Análisis de la herramienta TINA – Timed Petri Nets Analyzer

Victoria Lopez[1], Tomás Ponce[1], and Pedro Salas[1]

Facultad de Matemática, Astronomía, Física y Computación, Universidad Nacional de Córdoba, Ciudad Universitaria, Córdoba, Argentina
{victorialopez23,tomaspgessi,pedrosalaspinero}@mi.unc.edu.ar

Abstract. Este trabajo analiza la herramienta TINA (Time Petri Net Analyzer), desarrollada como una solución especializada para modelar y verificar sistemas concurrentes y de tiempo real mediante redes de Petri temporizadas. A lo largo del documento se describe su origen, objetivos, funcionamiento, características técnicas y aplicaciones concretas. Se incluye un caso de estudio representativo, en el que se modela un sistema de procesamiento de video aéreo.

1 Introducción

En este informe se presenta un análisis de TINA (Time Petri Net Analyzer), una herramienta desarrollada para el modelado, simulación y verificación formal de sistemas concurrentes con restricciones temporales. TINA se basa en el uso de redes de Petri temporizadas (Time Petri Nets, TPN), un modelo utilizado en ingeniería de sistemas para representar procesos que involucran concurrencia, sincronización y temporización. La motivación de este trabajo es explorar en profundidad las capacidades de TINA, no sólo desde el punto de vista técnico, sino también en su aplicación práctica. A través de un caso de estudio representativo, se busca ilustrar cómo esta herramienta puede ser utilizada para garantizar el correcto funcionamiento de un sistema con restricciones temporales, verificando propiedades críticas en contextos reales de ingeniería de software. La contribución principal de este informe consiste en sistematizar la información técnica y de uso sobre TINA, evaluando sus ventajas y limitaciones, y ejemplificando su funcionamiento mediante una aplicación concreta.

2 Marco Teórico

Para hablar sobre TINA (TIme petri Net Analizer), es fundamental dar un contexto teórico. En particular, en esta sección vamos a hablar de las redes de Petri clasicas (PN), que no tienen una noción de tiempo. Luego introducimos las redes de Petri con tiempo (TPN), que son un ejemplo de las redes de Petri con dependencia temporal, y finalmente mencionaremos algunos aspectos de las redes de Petri con tiempo extendidas (eTPN), que nos van a ser útiles para entender el caso de uso presentado.

2.1 Redes de Petri clásicas

Las **redes de Petri (PN)** son una herramienta teórica muy fuerte para el modelado, simulación y análisis de sistemas concurrentes. En el modelado de sistemas con PN clásicas sin indicación del tiempo, los eventos son representados como transiciones, que se denotan con rectángulos o barras. Tanto las precondiciones como las postcondiciones de los eventos son representadas como lugares, denotados con círculos. Se dibujan aristas dirigidas entre un lugar y una transición para representar la relación de precondición. Análogamente, se dibujan aristas dirigidas entre una transición y un lugar para representar la postcondición.

Para representar una situación en el modelado, se marcan a los lugares con cierta cantidad de puntos o tokens. Luego, una transición esta habilitada en una PN clásica si sus precondiciones (i.e. lugares que están conectados con una flecha hacia la transición) están satisfechas, es decir, tienen tokens. En particular, cuando una flecha tiene indicado un peso en el diagrama, la precondición debe tener esa cantidad de tokens en el lugar asignado. Las transiciones habilitadas pueden dispararse. Una forma imprecisa pero intuitiva de pensar los disparos de una transición, es que esta "consume" tokens de sus precondiciones y "produce" tokens en sus postcondiciones. De esta forma, podemos hablar de una secuencia de marcados o estados, donde para pasar de un estado a otro, se dispara alguna transición de la red [1].

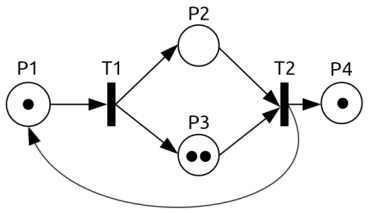

Fig. 1. Ejemplo de red de Petri clásica

En esta figura podemos ver un ejemplo de PN, en el cual tenemos cuatro lugares: $\{p1, p2, p3, p4\}$; y dos transiciones, $\{t1, t2\}$. La transición $t1$ está habilitada, mientras que $t2$ no lo está. Si consideramos el estado actual como $(1, 0, 2, 1)$, podemos ver que el estado resultante de que se dispare $t1$ es $(0, 1, 3, 1)$. En este otro estado, deja de estar habilitada la transición $t1$ pero pasa a habilitarse $t2$. Esta simulación puede seguir infinitamente hasta que no haya transiciones habilitadas.

