_x)$

$[P]_x$



$Spec \sqsubseteq Imp$ Hence \sqsubseteq defines the meaning of a *Spec*.

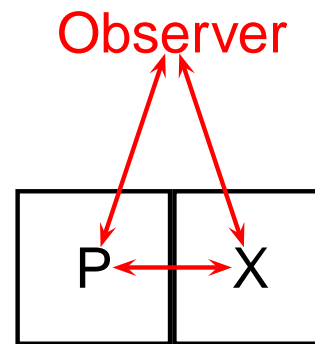
$A \sqsubseteq C$ if a **user** of *A* can not **observe** if they were given *C*

Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

$Obs([P]_x)$

$[P]_x$



$Spec \sqsubseteq Imp$ Hence \sqsubseteq defines the meaning of a $Spec$.

$A \sqsubseteq C$ if a **user** of A can not **observe** if they were given C

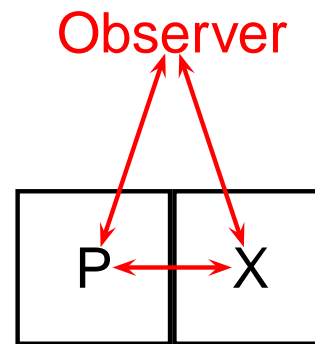
$$A \sqsubseteq_{(\Xi, Obs)} C \triangleq \forall x \in \Xi. Obs([C]_x) \subseteq Obs([A]_x)$$

Our Bridge

- The meaning of a specification is given by the set of implementations that satisfy it.

$Obs([P]_x)$

$[P]_x$



$Spec \sqsubseteq Imp$ Hence \sqsubseteq defines the meaning of a $Spec$.

$A \sqsubseteq C$ if a **user** of A can not **observe** if they were given C

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

can be specialised to give known refinements

What have we found - State based

- Using this bridge we can compare formalisms+methodologies+ semantics that have developed in relative isolation for some decades.

What have we found - State based

- Using this bridge we can compare formalisms+methodologies+ semantics that have developed in relative isolation for some decades.
- Both Z and B use ambiguous relational semantics with a notion of sequential composition that is “not what you would expect”.

What have we found - State based

- Using this bridge we can compare formalisms+methodologies+ semantics that have developed in relative isolation for some decades.
- Both Z and B use ambiguous relational semantics with a notion of sequential composition that is “not what you would expect”.
- Both have methodological solutions B has explicit methodology Z either has implicit methodology or is just mathematics so users beware!

What have we found - Event based

- Processes abstract away from **pushing a button** is an operation that causes **the button has been pushed** operation to occur.

What have we found - Event based

- Processes abstract away from **pushing a button** is an operation that causes **the button has been pushed** operation to occur.
- Processes can not be implemented

What have we found - Event based

- Processes abstract away from **pushing a button** is an operation that causes **the button has been pushed** operation to occur.
- Processes can not be implemented
- Process determinism is different from State based determinism.

What have we found - Event based

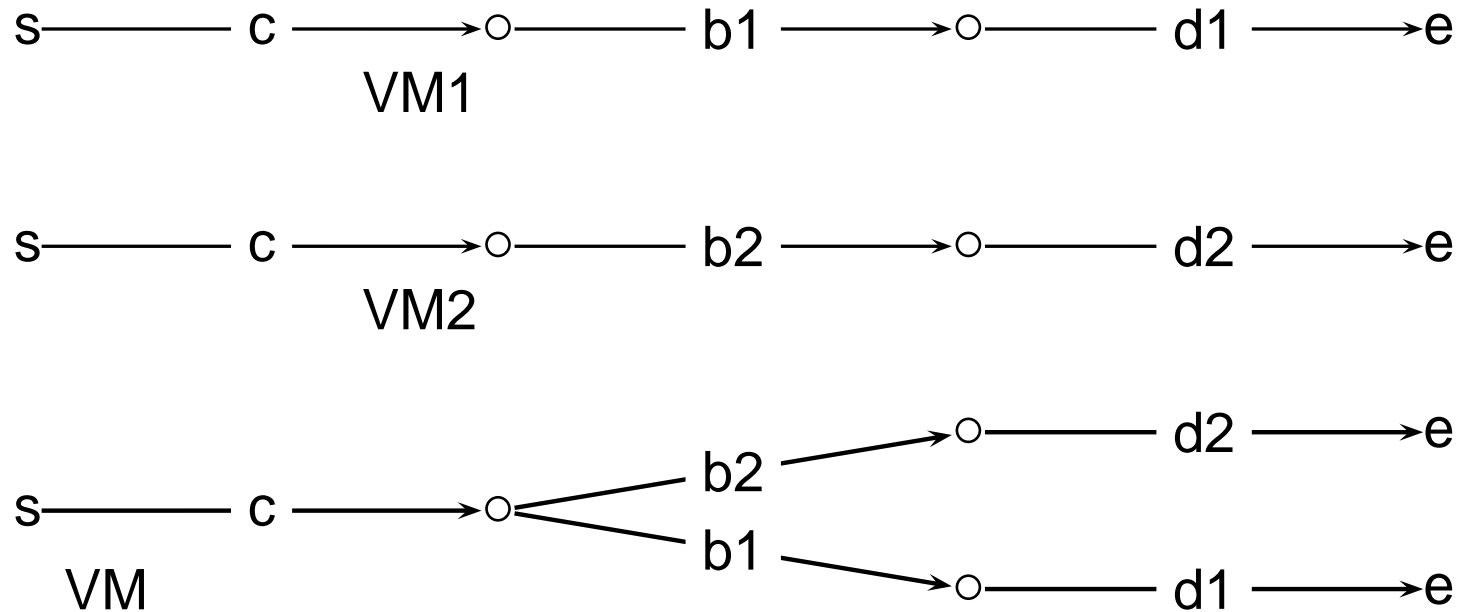
- Processes abstract away from **pushing a button** is an operation that causes **the button has been pushed** operation to occur.
- Processes can not be implemented
- Process determinism is different from State based determinism.
- Use Cases may cause a redefinition of *what is an atomic operation*

What have we found - Event based

- Processes abstract away from **pushing a button** is an operation that causes **the button has been pushed** operation to occur.
- Processes can not be implemented
- Process determinism is different from State based determinism.
- Use Cases may cause a redefinition of *what is an atomic operation*

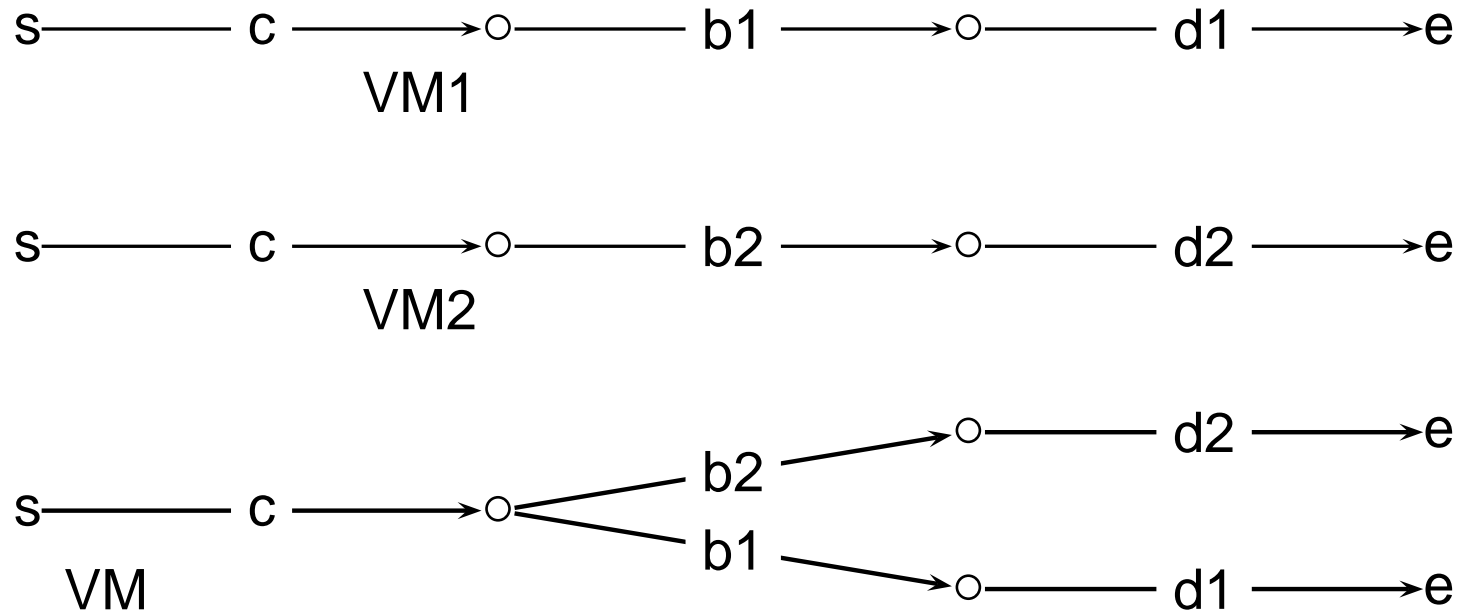
Specification

Using handshake actions.



Specification

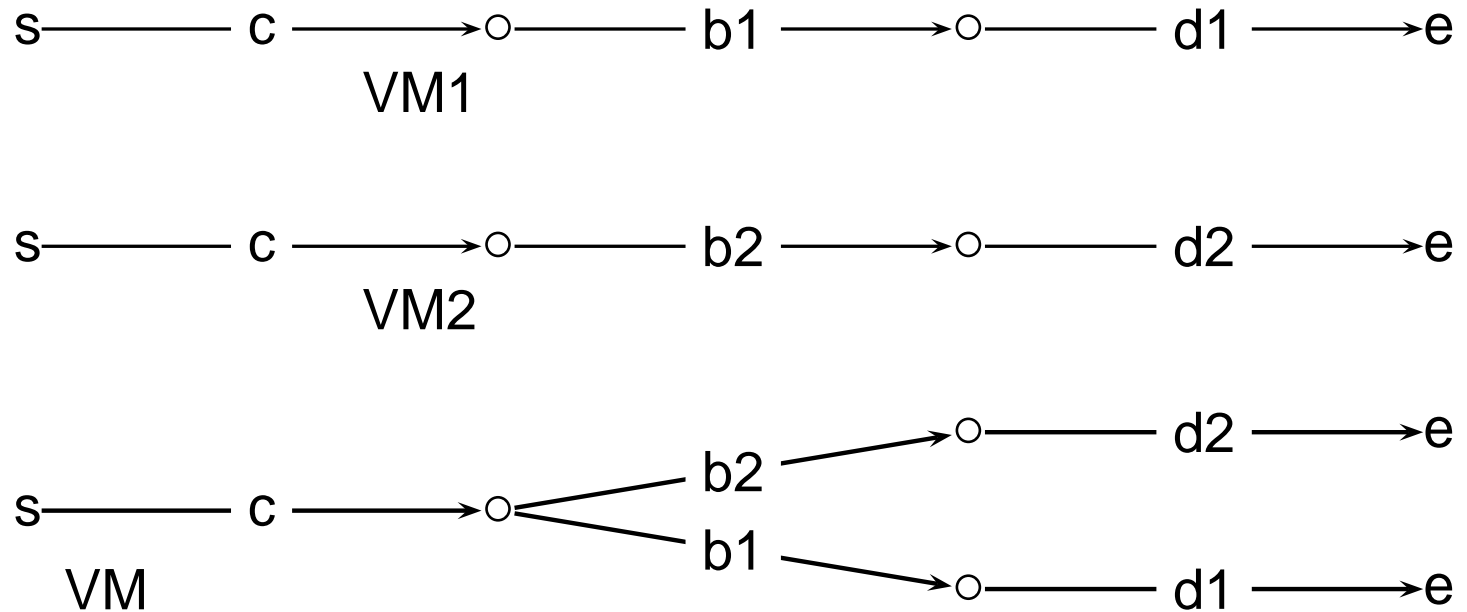
Using handshake actions.



Actions synchronise via $b1$ and $\overline{b1}$.

Specification

Using handshake actions.



Robot use cases:

- a** - drink d1 from VM1
- b** - drink d2 from VM2
- c** - drink d1 from VM.

Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\overline{b1};\overline{d1};r1$$

2. But **R1** fails to satisfy use case **b** - drink d2 from VM2.

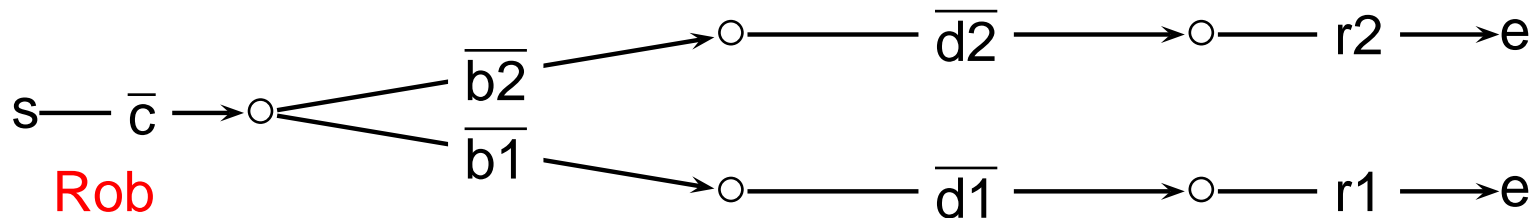
Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

2. But **R1** fails to satisfy use case **b** - drink d2 from VM2.

3. **Rob** satisfies use cases a and b.



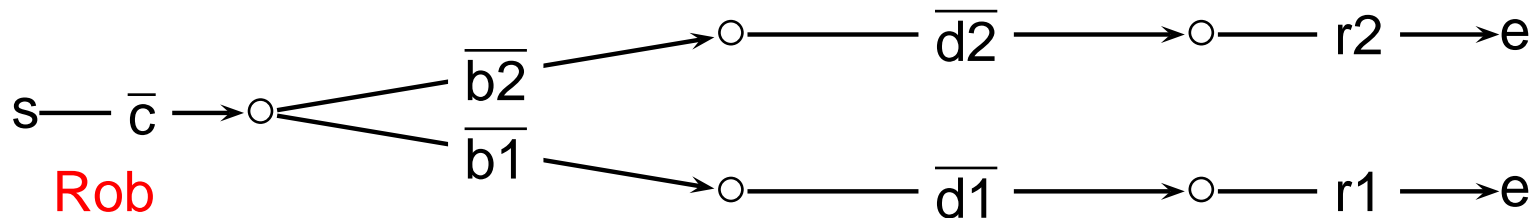
Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

2. But **R1** fails to satisfy use case **b** - drink d2 from VM2.

3. **Rob** satisfies use cases a and b.



4. **R1** \sqsubseteq_x **Rob** can be established using LOTOS's *ext*, *conf* or $\sqsubseteq_{F\delta}$ “weak sub-typing”

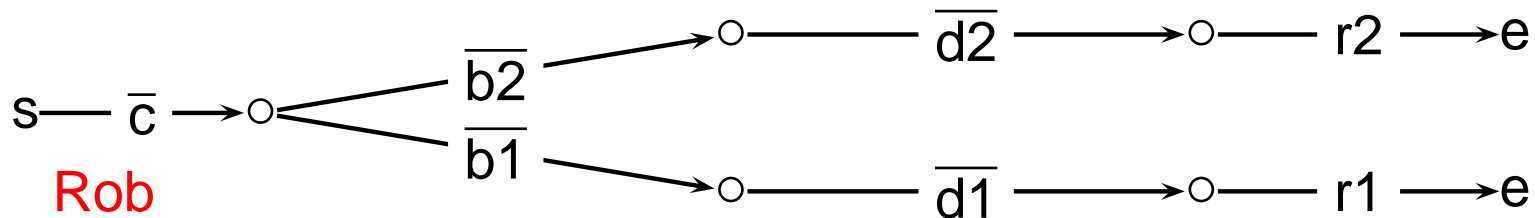
Stepwise development

1. Use case a - drink d1 from VM1 :

$$R1 \stackrel{\text{def}}{=} \bar{c};\bar{b1};\bar{d1};r1$$

2. But **R1** fails to satisfy use case **b** - drink d2 from VM2.

3. **Rob** satisfies use cases a and b.



4. **R1** \sqsubseteq_x **Rob** can be established using LOTOS's *ext*, *conf* or $\sqsubseteq_{F\delta}$ "weak sub-typing"
5. At this point actions are viewed as being defined at an adequate level of abstraction.

Problem

1. We are unable to satisfy the third use case c.

Problem

1. We are unable to satisfy the third use case c.
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.

Problem

1. We are unable to satisfy the third use case c.
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.
3. Using event based FM we would rewrite specification using different actions.

Problem

1. We are unable to satisfy the third use case c.
2. From this we conclude that the actions we have chosen are not at an adequate level of abstraction.
3. Using event based FM we would rewrite specification using different actions.
4. For large processes, changing the formal specification could entail a huge amount of work.

Conclusion

- We have taken a simple abstract approach to bridge two distinct styles of FM

Conclusion

- We have taken a simple abstract approach to bridge two distinct styles of FM
- We have then used the bridge to transfer ideas.

Conclusion

- We have taken a simple abstract approach to bridge two distinct styles of FM
- We have then used the bridge to transfer ideas.
- We are able to compare hidden assumptions
- We have have redefined formal semantics to broaden there scope

Conclusion

- We have taken a simple abstract approach to bridge two distinct styles of FM
- We have then used the bridge to transfer ideas.
- We are able to compare hidden assumptions
- We have have redefined formal semantics to broaden there scope

Conclusion

- We have taken a simple abstract approach to bridge two distinct styles of FM
- We have then used the bridge to transfer ideas.
- We are able to compare hidden assumptions
- We have have redefined formal semantics to broaden there scope