

Contexts and Observations

As a Bridge between Formal Models

Steve Reeves and David Streader

{stever,dstr}@cs.waikato.ac.nz.

Department of Computer Science

University of Waikato

Hamilton, New Zealand

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines
- Operational semantics.

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines
- Operational semantics.
- Verification and Validation

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines
- Operational semantics.
- Verification and Validation
- Contexts and observations are parameters to an abstract theory of interaction. Fix the contexts and observations fix a the theory.

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines
- Operational semantics.
- Verification and Validation
- Contexts and observations are parameters to an abstract theory of interaction. Fix the contexts and observations fix a the theory.
- Morphisms between theories.

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines
- Operational semantics.
- Verification and Validation
- Contexts and observations are parameters to an abstract theory of interaction. Fix the contexts and observations fix a the theory.
- Morphisms between theories.
- What we have learnt about particular theories.

Overview

- This talk discusses on going work to build a formal and flexible definition of refinement to encompass as much of software development as we can.
- Entities are Operations or Machines
- Operational semantics.
- Verification and Validation
- Contexts and observations are parameters to an abstract theory of interaction. Fix the contexts and observations fix a the theory.
- Morphisms between theories.
- What we have learnt about particular theories.
- Conclusion

Operational semantics

- We develop a general theory of entities. An entity can be a single operation or a machine consisting of a set of operations. (process, object, an ADT or a (Event-)B machine).

Operational semantics

- We develop a general theory of entities. An entity can be a single operation or a machine consisting of a set of operations. (process, object, an ADT or a (Event-)B machine).
- One relation/ LTS have many interpretations.

Operational semantics

- We develop a general theory of entities. An entity can be a single operation or a machine consisting of a set of operations. (process, object, an ADT or a (Event-)B machine).
- One relation/ LTS have many interpretations.
- An interpretation can be formalised by a refinement relation.

Operational semantics

- We develop a general theory of entities. An entity can be a single operation or a machine consisting of a set of operations. (process, object, an ADT or a (Event-)B machine).
- One relation/ LTS have many interpretations.
- An interpretation can be formalised by a refinement relation.
- Partial relations can be interpreted as partially or totally correct.

Operational semantics

- We develop a general theory of entities. An entity can be a single operation or a machine consisting of a set of operations. (process, object, an ADT or a (Event-)B machine).
- One relation/ LTS have many interpretations.
- An interpretation can be formalised by a refinement relation.
- Partial relations can be interpreted as partially or totally correct.
- An LTS can represent handshake, broadcast or method-calling actions.

Operational semantics

Operational semantics as a common ground for state- and event-based theories. (of Entities)

Operational semantics

Operational semantics as a common ground for state- and event-based theories. (of Machines)

B
Event B
Z

Undef
Gard

$O \rightarrow Pr$

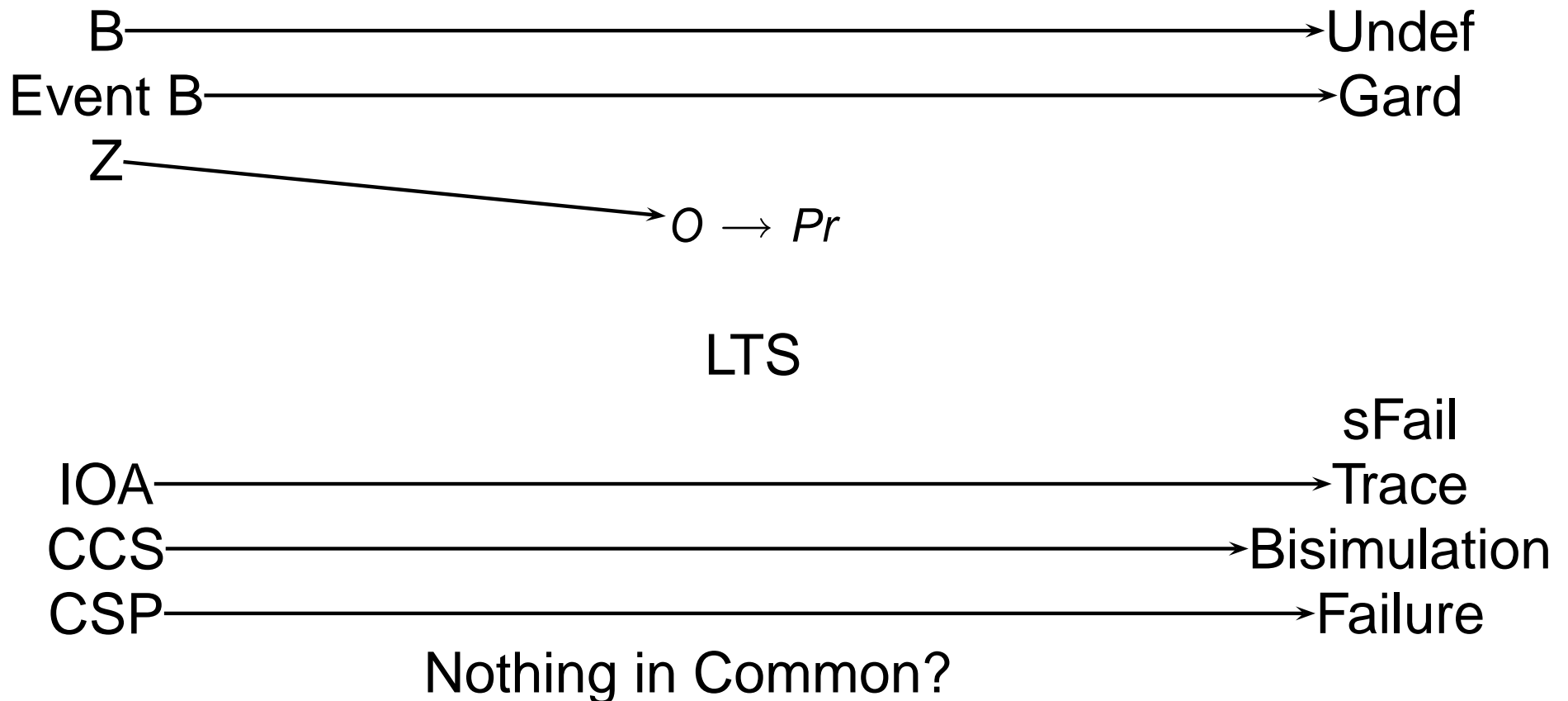
LTS

IOA
CCS
CSP

sFail
Trace
Bisimulation
Failure

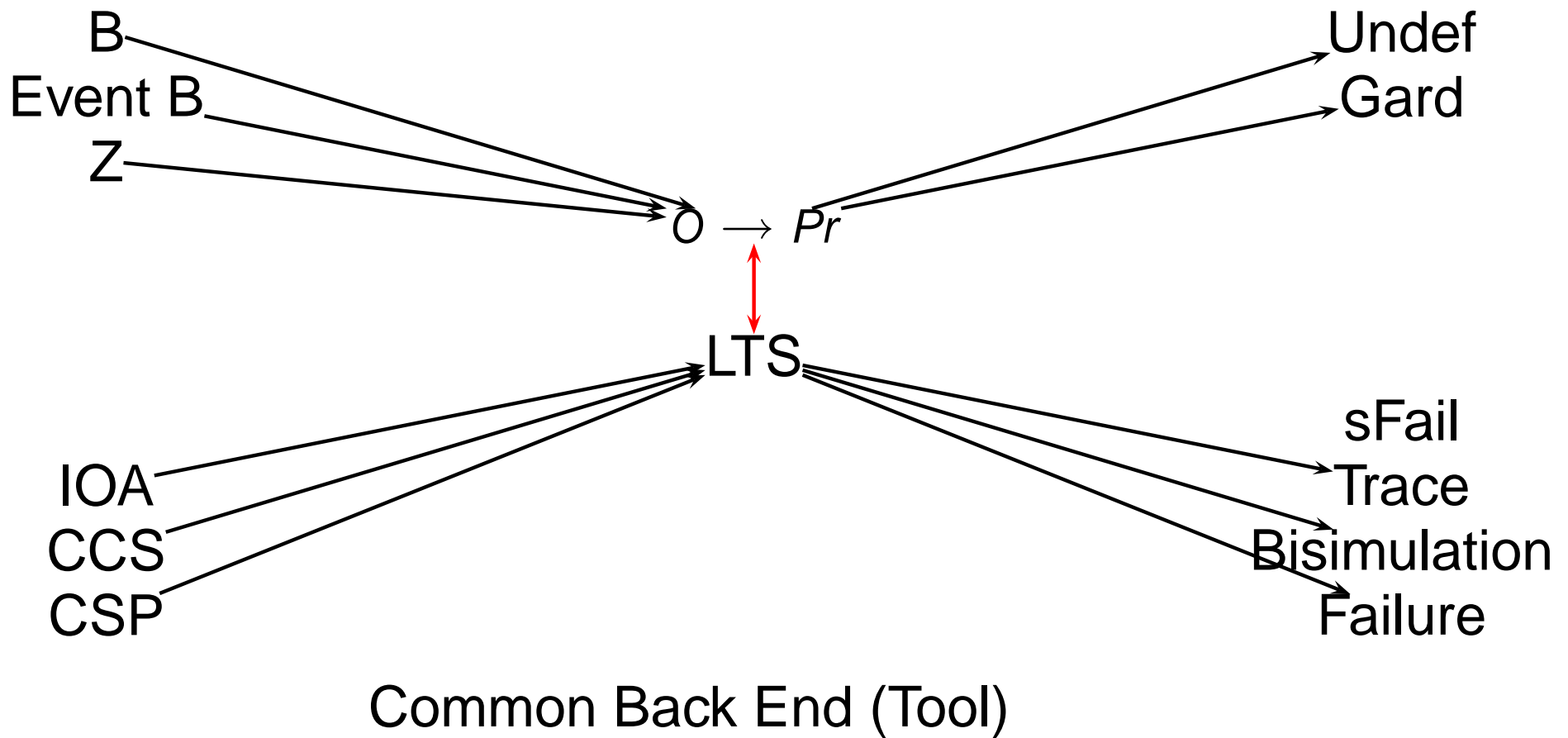
Operational semantics

Operational semantics as a common ground for state- and event-based theories. (of Machines)



