COMP2212 Programming Language Concepts
Example - an evil coffee machine

- $z_0$ can act like $x_0$, by going to $z_1$ but sometimes (maybe a race condition in the implementation?) it may go to $z_2$ or $z_3$

- $x_0$ and $z_0$ are simulation equivalent (why?). 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 evil coffee machine it is true that both $x_0$ is simulated by $z_0$ and $z_0$ is simulated by $x_0$ - so they are simulation equivalent states.
- This “feels” weird: the crucial insight is that the two simulations are not the same relation on states!
A bisimulation is a simulation that goes both ways — that is the relation and its reverse are both simulations.

So, intuitively if x and y are bisimilar then both x can respond to the moves of, and y can respond to the moves of x — and stay in the same relation.
Suppose that \((X, \Sigma, L)\) is a labelled transition system. A bisimulation is a relation on states that satisfies the following conditions, whenever \((x, y) \in R\):

- if \(x \xrightarrow{a} x'\) then there exists \(y'\) such that \(y \xrightarrow{a} y'\) and \((x', y') \in R\)
- if \(y \xrightarrow{a} y'\) then there exists \(x'\) such that \(x \xrightarrow{a} x'\) and \((x', y') \in R\)

We sometimes write \(x \sim y\) if there is a bisimulation that contains \((x, y)\).
Example

\[ \{ (x_0,y_0), (x_0, y_1) \} \text{ is a bisimulation} \]
More examples
Properties of bisimulation

- **Theorem** Unions of bisimulations are bisimulations
- There exists the largest bisimulation, it is the union of all bisimulations
  - The largest bisimulation, written ~, is called **bisimilarity**
- Just as simulation, bisimulation is a coinductive concept — more on this next week
Bisimilarity

- To show that $x \sim y$, it is enough to construct a bisimulation that contains $(x, y)$ (why?)
- 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

- We play against a demon. The game starts at position \((x,y)\)

1. The demon picks where to play, either at \(x\) or at \(y\)
2. If demon chose \(x\), he picks a move \(x \xrightarrow{\alpha} x'\)
   - in this case we must respond with \(y \xrightarrow{\alpha} y'\)
3. If demon chose \(y\), he picks a move \(y \xrightarrow{\alpha} y'\)
   - in this case we must respond with \(x \xrightarrow{\alpha} x'\)
4. The game goes back to step 1, changing position to \((x', y')\)
Bisimulation game, continued

- If at any point a player cannot make a move, that player loses.
- If the game goes on forever, we win.
- If the Demon has a winning strategy, then \( x \) and \( y \) are not bisimilar.
- If we have a winning strategy then \( x \sim y \).
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 \(x_1\)
- The game continues from position \((x_1, z_2)\) but now the demon switches positions and plays from \(x_1\) - and picks the \(c\) move to \(x_3\)
- we are stuck, because there is no \(c\) move from \(z_2\) - so we lose!
Anything finer?

- We have a new candidate for a relation, **bisimilarity**, to distinguish processes — but we already had two
  - trace equivalence
  - simulation equivalence
- bisimilar implies simulation equivalent implies trace equivalent
  - but the implications do not go the other way
- Can we cook up an even more evil coffee machine example to cast doubts on bisimilarity?
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 state space.