

A Robust Semantics Hides Fewer Errors

S. Reeves D. Streader

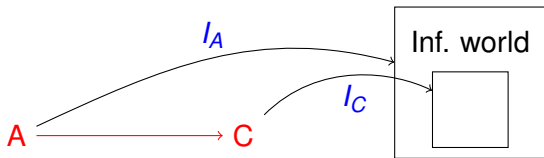
Department of Computer Science
The University of Waikato

FM2009

Outline of Talk

- 1 Semantics and Interpretations
- 2 Example
- 3 General theory
- 4 Robust semantics
Event-based
State-based
- 5 State-based steps
- 6 Conclusions

Semantics and Interpretations



- As engineers we want real, informal, things to work
 - Formality without **interpretation** is useless
 - More abstract - more interpretations
- 1 **Refinement**, a step from abstract **A** to concrete **C**
 - 2 Semantics are closer to real world interpretations
But it is surprisingly easy to select the wrong semantics

Example

- 1 By reasoning about proofs we can establish that a formal statement is not valid
- 2 **FS** from $(Pa \wedge Pb) \rightarrow R$ show $(Pa \rightarrow R) \vee (Pb \rightarrow R)$
- 3 Informal but rigorous argument
 - The assumption is that from a proof of $(Pa \wedge Pb)$ we can construct a proof of R
 - To know Pa (or Pb) is to know less than to know $(Pa \wedge Pb)$
 - Hence we cannot show $(Pa \rightarrow R)$ or $(Pb \rightarrow R)$ as both Pa and Pb might have been needed in the proof of R
- 4 But!

Example

- We have a formal proof!

1.	$(Pa \wedge Pb) \rightarrow R$	
2.	$\neg((Pa \rightarrow R) \vee (Pb \rightarrow R))$	Ass
3.	$\neg((\neg Pa \vee R) \vee (\neg Pb \vee R))$	Def \rightarrow .2
4.	$Pa \wedge \neg R \wedge Pb \wedge \neg R$	DeMorgan, $\neg E$
5.	$\neg(Pa \wedge Pb) \vee R$	Def \rightarrow .1
6.	$\neg(Pa \wedge Pb)$	Ass
7.	\perp	From – 4, 6
8.	R	Ass
9.	\perp	From – 4, 8
10.	\perp	$\vee E$, 5, 6, 7, 8, 9
11.	$(Pa \rightarrow R) \vee (Pb \rightarrow R)$	Cont, 2, 10

- What went Wrong?

Example

- The first mistake we made is to assume that formal statements have only one interpretation
 - ① constructively the informal argument is valid
 - ② classically the formal proof is valid
- With a classical interpretation
 - ① a mistake was made prior to stating the problem!
 - ② the first sentence was based on the wrong semantics
 - ③ classical logic is about truth (truth tables)
 - ④ constructive logic is about proof
- Changing the semantics easily causes problems
- **Using different semantics is both useful and common!**

CSP, ATP, Z

General Theory

- A general parameterised theory spanning both state- and event-based theories
- Fixing the parameters fixes the semantics and refinement

Failure, singleton failures, data refinement . . .

- A Unified Theory of Testing Hennessy, De Nicola

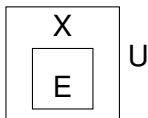
- Like simple event-based theories

- 1 homomorphic mappings $\llbracket A \oplus B \rrbracket = \llbracket A \rrbracket \llbracket \oplus \rrbracket \llbracket B \rrbracket$
- 2 congruences $A =_X A^* \rightarrow A \oplus B =_X A^* \oplus B$
- 3 the meaning is in the semantics More later
- 4 refinement tied to a semantics

- Not so often true in state-based models

A Unified Theory of Testing

- We use Testing semantics for ease of interpretation



- X is a context
- E is an entity
- U is a user

- A set of contexts Ξ and a set of observations \mathbb{O}
- $[E]_X$ is entity E in a context $X \in \Xi$
- An observation function $O([E]_X)$ from entities and contexts to sets of observations $\wp(\mathbb{O})$
- We refer to $\Xi \times \mathbb{O}$ as the *Frame* of the theory

Three semantics, refinements

- 1 • specification as a contract: if $X \in \Xi$ and only \mathbb{O} observed then $O([E]_X)$ defines what can be observed
- refinement as “satisfies contract” (subset of observations)

$$A \sqsubseteq_{\Xi, \mathbb{O}} C \triangleq \forall X \in \Xi. O([C]_X) \subseteq O([A]_X)$$

