# Operational Semantics and Behavioural Equivalence

**Ernesto Posse**

**August 2, 2001**

## Introduction

- Languages and formalisms for describing, modelling and simulating systems (models $\approx$ programs).
- Syntax:
  - What are the entities described in the language or formalism.
  - How to combine the entities to build composite models of the systems.
- Semantics: The meaning of the programs or models described in the formalism. Without it we only have a bunch of meaningless symbols and/or diagrams.

## Semantics

- Motivation for studying semantics
  - Guideline for implementing formalisms.
  - Reasoning about formalisms and the systems described by them.
- Types of semantics
  - Operational: What is the computation that a program performs, or what is the state trajectory of a system.
  - Denotational: What abstract mathematical entity is represented (denoted) by a program or model (e.g. input-output function).
  - Translational: The semantics of a program or model is given by translating it to another language or formalism for which we already know the semantics.
  - Axiomatic: Preconditions and postconditions.

## Reasoning about systems

- Deduce properties satisfied by a system, e.g. termination, correctness, complexity, liveness, fairness, etc.
- Applications: verification, optimization, automatic program generation.
- Behavioural Equivalence:
  - Practical motivation: verification, optimization, automatic program generation.
  - Theoretical motivation: system equivalence is fundamental for semantics. A semantics without a notion of equivalence is incomplete.
  - Process/System Algebra
  - For black-box approach it is better treated in the context of Denotational Semantics, but for studying state-trajectories and interaction, Operational Semantics is better suited.

## Operational Semantics

- Term/Graph Rewritting Systems
  - A **TRS** is a $(\mathcal{A}, \rightarrow)$, where $\mathcal{A}$ is *any* set, and $\rightarrow \subseteq \mathcal{A} \times \mathcal{A}$ is called a *reduction* or *reaction* relation.
  - We write $s_1 \rightarrow s_2$ instead of $(s_1, s_2) \in \rightarrow$ to mean "$s_1$ evolves in one time step into $s_2$", or "$s_1$ is substituted by $s_2$"
  - The operational semantics of a language are given by defining the appropriate $\mathcal{A}$ and $\rightarrow$.
  - In general it is straightforward to generate an interpreter from a TRS.

## TRS example: CCS

- Communicating Concurrent Systems (Milner '79)
  - Several concurrent *processes* or *agents*, possibly composite.
  - Communication by channels: synchronous message passing.
- Syntax
  - Nil process: 0
  - Send a signal: $\overline{a}.P$
  - Receive a signal: $a.P$
  - Parallel composition: $P_1 \mid P_2$
  - Restriction (scoping): $\nu x.P$
- Example 1: $\overline{a}.0 \mid a.0$
- Example 2: $\overline{a}.b.P_1 \mid \overline{b}.a.P_2$
- Example 3: $\overline{b}.P_1 \mid b.\nu a.(\overline{a}.P_2 \mid a.P_3)$

## TRS example: CCS (cont.)

- Let $\mathcal{A}_{ccs}$ be the set of all CCS programs, so the operational semantics of CCS is the TRS $(\mathcal{A}_{ccs}, \rightarrow)$ where $\rightarrow$ is defined (inductively) as follows:
  - Comm: $\overline{a}.P_1 \mid a.P_2 \rightarrow P_1 \mid P_2$
  - Par: if $P_1 \rightarrow P_1'$ then $P_1 \mid P_2 \rightarrow P_1' \mid P_2$.
  - Restr: if $P_1 \rightarrow P_1'$ then $\nu x.P_1 \rightarrow \nu x.P_1'$.
- Example 1: $\overline{a}.0 \mid a.0 \rightarrow 0 \mid 0$
- Example 2: $\overline{a}.b.P_1 \mid \overline{b}.a.P_2 \not\rightarrow$
- Example 3: $\overline{b}.P_1 \mid b.\nu a.(\overline{a}.P_2 \mid a.P_3) \rightarrow P_1 \mid \nu a.(\overline{a}.P_2 \mid a.P_3) \rightarrow P_1 \mid \nu a.(P_2 \mid P_3) \rightarrow \ldots$

