UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

Modelado y análisis de un sistema de semáforos inteligentes con UPPAAL

Modelling and analysis for an intelligent traffic lights system with UPPAAL

Realizado por
Pablo Villarejo Pérez

Tutorizado por
Laura Panizo Jaime
María del Mar Gallardo Melgarejo

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, septiembre de 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

Modelado y análisis de un sistema de semáforos inteligentes con UPPAAL

Modelling and analysis for an intelligent traffic lights system with UPPAAL

Realizado por
Pablo Villarejo Pérez

Tutorizado por
Laura Panizo Jaime
María del Mar Gallardo Melgarejo

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2025

Fecha defensa: septiembre de 2025

Agradecimientos

A mis padres, por nunca dejar de confiar en mí.

A Luz María, por ser mi compañera de vida y la razón de que todo esto tenga sentido.

A mis amigos del grado, por convertir los buenos momentos en extraordinarios, y los malos en llevaderos.

Al resto de mi familia y amigos, por multiplicar mis alegrías y dividir mis penas.

A Laura y María del Mar, por su tiempo y por su paciencia.

Gracias a todos.

Resumen

En este Trabajo de Fin de Grado se aborda el análisis y verificación formal de un sistema de semáforos inteligentes aplicado a un cruce complejo en el campus universitario de Teatinos (Universidad de Málaga), donde confluyen vehículos, metro en superficie y ambulancias debido a la cercanía del hospital. El objetivo principal es garantizar la seguridad en el cruce minimizando el riesgo de colisiones, al mismo tiempo que se asegura un flujo vehicular eficiente. Para ello se emplea UPPAAL, una herramienta de model checking que permite modelar, simular y verificar sistemas de tiempo real mediante autómatas temporizados y autómatas temporizados estocásticos.

El trabajo se desarrolla de forma iterativa, comenzando con un modelo básico que incluye coches, metro y semáforos, y ampliándose posteriormente con un segundo cruce regulado y, finalmente, incluyendo también ambulancias. A lo largo de estas iteraciones se definen requisitos, se modelan las restricciones y se verifican propiedades de seguridad, viveza y rendimiento, tanto con verificación exhaustiva como con verificación estocástica. Los resultados muestran que los métodos formales permiten no solo detectar errores tempranos en el diseño, sino también garantizar de manera rigurosa la corrección y robustez de un sistema crítico como el control de tráfico urbano.

Palabras clave: UPPAAL, modelado y verificación, autómatas temporizados, semáforo inteligente.

Abstract

This Final Degree Project addresses the analysis and formal verification of an intelligent traffic light system applied to a complex intersection in the Teatinos campus of the University of Málaga, where vehicles, surface metro lines, and ambulances converge due to the proximity of the university hospital. The main objective is to ensure safety at the intersection by minimizing the risk of collisions while maintaining an efficient traffic flow. To achieve this, UPPAAL is employed as the model checking tool, allowing the modeling, simulation, and verification of real-time systems through timed automata and stochastic timed automata.

The project follows an iterative approach, starting with a basic model including cars, metro, and traffic lights, later extending it with a second regulated intersection and, finally, also including ambulances. Across these iterations, requirements are defined, restrictions are modeled, and properties related to safety, liveness, and performance are verified, using both exhaustive and stochastic verification techniques. The results demonstrate that formal methods not only enable the early detection of design flaws but also rigorously guarantee the correctness and robustness of a critical system such as urban traffic control.

Keywords: UPPAAL, modeling and verification, timed automata, intelligent traffic light.

Índice

1 Introducción

1.1. Motivación

La gestión del tráfico en entornos urbanos es un problema complejo en el que influyen múltiples factores, como la densidad de vehículos, la coexistencia de distintos medios de transporte y la necesidad de dar prioridad a servicios de emergencia. En el campus de Teatinos, la intersección entre el Boulevard Louis Pasteur y la calle Jiménez Fraud, que podemos observar en la Figura 1, constituye un punto crítico debido a su alto volumen de tráfico y a la presencia de coches, metro en superficie y ambulancias que se dirigen al hospital Universitario Virgen de la Victoria. Esta situación hace necesario el desarrollo de sistemas que reduzcan al mínimo el riesgo de accidentes y, al mismo tiempo, mantengan la fluidez del tráfico.

El uso de semáforos inteligentes se presenta como una solución prometedora para afrontar este reto, al permitir un control dinámico y adaptativo en lugar de depender de ciclos fijos. Sin embargo, al tratarse de un sistema crítico que afecta directamente a la seguridad de peatones y conductores, resulta imprescindible garantizar su correcto funcionamiento antes de una posible implementación. Por ello, se plantea la aplicación de métodos formales y, en particular, el uso de la herramienta UPPAAL [1], con el fin de modelar, simular y verificar que el sistema propuesto cumple con los requisitos de seguridad y viveza.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado es aplicar métodos formales al diseño y análisis de sistemas de semáforos inteligentes, empleando las funcionalidades de modelado, simulación y verificación que ofrece la herramienta UPPAAL. Para ello se utilizará como caso de estudio real el cruce del Boulevard Louis Pasteur con Jiménez Fraud en el campus de Teatinos.

Figura 1: Boulevard Louis Pasteur y calle Jiménez Fraud en Google Maps.

De este objetivo principal se derivan los siguientes objetivos específicos:

- Estudiar los sistemas de semáforos inteligentes y las metodologías de modelado y análisis existentes basadas en métodos formales.

- Analizar el cruce del Boulevard Louis Pasteur con Jiménez Fraud para identificar sus principales características y requisitos.

- Desarrollar un modelo formal genérico de un cruce gestionado por semáforos inteligentes en UPPAAL y adaptarlo posteriormente al caso de estudio.

- Especificar los requisitos de seguridad y viveza y formalizarlos mediante expresiones lógicas en TCTL.

- Verificar formalmente que el modelo particularizado para el caso de estudio cumple con los requisitos definidos.

1.3. Estructura del documento

El Trabajo de Fin de Grado se organiza en siete capítulos que guían al lector de manera progresiva desde los fundamentos teóricos hasta el análisis de resultados y las conclusiones. El capítulo 1 actúa como introducción general, donde se exponen la motivación y los objetivos del trabajo, las tecnologías utilizadas, la contribución realizada y, finalmente, la metodología. Además, se presenta la organización del documento para facilitar la lectura.

El capítulo 2 recoge los preliminares y trabajos relacionados, que constituyen el marco teórico de referencia. En él se revisan los sistemas de semáforos inteligentes y el papel de los métodos formales en sistemas críticos. A continuación, se introducen los conceptos fundamentales de los autómatas temporizados y la herramienta UPPAAL, dedicando también un apartado a los trabajos relacionados más relevantes. Posteriormente, el capítulo 3 se centra en el caso de estudio, detallando las características del cruce analizado, los requisitos identificados y la justificación de su elección.

En los capítulos 4, 5 y 6 se desarrolla el núcleo del trabajo, centrado en la construcción de los modelos formales del cruce. Cada capítulo aborda un modelo distinto, que incrementa progresivamente la complejidad del sistema. Empezando desde el cruce superior inicial, hasta la ampliación con un cruce adicional y, finalmente, la incorporación de ambulancias como vehículos prioritarios. En cada caso se presentan las consideraciones y problemas de diseño, los autómatas principales que conforman el modelo, así como los resultados obtenidos en simulaciones y en la verificación de propiedades.

Para finalizar, el capítulo 7 recoge las conclusiones generales y plantea posibles líneas futuras de investigación y mejora, completando así el recorrido del trabajo.

1.4. Tecnologías usadas

Para el desarrollo de este Trabajo de Fin de Grado se han empleado distintas tecnologías y herramientas que han facilitado tanto la construcción del modelo como la organización y redacción de la memoria:

- **UPPAAL**: herramienta principal de modelado, simulación y verificación formal mediante autómatas temporizados y estocásticos [1].

- **Git**: utilizado para el control de versiones del proyecto de manera local [2].
- **GitHub**: plataforma empleada para el almacenamiento remoto del proyecto y la gestión del repositorio, facilitando la trazabilidad y el acceso desde cualquier dispositivo [3].
- **Visual Studio Code**: editor empleado para organizar los distintos documentos en formato Markdown y gestionar los archivos auxiliares del proyecto [4].
- **LaTeX**: lenguaje de tipografía científica usado para redactar la memoria con un formato profesional y un control preciso de referencias, índices y fórmulas [5].
- **Overleaf**: entorno colaborativo en la nube para compilar y gestionar documentos en LaTeX de forma accesible y sin necesidad de instalaciones adicionales [6].
- **Canva**: herramienta usada para la creación de esquemas y diagramas ilustrativos de los cruces y de la organización del sistema modelado [7].

El uso combinado de estas herramientas ha permitido estructurar el trabajo de forma iterativa, manteniendo el control sobre los avances en el modelo y facilitando la comunicación de los resultados mediante documentación clara y bien organizada.

1.5. Contribución del trabajo

La aportación principal de este Trabajo de Fin de Grado consiste en el modelado formal y el análisis de un sistema de semáforos inteligentes aplicado a un cruce real con condiciones de especial complejidad, debido a la interacción entre vehículos, ambulancias y el metro en superficie. A partir de este caso de estudio se han desarrollado, en la herramienta UPPAAL, distintos modelos que evolucionan de forma incremental, incorporando progresivamente nuevas funcionalidades y restricciones. Este proceso ha permitido mostrar cómo un sistema crítico puede construirse, validarse y refinarse a través de sucesivas iteraciones.

La contribución no se limita a la construcción de los modelos, sino que incluye también el análisis de su corrección mediante simulaciones y verificación formal de propiedades de seguridad, viveza y rendimiento. De esta manera, se demuestra la capacidad de los métodos formales para detectar situaciones de riesgo y asegurar el cumplimiento de requisitos fundamentales antes de una posible implementación real.

Además, el trabajo ofrece una aplicación práctica de los autómatas temporizados a un problema de tráfico urbano, lo que refuerza la utilidad de este enfoque en el ámbito de la movilidad inteligente. La documentación detallada de cada iteración y de los resultados obtenidos pretende servir de referencia para futuros desarrollos que aborden sistemas similares.

1.6. Metodología

1.6.1. Enfoque metodológico

Para el desarrollo del Trabajo de Fin de Grado se adoptó una metodología en cascada con un enfoque iterativo. La elección de este método se debió a la necesidad de trabajar de manera estructurada, pero al mismo tiempo con la flexibilidad suficiente para ir incorporando progresivamente nuevas funcionalidades al modelo. El esquema en cascada proporcionó un orden lógico y una división clara de fases, mientras que las iteraciones sucesivas permitieron evolucionar el modelo de un diseño sencillo hacia otro más realista y complejo.

El carácter iterativo se ve presente principalmente en las fases de modelado, simulación y verificación. Este, resultó fundamental para el éxito del trabajo, ya que permitió verificar y validar cada incremento de manera independiente, reduciendo el riesgo de acumular errores en etapas avanzadas. De este modo, se pudo garantizar que cada nueva ampliación del modelo estuviera correctamente fundamentada y validada antes de continuar con fases posteriores.

Este enfoque también facilitó la trazabilidad del desarrollo, ya que cada iteración del modelo quedó reflejada como una versión independiente, permitiendo comparar entre diferentes modelos, identificar las mejoras introducidas y comprender de manera clara cómo fue creciendo el sistema. La metodología, por tanto, no solo guió el proceso de trabajo, sino que también aportó rigor, claridad y eficiencia en la gestión del proyecto.

Cabe destacar que esta combinación entre la estructura del modelo en cascada y la flexibilidad del modelo iterativo coincide con propuestas metodológicas más recientes en la literatura, como el *waterative model* [8]. Dicho enfoque demuestra que integrar ambos paradigmas permite aprovechar las ventajas de cada uno. Estas son la organización secuencial del método cascada y la capacidad de adaptación y validación continua propia de la iteración, lo cual redunda en una mayor calidad del producto final y en un incremento en la satisfacción del cliente.

1.6.2. Fases de trabajo

El desarrollo del Trabajo de Fin de Grado se organizó en una serie de fases sucesivas, cada una con un papel específico en la construcción del modelo final:

- **Estudio de las tecnologías**:

  Esta fase fue imprescindible dado que no se disponía de experiencia previa con UPPAAL, la herramienta central del proyecto. Antes de abordar el caso de estudio real, se realizaron ejercicios básicos y tutoriales sobre autómatas temporizados [9]. Estos ejercicios iniciales, aunque no estaban relacionados con el tráfico ni con los semáforos, resultaron fundamentales para familiarizarse con la sintaxis de la herramienta, la creación de plantillas, la declaración de variables y canales, y el uso de relojes. Esta etapa de aprendizaje proporcionó la base técnica necesaria para afrontar posteriormente el modelado del cruce real.

- **Estudio del caso**:

  Tras adquirir la base tecnológica, se analizó en detalle el cruce del Boulevard Louis Pasteur con Jiménez Fraud en el campus de Teatinos. En esta fase se identificaron los actores principales (coches, metro, ambulancias...), sus interacciones y los riesgos asociados al cruce. El estudio permitió extraer los requisitos iniciales de seguridad y viveza que debía cumplir el sistema de semáforos inteligentes.

- **Diseño y modelado**:

  A partir de los requisitos identificados, se inició el desarrollo del modelo formal en UPPAAL. De forma iterativa, se comenzó con un modelo reducido, compuesto únicamente por coches, un metro y semáforos básicos, que servía para comprobar la viabilidad de la propuesta. Posteriormente, el modelo fue creciendo mediante iteraciones. Cada incremento se realizó sobre la base validada de alguna de las iteraciones anteriores, asegurando que el sistema mantenía las propiedades fundamentales de seguridad y corrección.

- **Pruebas y validación**:

  Una parte central del trabajo fue la verificación formal de las propiedades definidas, también de forma iterativa. En cada iteración se especificaron requisitos concretos, como la

ausencia de colisiones, la ausencia de bloqueos, la eventualidad de que todos los actores cruzaran y el cumplimiento de restricciones temporales. Para ello se aplicaron métodos de verificación formales, que se explican con detalle en el Capítulo 2. Esta fase no solo sirvió para confirmar que el modelo cumplía con los requisitos, sino también para identificar problemas de diseño y realizar ajustes que mejoraron la robustez del sistema.

- **Redacción de la memoria**:

  Finalmente, todo el trabajo desarrollado se documentó en la memoria. En este proceso se integraron los fundamentos teóricos, la metodología, las iteraciones del modelo y los resultados de la verificación. La redacción no fue una fase aislada, sino que se realizó en paralelo a lo largo del desarrollo, de manera que cada avance quedara registrado y bien contextualizado.

En conjunto, este proceso metodológico permitió abordar el proyecto de manera progresiva, reduciendo riesgos, asegurando la validez de cada avance y garantizando un resultado final sólido y fundamentado.

2

Preliminares y trabajos relacionados

2.1. Sistemas de semáforos inteligentes

Los semáforos inteligentes constituyen una evolución de los sistemas tradicionales de control del tráfico. A diferencia de los ciclos fijos, estos sistemas incorporan sensores, algoritmos y tecnologías de comunicación que permiten adaptar los tiempos de señal en función de las condiciones reales de circulación. Su objetivo principal es mejorar la fluidez del tráfico, reducir los tiempos de espera y aumentar la seguridad en intersecciones críticas [10].

Además de la regulación de vehículos, estos sistemas suelen integrar la gestión de peatones, bicicletas y transporte público, priorizando en ocasiones a determinados actores según las necesidades del entorno. Por ejemplo, se pueden dar fases preferentes a autobuses o tranvías para fomentar el transporte colectivo, o a ambulancias en situaciones de emergencia. Esta capacidad de adaptación dinámica convierte a los semáforos inteligentes en una herramienta clave para el desarrollo de ciudades más seguras, sostenibles y eficientes dentro del marco de las denominadas *ciudades inteligentes*.

2.2. Métodos formales en sistemas críticos

Los métodos formales son técnicas matemáticas de modelado y verificación utilizadas en el desarrollo de sistemas en los que la seguridad y la fiabilidad son fundamentales. Se aplican ampliamente en sectores como la aviación [11], la automoción [12] o el transporte ferroviario [13], donde los errores pueden tener consecuencias graves. En el contexto del tráfico urbano, como

es el caso de este trabajo, los métodos formales permiten garantizar que un sistema de semáforos inteligentes cumple con requisitos de seguridad y viveza antes de ser implementado en la realidad.

Entre las distintas técnicas existentes que forman parte de los métodos formales, una de las más empleadas es el *model checking*, que consiste en la exploración exhaustiva de un modelo para comprobar el cumplimiento de propiedades especificadas. Una de sus principales ventajas es que permite detectar errores de diseño en fases tempranas del desarrollo, antes de la implementación física del sistema.

Gracias al *model checking*, a través de modelos abstractos y verificaciones exhaustivas, es posible demostrar propiedades críticas como la ausencia de bloqueos, la correcta sincronización entre señales o, por ejemplo, la imposibilidad de que dos actores ocupen un cruce simultáneamente. De esta manera, los métodos formales se presentan como un complemento esencial a las pruebas tradicionales, ofreciendo un alto grado de confianza en la corrección del sistema final.

2.3. Autómatas temporizados y UPPAAL

Los autómatas finitos clásicos [14] constituyen un formalismo matemático que permite representar sistemas discretos mediante un conjunto de estados y transiciones. Su utilidad radica en la capacidad de describir de manera clara el comportamiento secuencial de un sistema, indicando en qué estado se encuentra en cada momento y bajo qué condiciones se producen los cambios. Sin embargo, este enfoque resulta insuficiente para modelar sistemas de tiempo real, en los que no basta con conocer el orden de los eventos, sino también el instante en el que ocurren.

