UNIVERSIDAD NACIONAL DE CÓRDOBA

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

CÁTEDRA DE PROGRAMACIÓN CONCURRENTE

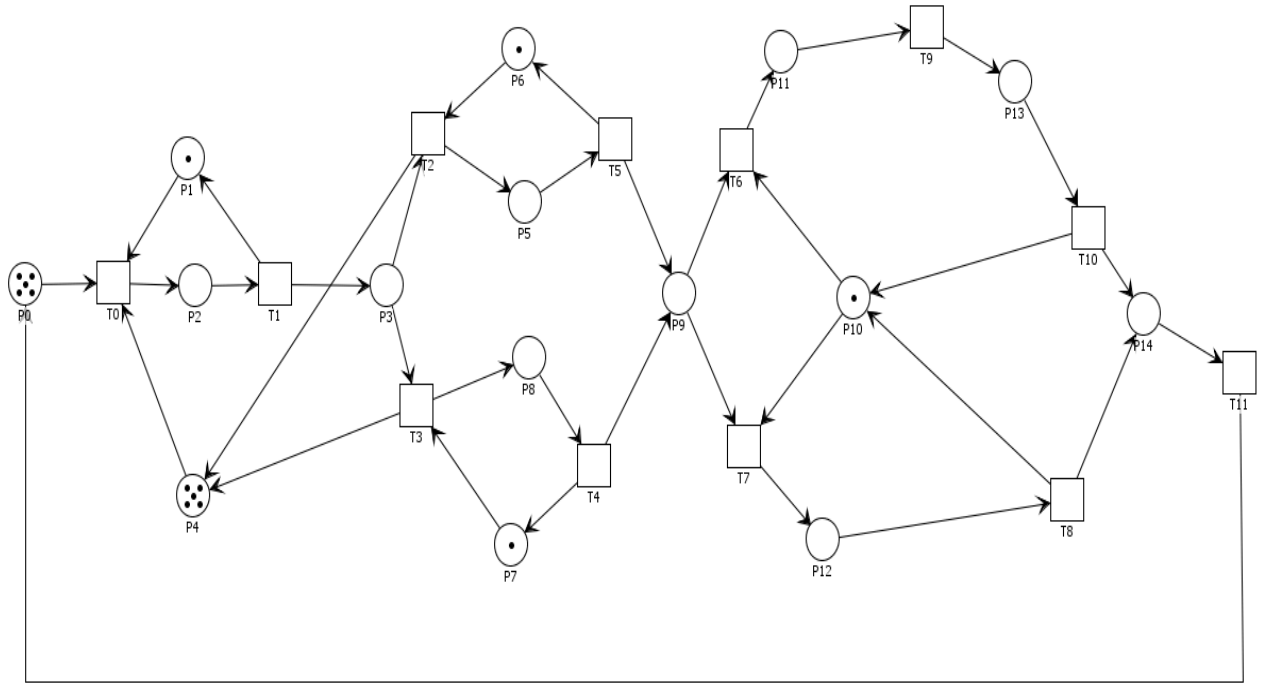
TRABAJO PRÁCTICO Nº 2

“Sistema de reservas de vuelos”

Grupo Nº 8

Integrantes		
Nombre	DNI	email
Piñera Nicolas	44195541	nicolas.pinera@mi.unc.edu.ar
Ñañez Josias Tomas	41886293	josias.nanez@mi.unc.edu.ar
Fatu Javier Alejandro	44741112	javier.fatu@mi.unc.edu.ar

Análisis red de petri



Esta red de Petri modela un sistema de agencia de viajes. Las plazas $\{P_1, P_4, P_6, P_7, P_{10}\}$ representan recursos compartidos en el sistema.

La plaza $\{P_0\}$ es una plaza idle que corresponde al buffer de entrada de clientes de la agencia.

La plaza $\{P_2\}$ representa el ingreso a la agencia.

La plaza $\{P_3\}$ representa la sala de espera de la agencia.

Las plazas $\{P_5, P_8\}$ representan los estados en los cuales se realiza la gestión de las reservas.

La plaza $\{P_9\}$ representa la espera de los clientes para pasar por el agente que aprueba o rechaza la reserva.

En la plaza $\{P_{12}\}$ se modela la cancelación de la reserva.

En las plazas $\{P_{11}, P_{13}\}$ se modela la confirmación y el pago respectivamente de la reserva.

En la plaza $\{P_{14}\}$ se modela la instancia previa a que el cliente se retire.

Para realizar el análisis de la red de petri decidimos utilizar el simulador petrinator

Matriz de post-incidencia W^+:

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
P0	0	0	0	0	0	0	0	0	0	0	0	1
P1	0	1	0	0	0	0	0	0	0	0	0	0
P2	1	0	0	0	0	0	0	0	0	0	0	0
P3	0	1	0	0	0	0	0	0	0	0	0	0
P4	0	0	1	1	0	0	0	0	0	0	0	0
P5	0	0	1	0	0	0	0	0	0	0	0	0
P6	0	0	0	0	0	1	0	0	0	0	0	0
P7	0	0	0	0	1	0	0	0	0	0	0	0
P8	0	0	0	1	0	0	0	0	0	0	0	0
P9	0	0	0	0	1	1	0	0	0	0	0	0
P10	0	0	0	0	0	0	0	0	1	0	1	0
P11	0	0	0	0	0	0	1	0	0	0	0	0
P12	0	0	0	0	0	0	0	1	0	0	0	0
P13	0	0	0	0	0	0	0	0	0	1	0	0
P14	0	0	0	0	0	0	0	0	1	0	1	0