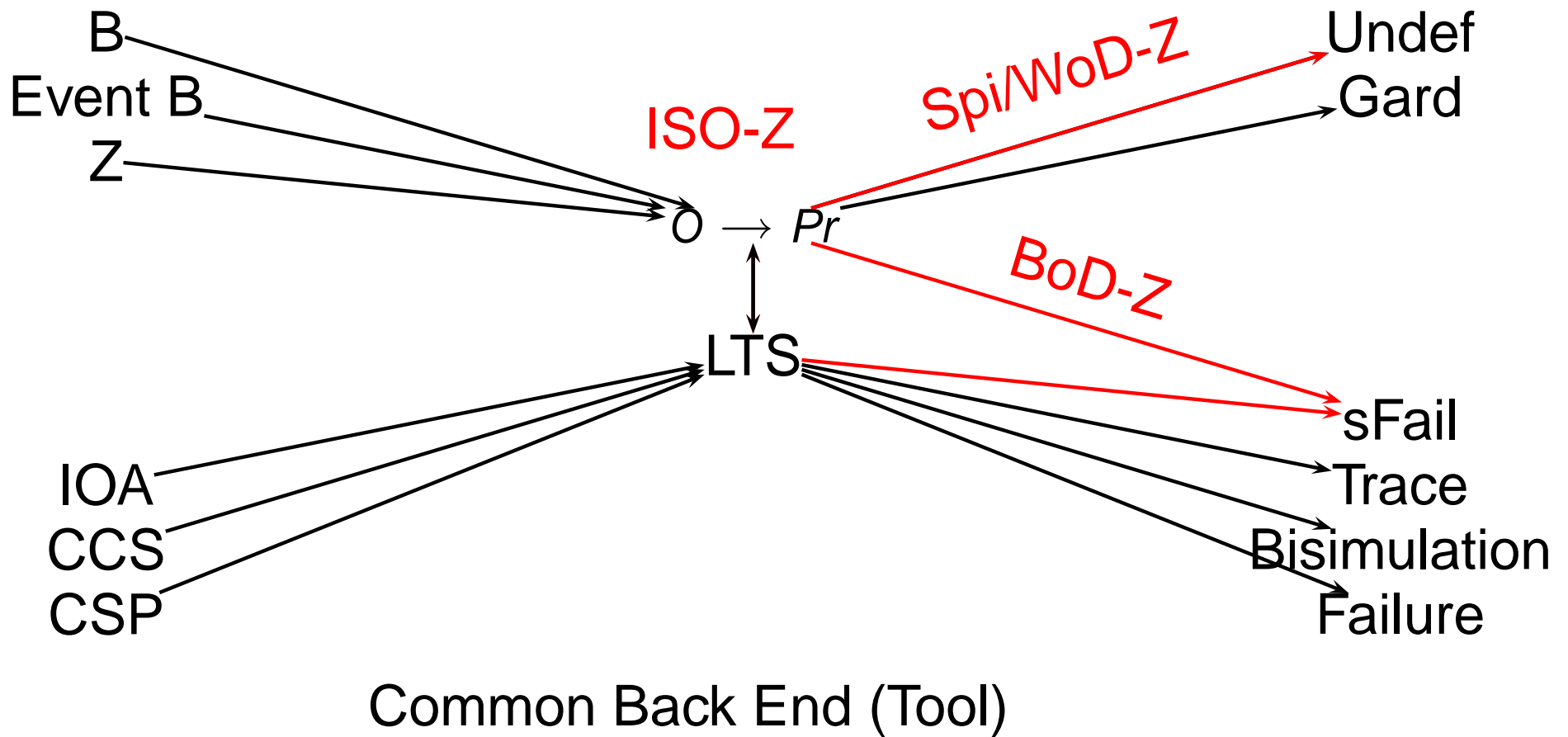
Operational semantics

Operational semantics as a common ground for state- and event-based theories. (of Machines)

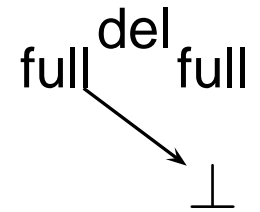
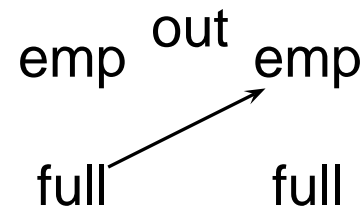
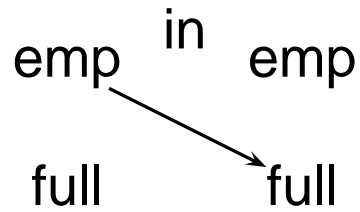
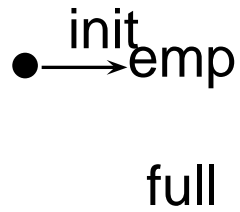


Operational semantics

Operational semantics as a common ground for state- and event-based theories. (of Machines)

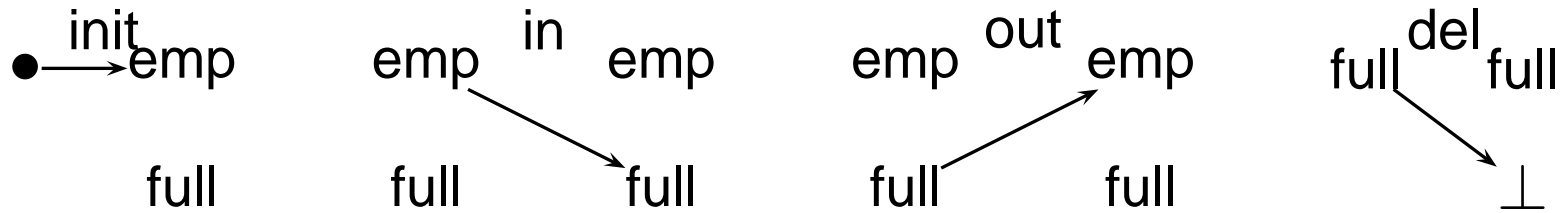


LTS and $O \rightarrow Pr$

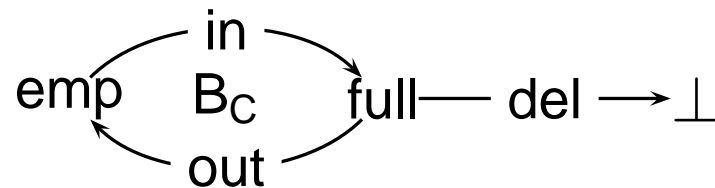
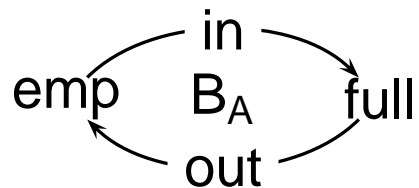


$$Alp_{BA} \triangleq \{in, out\}$$

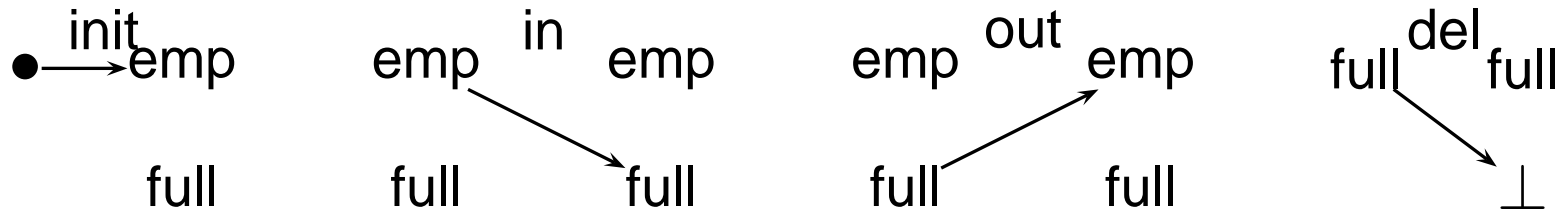
LTS and $O \rightarrow Pr$



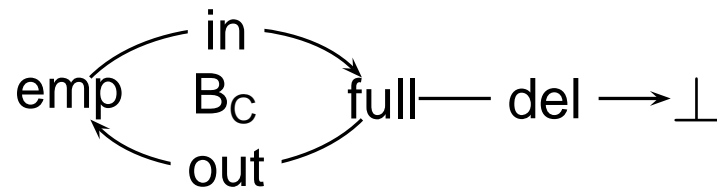
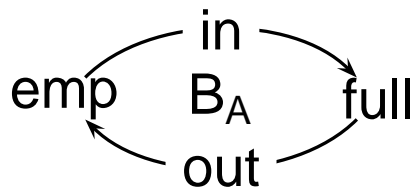
$$Alp_{B_A} \triangleq \{in, out\}$$



LTS and $O \rightarrow Pr$



$$Alp_{B_A} \triangleq \{in, out\}$$



$$Npr_A \triangleq \{(n, R) \mid n \in Alp_A \wedge (x, y) \in R \Leftrightarrow x \xrightarrow{n} y\}$$

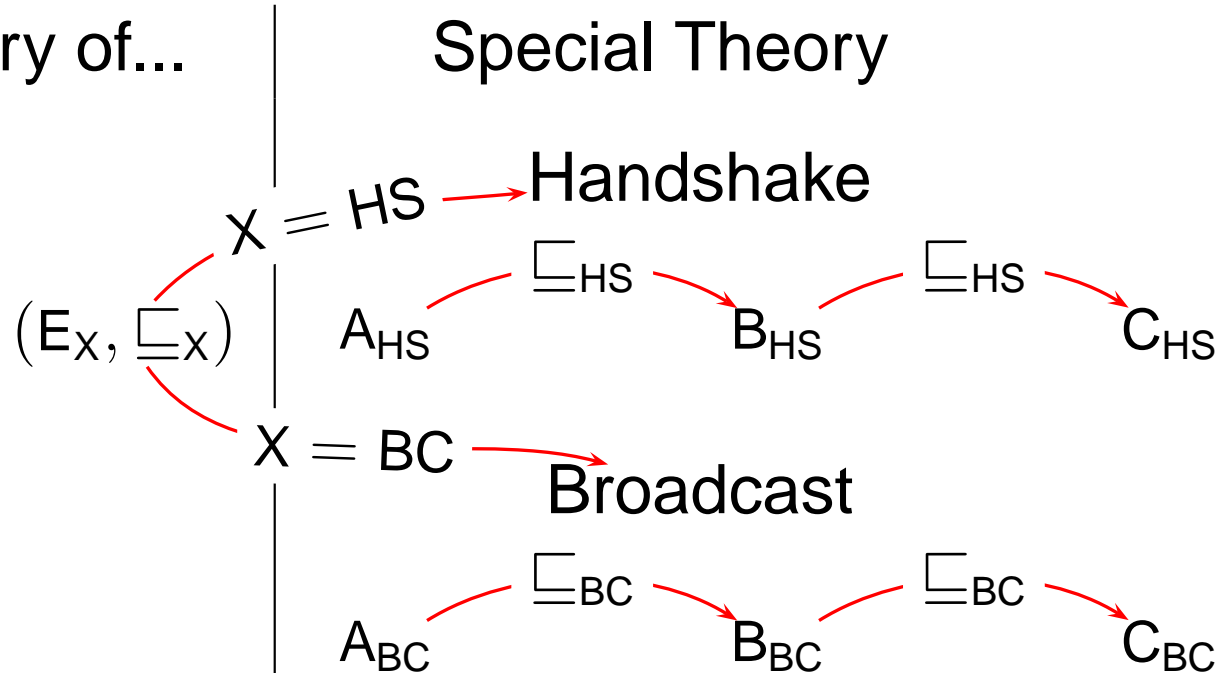
$$T_A \triangleq \{(x, n, y) \mid (n, R) \in Npr_A \wedge (x, y) \in R\}$$

General Theory

General Theory of...

Special Theory

...Entities and
Refinement



General Theory

General
Model of ...

Two Specific Models in two layers

... Entities
(E_X, \sqsubseteq_X)

$X = \text{IBP}$

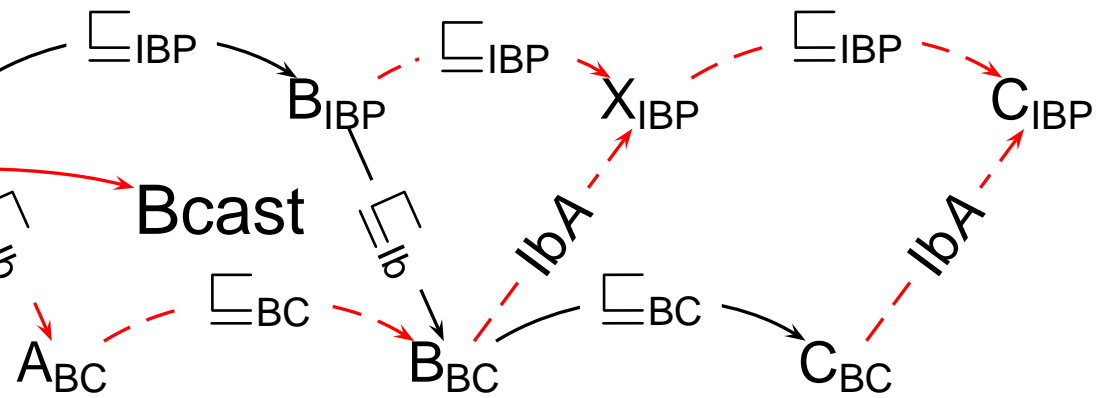
IBP

$X = \text{BC}$

Bcast

$\sqsubseteq_v = (\sqcup_v, vA)$
... Theory
Morphism

$v = \text{Ib}$



General Theory

In the General Theory we define:

1. Refinement $\sqsubseteq_{(\Xi, \mathcal{O})}$ contexts- Ξ , observations- \mathcal{O}
2. Relational theory/ Testing theory
3. Theory morphisms
4. Determinism/Specification/Implementation
5. On the operational semantics of Machines
 - (a) $- + -$; $-$ $- \parallel -$
 - (b) Abstraction or Hiding $(-\backslash \mathcal{T})$
 - (c) Use Cases/Simulation (HHS)

General Theory

In the General Theory we define:

1. Refinement $\sqsubseteq_{(\Xi, O)}$ contexts- Ξ , observations- O
2. Relational theory/ Testing theory
3. Theory morphisms
4. Determinism/Specification/Implementation
5. On the operational semantics of Machines
 - (a) $- + -$; $-$ \parallel $-$
 - (b) Abstraction or Hiding $(-\backslash \mathcal{T})$
 - (c) Use Cases/Simulation (HHS)

We can use the general theory both to compare and to combine special theories.

Refinement $A \sqsubseteq C$

1. Any user of A will not be able to observe if they had actually been given C in its place

Refinement $A \sqsubseteq C$

1. Any user of A will not be able to observe if they had actually been given C in its place
2. We first place an entity A in a context $x \in \Xi$ written $[A]_x$. Let A and x interact **privately**. Observe them both O .

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

Refinement $A \sqsubseteq C$

1. Any user of A will not be able to observe if they had actually been given C in its place
2. We first place an entity A in a context $x \in \Xi$ written $[A]_x$. Let A and x interact **privately**. Observe them both O .

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

3. This abstract definition of refinement can be specialised by fixing both Ξ and O .

Refinement $A \sqsubseteq C$

1. Any user of A will not be able to observe if they had actually been given C in its place
2. We first place an entity A in a context $x \in \Xi$ written $[A]_x$. Let A and x interact **privately**. Observe them both O .

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

3. This abstract definition of refinement can be specialised by fixing both Ξ and O .
4. We have defined (Ξ, O) pairs so that the $\sqsubseteq_{(\Xi, O)}$ refinement is well known. Failure, singleton Failure, Abstract Data Type, Broadcast.

Refinement – testing – theory

1. Which of the very many Semantics should we use?

Refinement – testing – theory

1. Which of the very many Semantics should we use?
2. Our definition of Refinement is a Testing Semantics.

Refinement – testing – theory

1. Which of the very many Semantics should we use?
2. Our definition of Refinement is a Testing Semantics.
3. From our definition of Refinement we can define a relational semantics:

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

Refinement – testing – theory

1. Which of the very many Semantics should we use?
2. Our definition of Refinement is a Testing Semantics.
3. From our definition of Refinement we can define a relational semantics:

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

4. These are not the relations from UTP.

Refinement – testing – theory

1. Which of the very many Semantics should we use?
2. Our definition of Refinement is a Testing Semantics.
3. From our definition of Refinement we can define a relational semantics:

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

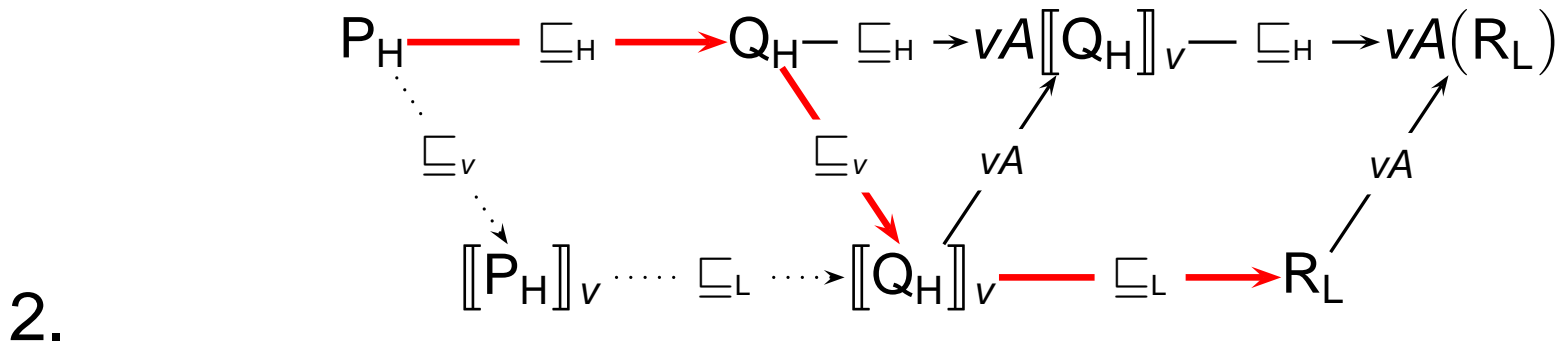
4. These are not the relations from UTP.
5. Refinement becomes implication in a logical theory based on giving entities a $\Xi \times O$ relational semantics.

Theory morphisms/ Galois connections

1. Semantic mapping $\llbracket - \rrbracket_v$ *interprets*, high-level E_H entities as low-level entities E_L and vA *interprets*, low-level entities as high-level entities.

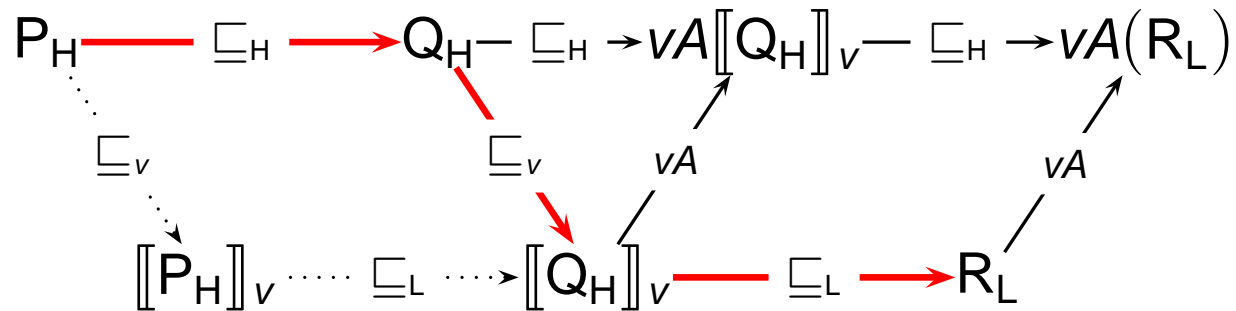
Theory morphisms/ Galois connections

1. Semantic mapping $\llbracket - \rrbracket_v$ interprets, high-level E_H entities as low-level entities E_L and vA interprets, low-level entities as high-level entities.



Theory morphisms/ Galois connections

- Semantic mapping $\llbracket - \rrbracket_v$ interprets, high-level E_H entities as low-level entities E_L and vA interprets, low-level entities as high-level entities.



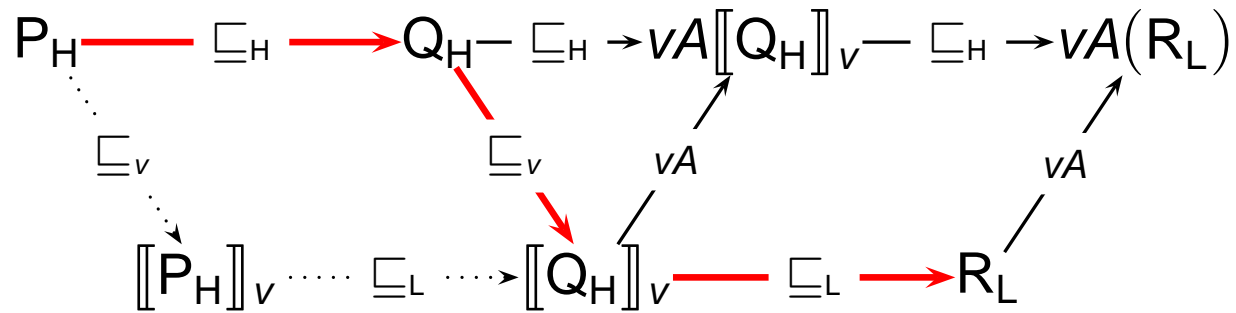
2.

3.

$$\forall P_H, R_L. \llbracket P_H \rrbracket_v^{HL} \sqsubseteq_L R_L \Leftrightarrow P_H \sqsubseteq_H vA^{HL}(R_L)$$

Theory morphisms/ Galois connections

- Semantic mapping $\llbracket - \rrbracket_v$ interprets, high-level E_H entities as low-level entities E_L and vA interprets, low-level entities as high-level entities.



2.

3.

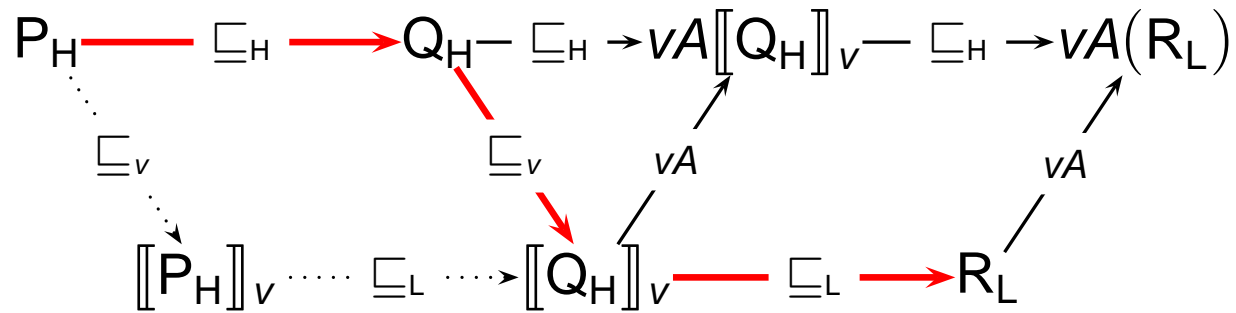
$$\forall P_H, R_L. \llbracket P_H \rrbracket_v^{HL} \sqsubseteq_L R_L \Leftrightarrow P_H \sqsubseteq_H vA^{HL}(R_L)$$

4.

$$\sqsubseteq_H \triangleq \sqsubseteq_{\Xi_H, O_H} \quad \llbracket \sqsubseteq_H \rrbracket_v \triangleq \sqsubseteq_{\llbracket \Xi_H \rrbracket_v, \llbracket O_H \rrbracket_v}$$

Theory morphisms/ Galois connections

1. Semantic mapping $\llbracket - \rrbracket_v$ interprets, high-level E_H entities as low-level entities E_L and vA interprets, low-level entities as high-level entities.



2.

3.

$$\forall P_H, R_L. \llbracket P_H \rrbracket_v^{HL} \sqsubseteq_L R_L \Leftrightarrow P_H \sqsubseteq_H vA^{HL}(R_L)$$

4.

$$\sqsubseteq_H \triangleq \sqsubseteq_{\Xi_H, O_H} \quad \llbracket \sqsubseteq_H \rrbracket_v \triangleq \sqsubseteq_{\llbracket \Xi_H \rrbracket_v, \llbracket O_H \rrbracket_v}$$

5. $H \sqsubseteq_v^{HL} L$ guarantees that the high-level vA -interpretation of entity L behaves like entity H when in context $x \in \Xi_H$ and only O_H observations are made.

Subset Morphisms

1. Subset of relational semantics

$$\Xi_A \times O_A \subseteq \Xi_C \times O_C$$

- (a) $\llbracket - \rrbracket_{(\Xi_C \setminus \Xi_A, O_C \setminus O_A)}$ embeds the abstract in the concrete
- (b) $\nu A_{(\Xi_C \setminus \Xi_A, O_C \setminus O_A)}$ projects back to the abstract

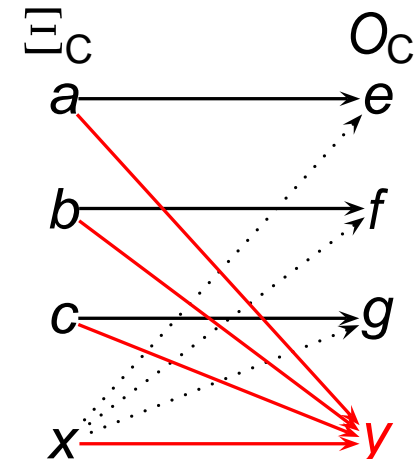
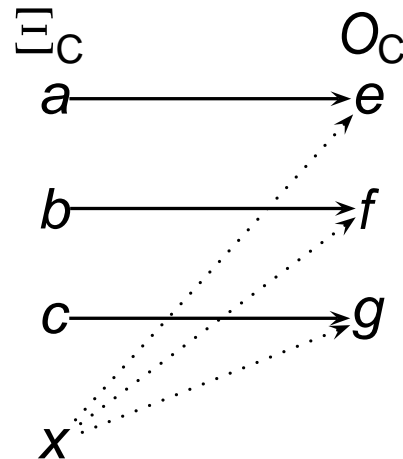
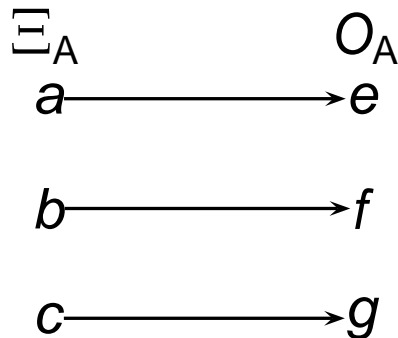
Subset Morphisms

1. Subset of relational semantics

$$\Xi_A \times O_A \subseteq \Xi_C \times O_C$$

(a) $\llbracket - \rrbracket_{(\Xi_C \setminus \Xi_A, O_C \setminus O_A)}$ embeds the abstract in the concrete

(b) $\nu A_{(\Xi_C \setminus \Xi_A, O_C \setminus O_A)}$ projects back to the abstract



2.

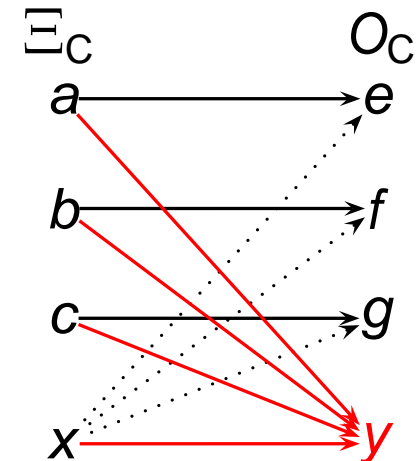
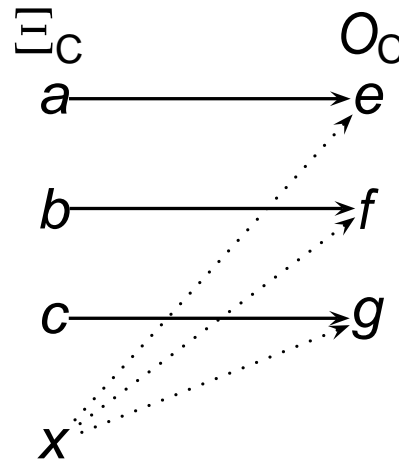
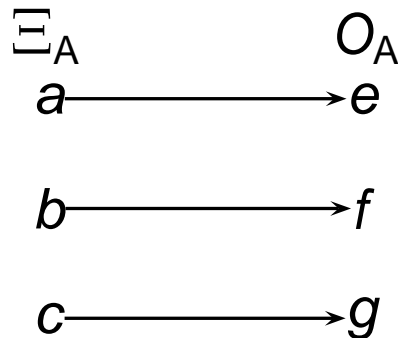
Subset Morphisms

1. Subset of relational semantics

$$\Xi_A \times O_A \subseteq \Xi_C \times O_C$$

(a) $\llbracket - \rrbracket_{(\Xi_C \setminus \Xi_A, O_C \setminus O_A)}$ embeds the abstract in the concrete

(b) $\nu A_{(\Xi_C \setminus \Xi_A, O_C \setminus O_A)}$ projects back to the abstract



2.

3. What is unusual is the addition of **new observations**

What have we found out?

- Data refinement (guarded ops) is not sF refinement

What have we found out?

- Data refinement (guarded ops) is not sF refinement
- Refinement can add Features or fix interpretations:
 - ISO-Z fixed to WoD-Z or BoD-Z
 - add new operations
 - mix operations of different interaction style
 - fix size of data type
 - change atomic operations into value passing

What have we found out?

- Data refinement (guarded ops) is not sF refinement
- Refinement can add Features or fix interpretations:
 - ISO-Z fixed to WoD-Z or BoD-Z
 - add new operations
 - mix operations of different interaction style
 - fix size of data type
 - change atomic operations into value passing
- Much of retrenchment can be done by refinement

What have we found out?

- Data refinement (guarded ops) is not sF refinement
- Refinement can add Features or fix interpretations:
 - ISO-Z fixed to WoD-Z or BoD-Z
 - add new operations
 - mix operations of different interaction style
 - fix size of data type
 - change atomic operations into value passing
- Much of retrenchment can be done by refinement
- Implement one style of interaction using another style of interaction

What have we found out?

- Data refinement (guarded ops) is not sF refinement
- Refinement can add Features or fix interpretations:
 - ISO-Z fixed to WoD-Z or BoD-Z
 - add new operations
 - mix operations of different interaction style
 - fix size of data type
 - change atomic operations into value passing
- Much of retrenchment can be done by refinement
- Implement one style of interaction using another style of interaction
- Handshake process algebras (CSP, CCS...) differ from most models in how they define determinism

Deterministic Entities

- The behaviour of a deterministic entity in a deterministic context is determined.

Deterministic Entities

- The behaviour of a deterministic entity in a deterministic context is determined.
- Let Ξ_D be the set deterministic contexts:
D is deterministic iff $\llbracket D \rrbracket_{\Xi_D, O}$ is a function.

Deterministic Entities

- The behaviour of a deterministic entity in a deterministic context is determined.
- Let Ξ_D be the set deterministic contexts:
D is deterministic iff $\llbracket D \rrbracket_{\Xi_D, O}$ is a function.
- I is implementable iff it is deterministic

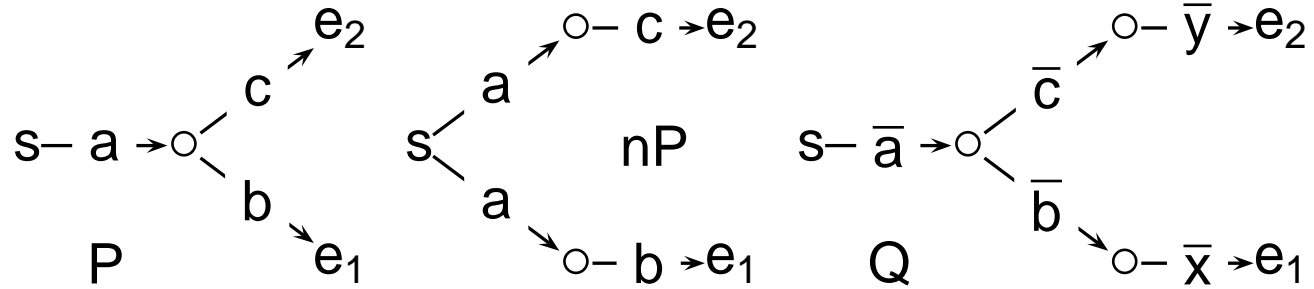
Deterministic Entities

- The behaviour of a deterministic entity in a deterministic context is determined.
- Let Ξ_D be the set deterministic contexts:
D is deterministic iff $\llbracket D \rrbracket_{\Xi_D, O}$ is a function.
- I is implementable iff it is deterministic
- These definitions in special theories:
 - Operations - as expected
 - ADT (B machines) - as expected
 - Broadcast processes - as in CBS Prassad
 - Handshake processes - **Not as expected**

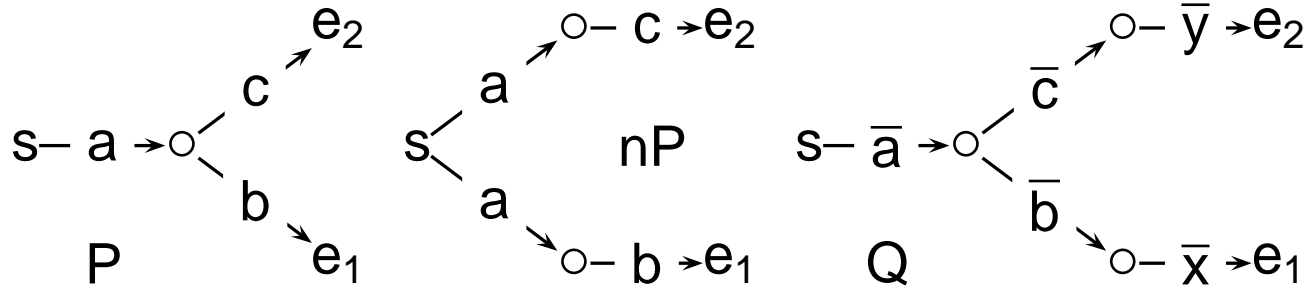
Determinism and CSP/CCS/ACP

- Determinism as defined: $n \xrightarrow{a} x \wedge n \xrightarrow{a} y \Rightarrow x = y$

Determinism and CSP/CCS/ACP



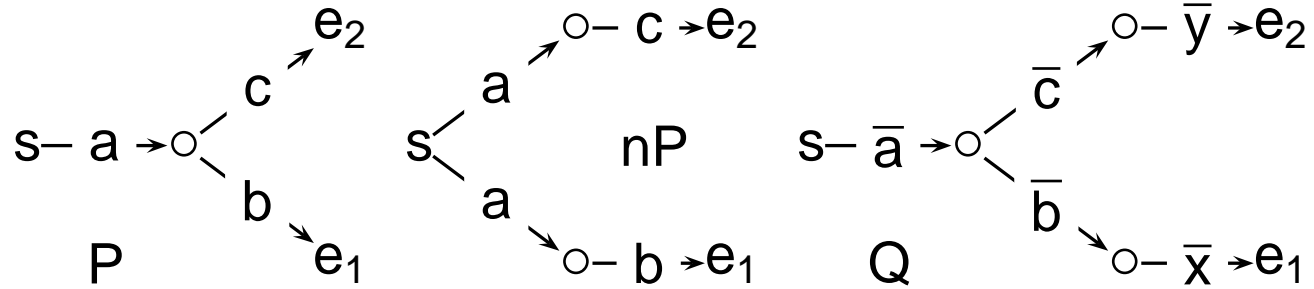
Determinism and CSP/CCS/ACP




Hoare's view

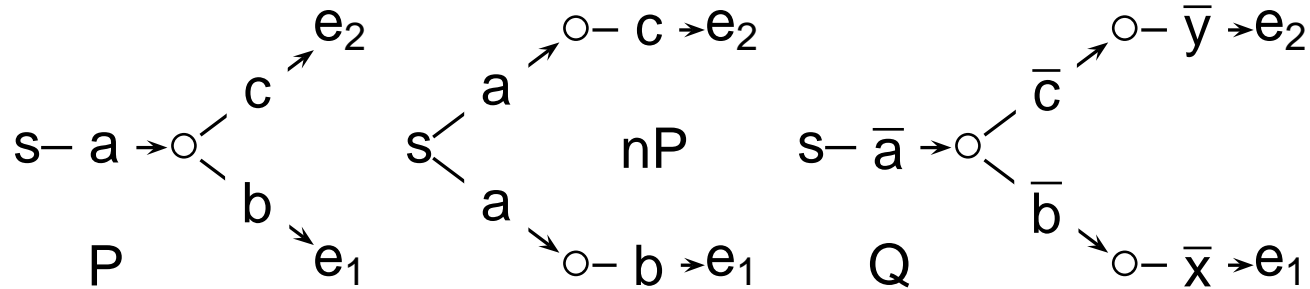
“nondeterminism: it arises from a deliberate decision to ignore the factors which influence the selection”

Determinism and CSP/CCS/ACP




CSP $\gamma(a, a) = a$ and **CCS** $\gamma(a, \bar{a}) = \tau$

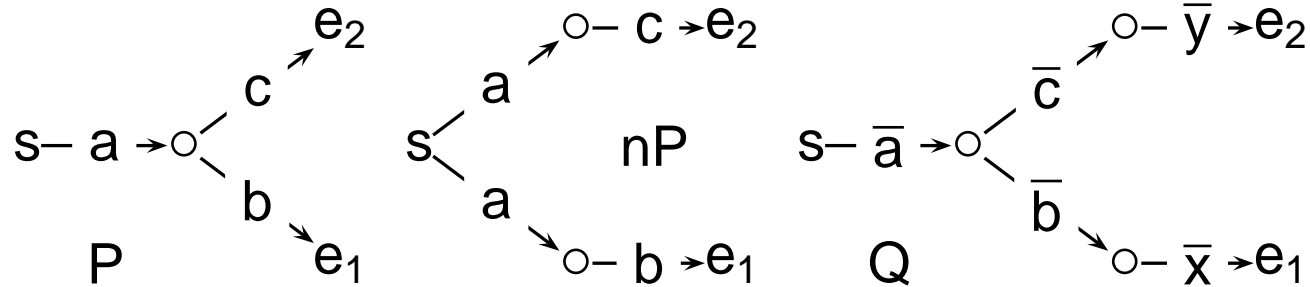
Determinism and CSP/CCS/ACP



● Milner's comment about determinism:

“Whatever its precise definition, it certainly must have a lot to do with predictability; if we perform the same experiment twice on a determinate system – starting each time in its initial state – then we expect to get the same result, or behaviour, each time.”

Determinism and CSP/CCS/ACP



● Milner's comment about determinism:

“Whatever its precise definition, it certainly must have a lot to do with predictability; if we perform the same experiment twice on a determinate system – starting each time in its initial state – then we expect to get the same result, or behaviour, each time.”



● If P is a valid experiment then Q is not deterministic!

Determinism and CSP/CCS/ACP

- Our definition is no more than a formalisation of Milner's comment

Determinism and CSP/CCS/ACP

- Our definition is no more than a formalisation of Milner's comment
- **Deterministic behaviour** Let ρ be a sequence of observable events

$$det_beh_{\Xi}(A) \triangleq n \xRightarrow{\rho} r \wedge n \xRightarrow{\rho} t \Rightarrow r =_{\Xi} t$$

Determinism and CSP/CCS/ACP

- Our definition is no more than a formalisation of Milner's comment
- **Deterministic behaviour** Let ρ be a sequence of observable events

$$det_beh_{\Xi}(A) \triangleq n \xrightarrow{\rho} r \wedge n \xrightarrow{\rho} t \Rightarrow r =_{\Xi} t$$



$$\forall A \in D. \forall x \in \Xi_D. det_beh_{\Xi_D}([A]_x)$$

Determinism and CSP/CCS/ACP

- Our definition is no more than a formalisation of Milner's comment
- **Deterministic behaviour** Let ρ be a sequence of observable events

$$det_beh_{\Xi}(A) \triangleq n \xrightarrow{\rho} r \wedge n \xrightarrow{\rho} t \Rightarrow r =_{\Xi} t$$



$$\forall A \in D. \forall x \in \Xi_D. det_beh_{\Xi_D}([A]_x)$$

- As entities and contexts are “the same” this is condition not a definition!

Determinism and CSP/CCS/ACP

- Our definition is no more than a formalisation of Milner's comment
- **Deterministic behaviour** Let ρ be a sequence of observable events

$$det_beh_{\Xi}(A) \triangleq n \xrightarrow{\rho} r \wedge n \xrightarrow{\rho} t \Rightarrow r =_{\Xi} t$$



$$\forall A \in D. \forall x \in \Xi_D. det_beh_{\Xi_D}([A]_x)$$

- As entities and contexts are “the same” this is condition not a definition!
- Only handshake theories CSP/CCS/... do not distinguish cause and effect in γ .

Vertical implementation

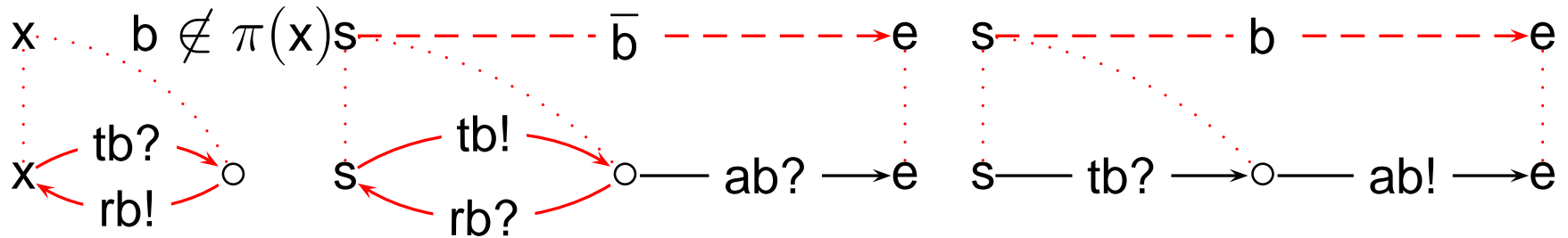
Theory morphisms can be used to “implement” one interaction style on another.

Vertical implementation

Implement ADTs on a Broadcast theory (layer)

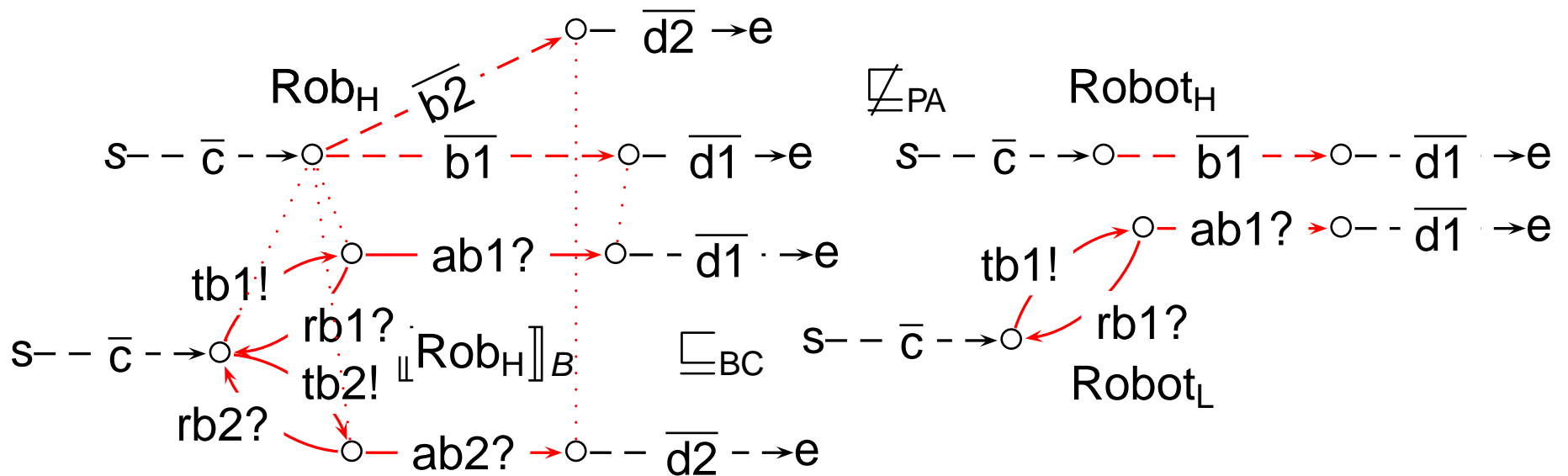
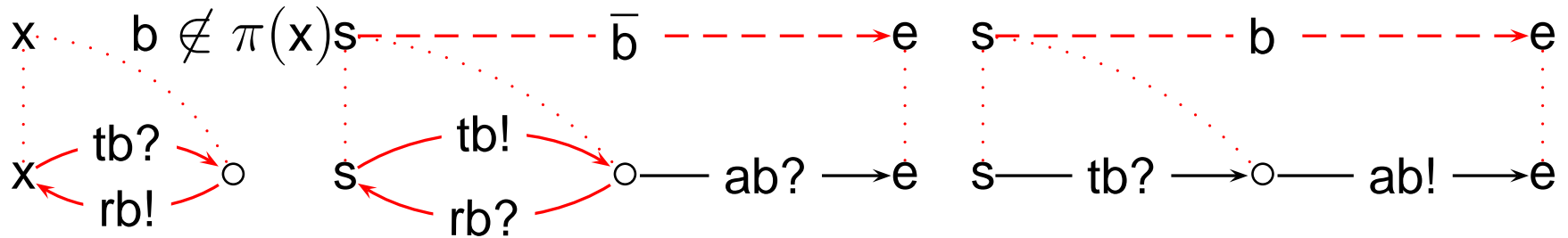
Vertical implementation

Implement ADTs on a Broadcast theory (layer)



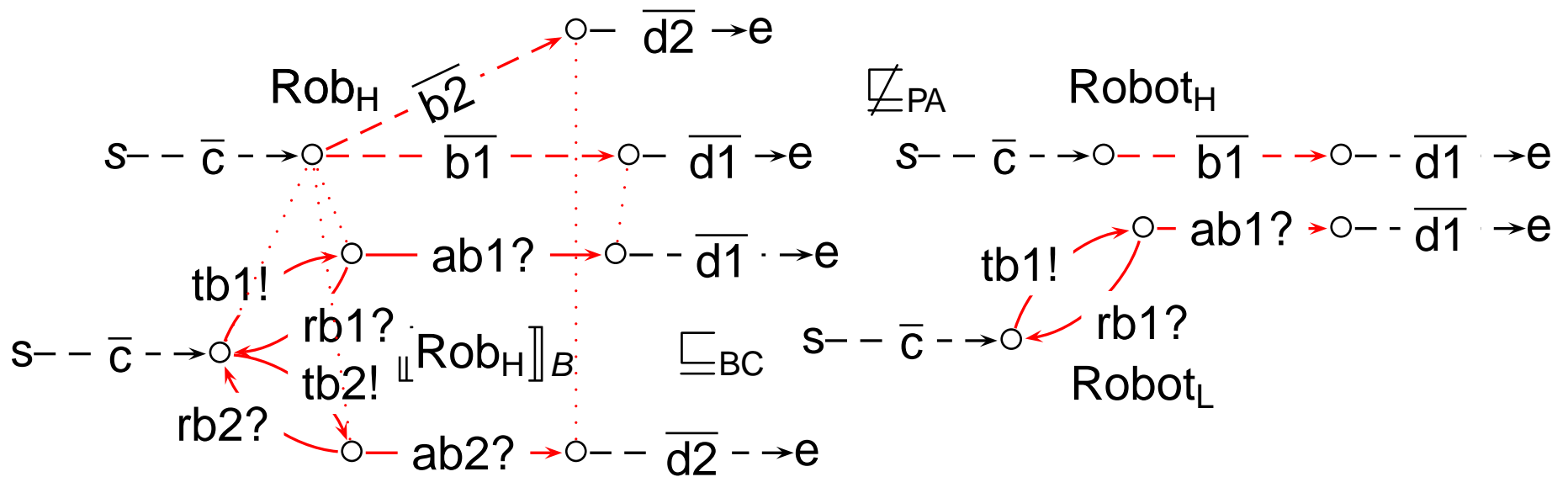
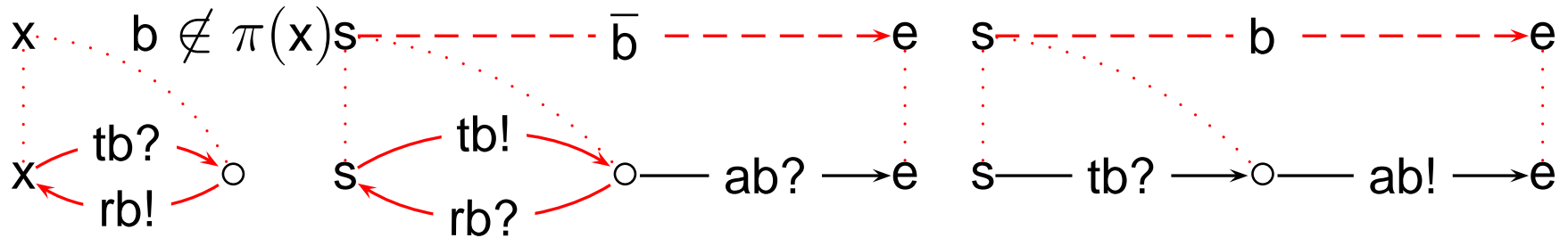
Vertical implementation

Implement ADTs on a Broadcast theory (layer)



Vertical implementation

Implement ADTs on a Broadcast theory (layer)



Operations

1. Partial relations have many interpretations.

Operations

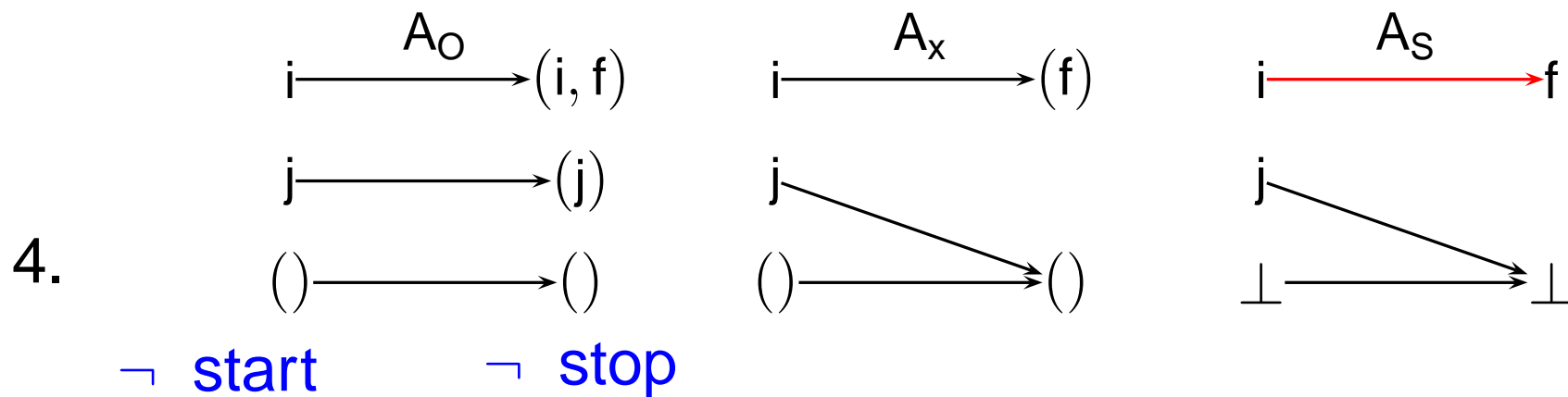
1. Partial relations have many interpretations.
2. The interpretation can be fixed by
 - (a) defining refinement and/or
 - (b) lifting and totalising the relations.

Operations

1. Partial relations have many interpretations.
2. The interpretation can be fixed by
 - (a) defining refinement and/or
 - (b) lifting and totalising the relations.
3. Observing individual states is not observing behavior
 $\{x = 1, x = 2, x' = 1, x' = 2\}$ observing a sequence of states is $\{(x = i, x' = f)\}$.

Operations

1. Partial relations have many interpretations.
2. The interpretation can be fixed by
 - (a) defining refinement and/or
 - (b) lifting and totalising the relations.
3. Observing individual states is not observing behavior
 $\{x = 1, x = 2, x' = 1, x' = 2\}$ observing a sequence of states is $\{(x = i, x' = f)\}$.



Operations ST

8 interpretations - 4 with \perp as Nontermination and 4 with \perp as Chaos

Operations ST

8 interpretations - 4 with \perp as Nontermination and 4 with \perp as Chaos

$$a \xrightarrow{[[U]]_{pu_}} a'$$

$$b \quad b'$$

$$\perp \quad \perp'$$

$$a \xrightarrow{[[U]]_{tu_}} a'$$

$$b \quad b'$$

$$\perp \quad \perp'$$

$$a \xrightarrow{[[U]]_{pg_}} a'$$

$$b \quad b'$$

$$\perp \quad \perp'$$

$$a \xrightarrow{[[U]]_{tg_}} a'$$

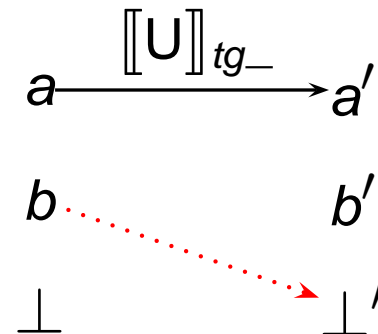
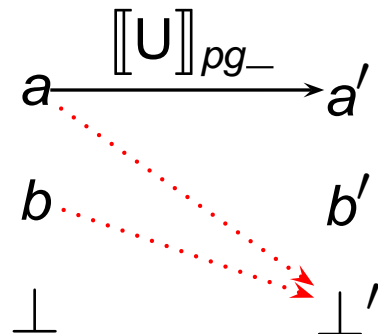
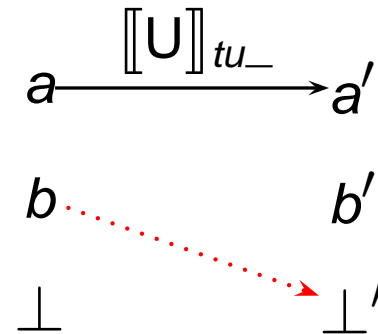
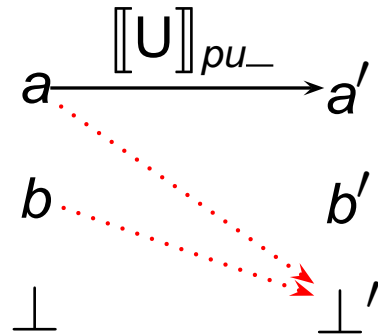
$$b \quad b'$$

$$\perp \quad \perp'$$

Partial relation

Operations ST

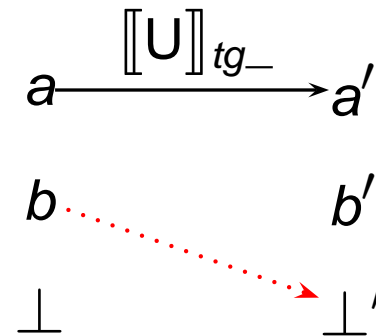
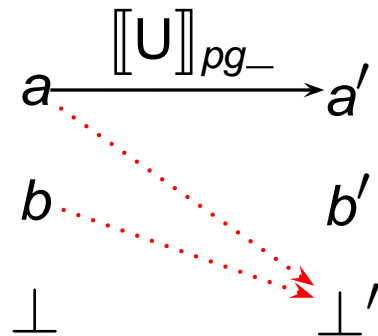
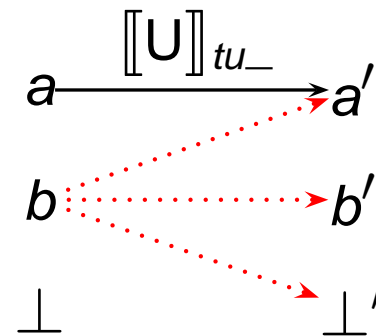
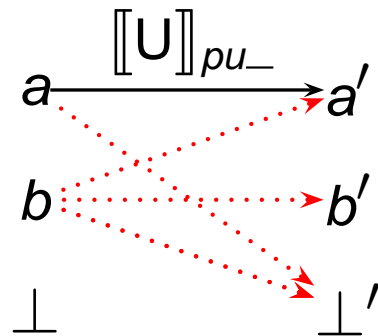
8 interpretations - 4 with \perp as Nontermination and 4 with \perp as Chaos



