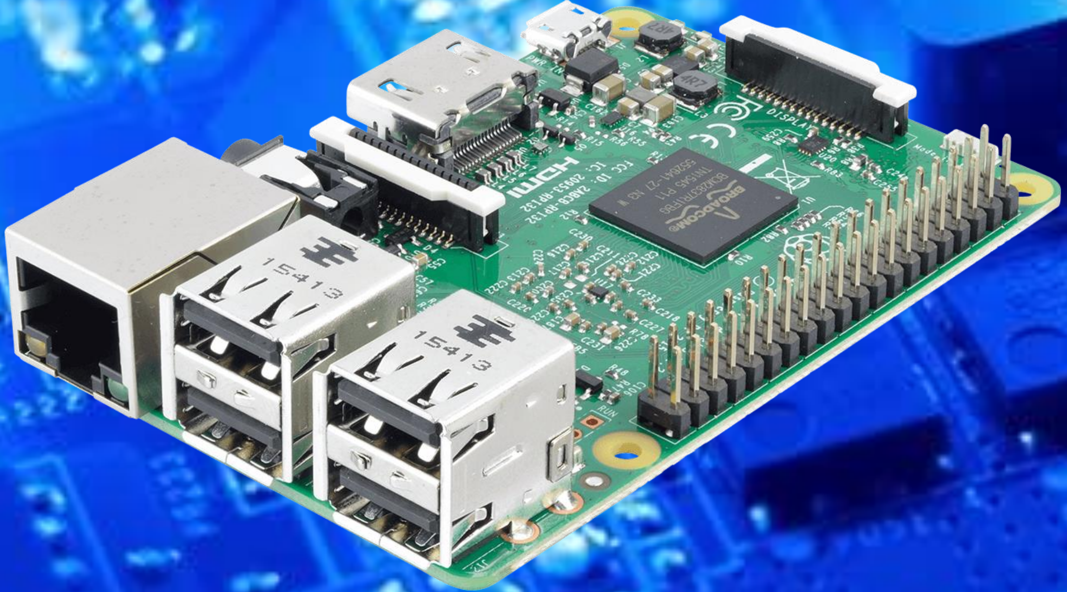
EEE3096S



Specifications and Modeling (3)

Section 2.4 of textbook relates

L7

Embedded Systems II

Dr Simon Winberg

Electrical Engineering
University of Cape Town

Outline of Lecture

- Requirements Considerations
  - The Customer Tour
- Modelling
  - Mealy and Moore FSM
  - FSM exercise
  - Communicating finite state machines

We starting with a visit to requirements and then moving out of specifications into design…

And here on your left you can see Cindy, our CEO's ex-wife shopping at Harrold's. Let's see what the target market likes!
FOR YOUR OWN SAFETY PLEASE REMAIN SEATED
PLEASE LEAVE THIS SEAT FREE FOR YOUR GUIDE

EEE3096S

THE CUSTOMER RESEARCH TOUR

Embedded Systems II

The Customer Research Tour:

Laying foundations for Requirements & Specifications

- Four People Recommended
  - Including Engineers & Marketing/Sales people
- Prepare for the visit – plan interview questions
- Each member has a specific role
  - 1 Leader
  - 1 Technical Writer / Note Taker
  - Rest of team offer technical input or they just observe
- Followed by a debriefing

Reflections on a past experience

CMAASM : Continuous Miner Automation-Assist and Safety Monitor

REQUIREMENTS PHASE

- Started with intense literature review
- Business meeting with clients
- Few back and forth
- Visit to mines
- Meeting miners
- Viewing operation, miners do demo
- Writing up report

Requirements & Specifications: After the Tour

- Translate notes into a set of formal product requirements (see later slide)
- Requirements intended to guide the team in the specifications phase
- Further tours might be useful (or required) to fine-tune requirements
- Incorrect requirements can be extremely costly

