Lecture 2: Towards Bisimulation

Concurrency Theory Summer 2024

Dr. Stephan Mennicke

April 9th, 2024

TU Dresden, Knowledge-Based Systems Group

Review

Overview

Part 0: Completing the Introduction (**today**)

- learning about *bisimilarity* and *bisimulations*

Part 1: Semantics of (Sequential) Programming Languages

- WHILE – an old friend
- denotational semantics (a baseline and an exercise of the inductive method)
- natural semantics and (structural) operational semantics

Part 2: Towards Parallel Programming Languages

- bisimilarity and its success story
- deep-dive into induction and coinduction
- algebraic properties of bisimilarity

Part 3: Expressive Power

- Calculus of Communicating Systems (CCS)
- Petri nets

Overview

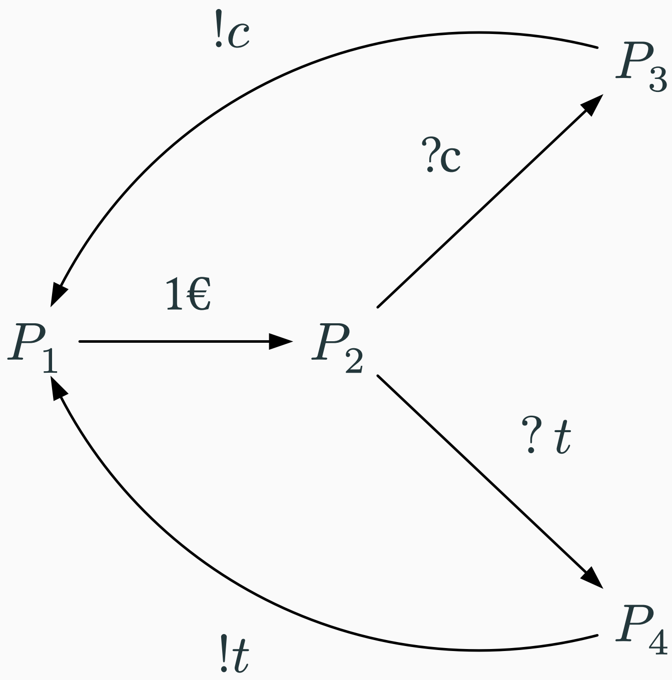
- denotations as sound basis for *sequential* programming language semantics
- denotations insufficient when concurrency is involved
  - computation is interaction
  - interaction between processes
- labeled transition systems (Definition 3) as **the** model for behavior
  - basic notions and notations
  - classes of LTSs and processes (Definition 5)

Central Questions:

1. What is a *process*, mathematically?
2. What does it mean for two processes to be *equal*?
   - seek notions of equality that are effective
   - equality must be justifiable, according to the notion of *process*

Definition 3 (Labeled Transition System): A *labeled transition system* (LTS) is a triple $(\mathrm{Pr}, \mathrm{Act}, \rightarrow)$ where Pr is a non-empty set, the *domain* of the LTS; Act is the set of *actions*; and $\rightarrow \subseteq \mathrm{Pr} \times \mathrm{Act} \times \mathrm{Pr}$ is the *transition relation*.

The Vending Machine as an LTS



This is the LTS $V = (\mathrm{Pr}, \mathrm{Act}, \rightarrow)$ where $\mathrm{Pr} = \{P_1, P_2, P_3, P_4\}$, $\mathrm{Act} = \{1€, ?\,c, ?\,t, !c, !t\}$, and $\rightarrow = \{(P_1, 1€, P_2), (P_2, ?\,c, P_3), (P_2, ?\,t, P_4), (P_3, !c, P_1), (P_4, !t, P_1)\}$

An LTS is

- *image-finite* if for each μ, relation $\xrightarrow{\mu}$ is image-finite (i.e., for all P, the set $\left\{P' \,\middle|\, P \xrightarrow{\mu} P'\right\}$ is finite);
- *finitely branching* if it is image-finite and, for each P, the set $\left\{\mu \,\middle|\, P \xrightarrow{\mu}\right\}$ is finite;
- *finite-state* if it has a finite number of states;
- *finite* if it is finite-state and acyclic;
- *deterministic* if all processes are deterministic (i.e., for P and μ, $P \xrightarrow{\mu} P_1$ and $P \xrightarrow{\mu} P_2$ implies $P_1 = P_2$)

Getting Inspiration for Process Equality

By equality we mean *equivalence relations* for LTSs (i.e., binary relations on processes).

A *process relation* is a binary relation on the processes of an LTS.

Reminder: A process relation $\mathcal{R} \subseteq \mathrm{Pr} \times \mathrm{Pr}$ is an equivalence relation if $\mathcal{R}$ is

1. *reflexive* (i.e., for all $P \in \mathrm{Pr}$, $(P, P) \in \mathcal{R}$)
2. *symmetric* (i.e., for all $(P, Q) \in \mathcal{R}$, $(Q, P) \in \mathcal{R}$), and
3. *transitive* (i.e., for all $(P, Q) \in \mathcal{R}$ and $(Q, R) \in \mathcal{R}$, $(P, R) \in \mathcal{R}$).

Intuition: Two processes should be equivalent if they cannot be distinguished by interacting with them.

Because of the resemblence of LTSs to (1) edge-labeled directed graphs and (2) nondeterministic finite automata, we let both fields try.

Equality Stolen from Graph Theory

In graph theory, or generally relational structures, equality is established by means of *isomorphisms*.

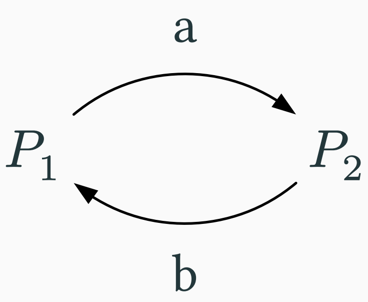
Definition: Two LTSs $T = (\mathrm{Pr}, \mathrm{Act}, \rightarrow)$ and $T' = \left(\mathrm{Pr}', \mathrm{Act}, \xrightarrow{a}{}'\right)$ are *isomorphic*, denoted $T \cong T'$, if there is a *bijective function* $f : \mathrm{Pr} \rightarrow \mathrm{Pr}'$ such that $P \xrightarrow{\mu} P'$ if, and only if, $f(P) \xrightarrow{\mu}{}' f(P')$. (For experts: f is a bijective and strong homomorphism)

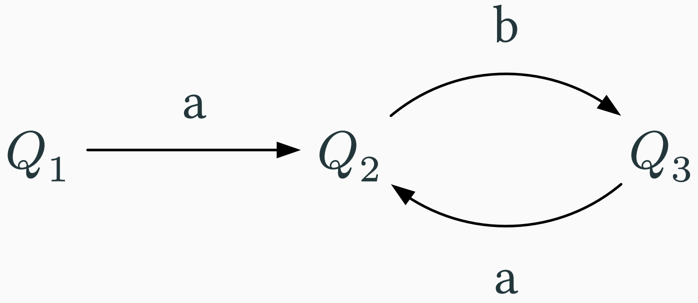
Two processes P and Q are *isomorphic*, denoted $P \cong Q$, if their induced LTSs are isomorphic.

Exercise: Is $\cong$ an equivalence relation for LTSs/processes?

Exercise: Is it a good equivalence for processes?

Counterexample for Graph Theory





Equality Stolen from Automata Theory

Two nondeterministic finite automata (NFAs) are equal if they accept the same language.

LTSs neither have initial nor final states.

The LTS analogue to NFA language equivalence is called *trace equivalence*: Two processes P and Q are equal if they can produce the same **finite** sequences of transitions.