- 2 • specification as a proposition (relation)

$$A_{\Xi, \mathbb{O}}(X, o) \triangleq X \in \Xi \wedge o \in O([A]_X)$$

- refinement as implication (subset)

$$A \sqsubseteq_{\rightarrow, \Xi, \mathbb{O}} C \triangleq C_{\Xi, \mathbb{O}} \rightarrow A_{\Xi, \mathbb{O}}$$

Three semantics, refinements

- 3 • specification as set of implementations

$$\llbracket A \rrbracket_{I,\exists,O} \triangleq \{J \mid Det_{\exists,O}(J) \wedge J_{\exists,O} \rightarrow A_{\exists,O}\}$$

- refinement as subset

$$A \sqsubseteq_{I,\exists,O} C \triangleq \llbracket C \rrbracket_{I,\exists,O} \subseteq \llbracket A \rrbracket_{I,\exists,O}$$

semantics can be built from refinement

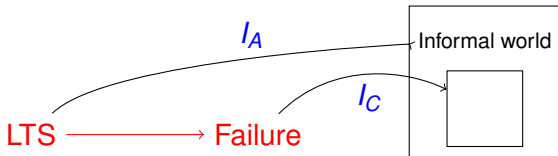
- The three refinement relations are equivalent

$$A \sqsubseteq_{I,\exists,O} C \quad \Leftrightarrow \quad A \sqsubseteq_{\rightarrow,\exists,O} C \quad \Leftrightarrow \quad A \sqsubseteq_{\exists,O} C$$

Concrete theories

- By fixing the parameters of our general theory we get concrete theories
 - ① handshake events - failure refinement
 - ② broadcast events - quiescent refinement
 - ③ contexts as programs - ADT refinement
 - ④ contexts as start states and observations as start/end state traces - operational refinement
- All are treated uniformly and all have versions of the three refinements and semantics.
- **Robust - has several equivalent interpretations**
Many semantics, same interpretation

Event-based theories



- Operational semantics as LTS
- semantics and refinement have been defined for Failures, Failures+Divergences ...
- It would be strange (and less robust) to define
 - Failures semantics but Failures+Divergences refinement or
 - LTS semantics but Failures refinement
- One definition of an operator on LTS
- Simple event-based definitions are robust

State-based semantics

- Operations are often given a partial relation semantics and may be interpreted differently

$$Op = \{(1, 1)\} \text{ and } State = \{1, 2\}$$

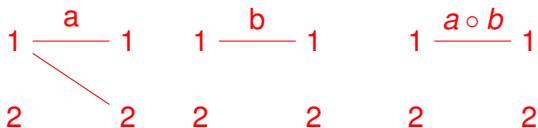
- ① partial correctness (may terminate from 1)
 - ② total correctness (must terminate from 1)
 - ③ undefined outside of precondition (anything from 2)
 - ④ guarded outside of precondition (nothing from 2)
- One semantics but many interpretations

Not Robust !

- One semantics but many refinements
- Is the meaning given by the semantics or the refinement?

Spivey's problem

- Sequential composition has unexpected behaviour!

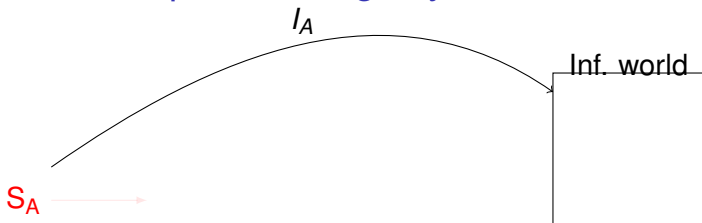


- We can define sequential composition to behave as expected for each interpretation but they are different
- This is very different to event-based theories

State-based methodology

- Z experts adopt a two step informal methodology
 - ① partial relations can be glued together (\wedge, \vee)
 - ② **total relations can be sequenced**
- B formalizes the methodology in its tool kit
- Z methodology for data refinement
 - ① Semantics as partial relations
 - ② Prior to sequencing lift and totalise according to interpretation
 - ③ I/O seen upon termination of program or during execution
- Two semantics transformations
- The I/O transformation changes guarded operations

Stepwise Design by Refinement



- Step 1 define specification and interpretation

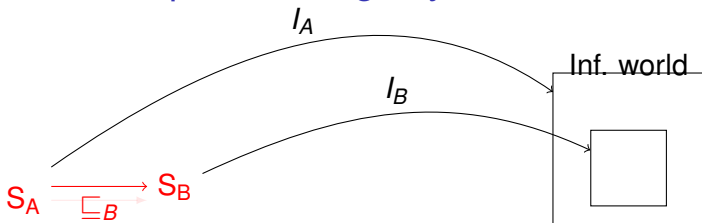
S_A, I_A

- Step 2 define $\sqsubseteq_B S_B$ and I_B

- Step 3 $\sqsubseteq_C S_C$ and I_C

- Refinement = pre-order, \sqsubseteq + guarantee
- Specifications, S_A, S_B and S_C in different theories!