# Operational Semantics

- Labelled Transition Systems (LTS)
  - Finer grained treatment of how the system behaves
  - An **LTS** is a $(\mathcal{A}, \mathcal{L}, \rightarrow)$, where $\mathcal{A}$ and $\mathcal{L}$ are *any* sets, and $\rightarrow \subseteq \mathcal{A} \times \mathcal{L} \times \mathcal{A}$ is called a *transition* relation.
  - We write $s_1 \stackrel{\alpha}{\rightarrow} s_2$ instead of $(s_1, \alpha, s_2) \in \rightarrow$ to mean "$s_1$ evolves in one time step into $s_2$ by performing the action $\alpha$".
  - Context sensitive: $\mathcal{L}$ represents context information.
  - Closely related to automata: given an LTS $(\mathcal{A}, \mathcal{L}, \rightarrow)$ and a system $s \in \mathcal{A}$ we can obtain a state automata representing $s$.

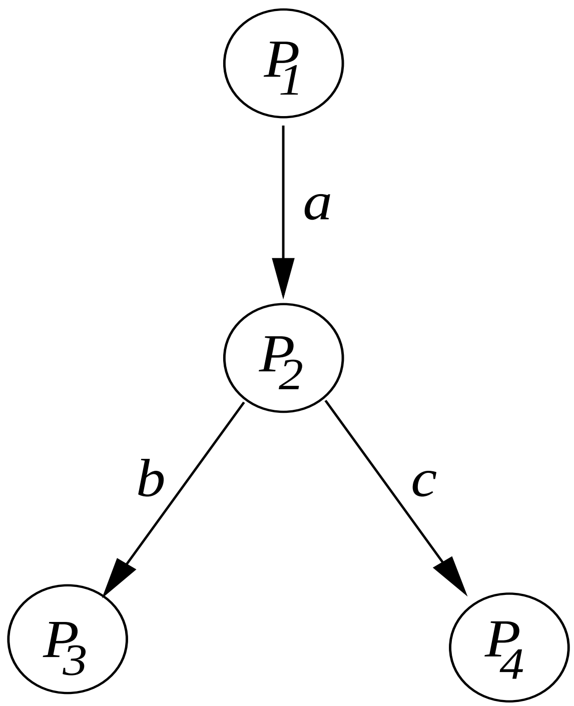
## LTS example: CCS

- Given $\mathcal{A}_{ccs}$ as before, and $\mathcal{L}_{ccs} \stackrel{def}{=} \{\tau\} \cup \bigcup\{x, \overline{x}\}$ for all channel names $x$, the semantics of CCS is given by the LTS $(\mathcal{A}_{ccs}, \mathcal{L}_{ccs}, \rightarrow)$ where $\rightarrow$ is defined as follows:
  - Pref: $\alpha.P \stackrel{\alpha}{\rightarrow} P$
  - Comm: if $P \stackrel{\overline{a}}{\rightarrow} P'$ and $Q \stackrel{a}{\rightarrow} Q'$ then $P \mid Q \stackrel{\tau}{\rightarrow} P' \mid Q'$
  - Par: if $P_1 \stackrel{\alpha}{\rightarrow} P_1'$ then $P_1 \mid P_2 \stackrel{\alpha}{\rightarrow} P_1' \mid P_2$.
  - Restr: if $P_1 \stackrel{\alpha}{\rightarrow} P_1'$ and $x$ is not in $\alpha$ then $\nu x.P_1 \stackrel{\alpha}{\rightarrow} \nu x.P_1'$.

