Testing of Reactive Systems

Lecture 3: Semantics of Reactive Systems, Implementation Relations

Henrik Bohnenkamp

Lehrstuhl Informatik 2 (MOVES)
RWTH Aachen

Summer Semester 2009

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

What happened so far?

Labelled Transition Systems

- Actions Act , atomic, observable, τ unobservable
- States, Transitions
- $\Longrightarrow$, traces, reachable states, **after** ,
- non-determinism

Processes

- prefix $a.P$
- choice $P + Q$,
- parallel composition $P \|_A Q$
- process variables $\mathcal{P}$
- (recursive) process definitions $\mathcal{P}$

1 Structural Operational Semantics for $\mathbb{IP}$

2 Summary Part 1

3 Preorders

4 Some Implementation Relations

A formal semantics of processes

Approach

- Process semantics in terms of LTS
- States are processes
- Transitions derived inductively over the syntactic structure of processes
- We define one big LTS $L_{\mathbb{P}}$

A formal semantics of processes

Approach

- Process semantics in terms of LTS
- States are processes
- Transitions derived inductively over the syntactic structure of processes
- We define one big LTS $L_{\mathbb{P}}$

A formal semantics of processes

Approach

- Process semantics in terms of LTS
- States are processes
- Transitions derived inductively over the syntactic structure of processes
- We define one big LTS $L_{\mathbb{P}}$

A formal semantics of processes

Approach

- Process semantics in terms of LTS
- States are processes
- Transitions derived inductively over the syntactic structure of processes
- We define one big LTS $L_{\mathbb{P}}$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 1): Prefix

for all $a \in Act_\tau$, $p \in \mathbb{IP}$: $a.p \xrightarrow{a} p$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 2): Choice

for $p, q \in \mathbb{IP}$: if $p \xrightarrow{a} p'$ for $a \in Act_\tau$, then

$$p + q \xrightarrow{a} p'$$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 3): Choice again

if $q \xrightarrow{a} q'$ for $a \in Act_\tau$, then

$$p + q \xrightarrow{a} q'$$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 4): Process variables

For $P \in \mathcal{P}$: if $P \hat{=} p$ and $p \xrightarrow{a} p'$, then

$$P \xrightarrow{a} p'$$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 5): Parallel composition (non-synchronising)

For $A \subseteq Act$ and $p\|_A q \in \mathbb{IP}$: if $p \xrightarrow{a} p'$ and $a \notin A$, then

$$p\|_A q \xrightarrow{a} p'\|_A q$$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 6): Parallel composition (non-synchronising) again

For $A \subseteq Act$ and $p\|_A q \in \mathbb{IP}$: if $q \xrightarrow{a} q'$ and $a \notin A$, then

$$p\|_A q \xrightarrow{a} p\|_A q'$$

A formal semantics of processes

Definition: $L_{\mathbb{IP}}$

$L_{\mathbb{IP}} = (S, Act \cup \{\tau\}, \rightarrow)$ LTS, where:

- $S = \mathbb{IP}$;
- the actions set Act the same as used to define $\mathbb{IP}$;
- $\rightarrow$ contains all and only those transitions (is the least set of transitions) that can be derived with the following rules:

Rule 7): Parallel composition (synchronising)

if $p \xrightarrow{a} p'$, $q \xrightarrow{a} q'$, and $a \in A$, then

$$p \|_A q \xrightarrow{a} p' \|_A q'$$

A formal semantics of processes

A more compact way to write the rules

1) $\dfrac{}{a.p \xrightarrow{a} p}$ 2) $\dfrac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$ 3) $\dfrac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$

4) $\dfrac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$

5) $\dfrac{p \xrightarrow{a} p'}{p \|_A q \xrightarrow{a} p' \|_A q}(a \notin A)$ 6) $\dfrac{q \xrightarrow{a} q'}{p \|_A q \xrightarrow{a} p \|_A q'}(a \notin A)$

7) $\dfrac{p \xrightarrow{a} p' \quad q \xrightarrow{a} q'}{p \|_A q \xrightarrow{a} p' \|_A q'}(a \in A)$

A formal semantics of processes

A more compact way to write the rules

1) $\dfrac{}{a.p \xrightarrow{a} p}$ 2) $\dfrac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$ 3) $\dfrac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$

4) $\dfrac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$

5) $\dfrac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A)$ 6) $\dfrac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$

7) $\dfrac{p \xrightarrow{a} p' \;\; q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$

A formal semantics of processes

A more compact way to write the rules

$$1)\ \frac{}{a.p \xrightarrow{a} p} \qquad 2)\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3)\ \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$4)\ \frac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$$

$$5)\ \frac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A) \qquad 6)\ \frac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$$

$$7)\ \frac{p \xrightarrow{a} p' \quad q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$$

A formal semantics of processes

A more compact way to write the rules

$$1)\ \frac{}{a.p \xrightarrow{a} p} \qquad 2)\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3)\ \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$4)\ \frac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$$

$$5)\ \frac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A) \qquad 6)\ \frac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$$

$$7)\ \frac{p \xrightarrow{a} p' \quad q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$$

A formal semantics of processes

A more compact way to write the rules

$$1)\ \frac{}{a.p \xrightarrow{a} p} \qquad 2)\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3)\ \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$4)\ \frac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$$

$$5)\ \frac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A) \qquad 6)\ \frac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$$

$$7)\ \frac{p \xrightarrow{a} p' \quad q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$$

A formal semantics of processes

A more compact way to write the rules

$$1)\ \frac{}{a.p \xrightarrow{a} p} \qquad 2)\ \frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'} \qquad 3)\ \frac{q \xrightarrow{a} q'}{p+q \xrightarrow{a} q'}$$

$$4)\ \frac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$$

$$5)\ \frac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A) \qquad 6)\ \frac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$$

$$7)\ \frac{p \xrightarrow{a} p' \quad q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$$

A formal semantics of processes

A more compact way to write the rules

$$1)\ \frac{}{a.p \xrightarrow{a} p} \qquad 2)\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad 3)\ \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$4)\ \frac{p \xrightarrow{a} p'}{P \xrightarrow{a} p'}(P \hat{=} p)$$

$$5)\ \frac{p \xrightarrow{a} p'}{p\|_A q \xrightarrow{a} p'\|_A q}(a \notin A) \qquad 6)\ \frac{q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p\|_A q'}(a \notin A)$$

$$7)\ \frac{p \xrightarrow{a} p' \ q \xrightarrow{a} q'}{p\|_A q \xrightarrow{a} p'\|_A q'}(a \in A)$$

Operational Rules

Example 1.4.2: Three process equations:

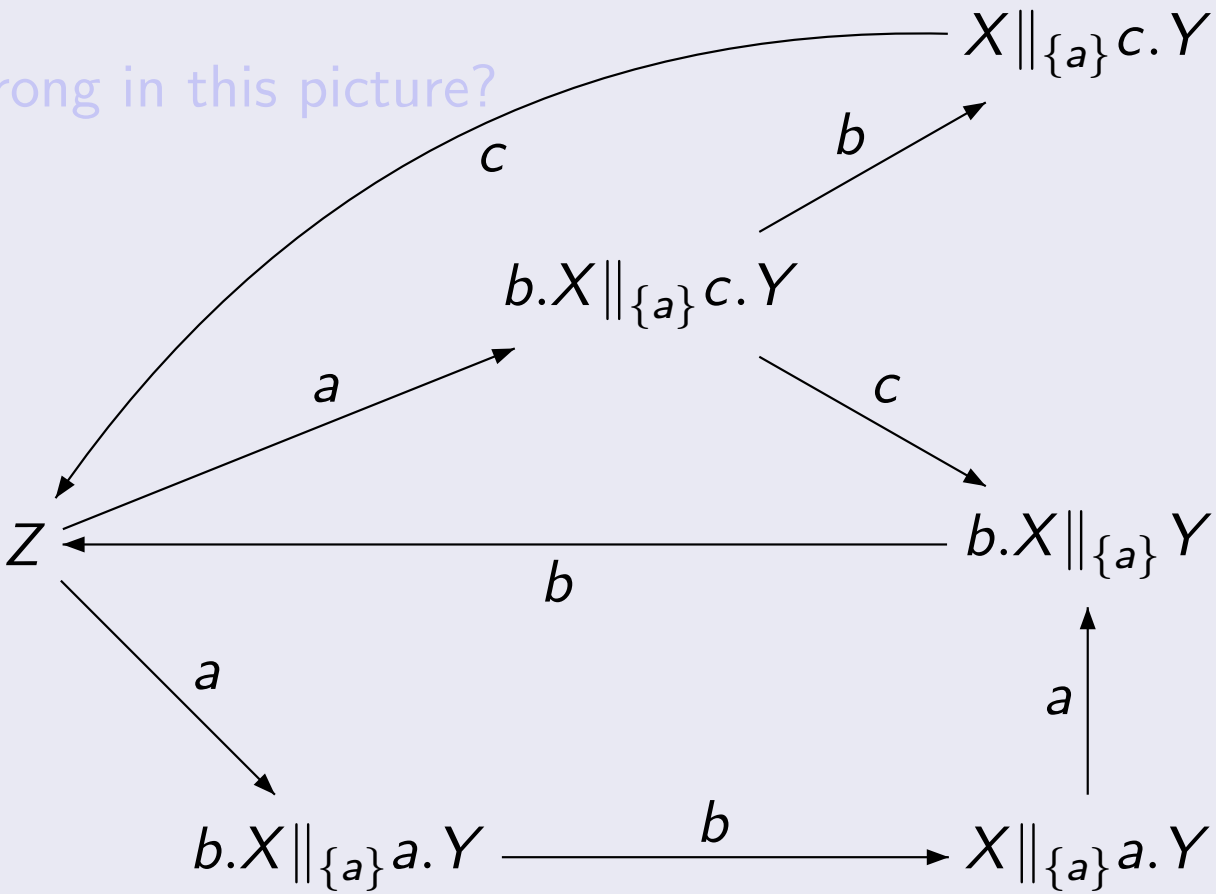
$$\begin{aligned} X &\hat{=} a.b.X \\ Y &\hat{=} a.c.Y + a.a.Y \\ Z &\hat{=} X \|_{\{a\}} Y \end{aligned}$$

$\Longrightarrow$ Blackboard

Example 1.4.2

The complete transitions system of Z:

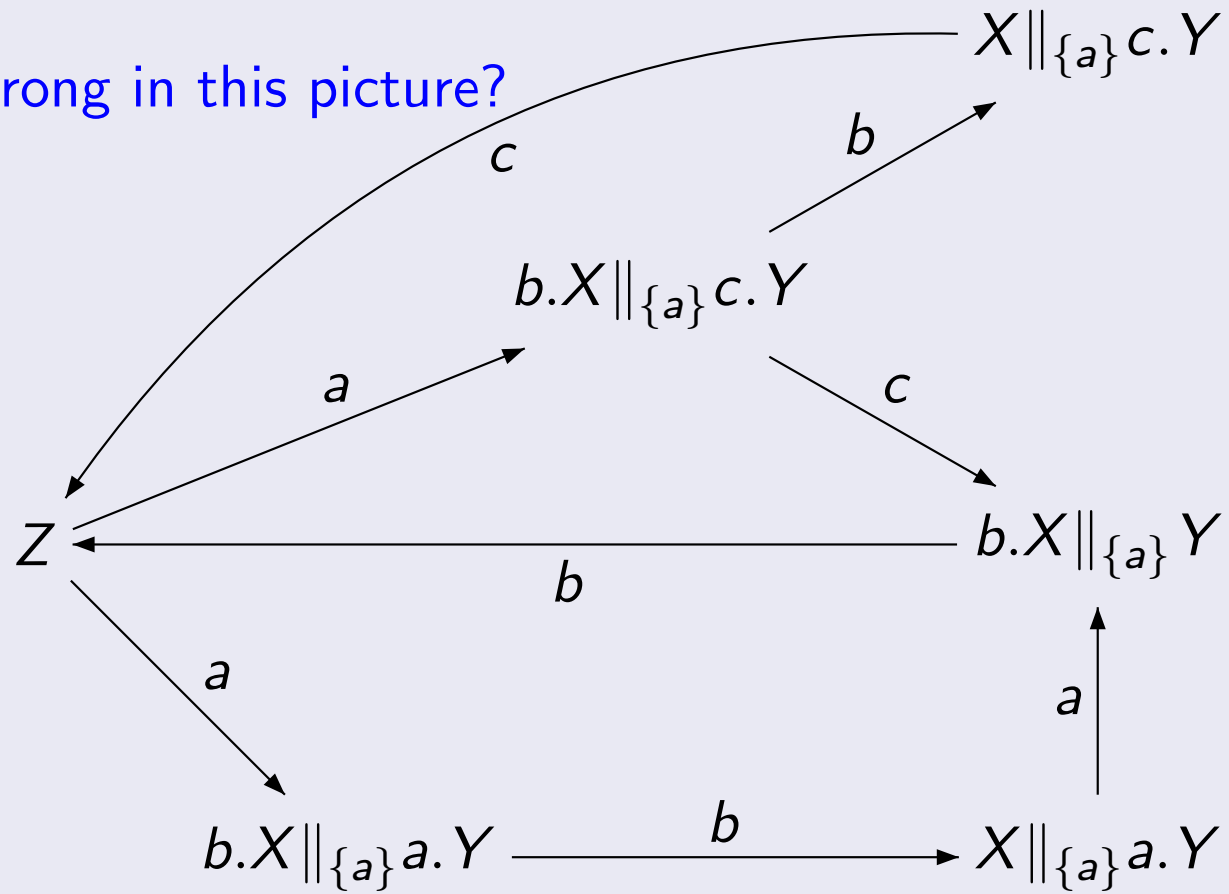
What is wrong in this picture?