Stepwise Design by Refinement



- Step 1 define specification and interpretation

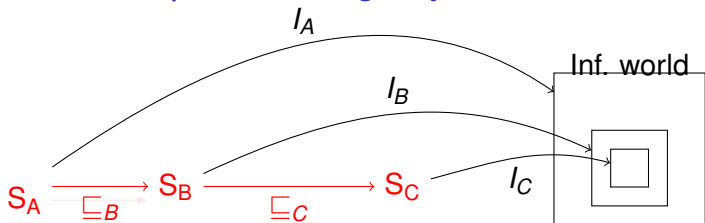
S_A, I_A

- Step 2 define $\sqsubseteq_B S_B$ and I_B

- Step 3 $\sqsubseteq_C S_C$ and I_C

- Refinement = pre-order, \sqsubseteq + guarantee
- Specifications, S_A, S_B and S_C in different theories!

Stepwise Design by Refinement



- Step 1 define specification and interpretation

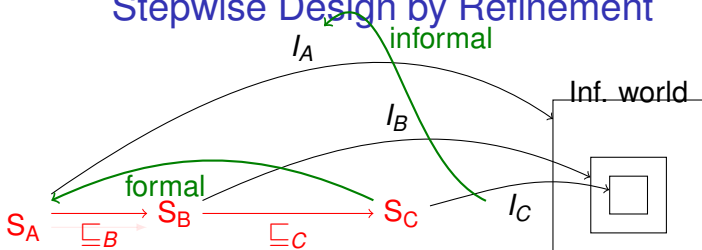
S_A, I_A

- Step 2 define $\sqsubseteq_B S_B$ and I_B

- Step 3 $\sqsubseteq_C S_C$ and I_C

- Refinement = pre-order, \sqsubseteq + guarantee
- Specifications, S_A, S_B and S_C in different theories!

Stepwise Design by Refinement



- Step 1 define specification and interpretation

S_A, I_A

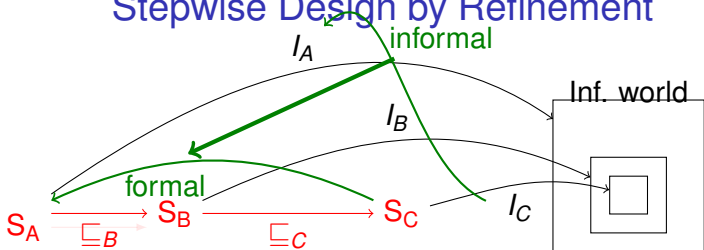
- Step 2 define $\sqsubseteq_B S_B$ and I_B

- Step 3 $\sqsubseteq_C S_C$ and I_C

- Refinement = pre-order, \sqsubseteq + **guarantee**

- Specifications, S_A, S_B and S_C in different theories!

Stepwise Design by Refinement



- Step 1 define specification and interpretation

S_A, I_A

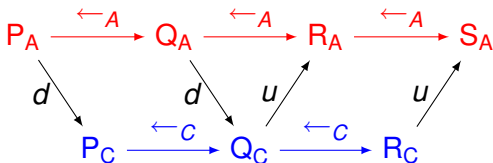
- Step 2 define $\sqsubseteq_B S_B$ and I_B

- Step 3 $\sqsubseteq_C S_C$ and I_C

- Refinement = pre-order, \sqsubseteq + **guarantee**
- Specifications, S_A, S_B and S_C in different theories!

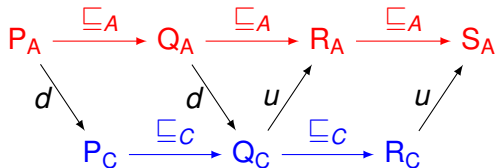
Galois Connections

- A weak form of isomorphism between logical theories
- defined by a pair of mappings (d, u)



Galois Connections

- A weak form of isomorphism between logical theories
- defined by a pair of mappings (d, u)



- Galois Steps UTT+GC+UTT
 - ① both preserves and reflects refinement
 - ② very strict step - may be regarded as a refinement.

$$d(P_A) \sqsubseteq_C R_C$$

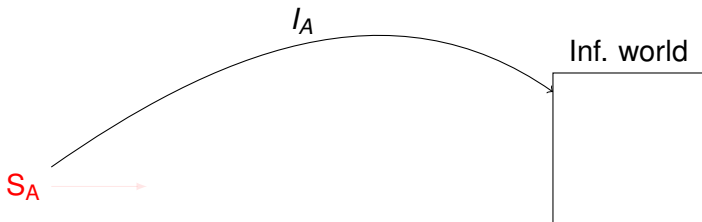
$$P_A \sqsubseteq_A u(R_C)$$

Galois Steps

- Embedding and forgetful pair is a simple Galois step
 - 1 silent outside of abstract frame
 - 2 subset inside the abstract frame
 - Galois steps add flexibility
 - 1 we can do much of the work of retrenchment
 - 2 we can compare different theories
 - 3 implement one theory in another

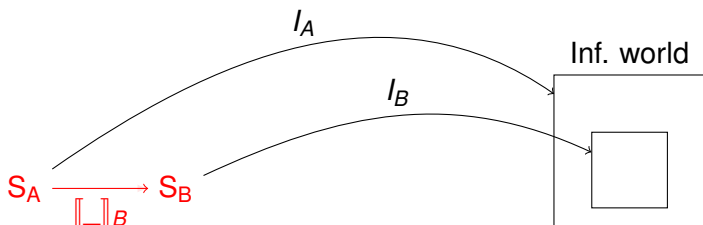
 - 4 postpone defining semantic details
 - 5 avoid Spivey's problem
- no abstract seq.

State-based Galois Steps



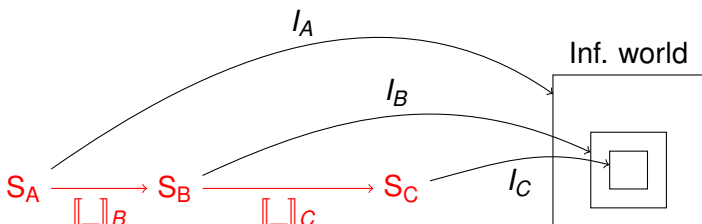
- Step 1** Let $\Sigma \triangleq (_ \oplus _, Act)$ $(Act \triangleq \{a, b, c\})$
 $S_A \triangleq T_\Sigma$, and I_A of Act is a set
- Step 2** $\llbracket _ \rrbracket_B \triangleq Act \rightarrow St \times St$, $\llbracket \oplus \rrbracket_B \triangleq \cup$ $(St \triangleq \{1, 2\})$
 I_B of Act is a set of operations $(\llbracket a \rrbracket_B \triangleq \{(1, 1)\})$
- Step 3** lift and totalise the relations
 $\llbracket \cup \rrbracket_C \triangleq \cup$ $(\llbracket \{(1, 1)\} \rrbracket_C \triangleq \{(1, 1), (2, \perp), (\perp, \perp)\})$

State-based Galois Steps



- Step 1 Let $\Sigma \triangleq (_ \oplus _, Act)$ $(Act \triangleq \{a, b, c\})$
 $S_A \triangleq T_\Sigma$, and I_A of Act is a set
- Step 2 $[]_B \triangleq Act \rightarrow St \times St$, $[\oplus]_B \triangleq \cup$ $(St \triangleq \{1, 2\})$
 I_B of Act is a set of operations $([a]_B \triangleq \{(1, 1)\})$
- Step 3 lift and totalise the relations
 $[\cup]_c \triangleq \cup$ $([\{(1, 1) \}]_c \triangleq \{(1, 1), (2, \perp), (\perp, \perp)\})$

State-based Galois Steps



- Step 1 Let $\Sigma \triangleq (_ \oplus _, Act)$ $(Act \triangleq \{a, b, c\})$
 $S_A \triangleq T_\Sigma$, and I_A of Act is a set
- Step 2 $[[_]]_B \triangleq Act \rightarrow St \times St$, $[[\oplus]]_B \triangleq \cup$ $(St \triangleq \{1, 2\})$
 I_B of Act is a set of operations $([[a]]_B \triangleq \{(1, 1)\})$
- Step 3 lift and totalise the relations
 $[[\cup]]_C \triangleq \cup$ $([[\{(1, 1)\}]]_C \triangleq \{(1, 1), (2, \perp), (\perp, \perp)\})$

Conclusions

- Unified Theory of Testing and Galois Steps
 - 1 Robust general theory
 - 2 prevents us from falling into Spivey's problem
 - 3 Allows the decision as to the detailed semantics and interpretation of specifications to be made late in the design process