Para superar esta limitación surgen los autómatas temporizados [15], una extensión de los autómatas finitos que incorpora de manera explícita la dimensión temporal. En este formalismo, el paso del tiempo se modela mediante relojes, variables que avanzan de forma continua y uniforme. Estos relojes permiten imponer restricciones tanto en los estados como en las transiciones, de modo que el modelo refleje con mayor precisión los requisitos temporales de un sistema crítico.

En este contexto aparece UPPAAL, que es una herramienta de modelado, simulación y verificación formal que permite representar sistemas exclusivamente como redes de autómatas

temporizados [1]. En el resto de la sección introduciremos las principales características de los autómatas temporizados, utilizando un ejemplo de un sistema ferroviario, que se proporciona en uno de los tutoriales de la herramienta [9]. De forma resumida, el sistema consiste en varios trenes que se aproximan a un puente con una única vía. Para evitar colisiones, un controlador central regula el paso de los trenes, deteniéndolos cuando es necesario y permitiendo su avance únicamente cuando el cruce está libre.

El entorno gráfico de UPPAAL permite traducir los autómatas temporizados a diagramas visuales, donde estados, transiciones, relojes y condiciones se representan de manera estructurada. De esta forma, ofrece una forma clara y consistente de trabajar sobre el formalismo, posibilitando la construcción, simulación y análisis de sistemas de tiempo real de forma precisa.

Cada componente del sistema se define en forma de plantilla, las cuales se pueden instanciar, y que corresponden a un autómata temporizado. Por lo tanto, el conjunto de las instancias de dichos autómatas constituye el sistema.

En el editor gráfico, los estados se representan mediante círculos, y las transiciones, mediante flechas que los conectan. Como puede observarse en la Figura 2, en la parte superior aparece el nombre de la plantilla del autómata y sus parámetros, los cuales se definen cuando se crea una instancia. Se observan además los elementos adicionales que acompañan a los estados y transiciones, que permiten expresar restricciones y condiciones temporales:

- Los **invariantes**, en morado, aparecen asociados a los estados, limitando el tiempo que el sistema puede permanecer en ellos. Por ejemplo, en el estado *Appr*, se encuentra una invariante ”x <= 20”, que fuerza al autómata a no permanecer más de 20 unidades de tiempo en dicho estado.

- En las transiciones pueden encontrarse **guardas**, en verde, que son condiciones que deben cumplirse para que la transición sea posible. También en el estado *Appr*, para realizar la transición hacia *Cross*, aparece la guarda ”x >= 10”, imposibilitando que esta transición se realice antes de que hayan pasado 10 unidades de tiempo.

- También en las transiciones, las **sincronizaciones**, en azul claro, coordinan acciones con otros autómatas mediante canales. Al pasar del estado *Safe* a *Appr*, el autómata envía una señal a través del canal *appr* con el *id* del tren.

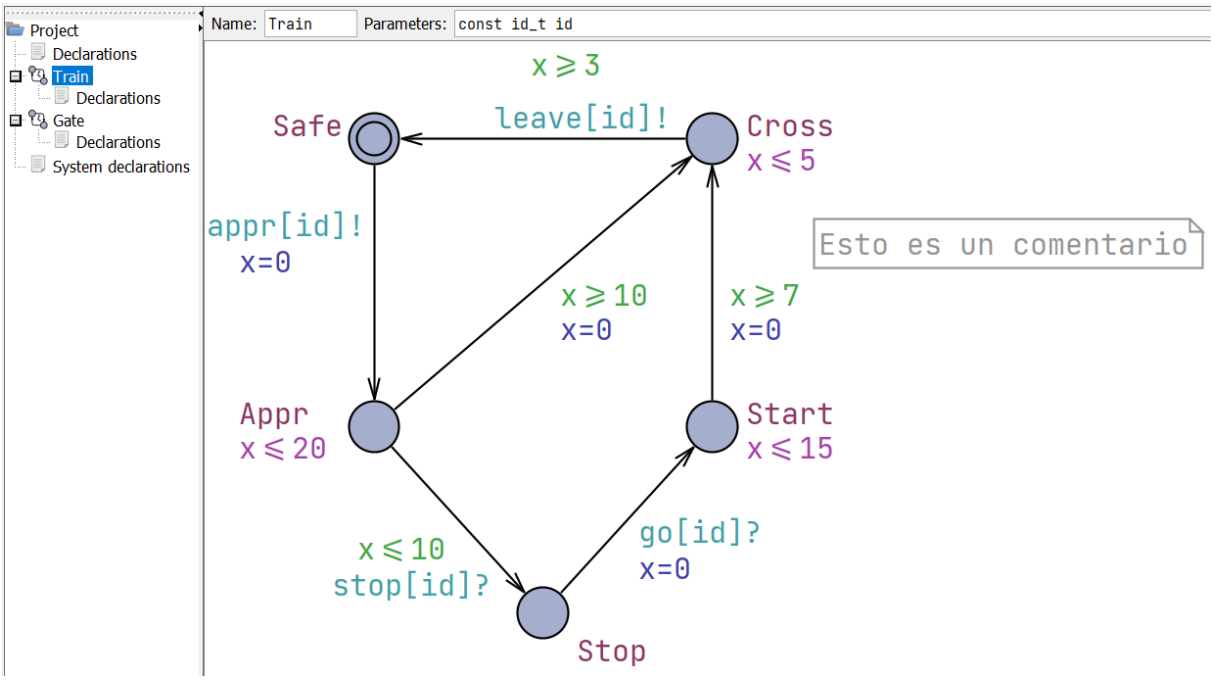

Figura 2: Ejemplo de plantilla de un autómata.

- Las **actualizaciones**, en azul oscuro y también pertenecientes a las transiciones, modifican relojes o variables al realizar la transición. Por ejemplo, al pasar de *Stop* a *Start*, se reinicia el reloj con ”x = 0”.

- También se pueden observar, en gris, **comentarios** que ayudan a comprender el modelo. Pueden aparecer tanto en estados como en transiciones. Se puede observar uno en la transición de *Start* a *Cross*.

Gracias a esta notación visual, resulta considerablemente más sencillo entender el funcionamiento básico de un modelo. Cabe mencionar, además, la existencia de los estados especiales llamados *commit*. Estos estados no permiten que transcurra tiempo entre las transiciones que pasan por ellos. Se utilizan principalmente para forzar secuencias inmediatas de acciones o evitar asincronías no deseadas entre autómatas.

Además del editor de plantillas, la herramienta cuenta con paneles de declaraciones donde se especifican parámetros globales, constantes, variables y canales que gobiernan la interacción entre los distintos autómatas. En concreto, los canales de sincronización actúan como mecanismos de comunicación entre autómatas. Para utilizar un canal, como sería *appr*, un

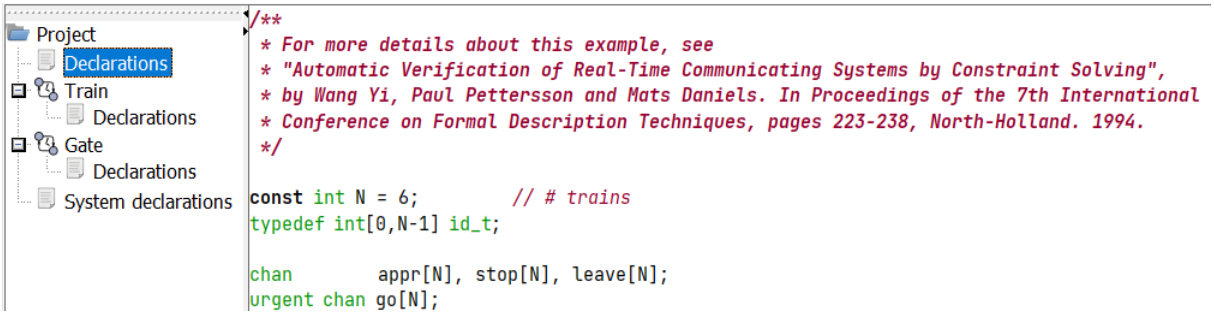

Figura 3: Declaraciones globales en UPPAAL.

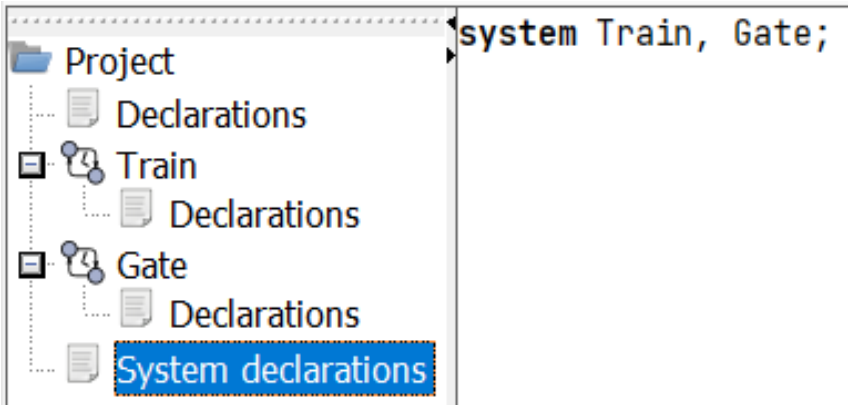

Figura 4: Declaraciones del sistema en UPPAAL.

autómata emite un mensaje por dicho canal, añadiendo el símbolo "*!*", mientras que otro lo recibe con el símbolo "*?*". La transición comunica a ambos extremos, lo que permite coordinar comportamientos que deben ocurrir de manera simultánea. En la Figura 3 se muestra un ejemplo de estos canales y demás declaraciones globales, que definen las condiciones iniciales y la estructura básica del modelo.

De forma complementaria, en las declaraciones de sistema se determina qué instancias de autómatas se ejecutan en paralelo, componiendo así el comportamiento global del modelo (Figura 4). También se puede observar en esta figura que cada autómata tiene su propio apartado para declaraciones, para relojes, variables o funciones. Esta organización permite representar de forma modular incluso sistemas considerablemente complejos.

La combinación de estos elementos permite modelar sistemas concurrentes y/o distribuidos cuyo comportamiento depende del tiempo. Mientras los estados y transiciones recogen la lógica del sistema, los relojes, las guardas y los invariantes introducen la dimensión temporal,

asegurando que las acciones se produzcan en los intervalos adecuados.

A continuación se da una semántica intuitiva del comportamiento de los autómatas temporizados utilizando el autómata de la Figura 2. Este autómata representa el comportamiento de un tren *Train(id)* que, de vez en cuando, quiere cruzar un puente estrecho cuyo acceso está controlado por otro autómata (*Gate*) que no aparece en la figura. El reloj x se utiliza para gestionar el tiempo que el tren pasa en los distintos estados. El estado inicial (*Safe*) no tiene invariante, por lo que el tren puede estar en este estado de forma indefinida. De *Safe* puede pasar a aproximarse al cruce, para lo que envía un mensaje *appr[id]!* a *Gate* y transita al estado *Appr*. En este estado puede estar a lo sumo 20 unidades de tiempo. Si antes de que pasen 10 unidades recibe el mensaje *stop[id]?* transita al estado *Stop* para esperar a que le llegue su turno para cruzar. Si no llega el mensaje *stop* puede transitar a *Cross* (estado que representa el momento en el que el tren está cruzando) para, a continuación, volver de nuevo a *Safe*. Cuando el tren está en *Stop* y recibe el mensaje *go[id]?* se pone en marcha de nuevo (transita a *Start*) y cruza el puente.

Una vez construido el modelo, UPPAAL ofrece diferentes formas de analizar su comportamiento. El simulador simbólico permite explorar las posibles ejecuciones del sistema, ya sea de manera aleatoria o eligiendo las transiciones que se quieren observar, siempre dentro de las disponibles. Adicionalmente, ofrece dos posibles vistas, como se aprecia en la Figura 5. En la que aparece en la parte superior, se observa el diagrama completo de cada instancia de nuestro modelo, mostrando el estado en el que se encuentra y las transiciones disponibles. En la vista inferior, se aprecia un diagrama donde cada instancia muestra simplemente su estado actual y los mensajes que se producen a través de los canales de sincronización. Esto ofrece una visión general de las transiciones que puede seguir el sistema sin necesidad de analizar cada valor concreto de los relojes en todas las ejecuciones, facilitando la detección temprana de errores en las fases iniciales del diseño.

Por su parte, el verificador (Figura 6) constituye la parte más potente de la herramienta, ya que permite verificar propiedades. Esta verificación de propiedades se realiza mediante lógica temporal TCTL (Timed Computation Tree Logic), utilizando un algoritmo de model checking. La lógica TCTL extiende la lógica CTL con restricciones temporales. Algunos operadores básicos son:

- **A[]** φ: en todas las ejecuciones y en todos los estados, la condición φ se cumple.

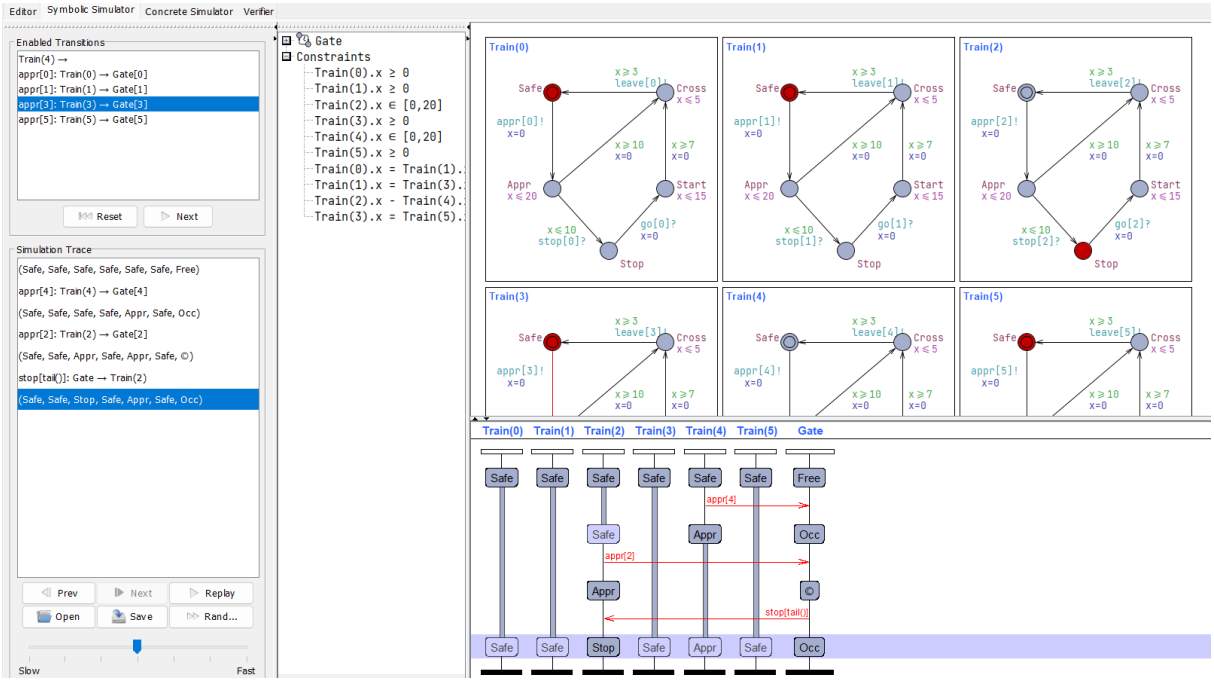

Figura 5: Simulador simbólico de UPPAAL.

- **A<>** φ: en todas las ejecuciones, eventualmente se cumple φ.
- **E<>** φ: existe al menos una ejecución en la que eventualmente se cumple φ.
- **E[]** φ: existe al menos una ejecución en la que siempre se cumple φ.
- $p - - > q$: Cuando p se cumple, eventualmente q también se cumple.
- **Pr[**$\leq T$**]** ($<>$ φ): probabilidad de alcanzar φ en un tiempo menor o igual que T.

Por ejemplo, en la Figura 6, la propiedad $E <> \ Train(0).Cross$ expresa que existe al menos una ejecución en la que eventualmente el autómata *Train(0)* llega al estado *Cross*. Por otro lado, $Train(0).Appr - - > Train(0).Cross$ expresa que, cuando *Train(0)* llega al estado *Appr*, después, eventualmente también llegará al estado *Cross*.

En definitiva, UPPAAL reúne en un único entorno las funciones necesarias para trabajar con autómatas temporizados, como son la edición gráfica, la definición modular de sus componentes, la simulación y la verificación de propiedades. Esta integración permite abordar sistemas de tiempo real de manera rigurosa, manteniendo al mismo tiempo una representación clara y manejable de su comportamiento.

Figura 6: Verificador de propiedades en UPPAAL.

2.4. Trabajos relacionados

En la literatura especializada se encuentran diversos trabajos que emplean técnicas de verificación formal para abordar la gestión del tráfico y, en particular, el control de intersecciones reguladas por semáforos. Una de las líneas más relevantes es el uso de UPPAAL como herramienta de modelado y verificación, gracias a su capacidad para representar sistemas concurrentes y temporizados de manera precisa.

En [16], los autores presentan el uso de UPPAAL *Stratego* [17] para el diseño de estrategias de control de semáforos inteligentes, integrando no solo la verificación formal, sino también la optimización bajo distintos criterios. Este enfoque permite explorar configuraciones que buscan equilibrar la seguridad con la eficiencia del tráfico, adaptando la temporización de los semáforos en función de las condiciones observadas. El trabajo demuestra que el uso de técnicas de síntesis de estrategias puede aportar soluciones más flexibles y adaptativas frente a sistemas tradicionales de control fijo.

Por otro lado, estudios como [18] se centran en el modelado formal de semáforos mediante autómatas temporizados, con el objetivo de verificar propiedades críticas como la ausencia de colisiones, la equidad en el acceso de los vehículos y la correcta sincronización entre fases. Este tipo de trabajos refuerza la importancia de la verificación formal para garantizar que los sistemas de tráfico urbano no solo cumplen con los requisitos de seguridad, sino que además lo hacen de forma verificable y reproducible.

Estos trabajos relacionados muestran que la aplicación de UPPAAL y sus extensiones al

ámbito del tráfico es una línea de investigación consolidada y en expansión. Sin embargo, la mayoría de los estudios se han enfocado en intersecciones genéricas. En contraste, el presente trabajo aporta un estudio detallado y aplicado a un caso real con particularidades propias —el cruce del Boulevard Louis Pasteur con Jiménez Fraud en el campus universitario de Teatinos— lo que contribuye a trasladar los beneficios de los métodos formales a un escenario crítico y de relevancia práctica.

3

Caso de estudio

3.1. Descripción del entorno

El caso de estudio seleccionado corresponde al cruce entre Boulevard Louis Pasteur y calle Jiménez Fraud, que podemos observar en la Figura 7. Situado en el campus de Teatinos de la Universidad de Málaga, se trata de una intersección con una elevada densidad de tráfico y una notable complejidad derivada de la concurrencia de múltiples factores, como son la circulación del metro en superficie que atraviesa la calzada, el tránsito frecuente de ambulancias debido a la proximidad del Hospital Virgen de la Victoria, y la presencia de numerosos peatones al encontrarse en un entorno universitario.

Desde el punto de vista del modelado, el entorno se compone de dos intersecciones adyacentes que comparten tráfico en la calle principal. Estas son un cruce superior, donde coinciden metro y coches que circulan por Jiménez Fraud, y un cruce inferior, donde interactúan los coches de esta misma calle con los del Boulevard Louis Pasteur. Actualmente, el cruce inferior no cuenta con un semáforo que regule el tráfico del Boulevard, pero en este trabajo se plantea cómo debería gestionarse en caso de instalarse uno, de manera que ambos semáforos funcionen coordinados entre sí y en relación con el metro y los vehículos prioritarios.

La elección de este escenario no es arbitraria. Por su localización, es un entorno cercano y familiar, lo que facilita el análisis empírico, y al mismo tiempo presenta la complejidad suficiente para un estudio académico de carácter técnico. A diferencia de un cruce simple, aquí resulta necesario abordar aspectos avanzados como la sincronización entre múltiples actores, la gestión de prioridades y la validación de propiedades temporales. Además, se han registrado accidentes en los que se han visto implicados tanto vehículos como el metro [19], [20], lo que subraya la importancia de una coordinación rigurosa orientada a la seguridad y al rendimiento del sistema.

Figura 7: Imagen satélite de Google Maps del cruce entre Boulevard Louis Pasteur y Jiménez Fraud.

Finalmente, cabe señalar que no todos los detalles físicos y operativos se representan íntegramente en los modelos. Por razones de trazabilidad, complejidad y limitaciones computacionales, ciertos aspectos se abstraen o simplifican, mientras que otros se introducen progresivamente en sucesivas iteraciones. En los capítulos siguientes se especificará qué componentes se incluyen en cada modelo, qué supuestos se han adoptado y qué aspectos se han omitido o aproximado, así como la estrategia empleada para la verificación según la escala del sistema.

3.2. Requisitos identificados

El análisis del entorno permitió extraer un conjunto de requisitos iniciales que guían el diseño del sistema de semáforos inteligentes:

- Minimizar el riesgo de colisiones entre coches y metro, garantizando que nunca coincidan en el cruce.
- Coordinar los semáforos en ambos sentidos de la calle principal para mantener la cohe-

rencia del flujo de tráfico.

- Dar servicio también al Boulevard Louis Pasteur, asegurando que los vehículos que circulan por él no queden en espera indefinida.
- Permitir el paso fluido de ambulancias, evitando retrasos en situaciones críticas.
- Lograr un equilibrio entre seguridad y eficiencia, priorizando la primera pero sin comprometer de forma excesiva la fluidez del tráfico.

Estos requisitos fueron refinándose a lo largo de las distintas iteraciones de modelado, de modo que algunos se formularon inicialmente de forma general y posteriormente, como veremos en las secciones referentes a verificación y resultados, se concretaron como propiedades verificables con UPPAAL.

4

Primer modelo: Cruce superior

El primer modelo desarrollado, cuyo diagrama podemos ver en la Figura 8, tuvo como objetivo reproducir de manera simplificada el funcionamiento de un cruce en el que confluyen coches y metro en superficie. Se trata de una primera aproximación destinada a comprobar la viabilidad de representar el sistema mediante autómatas temporizados en UPPAAL y verificar propiedades básicas de seguridad y viveza. Concretamente, los actores serán dos metros, uno en cada sentido de la marcha; dos coches, igualmente uno en cada sentido; y dos semáforos de coche, que regularán el tráfico, uno para cada dirección.

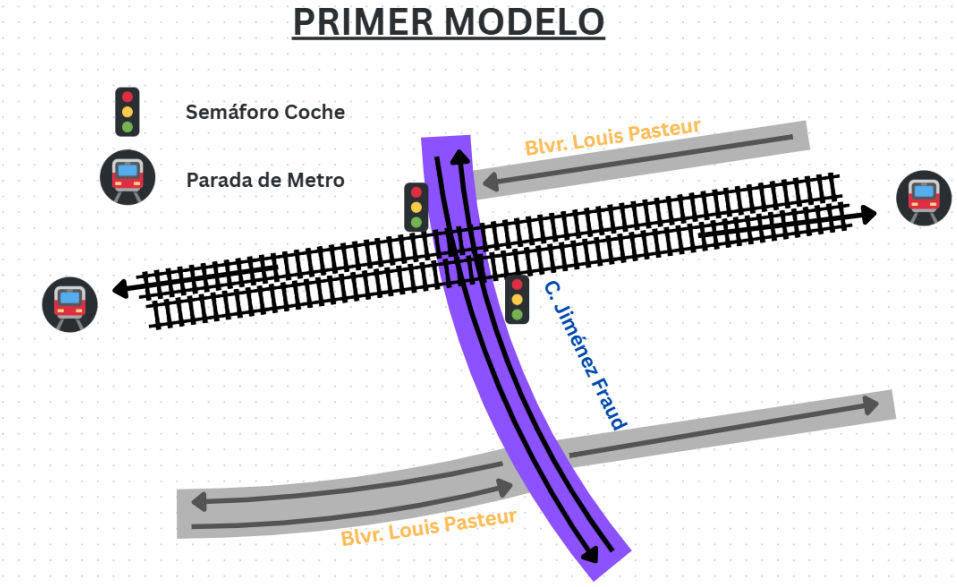

Figura 8: Diagrama del cruce del primer modelo hecho en Canva.

La lógica del diseño se basó en darle prioridad al metro, antes que a los coches, para cruzar. De esta forma, cuando un metro llega a su parada, debe hacer que los semáforos de los coches pasen a rojo, para que al salir de la parada pueda cruzar con prioridad. En las secciones

siguientes se explicará más detalladamente el comportamiento de cada actor por separado, las simulaciones realizadas y las propiedades verificadas, facilitando la comprensión del modelo como conjunto.

4.1. Consideraciones y problemas de diseño

El diseño del primer modelo estuvo condicionado por la necesidad de garantizar un sistema seguro, aunque ello implicara cierto grado de simplificación. Por ello, se otorgó al metro prioridad absoluta en la intersección, de manera que no fue necesario incluir un semáforo específico para regular su paso. Su capacidad de atravesar el cruce dependía únicamente de que no hubiera coches en él, lo que permitió un modelado inicial más abarcable.

Al incluir varias instancias de cada actor, como hemos comentado, se descartó el uso de variables booleanas para señalar la ocupación del cruce, ya que resultaban inadecuadas ante múltiples instancias. En su lugar, se emplearon contadores, como *cochesEnCruce* y *metrosEnCruce*, que ofrecieron mayor precisión y evitaron conflictos de sincronización. Se podría pensar que esto describe un simple problema de exclusión mutua; sin embargo, la naturaleza de los estados *commit* en los autómatas de los actores, hace que la solución más sencilla y efectiva sea la empleada.

Durante el desarrollo también se ajustaron cuidadosamente los parámetros de tiempo de los actores, con el doble objetivo de acercar el modelo al comportamiento real y garantizar que ocurrieran sucesos menos comunes, pero posibles, como que un metro pueda entrar en estado de alarma. Este ajuste permitió comprobar situaciones de riesgo controlado y asegurar que el sistema no quedara bloqueado. Esta configuración puede observarse en la Figura 9, donde, además de los parámetros temporales, también se muestran los canales de sincronización necesarios y las variables globales empleadas en el modelo. Entre estas variables, nos encontramos 3 contadores con nombres autoexplicativos: *cochesEnCruce*, *metrosEnCruce* y *metrosEnParada*. Además, también se muestra la variable que representa el valor de los semáforos, uno de cada sentido. Como aparece en la figura, para esta variable de semáforos, el valor 0 representa el rojo y el valor 1, el verde.

Finalmente, para mantener la coherencia en el flujo de actores, se definió la circulación como un ciclo infinito en el que coches y metros reaparecen al inicio tras completar su trayecto. En conjunto, estas decisiones y ajustes permitieron consolidar un modelo robusto, con un

```
// Parámetros de tiempo de Metro
const int tParadaMax = 15;
const int tParadaMin = 10;
const int tMetroEnCruceMax = 10;
const int tMetroEnCruceMin = 5;
const int tMetroLejosMax = 90;
const int tMetroLejosMin = 50;

// Parámetros de tiempo de Coche
const int tCocheLLegaSemaforoMax = 30;
const int tCocheLLegaSemaforoMin = 15;
const int tCocheEnCruceMax = 15;
const int tCocheEnCruceMin = 5;
const int tCocheLejosMax = 90;
const int tCocheLejosMin = 50;

bool semaforoCoche[2]= {0,0} ;//0 -> rojo 1 -> verde
int cochesEnCruce = 0;
int metrosEnCruce = 0;
int metrosEnParada = 2; //Empieza a 2 ya que "enLaParada" es el estado inicial

// Canal de sincronización para indicar paso permitido
broadcast chan greenCoche[2];
broadcast chan metroAproximando; //No es necesario que sea urgent
broadcast chan metroSaliendo;    //No es necesario que sea urgent
```

Figura 9: Parámetros de tiempo, variables y canales del modelo 1 en UPPAAL.

comportamiento consistente y adecuado para servir como base de las iteraciones posteriores.

4.2. Autómatas principales

El sistema se estructura a partir de tres autómatas fundamentales: metro, coche y semáforo de coches.

Metro. La Figura 10 muestra el autómata (la plantilla) que modela el comportamiento del metro. Este se inicializa con un valor *d* binario, que representa la dirección de viaje. El valor 0 indica que viaja hacia la izquierda, y el valor 1, por su parte, representa que viaja hacia la derecha. Como hemos mencionado, en el sistema habrá dos instancias de esta plantilla, cada una modelará un convoy moviéndose en cada uno de los sentidos. El autómata reproduce, de forma resumida, su dinámica al aproximarse, atravesar y abandonar el cruce.

Parte del estado inicial *enLaParada*, donde permanece detenido un tiempo máximo correspondiente al valor *tParadaMax*. Desde este estado, puede evolucionar hacia *DentroCruce* o hacia *Alarma*. Veamos primero la ejecución más común, hacia la primera opción. Una vez haya transcurrido el tiempo *tParadaMin*, y siempre que no haya ningún coche en el cruce, el metro

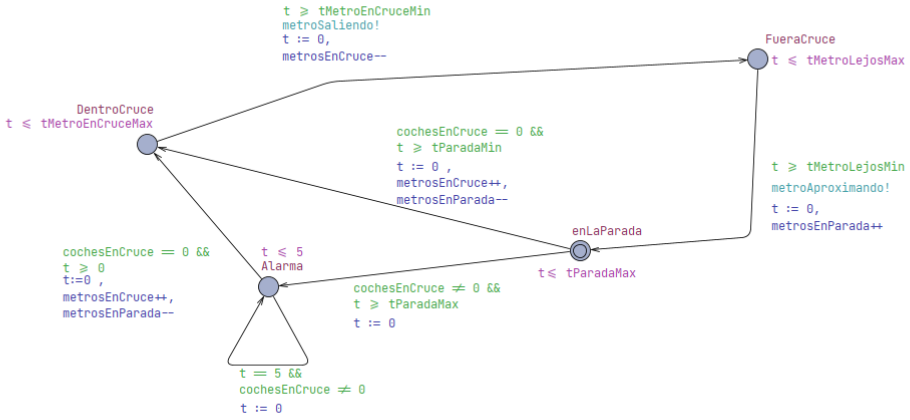

Figura 10: Autómata de metro en el modelo 1 en UPPAAL.

puede realizar la transición hacia *DentroCruce*, actualizando los valores de *metrosEnCruce* y de *metrosEnParada*; sumando una unidad al primero y restando otra al segundo.

La otra opción es que el metro llegue al estado *Alarma*. Esto ocurre cuando el metro agota el tiempo máximo que puede permanecer en la parada y sigue habiendo algún coche en el cruce. En ese caso, se considera que existe una emergencia, y el metro pasa al estado *Alarma*. En este estado, el metro comprueba cada cinco unidades de tiempo si sigue habiendo un coche en el cruce. En el caso de que así sea, permanece en el estado de alarma y vuelve a comprobarlo. En cambio, cuando se libere el cruce, el metro sigue hacia el estado *DentroCruce*, actualizando, igual que en el caso anterior, los valores de *metrosEnCruce* y de *metrosEnParada*.

Independientemente de si el metro pasa por el estado de alarma o no, cuando se encuentra dentro del cruce, se comporta de igual manera. Una vez pasado un tiempo mínimo de *tMetroEnCruceMin* y antes de un máximo de *tMetroenCruceMax*, el metro realiza la transición hacia *FueraCruce*. Al completarla, actualiza el valor de los metros que se encuentran en el cruce, restándose a él mismo, y envía por el canal de sincronización *metroSaliendo* el mensaje de que se encuentra en proceso de salir del cruce. Cuando sale del cruce, se considera que, pasado un tiempo mayor que el de las demás transiciones, vuelve a llegar al estado *enLaParada*. Al hacerlo, actualiza el valor *metrosEnParada* y envía por el canal de sincronización *metroAproximando*

el aviso de que se está volviendo a aproximar al cruce.

Coche.

El autómata que modela al coche, que se puede consultar en la Figura 11, también se inicializa con un valor *d* binario, que representa la dirección de viaje. En este caso, el valor 0 se corresponde con la dirección ascendente; y el valor 1 con la descendente. De manera similar al metro, y de forma resumida, este también sigue un ciclo repetitivo que simula un flujo infinito en el que constantemente se acerca al cruce, lo atraviesa y sale de él para volver a aproximarse después.

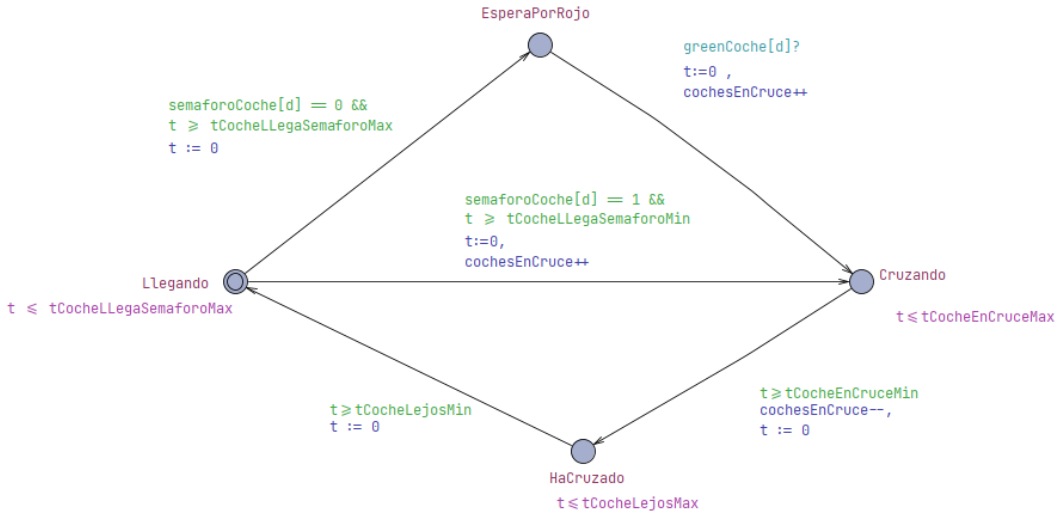

Figura 11: Autómata de coche en el modelo 1 en UPPAAL.

Los coches se inician desde el estado *Llegando*, el cual representa que se están aproximando al semáforo que regula su paso. Si el semáforo de su dirección está en verde, y una vez haya transcurrido un tiempo mínimo *tCocheLlegaSemaforoMin*, el coche entra al cruce y actualiza el valor *cochesEnCruce* sumando una unidad. Por lo tanto, pasa al estado *Cruzando*.

En el caso de que el semáforo no se encuentre en verde, el coche realiza la transición hacia *EsperaPorRojo*. Esta solo se realiza cuando ha transcurrido el tiempo máximo *tCocheLlegaSemáforoMax*. De esta forma se consigue que solo transite a este estado cuando no tiene otra opción y se le agota el tiempo estimado. Con esta decisión se busca simular el comportamiento de re-

ducir la velocidad de tu vehículo si ves un semáforo rojo de lejos, esperando a que cambie a verde sin llegar a detener el coche. Una vez el coche se encuentra en el estado *EsperaPorRojo*, espera a que el semáforo se ponga en verde, sin invariantes ni guardas de tiempo, ya que no podrá cruzar nunca mientras el semáforo siga rojo, pase el tiempo que pase. Este cambio de color lo recibe a través del canal de sincronización *greenCoche[d]*, siendo *[d]* la dirección del semáforo que cambia a verde. Al recibirlo, actualiza la variable *cochesEnCruce* sumando uno y realiza la transición hacia *Cruzando*.