Matriz de pre-incidencia W^-:

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
P0	1	0	0	0	0	0	0	0	0	0	0	0
P1	1	0	0	0	0	0	0	0	0	0	0	0
P2	0	1	0	0	0	0	0	0	0	0	0	0
P3	0	0	1	1	0	0	0	0	0	0	0	0
P4	1	0	0	0	0	0	0	0	0	0	0	0
P5	0	0	0	0	0	1	0	0	0	0	0	0
P6	0	0	1	0	0	0	0	0	0	0	0	0
P7	0	0	0	1	0	0	0	0	0	0	0	0
P8	0	0	0	0	1	0	0	0	0	0	0	0
P9	0	0	0	0	0	0	1	1	0	0	0	0
P10	0	0	0	0	0	0	1	1	0	0	0	0
P11	0	0	0	0	0	0	0	0	0	1	0	0
P12	0	0	0	0	0	0	0	0	1	0	0	0
P13	0	0	0	0	0	0	0	0	0	0	1	0
P14	0	0	0	0	0	0	0	0	0	0	0	1

Matriz de incidencia $W = W^{+} - W^{-}$***:***

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
P0	-1	0	0	0	0	0	0	0	0	0	0	1
P1	-1	1	0	0	0	0	0	0	0	0	0	0
P2	1	-1	0	0	0	0	0	0	0	0	0	0
P3	0	1	-1	-1	0	0	0	0	0	0	0	0
P4	-1	0	1	1	0	0	0	0	0	0	0	0
P5	0	0	1	0	0	-1	0	0	0	0	0	0
P6	0	0	-1	0	0	1	0	0	0	0	0	0
P7	0	0	0	-1	1	0	0	0	0	0	0	0
P8	0	0	0	1	-1	0	0	0	0	0	0	0
P9	0	0	0	0	1	1	-1	-1	0	0	0	0
P10	0	0	0	0	0	0	-1	-1	1	0	1	0
P11	0	0	0	0	0	0	1	0	0	-1	0	0
P12	0	0	0	0	0	0	0	1	-1	0	0	0
P13	0	0	0	0	0	0	0	0	0	1	-1	0
P14	0	0	0	0	0	0	0	0	1	0	1	-1

Invariantes de transición:

T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
1	1	0	1	1	0	0	1	1	0	0	1
1	1	0	1	1	0	1	0	0	1	1	1
1	1	1	0	0	1	0	1	1	0	0	1
1	1	1	0	0	1	1	0	0	1	1	1

Los invariantes de transición son aquellas secuencias de disparo de transiciones, la cual nos devuelve el marcado inicial de la red. Podemos ver que para las siguientes secuencias de disparo, la red mantiene el mismo marcado.

$$T_0 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_7 \rightarrow T_8 \rightarrow T_{11}$$

$$T_0 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_6 \rightarrow T_9 \rightarrow T_{10} \rightarrow T_{11}$$

$$T_0 \rightarrow T_1 \rightarrow T_2 \rightarrow T_5 \rightarrow T_7 \rightarrow T_8 \rightarrow T_{11}$$

$$T_0 \rightarrow T_1 \rightarrow T_2 \rightarrow T_5 \rightarrow T_6 \rightarrow T_9 \rightarrow T_{10} \rightarrow T_{11}$$

Invariantes de Plaza:

P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	1	1	1	0
1	0	1	1	0	1	0	0	1	1	0	1	1	1	1

Los invariantes de plaza son un conjunto de plazas cuyo número total de tokens se mantiene constante a lo largo del tiempo, independientemente de las transiciones que se disparen. Podemos observar los siguientes invariantes de plaza

1) $m(P_1) + m(P_2) = 1$

2) $m(P_2) + m(P_3) + m(P_4) = 5$

3) $m(P_5) + m(P_6) = 1$

4) $m(P_7) + m(P_8) = 1$

5) $m(P_{10}) + m(P_{11}) + m(P_{12}) + m(P_{13}) = 1$

6) $m(P_0) + m(P_2) + m(P_3) + m(P_5) + m(P_8) + m(P_9) + m(P_{11}) + m(P_{12}) + m(P_{13}) + m(P_{14}) = 5$

En la ecuación 1) podemos ver que la cantidad de personas que van a pasar es una por vez.

En la ecuación 2) vemos que la cantidad máxima de personas que puede haber en la sala de espera es de cinco.

En las ecuaciones 3) y 4) vemos que cada agente de reserva atenderá a una sola persona por vez.

