Práctica PetriDesign
Proyecto: Diseño de un sistema concurrentes.
Autor: José M. Drake
Fecha del documento: 19-12-11

Objetivo:

Esta práctica propone concebir y realizar el diseño detallado de un sistema concurrente a partir de su especificación.

Se diseña una aplicación para el control de cualquier entorno físico a partir de una descripción de la estrategia reactiva de control formulada como una red de Petri (RdP).

Haciendo uso de UML (no escribir código Java) hay que proponer un conjunto de clases que implementen la funcionalidad de los elementos propios de las redes de Petri (transición, plaza, testigo) y de un entorno físico (eventos, acciones). Estos elementos deben estar concebidos a fin de que se construya automáticamente el programa que controle el sistema a partir de la descripción de la red de Petri y del entorno.

El diseño debe quedar completamente documentado, a fin de que un programador (sin capacidad de iniciativa de diseño pueda escribir su código Java completo).

A cada grupo de prácticas se le establecerán ciertos criterios específicos de diseño.

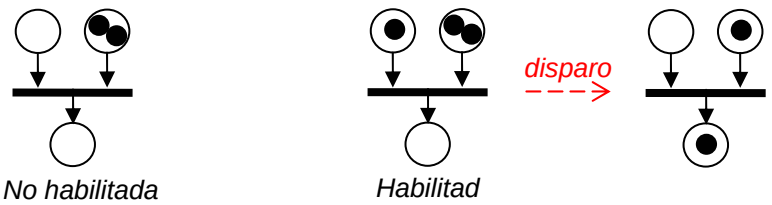
Implementación de redes de Petri para el control de un entorno físico.

Red de Petri (RdP)

Una RdP es una abstracción concebida para modelar sistemas basados en eventos discretos, que en este trabajo se quiere utilizar para desarrollar aplicaciones Java concurrentes de control de un entorno físico.

Los elementos definidos en una RdP son los siguientes:

Transición (Transition): Una transición representa un elemento que está asociado en la RdP a un conjunto de plazas de entrada y a un conjunto de plazas de salida. Una transición se dispara de forma autónoma cuando en todas sus plazas de entrada existen al menos un testigo. El disparo de la transición extrae de cada plaza de entrada un testigo y deposita en cada plaza de salida un nuevo testigo. La ejecución de una transición supone el cambio de estado del sistema.



Plaza(place): Es un elemento contendor de la RdP que puede estar vacía o contener un conjunto de testigos. Cada plaza puede estar conectada en la RdP como salida de una transición y en ese caso recibe un nuevo testigo cada vez que esa transición se dispara. Así mismo, una plaza puede esta conectada en la RdP como una salida de una transición, y en ese caso, el que disponga de testigo es una condición para que esa transición se pueda disparar. Cuando esto ocurre se extrae un testigo de la plaza. Todos los testigos de una plaza son del mismo tipo (representan la misma situación), y el número de testigos que contienen el conjunto de todas las plazas de la RdP representan el estado del sistema.

Testigo(Tocken): Representa la ocurrencia de un evento de sincronización y no lleva asociados ninguna información. Los testigos se almacenan en las plazas. Los eventos se generan y se eliminan como efecto del disparo de las transiciones. Entre los eventos que se eliminan por el disparo de una transición, y los eventos que se genera por la misma no hay ninguna relación.

Red de Petri (RdP): Es un grafo cuyos nudos son transiciones o plazas, y los arcos del grafo relacionan transiciones con plazas. Una red de Petri representa completamente una aplicación.

Entorno físico

El entorno físico (sistema) es una infraestructura compuesta por equipos, con sensores y actuadores que realizan algún tipo de proceso industrial (mecánico, eléctrico, químico, etc.). El sistema evoluciona de acuerdo con su dinámica propia. El control del sistema se realiza en base a una estrategia reactiva, basada en señales y acciones:

Señal: Es una variable booleana que indica que el sistema se encuentra en un estado interno relevante. El estado de la señal puede ser consultado.

