14 de marzo de 2019

Laura Aguilera Checa

Ignacio Aguilera Gómez

Grado en ingeniería informática

Universidad de Almería

Relación de ejercicios

Sistemas de tiempo real

Contenido

[1. Autores 2](#_Toc5124876)

[2. Introducción 2](#_Toc5124877)

[3. Actividades a realizar 3](#_Toc5124878)

[3.1. Descripción de la planta 3](#_Toc5124879)

[3.2. Red de Petri temporal 3](#_Toc5124880)

[3.3. Implementación en ADA sin tareas concurrentes 4](#_Toc5124881)

[3.4. Implementación en ADA con tareas concurrentes 4](#_Toc5124882)

[3.5. Conclusión 4](#_Toc5124883)

[Anexo. Descripción del problema: Control de una instalación de destilación por membranas 4](#_Toc5124884)

# Autores

|  |  |  |
| --- | --- | --- |
| Apellidos | Aguilera | Checa |
| Nombre | Laura |  |
| Titulación | Grado de Ingeniería Informática | |

|  |  |  |
| --- | --- | --- |
| Apellidos | Aguilera | Gómez |
| Nombre | Ignacio |  |
| Titulación | Grado de Ingeniería Informática | |

# Ejercicio 6. Modelado con Redes de Petri

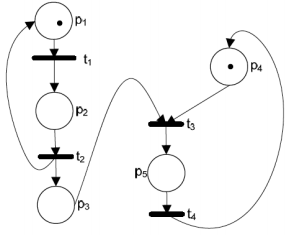
## 1. Introducción

Con estos ejercicios se pretende realizar un acercamiento del alumno al modelado discreto basado en eventos utilizando Redes de Petri. Para ello, se plantean tres ejercicios sencillos de uso e interpretación de esta herramienta de modelado. Dicho ejercicio deberá de ser remitido al profesor a través del módulo de Tareas de la plataforma Aula Virtual antes de la finalización del plazo de envío indicado. El formato de entrega de los ejercicios deberá de ser un documento PDF en el que se incluyan los comentarios y la solución de los ejercicios realizados.

## 2. Actividades a realizar

### Ejercicio 1. Interpretando redes de Petri

Dada la Red de Petri mostrada en la Figura 1:



*1. Describir formalmente (usando la notación de quíntupla de Red de Petri) el modelo.*

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

T = {t1, t2, t3, t4}

A = {(p1, t1), (t1, p2), (p2, t2), (t2, p1), (t2, p3), (p3, t3), (p4, t3), (t3, p5), (p5, t4), (t4, p4)}

W(p1, t1) = 1

W(t1, p2) = 1

W(p2, t2) = 1

W(t2, p1) = 1

W(t2, p3) = 1

W(p3, t3) = 1

W(p4, t3) = 1

W(t3, p5) = 1

W(p5, t4) = 1

W(p5, t4) = 1

W(t4, p4) = 1

M0 = {1, 0, 0, 1, 0}

*2. Explicar en qué consiste el modelo propuesto de forma general. ¿Conoce algún proceso que pueda ser representado por esta Red de Petri?*

Se ejecuta P2 repetidamente, accediendo al recurso compartido P1. Por otra parte, una vez se ha completado la primera ejecución de P2, se ejecuta P3 y se queda en espera hasta que finalice el proceso P4 para, a continuación, ejecutar P4 en bucle.

Un ejemplo que pueda ser representado con esta red de Petri podría ser un sistema de riego de un invernadero, que comprueba la temperatura del ambiente antes de empezar a regar. Si la temperatura es muy alta, se abren las compuertas de ventilación. Esto se comprueba periódicamente para garantizar la temperatura adecuada. A su vez, se comprueba la humedad de la tierra y se riega en función al valor obtenido.

P1: Sensor de temperatura

P2: Controlar las compuertas de ventilación

P3: Nodo auxiliar

P4: Sensor de la humedad del agua

P5: Riego

### Ejercicio 2. Modelado de un proceso de fabricación

Dado el proceso descrito en la Figura 2:



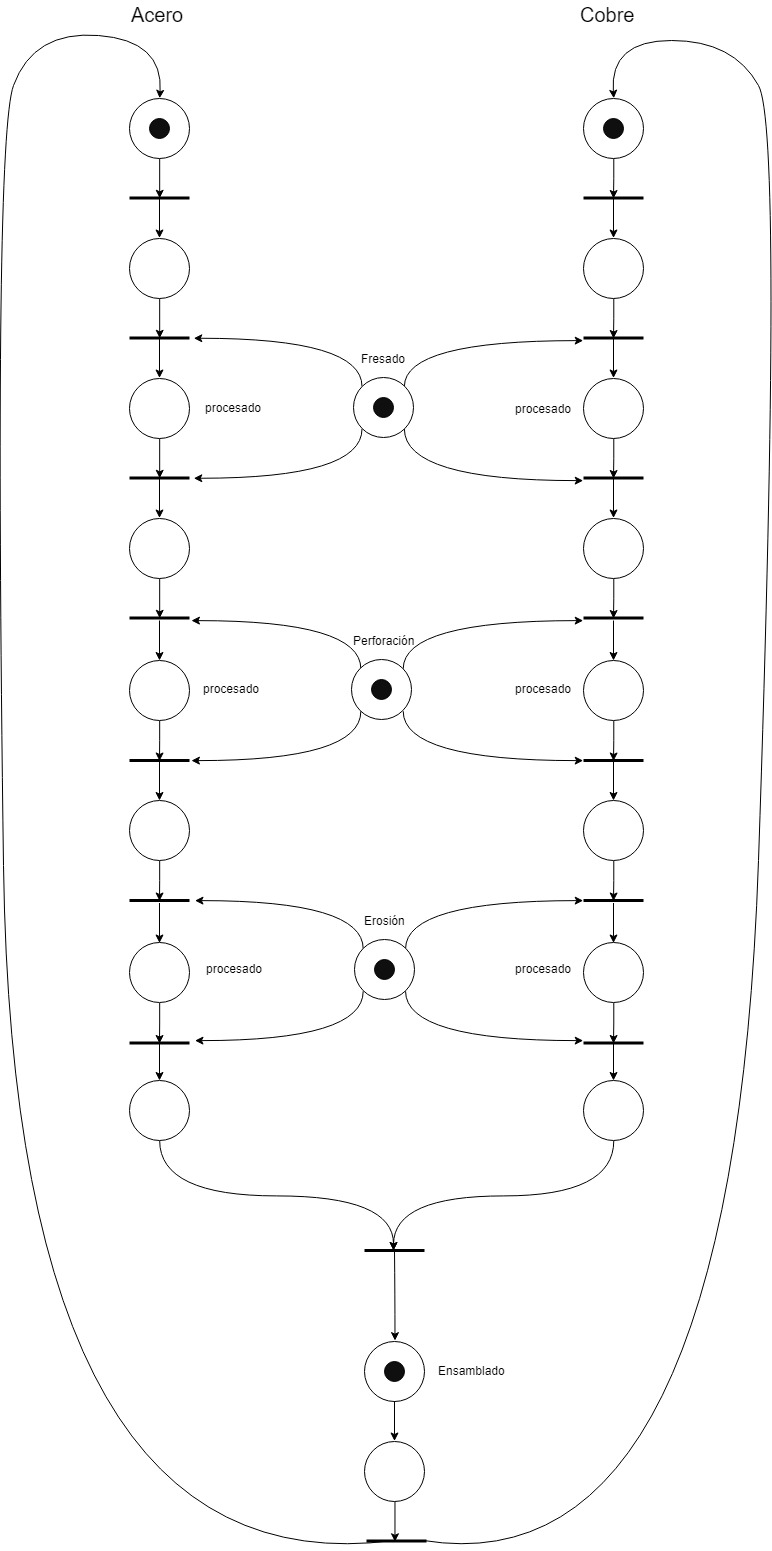
Figura 2. Proceso de fabricación para Ejercicio 2

*1. Describir los recursos compartidos y los almacenes existentes.*

Como recursos compartidos tenemos las máquinas de fresado, perforación, erosión y ensamblado y los almacenes existentes son los almacenes de materia prima y de productos finales.

*2. Realizar la Red de Petri que lo modela suponiendo que:*

* *Existen dos tipos de piezas (A: Acero, C: Cobre) que llegan indistintamente a un almacén de materia prima.*
* *Dichas piezas tienen que recorrer todas las máquinas del proceso (fresado, perforación, erosión y ensamblado) en el mismo orden.*
* *El tiempo de operación de los tratamientos de fresado, perforación y erosión depende del tipo de pieza.*
* *Durante el ensamblado es necesaria una pieza de cada tipo para generar el producto final.*



### Ejercicio 3. Modelado de tareas concurrentes con recursos compartidos

Dado el diagrama de tareas expuesto en la Figura 3:

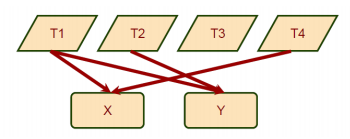
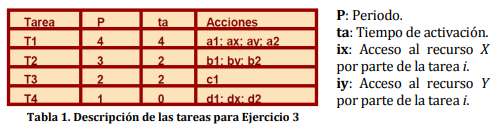
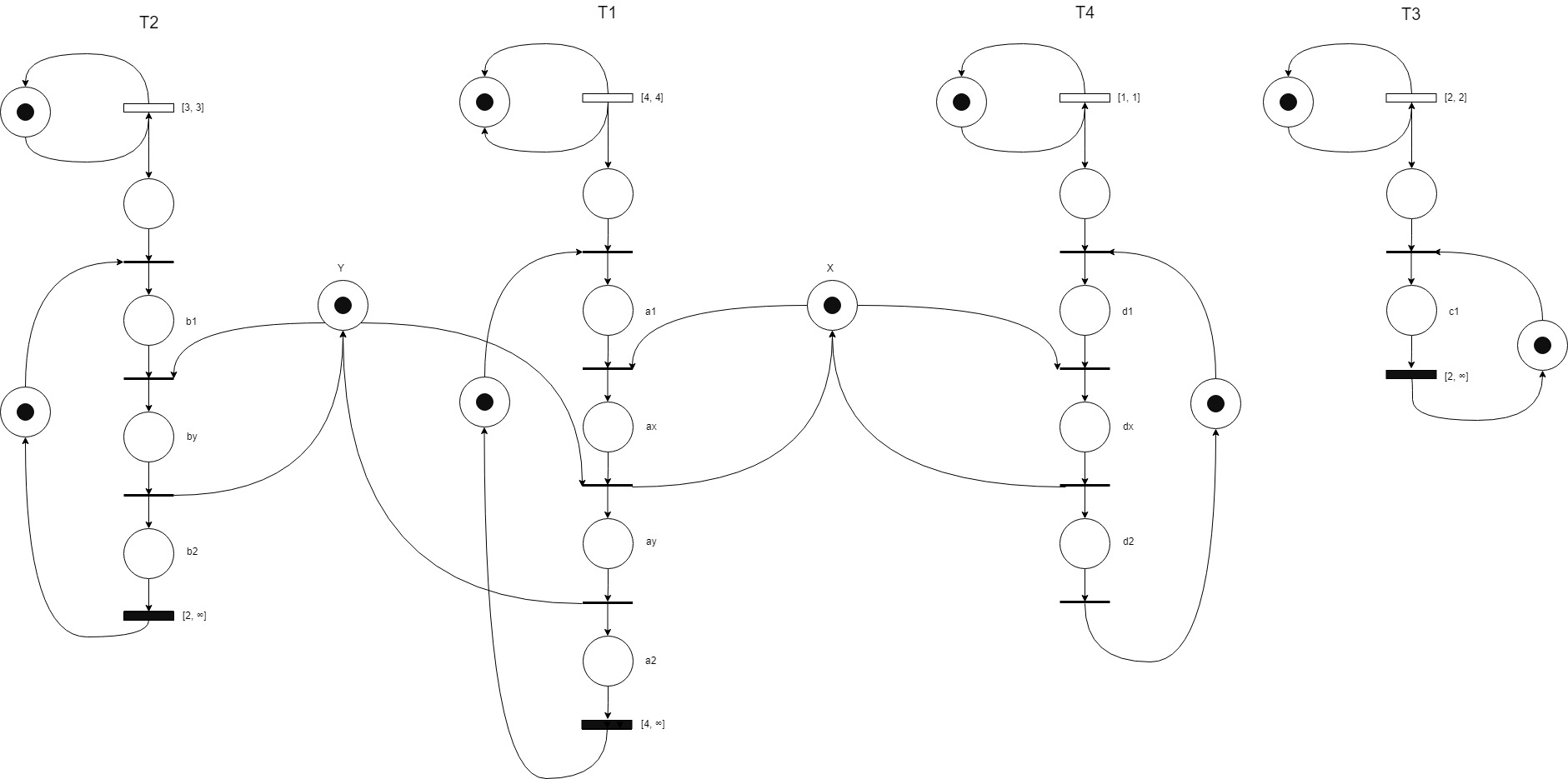


Figura 3. Diagrama de tareas para Ejercicio 3

1. Realizar el modelado concurrente mediante Redes de Petri a partir de la Tabla 1



# Ejercicio 7. Modelado con Redes de Petri Temporales

## 1. Introducción

Con estos ejercicios se pretende realizar un acercamiento del alumno al modelado discreto basado en eventos utilizando Redes de Petri Temporales. Para ello, se plantean tres ejercicios sencillos de uso e interpretación de esta herramienta de modelado. Dicho ejercicio deberá de ser remitido al profesor a través del módulo de Tareas de la plataforma Aula Virtual antes de la finalización del plazo de envío indicado. El formato de entrega de los ejercicios deberá de ser un documento PDF en el que se incluyan los comentarios y la solución de los ejercicios realizados.

## 2. Actividades a realizar

### Ejercicio 1. Interpretando Redes de Petri Temporales

Dada la Red de Petri Temporal mostrada en la Figura 1:

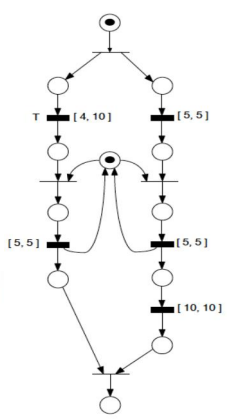


Figura 1. Red de Petri para Ejercicio 1

*1. Calcular el tiempo de ejecución total suponiendo que todos los elementos se ejecutan en su tiempo más lento.*

20 = 5 + 5 + 10

*2. Calcular el tiempo de ejecución total suponiendo que todos los elementos se ejecutan en su tiempo más rápido.*

24 = 5 + 4 + 5 + 10

### Ejercicio 2. Modelado de un proceso de fabricación

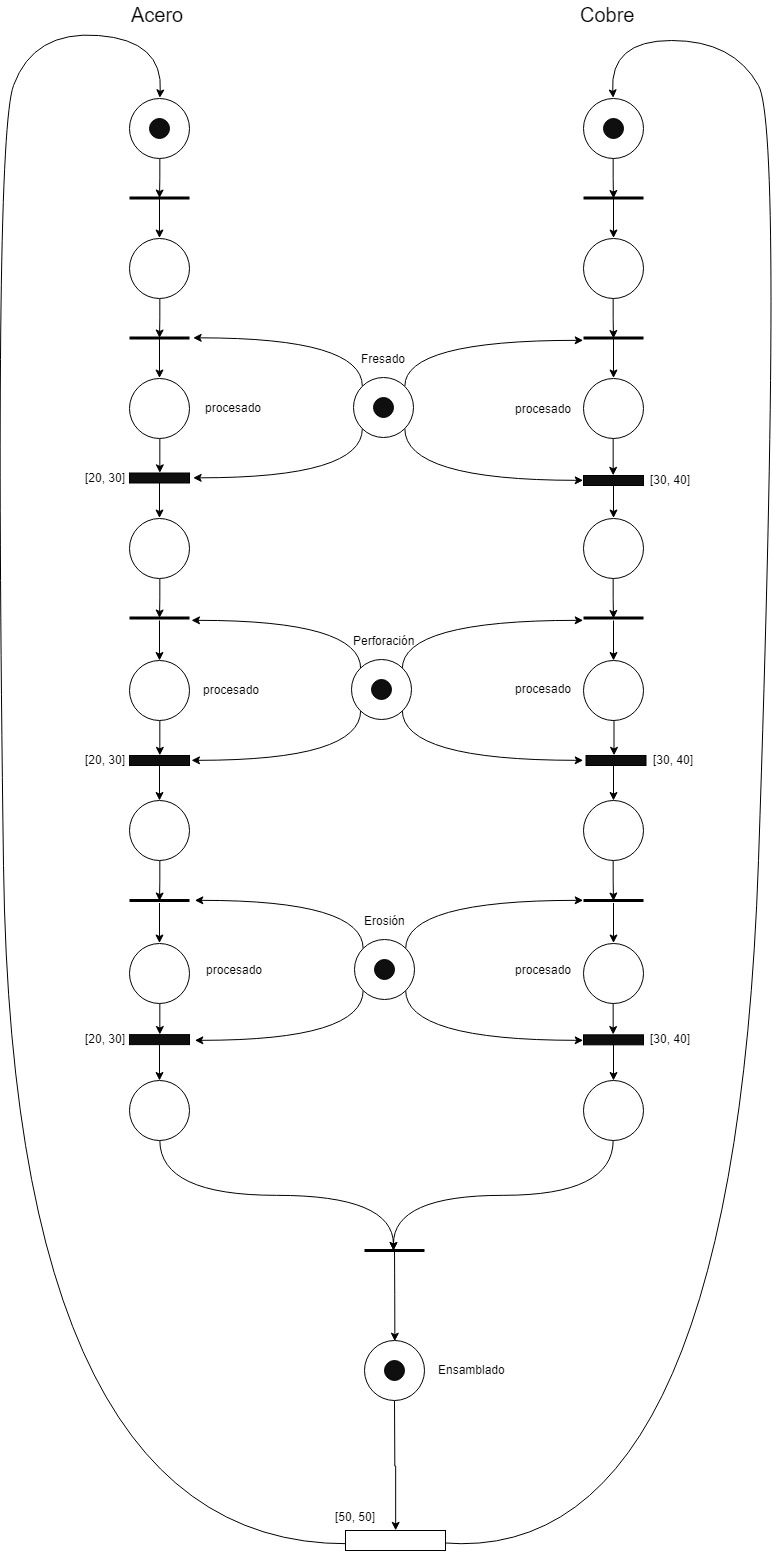
Dado el proceso descrito en la Figura 2:



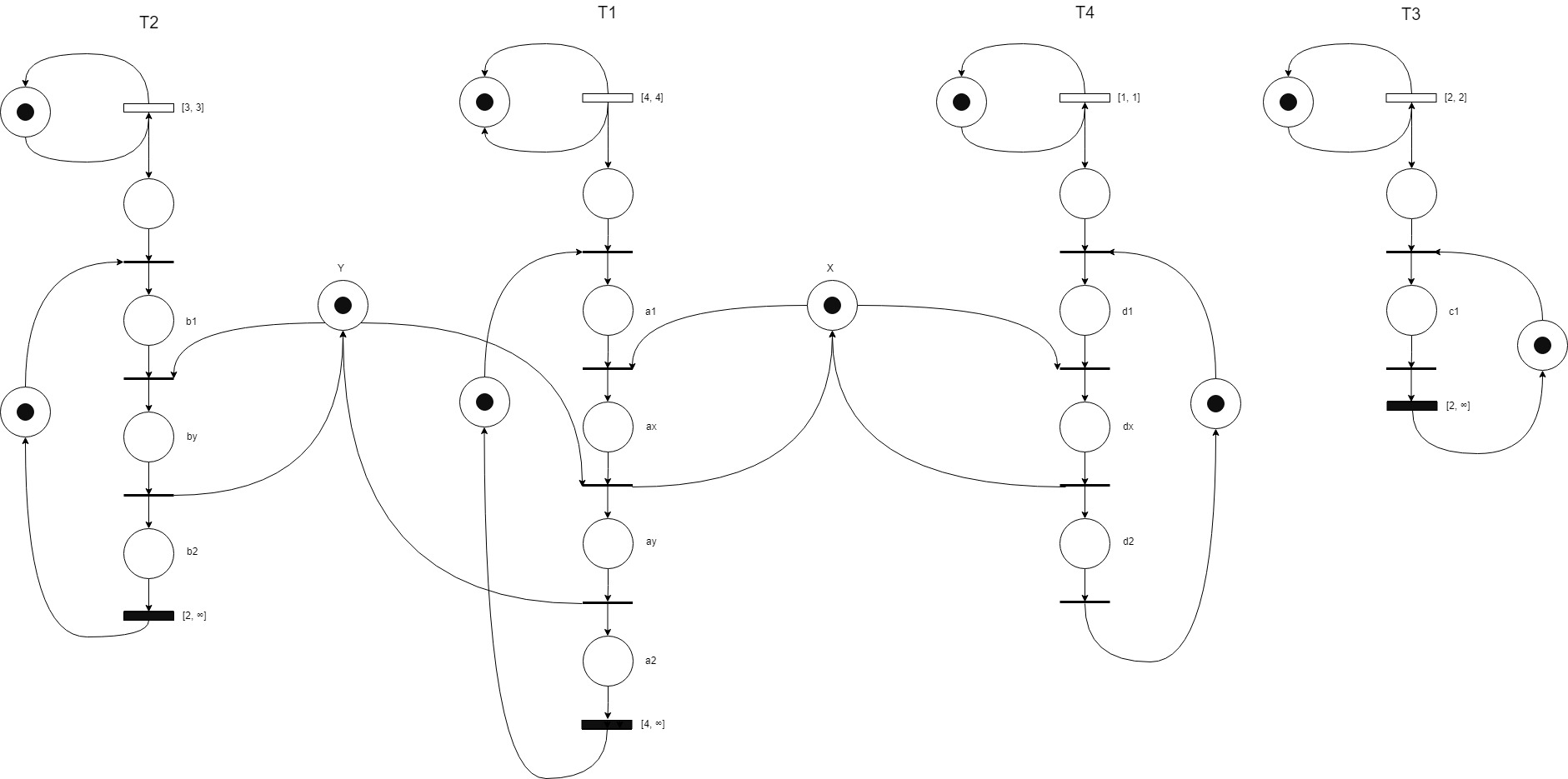
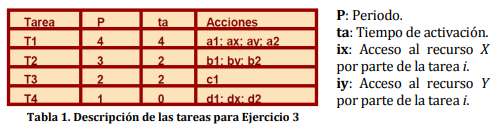
Figura 2. Proceso de fabricación para Ejercicio 2

1. Realizar la Red de Petri Temporal que lo modela suponiendo que:

* Existen dos tipos de piezas (A: Acero, C: Cobre) que llegan indistintamente a un almacén de materia prima.
* Dichas piezas tienen que recorrer todas las máquinas del proceso (fresado, perforación, erosión y ensamblado) en el mismo orden.
* El tiempo de operación de los tratamientos de fresado, perforación y erosión depende del tipo de pieza.
* Durante el ensamblado es necesaria una pieza de cada tipo para generar el producto final.
* Las operaciones con la pieza A requieren de 20 a 30 minutos y con la pieza B de 30 a 40 minutos.
* La operación de ensamblado requiere de 35 a 45 minutos y deberá ser abortada en el caso de que se alcancen los 50 minutos.



### Ejercicio 3. Modelado de tareas concurrentes con recursos compartidos

1. Realizar el modelado concurrente mediante Redes de Petri Temporales a partir de la Tabla NOTA: Ignorar tiempos de cómputo de las acciones.

# Ejercicio 8. Concurrencia de tareas en ADA

## 1. Introducción

Con estos ejercicios se pretende realizar un acercamiento del alumno a la concurrencia de tareas en ADA. Para ello, se plantean dos ejercicios sencillos de programación concurrente basada en SO (Sistema Operativo) y lenguaje de programación contra la programación secuencial de un ejecutivo cíclico. Dichos ejercicios deberán de ser remitidos al profesor a través del módulo de Tareas de la plataforma Aula Virtual antes de la finalización del plazo de envío indicado. El formato de entrega de los ejercicios deberá de ser un documento PDF en el que se incluyan los comentarios y la solución de los ejercicios realizados.

## 2. Actividades a realizar por los alumnos

### Ejercicio 1. Concurrencia basada en SO vs. lenguaje de programación

*Explicar la diferencia entre concurrencia basada en SO y concurrencia basada en tasks de ADA.*

La concurrencia basada en SO es más flexible y sencilla de implementar, puesto que es independiente del lenguaje, pero no podemos hacer uso de los tipos de nuestro programa para comunicar varios procesos, cosa que usando tasks si se nos permite. También, al usar la concurrencia del SO nuestro compilador no detectará errores relacionados con esta concurrencia. Y, por último, no sabemos si tan siquiera nuestro SO soporta concurrencia.

### Ejercicio 2. Modelado de un proceso de fabricación

1. Implementación secuencial (A -> B -> C).

Puesto que la implementación es secuencial, simplemente vamos a imprimir por consola las tres tareas en bucles *for* individualmente una a una.

procedure apartado1 is

i :integer := 0;

begin

for i in 1..3 loop

     Put\_Line ("Tarea A: " & i'image);

    end loop;

i := 0;

for i in 1..3 loop

     Put\_Line ("Tarea B: " & i'image);

    end loop;

i := 0;

for i in 1..3 loop

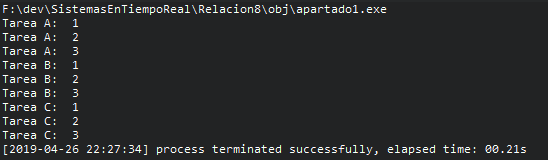
     Put\_Line ("Tarea C: " & i'image);

    end loop;

i := 0;

end apartado1;

Salida por consola:



1. Implementación basada en concurrencia del lenguaje de programación con tareas estáticas.

Aquí no necesitamos añadir nada a nuestro bloque de ejecución, nos basta con declarar las tareas y si “body”, en el cual se imprime las tareas. Cuando ejecutemos el programa se van a ejecutar en el orden declarado.

procedure apartado2 is

procedure apartado2 is

i : integer := 0;

task TA;

task TB;

task TC;

task body TA is

begin

    i := 0;

    for i in 1 .. 3 loop

     Ada.Text\_IO.Put\_Line ("Tarea A:" & i'image);

    end loop;

end TA;

task body TB is

begin

    i := 0;

    for i in 1 .. 3 loop

     Ada.Text\_IO.Put\_Line ("Tarea B:" & i'image);

    end loop;

end TB;

task body TC is

begin

    i := 0;

    for i in 1 .. 3 loop

     Ada.Text\_IO.Put\_Line ("Tarea C:" & i'image);

    end loop;

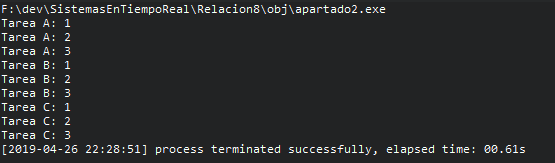
end TC;

begin

null;

end apartado2;

Salida por consola:



1. Implementación basada en concurrencia del lenguaje de programación con tareas dinámicas.

Para implementar este proceso con concurrencia basada en el lenguaje ada primero debemos declarar las tareas al igual que unas referencias a estas, las cuales luego utilizaremos para crearlas dinámicamente.

procedure apartado3 is

i : integer := 0;

task type TA;

type REF\_TA is access TA;

task body TA is

begin

    i := 0;

    for i in 1 .. 3 loop

     Ada.Text\_IO.Put\_Line ("Tarea A:" & i'image);

    end loop;

end;

task type TB;

type REF\_TB is access TB;

task body TB is

begin

    i := 0;

    for i in 1 .. 3 loop

     Ada.Text\_IO.Put\_Line ("Tarea B:" & i'image);

    end loop;

end;

task type TC;

type REF\_TC is access TC;

task body TC is

begin

    i := 0;

    for i in 1 .. 3 loop

     Ada.Text\_IO.Put\_Line ("Tarea C:" & i'image);

    end loop;

end;

begin

declare

    m\_ta : REF\_TA := new TA;

    m\_tb : REF\_TB := new TB;

    m\_tc : REF\_TC := new TC;

begin

    Ada.Text\_IO.Put ("");

end;

end apartado3;

Salida por consola:

