University of
Southampton

COMP2212 Programming Language Concepts

Bisimulation

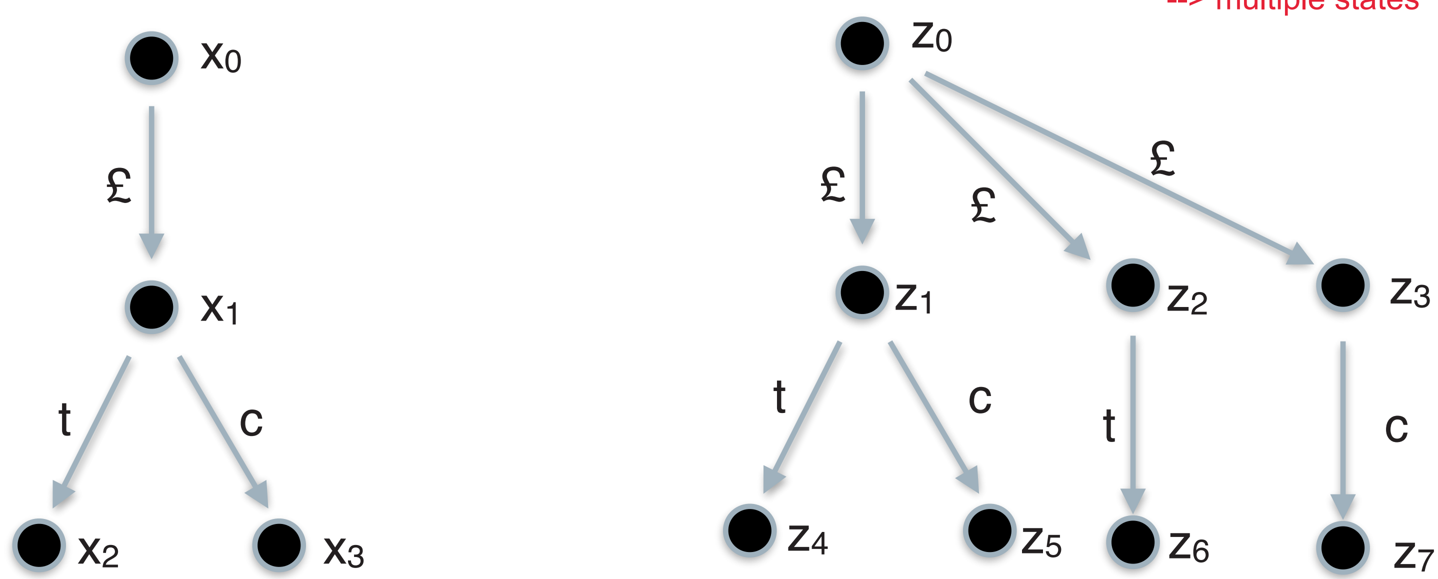
Dr Julian Rathke

University of Southampton

Example - a confused vending machine

- Machine x_0, after receiving a £, will always offer both "tea" and "coffee".
- Machine z_0, after receiving a £, will sometimes (maybe due to a race condition in the implementation) may move to a state in which only one of "tea" or "coffee" is offered.

Z0 can be confused
--> multiple states



Are x_0 and z_0 are simulation equivalent? Should we be able to distinguish them?

