

---

# Parallel Programming: Challenges and Possible Solutions

**Marian V. Iordache**

LeTourneau University

February 20, 2009

## Background

---

- Parallel programming can be considerably more difficult than sequential programming.
- Tools for converting sequential programs to parallel programs are available.
- Often programs are not nearly as efficient as when written from the beginning as parallel programs.

# Parallelism

---

- Data parallelism

`a[0] = b[0] || a[1] = b[1] || ... || a[n] = b[n]`

- Task parallelism

`monitor_temp() || control_speed() || user_interface()`

## Parallel Programming

---

- Determine what operations can be executed in parallel.
- Load distribution.
- Ensure memory consistency.
- Concurrent execution of tasks raises more issues.

# Concurrent Programming

---

“Concurrent programming is notoriously difficult, even for experts.”

In “Design Considerations for Parallel Programming” by D. Callahan, MSDN Magazine, Oct. 2008

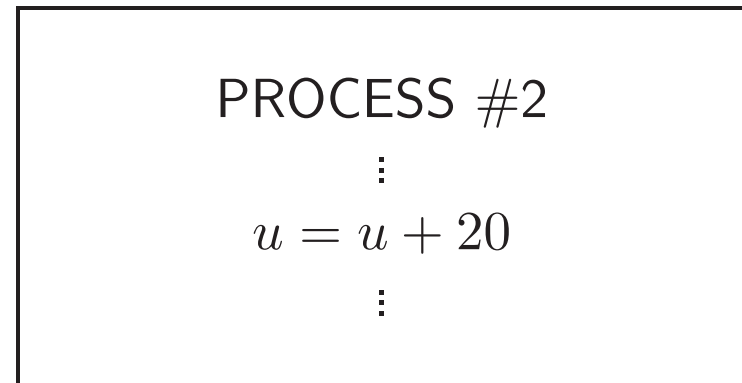
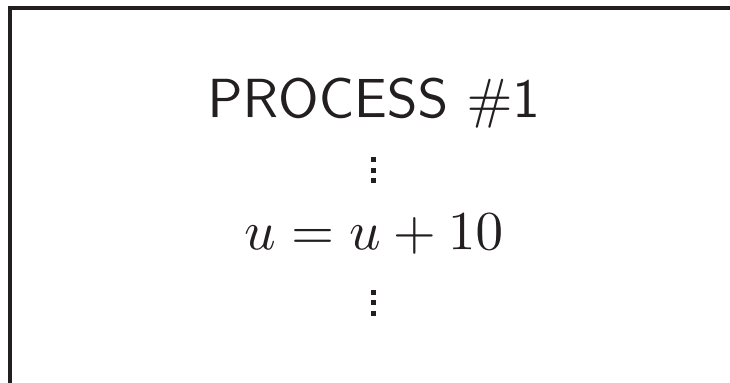
- Difficulties because of shared resources and variables.
- These can cause race conditions.
- “Locks” used to address race conditions.
- Locking may result in deadlock or livelock!

## Race Conditions

---

When the result depends on the order of execution of the processes.

- Initial value of shared variable  $u$ : 30.
- This is how  $u$  is modified.



- Correct result:  $u = 60$ .
- Possible results:  $u = 40$ ,  $u = 50$ , and  $u = 60$ .

In our example the instructions are not atomic.

Solution: use locks.

Semaphores: a type of locks.

## Semaphore Description

---

A **semaphore**  $x$  has the following operations:

- $wait(x)$

**if**  $x > 0$  **then**  $x = x - 1$   
**else if**  $x = 0$  **then** suspend process

- $signal(x)$

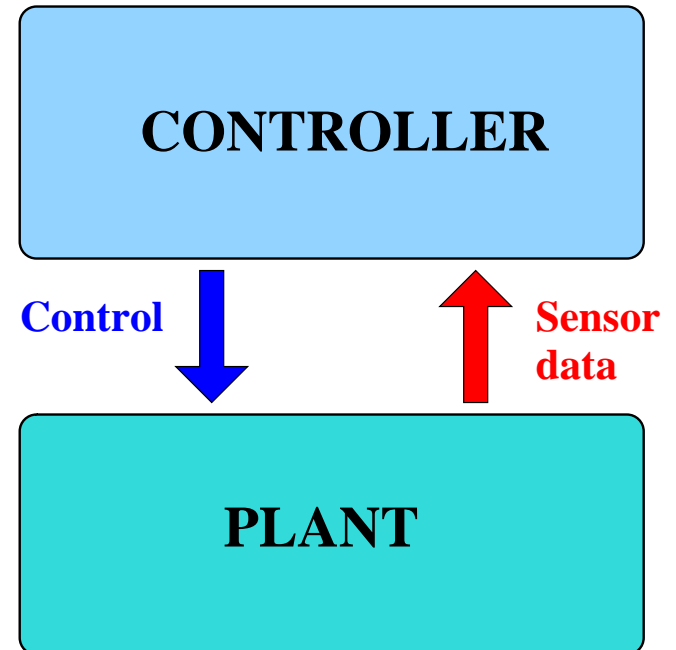
**if** there are suspended processes **then** activate one process  
**else**  $x = x + 1$

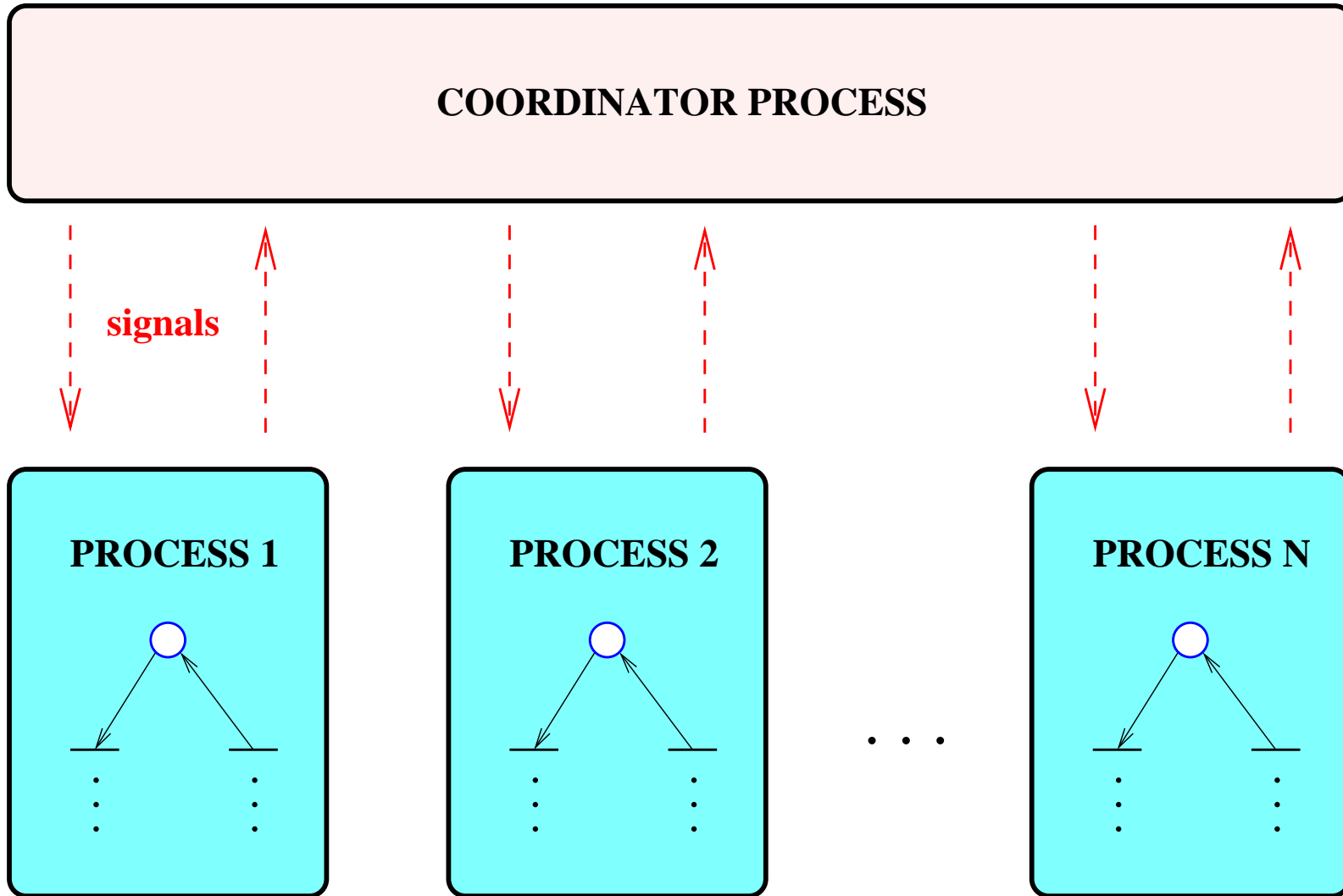
The use of locks may lead to deadlock.

Some of the parallel code executed sequentially (not in parallel).

# Control Systems

---





- Supervisory Control: Control Systems methods for discrete event systems.
- Control Systems methods require a plant model.
  1. Finite state machines.
  2. Petri nets.
  3. . . .

Petri nets model concurrent systems.

Petri net models have been used in: *distributed algorithms, flexible manufacturing, program specification, communication protocols*, and others.

An **ordinary Petri net structure** is a tuple  $\mathcal{N} = (P, T, F)$  where:

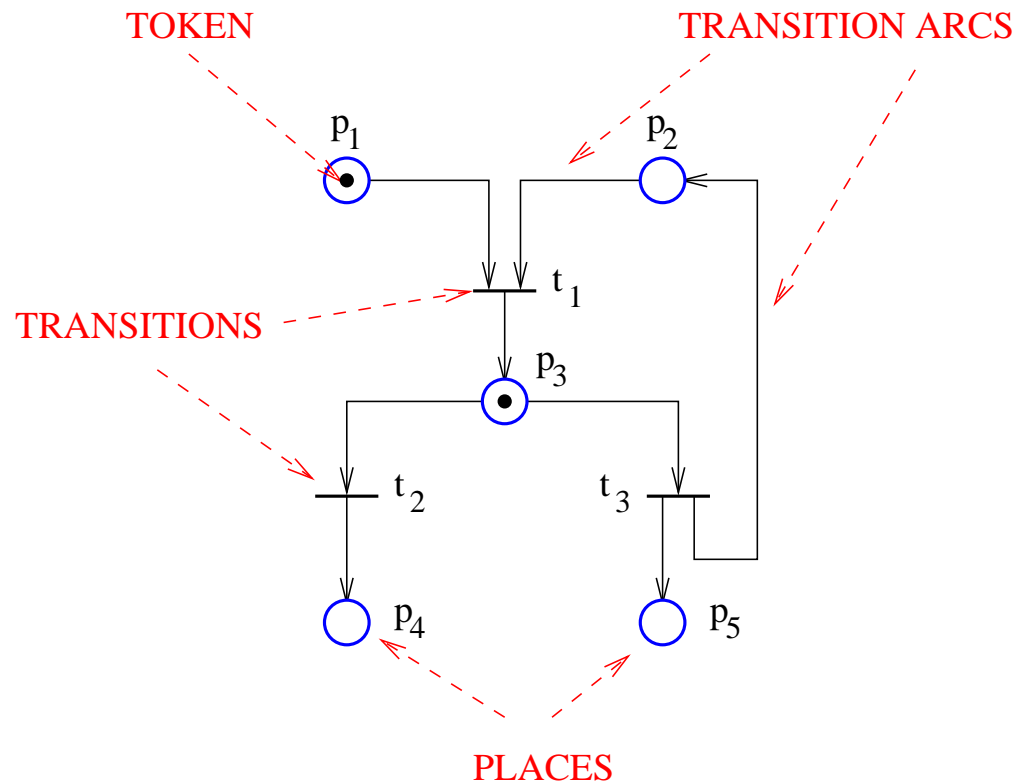
$P$  - is the **set of places**

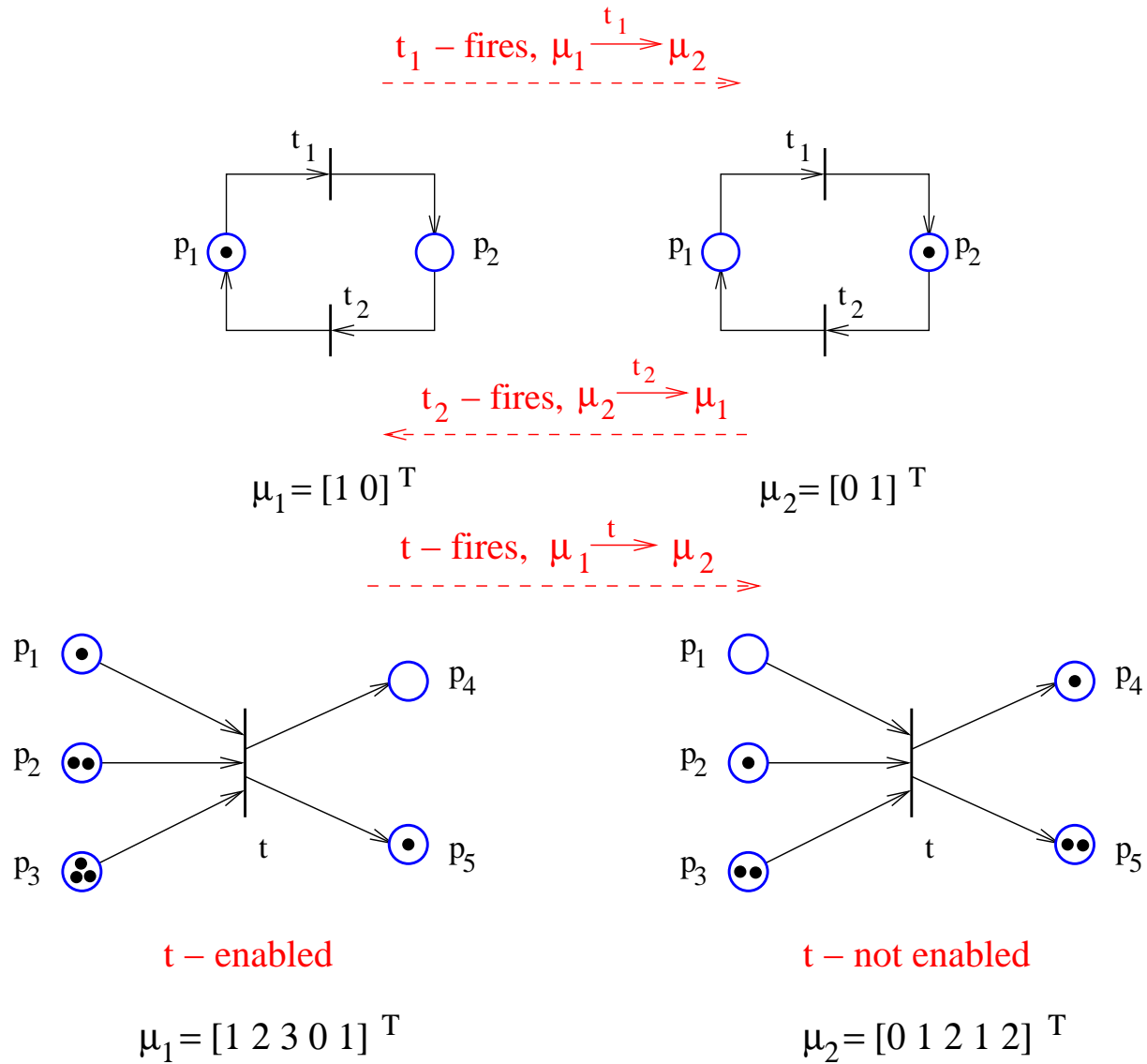
$T$  - is the **set of transitions**

$F$  - is the **set of transition arcs**

A **marking**  $\mu$  is a vector containing the number of tokens for each place.

A Petri net structure  $\mathcal{N}$  of **initial marking**  $\mu_0$  is denoted as  $(\mathcal{N}, \mu_0)$ .

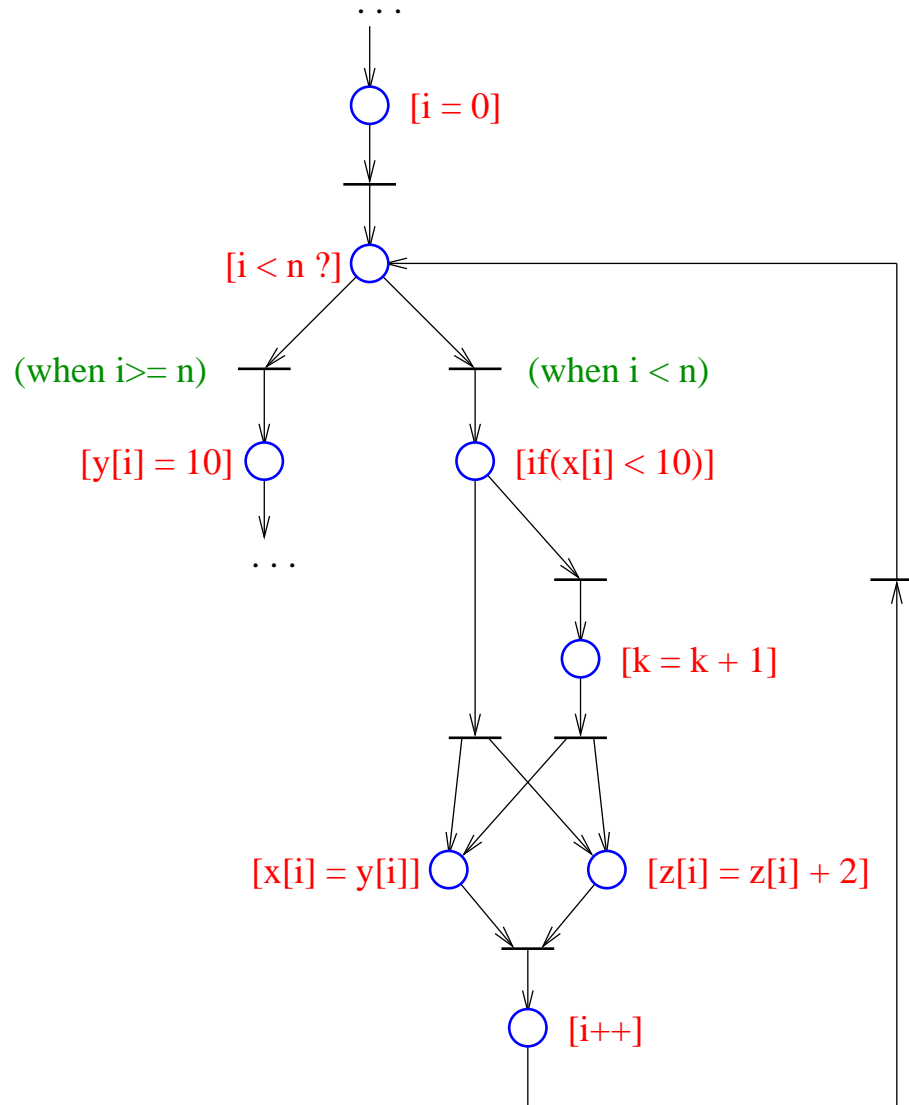


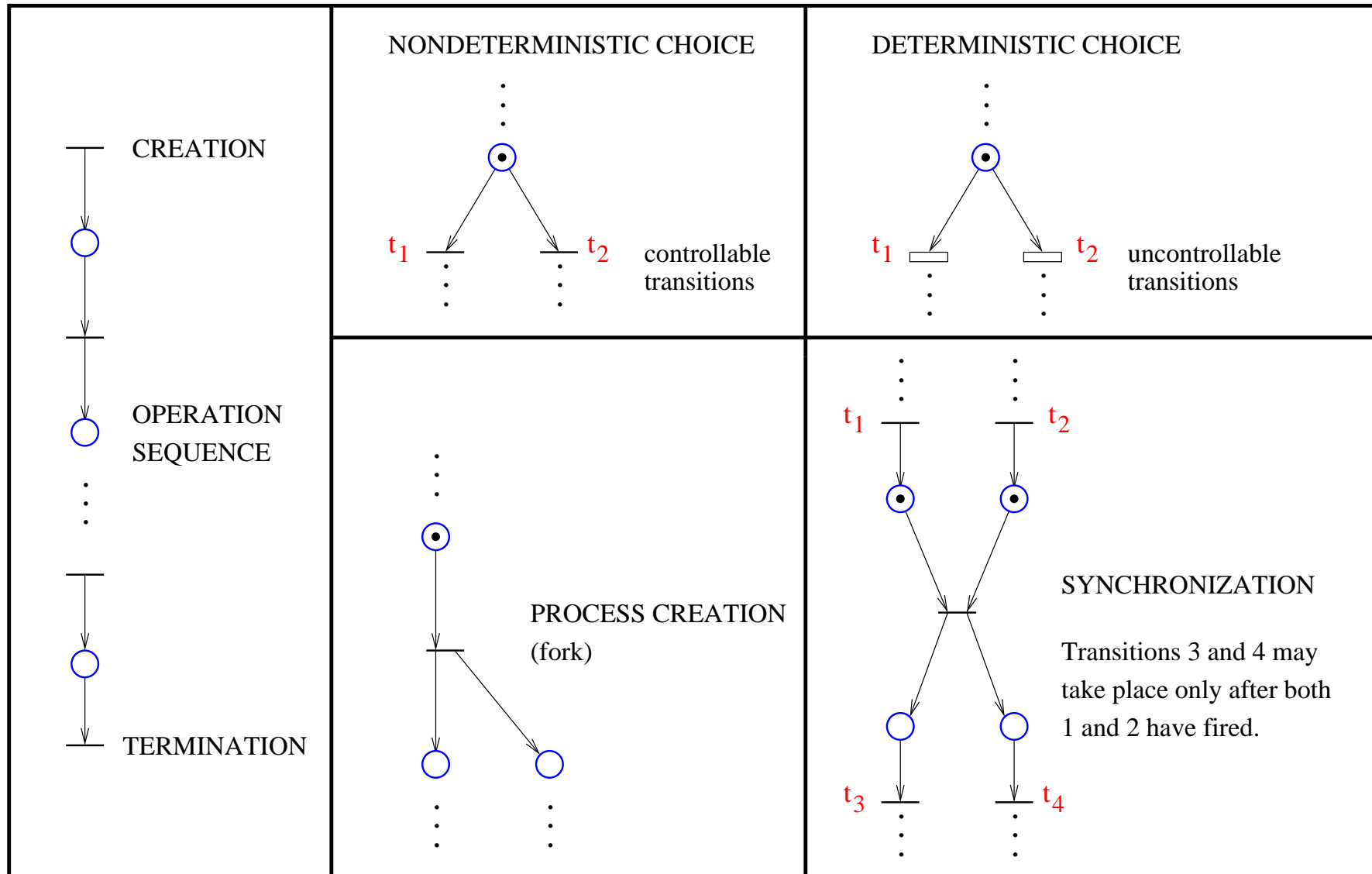


```

/* the code */
...
for(i = 0; i < n; i++) {
  if(x[i] < 10)
    k = k + 1;
  x[i] = y[i];
  z[i] = z[i] + 2;
}
y[0] = 10;
...

```





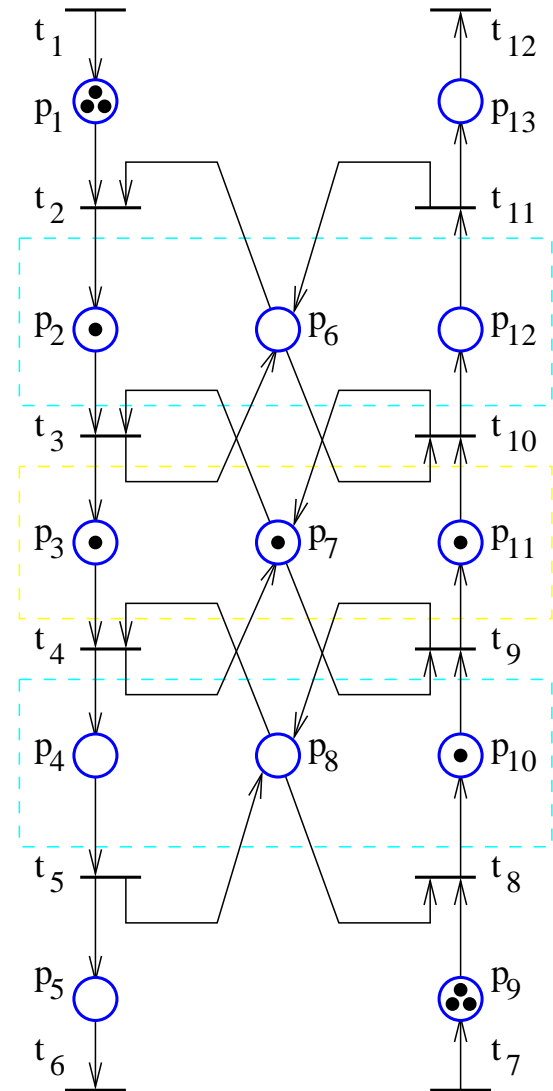
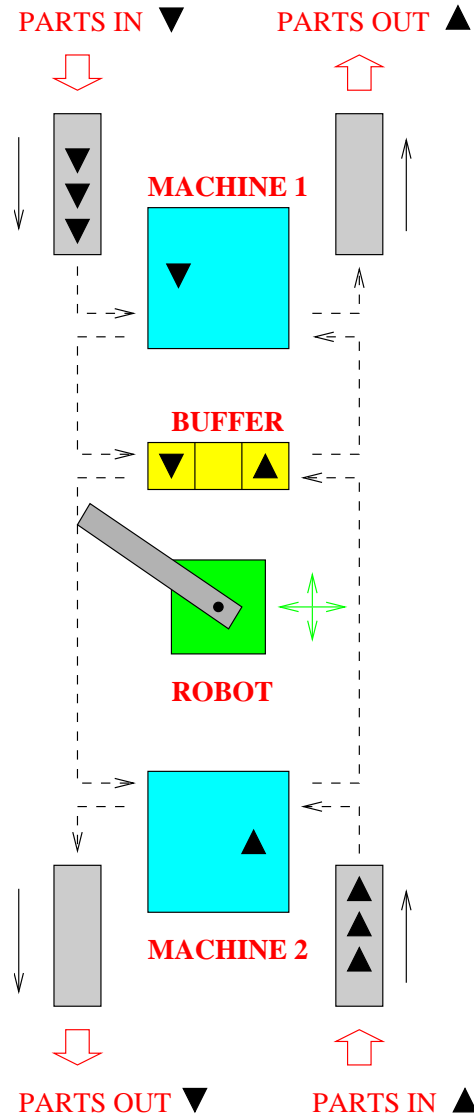
# Deadlock Prevention

# Manufacturing Cell Example

Each machine can only process one part at a time.

The robot moves the parts.

The buffer is the only storage space available to the robot.





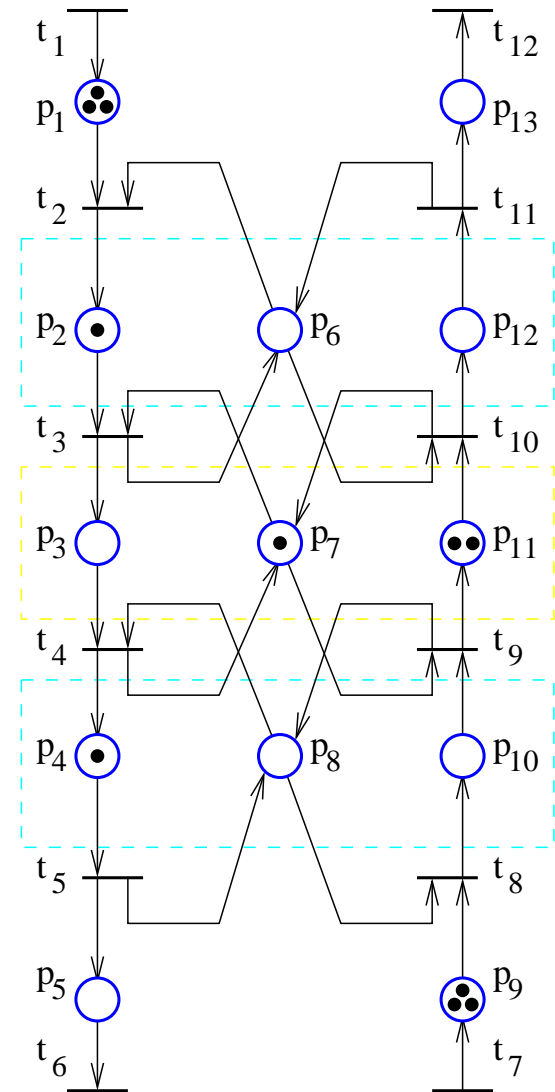
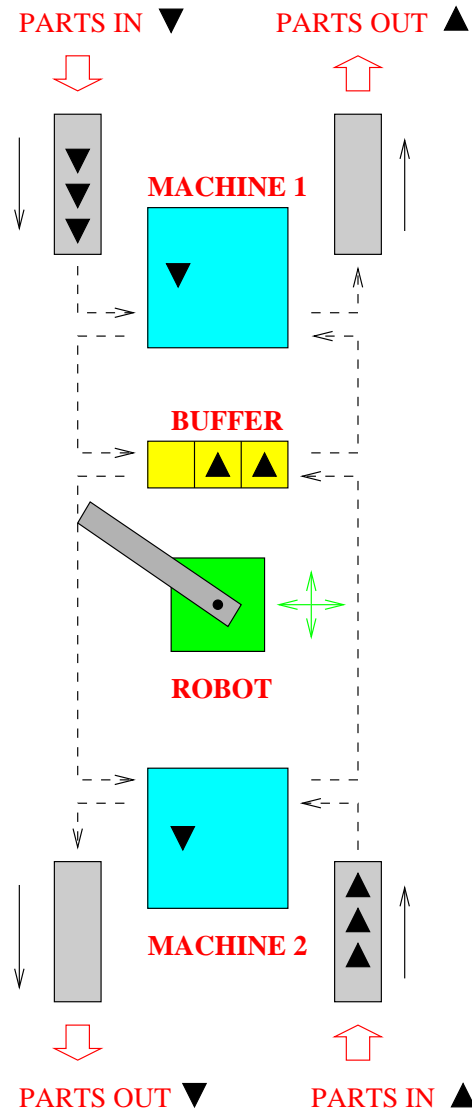
# Deadlock Prevention

# Manufacturing Cell Example

Each machine can only process one part at a time.

The robot moves the parts.

The buffer is the only storage space available to the robot.





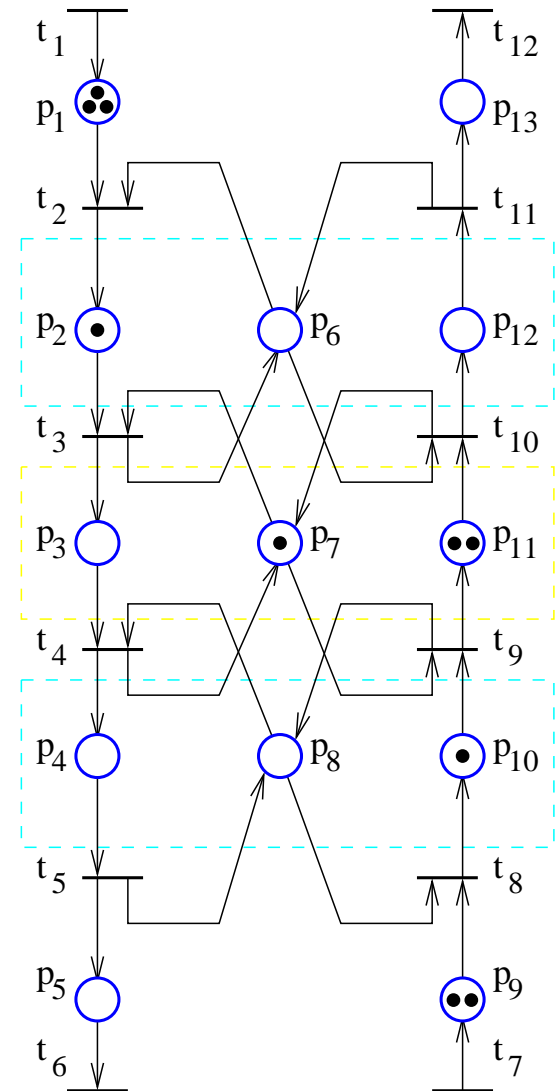
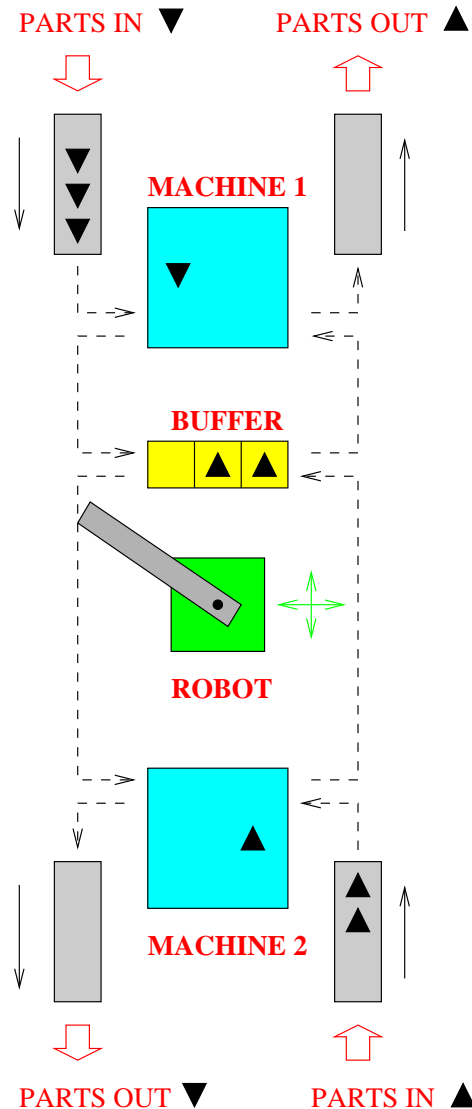
# Deadlock Prevention

# Manufacturing Cell Example

Each machine can only process one part at a time.

The robot moves the parts.

The buffer is the only storage space available to the robot.



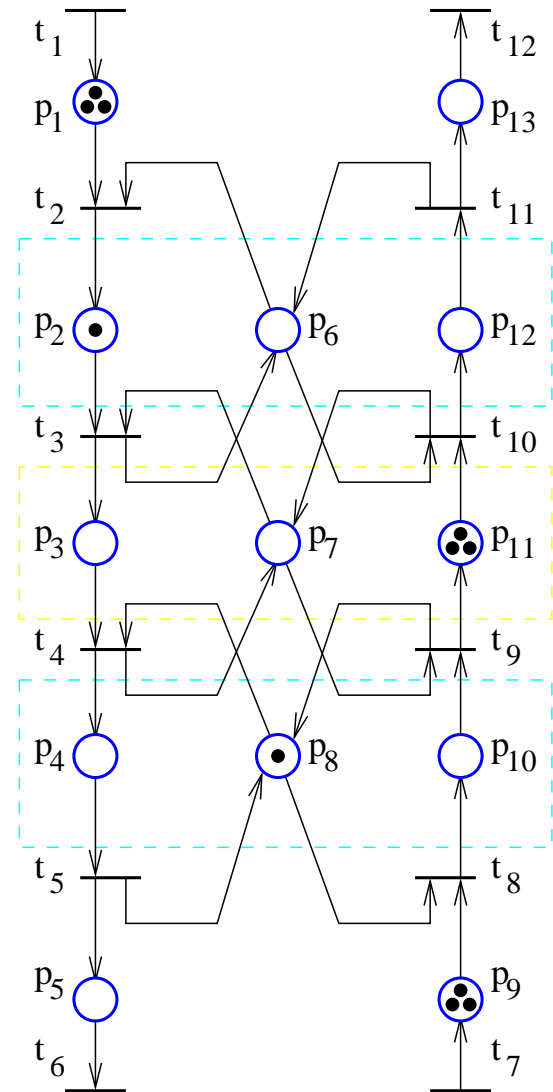
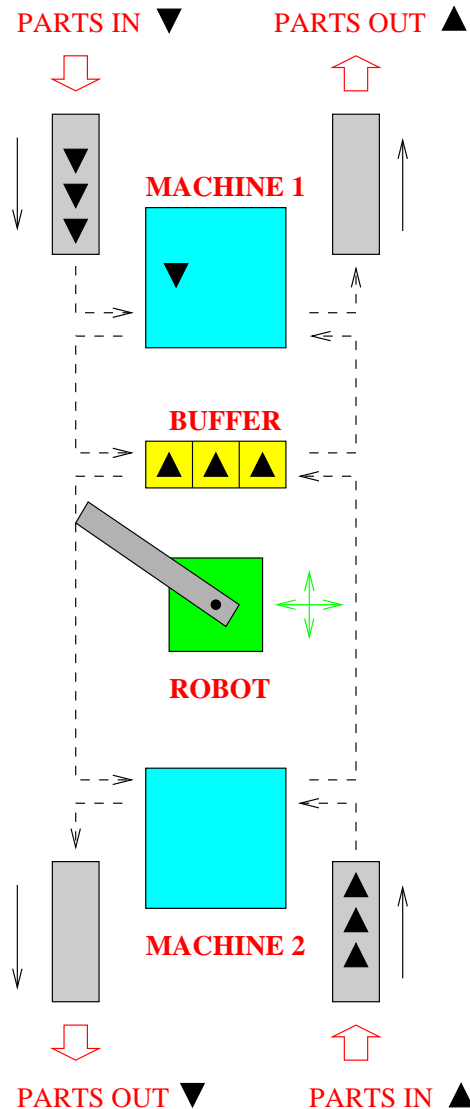
# Deadlock Prevention

# Manufacturing Cell Example

Each machine can only process one part at a time.

The robot moves the parts.

The buffer is the only storage space available to the robot.



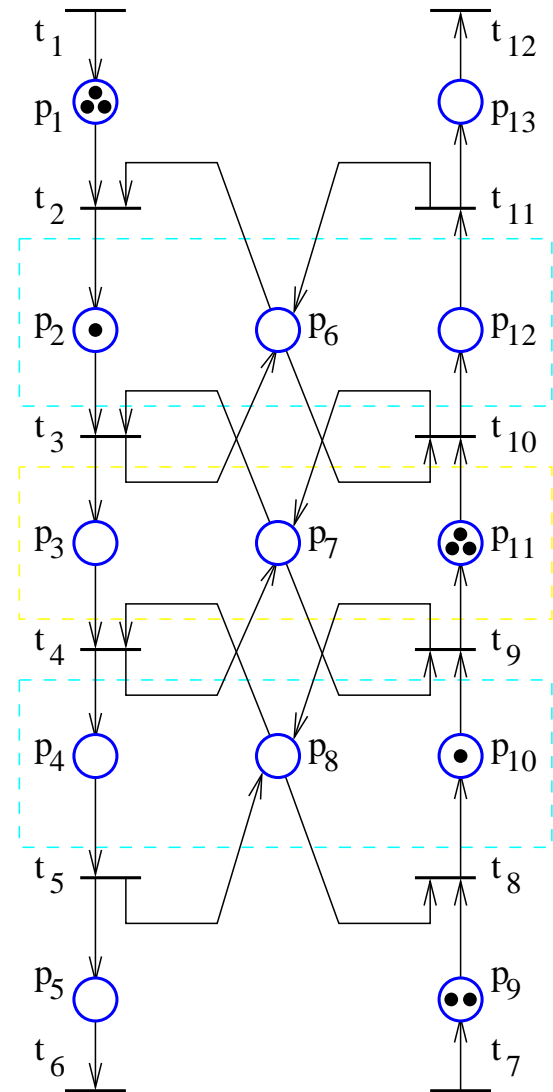
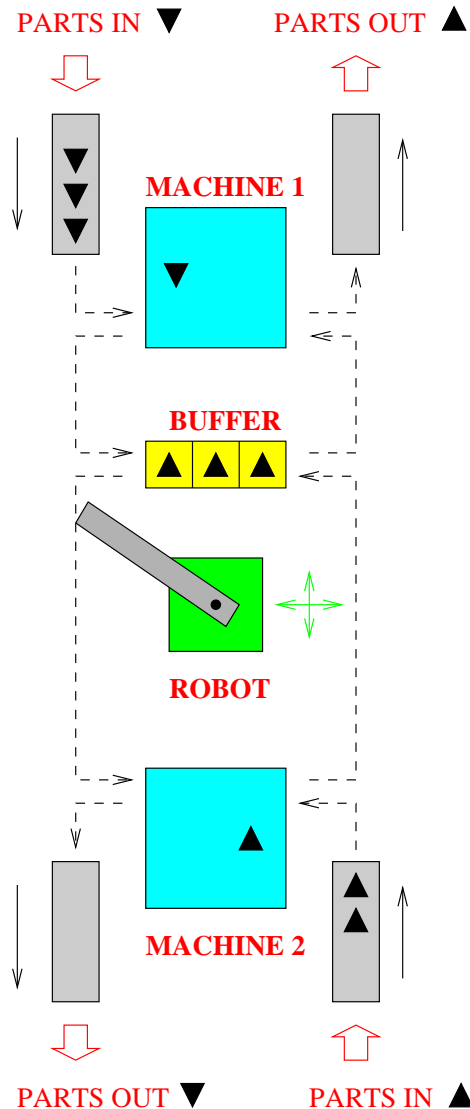
# Deadlock Prevention

# Manufacturing Cell Example

Each machine can only process one part at a time.

The robot moves the parts.

The buffer is the only storage space available to the robot.





# Deadlock Prevention

# Manufacturing Cell Example

The inequalities to be enforced:

$$\mu_3 + \mu_6 + \mu_7 + \mu_{12} \geq 1$$

$$\mu_4 + \mu_7 + \mu_8 + \mu_{11} \geq 1$$

$$\mu_4 + \mu_6 + \mu_7 + \mu_8 + \mu_{12} \geq 1$$

$$2\mu_3 + \mu_4 + 2\mu_6 + 2\mu_7 +$$

$$\mu_8 + \mu_{11} + 2\mu_{12} \geq 3$$

In addition to the inequalities above, the initial marking  $\mu_0$  of the plant must satisfy:

$$\mu_{0,2} + \mu_{0,6} + \mu_{0,12} \geq 1$$

$$\mu_{0,3} + \mu_{0,7} + \mu_{0,11} \geq 1$$

$$\mu_{0,4} + \mu_{0,8} + \mu_{0,10} \geq 1$$

