

Generic Tools via General Refinement

S. Reeves D. Streader

Department of Computer Science
University of Waikato

Harnessing Theories for Tool Support in Software

Outline of Talk

Motivation

Formal Methods
straight jacket
Example

- 1 Motivation
Formal Methods straight jacket
Example

Lax Refinement

Lax Refinement in
two parts
General refinement
Vertical refinement/
Galois Connection
Example

- 2 Lax Refinement
Lax Refinement in two parts
General refinement
Vertical refinement/ Galois Connection
Example

Summary

Where are we?

Motivation

Formal Methods
straight jacket

Example

Lax Refinement

Lax Refinement in
two parts

General refinement

Vertical refinement/
Galois Connection

Example

Summary

1 Motivation

Formal Methods straight jacket

Example

2 Lax Refinement

Lax Refinement in two parts

General refinement

Vertical refinement/ Galois Connection

Example

Stepwise Refinement \sqsubseteq

- $A \sqsubseteq C$ is a formal guarantee that C satisfies A
- Two parts of the straight jacket
 - 1 Fixed interaction
 - 2 Refinement is reduction of nondeterminism.
- Lax Refinement tackles both

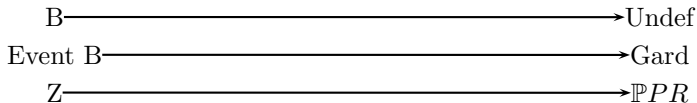
Stepwise Refinement \sqsubseteq

- $A \sqsubseteq C$ is a formal guarantee that C satisfies A
- Two parts of the straight jacket
 - 1 Fixed interaction
 - 2 Refinement is reduction of nondeterminism.
- Lax Refinement tackles both

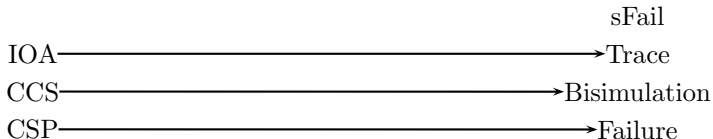
Stepwise Refinement \sqsubseteq

- $A \sqsubseteq C$ is a formal guarantee that C satisfies A
- Two parts of the straight jacket
 - 1 Fixed interaction
 - 2 Refinement is reduction of nondeterminism.
- Lax Refinement tackles both