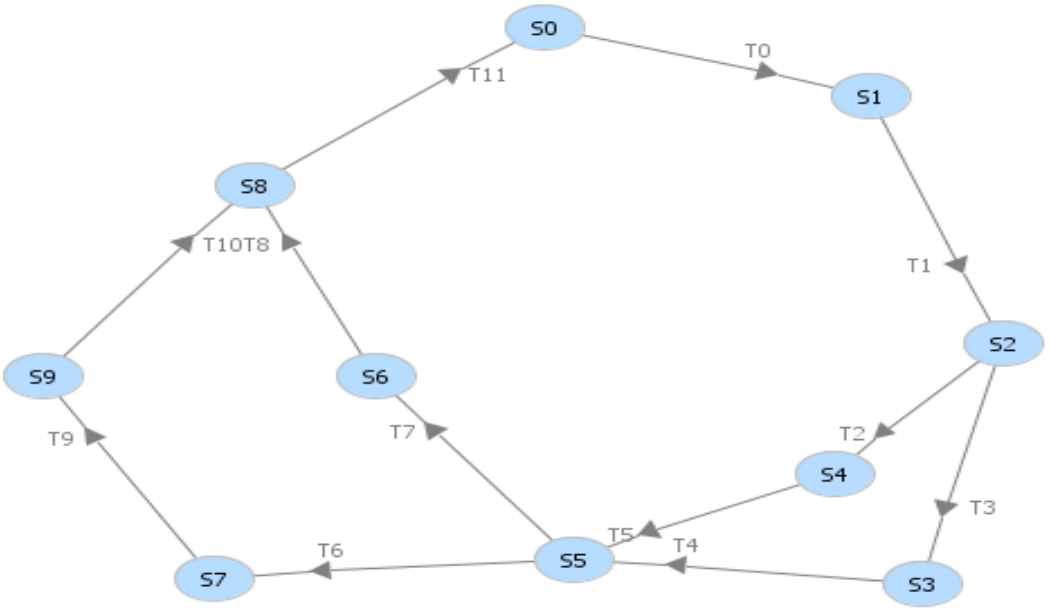
En la ecuación 5) vemos que se verificará el pago o cancelación de una persona a la vez.

En este caso para poder estudiar las propiedades de la red, los clientes que salen vuelven a entrar en la agencia eso es lo que observamos en la ecuación 6), la cantidad de clientes es de cinco.

Grafo de alcanzabilidad y deadlock:

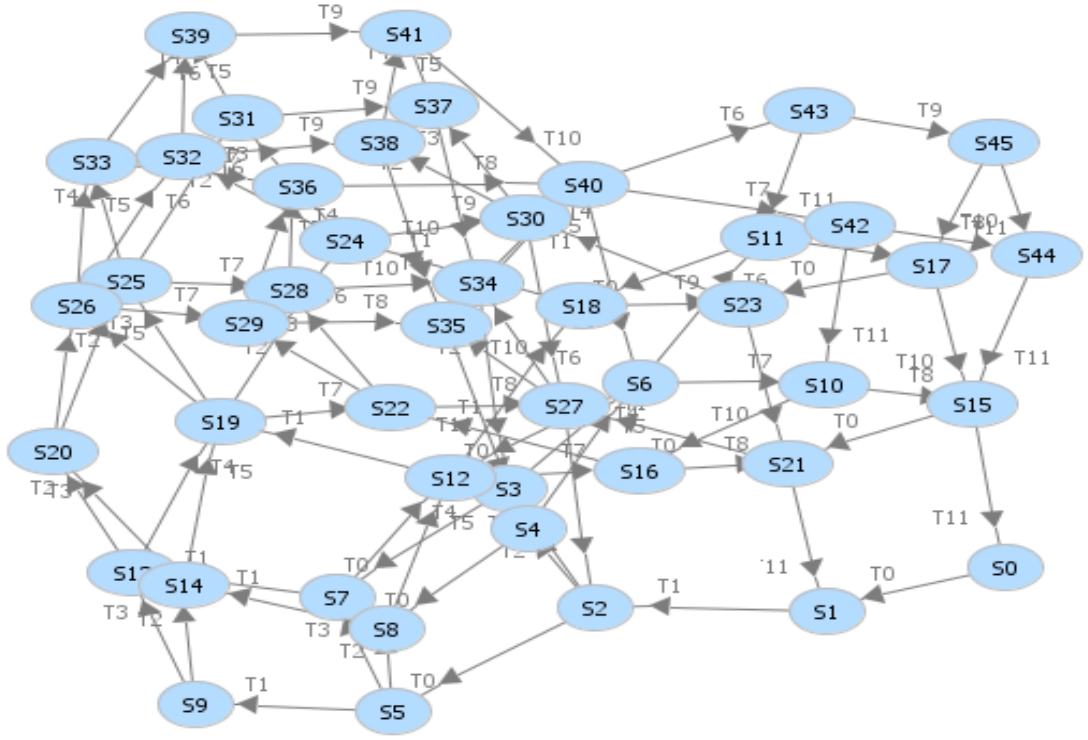
Marcado inicial: $(1, 1, 0, 0, 5, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0)$

En el grafo de alcanzabilidad podemos notar que no se produce deadlock, también vemos los invariantes de transición, por ejemplo, si disparamos la secuencia $T_0 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_7 \rightarrow T_8 \rightarrow T_{11}$ regresamos al marcado inicial.



.

Grafo de alcanzabilidad con marcado inicial $(2, 1, 0, 0, 5, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0)$



Vemos que al aumentar el marcado de la plaza P_0 El número de combinaciones de transición crece exponencialmente pero seguimos viendo que las propiedades se mantienen.

Seguridad:

Una red de petri se denomina segura cuando el marcado máximo de cada una de sus plazas es uno. Podemos ver que en la plazas que representan salas de espera como por ejemplo P_3 pueden esperar más de un cliente por lo tanto esta red de petri no se considera segura.

Tabla de estados y eventos:

Estado inicial	Evento	Estado final
No hay clientes	Arriba una persona	Una persona está entrando
Una persona está entrando	Termina de pasar	El cliente está en la sala de espera
El cliente está en la sala de espera	El cliente pasa a ser atendido por el agente superior	El cliente está siendo atendido por agente superior
El cliente está en la sala de espera	El cliente pasa a ser atendido por el agente inferior	El cliente está siendo atendido por agente inferior
El cliente está siendo atendido por el agente superior	El cliente termina de ser atendido	El cliente espera a ser aprobado o rechazado
El cliente está siendo atendido por el agente inferior	El cliente termina de ser atendido	El cliente espera a ser aprobado o rechazado
El cliente espera a ser aprobado o rechazado	Se acepta la reserva	La reserva está aceptada

El cliente espera a ser aprobado o rechazado	Se rechaza la reserva	La reserva está rechaza
La reserva está aceptada	Se efectúa el pago	La reserva está pagada
La reserva está rechaza	Se despide al cliente	El cliente se va
La reserva está pagada	Se despide al cliente	El cliente se va

Tabla de estados			
Tabla de estados del sistema		**Tabla de eventos del sistema**	
P0	Entrada de clientes de la agencia.	T0	Entra una persona
P1	Limita entrada clientes		
P2	Cliente entrando	T1	Pasa a la sala de espera
P3	Sala de espera para hacer reserva		
P4	Libera lugar en sala de espera	T2	Pasa a ser atendido por el agente superior
P5	Cliente atendido por agente superior		
P6	Agente superior	T3	Pasa a ser atendido por el agente inferior
P7	Agente inferior	T4	Termina de ser atendido por el agente inferior

P8	Cliente atendido por agente inferior	T5	Termina de ser atendido por el agente superior
P9	Sala de espera para ser aprobado o rechazado	T6	Se acepta la reserva
P10	Agente que aprueba, cobra y rechaza	T7	Se cancela la reserva
P11	Aceptación de reserva	T8	Termina de ser atendido
P12	Cancelación de reserva	T9	Se paga su reserva
P13	Pago de reserva	T10	Termina de ser atendido
P14	Salida de la agencia	T11	Cliente se retira

Determinación de hilos máximos activos simultáneos

1) IT de la red:

$$IT_1 = \{T_0, T_1, T_3, T_4, T_7, T_8, T_{11}\}$$

$$IT_2 = \{T_0, T_1, T_3, T_4, T_6, T_9, T_{10}, T_{11}\}$$

$$IT_3 = \{T_0, T_1, T_2, T_5, T_7, T_8, T_{11}\}$$

$$IT_4 = \{T_0, T_1, T_2, T_5, T_6, T_9, T_{10}, T_{11}\}$$

2) Obtenemos el conjunto de plazas PI de cada IT:

$$PI_1 = \{P_0, P_1, P_2, P_3, P_4, P_7, P_8, P_9, P_{12}, P_{10}, P_{14}\}$$

$$PI_2 = \{P_0, P_1, P_2, P_3, P_4, P_7, P_8, P_9, P_{11}, P_{13}, P_{10}, P_{14}\}$$

$$PI_3 = \{P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_9, P_{12}, P_{10}, P_{14}\}$$

$$PI_4 = \{P_0, P_1, P_2, P_3, P_4, P_5, P_6, P_9, P_{11}, P_{13}, P_{10}, P_{14}\}$$

3) Obtenemos el conjuntos de plazas de acción PA de cada IT

$$PA_1 = \{P_2, P_3, P_8, P_9, P_{12}, P_{14}\}$$

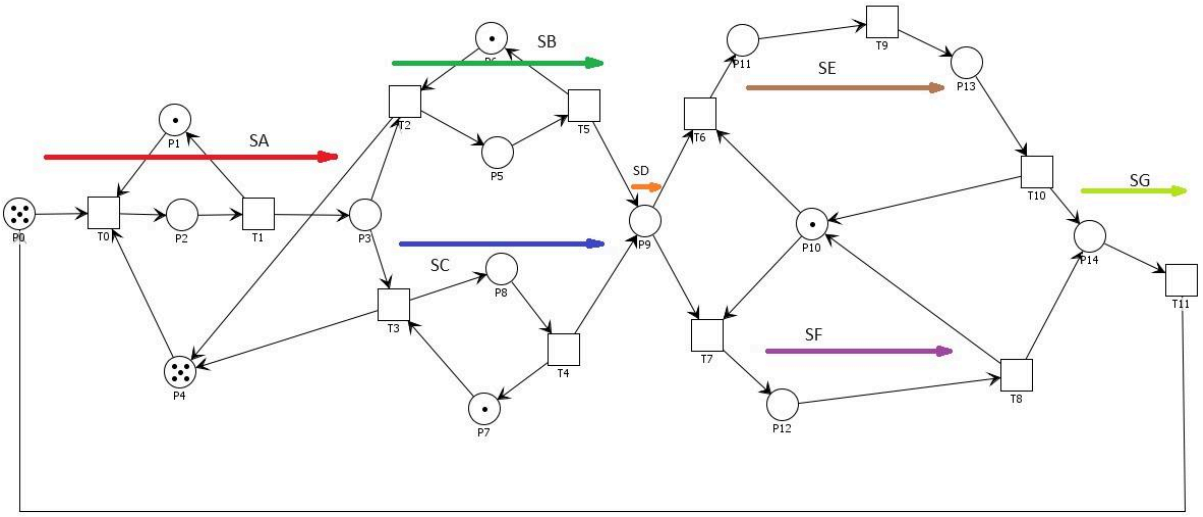
$$PA_2 = \{P_2, P_3, P_8, P_9, P_{11}, P_{13}, P_{14}\}$$

$$PA_3 = \{P_2, P_3, P_5, P_9, P_{12}, P_{14}\}$$

$$PA_4 = \{P_2, P_3, P_5, P_9, P_{11}, P_{13}, P_{14}\}$$

$$PA = \{P_2, P_3, P_5, P_8, P_9, P_{11}, P_{12}, P_{13}, P_{14}\}$$

4) De todos los marcados posibles de PA Se busca el marcado máximo, este valor determina la máxima cantidad de hilos activos simultáneos. En nuestro caso son 5 hilos.



La red cuenta con 7 segmentos: el segmento S_A previo al primer conflicto (fork), el segmento superior S_B e inferior S_C, luego el segmento S_D que cumple con la condición de unión y de conflicto, luego sigue el segmento S_E y S_F y por último el segmento de unión S_G. Una vez determinados los segmentos, calculamos los hilos máximos necesarios por segmento de ejecución.

$$PS_A = \{P_2, P_3\} \;\; Max(MS_A) = 5$$

$$PS_B = \{P_5\} \;\; Max(MS_B) = 1$$

$$PS_C = \{P_8\} \;\; Max(MS_C) = 1$$

$$PS_D = \{P_9\} \;\; Max(MS_D) = 5$$

$$PS_E = \{P_{11}, P_{13}\} \;\; Max(MS_E) = 1$$

$$PS_F = \{P_{12}\} \;\; Max(MS_F) = 1$$

$$PS_G = \{P_{14}\} \;\; Max(MS_G) = 5$$

Debido que la plaza P_9 es un fork y join por lo tanto las siguientes transiciones cuentan con un conflicto decidimos no utilizar los 5 hilos de ese segmento y utilizar los hilos de los segmentos siguientes al conflicto.

Como el simulado de los clientes termina con las transiciones T_{10} y T_8 decidimos no utilizar la transición T_{11} y por lo tanto no usar el segmento S_G.

Una vez determinados los hilos máximos por segmento, la suma de todos estos determina la máxima cantidad de hilos necesarios del sistema. En nuestro caso la cantidad de hilos necesarios es de 9 hilos.

Analisis de politicas e Invariantes

Analisis de politicas e Invariantes									
Política	**Ejec**	**IT 1**	**%**	**IT 2**	**%**	**IT3**	**%**	**IT 4**	**%**
1	1	0	0%	93	50%	93	50%	0	0%
	2	0	0%	93	50%	93	50%	0	0%
	3	1	1%	92	49%	92	49%	1	1%
	4	0	0%	93	50%	93	50%	0	0%
	5	0	0%	93	50%	93	50%	0	0%
	6	0	0%	93	50%	93	50%	0	0%
	7	0	0%	93	50%	93	50%	0	0%
	8	0	0%	93	50%	93	50%	0	0%
	9	0	0%	93	50%	93	50%	0	0%
	10	0	0%	93	50%	93	50%	0	0%
	Total	1	0%	929	50%	929	50%	1	0%
2	1	10	5%	37	20%	27	15%	112	60%
	2	10	5%	37	20%	27	15%	112	60%
	3	10	5%	37	20%	27	15%	112	60%
	4	10	5%	37	20%	27	15%	112	60%
	5	10	5%	37	20%	27	15%	112	60%
	6	10	5%	37	20%	27	15%	112	60%
	7	10	5%	37	20%	7	4%	112	60%
	8	10	5%	37	20%	27	15%	112	60%
	9	10	5%	37	20%	27	15%	112	60%
	10	10	5%	37	20%	27	15%	112	60%
	Total	100	5%	370	20%	250	14%	1120	61%

En esta tabla podemos ver un total de 20 ejecuciones de nuestro programa, 10 ejecuciones utilizando la política balanceada y 10 con la política priorizada. En cada ejecución tomamos la secuencia de disparo total de la red y lo analizamos con nuestra expresión regular, tenemos 186 invariantes de transición disparados, los cuales son:

- **Invariante de Transición 1: T0T1T3T4T7T8T11** Representa a un cliente que ingresa a la agencia, es atendido por el agente inferior, y luego su reserva es cancelada.
- **Invariante de Transición 2: T0T1T3T4T6T9T10T11**: Representa a un cliente que ingresa a la agencia, es atendido por el agente inferior y luego su reserva es confirmada y pagada.
- **Invariante de Transición 3: T0T1T2T5T7T8T11**: Representa a un cliente que ingresa a la agencia, es atendido por el agente superior y luego su reserva es cancelada.
- **Invariante de Transición 4: T0T1T2T5T6T9T10T11**: Representa a un cliente que ingresa a la agencia, es atendido por el agente superior y luego su reserva es confirmada y pagada.

Luego de este análisis podemos ver el correcto funcionamiento de las políticas, ya que en la balanceada tenemos un 50% de los clientes atendido por cada agente y un 50% de las reservas son canceladas y la otra mitad es confirmada y pagada.

Al ejecutar la política 2, vemos que si sumamos los porcentajes de los invariantes 3 y 4 obtenemos que el 75% de los clientes son atendidos por el agente superior y si sumamos los invariantes 2 y 4 obtenemos que un 81% de las reservas son confirmadas y pagadas. Hay que considerar que la política funciona para estos tiempos ya que se genera conflictos efectivos constantemente lo cual hace decidir a la política.

Análisis de tiempos

	T1 [s]	T4 [s]	T5 [s]	T8 [s]	T9 [s]	T10 [s]	Tiempo Total si varia T1[s]
1	0,11	0,15	0,15	0,04	0,03	0,05	20,8
2	0,13	0,15	0,15	0,04	0,03	0,05	24,5
3	0,15	0,15	0,15	0,04	0,03	0,05	28,3
Tiempo Inicial	0,16	0,15	0,15	0,04	0,03	0,05	30,2
5	0,17	0,15	0,15	0,04	0,03	0,05	32
6	0,19	0,15	0,15	0,04	0,03	0,05	35,7
7	0,21	0,15	0,15	0,04	0,03	0,05	39,4
	T1 [s]	**T4 [s]**	**T5 [s]**	**T8 [s]**	**T9 [s]**	**T10 [s]**	**Tiempo Total si varía T4 y T5[s]**
1	0,16	0,1	0,1	0,04	0,03	0,05	30,1
2	0,16	0,12	0,12	0,04	0,03	0,05	30,1
3	0,16	0,14	0,14	0,04	0,03	0,05	30,1
Tiempo Inicial	0,16	0,15	0,15	0,04	0,03	0,05	30,2
5	0,16	0,16	0,16	0,04	0,03	0,05	30,2
6	0,16	0,18	0,18	0,04	0,03	0,05	30,2
7	0,16	0,2	0,2	0,04	0,03	0,05	30,3
	T1 [s]	**T4 [s]**	**T5 [s]**	**T8 [s]**	**T9 [s]**	**T10 [s]**	**Tiempo Total si varía T8[s]**
1	0,16	0,15	0,15	0,02	0,03	0,05	30,1
2	0,16	0,15	0,15	0,03	0,03	0,05	30,1
3	0,16	0,15	0,15	0,035	0,03	0,05	30,2
Tiempo Inicial	0,16	0,15	0,15	0,04	0,03	0,05	30,2
5	0,16	0,15	0,15	0,045	0,03	0,05	30,2
6	0,16	0,15	0,15	0,5	0,03	0,05	30,2
7	0,16	0,15	0,15	0,6	0,03	0,05	30,2

	T1 [s]	T4 [s]	T5 [s]	T8 [s]	T9 [s]	T10 [s]	Tiempo Total si varía T9[s]
1	0,16	0,15	0,15	0,04	0,01	0,3	30,1
2	0,16	0,15	0,15	0,04	0,015	0,3	30,1
3	0,16	0,15	0,15	0,04	0,02	0,3	30,1
Tiempo Inicial	0,16	0,15	0,15	0,04	0,03	0,05	30,2
5	0,16	0,15	0,15	0,04	0,04	0,3	30,1
6	0,16	0,15	0,15	0,04	0,05	0,3	30,2
7	0,16	0,15	0,15	0,04	0,06	0,3	30,4
	T1 [s]	**T4 [s]**	**T5 [s]**	**T8 [s]**	**T9 [s]**	**T10 [s]**	**Tiempo Total si varía T10[s]**
1	0,16	0,15	0,15	0,04	0,03	0,03	30
2	0,16	0,15	0,15	0,04	0,03	0,04	30,1
3	0,16	0,15	0,15	0,04	0,03	0,045	30,1
Tiempo Inicial	0,16	0,15	0,15	0,04	0,03	0,05	30,2
5	0,16	0,15	0,15	0,04	0,03	0,06	30,2
6	0,16	0,15	0,15	0,04	0,03	0,07	30,2
7	0,16	0,15	0,15	0,04	0,03	0,08	30,2

Podemos apreciar que la mayor variación del tiempo total se da cuando variamos el tiempo de la transición T1 que simula la entrada de clientes, esto se puede deber a que a menores tiempo en T1 los demás procesos se ejecutan casi al mismo tiempo sin tener que esperar la llegada de un nuevo cliente, mientras un cliente está siendo atendido otros entran, al aumentar el tiempo de la entrada, las otras transiciones se sensibilizan y disparan, al querer continuar con el siguiente cliente se encuentran con que todavía no ingreso ninguno y los hilos encargados de las reservas, cancelación, confirmación y pago, duermen esperando que un nuevo cliente ingrese a la agencia.

En conclusión, un menor tiempo de T1 genera una sobrecarga de los procesos siguientes y por lo tanto es probable que las políticas tampoco tengan oportunidad de decidir.

Invariante	Tiempo Teórico [ms]	Tiempo código [ms]
T0T1T2T5T7T8T11	350	441
T0T1T2T5T6T9T10T11	390	448

En la tabla anterior vemos que el tiempo teórico de disparar el invariante T0T1T2T5T7T8T11 es de 350 [ms] esto se debe a la suma de los tiempos de las transiciones en este caso $160\,[ms] + 150\,[ms] + 40\,[ms] = 350\,[ms]$ mientras que para el invariante T0T1T2T5T6T9T10T1 el tiempo teórico es de $160\,[ms] + 150\,[ms] + 30\,[ms] + 50\,[ms] = 390\,[ms]$ vemos que el tiempo que tarda en ejecutarse los invariantes es de $441\,[ms]$ y de $448\,[ms]$ esta diferencia se puede deber a los tiempos que se tarda en hacer los demás procesos como por ejemplo los cálculos del monitor, pero podemos apreciar com el invariante T0T1T2T5T6T9T10T1 tarda mas que el invariante T0T1T2T5T7T8T11.

También vemos que el tiempo promedio de la política 1 es de 30,2 segundos y el de la política 2 es de 31,3 segundos, esta diferencia se debe a que en la política 2 el 80 % de los clientes ingresan al proceso de pagar y confirmar que tarda más que el proceso de cancelación.

Informe de Código

A continuación se detallan todas las clases de nuestro trabajo y cada uno de sus variables y métodos.

Clase: Atencion Agente

En esta clase se modela la atención de un cliente por un agente de viajes para realizar la gestión del viaje, se implementa la interfaz Runnable. En la misma tenemos el número de agente para saber si es el agente número 1 (superior) o el número 2 (inferior) ya que nuestra agencia de vueltos tiene 2 agentes, y por último tenemos acceso a la única instancia de nuestro monitor.

Método Run

Dentro de este método nos fijamos que agente es el que está por atender, según esta información vamos a disparar la transición 2 si el que atiende es el agente 1 y una vez finalizada la atención dispara la transición 5 o la transición 3 si atiende el agente 2 y una vez finalizada la atención se dispara la transición 4. Este proceso tiene una duración de 150 milisegundos el cual lo simulamos con un sleep.

Clase: Cancelación

En esta clase se modela el proceso de cancelación de la reserva de viaje, la misma implementa la interfaz Runnable, y tiene acceso a la única instancia de nuestro monitor.

Método Run

Dentro del método se dispara la transición 7 la cual hace que los clientes entren al proceso de cancelación de viaje, este proceso dura 40 milisegundos, luego se dispara la transición 8 la cual hace que los clientes pasen a la etapa previa a que se retiren de la agencia y por último la transición 11 para que se retiren.

Clase: Confirmación y Pago

En esta clase se modela tanto el proceso de confirmación como el pago de dicha reserva, la misma implementa la interfaz Runnable, y tiene acceso a la única instancia de nuestro monitor.

Método Run

Dentro del método se dispara la transición 6 la cual hace que los clientes entren el proceso de confirmación del viaje, este proceso dura 30 milisegundos, luego se dispara la transición 9 pasando a realizar el proceso de pago que dura 50 milisegundos y finalizando con la transición 10, lo cual lleva al cliente a la plaza 14 de clientes salientes y por último se dispara la transición 11 para que el cliente se retire de la agencia.

Clase: Entrada de clientes

Esta clase modela la entrada de los clientes a nuestra agencia de viajes, la misma implementa la interfaz runnable y tiene acceso a la única instancia de nuestro monitor.

Método Run

Dentro de este método, se dispara la transición 0 la cual hace que los clientes ingresen a la agencia, este proceso demora 160 milisegundos y luego se dispara la transición 1 para que los clientes pasen a la sala de espera de la agencia.

Clase: Locks

Esta clase se utiliza como cerrojo de los bloques synchronized de los diferentes métodos dentro del monitor.

Clase: Main

Es nuestra clase principal. Primero se inicializa un scanner para poder pedir por consola un número entero entre 1 y 2 el cual representa la política que implementará el monitor sobre los conflictos estructurales que se generen, siendo 1 la política balanceada donde la cantidad de clientes atendidos por el agente de reservas 1, debe ser equitativo a la cantidad de clientes atendidos por el agente de reservas 2. La cantidad de reservas canceladas debe ser equitativa a la cantidad de reservas confirmadas.

Y 2 una política de procesamiento priorizada en la cual Se debe priorizar al agente de reservas 1 con el 75% de las reservas deben ser creadas por dicho

agente y se debe priorizar la confirmación de reservas, obteniendo al finalizar, un 80% del total de reservas confirmadas.

Se inicializa tanto la fábrica de hilos y un ArrayList de hilos y se agregan hilos en los cuales se le asignan diferentes procesos, los cuales son: uno por cada agente (1 y 2), uno proceso de cancelación, uno para la confirmación y pago, 5 para la generación y entrada de clientes, y uno para el log. Y luego se inicializan todos los hilos.

Clase: Monitor

Esta clase está encargada de la centralización de la concurrencia. Los hilos ingresan al monitor, corroboran si las condiciones para realizar su proceso están dadas y salen del mismo para hacer su trabajo o se duermen e ingresan a una cola de espera para que no se produzca un deadlock. No es una clase viva, los que trabajan, duermen y señalan son los mismos hilos.

Dentro de las variables del monitor encontramos: El número máximo de clientes 186, la instancia del monitor ya que usamos un patrón de diseño singleton para que nuestras variables compartidas sean únicas y se use una sola instancia de esta clase, un semáforo para la exclusión mutua a la hora de ingresar al monitor, un array list de los diferentes locks, la matriz incidencia de la red, el marcado actual, un booleano para saber si termino, cantidad de clientes atendidos por cada agente (superior e inferior), booleanos para saber si los agentes están trabajando o no, la cantidad de clientes confirmados y cancelados, la secuencia de disparo de las transiciones para poder analizar los invariantes con nuestra expresión regular y nuestra política empleada.

Método GetInstance

Este método público y estático nos devuelve la única instancia de nuestro monitor, es necesaria por el patrón de diseño utilizado en este sistema

Método Monitor

Es el constructor de nuestro monitor el cual es un método privado, en el mismo se genera la matriz de incidencia de nuestra red, las llaves a utilizar y el marcado inicial.