Example 1.4.2

The complete transitions system of Z:

What is wrong in this picture?



Example 1.4.3

Deadlock

$(a.\mathrm{STOP} \|_{\{a,b\}} b.\mathrm{STOP}) \not\xrightarrow{c}$ for all $c \in Act_\tau$.

Example 1.4.3

Parallel Execution and Nondeterminism

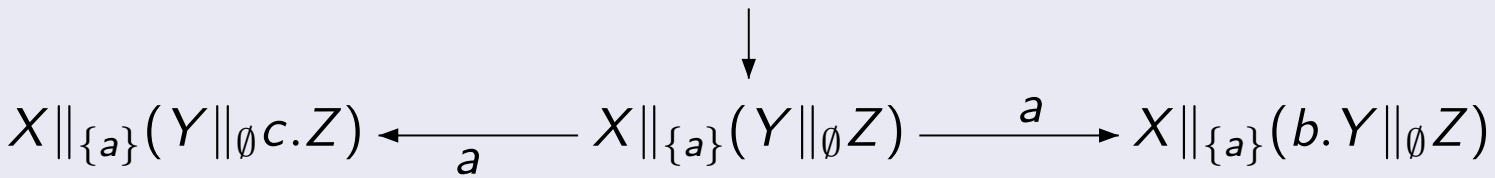
$X \hat{=} a.X$ $Y \hat{=} a.b.Y$ $Z \hat{=} a.c.Z$

Example 1.4.3

Parallel Execution and Nondeterminism

$$X \hat{=} a.X \qquad Y \hat{=} a.b.Y \qquad Z \hat{=} a.c.Z$$

Initial transitions $X \|_{\{a\}} (Y \|_{\emptyset} Z)$:



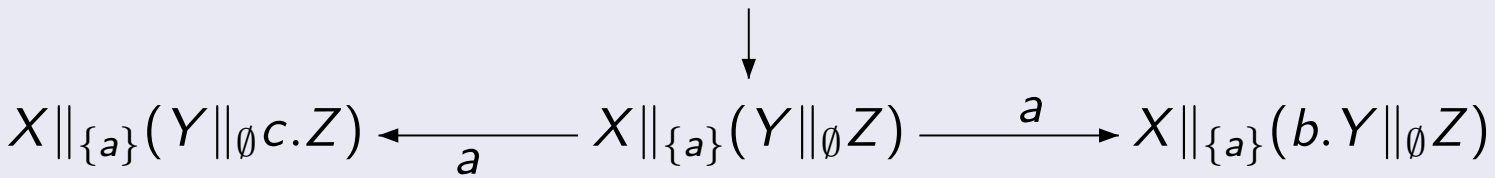
- X, Y, Z all deterministic.
- Parallelism causes non-determinism.

Example 1.4.3

Parallel Execution and Nondeterminism

$$X \hat{=} a.X \qquad Y \hat{=} a.b.Y \qquad Z \hat{=} a.c.Z$$

Initial transitions $X \|_{\{a\}} (Y \|_{\emptyset} Z)$:



- X, Y, Z all deterministic.
- Parallelism causes non-determinism.

Exercise: Derive all transitions of $X \|_{\{a\}} (Y \|_{\emptyset} Z)$

Note

- The structure of $L_{\mathbb{IP}}$ depends actually on the given process definitions, which are part of the process specification.
- If there are no process definitions, then $L_{\mathbb{IP}}$ is still well defined (why?).

1 Structural Operational Semantics for $\mathbb{IP}$

2 Summary Part 1

3 Preorders

4 Some Implementation Relations

Summary

- LTS: states, actions, transitions
- Traces
- Non-determinism
- The language $\mathbb{IP}$
- The SOS of $\mathbb{IP}$

Summary

- LTS: states, actions, transitions
- Traces
- Non-determinism
- The language $\mathbb{IP}$
- The SOS of $\mathbb{IP}$

Summary

- LTS: states, actions, transitions
- Traces
- Non-determinism
- The language $\mathbb{IP}$
- The SOS of $\mathbb{IP}$

Summary

- LTS: states, actions, transitions
- Traces
- Non-determinism
- The language $\mathbb{IP}$
- The SOS of $\mathbb{IP}$

Summary

- LTS: states, actions, transitions
- Traces
- Non-determinism
- The language $\mathbb{IP}$
- The SOS of $\mathbb{IP}$

Part 2: Differentiating Behaviour

1 Structural Operational Semantics for $\mathbb{IP}$

2 Summary Part 1

3 Preorders

4 Some Implementation Relations

Motivation

Goal

- Describing formally when an implementation is (in)correct with respect to a specification
- We use (binary) relations $\leq$: implementation relations.
- Here:

$$\leq \subseteq \mathbb{IP} \times \mathbb{IP}$$

- We say:

$$i \leq s \quad \text{iff} \quad i \text{ is implementation of } s$$

- Often: implementation relations are preorders

Preorders

Definition 2.1.1: Preorders

Let X be a set. $\leq \subseteq X \times X$ is called a preorder, iff

1. $(x, x) \in \leq$ (reflexivity)
2. $(x, y), (y, z) \in \leq \implies (x, z) \in \leq$ (transitivity)

We write $x \leq y$ for $(x, y) \in \leq$

Preorders and equivalence relations

Lemma 2.1.2

- Let $\leq$ be a preorder.
- Define $\sim_{\leq} \subseteq X \times X$ as

$$\{(x, y) |\ (x \leq y \text{ and } y \leq x\}.$$

- Then $\sim_{\leq}$ is an equivalence relation, called the kernel of preorder $\leq$.

Proof:

Show for $\sim_{\leq}$:

1. reflexivity
2. symmetry
3. transitivity

(Exercise)

Preorders and equivalence relations

Lemma 2.1.2

- Let $\leq$ be a preorder.
- Define $\sim_{\leq} \subseteq X \times X$ as

$$\{(x, y) | \ (x \leq y \text{ and } y \leq x\}.$$

- Then $\sim_{\leq}$ is an equivalence relation, called the kernel of preorder $\leq$.

Proof:

Show for $\sim_{\leq}$:

1. reflexivity
2. symmetry
3. transitivity

(Exercise)

Preorders and equivalence relations

Lemma 2.1.2

- Let $\leq$ be a preorder.
- Define $\sim_{\leq} \subseteq X \times X$ as

$$\{(x, y) |\ (x \leq y \text{ and } y \leq x\}.$$

- Then $\sim_{\leq}$ is an equivalence relation, called the kernel of preorder $\leq$.