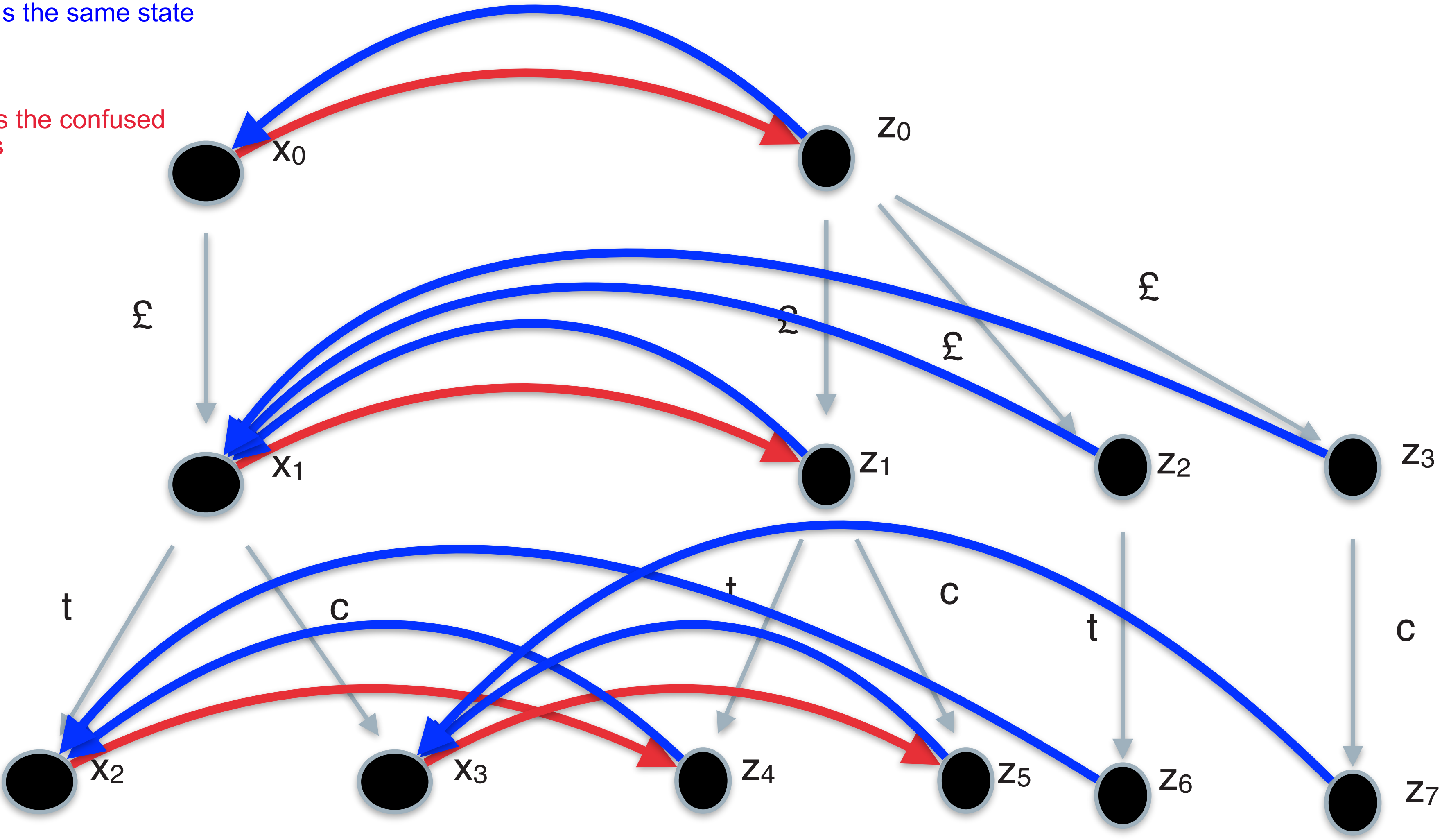
Simulations

- Recall that to know that state y simulates x it suffices to construct a simulation that contains (x,y)
- In the confused coffee machine it is true that both x0 is simulated by z0 **and** z0 is simulated by x0 - so they are **simulation equivalent** states
- This “feels” weird: the crucial insight is that the two simulations relations used to prove equivalence are **not the same relation** on states!
- Let's see the actual simulations in each direction.

Simulation Equivalence Example

Blue is the same state
x to y

Red is the confused
states



The red and blue mappings form different relations.