Una vez en *Cruzando*, al transcurrir un tiempo entre *tCocheEnCruceMax* y *tCocheEnCruceMin*, se resta una unidad al contador de los coches en el cruce y se pasa al estado *HaCruzado*. Al igual que ocurre con el metro, posteriormente simula que, una vez pasado un tiempo, vuelve a aparecer en el estado *Llegando*, aproximándose otra vez al cruce.

Semáforo de coches. El tercer autómata es el encargado de regular el paso de los vehículos. Como parámetro inicial, también almacena un valor binario *d* que indica el sentido de la marcha sobre el que tiene control. Sus estados básicos son *Rojo* y *Verde*, que representan las dos configuraciones posibles del semáforo. Se pueden ver sus estados y transiciones en la Figura 12.

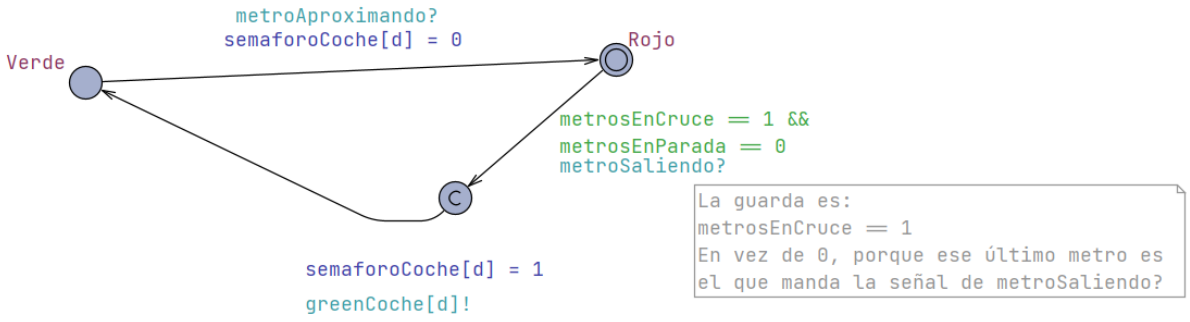

Figura 12: Autómata del semáforo de coche en el modelo 1 en UPPAAL.

De estos, el estado inicial es *Rojo*, ya que los metros se inician en la parada, próximos al cruce, y los coches no tienen permitido el paso cuando la parada está ocupada. Estando en rojo, el semáforo espera a recibir un mensaje por el canal *metroSaliendo*, que indique que hay un metro abandonando el cruce. Cuando esto ocurra, comprueba dos condiciones: que no haya ningún otro metro en la parada, y que solo haya un metro en el cruce. Esto puede parecer un

error, ya que necesitamos que el cruce esté libre para que el semáforo cambie a verde, pero es necesario, ya que este metro que aparece en el cruce es el que ha enviado el mensaje *metroSaliendo*, y por tanto, está prácticamente fuera del cruce, pero aún no ha actualizado la variable *metrosEnCruce*. Al cumplirse ambas condiciones, no pasamos directamente a *Verde*, sino que pasamos a un estado *commit* intermedio. Gracias a este, podemos realizar otra sincronización, en este caso avisar a los coches que esperen en *EsperaPorRojo* de que ya se encuentra en verde, además de actualizar las variables del semáforo al valor 1, que representa el color verde.

Una vez se encuentra en *Verde*, la transición hacia *Rojo* de vuelta es mucho más sencilla. Simplemente, cuando se reciba el mensaje de que hay un metro cercano al cruce, a través del canal *metroAproximando*, se actualizan los valores del semáforo de su dirección al valor 0, representando el color rojo. Se cambia así al estado *Rojo*, y vuelve a empezar.

4.3. Resultados de las simulaciones

Las simulaciones realizadas con el primer modelo evidencian que el sistema reproduce de manera coherente la interacción entre coches, semáforo y metro. Los coches obedecen la señal del semáforo y se detienen cuando este se encuentra en rojo, mientras que reanudan la marcha al ponerse en verde, siempre que no haya un metro en el cruce. Por su parte, los metros llegan a la parada, avisan a los semáforos de coches y atraviesan el cruce con prioridad.

En las trazas generadas se observa cómo, en determinados casos, muy poco comunes, el metro entra en el estado de *Alarma* cuando coincide con coches aún dentro del cruce. Se puede observar una traza donde esto ocurre en la Figura 13. No obstante, en todas estas situaciones, el metro logra eventualmente reanudar su trayecto, confirmando que el sistema no queda bloqueado.

Asimismo, el comportamiento de los semáforos refleja la sincronización prevista. Ambos se activan de manera coordinada, evitando configuraciones inconsistentes como un semáforo en rojo y otro en verde. El estado *commit* incorporado en el modelo justifica pequeñas asincronías, pero estas no afectan a la corrección global del sistema.

En la Figura 14 se puede apreciar, en la otra vista que ofrece el simulador simbólico, una cadena de transición de estados que evidencia el buen funcionamiento del sistema. Un metro sale del cruce mientras el otro se encuentra ya fuera, enviando así la señal de que los semáforos cambien a verde. Uno de los cuales permite cruzar al *Coche1* después de encontrarse este

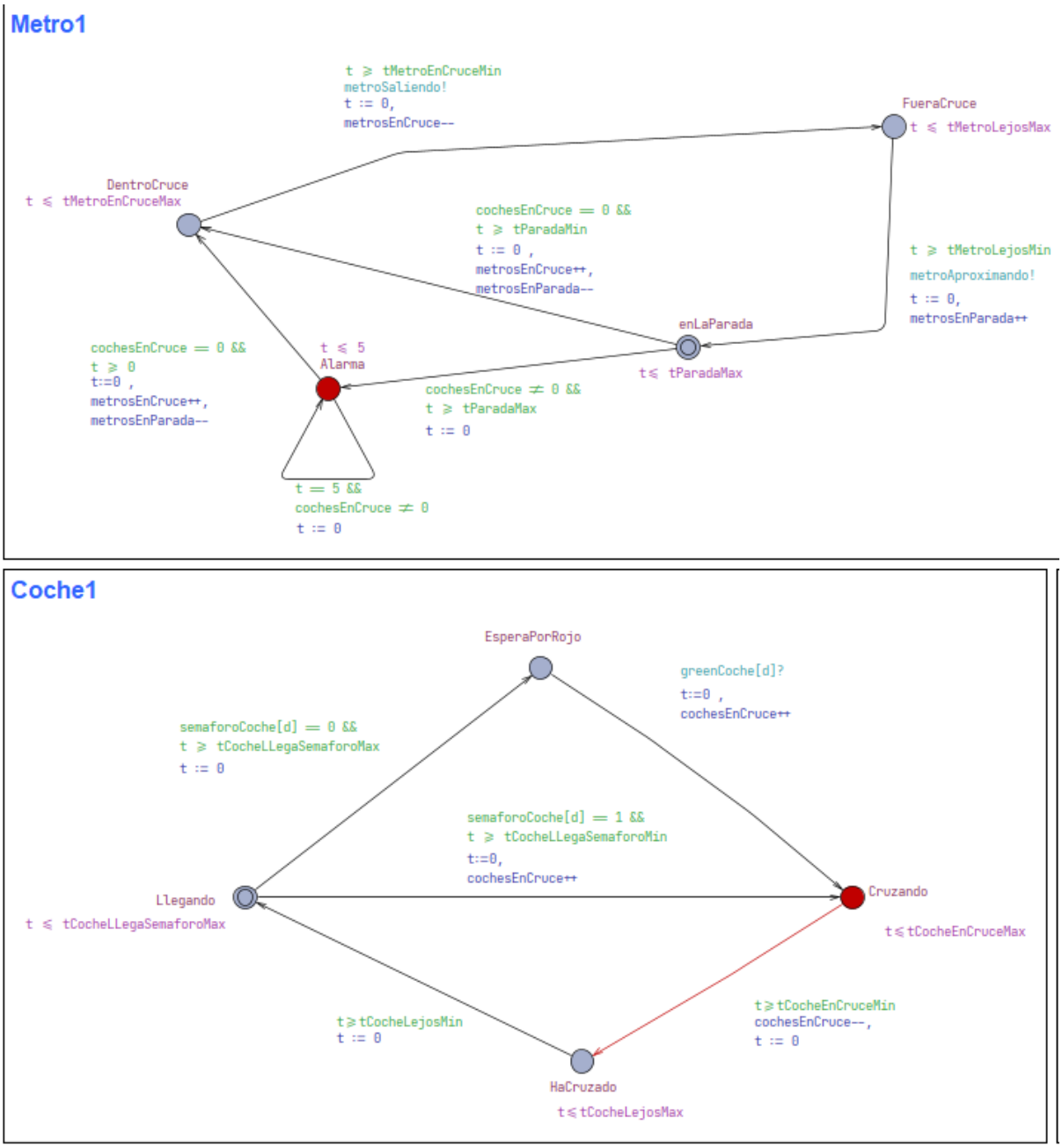

Figura 13: Estado del modelo 1 en el simulador simbólico donde un metro entra en estado de alarma.

esperando en el estado *EsperaPorRojo*.

Finalmente, la naturaleza cíclica del modelo garantiza un flujo constante tanto de coches como de metros, lo que permite comprobar de manera reiterada los distintos escenarios de interacción y validar el comportamiento en múltiples iteraciones.

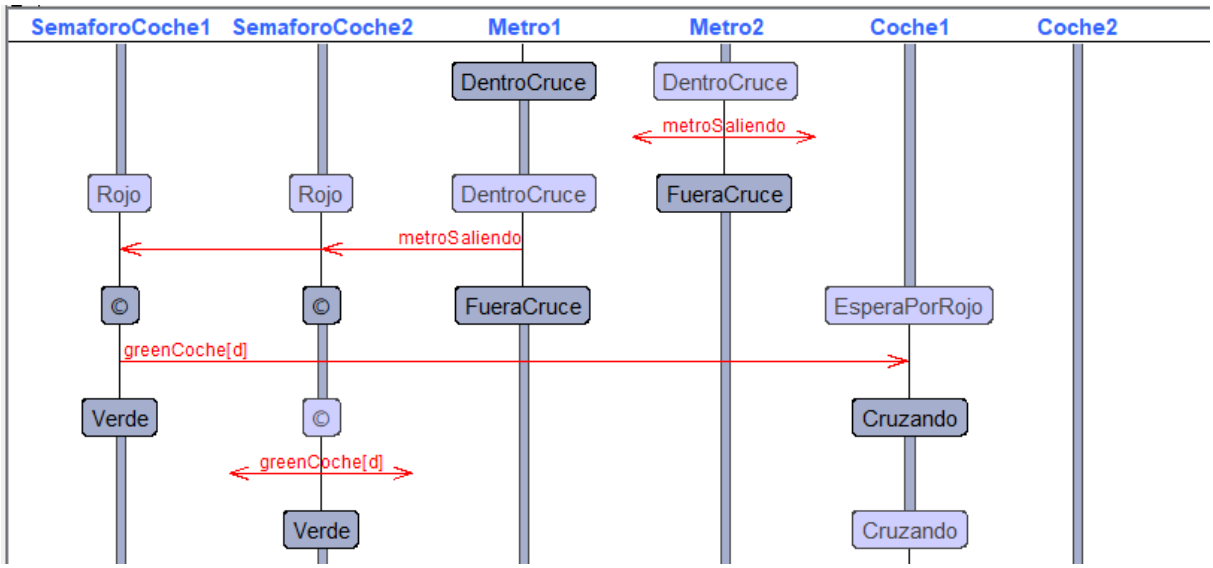

Figura 14: Traza del modelo 1 en el simulador simbólico de UPPAAL.

4.4. Verificación de las propiedades

El verificador de UPPAAL permitió comprobar de manera exhaustiva un conjunto de propiedades de seguridad, viveza y rendimiento, tanto mediante técnicas exhaustivas como con análisis estocástico. Los resultados obtenidos muestran, en primer lugar, que el sistema nunca entra en *deadlock*, lo que garantiza la ausencia de bloqueos y asegura que siempre exista alguna transición posible para los actores. Asimismo, se verificó que tanto los coches como los metros terminan cruzando en todas las trazas, independientemente del estado inicial, lo que confirma la viveza del sistema y la imposibilidad de bloqueos permanentes.

En relación con los estados de alarma, se observó que los metros pueden alcanzarlos en determinadas trazas, pero que desde ellos siempre logran continuar y finalizar el cruce, evidenciando que el modelo contempla situaciones de riesgo controlado y resuelve de forma correcta la recuperación. También se confirmó la coherencia de los semáforos, que pueden estar en verde al mismo tiempo, pero nunca en configuraciones contradictorias, como uno en rojo y otro en verde; las pequeñas asincronías detectadas se explican únicamente por la presencia de estados *commit*.

Por otro lado, las propiedades individuales de los coches muestran que, tanto al aproximarse como al esperar en rojo, siempre acaban cruzando, mientras que el metro, desde la parada

o incluso desde el estado de alarma, logra igualmente atravesar la intersección. Esto asegura un flujo constante de todos los actores. Además, se comprobó que nunca se da la situación en la que un coche y un metro coincidan dentro del cruce, garantizando la seguridad del sistema, aunque sí se permite la concurrencia de varios coches o varios metros simultáneamente cuando no supone riesgo de colisión.

Finalmente, el análisis estocástico evidenció que la probabilidad de que un coche llegue al cruce en menos de 30 unidades de tiempo supera el 95 %, mientras que la de que un metro permanezca en alarma durante 1000 unidades de tiempo se mantiene por debajo del 5 %. La Figura 15 muestra la verificación de estas propiedades en la herramienta UPPAAL. Para consultar la especificación de estas, se puede consultar la Tabla 1 del Apéndice A. En conjunto, estos resultados confirman que el primer modelo cumple con los requisitos fundamentales de seguridad y viveza, constituyendo una base sólida para el desarrollo de versiones posteriores más complejas.

Figura 15: Propiedades verificadas del modelo 1 en UPPAAL.

5

Segundo modelo: Ampliación con cruce inferior

El segundo modelo desarrollado amplía el escenario anterior incorporando el cruce situado en la parte inferior de la calle Jiménez Fraud, lo que añade nuevos elementos y una mayor complejidad en la gestión de semáforos. En la versión completa del modelo, cuyo esquema puede observarse en la Figura 16, se introducen dos semáforos adicionales para regular cada sentido de circulación en Jiménez Fraud y un semáforo más para los vehículos que acceden desde el Boulevard Louis Pasteur. La lógica de coordinación sigue estando gobernada por el paso del metro, es decir, mientras el metro no se aproxima ni cruza la intersección, los vehículos de Jiménez Fraud disponen de paso en verde, manteniendo en rojo el semáforo del Boulevard Louis Pasteur. Cuando el metro entra en zona de influencia, los semáforos de Jiménez Fraud se ponen en rojo y se habilita el verde para el Boulevard, reproduciendo así un mecanismo de prioridad análogo al definido en el primer modelo para el cruce superior.

No obstante, este incremento en el número de elementos supuso un notable crecimiento del espacio de estados en la verificación, lo que hacía impracticable comprobar algunas propiedades en tiempos razonables. Para solventar este problema, se diseñó una versión reducida del modelo, en la que los semáforos inferiores de Jiménez Fraud fueron sustituidos por un tramo de paso regulado mediante un cebreado amarillo. En esta configuración, que podemos ver en la Figura 17, los vehículos de Jiménez Fraud no pueden detenerse sobre la zona marcada, lo que permite que, cuando se les da la orden de alto, dejen libre el acceso a los vehículos que circulan por el Boulevard Louis Pasteur, que disponen entonces de semáforo en verde.

De esta forma, los actores que se concretan para este modelo reducido, sobre el que se ve-

Figura 16: Diagrama del cruce del segundo modelo completo.

rificarán las propiedades, serán: Los dos coches, semáforos y metros ya definidos en el modelo anterior, además de un nuevo coche que circulará por el Boulevard Louis Pasteur, y el semáforo que regula su paso. Esta simplificación mantuvo la lógica general del sistema, a la vez que redujo de manera significativa la complejidad de la verificación formal, permitiendo analizar propiedades de seguridad y viveza de forma efectiva.

5.1. Consideraciones y problemas de diseño

En el segundo modelo se ajustaron cuidadosamente los parámetros temporales de todos los actores, con el fin de lograr un comportamiento lo más fiel posible a la realidad y garantizar que las propiedades verificadas tuvieran un significado consistente. Estos parámetros, junto con las variables y canales empleados, pueden observarse en la Figura 18. Concretamente, se puede ver un nuevo contador *cochesEnCruceInferior*, dos nuevos canales que comunicarán a los coches con los nuevos semáforos y las variables referentes a dichos semáforos. En el modelo reducido, aunque no se utilizan todos los parámetros, tampoco se introducen nuevos, manteniéndose la coherencia con la versión más amplia.

Al ampliarse la circulación de los coches de Jiménez Fraud para incluir un cruce adicional, surgió la duda de si un mismo autómata podía seguir representando vehículos en direcciones opuestas. Sin embargo, se confirmó que esta aproximación continuaba siendo válida, ya que el

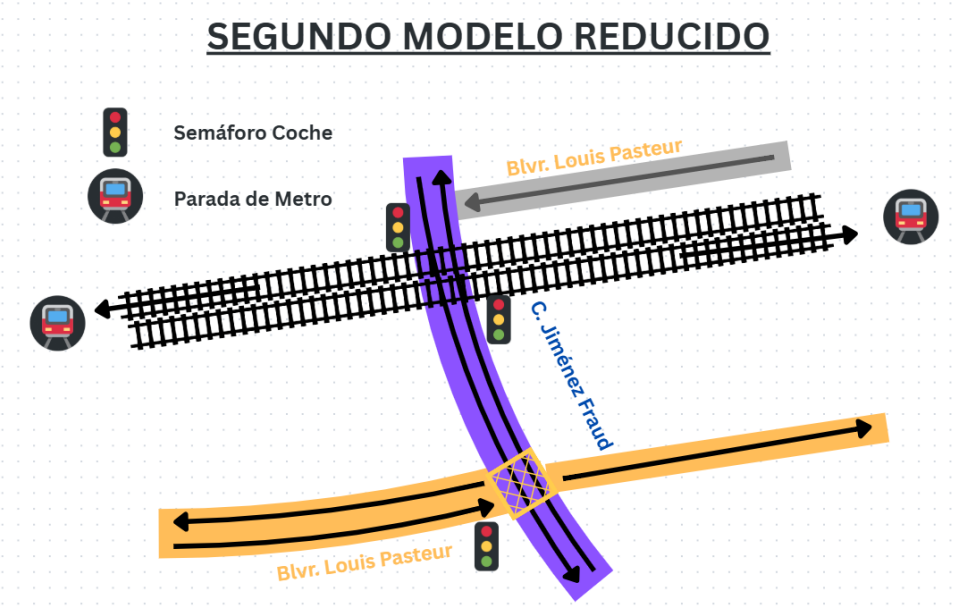

Figura 17: Diagrama del cruce del segundo modelo reducido.

bucle infinito de coches reproduce de manera cíclica los mismos estados, independientemente del sentido de circulación. En la misma línea, se mantuvo la hipótesis de bucles infinitos para los coches que atraviesan el Boulevard Louis Pasteur, asumiendo que tras un tiempo de circulación regresan a su punto inicial.

El principal problema detectado fue el elevado tiempo de verificación, consecuencia del crecimiento del espacio de estados. Para solucionarlo, como ya se ha mencionado, se construyó una versión reducida del modelo. En el contexto de esta reducción, además, la circulación de los coches de Jiménez Fraud se limita a dos vueltas. Esta restricción redujo significativamente el número de estados y transiciones, acortando el tiempo de verificación sin comprometer la validez de las comprobaciones.

5.2. Autómatas principales

En esta sección, se hace una distinción entre los autómatas referentes al modelo completo y los referentes al modelo reducido.

```
// ITERACIÓN 2:

int cochesEnCruceInferior = 0;

bool semaforoCocheInferior[2]= {0,0} ; //0 -> rojo 1 -> verde
bool semaforoCocheInferiorPasteur= 1 ; //0 -> rojo 1 -> verde

broadcast chan cochePasteurVerde;

broadcast chan verdeCocheInf[2];

// Parámetros de tiempo de Coche Inferior Louis Pasteur
const int tCocheInferiorLLegaSemaforoMax = 15;
const int tCocheInferiorLLegaSemaforoMin = 8;
const int tCocheInferiorEnCruceMax = 7;
const int tCocheInferiorEnCruceMin = 2;
const int tCocheInferiorLejosMax = 60;
const int tCocheInferiorLejosMin = 30;

// Nuevos parámetros de tiempo de Coche
const int tCocheLLegaSemaforoInfMax = 30;
const int tCocheLLegaSemaforoInfMin = 15;
const int tCocheEnCruceInfMax = 10;
const int tCocheEnCruceInfMin = 5;
const int tCocheHaCruzadoInfMax = 25;
const int tCocheHaCruzadoInfMin = 10;
```

Figura 18: Parámetros de tiempo, variables y canales del modelo 2 completo en UPPAAL.

5.2.1. Modelo completo

Para este modelo completo, se añaden tres nuevos autómatas. Estos son el semáforo inferior de Jiménez Fraud, el semáforo del Boulevard Louis Pasteur, y los coches que circulan por este último. Además, respecto al primer modelo, se modifica el autómata referente al coche y se mantienen el semáforo de coche (que ahora se convierte en el semáforo superior de Jiménez Fraud) y el metro.

Coche. Como se ha mencionado, se modifican los estados y transiciones del coche del anterior modelo para que continúe funcionando correctamente en este nuevo sistema. Se puede ver el nuevo diagrama en la Figura 19. A primera vista, este autómata puede parecer considerablemente más difícil de entender que el del modelo anterior. Sin embargo, simplemente se ha duplicado el comportamiento de dicho modelo, para que suceda tanto en el cruce superior

como en el inferior.

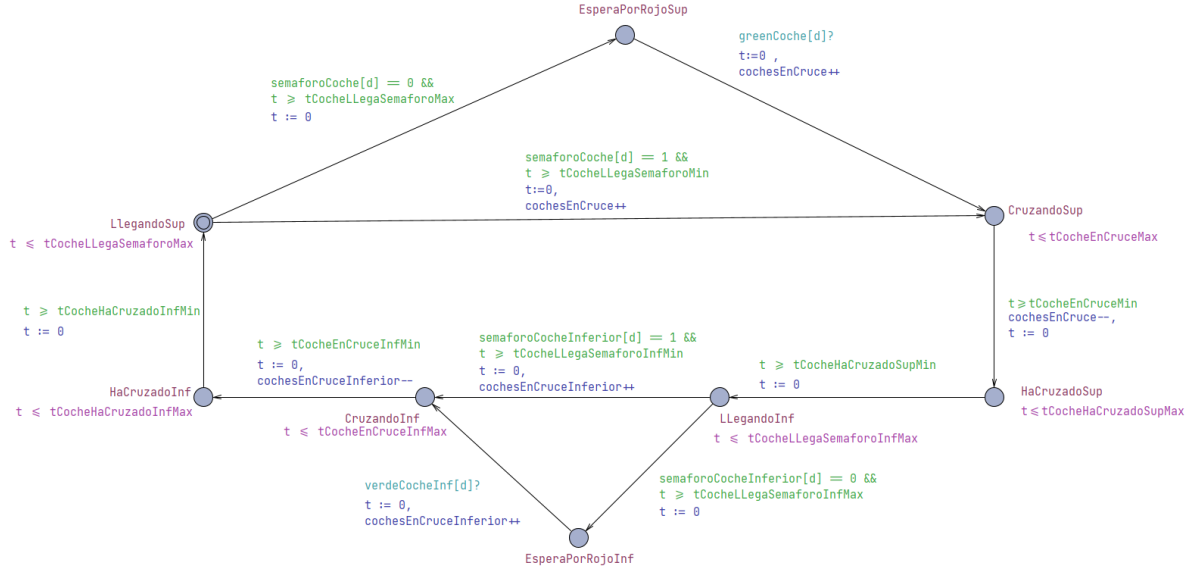

Figura 19: Autómata de coche en el modelo 2 completo en UPPAAL.

Comienza en el estado *LlegandoSup*, que representa que está aproximándose al semáforo superior. Después, al igual que se ha explicado en el capítulo anterior, o bien entra en el cruce pasando a *CruzandoSup*, o bien realiza la transición a *EsperaPorRojoSup* para después continuar hacia *CruzandoSup*. Todo esto ocurre con las mismas restricciones temporales, actualizaciones de variables y mensajes a través de canales de sincronización ya explicados anteriormente.

Una vez ha atravesado el cruce superior, realiza las transiciones análogas a través de los estados análogos en el cruce inferior. Esto es, pasa por el estado *LlegandoInf*, de ahí puede realizar la transición hacia *CruzandoInf* si el semáforo se encuentra en verde, o a *EsperaPorRojoInf* si se le agota el tiempo de espera y el semáforo continúa en rojo; desde este último estado, pasa a *CruzandoInf* cuando reciba el mensaje de que el valor del semáforo ha cambiado. Todas estas transiciones se realizan, al igual que en el cruce superior, con las actualizaciones, comprobaciones y sincronizaciones pertinentes, y sin violar ninguna invariante de ningún estado. Tras atravesar el cruce inferior, vuelve al principio, aproximándose al semáforo superior y repitiendo el bucle.

SemaforoCocheInferior. Es prácticamente idéntico al semáforo del cruce superior, visto en el modelo anterior. De igual manera, comienza en rojo, ya que los metros se inician en la parada. Cuando sale el último metro del cruce y no hay otro en la parada, pasa a *Verde*.

Después, cuando un metro se aproxima, pasa a *Rojo*. La única diferencia es, obviamente, que este semáforo controla los canales de sincronización y actualiza las variables referentes a los semáforos del cruce inferior. Se pueden observar estos estados y transiciones en la Figura 20.

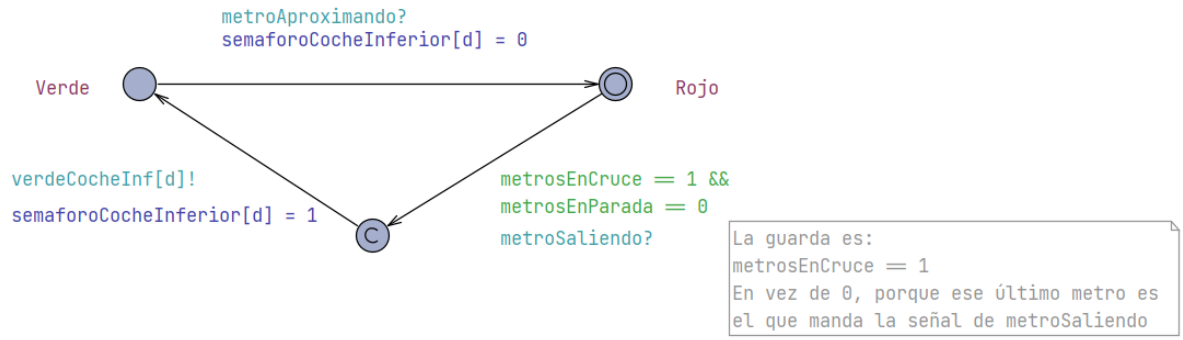

Figura 20: Autómata del semáforo inferior de Jiménez Fraud en el modelo 2 completo en UPPAAL.

SemaforoCocheInferiorLouisPasteur. Su comportamiento, que se puede apreciar en la Figura 21, es totalmente opuesto a ambos semáforos de la calle Jiménez Fraud, tanto el superior como el inferior. Este comienza en el estado *Verde*, ya que siempre debe tener el valor opuesto a los demás, y el estado inicial de estos es el rojo.

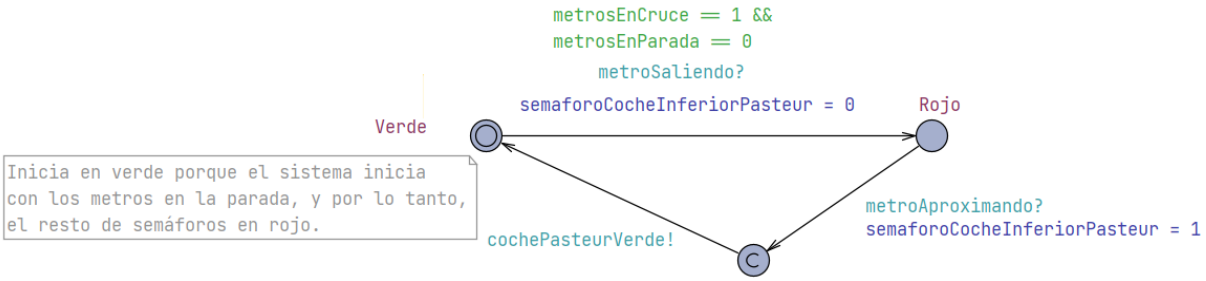

Figura 21: Autómata del semáforo de Boulevard Louis Pasteur en el modelo 2 completo en UPPAAL.

Así, en primera instancia, permite el paso a los coches que circulan por el Boulevard Louis Pasteur. Después, se puede observar que la sincronización y guarda que anteriormente se usaba para cambiar a verde, en este caso se usa para realizar la transición hacia el estado *Rojo*. Esto es, cuando se comprueba que no hay ningún metro en la parada y el único que hay en el cruce,

nos envía el mensaje de que está saliendo. Esta transición permite la circulación por la calle Jiménez Fraud y, por lo tanto, detiene el tráfico del Boulevard Louis Pasteur.

También de forma opuesta, cuando se reciba el mensaje de que hay un metro aproximándose y los semáforos de Jiménez Fraud cambien a rojo, en este autómata, se actualizará el valor del semáforo a 1, es decir, a verde, y enviaremos la sincronización a través del canal *cochePasteurVerde*, por si existe algún coche en el Boulevard Louis Pasteur esperando en el semáforo en rojo en ese mismo momento.

CocheInferiorLouisPasteur. Finalmente, el coche que circula por el Boulevard Louis Pasteur (Figura 22) funciona de forma muy similar al coche del primer modelo, con estados semejantes como *Llegando*, *EsperaPorRojo*, *Cruzando* y *HaCruzado*. Al igual que este, el coche al que representa este autómata se aproxima al semáforo, comprueba su estado; si está verde, cruza. En cambio, si está rojo, espera a que se cambie a verde y entonces cruza. También simula un bucle infinito, por lo que al cruzar reaparece otra vez aproximándose al semáforo. La variable que actualiza, los semáforos que comprueba, y las sincronizaciones que recibe, además de todos los parámetros referentes al tiempo que puede tardar en realizar estas transiciones, son todos referentes a su dominio, y constituyen la mayor diferencia entre el autómata de coche del primer modelo y este.

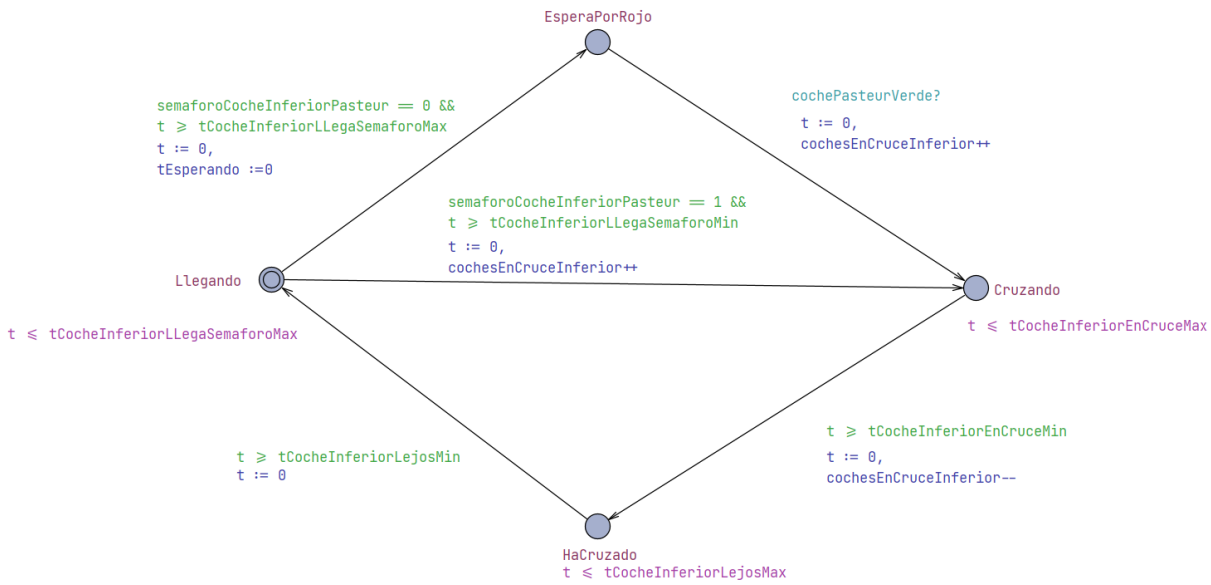

Figura 22: Autómata de coche de Boulevard Louis Pasteur en el modelo 2 completo en UPPAAL.

5.2.2. Modelo reducido

Para este modelo reducido, se prescinde de los semáforos inferiores de Jiménez Fraud. Además, el autómata del semáforo del Boulevard Louis Pasteur (Figura 21), y el de los coches que circulan por este (Figura 22) permanecen igual que en el modelo completo. El otro cambio significativo se realiza con el autómata de los coches de Jiménez Fraud, como se explica a continuación.

Coche. Para el modelo reducido, como se aprecia en la Figura 23 se devuelve al coche de Jiménez Fraud a la simplicidad del primer modelo. Simplemente, y como ya hemos explicado más en detalle, se acerca al semáforo y, o bien espera que cambie a verde y cruza, o bien cruza directamente si está en verde. Después, se vuelve a aproximar al semáforo.

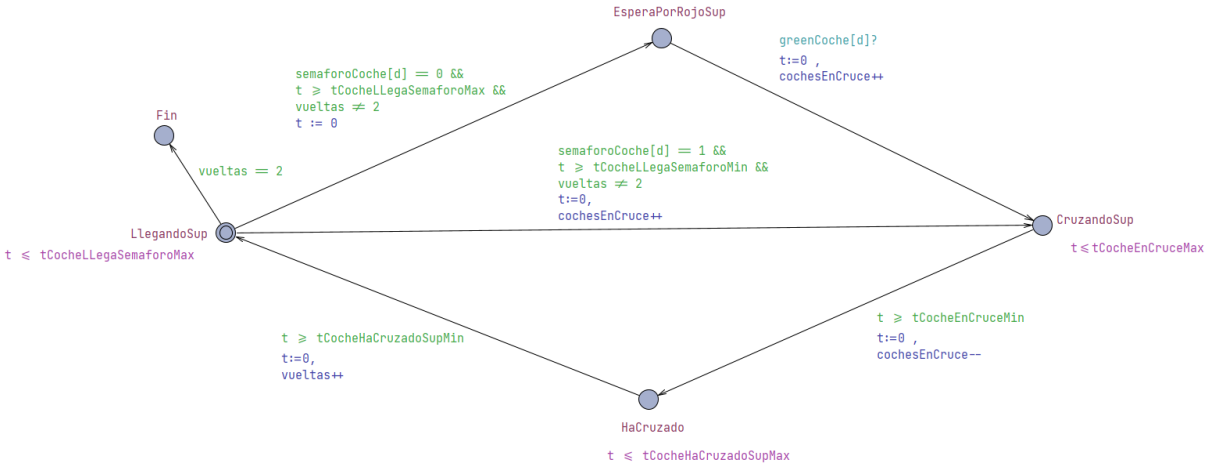

Figura 23: Autómata de coche en el modelo 2 reducido en UPPAAL.