Requirements documents

- Typically written up as a lengthy document
- Use cases and UML use cases, scenarios / sequence diagrams, etc. are often used as a means to work out the requirements & specifications; the requirements document itself often doesn't include UML (besides perhaps use cases) as it needs to be a document that the client can read and understand.
- The requirements document often becomes, or is translated into, a contract and acceptance test checklist.
- The doc itself if mostly text with figures to aid understanding

(the less patient the client the more figures 😉)

Requirements documents

Sample table of contents

Product Requirements Document

Requirements documents Examples

- Optional readings:
  - http://www.embedded-systems-portal.com/CTB/Overview,10084.html

Models of computation in this course

Communication/ local computations	**Shared memory**	**Message passing** **Synchronous**	**Asynchronous**
Undefined components	Plain text ✓, use cases ✓	| Sequence Charts ✓, ICD	
Communicating finite state machines	StateCharts		SDL
Data flow	Scoreboarding + Tomasulo Algorithm (☞ Comp.Archict.)		Kahn networks, SDF
Petri nets		C/E nets, P/T nets, …	
Discrete event (DE) model	VHDL*, Verilog*, SystemC*, …	Only experimental systems, e.g. distributed DE in Ptolemy (Ptolemy only discussed briefly)	
Von Neumann model	C, C++, Java	C, C++, Java with libraries CSP, ADA |	

* Classification based on implementation with centralized data structures
SystemC will not be delved into detail. Only brief flavour of VHDL and Verilog given

Moore & Mealy Machines

What is the difference between Moore and Mealy Machines?

Are these are state machines in which …

~~Moore = Plentiful state machine, with many states ?~~

~~Mealy = Measly state machine, with few states ?~~

NOT SO

Moore & Mealy Machines

What is the difference between Moore and Mealy Machines?

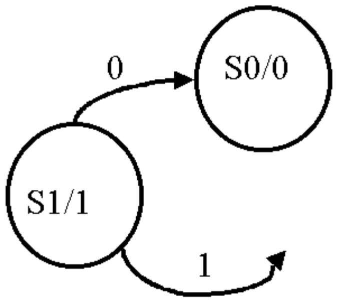
These are state machines in which…

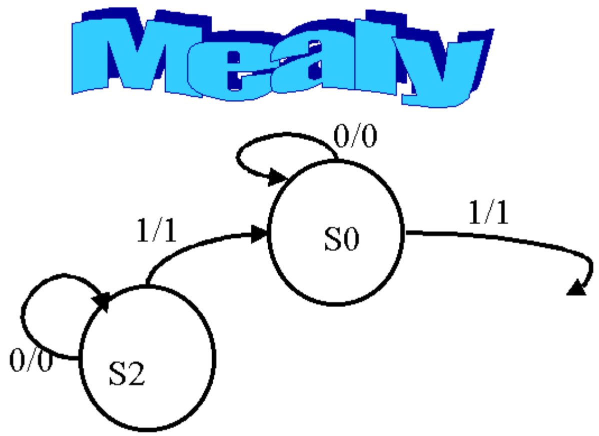
Moore Machine

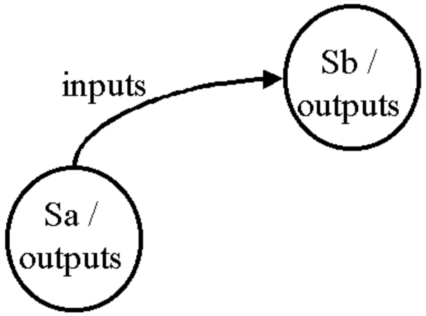
= finite-state machine whose output values are determined only by its current state

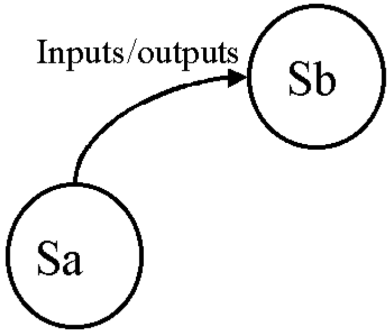
Mealy Machine

= finite-state machine whose outputs values are determined both by its current state and the current inputs.