Proof:

Show for $\sim_{\leq}$:

1. reflexivity
2. symmetry
3. transitivity

(Exercise)

Preorders and equivalence relations

Lemma 2.1.2

- Let $\leq$ be a preorder.
- Define $\sim_{\leq} \subseteq X \times X$ as

$$\{(x, y) |\ (x \leq y \text{ and } y \leq x\}.$$

- Then $\sim_{\leq}$ is an equivalence relation, called the kernel of preorder $\leq$.

Proof:

Show for $\sim_{\leq}$:

1. reflexivity
2. symmetry
3. transitivity

(Exercise)

Note

Preorders and Equivalences on $\mathbb{IP}$

- Preorders on $\mathbb{IP}$ used as implementation relations
- Kernels of preorders on $\mathbb{IP}$ then equivalences on processes: if $p \sim_{\leq} q$, then p, q behave alike, according of the chosen preorder.
- Usually it is easier to reason about preorders, rather than the respective kernels.

1 Structural Operational Semantics for $\mathbb{IP}$

2 Summary Part 1

3 Preorders

4 Some Implementation Relations

Until further notice we consider only processes that do not make any τ-steps!

Consequences

1. $\Longrightarrow = \rightarrow \cup \{(p, \varepsilon, p) \mid p \in \mathbb{P}\}$
2. For consistency sake, we write then also $p \xrightarrow{\varepsilon} p'$ iff $p \overset{\varepsilon}{\Longrightarrow} p'$.

Until further notice we consider only processes that do not make any τ-steps!

Consequences

1. $\Longrightarrow = \rightarrow \cup \{(p, \varepsilon, p) \mid p \in \mathbb{P}\}$
2. For consistency sake, we write then also $p \xrightarrow{\varepsilon} p'$ iff $p \overset{\varepsilon}{\Longrightarrow} p'$.

Definition 2.2.1

Trace preorder

Let p, q be processes. We define

$$p \leq_{tr} q \quad :\Longleftrightarrow \quad traces(p) \subseteq traces(q)$$

Trace Preorder

Properties

- $\leq_{tr}$ is a preorder: it follows from reflexivity and transitivity of $\subseteq$.
- if $p \leq q$ and p can execute $\sigma \in Act^*$, then q can as well.
- q describes the legal traces that an implementation is allowed to do.
- An implementation does not need to be complete.
- The kernel is set equality of the trace sets, called trace equivalence.

Trace Preorder

Properties

- $\leq_{tr}$ is a preorder: it follows from reflexivity and transitivity of $\subseteq$.
- if $p \leq q$ and p can execute $\sigma \in Act^*$, then q can as well.
- q describes the legal traces that an implementation is allowed to do.
- An implementation does not need to be complete.
- The kernel is set equality of the trace sets, called trace equivalence.

Trace Preorder

Properties

- $\leq_{tr}$ is a preorder: it follows from reflexivity and transitivity of $\subseteq$.
- if $p \leq q$ and p can execute $\sigma \in Act^*$, then q can as well.
- q describes the legal traces that an implementation is allowed to do.
- An implementation does not need to be complete.
- The kernel is set equality of the trace sets, called trace equivalence.

Trace Preorder

Properties

- $\leq_{tr}$ is a preorder: it follows from reflexivity and transitivity of $\subseteq$.
- if $p \leq q$ and p can execute $\sigma \in Act^*$, then q can as well.
- q describes the legal traces that an implementation is allowed to do.
- An implementation does not need to be complete.
- The kernel is set equality of the trace sets, called trace equivalence.

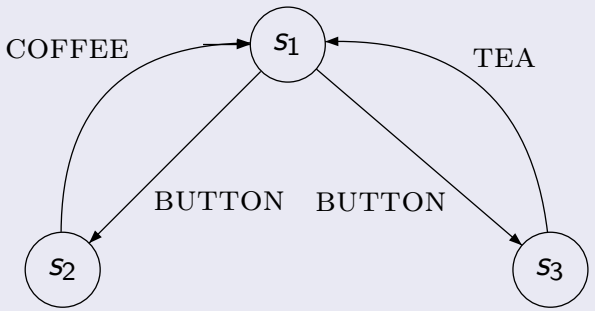
Trace Preorder

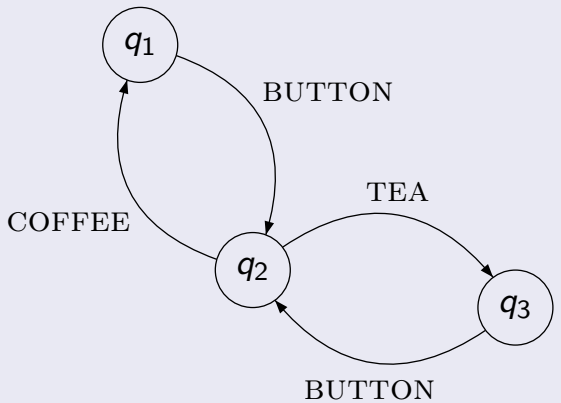
Properties

- $\leq_{tr}$ is a preorder: it follows from reflexivity and transitivity of $\subseteq$.
- if $p \leq q$ and p can execute $\sigma \in Act^*$, then q can as well.
- q describes the legal traces that an implementation is allowed to do.
- An implementation does not need to be complete.
- The kernel is set equality of the trace sets, called trace equivalence.

Examples Trace Preorder

Example 1

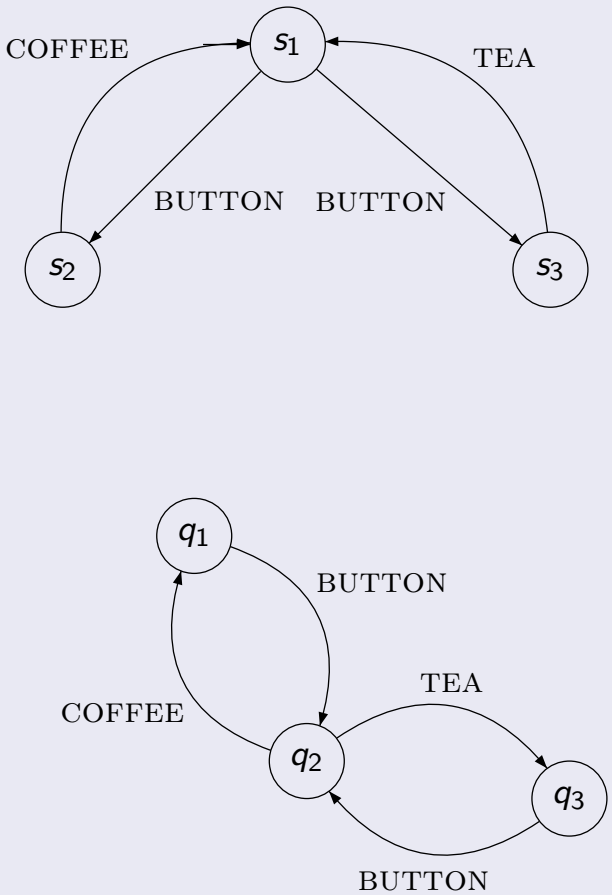




$s_1 \leq_{tr} q_1$ $q_3 \leq_{tr} s_1$ $q_1 \leq_{tr} s_1$ $s_1 \leq_{tr} q_3$

Examples Trace Preorder

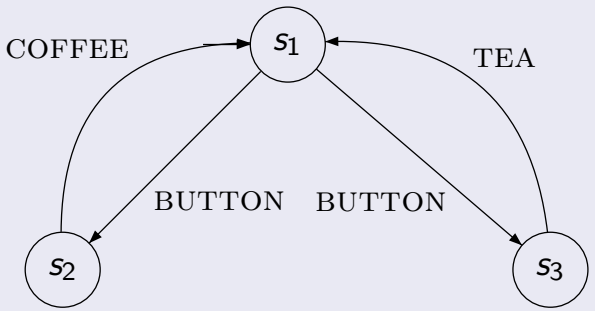
Example 1

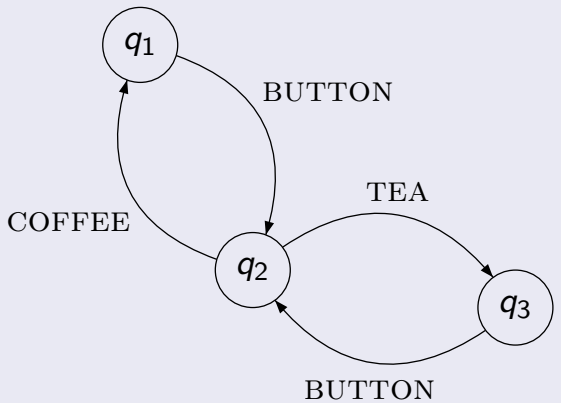


$s_1 \leq_{tr} q_1$ $q_3 \leq_{tr} s_1$ $q_1 \leq_{tr} s_1$ $s_1 \leq_{tr} q_3$

Examples Trace Preorder

Example 1

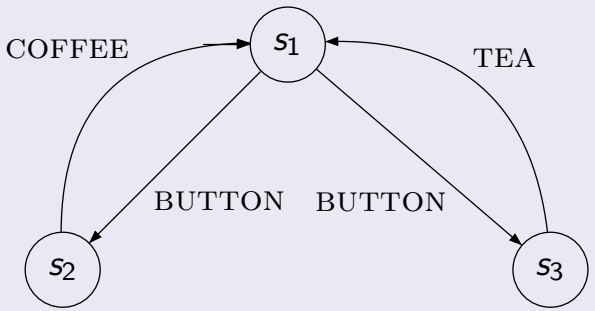


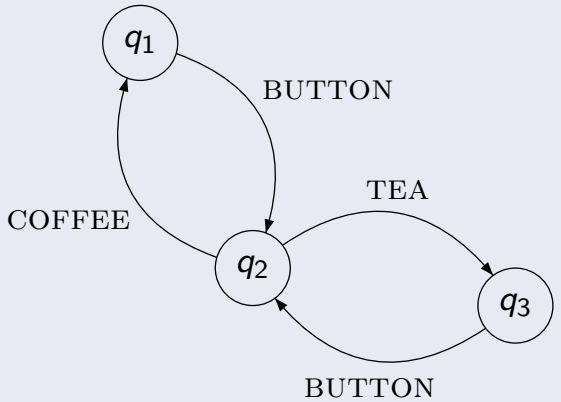


$s_1 \leq_{tr} q_1$ $q_3 \leq_{tr} s_1$ $q_1 \leq_{tr} s_1$ $s_1 \leq_{tr} q_3$

Examples Trace Preorder

Example 1

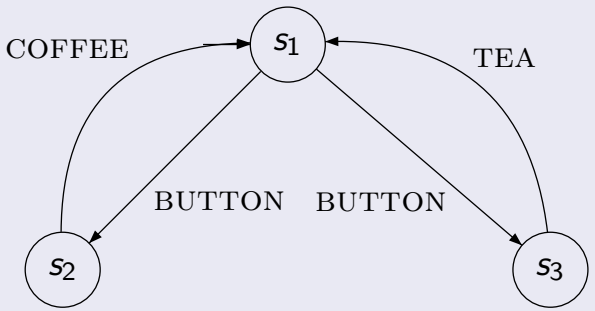


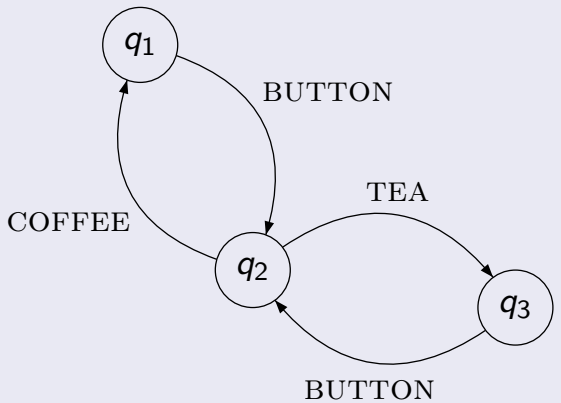


$s_1 \leq_{tr} q_1$ $q_3 \leq_{tr} s_1$ $q_1 \leq_{tr} s_1$ $s_1 \leq_{tr} q_3$

Examples Trace Preorder

Example 1

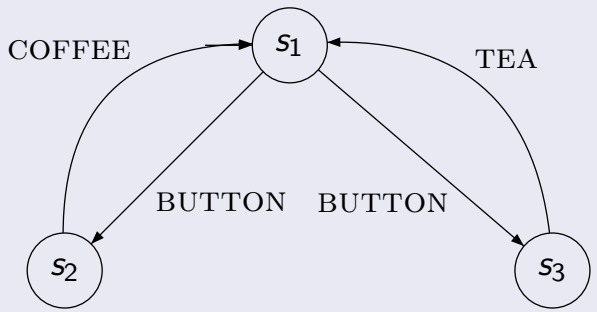


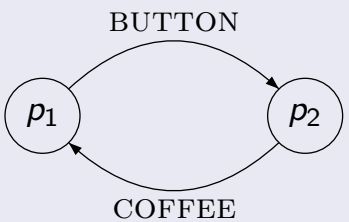


$s_1 \leq_{tr} q_1$ $q_3 \leq_{tr} s_1$ $q_1 \leq_{tr} s_1$ $s_1 \leq_{tr} q_3$

Examples Trace Preorder

Example 2

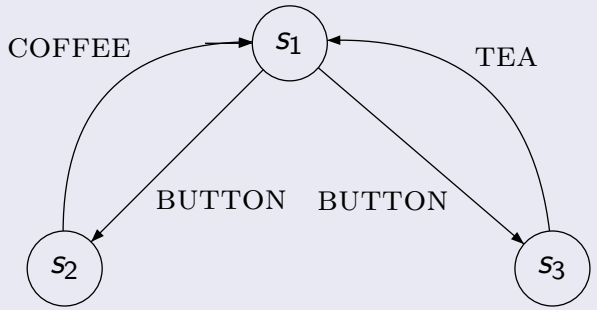


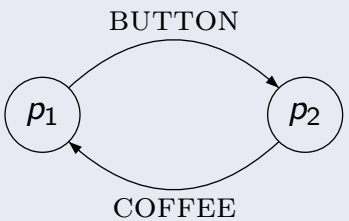


$p_1 \leq_{tr} s_1$ $p_2 \leq_{tr} s_2$ $s_1 \not\leq_{tr} p_1$ $s_2 \not\leq_{tr} p_2$

Examples Trace Preorder

Example 2

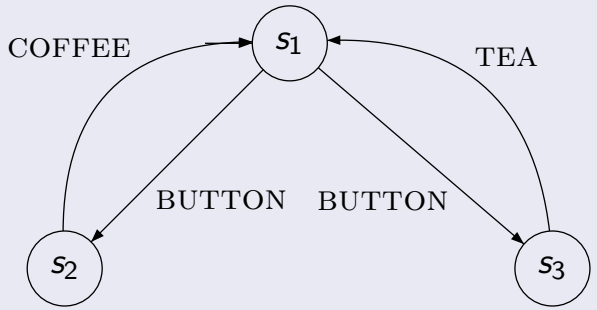


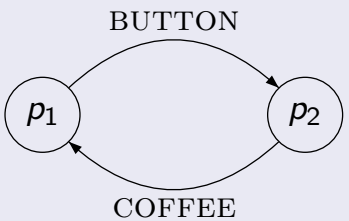


$p_1 \leq_{tr} s_1$ $p_2 \leq_{tr} s_2$ $s_1 \not\leq_{tr} p_1$ $s_2 \not\leq_{tr} p_2$

Examples Trace Preorder

Example 2

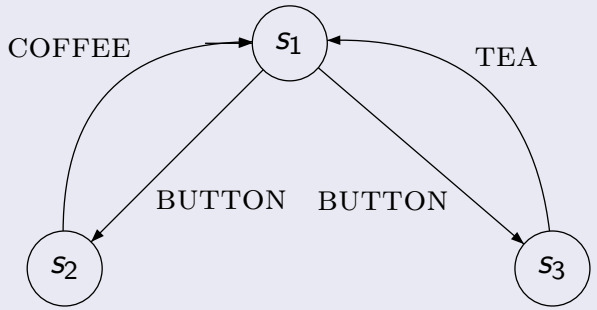


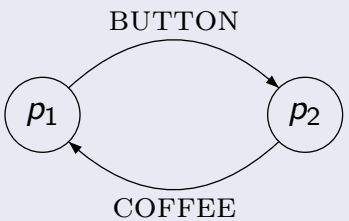


$p_1 \leq_{tr} s_1$ $p_2 \leq_{tr} s_2$ $s_1 \not\leq_{tr} p_1$ $s_2 \not\leq_{tr} p_2$

Examples Trace Preorder

Example 2

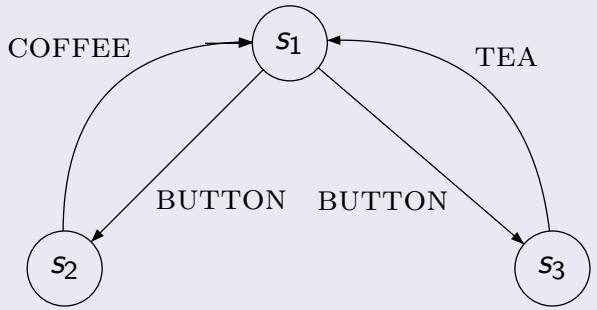


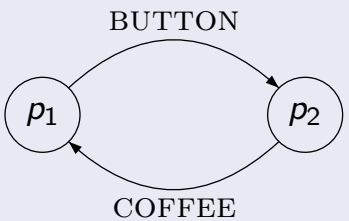


$p_1 \leq_{tr} s_1$ $p_2 \leq_{tr} s_2$ $s_1 \not\leq_{tr} p_1$ $s_2 \not\leq_{tr} p_2$

Examples Trace Preorder

Example 2





$p_1 \leq_{tr} s_1$ $p_2 \leq_{tr} s_2$ $s_1 \not\leq_{tr} p_1$ $s_2 \not\leq_{tr} p_2$

Definition 2.2.3

Bisimulation Equivalence

A bisimulation is a binary relation $R \subseteq \mathbb{P} \times \mathbb{P}$ satisfying:
For all $a \in Act$:

1) if pRq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathbb{P} : q \xrightarrow{a} q'$ and $p' R q'$.

2) if pRq and $q \xrightarrow{a} q'$, then $\exists p' \in \mathbb{P} : p \xrightarrow{a} p'$ and $p' R q'$.

We say p is bisimulation equivalent to q ($\sim_B$) if there is a bisimulation R such that pRq.

Definition 2.2.3

Bisimulation Equivalence

A bisimulation is a binary relation $R \subseteq \mathbb{P} \times \mathbb{P}$ satisfying:
For all $a \in Act$:

1) if pRq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathbb{P} : q \xrightarrow{a} q'$ and $p' R q'$.

2) if pRq and $q \xrightarrow{a} q'$, then $\exists p' \in \mathbb{P} : p \xrightarrow{a} p'$ and $p' R q'$.

We say p is bisimulation equivalent to q ($\sim_B$) if there is a bisimulation R such that pRq.

Definition 2.2.3

Bisimulation Equivalence

A bisimulation is a binary relation $R \subseteq \mathbb{P} \times \mathbb{P}$ satisfying:
For all $a \in Act$:

1) if pRq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathbb{P} : q \xrightarrow{a} q'$ and $p' R q'$.

2) if pRq and $q \xrightarrow{a} q'$, then $\exists p' \in \mathbb{P} : p \xrightarrow{a} p'$ and $p' R q'$.

We say p is bisimulation equivalent to q ($\sim_B$) if there is a bisimulation R such that pRq.

Definition 2.2.3

Bisimulation Equivalence

A bisimulation is a binary relation $R \subseteq \mathbb{P} \times \mathbb{P}$ satisfying:
For all $a \in Act$:

1) if pRq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathbb{P} : q \xrightarrow{a} q'$ and $p'Rq'$.

2) if pRq and $q \xrightarrow{a} q'$, then $\exists p' \in \mathbb{P} : p \xrightarrow{a} p'$ and $p'Rq'$.

We say p is bisimulation equivalent to q ($\sim_B$) if there is a bisimulation R such that pRq.

