Sangiorgi: Introduction to Bisimulation and Coinduction

Johannes Åman Pohjola

Uppsala University

June 1, 2012

Outline

After this seminar, you should (hopefully):

- Understand what bisimulation is.
- Know how to use the bisimulation proof method.
- Kind of sort of vaguely understand coinduction and its duality with induction.

Introduction

We seek a *behavioural equivalence* between processes.

That is, an equivalence relation that relates processes that exhibit the same observable behaviour.

What would be a sensible such equivalence?

And besides, what is a process anyway?

LTS

We will model processes and their behaviour using *labelled transition systems* (LTS).

Definition (LTS)

An LTS is a triple $(Pr, Act, \longrightarrow)$ where

- Pr is a non-empty set called the *domain*
- Act is the set of *labels*
- $\longrightarrow \subseteq Pr \times Act \times Pr$ is the *transition relation*.

We will write $P \xrightarrow{\alpha} Q$ when $(P, \alpha, Q) \in \longrightarrow$.

We call elements of Pr *states*, or interchangeably, *processes*.

Graph isomorphism

An LTS is similar to a graph, so let's try graph isomorphism!

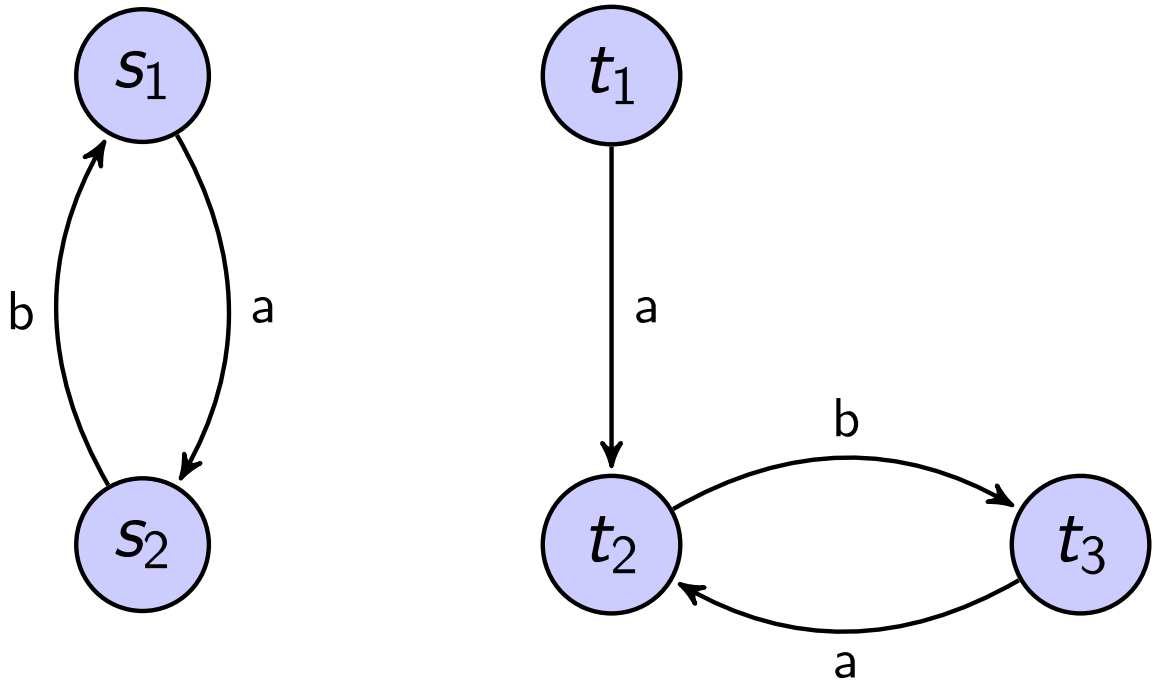
Two graphs are isomorphic if there is a bijection between their components: the states and the transitions.

Two isomorphic LTSs would certainly have the same behaviour.

Unfortunately, the converse is false.

Graph isomorphism

These two LTSs have the same behaviour:



But they are not isomorphic.

Trace equivalence

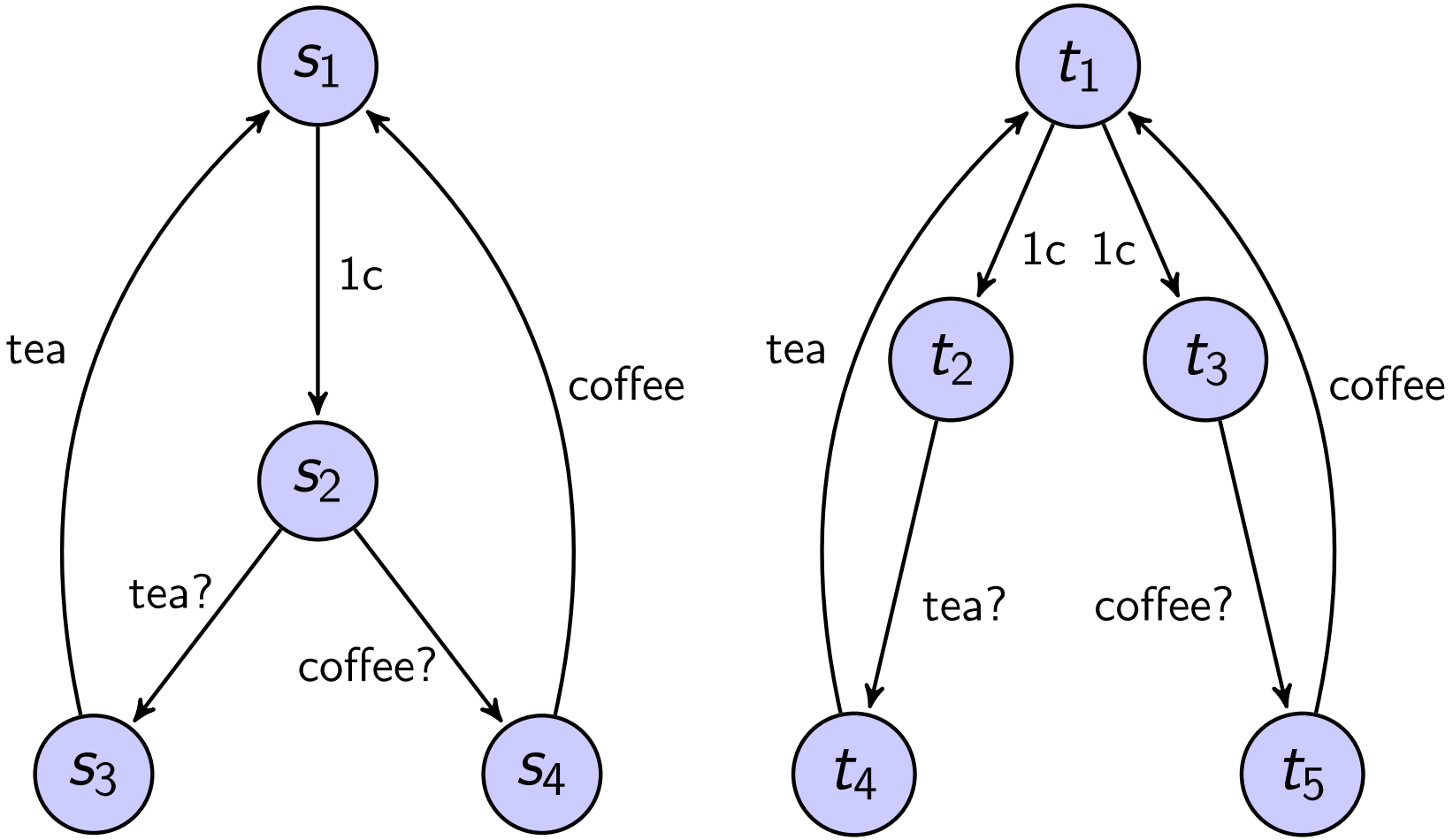
LTSs are similar to automata, so let's try language equivalence!

Two automata are language equivalent if they accept the same set of strings.

Analogously, two LTS processes are *trace equivalent* if they give rise to the same (finite) transition sequences.

Trace equivalence

These coffee machines are trace equivalent.



But not behaviourally equivalent.

Bisimulation

Definition (Bisimulation)

A binary relation $\mathcal{R}$ on the states of an LTS is a *bisimulation relation* iff whenever $P \mathcal{R} Q$,

1. For all P' such that $P \xrightarrow{\alpha} P'$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$
2. For all Q' such that $Q \xrightarrow{\alpha} Q'$, there exists P' such that $P \xrightarrow{\alpha} P'$ and $P' \mathcal{R} Q'$

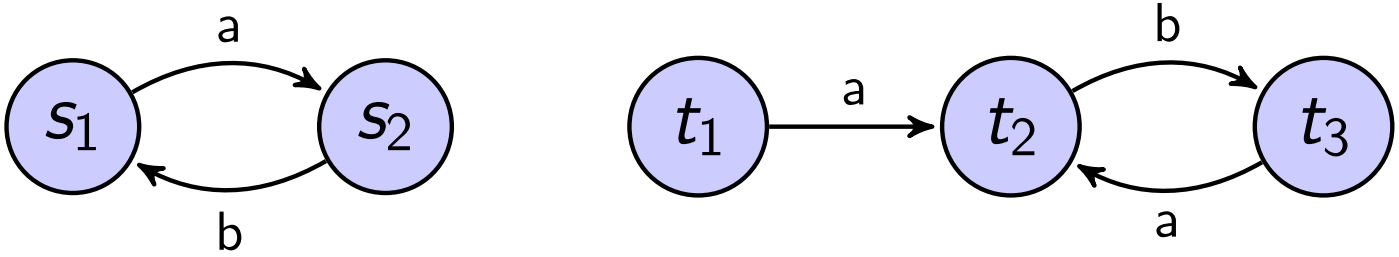
Bisimilarity, denoted $\sim$, is the union of all bisimulations. Hence $P \sim Q$ iff there exists a bisimulation $\mathcal{R}$ with $P \mathcal{R} Q$.

Bisimulation proof method

Immediately from the definition of bisimulation, we obtain the *bisimulation proof method*:

To prove $P \sim Q$, find a bisimulation $\mathcal{R}$ such that $P \mathcal{R} Q$.

Bisimulation proof method: examples



Theorem

$s_1 \sim t_1$

Proof.

Pick $\mathcal{R} = \{(s_1, t_1), (s_2, t_2), (s_1, t_3)\}$.
Then check that all transitions from $\mathcal{R}$ take us back to $\mathcal{R}$. □

Hint: be lazy! Pick the smallest possible candidate relation!

Bisimulation proof method: examples

Theorem

$\sim$ *itself is a bisimulation.*

Proof.

By intimidation. □

Theorem

$\sim$ *is an equivalence relation, ie for all* P, Q *and* R*:*

1. $P \sim P$
2. $P \sim Q \Rightarrow Q \sim P$
3. $P \sim Q \wedge Q \sim R \Rightarrow P \sim R$

Proof.

On blackboard. □

Simulation

Definition (Simulation)

A binary relation $\mathcal{R}$ on the states of an LTS is a *simulation relation* iff whenever $P \mathcal{R} Q$,

1. For all P' such that $P \xrightarrow{\alpha} P'$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \mathcal{R} Q'$
2. There is no second clause

We say that Q *simulates* P iff there exists a simulation $\mathcal{R}$ and $P \mathcal{R} Q$.

Intuitively, this means that the behaviour of Q includes the behaviour of P.

Simulation

One might expect the following to hold:

Proposition

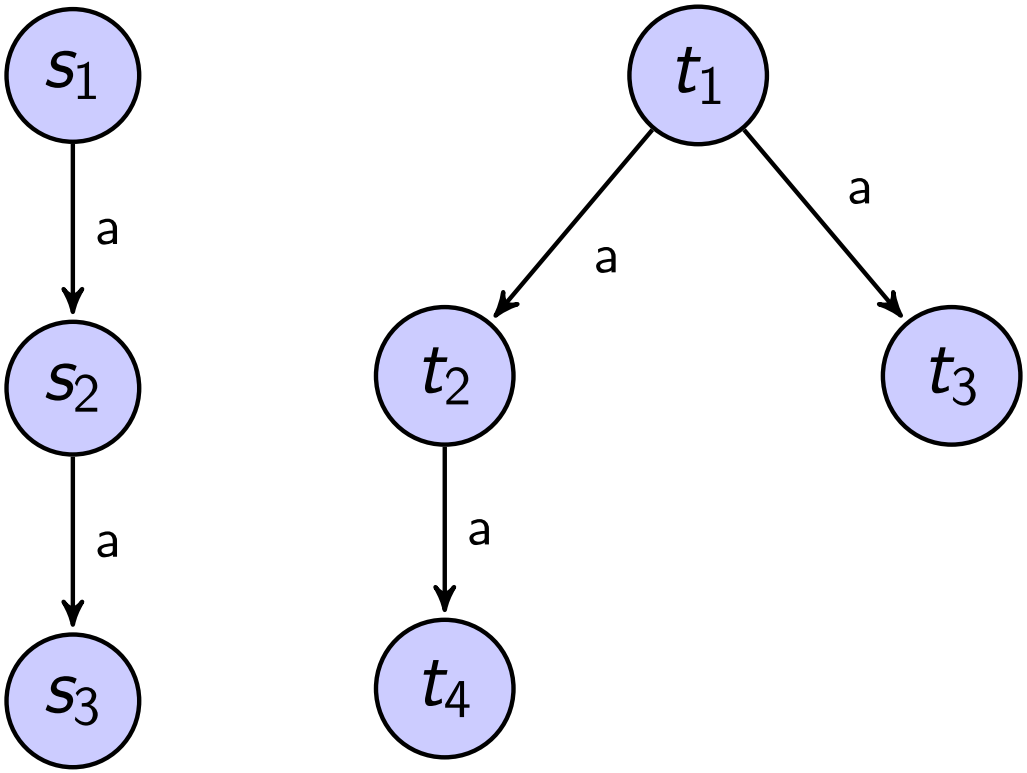
If P simulates Q and Q simulates P, then $P \sim Q$.

And one would rejoice, since it would allow for simpler bisimulation proofs.

Unfortunately, it is false.

Simulation

Here s_1 simulates t_1, and vice versa.



But $s_1 \sim t_1$ does *not* hold.

Coinduction

Since $\sim$ itself is a bisimulation, we could define $\sim$ as:

Definition

$\sim$ is the largest bisimulation, ie the largest relation such that $P \sim Q$ implies:

1. For all P' such that $P \xrightarrow{\alpha} P'$, there exists Q' such that $Q \xrightarrow{\alpha} Q'$ and $P' \sim Q'$
2. The converse

Looks kinda like an inductive definition, but it's not.

Where's the base case? What's the well-founded order?

In fact, it's a *coinductive* definition!

Coinduction

$$\frac{}{nil \in \mathcal{L}} \qquad \frac{l \in \mathcal{L} \qquad a \in A}{cons(a, l) \in \mathcal{L}}$$

The set *inductively* defined by these rules is the *smallest* set *closed forward* under the rules.

Ie, the set of finite lists over A.

Coinduction

$$\frac{}{nil \in \mathcal{L}} \qquad \frac{l \in \mathcal{L} \qquad a \in A}{cons(a, l) \in \mathcal{L}}$$

The set *coinductively* defined by these rules is the *largest* set *closed backward* under the rules.

Ie, the set of finite *and infinite* lists over A.

Coinduction

$$\frac{}{nil \in \mathcal{L}} \qquad \frac{l \in \mathcal{L} \qquad a \in A}{cons(a, l) \in \mathcal{L}}$$

Let X be the strings over the alphabet $A \cup \{nil, cons, (,), , \}$.

The *rule functional* $F : X \to X$ of the above rules is:

$F(S) = \{nil\} \cup \{cons(a, s) : \ a \in A, s \in S\}$

The least fixed point of F is the set inductively defined by the rules.

The greatest fixed point of F is the set coinductively defined by the rules.

The end