Bisimulation

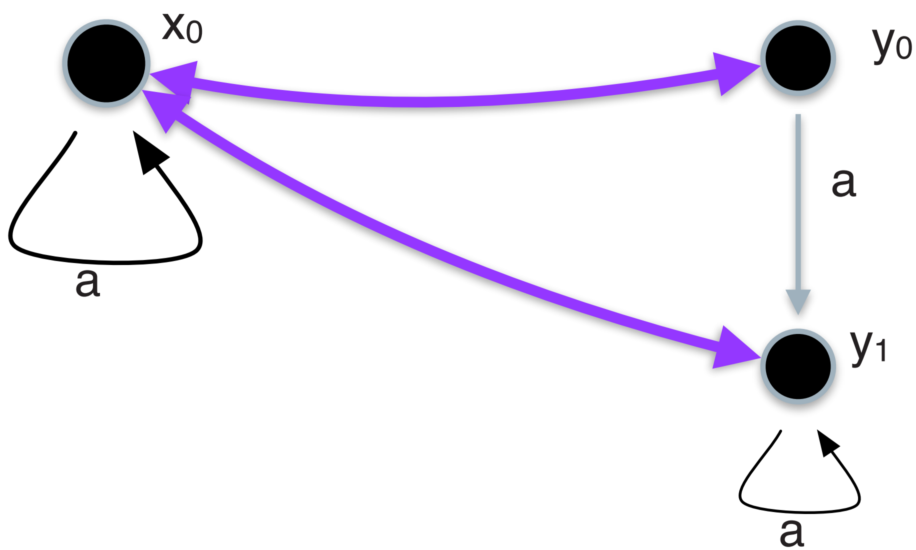
- A bisimulation is a simulation that goes both ways — that is the relation and its reverse are **both** simulations
- Suppose that (X, Σ, L) is a labelled transition system
- A binary **relation** R on states of **L** is called a **bisimulation** on **L** if R satisfies the following conditions:

Whenever x R y and $x \xrightarrow{a} x'$ for some state x' then there exists a state y' such that $y \xrightarrow{a} y'$ and $(x', y') \in R$
Whenever x R y and $y \xrightarrow{a} y'$ for some state y' then there exists a state x' such that $x \xrightarrow{a} x'$ and $(x', y') \in R$

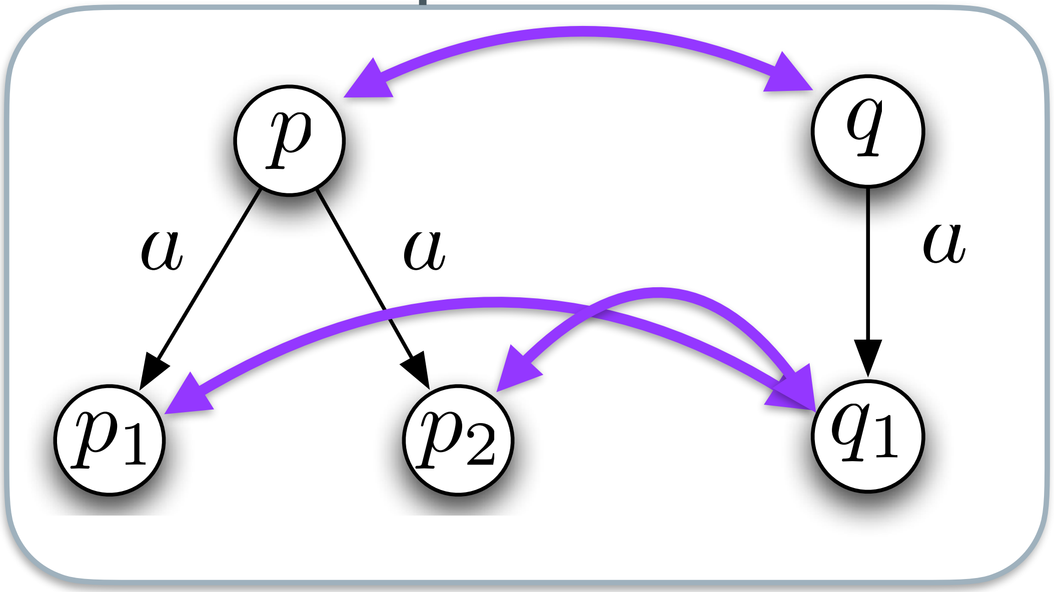
- We write x ~ y if there is a bisimulation that contains (x, y)

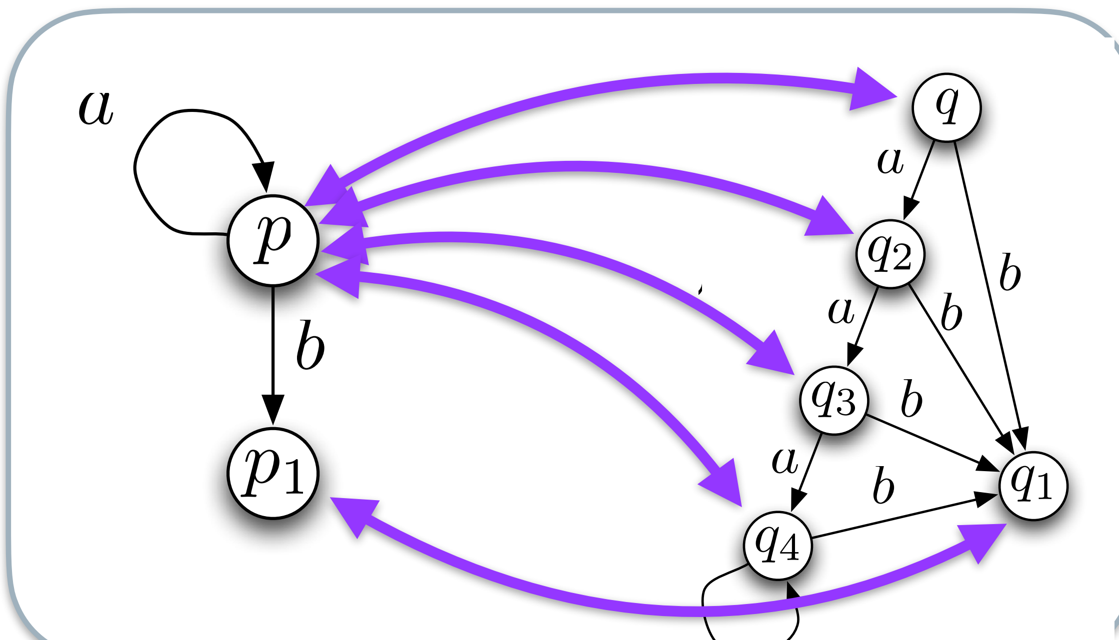
Example

- $\{ (x_0,y_0), (x_0, y_1) \}$ is a bisimulation



More examples





Bisimilarity

University of Southampton

Theorem : the union of two bisimulation relations R1, R2 on an LTS **L** is a bisimulation relation on **L**

Theorem : for any given LTS L there is a largest bisimulation on L

We write ~ to denote the largest bisimulation on an LTS - we call this relation **bisimilarity**

Similar to similarity, bisimilarity is a coinductively defined relation that comes along with a coinductive proof technique.

Coinduction for Bisimilarity

- To show that $x \sim y$, it is enough to construct a bisimulation that contains (x, y)
- If $(x,y) \in R$ and R is a bisimulation, then because $\sim$ is the largest simulation then $R \subseteq \sim$ and hence $x \sim y$ also.

- Again, it is less clear how to show that two states **are not** bisimilar
  - just like for similarity, there is a game we can play!

The bisimulation game

University of Southampton

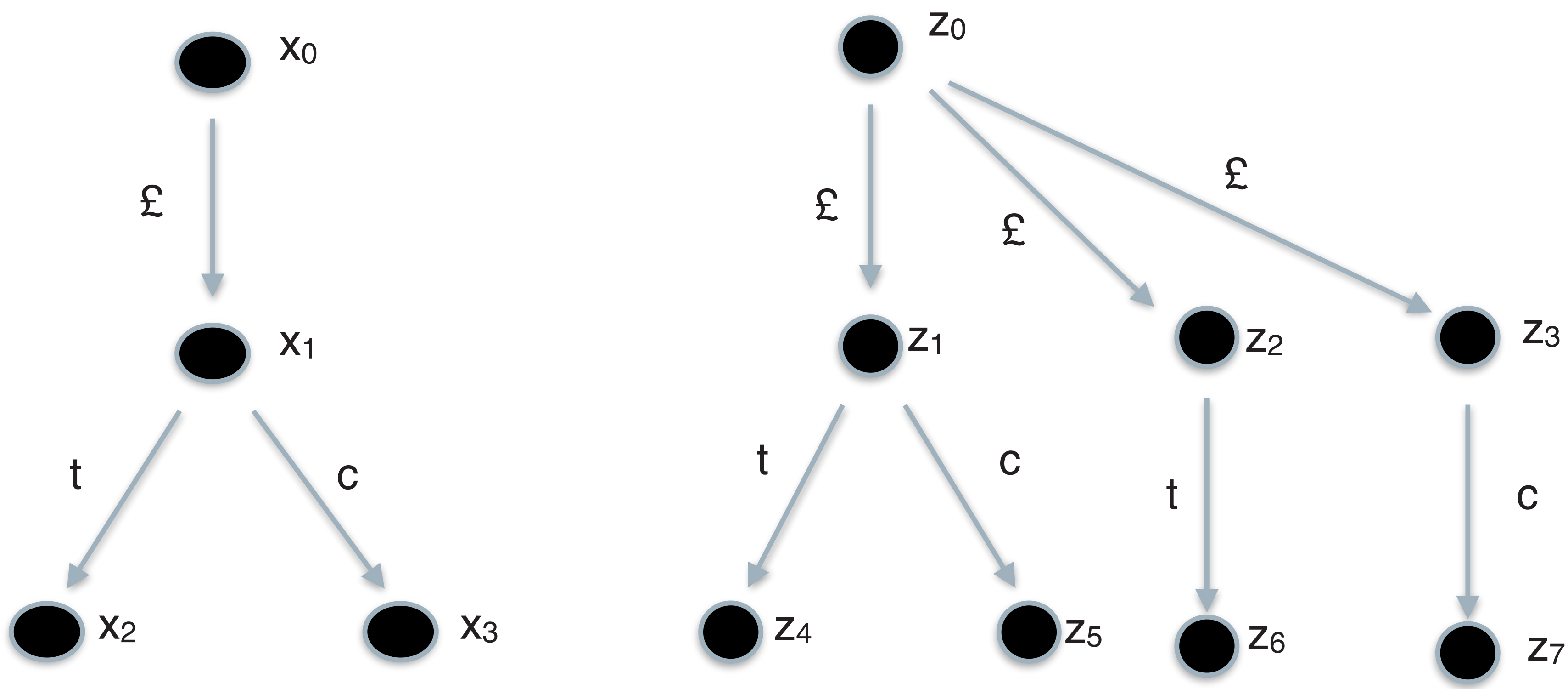
Imagine a game in which two players must pick matching moves in an LTS trying force each other to fail to match the next move. We can use this idea as a proof technique!

Rules of the Game:

- You are playing against a demon 😈 . The game starts at position (x,y).
  1. The demon first picks where to play, either from x or from y
  2. The demon then picks a move from their chosen start state
  3. You must start from the other start state and choose a matching move
  4. The game goes back to **Step 1**, changing the position to (x’,y’) where (x',y') are the states reached by both demon and player.
- If at any point a player cannot make a move, that player loses
- If the game goes on forever, you win.
- nb. The demon always gets to choose whether to play in the "x" states or the "y" states when making its next move.

Let's play!

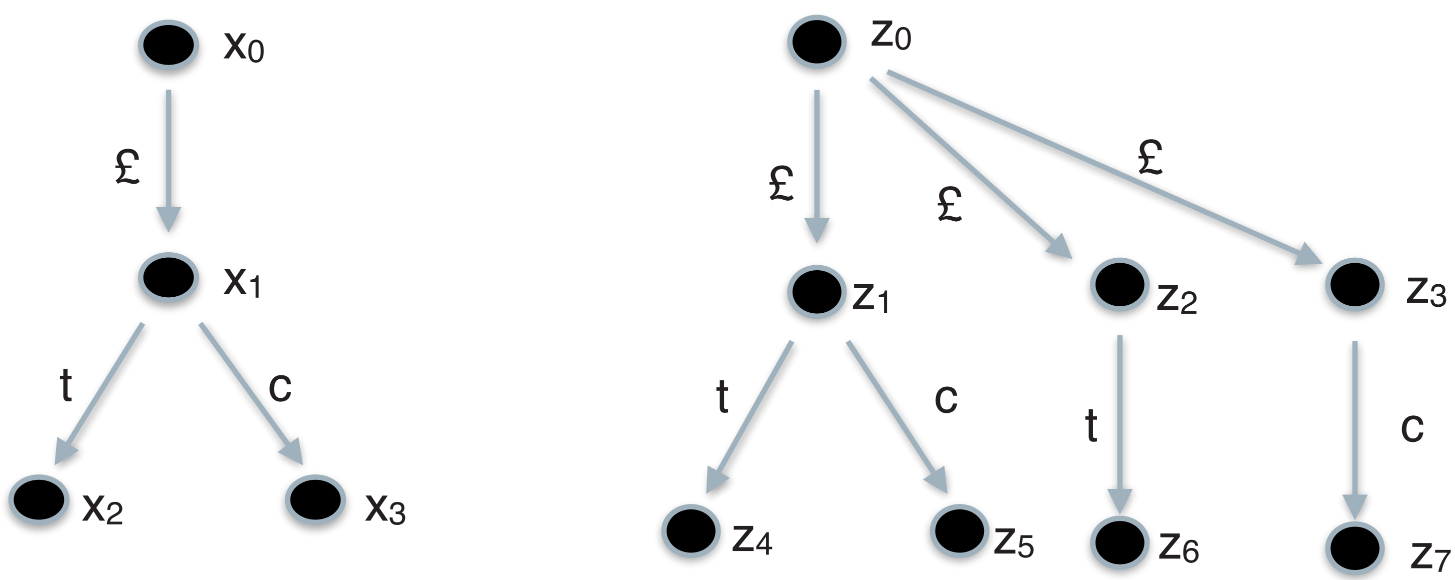
I'll be the demon



Anyone reckon they can beat me this time?

Bisimulation game, example

- Here's a winning strategy for the demon, starting in position (x_0, z_0)
  - The demon picks z_0 to play in and plays the £ move to z_2
  - We have to match with the £ move to x1
  - The game continues from position (x1,z2) but now the demon **switches positions** and plays from x1 - and picks the c move to x3
  - we are stuck, because there is no c move from z2 - so we lose!



Winning strategies

- **Theorem** : for any two states, $x \sim y$ if and only if the non-demon player has a winning strategy in the bisimulation game.
- This gives us a proof technique to show that two states are not in the bisimilarity relation - i.e. demon has a winning strategy from (x,y) means $x \nsim y$
- **Corollary** : bisimilarity implies simulation equivalence
- **Proof** : we prove the contrapositive, i.e. assume that x is not simulation equivalent to y. Then we prove that x is not bisimilar to y. We know, by assumption, that demon has a winning strategy in the simulation game starting from (x,y). This can be turned in to a strategy in the bisimulation game for demon by always choosing the "x" state as its starting state. This strategy is a winning strategy in the bisimulation game also so x is not bisimilar to y.

Anything finer?

- We have a new candidate for a relation, **bisimilarity**, to distinguish processes — but we already had two
  - trace equivalence
  - simulation equivalence
- bisimilarity implies simulation equivalence implies trace equivalence
  - but the implications do not go the other way
- **Two Questions:**
- Are there other relations that might be used?
- Can we cook up an even more confused coffee machine example to cast doubts on bisimilarity?

Two Answers

- Yes!
  - There are loads of variations of bisimilarity and trace equivalence that have been researched, each with subtly different observational properties.
  - There is no single "correct" notion of equivalence ... but
- No!
  - An observer cannot tell the difference between any two bisimilar states if all they can see are the capability of performing actions
  - Bisimilarity is the **finest** “reasonable” equivalence
    - “reasonable” here means roughly that we can only observe behaviour and not look directly at the statespace

Next Lecture

Shared Variable Concurrency