+ nontermination

Operations ST

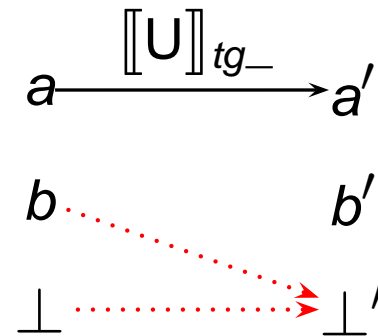
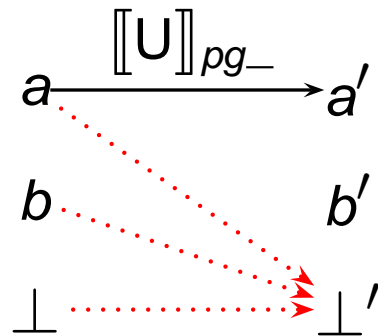
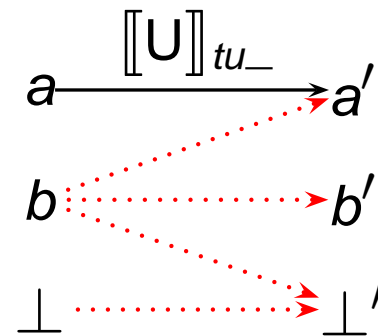
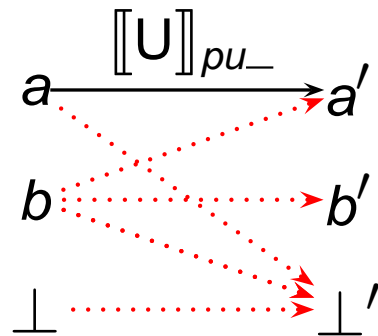
8 interpretations - 4 with \perp as Nontermination and 4 with \perp as Chaos



+ outside of precondition

Operations ST

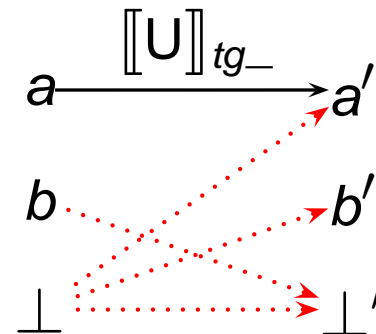
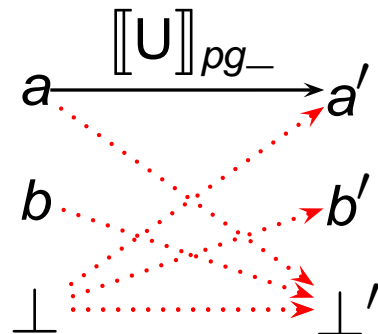
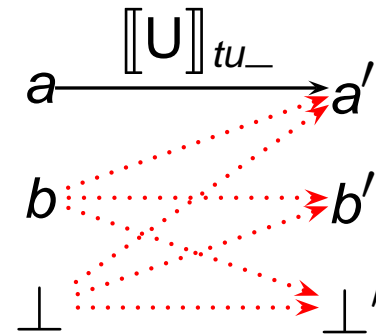
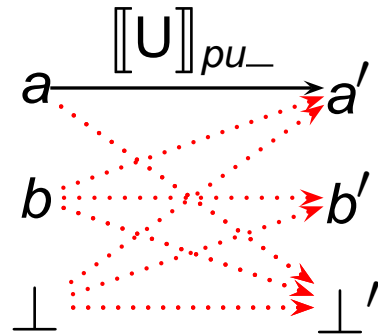
8 interpretations - 4 with \perp as Nontermination and 4 with \perp as Chaos



Strict lifting \perp as nontermination

Operations ST

8 interpretations - 4 with \perp as Nontermination and 4 with \perp as Chaos



Unstrict lifting \perp a chaos

Refineing interpretations

Lifting: introducing \perp is a subset morphism $\llbracket - \rrbracket_{(X,O)}$ where

$$\perp \in X \wedge \perp \in O$$

Refineing interpretations

Lifting: introducing \perp is a subset morphism $\llbracket - \rrbracket_{(X,O)}$ where $\perp \in X \wedge \perp \in O$

$$a \xrightarrow{\llbracket U \rrbracket} a' \quad \llbracket U \rrbracket_{(\{b, \perp\}, \{b, \perp\})} = a \xrightarrow{\llbracket U \rrbracket_{puc}} a'$$

Refineing interpretations

Lifting: introducing \perp is a subset morphism $\llbracket - \rrbracket_{(X,O)}$ where $\perp \in X \wedge \perp \in O$

$$a \xrightarrow{\llbracket U \rrbracket} a' \quad \llbracket U \rrbracket_{(\{b, \perp\}, \{b, \perp\})} = a \xrightarrow{\llbracket U \rrbracket_{puc}} a'$$

$$\llbracket U \rrbracket_{(\{b, \perp\}, \{b, \perp\})} = \llbracket U \rrbracket_{puc} \text{ or } U \sqsubseteq_{(\{b, \perp\}, \{b, \perp\})} = \llbracket U \rrbracket_{puc}$$

Refineing interpretations

Lifting: introducing \perp is a subset morphism $\llbracket - \rrbracket_{(X,O)}$ where $\perp \in X \wedge \perp \in O$

$$a \xrightarrow{\llbracket U \rrbracket} a' \quad \llbracket U \rrbracket_{(\{b,\perp\},\{b,\perp\})} = a \xrightarrow{\llbracket U \rrbracket_{puc}} a'$$

$$\llbracket U \rrbracket_{(\{b,\perp\},\{b,\perp\})} = \llbracket U \rrbracket_{puc} \text{ or } U \sqsubseteq_{(\{b,\perp\},\{b,\perp\})} = \llbracket U \rrbracket_{puc}$$

$$\llbracket U \rrbracket_{puc} \sqsubseteq \llbracket U \rrbracket_{pun} \sqsubseteq \llbracket U \rrbracket_{pgn} \sqsubseteq \llbracket U \rrbracket_{tgn}$$

Event based unobservable

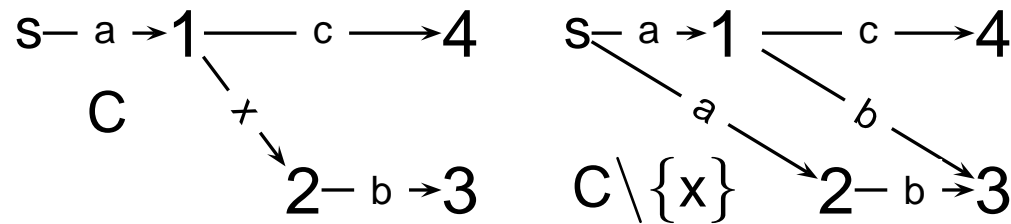
1. τ not seen or blocked - δ not seen or performed.

Event based unobservable

1. τ not seen or blocked - δ not seen or performed.
2. Removing δ operations is **Restriction** $_{-\delta_D}$.

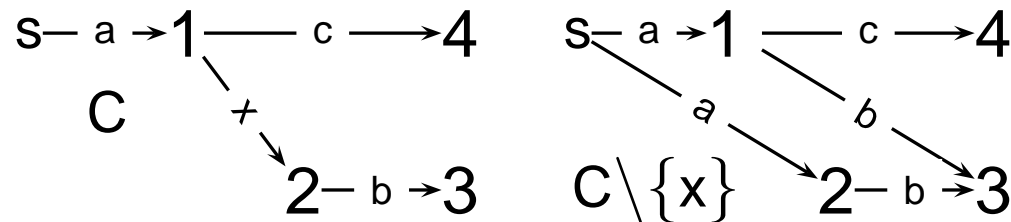
Event based unobservable

1. τ not seen or blocked - δ not seen or performed.
2. Removing δ operations is **Restriction** $_{-\delta_D}$.
3. Removing τ operations is **Hiding** or **Abstraction** $_{-\backslash T}$.



Event based unobservable

1. τ not seen or blocked - δ not seen or performed.
2. Removing δ operations is **Restriction** $_{-\delta_D}$.
3. Removing τ operations is **Hiding** or **Abstraction** $_{-\backslash T}$.

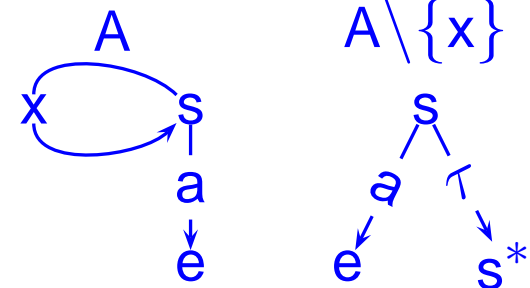


4. Three ways to hide τ -loops:

(a) Fair - CCS,ACP, Fair Failure

(b) Chaotic - CSP \Rightarrow **Eager Lazy**

(c) **Chaos free** CFFD,NDFD



Conclusion

- We have built a General Theory

Conclusion

- We have built a General Theory
- The General Theory can be specialised to known theories

Conclusion

- We have built a General Theory
- The General Theory can be specialised to known theories
- By implementing this General Theory as a tool we could have a common back end to many special tools