Aunque en este caso, y aquí se encuentra la única diferencia con el autómata del primer modelo, no realiza un bucle infinito. Simplemente, con la misión de reducir el número de estados, se limita a realizar dos vueltas completas. Esto se realiza añadiendo una variable *vueltas* en las declaraciones del propio autómata. Esta variable se inicializa a 0, y cada vez que el coche vuelve a aproximarse al semáforo, al pasar de *HaCruzado* a *LlegandoSup*, se suma una unidad al valor que tenga. Finalmente, cuando llegue al valor 2, le será imposible pasar a los estados *EsperaPorRojoSup* o *CruzandoSup*, y simplemente podrá realizar la transición hacia el estado *Fin* y terminar su ejecución.

Esta simplificación no supone ningún problema, debido a que las propiedades entre él y el metro están probadas ya en el modelo anterior; y las propiedades entre él y los coches inferiores se verificarán a través de sus semáforos, como se verá en la sección de verificación.

5.3. Resultados de las simulaciones

Las simulaciones realizadas con el segundo modelo muestran que el sistema reproduce adecuadamente la dinámica del cruce ampliado. En la versión completa, los semáforos de la parte inferior de Jiménez Fraud y del Boulevard Louis Pasteur responden de manera coherente al paso del metro, o sea, cuando este se aproxima, los vehículos de Jiménez Fraud se detienen en rojo y se habilita el paso para los que circulan por el Boulevard, manteniendo así la lógica de prioridad ya establecida en el primer modelo. El comportamiento observado confirma que tanto los coches como el metro cumplen con las reglas definidas. Por ejemplo, en la Figura 24, se aprecia cómo al aproximarse el *Metro2*, este cambia los semáforos de Jiménez Fraud (superiores e inferiores) a rojo, y el *SemaforoCocheInferiorPasteur* a verde. Permitiendo a su vez que el *CocheInferior*, que espera en el Boulevard Louis Pasetur, pase de *EsperaPorRojo* a *Cruzando*.

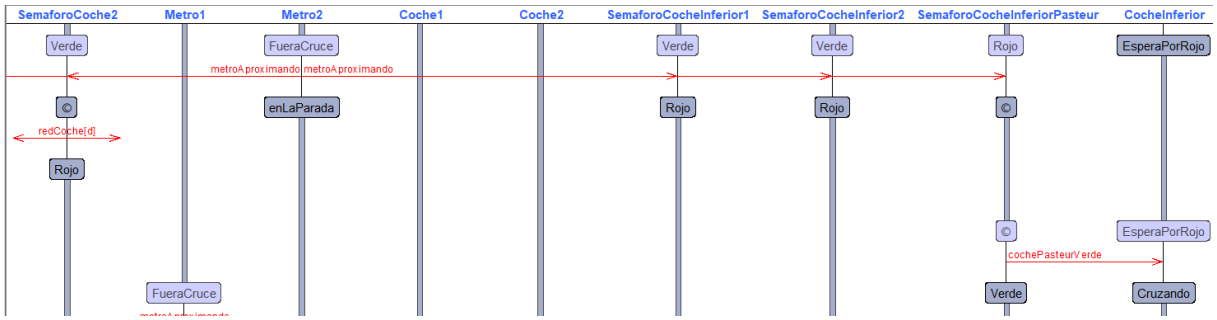

Figura 24: Traza del modelo 2 completo en el simulador simbólico de UPPAAL.

En la versión reducida, donde los semáforos inferiores de Jiménez Fraud se sustituyen por un tramo marcado con cebreado amarillo, el sistema conserva la coherencia en el flujo de vehículos. Aunque la representación es más simple, las trazas obtenidas evidencian que la alternancia entre el flujo de tráfico de Jiménez Fraud (*Coche1* y *Coche2*) y el tráfico del Boulevard Louis Pasteur (*CocheInferior*) junto a ambos metros se mantiene sin inconsistencias; como se puede observar en la Figura 25.

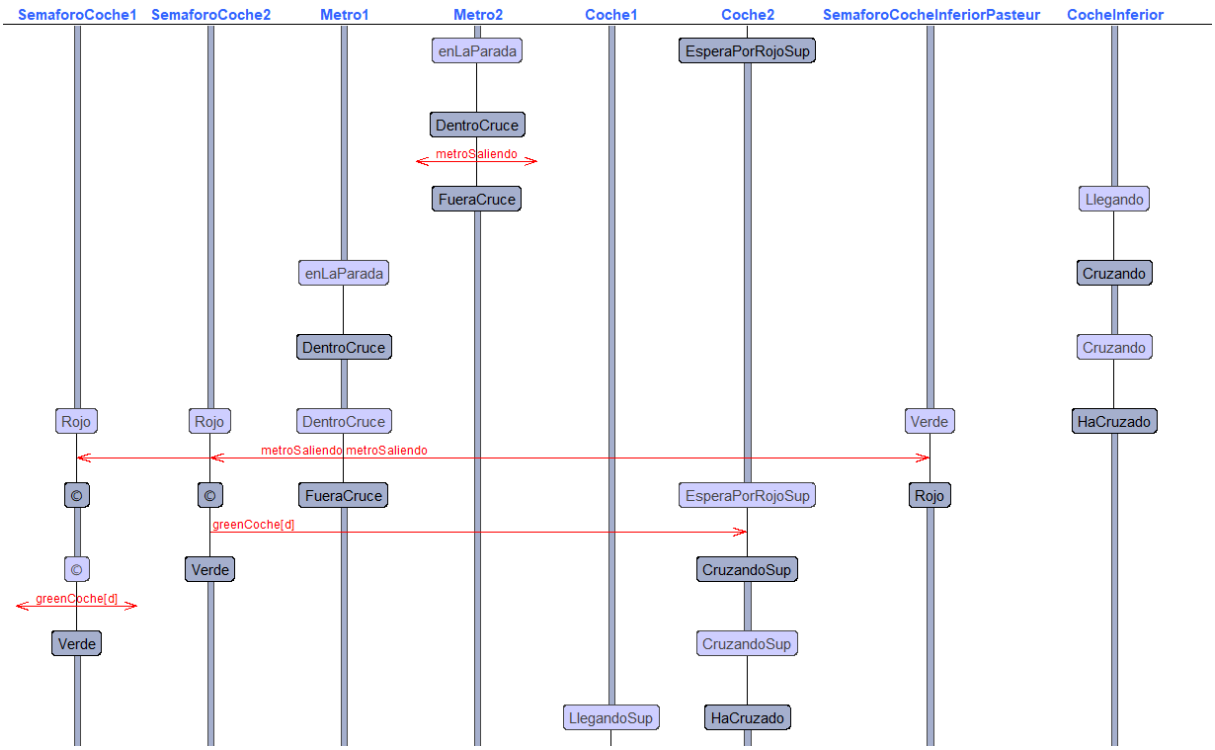

Figura 25: Traza del modelo 2 reducido en el simulador simbólico de UPPAAL.

En ambas versiones, las simulaciones reflejan un comportamiento cíclico estable que permite comprobar repetidamente las interacciones relevantes, validando así la corrección del diseño antes de proceder a la verificación formal de propiedades.

5.4. Verificación de las propiedades

El análisis realizado con el verificador de UPPAAL permitió confirmar que el segundo modelo satisface un conjunto de propiedades esenciales relacionadas con la seguridad y la viveza de los actores. En primer lugar, se comprobó que el coche inferior siempre dispone de la posibilidad de cruzar, tanto al aproximarse al semáforo como cuando se encuentra en espera por rojo. Esto garantiza la viveza del sistema y asegura que ningún vehículo queda permanentemente bloqueado.

En cuanto a la sincronización de los semáforos, las propiedades verificadas confirman que no existen configuraciones contradictorias entre los situados en la calle Jiménez Fraud y el del Boulevard Louis Pasteur. El sistema impide que ambos estén simultáneamente en verde o en

rojo, manteniendo así la coherencia en la regulación del tráfico. Cabe señalar que, debido a la presencia de estados *commit*, no siempre se encuentran en estados estrictamente opuestos (es decir, uno en rojo y otro en verde), pero en ningún caso se permite que ambos presenten la misma señal. Estas comprobaciones, además de asegurar la correcta coordinación, constituyen también un requisito de seguridad, ya que evitan situaciones de riesgo en la intersección.

Finalmente, el análisis estocástico permitió evaluar aspectos de rendimiento y eficiencia. Los resultados mostraron que la probabilidad de que el coche inferior alcance el estado de cruce en menos de veinte unidades de tiempo es elevada, superior al 95 %. Asimismo, la probabilidad de que permanezca en espera durante al menos diez unidades de tiempo dentro de las primeras 150 se mantiene baja, alrededor del 5 %. Estos resultados reflejan que el sistema favorece la fluidez del tráfico al mismo tiempo que preserva condiciones seguras de circulación. La verificación de las propiedades anteriormente mencionadas aparece en la Figura 26. Por su parte, la especificación de estas se puede consultar en la Tabla 2 del Apéndice A.

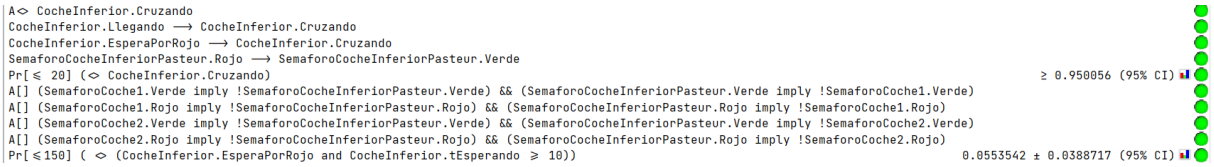

Figura 26: Propiedades verificadas del modelo 2 en UPPAAL.

6

Tercer modelo: Cruce con ambulancia

Para el tercer modelo, se añade en consideración la incorporación de vehículos prioritarios, como son las ambulancias, a nuestro sistema. Estas resultan más frecuentes en este entorno concreto que en otros escenarios habituales de tráfico, debido a la proximidad del Hospital Virgen de la Victoria. Se puede observar la posición exacta del hospital en la parte superior de la Figura 27. En la parte inferior aparece nuestro dominio del cruce.

Este tercer modelo toma como punto de partida la versión completa del segundo modelo, incorporando ahora, como se ha mencionado, la presencia de ambulancias, las cuales deben tener prioridad en el paso en condiciones de emergencia. Esta ampliación permite analizar cómo interactúa un actor con prioridad absoluta en un entorno ya de por sí complejo, donde confluyen los flujos del metro y de los coches regulados por semáforos. En la versión completa, el sistema mantiene toda la infraestructura de semáforos descrita en el modelo anterior y añade además la lógica necesaria para garantizar que, cuando una ambulancia se aproxima al cruce, pueda acceder con seguridad sin riesgo de colisión con el metro.

El incremento de componentes y restricciones asociado a este modelo conllevó, como en el modelo anterior, un crecimiento considerable del espacio de estados en la verificación, lo que dificultaba comprobar las propiedades de manera eficiente. Para abordar esta limitación también se elaboró una versión reducida del modelo, en la que la ambulancia se introdujo sobre la base del primer modelo. De esta forma, se eliminó la complejidad adicional derivada de los semáforos inferiores de Jiménez Fraud y el análisis se centró en la interacción que se considera más esencial, es decir, la prioridad de paso de la ambulancia frente al metro en superficie.

Esta versión reducida mantiene los actores concretos del primer modelo, es decir, dos co-

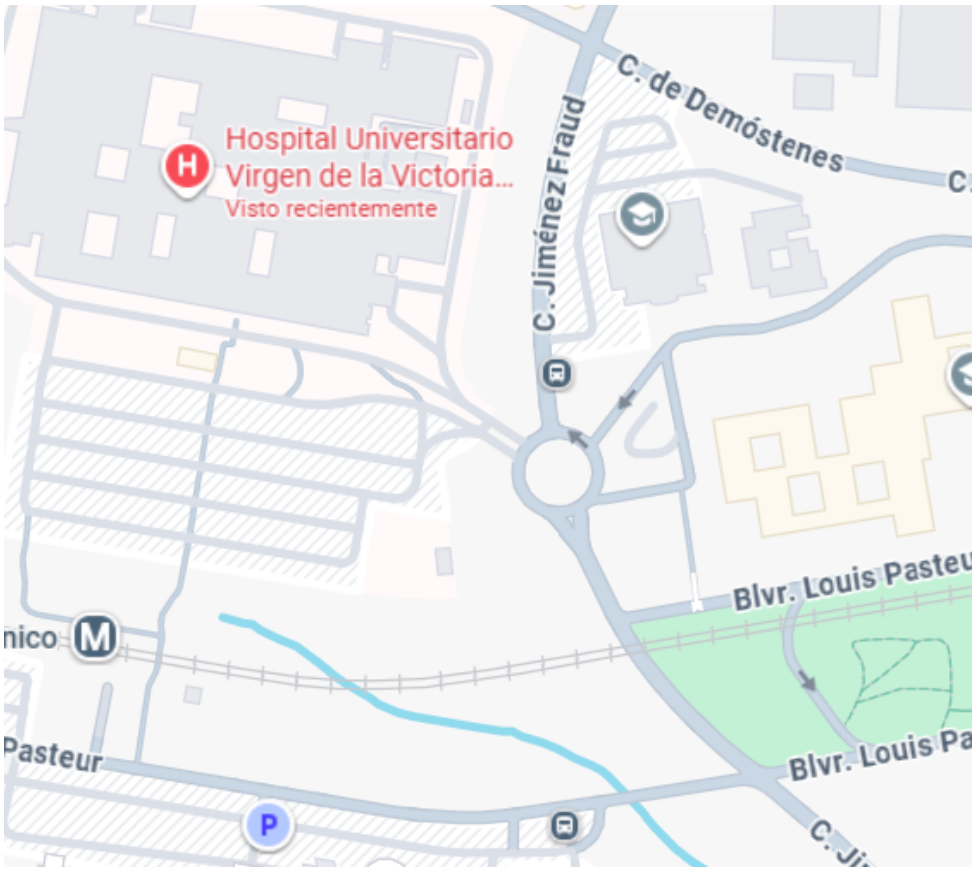

Figura 27: Imagen del cruce del tercer modelo en Google Maps.

ches, dos semáforos y dos metros, añadiendo además, una instancia de ambulancia. En resumen, esta reducción del modelo permitió estudiar de manera más clara y directa el comportamiento de la ambulancia en situaciones críticas, evaluando con precisión las propiedades de seguridad y viveza relacionadas con su circulación.

6.1. Consideraciones y problemas de diseño

Cuando se habla de ambulancia como un nuevo actor, está representando exclusivamente la situación en la que circula con las luces de emergencia encendidas. Una ambulancia sin prioridad se comportaría de forma equivalente a un coche normal, por lo que no se consideró necesario realizar su modelo de forma independiente. Se asumió, además, que las ambulancias circulan únicamente por la calle Jiménez Fraud y no por el Boulevard Louis Pasteur.

En cuanto a los tiempos de este nuevo actor, se ajustaron para reflejar tanto su baja frecuencia como su mayor velocidad de circulación. Estos parámetros y la nueva variable global

incorporada *ambulanciaEnCruce* pueden observarse en la Figura 28.

```
// ITERACIÓN 3:

// Parámetros de tiempo de Ambulancia
const int tAmbLLegaSemaforoInfMax = 20;// Tiempos bajos, ya que va deprisa
const int tAmbLLegaSemaforoInfMin = 5;
const int tAmbEnCruceInfMax = 8;
const int tAmbEnCruceInfMin = 3;
const int tAmbHaCruzadoInfMax = 10;
const int tAmbHaCruzadoInfMin = 5;
const int tAmbLLegaSemaforoSupMax = 10;
const int tAmbLLegaSemaforoSupMin = 3;
const int tAmbEnCruceSupMax = 8;
const int tAmbEnCruceSupMin = 3;
const int tAmbHaCruzadoSupMax = 500;// Estos tiempos altos, ya que las ambulancias no son tan comunes como los coches
const int tAmbHaCruzadoSupMin = 200;

int ambulanciaEnCruce = 0;
```

Figura 28: Parámetros de tiempo, variables y canales del modelo 3 completo en UPPAAL.

El desarrollo del modelo presentó algunas dificultades. Inicialmente, se asumió que la ambulancia, por su condición de urgencia, podía cruzar sin ninguna restricción, lo que generaba situaciones inseguras en las que coincidía con un metro en el cruce. Este problema se resolvió introduciendo un estado intermedio en el que la ambulancia espera a que el metro finalice su trayecto. Otro obstáculo ya mencionado fue la complejidad del modelo completo, que hacía inviables las verificaciones en tiempos razonables. Por ello, al igual que en el modelo anterior, se construyó una versión reducida.

6.2. Autómatas principales

En esta sección, se hace una distinción entre los autómatas referentes al modelo completo y los referentes al modelo reducido.

6.2.1. Modelo completo

Para el modelo completo de la tercera iteración, se añade un solo autómata respecto al segundo modelo completo, el referente a la ambulancia. Además, se modifica el autómata de metro para que tenga en cuenta a este último.

Ambulancia. De forma general, la ambulancia se comporta de manera similar a como lo haría un coche en el modelo 2 completo. Sin embargo, las diferencias que existen son clave para diferenciar ambos autómatas. Como se puede observar en la Figura 29 que muestra sus estados y transiciones, la ambulancia comienza en el estado *LlegandoInf*, aproximándose al semáforo inferior de Jiménez Fraud. Atraviesa este cruce inferior con prioridad frente a los

coches, y sin consultar el estado del semáforo que regula esta intersección, simplemente con las guardas e invariantes de tiempo como limitación. De esta manera, llega al estado *CruzandoInf* y posteriormente a *HaCruzadoInf*.

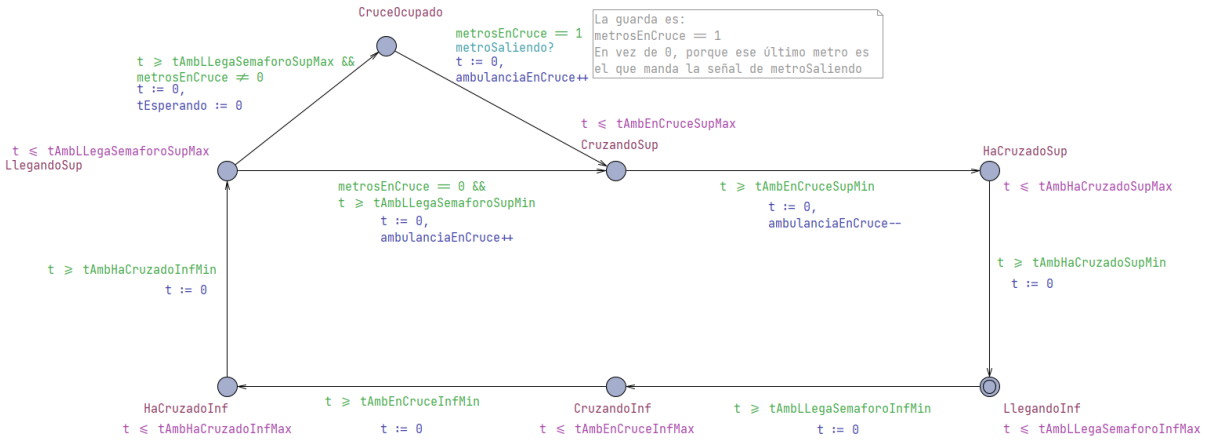

Figura 29: Autómata de ambulancia en el modelo 3 completo en UPPAAL.

Después, al aproximarse al cruce superior, encontrándose en el estado *LlegandoSup*, sí realiza una comprobación. Además de las guardas de tiempo, comprueba si hay o no algún metro en el cruce. En el caso de que no se encuentre ninguno y haya transcurrido el tiempo *tAmbLlegaSemaforoSupMin*, realiza la transición hacia *CruzandoSup*, actualizando la variable *ambulanciaEnCruce*. Sin embargo, si agota el tiempo *tAmbLlegaSemaforoSupMax* y el cruce sigue ocupado por algún metro, pasa al estado *CruceOcupado*. En este, realiza una sincronización y comprueba una guarda similar a la de los semáforos, aunque no igual. Nuestra ambulancia espera a que el último metro del cruce envíe a través del canal *metroSaliendo* que ha terminado de cruzar. En ese instante, pasa al estado *CruzandoSup*, actualizando también la variable *ambulanciaEnCruce*.

Se puede ver entonces que, aunque pudiera parecer que, cuando una ambulancia espera a que un metro desaloje el cruce, equivale a esperar un semáforo en rojo, la lógica es distinta. Mientras los semáforos solo permiten el paso cuando no hay metros ni en el cruce ni en la parada, la ambulancia puede atravesar tan pronto como detecta que el cruce está libre, incluso si un metro permanece detenido en la parada. Esto refleja una situación más realista, dotando de mayor prioridad a la ambulancia.

Finalmente, pasará del estado *CruzandoSup* a *HaCruzadoSup*, y vuelta al estado inicial. Tras completar una vuelta de este bucle, la ambulancia volverá a realizarlo, como ya hemos matizado, con algo más de tiempo de espera entre una vuelta y otra, representando así su carácter menos común que el de los coches.

Metro. En cuanto al autómata de Metro (Figura 30), se puede apreciar que es prácticamente igual al de los modelos anteriores. La única diferencia reside en el hecho de que ahora, al realizar transiciones entre los estados *enLaParada*, *Alarma* y *DentroCruce*, también tiene en cuenta el hecho de que puede haber una ambulancia en el cruce, no solo coches.

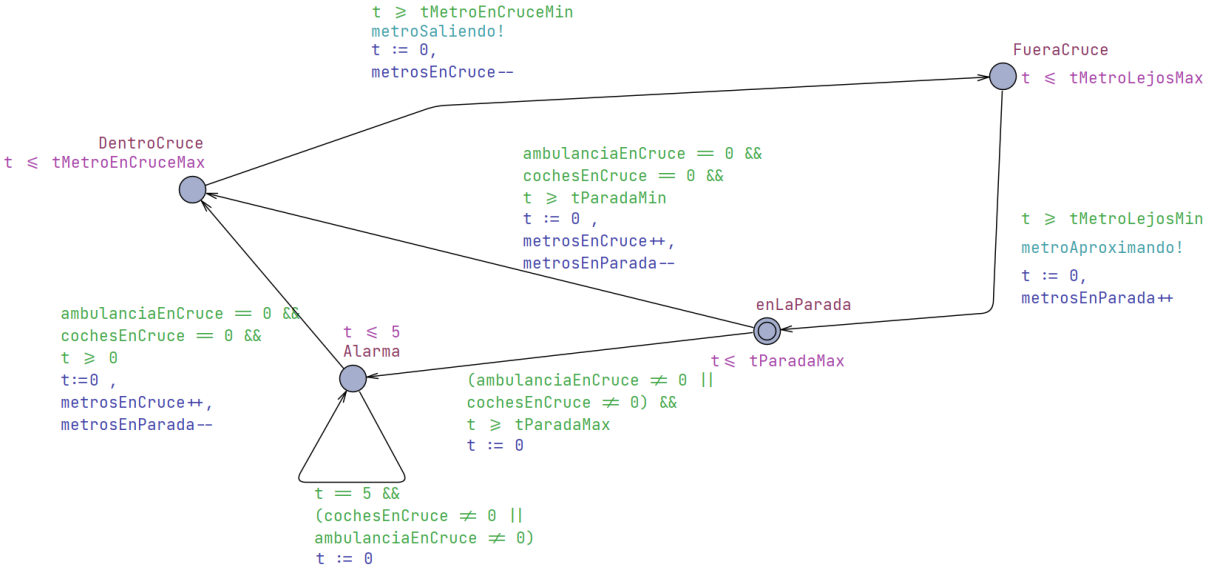

Figura 30: Autómata de metro en el modelo 3 completo en UPPAAL.

6.2.2. Modelo reducido

Para la versión reducida, se parte con el primer modelo como base. De este, los autómatas de coche y de semáforo de coche quedan exactamente iguales. El de metro se modifica levemente para que quede como en la versión completa de este tercer modelo (Figura 30), que ya se ha explicado. Y, obviamente, sobre esta base se añade el autómata de la ambulancia.

Ambulancia. En esta versión reducida, la ambulancia se comporta de manera similar a la versión completa, quitando eso sí, los estados y transiciones del cruce inferior (el cual entendemos como un cebreado amarillo). Así, también atraviesan el cruce superior de Jiménez Fraud

sin tener en cuenta los semáforos, simplemente realizando las comprobaciones y transiciones necesarias dependiendo de si hay algún metro que se encuentre en el cruce o no.

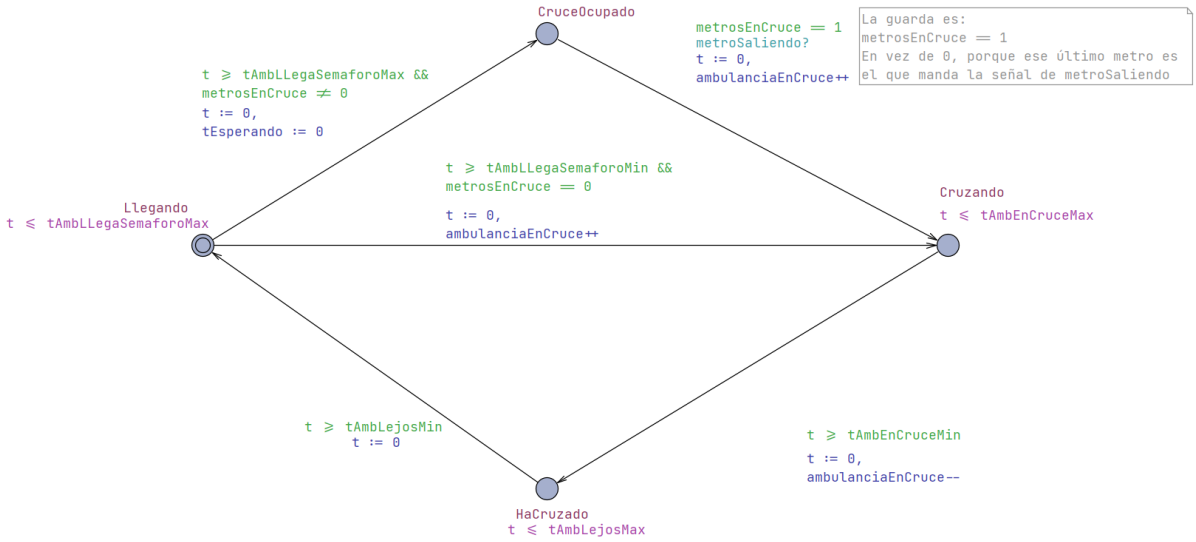

Figura 31: Autómata de ambulancia en el modelo 3 reducido en UPPAAL.

6.3. Resultados de las simulaciones

Las simulaciones del tercer modelo mostraron un comportamiento coherente en ambos casos, tanto en la versión completa como en la reducida. En los dos, la ambulancia atraviesa el cruce con prioridad respecto a los coches, deteniéndose únicamente cuando detecta la presencia de un metro en plena intersección. Una vez que el metro ha salido, la ambulancia puede reanudar la marcha de inmediato, incluso aunque otro metro permanezca detenido en la parada.

En la Figura 32 se puede apreciar una traza del modelo completo, en la que al intentar cruzar, la *Ambulancia1* encuentra el cruce ocupado. Viéndose obligada a pasar al estado *CruceOcupado*, del que posteriormente sale en cuanto el cruce se desaloja, sin esperar siquiera a que el semáforo cambie a verde.

Por otra parte, en la Figura 33, se observa cómo funcionaría, en este caso en el modelo reducido, una traza diferente. En concreto, una en la que la *Ambulancia1* cruza aunque los semáforos estén en rojo, ya que no detecta ningún metro en el cruce. Al hacerlo, provoca que el *Metro2* pase momentáneamente al estado de *Alarma*, ya que intenta cruzar y se encuentra

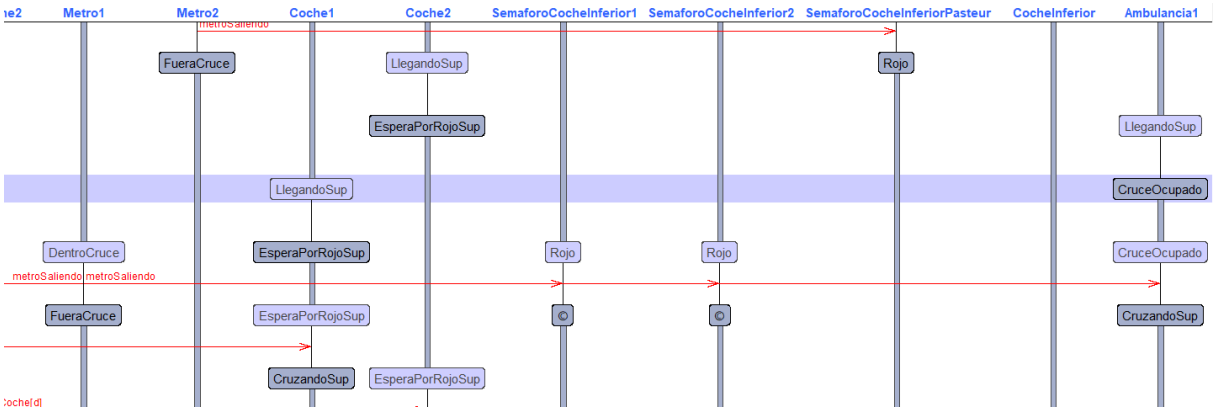

Figura 32: Traza del modelo 3 completo en el simulador simbólico de UPPAAL.

con un vehículo (*Ambulancia1*) en el cruce. Así, se produce el cruce de una ambulancia, con prioridad y sin dejar de lado la seguridad.

En la versión completa, este comportamiento se refleja con mayor detalle gracias a la inclusión de todos los semáforos y actores, lo que permite observar la coordinación entre múltiples componentes en escenarios complejos. No obstante, el elevado número de estados dificulta la verificación formal. En conjunto, las trazas generadas evidencian que el sistema, en ambos modelos, mantiene la fluidez del tráfico y garantiza la prioridad de paso de la ambulancia, respetando al mismo tiempo las condiciones de seguridad necesarias en la interacción con los metros.

6.4. Verificación de las propiedades

La verificación formal del tercer modelo permitió confirmar tanto la corrección como la coherencia de las interacciones introducidas con la incorporación de la ambulancia. En primer lugar, se comprobó que el sistema preserva la seguridad en el cruce, es decir, nunca se produce una situación en la que una ambulancia y un metro coincidan dentro de la intersección, lo que garantiza que la prioridad no compromete la integridad del tráfico.

Las propiedades de viveza verificadas muestran que la ambulancia siempre consigue cruzar, ya sea desde el estado de llegada o desde el estado de espera en el cruce ocupado, evitando bloqueos permanentes. De manera complementaria, el análisis estocástico evidencia que la probabilidad de que la ambulancia cruce en un tiempo reducido es elevada, lo que refleja una

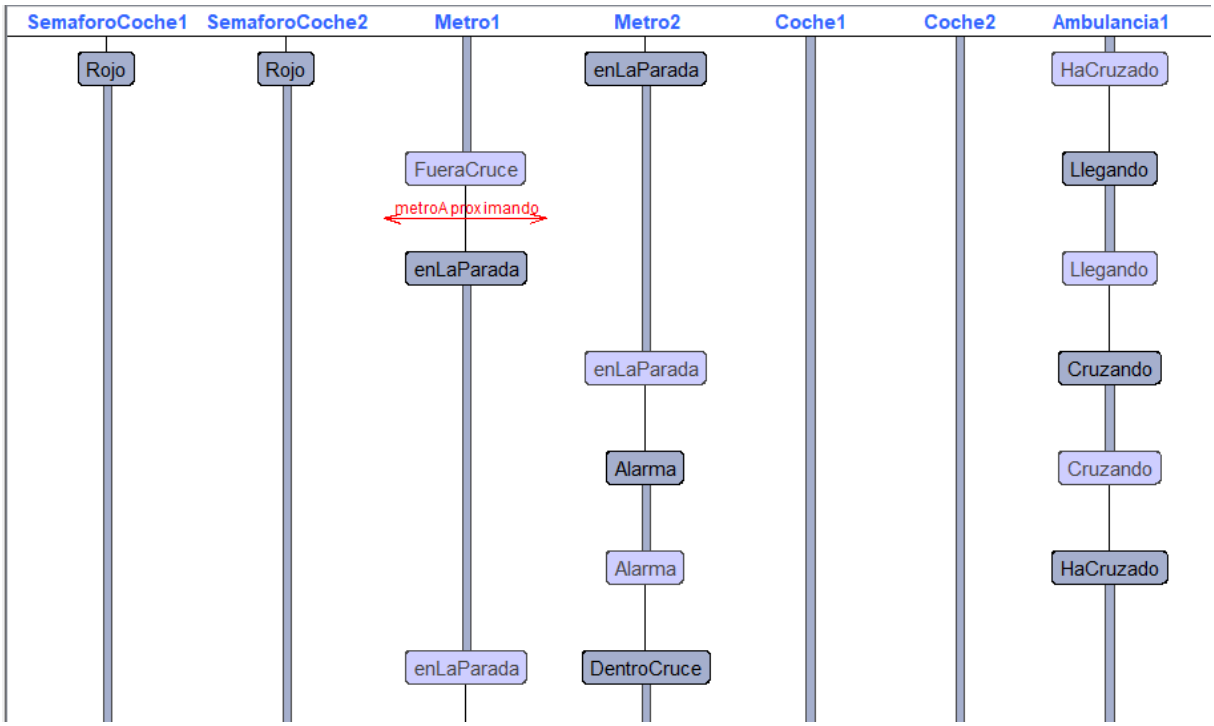

Figura 33: Traza del modelo 3 reducido en el simulador simbólico de UPPAAL.

respuesta rápida y coherente con el carácter prioritario de este actor.

Asimismo, se comprobó que la ambulancia puede llegar a encontrarse en situaciones de concurrencia con otros vehículos, como coches o semáforos en rojo, o incluso con un metro en estado de alarma. Estas trazas, aunque posibles, no vulneran la seguridad del sistema, sino que reflejan escenarios realistas en los que la ambulancia mantiene su prioridad. Finalmente, las propiedades probabilísticas aplicadas al estado de espera en el cruce ocupado muestran que este tiempo se mantiene reducido en la gran mayoría de los casos, confirmando que el modelo proporciona un paso ágil y seguro a este actor esencial. Podemos consultar la verificación de estas propiedades en la Figura 34. Para su especificación, ver la Tabla 3 del Apéndice A.

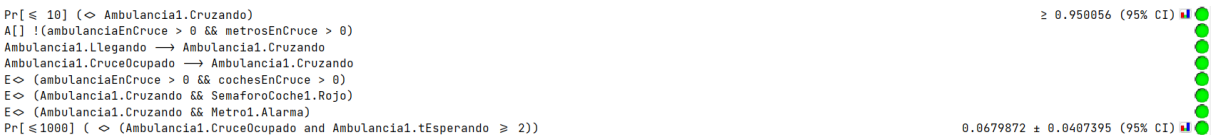

Figura 34: Propiedades verificadas del modelo 3 en UPPAAL.