Note in diagrams above you can assume there is only one input in the system so the transitions only indicate the value of the input, opposed to indicating input=0

Short Exercise

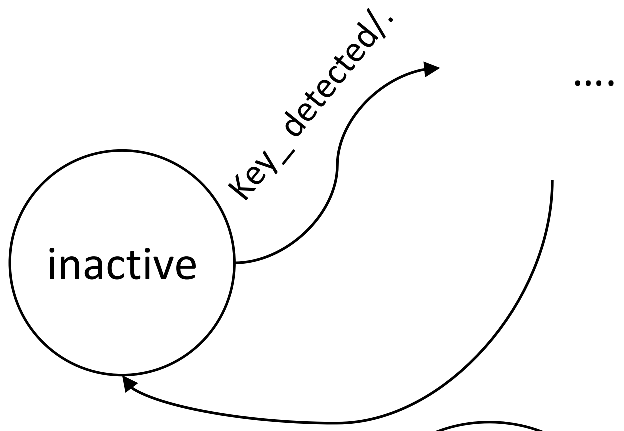
Draw a **Mealy machine** to illustrate the operation of the motorcar ignition system described below.

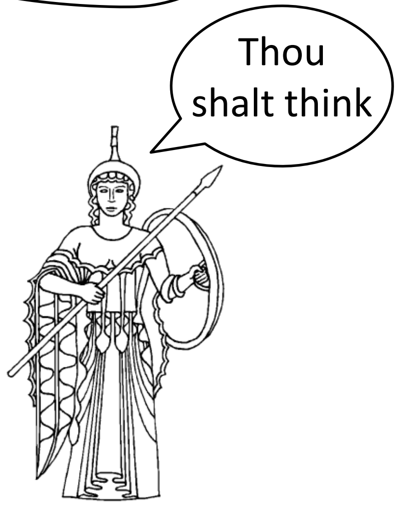
<u>Motorcar Ignition Systems</u>

This system has four parts. The key detector, the ignition button, ignition system, the engine. The system is in **inactive** mode until the system detects the key. Once the key is detected the system is in **ready** mode. If the key is no longer sensed the system reverts to **inactive** mode. If the user presses the ignition button then the ignition sequence is started. The ignition sequence starts with a **gear check**, to see if the car is in neutral, if it isn't the system does a beep and reverts to **user fault** mode. The system remains in user fault mode until the user has released the ignition button and after 2 seconds have elapsed from being in that mode, and returns to **ready** mode once this is the case. If in **gear check** the gear was in neutral then an **engine start** message is sent to the engine. The system then goes into **driving** mode. If the user presses the ignition button for more than 300ms in drive mode then a **shutdown** message is sent to the engine and control returns to **ready mode**.