2.2 Red de Petri con tiempo

Como mencionamos anteriormente, estos modelos matemáticos no tienen una noción de tiempo asociada. A partir de esta limitación, se introdujeron muchas

variantes de PN que dependían en alguna forma del tiempo. Cada alternativa tiene sus ventajas y desventajas frente al modelado de un sistema particular.

La variante que nos interesa a nosotros es la **red de Petri con tiempo (TPN)**. Esta variante introduce la idea de un intervalo de tiempo $[a_t, b_t]$ a cada transición t, y modifica la regla de activación para que las transiciones no se puedan activar apenas están disponibles (i.e. sus precondiciones se cumplen). En cambio, tienen que pasar al menos a unidades de tiempo desde que la transición t se habilitó para que se dispare, y no puede dispararse después de b unidades de tiempo después de habilitarse, excepto si la transición deja de estar habilitada antes. En estas redes, el inicio y el final del intervalo son relativos al momento en el que la transición se habilita. Además, la activación de la transición es instantánea [1].

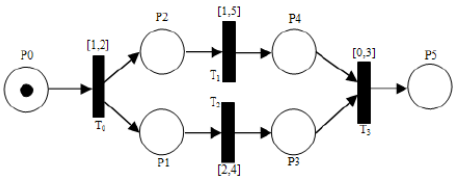

Fig. 2. Ejemplo de TPN

En el modelado con TPN, una situación o estado se representa con un marcado de los lugares y una variable externa *tiempo*. A diferencia de una PN, hay dos mecanismos que pueden modificar el estado, o bien se dispara una transición, o avanza el tiempo.

2.3 Variaciones de redes de Petri

Además de incorporar la dimensión temporal, existen otras extensiones a las redes de Petri que permiten modelar con mayor expresividad ciertas restricciones lógicas o de control que aparecen en sistemas reales. En particular, algunas herramientas como TINA permiten modelar arcos especiales entre lugares y transiciones que no siguen la semántica tradicional de consumo y producción de tokens. Los dos tipos más comunes de estos arcos son los arcos inhibidores y los arcos lectores. Estas extensiones forman parte de lo que se conoce como redes de Petri con tiempo extendidas (eTPN).

Un arco inhibidor se representa gráficamente con una línea dirigida desde un lugar hacia una transición, terminando en un pequeño círculo blanco. A diferencia del arco normal, con este tipo de arco la transición solo estará habilitada si el lugar conectado tiene una cantidad estrictamente menor que el umbral especificado. Esto permite modelar condiciones de exclusión o restricciones negativas, como por ejemplo la necesidad de que un recurso no esté en uso para que una operación pueda ejecutarse.

Por otro lado, los arcos lectores se representan con una línea que termina en un círculo negro y conectan también un lugar con una transición. A diferencia de un arco tradicional, el disparo de la transición no consume los tokens del lugar, es decir, simplemente los observa o "lee". Esta propiedad resulta útil para modelar chequeos de disponibilidad o condiciones de entorno que deben mantenerse durante la ejecución del sistema.

Este tipo de extensiones no forman parte de la definición estándar de las PN o TPN, pero son ampliamente utilizadas en herramientas de análisis y verificación formal debido a su utilidad práctica. En el contexto del caso de uso que presentaremos más adelante, estas variaciones permiten modelar de forma más precisa ciertas restricciones operativas sin recurrir a construcciones artificiales o complejas [2].

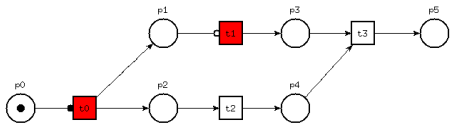

Fig. 3. Ejemplo de una ePN en TINA

En la figura podemos ver un modelado de una ePN en TINA. Las transiciones rojas indican que están habilitadas. Notar que la transición $t1$ se encuentra habilitada y $p1$ no tiene tokens. Si se hiciera la activación de la transición $t0$, el lugar $p0$ conserva su token, porque los une un arco de lectura

3 TINA

3.1 Historia de la herramienta

TINA es una herramienta desarrollada en el ámbito académico por el laboratorio LAAS-CNRS en Toulouse, Francia, bajo la dirección de Bernard Berthomieu y colaboradores. Surge de líneas de investigación iniciadas desde los años 1980 en métodos de análisis de redes de Petri temporizadas.

Originalmente concebida en el entorno académico, TINA se consolidó a inicios de los 2000 como un conjunto de herramientas para editar y analizar redes de Petri clásicas y temporizadas. Se distribuye de forma gratuita para múltiples plataformas aunque no es de código abierto, manteniéndose su desarrollo principalmente en el equipo original.

A lo largo de dos décadas, TINA ha pasado de ser un prototipo de investigación a una herramienta madura utilizada en la academia e incluso en competencias internacionales de verificación. El desarrollo continúa activo, su uso se concentra en el ámbito académico y de I+D, donde investigadores, estudiantes de posgrado y desarrolladores de herramientas la emplean para modelar y verificar sistemas de tiempo real. Si bien no es una herramienta industrial de uso

masivo, TINA ha sido aplicada en proyectos bajo contrato y casos de estudio en colaboración con la industria, gracias a su acceso gratuito y reputación de confianza a lo largo de los años.

3.2 Objetivo de TINA

Una de las funciones fundamentales de TINA es la generación del espacio de estados alcanzables. A partir del estado inicial de un sistema, es posible calcular todos los estados que puede alcanzar, teniendo en cuenta la disposición de tokens como los tiempos asociados a las transiciones. Esto nos ayuda a detectar condiciones de carrera, verificar alcanzabilidad de estados y construir modelos de comportamiento como sistemas de transición temporizados (TTS).

Para realizar esto utilizamos distintas herramientas, todas partes del suit de herramientas de TINA, entre ellas usamos **nd** para generar el grafo de estados alcanzables, **sift** para verificación de propiedades temporales sobre los grafos, y finalmente **tedd** para que nos permita una construcción simbólica eficiente del espacio de estados. Una vez obtenido el espacio de estados, es posible hacer una verificación de propiedades temporales y de seguridad, lo cual nos permite garantizar el correcto funcionamiento de los sistemas modelados. Podemos verificar automáticamente las propiedades críticas de dichos sistemas. Entre estas propiedades se encuentran, por ejemplo, las de seguridad, como asegurar que nunca se alcance un estado en el que dos procesos accedan simultáneamente al mismo recurso. También se pueden verificar propiedades temporales, tales como garantizar que, si ocurre un determinado evento, otro suceda en menos unidades de tiempo. Para hacer estas verificaciones TINA ofrece herramientas como **selt** para verificar las fórmulas en lógica temporal lineal (LTL), **muse** para verificar con mu-cálculo modal y **scan** para verificar propiedades de alcanzabilidad expresadas como fórmulas lógicas.

Además de estos análisis formales, TINA ofrece la posibilidad de observar el comportamiento del sistema a través de simulaciones, lo cual resulta muy útil para depurar y entender mejor el modelo. Con la herramienta **play**, que simula paso a paso el sistema de ejecución del sistema, con esta herramienta también podemos reconstruir trazas de ejecución o reproducir errores, y **walk,** que realiza exploraciones aleatorias útiles para observar comportamientos probables y depurar.

Cuando tenemos modelos grandes o complejos podemos simplificarlos y hacerlos más efectivos utilizando las herramientas de **reduce**, que aplica técnicas de reducción de redes y preserva propiedades claves, y **struct**, que realiza un análisis estructural de la red.

Finalmente, otro aspecto importante es la conversión y compatibilidad entre formatos. TINA permite transformar modelos de un formato a otro, lo cual facilita su integración con herramientas externas y distintos entornos de análisis. Para eso se utilizan varios conversores. Por ejemplo, **ndrio** se encarga de convertir modelos entre diferentes representaciones internas, lo que permite trabajar con ellos desde distintas perspectivas. **Ktzio** sirve para exportar e importar modelos como sistemas de transición etiquetados, lo cual es útil si se quiere analizar

el sistema con herramientas externas. Por ultimo, **frac** permite compilar modelos escritos en Fiacre, un lenguaje de alto nivel pensado para describir sistemas complejos de manera más clara y estructurada.

3.3 TINA desde el lado del usuario

TINA ofrece una interfaz gráfica, que permite construir redes de Petri de forma visual. Desde ahí, se pueden definir plazas y transiciones (ya sean inmediatas o temporizadas) y conectarlas usando distintos tipos de arcos, como los normales, inhibidores o lectores. Este editor facilita muchísimo el modelado, porque permite ver y probar cómo se comporta el sistema paso a paso, viendo cómo se mueven los tokens. Esta simulación visual es muy útil para detectar errores o entender mejor el comportamiento del modelo sin tener que escribir todo en texto.

Además de la parte gráfica, TINA incluye un conjunto de herramientas por línea de comandos que amplían sus capacidades, las cuales mencionamos anteriormente. El modelado puede hacerse tanto de forma gráfica como textual. Cuando se usa el editor gráfico, los modelos se guardan en archivos con extensión .ndr, mientras que los formatos .net o .tpn se usan para escribir redes de forma textual, dependiendo si son clásicas o temporizadas.

Algo muy interesante de TINA es que no hace falta programar ni instrumentar código fuente para usarlo. Su enfoque está en la modelación formal, permitiendo que te concentres en cómo se comporta el sistema sin preocuparte por cómo se implementa. Si necesitás un nivel más alto de descripción, podés usar herramientas como Fiacre para definir componentes y tareas, y luego generar modelos compatibles con TINA automáticamente.

4 Caso de uso: Análisis de latencia en un sistema de seguimiento de video aéreo

Para ilustrar el uso de TINA , consideramos un caso de estudio de un sistema de seguimiento de video aéreo en tiempo real, este problema fue propuesto en el desafío académico *WATERS15 Workshop* [4]. Se trata de un sistema de seguimiento de video compuesto por varias tareas periódicas que procesan imágenes de un dron para identificar objetivos. El objetivo del desafió era determinar la latencia extremo a extremo (tiempo desde que se captura una imagen hasta que se muestra en pantalla) y verificar si bajo ciertos parámetros del sistema podría perderse alguna imagen (frame) por atraso del sistema.

4.1 Descripción general del sistema

El sistema tiene cuatro procesos principales (T1, T2, T3, T4) [5], cada uno encargado de realizar ciertas funciones y con ciertos parámetros que modifican su comportamiento:

– **T1** (Pre-procesamiento): Recibe el frame de la cámara de video con periodo fijo $40ms \pm 0,01\%$ y aplica un preprocesamiento de la imagen que tarda exactamente $28ms$.
– **T2** (Procesado principal): Recibe el frame pre-procesado de T2 y aplica el procesado principal que demora entre $17ms$ y $19ms$. El resultado lo envía a un registro.
– **T3** (Filtrado): Filtra los frames procesados de T2 periódicamente con periodo fijo $\frac{40}{3}ms \pm 0,05\%$. La demora de ejecución es de $8ms$ y guarda los resultados en un buffer.
– **T4** (Convertidor digital/analógico): Recibe los frames del buffer de T3 y los muestra en pantalla, la tarea tiene un periodo fijo $40ms \pm 0,01\%$ y demora $1ms$ en caso de no haber un frame y $10ms$ en caso de que si

Para analizar este sistema construyó un modelo de red de petri temporizada que representa el flujo de frames y los temporizadores sobre ciertas *unidades de tiempo u.t.*:

– Transiciones representan eventos como *"frame generado por T1"*, *"frame procesado por T3"*, *"frame enviado por T4"*. Cada transición tiene asociado un intervalo correspondientes a los periodos de ejecución de sus tareas.
– Ciertos lugares en la red representan estados como *"frame en buffer listo para procesar"*, *"frame disponible para mostrar"*, etc.
– Ademas del modelo del sistema, se coloco una transición especial *DEADLINE* que modela el tiempo limite de procesamiento de una trama, esta transición se habilita cuando un frame es generado y se dispara después de M *u.t.* si para entonces el frame no ha sido mostrado. Básicamente M *u.t.* representa la máxima latencia aceptada, de tal modo que si el tiempo de procesamiento es mayor a M *u.t.* entonces el sistema demoro demasiado.

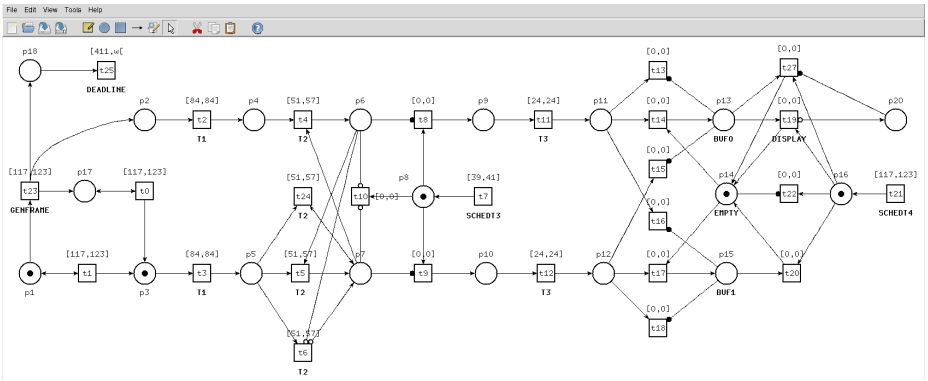

Fig. 4. Modelado del sistema en la herramienta TINA

4.2 Verificación de propiedades y datos de interés

Ningún frame se pierde por exceso de latencia: El principal requerimiento a verificar era que ningún frame se pierda por exceso de latencia, es decir siempre vale *"un frame es mostrado sin que haya expirado su deadline"*. En el modelo planteado esta condición se da siempre que *cada vez que se dispara DISPLAY, entonces no se disparo DEADLINE.*

1. **Verificación de la propiedad en un caso base:** Utilizando un deadline de 412 $u.t$ o mayor y los parámetros iniciales dados, TINA fue capaz de verificar que la propiedad anterior se cumplía siempre, es decir, para ese tiempo limite nunca se perdía un frame
2. **Verificación de la propiedad con un deadline mas ajustado:** Bajo los mismos parámetros anteriores pero con un deadline de 411 $u.t$ o menor se pudo verificar que existía una secuencia de transiciones que invalidaba la propiedad, de tal modo que era posible perder frames.

Latencia mínima y máxima: Un dato de interés en este sistema es determinar la latencia mínima y máxima del sistema, es decir, cual era el menor tiempo posible para el cual se recibía un frame procesado y cual era respectivamente el mayor tiempo posible. Si bien estos datos no son posibles de obtener directamente con TINA son posibles de buscar analizando las secuencias de ejecución, determinado así la latencia mínima y máxima.

Como es posible observar a través de los resultados anteriores, el caso de estudio demostró como TINA puede usarse para verificar invariantes del sistema y también para extraer información de ejecución del sistema, probando así el potencial de TINA para chequeo de modelos de sistemas temporales y complejos.

5 Conclusiones

La herramienta TINA ofrece una solución potente para el modelado y análisis de redes de Petri con tiempo. Su instalación es relativamente sencilla en entornos Unix y en cuanto a su comprensión, TINA exige una base teórica sólida, ya que la lógica detrás de los analizadores y los modelos utilizados no es trivial, lo cual dificulta su adopción por parte de usuarios principiantes. Pese a esto, la herramienta probo su utilidad en el modelado y verificado de sistemas complejos y temporales, pudiendo así verificar invariantes y propiedades deseables del sistema, algo fundamental en sistemas críticos.

Bajo el análisis del caso de uso dado, es posible chequear sin mucha complejidad sistemas a gran escala, partiendo de los componentes principales y sus interacciones se deducen propiedades del sistema. En conjunto, la suit de herramientas TINA provee una serie de utilidades orientadas al modelado formal de sistemas, que sin tanto esfuerzo permite obtener resultados relevantes.

References

1. Popova-Zeugmann L. (2013) *Time and Petri Nets*. Springer
2. Berthomieu, B., & Diaz, M. (1991). *Modeling and Verification of Time Dependent Systems Using Time Petri Nets*. IEEE Transactions on Software Engineering.
3. B. Berthomieu and F. Vernadat LAAS / CNRS - Time Petri Nets Analysis with TINA.
4. WATERS15 Workshop
5. WATERS15 Challenge Specs
6. Fiace Use Case An Aerial Video Tracking System
7. TINA