Testing Concurrent Conformance

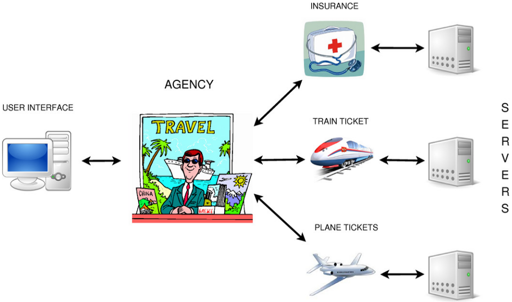
Hernán Ponce de León

INRIA and LSV, École Normale Supérieure de Cachan and CNRS, France

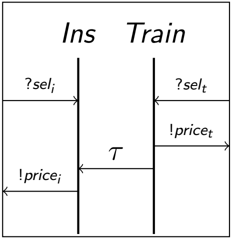
KOSMOS Workshop - November 2013

Testing concurrent systems

- Some actions are naturally concurrent (distributed systems)
- Interleaving may be artificial

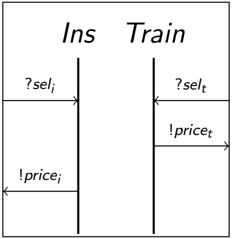


$$i \textbf{ ioco } s \Leftrightarrow \forall \sigma \in \text{traces}(s) : \text{out}(i \textbf{ after } \sigma) \subseteq \text{out}(s \textbf{ after } \sigma)$$



Imp

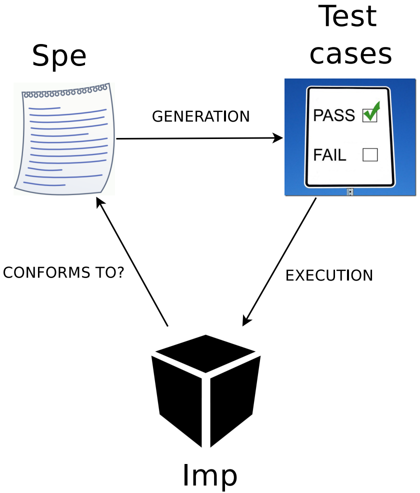
ioco



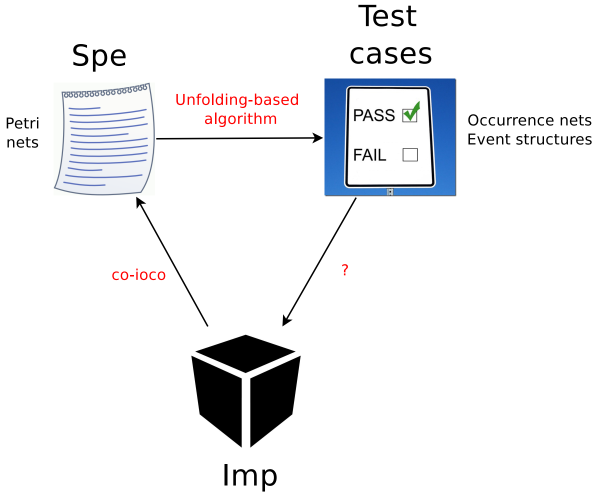
Spe

Concurrency is interpreted as interleavings

Formal Black Box Testing



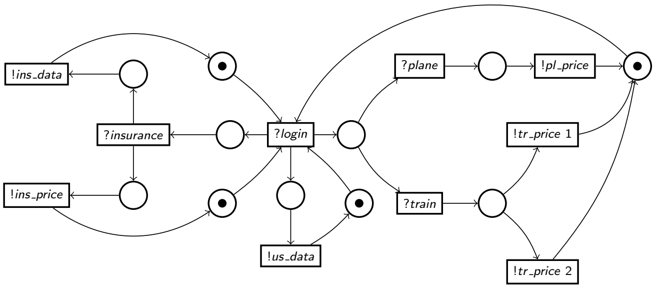
Formal Black Box Testing



Content

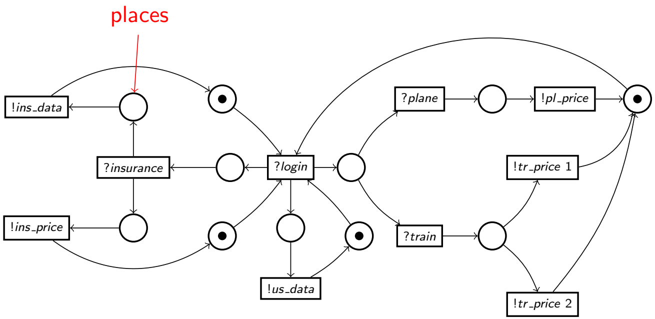
1. Model of the systems
   - Petri nets
   - Occurrence nets - Unfolding
   - Event Stuctures
2. Testing Framework
   - The conformance relation
   - Test suite
   - Test generation
3. Unsolved questions
   - Can we observe concurrency?
   - What is the meaning of concurrency?
   - Test execution
   - IICS sets

Petri nets



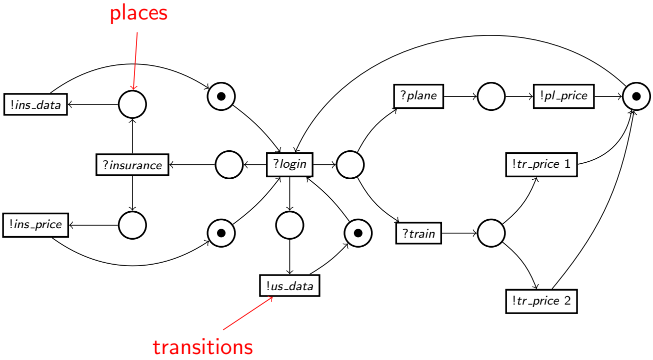
!ins_data
?insurance
!ins_price
?login
!us_data
?plane
!pl_price
!tr_price 1
?train
!tr_price 2

Petri nets

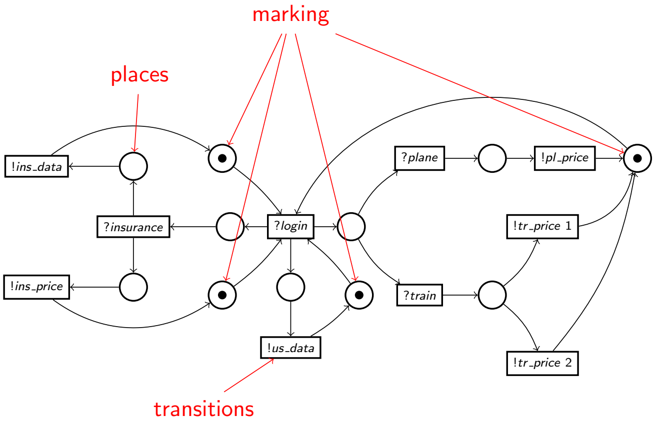


places
!ins_data
?insurance
!ins_price
?login
!us_data
?plane
!pl_price
?train
!tr_price 1
!tr_price 2

Petri nets



places
!ins_data
?insurance
!ins_price
?login
!us_data
?plane
!pl_price
!tr_price 1
?train
!tr_price 2
transitions



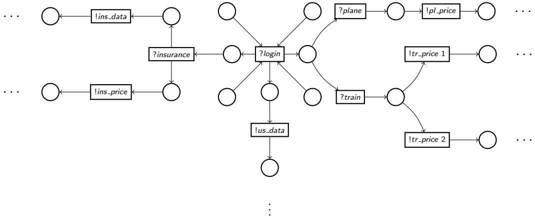
marking
places
!ins_data
?insurance
!ins_price
?login
!us_data
?plane
!pl_price
!tr_price 1
?train
!tr_price 2
transitions

Occurrence net - (Partial Order) Unfolding

Acyclic (possible infinite) Petri net that highlights conflict

places → conditions
transitions → events

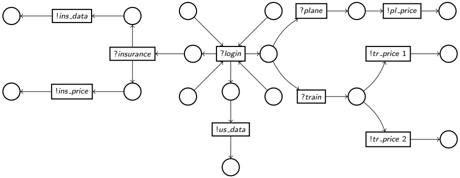
- only one arrow entering to conditions
- no self conflict

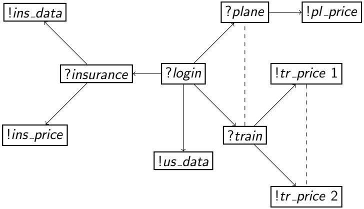


Event Structures

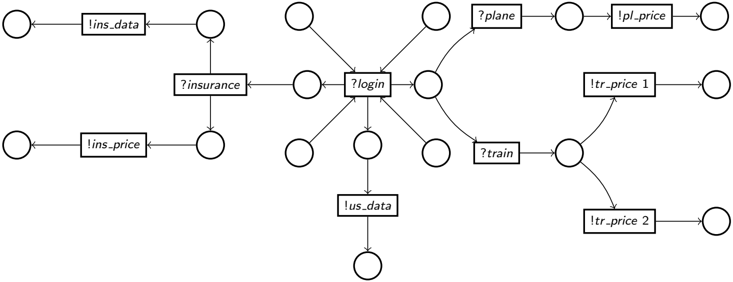
Set of events E equipped with

- causality: $\leq$
- conflict: $\#$ (inherited w.r.t $\leq$)
- concurrency: **co** (events not related by $\leq$ or $\#$)
- labeling: $\lambda : E \to \mathcal{I} \uplus \mathcal{O}$

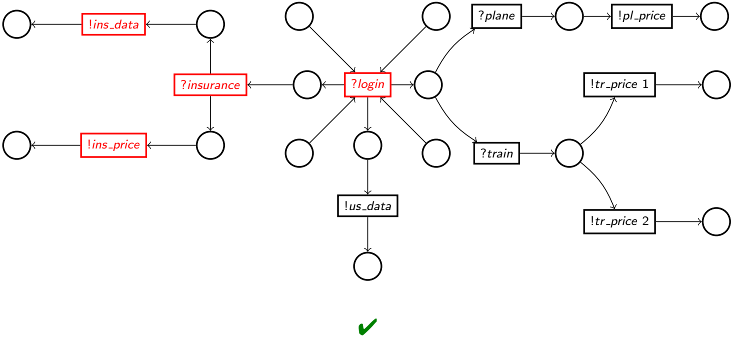

!ins_data
?insurance
!ins_price
?login
!us_data
?plane
!pl_price
?train
!tr_price 1
!tr_price 2


!ins_data
?insurance
!ins_price
?login
!us_data
?plane
!pl_price
?train
!tr_price 1
!tr_price 2

- Causally closed and conflict free set of events
- Represents executions

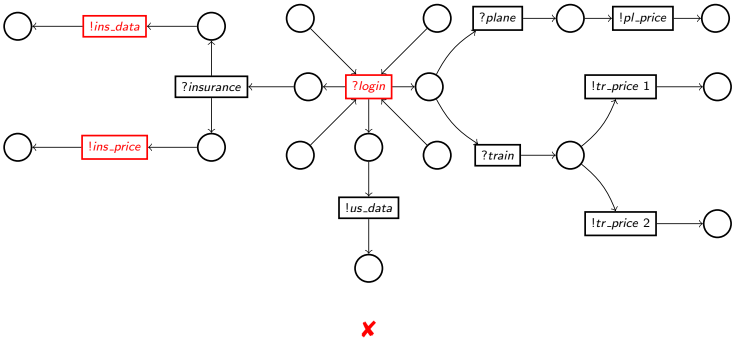


- Causally closed and conflict free set of events
- Represents executions

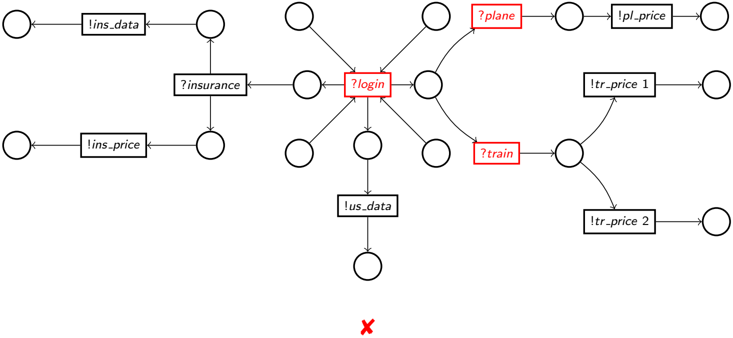


Configuration

- Causally closed and conflict free set of events
- Represents executions



- Causally closed and conflict free set of events
- Represents executions



Content

LTS vs LES

- States $\rightarrow$ Configurations
- Traces

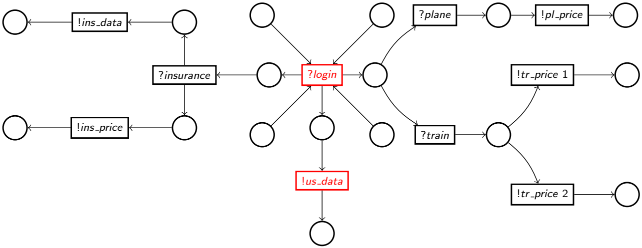
  - labeled partial order (LPO) $\rightarrow$ concurrency is preserved
- Outputs / Quiescence

  Also LPOs

Produced outputs

A configuration is quiescent iff only enabled input events

Outputs(C): LPOs of outputs leading to a quiescent configuration, δ if quiescent

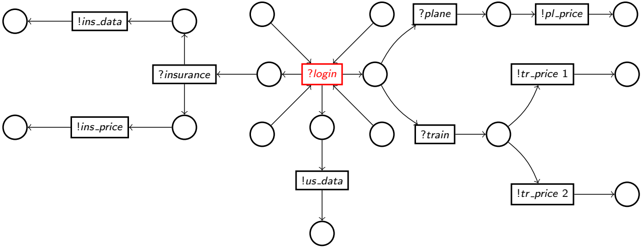


$\mathrm{out}(\{?login, !us\_data\}) = \{\delta\}$

Produced outputs

A configuration is quiescent iff only enabled input events

Outputs(C): LPOs of outputs leading to a quiescent configuration, δ if quiescent

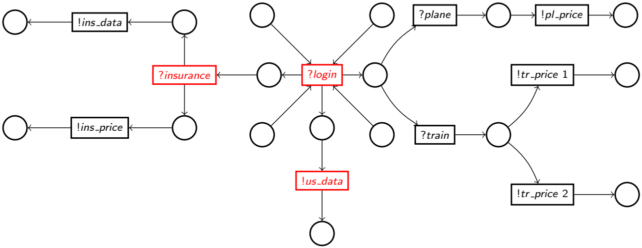


$$\mathrm{out}(\{?\mathit{login}\}) = \{!\mathit{us\_data}\}$$

Produced outputs

A configuration is quiescent iff only enabled input events

Outputs(C): LPOs of outputs leading to a quiescent configuration, δ if quiescent

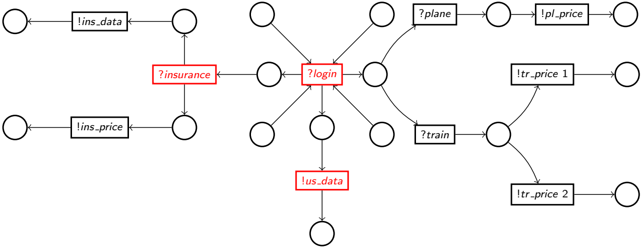


$!\mathit{ins\_price} \notin \mathrm{out}(\{?\mathit{login}, !\mathit{us\_data}, ?\mathit{insurance}\})$

Produced outputs

A configuration is quiescent iff only enabled input events

Outputs(C): LPOs of outputs leading to a quiescent configuration, δ if quiescent



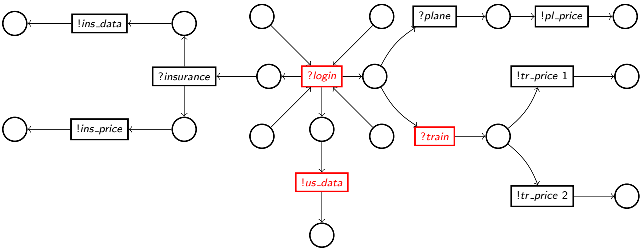
$$\mathrm{out}(\{?login, !us\_data, ?insurance\}) = \{!ins\_price\ \mathbf{co}\ !ins\_data\}$$

Concurrency is preserved

Produced outputs

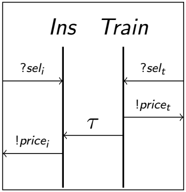
A configuration is quiescent iff only enabled input events

Outputs(C): LPOs of outputs leading to a quiescent configuration, δ if quiescent



$$\mathrm{out}(\{?login, !us\_data, ?train\}) = \{!tr\_price\ 1, !tr\_price\ 2\}$$

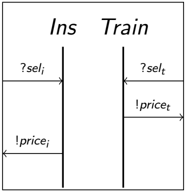
$$\mathcal{E}_i \ \textbf{co-ioco}\ \mathcal{E}_s \Leftrightarrow \forall \omega \in \text{traces}(\mathcal{E}_s) : \text{out}(\mathcal{E}_i \ \textbf{after}\ \omega) \subseteq \text{out}(\mathcal{E}_s \ \textbf{after}\ \omega)$$



Imp

ioco

~~**co-ioco**~~



Spe

no concurrency = **ioco**

Test case: finite deterministic Occurrence net with no immediate conflict between inputs

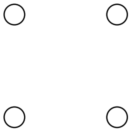
Sufficient conditions for soundness

1. $\forall \mathcal{E}_t \in T : traces(\mathcal{E}_t) \subseteq traces(\mathcal{E}_s)$
2. $\forall \mathcal{E}_t \in T, \omega \in traces(\mathcal{E}_t) : out_t(\perp \textbf{ after } \omega) = out_s(\perp \textbf{ after } \omega)$

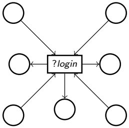
Sufficient conditions for exhautiveness

1. $\forall \omega \in traces(\mathcal{E}_s), \exists \mathcal{E}_t \in T : \omega \in traces(\mathcal{E}_t)$;
2. $\forall \mathcal{E}_t \in T, \omega \in traces(\mathcal{E}_t) : (\perp_t \textbf{ after } \omega)$ *is quiescent implies* $(\perp_s \textbf{ after } \omega)$ *is quiescent;*

From a finite deterministic occurrence net, resolve immediate conflict between inputs:

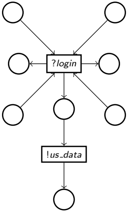


From a finite deterministic occurrence net, resolve immediate conflict between inputs:



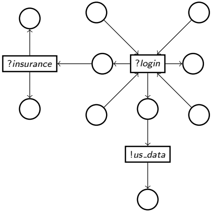
Building test cases

From a finite deterministic occurrence net, resolve immediate conflict between inputs:



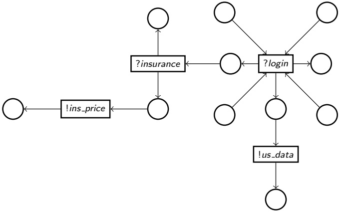
Building test cases

From a finite deterministic occurrence net, resolve immediate conflict between inputs:

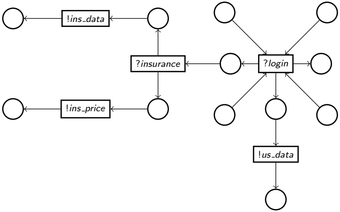


Building test cases

From a finite deterministic occurrence net, resolve immediate conflict between inputs:

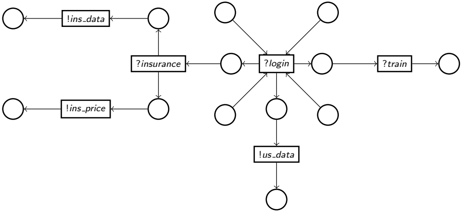


From a finite deterministic occurrence net, resolve immediate conflict between inputs:

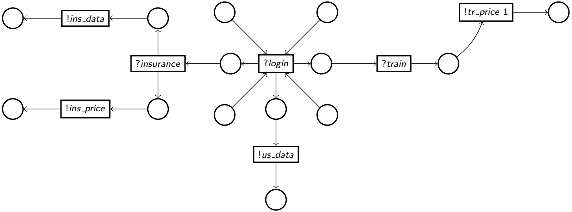


Building test cases

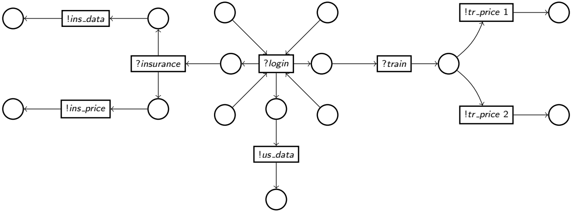
From a finite deterministic occurrence net, resolve immediate conflict between inputs:



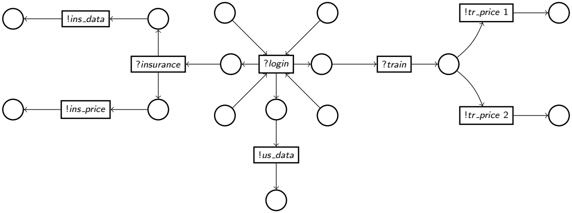
From a finite deterministic occurrence net, resolve immediate conflict between inputs:



From a finite deterministic occurrence net, resolve immediate conflict between inputs:



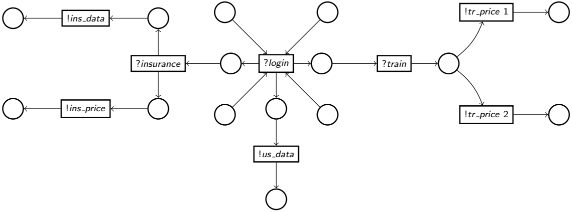
From a finite deterministic occurrence net, resolve immediate conflict between inputs:



?*plane* (and its future) can not be added: they introduce immediate input conflict

Building test cases

From a finite deterministic occurrence net, resolve immediate conflict between inputs:



Completeness: we need another test case containing ?*plane*

The set of linearization should consider every resolution of immediate conflict inputs.

Definition

Let $\mathcal{L}$ be a set of linearizations of $\leq$. Then $\mathcal{L}$ is an *immediate input conflict saturated* set, or *iics* set, for $\mathcal{E}$ iff for all $e_1, e_2 \in E^{\mathcal{I}}$ such that $e_1 \#^{\mu} e_2$, there exist $\mathcal{R}_1, \mathcal{R}_2 \in \mathcal{L}$ with $\forall e \in [e_1] : e \mathcal{R}_1 e_2$ and $\forall e \in [e_2] : e \mathcal{R}_2 e_1$.

A complete test suite

Proposition

Every event is represented by at least one test case if we use the algorithm to resolve immediate conflict and an iics set.

Theorem

From the set of all finite prefixes of the specification, the algorithm to resolve immediate conflict and an iics set yield a complete test suite.

Test selection

- Exhaustive test suites are usually infinite:
  - we need a finite prefix of the unfolding
- How to choose it?
  - basic behaviors are cycles
  - unfold each cycle once (adding outputs if necessary)

$$\downarrow$$

output closure of a (cycling) complete prefix

Cycling criterion: sound test suite

Proposition

The output closure of a complete prefix preserves traces and outputs.

Theorem

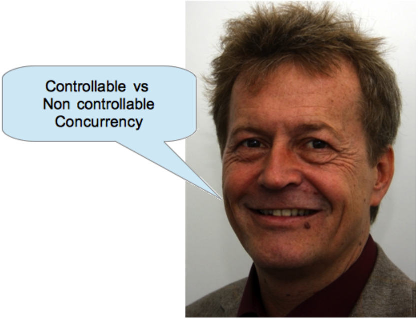
The resolving immediate conflict algorithm applied to the output closure of a complete prefix yields a sound test suite.

Result: sound test suite that covers all the basic behaviors

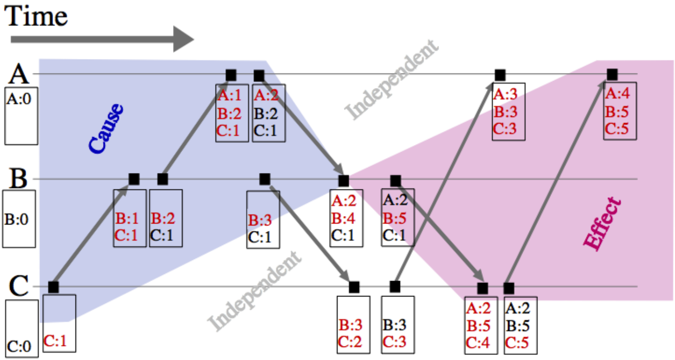
Content

Observable concurrency

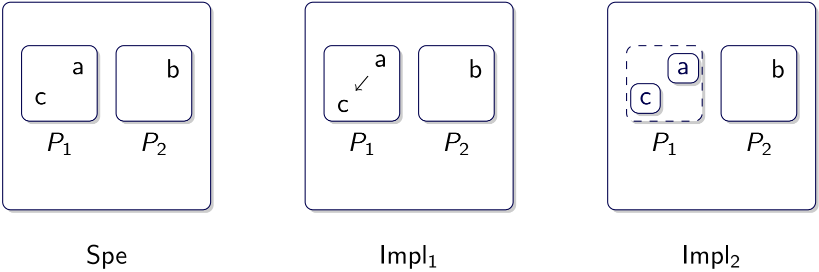
Non observable concurrency



Update clocks over synchronization: vector clocks



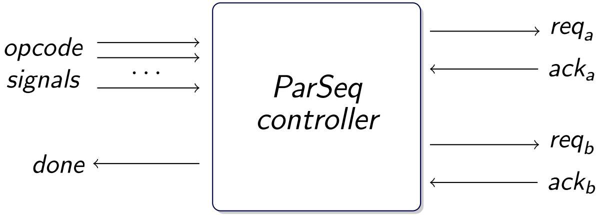
How to interpret concurrency?



weak concurrency: further refinement

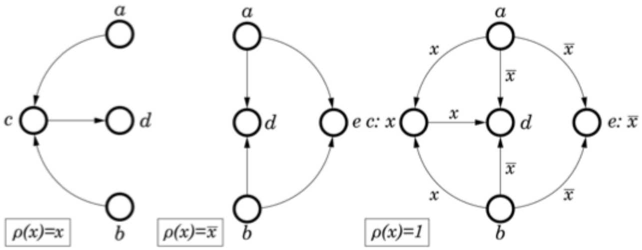
strong concurrency: distribution

How to interpret concurrency? (2)

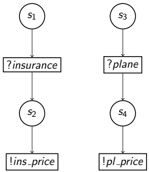


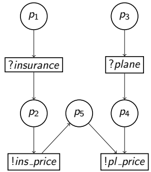
weak concurrency: external choice

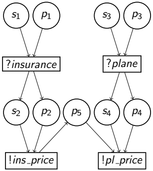
Conditional Partial Order Graphs [Mokhov,Yakovlev]



Test execution: product of LTS







Product of nets do not preserve concurrency!

Partial order commutation

Let Σ be an alphabet and $I \subseteq \Sigma \times \Sigma$ a symmetric and irreflexive relation called the independence relation.

For $x, y \in \Sigma^*$ we have $x \equiv_I y$ if we can obtain y from x by successive commutation of neighboring independent letters.

$$[x]_I \triangleq \{y \in \Sigma^* \mid x \equiv_I y\}$$

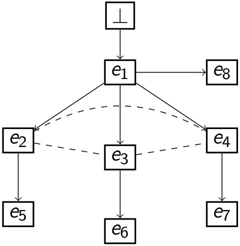
Example

If $I = \{(a, d)(d, a)(b, c)(c, d)\}$, we have:

$$[baadcb]_I = \{baadcb, baadbc, badacb, badabc, bdaacb, bdaabc\}$$

Constructing an iics set

$$I \triangleq (E \times E) \setminus (\leq \cup (\# \cap E^{\mathcal{I}} \times E^{\mathcal{I}}))$$



Normal form of any linearization:

$$\mathcal{R}_1 = (\perp)(e_1)(e_2)(e_3)(e_4)(e_5\, e_6\, e_7\, e_8) \quad \mathcal{R}_2 = (\perp)(e_1)(e_2)(e_4)(e_3)(e_5\, e_6\, e_7\, e_8)$$
$$\mathcal{R}_3 = (\perp)(e_1)(e_3)(e_2)(e_4)(e_5\, e_6\, e_7\, e_8) \quad \mathcal{R}_4 = (\perp)(e_1)(e_3)(e_4)(e_2)(e_5\, e_6\, e_7\, e_8)$$
$$\mathcal{R}_5 = (\perp)(e_1)(e_4)(e_2)(e_3)(e_5\, e_6\, e_7\, e_8) \quad \mathcal{R}_6 = (\perp)(e_1)(e_4)(e_3)(e_2)(e_5\, e_6\, e_7\, e_8)$$

Thank you!