# Lecture 25
# Petri Nets 2

**David Garlan**
**Carnegie Mellon University**

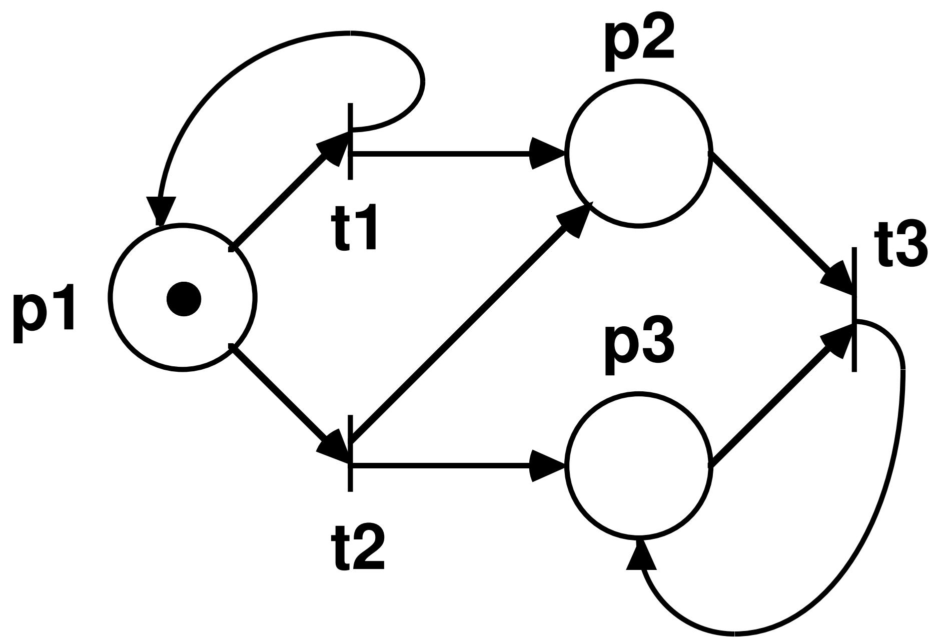
**Models of Software Systems**
**15-671**
**Fall 1995**

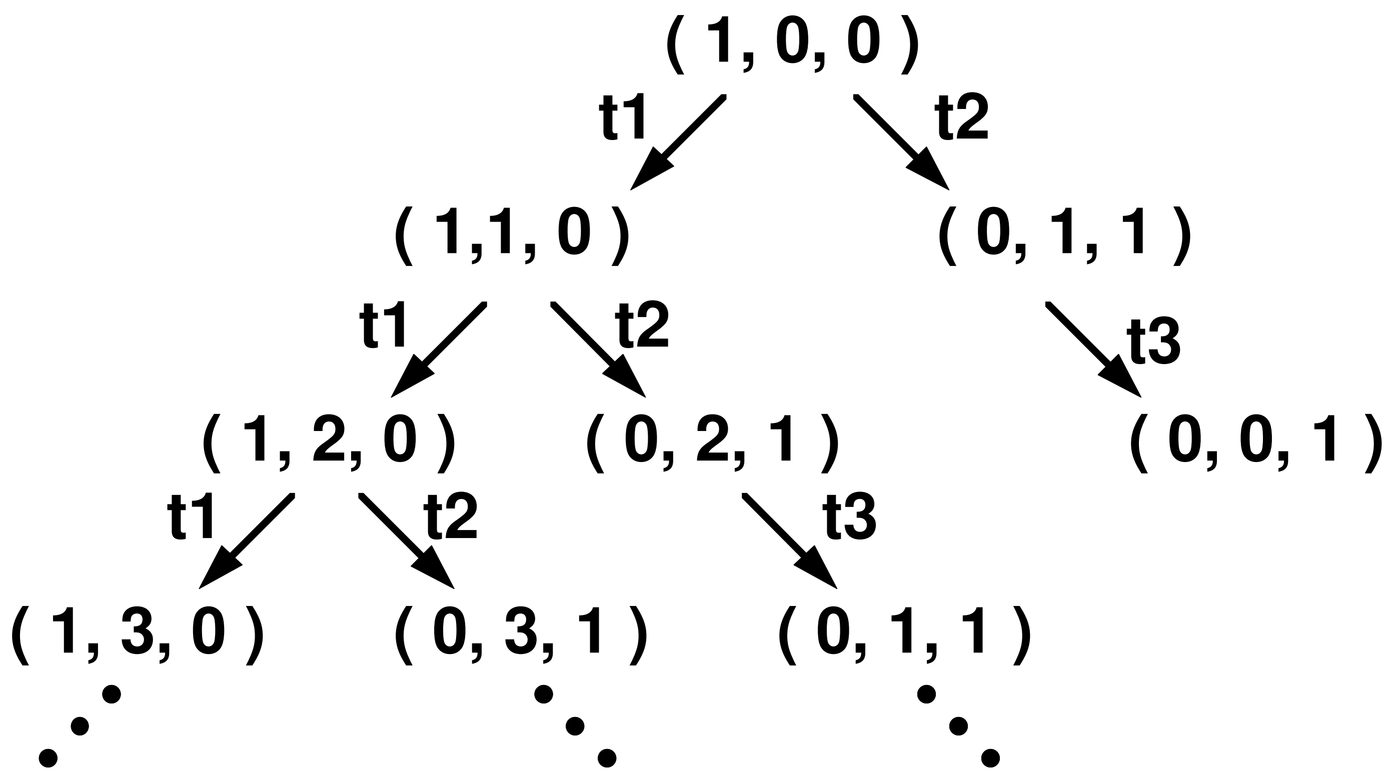
# This Lecture

- Reachability
  > Calculating finite reachability trees
- Modelling techniques and applications
  > Duration
  > Message passing
  > Token sources and sinks
  > Alternative notations
  > Readers-writers, dining philosophers, etc.
- Extensions to Petri Nets
  > Colored Petri Nets
  > Hierarchical Petri Nets

# Reachability



# Infinite Reachability Tree




**What can we conclude from this tree?**

# Computing a Finite Reachability Tree

**Start with the initial marking**

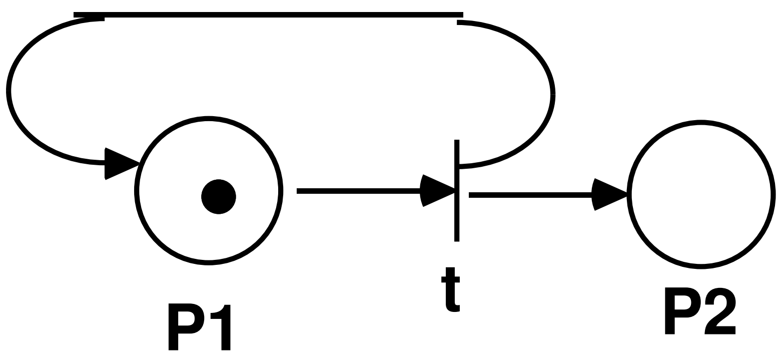
**At each stage of expansion**

- **(1) If marking repeats - make it a terminal.**
- **(2) If new marking is greater than another marking on path to root, replace larger values with $\omega$.**
  **(where ordering on markings is as described earlier)**

**Use the following rules for $\omega$**

- **$a < \omega$, for any value of a**
- $\omega \leq \omega$
- $\omega + a = \omega$
- $\omega - a = \omega$

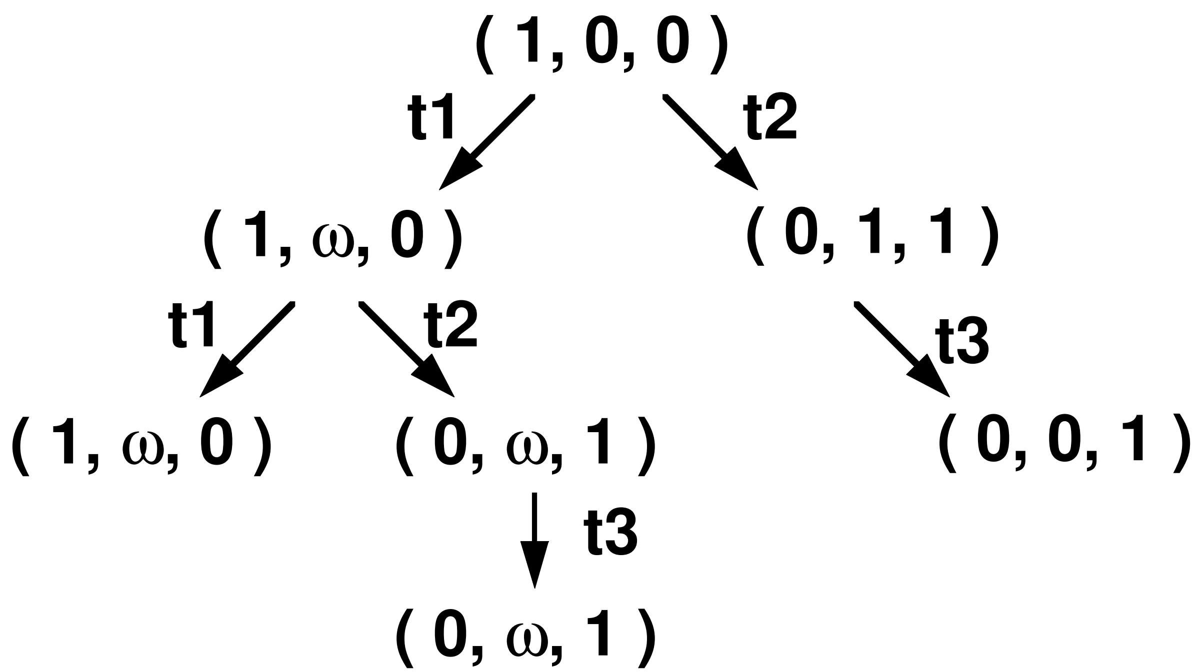
# Simple Example




$$(1,0) \xrightarrow{t} (1,1) \xrightarrow{t} (1,2) \xrightarrow{t} \cdots$$

$$(1,0) \xrightarrow{t} (1,\omega) \xrightarrow{t} (1,\omega)$$

# Finite Reachability Tree



# Uses of Reachability Trees

**1. Is a given marking reachable?**

> **Check for illegal combinations of tokens**

**2. Boundedness**

> ***Check for resource overflow***

**But some problems are undecidable:**

> **Example: is the reachability set of one marked net equal to that of another.**

> **Important in producing optimized nets - for example, by eliminating dead transitions.**

# Examples

## For the earlier example

> Is the net unbounded?

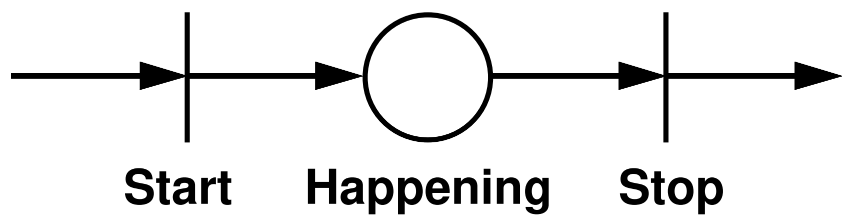
> Is it ever the case that P1 and P3 simultaneously have tokens?

> Is the sum of tokens on P1 and P2 always greater than 0?

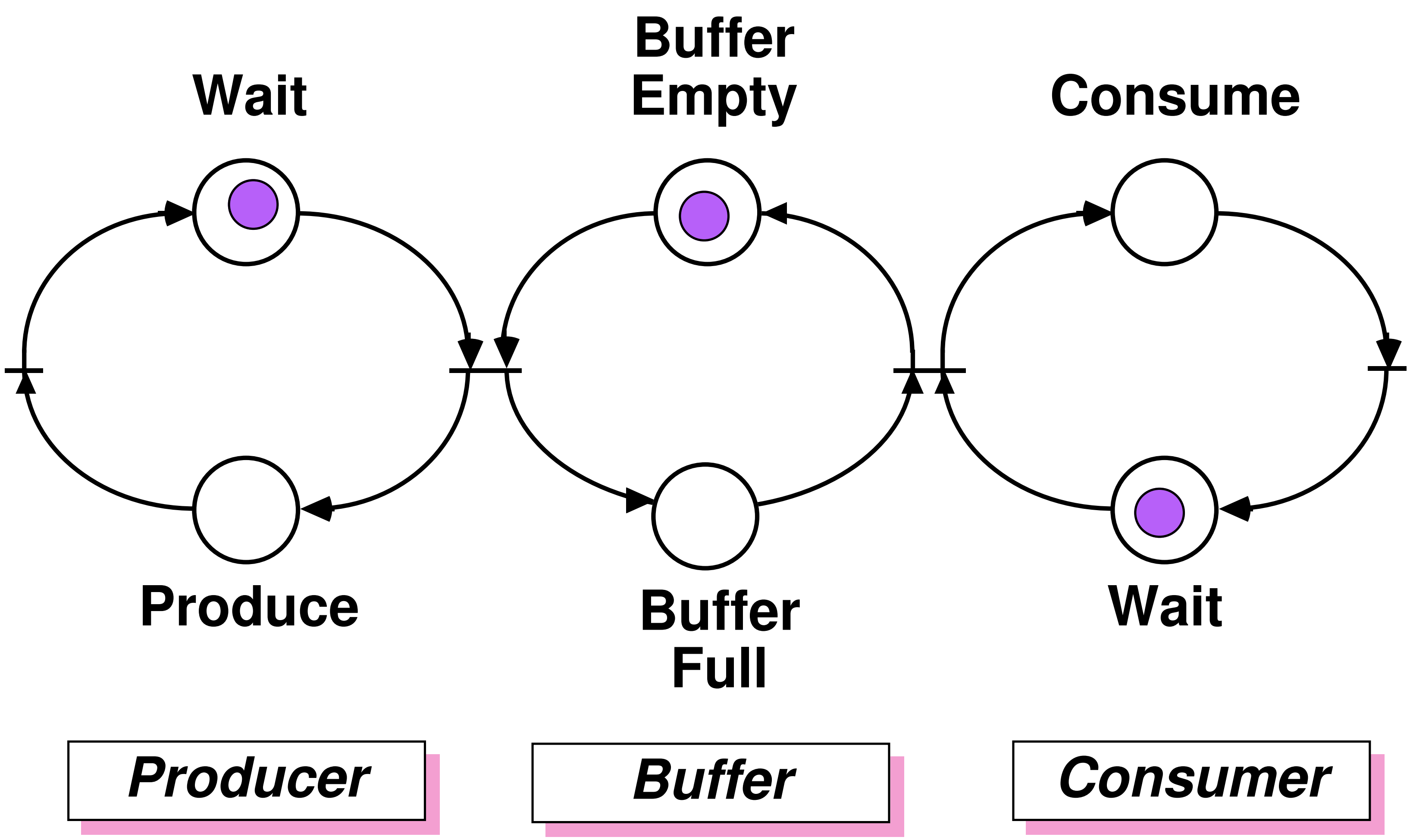
> Is the sum of tokens on P1 and P3 always greater than 0?

# Modeling Techniques: Durations

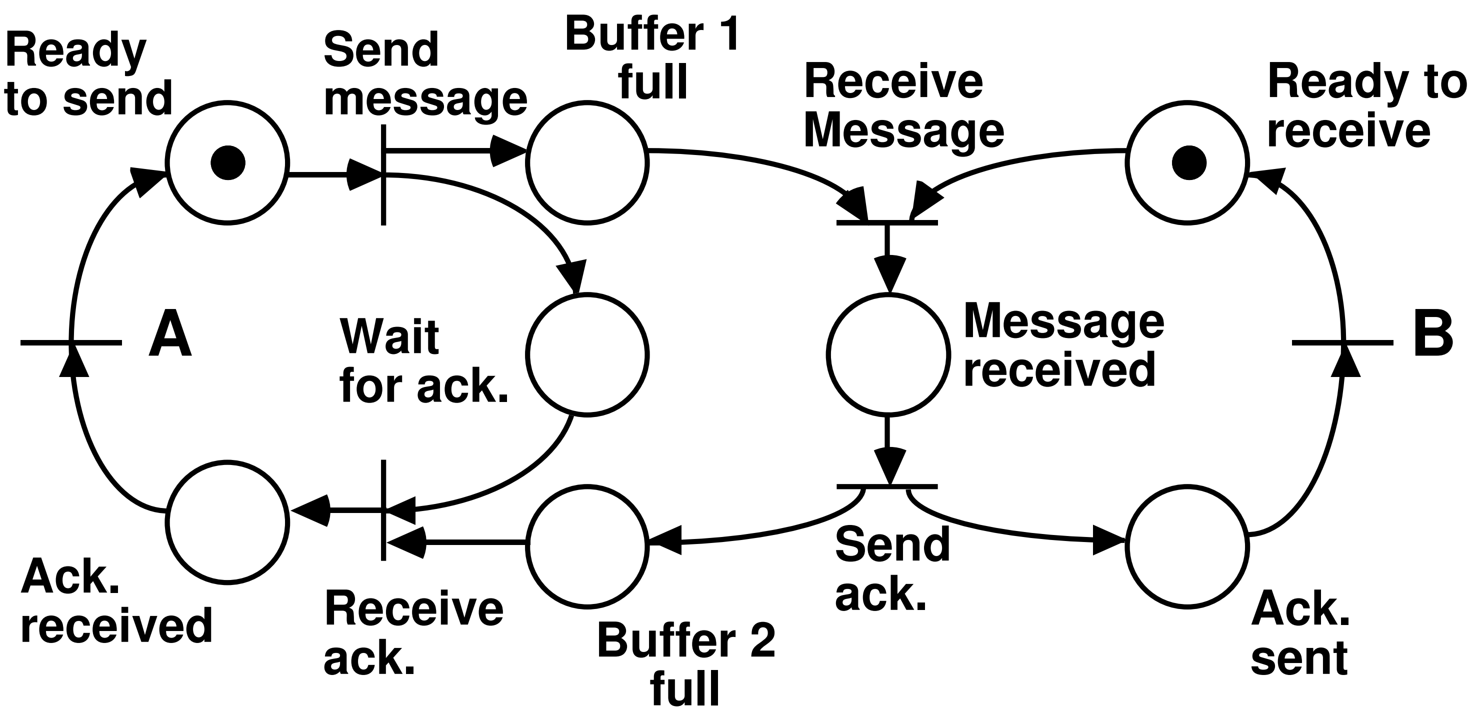
- Petri Nets can model primitive (instantaneous) and non-primitive events
  - > Use a transition to represent a primitive event.
  - > Use a place between 2 transitions to represent a non-primitive event.



# Modelling Techniques
# Producer/Consumer: Shared Buffer

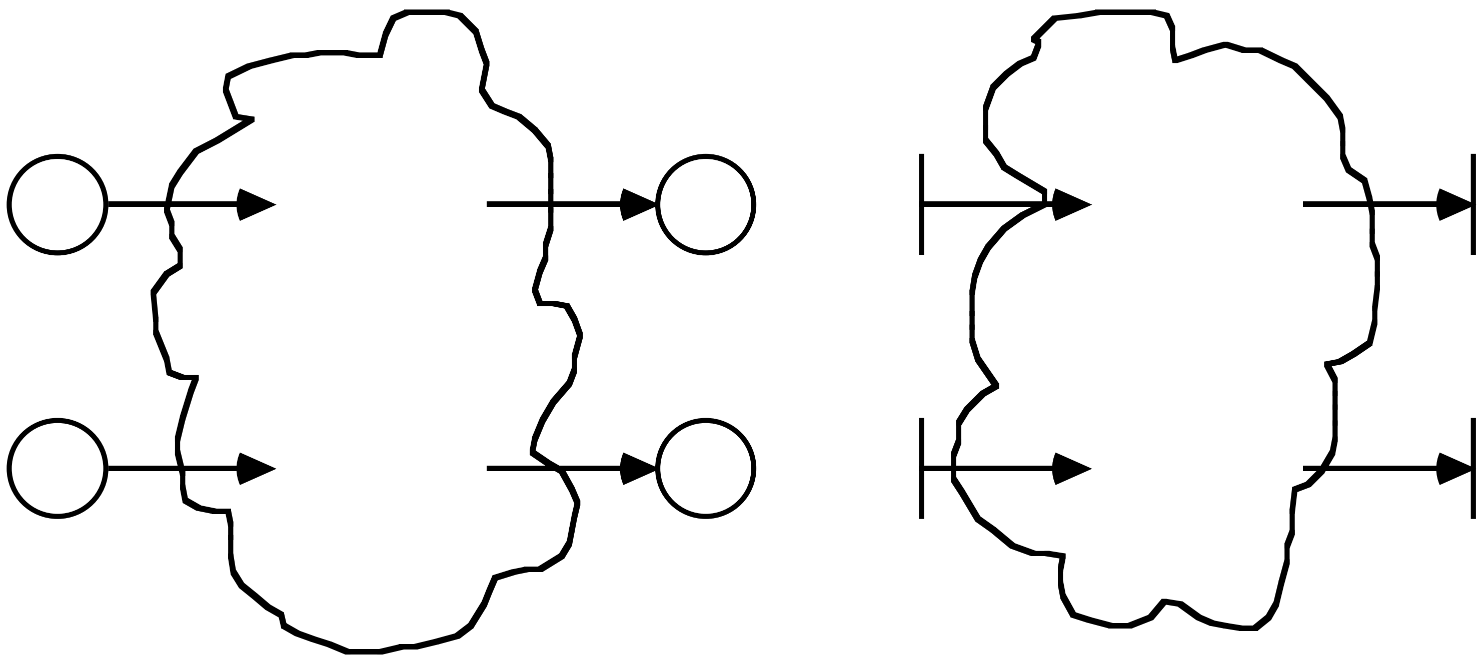


# Producer/Consumer: Synchronous Message Passing



# Modelling Techniques: Input Sources and Output Sinks

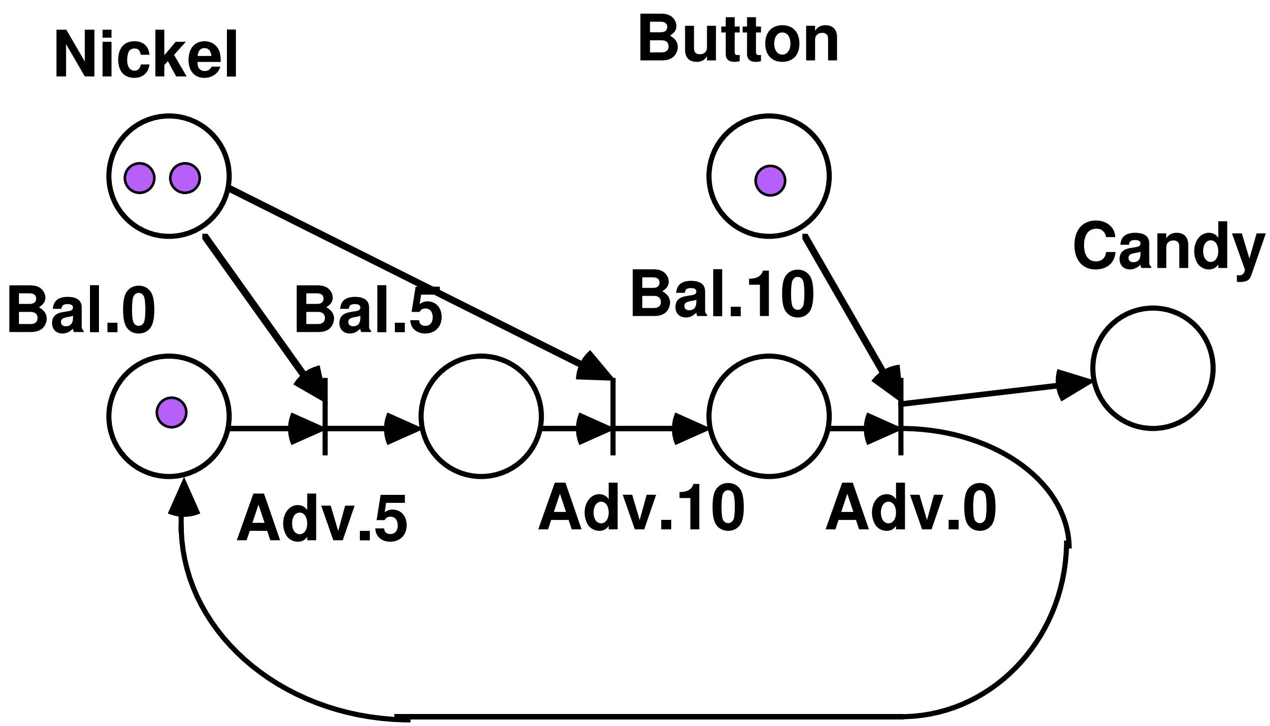
- Input/Output may be modeled by either external places or external transitions



# Modelling Techniques: Alternative Notations

- Like most graphical notations, Petri Nets can become unmanageable when problems become large.
- Alternative notations can be used.
  - Enumerate place names and transition names.
  - State the initial marking.
  - Describe transitions in terms of inputs and outputs.
  - Use subscripts to abstract away from repetition.

# Example (1)



# Example (2)

**Places:**

**{ Nickel, Button, Candy, Bal.i }**

**where i = 0, 5, 10**

**Transitions:**

**{Adv.i } where i = 0, 5, 10**

**Initial Marking:**

**{Nickel -> 2,  Button -> 1, Bal.0 -> 1}**

# Example (3)

**In(Adv.5) = { Bal.0, Nickel }**
**Out(Adv.5) = { Bal.5 }**

**In(Adv.10) = { Bal.5, Nickel }**
**Out(Adv.10) = { Bal.10 }**

**In(Adv.0) = { Bal.10, Button }**
**Out(Adv.0) = { Bal.0, Candy }**

# Example (4)

## Alternatively:

for i = 5 or 10:
In(Adv.i) = { Bal.(i-5), Nickel }
Out(Adv.i) = { Bal.i }

In(Adv.0) = { Bal.10, Button }
Out(Adv.0) = { Bal.0, Candy }

# Readers/Writers Problem

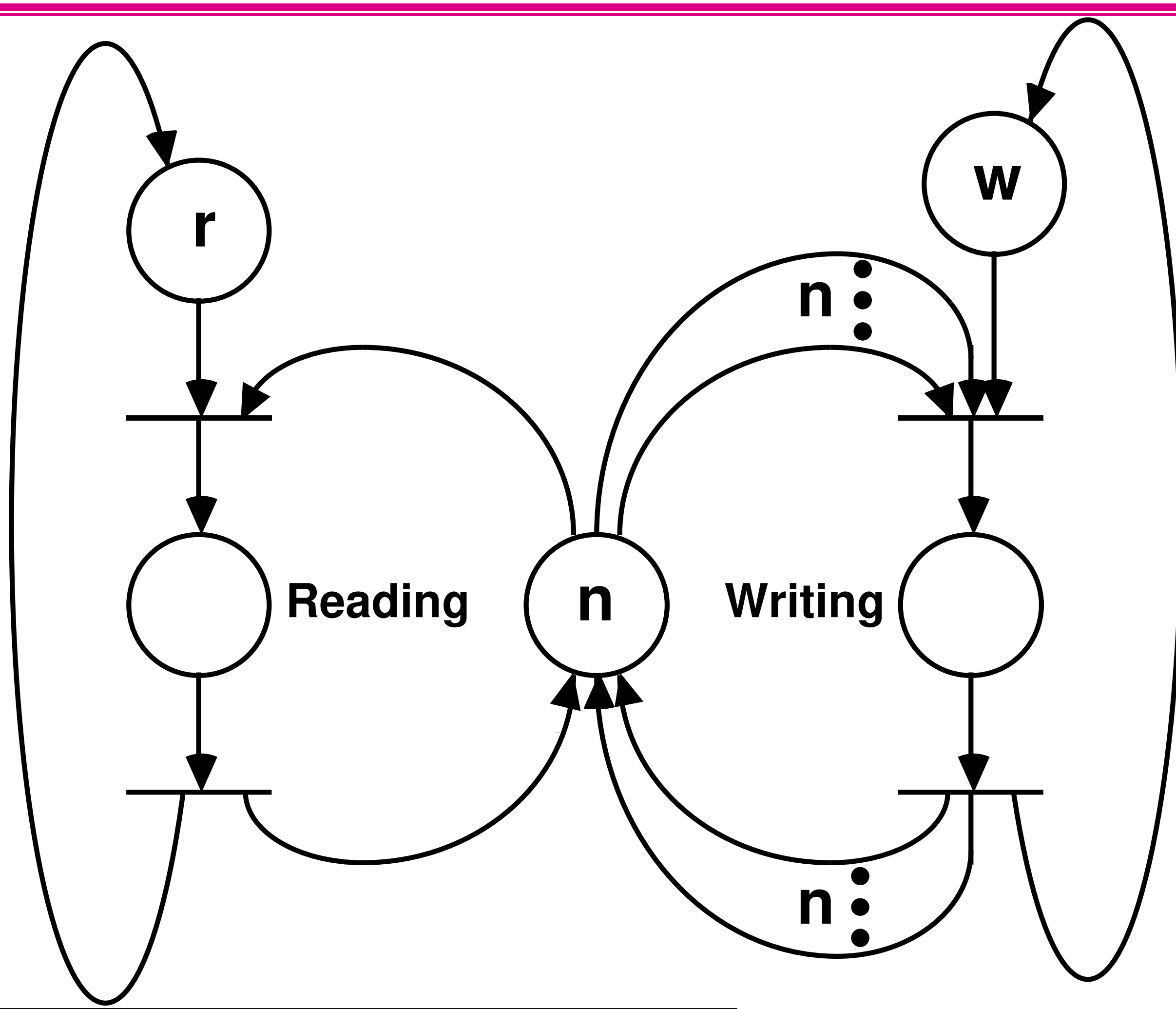
- **Problem Statement:**

  **Readers do not modify the object.**
  **Writers modify the object.**

  **Readers can share access.**
  **Writers block all others.**

- **Petri nets may be used if the readers are bounded (e.g., up to n readers at a time).**

# Readers/Writers Solution




Note non-standard use of multiple arcs

# Semaphores (1)

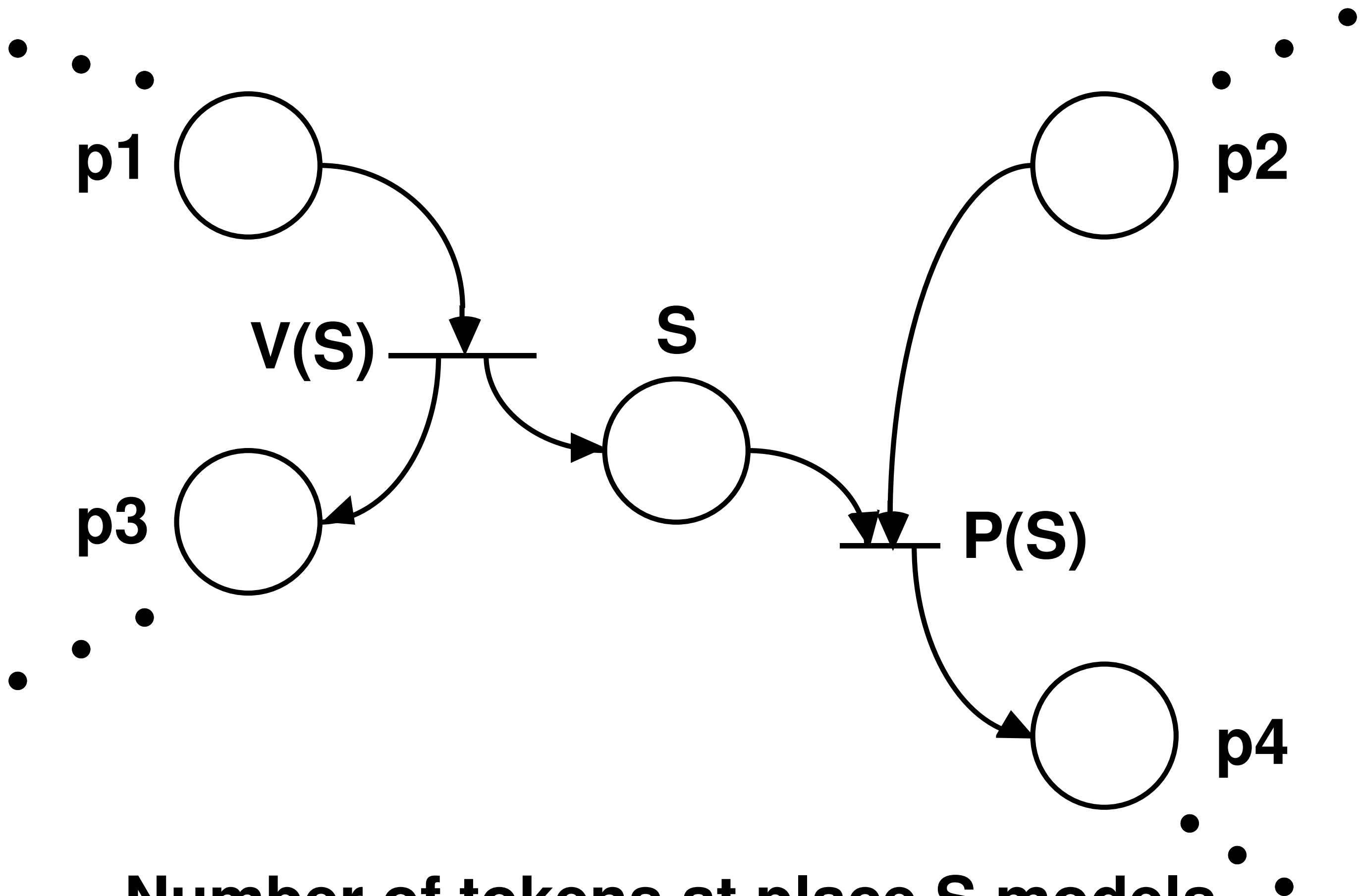
$P(S) \equiv$ when $S > 0$ then $S := S - 1$

$V(S) \equiv S := S + 1$

Each operation is atomic. (I.e., it may not be interrupted nor disturbed.)

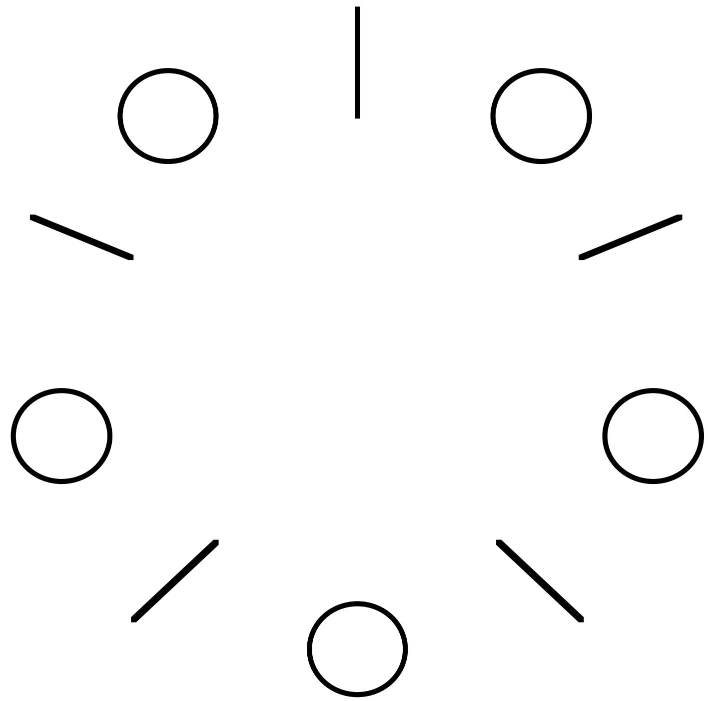
| Process 1 | Process 2 |
|---|---|
| ⋮ | ⋮ |
| P(S) | P(S) |
| (critical section) | (critical section) |
| V(S) | V(S) |
| ⋮ | ⋮ |

# Semaphores




**Number of tokens at place S models the value of the semaphore.**

# Dining Philosophers Problem

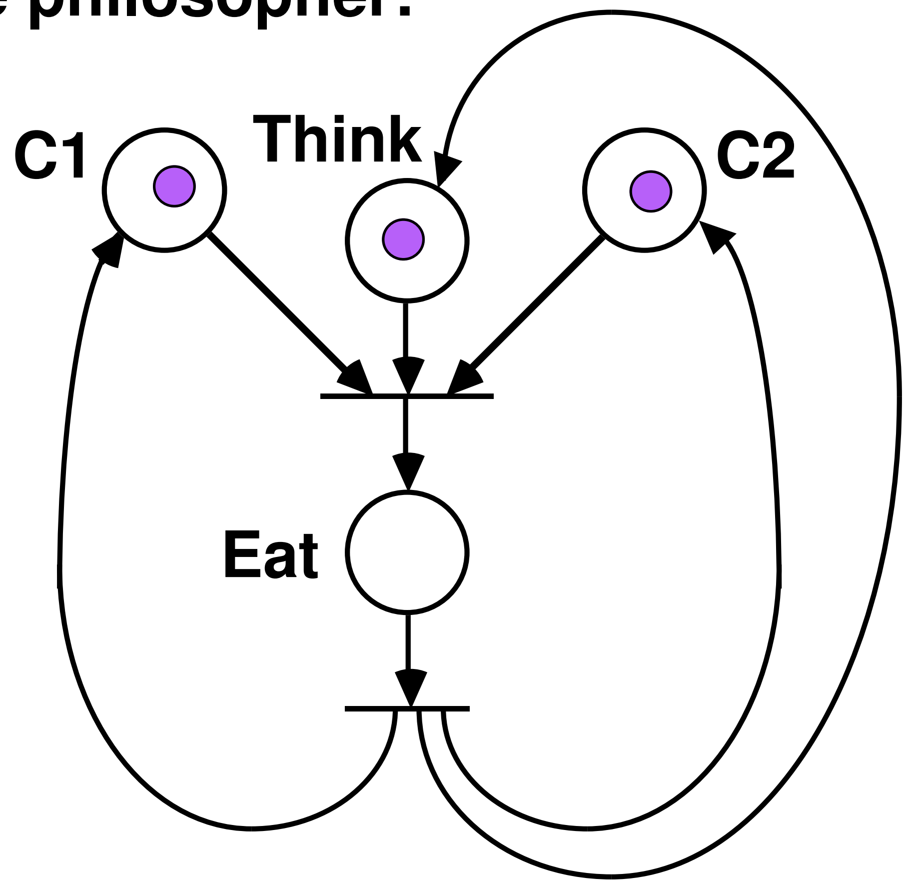


**Philosopher:** → **Think** → **Eat**

**Eat: Pick-up 2 chopsticks** → **use** → **Put-down chopsticks**

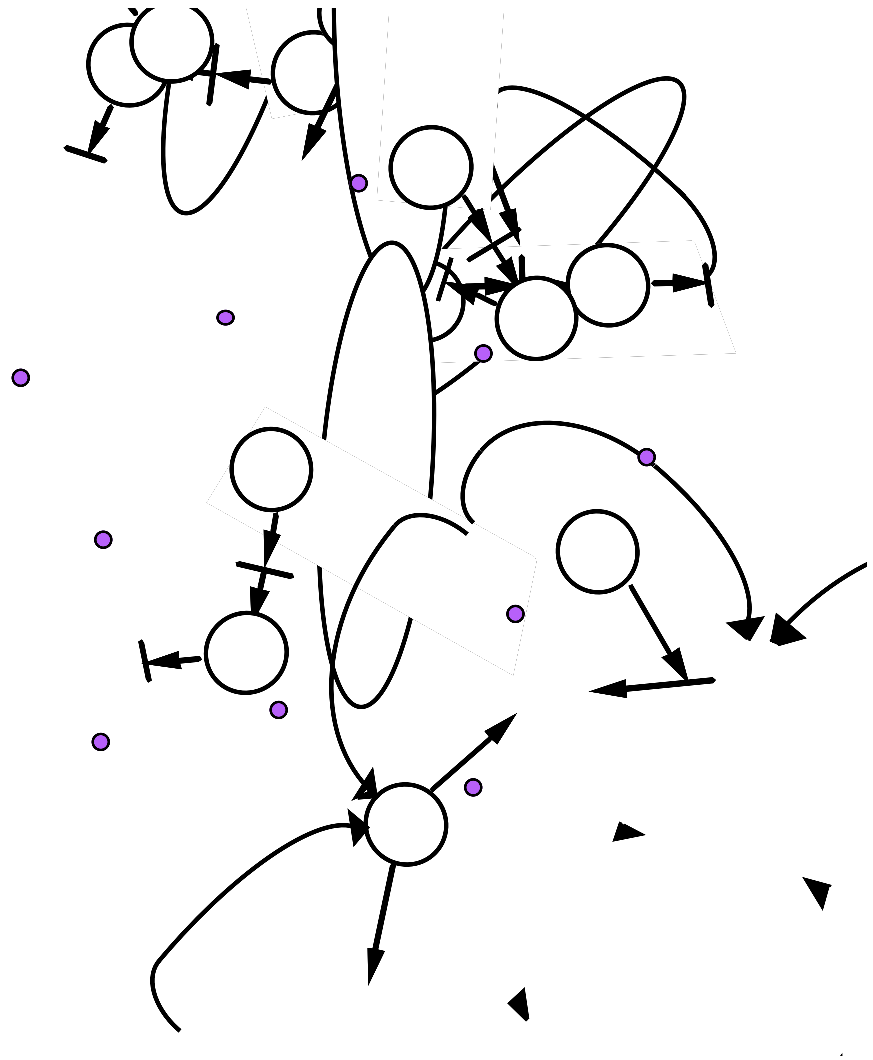
# Dining Philosophers Solution

**One philosopher:**

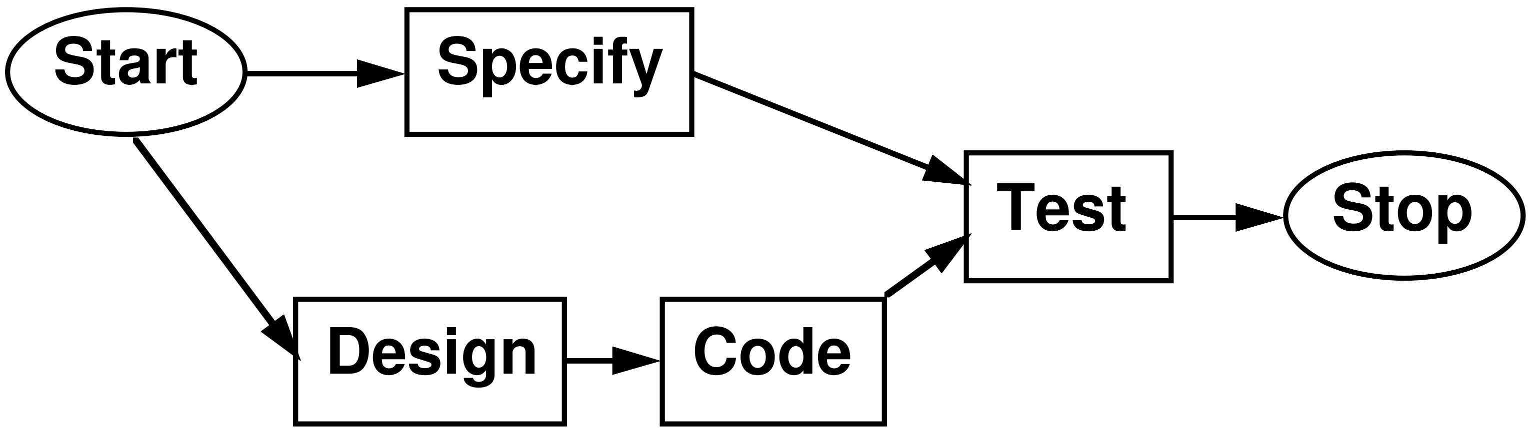


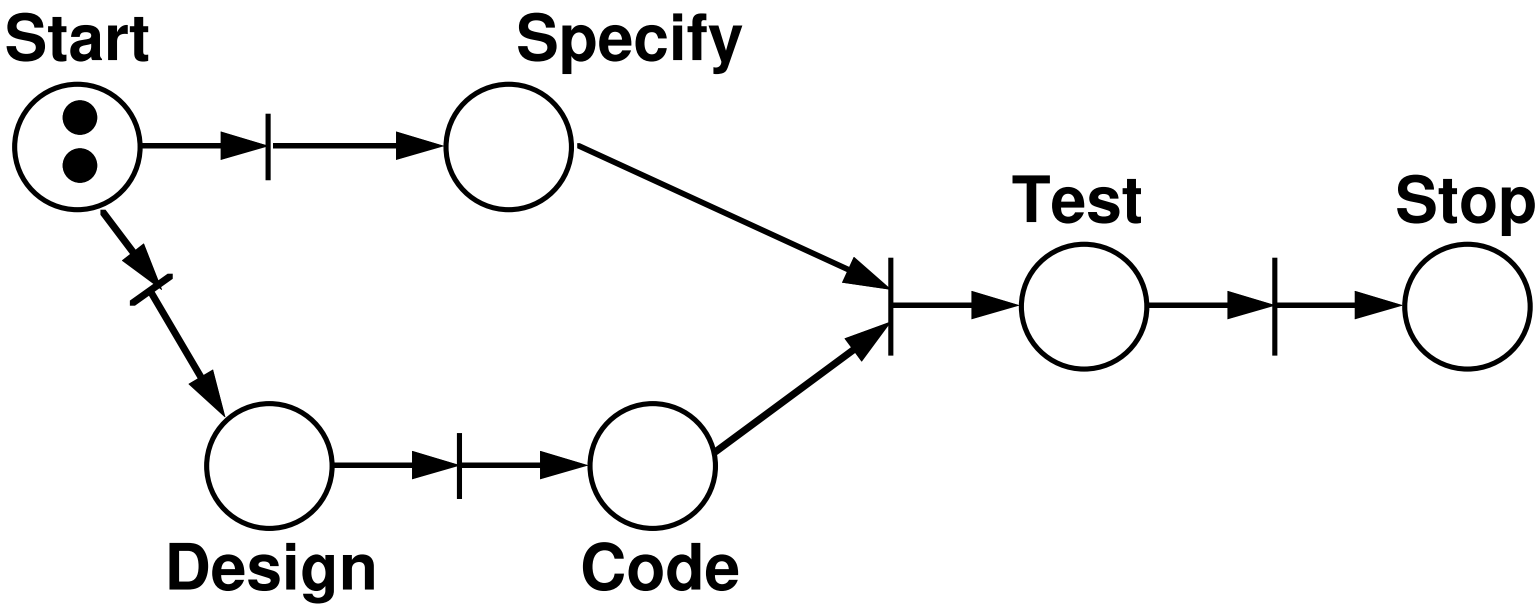
# Dining Philosophers (contd)

**This page did not print properly in 1995 using laserwriter 8 but did with the old laserwriter software**

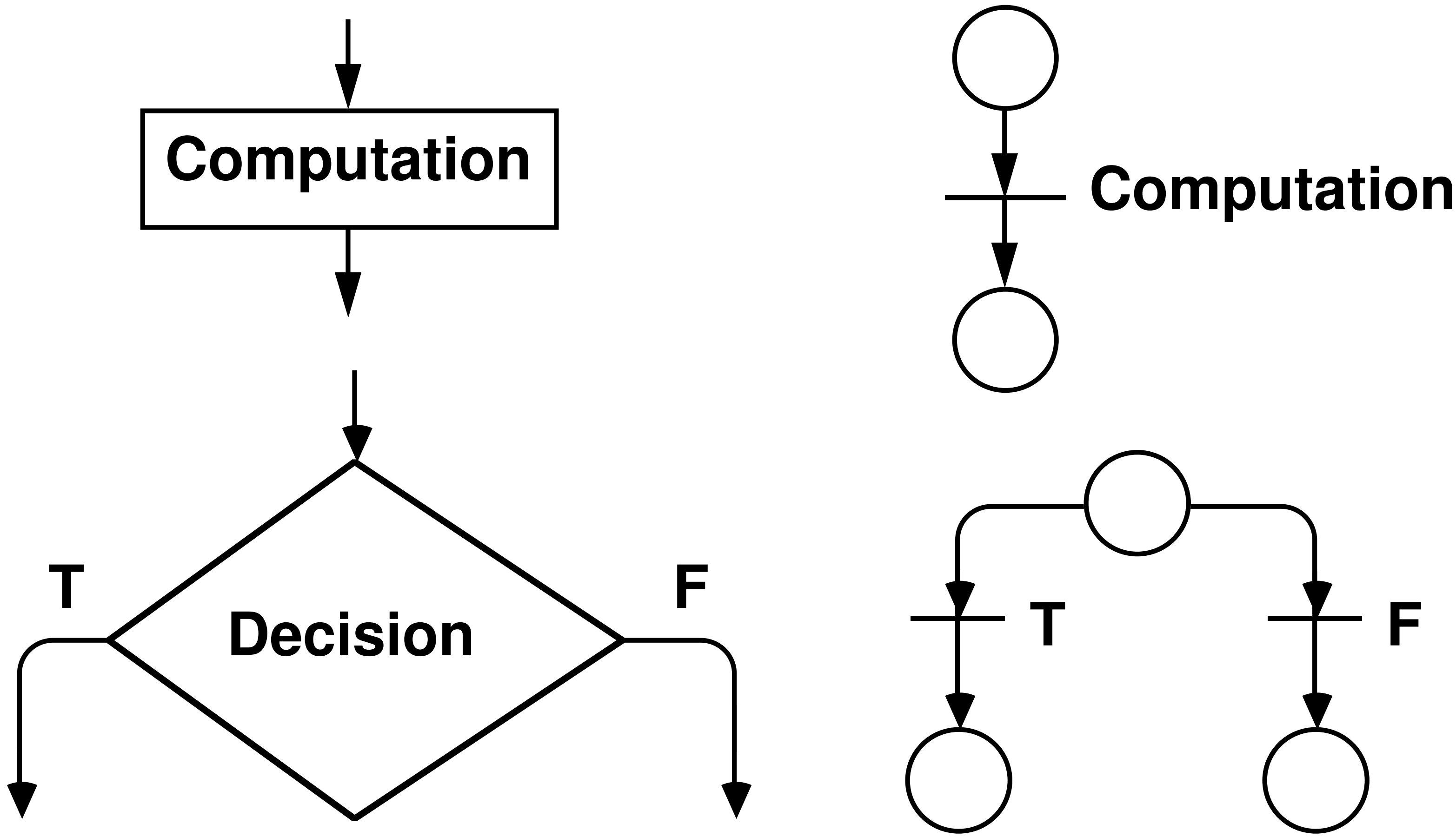


# Applications: PERT Charts (1)



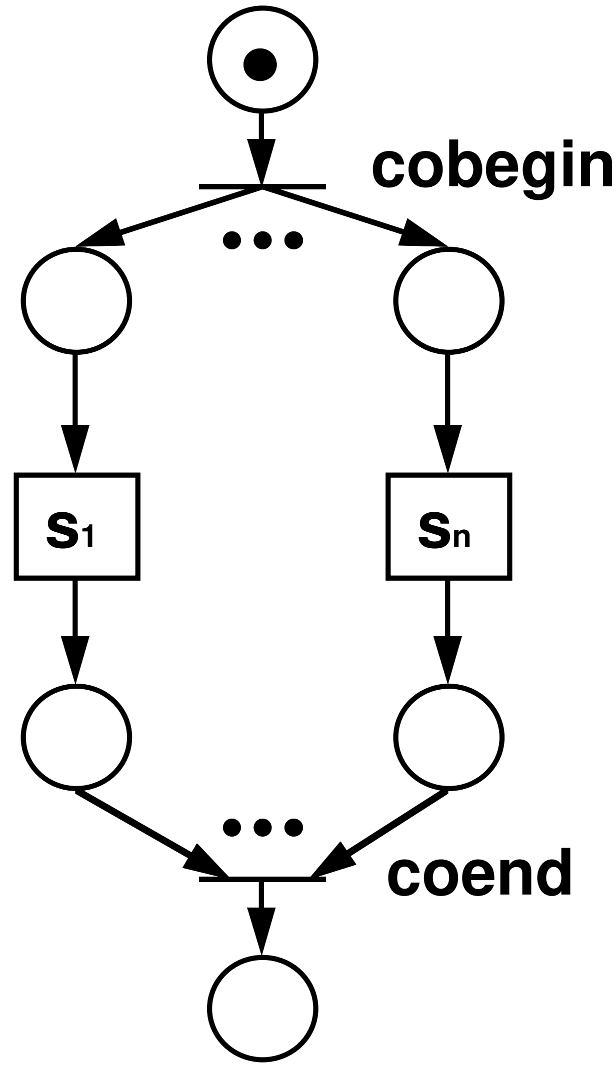

Start
Specify
Test
Stop
Design
Code

# Flowcharts



# Concurrent Programming

**cobegin**
  **s1 ;**
  **...**
  **sn**
**coend**

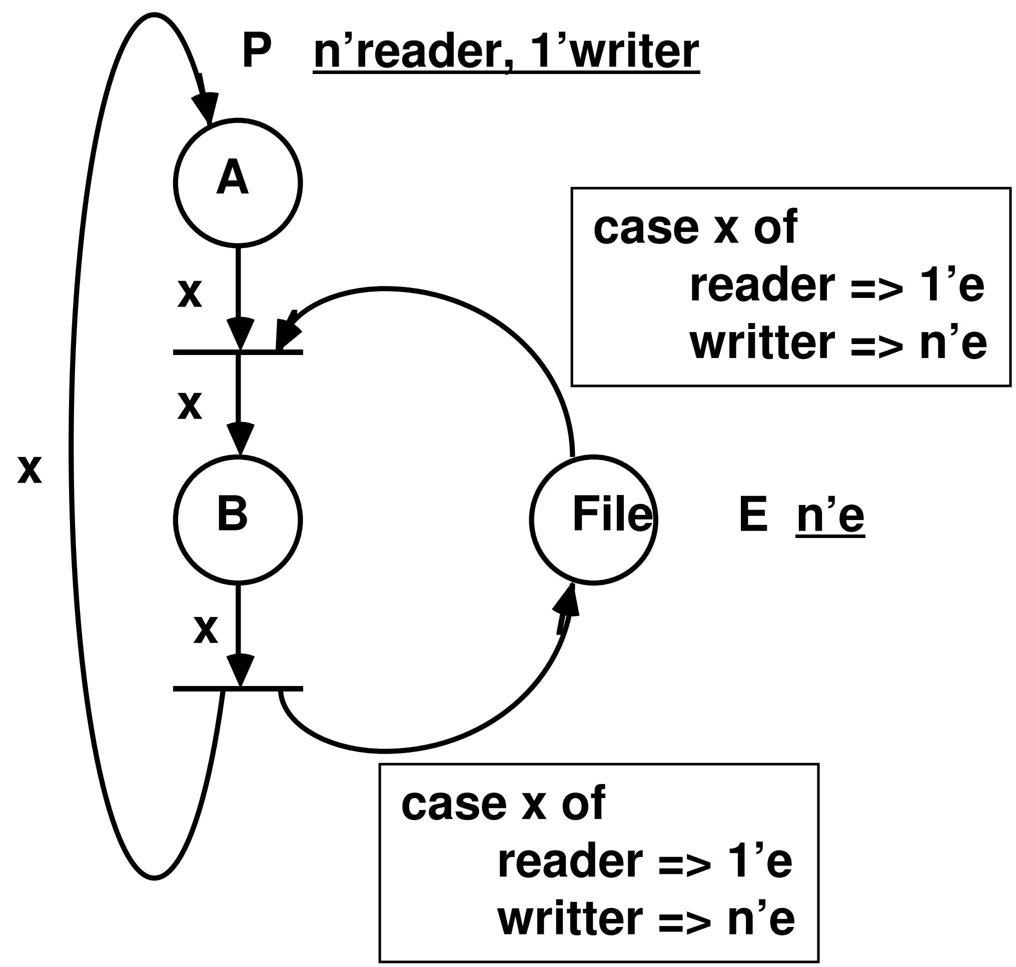


# Extensions: Colored Petri Nets

- Tokens represent data values
  > integers, records, strings, unions, etc.
  > each data value has a type -- the token "color"
- Places are labelled with
  > the type of token they can hold
  > an (optional) initialization expression
- Transitions represent state changes
  > they may be guarded with a predicate that must be satisfied if the transition is to take place
- Arcs are labelled with expressions that tell how many (if any) tokens to transfer

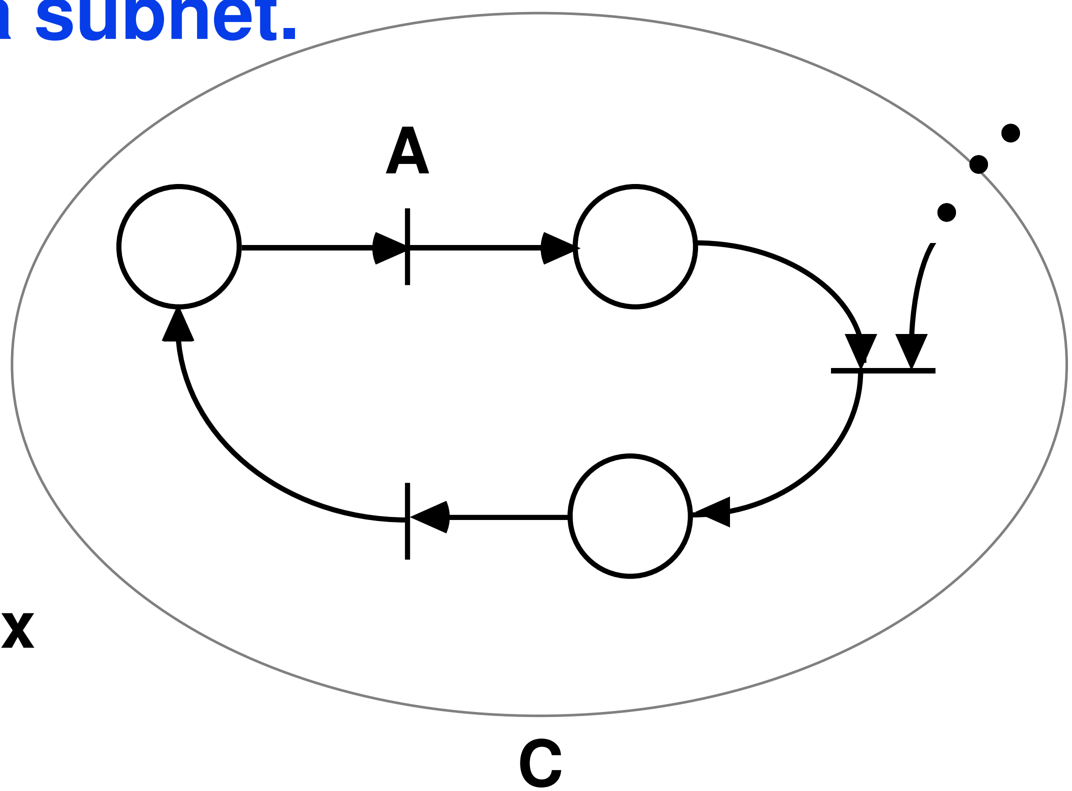
# Example

color P = with reader | writer
color E = with e;
var x: P



# Extensions: Hierarchical Nets

- When structure of a complex transition is to be represented you can model with it with a subnet.




**A - atomic**
**C - complex**

# Hierarchical Nets (continued)

## Hierarchical Petri Nets: