Universidad Nacional de Córdoba

Facultad de Ciencias Exactas, Físicas y Naturales

Programación Concurrente

Trabajo Práctico Final

Nombre del grupo: Pionono

Integrantes del grupo:

- Careggio, Camila
- Casanueva, María Constanza
- Erlicher, Ezequiel
- Riba, Franco

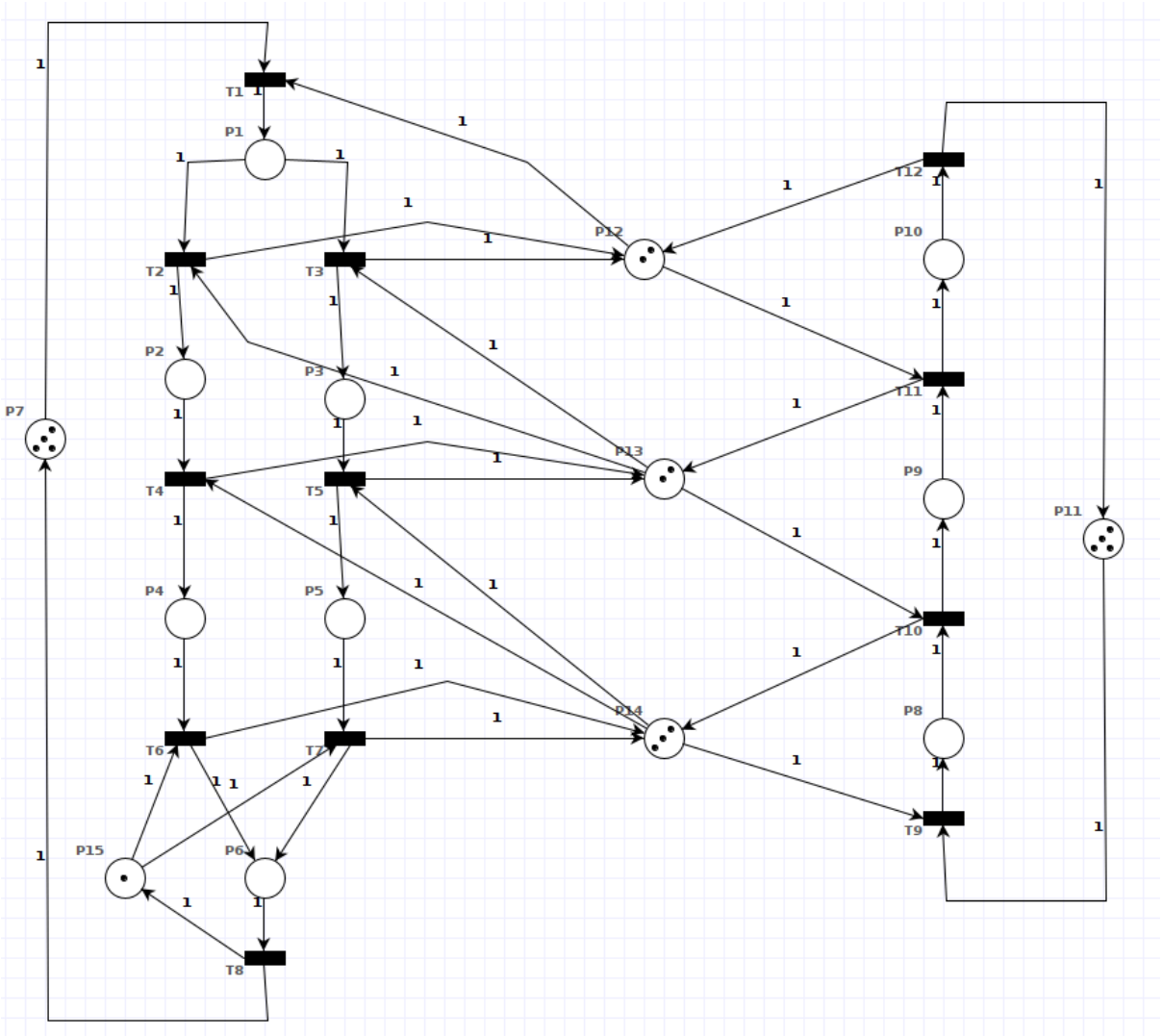
Profesor/es:

- Ventre, Luis Orlando
- Ludemann, Mauricio

Abril 2024

Índice

Red Inicial


T1
P1
T2
T3
P12
T12
P10
P2
P3
T11
P7
T4
T5
P13
P9
P11
P4
P5
T10
T6
T7
P14
P8
P15
P6
T9
T8

Clasificación

Petri net classification results

State Machine	false
Marked Graph	false
Free Choice Net	false
Extended Free Choice Net	false
Simple Net	false
Extended Simple Net	true

Propiedades de la Red

Observamos las propiedades en State Space Analysis de PIPE:

Petri net state space analysis results

Bounded	true
Safe	false
Deadlock	true

Limitada

Una plaza está limitada si el número de tokens nunca será mayor a k, mientras que una RdP está limitada si todas sus plazas están limitadas. Esto depende de la estructura y la marca inicial.

Esta red está limitada a 4.

Seguridad

Una RdP es segura para una marca inicial si para cada marca alcanzable, cada plaza contiene cero o un token. También depende de la estructura y de la marca inicial.

Se dice que una red es segura si es acotada a uno, esto es si las plazas tienen como máximo un token.

Como ya establecimos que la red está acotada a 4, por lo tanto no es segura.

Vivacidad

La red es Quasi viva ya que todas las transiciones tienen al menos un marcado alcanzable desde el marcado inicial en el cuál están habilitadas. Esta opción surge de las definiciones de red Quasi-viva en la página 13 de:
https://postgrado.info.unlp.edu.ar/wp-content/uploads/2017/11/2015_Aaron_Gustavo_Horacio_Wolfmann.pdf

2. El problema de la sobrevivencia (*Liveness*). Un marcado sin Transiciones habilitadas es llamado muerto. Una PN es pseudo-viva, si $\forall M \in \Re(M_0) \exists t \in T : t$ está habilitada, lo que significa que todos los marcados alcanzables tienen una Transición que puede ser disparada. Una PN es quasi-viva si $\forall t \in T, \exists M \in \Re(M_0) : t$ está habilitada en M, lo que significa que todas las Transiciones tienen al menos un marcado en que están habilitadas. Una PN es viva si $\forall M \in \Re(M_0)$ y $\forall t \in T, \exists M' \in \Re(M) : t$ está habilitada M', lo que significa que para todos los marcados alcanzables, todas las Transiciones quedarán habilitadas en al menos un marcado posterior.

En una RdP viva se garantiza una operación libre de puntos muertos independientemente de la secuencia de disparo elegida. Esta propiedad es muy importante para caracterizar el bloqueo total o parcial de un sistema.

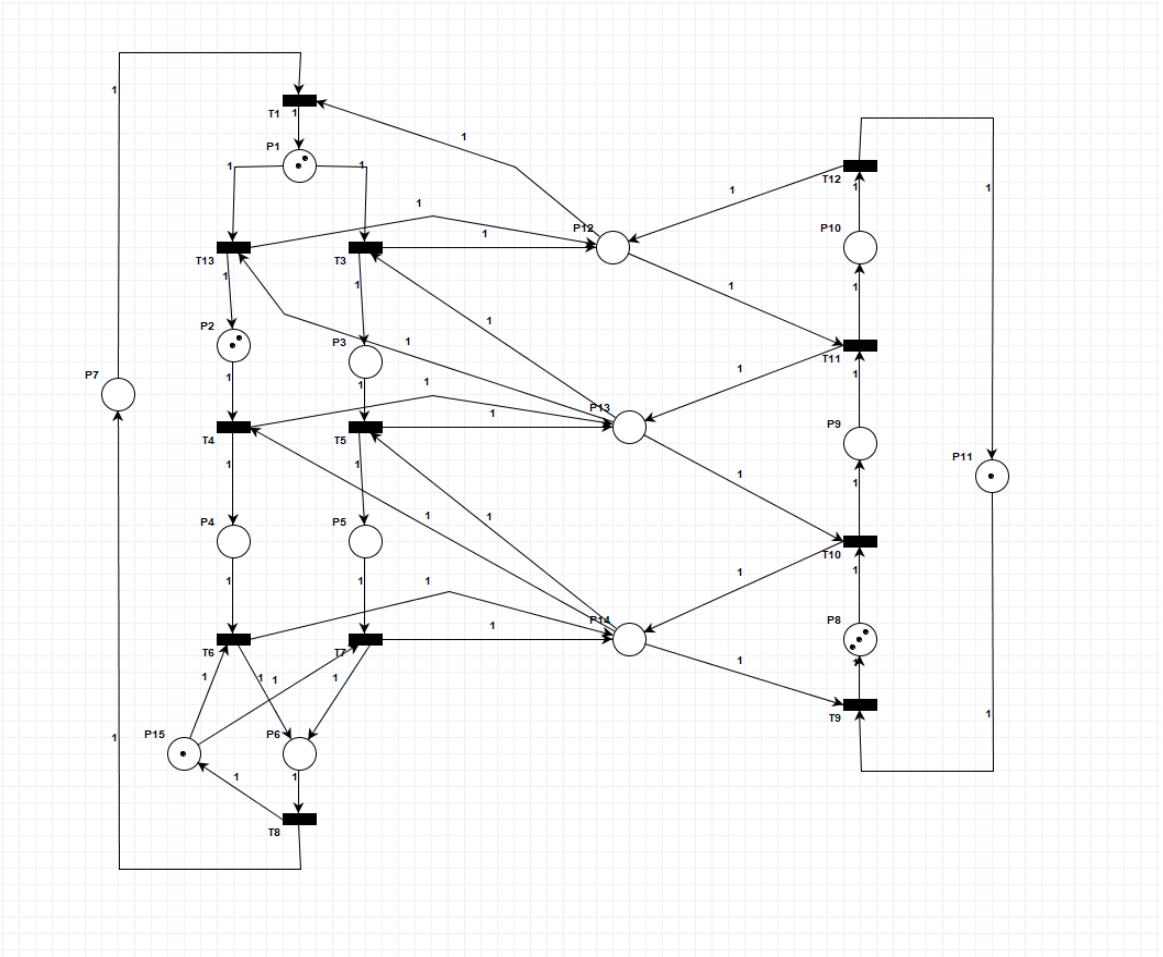
Cuando se produce el deadlock, hay transiciones que no se disparan ni una sola vez, esto significa que no es Quasi viva.

Deadlock

Un Deadlock sucede cuando no se puede disparar ninguna transición. Nuestra red sí tiene deadlock, siendo la secuencia de disparo más corta para llegar a este estado (partiendo desde el marcado inicial y según el simulador PIPE) la siguiente:

T1 T1 T2 T1 T2 T1 T9 T9 T9

Un ejemplo de una situación con deadlock es la siguiente, ya que no se puede disparar ninguna transición más:



Invariantes

Invariantes de plaza

Un invariante de plaza se obtiene de la siguiente manera:

$$Subconjunto\ de\ plazas\ P\ \ *\ \ Vector\ de\ ponderación\ q\ \ =\ \ constante\ (invariante\ de\ plaza)$$

El invariante de plaza depende de la marca inicial, mientras que P es un conjunto conservador, independiente de la marca inicial.

En la red inicial, estos son:

P-Invariants

P1	P10	P11	P12	P13	P14	P15	P2	P3	P4	P5	P6	P7	P8	P9
0	1	1	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	1	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	1	1	1	1	1	0	0

P-Invariant equations

M(P10) + M(P11) + M(P8) + M(P9) = 4
M(P1) + M(P10) + M(P12) = 2
M(P13) + M(P2) + M(P3) + M(P9) = 2
M(P14) + M(P4) + M(P5) + M(P8) = 2
M(P15) + M(P6) = 1
M(P1) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P7) = 4

Invariantes de transición

Son las secuencias repetitivas que causan retorno al estado inicial en la red:

T-Invariants

T1	T10	T11	T12	T2	T3	T4	T5	T6	T7	T8	T9
0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	1	0	1	0

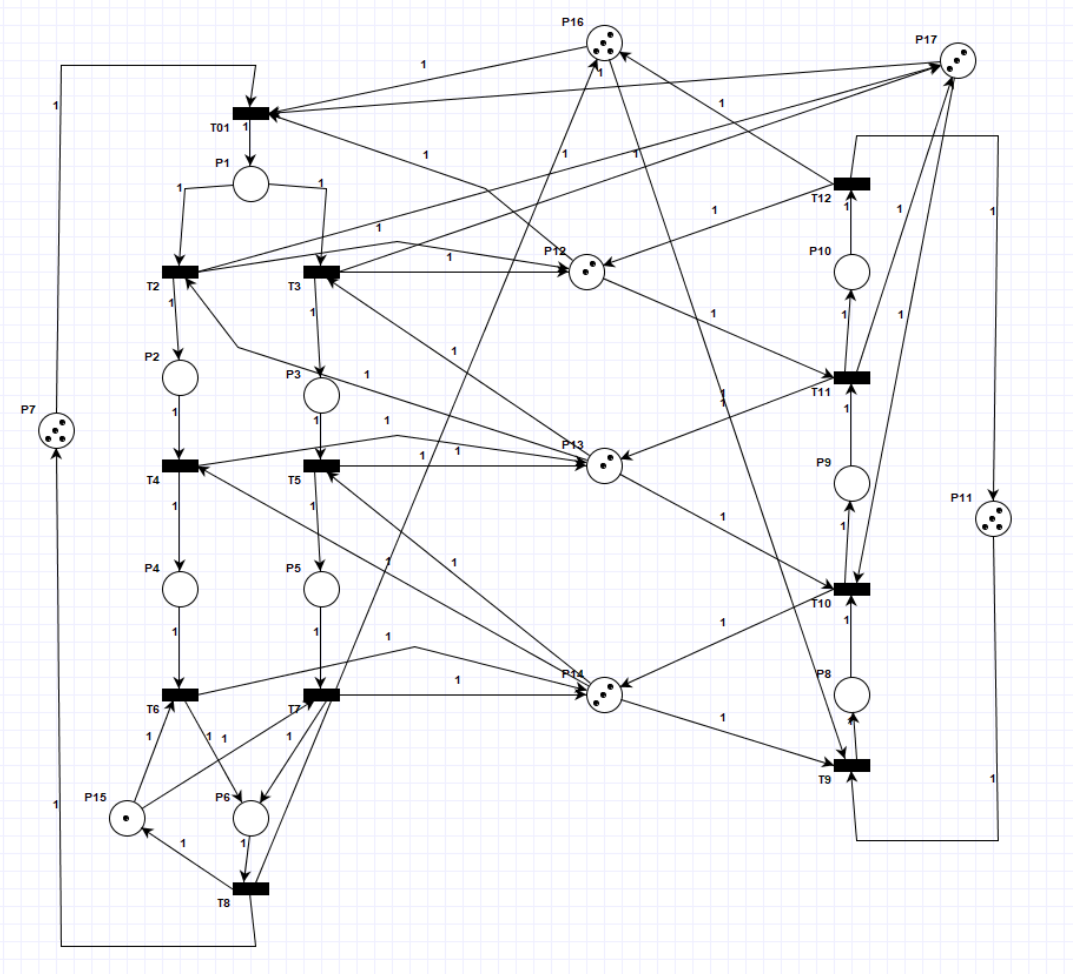
$S1 = (T9\ T10\ T11\ T12)$

$S2 = (T1\ T3\ T5\ T7\ T8)$

$S3 = (T1\ T2\ T4\ T6\ T8)$

Solución al Deadlock - Alternativa 1

Se agregan las plazas P16 y P17 con cuatro y tres tokens respectivamente para desbloquear la red:



Verificamos que ya no exista deadlock:

Petri net state space analysis results

Bounded	true
Safe	false
Deadlock	false

Y verificamos que no se hayan modificado los invariantes de transición:

Petri net invariant analysis results

T-Invariants

T01	T10	T11	T12	T2	T3	T4	T5	T6	T7	T8	T9
0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	1	0	1	0

The net is covered by positive T-Invariants, therefore it might be bounded and live.

Invariantes de plaza

P-Invariants

P1	P10	P11	P12	P13	P14	P15	P16	P17	P2	P3	P4	P5	P6	P7	P8	P9
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0

The net is covered by positive P-Invariants, therefore it is bounded.

P-Invariant equations

M(P10) + M(P11) + M(P8) + M(P9) = 4
M(P1) + M(P10) + M(P12) = 2
M(P13) + M(P2) + M(P3) + M(P9) = 2
M(P14) + M(P4) + M(P5) + M(P8) = 3
M(P15) + M(P6) = 1
M(P1) + M(P10) + M(P16) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P8) + M(P9) = 4
M(P1) + M(P17) + M(P9) = 3
M(P1) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P7) = 4

Estos invariantes son los que se agregan al tener las nuevas plazas P16 y P17:

$m(P_1) + m(P_{17}) + m(P_9) = 3$

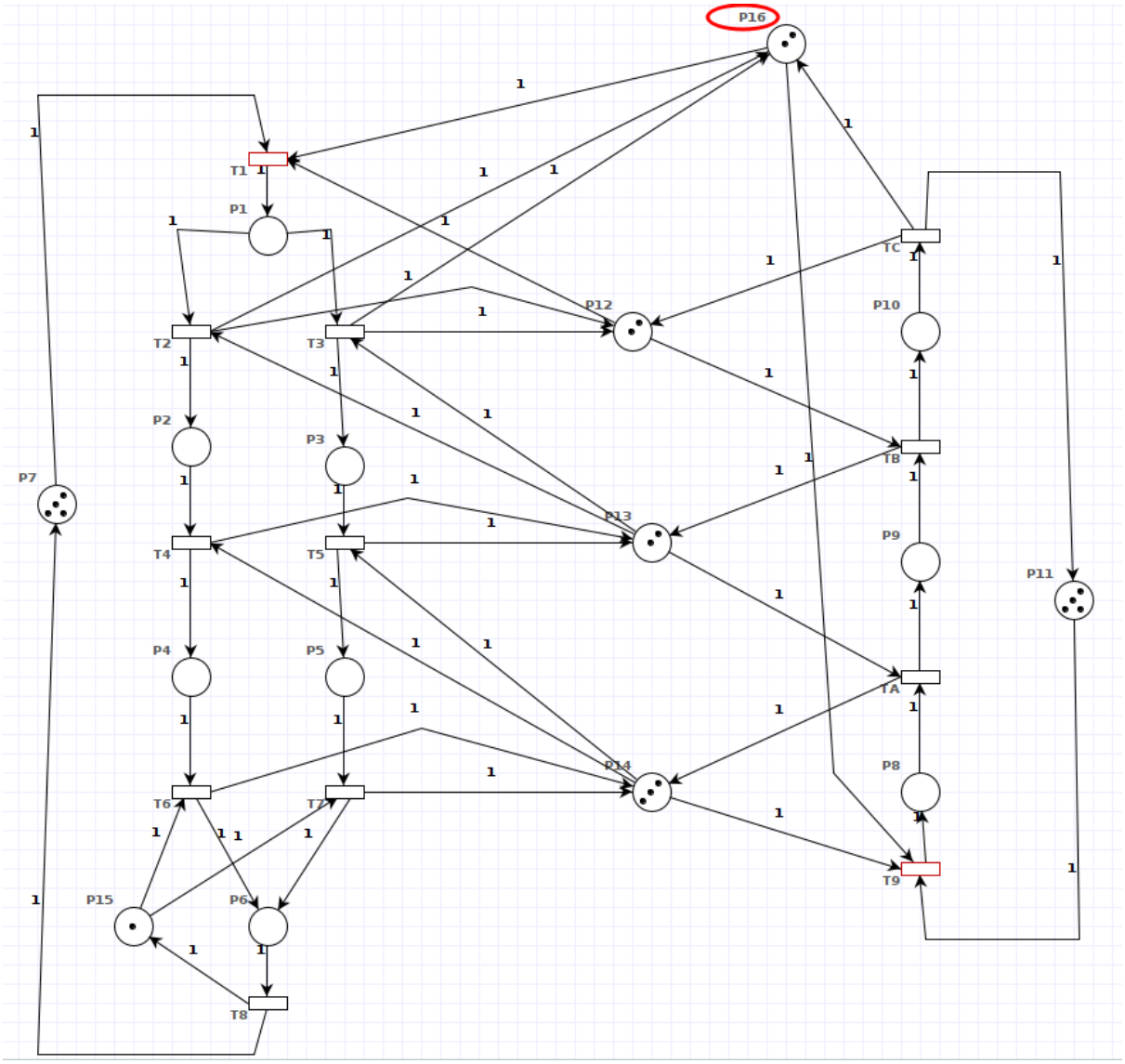
$m(P_1) + m(P_{10}) + m(P_{16}) + m(P_2) + m(P_3) + m(P_4) + m(P_5) + m(P_6) + m(P_8) + m(P_9) = 4$

Invariantes de transición

Se mantienen las ya analizadas en la primera PN propuesta.

Solución al Deadlock - Alternativa 2

Esta es otra alternativa de desbloqueo de la red que permite eliminar el Deadlock de la PN original .



Nota: a partir de aquí reemplazamos T10, T11 y T12 por TA, TB y TC respectivamente a fines poder generar simulaciones de secuencias de transiciones que podamos analizar posteriormente con nuestro script de Python. Teniendo en cuenta estos cambios, las invariantes de transición pasarían a quedar expresadas como:

$S1 = (T9\ TA\ TB\ TC)$
$S2 = (T1\ T3\ T5\ T7\ T8)$
$S3 = (T1\ T2\ T4\ T6\ T8)$

Verificamos que ya no exista deadlock:

Petri net state space analysis results

Bounded	true
Safe	false
Deadlock	false

Verificamos también que no se hayan modificado los invariantes de transición:

Petri net invariant analysis results

T-Invariants

T1	T10	T11	T12	T2	T3	T4	T5	T6	T7	T8	T9
0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	1	0	1	0

The net is covered by positive T-Invariants, therefore it might be bounded and live.

Invariantes de plaza

P-Invariants

P1	P10	P11	P12	P13	P14	P15	P16	P2	P3	P4	P5	P6	P7	P8	P9
0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0

The net is covered by positive P-Invariants, therefore it is bounded.

P-Invariant equations

M(P10) + M(P11) + M(P8) + M(P9) = 4
M(P1) + M(P10) + M(P12) = 2
M(P13) + M(P2) + M(P3) + M(P9) = 2
M(P14) + M(P4) + M(P5) + M(P8) = 3
M(P15) + M(P6) = 1
M(P1) + M(P16) + M(P8) + M(P9) = 2
M(P1) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P7) = 4

M(P10) + M(P11) + M(P8) + M(P9) = 4
M(P1) + M(P10) + M(P12) = 2

M(P13) + M(P2) + M(P3) + M(P9) = 2
M(P14) + M(P4) + M(P5) + M(P8) = 3
M(P15) + M(P6) = 1
M(P1) + M(P16) + M(P8) + M(P9) = 2
M(P1) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P7) = 4

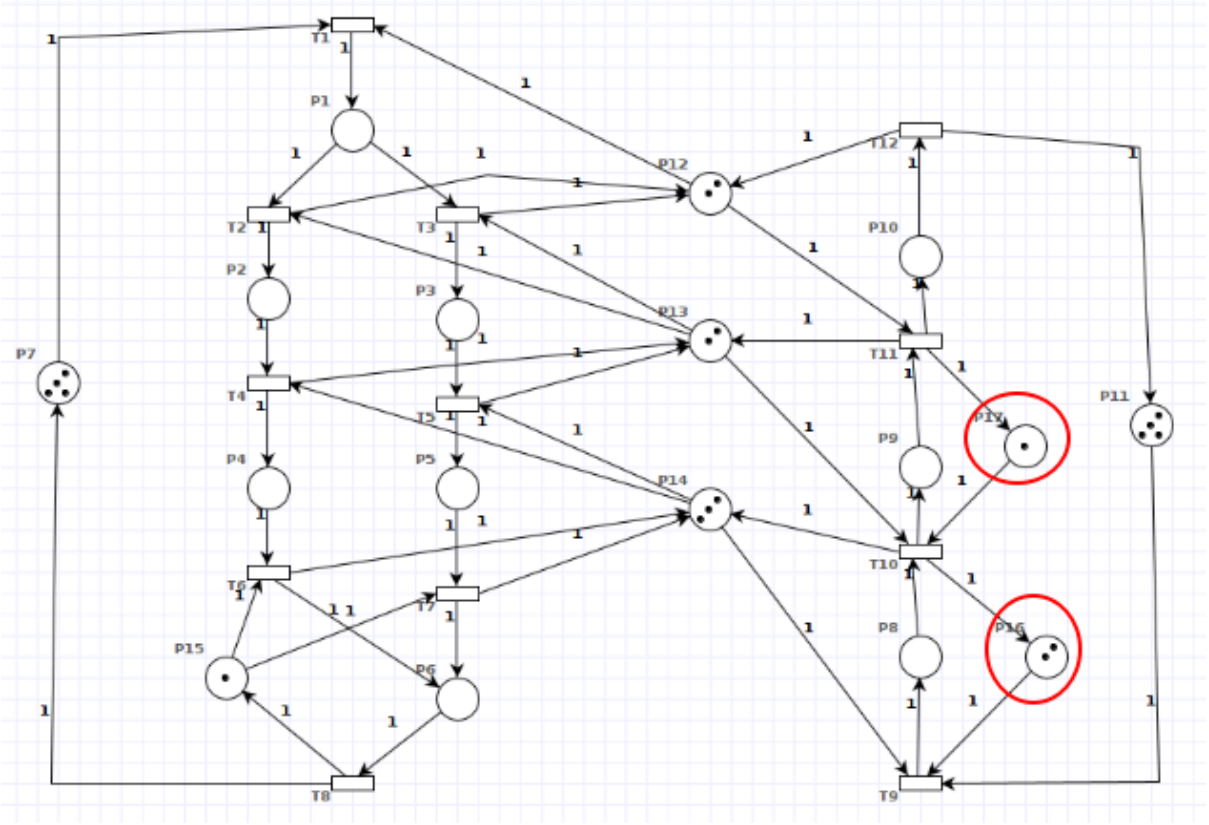
Podemos observar una nueva ecuación que involucra a la plaza agregada P16.

Invariantes de transición

Se mantienen las ya analizadas en la primera PN propuesta.

Solución al Deadlock - Alternativa 3

Esta es otra alternativa de desbloqueo de la red que permite eliminar el Deadlock de la PN original.



Verificamos que ya no exista deadlock:

Petri net state space analysis results

Bounded	true
Safe	false
Deadlock	false

Verificamos también que no se hayan modificado los invariantes de transición:

T-Invariants

T1	T10	T11	T12	T2	T3	T4	T5	T6	T7	T8	T9
1	0	0	0	0	1	0	1	0	1	1	0
1	0	0	0	1	0	1	0	1	0	1	0
0	1	1	1	0	0	0	0	0	0	0	1

The net is covered by positive T-Invariants, therefore it might be bounded and live.

Invariantes de plaza

P-Invariants

P1	P10	P11	P12	P13	P14	P15	P16	P17	P2	P3	P4	P5	P6	P7	P8	P9
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0

The net is covered by positive P-Invariants, therefore it is bounded.

M(P10) + M(P11) + M(P8) + M(P9) = 4
M(P1) + M(P10) + M(P12) = 2
M(P13) + M(P2) + M(P3) + M(P9) = 2
M(P14) + M(P4) + M(P5) + M(P8) = 3
M(P15) + M(P6) = 1
M(P16) + M(P8) = 2
M(P17) + M(P9) = 1
M(P1) + M(P2) + M(P3) + M(P4) + M(P5) + M(P6) + M(P7) = 4

Combined incidence matrix *I*

	T1	T10	T11	T12	T2	T3	T4	T5	T6	T7	T8	T9
P1	1	0	0	0	-1	-1	0	0	0	0	0	0
P10	0	0	1	-1	0	0	0	0	0	0	0	0
P11	0	0	0	1	0	0	0	0	0	0	0	-1
P12	-1	0	-1	1	1	1	0	0	0	0	0	0
P13	0	-1	1	0	-1	-1	1	1	0	0	0	0
P14	0	1	0	0	0	0	-1	-1	1	1	0	-1
P15	0	0	0	0	0	0	0	0	-1	-1	1	0
P16	0	1	0	0	0	0	0	0	0	0	0	-1
P17	0	-1	1	0	0	0	0	0	0	0	0	0
P2	0	0	0	0	1	0	-1	0	0	0	0	0
P3	0	0	0	0	0	1	0	-1	0	0	0	0
P4	0	0	0	0	0	0	1	0	-1	0	0	0
P5	0	0	0	0	0	0	0	1	0	-1	0	0
P6	0	0	0	0	0	0	0	0	1	1	-1	0
P7	-1	0	0	0	0	0	0	0	0	0	1	0
P8	0	-1	0	0	0	0	0	0	0	0	0	1
P9	0	1	-1	0	0	0	0	0	0	0	0	0

Comparación de las tres soluciones al deadlock

En base a los posibles marcados de la red, obtenidos mediante la funcionalidad "GSPN Steady State Analysis Results" de PIPE, se realiza el análisis.
Para comparar las tres soluciones encontradas, se toma como criterio el grado de paralelismo en cada una de estas. Dicho parámetro, se puede cuantificar al observar la cantidad de tokens promedio y máximo por marcado, es decir, la cantidad promedio y máxima de actividades simultáneas que pueden llevarse a cabo en cada red.

El análisis completo se encuentra en Possible markings. De aquí obtenemos:

Solución	Promedio de tokens por marcado	Máxima cantidad de tokens por marcado
Red original	**4,945**	**8**
Alternativa 1: plazas P16 y P17 con cuatro y tres tokens respectivamente	**3,497**	**4**
Alternativa 2: agregando la plaza P16 con dos tokens, tres arcos de entrada y dos arcos de salida	**4,595**	**8**
Alternativa 3: Agregando las plazas P16 con dos tokens y P17 con un token	**4,813**	**8**

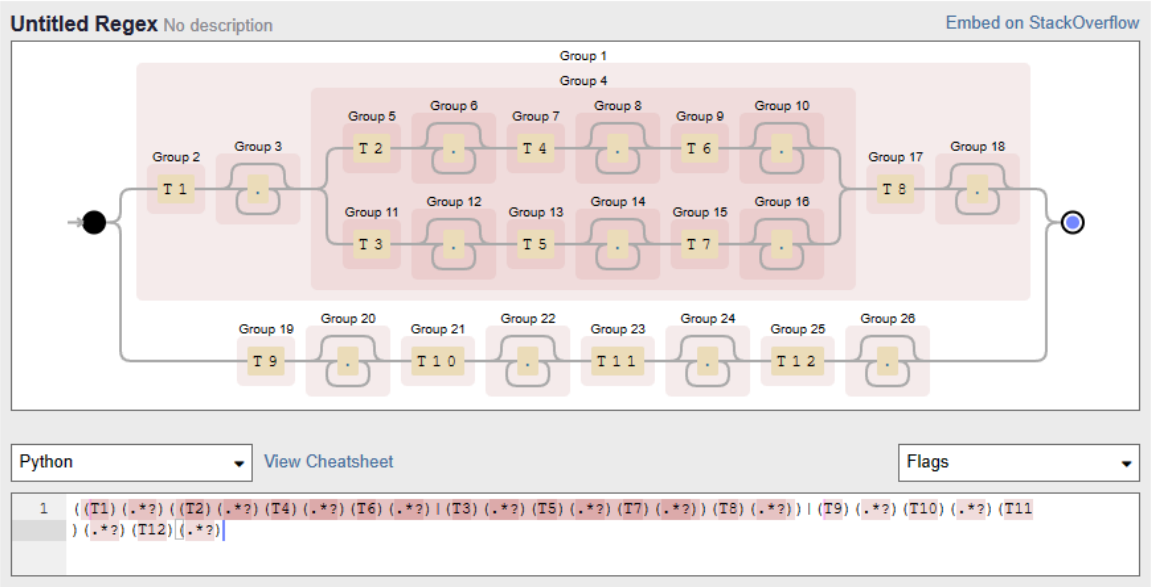
Claramente la alternativa 3 tiene un grado de paralelismo muy superior a las otras 2 alternativas , asemejándose incluso a la performance de la red de petri original.

Regex

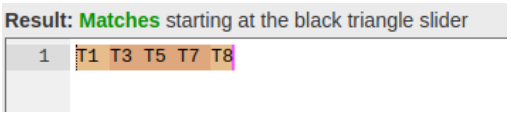
La expresión regular que define los invariantes de transición es:

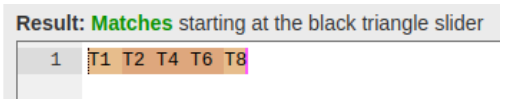
((T1)(.?)((T2)(.*?)(T4)(.*?)(T6)(.*?)|(T3)(.*?)(T5)(.*?)(T7)(.*?))(T8)(.*?))|(T9)(.*?)(T10)(.*?)(T11)(.*?)(T12)(.*?)*

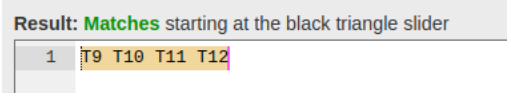
Haciendo uso de la herramienta web Debuggex podemos observar gráficamente nuestra expresión regular.



Verificamos que la expresión regular sea capaz detectar las invariantes de transición:







Con el script en Python *check-invariants.py* verificamos que las transiciones ejecutadas, cumplan con la regex definida.

El script remplaza los invariantes obtenidos con la ejecución de nuestro programa y los va filtrando a medida que los encuentra.

```
('T9TATBTCT9TATBTCT9TATBTCT9TATBTCT9TATBTCT
T9T9T1T2TBTCT3T1T4T3T1T6\n', 113)

('T1T9T9T1T2T3T1T4T3T1T6\n', 21)

('T1T9T9T1T2T3T1T4T3T1T6\n', 0)

Incomplete sequences found:

T1T9T9T1T2T3T1T4T3T1T6
```

Como podemos observar se indican secuencias incompletas, esto es porque nuestro script está diseñado para hallar invariables T completos, las transiciones de la secuencia incompleta pertenecen a invariantes que, por la cantidad de ejecuciones/disparos de la red de petri modelada en nuestro programa, no llegaron a completarse. Analicemos a que invariantes pertenecen las transiciones restantes de forma manual.

T-Invariants

	T1	T10	T11	T12	T2	T3	T4	T5	T6	T7	T8	T9
IT1	1	0	0	0	0	1	0	1	0	1	1	0
IT2	1	0	0	0	1	0	1	0	1	0	1	0
IT3	0	1	1	1	0	0	0	0	0	0	0	1

T1T9T9T1T2T3T1T4T3T1T6

Secuencia Amarillo → IT2

Secuencia Rojo → IT3

Secuencia Verde → IT3

Secuencia Azul → IT2

Secuencia Violeta → IT2

Secuencia Naranja → IT2

Notemos que si bien las invariantes no estan completas, el orden de las transiciones hace coincide con los invariantes T de la red.

Implementación

Hay que definir qué es el sistema, sabiendo que:

- Las plazas {P12, P13, P14, P15, P16, P17} representan recursos compartidos en el sistema.

- Las plazas {P7, P11} son plazas idle, que corresponden a buffers del sistema.

- Las plazas {P1, P2, P3, P4, P5, P6, P8, P9, P10} son plazas donde se realizan actividades relacionadas con el proceso

Decidimos elaborar el proceso de **producción de utensillos (cuchillos y tenedores) de metal**. A continuación se muestra la imagen que describe el proceso y las tablas de estados y eventos.

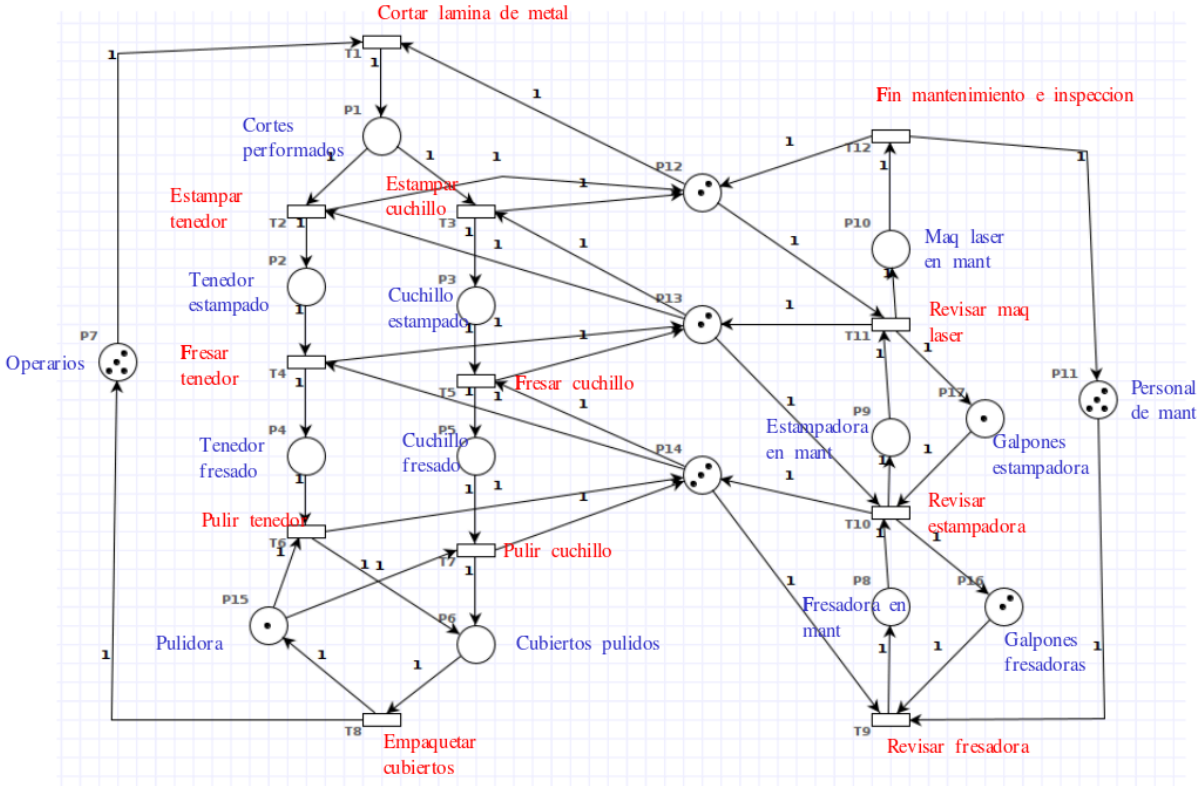


Tabla de estados

P1	Cortes de metal preformados

P2	Tenedor estampado
P3	Cuchillo estampado
P4	Tenedor fresado
P5	Cuchillo fresado
P6	Cubiertos pulidos
P7	Operarios
P8	Fresadora en mantenimiento
P9	Estampadora en mantenimiento
P10	Máquina laser en mantenimiento
P11	Personal de mantenimiento
P12	Máquinas láser
P13	Estampadoras
P14	Fresadoras
P15	Pulidora
P16	Galpones de fresadoras en mantenimiento
P17	Galpón de estampadoras en mantenimiento

Tabla de eventos

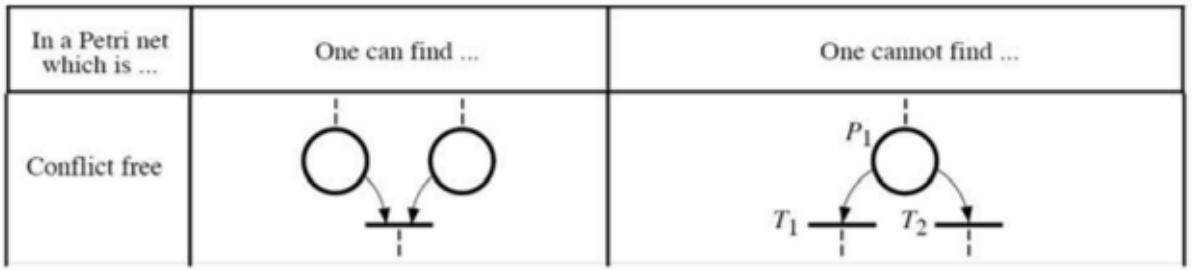
T1	Cortar láminas de metal
T2	Estampar tenedor
T3	Estampar cuchillo
T4	Fresar tenedor
T5	Fresar cuchillo
T6	Pulir tenedor

T7	Pulir cuchillo
T8	Empaquetado de los utensilios
T9	Mantenimiento de Fresadora
TA	Mantenimiento de Estampadora
TB	Estampadora Examinada
TC	Termina el mantenimiento e inspección

Determinación de hilos

¿Cómo se determinan los hilos?

Un conflicto en una RdP es un lugar con dos o más transiciones de salida.

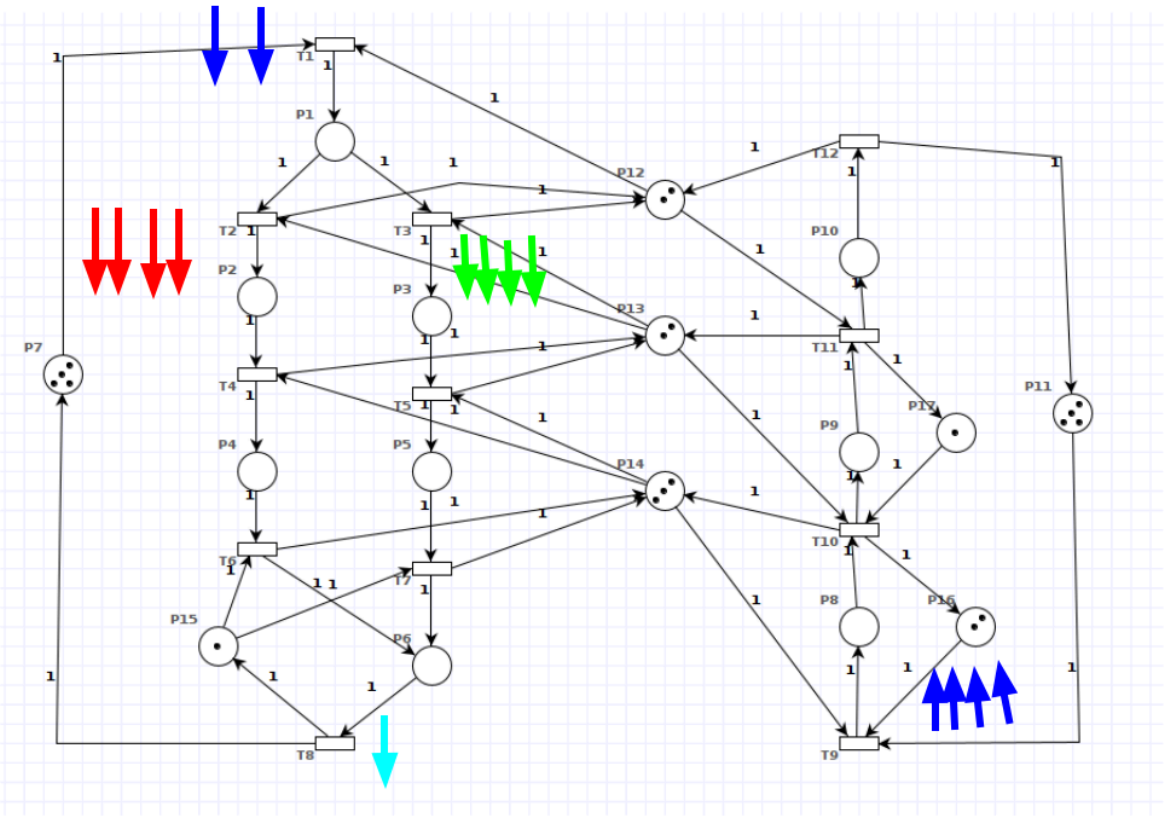


1. Si el invariante de transición no tiene conflictos, un hilo debe estar encargado de la ejecución de las transiciones de ese invariante.

2. Si el invariante de transición tiene un conflicto, con otro invariante, debe haber un hilo encargado de la ejecución de la transición anterior al conflicto y luego un hilo por invariante.

3. Si el invariante de transición presenta un join, con otro invariante de transición, luego del join debe haber tantos hilos, como token simultáneos en la plaza, encargados de las transiciones restantes dado que hay un solo camino.

A partir de estos puntos, determinamos la cantidad de hilos posibles en la red desbloqueada:



Como vemos, son 15 los hilos posibles. Sin embargo, como habíamos analizado anteriormente en nuestra red, el máximo de hilos simultáneos posibles es 8, al igual que en la red original bloqueada:

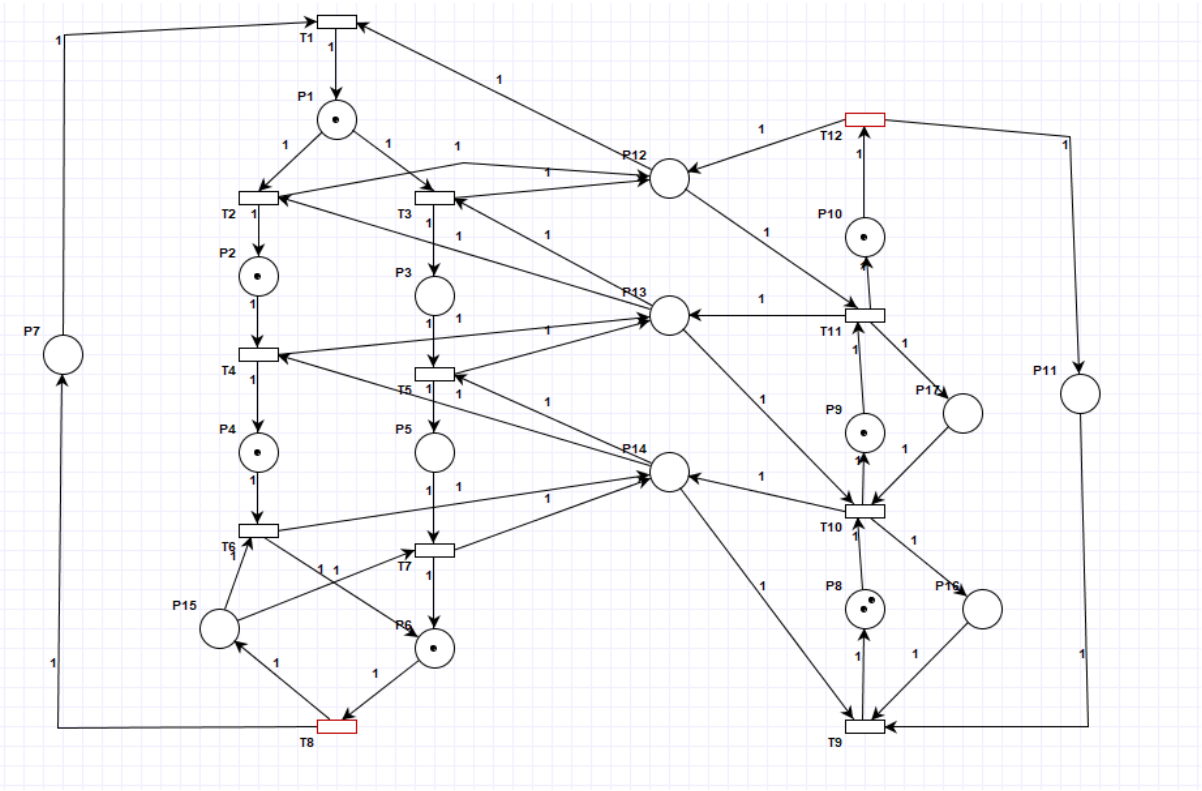
Avg	4,945196211
Max	8

Red bloqueada - original

Avg	4,813343924
Max	8

Red desbloqueada - Alternativa 3

Un ejemplo de la red desbloqueada con el máximo de 8 actividades simultáneas:


T1
P1
T2
T3
P2
P3
T4
T5
P4
P5
T6
T7
P15
P6
T8
P7
P12
P13
P14
T12
P10
T11
P17
P9
T10
P16
P8
T9
P11

Código

Diagrama de clases

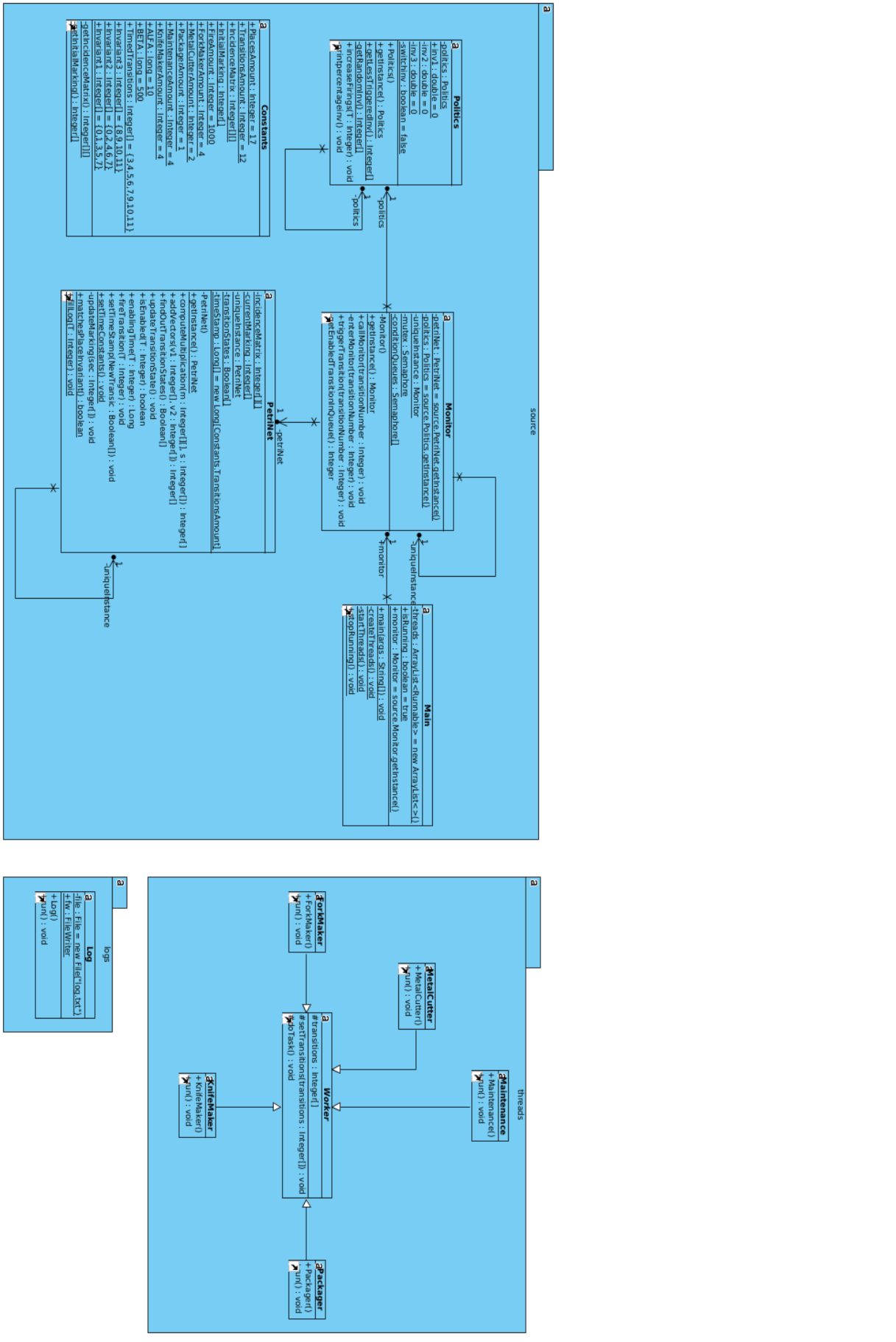
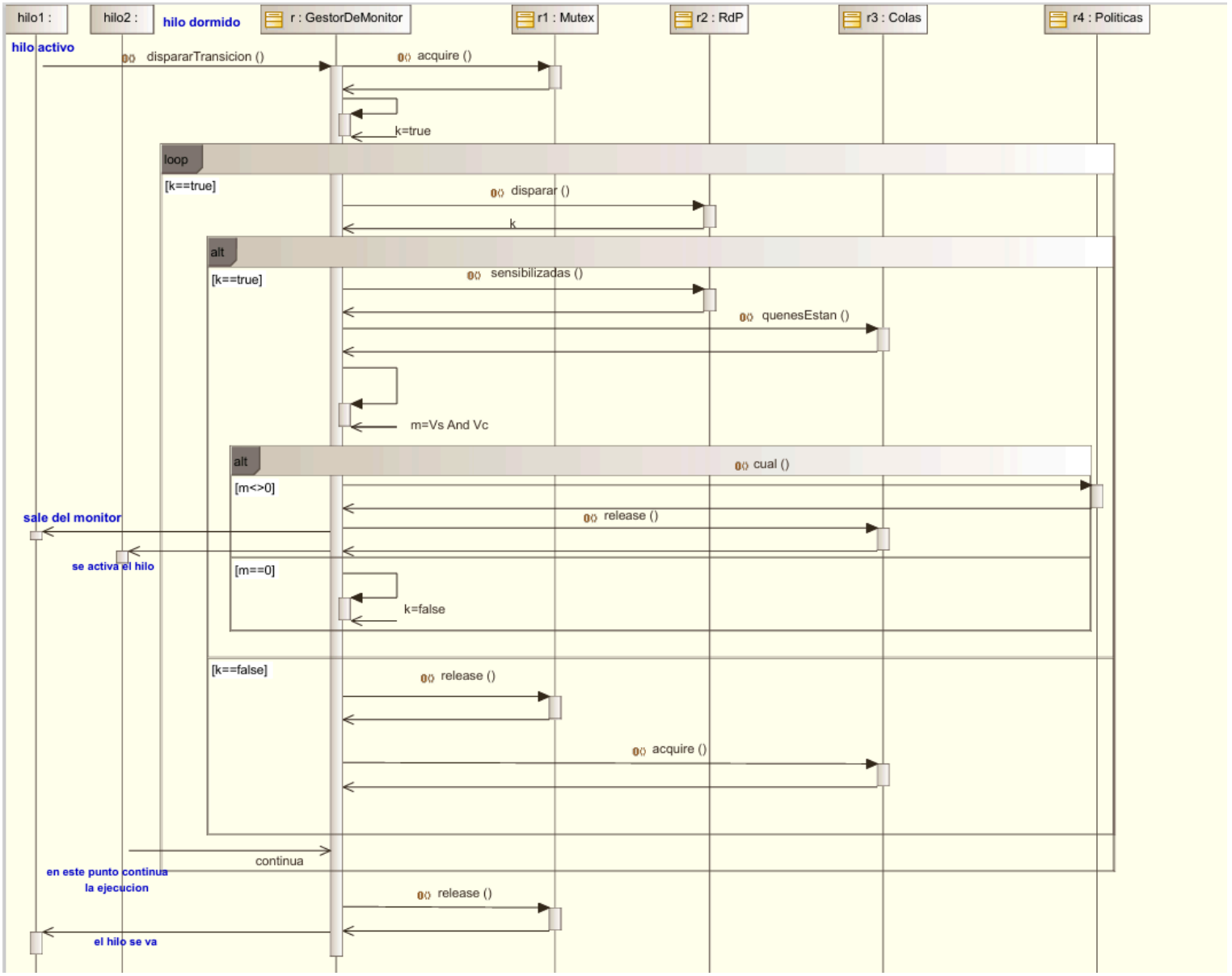
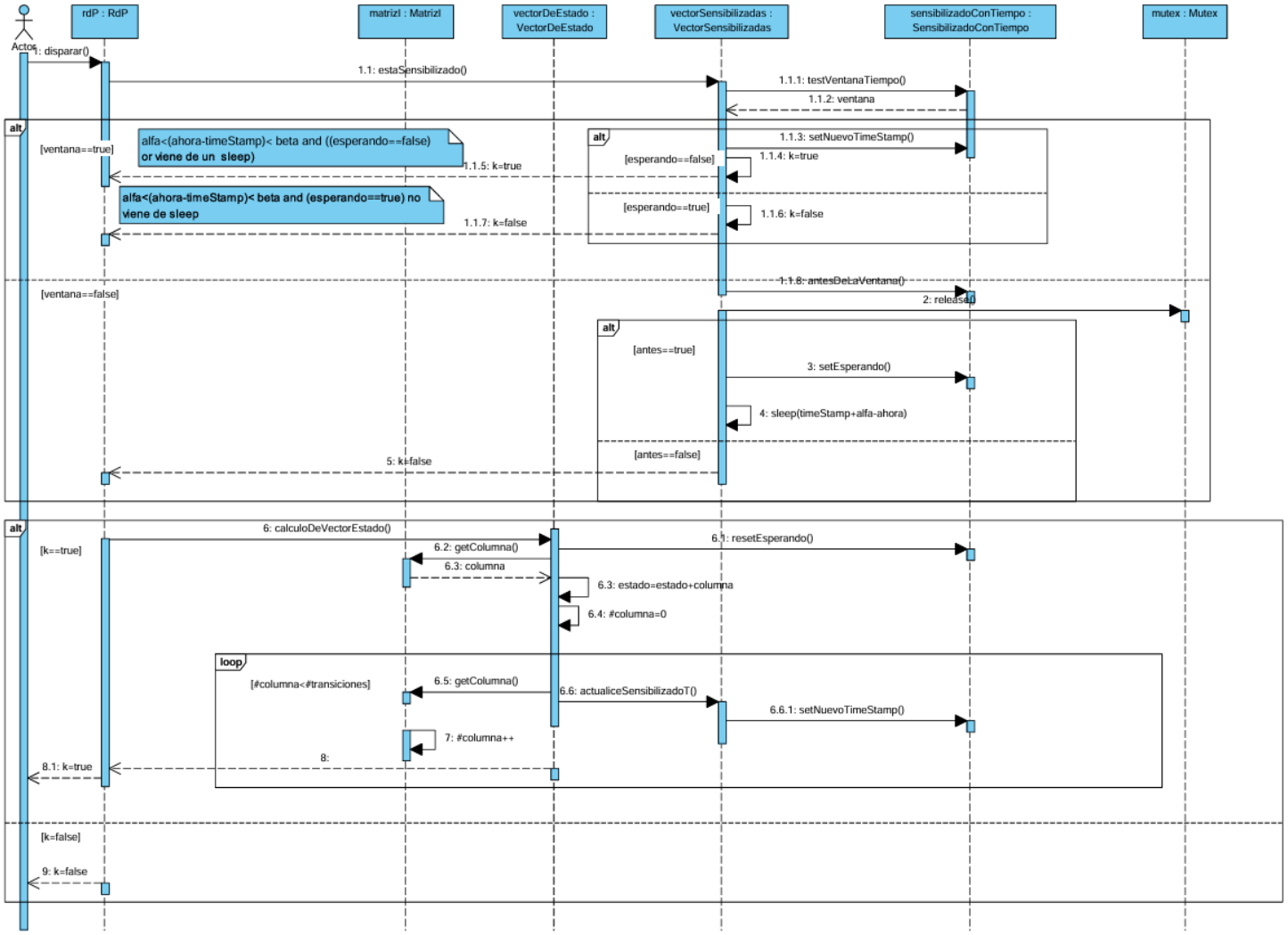


Diagrama de Secuencia

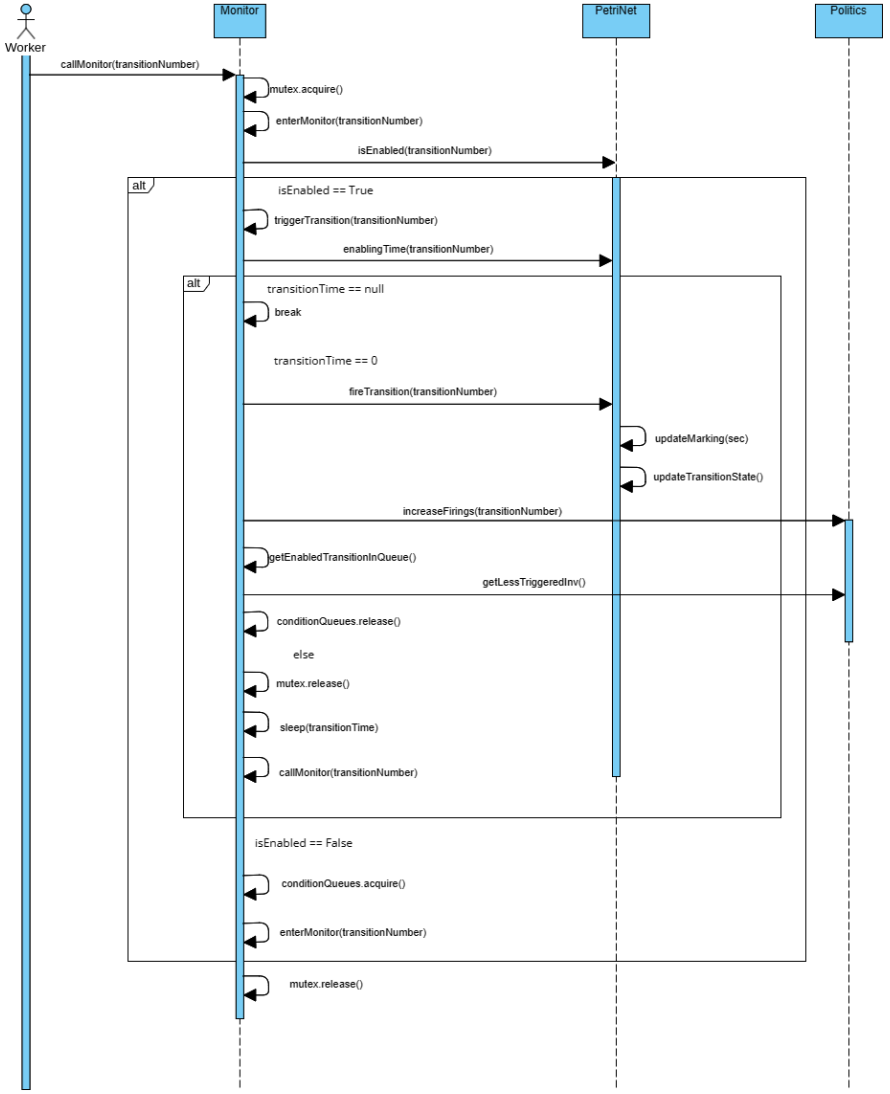
Basándonos en una primera instancia en el siguiente diagrama del LEV:



Y en una segunda instancia al implementar la ventana temporal, nos guiamos con el siguiente diagrama, también del LEV:



Obtuvimos el siguiente diagrama de secuencia en nuestra implementación:



Cumplimiento de Invariantes

Análisis temporal

Con el fin de evaluar el rendimiento de la red, se experimentó variando tres parámetros. Por un lado, se probaron distintos valores de alfa y beta en distintas computadoras. Se concluyó que para intervalos de alfa-beta iguales o superiores a 450 (aproximadamente) milisegundos la red no presenta deadlocks, con lo cual se respeta este intervalo mínimo en los siguientes análisis.

Por otra parte, también se asignaron diferentes prioridades de disparo a los tres invariantes de plaza. Finalmente, en el caso de que los tres invariantes hubieran sido disparados la misma cantidad de veces, se usaron distintos criterios para elegir cual de estos se dispara (consulte el método *isEqual()* en la clase *Politics*). En cada caso, se analizó la distribución del porcentaje total de disparos entre los tres invariantes de plaza.

Caso 1: Invariante 1 con la mayor prioridad

Para este primer caso, se decidió priorizar siempre el disparo del primer invariante por sobre el resto (que en la red representa la línea de mantenimiento). El invariante 1 supera ampliamente a los otros por al menos un 10% en todos los casos, salvo para la situación en donde alfa=1 y beta=5000, donde dicho invariante representa el 64% del total de los disparos.

```
Avg invariante 1: 64.1%
Avg invariante 2: 18.0%
Avg invariante 3: 17.9%
```

alfa=1, beta=5000

```
Avg invariante 1: 42.699999999999996%
Avg invariante 2: 28.7%
Avg invariante 3: 28.599999999999998%
```

alfa=15,beta=5000

```
Avg invariante 1: 41.099999999999994%
Avg invariante 2: 29.4%
Avg invariante 3: 29.5%
```

alfa=50,beta=500

```
Avg invariante 1: 41.0%
Avg invariante 2: 29.5%
Avg invariante 3: 29.5%
```

alfa=100,beta=5000

```
Avg invariante 1: 41.0%
Avg invariante 2: 29.5%
Avg invariante 3: 29.5%
```

alfa=15,beta=3000

Caso 2: Invariante 2 y 3 poseen prioridad frente a un mismo número de disparos en los 3 invariantes

En este caso, en la situación en la que los tres invariantes poseen la misma cantidad de disparos, los invariantes 2 y 3 poseen prioridad frente al invariante 1. Los resultados reflejan que a pesar de la prioridad mencionada, el invariante 1 siempre posee el mayor porcentaje del total de disparos, independientemente de los valores de alfa y beta que se elijan.

```
Avg invariante 1: 70.7%
Avg invariante 2: 15.299999999999999%
Avg invariante 3: 14.000000000000002%
```

alfa=5,beta=5000

```
Avg invariante 1: 40.0%
Avg invariante 2: 30.0%
Avg invariante 3: 30.0%
```

alfa=10,beta=5000

```
Avg invariante 1: 40.2%
Avg invariante 2: 29.9%
Avg invariante 3: 29.9%
```

alfa=20,beta=5000

Caso 3: Se dispara siempre Invariante 2 o 3 frente a un mismo número de disparos en los tres invariantes

Para este caso, si la cantidad de disparos en los 3 invariantes, a diferencia del caso 2, siempre se dispara el invariante 2 o 3 (nunca el invariante 1). Fue bajo esta condición que se logró la mayor equitatividad en el porcentaje de disparos para cada invariante.

```
Avg invariante 1: 35.099999999999994%
Avg invariante 2: 32.4%
Avg invariante 3: 32.5%
```

alfa=1, beta=5000

```
Avg invariante 1: 33.2%
Avg invariante 2: 33.5%
Avg invariante 3: 33.300000000000004%
```

alfa=20, beta=5000

```
Avg invariante 1: 33.2%
Avg invariante 2: 33.300000000000004%
Avg invariante 3: 33.5%
```

alfa=50, beta=5000

```
Avg invariante 1: 33.300000000000004%
Avg invariante 2: 33.300000000000004%
Avg invariante 3: 33.4%
```

alfa=50, beta=3000

```
Avg invariante 1: 33.300000000000004%
Avg invariante 2: 33.4%
Avg invariante 3: 33.300000000000004%
```

alfa=100, beta=5000

Conclusión

Luego de haber analizado nuestra red de Petri y las posibles soluciones al deadlock, llegamos a las siguientes conclusiones.

En primer lugar, identificamos que nuestra red original presentaba un problema de deadlock. Luego, presentamos tres alternativas para resolver este deadlock. Al comparar estas soluciones, observamos que cada una tiene su propio grado de paralelismo y eficiencia, representado por la cantidad promedio y máxima de actividades simultáneas en la red. Observamos que la tercera alternativa mostró el mayor grado de paralelismo, por lo que la elegimos para la implementación

Posteriormente, determinamos la cantidad de hilos necesarios para la ejecución de la red. Basándonos en los conflictos, joins y forks del sistema, la cantidad total de hilos es 15 pero la cantidad máxima de hilos simultáneos es 8. Además, le dimos un propósito a nuestro sistema, interpretando las plazas y transiciones como actividades del proceso de fabricación de cubiertos.

También verificamos el cumplimiento de los invariantes de transición e implementamos una expresión regular. Durante nuestro análisis, nos aseguramos que las transiciones ejecutadas cumplieran con la Regex definida. Esta validación confirma que nuestro sistema opera de acuerdo con las restricciones establecidas.

Luego, implementamos la red en Java, haciendo uso de un monitor para gestionar la concurrencia de los hilos y estableciendo una ventana temporal para la ejecución de las transiciones.

Por último, realizamos un análisis temporal de la red, al mismo tiempo que fuimos implementando distintas políticas y ventanas temporales que mejoren los tiempos, que no provoquen deadlock y que ejecuten los invariantes de forma equitativa.

En resumen, nuestro análisis e implementación nos ha proporcionado una comprensión más profunda de los desafíos asociados con la programación concurrente y la resolución de deadlock en redes de Petri. Al considerar cuidadosamente las diferentes alternativas y sus implicaciones, podemos tomar decisiones informadas para mejorar la eficiencia de nuestro sistema.

Bibliografía y Herramientas

Regex101
Debuggex
PIPE

"Redes de Petri: Modelado e implementación de algoritmos para autómatas programables", *Murillo, Luis Diego,* Tecnología en Marcha, Vol. 21, N.° 4, Octubre-Diciembre 2008, pp. 102-125

"Algoritmos para determinar cantidad y responsabilidad de hilos en sistemas embebidos modelados con Redes de Petri S3PR" *Micolini, Orlando. Ventre, Luis,* Octubre 2021

"Ecuación de estado generalizada para redes de Petri no autónomas y con distintos tipos de arcos" *Micolini, Orlando. Ventre, Luis*.