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

Winter 2018

23 Finite State Machines (Ch. 5)

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 7, 2018

23 Finite State Machines (Ch. 5)

- 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. Wassyng's slides (and Ghezzi et al)
- A3
  - Part 1 - Specification: due 11:59 pm Mar 12
  - Part 2 - Code: due 11:59 pm Mar 26
  - spec.tex moved to se2aa4_cs2me3 repo
  - Minor corrections
    - LineT len output type better as $\mathbb{N}$ (rather than $\mathbb{Z}$)
    - LineT strt output should be $out := \text{PointT}(s.x(), s.y())$
    - PathT len output the total number of points (grid cells) on the path, including the beginning and end points (cells).
- A4
  - Your own design and specification
  - Due April 9 at 11:59 pm

A3

- `make experiment`
- `make doc`
- `make test` (uses catch2)
- Version 7 of g++
- Strongly recommend testing on mills
- On mills, modify `.bashrc` to include
  `. /opt/rh/devtoolset-7/enable`
- Use vectors for Seq2D constructor
- No pointers
- Follow tutorial approach for implementing templates

```
template <class T>
Seq2D<T>::Seq2D(vector<vector<T>> s, double scal
{ //details
}
```

Operational Versus Descriptive Spec

- 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
  - What is the corresponding descriptive spec?
  - How can we further specify (formalize) the notion of sorted?

Operational Versus Descriptive Spec

- 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
  - What is the corresponding descriptive spec?
  - "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?

Operational Versus Descriptive Spec

- 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
  - What is the corresponding descriptive spec?
  - "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?
  - $\text{sorted}(A) \equiv \forall(i : \mathbb{N} | 0 \leq i \leq (|A| - 2) : A[i] \leq A[i+1])$

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

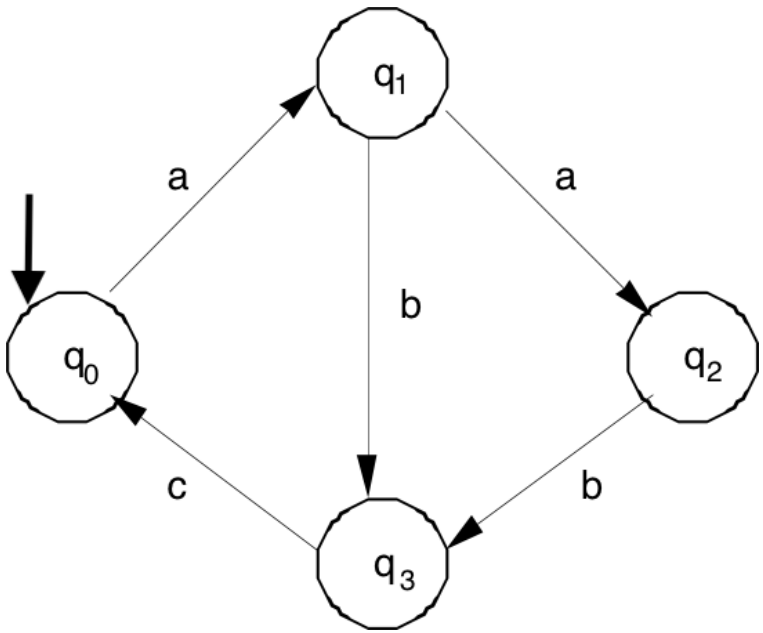
- How might you verify a specification?

How to Verify a Specification

- How might you verify a specification?
- *Observe* dynamic behaviour of the specified system
  - Simulation
  - Prototyping
  - "testing" the specification
- Mathematically analyze properties of the specified system, including proof
- 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)

