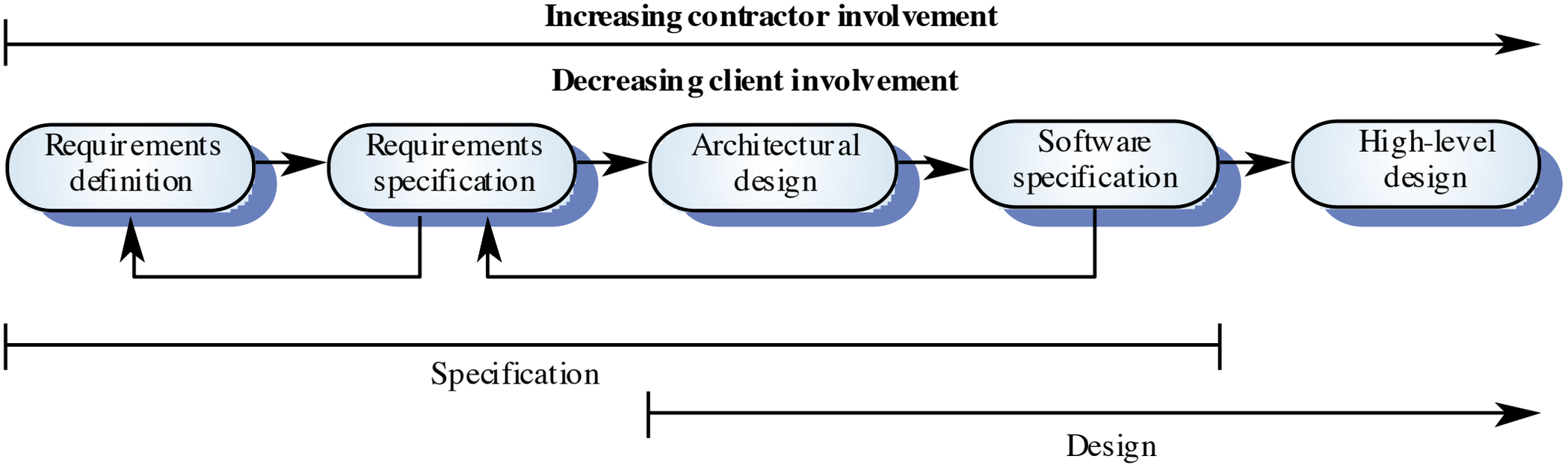
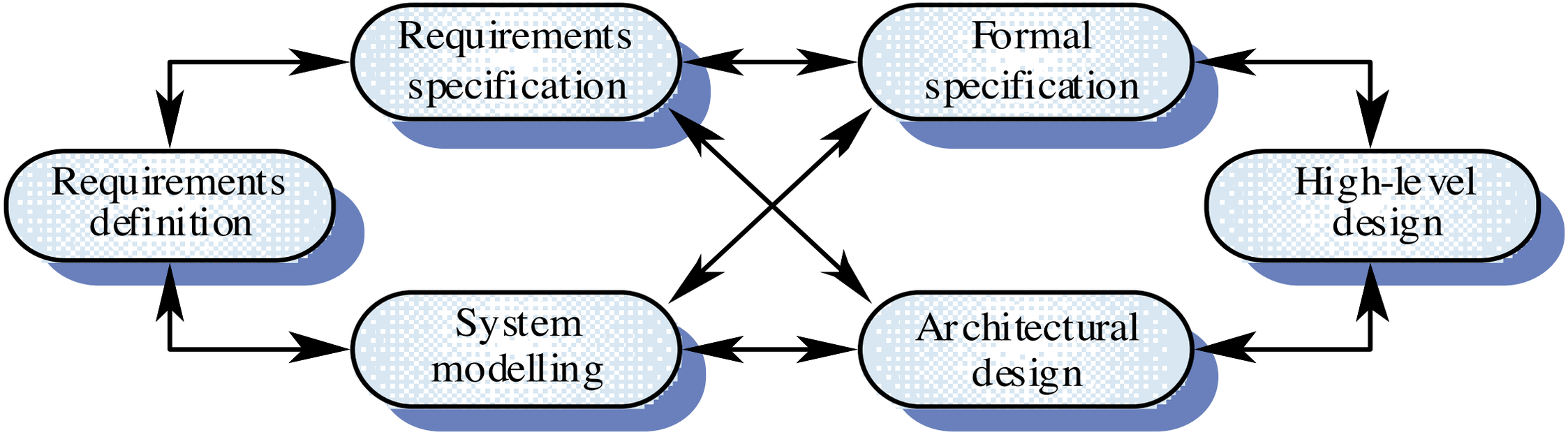
Specifications

- Specification is a precise statement about …..???
  - a statement of contract between producer/implementer and consumer/user of the service.
- Normally associated with requirements …..
- But is present at each stage….
  - specify design, specify architectures, specify testing strategies etc…
- Specification at some level states
  - the requirements for the implementation at a lower level.

Specification and design



Specification in the software process



Types of specifications

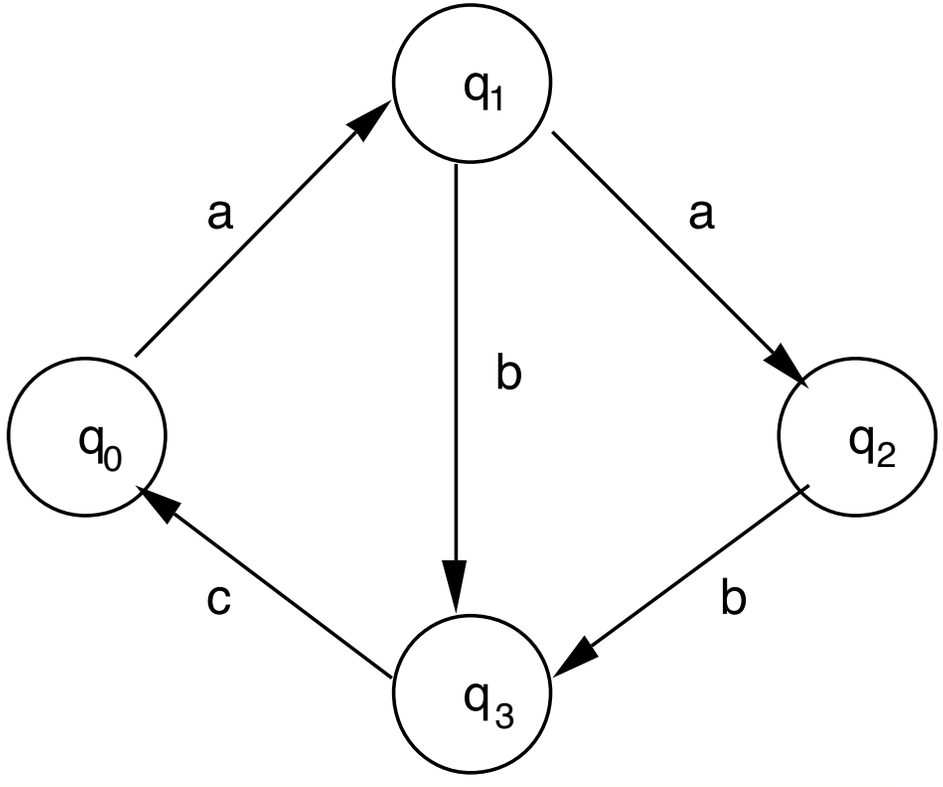
- operational
  - Data Flow Diagrams
  - (Some) UML diagrams
  - Finite State Machines
  - Petri Nets
- descriptive
  - Entity Relationship Diagrams
  - Logic-based notations
  - Algebraic notations
- Languages for modular specifications
  - Statecharts, Z, Alloy

Formal Specification techniques

- Finite state machines
- Petri nets
- Logic specification
- Algebraic specifications.