Evento: Es un aviso que se genera cuando una señal cambia de estado, esto es, cuando el sistema alcanza o abandona un estado relevante señalizado.

Acción: Es una actuación instantánea que se ejecuta en el sistema para cambiar su estado.

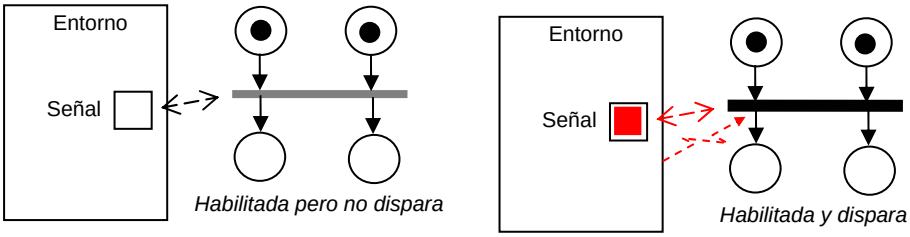
Estrategia de control: Consiste en gobernar el funcionamiento del sistema asociando a cada evento una respuesta compuesta por diferentes acciones.

Interacciones entre el sistema y la RdP de control.

Se pueden utilizar diferentes opciones de interacción entre el sistema físico y la RdP que implementa su estrategia de control. En cada una de estas opciones se formaliza como los eventos del sistema afectan el estado de la RdP, y como los cambios de estado de la RdP se traducen en acciones que se ejecutan en el sistema.

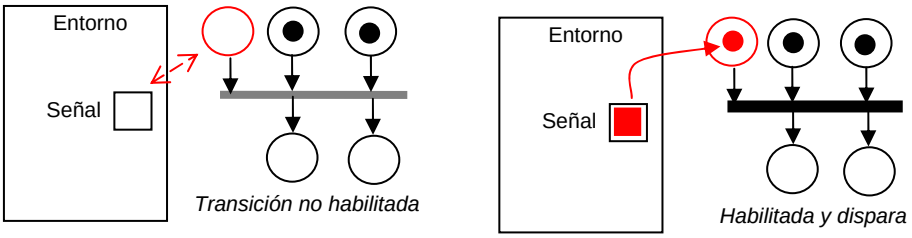
Opciones de comunicación de eventos:

1. *El disparo de una transición se condiciona con el estado de una señal del sistema*: En este caso cuando la transición está habilitada por el estado de las plazas de entrad de la RdP, sólo dispara si está establecida la señal del sistema a la que está asociada.



Para implementar esta opción a cada transición de la RdP se asocia (o nó) una señal del entorno que debe ser verificada para que la transición se pueda disparar. La transacción puede consultar el estado de la señal asociada, y el entorno puede enviar un evento cuando el estado de la señal cambia.

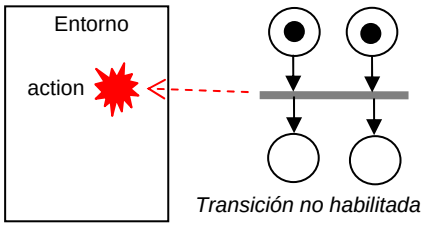
2. *El evento de establecimiento de una señal del sistema se traduce en la introducción de un testigo en una plaza*: Cuando una señal se establece en el sistema, en la plaza de la RdP asociada a ella se introduce un nuevo testigo. Este testigo es creado e introducido por el entorno, y no por el disparo de una transición.



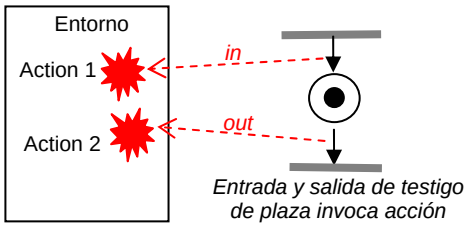
Para implementar esta opción a cada señal del sistema se le asocia una plaza de la RdP, en la que se introduce un testigo con cada evento de establecimiento de la señal.

Opciones de ejecución de acciones:

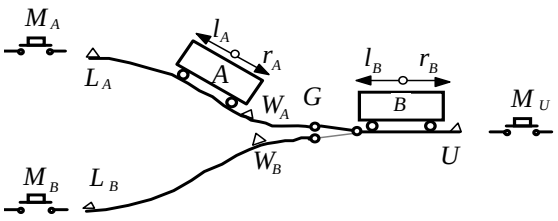
1. *La acción se asocia al disparo de una transición*: A ciertas transiciones del la RdP se le asocia una acción. Cuando la transición se dispara, la acción se ejecuta sobre el sistema.



2. *La acción se asocia a la entrada o a la salida de un testigo en una plaza*: A ciertas plazas de la RdP se le asocian acciones que se ejecutan cuando en la plaza entra un evento y/o cuando de ella sale un evento.



CoalMine: Ejemplo de control de un sistema de transporte.



Se trata de controlar el sistema de transporte de una mina que se muestra en la figura. Se controlan los movimientos de los vagones (avance hacia la derecha (r), avance hacia la izquierda (l) y parada (s), en base a las señales que se establecen con los pulsadores en las estaciones terminales (M_A,M_B y M_U), y de los sensores de paso por determinados puntos de las vías (L_A,L_B,W_A,W_B y U). En el recorrido de los vagones hay una sección común, a la que se debe acceder con exclusión mútua.

La estrategia de control de este sistema se realiza con la lógica del la RdP que se muestra en la siguiente figura.

En este sistema se disponen de las siguientes señales del entorno:

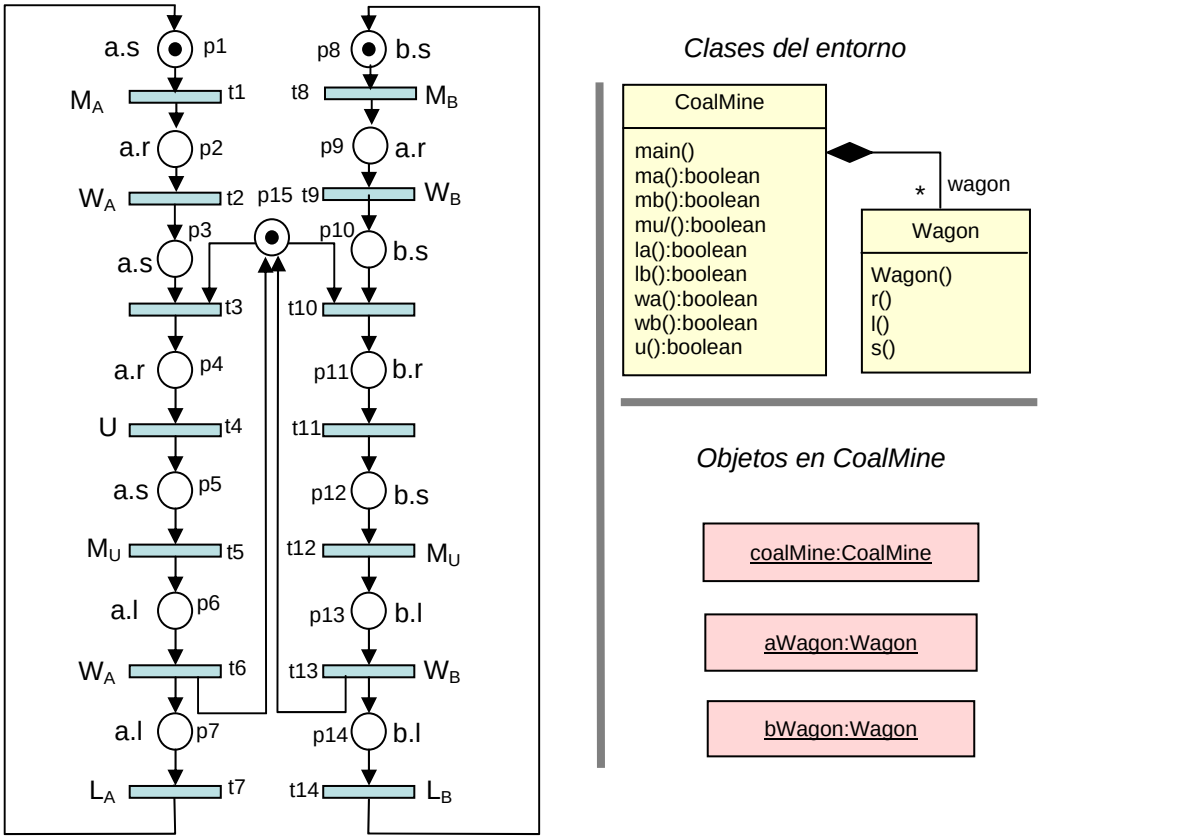
- **M_A** y **M_B** : Se ha pulsado el botón para que el vagón correspondiente salga de su terminal izquierdo.
- **M_U**: Se ha pulsado el botón para que el vagón presente abandone el terminal derecho.
- **L_A** y **L_B**: Sensor de la vía que indica que el correpondiente vagón ha alcanzado la estación del extremo izquierdo.
- **W_A** y **W_B**: Sensor de la vía que indica que el vagón correspondiente ha alcanzado el punto de bufurcación de las vías
- **U**: Sensor de la vía que indica que un vagón ha alcanzado la estación derecha del recorrido.

Las acciones que están definidas en el sistema son:

- **a.r** y **b.r**: El vagón a o b inicia su marcha hacia la derecha.
- **a.l** y **b.l**: El vagón a o b inicia su marcha hacia la izquierda.
- **a.s** y **b.s**: El vagón a o b se para.

A modo de ejemplo, en la Tabla I se muestra una posible descripción del sistema de control en un fichero XML. Este ejemplo corresponde a las opciones de asociar las señales a condiciones de disparo de las transacciones, y las acciones al disparo de transiciones.

El ejemplo se ha formulado en base a la existencia de un modelo de la infraestructura con dos tipos de elementos CoalMine, y Wagon, con las instancias tres instancias que se muestran en la siguiente figura.



El objetivo de diseño que se propone es diseñar un conjunto de clases que corresponden a los elementos básicos de la RdP: RdP, Transicion, Place, Tocken, que permitan construir la aplicación de forma automática por la simple invocación de la aplicación desde el sistema operativo:

RdP.main("CoalMain.xml);

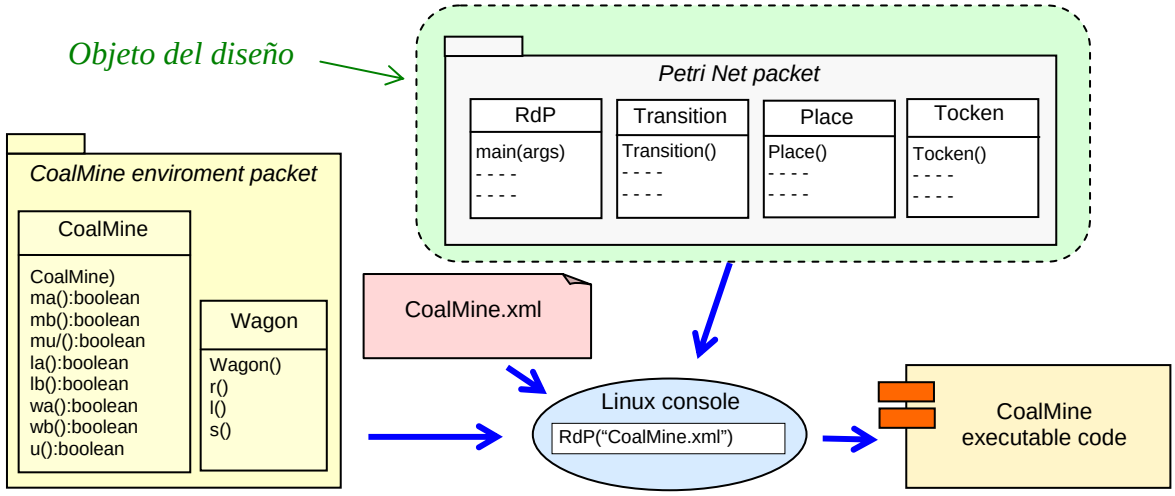


Tabla I: Fichero "CoalMineDescr.xml" con la descripción de la RdP que constituye la aplicación de control de la aplicación CoalMine.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--************************************************
 Programacion Concurrente y distribuido
 Grupo de Computadores y tiempo real (CTR)

File: Descripción del ejemplo CoalMine
Authors:: J.M. Drake
Version: 18-12-2011
************************************************-->
<PETRI_DESCRIPTION
                        Sistem="PowerDistortion"
                        Model_Date="2011-12-18">
<place name="p1" initMark="1"/>
<place name="p2" initMark="0"/>
<place name="p3" initMark="0"/>
<place name="p4" initMark="0"/>
<place name="p5" initMark="0"/>
<place name="p6" initMark="0"/>
<place name="p7" initMark="0"/>
<place name="p8" initMark="1"/>
<place name="p9" initMark="0"/>
<place name="p10" initMark="0"/>
<place name="p11" initMark="0"/>
<place name="p12" initMark="0"/>
<place name="p13" initMark="0"/>
<place name="p14" initMark="0"/>
<place name="p15" initMark="1"/>
<transition name="t1">
            <inputPlace name="p1"/>
            <outputPlace name="p2"/>
            <condition object="coalMine" method="ma"/>
            <action object="aWagon" method="r"/>
</transition>
<transition name="t2">
            <inputPlace name="p2"/>
            <outputPlace name="p3"/>
            <condition object="coalMine" method="wa"/>
            <action object="aWagon" method="s"/>
</transition>
<transition name="t3">
            <inputPlace name="p3"/>
            <inputPlace name="p15"/>
            <outputPlace name="p4"/>
            <action object="aWagon" method="r"/>
</transition>
<transition name="t4">
            <inputPlace name="p4"/>
            <outputPlace name="p5"/>
            <condition object="coalMine" method="u"/>
            <action object="aWagon" method="s"/>
</transition>
<transition name="t5">
            <inputPlace name="p5"/>
            <outputPlace name="p6"/>
            <condition object="coalMine" method="mu"/>
            <action object="aWagon" method="l"/>
</transition>
<transition name="t6">
            <inputPlace name="p6"/>
            <outputPlace name="p7"/>
            <outputPlace name="p15"/>
            <condition object="coalMine" method="wa"/>
</transition>
<transition name="t7">
            <inputPlace name="p7"/>
            <outputPlace name="p1"/>
            <condition object="coalMine" method="la"/>
            <action object="aWagon" method="s"/>
</transition>
<transition name="t8">
            <inputPlace name="p8"/>
            <outputPlace name="p9"/>
            <condition object="coalMine" method="mb"/>
            <action object="bWagon" method="r"/>
</transition>
<transition name="t9">
            <inputPlace name="p9"/>
            <outputPlace name="p10"/>
            <condition object="coalMine" method="wb"/>
            <action object="bWagon" method="s"/>
</transition>
<transition name="t10">
            <inputPlace name="p10"/>
            <inputPlace name="p15"/>
            <outputPlace name="p11"/>
            <action object="bWagon" method="r"/>
</transition>
<transition name="t11">
            <inputPlace name="p11"/>
            <outputPlace name="p12"/>
            <condition object="coalMine" method="u"/>
            <action object="bWagon" method="s"/>
</transition>
<transition name="t12">
            <inputPlace name="p12"/>
            <outputPlace name="p13"/>
            <condition object="coalMine" method="mu"/>
            <action object="bWagon" method="l"/>
</transition>
<transition name="t13">
            <inputPlace name="p13"/>
            <outputPlace name="p14"/>
            <outputPlace name="p15"/>
            <condition object="coalMine" method="wb"/>
</transition>
<transition name="t14">
            <inputPlace name="p14"/>
            <outputPlace name="p8"/>
            <condition object="coalMine" method="lb"/>
            <action object="bWagon" method="s"/>
</transition>
</PETRI_DESCRIPTION>
```

Desarrollo de la práctica.

El objetivo de la práctica es el **diseño** de una aplicación PetriNet que permita controlar un entorno físico utilizando la estrategia de red de Petri. Lanzando la aplicación, pasándole como argumento la descripción de la red de Petri de control, debe ejecutar la aplicación que produce el control.

El diseño debe estar suficientemente detallado para que un programador pudiera escribir el código de las clases sin necesidad de tomar decisiones de diseño.

Para ello se necesita diseñar la interfaz y la funcionalidad de las clases del paquete PetriNet que junto con las clases específicas del entorno constituyan estas aplicaciones. Mientras que las clases que definen el entorno (por ejemplo *CoalMineEnviroment*) son especificas de una aplicación, el paquetes de clases *PetriNet* debe ser genérico y reutilizable en cualquier aplicación (CoalMine, SemaphoreControl,...).

A cada grupo de prácticas se le introducen restricciones adicionales de diseño:

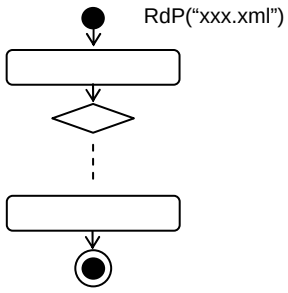
		Acción en I/O de plaza	Acción en disparo transición
La transición es Thread	Condición de disparo	**Grupo 6**	**Grupo1**
	Testigo en plaza	**Grupo 12**	**Grupo 7**
La plaza es Thread	Condición de disparo	**Grupo 2**	**Grupo 11**
	Testigo en plaza	**Grupo 8**	**Grupo 5**
El testigo es Thread	Condición de disparo	**Grupo 10**	**Grupo 3**
	Testigo en plaza	**Grupo 4**	**Grupo 9**

A tal fin cada grupo debe realizar:

1. **Concebir la aplicación**, esto es proponer su funcionamiento de las restricciones que a cada uno se le han establecido. Esta es la parte esencial de la práctica.
2. **Documentar el diseño** que se ha concebido:
   a. Formulando mediante diagramas de clases la interfaz de las clases RdP, Transition, Place y Tocken. Métodos y atributos.
   b. Cuando se requiera, describir mediante diagramas de estado los diferentes estados en que se puede encontrar cada objeto de las clases PetriNet definidas.
   c. Describir mediante pseudocódigo o diagrama de actividad la función de los métodos de la interfaz de las clases.
   d. Formulando mediante diagramas de clases la interfaz de las clases que constituyen el entorno. Estas pueden formularse en base al ejemplo CoalMine.
   e. Mediante diagramas de colaboración (diagramas de secuencias o diagramas de objetos) verificar como opera el diseño en un conjunto de escenarios representativos.
3. Formular el diseño mediante un documento PowerPoint (Figura y notas) que permita explicar el diseño.

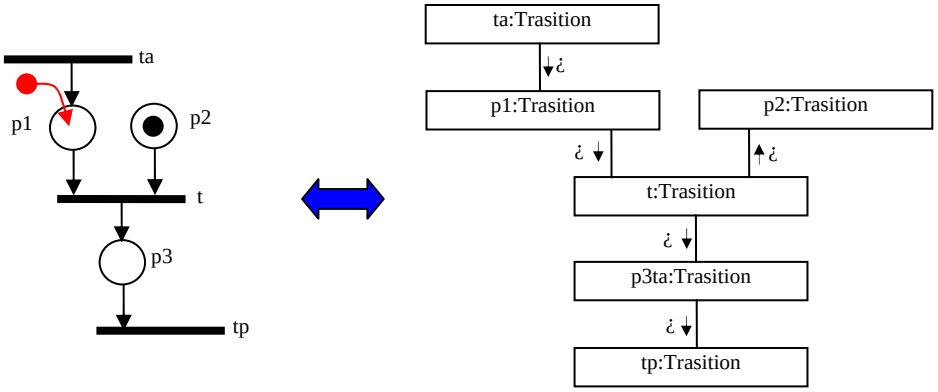
Ejemplos de escenarios representativos

- Construcción y lanzamiento de la aplicación.

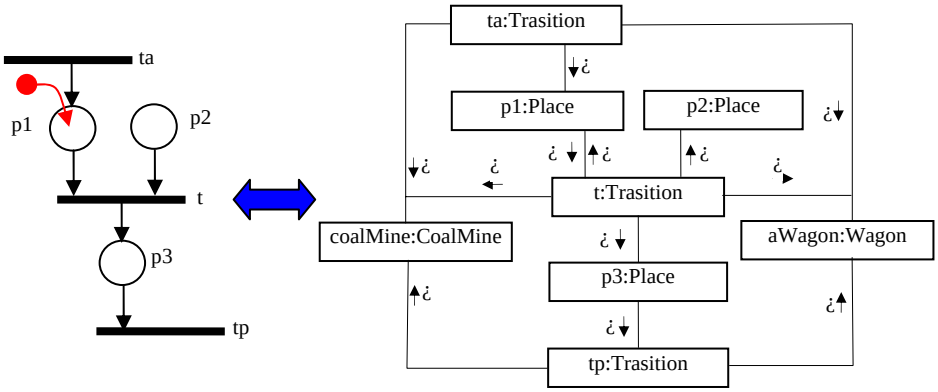


- Finalización de la aplicación.

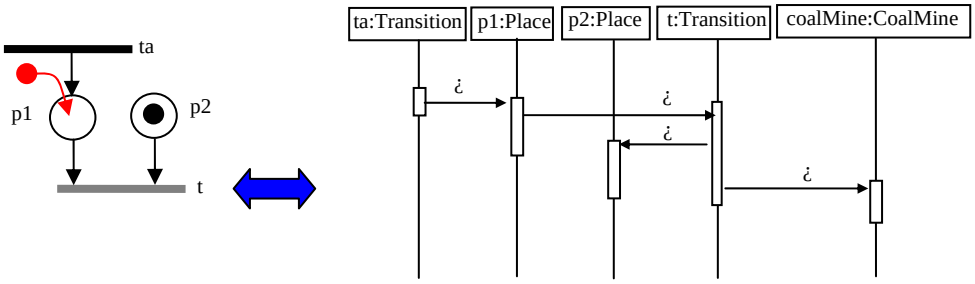
- Evolución de respuesta a la llegada de un testigo a una plaza que habilita el disparo de una transición.



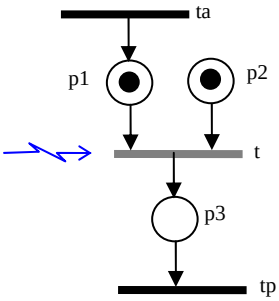
- Evolución de respuesta a un evento a una plaza que no habilita el disparo de ninguna transición.



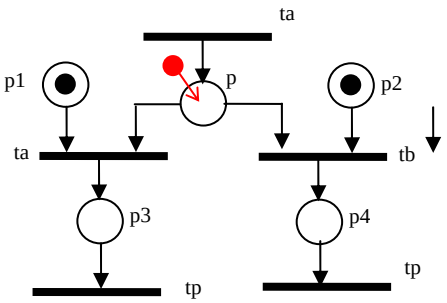
- Evolución de respuesta a un event a una plaza que habilita la transición, pero no dispara como consecuencia de que la señal asociada al transición no está establecida.



- Evolución de respuesta a un evento procedente del entorno que habilita una transacción.



- Evolución de respuesta a la llegada de un testigo que provoca con conflicto de ejecución en la red de Petri.



(Estos escenarios deben adaptarse a la estrategia de diseño que imponen las restricciones que se establecen en cada grupo de prácticas.)

Calendario

Lunes 18 Diciembre: Planteamiento del trabajo y formación de grupos de prácticas.
Martes 10 Enero: Consultas y discusiones.
Viernes 13 Enero: Presentación de las prácticas.