Concurrency: Languages, Programming and Theory

– Equivalences for Concurrency –

Session 10 – January 7, 2004

Uwe Nestmann

EPFL-LAMP

Repetition of Algebraic Notions

relations/functions

- composition
- comparison, containment

preorder/equivalence

- reflexivity
- symmetry
- transitivity
- kernel of a (reflexive) preorder
- comparison, containment vs fine/coarse

congruence

- by definition?

Automata

An **automaton** $A = (Q, q_0, F, T)$
over an **action alphabet** *Act*:

- □ a set $Q = \{q_0, q_1 \ldots\}$: the **states**
- □ a state $q_0 \in Q$: the **start state**
- □ a subset $F \subseteq Q$: the **accepting states**
- □ a subset $T \subseteq (Q \times \mathit{Act} \times Q)$: the **transitions**

A transition $(q, \alpha, q') \in T$ is also written $q \xrightarrow{\alpha} q'$.

Example Automaton

Let *Act* be $\{a, b, c\}$.
Let A be defined as

$$
\begin{array}{ll}
(& \{q_0, q_1, q_2, q_3\}, \\
 & q_0, \\
 & \{q_1\}, \\
 & \{ \quad (q_0, b, q_3), (q_0, c, q_3), (q_0, a, q_1), \\
 & \qquad (q_1, c, q_0), (q_1, a, q_3), (q_1, b, q_2), \\
 & \qquad (q_2, c, q_0), (q_2, a, q_3), (q_2, b, q_3), \\
 & \qquad (q_3, c, q_3), (q_3, a, q_3), (q_3, b, q_3), \\
 & \} \\
) &
\end{array}
$$

Automata (II)

An automaton A is

- **finite-state**, if Q is finite, and
- **deterministic** if for each pair $(q, \alpha) \in Q \times \mathit{Act}$ there is **exactly one transition** $q \xrightarrow{\alpha} q'$.
  (Note the similarity to a function $Q \times \mathit{Act} \rightarrow Q$.)

Question: Would the formulation "**at most one transition**" yield less deterministic automata?

Note: "Complete" an automaton?

Behavior: *Language* of an Automaton

Let A be an automaton over *Act*.
Let $s = \alpha_1 \ldots \alpha_n$ be a string over *Act*. Then:

- A is said to **accept** s, if there is a path in A — from q_0 to some accepting state — whose arcs are labeled successively $\alpha_1 \ldots \alpha_n$.
- The **language** of A, denoted by $\widehat{A}$, is the set of strings accepted by A.

ϵ denotes the empty string.

Fact: The language $\widehat{A}$ of any finite-state automaton A is *regular*.

Regular Sets

(* *a mathematical model* *)

Definition: A set of strings over *Act* is **regular** if it can be built from

- the **empty set** $\emptyset$ and the **singleton** sets $\{\alpha\}$ ($\forall \alpha \in$ *Act*),
- using the operations of
  - **union** ($\cup$),
  - **concatenation** ($\cdot$), and
  - **iteration** (*).

$$S_1 \cdot S_2 \stackrel{\mathrm{def}}{=} \{s_1 \cdot s_2 \mid s_1 \in S_1 \wedge s_2 \in S_2\}$$
$$S^* \stackrel{\mathrm{def}}{=} \{\epsilon\} \cup S \cup S{\cdot}S \cup S{\cdot}(S{\cdot}S) \cup \ldots$$

In regular sets, we sometimes write α for $\{\alpha\}$ and ϵ for $\{\epsilon\}$.

Regular Expressions

(* *syntax to indicate the elements of the mathematical model* *)

Definition: The set of **regular expressions** over *Act* is generated by the following grammar:

$$E \quad ::= \quad \epsilon \quad \Big| \quad \alpha \quad \Big| \quad E + E \quad \Big| \quad E \cdot E \quad \Big| \quad E^*$$

where $\alpha \in$ *Act*.
In regular expressions, we often write $\alpha\beta$ for $\alpha \cdot \beta$ . . .

regular expressions	*regular sets*
$(a+b)c$, $ac+bc$	$\{ac, bc\}$
$a+bc$	$\{a, bc\}$

“Denotational Semantics”

$$
\begin{array}{rcl}
\text{RegExps} & \longrightarrow & \text{RegSets} \\
\hline
[\![\, \epsilon \,]\!] & \stackrel{\text{def}}{=} & \{\epsilon\} \\
[\![\, \alpha \,]\!] & \stackrel{\text{def}}{=} & \{\alpha\} \\
[\![\, E_1 + E_2 \,]\!] & \stackrel{\text{def}}{=} & [\![\, E_1 \,]\!] \cup [\![\, E_2 \,]\!] \\
[\![\, E_1 \cdot E_2 \,]\!] & \stackrel{\text{def}}{=} & [\![\, E_1 \,]\!] \cdot [\![\, E_2 \,]\!] \\
[\![\, E^* \,]\!] & \stackrel{\text{def}}{=} & [\![\, E \,]\!]^*
\end{array}
$$

- in the image of the semantics function $[\![\ ]\!]$, all of $\cup$, $\cdot$, and *, are operators on sets so they entail the calculation of the actual set that they represent
- compare to Arithmetic Expressions and Natural Numbers
- note that $[\![\ ]\!]$ is not surjective . . . ***why?***

Some Laws on Regular Expressions

$$\begin{aligned}(E_1 \cdot E_2) \cdot E_3 &= E_1 \cdot (E_2 \cdot E_3)\\ E \cdot \epsilon &= E\\ E \cdot \emptyset &= \emptyset\end{aligned}$$

$$\begin{aligned}(E_1 + E_2) \cdot E_3 &= E_1 \cdot E_3 + E_2 \cdot E_3\\ E_3 \cdot (E_1 + E_2) &= E_3 \cdot E_1 + E_3 \cdot E_2\end{aligned}$$

$$E_1 \cdot (E_2 \cdot E_1)^* = (E_1 \cdot E_2)^* \cdot E_1$$

Be Careful . . .

Note:
The regular set $\emptyset$ means "no path". But:
The regular expression ϵ means "empty path".

$$\emptyset \neq \{\epsilon\}$$

As an example, compare $\{\alpha\beta\} \cdot \{\epsilon\}$ with $\{\alpha\beta\} \cdot \emptyset$.

Arden's rule

Theorem:
For any sets of strings S and T, the equation

$$X = S \cdot X + T \qquad \text{has} \qquad X = S^* \cdot T$$

as a **solution**.
Moreover, this solution is unique if $\epsilon \notin S$.

Example Automaton

Determine the language of the previous automaton
as the regular expression describing
the strings accepted in the initial state.

Write down a set of equations,
one equation for each state.

Solve the set of equations . . .

Determinism / Nondeterminism

Analyze the two automata of § 2.4 of [Mil99].

Message1:
Language equivalence is blind for nondeterminism!

In fact, every nondeterministic automaton can be converted into a determinstic one that accepts the same language.

Message2:
Language equivalence is blind for deadlocks!

Example?

Message3 (less important):
Language equivalence requires accepting states.

Labeled Transition Systems

Definition:

An **LTS** $L = (Q, T)$ over an **action alphabet** *Act*:

- a set of **states** $Q = \{q_0, q_1 \ldots\}$
- a ternary **transition relation** $T \subseteq (Q \times \mathit{Act} \times Q)$

A transition $(q, \alpha, q') \in T$ is also written $q \stackrel{\alpha}{\longrightarrow} q'$.

If $q \stackrel{\alpha_1}{\longrightarrow} q_1 \cdots \stackrel{\alpha_n}{\longrightarrow} q_n$ we call q_n a **derivative** of q.

Equivalence on LTS ?

Example: Compare p_0 and q_0 in

$$\{ \quad (p_0, a, p_1), (p_1, b, p_2), (p_1, c, p_3), \\ (q_0, a, q_1), (q_0, a, q_1'), (q_1, b, q_2), (q_1', c, q_3) \quad \}$$

Induce simulation of paths
through step-by-step simulation of actions . . .

(Strong) Simulation on LTS

<u>Definition:</u> **(learn it by heart!)**
Let (Q, T) be an LTS.

1. Let $\mathcal{S}$ be a binary relation over Q.
   $\mathcal{S}$ is a **(strong) simulation** over (Q, T) if, whenever $p \; \mathcal{S} \; q$,

if $p \stackrel{\alpha}{\longrightarrow} p'$ then there is $q' \in Q$ such that $q \stackrel{\alpha}{\longrightarrow} q'$ and $p' \; \mathcal{S} \; q'$

   .

2. q **(strongly) simulates** p, written $p \preceq q$,
   if there is a (strong) simulation $\mathcal{S}$ such that $p \; \mathcal{S} \; q$.

The relation $\preceq$ is sometimes called *similarity*.

Properties of Simulations

Lemma:

If $\mathcal{S}_1$ and $\mathcal{S}_2$ are simulations, then

- $\mathcal{S}_1 \cup \mathcal{S}_2$ is also a simulation.
- $\mathcal{S}_1 \cap \mathcal{S}_2$ is also a simulation ?
- $\mathcal{S}_1\mathcal{S}_2$ is also a simulation ?

Definition: Let (Q, T) be a LTS.

$$\preceq \stackrel{\text{def}}{=} \bigcup \{ \ \mathcal{S} | \mathcal{S} \text{ is simulation over } (Q, T) \ \}$$

Lemma:

- $\preceq$ is the largest simulation over (Q, T).
- $\preceq$ is a reflexive preorder over $Q \times Q$.

Is any simulation a preorder?

Working with Simulation

What do we do with simulations?

- exhibiting a simulation:
  "*guessing*" a relation $\mathcal{S}$ that contains (p, q)
- checking a simulation:
  check that a given relation $\mathcal{S}$ is in fact a simulation.

Fortunately, clever people developed algorithms and respective tools (CWB, ABC) that are good at "guessing" simulations.

In fact, they ***generate*** relations algorithmically that—by construction—fulfil the property of being a simulation.

Results on (semi-)decidability are very important for such tools.

Home-Working with Simulation

Example: Find all non-trivial simulations in

$$\{(1, b, 2), (1, c, 3), (4, b, 5), (6, b, 7), (6, c, 8), (6, c, 9)\}$$

How many are there ?

Trivial pairs are any pairs with elements from $\{2, 3, 5, 7, 8, 9\}$ (because there are no transitions),
as well as any identity on $\{1, 4, 6\}$.

Trivial simulations are those that either
(0) are empty, or
(1) contain only trivial pairs, or
(2) contain at least one trivial pair that is not reachable from a contained non-trivial one.

Towards Equivalence

Simulation is only a preorder,
thus it allows us to *distinguish* states.

We want instead an equivalence,
which would allow us to *equate* states.

The mathematical way: just take the “kernel”

$$p = q \qquad \text{if} \qquad p < q \qquad \text{and} \qquad q < p$$

However, there are two different natural candidates !

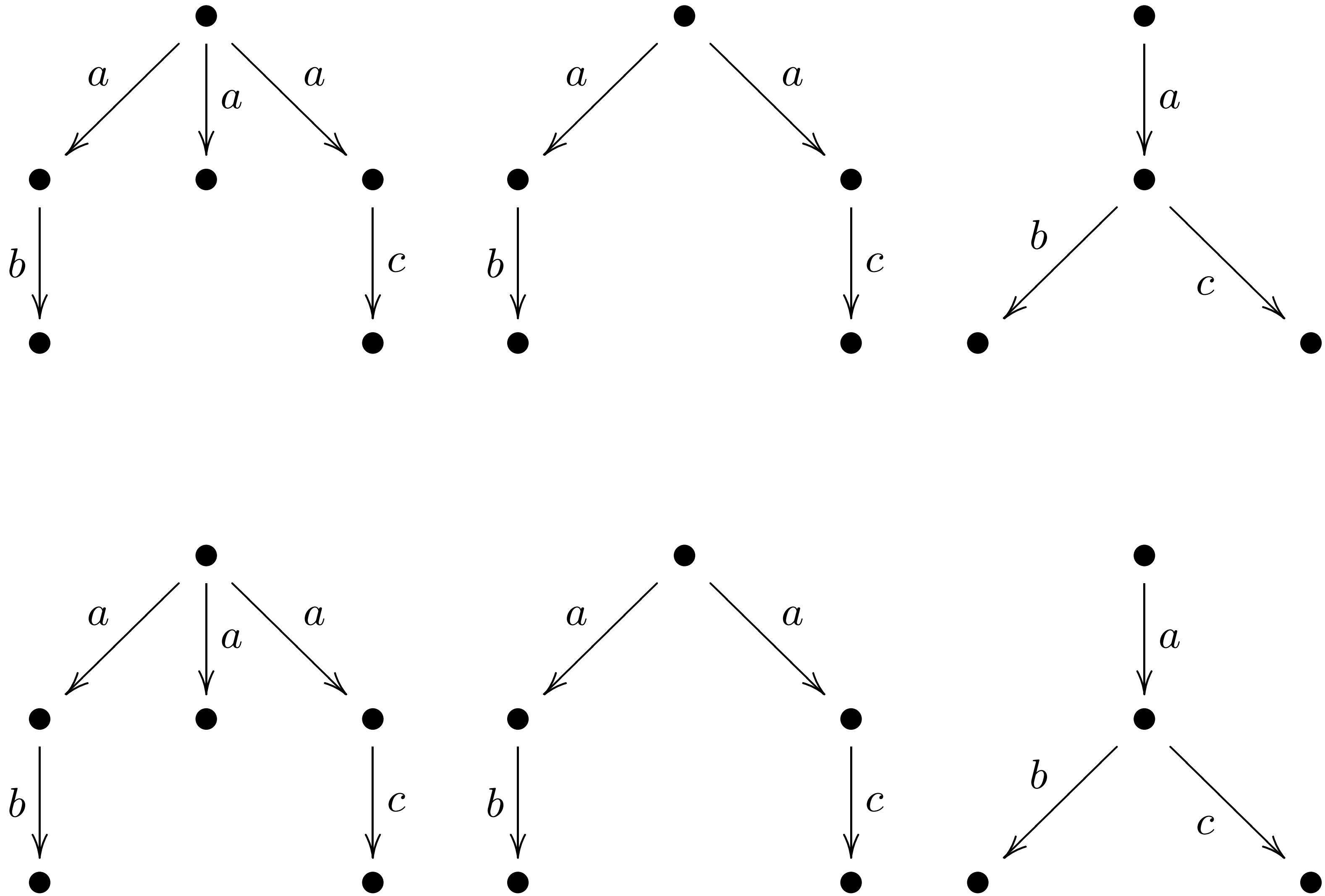
- mutual simulation
- bisimulation

Mutual Simulation: Back and Forth

Definition:
Let (Q, T) be a LTS. Let $\{p, q\} \subseteq Q$.

p and q are **mutually similar**, written $p \gtrless q$,
if there is a pair $(\mathcal{S}_1, \mathcal{S}_2)$ of simulations $\mathcal{S}_1$ and $\mathcal{S}_2$
with $p \; \mathcal{S}_1 \; q \; \mathcal{S}_2 \; p$ (i.e., with $p \; \mathcal{S}_1 \; q$ and $q \; \mathcal{S}_2 \; p$).

Example: Mut. Sim. vs Lang. Equiv.



Mutual Simulation (II)

Proposition:

- ☐ $\gtrless$ is an equivalence relation.

Proof?

Typical research-craftsmen work . . .

$$p \gtrless q \qquad\qquad \mathrm{Lang}(p) = \mathrm{Lang}(q)$$

$$\gtrless \qquad\qquad =_{\mathrm{Lang}}$$

(Strong) Bisimulation

Definition:**(learn it by heart!)**
A binary relation $\mathcal{B}$ over Q is
a **(strong) bisimulation** over the LTS (Q, T)
if both $\mathcal{B}$ and its converse $\mathcal{B}^{-1}$ are (strong) simulations.

p and q are **(strongly) bisimilar**, written $p \sim q$,
if there is a (strong) bisimulation $\mathcal{B}$ such that $p \; \mathcal{B} \; q$.