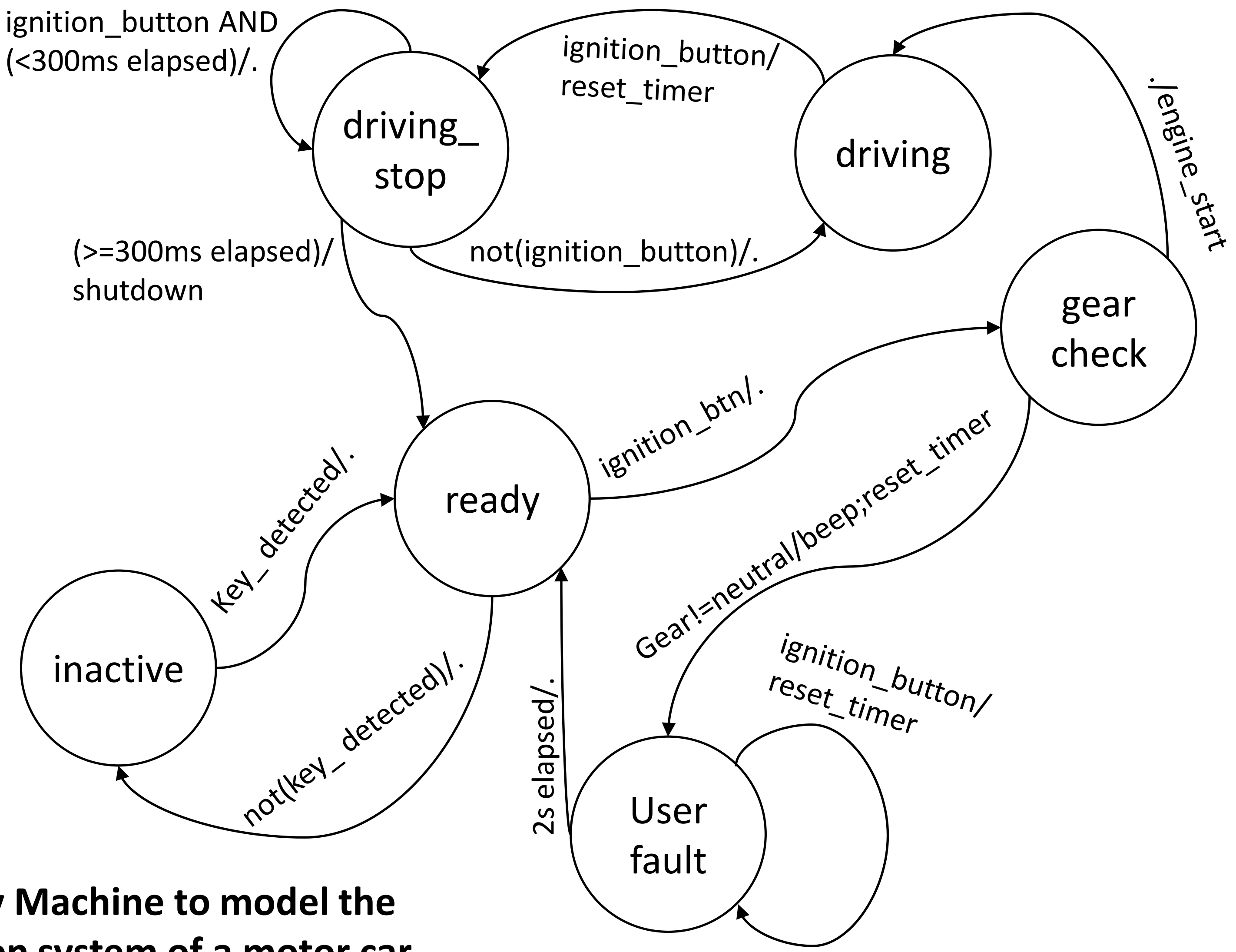
Obviously there are some safety conditions that are being ignored to keep this simple. E.g. if the driver presses the ignition button while in gear and driving should the car shut down?

Short Exercise Solution





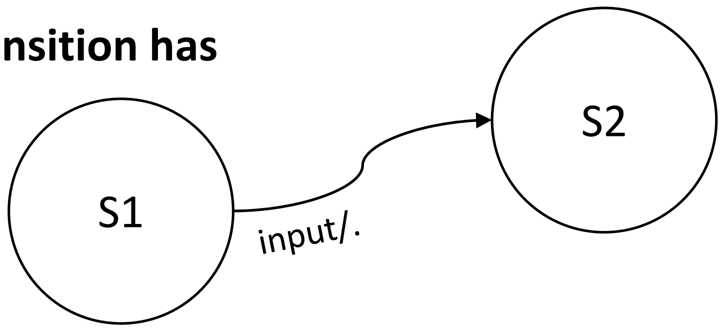
Short Exercise Solution


ignition_button AND (<300ms elapsed)/.
driving_ stop
ignition_button/ reset_timer
driving
./engine_start
(>=300ms elapsed)/ shutdown
not(ignition_button)/.
gear check
ignition_btn/.
Key_ detected/.
ready
Gear!=neutral/beep;reset_timer
inactive
not(key_ detected)/.
2s elapsed/.
ignition_button/ reset_timer
User fault