Definition 2.2.3

Bisimulation Equivalence

A bisimulation is a binary relation $R \subseteq \mathbb{P} \times \mathbb{P}$ satisfying:
For all $a \in Act$:

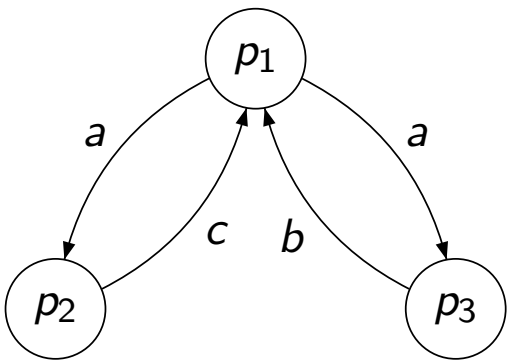
1) if pRq and $p \xrightarrow{a} p'$, then $\exists q' \in \mathbb{P} : q \xrightarrow{a} q'$ and $p' R q'$.

2) if pRq and $q \xrightarrow{a} q'$, then $\exists p' \in \mathbb{P} : p \xrightarrow{a} p'$ and $p' R q'$.

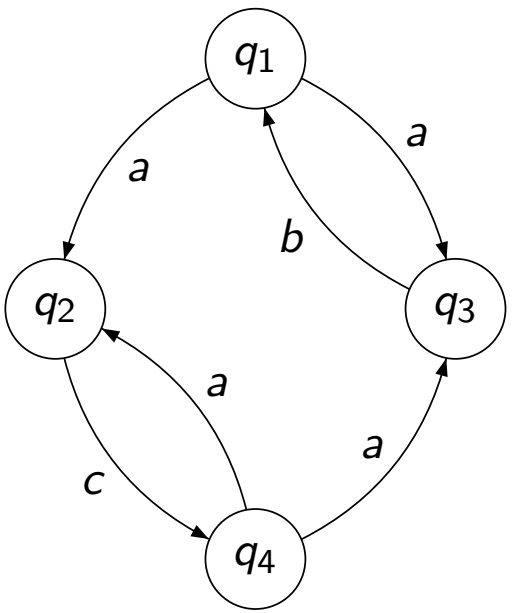
We say p is bisimulation equivalent to q ($\sim_B$) if there is a bisimulation R such that pRq.

Example 2.2.4

L_1 :



L_2 :



Assertion: p_1 and q_1 are bisimulation equivalent

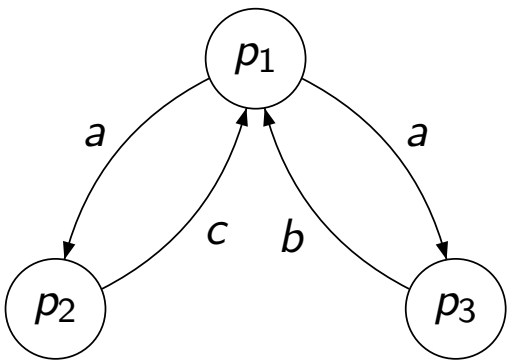
Consider:

$$p_1 R\, q_1 \quad p_2 R\, q_2 \quad p_3 R\, q_3 \quad p_1 R\, q_4$$

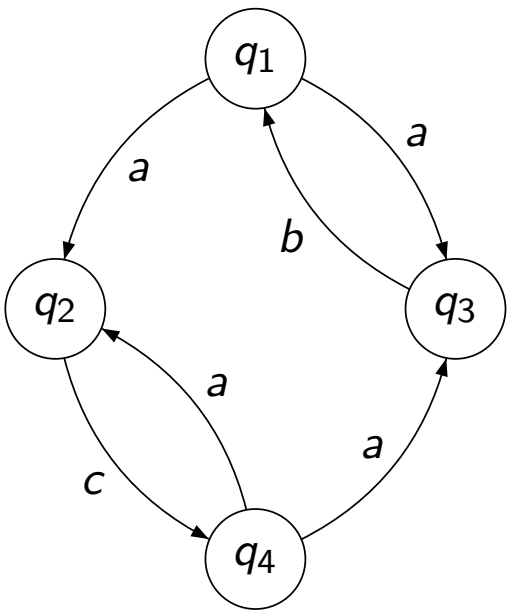
To show: R is a bisimulation

Example 2.2.4

L_1 :



L_2 :



Assertion: p_1 and q_1 are bisimulation equivalent

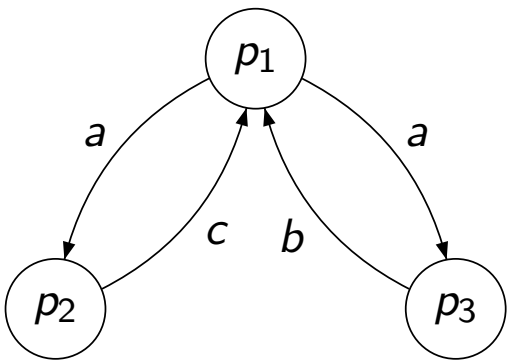
Consider:

$$p_1 R\, q_1 \quad p_2 R\, q_2 \quad p_3 R\, q_3 \quad p_1 R\, q_4$$

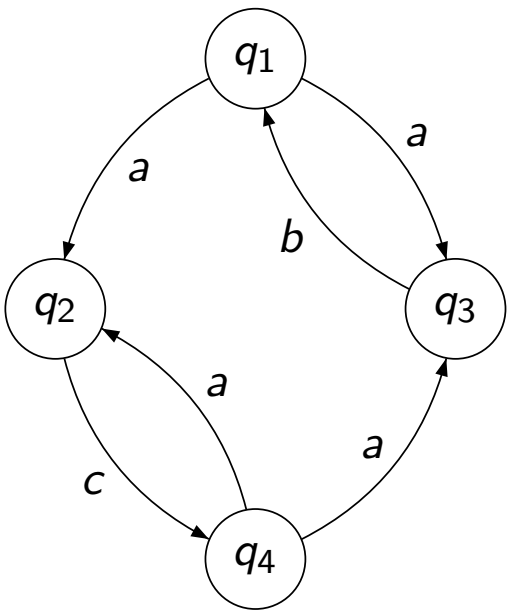
To show: R is a bisimulation

Example 2.2.4

L_1 :



L_2 :