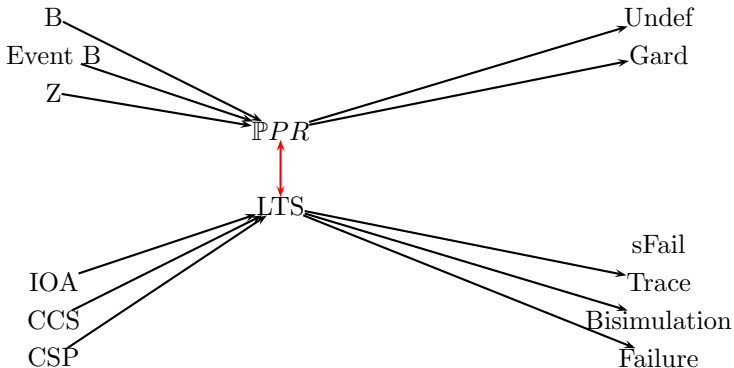
1 - Fixed interaction



Seperate theories



1 - Fixed interaction



Motivation

Formal Methods
straight jacket

Example

Lax Refinement

Lax Refinement in
two parts

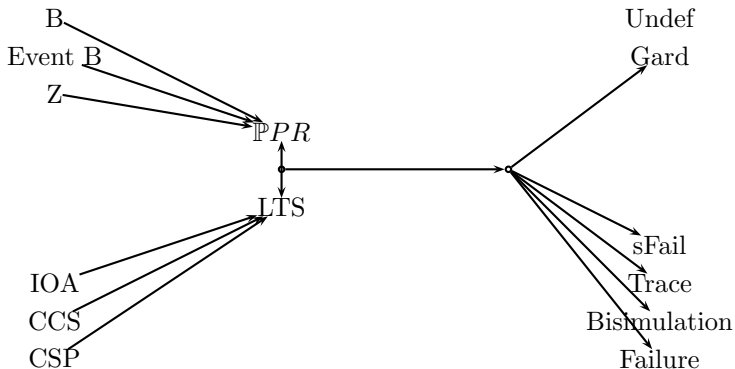
General refinement

Vertical refinement/
Galois Connection

Example

Summary

1 - Fixed interaction



Motivation

Formal Methods
straight jacket

Example

Lax Refinement

Lax Refinement in
two parts

General refinement

Vertical refinement/
Galois Connection

Example

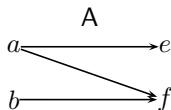
Summary

Refinement is **strictly** reduction of nondeterminism

- A strict view of reduction of nondeterminism.
- Undefined outside of domain
- A more Lax view outside of the frame

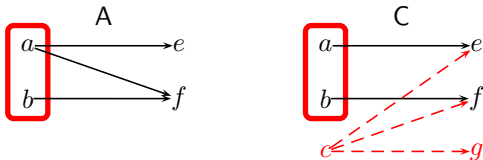
Refinement is **strictly** reduction of nondeterminism

- A strict view of reduction of nondeterminism.
- Undefined outside of domain
- A more Lax view outside of the frame



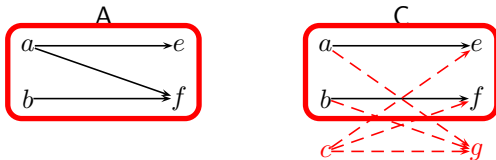
Refinement is **strictly** reduction of nondeterminism

- A strict view of reduction of nondeterminism.
- Undefined outside of domain
- A more Lax view outside of the frame



Refinement is **strictly** reduction of nondeterminism

- A strict view of reduction of nondeterminism.
- Undefined outside of domain
- A more Lax view outside of the frame



Where are we?

Motivation

Formal Methods
straight jacket

Example

Lax Refinement

Lax Refinement in
two parts

General refinement

Vertical refinement/
Galois Connection

Example

Summary

1 Motivation

Formal Methods straight jacket

Example

2 Lax Refinement

Lax Refinement in two parts

General refinement

Vertical refinement/ Galois Connection

Example

First specify correct behaviour later specify errors

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

Correct Behaviour

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_C

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

First specify correct behaviour later specify errors

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

Correct Behaviour

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 $t' = \emptyset$

First specify correct behaviour later **specify errors**
 $\text{Set}_C = \text{Set}_A + \text{errors}$

Correct Behaviour

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_C

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

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

- In most formalisms NO!
 - In most formalisms Set_A allows Put^∞
- But informally we might want to do just this
- If so what could Set_A formally specify
 - We interpret Set_A as only guaranteeing behaviour when no errors occur. Thus when:
 1. state remains in State_A and
 2. only operations in Set_A are called/observed.
- Lax refinement will formalise such development steps

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

- In most formalisms NO!
 - In most formalisms Set_A allows Put^∞
- But informally we might want to do just this
- If so what could Set_A formally specify
 - We interpret Set_A as only guaranteeing behaviour when no errors occur. Thus when:
 1. state remains in State_A and
 2. only operations in Set_A are called/observed.
- Lax refinement will formalise such development steps

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

- In most formalisms NO!
 - In most formalisms Set_A allows Put^∞
- But informally we might want to do just this
- If so what could Set_A formally specify
 - We interpret Set_A as only guaranteeing behaviour when no errors occur. Thus when:
 1. state remains in State_A and
 2. only operations in Set_A are called/observed.
- Lax refinement will formalise such development steps

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

- In most formalisms NO!
 - In most formalisms Set_A allows Put^∞
- But informally we might want to do just this
- If so what could Set_A formally specify
 - We interpret Set_A as only guaranteeing behaviour when no errors occur. Thus when:
 1. state remains in State_A and
 2. only operations in Set_A are called/observed.
- Lax refinement will formalise such development steps

Where are we?