Mealy Machine to model the ignition system of a motor car

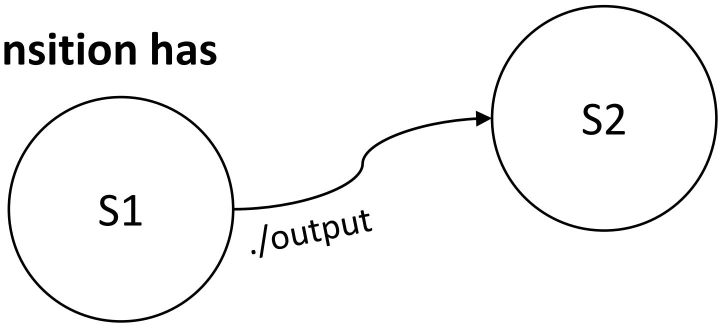
Mealy Connections

Situation where a transition has no output



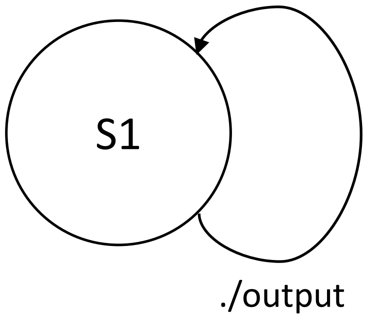
You can just put a dot or blank after slash, e.g.
in/. in/
means the same thing

Situation where a transition has no input



You can just put a dot or blank before slash, e.g.
./out /out
means the same thing

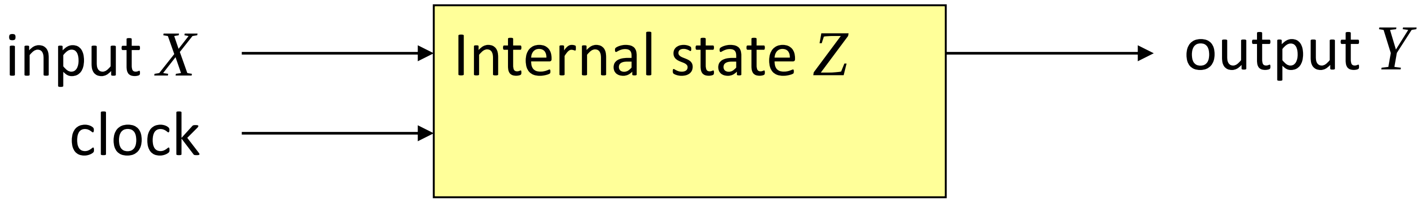
Busy output …



This can be used to emphasise by default the output is continuously asserted (but this is something one may include in comments for the diagram to keep diagrams neater)

StateCharts: recap of classical automata

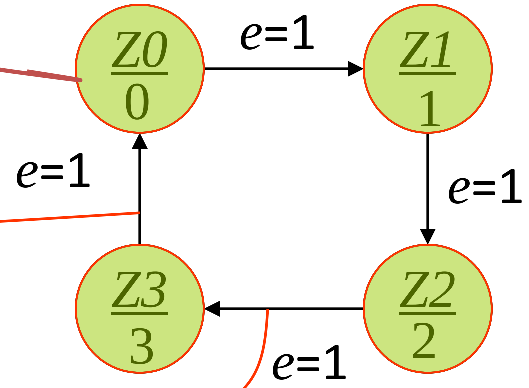
Classical automata:



Next state Z^+ computed by function δ
Output computed by function λ

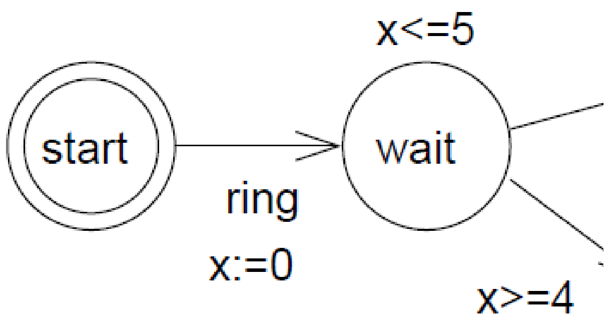
Moore / Mealy automata = finite state machines (FSMs)

