Lección 2: Conceptos básicos de *PC*

- Procesos y programas concurrentes
- Aplicaciones de la programación concurrente
- Sincronización
- Propiedades de un programa
- Especificación de algoritmos
- Verificación de propiedades de un programa
- Una manera de modelar y analizar programas concurrentes

Procesos y programas concurrentes

- **Proceso**: programa secuencial que ejecuta una secuencia de acciones
- **Programa concurrente**: programa en el que intervienen dos o más procesos secuenciales que cooperan en la realización de una tarea
- **Cooperar** implica **comunicar**
  - mediante memoria compartida
  - mediante paso de mensajes

Procesos y programas concurrentes

- ¿Qué hay de "distinto"?
  - problemas de **interferencia**
- Ejemplo:

```
Vars x:Ent := 0

x := x + 1
--x = 1
```

```
    Vars x:Ent := 0

x := x + 1  ||   x := x + 1
        --x =?????
```

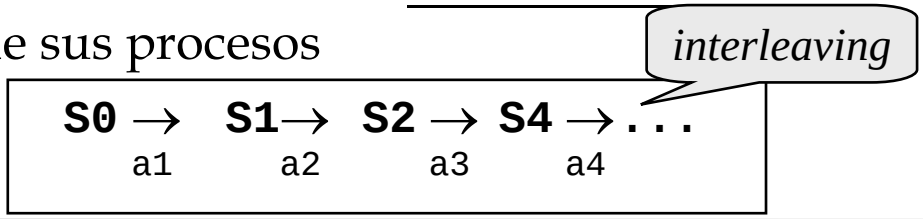
- El diseño de programas concurrentes es complejo:
  - Conveniencia de conocer **métodos formales** para el diseño y verificación de programas concurrentes

Algunas aplicaciones de la PC

- En todas partes:
  - Sistemas operativos
  - Bases de datos multiusuario
  - Internet
  - Algoritmos de cálculo vectorial (programas paralelos)
  - Sistemas de fabricación
  - Sistemas distribuídos
  - ....

Sobre la sincronización

- **Estado de un programa secuencial**: tupla de valores y contador de programa
- **Estado de un programa concurrente**: tupla de los estados de los procesos que lo componen
- **Acción de un proceso**: secuencia de **acciones atómicas**
  - transformaciones de estado indivisibles
    - durante su ejecución, ningún estado intermedio es visible
    - carga o descarga de registros, operaciones elementales de cálculo en la CPU, etc.
- **Ejecución de un programa concurrente**: un entrelazado de las acciones atómicas de sus procesos



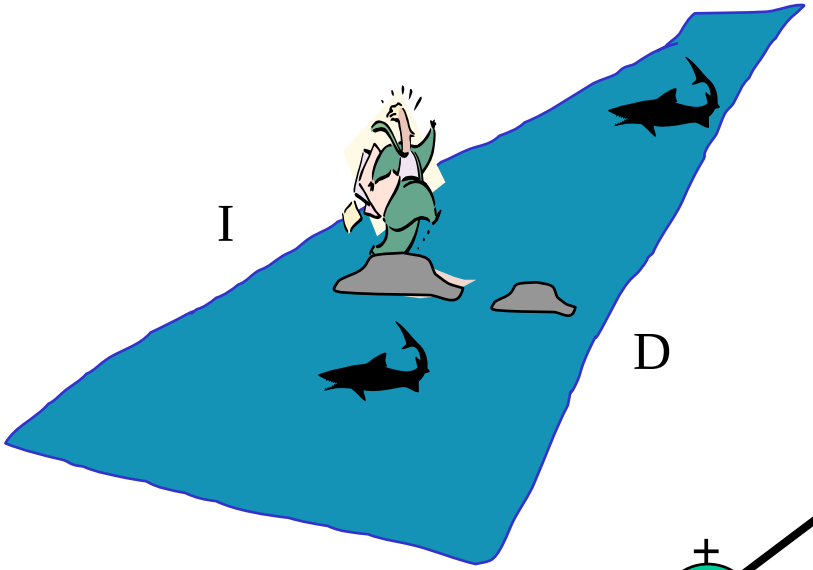
Sobre la sincronización

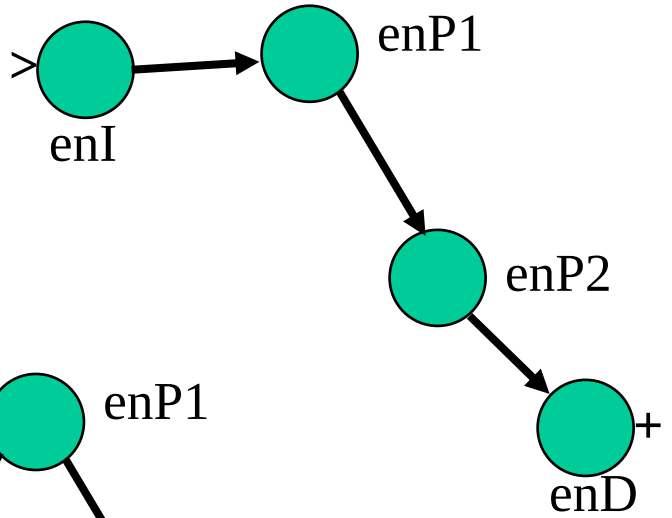
- Cada ejecución de un programa concurrente define una **historia**
  - el número de historias diferentes posibles puede ser enorme
- La **sincronización** entre procesos
  - restringe el número de historias posibles
  - permite evitar las no deseables
    - problemas de bloqueo, de inanición, etc.
    - escasa utilización de recursos eficientes, mal balance en el uso, etc.

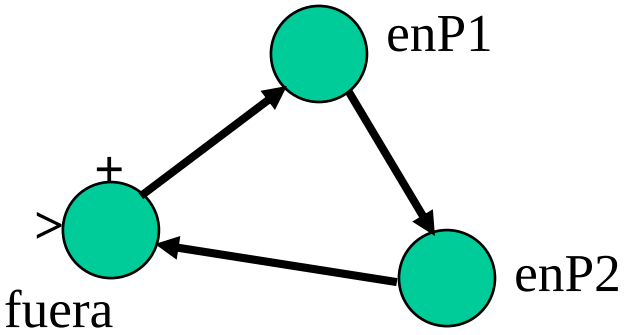
Propiedades de un Programa

- **Propiedad de un programa**: atributo cierto para cualquier posible historia del programa
- Básicamente, dos clases de propiedades:
  - Propiedades de **seguridad**: el programa nunca alcanza un "mal" estado
    - corrección parcial, exclusión mutua y ausencia de bloqueos
  - Propiedades de **vivacidad**: algo “bueno” ocurrirá
    - terminación, equidad
    - dependen en gran medida de la política de “scheduling”
- **Ejemplo**: la corrección total de un programa (secuencial o concurrente), combina corrección parcial (seguridad) y terminación (vivacidad)

Una manera de modelar sistemas concurrentes



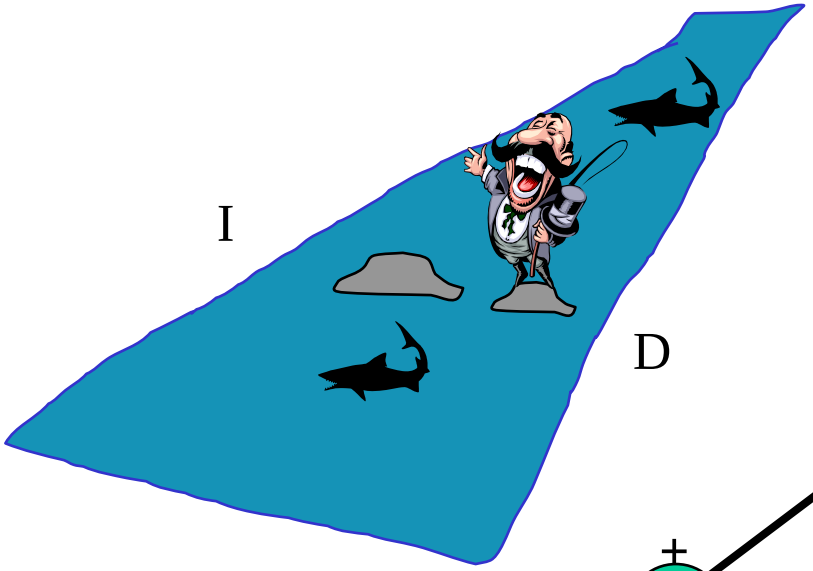


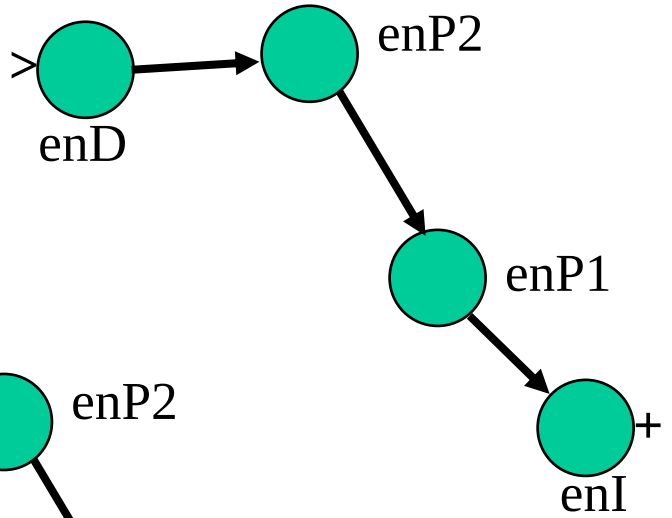


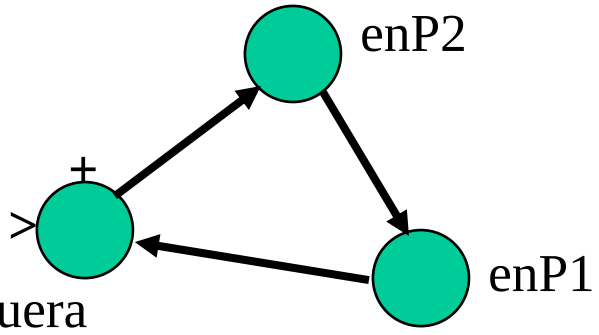
Una manera de modelar sistemas concurrentes

- **Modelo**: abstracción de la realidad
  - visión simplificada
  - centrada en determinados aspectos
- Un modelo se debe validar
- Un modelo debe servir para:
  - entender el sistema real
  - inferir propiedades del sistema a partir de propiedades del modelo

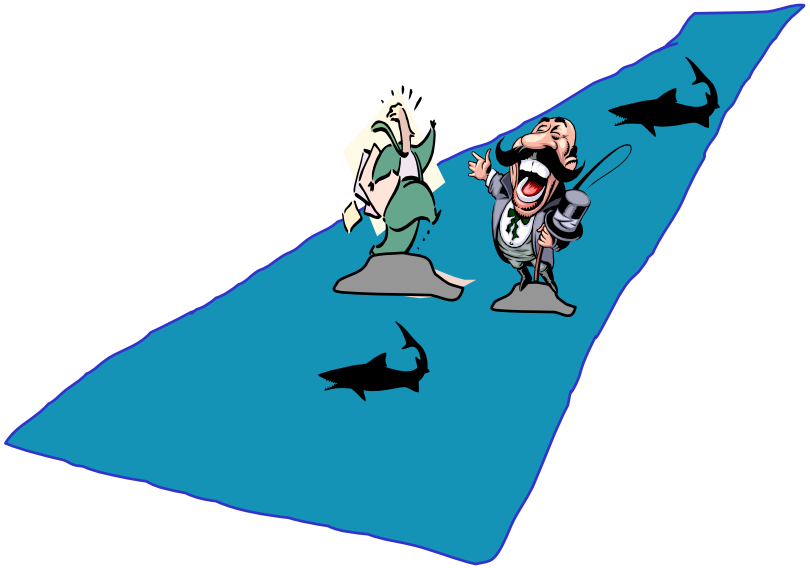
Una manera de modelar sistemas concurrentes



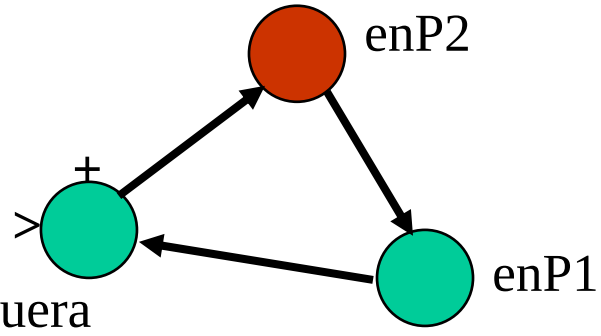


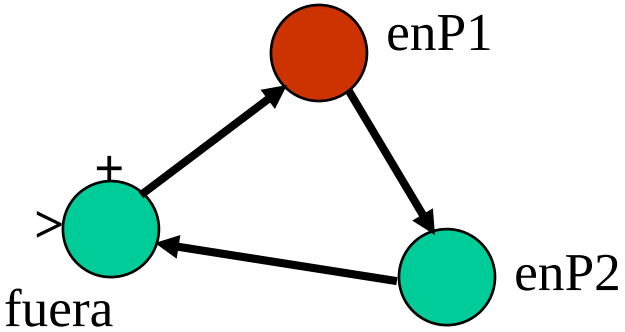


Una manera de modelar sistemas concurrentes

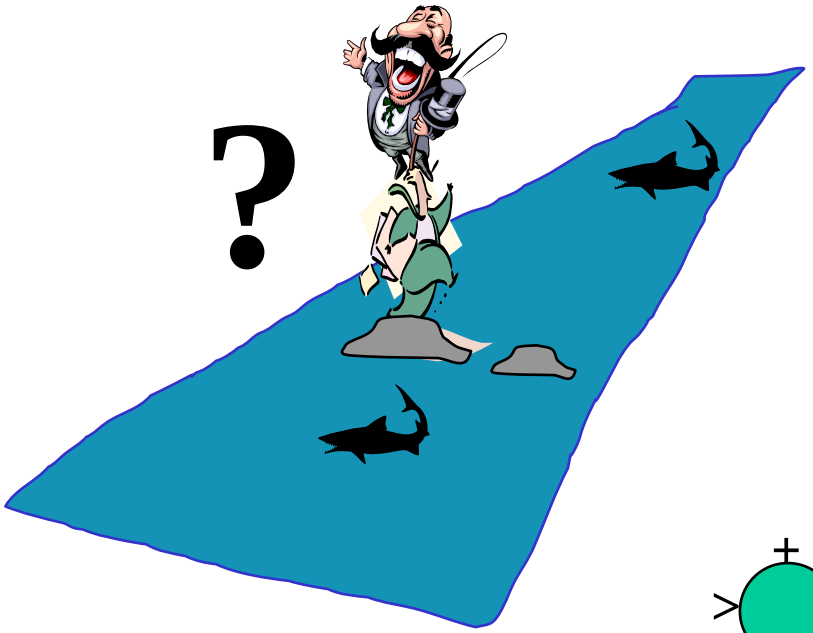


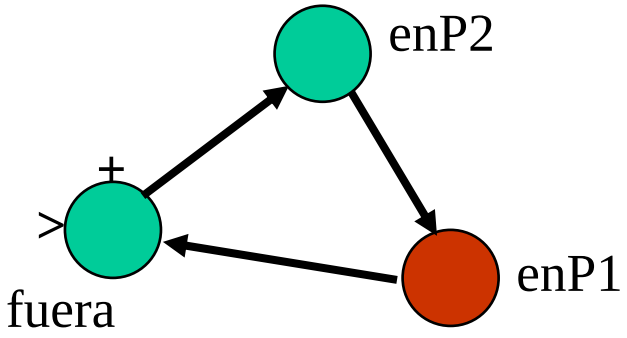
Bloqueo

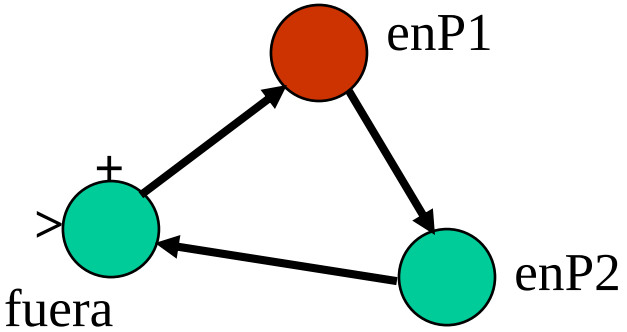




Una manera de modelar sistemas concurrentes

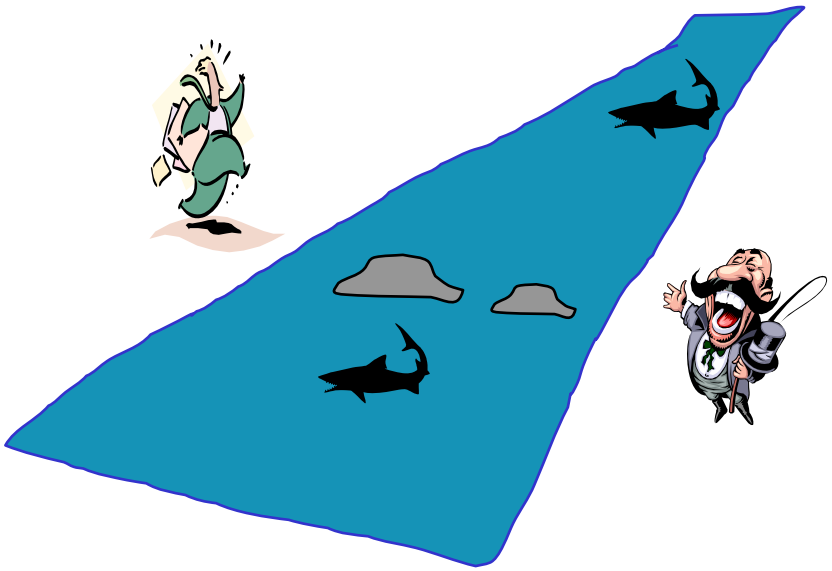


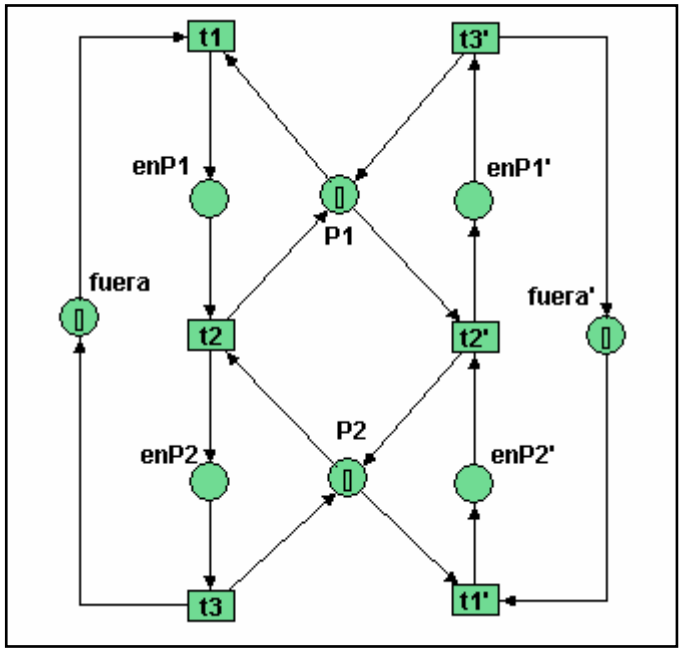




Exclusión mutua

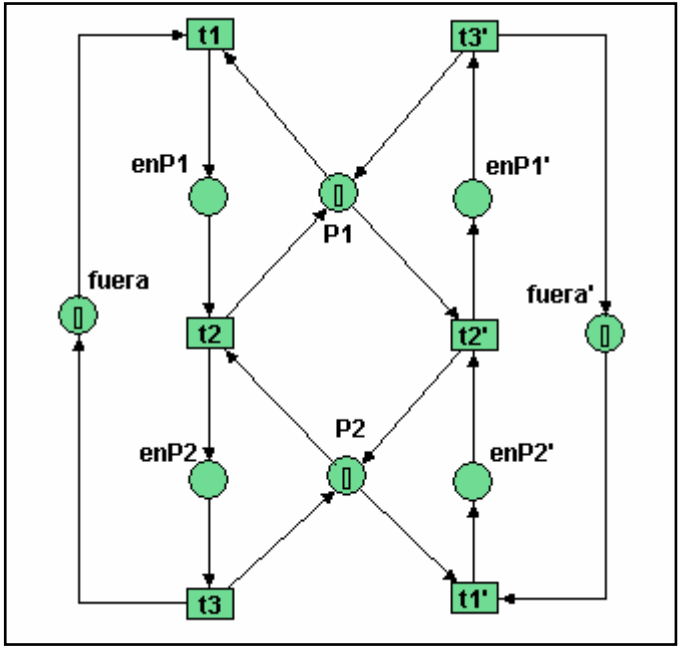
Una manera de modelar sistemas concurrentes



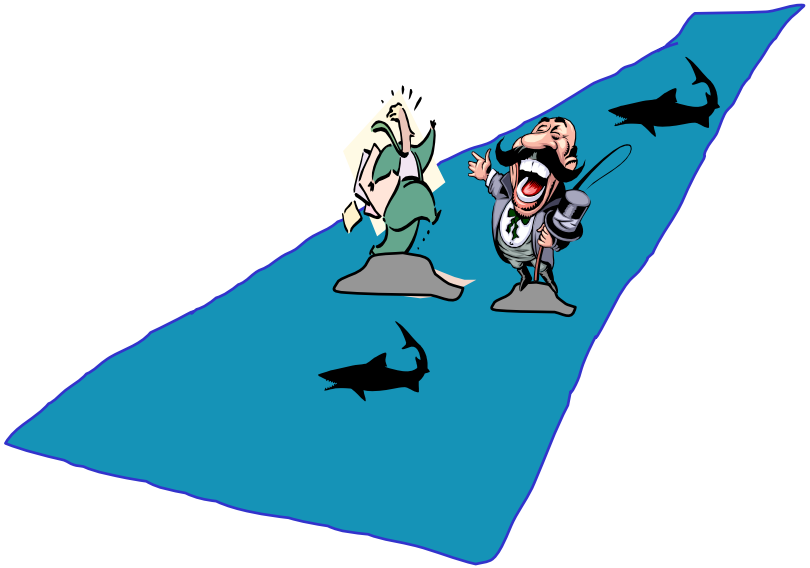


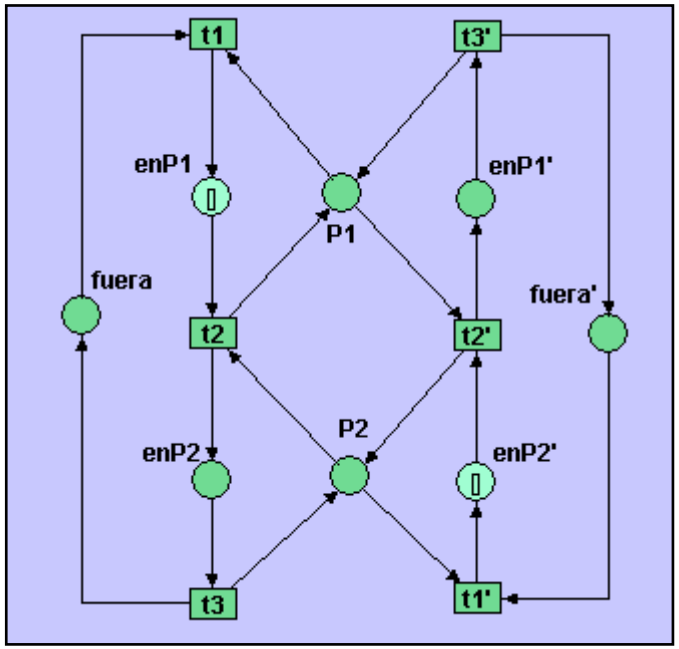
Una manera de modelar sistemas concurrentes

- Recordatorio de conceptos básicos y mínimos sobre redes de Petri:
  - lugar
  - transición
  - marcado
  - transición sensibilizada
  - disparo de transición
  - redes ordinarias
  - redes coloreadas

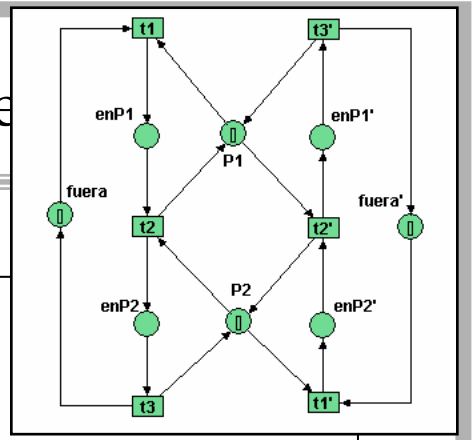


Una manera de modelar sistemas concurrentes

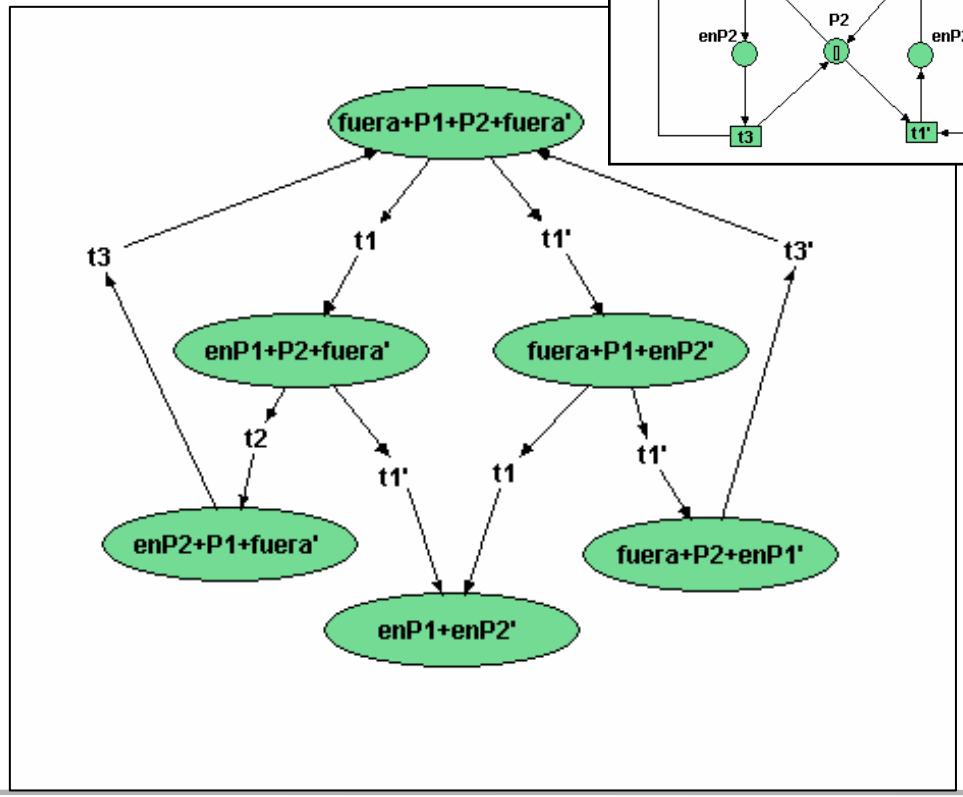




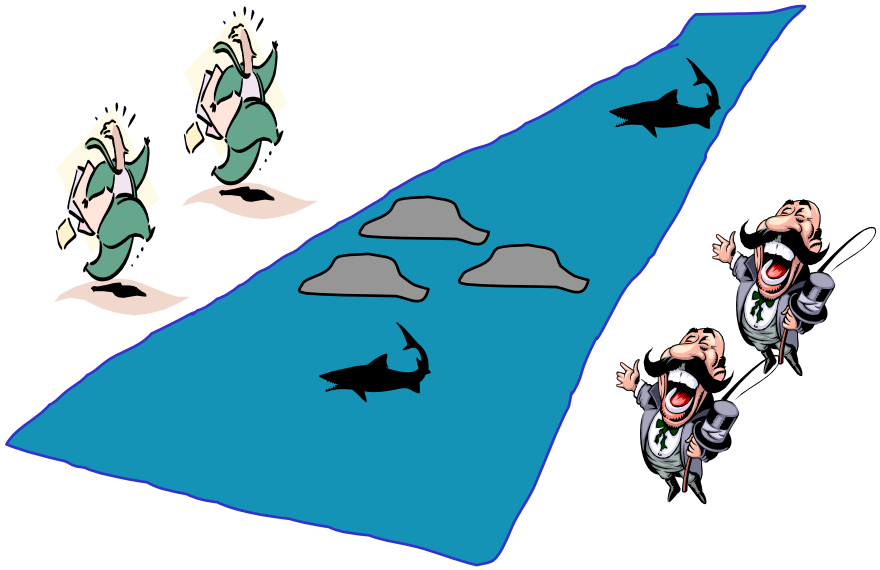
Una manera de modelar sistemas concurre

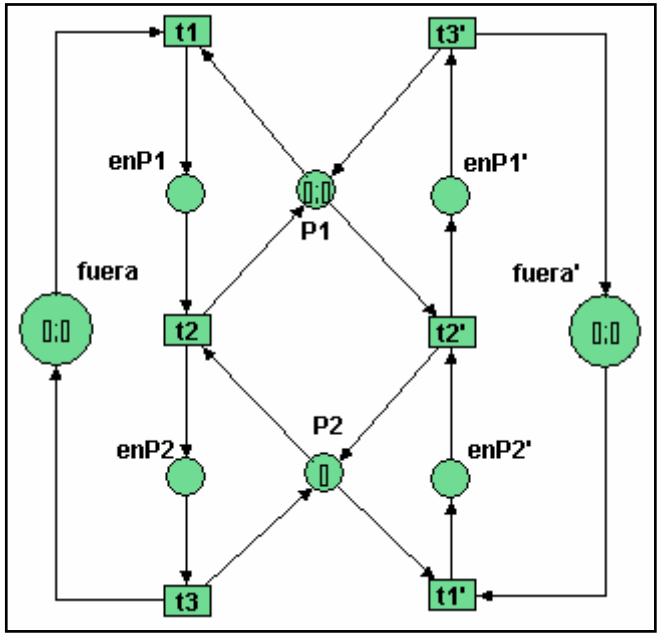


- Grafo de **estados alcanzables** del ejemplo anterior
- Estudio de propiedades:
  - repetitividad
  - ausencia de bloqueos totales
  - ausencia de bloqueos parciales
  - vivacidad
  - equidad/inanición



Una manera de modelar sistemas concurrentes

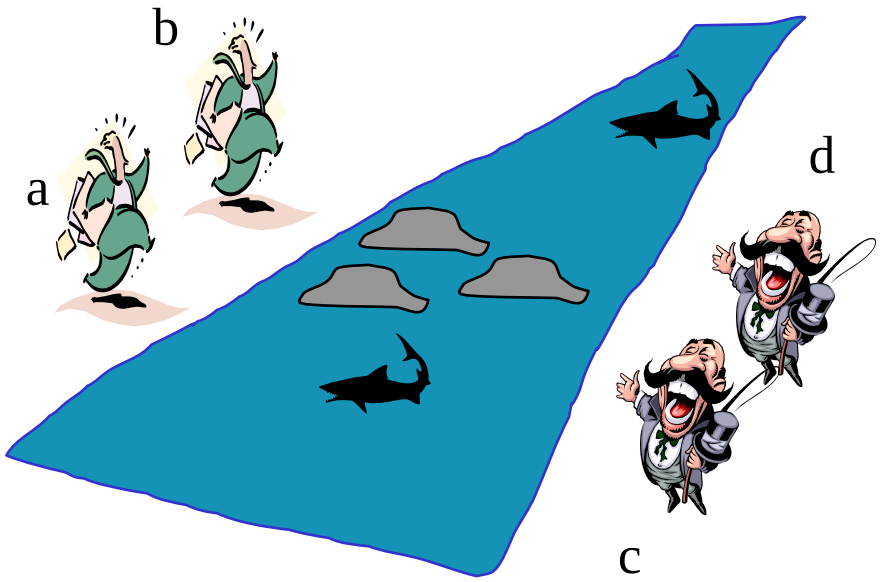


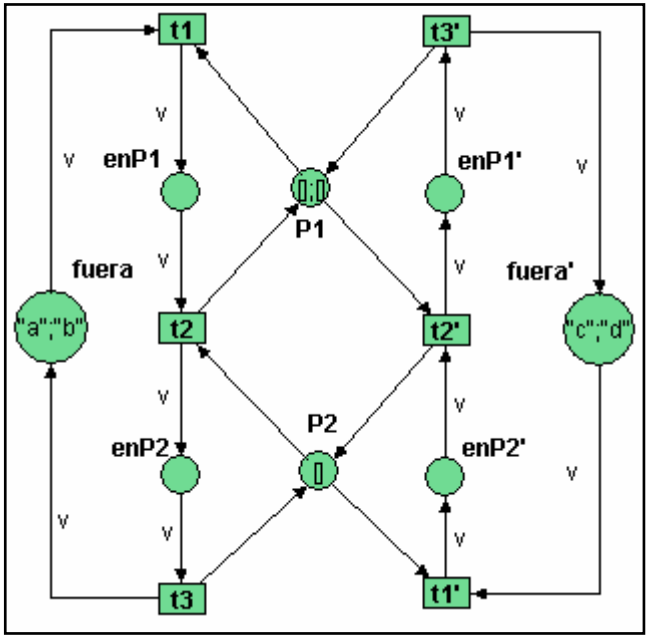


Una manera de modelar sistemas concurrentes

- Ejercicio:
  - modelar el sistema asumiendo que queremos distinguir la identidad de los transeúntes, pero no distinguir cada una de las dos piedras cercanas a la orilla izquierda
  - modelar el sistema asumiendo que queremos distinguir las dos piedras de la orilla izquierda, pero no nos interesa la identidad de los transeúntes
  - modelar el sistema asumiendo que queremos distinguir la identidad de los transeúntes y también queremos distinguir las dos piedras de la orilla izquierda

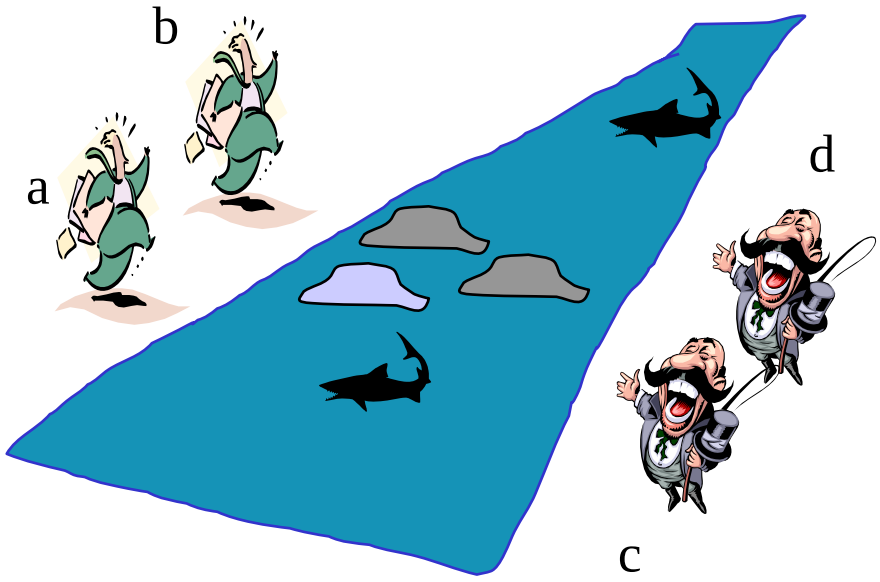
Una manera de modelar sistemas concurrentes

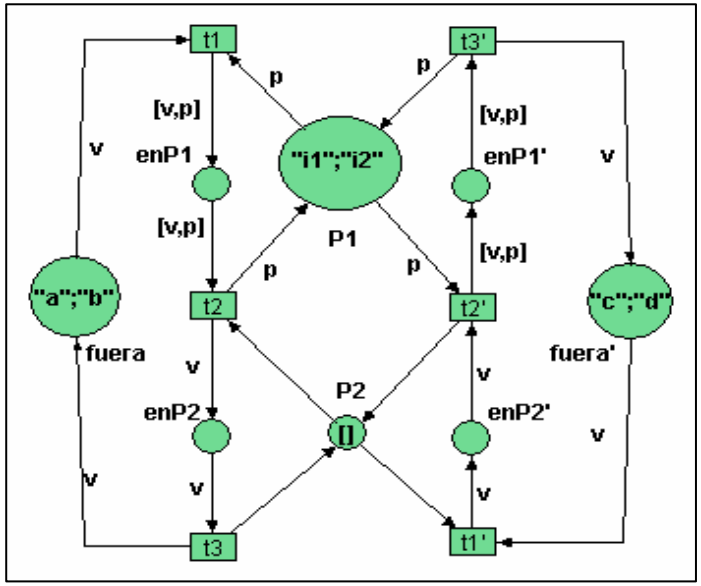




Una manera de modelar sistemas concurrentes

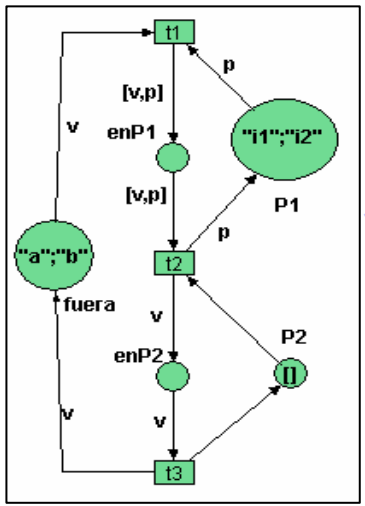
- Extendamos el sistema
  - identificando, además, las piedras de la izquierda

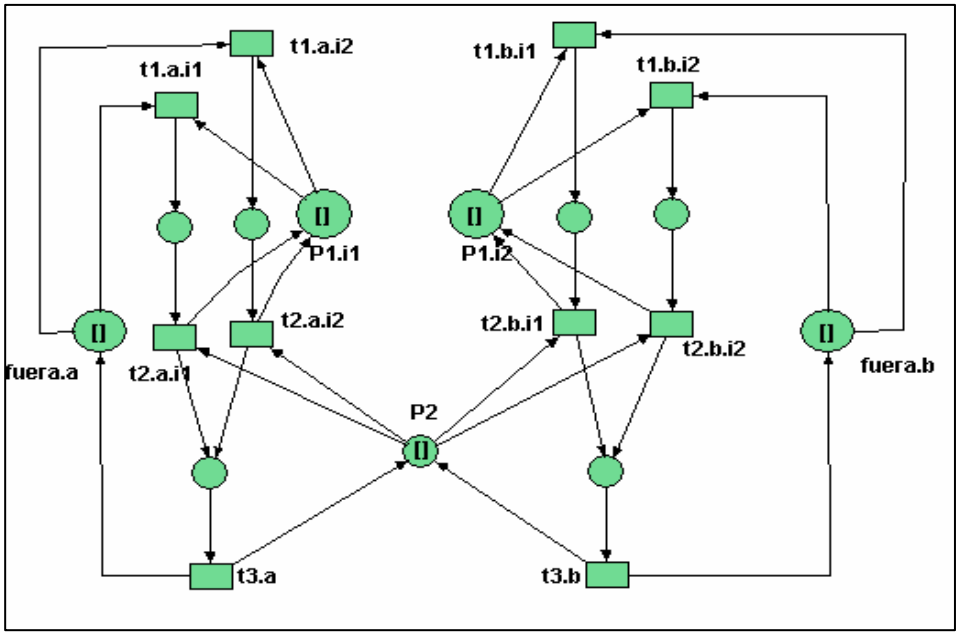




Una manera de modelar sistemas concurrentes

- En coloreadas hay una equivalente, pero...
- El objetivo es hacer un tratamiento simbólico
- Útil y complicado





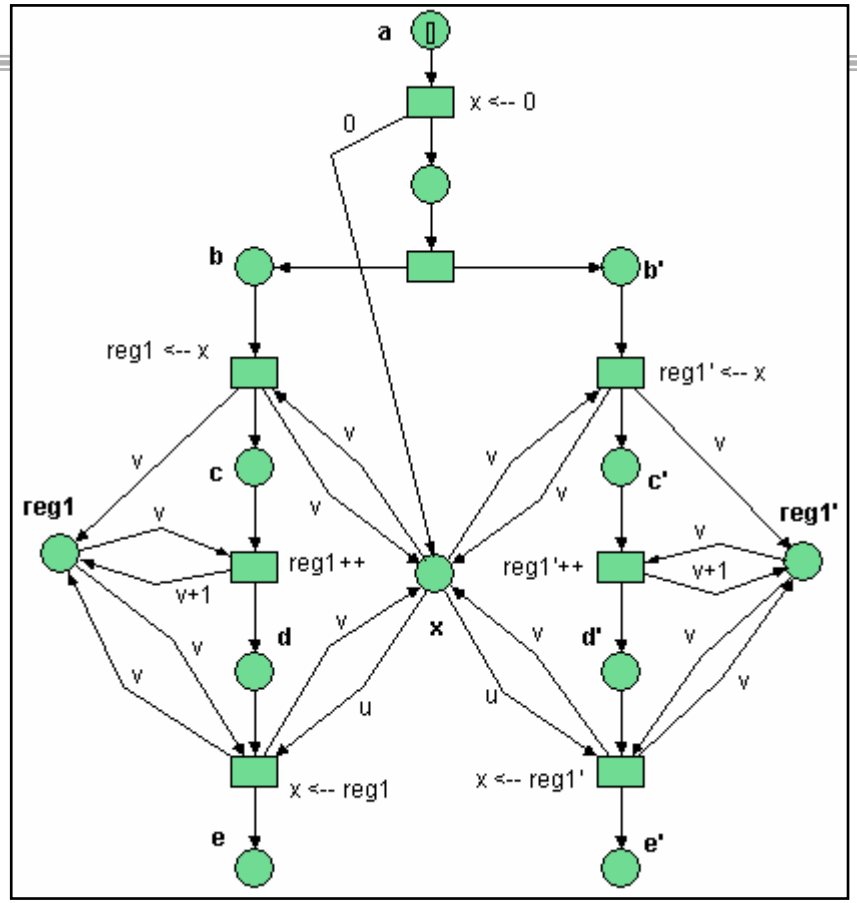
Un primer programa

```
Vars x:Ent := 0

x := x + 1  ||  x := x + 1
      --x =?????
```

```
x := x + 1


   reg1 <-- x
   reg1++
   x <-- reg1
```



Un primer programa