# Concurrency Theory

## Lecture 2: Towards Bisimulation

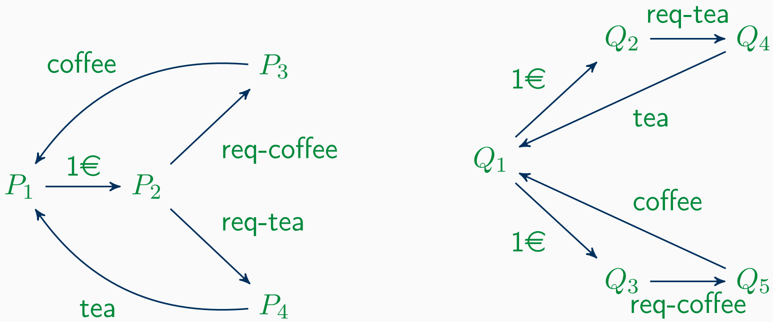
Stephan Mennicke
Knowledge-Based Systems Group

April 5, 2023

# Review

- Functions vs. processes
- LTSs for specification of process behaviors

Now: $P_1$ vs. $Q_1$



# Equality 1.0: Graph Theory – Isomorphisms?

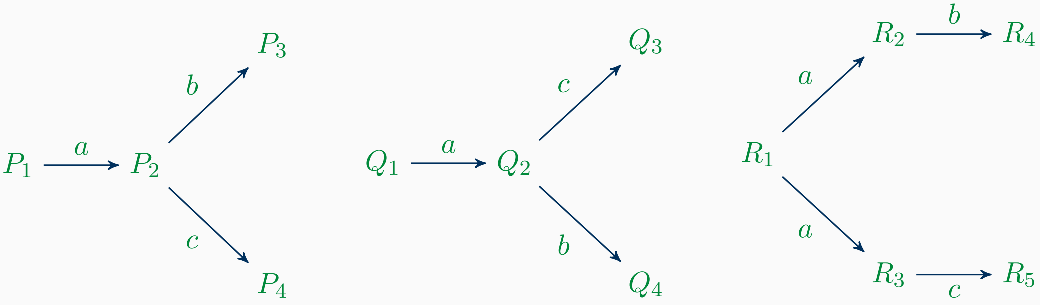
We use edge-labeled directed graphs to depict LTSs anyway.

**Definition 1 (Process Isomorphism)**
Let $\mathcal{T} = (Pr, Act, \rightarrow)$ be an LTS. A bijective function $f : Pr \rightarrow Pr$ is called an isomorphism if $p \xrightarrow{a} q$ if, and only if, $f(p) \xrightarrow{a} f(q)$. Process $P \in Pr$ is isomorphic to process $Q \in Pr$, denoted by $P \cong Q$, if there is an isomorphism $f$ such that $f(P) = Q$.

Remarks:

- single LTS $\mathcal{T}$ assumed; alternative: relation between two LTSs
- define equalities (equivalences, resp.) on process levels;
- call a relation $R \subseteq Pr \times Pr$ a process relation;

# Equality 1.0: Graph Isomorphisms in Action




Neither $P_1$ nor $Q_1$ are isomorphic to $R_1$. How do we prove it?

# Equality 1.0: Proof Methods for $\cong$

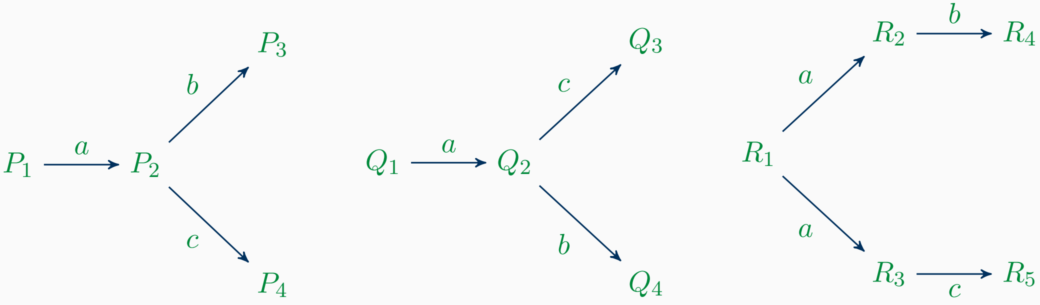
**Definition 2 (Isomorphism between LTSs)**
Let $\mathcal{T} = (Pr_1, Act, \rightarrow_1)$ and $\mathcal{U} = (Pr_2, Act, \rightarrow_2)$ be LTSs. $\mathcal{T}$ is isomorphic to $\mathcal{U}$ if there is a bijective function $f : Pr_1 \rightarrow Pr_2$, such that $p \xrightarrow{a}_1 q$ if, and only if, $f(p) \xrightarrow{a}_2 f(q)$.

**Theorem 3**

*Two processes $P$ and $Q$ are isomorphic if, and only if, their induced LTSs are (i. e., if $\mathcal{T}(P)$ and $\mathcal{T}(Q)$ are isomorphic).*

- To prove that $P \cong Q$, it suffices to give an isomorphism.
- To disprove the relation, the existence of one must be excluded.

# Equality 1.0: Graph Isomorphisms in Action (1/2)




We show that $P_1 \ncong R_1$. Consider $\mathcal{T}(P_1)$ with 4 states and $\mathcal{T}(R_1)$ having 5 states. There cannot by a bijection between these two LTSs. By Theorem 3, $P_1 \not\approx R_1$.

# Equality 1.0: Issues with ≅ as Process Equality

## Sequential Process Equality Issue


## Not Compositional w. r. t. External Nondeterminism

To be seen later...

## Conclusion

Process isomorphism (≅) is too strong.

# Equality 2.0: Automata Theory to the Rescue, Again?

Owing to the resemblence of LTSs to Nondeterministic Finite Automata (NFA), we may adopt NFA's language equivalence for processes.

To check whether $P$ and $Q$ are equivalent, consider $\mathcal{T}(P)$ and $\mathcal{T}(Q)$ as NFAs $\mathcal{A}(P)$ and $\mathcal{A}(Q)$ as follows:

- Initial state of $\mathcal{A}(P)$ is $P$ and that of $\mathcal{A}(Q)$ is $Q$;
- All states are final states;

This translation does not generally work. Why?

In the finite-state case, the words accepted by $\mathcal{A}(P)$ are called the traces of $P$. $P$ and $Q$ are trace equivalent if they have the same traces.

We use this resemblence but need to get rid of the finite-state assumption.

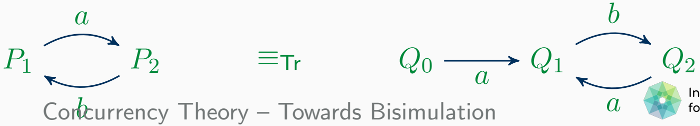
# Equality 2.0: Trace Equivalence

As for automata, the labeled transition relation can be generalized to words/traces:

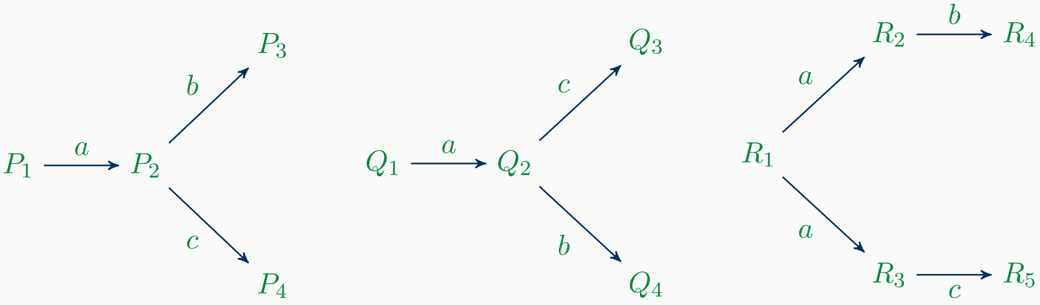
- $P \stackrel{\varepsilon}{\Rightarrow} P$ where $\varepsilon$ is the empty word;
- if $P \stackrel{\sigma}{\Rightarrow} P'$ and $P' \stackrel{a}{\rightarrow} P''$, then $P \stackrel{\sigma a}{\Longrightarrow} P''$.

Notations like $P \stackrel{\sigma}{\Rightarrow}$ and $P \stackrel{\sigma}{\not\Rightarrow}$ can also be generalized.

**Definition 4 (Trace Equivalence)**
Let $\mathcal{T} = (Pr, Act, \rightarrow)$ be an LTS and $P \in Pr$. The set of traces of $P$, denoted by $\mathsf{Tr}(P)$, is defined as the set $\{\sigma \in Act^* \mid \exists P' \in Pr : P \stackrel{\sigma}{\Rightarrow} P'\}$. $P$ and $Q$ are trace equivalent, denoted $P \equiv_{\mathsf{Tr}} Q$, if, and only if, $\mathsf{Tr}(P) = \mathsf{Tr}(Q)$.



# Equality 2.0: Trace Equivalence in Action




How to prove or disprove trace equivalence? What is the complexity of doing so?

# Equality 2.0: Trace Equivalence is Pspace-complete (1/2)

Of course, we consider the finite-state case only: For finite-state processes $P$ and $Q$, check whether $P \equiv_{\mathsf{Tr}} Q$ holds.

Membership in Pspace is immediate from the following observation:

- $\mathsf{Tr}(P)$ is a regular language;
- for languages $\mathbb{L}_1, \mathbb{L}_2$, $\mathbb{L}_1 = \mathbb{L}_2 \iff \mathbb{L}_1 \subseteq \mathbb{L}_2$ and $\mathbb{L}_2 \subseteq \mathbb{L}_1$ and $\mathbb{L}_1 \subseteq \mathbb{L}_2 \iff \mathbb{L}_1 \cap \overline{\mathbb{L}_2} = \emptyset$.
- Take $\mathcal{A}(P)$ and $\mathcal{A}(Q)$;
- construct the complement of $\mathcal{A}(Q)$ (yields exponential blow-up);
- to keep up with Pspace, compute the complement of $\mathcal{A}(Q)$ on-demand.

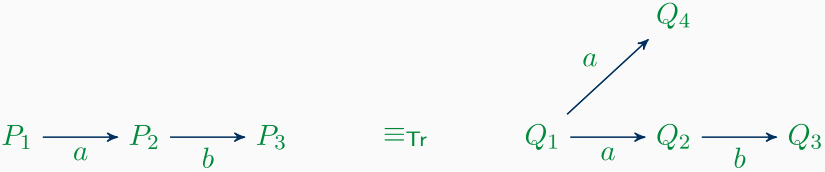
# Equality 2.0: Trace Equivalence is Pspace-complete (2/2)

Hardness by reduction from language equivalence of NFAs:

- Turn an NFA $\mathcal{A} = (\mathcal{Q}, q_0, F, \delta)$ into an LTS; (how do we do that?)
- Do the same for NFA $\mathcal{B}$;
- Now $\mathcal{A}$ and $\mathcal{B}$ accept the same language if, and only if, their respective processes (represented inside the translated LTSs) are trace equivalent.

# Equality 2.0: Issues with Trace Equivalence

## Deadlocks




What if $a$ means *insert money* and $b$ is for *receive product*?

We say that $Q_4$ is a blocked process or a deadlock.

# Formalizing Deadlocks

## First Attempt

A process $P$ is a deadlock if for all $a \in Act$, $P \overset{a}{\not\rightarrow}$.

We aim for process relations that preserve traces as well as deadlocks. But how?

## Completed Traces Approach

A completed trace is a trace that may end in a deadlock.

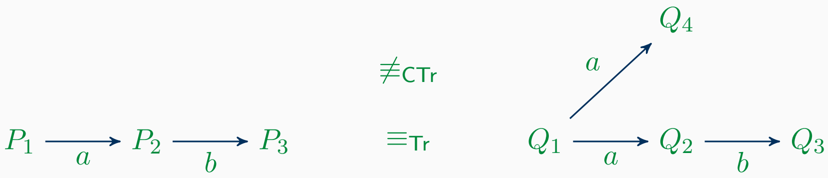
# Equality 2.1: Completed Trace Equivalence

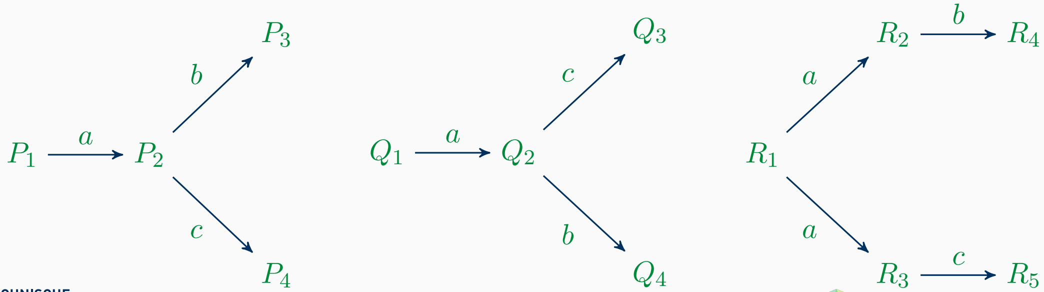
**Definition 5 (Completed Trace Equivalence)**
A trace $\sigma \in \mathsf{Tr}(P)$ is a completed trace of $P$ if there is a $P'$ such that $P \stackrel{\sigma}{\Longrightarrow} P'$ and $P'$ is a deadlock. The set of all completed traces is denoted by $\mathsf{CTr}(P)$. Processes $P$ and $Q$ are completed trace equivalent, denoted $P \equiv_{\mathsf{CTr}} Q$, if (1) $P \equiv_{\mathsf{Tr}} Q$ and (2) $\mathsf{CTr}(P) = \mathsf{CTr}(Q)$.

**Theorem 6**
$\equiv_{CTr} \subseteq \equiv_{\mathsf{Tr}}$.

**Proof.**
Immediate consequence of the definition of $\equiv_{\mathsf{CTr}}$. □

# Equality 2.1: Completed Traces in Action



# On Good Process Relations

**Trace Preservation:** traces formalize the sequential (observable) runs of a system. If two systems differ in their observable runs, they also interact differently with their environment.

**Deadlock-Sensitiveness:** we want to distinguish processes that have different deadlocks since deadlocks are those processes with which no environment may interact.

**Compositional Reasoning:** in order to formalize this point properly, we need some more tools first (lecture 4 on CCS).

A good process relation $\equiv$ preserves traces, is deadlock-sensitive, and admits compositional reasoning.

The first two items imply that $\equiv \subseteq \equiv_{\mathsf{CTr}}$ must hold. Is $\equiv_{\mathsf{CTr}}$ a *good process relation*?

# Bisimilarity – A Good Process Relation

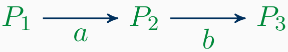
**Definition 7 (Bisimilarity)**
A process relation $R$ is a bisimulation if, whenever $P\,R\,Q$,

- for all $P' \in Pr$ and $a \in Act$, $P \xrightarrow{a} P'$ implies there is a $Q' \in Pr$ such that $Q \xrightarrow{a} Q'$ and $P'\,R\,Q'$; and
- for all $Q' \in Pr$ and $a \in Act$, $Q \xrightarrow{a} Q'$ implies there is a $P' \in Pr$ such that $P \xrightarrow{a} P'$ and $P'\,R\,Q'$.

Processes $P$ and $Q$ are bisimilar, written $P \leftrightarrow Q$, if there is a bisimulation $R$ such that $P\,R\,Q$. Process relation $\leftrightarrow$ is also called bisimilarity.

**Theorem 8**
*$\leftrightarrow$ is a good process relation. For now, $\leftrightarrow \subseteq \equiv_{CTr}$.*

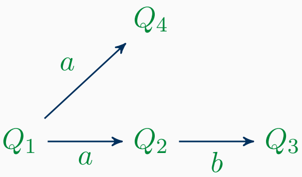
# Bisimilarity in Action

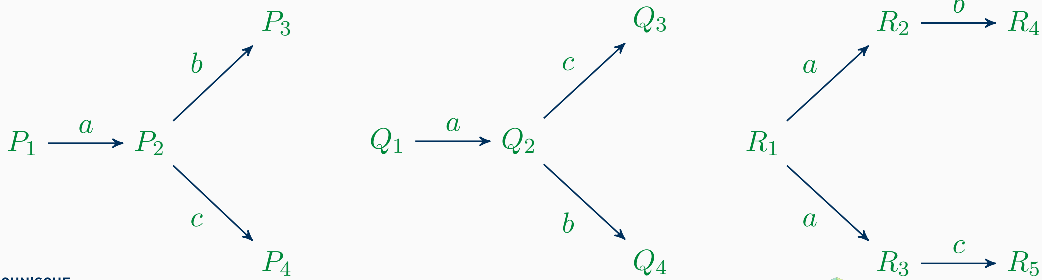



$\not\sim$

$\not\equiv_{\mathsf{CTr}}$

$\equiv_{\mathsf{Tr}}$



# Bisimilarity – Some Useful Results

**Theorem 9**
$\leftrightarrow$ *is an equivalence relation.*

**Theorem 10**
$\leftrightarrow$ *is the largest bisimulation.*

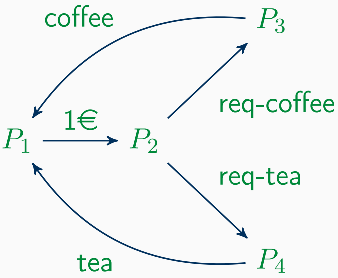
**Theorem 11**
*A process relation* $R$ *is a bisimulation up to* $\leftrightarrow$ *if, whenever* $P \, R \, Q$, *we have for all* $a \in Act$*:*

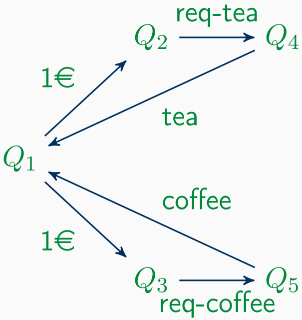
1. $P \xrightarrow{a} P'$ *implies there is a* $Q' \in Pr$ *such that* $Q \xrightarrow{a} Q'$ *and* $P' \leftrightarrow R \leftrightarrow Q'$, *and*
2. $Q \xrightarrow{a} Q'$ *implies there is a* $P' \in Pr$ *such that* $P \xrightarrow{a} P'$ *and* $P' \leftrightarrow R \leftrightarrow Q'$.

*If* $R$ *is a bisimulation up to* $\leftrightarrow$, *then* $R \subseteq \leftrightarrow$.

# Summary

- (Completed) trace equivalence;
- Bisimilarity as a good equivalence



# Outlook

- A lot of examples
- The bisimulation proof method (coinduction)
- More on coinduction
- Fixpoints and bisimulation games