- Moore-automata:
  $Y = \lambda(Z); \quad Z^+ = \delta(X, Z)$
- Mealy-automata
  $Y = \lambda(X, Z); \quad Z^+ = \delta(X, Z)$



Timed automata

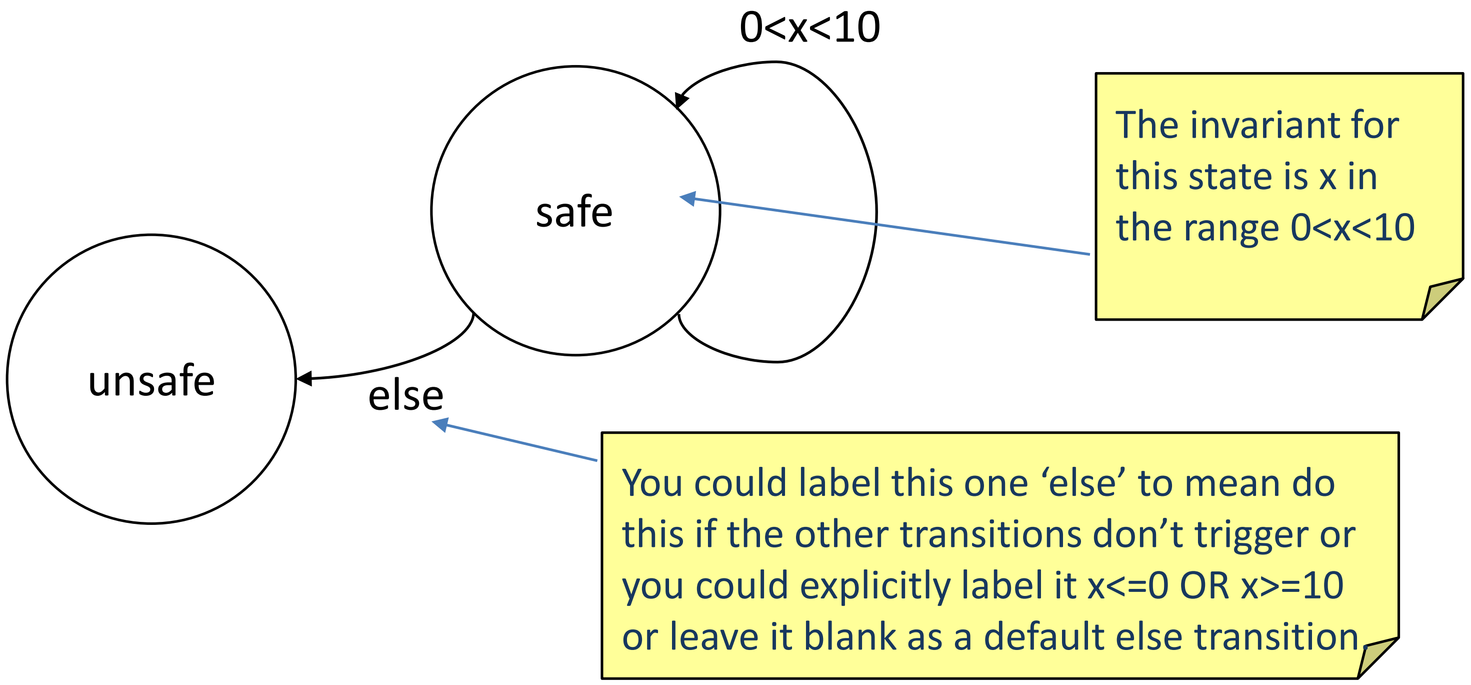
- Timed automata = automata + models of time
- *Variables that model logical clocks in system, that are initialized with zero when the system is started, and then increase synchronously with the same rate.*
- *Clock constraints i.e. guards on edges are used to restrict the behavior of the automaton.*
- *A transition represented by an edge can be taken when the clocks values satisfy the guard labeled on the edge.*
- Additional invariants* make sure the transition is taken.
- *Clocks may be reset to zero when a transition is taken* [Bengtsson and Yi, 2004].



