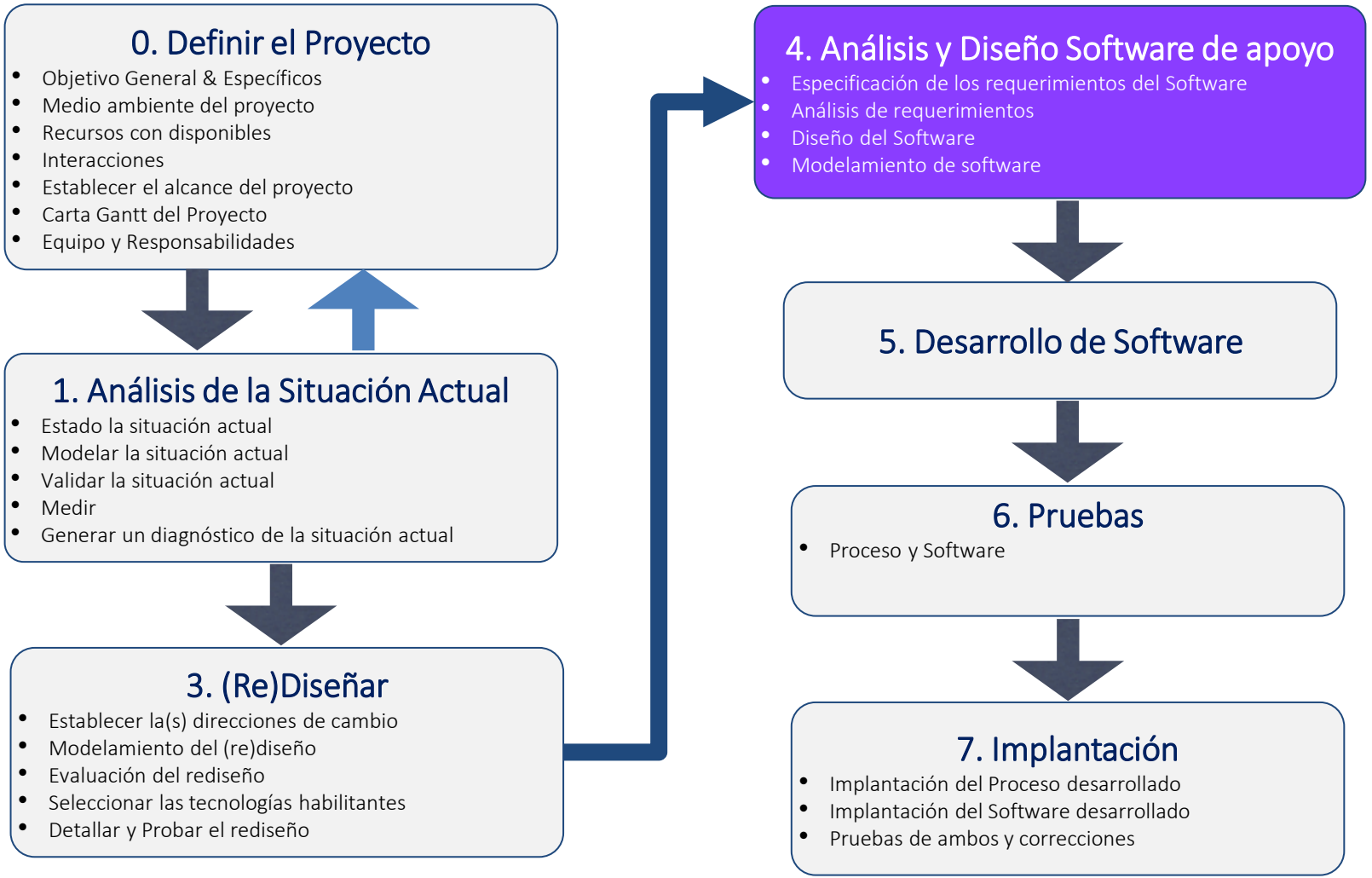
Análisis y Diseño Lógico de Sistemas

3419B201-A


0. Definir el Proyecto
Objetivo General & Específicos
Medio ambiente del proyecto
Recursos con disponibles
Interacciones
Establecer el alcance del proyecto
Carta Gantt del Proyecto
Equipo y Responsabilidades
1. Análisis de la Situación Actual
Estado la situación actual
Modelar la situación actual
Validar la situación actual
Medir
Generar un diagnóstico de la situación actual
3. (Re)Diseñar
Establecer la(s) direcciones de cambio
Modelamiento del (re)diseño
Evaluación del rediseño
Seleccionar las tecnologías habilitantes
Detallar y Probar el rediseño
4. Análisis y Diseño Software de apoyo
Especificación de los requerimientos del Software
Análisis de requerimientos
Diseño del Software
Modelamiento de software
5. Desarrollo de Software
6. Pruebas
Proceso y Software
7. Implantación
Implantación del Proceso desarrollado
Implantación del Software desarrollado
Pruebas de ambos y correcciones

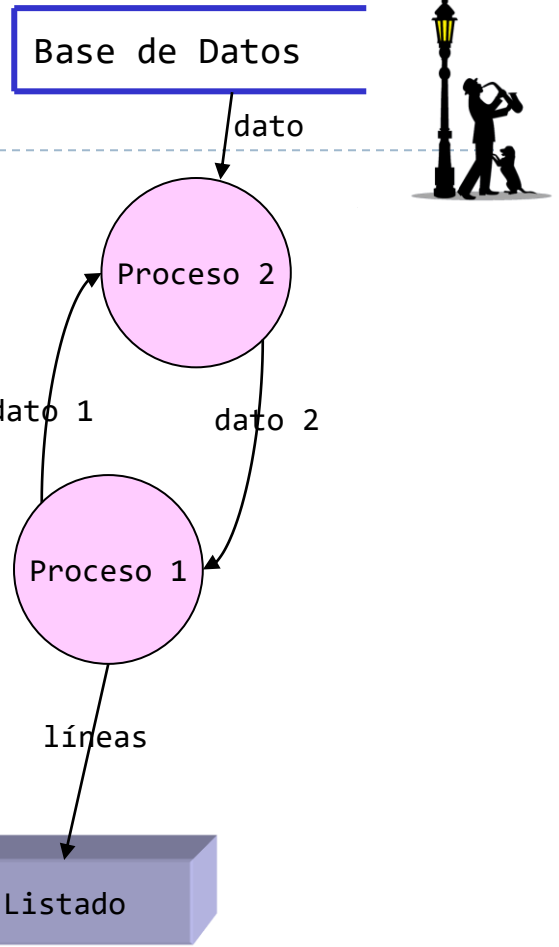
Especificaciones operacionales

Diagramas de Flujo de Datos

- Los Diagramas de Flujo de Datos (DFDs) son populares para especificar funcionalidad en sistemas de información.
- Los sistemas se ven como colecciones de datos manipulados por ciertos procesos.
- Los datos pueden ser:
  - estáticos - en repositorios de datos,
  - dinámicos - comunicación entre procesos y entidades externas.

DFDs: Notación

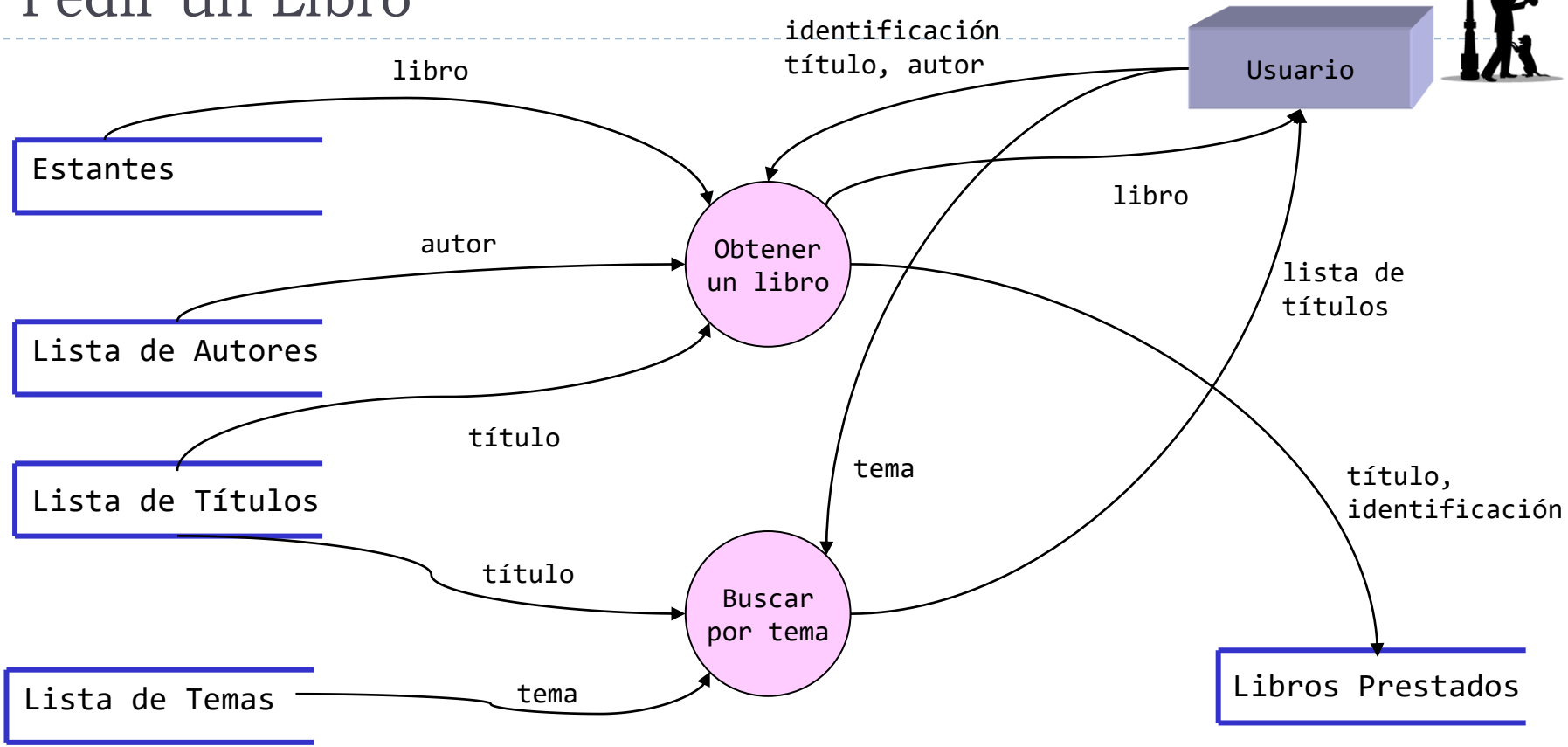
- Burbujas: representan funciones.
- Flechas: flujos de datos.
  - Pueden entrar o salir de una burbuja.
- Cajas abiertas: almacenamiento de datos.
  - Flechas saliendo de las cajas son lectura de datos, y entrando son escritura de datos.
- Cajas cerradas: entidades externas al sistema.
  - Pueden ser usuarios u otros sistemas.



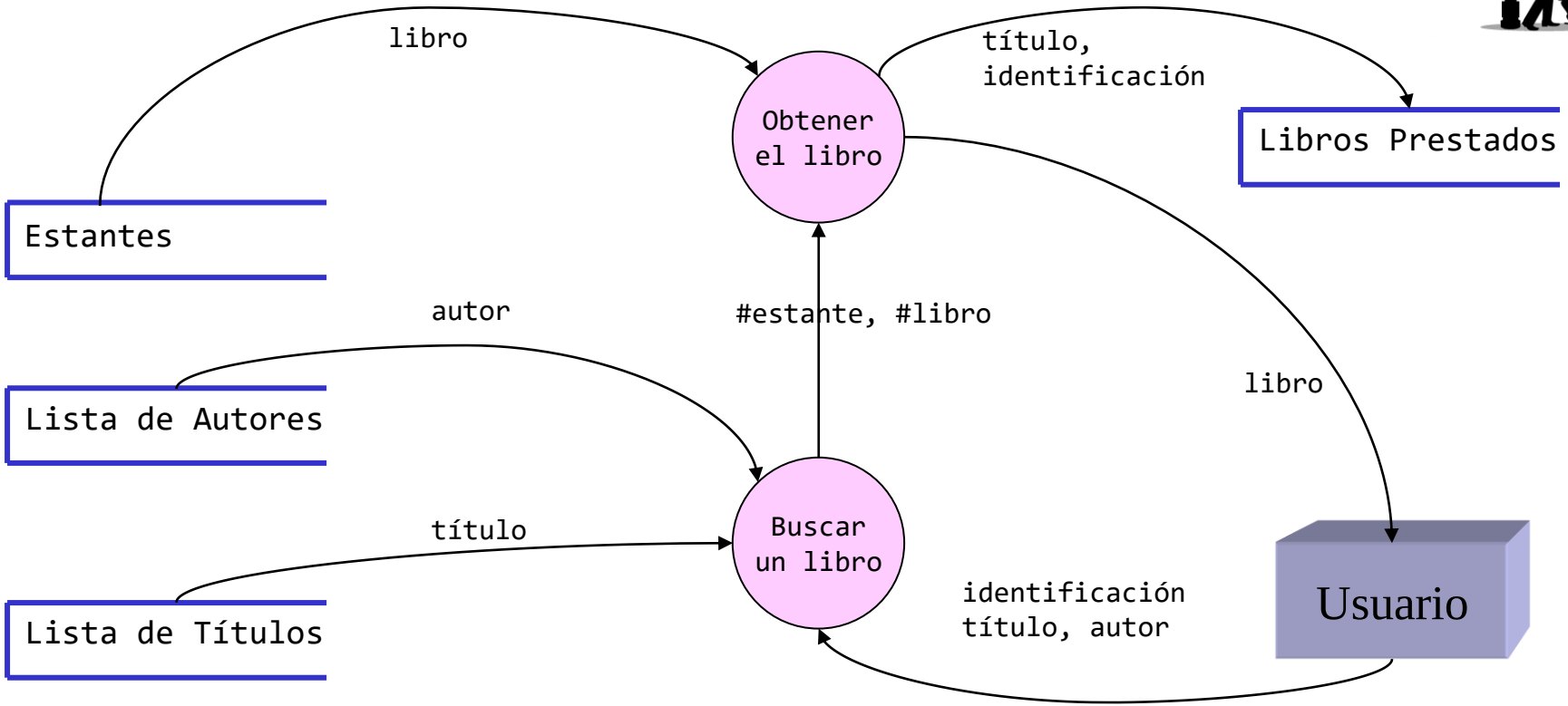
Ejemplo: La Biblioteca

- Para obtener un libro se requiere:
  - una solicitud del usuario conteniendo su identificación, así como también el nombre y el autor del libro;
  - disponer de una lista de libros y autores para localizar el libro;
  - acceder a los estantes donde se encuentra el libro.
- Un DFD es una primera aproximación acerca de la forma en que ocurre este procedimiento.

Pedir un Libro



Obtener un Libro (refinamiento o explosión)



Precisión

- El refinamiento de procesos da mayor precisión a la especificación, pero no elimina la ambigüedad:
  - ¿es necesario dar el título del libro y el autor siempre?
  - ¿alcanza sólo con el nombre del libro?
  - ¿qué sucede si el libro no está en el estante?
  - ¿qué pasa si el usuario debe libros atrasados?

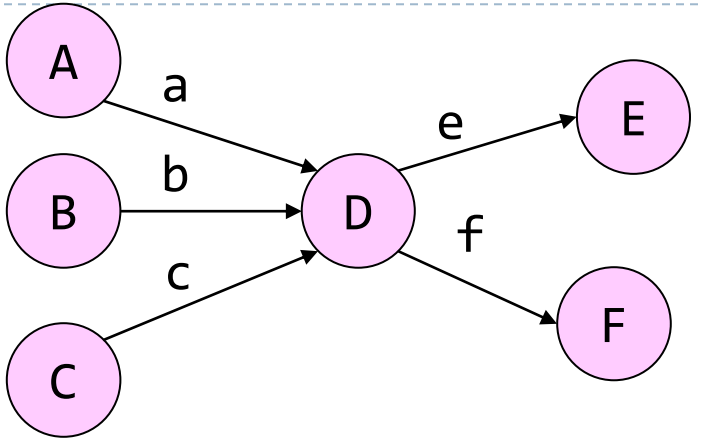
Causas de Imprecisión

- Semántica
  - el significado de los nombres de los elementos del diagrama es elegido (imprecisamente) por el especificador.

```
si el usuario da el título y el autor entonces
     determinar la posición del libro
     si el libro existe entonces
          obtener el libro
     sino dar un mensaje de error;
sino si se da sólo el autor del libro entonces
           dar una lista de libros de ese autor
           pedir un título de la lista;
      sino si se da sólo el título entonces
        ...
```

Más Causas de Imprecisión

- El modelo no define elementos de control.
- No se explicita cómo producir las salidas a partir de las entradas:
  - D necesita a, b y c, para poder producir e y f.
  - D necesita a, b o c para producir e y f.
  - D produce e y f simultáneamente.
  - D produce primero e y luego f.



- Otro posible problema de control:
  - A manda el dato a en cuanto lo tiene disponible.
  - D solicita a A el dato a.

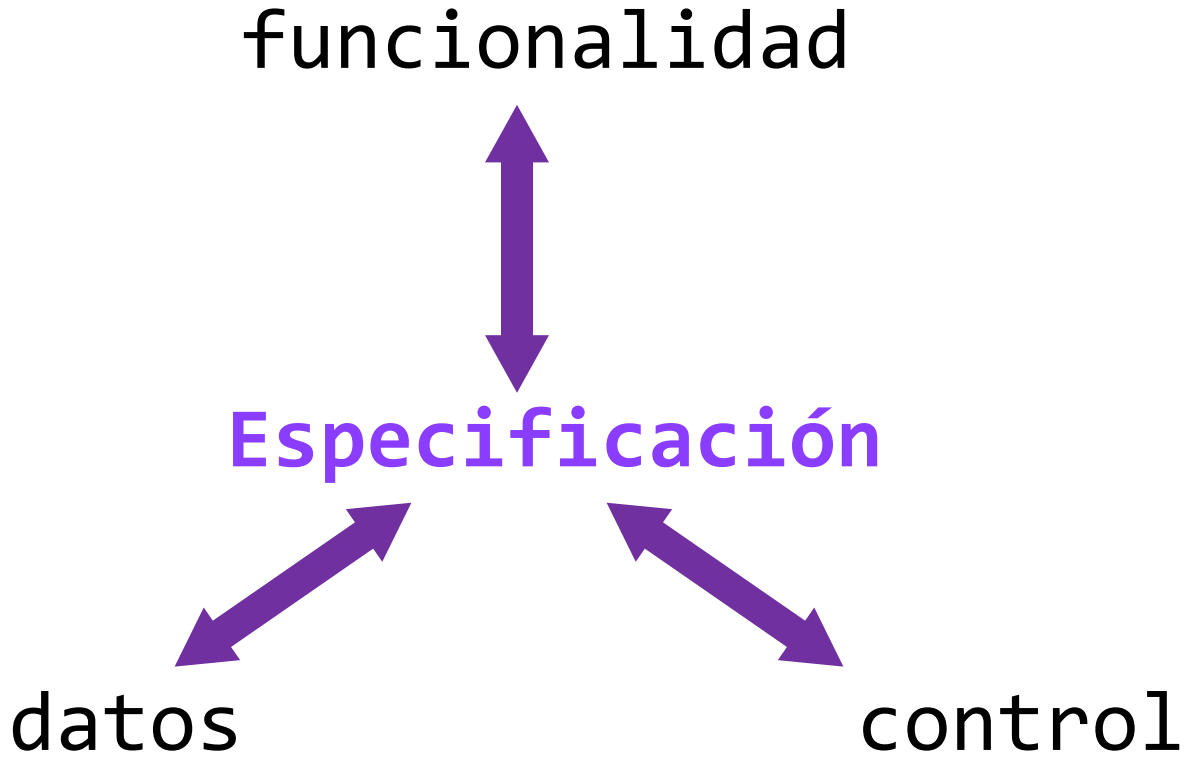
Consecuencias

- Los DFDs son diagramas intuitivos y fácilmente comprensibles para describir el flujo de datos y operaciones en un sistema de información.
- Pero tiene una semántica imprecisa:
  - si se requiere una descripción precisa del sistema, los DFDs no son una herramienta adecuada;
  - no se puede derivar un prototipo automáticamente de un DFD porque existen muchas ambigüedades.
- Los DFDs son una notación semi-formal:
  - la sintaxis es formal pero la semántica no.

Alternativas

- Formas de superar la ambigüedad de los DFDs:
  - usar una notación complementaria para especificar los aspectos ambiguos;
  - extender la notación para incluir elementos de control y sincronización;
  - redefinir los DFDs para hacerlos completamente formales:
    - diferentes flechas para flujos de datos y flujo de control,
    - and/or entre las entradas y las salidas,
    - especificación formal de la función de las burbujas.

Enfoques de Especificación



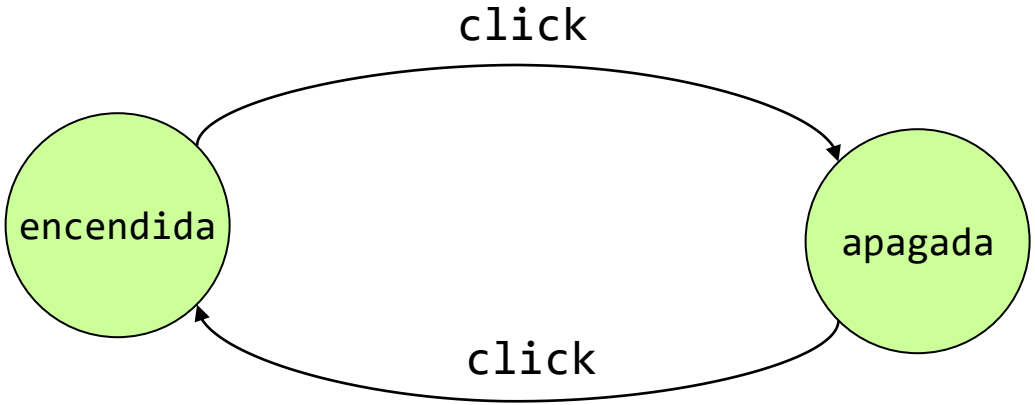
Máquinas de estados finitos

Máquinas de Estados Finitos

- Las máquinas de estados finitos (FSMs) son una notación formal para especificar aspectos de control de un sistema.
- Sus elementos son:
  - un conjunto finito de estados Q;
  - un conjunto finito de entradas I;
  - una función parcial de transición $\delta : Q \times I \rightarrow Q$.
- Gráficamente una máquina de estados finitos se representa como un grafo donde:
  - los nodos representan estados, y
  - existe un arco i de q1 a q2 $\Leftrightarrow \delta(q1,i) = q2$.

Usos de FSMs

- Los FSMs se usan para modelar sistemas que pueden encontrarse en un número finito de estados y que pasan de un estado a otro como reacción a una entrada:
  - una lámpara puede estar encendida o apagada dependiendo de la acción del switch.



Características

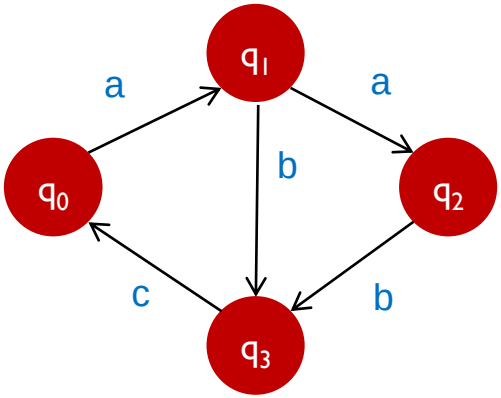
- Simple
- Formal
- Operacional
- Número finito de estados

Definición

- Una máquina de estados finitos M consiste de:
  - Un conjunto finito de estados, Q;
  - Un conjunto finito de entradas, I;
  - Una función de transición $\delta: Q \times I \rightarrow Q$.
  - δ puede ser una función parcial

Representación gráfica

- Grafo
  - nodos representan los estados;
  - un arco rotulado i va del estado q a q’ si y sólo si $\delta(q, i) = q'$.



$Q = \{q_0, q_1, q_2, q_3\}$ $I = \{a, b, c\}$

$\delta(q_0, a) = q_1$

$\delta(q_1, a) = q_2$

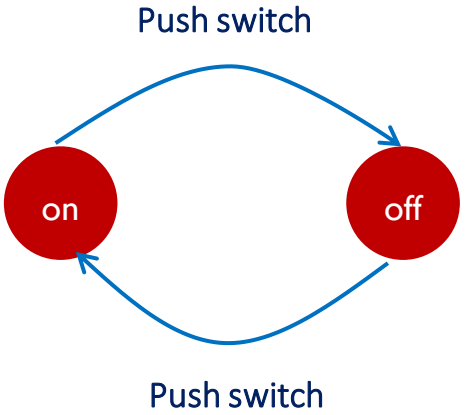
$\delta(q_1, b) = q_3$

$\delta(q_2, b) = q_3$

$\delta(q_3, c) = q_0$

Modelado con una MEF

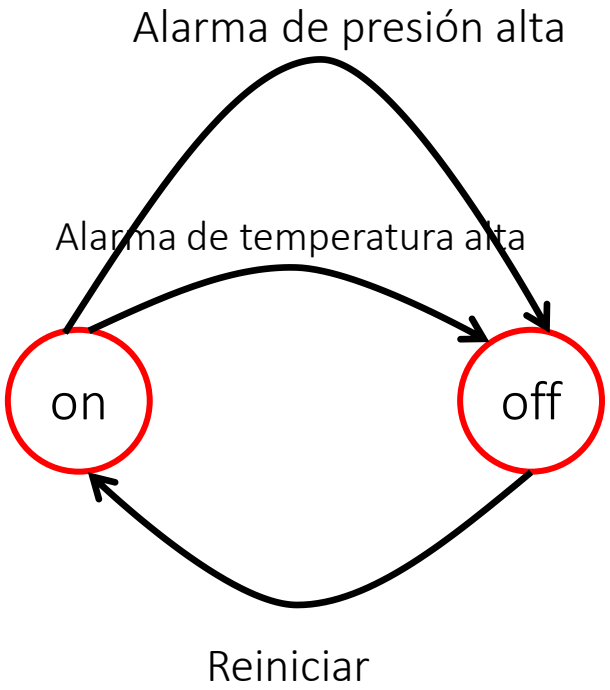
- MEF que modela el switch de una lámpara



Problema

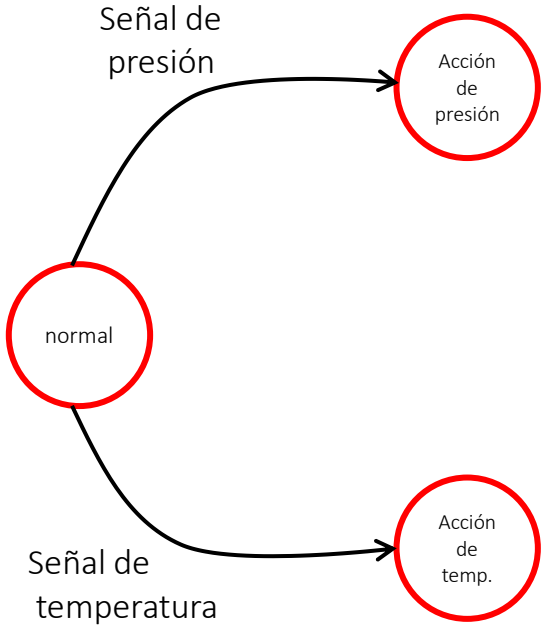
- Una planta química
  - Los niveles de temperatura y de presión deben ser monitoreados por razones de seguridad. Se han instalado sensores para generar señales adecuadas cuando alguno de dichos niveles excede un valor predefinido.
  - Política trivial: cuando alguna de las señales es originada por el correspondiente sensor, el sistema de control apaga la planta y emite una señal de alarma; el sistema es re-iniciado manualmente cuando la causa de la falla ha sido corregida.

Problema


Alarma de presión alta
Alarma de temperatura alta
on
off
Reiniciar

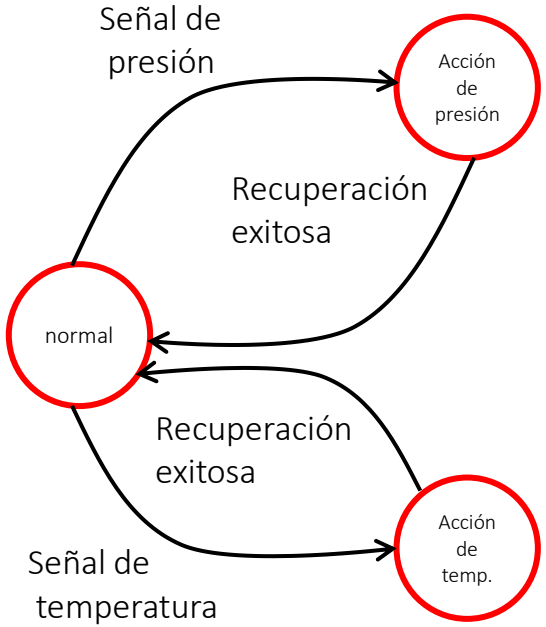
Problema

- Cuando se origina alguna de las dos señales, se invoca automáticamente una acción de recuperación.
  - existe una “acción de temperatura”
  - y una “acción de presión”.



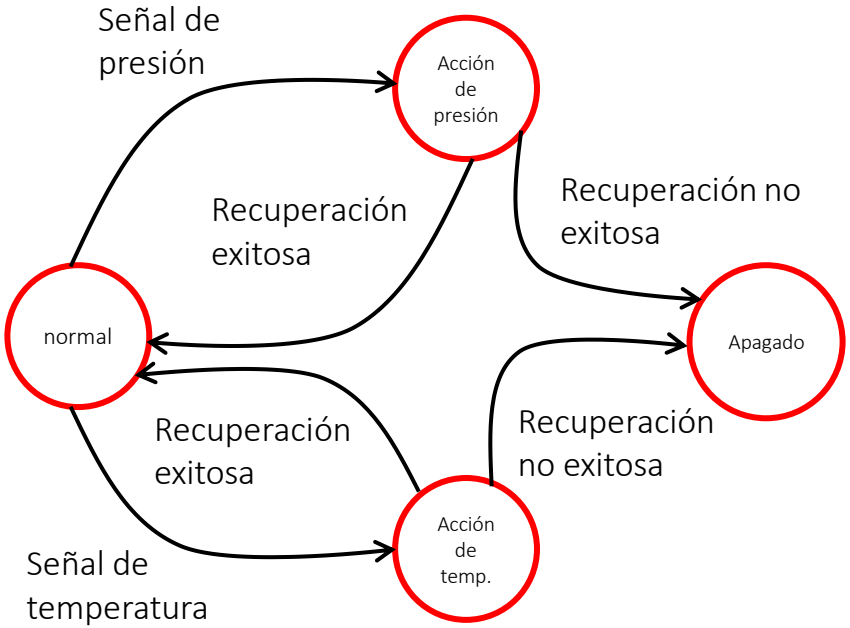
Problema

- Si, después de un rato, la acción de recuperación tiene éxito, el sistema es automáticamente re-inicializado al estado “normal” y un mensaje de “todo OK” es emitido.



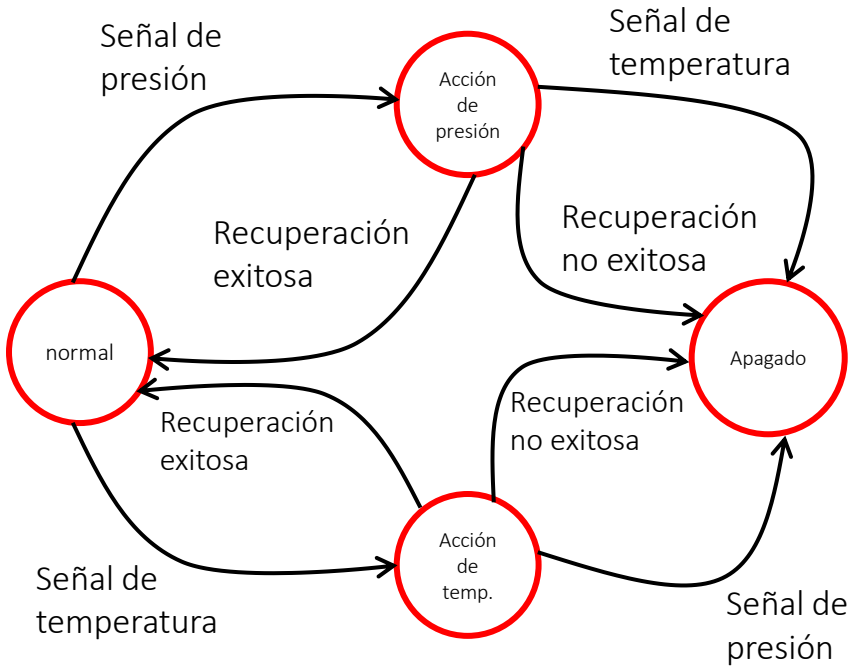
Problema

- Si la recuperación no tiene éxito, la señal de alarma debe ser lanzada y la planta debe ser apagada.

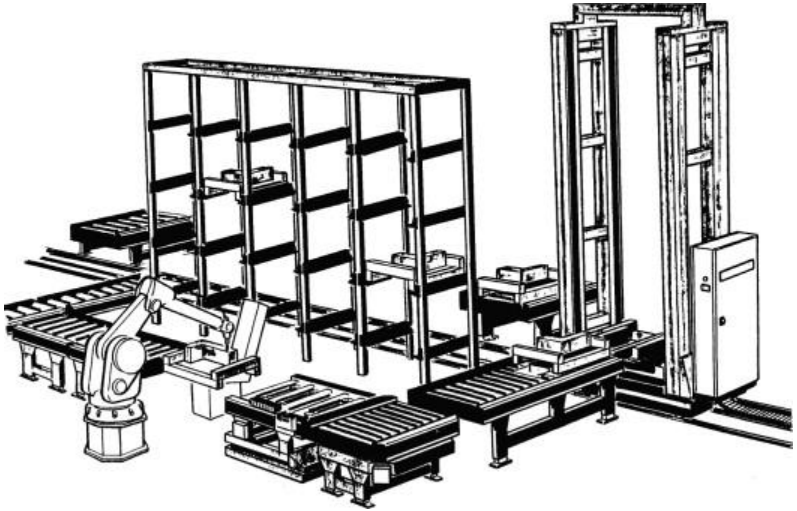


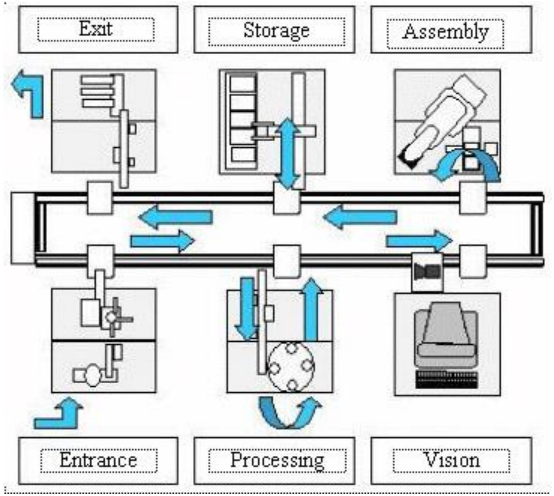
Problema

- El sistema debe también ser apagado si se está tratando de recuperar de algún tipo de anormalidad (de temperatura o presión) y la otra señal aparece.

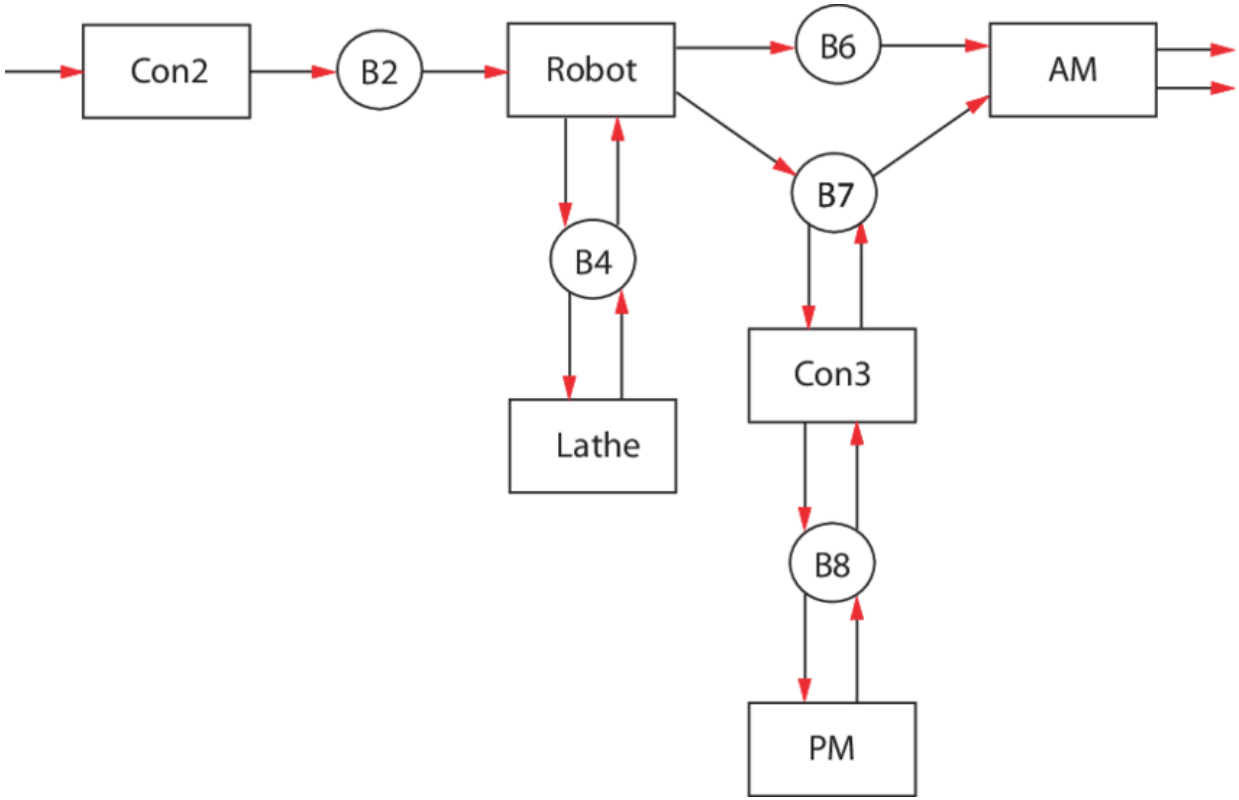


Problema



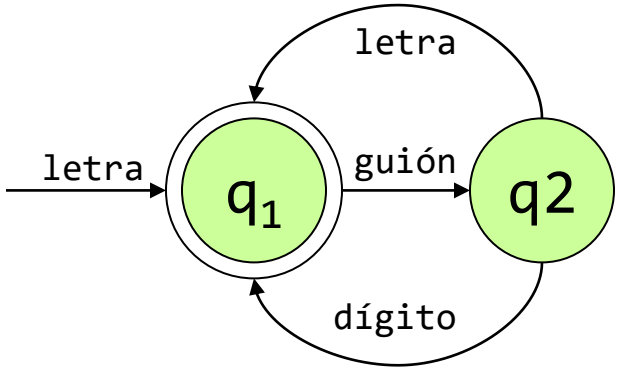


Problema



Otros Usos de FSMs

- Las máquinas de estados finitos se usan para el reconocimiento de secuencias aceptables (lenguajes formales):
  - se agrega la definición de un estado inicial $q_0 \in Q$;
  - un conjunto de estados finales o aceptables;
  - una secuencia es aceptable si existe un camino entre q_0 y un estado final.



Otros Usos Más

- Las máquinas de estados finitos son sencillas para especificar:
  - sistemas de control,
  - compilación,
  - comparación de patrones,
  - diseño de hardware,
  - otras aplicaciones no relacionadas con la computación.

Introducción a las Redes de Petri

Descripción del Problema

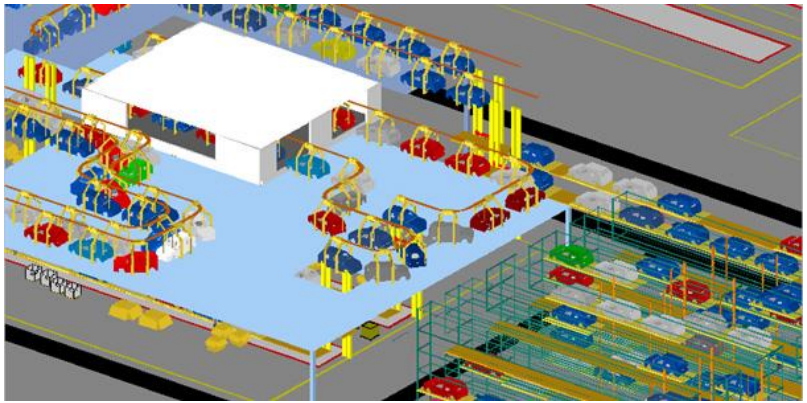
- En diversos sectores productivos, la programación de actividades es de gran importancia en la incidencia de la productibilidad y competitividad de las empresas en el mercado, por lo cual es fundamental que las empresas del sector productivo se acoplen al mercado cambiante, en este contexto, las empresas se ven en la necesidad de contar con una programación de la producción efectiva. Los sectores productivos deben tener la capacidad de competir y una formas para lograrlo es la producción por FMS (acrónimo en inglés de Flexible Manufacture System) o Sistema de Manufactura Flexible siendo una gran promesa para el futuro ofreciendo beneficios en la calidad de los productos, reducción de costos y un mejor manejo en los procesos.

Planteamiento del Problema

- En la simulación y análisis de FMS sería útil el uso de un Software de Simulación Gráfico enfocado en el usuario final permitiendo reducir el nivel de complejidad de modelamiento de los sistemas sin la necesidad que el usuario requiera de conocimientos medios o avanzados de programación e ingeniería en procesos de manufactura, para lo cual se emplea un método como las Redes de Petri (RdP) el cual es usado en varias aplicaciones para modelar y describir el comportamiento de sistemas dinámicos discretos tratando de que su representación sea lo más cercana a la realidad.

Simulación

- “El proceso de diseñar y desarrollar un modelo de sistema o proceso, y conducir experimentos con este modelo con el propósito de entender el comportamiento del sistema o evaluar estrategias con las cuales se puede operar sobre él”. (Shannon, 1988)



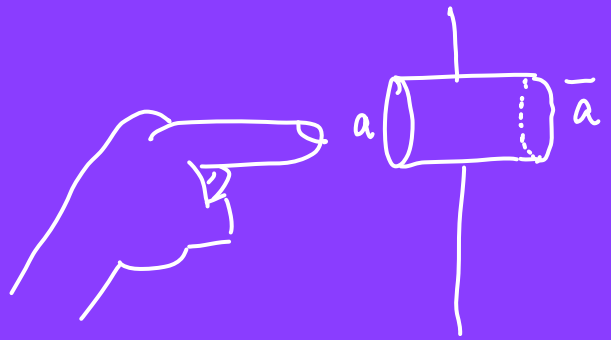
Sistemas de Manufactura Flexible (FMS)

- Un FMS según (Groove, 1990) “consiste de un grupo de estaciones de procesamiento (predominantemente máquinas herramientas CNC), interconectadas por medio de un sistema de manejo y recuperación de material automático. Lo que da su nombre al FMS es su capacidad de procesar una variedad de diferentes tipos de partes simultáneamente bajo un programa de control NC en varias estaciones”.
- La principal característica de un FMS, es su capacidad para procesar múltiples productos con volúmenes de producción diversos brindando gran flexibilidad y adaptabilidad dentro del sistema.

Redes de Petri

- Es una herramienta de modelado muy efectiva para la representación y el análisis de procesos concurrentes.
  1. El sistema completo es a menudo más fácil de entender debido a la naturaleza gráfica y precisa del esquema de representación.
  2. El comportamiento del sistema puede ser analizado utilizando la teoría de las redes de Petri, que incluye herramientas para el análisis tales como los árboles enmarcados y establece relaciones entre ciertas estructuras de redes y el comportamiento dinámico de la Programación Concurrente.
  3. Puesto que las redes de Petri pueden sintetizarse es posible diseñar automáticamente sistemas cuyo comportamiento es conocido o fácilmente verificable

Introducción a las redes de Petri



Redes de Petri - Definición

Una **Red de Petri** es un *multigrafo bipartito dirigido* representado por una cuádrupla $PN = <P, T, I, O>$ donde

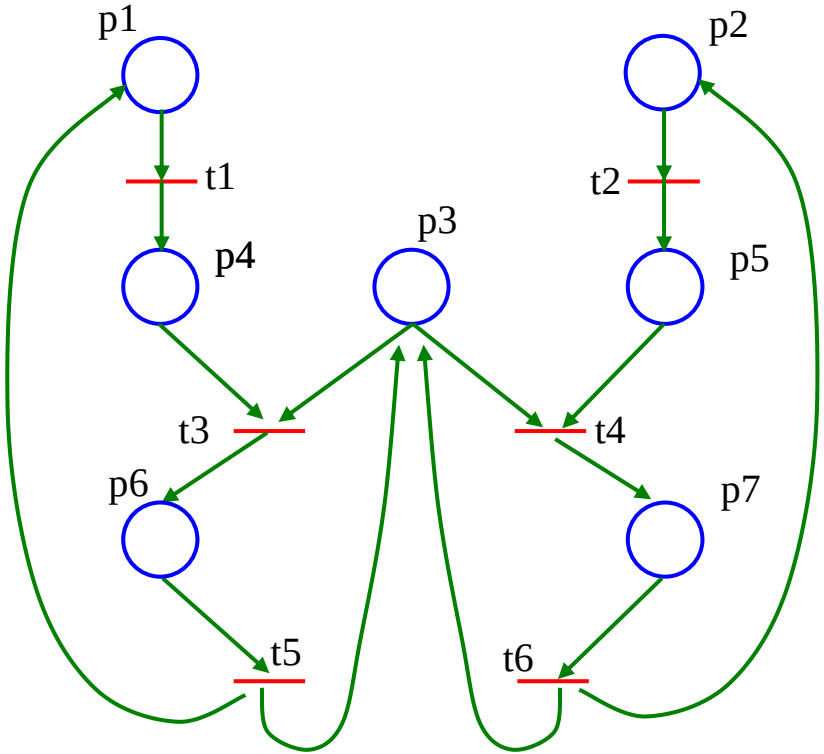
- $P = \{p_1, p_2, ..., p_n\}$ conjunto finito de lugares (places), $n \geq 0$
- $T = \{t_1, t_2, ..., t_m\}$ conjunto finito de transiciones, $m \geq 0$
- $I: T \rightarrow P^\infty$ P^∞ denota los multisets o bolsas de P.

  $p_i \in I(t_j)$ si existe un arco de p_i a t_j

- $O: T \rightarrow P^\infty$

  $p_i \in O(t_j)$ si existe un arco de t_j a p_i

Ejemplo 1



P = {p1, p2, p3, p4, p5, p6, p7}

T = {t1, t2, t3, t4, t5, t6}

I(t1) = {p1} O(t1) = {p4}

I(t2) = {p2} O(t2) = {p5}

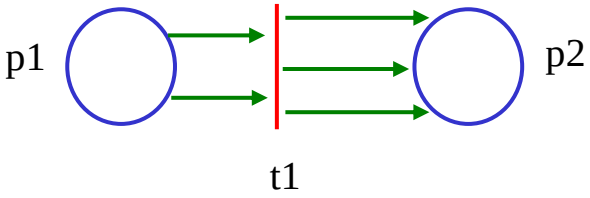
I(t3) = {p3, p4} O(t3) = {p6}

I(t4) = {p3, p5} O(t4) = {p7}

I(t5) = {p6} O(t5) = {p1, p3}

I(t6) = {p7} O(t6) = {p2, p3}

Ejemplo 2



P = {p1, p2}

T = {t1}

I(t1) = {p1, p1}

O(t1) = {p2, p2, p2}

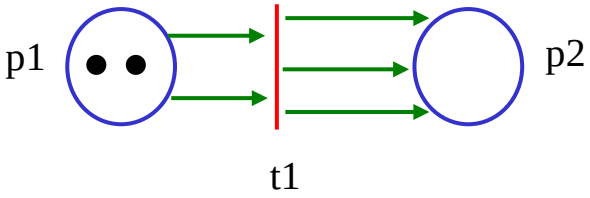
Redes de Petri

- Si un arco va desde un place a una transición, al place se le llama input place (lugar de entrada) de la transición.
- Si un arco va desde una transición a un place, al place se le llama output place (lugar de salida) de la transición.

Redes de Petri - Marcación y ejecución

- Una red de Petri recibe su estado marcando sus places.
- Una marcación (marking) consiste en asignar un número entero no negativo de tokens a cada place.
- En las redes tradicionales de Petri los tokens son indistinguibles y no representan información específica.

Ejemplo de red marcada



P = {p1, p2}

T = {t1}

I(t1) = {p1, p1}

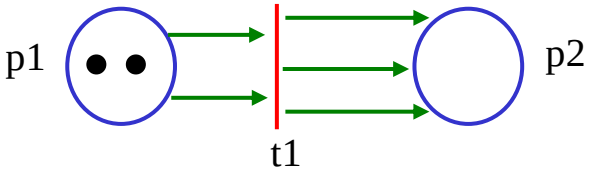
O(t1) = {p2, p2, p2}

Marcación

- Una *marcación* de una Red de Petri
- *PN = <P, T, I, O>* es una función:
- $\mu: P \rightarrow \{0, 1, 2, \ldots\}$
- que asigna un número entero no negativo de tokens a cada place de la red.

Marcación

- Una marcación puede representarse como un vector de dimensión n (n es el número de lugares de la red).



$\mu(p1) = 2 \quad \mu(p2) = 0$

$\mu = (2,0)$

Red de Petri marcada

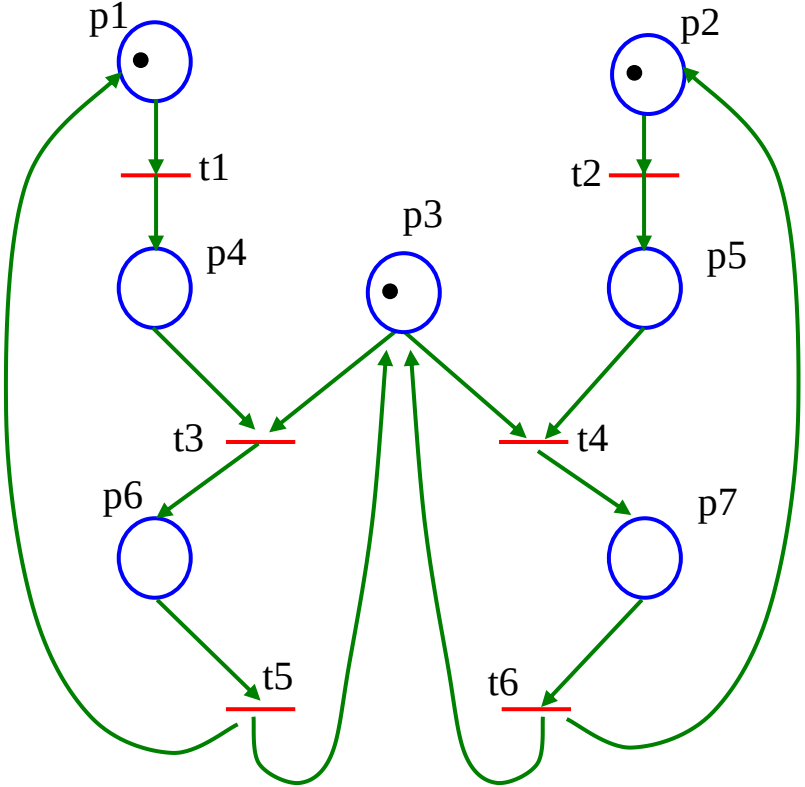
Una *Red de Petri marcada*

$$M = \langle PN, \mu \rangle$$

es un una estructura de Red de Petri

$PN = \langle P, T, I, O \rangle$ y un marking μ.

Red de Petri marcada



P = {p1, p2, p3, p4, p5, p6, p7}

T = {t1, t2, t3, t4, t5, t6}

I(t1) = {p1} O(t1) = {p4}

I(t2) = {p2} O(t2) = {p5}

I(t3) = {p3, p4} O(t3) = {p6}

I(t4) = {p3, p5} O(t4) = {p7}

I(t5) = {p6} O(t5) = {p1, p3}

I(t6) = {p7} O(t6) = {p2, p3}

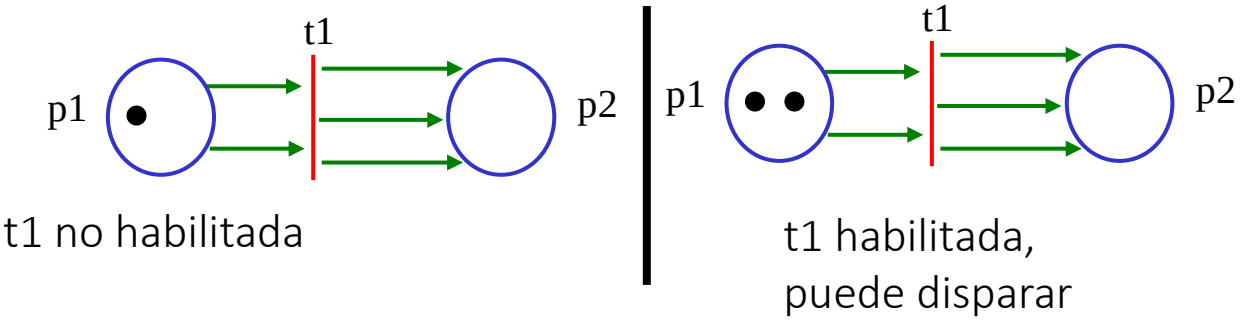
$\mu = (1, 1, 1, 0, 0, 0, 0)$

Ejecución de una Red de Petri

- Durante la ejecución de la red el número y posición de los tokens puede variar dando lugar a una nueva marcación.
- Cada marcación corresponde a un estado de la red.

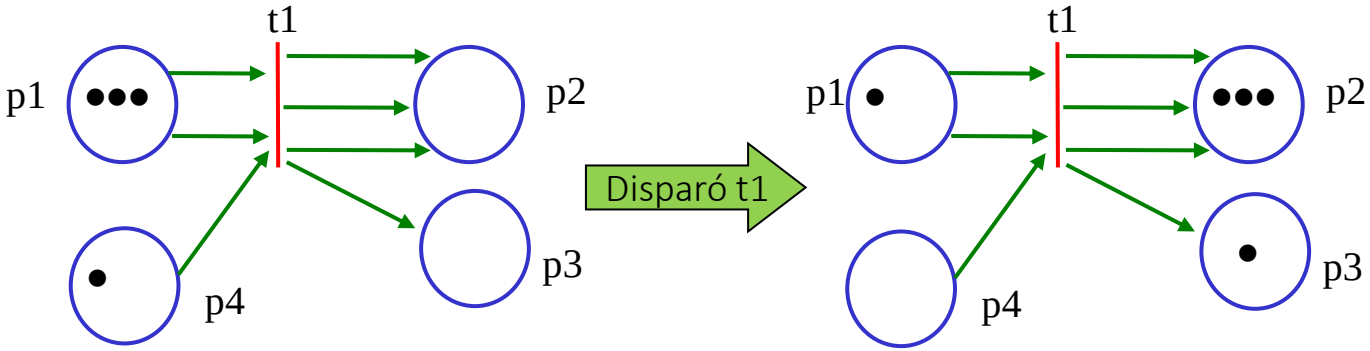
Reglas de Ejecución

1. Una transición puede disparar si está habilitada.
2. Una transición está habilitada si en cada uno de sus input places existen al menos tantos tokens como arcos existan desde el place a la transición.



Reglas de Ejecución

3. Cuando una transición dispara, en cada uno de sus input places se remueven tantos tokens como arcos existan desde el input place hacia la transición, y en cada uno de los output places de la transición se insertan tantos tokens como arcos existan desde la transición al output place.



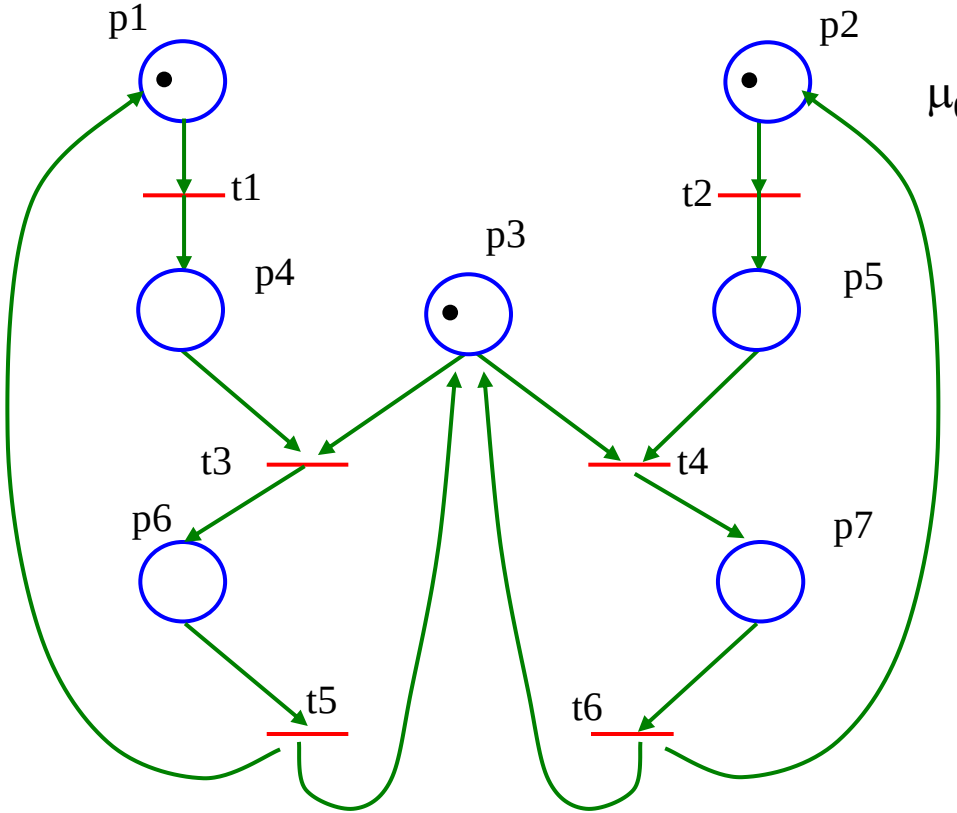
Reglas de Ejecución

- Formalmente,
  - una transición $t \in T$ está habilitada en una red de Petri PN = <P, T, I, O> sii
- $\forall p \in I(t) : \mu(p) \geq \#(p, I(t))$
- donde:
- #(p, I(t)) es el número de arcos desde p a t.

Reglas de Ejecución

- Formalmente, el disparo de una transición $t \in T$ con un marking μ resulta en un nuevo marking μ' definido por:
  - $\forall p \in P : \mu'(p) = \mu(p) - \#(p, I(t)) + \#(p, O(t))$
- donde
  - $\#(p, I(t))$ es el número de arcos desde p a t.
  - $\#(p, O(t))$ es el número de arcos desde t a p.

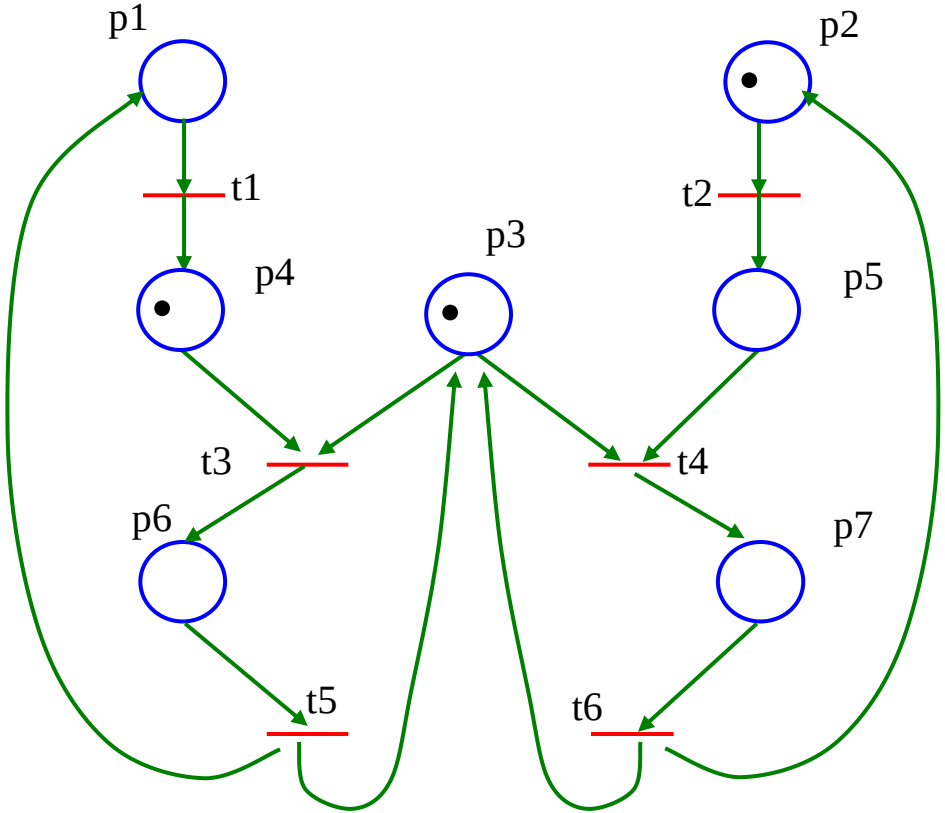
Ejecución de una Red de Petri



$\mu_0 = (1, 1, 1, 0, 0, 0, 0)$

t1 y t2 están habilitadas.

Ejecución de una Red de Petri

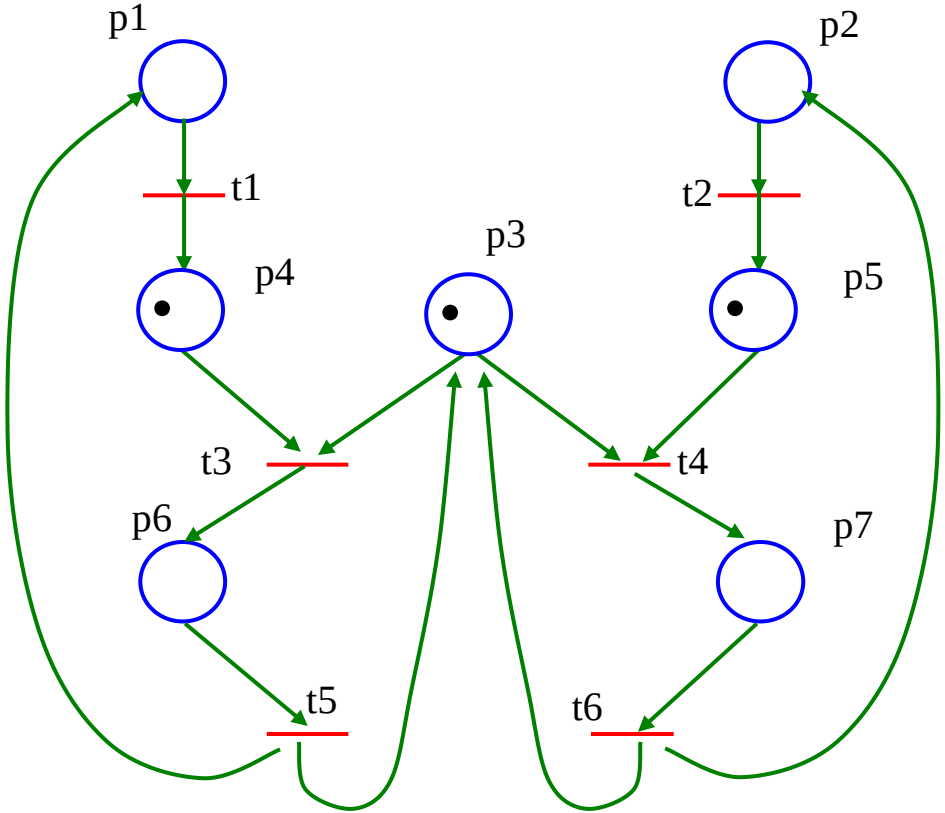


$\mu_0 = (1, 1, 1, 0, 0, 0, 0)$

$\mu_1 = (0, 1, 1, 1, 0, 0, 0)$

- secuencia de disparos: (t1)
- t2 y t3 quedaron habilitadas.

Ejecución de una Red de Petri



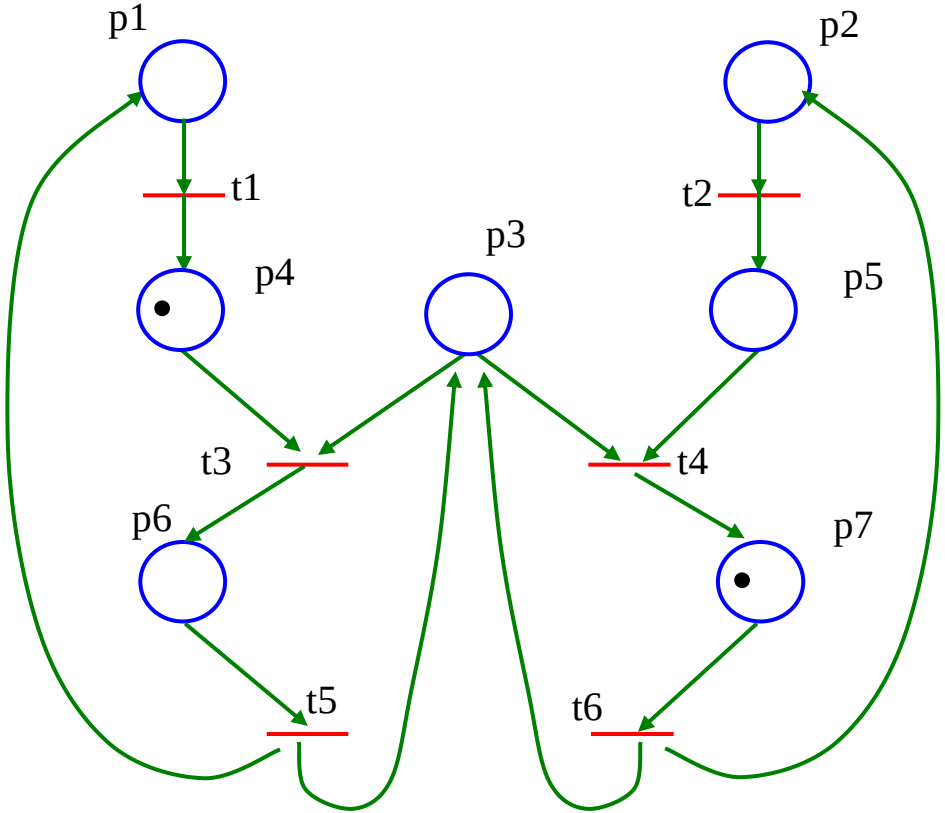
$\mu_0 = (1, 1, 1, 0, 0, 0, 0)$

$\mu_1 = (0, 1, 1, 1, 0, 0, 0)$

$\mu_2 = (0, 0, 1, 1, 1, 0, 0)$

- secuencia de disparos: (t1,t2)
- t3 y t4 quedaron habilitadas.

Ejecución de una Red de Petri



$\mu_0 = (1, 1, 1, 0, 0, 0, 0)$

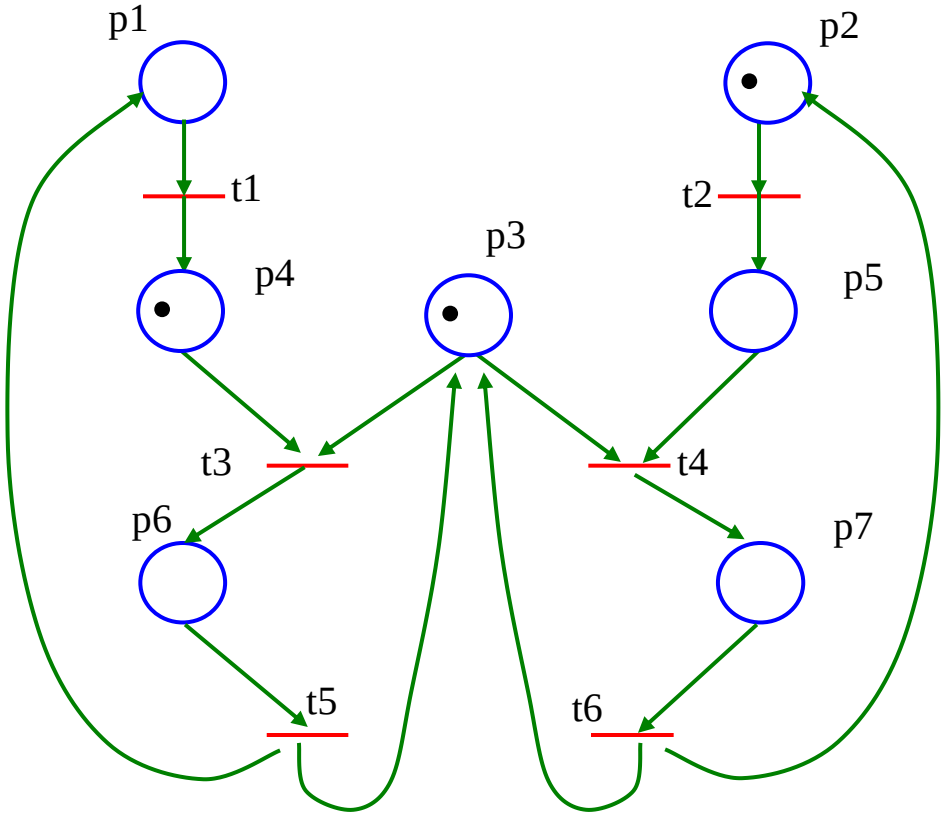
$\mu_1 = (0, 1, 1, 1, 0, 0, 0)$

$\mu_2 = (0, 0, 1, 1, 1, 0, 0)$

$\mu_3 = (0, 0, 0, 1, 0, 0, 1)$

- secuencia de disparos: (t1,t2,t4)
- t6 quedó habilitada.

Ejecución de una Red de Petri



$\mu_0 = (1, 1, 1, 0, 0, 0, 0)$

$\mu_1 = (0, 1, 1, 1, 0, 0, 0)$

$\mu_2 = (0, 0, 1, 1, 1, 0, 0)$

$\mu_3 = (0, 0, 0, 1, 0, 0, 1)$

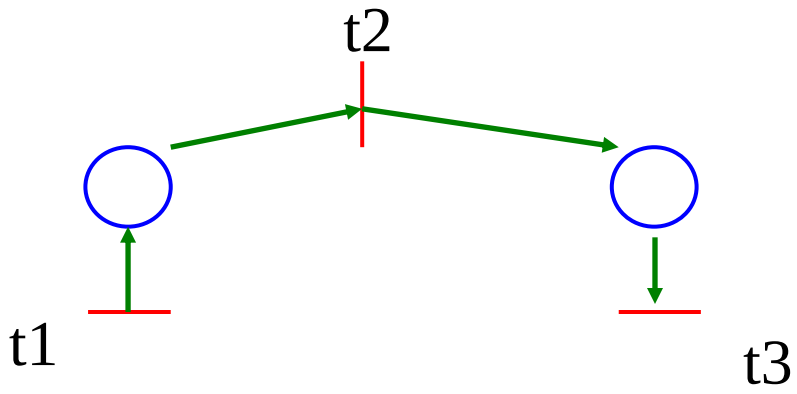
$\mu_4 = (0, 1, 1, 1, 0, 0, 0)$

- secuencia de disparos: (t1,t2,t4,t6)
- t2 y t3 quedaron habilitadas. ...

Ejecución de una Red de Petri

- La secuencia de disparos dada es una de las posibles secuencias, no la única.
- El modelo es *no determinístico*, en el sentido que dado un marking inicial μ_0 distintas evoluciones de la red son posibles.

Ejecución de una Red de Petri

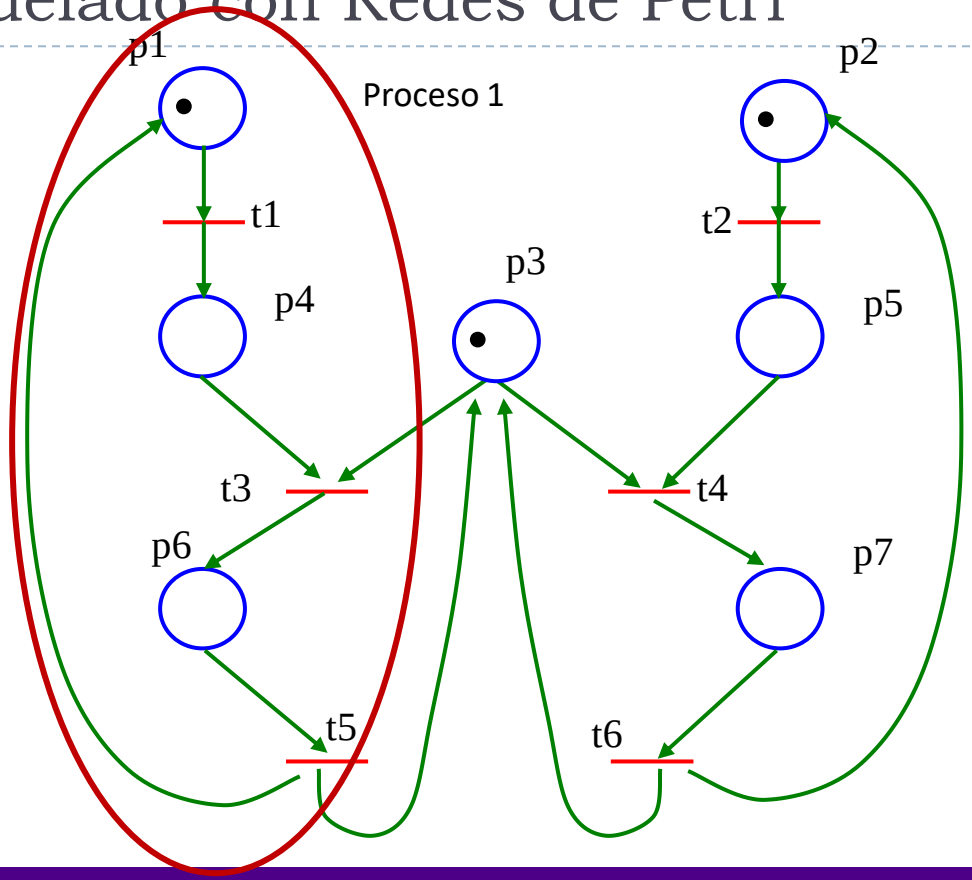


t1 siempre habilitada

Modelado con Redes de Petri

Una transición, usualmente, modela un evento o acción.
El disparo de la transición representa la ocurrencia de tal evento o la ejecución de la acción.
Una transición está habilitada, si se satisfacen las condiciones necesarias para la ocurrencia de tal evento o acción.
La presencia de tokens en los input places modelan tales condiciones.

Modelado con Redes de Petri



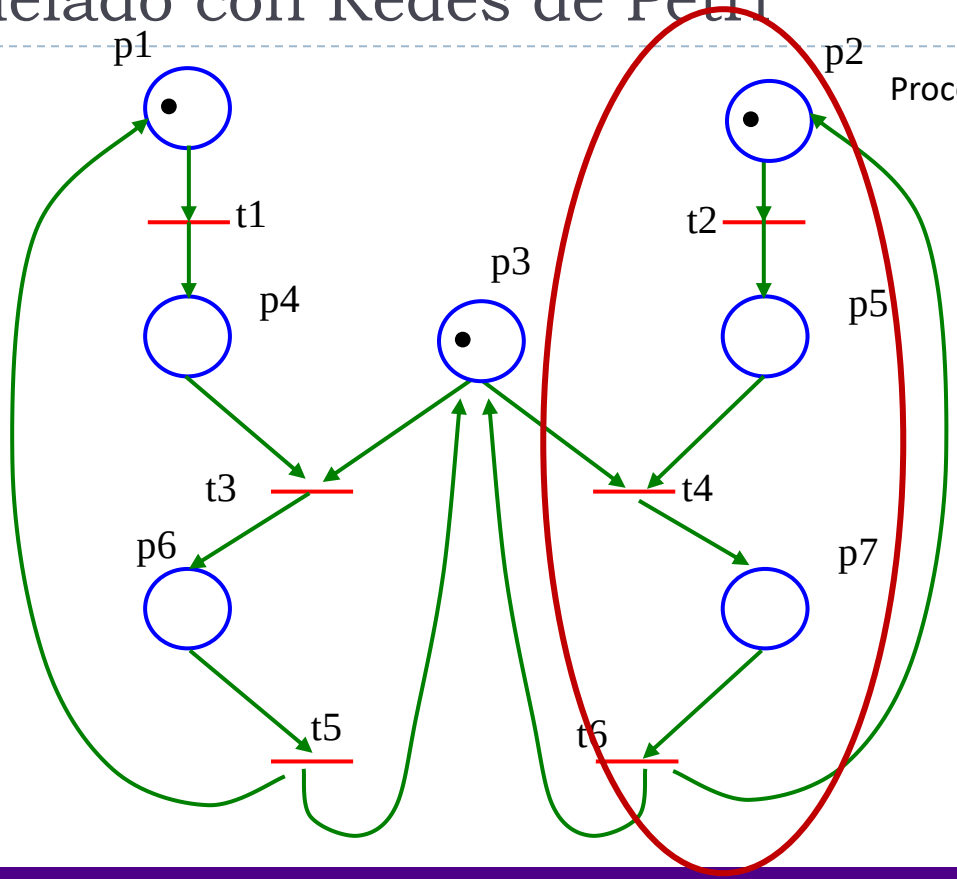
Eventos del proceso 1

t1 = proceso 1 solicita recurso

t3 = proceso 1 toma recurso

t5 = proceso 1 libera recurso

Modelado con Redes de Petri



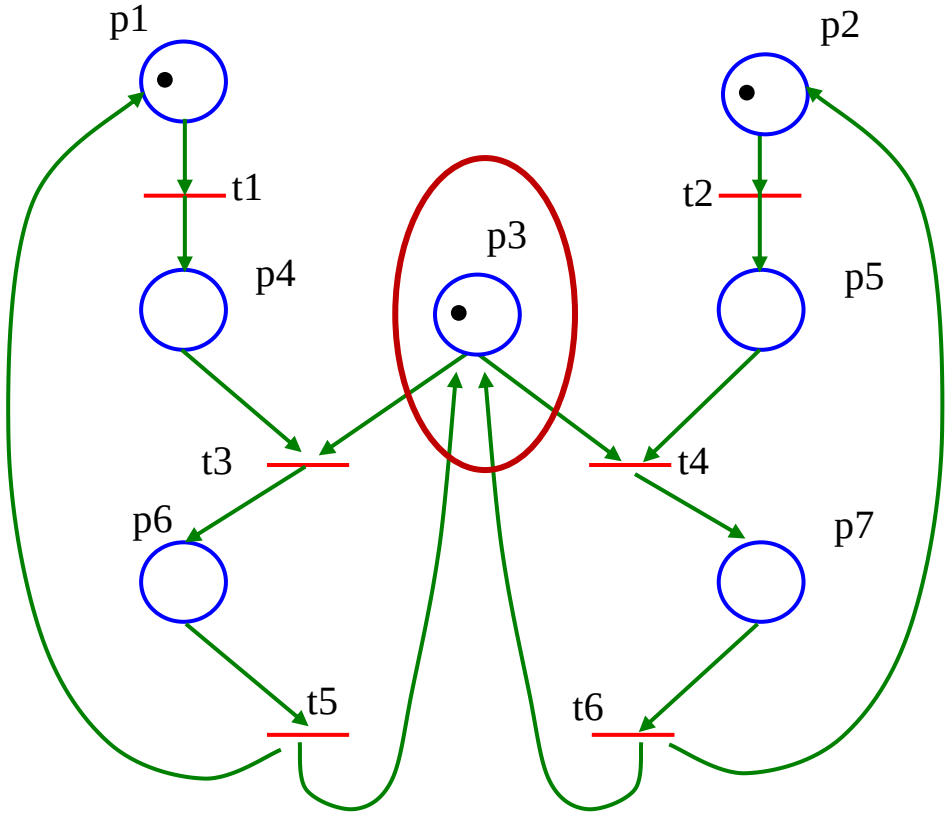
Eventos del proceso 2

t2 = proceso 2 solicita recurso

t4 = proceso 2 toma recurso

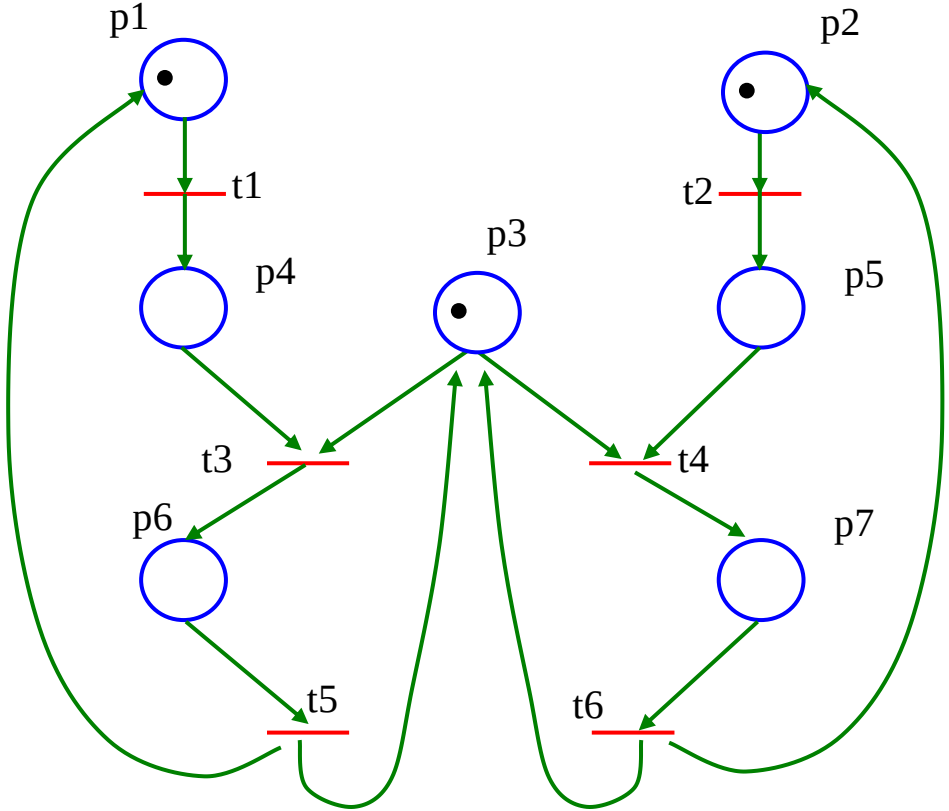
t6 = proceso 2 libera recurso

Modelado con Redes de Petri



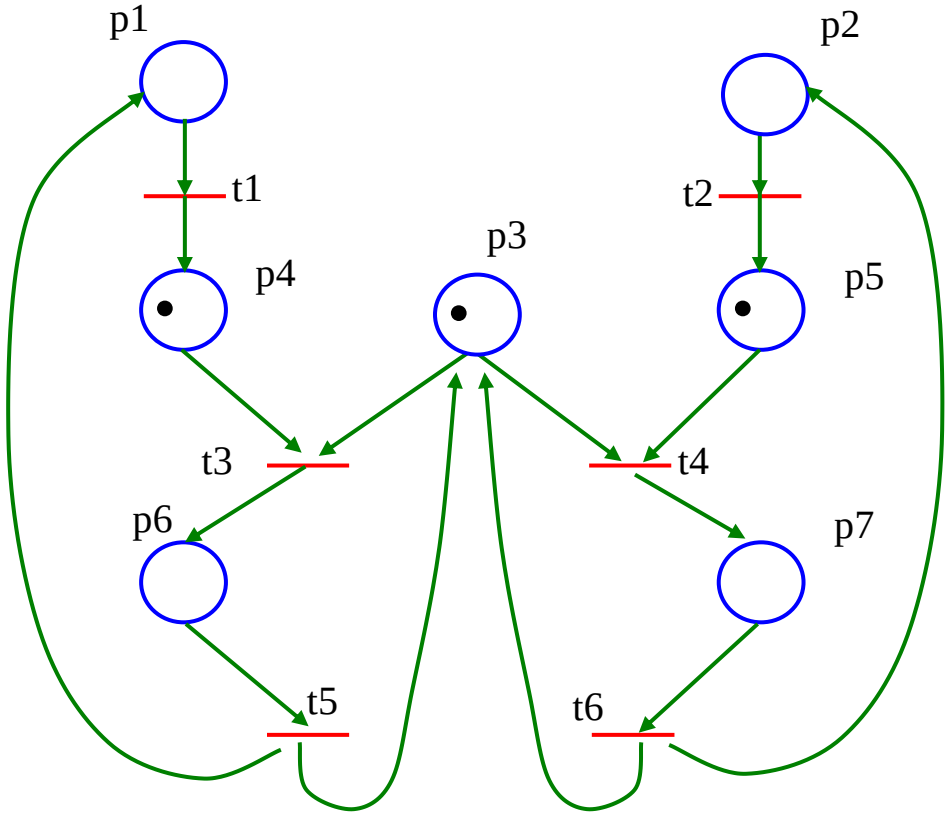
p3 representa la disponibilidad o no de un recurso.

Modelado con Redes de Petri Transiciones concurrentes



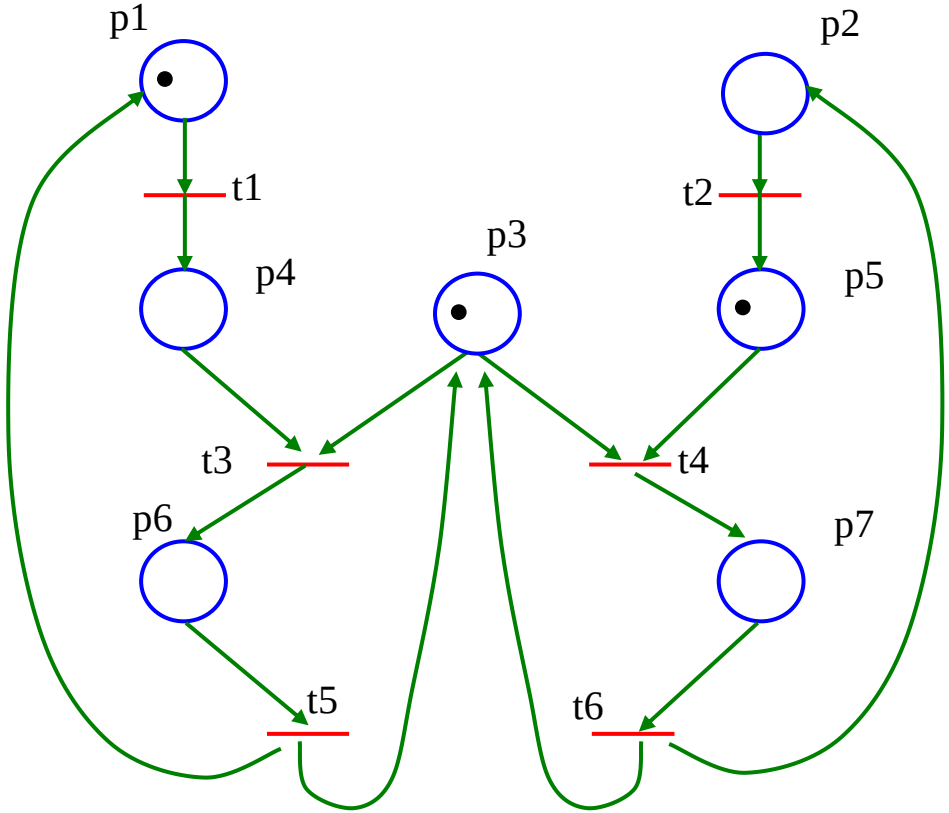
t1 y t2 habilitadas y el disparo de una no deshabilita la otra => *concurrentes*

Modelado con Redes de Petri Transiciones en conflicto



t3 y t4 habilitadas
y el disparo de una
deshabilita la otra
=> *en conflicto*

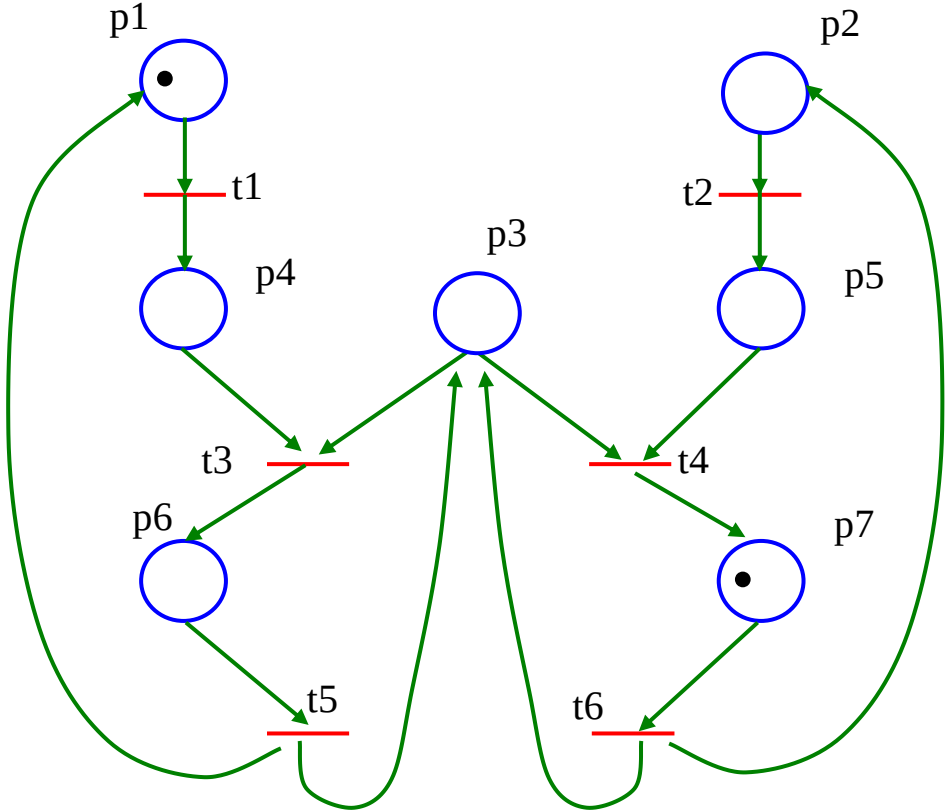
Modelado con Redes de Petri - Starvation



secuencia de disparos:

(t2)

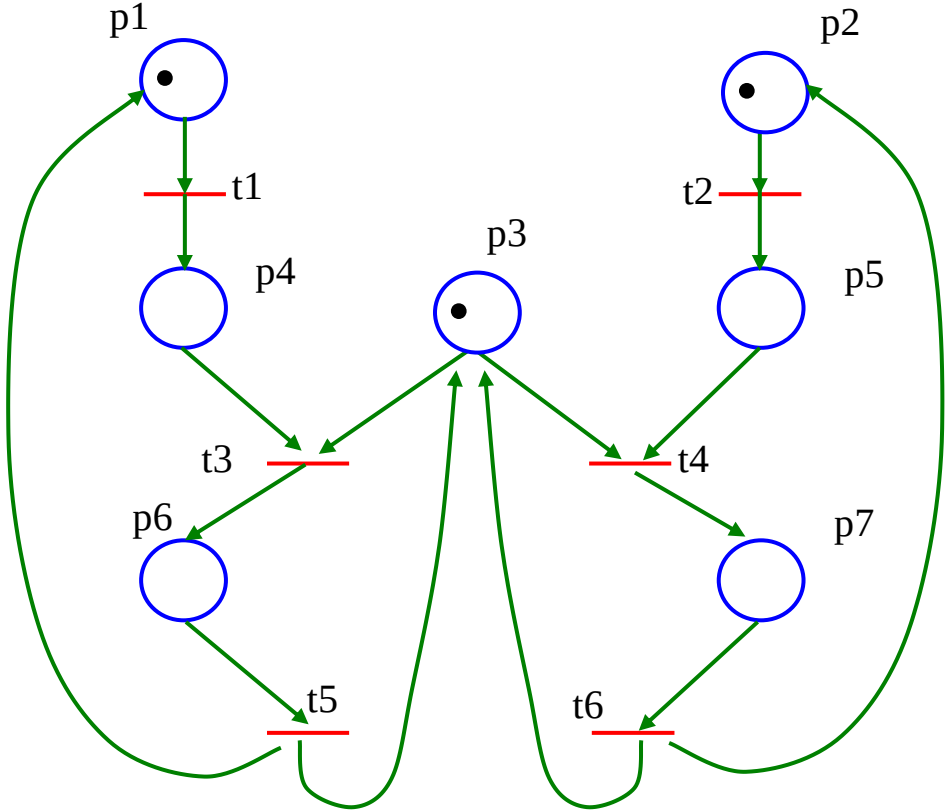
Modelado con Redes de Petri - Starvation



secuencia de disparos:

(t2,t4)

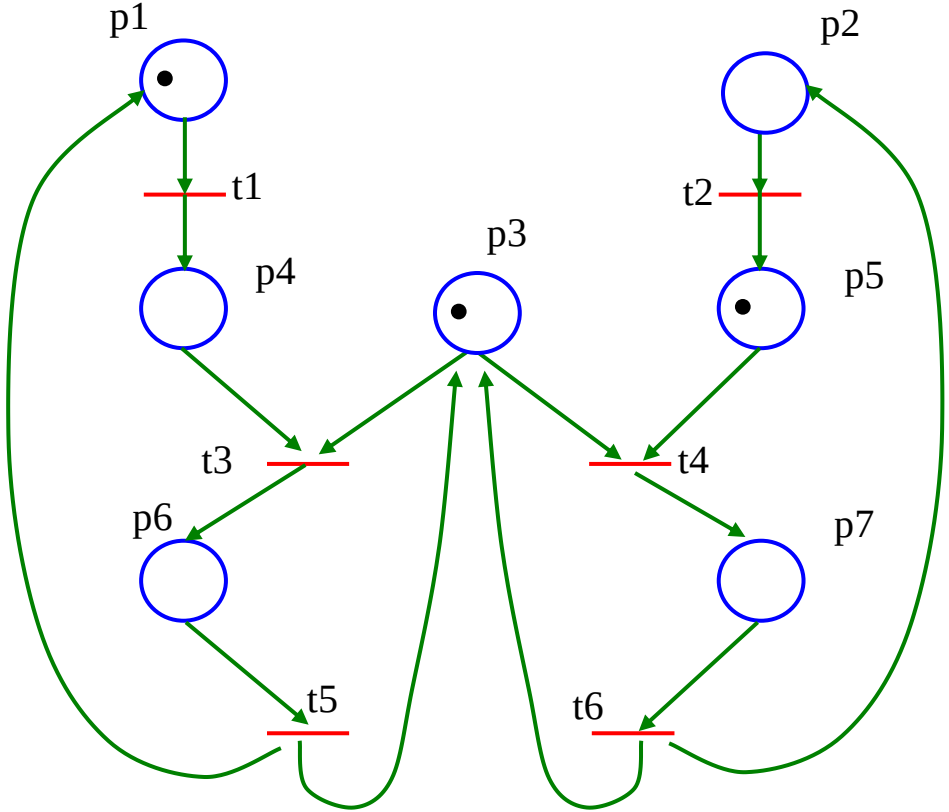
Modelado con Redes de Petri - Starvation



secuencia de disparos:

(t2,t4,t6)

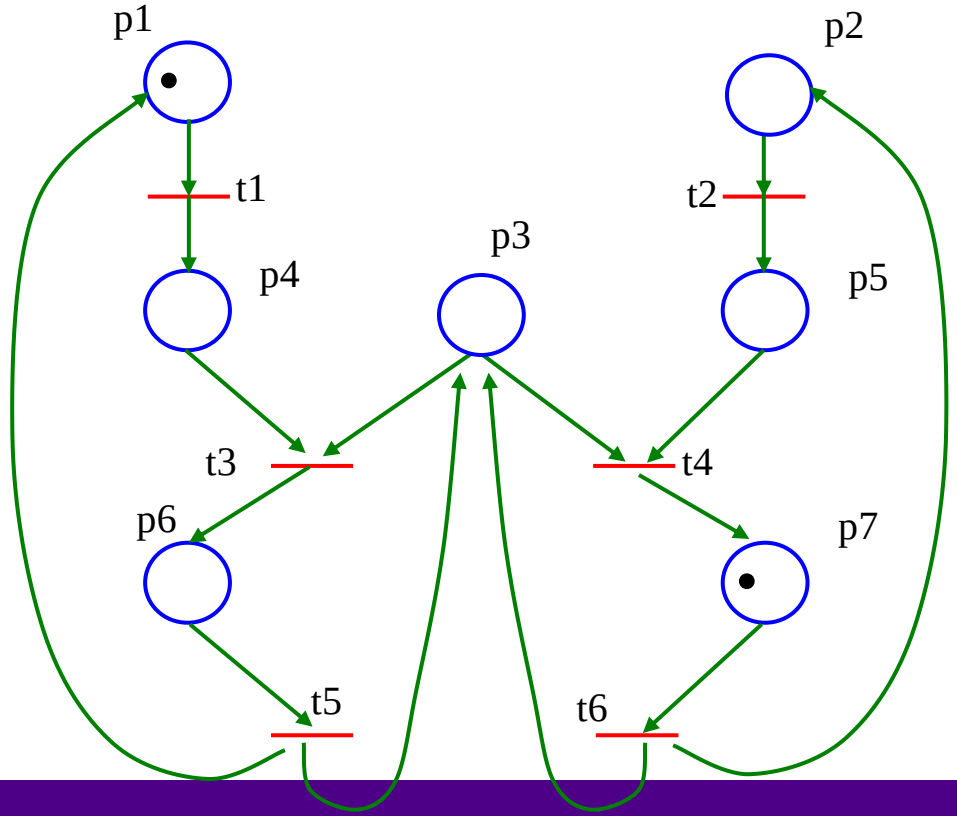
Modelado con Redes de Petri - Starvation



secuencia de disparos:

(t2,t4,t6,t2)

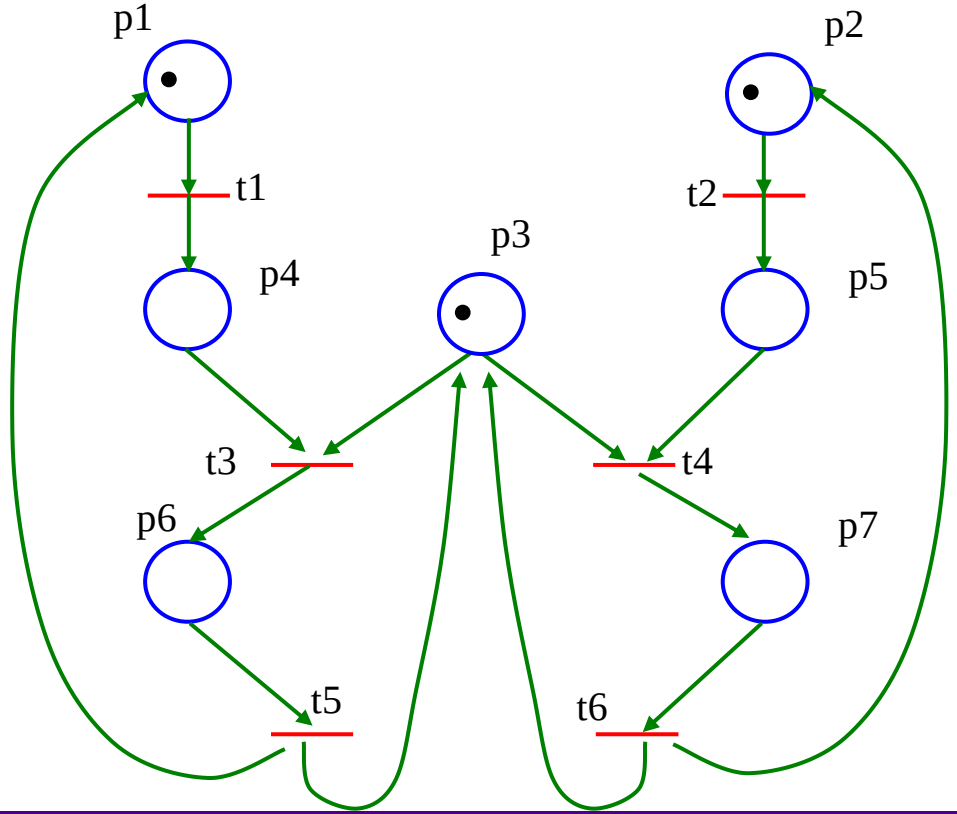
Modelado con Redes de Petri - Starvation



secuencia de disparos:

(t2,t4,t6,t2,t4)

Modelado con Redes de Petri - Starvation



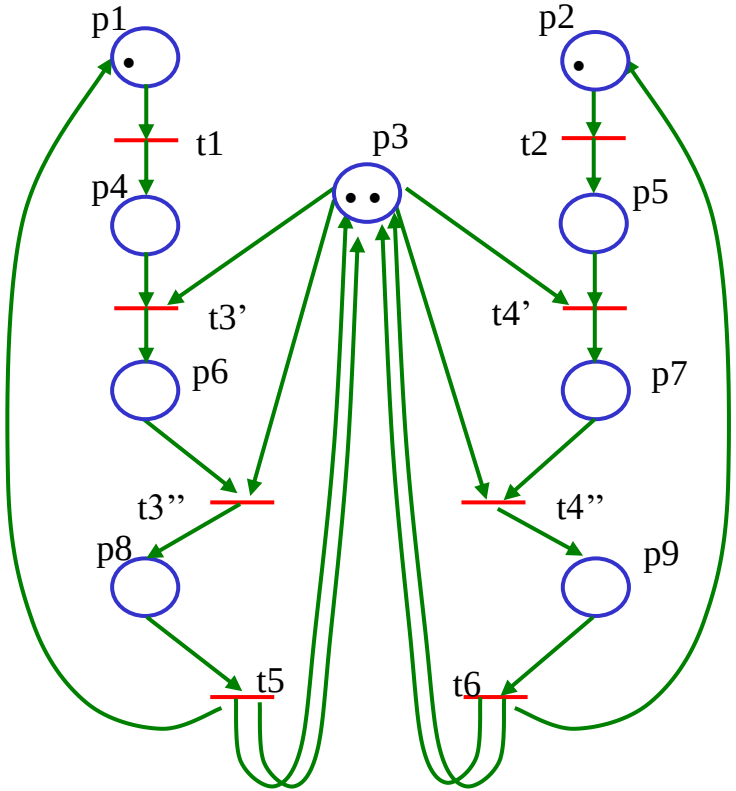
secuencia de disparos:

(t2,t4,t6,t2,t4,t6,...)

⇓

starvation de proceso 1

Modelado con Redes de Petri - Deadlock



t1: proc1 solicita ambos recursos

t3': proc1 toma un recurso

t3'': proc1 toma otro recurso

t5: proc1 libera ambos recursos

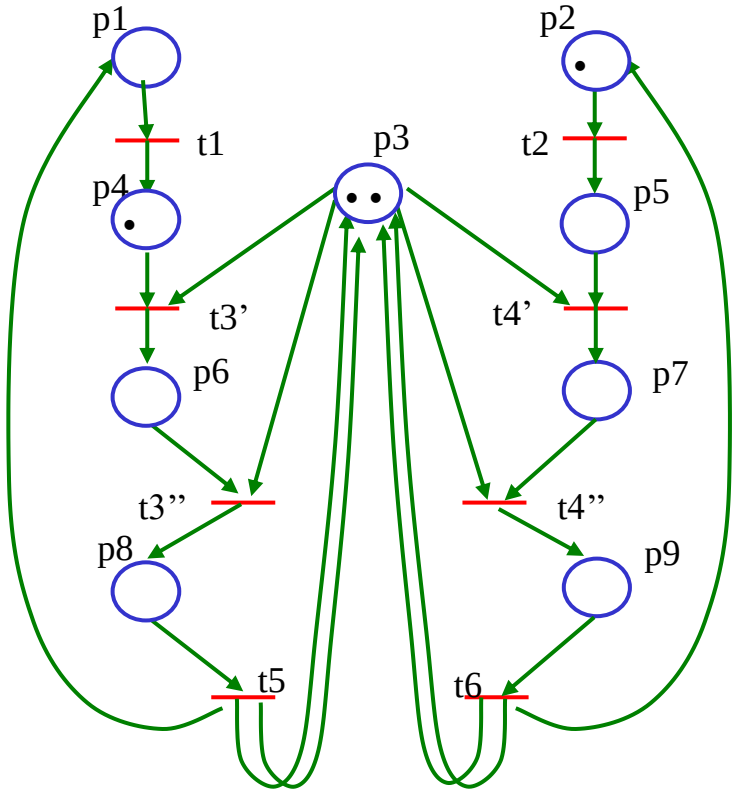
t2: proc2 solicta ambos recursos

t4': proc2 toma un recurso

t4'': proc2 toma otro recurso

t6: proc2 libera ambos recursos

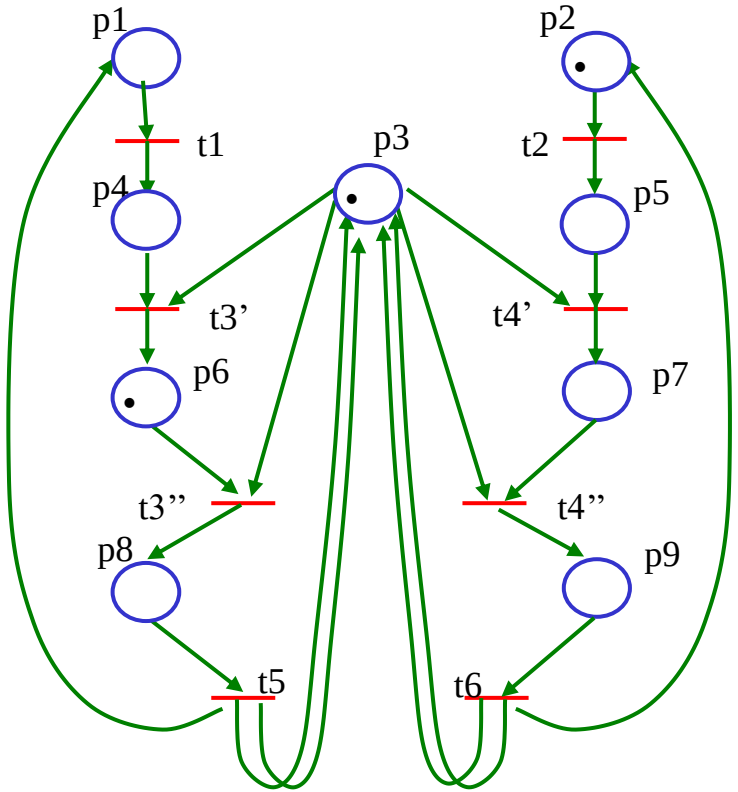
Modelado con Redes de Petri
Una posible ejecución sin deadlock



secuencia de disparos:

(t1)

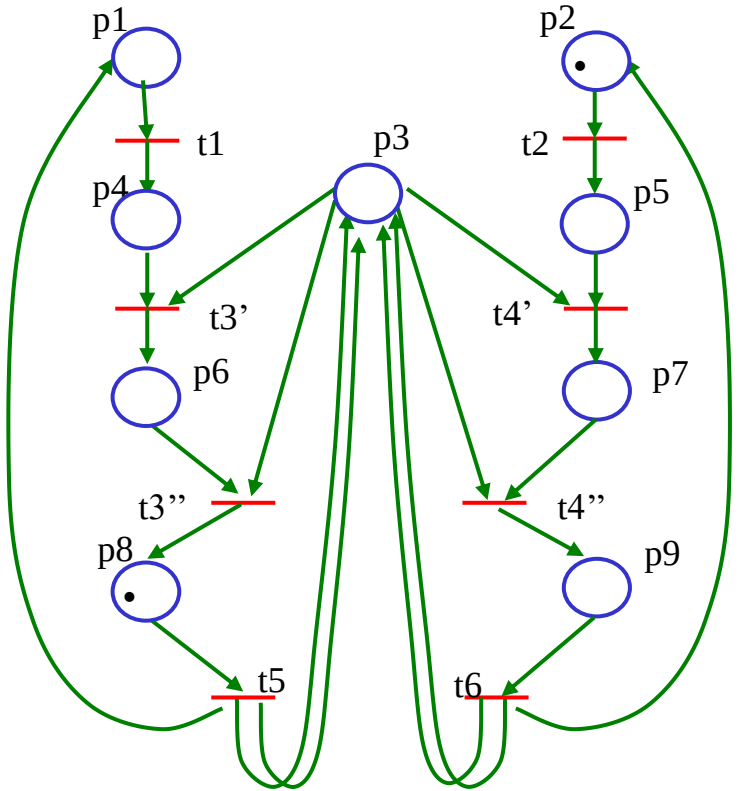
Modelado con Redes de Petri
Una posible ejecución sin deadlock



secuencia de disparos:

(t1, t3')

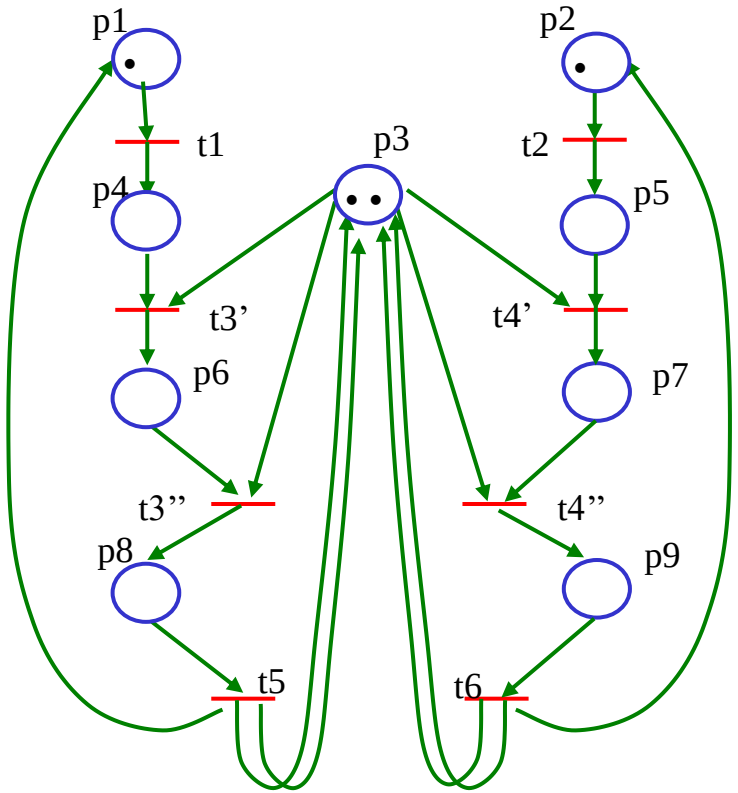
Modelado con Redes de Petri
Una posible ejecución sin deadlock



secuencia de disparos:

(t1, t3', t3'')

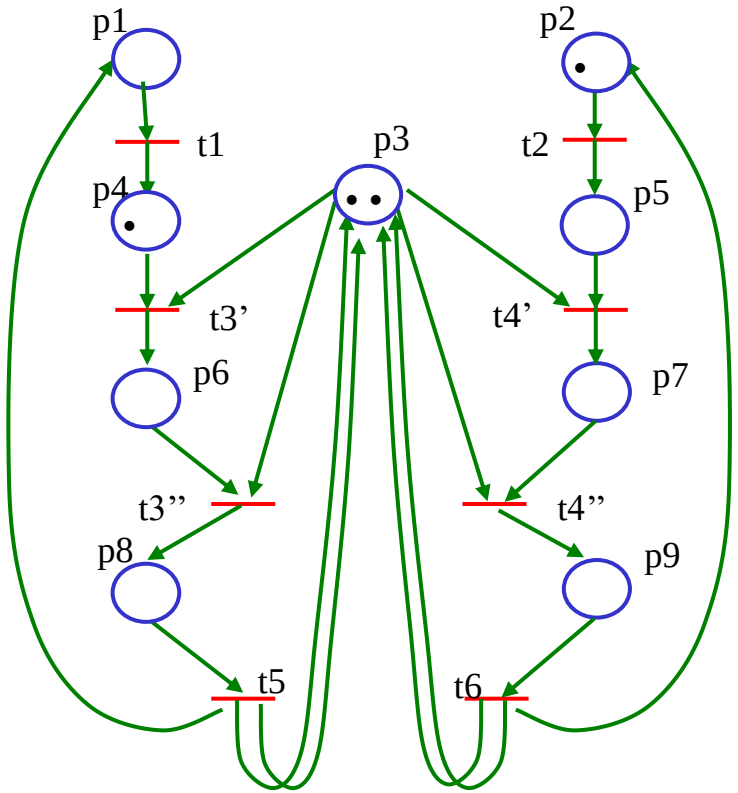
Modelado con Redes de Petri
Una posible ejecución sin deadlock



secuencia de disparos:

(t1, t3', t3'', t5)

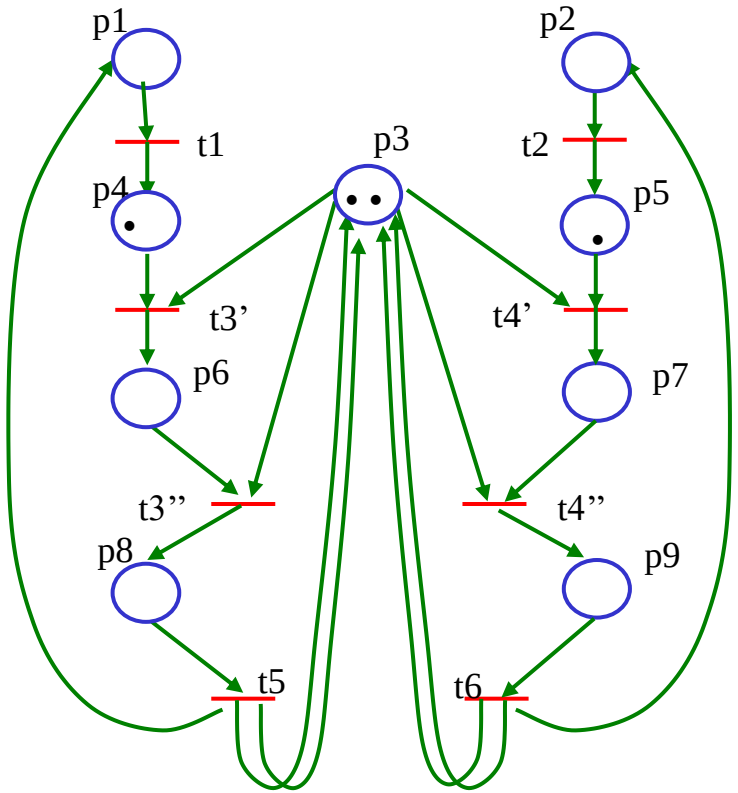
Modelado con Redes de Petri
Una posible ejecución con deadlock



secuencia de disparos:

(t1)

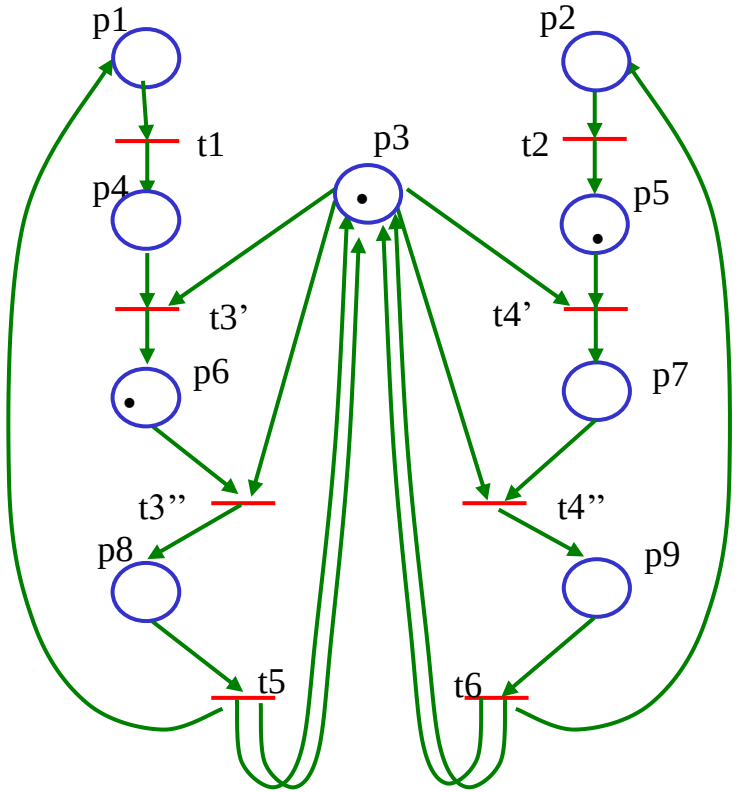
Modelado con Redes de Petri
Una posible ejecución con deadlock



secuencia de disparos:

(t1, t2)

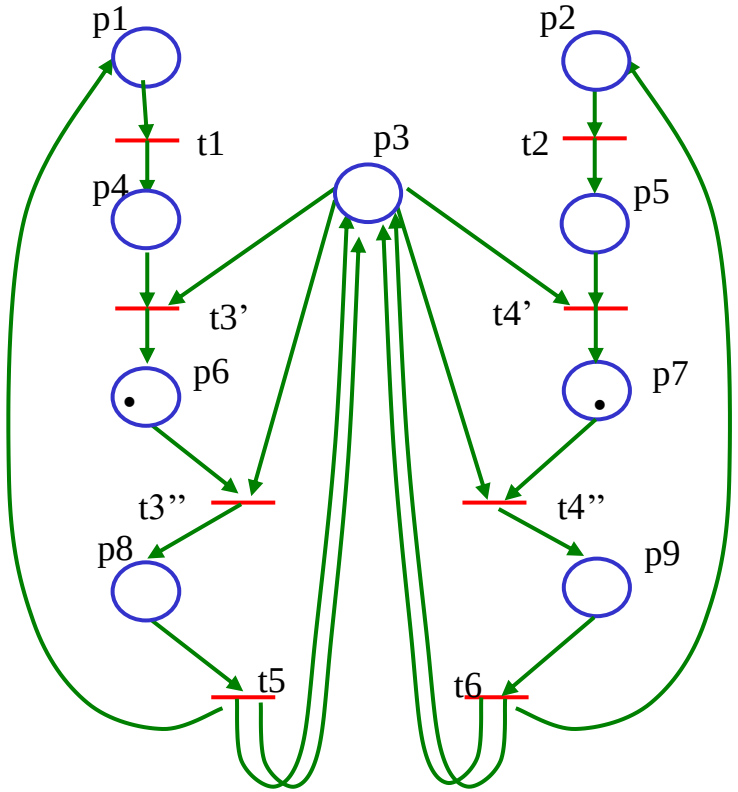
Modelado con Redes de Petri
Una posible ejecución con deadlock



secuencia de disparos:

(t1, t2, t3')

Modelado con Redes de Petri
Una posible ejecución con deadlock

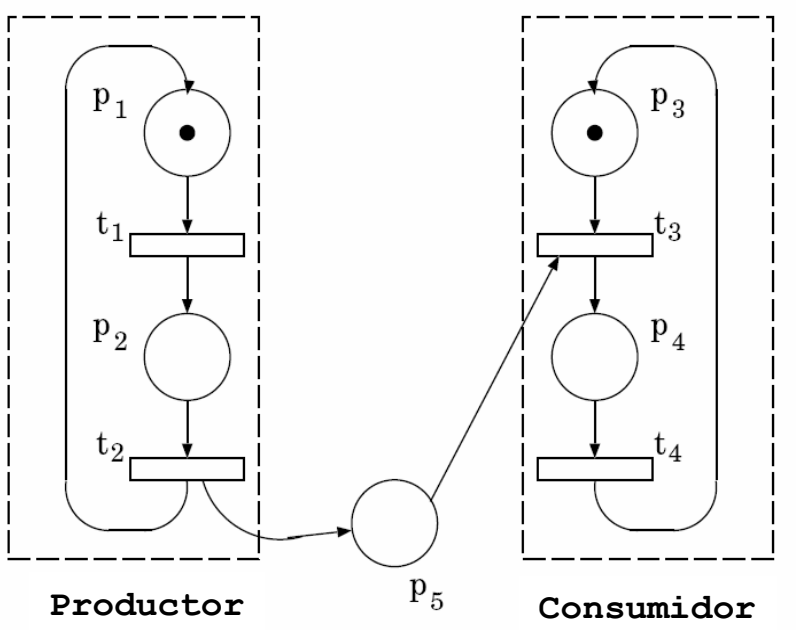


secuencia de disparos:

(t1, t2, t3', t4')

deadlock (ninguno de los procesos puede continuar)

PN para Productor-Consumidor



P_1: Dispuesto a producir
T_1: Produce elemento
P_2: Dispuesto a entregar
T_2: Entrega elemento
P_3: Dispuesto a recibir
T_3: Recibe elemento
P_4: Dispuesto a consumir
T_4: Consume elemento
P_5: Buffer

Muchas gracias por tu tiempo y participación