Motivation

Formal Methods
straight jacket
Example

Lax Refinement

Lax Refinement in
two parts

General refinement
Vertical refinement/
Galois Connection
Example

Summary

1 Motivation
Formal Methods straight jacket
Example

2 Lax Refinement
Lax Refinement in two parts
General refinement
Vertical refinement/ Galois Connection
Example

Motivation

Formal Methods

straight jacket

Example

Lax
Refinement

Lax Refinement in
two parts

General refinement

Vertical refinement/
Galois Connection

Example

Summary

Lax = General + Vertical

- 1 General refinement (2003)
- 2 Vertical refinement

Lax = General + Vertical

Motivation

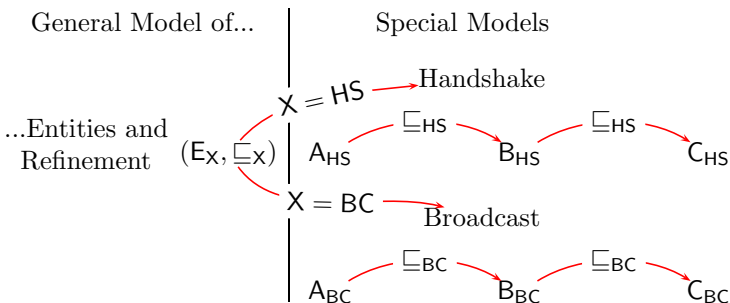
Formal Methods
straight jacket
Example

Lax Refinement

Lax Refinement in
two parts
General refinement
Vertical refinement/
Galois Connection
Example

Summary

- 1 General refinement (2003)
- 2 Vertical refinement



Where are we?

Motivation

Formal Methods
straight jacket
Example

Lax Refinement

Lax Refinement in
two parts

General refinement

Vertical refinement/
Galois Connection
Example

Summary

1 Motivation
Formal Methods straight jacket
Example

2 Lax Refinement
Lax Refinement in two parts
General refinement
Vertical refinement/ Galois Connection
Example

What is refinement?

- 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 $O : \mathbb{E} \rightarrow \mathbb{O}$

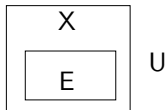
- \mathbb{O} is the set of complete traces

-

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

What is refinement?

- 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 $O : \mathbb{E} \rightarrow \mathbb{O}$



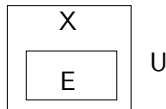
- \mathbb{O} is the set of complete traces

•

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

What is refinement?

- 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 $O : \mathbb{E} \rightarrow \mathbb{O}$



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

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

Concrete refinements

- General Refinement is made concrete by fixing \mathbb{E} , Ξ and O
- The denotational semantics of entities is a relation:

$$\llbracket A \rrbracket_{\Xi, O} \triangleq \{(x, o) \mid x \in \Xi, o \in O(\llbracket A \rrbracket_x)\}$$

- Refinement is subset or implication in a logical theory:
A theory T is pair (E_T, \sqsubseteq_T)

Concrete refinements

- General Refinement is made concrete by fixing \mathbb{E} , Ξ and O
- The denotational semantics of entities is a relation:

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

- Refinement is subset or implication in a logical theory:
A theory T is pair (E_T, \sqsubseteq_T)

Concrete refinements

- General Refinement is made concrete by fixing \mathbb{E} , Ξ and O
- The denotational semantics of entities is a relation:

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

- Refinement is subset or implication in a logical theory:
A theory T is pair (E_T, \sqsubseteq_T)

Where are we?

Motivation

Formal Methods
straight jacket
Example

Lax Refinement

Lax Refinement in
two parts
General refinement

Vertical refinement/
Galois Connection
Example

Summary

1 Motivation
Formal Methods straight jacket
Example

2 Lax Refinement
Lax Refinement in two parts
General refinement
Vertical refinement/ Galois Connection
Example

Lax = General + Vertical

Motivation

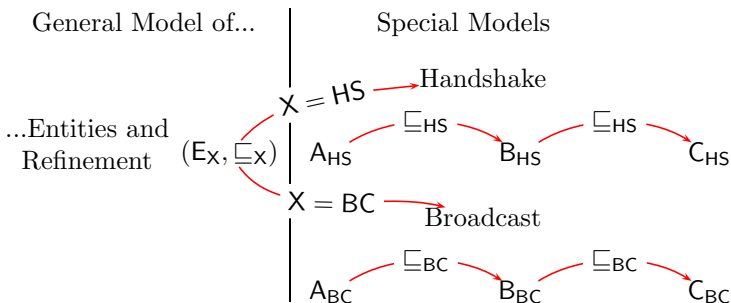
Formal Methods
straight jacket
Example

Lax Refinement

Lax Refinement in
two parts
General refinement
Vertical refinement/
Galois Connection
Example

