PLPDI: Verification Revision Notes

Sam Barrett

January 10, 2021

Formal Verification

This is the application of rigorous, mathematical models to establish the **correctness** of computerised systems.

Computer-aided Verification

Computer-aided verification is simply automated formal verification methods via tools and algorithms etc.

1 Labelled Transition Systems

Labelled transition systems are comprised of two parts:

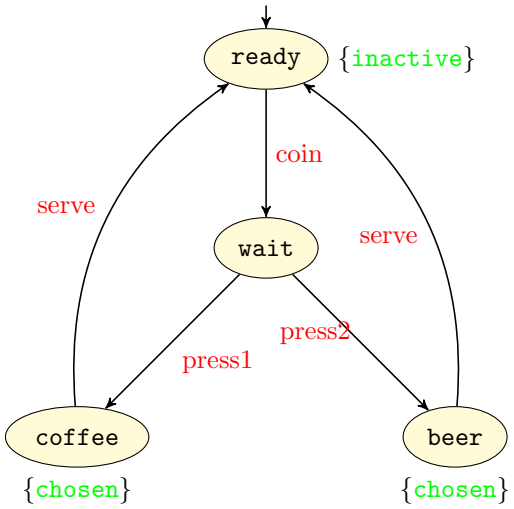
- States, representing possible configurations of a systems.
  - Values of program variables
  - Values of registers in a hardware circuit
- Transitions, possible ways/routes a system can *evolve*
  - Execution of a program statement
  - Sequential circuit update

Formally, they can be defined as a tuple $\texttt{S,Act},\rightarrow,\texttt{I,AP,L}$. Where:

- $\texttt{S}$ is a set of **states**
- $\texttt{Act}$ is a set of **actions**
- $\rightarrow \subseteq \texttt{S} \times \texttt{Act} \times \texttt{S}$ is a **transition system**
- $\texttt{I} \subseteq S$ is a set of **initial states**
- $\texttt{AP}$ is a set of **atomic propositions**
- $\texttt{L:S} \rightarrow 2^{\texttt{AP}}$ is a **labelling function**

1.1 Example LTS

- $\texttt{S} = \{\texttt{ready, wait, coffee, beer}\}$
- $\texttt{Act} = \{\texttt{coin, press1, press2, serve}\}$
- $\rightarrow = \{(\texttt{ready, coin, wait}),$
  $(\texttt{wait, press1, coffee}),$
  $(\texttt{wait, press1, beer}),$
  $(\texttt{coffee, serve, ready}),$
  $(\texttt{beer, serve, ready})\}$
- $\texttt{I} = \{\texttt{ready}\}$
- $\texttt{AP} = \{\texttt{inactive, chosen}\}$
- $\texttt{L(ready)} = \{\texttt{inactive}\},$
  $\texttt{L(wait)} = \emptyset,$
  $\texttt{L(coffee) = L(beer)} = \{\texttt{chosen}\}$



1.2 Labelling & Finiteness

LTS states are labelled with **atomic propositions** $a, b, \ldots \in \texttt{AP}$. They represent facts or observations of the system.

LTS transitions are labelled with actions $\alpha, \beta, \ldots \in \texttt{Act}$ and are used to communicate between *components*

An LTS is said to be *finite* if `S,Act` and `AP` are finite. We tend to assume our LTSs are finite.

1.3 Transitions

We denote transitions as $s - \alpha \to s\prime$ if $(s, \alpha, s\prime) \in \to$

We define the direct successors and predecessors as:

- $\texttt{Post}(s, \alpha) = \{s\prime \in S | s - \alpha \to s\prime\}$ and $\texttt{Post}(s) = \cup_{\alpha \in \texttt{Act}} \texttt{Post}(s, \alpha)$
- $\texttt{Pre}(s, \alpha) = \{s\prime \in S | s\prime - \alpha \to s\}$ and $\texttt{Pre}(s) = \cup_{\alpha \in \texttt{Act}} \texttt{Pre}(s, \alpha)$

We say that a state, s is **terminal** if $\texttt{Post}(\text{s}) = \emptyset$, i.e. it has no outgoing transitions. Such states can be used to represent the termination of a program or erroneous or undesired behaviour.