COMP2212 Programming Language Concepts

Labelled transition systems
Reasoning about concurrent programs

- There are some essential aspects of concurrency that make it different from sequential computation, e.g.:
  - **communication** - processes communicate with other processes either through shared memory or with message passing
  - **synchronisation** - processes must sometimes synchronise their actions to ensure atomicity
  - **nondeterminism** - what can be observed about a program changes from one run to the next
- Some things that we take for granted need to be rethought: e.g. how to test concurrent code?
## Nondeterminism example

Suppose we run the above three pieces of code concurrently. What will be printed?

<table>
<thead>
<tr>
<th>Thread 1</th>
<th>Thread 2</th>
<th>Thread 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = 0;</code></td>
<td><code>x = 1;</code></td>
<td><code>x = 2;</code></td>
</tr>
<tr>
<td><code>print &quot;x is &quot;.i_to_s(x);</code></td>
<td><code>print &quot;x is &quot;.i_to_s(x);</code></td>
<td><code>print &quot;x is &quot;.i_to_s(x);</code></td>
</tr>
</tbody>
</table>
Possibilities

- We can use a mathematical structure called a Labelled Transition System (LTS) in order to capture what can be observed about programs.
  - LTS are a mathematical structure for reasoning about nondeterminism.
  - The labels of transitions say “what can be observed”.
  - Eg.

![Diagram showing LTS transitions and states with labels x is 0, x is 1, and x is 2.]

States

Transitions
Labelled transition systems

- We have already seen two examples of labelled transition systems.
- A labelled transition system is a mathematical structure $(X, \Sigma, L)$ where:
  - $X$ is a set of states.
  - $\Sigma$ is an alphabet of actions.
  - $L \subseteq X \times \Sigma \times X$. 


Labelled transition systems are similar to finite state automata, which also have states and transitions, but there are important differences.

- The set of states in an LTS can be infinite: we cannot assume that our systems have only a finite number of possible states!
- LTSs typically do not have initial and final states.
Example, formally

\[ X = \{ s_0, s_1, s_2, s_3 \} \]

\[ \Sigma = \{ x \text{ is } 0, x \text{ is } 1, x \text{ is } 2 \} \]

\[ L = \{ (s_0, x \text{ is } 0, s_1), (s_0, x \text{ is } 1, s_2), (s_0, x \text{ is } 2, s_3) \} \]
Kinds of nondeterminism

- Internal - “the machine chooses”
  - e.g. the simple code example we have examined
  - the nondeterminism is resolved by the scheduler
- External - “the environment chooses”
  - e.g. interactive systems such as vending machines
  - the combination of a vending machine and user can be thought of as a concurrent system
External nondeterminism - Coffee machine

- The user puts in money - this is the £ action
- The machine now offers a choice between tea (t) and coffee (c)
Process equivalence

- Non determinism is inherent in concurrent and interactive systems

- What does it mean that a system is correct?
  - one answer: it should behave like (be equivalent to) some specification
  - but what should equivalent mean?
  - this is a surprisingly subtle question that has resulted in a lot of research over the last 40 years
First try

- Give the specification as a set of **traces**
  - a **trace** is a sequence of observations from some state
  - example:

    ![Diagram](image.png)

    Traces from $x_0$: $\varepsilon$, £, £t, £c

- Say that two states are **trace equivalent** when they have the same traces
Some systems have an infinite set of traces

Traces from $x_0$: $b$, $ab$, $aab$, $aaab$, $aaaaab$, ....

Indeed, all the words matched by the regular expression $a^*b$
Example: coffee machines

- Should we consider these two coffee machines as equivalent?

- Note that $x_0$ and $y_0$ are trace equivalent, the difference is when the nondeterministic choice happens.
In some cases, trace equivalence is too coarse: it equates too much.

- we want to distinguish the two coffee machine examples

- In the next few lectures we will discuss finer ways of distinguishing between labelled transitions systems: simulation and bisimulation.