Summary

- 1 General refinement (2003)
- 2 Vertical refinement



Lax = General + Vertical

Motivation

Formal Methods
straight jacket
Example

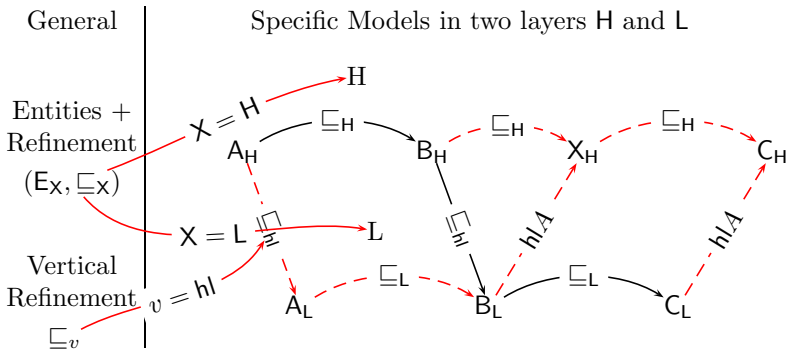
Lax Refinement

Lax Refinement in
two parts
General refinement
Vertical refinement/
Galois Connection
Example

Summary

1 General refinement (2003)

2 Vertical refinement



Galois Connections are theory morphism

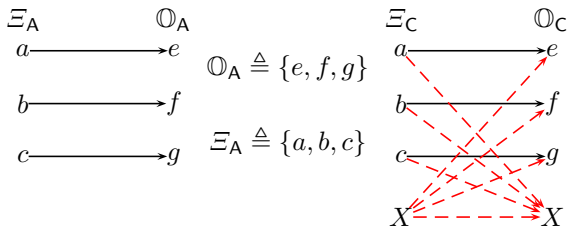
- Different kinds of entities have different theories
- Galois connections interpret one theory in another and vice versa
- A Galois connection is a pair of functions: $\llbracket _ \rrbracket_V^{HL}$ one from the high- to low-level theory the other $\nu A^{HL} _$ from the low- to high-level theory.

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

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

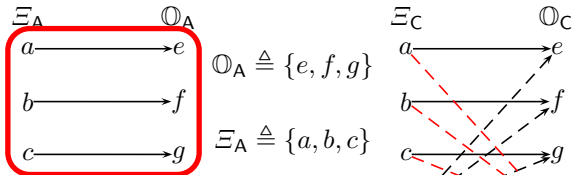
Subset morphism

- 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 $A \sqsubseteq_{sub} C$



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 $A \sqsubseteq_{sub} C$



Where are we?

Motivation

Formal Methods
straight jacket
Example

Lax Refinement

Lax Refinement in
two parts
General refinement
Vertical refinement/
Galois Connection
Example

Summary

1 Motivation
Formal Methods straight jacket
Example

2 Lax Refinement
Lax Refinement in two parts
General refinement
Vertical refinement/ Galois Connection
Example

Motivation

Formal Methods
straight jacket
Example

Lax
Refinement

Lax Refinement in
two parts
General refinement
Vertical refinement/
Galois Connection
Example

Summary

From Set_A to Set_C in three steps

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

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

Step 1 from Set_A to Set_X $\{X\}$ is outside the frame

Set_A

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 Set_X to Set_B General Refinement

Set_B

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 Set_B to 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_C

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

Summary

- Tools in two parts decouples semantics analysis from syntax.
- General refinement factors out a common theory
- Vertical refinement liberalises refinement
- lax refinement = general + vertical refinement
- Much of what once needed retrenchment can now be performed by lax refinement

- To Do
 - Build a tool similar to CSPproover
 - Build Denotational semantics as metric spaces