Week 2/3
1. System interconnections (block diagrams)
2. Finite State Machines (Chapter 3)

A system $S$ is a function: $\text{InputSignals} \rightarrow \text{OutputSignals}$ with

- $\text{InputSignals} = [D \rightarrow R]$,
- $\text{OutputSignals} = [D' \rightarrow R']$

$$x \in \text{InputSignals} \quad y = S(x) \in \text{OutputSignals}$$

Recall

$10\text{Evans}: \text{[Ints, } \rightarrow \text{[enter, leave]} \text{]} \rightarrow \text{[Ints, } \rightarrow \text{Integers, ]}$

$$u = (e, l, i, e, e, \ldots) \quad \text{10Evans} \quad y = (6, 5, 4, 5, 6, \ldots)$$

initial number in room
The system $10\text{Evans}$ can be specified as follows:

$$y = 10\text{Evans}(u), \text{ and for all } n \in \text{Ints}. , y(n) \text{ is given by}$$

$$y(0) = 5 + 1(u(0) = c) - 1(u(0) = l) = 6$$

$$\cdots \cdots$$

$$y(n) = 5 + \sum_{0 \leq k \leq n} [1(u(k) = c) - 1(u(k) = l)]$$

where

$$1 : \{\text{true, false}\} \rightarrow \{1, 0\} \text{ is the function}$$

$$1(\text{true}) = 1, \ 1(\text{false}) = 0.$$
**Finite State Machines**

\[ v_1 = \text{Microphone}(u) \cap \text{AGC}(v_1, v_2) \]

4 equations

\[ v_2 = \text{VolumeSensor}(s) \]

4 unknowns

**Finite State Machines**

\[ x = (x(0), x(1), x(2), \ldots) \]

\[ y = (y(0), y(1), y(2), \ldots) \]

\[ x \in \text{InputSignals} \]

\[ y \in \text{OutputSignals} \]

\[ x = (0, 1, 0, 0, 0, 1, 1, \ldots) \]

\[ y = (f, f, f, t, t, f, f, \ldots) \]

\[ s(0) = \text{init} \]

\[ (s(n+1), y(n)) = \text{update}(s(n), x(n)), n = 0, 1, \ldots \]
What does this machine with 4 states do?

It also implements Recognizer

Can a machine with 2 states implement Recognizer?
For Recognizer

States = \{init, a, b\}
Inputs = \{0,1\}
Outputs = \{t, f\}
initialState = init

update: States \times Inputs \rightarrow States \times Outputs
is given by:

\[
\begin{align*}
\text{update}(\text{init}, 0) &= (0, f) \\
\text{update}(\text{init}, 1) &= (\text{init}, f) \\
\end{align*}
\]

It is useful sometimes to express the update function as two functions:

update = (nextState, output),
nextState: States \times Inputs \rightarrow States
output: States \times Inputs \rightarrow Outputs

so that

\[
\begin{align*}
\text{update}(s(n), x(n)) &= (s(n+1), y(n)) \\
\text{nextState}(s(n), x(n)) &= s(n+1) \\
\text{output}(s(n), x(n)) &= y(n)
\end{align*}
\]

Operation of state machine

Given S = (States, Inputs, Outputs, update, initialState)

InputSignals = [Nats0 \rightarrow Inputs]
OutputSignals = [Nats0 \rightarrow Outputs]

An input signal \( x = (x(0), x(1), \ldots, x(n), \ldots) \) determines the state response \( s: \text{Nats}_0 \rightarrow \text{States} \) and the output signal \( y: \text{Nats}_0 \rightarrow \text{Outputs} \)

by the recursion, \( s(0) = \text{initState} \), and \( n \geq 0 \),
\[
(s(n+1), y(n)) = \text{update}(s(n), x(n))
\]

Thus \( S \) determines an input-output function (system) \( F: \text{InputSignals} \rightarrow \text{OutputSignals} \)

For Recognizer

InputSignals = [Nats0 \rightarrow \{0,1\}]
OutputSignals = [Nats0 \rightarrow \{t,f\}]

The input-output function (system) is \( F: \)

\[
\forall x \in \text{InputSignals}, \forall n \in \text{Nats}_0
\]

\[
y(n) = F(x)(n) = \begin{cases} 
  t, & \text{if } (x(n-2), x(n-1), x(0)) = (0,0,0) \\
  f, & \text{else}
\end{cases}
\]
Stuttering

It is useful to introduce a stuttering symbol \texttt{absent}

We insist that \texttt{absent} ∈ Inputs and \texttt{absent} ∈ Outputs

We require that for all states \( s \) ∈ States
\[ \text{update}(s, \texttt{absent}) = (s, \texttt{absent}) \]
The guards are subsets of Inputs

- guards on different arcs from same state are disjoint (determinism)
- union of all guards (from same state) = Inputs; (reactive system)
- guard on 'else' is complement of remaining guards
- guards may be defined using convenient notation, eg. if Inputs = \{a, b, c\}, \neg a = \{b, c\}
All information about a state machine is given its update function. If States and Inputs are finite, update can be given as a table.

<table>
<thead>
<tr>
<th>current state</th>
<th>(next state, output) if input is</th>
</tr>
</thead>
<tbody>
<tr>
<td>init</td>
<td>(a, f)</td>
</tr>
<tr>
<td>a</td>
<td>(b, f)</td>
</tr>
<tr>
<td>b</td>
<td>(b, t)</td>
</tr>
</tbody>
</table>

System requirements in English

System specification as

\( F: \text{InputSignals} \to \text{OutputSignals} \)

System implementation as state machine described by set-and-functions 5-tuple, transition diagram, or table.

Parking meter

\{\text{tick, coin5, coin25, absent}\} \to \{\text{safe, expired, absent}\}

topics/state/parking meter
Answering machine

topics/state/answering machine

table