* explained in a moment

Terminology Aside: Invariant

- Invariant:
  - A quantity (e.g. variable range limits) that remains the same in a state (or process) or in applying a transition (or function)



The invariant for this state is x in the range 0<x<10

You could label this one ‘else’ to mean do this if the other transitions don’t trigger or you could explicitly label it x<=0 OR x>=10 or leave it blank as a default else transition

Terminology Aside: Pre- and Post-condition

- The book also makes mention of pre-conditions and post-conditions
- These relate to methods of formal verification or correctness proofs
  - you don't need to worry about these issues for now, but it is nevertheless useful to know what these terms
- Pre-condition: an assertion that is true before a function is applied
- Post-condition: an assertion that is true after the function is applied

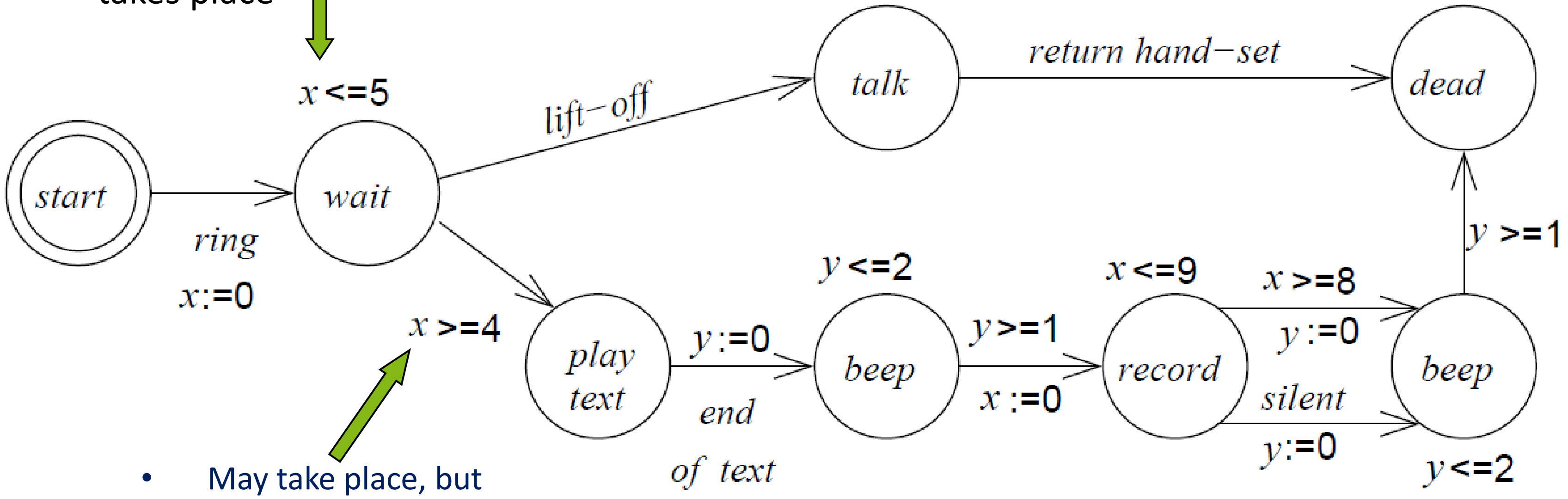
e.g.

```
int sqrt ( int x ) {  // not this is for integers
  PRECOND(x>=0)
   … do square root operation …
  POSTCOND(result<=x)
}
```