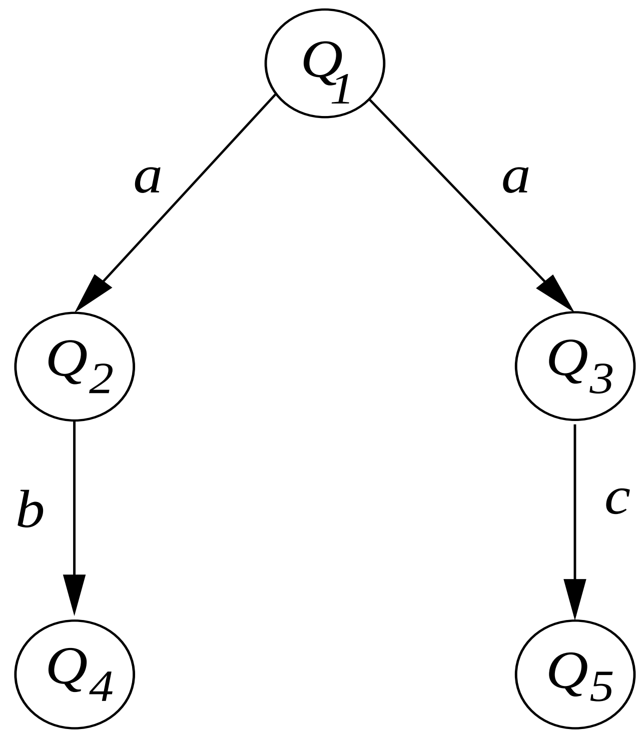
## Simulation and similarity

- Formalize the notion of "one system/model imitates another system/model".
  - Do no separate the world from the formalism domain.
  - The "simulating" system must match the actions of the "simulated" system.
- Given an LTS $(\mathcal{A}, \mathcal{L}, \rightarrow)$, a binary relation $\mathcal{S} \subseteq \mathcal{A} \times \mathcal{A}$ is called *a simulation* if for any $P, Q \in \mathcal{A}$, $(P, Q) \in \mathcal{S}$ implies that whenever $P \stackrel{\alpha}{\rightarrow} P'$ for some $\alpha \in \mathcal{L}$ and some $P' \in \mathcal{A}$ then $Q \stackrel{\alpha}{\rightarrow} Q'$ for some $Q' \in \mathcal{A}$ and $(P', Q') \in \mathcal{S}$.
- We say that $P$ and $Q$ are *similar* (or that $Q$ *simulates* $P$), written $P \preceq Q$ if there is a simulation $\mathcal{S}$ such that $(P, Q) \in \mathcal{S}$.

## Simulation example




Machine P




Machine Q

$Q_1 \preceq P_1$ because

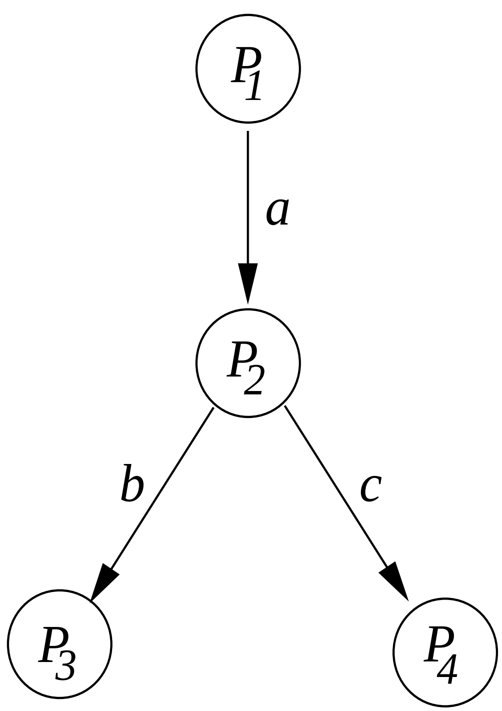
$\mathcal{S} \stackrel{def}{=} \{(Q_1, P_1), (Q_2, P_2), (Q_3, P_2), (Q_4, P_3), (Q_5, P_4)\}$ is a simulation.

...but $P_1 \not\preceq Q_1$

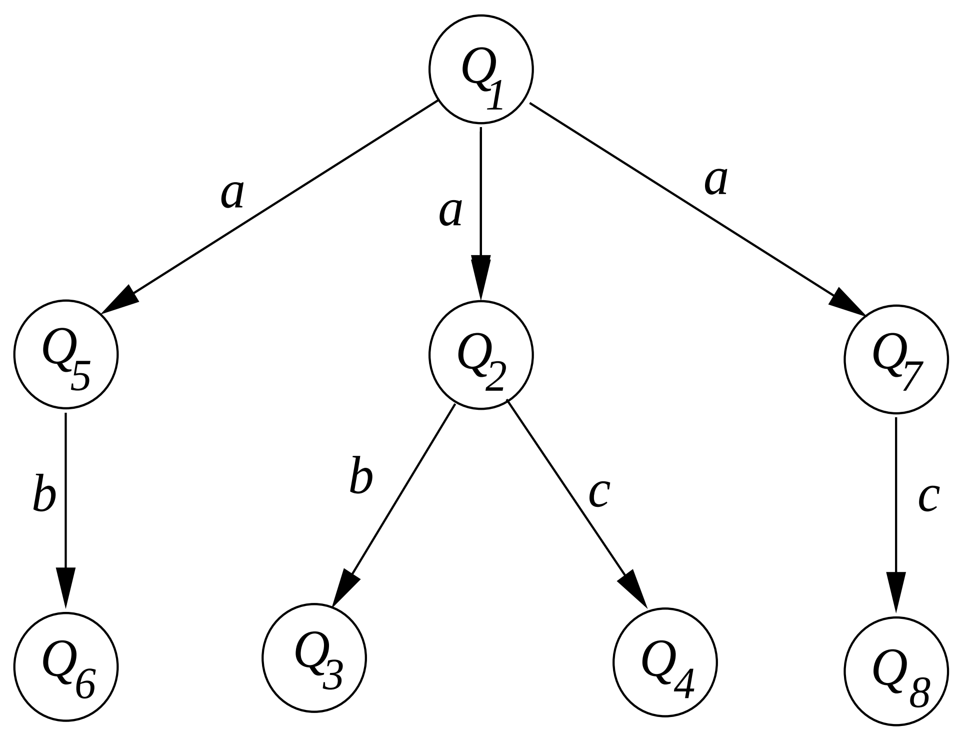
## Behavioural equivalence: first attempt

- Two-way simulation: we consider $P$ and $Q$ equivalent if $P \preceq Q$ and $Q \preceq P$.
- Is this an equivalence relation? Yes.
- Is it a good behavioural equivalence relation? No. A good behavioural equivalence should differentiate between systems that do not have the same state trajectories.

## Are these equivalent?




Machine P




Machine Q

## Behavioural equivalence: second attempt

- Two-way simulation fails in capturing the idea that two equivalent systems must interact in the same manner with the external world (i.e. it is not a congruence)
- Given an LTS $(\mathcal{A}, \mathcal{L}, \rightarrow)$, a binary relation $\mathcal{S} \subseteq \mathcal{A} \times \mathcal{A}$ is called *a bisimulation* if for any $P, Q \in \mathcal{A}$, $(P, Q) \in \mathcal{S}$ implies that for any $\alpha \in \mathcal{L}$:
  - Whenever $P \xrightarrow{\alpha} P'$ then $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{S}$.
  - Whenever $Q \xrightarrow{\alpha} Q'$ then $P \xrightarrow{\alpha} P'$ and $(P', Q') \in \mathcal{S}$.
- Alternative definition: $\mathcal{S}$ is a bisimulation if it is a simulation and $\mathcal{S}^{-1}$ is also a simulation.
- We say that $P$ and $Q$ are *bisimilar*, written $P \sim Q$ if there is a bisimulation $\mathcal{S}$ such that $(P, Q) \in \mathcal{S}$.

## Congruence relations

- When an equivalence relation is not good enough.
- Given an algebra (or language) with some operators (or combinators), we can define the notion of "context" as a term with a "hole": if $\mathcal{C}[\cdot]$ is a context in our formalism and $P$ a system in the formalism (element in the algebra), then $\mathcal{C}[P]$ is the system resulting from putting $P$ in place of the hole $[\cdot]$.
- A congruence relation is an equivalence relation $\cong$ such that whenever $P \cong Q$, then $P$ and $Q$ are interchangeable in all possible contexts, i.e. it is preserved by all contexts: for all contexts $\mathcal{C}[\cdot]$, $P \cong Q$ implies $\mathcal{C}[P] \cong \mathcal{C}[Q]$.
- In CCS, bisimilarity is a congruence.

## Final remarks

- A notion of behavioural equivalence should be a congruence
- ...but the notions of simulation and behavioural equivalence can (and sometimes must) be relaxed to be more useful.
  - Weak (bi)simulation
  - Barbed (bi)simulation
  - etc.
- The notion of (bi)simulation depends on the formalism, and sometimes it is not a congruence (yet might be useful). There are specialized notions, e.g. markovian bisimulation.
- The notion of bisimulation induces an algebra (set of axioms for equations).
- Bisimulation is decidable, and there are standard algorithms for testing it (but they are also formalism dependant).