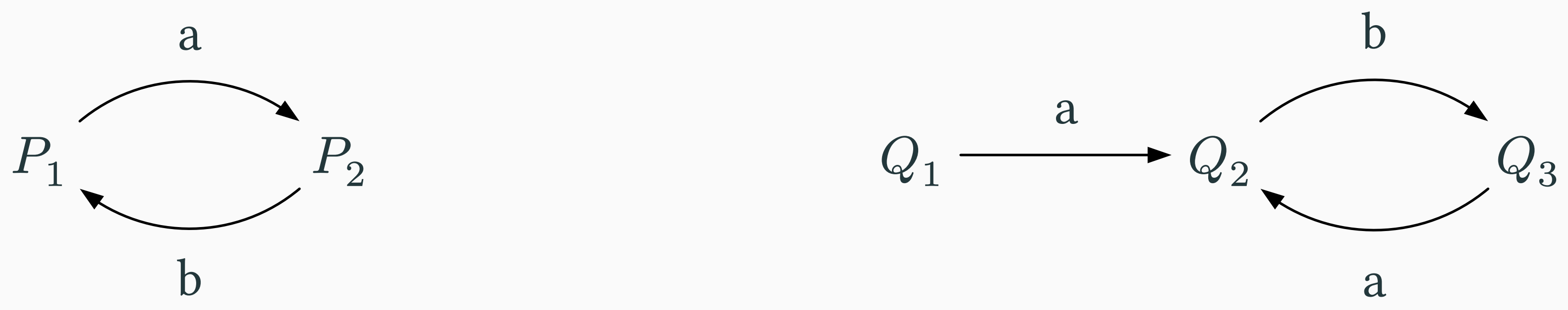
Definition: Let $T = (\mathrm{Pr}, \mathrm{Act}, \rightarrow)$ be an LTS and $P, Q \in \mathrm{Pr}$. The *set of traces* of process P is defined by $\mathrm{tr}(P) := \left\{ s \in \mathrm{Act}^\star \,\middle|\, P \xrightarrow{s} \right\}$.

P and Q are *trace equivalent*, denoted $P \overset{\mathrm{tr}}{\equiv} Q$, if $\mathrm{tr}(P) = \mathrm{tr}(Q)$.

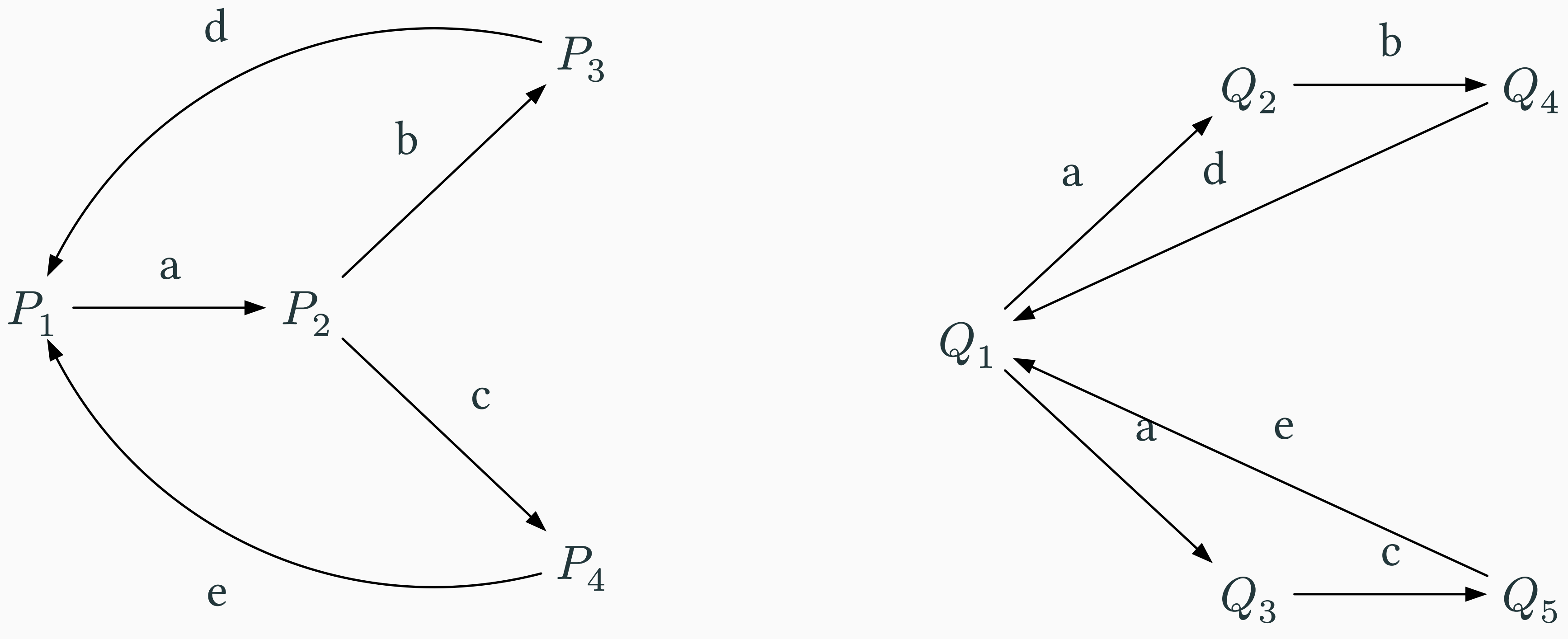
Exercise: Is $\overset{\mathrm{tr}}{\equiv}$ an equivalence relation?

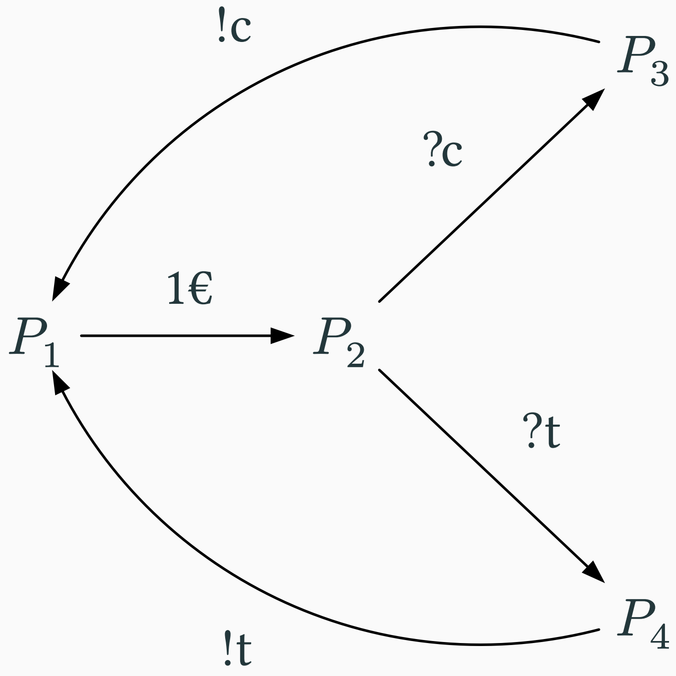
Exercise: Is it a good one?

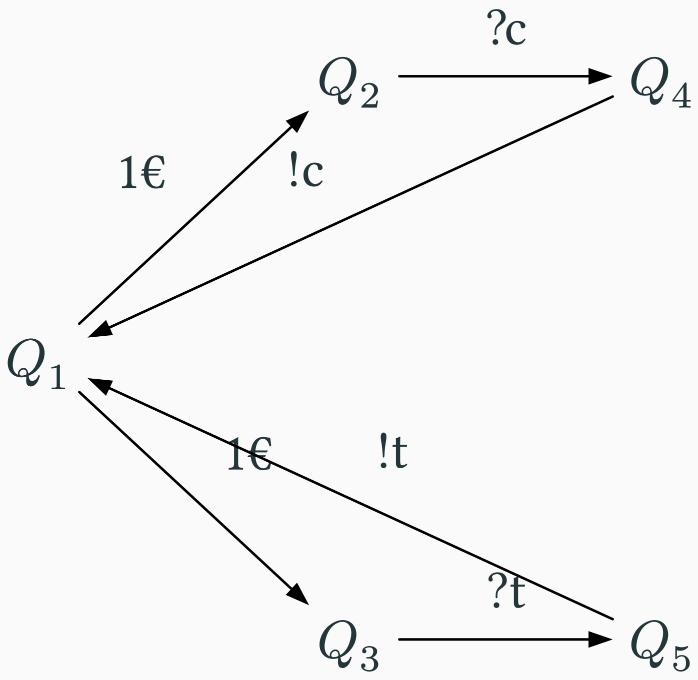
Counterexample for Graph Theory Revisited



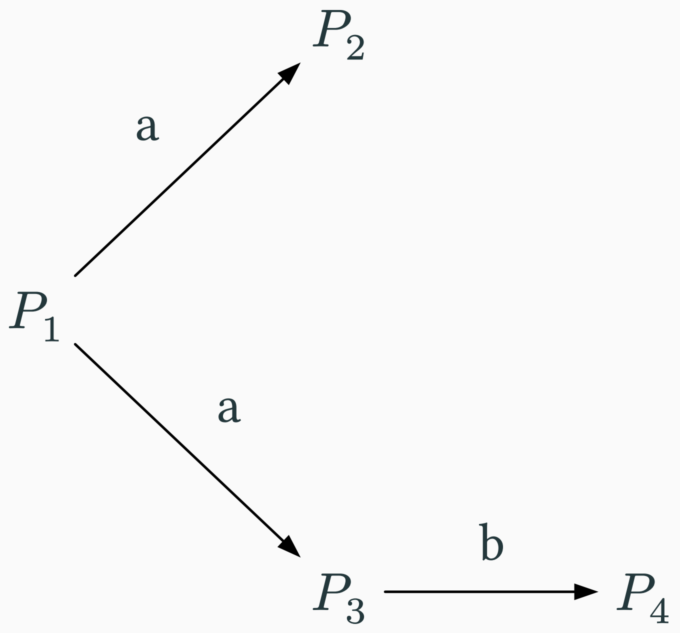
Example (1) for Automata Theory

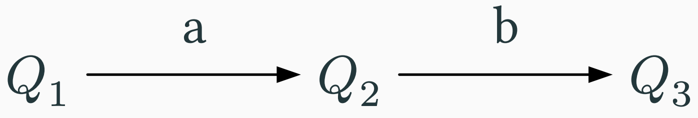



!c
P_3
?c
1€
P_1
P_2
?t
P_4
!t


?c
Q_2
Q_4
1€
!c
Q_1
1€
!t
?t
Q_3
Q_5

Counterexample (2) for Automata Theory





Summary from Counterexamples

- look for something that distinguishes more than trace equivalence
- rather transition-based than structure-based

Intuition: If we do something with the one process, we should be able to do the same with the other.

Bisimulation

Bisimilarity

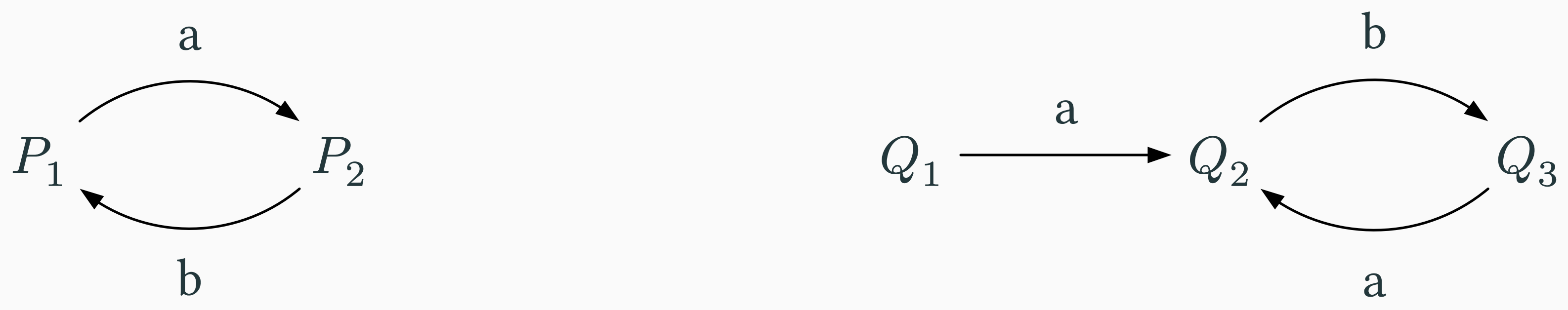
Definition 6 (Bisimilarity): A process relation $\mathcal{R}$ is a *bisimulation* if, for all $(P, Q) \in \mathcal{R}$ and all $\mu \in \mathrm{Act}$:

1. for P' with $P \xrightarrow{\mu} P'$, a Q' exists such that $Q \xrightarrow{\mu} Q'$ and $(P', Q') \in \mathcal{R}$;
2. for Q' with $Q \xrightarrow{\mu} Q'$, a P' exists such that $P \xrightarrow{\mu} P'$ and $(P', Q') \in \mathcal{R}$.

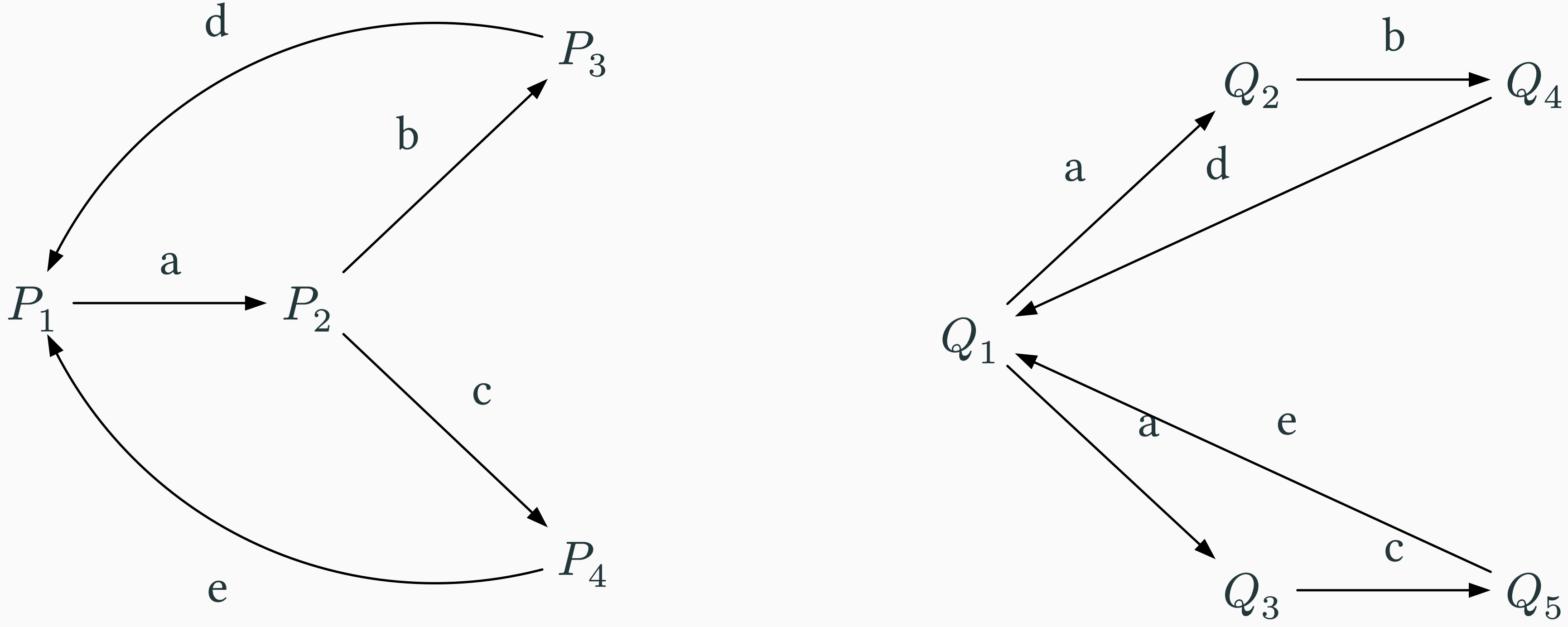
Bisimilarity, denoted by $\Leftrightarrow$, is the union of all bisimulations.

Processes P and Q are *bisimilar*, consequently denoted by $P \Leftrightarrow Q$, if there is a bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.

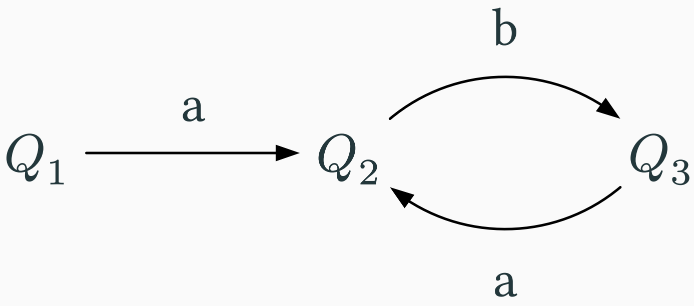
Counterexample for Graph Theory Revisited

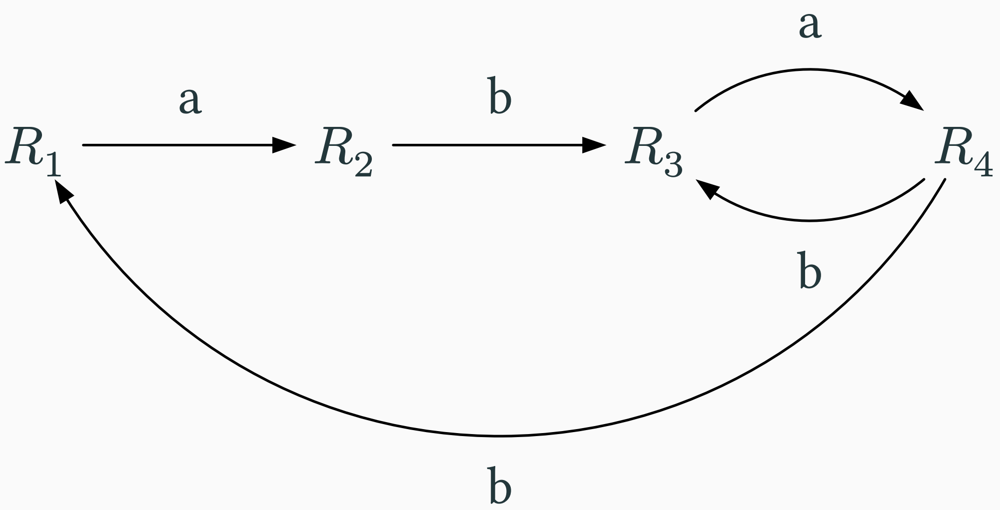


Counterexample (1) for Automata Theory Revisited

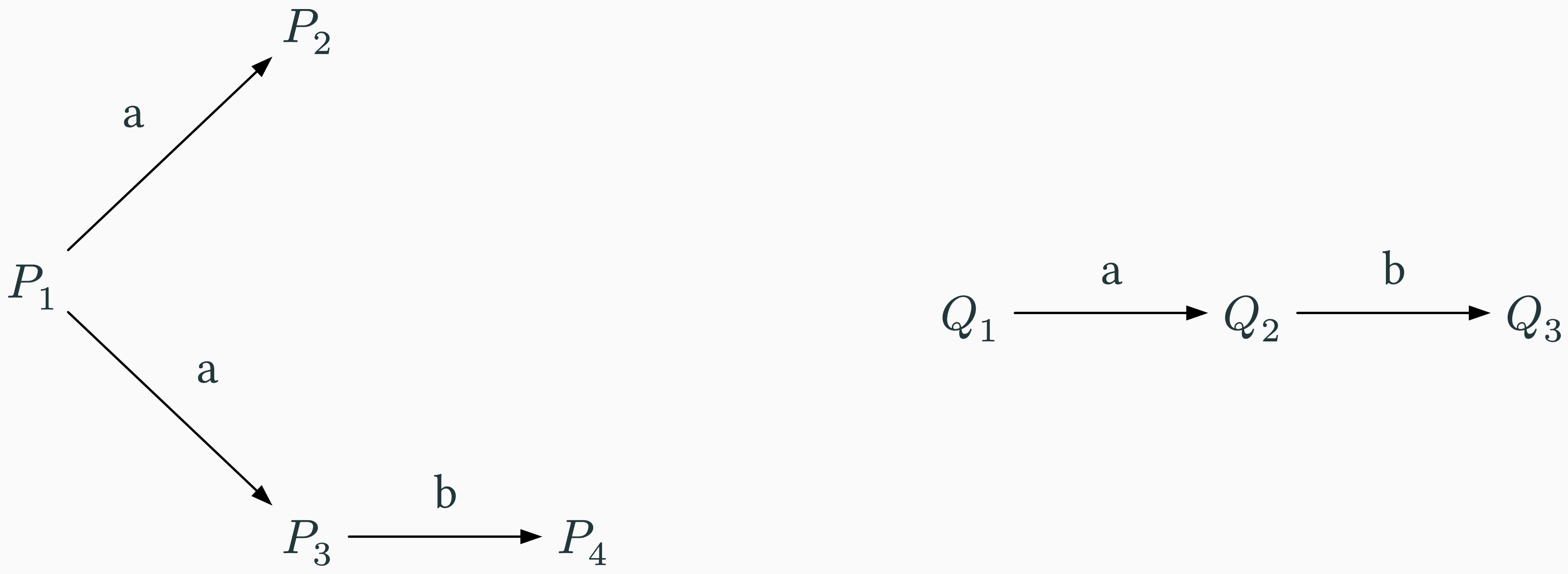


Another Example: $Q_1 \Leftrightarrow R_1$?





Counterexample (2) for Automata Theory Revisited



Definition 6 (Bisimilarity): A process relation $\mathcal{R}$ is a *bisimulation* if, for all $(P, Q) \in \mathcal{R}$ and all $\mu \in \mathrm{Act}$:

1. for P' with $P \overset{\mu}{\rightarrow} P'$, a Q' exists such that $Q \overset{\mu}{\rightarrow} Q'$ and $(P', Q') \in \mathcal{R}$;
2. for Q' with $Q \overset{\mu}{\rightarrow} Q'$, a P' exists such that $P \overset{\mu}{\rightarrow} P'$ and $(P', Q') \in \mathcal{R}$.

Bisimilarity, denoted by $\Leftrightarrow$, is the union of all bisimulations. Processes P and Q are *bisimilar*, consequently denoted by $P \Leftrightarrow Q$, if there is a bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.

Theorem 7:

1. $\Leftrightarrow$ is an equivalence relation.
2. $\Leftrightarrow$ is itself a bisimulation.

Corollary: $\Leftrightarrow$ is the largest bisimulation.

Proof of Theorem 7

ToDo: write down the proof here

Dwelling on the Corollary

Corollary: $\Leftrightarrow$ is the largest bisimulation.

Definition 8 (Bisimilarity): *Bisimilarity*, denoted by $\Leftrightarrow$, is the largest process relation such that $P \Leftrightarrow Q$ implies for all $\mu \in \mathrm{Act}$:

1. for P' with $P \xrightarrow{\mu} P'$, a Q' exists such that $Q \xrightarrow{\mu} Q'$ and $P' \Leftrightarrow Q'$;
2. for Q' with $Q \xrightarrow{\mu} Q'$, a P' exists such that $P \xrightarrow{\mu} P'$ and $P' \Leftrightarrow Q'$.

Strange: circular definition?

Strange: proof technique requires bisimulations $\mathcal{R}$ that have the same properties as $\Leftrightarrow$?

Outlook: $\Leftrightarrow$ is defined *coinductively*

$\rightsquigarrow$ see you again in four weeks

Next: Baseline Semantics of Programming Languages