assertion checked

FSM Example: Answering machine

Ensures that transition takes place



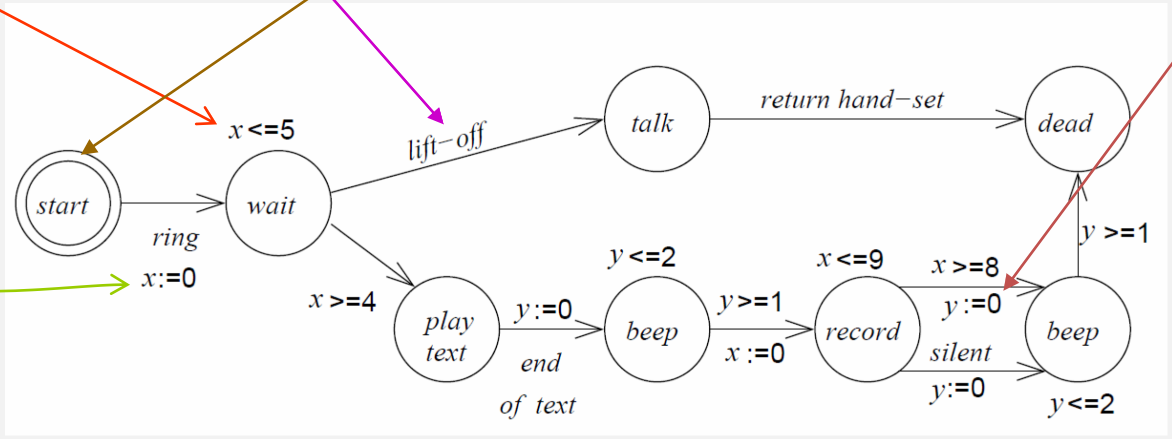
- May take place, but does not have to

Definitions – timed automaton

- Let C: real-valued variables C representing clocks.
- Let Σ: finite alphabet of possible inputs.
- **Definition**: A **clock constraint** is a conjunctive formula of atomic constraints of the form
  $x \circ n$ or $x - y \circ n$ for $x, y \in C$, $\circ \in \{\leq,<,=,>,\geq\}$ and $n \in N$
- Let $B(C)$ be the set of clock constraints.
- **Definition**: A **timed automaton** A is a tuple (S, s_0, E, I) where
- S is a finite set of states, s_0 is the initial state,
- $E \subseteq S \times B(C) \times \Sigma \times 2^C \times S$ is the set of edges,
  $B(C)$: conjunctive condition, 2^C: variables to be reset
- $I : S \rightarrow B(C)$ is the set of invariants for each of the states
  $B(C)$: invariant that must hold for state S

Definitions – math. rep. of timed automaton

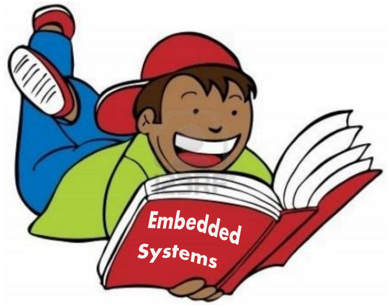
- Let C: real-valued variables C representing clocks.
- Let Σ: finite alphabet of possible inputs.
- **Definition**: A **clock constraint** is a conjunctive formula of atomic constraints of the form $x \circ n$ or $x - y \circ n$ for $x, y \in C$, $\circ \in \{\leq,<,=,>,\geq\}$ and $n \in N$
- Let $B(C)$ be the set of clock constraints.
- **Definition**: A **timed automaton** A is a tuple (S, s_0, E, I) where S is a finite set of states, s_0 is the initial state,
- $E \subseteq S \times B(C) \times \Sigma \times 2^C \times S$ is the set of edges, $B(C)$: conjunctive condition, 2^C: variables to be reset
- $I : S \to B(C)$ is the set of invariants for each of the states, $B(C)$: invariant that must hold for state S



The Next Episode…

Lecture L08

L08: State Charts, ICD



Reminder: Read section 2.4