Alternatively:

$$\sim \stackrel{\mathrm{def}}{=} \bigcup \{ \; \mathcal{B} \mid \mathcal{B} \text{ is (strong) bisimulation over } (\mathcal{Q}, \mathcal{T}) \; \}$$

(Strong) Bisimulation (II)

Proposition:

- $\sim$ is (itself) a (strong) bisimulation.
- $\sim$ is an equivalence relation.

Proof?

Again, typical research-craftsmen work . . .

Example

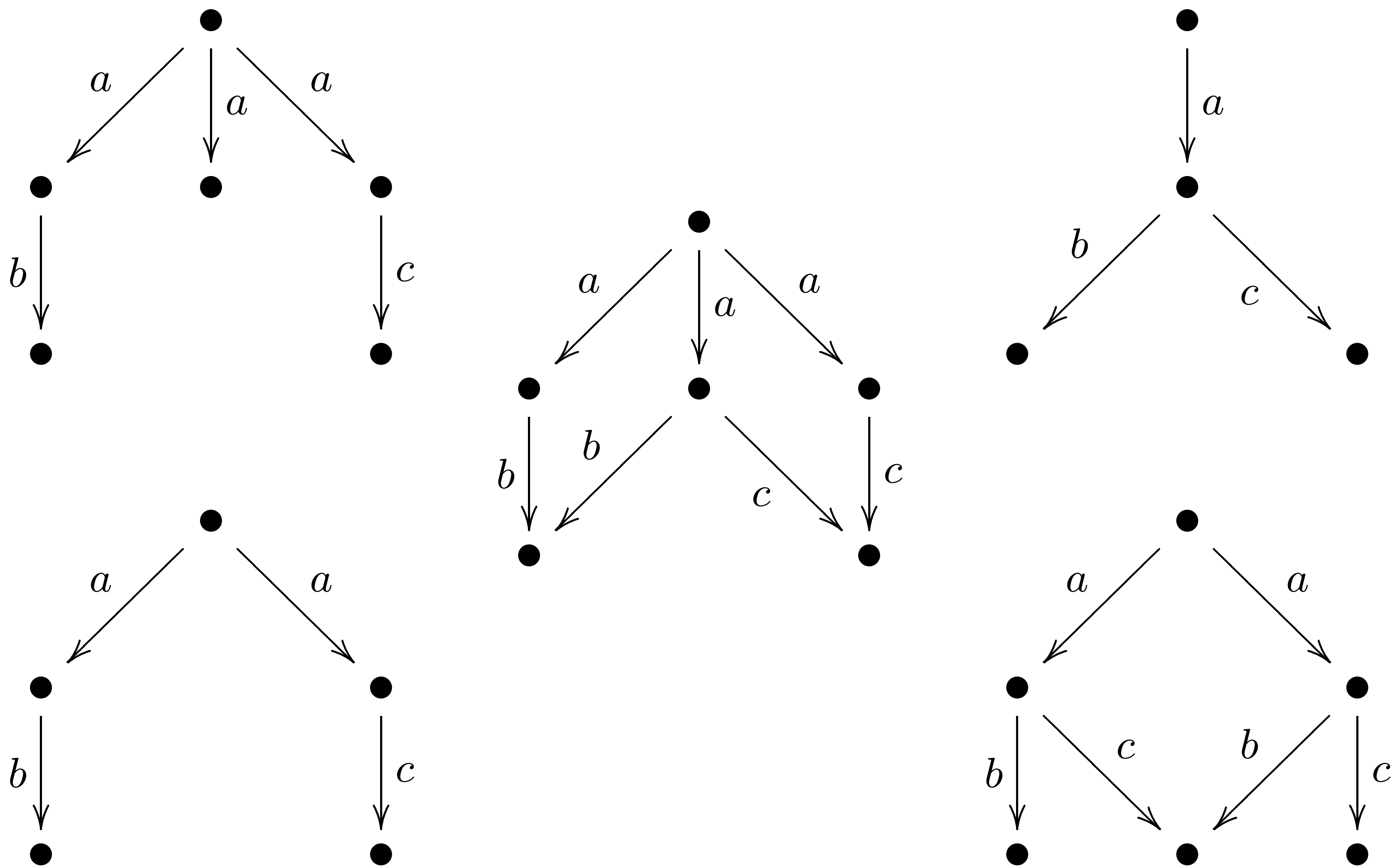
$$\{ (1,a,2),(1,a,3),(2,a,3),(2,b,1),(3,a,3),(3,b,1),$$
$$(4,a,5),(5,a,5),(5,b,6),(6,a,5),$$
$$(7,a,8),(8,a,8),(8,b,7) \}$$

Prove $1 \sim 4 \sim 6 \sim 7$.

Write out $\sim \ldots$

Minimization ?!

Example: Mutual vs Bi



Isomorphism on LTS

Definition:
Let (Q_i, T_i) be two LTS over *Act* for $i \in \{1, 2\}$.

(Q_1, T_1) and (Q_2, T_2) are **isomorph(ic)**,
written $(Q_1, T_1) \cong (Q_2, T_2)$,
if there is a **bijection** f on between Q_1 and Q_2
that preserves T, i.e., $f : Q_1 \to Q_2$ with

$$q \xrightarrow{\alpha} q' \quad \text{iff} \quad f(q) \xrightarrow{\alpha} f(q').$$

Isomorphism on LTS (II)

Proposition:

- $\cong$ is an equivalence relation
  (on the domain of LTSs).

Proof?

Be careful with the interpretation of reflexivity, symmetry, and transitivity . . .

Isomorphism vs Bisimulation

"Problem":
Isomorphism compares two transition systems;
Bisimulation (at least as we have defined it) compares two states.

Redefine $\mathcal{B} \subseteq Q_1 \times Q_2$ to be a bisimulation
if $\mathcal{B}$ and $\mathcal{B}^{-1}$ are simulations on their respective domains, i.e., $\mathcal{B}^{-1} \subseteq Q_2 \times Q_1$.

Redefine $\sim$ to the whole domain of LTSs.
Be careful with the interpretation of reflexivity, symmetry, and transitivity . . .

Isomorphism vs Bisimulation

1. **reachability**
   $(Q_1, T_1) = (\{q_1^0, q_1^1, q_1^2\}, \{(q_1^0, a, q_1^1)\})$
   $(Q_2, T_2) = (\{q_2^0, q_2^1\}, \{(q_2^0, a, q_2^1)\})$

Isomorphism vs Bisimulation

2. **copying**

$$
\begin{aligned}
(Q_1, T_1) = (\ & \{\ q_1^0, q_1^1, q_1^2\ \}, \\
& \{\ (q_1^0, a, q_1^1), (q_1^1, b, q_1^2), (q_1^1, c, q_1^3)\ \}\ ) \\
(Q_2, T_2) = (\ & \{\ q_2^0, q_2^1, q_2^2, q_2^3, \underline{{q'}_2^1, {q'}_2^2, {q'}_2^3}\ \}, \\
& \{\ (q_2^0, a, q_2^1), (q_2^1, b, q_2^2), (q_2^1, c, q_2^3), \\
& \quad (q_2^0, a, {q'}_2^1), ({q'}_2^1, b, {q'}_2^2), ({q'}_2^1, c, {q'}_2^3)\ \}\ )
\end{aligned}
$$

Isomorphism vs Bisimulation

3. **recursion/unfolding**

$$(Q_1, T_1) = (\{q_1^i \mid i \in \mathbb{N}_0\}, \{(q_1^i, a, q_1^{i+1}) \mid i \in \mathbb{N}_0\})$$
$$(Q_2, T_2) = (\{q_2^0\}, \qquad \{(q_2^0, a, q_2^0)\} \qquad)$$

Which is the Best Equivalence ?

language equivalence
mutual simulatity
bisimilarity
isomorphism
identity

$$= \qquad \cong \qquad \sim \qquad \gtrless \qquad =_{\mathrm{L}}$$

To be remembered:
What are the intuitive distinguishing aspects
between all of these notions of equivalence? ($\rightarrow$ Exam . . .)