Metodo Generar Marcado Inicial

En este método privado se genera un array el cual representa el marcado inicial de nuestra red de petri.

Metodo Generar Matriz de Incidencia

Este método privado nos genera nuestra matriz de incidencia de nuestra red, la cual se calcula como la resta de la matriz de post incidencia y pre incidencia, obtenemos una matriz plaza (filas) por transición (columnas).

Metodo Generar Llaves

Este método privado agrega a nuestro arraylist de llaves, objetos de la clase Locks, cada uno con un nombre específico.

Métodos Getters y Setters

Los métodos getters nos devuelven información sobre el estado de algunas variables vinculadas a nuestra red. El único método setter que tenemos es establecer política.

Método Fire Transition

Método público que recibe como parámetro el número de transición a disparar. El hilo que ejecute este método, primero toma el mutex para poder ingresar al monitor, luego verifica si la transición que quiere disparar está sensibilizada, si lo está se llama al método disparar corrobora si la red termino y devuelve el mutex, si no lo está se llama al método dormir hilo.

Metodo Dormir Hilo

En este método privado entran todos aquellos hilos que quisieron disparar una transición que no estaba sensibilizada. Primero se devuelve el mutex y vuelve a la cola de entrada siguiendo con nuestra política de señalizado (signal and continue), se lo hace dormir al hilo con un wait.

Método Llave Dormir Hilo

Este Método privado recibe por parámetro el número de transición que queremos disparar, y él mismo nos devuelve los locks correspondiente al proceso que estamos ejecutando para poder dormir a ese hilo.

Metodo Disparar

Este método privado recibe por parámetro el número de transición que vamos a disparar. Primero agrega la transición a la secuencia, según sea la transición se incrementa la cantidad de clientes atendidos por el agente 1 o 2, clientes confirmados o cancelados (Transiciones 2, 3 ,6 y 7), se genera un nuevo mercado correspondiente al estado actual de la red y despierta a los hilos correspondientes.

Método Despertar Hilo

En este método privado recibe por parámetro el número de transición que queremos disparar, comprueba si es alguna de las transiciones involucradas en un conflicto estructural llama a la política para ver cómo se prosigue, si la transición es T1, T2 o T3, se levanta a los hilos de la entrada de clientes.

Método Nuevo Marcado

En este método privado recibe por parámetro el número de transición que queremos disparar, se va a generar el nuevo marcado de la red que representa el nuevo estado de la misma, se va a calcular utilizando la ecuación fundamental, se crea un vector de disparo, se hace el producto matricial entre la matriz de incidencia y el vector de disparo, por último se suma el marcado actual al resultado del producto y obtenemos el nuevo marcado.

Método Sensibilizado

En este método privado recibe por parámetro el número de transición que queremos disparar, se llama al nuevo marcado que se generaría con el disparo de esa transición y se verifica que en todo el nuevo marcado no encuentra un número negativo, si sucede ese la transición no esta sensibilizada y se devuelve un false, caso contrario si lo está y se devuelve un true.

Clase: Política

En esta clase se modela la política que se implementa en los conflictos estructurales que posee nuestra red, tiene como variable un número entero que representa la política a utilizar por nuestra red.

Metodo Política

Es el constructor de nuestra clase, recibe por parámetro un número entero que representa la política empleada ya sea 1 para una política balanceada o 2 para una política priorizada

Metodo Llamado a Política

Método público que recibe por parámetro la transición a disparar, luego verifica si la política a utilizar es la 1 o la 2 llamando a los métodos que correspondan.

Métodos Politica Priorizada 1 y 2

Ambos métodos privados se encargan de la política priorizada en donde la primera parte se encarga de darle prioridad al agente superior haciendo que atienda al 75% de las reservas, y la segunda parte se encarga de priorizar la cantidad de reservas confirmadas obteniendo al finalizar un 80% de reservas confirmadas.

Metodos Política Balanceada 1 y 2

Ambos métodos privados se encargan de la política balanceada en donde la primera parte se encarga de balancear la cantidad de gente atendida por cada agente obteniendo al finalizar que cada agente atendió al 50% de los clientes, y la segunda parte se encarga de balancear la cantidad de reservas confirmadas y canceladas.

Interface: Monitor Interface

Es una interface que nos da mayor capacidad de abstracción, posee el método abstracto fireTransition() el cual implementamos en nuestro monitor siendo este el único método público de nuestro monitor teniendo un único punto de acceso,

centralizando nuestros recursos compartidos, la concurrencia y la toma de decisiones viendo cual hilo se dispara dependiendo de su sensibilizado.

Enumerado: Número de Agente

Este enumerado es una variable en la clase Atencion de Agente, la cual nos permite definir el comportamiento de los agentes según se defina como agente uno (superior) o agente dos (Inferior). Usar enumerados tiene muchas ventajas, como la facilidad de mantención cuando crece nuestro software.

Our Thread Factory

Este patrón de diseño sirve para centralizar la creación de hilos en nuestro código y lleva registros y estadísticas de estos

Conclusión

En este trabajo pudimos comprobar las ventajas de modelar con redes de Petri, ya que nos permite tener herramientas matemáticas que nos aseguran que si implementamos bien estas redes, no tendremos problemas por condiciones de carrera. Por otra parte el uso del monitor, nos ayuda a centralizar todos nuestros problemas en una sola clase. Y a través de un lenguaje poder comprobar de manera sencilla si existió algún error inesperado con el uso de expresiones regulares.