Finite State Machines

- Can specify control flow aspects
- defined as
  - a finite set of states, Q;
  - a finite set of inputs, I;
  - a transition function d : Q x I → Q
  - (d can be a partial function)



Finite State Machines…

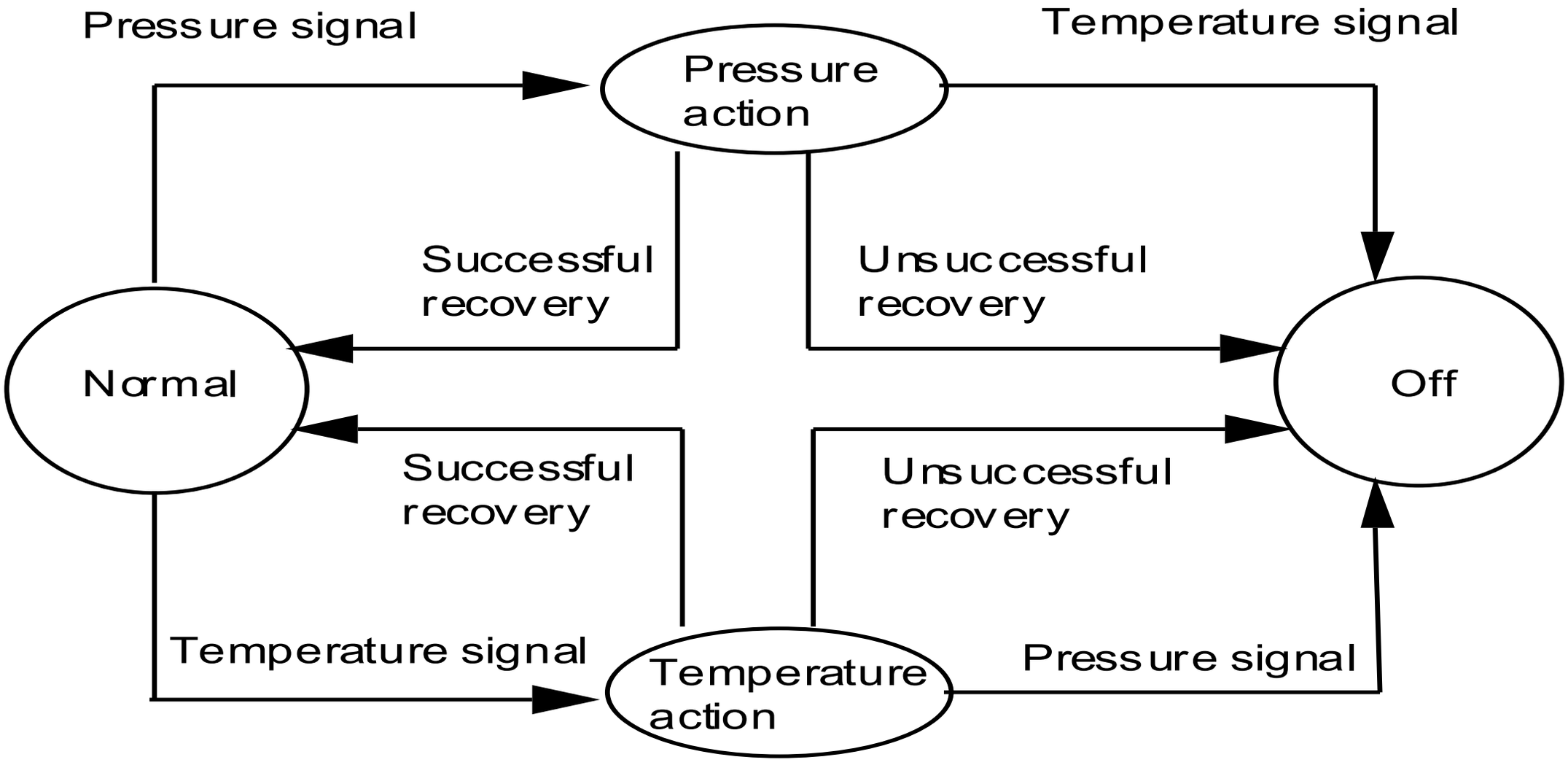
- Design a FSM to model the behaviour of a lamp with switch actions: Push_1 leading to OFF state and Push_2 leading to ON state.
- Design a FSM for a Chemical Plant Control system to model the following behaviour:
  - if there is a High temperature OR a High Pressure Signal the plant is moved to the OFF state from the normal ON state…
  - the plant is to be restarted once in the OFF state.

Tutorial Exercise

- Draw a simple FSM to model the following behavior:
  - In a Chemical plant, if there is a High-pressure signal OR there is a High-temperature signal then the plant is shut OFF.
- Modify the FSM given above (chemical plant), by considering the case…
  - where temperature & pressure have two different signals – one indicating a slight deviation from the acceptable & the other a dangerous deviation from the acceptable value. In the latter case, system must shut off immediately.
- Modify the chemical plant FSM to cope also with simultaneous signals.

Finite State machines

- Consider the Chemical plant control system – modified as follows



Types of FSMs

- Deterministic/nondeterministic
- FSMs as recognizers
  - introduce final states
  - drawn as doubly circled nodes
- FSMs as transducers
  - introduce set of outputs – Mealy & Moore machines

Tutorial Exercises……

- Describe a system with two lamps and a button. When the lights are OFF, pushing the button causes the first lamp to go on. Pushing the button again causes the second lamp to go ON and the first to go OFF. Pushing the button yet again causes both the lamps to go ON and pushing it once more causes both to go OFF.

Tutorial Exercise…

- Consider a network of sensors could be used in an industrial setting to monitor groundwater contamination. Power consumption is a major issue for sensors, so a monitoring system should be designed to conserve power as much as possible. For this exercise, you will draw a finite state machine (FSM) to model the following sensor system.
  - The system will be initialized upon power up
  - If a normal reading is received, no action will occur. If an abnormal reading is received, the system will send an alert. Additional abnormal readings will not be reported (to conserve power). When a normal reading is received, the system will again send an alert.
    Draw your FSM to model the above description.

Limitations

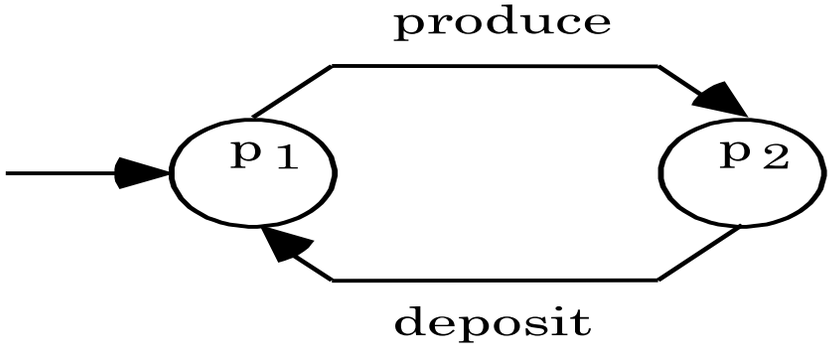
- Finite memory
- State explosion

} limit expressive power

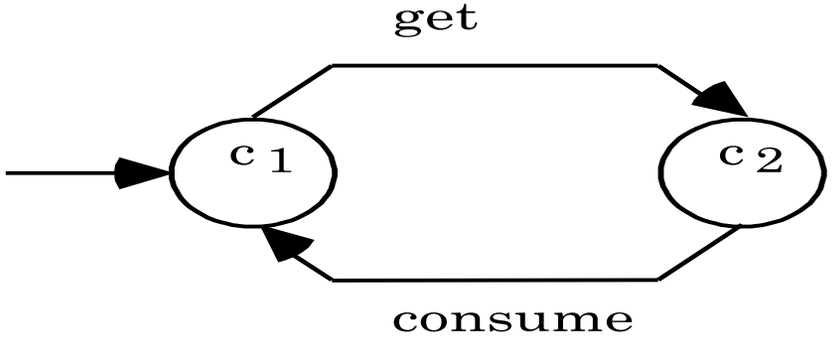
- Given a number of FSMs with k_1, k_2, … k_n states,
  - their composition is a FSM with $k_1 * k_2 *… * k_n$.
  - this growth is exponential with the number of FSMs, not linear (we would like it to be $k_1 + k_2 +… + k_n$)

State space explosion – an example

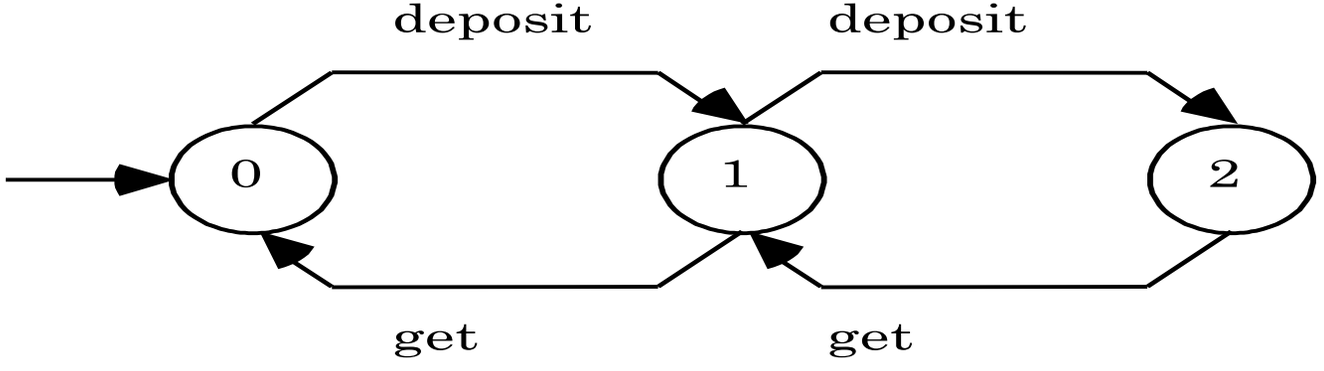
Producer



Consumer



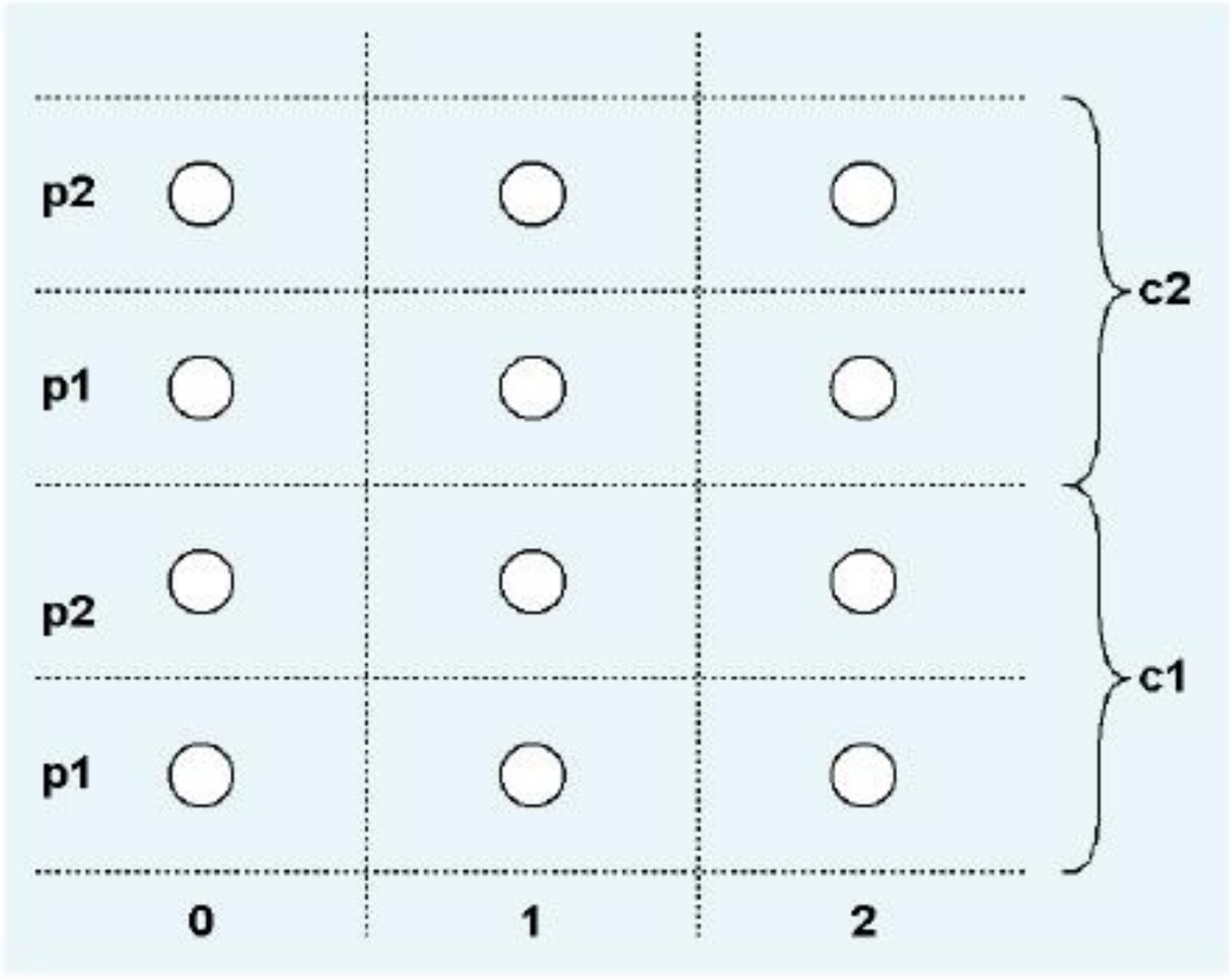
Storage



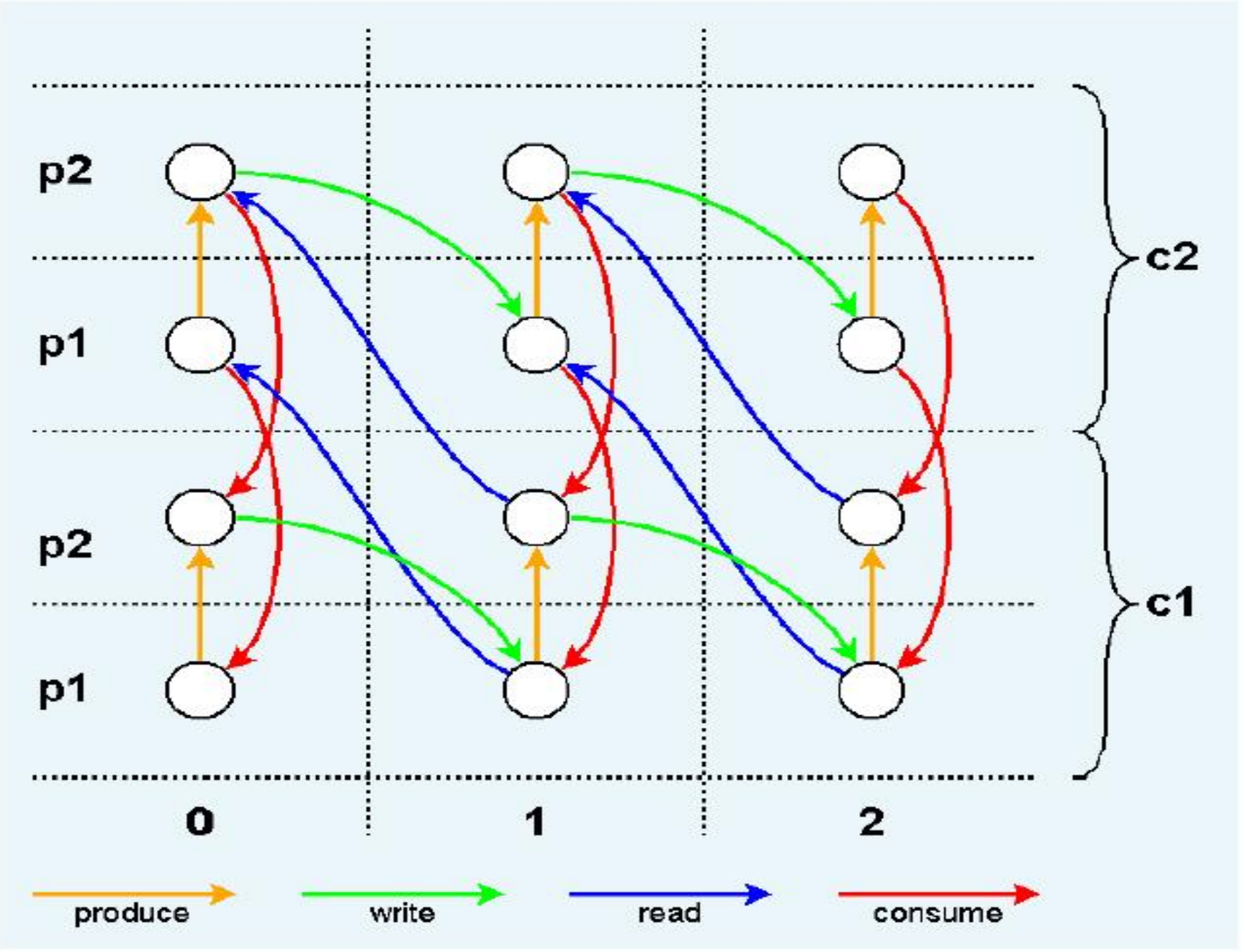
State space explosion – an example

- Draw a parallel FSM from the above with concurrent Producer Consumer activities……

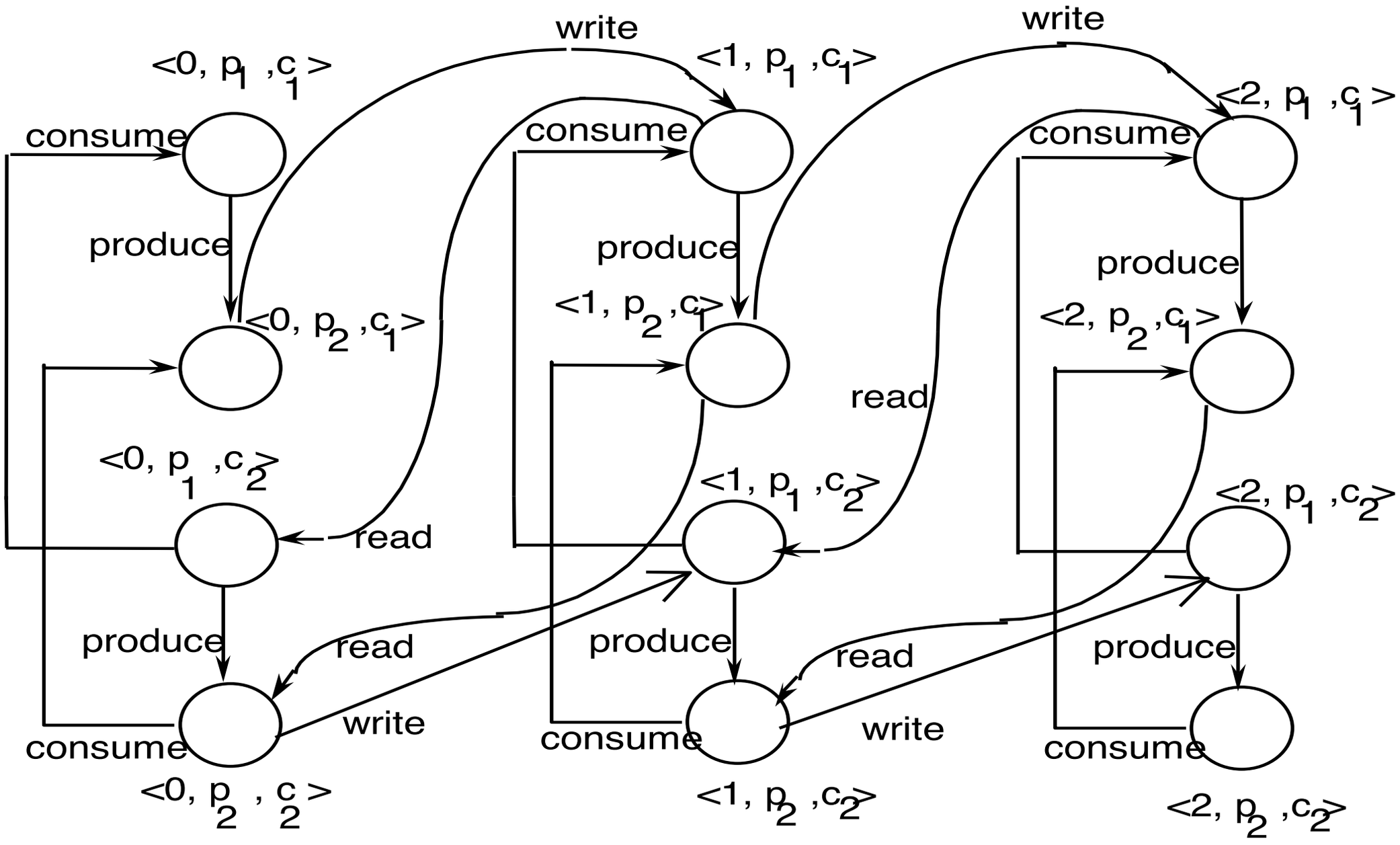
State space explosion – an example…..



State space explosion – an example…..



Integrated description



Review

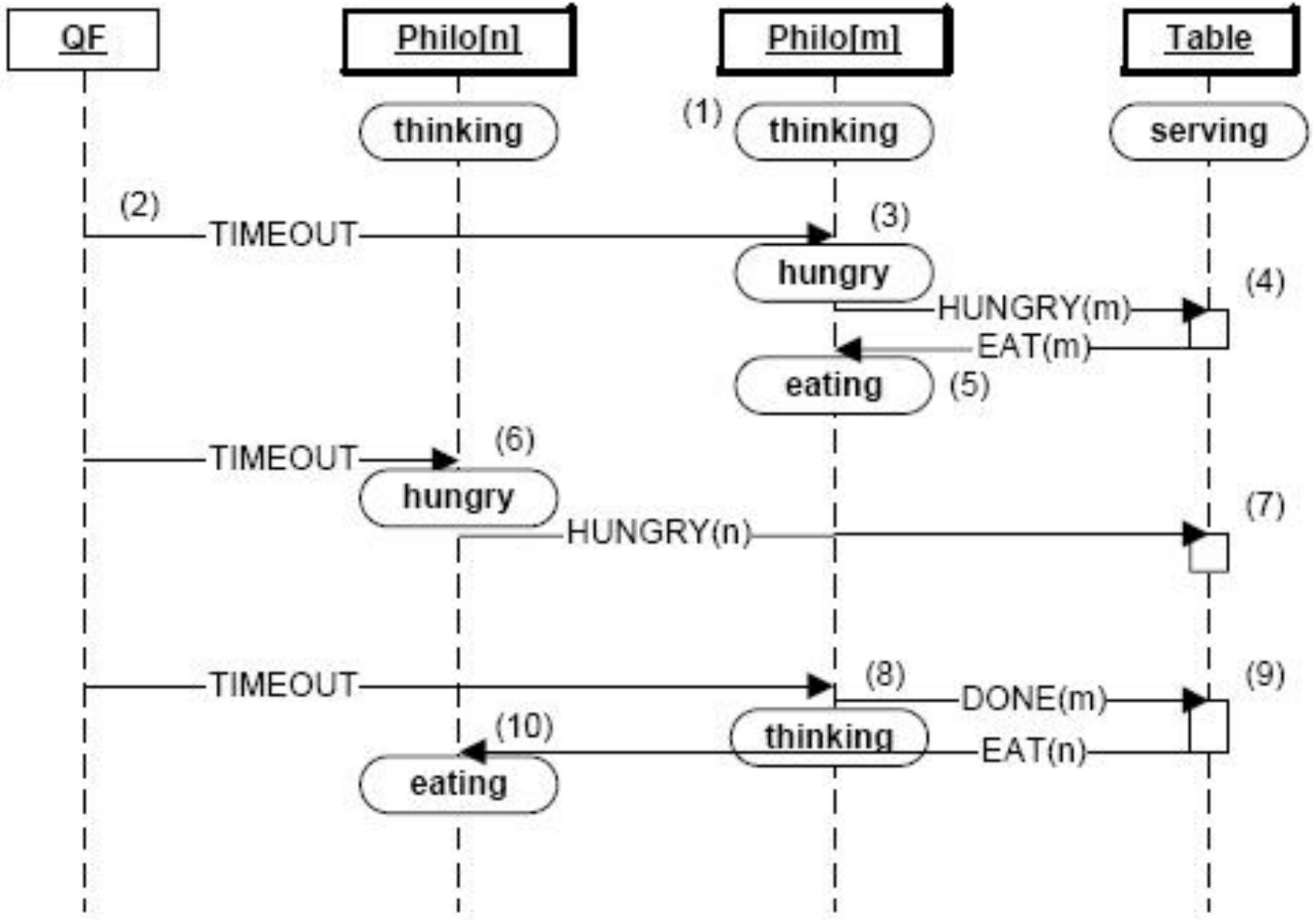
- The cardinality of the state space grows dramatically.
- Can we use FSM for concurrent systems – with several concurrent units ?
- Can we model further required details e.g. in Chemical plant example, to model "cooling-effort" applied proportional rise in the temperature in case of a "temperature signal"…….
- Is there a serialization between the production by the producer and consumption by the consumer, above ?
- Are we assuming that "consuming" and "producing" are sort of instantaneous ?
- FSMs are essentially a synchronous model – asynchronous transitions aren't described by FSM.
- FSM is not explicitly aimed at concurrent systems.

Tutorial Exercise….

- Five philosophers are gathered around a table with a big plate of spaghetti in the middle. Between each philosopher is a fork. The spaghetti is so slippery that a philosopher needs two forks to eat it. The life of a philosopher consists of alternate periods of thinking and eating. When a philosopher wants to eat, he tries to acquire forks. If successful in acquiring two forks, he eats for a while, then puts down the forks and continues to think.

  The key issue is that a finite set of tasks (philosophers) is sharing a finite set of resources (forks), and each resource can be used by only one task at a time. (An alternative oriental version replaces spaghetti with rice and forks with chopsticks, which perhaps explains better why philosophers need two chopsticks to eat.)

Tutorial Ex...Sequence Diagram [Quantum Leaps]

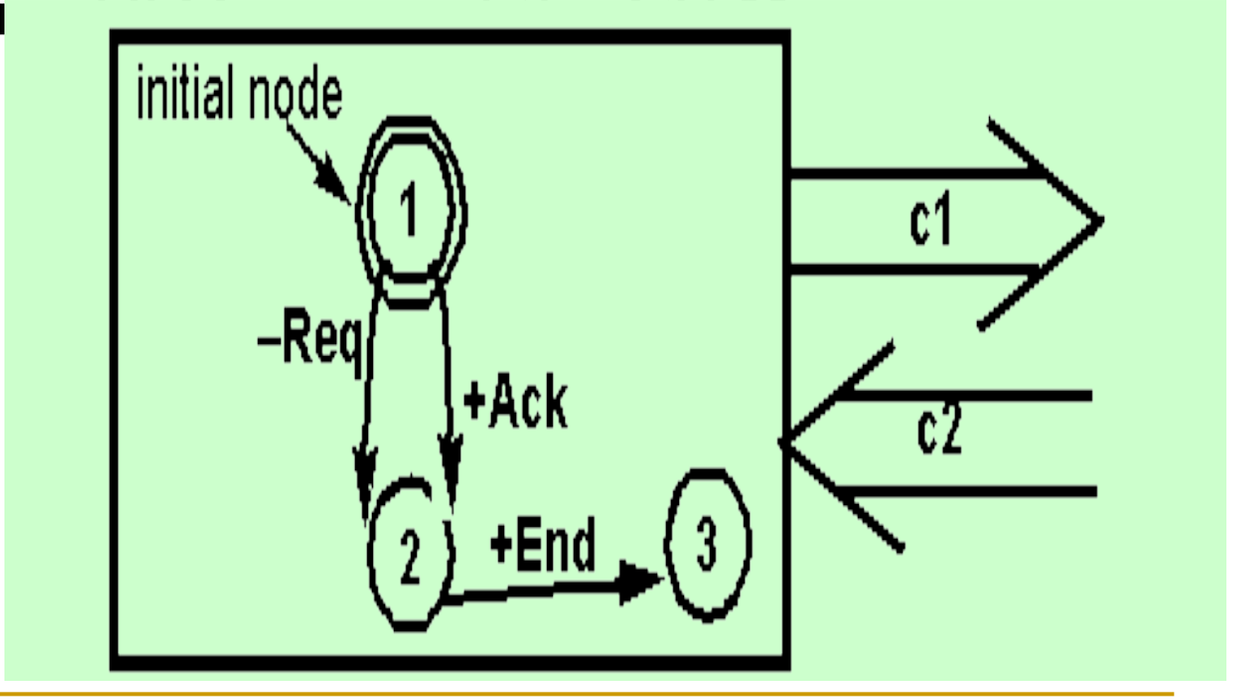


Communicating Finite State Machines (CFSM)

- A Protocol is described as a set of Communicating Finite State Machines.
- Each CFSM represents
  - a component (or process) of the network
    - i.e. in OSI term, a protocol entity, viz. a sender OR a receiver
- Each CFSM is represented by a labeled DAG where
  - nodes represent states (conditions) of the process;
  - edges represent transitions (events) of the process.

Communicating Finite State Machines (CFSM)

- Transitions include actions taken by the process (e.g. the sending of a message) or external stimuli (e.g. the reception of a message).
- Transitions are labeled as
  - the sending message transition is labelled as -Msg where Msg is the type of messages being sent
  - the receiving message transition is labelled as +Msg where Msg is the head message on the incomi

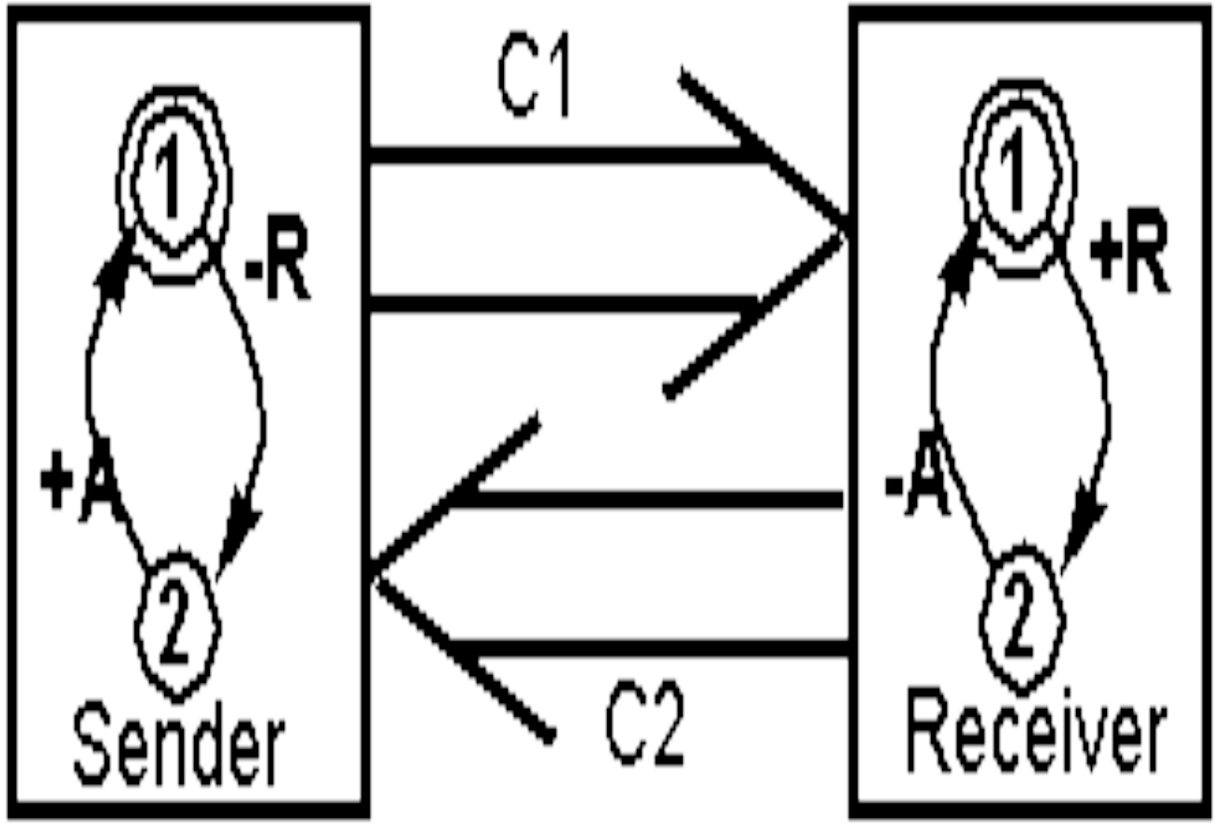


CFSM operating semantic [Quantum Leaps]

- The channels that connect CFSM's are assumed to be FIFO queues.
  - an error-prone channel is modeled as a CFSM.
- Node Status
  - Initial node……starting state of a CFSM.
  - Final node…..no transition.
  - Receiving node
    - all (outgoing) transitions are receiving transitions.
    - If no message or incorrect msg is in the channel, the node will be blocked.
  - Sending node
    - all transitions are sending transitions.
    - They are not blocked.
  - Mix node
    - has both receiving and sending transition.

CFSM illustrations [Quantum Leaps]

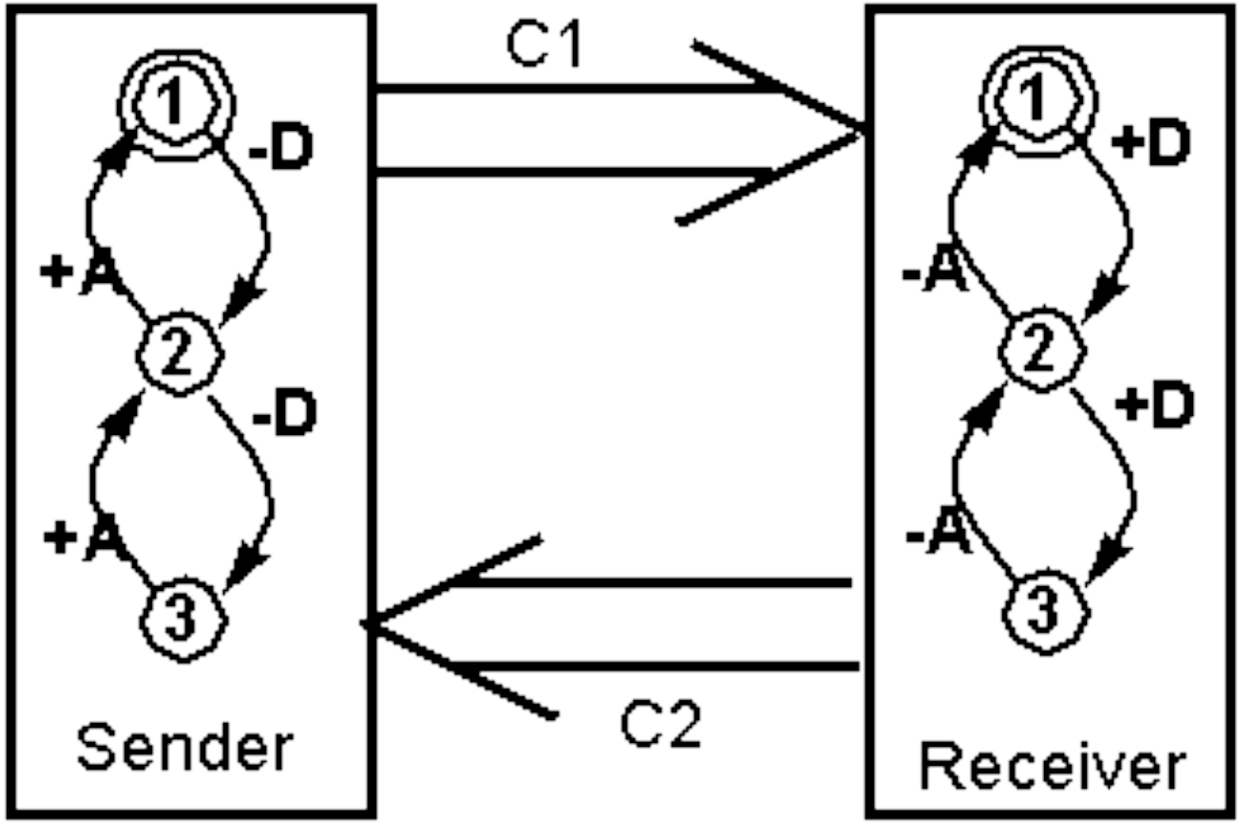
- Simple Request Response Protocol



R: request
A: Acknowledgment

CFSM operating semantic [Quantum Leaps]

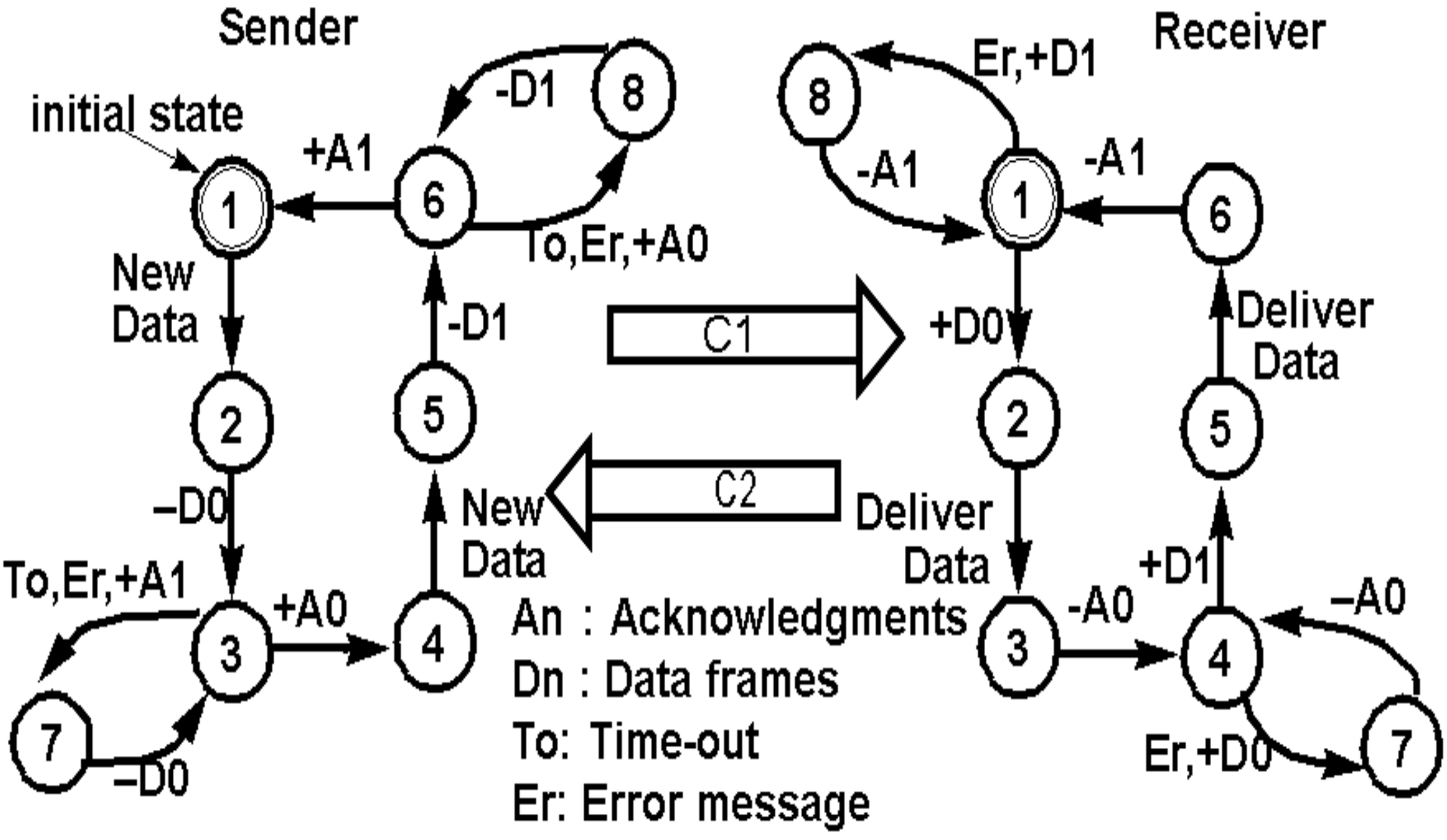
- A Sliding Window FSM with window size of 2



D: Data
A: Acknowledgment

What if we distinguish the two data messages and the two acks?

CFSM- Alternating-bit Protocol [Quantum Leaps]



Specification Properties and Types of Specifications

Uses of specification

- Specification & Implementation
  - *the what vrs how dichotomy.*
- Statement of user requirements
- "The hardest single part of building a software system is deciding precisely what to build" (F. Brooks)

Uses of specification (contd)

- A Specification can be
  - Statement of the i/f between the machine & the controlled environment
    - control function can have serious undesirable effects e.g. …..??
    - so, precisely define what are inputs, outputs, expected relationships and time constraints.
  - Statement of requirements for implementation
    - useful guideline during the design and verification
  - A reference point during maintenance
    - corrective maintenance only changes implementation
    - adaptive and perfective maintenance occur because of requirements changes
    - requirements specification must change accordingly

An example: defining requirements ..

- Problem Definition:
  - The software must provide a means of representing and accessing external files created by other tools.

An example: defining requirements(contd)

- Specification
  - The user should be provided with facilities to define the type of external files.
  - Each external file type may have an associated tool which may be applied to the file.
  - Each external file type may be represented as a specific icon on the user's display.
  - Facilities should be provided for the icon representing an external file type to be defined by the user.
  - When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected icon.

Spec qualities - Precise, clear, unambiguous

- Example: specification fragment for a word-processor
  - `Selecting is the process of designating areas of the document that you want to work on. Most editing and formatting actions require two steps: first you select what you want to work on, such as text or graphics; then you initiate the appropriate action.`
- Is it precise, unambiguous, clear ?

- Another example (from a real safety-critical system)
  - `The message must be triplicated. The three copies must be forwarded through three different physical channels. The receiver accepts the message on the basis of a two-out-of-three voting policy.`
- Again, is it precise, unambiguous, clear ?

Spec qualities - Precise, clear, unambiguous

- Bad
  - The Background Task Manager shall provide status messages at regular intervals not less than 60 seconds.
- Should be like…
  - The Background Task Manager (BTM) shall display status messages in a designated area of the user interface
    - The messages shall be updated every 60 plus or minus 10 seconds after background task processing begins.
    - The messages shall remain visible continuously.
    - Whenever communication with the background task process is possible, the BTM shall display the percent completed of the background task.

Source: Wiegers

Spec qualities - Precise, clear, unambiguous

- Bad
  - The XML Editor shall switch between displaying and hiding non-printing characters instantaneously.
- Should be like…
  - The user shall be able to toggle between displaying and hiding all XML tags in the document being edited with the activation of a specific triggering mechanism. The display shall change in 0.1 seconds or less.

Source: Wiegers

Spec qualities - Precise, clear, unambiguous

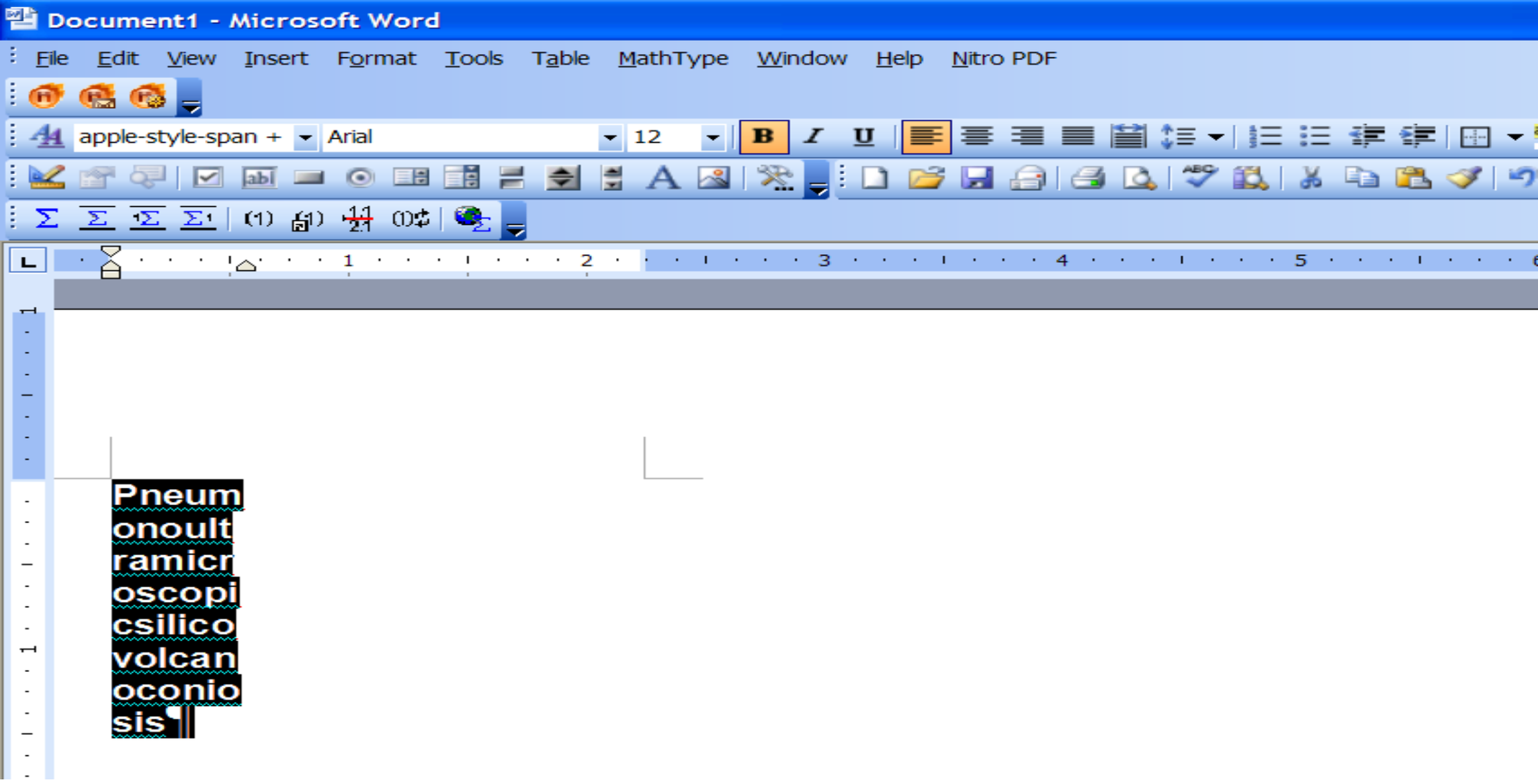
- Bad
  - The XML parser shall produce a markup error report that allows quick resolution of errors when used by XML novices.
- Should be like…
  - After the XML Parser has completely parsed a file, it shall produce an error report that contains the line number and text of any XML errors found in the parsed file and a description of each error found.
  - If no parsing errors are found, the parser shall not produce an error report.

Source: Wiegers

Consistent

- spec for the *justify* action button of the editor
  - `The whole text should be kept in lines of equal length. The length is specified by the user. Unless the user gives an explicit hyphenation command, a carriage return should occur only at the end of a word.`
  - An example of self-contradictory or inconsistent spec. Why ?
  - How would the editor deal with **`Pneumonoultramicroscopicsilicovolcanoconiosis`**

    with specified line length of less than 45 characters ?
  - this word according to the *Oxford English Dictionary is*….,
    - "a factitious word alleged to mean 'a lung disease caused by the inhalation of very fine silica dust, causing inflammation in the lungs. A condition meeting the word's definition is normally called silicosis. [Wiki]



- Where has the CR occurred ?
  - Is it at the end of the word ?
  - Is there an explicit hyphenation ?

Internal Consistency…

- the specified characteristics of real-world objects may conflict e.g.
  - the format of an output report may be described in one requirement as tabular but in another as textual.
  - one requirement may state that all lights shall be green while another may state that all lights shall be blue.
- there may be logical or temporal conflicts between two specified actions e.g.
  - one requirement may specify that the program will add two inputs and another may specify that the program will multiply them.
  - One requirement may state that “A” must always follow “B” while another may requirement may state “A” and “B” occur simultaneously.

Internal Consistency…

- Two or more requirements may describe the same real-world object but use different terms for that object e.g.
  - a program's request for a user input may be called as "prompt" once and may be called as a "cue" in another.
- The use of standard terminology and definitions promotes consistency.

Complete

- internal completeness
  - must define any new concept or terminology that it uses
    - e.g. in the absence of any outstanding requests, the elevator enters a *wait-for-request* state.
    - what is lack of internal completeness here ?
- external completeness
  - functional
  - non-functional or qualitative
- Is it realistic to insist on complete specifications?
  - e.g. "response time of about two seconds …"

Incremental specifications

- Referring to the specification process
  - start from a sketchy document and progressively add details
- Referring to the specification document
  - document is structured and can be understood in increments

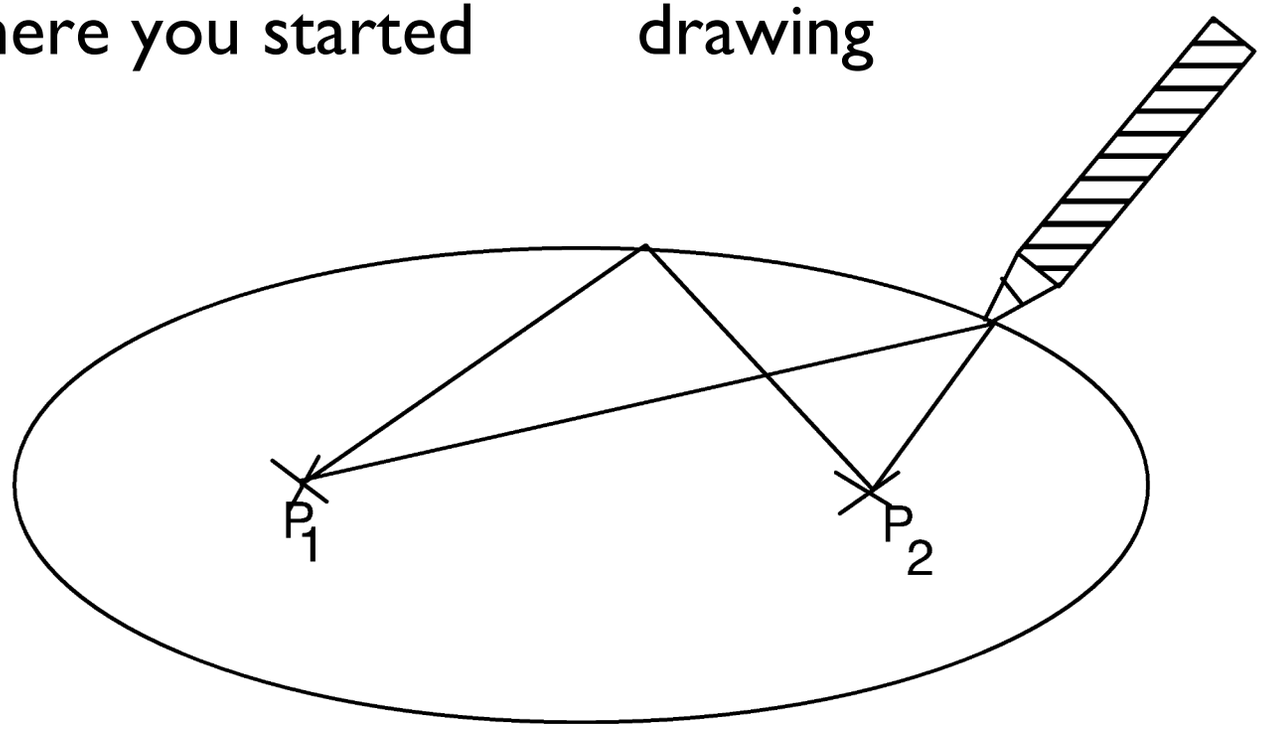
Classification of specification styles

- Informal, semi-formal, formal
- Operational
  - Behavior specification in terms of some abstract machine
- Descriptive
  - Behavior described in terms of properties

Example1: Operational Spec

- A Geometric Fig E can be drawn as follows:

1. Select two points P1 and P2 on a plane
2. Get a string of a certain length and fix its ends to P1 and P2
3. Position a pencil as shown in next figure
4. Move the pen clockwise, keeping the string tightly stretched, until you reach the point where you started drawing



Example 2: functional spec

- "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."

Example1: Descriptive Spec

- Geometric figure E is describe by the following equation

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

  where a, b, and c are suitable constants

Example2 : Descriptive spec

- "`The result of sorting array a is an array b which is a permutation of a and is sorted`"
- descriptive specs are more abstract than operational.
- A combination of descriptive & functional approaches may also be adopted
- Exercise 5.3, 5.4
  - Are the operational & descriptive specs given above unambiguous ?
  - Give a completely descriptive spec of the sort operation which takes care of duplicates.

Review…

OP

"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."

DES

"The result of sorting array a is an array b which is a permutation of a and is sorted."