

Formal methods

Salery by negotiation

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
- Requires a change of world view!

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Requires a change of world view!
- From assembler to Compiled imperative

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Requires a change of world view!
- From assembler to Compiled imperative
- From Imperative to Declarative (functional)

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Requires a change of world view!
- From assembler to Compiled imperative
- From Imperative to Declarative (functional)
- Specifications are at a higher level of abstraction than code

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Requires a change of world view!
- From assembler to Compiled imperative
- From Imperative to Declarative (functional)
- Specifications are at a higher level of abstraction than code
- FM: Specification \rightarrow Implementation

What is it?

- Engineering \leftarrow FM \rightarrow Maths
- Requires a change of world view!
- From assembler to Compiled imperative
- From Imperative to Declarative (functional)
- Specifications are at a higher level of abstraction than code
- FM: Specification \rightarrow Implementation
- Key is Modularity and Abstraction (information hiding)!

The shape of things to hide?

- Functions interact with their context at two **States**

The shape of things to hide?

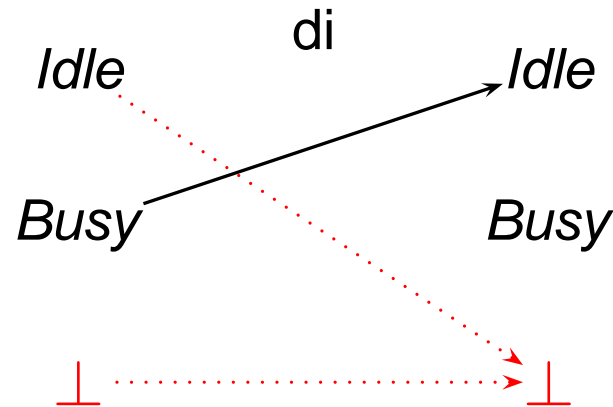
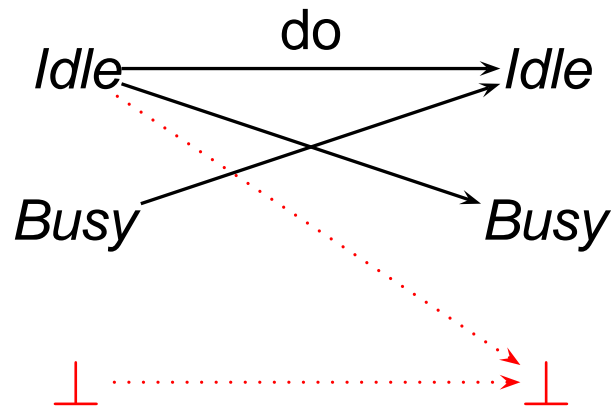
- Functions interact with their context at two **States**
- Transactional systems to restrictive - Interactive systems
- Interactive systems are built from **events**

The shape of things to hide?

- Functions interact with their context at two **States**
- **Events** interact with their context at three points

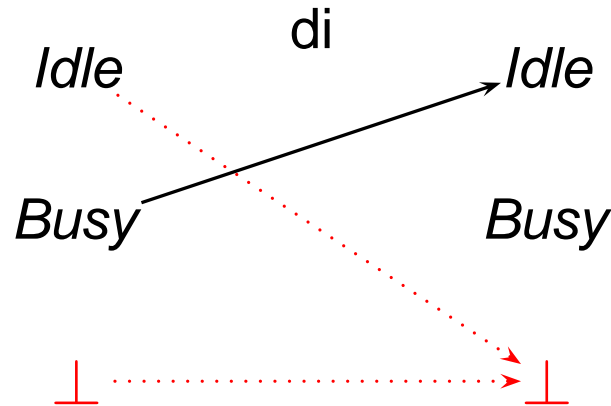
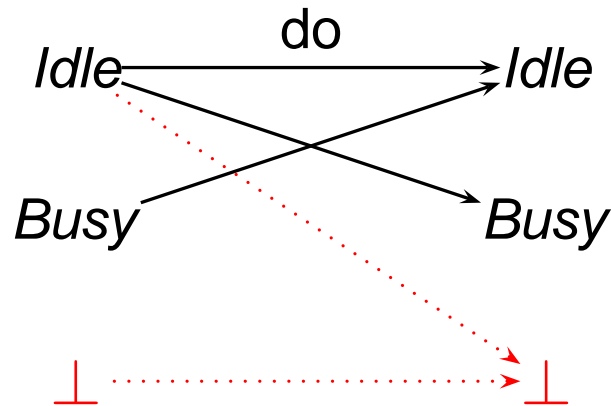
The shape of things to hide?

- Functions interact with their context at two **States**
- **Events** interact with their context at three points
- State based view



The shape of things to hide?

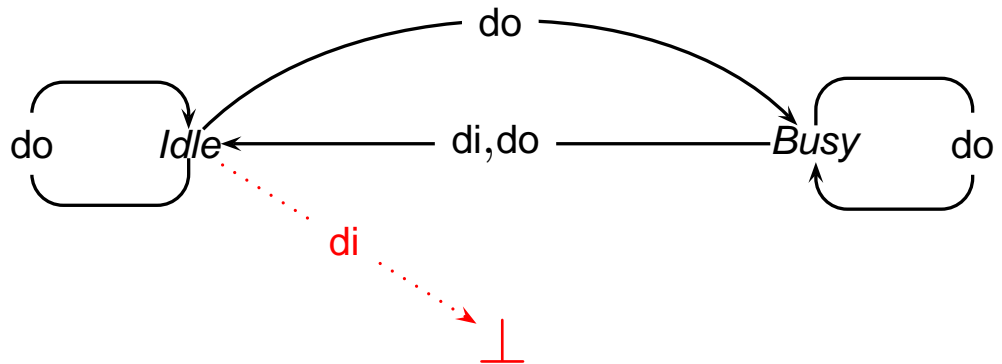
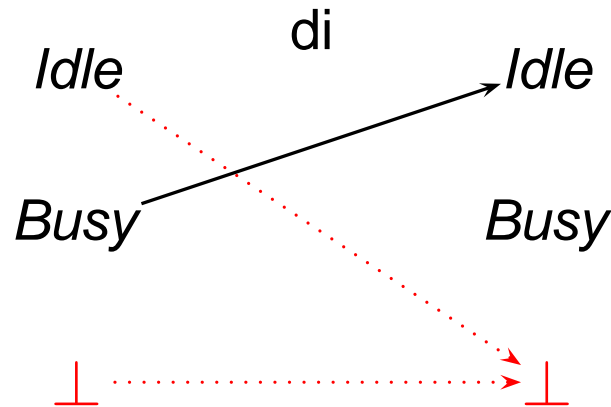
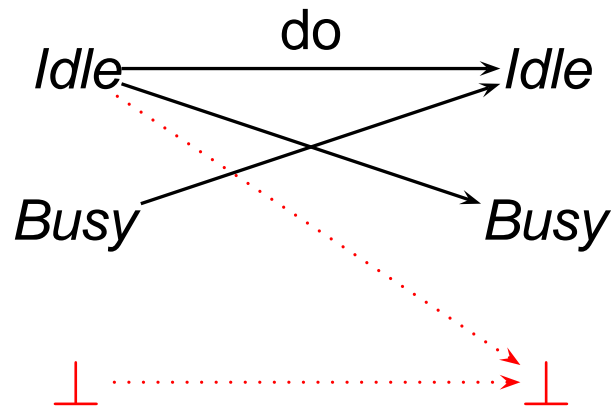
- Functions interact with their context at two **States**
- **Events** interact with their context at three points
- State based view



- What dose this mean?
- How many meanings?

The shape of things to hide?

- Functions interact with their context at two **States**
- **Events** interact with their context at three points
- State based view



- Event based view

A Bridge between Event and State views

- Simple isomorphism between sets of named relations and LTS
- But relations and LTS have many meanings

A Bridge between Event and State views

- Simple isomorphism between sets of named relations and LTS
- The meaning of a specification is given by the set of implementations that satisfy it.

A Bridge between Event and State views

- Simple isomorphism between sets of named relations and LTS
- The meaning of a specification is given by the set of implementations that satisfy it.
- $Spec \sqsubseteq Imp$ Hence refinement defines the meaning of a specification.

A Bridge between Event and State views

- Simple isomorphism between sets of named relations and LTS
- The meaning of a specification is given by the set of implementations that satisfy it.
- $Spec \sqsubseteq Imp$ Hence refinement defines the meaning of a specification.
- $A \sqsubseteq C$ if any user of A can not tell if they were actually given C

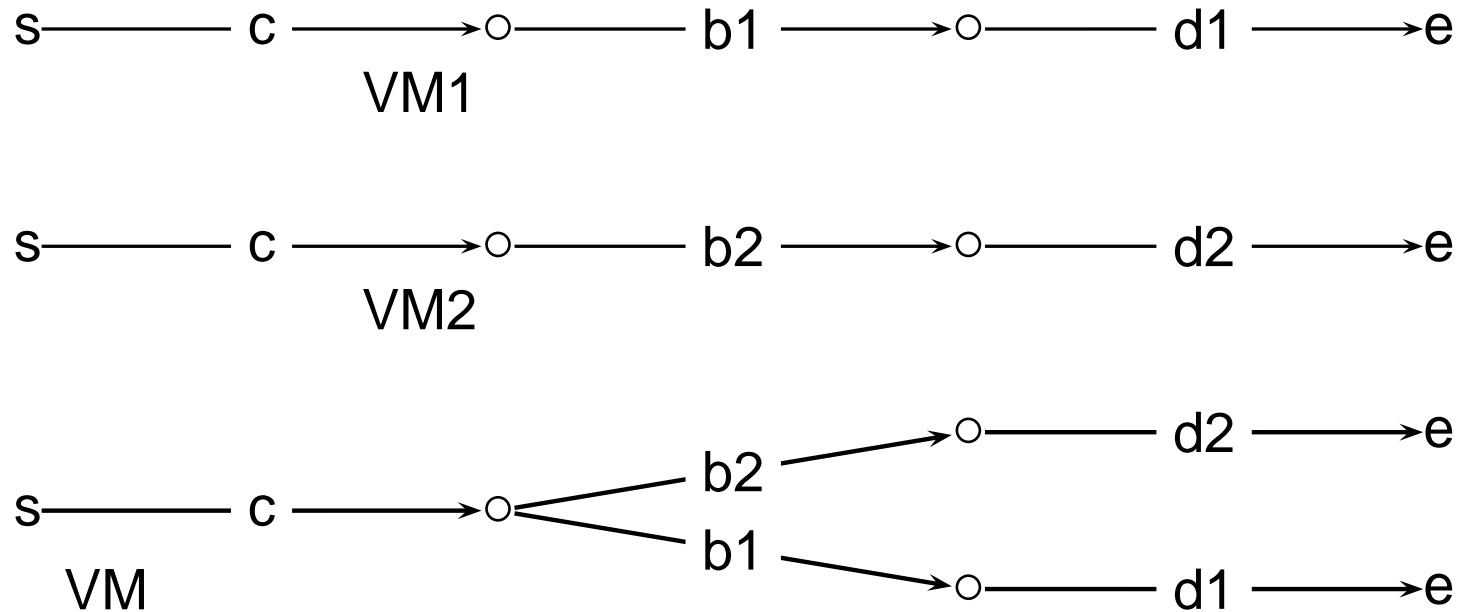
A Bridge between Event and State views

- Simple isomorphism between sets of named relations and LTS
- The meaning of a specification is given by the set of implementations that satisfy it.
- $Spec \sqsubseteq Imp$ Hence refinement defines the meaning of a specification.
- $A \sqsubseteq C$ if any user of A can not tell if they were actually given C
- Refinement that generalizes both state and event based definitions

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

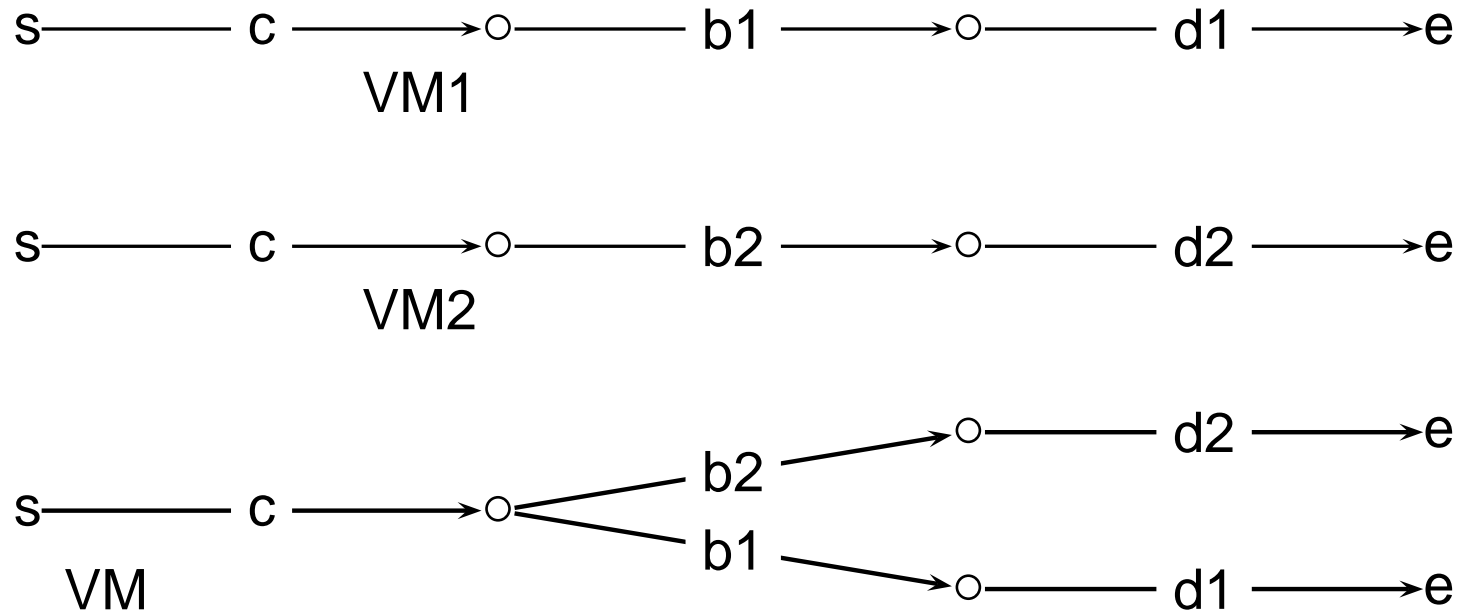
Specification

Using handshake actions.



Specification

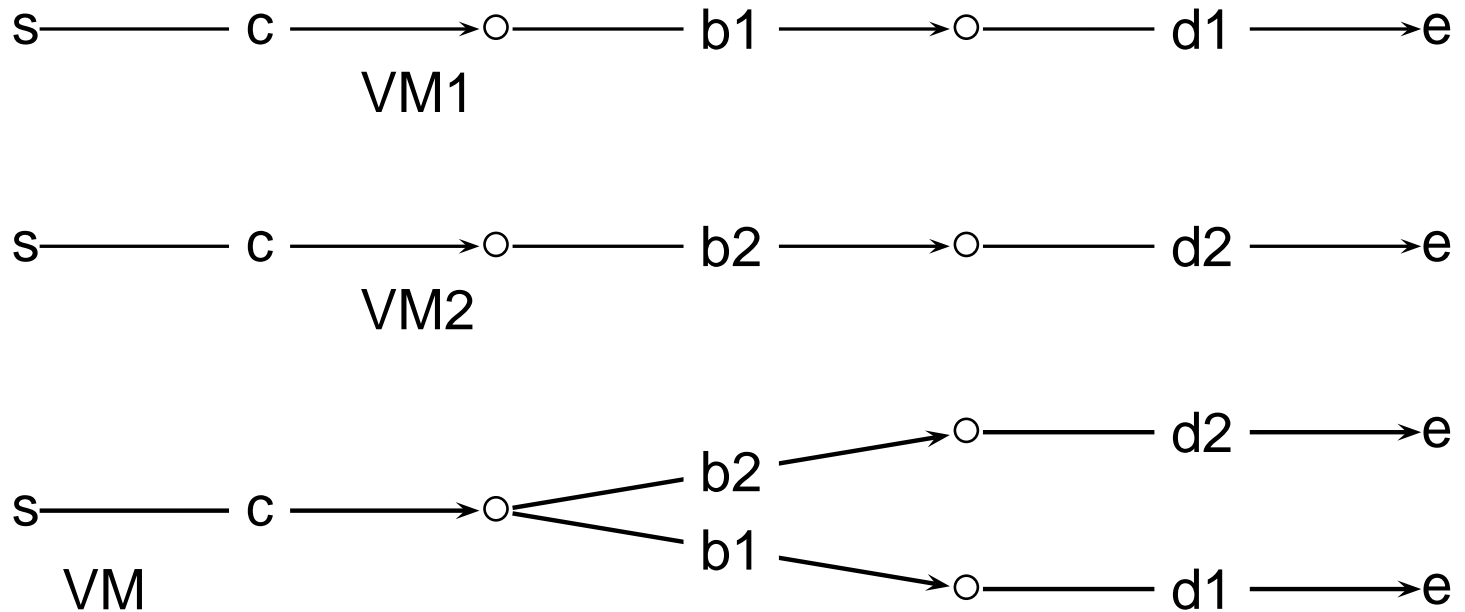
Using handshake actions.



Actions synchronize 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 :

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

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

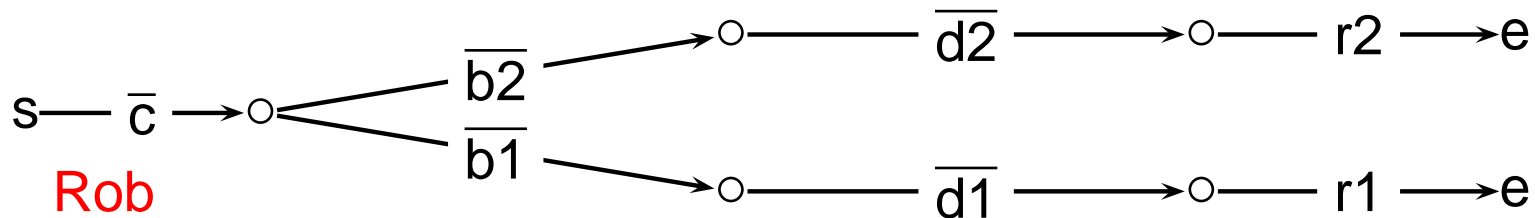
Stepwise development

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.



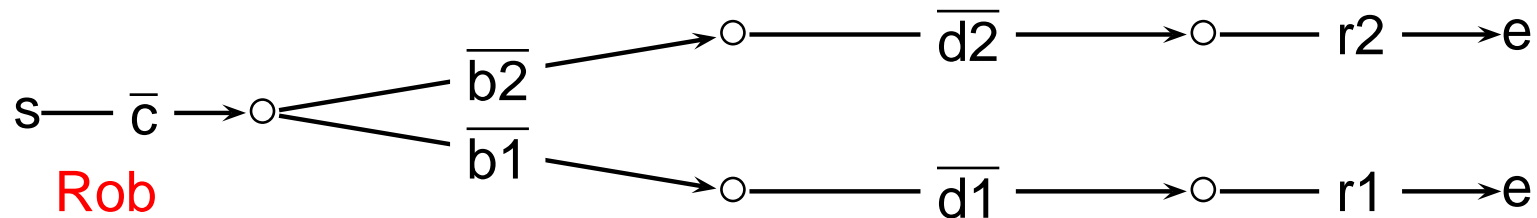
Stepwise development

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”

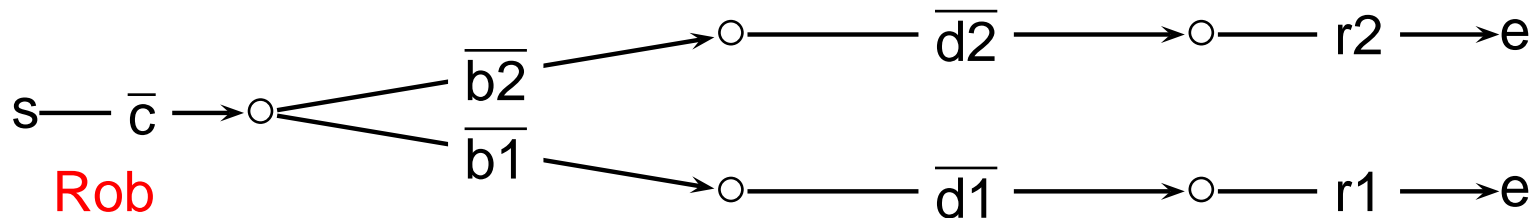
Stepwise development

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