Assertion: p_1 and q_1 are bisimulation equivalent

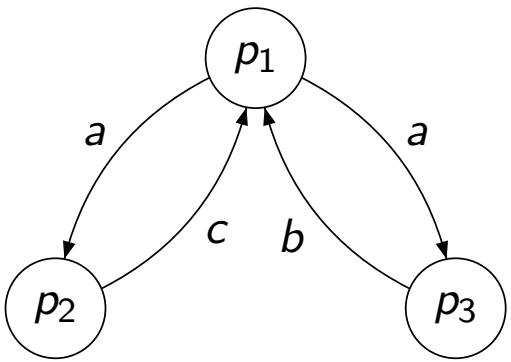
Consider:

$$p_1 R\, q_1 \quad p_2 R\, q_2 \quad p_3 R\, q_3 \quad p_1 R\, q_4$$

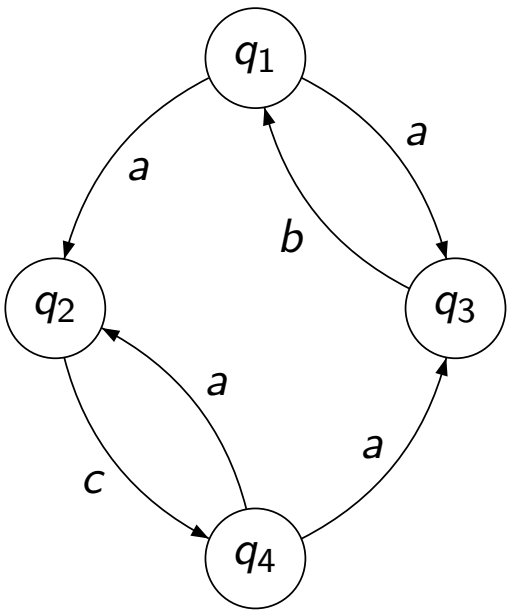
To show: R is a bisimulation

Example 2.2.4

L_1 :



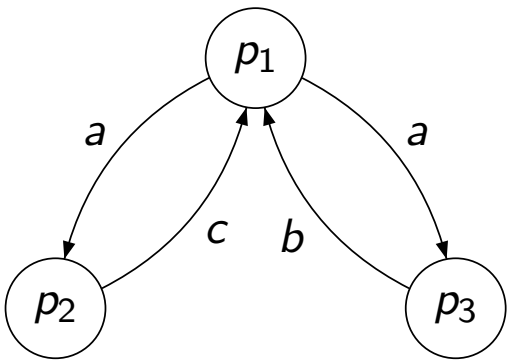
L_2 :



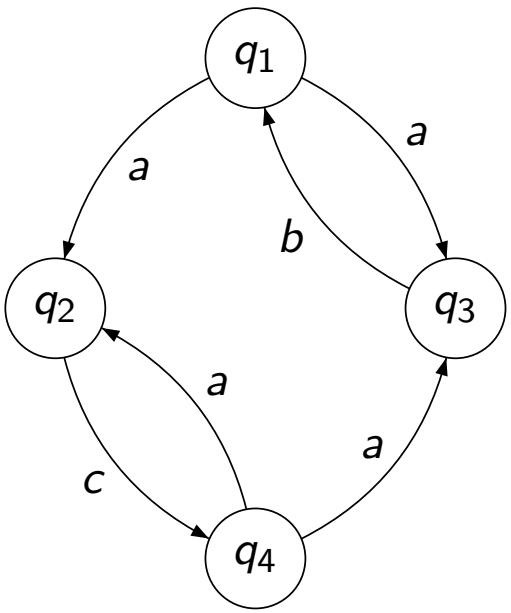
$p_1 R\, q_1$ $p_2 R\, q_2$ $p_3 R\, q_3$ $p_1 R\, q_4$

Example 2.2.4

L_1 :



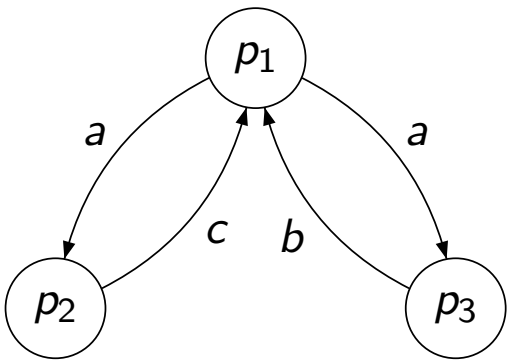
L_2 :



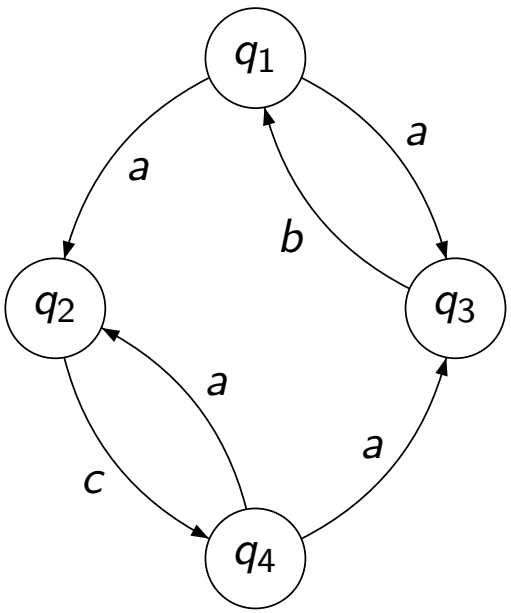
$p_1 R\, q_1$ $p_2 R\, q_2$ $p_3 R\, q_3$ $p_1 R\, q_4$

Example 2.2.4

L_1 :



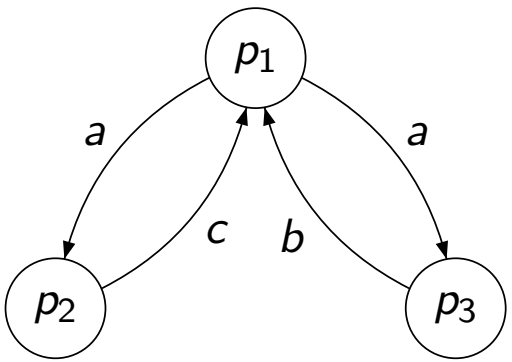
L_2 :



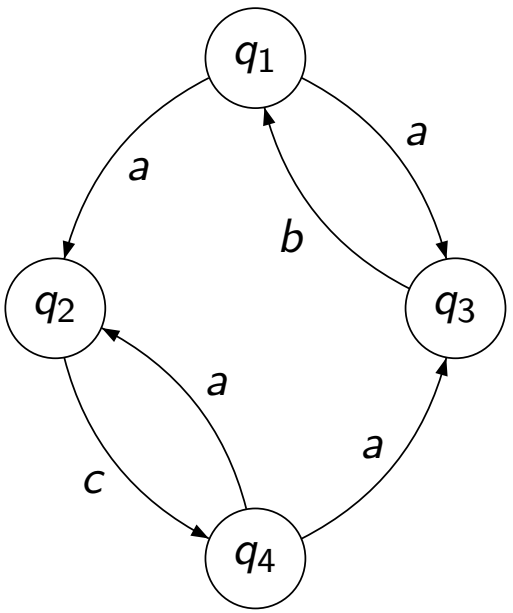
$p_1 R\, q_1$ $p_2 R\, q_2$ $p_3 R\, q_3$ $p_1 R\, q_4$

Example 2.2.4

L_1 :



L_2 :



$p_1 R\, q_1 \qquad p_2 R\, q_2 \qquad p_3 R\, q_3 \qquad p_1 R\, q_4$