

**SE 2AA4, CS 2ME3 (Introduction to Software  
Development)**

**Winter 2017**

## **23 Finite State Machines (Ch. 5)**

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 3, 2017



# Finite State Machines

- Administrative details
- Classification of specification styles
- Continuation on specification qualities
- Homework exercise
- How to verify a specification
- Finite state machines

# Administrative Details

- Some of today's slides adapted from Dr. Wassying's slides (and Ghezzi et al)
- A3 deadlines
  - ▶ Part 1 - Specification: due 11:59 pm Mar 8
  - ▶ Part 2 - Code: due 11:59 pm Mar 20
- A4
  - ▶ Your own design and specification
  - ▶ Due April 3 at 11:59 pm

# Same Symbol/Term Different Meaning

- Can you think of some symbols/terms that have different meanings depending on the context?
- Homonyms
  - ▶ Homograph - same spelling different meaning, maybe different pronunciation (ex. bank, bow, biweekly, ...)
  - ▶ Homophone - same pronunciation, but different meaning, origin or spelling (ex. new/knew, to/too/two, ...)

# Consistent

- Language and terminology must be consistent within the specification
- Potential problem with homonyms, for instance consider the symbol  $\sigma$ 
  - ▶ Represents standard deviation
  - ▶ Represents stress
  - ▶ Represents the Stefan-Boltzmann constant (for radiative heat transfer)
- Changing the symbol may be necessary for consistency, but it could adversely effect understandability
- Potential problem with **synonyms**
  - ▶ Externally funded graduate students, versus eligible graduate students, versus non-VISA students
  - ▶ Enter key versus Return key
  - ▶ **Other examples?**

# Complete

- Internal completeness
  - ▶ The specification must define any new concept or terminology that it uses
    - ▶ A glossary is helpful for this purpose
- External completeness
  - ▶ The specification must document all the needed requirements
    - ▶ Difficulty: when should one stop?

# Incremental

- Referring to the specification process
  - ▶ Start from a sketchy document and progressively add details
  - ▶ A document template can help with this
- Referring to the specification document
  - ▶ Document is structured and can be understood in increments
  - ▶ Again a document template can help with this

# Another Example

- Operational specification
  - ▶ “Let  $a$  be an array of  $n$  elements. The result of its sorting is an array  $b$  of  $n$  elements such that the first element of  $b$  is the minimum of  $a$  (if several elements of  $a$  have the same value, any one of them is acceptable); the second element of  $b$  is the minimum of the array of  $n - 1$  elements obtained from  $a$  by removing its minimum element; and so on until all  $n$  elements of  $a$  have been removed.”
- Descriptive specification
  - ▶ “The result of sorting array  $a$  is an array  $b$  which is a permutation of  $a$  and is sorted.”
  - ▶ How can we further specify (formalize) the notion of sorted?



# Another Example

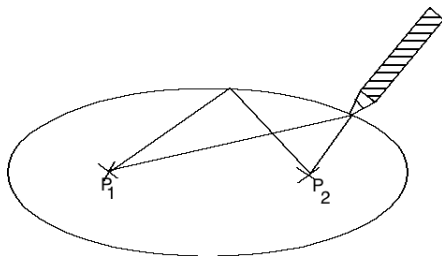
- Operational specification
  - ▶ “Let  $a$  be an array of  $n$  elements. The result of its sorting is an array  $b$  of  $n$  elements such that the first element of  $b$  is the minimum of  $a$  (if several elements of  $a$  have the same value, any one of them is acceptable); the second element of  $b$  is the minimum of the array of  $n - 1$  elements obtained from  $a$  by removing its minimum element; and so on until all  $n$  elements of  $a$  have been removed.”
- Descriptive specification
  - ▶ “The result of sorting array  $a$  is an array  $b$  which is a permutation of  $a$  and is sorted.”
  - ▶ How can we further specify (formalize) the notion of sorted?
  - ▶  $\text{sorted}(A) \equiv \forall(i : \mathbb{N} | 0 \leq i \leq (|A| - 2) : A[i] \leq A[i + 1])$

# Classification of Specification Styles

- Informal, semi-formal, formal
- Operational
  - ▶ Behaviour specification in terms of some abstract machine
- Descriptive
  - ▶ Behaviour described in terms of properties
- The module state machine specification that we use is a mix of operational and descriptive specification - Why?

# Example Operational Specification

- Specification of a geometric figure  $E$
- $E$  can be drawn as follows
  1. Select two points  $P_1$  and  $P_2$  on a plane
  2. Get a string of a certain length and fix its ends to  $P_1$  and  $P_2$
  3. Position a pencil as shown in the next figure
  4. Move the pen clockwise, keeping the string tightly stretched, until you reach the point where you started drawing



# Example Descriptive Specification

Geometric figure  $E$  is described by the following equation

$$ax^2 + by^2 + c = 0$$

where  $a$ ,  $b$  and  $c$  are suitable constants

# Homework Exercise

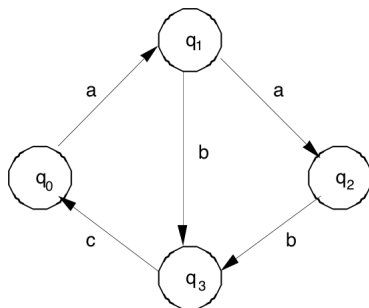
- Consider the **line formatter** specification and
  1. How well does the specification do with respect to the following qualities: abstract, correct, unambiguous, complete, consistent and verifiable?
  2. For a requirement specification like that given, what are the advantages and disadvantages of maintaining both a formal specification and a natural language specification?
- Even spending 5 minutes thinking about will help when we discuss next week
- In repo
  - ▶ The [line formatter specification](#)
  - ▶ [Meyer \(1985\)](#) “On Formalism in Specification”
- Will discuss next day

# How to Verify a Specification

- Observe dynamic behaviour of the specified system
  - ▶ Simulation
  - ▶ Prototyping
  - ▶ “testing” the specification
- Analyze properties of the specified system
- Analogy with traditional engineering
  - ▶ Physical model of a bridge (prototype)
  - ▶ Mathematical model of a bridge
- We will return to this topic when we cover verification (Chapter 6)

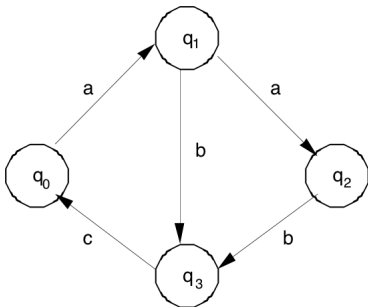
# Finite State Machines (FSMs)

- Can specify control flow aspects
- Defined as
  - ▶ A finite set of states  $Q$
  - ▶ A finite set of inputs  $I$
  - ▶ A transition function  $\delta : Q \times I \rightarrow Q$  ( $\delta$  can be a partial function)



# FSMs Continued

	<b>q<sub>0</sub></b>	<b>q<sub>1</sub></b>	<b>q<sub>2</sub></b>	<b>q<sub>3</sub></b>
<b>a</b>	$q_1$	$q_2$	-	-
<b>b</b>	-	$q_3$	$q_3$	-
<b>c</b>	-	-	-	$q_0$

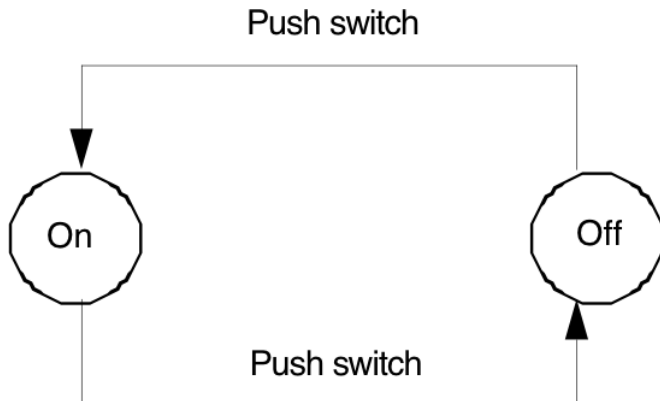




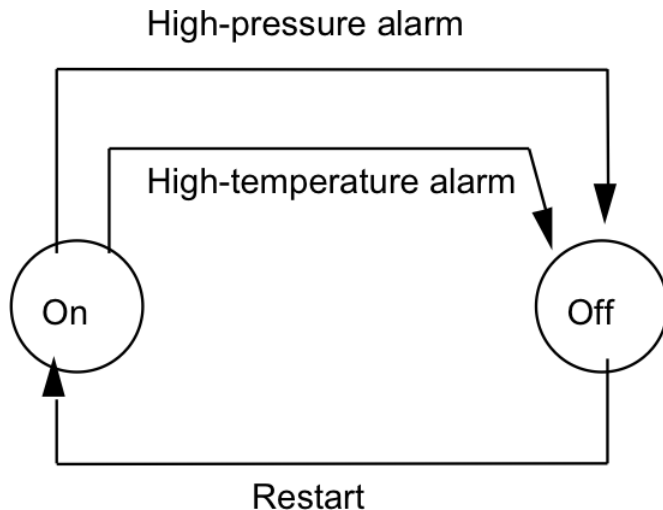
# Example: A Lamp

- What are the states  $Q$  for a typical lamp?
- What are the set of inputs  $I$
- What is the transition function  $\delta$ ?

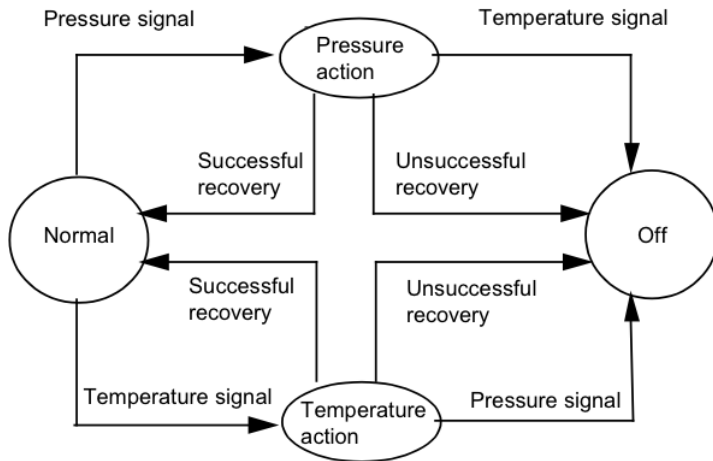
## Example: A Lamp



## Example: A Plant Control System



# A Refinement



# When to use FSMs for Specification?

- When is an FSM a good choice for specification?
- What are some examples of things we would specify using an FSM?

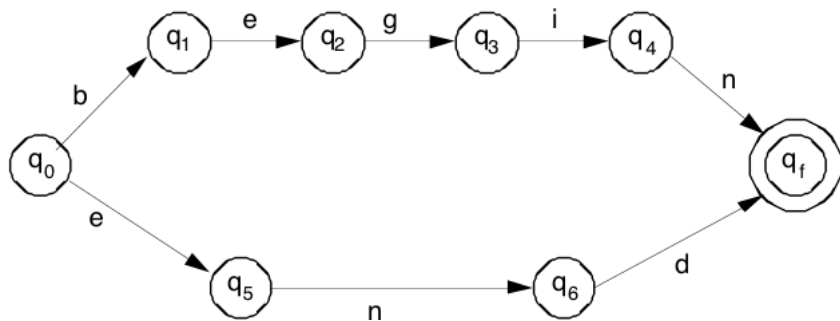
# When to Potentially use FSMs

- Describing control flow
- Clear finite set of states (or modes)
- Specify acceptable strings for a parser
- Specifying hardware design
- For synchronous models (at any time a global state must be defined and a single transition must occur)

# Classes of FSMs

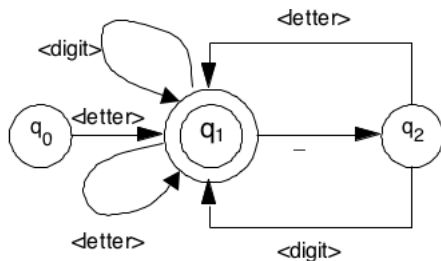
- Deterministic/nondeterministic
- FSMs as recognizers - introduce final states
- FSMs as transducers - introduce set of output
- ...

# FSMs as Recognizers





# FSMs as Recognizers Continued



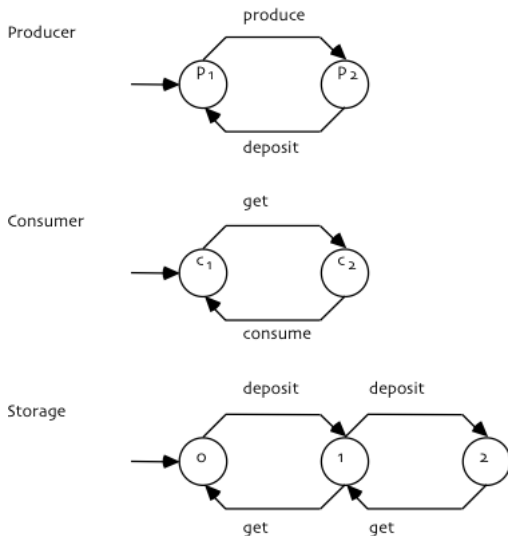
Legend:  $\xrightarrow{\langle \text{letter} \rangle}$  is an abbreviation for a set of arrows  
respectively

$\xrightarrow{\langle \text{digit} \rangle}$  is an abbreviation for a set of arrows  
labeled 0, 1, ..., 9, respectively

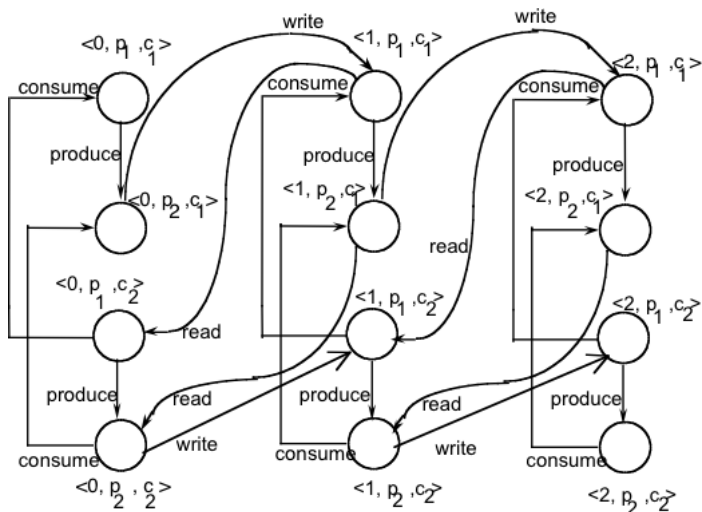
# Limitations

- Finite memory
- State explosion - Given a number of FSMs with  $k_1, k_2, \dots, k_m$  states, their composition is an FSM with  $k_1 \times k_2 \times \dots \times k_n$ . This growth is exponential with the number of FSMs, not linear (we would like it to be  $k_1 + k_2 + \dots + k_n$ )

# State Explosion: An Example



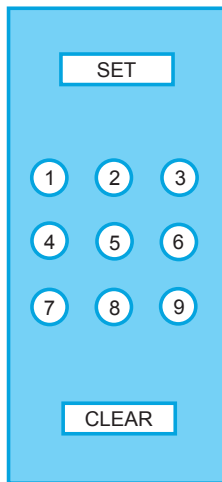
# The Resulting FSM



# Events Versus Conditions

- Events can be viewed as “pulses” in time - they do not last (retain their values)
- Conditions may retain their values indefinitely

# FSM Example: Security Alarm



# Security Alarm Example Continued