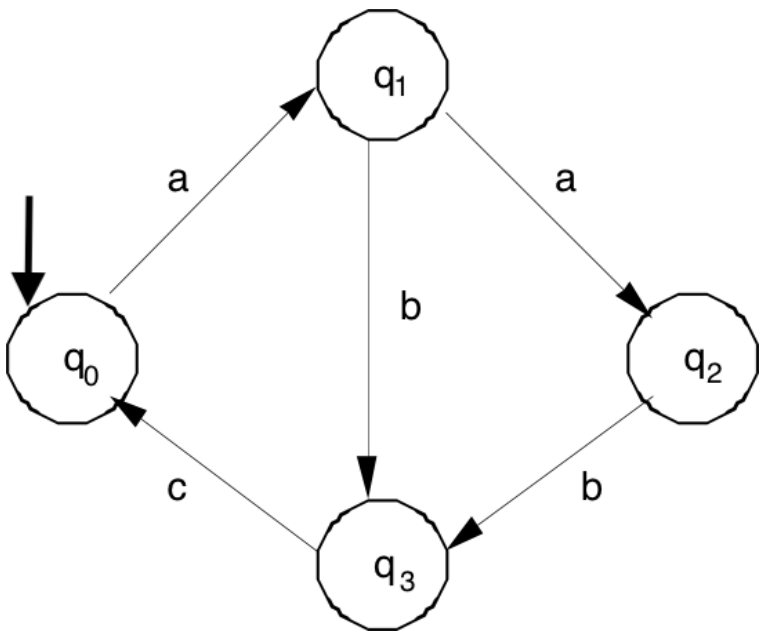
Finite State Machines (FSMs)

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



FSMs Continued

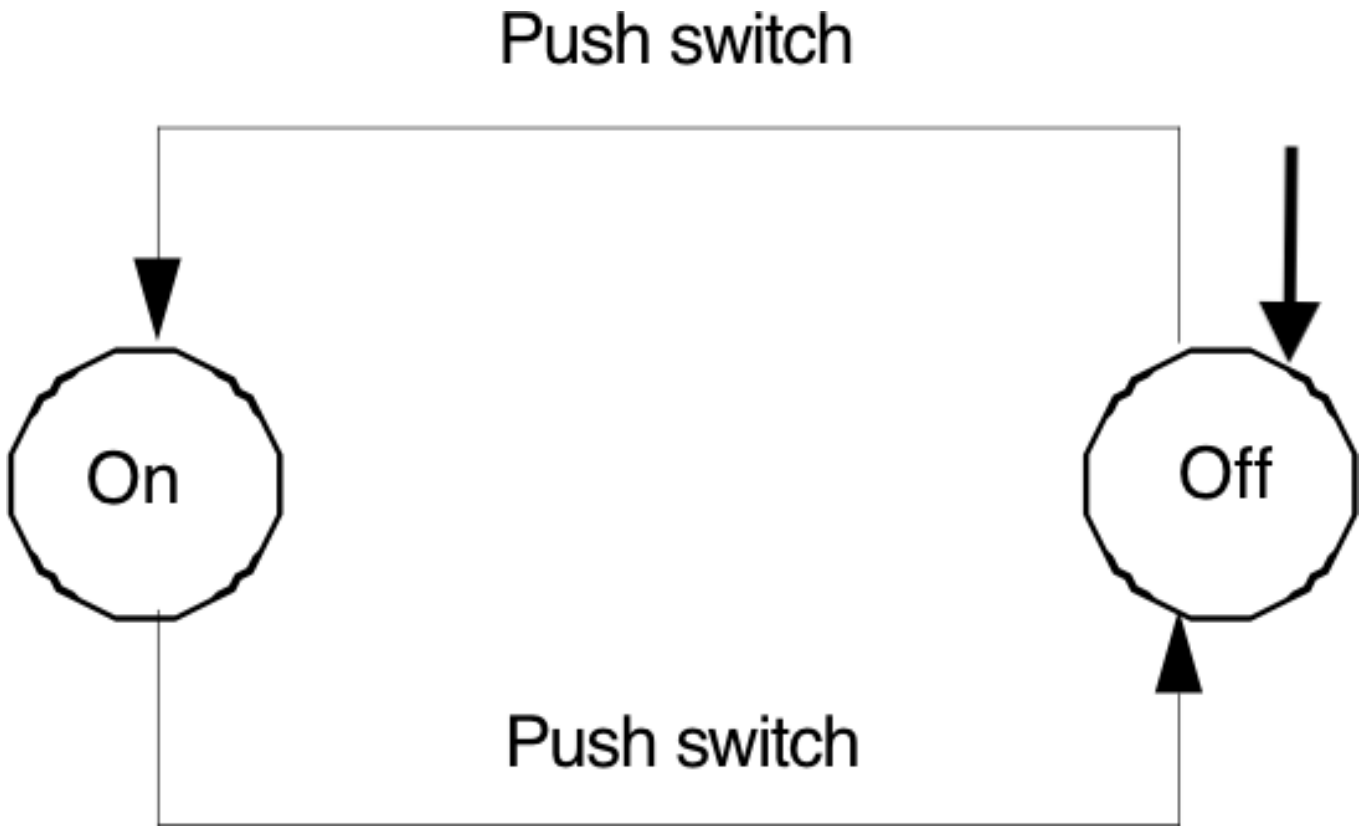
	$\mathbf{q_0}$	$\mathbf{q_1}$	$\mathbf{q_2}$	$\mathbf{q_3}$
a	q_1	q_2	-	-
b	-	q_3	q_3	-
c	-	-	-	q_0



Example: A Lamp

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

Example: A Lamp

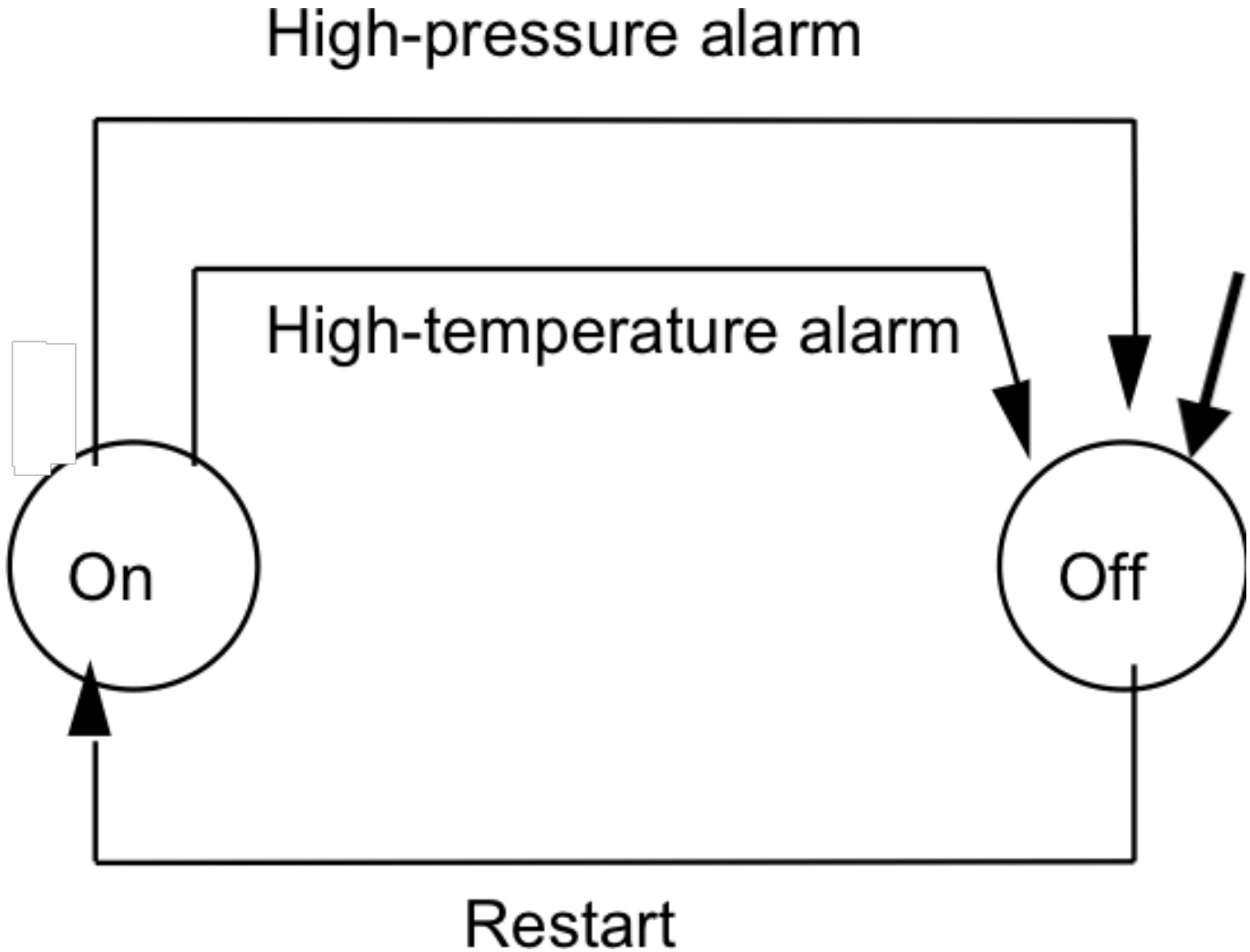


Example: A Plant Control System

Your plant is either On or Off. If an alarm occurs, what should be the state transition?

Example: A Plant Control System

Your plant is either On or Off. If an alarm occurs, what should be the state transition?

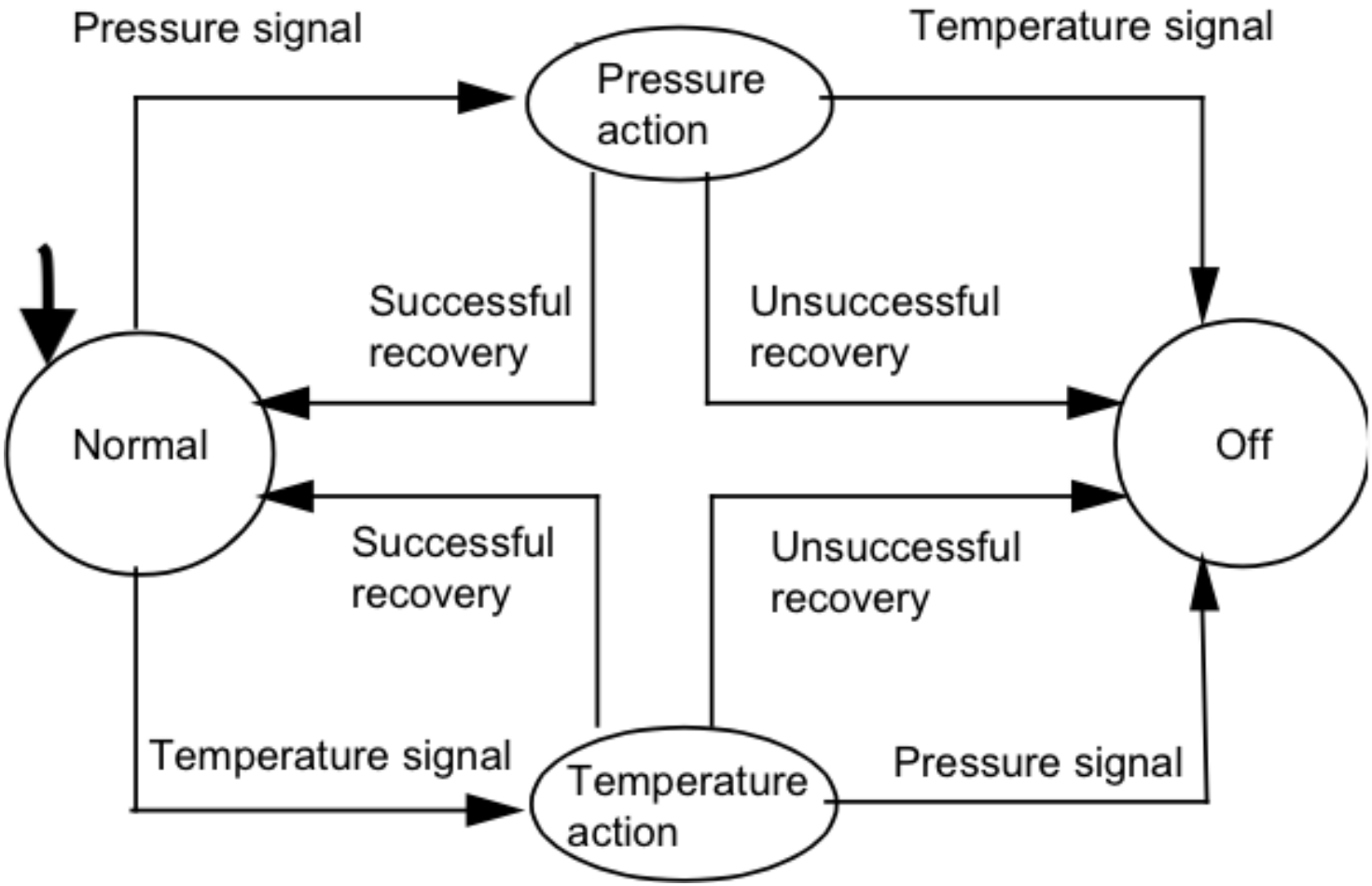


A Refinement

How might your refine the FSM if you have states for "pressure action" and "temperature action"?

A Refinement

How might your refine the FSM if you have states for "pressure action" and "temperature action"?



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?

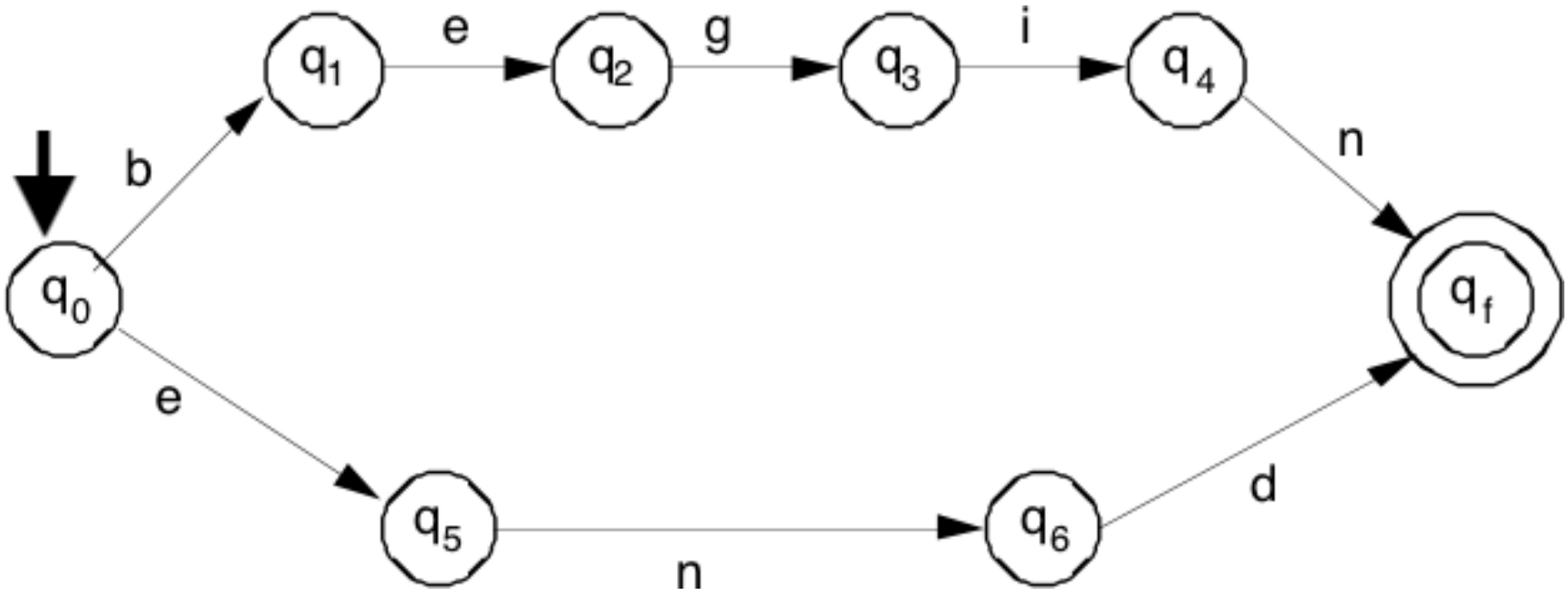
When to Potentially use FSMs

- 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

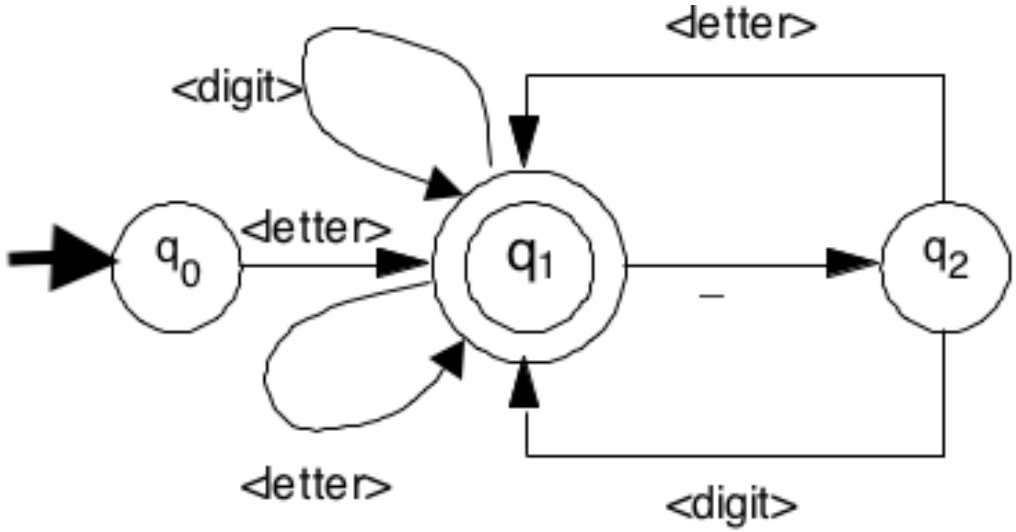
- Deterministic/nondeterministic
- FSMs as recognizers - introduce final states
- FSMs as transducers - introduce set of output
- ...

FSMs as Recognizers



What if an invalid character is entered?

FSMs as Recognizers Continued



Legend: <letter> ——→ is an abbreviation for a set of arrows

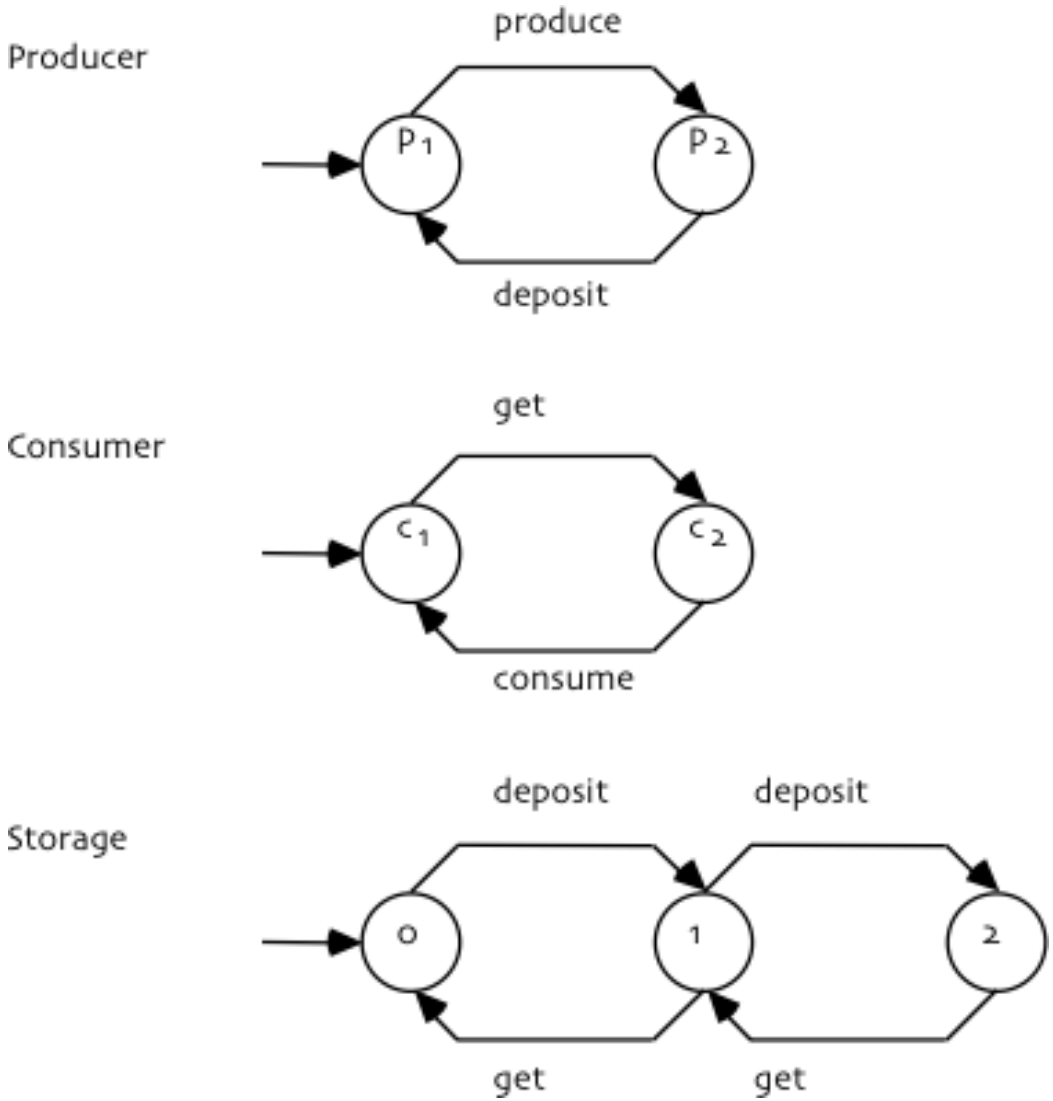
respectively

<digit> ——→ is an abbreviation for a set of arrows labeled 0, 1,..., 9, respectively

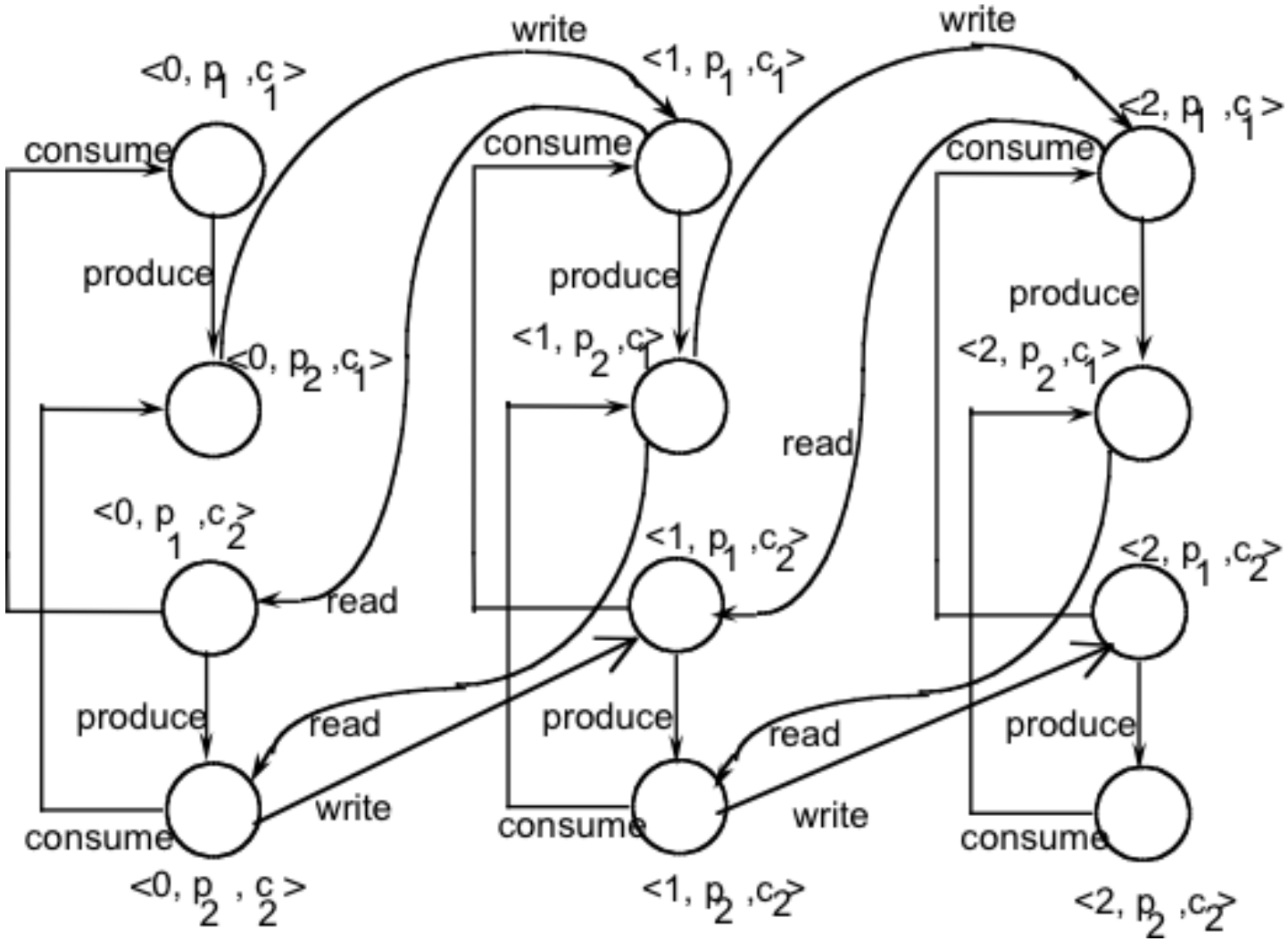
Limitations

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

State Explosion: An Example



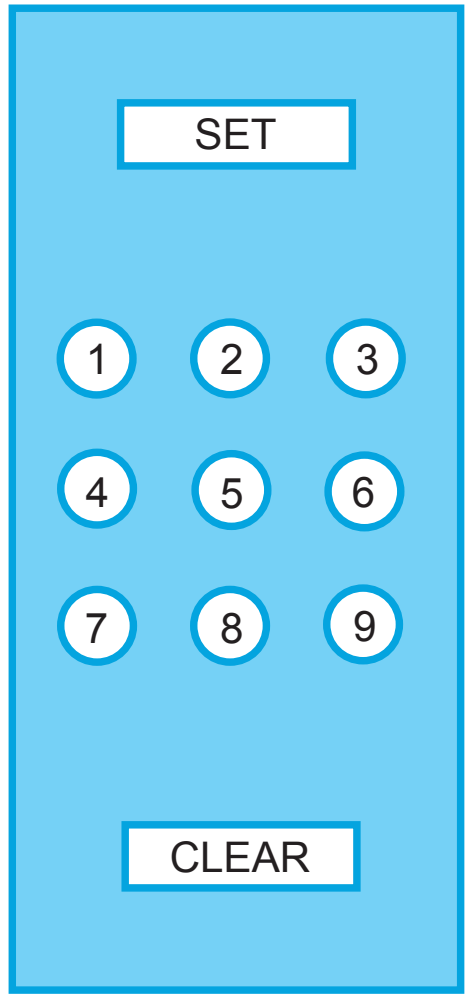
The Resulting FSM



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