7

Conclusiones y Líneas Futuras

7.1. Conclusiones

El presente Trabajo de Fin de Grado ha permitido aplicar métodos formales al análisis de un sistema de semáforos inteligentes en un entorno real, concretamente en el cruce entre el Boulevard Louis Pasteur y la calle Jiménez Fraud, en el campus de Teatinos de la Universidad de Málaga. A través de la herramienta UPPAAL se han desarrollado y verificado distintos modelos de creciente complejidad, lo que ha posibilitado evaluar de manera rigurosa propiedades críticas de seguridad, viveza y rendimiento.

El primer modelo sirvió como punto de partida para comprobar la viabilidad de representar el sistema mediante autómatas temporizados, logrando reproducir la interacción básica entre coches, semáforos y metro. Posteriormente, en el segundo modelo se amplió el alcance al incluir el cruce inferior, lo que aumentó notablemente la complejidad y obligó a recurrir a una versión reducida para que las verificaciones fueran computacionalmente abordables. Finalmente, en el tercer modelo se incorporó la ambulancia como actor prioritario, destacando la importancia de coordinar su comportamiento con el del metro y de garantizar tiempos de espera reducidos en situaciones de emergencia.

En conjunto, los resultados obtenidos confirman que los modelos desarrollados cumplen los requisitos fundamentales planteados. Los coches y el metro pueden circular de forma segura y sin bloqueos, los semáforos mantienen configuraciones coherentes y la ambulancia atraviesa la intersección con prioridad sin comprometer la seguridad del sistema. Además, las propiedades estocásticas verificadas evidencian un nivel aceptable de rendimiento en los tiempos de espera, lo que refuerza la aplicabilidad del enfoque.

Más allá de los resultados específicos, el trabajo demuestra el valor de los métodos formales para el diseño de sistemas críticos en el ámbito del tráfico urbano. La construcción iterativa y el uso de modelos reducidos han permitido gestionar la complejidad inherente al problema, a la vez que se han obtenido garantías sólidas de corrección. Este enfoque constituye, por tanto, una base sólida para futuros desarrollos orientados a la optimización y control inteligente de intersecciones urbanas.

7.2. Líneas Futuras

A partir del trabajo desarrollado, se identifican diversas líneas de investigación y mejora que podrían abordarse en el futuro para ampliar el alcance y la utilidad del modelo:

- **Optimización mediante UPPAAL Stratego:** Aunque en este trabajo se ha empleado UPPAAL en su versión clásica y SMC, una línea de desarrollo natural es el uso de UPPAAL Stratego [17]. Esta extensión permite no solo verificar propiedades, sino también sintetizar estrategias óptimas para el control de los semáforos en función de diferentes objetivos, como minimizar los tiempos de espera o priorizar el paso de ambulancias.

- **Ampliación del entorno de simulación:** El modelo actual se centra en un cruce concreto, pero podría extenderse a una red mayor de intersecciones, incorporando nuevos actores como peatones o bicicletas. Esto permitiría evaluar la escalabilidad del enfoque y analizar cómo se comportan las estrategias de coordinación en un contexto urbano más realista.

- **Uso de recursos computacionales más potentes para verificación completa:** Otra línea de investigación consistiría en replicar el sistema desarrollado, pero aprovechando ordenadores con capacidades de cálculo significativamente superiores. Esto permitiría analizar el modelo completo sin necesidad de recurrir a versiones reducidas, lo que facilitaría la verificación exacta de todas las propiedades de seguridad, viveza y rendimiento para todos los actores simultáneamente.

- **Integración de datos reales:** Otra línea prometedora sería la incorporación de parámetros obtenidos de mediciones reales de tráfico en el cruce, tales como tiempos de espera,

frecuencias de paso o patrones de congestión. Con ello se lograría un modelo más fiel a la realidad, facilitando su transferencia a un posible sistema de gestión en producción.

- **Exploración de otras herramientas de verificación:** Aunque UPPAAL ha demostrado ser eficaz para el análisis de sistemas de tiempo real, sería de interés contrastar los resultados con otras herramientas de modelado o de verificación formal. Esto permitiría validar la robustez del enfoque y, al mismo tiempo, evaluar posibles ventajas de otras plataformas en términos de rendimiento o expresividad.

Referencias

[1] G. Behrmann, A. David y K. G. Larsen. "UPPAAL: Model Checking Tool." Herramienta de modelado, simulación y verificación de sistemas de tiempo real. (1995), dirección: http://www.uppaal.org.

[2] L. Torvalds et al. "Git: Distributed Version Control System." Sistema de control de versiones distribuido, Software Freedom Conservancy. (2005), dirección: https://git-scm.com.

[3] GitHub, Inc. "GitHub: Collaborative Development Platform." Plataforma de alojamiento y gestión de repositorios. (2008), dirección: https://github.com.

[4] Microsoft Corporation. "Visual Studio Code." Editor de código multiplataforma, Microsoft. (2015), dirección: https://code.visualstudio.com.

[5] L. Lamport. "LaTeX - A Document Preparation System." Lenguaje de tipografía científica, Addison-Wesley. (1984), dirección: https://www.latex-project.org.

[6] Overleaf, Ltd. "Overleaf: Collaborative LaTeX Editing Platform." Entorno colaborativo en la nube para documentos LaTeX. (2011), dirección: https://www.overleaf.com.

[7] Canva Pty Ltd. "Canva: Graphic Design Platform." Herramienta para crear esquemas y diagramas. (2013), dirección: https://www.canva.com.

[8] M. S. Gharajeh, "Waterative model: An integration of the waterfall and iterative software development paradigms," *Database Syst. J*, vol. 10, págs. 75-81, 2019.

[9] A. David, K. G. Larsen, A. Legay, M. Mikucionis y D. B. Poulsen, *Uppaal SMC Tutorial*, Department of Computer Science, Aalborg University and INRIA/IRISA Rennes, Technical tutorial paper, ene. de 2018. dirección: https://uppaal.org/texts/uppaal-smc-tutorial.pdf.

[10] M. Wiering, J. v. Veenen, J. Vreeken, A. Koopman et al., "Intelligent traffic light control," 2004.

[11] A. Joshi, S. P. Miller y M. P. Heimdahl, "Mode confusion analysis of a flight guidance system using formal methods," en *Digital Avionics Systems Conference, 2003. DASC'03. The 22nd*, IEEE, vol. 1, 2003, págs. 2-D.

[12] J. H. Kim, K. G. Larsen, B. Nielsen, M. Mikučionis y P. Olsen, "Formal analysis and testing of real-time automotive systems using UPPAAL tools," en *International Workshop on Formal Methods for Industrial Critical Systems*, Springer, 2015, págs. 47-61.

[13] P. L. Laursen, V. A. T. Trinh y A. E. Haxthausen, "Formal modelling and verification of a distributed railway interlocking system using UPPAAL," en *International Symposium on Leveraging Applications of Formal Methods*, Springer, 2020, págs. 415-433.

[14] J. E. Hopcroft, R. Motwani y J. D. Ullman, "Introduction to automata theory, languages, and computation," *Acm Sigact News*, vol. 32, n.º 1, págs. 60-65, 2001.

[15] R. Alur y D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, n.º 2, págs. 183-235, 1994.

[16] A. Eriksen, C. Huang, J. Kildebogaard et al., "Uppaal Stratego for Intelligent Traffic Lights," en *12th ITS European Congress*, jul. de 2017.

[17] Aalborg University. "UPPAAL Stratego: Synthesis and Optimization for Stochastic Timed Systems." Extensión de UPPAAL para la síntesis de estrategias y optimización en sistemas temporizados estocásticos. (2015), dirección: https://uppaal.org/casestudies/stratego/.

[18] A. Kamput y C. Dechsupa, "Formal modelling and verification of the traffic light control system design with time-automata," en *Proceedings of the International MultiConference of Engineers and Computer Scientists. Lecture Notes in Engineering and Computer Science. IMEC*, 2023.

[19] D. Sur, *Colisión entre un coche y el metro de Málaga en el bulevar Louis Pasteur*, Accedido: agosto de 2025, 2017. dirección: https://www.diariosur.es/malaga-capital/nuevo-accidente-coche-20171120155521-nt.html.

[20] M. Hoy, *Un coche colisiona con el Metro de Málaga en el tercer accidente en dos meses*, Accedido: agosto de 2025, 2015. dirección: https://www.malagahoy.es/malaga/colisiona-Metro-tercer-accidente-meses_0_910409465.html.

Apéndice A
Especificación de las propiedades

A.1. Modelo 1

Propiedad (TCTL)	**Descripción en lenguaje natural**	**Verificación**
$A[]\ notdeadlock$	El sistema nunca entra en *deadlock*, es decir, nunca se bloquea.	Cumplida
$A <>\ Coche1.Cruzando$	El *Coche1* puede cruzar eventualmente en todas las trazas.	Cumplida
$A <>\ Coche2.Cruzando$	El *Coche2* puede cruzar eventualmente en todas las trazas.	Cumplida
$A <>\ Metro1.DentroCruce$	El *Metro1* puede cruzar eventualmente en todas las trazas.	Cumplida
$A <>\ Metro2.DentroCruce$	El *Metro2* puede cruzar eventualmente en todas las trazas.	Cumplida
$E <>\ Metro1.Alarma$	Existe alguna traza en la que, eventualmente, el *Metro1* entra en estado de alarma.	Cumplida
$E <>\ Metro2.Alarma$	Existe alguna traza en la que, eventualmente, el *Metro2* entra en estado de alarma.	Cumplida

$Metro1.Alarma \ -- > \ Metro1.DentroCruce$	Cuando el *Metro1* está en estado de alarma, después en algún momento, cruza.	Cumplida
$Metro2.Alarma \ -- > \ Metro2.DentroCruce$	Cuando el *Metro2* está en estado de alarma, después en algún momento, cruza.	Cumplida
$A <> \ SemaforoCoche1.Verde \ and$ $SemaforoCoche2.Verde$	Ambos semáforos pueden estar en verde al mismo tiempo.	Cumplida
$A[]$ $(SemaforoCoche1.Verde \ imply$ $!SemaforoCoche2.Rojo)$ && $(SemaforoCoche2.Verde \ imply$ $!SemaforoCoche1.Rojo)$	Ambos semáforos están sincronizados, es decir, no puede estar uno en verde y otro en rojo. A veces no están ambos verdes o rojos a la vez debido a que existen estados *commit*.	Cumplida
$Coche1.Llegando \ -- > \ Coche1.Cruzando$	Cuando el *Coche1* está llegando, después en algún momento, cruza.	Cumplida
$Coche1.EsperaPorRojo \ -- >$ $Coche1.Cruzando$	Cuando el *Coche1* está en espera por semáforo en rojo, después en algún momento, cruza.	Cumplida
$Metro1.enLaParada \ -- >$ $Metro1.DentroCruce$	Cuando el *Metro1* está en la parada, después en algún momento, cruza.	Cumplida
$Metro1.Alarma \ -- > \ Metro1.DentroCruce$	Cuando el *Metro1* está en estado de alarma, después en algún momento, cruza.	Cumplida
$SemaforoCoche1.Rojo \ -- >$ $SemaforoCoche1.Verde$	Cuando el *SemaforoCoche1* está rojo, después en algún momento, cambia a verde.	Cumplida

$A[]\ !(cochesEnCruce > 0 \ \&\& \ metrosEnCruce > 0)$	En ningún momento hay algún coche y metro en el cruce al mismo tiempo.	Cumplida
$E <> Coche1.Cruzando \ \&\& \ Coche2.Cruzando$	Existe alguna traza con dos coches en el cruce a la vez, es posible.	Cumplida
$E <> Metro1.DentroCruce \ \&\& \ Metro2.DentroCruce$	Existe alguna traza con dos metros en el cruce a la vez, es posible.	Cumplida
$Pr[<= 90] (<> Coche1.Cruzando)$	Probabilidad de que el *Coche1* cruce en menos de 30 unidades de tiempo.	$\geq 95\,\%$
$Pr[<= 30] (<> Coche2.Cruzando)$	Probabilidad de que el *Coche2* cruce en menos de 30 unidades de tiempo.	$\geq 95\,\%$
$Pr[<= 1000] (<> Metro1.Alarma)$	Probabilidad de que el *Metro1* entre en estado de alarma en 1000 unidades de tiempo.	$\leq 5\,\%$
$Pr[<= 1000] (<> Metro2.Alarma)$	Probabilidad de que el *Metro2* entre en estado de alarma en 1000 unidades de tiempo.	$\leq 5\,\%$

Cuadro 1: Propiedades verificadas, su descripción y resultados en el modelo 1.

A.2. Modelo 2

Propiedad (TCTL)	**Descripción en lenguaje natural**	**Verificación**
$A <> \ CocheInferior.Cruzando$	El *CocheInferior* puede cruzar eventualmente en todas las trazas.	Cumplida
$CocheInferior.Llegando \ - - > \ CocheInferior.Cruzando$	Cuando el *CocheInferior* está llegando, después en algún momento, cruza.	Cumplida
$CocheInferior.EsperaPorRojo \ - - > \ CocheInferior.Cruzando$	Cuando el *CocheInferior* está en espera por semáforo en rojo, después en algún momento, cruza.	Cumplida
$SemaforoCocheInferiorPasteur.Rojo \ - - > \ SemaforoCocheInferiorPasteur.Verde$	Cuando el *SemaforoCocheInferiorPasteur* está rojo, después en algún momento, cambia a verde.	Cumplida
$Pr[<= 20] \ (<> \ CocheInferior.Cruzando)$	Probabilidad de que el *CocheInferior* cruce en 20 unidades de tiempo o menos.	$\geq 95\,\%$
$A[](SemaforoCoche1.Verde \ imply \ !SemaforoCocheInferiorPasteur.Verde) \ \&\& \ (SemaforoCocheInferiorPasteur.Verde \ imply \ !SemaforoCoche1.Verde)$	*SemaforoCoche1* y *SemaforoCocheInferiorPasteur* están sincronizados, es decir, no pueden estar ambos en verde.	Cumplida
$A[](SemaforoCoche1.Rojo \ imply \ !SemaforoCocheInferiorPasteur.Rojo) \ \&\& \ (SemaforoCocheInferiorPasteur.Rojo \ imply \ !SemaforoCoche1.Rojo)$	*SemaforoCoche1* y *SemaforoCocheInferiorPasteur* están sincronizados, es decir, no pueden estar ambos en rojo.	Cumplida

$A[](SemaforoCoche2.Verde\ imply$ $!SemaforoCocheInferiorPasteur.Verde)$ && $(SemaforoCocheInferiorPasteur.Verde\ imply$ $!SemaforoCoche2.Verde)$	*SemaforoCoche2* y *SemaforoCocheInferiorPasteur* están sincronizados, es decir, no pueden estar ambos en verde.	Cumplida
$A[](SemaforoCoche2.Rojo\ imply$ $!SemaforoCocheInferiorPasteur.Rojo)$ && $(SemaforoCocheInferiorPasteur.Rojo\ imply$ $!SemaforoCoche2.Rojo)$	*SemaforoCoche2* y *SemaforoCocheInferiorPasteur* están sincronizados, es decir, no pueden estar ambos en rojo.	Cumplida
$Pr[<= 150]$ $(<> (CocheInferior.EsperaPorRojo\ and$ $CocheInferior.tEsperando >= 10))$	La probabilidad de que, en algún momento dentro de las primeras 150 unidades de tiempo, el *CocheInferior* espere por semáforo en rojo, habiendo acumulado al menos 10 unidades de tiempo en dicho estado.	$5\,\% \pm 3\,\%$

Cuadro 2: Propiedades verificadas, su descripción y resultados en el modelo 2 reducido.

A.3. Modelo 3

Propiedad (TCTL)	**Descripción en lenguaje natural**	**Verificación**
$Pr[<= 10]\ (<>\ Ambulancia1.Cruzando)$	Probabilidad de que el la *Ambulancia1* cruce en 10 unidades de tiempo o menos.	$\geq 95\,\%$
$A[]\ !(ambulanciaEnCruce > 0\ \ \&\&\ metrosEnCruce > 0)$	En ningún momento hay alguna ambulancia y metro en el cruce al mismo tiempo.	Cumplida
$Ambulancia1.Llegando\ -->\ Ambulancia1.Cruzando$	Cuando la *Ambulancia1* está llegando, después en algún momento, cruza.	Cumplida
$Ambulancia1.CruceOcupado\ -->\ Ambulancia1.Cruzando$	Cuando la *Ambulancia1* está esperando porque el cruce está ocupado, después en algún momento, cruza.	Cumplida
$E <>\ (ambulanciaEnCruce > 0\ \ \&\&\ cochesEnCruce > 0)$	Existe alguna traza en la que, eventualmente, una ambulancia y un coche coinciden en el cruce, es posible.	Cumplida
$E <>\ (Ambulancia1.Cruzando\ \ \&\&\ SemaforoCoche1.Rojo)$	Existe alguna traza en la que, eventualmente, la *Ambulancia1* cruza aún teniendo el semáforo en rojo, es posible.	Cumplida
$E <>\ (Ambulancia1.Cruzando\ \ \&\&\ Metro1.Alarma)$	Existe alguna traza en la que, eventualmente, la *Ambulancia1* cruza aún mientras el *Metro1* está en estado de alarma, es posible.	Cumplida

$Pr[<= 1000]$ ($<>$ ($Ambulancia1.CruceOcupado\ and$ $Ambulancia1.tEsperando >= 2$))	La probabilidad de que, en algún momento dentro de las primeras 1000 unidades de tiempo, la *Ambulancia1* esté en esperando porque el cruce está ocupado, habiendo acumulado al menos 2 unidades de tiempo en dicho estado.	$6\,\% \pm 4\,\%$

Cuadro 3: Propiedades verificadas, su descripción y resultados en el modelo 3 reducido.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA