Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

# Lógicas Temporales Lineales para la Verificación Formal de Sistemas de Tiempo Real



Tesis que presenta:
**Francisco Manzano Pinzón**

para obtener el grado de:
**Maestro en Ciencias**

en la especialidad de:
**Ingeniería Eléctrica**



Director de Tesis
**Dr. Raúl Ernesto González Torres**

Guadalajara, Jalisco, Septiembre de 2007.

# Lógicas Temporales Lineales para la Verificación Formal de Sistemas de Tiempo Real

**Tesis de Maestría en Ciencias**
**Ingeniería Eléctrica**

Por:
**Francisco Manzano Pinzón**
Licenciado en Ciencias de la Computación
Universidad Autónoma de Yucatán 1993-1998

Becario de Conacyt, expediente no. 181833

Director de Tesis
**Dr. Raúl Ernesto González Torres**

CINVESTAV del IPN Unidad Guadalajara, Septiembre de 2007.

*Agradecimientos:*

*A Dios, a mi Familia,*

*a mi Asesor, a mis Profesores y a CONACYT.*

Resumen:

La verificación formal es una metodología que prueba la validez de un conjunto de especificaciones o propiedades en el modelo de un sistema dado. La comprobación de modelos es un método de verificación formal de sistemas, el cual consiste en representar el modelo del sistema mediante alguna estructura de estados y un lenguaje formal para representar las propiedades que se quieren probar en el sistema, usualmente alguna lógica temporal.
Este trabajo se enfoca en la comprobación de modelos de sistemas de tiempo real. Se utilizan los autómatas temporizados para modelar sistemas de tiempo real. Para especificar propiedades de tiempo real se proponen dos lógicas temporales: LTPT (lógica temporal proposicional temporizada) en la cual los tiempos se cuantifican en un dominio discreto; y LTIM (lógica temporal de intervalos métricos) en la cual los tiempos se cuantifican en un dominio continuo. Se crea un algoritmo de comprobación de modelos de tiempo real resultante de una extensión al algoritmo de comprobación de modelos sin tiempos cuantificados.

Abstract:

The formal verification is a methodology that proves the validity of a set of specifications or properties on the model of a given system. The model checking is a method of formal verification of systems, which consists in represent the model of the system using some structure of states and a formal language for represent properties that want to be proved on the system, usually some temporal logic.
This work is focused on the model checking for real time systems. It uses temporized automatas for model real time systems. For specifiy real time properties it proposes two temporal logics: TPTL(timed propositional temporal logic) in which the times are cuantified over a discrete domain; and MITL (metric inteval temporal logic) in which the times are cuantified over a continuous domain; It creates a real time model checking algorithm that results from an extension of the model checking algorithm without cuantified times.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. Contenido del capítulo

Este capítulo presenta un breve panorama de la verificación de sistemas de tiempo real. En la sección 1.2 se resalta la necesidad que existe de garantizar que los sistemas realicen lo que se requiere. En la sección 1.3 se mencionan algunos de los beneficios de la comprobación de modelos como técnica automática para verificar sistemas. En la sección 1.4 se definen los sistemas de tiempo real; también se da una introducción a las lógicas temporales con tiempo cuantificado y a la comprobación de modelos extendida para sistemas de tiempo real. La sección 1.5 establece el objetivo de la tesis. Por último, se presenta la organización de éste trabajo.

## 1.2. Verificación automática de sistemas

La verificación de sistemas es un paso importante en el diseño de sistemas ya que:

*¡En la mayoría de los diseños se invierte más tiempo y esfuerzo en la verificación que en la construcción!* [18].

Esto no es algo sorprendente, ya que los errores pueden ser muy costosos. El error de división de punto flotante de la unidad Pentium Intel se estima que causó una pérdida aproximada de 500 millones de dólares. Considérese también el error de software en el cohete Ariane-5, que costó aproximadamente 500 millones de dólares, o el error en el sistema de control de equipaje de un aeropuerto de Denver, que provocó posponer su apertura por nueve meses con pérdidas de 1.1 millones de dólares por día.

Con el avance de la tecnología los sistemas que se construyen son cada vez más grandes y complejos. Esto dificulta cada vez más el hecho de garantizar que los sistemas cumplan con las especificaciones planteadas en el inicio del desarrollo del sistema. Así :

*Existe una fuerte necesidad de técnicas y herramientas que logren la verificación de sistemas de manera automática!* [18].

La verificación formal es una metodología que prueba la validez de un conjunto de especificaciones o propiedades en el modelo de un sistema dado.

La idea básica de la verificación formal es construir un modelo formal del sistema el cual representa el posible comportamiento del sistema. En adición, los requerimientos de buen funcionamiento son escritos en una especificación formal que representa el comportamiento deseable del sistema. Basado en estas dos especificaciones, se analiza por prueba formal si el posible comportamiento "está de acuerdo" con el comportamiento deseado.

## 1.3. Comprobación de modelos

La *comprobación de modelos* es un método de verificación formal de sistemas, el cual consiste en representar el modelo del sistema mediante alguna estructura de estados y transiciones, comúnmente redes de Petri, estructuras de Kripke y autómatas de estados finitos, por mencionar algunas. Por otro lado, también se requiere un lenguaje formal para representar las propiedades que se quieren probar en el sistema, usualmente alguna lógica temporal lineal [21, 22], o ramificada [7, 10].

El procedimiento de comprobación de modelos se divide en tres grandes actividades: modelado, especificación y prueba.

- Modelado

Consiste en representar el diseño del sistema con un formalismo aceptado por el método de comprobación de modelos. En muchos casos esto es una traducción directa en la que sólo se escribe el diseño con la sintaxis que acepta el método. En otros casos, dependiendo de las limitaciones de memoria, el modelo se construye haciendo una abstracción del diseño para eliminar todos aquellos detalles irrelevantes a la prueba. Por ejemplo, cuando se modelan circuitos, es muy común razonar en términos de compuertas y valores booleanos, y no en niveles de voltaje. De manera parecida, cuando se razona acerca de protocolos de comunicación, el interés principal está en el intercambio de mensajes y no en el contenido de ellos.

- Especificación

Se establecen todas las propiedades que debe cumplir el modelo. En un principio, estas propiedades son escritas en lenguaje natural. Sin embargo, para poder hacer el procesamiento de éstas es necesario representarlas con algún formalismo matemático; por ejemplo, como fórmulas de LTLP Es importante mencionar que una mala formulación de

alguna propiedad puede dar un resultado equivocado, por lo que se requiere de ser hábil para representar las propiedades correctamente.

- Prueba

Por conveniencia siempre se han utilizado herramientas de software que realizan esta actividad. Así por ejemplo, sólo basta ingresar el modelo y la propiedad en sus lenguajes aceptados correspondientes y la herramienta los procesa y arroja un resultado. La persona encargada de realizar la verificación formal normalmente interactúa con la herramienta para darse cuenta si la propiedad se cumple en el modelo o no. En caso de que la propiedad no se cumpla, analiza el contraejemplo y determina la parte del modelo en la que no se cumple para corregirlo.

En la figura 1.1 se puede observar el diagrama de la metodología de comprobación de modelos.

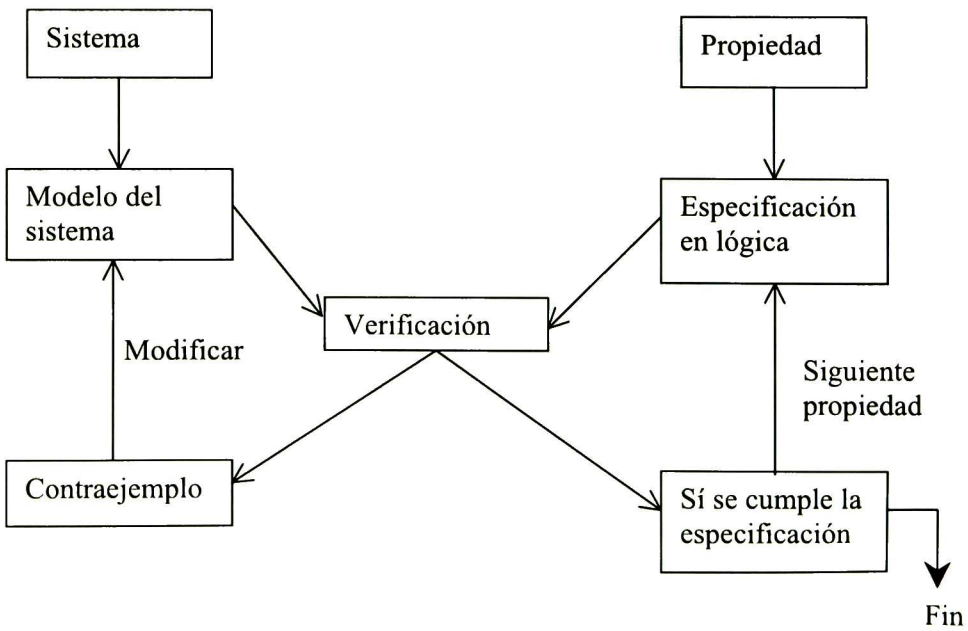


Figura 1.1: Metodología de comprobación de modelos.

Los principales beneficios de la comprobación de modelos son:

- Es un enfoque general aplicable a la verificación de hardware, ingeniería de software, sistemas multi-agentes, protocolos de comunicación, sistemas de control, sistemas embebidos, y otras más.
- Soporta verificación parcial: un diseño puede ser verificado sólo considerando un subconjunto de todos los requerimientos.
- Puede entregar un contraejemplo, en caso de que exista.
- Es totalmente automática, y por tanto no se requiere un alto grado de interacción entre las herramientas de comprobación de modelos y los usuarios.

## 1.4. Verificación de sistemas de tiempo real

Los *sistemas reactivos* son sistemas guiados por eventos y que continuamente tienen que reaccionar a estímulos internos y externos. El comportamiento de los sistemas reactivos no puede ser especificado solamente con las entradas y salidas del sistema; es necesario describir un mecanismo de control que determine el orden temporal de los procesos del sistema.

Los *sistemas de tiempo real* constituyen un subconjunto de los sistemas reactivos, en los cuales el orden temporal de los procesos debe ser cuantificado.

Un ejemplo típico de sistema de tiempo real es el cruce del tren: toda vez que se detecte la aproximación del tren, el cruce necesita cerrar sus compuertas con un cierto tiempo acotado a fin de detener el tráfico vehicular antes de que el tren llegue al cruce. Los protocolos de comunicación son otros ejemplos típicos de sistemas de tiempo real: después de la transmisión de un grupo de datos, se debe iniciar una retransmisión si la confirmación no es recibida dentro de un cierto tiempo acotado.

### 1.4.1. Modelado de sistemas de tiempo real

Los autómatas de estados finitos se han utilizado como modelos abstractos de sistemas reactivos. Cuando se quiere modelar sistemas temporizados con tiempo discreto, los autómatas de estados finitos también pueden ser utilizados para modelar estos sistemas.

Sin embargo, para modelar los sistemas de tiempo real se introduce la noción de *autómatas temporizados* [1, 2], los cuales son una extensión de los autómatas de estados finito equipados con relojes para medir el tiempo. Se pueden agregar tantos relojes como sean necesarios. También se pueden reiniciar los relojes deseados en cualquier transición predeterminada del autómata temporizado. Las restricciones temporizadas del sistema se ven reflejadas en restricciones de los valores de los relojes en los procesos y transiciones del autómata temporizado.

Desde su concepción, los autómatas temporizados han sido utilizados para modelar varios tipos de sistemas de tiempo real, yendo desde los protocolos de comunicación hasta los sistemas de seguridad crítica.

A continuación se describe el modelo del controlador de una lámpara. Este modelo es un ejemplo que manifiesta la necesidad de modelar los sistemas de tiempo real con autómatas temporizados.

**Ejemplo 1.1. Controlador de una lámpara.**

En un principio, la lámpara está apagada; si se presiona una vez el único botón de la lámpara, entonces ésta se enciende con una luz ligera. Si se presiona el botón cuando la luz es ligera y además hay poca diferencia de tiempo (digamos tres segundos) entre esta nueva presión del botón y la anterior, entonces la luz ahora pasará a ser muy brillante. Pero si la luz es ligera y además hay una diferencia de tiempo entre la nueva presión de tiempo y la anterior mayor a tres segundos, entonces la lámpara se apagará. Por último, si la luz es muy brillante y se presiona el botón, la lámpara se apagará.

La figura 1.2 muestra que los autómatas de estados finitos no pueden medir la diferencia de tiempo entre una presión del botón y la siguiente.

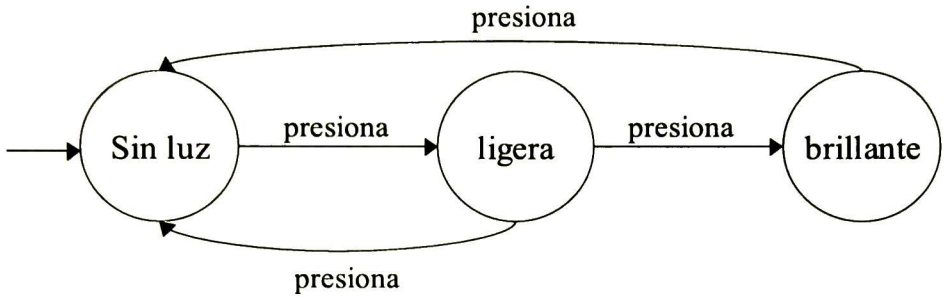


Figura 1.2: Modelo del controlador de una lámpara con un autómata de estados finito sin relojes.

La figura 1.3 muestra el modelo del controlador de la lámpara usando autómatas temporizados. La transición que tiene asociada la expresión 'x:=0' denota que se inicia el reloj 'x' cuando se ejecuta esta transición. La transición que tiene la restricción 'x<=3' indica que la transición puede ejecutarse siempre que el reloj cumpla con esa restricción.

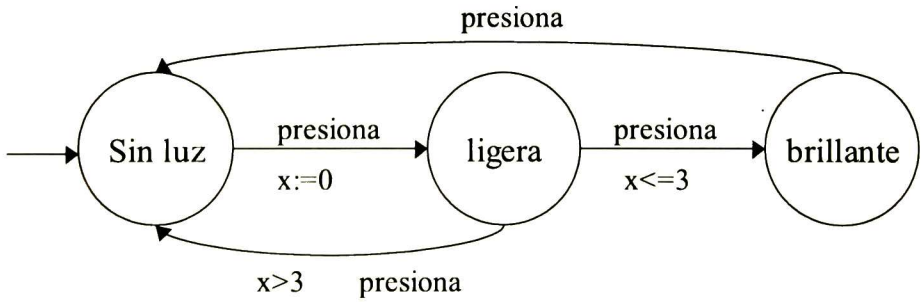


Figura 1.3: Modelo del controlador de una lámpara con un autómata temporizado.

### 1.4.2. Lógicas temporales con tiempo cuantificado

Las lógicas temporales como LTLP y CTL facilitan la especificación de propiedades que se enfocan en el orden temporal de eventos. En ellas, el orden temporal es una noción cualitativa; el tiempo no está cuantificado. Por ejemplo, si $p$ es la proposición correspondiente a la ocurrencia del evento A y $q$ es la proposición correspondiente a la ocurrencia del evento B, entonces la fórmula $\Box(p \rightarrow \Diamond q)$ de la LTLP expresa que siempre que ocurra el evento A eventualmente será seguido por el evento B. Pero esta fórmula no expresa nada acerca de qué tan largo será el periodo de tiempo entre las ocurrencias del evento A y el evento B. Esta ausencia de la noción cuantificada del tiempo es esencial para especificar propiedades de los sistemas de tiempo real.

Para considerar los tiempos cuantificados se han extendido las lógicas temporales cualitativas como LTLP y CTL. Algunas lógicas se han extendido tomando un dominio discreto de tiempo, por ejemplo, la lógica temporal proposicional temporizada (LTPT) [3], la cual es una lógica temporal lineal de tiempo discreto; otras lógicas tienen un dominio continuo del tiempo, por ejemplo, la lógica temporal de intervalos métricos (LTIM) [5], la cual es lineal de tiempo continuo. También se han extendido las lógicas de tiempo ramificado, para dar lugar, por ejemplo, a la lógica computacional arbórea temporizada (TCTL) [4] cuyo dominio de tiempo es continuo.

Ya que el tiempo es de naturaleza continua, la selección más obvia es aquella donde el tiempo tiene un dominio continuo, como el conjunto de los números reales no negativos. Para los sistemas asíncronos el dominio de tiempo continuo es lo apropiado. Sin embargo, para sistemas síncronos en los cuales los componentes funcionan por ciclos de reloj, el dominio de tiempo discreto es más apropiado.

**Ejemplo 1.2.**

Para definir un requerimiento de respuesta acotada que exprese que cada solicitud $p$ es seguida por una respuesta $q$ en a lo más 10 unidades de tiempo, se puede escribir la fórmula en LTPT como:

$$\Box x.\ (p \rightarrow \Diamond y.(q \wedge y \leq x + 10))$$

"Siempre que haya una petición $p$, y la variable $x$ sea *congelada* en ese instante, la petición es seguida por una respuesta $q$, en el tiempo $y$, tal que $y$ es a lo más $x$ +10"

Para expresar el mismo requerimiento de respuesta acotada en LTIM se escribe

$$\Box_{\geq 0}\ (p \rightarrow \Diamond_{(0,10]}\ q)$$

"Cada p-estado es seguido por un q-estado dentro de 10 unidades de tiempo"

En TCTL el requerimiento se expresa con la fórmula:

$$\forall\Box\ (p \rightarrow \forall\Diamond_{\leq 10}\ q)$$

"En toda trayectoria siempre ocurre que, si $p$, entonces en cualquier trayectoria eventualmente ocurre $q$ en a lo más 10 unidades de tiempo"

### 1.4.3. La comprobación de modelos usando lógicas temporales con tiempo cuantificado

Para realizar la comprobación de modelos con lógicas lineales de tiempo cuantificado en un dominio discreto, como es el caso de la LTPT, se pueden utilizar los algoritmos de la comprobación de modelos con lógicas lineales temporales cualitativas. Por un lado, se tiene el autómata del modelo del sistema, cuyos vértices están etiquetados con proposiciones especiales indicando las diferencias de tiempo con respecto a sus predecesores. Por otro lado, se obtiene el autómata correspondiente de la negación de una fórmula en LTPT; éste autómata tendrá tantos vértices según sean tan grandes los valores de las constantes

presentes en la misma fórmula en LTPT. En seguida, se construye el producto del autómata del modelo y del autómata de la negación de la propiedad. Por último, se busca en el producto paralelo un camino que parta desde algún estado inicial y termine en una componente fuertemente conexa donde las restricciones de reloj no impidan una ejecución infinita de la trayectoria establecida por dicho camino. Así, los algoritmos de la comprobación de modelos usando LTPT sólo son extensiones de los algoritmos de la comprobación usando lógicas lineales temporales cualitativas para considerar la cuantificación del tiempo.

En la comprobación de modelos con lógicas temporales lineales de tiempo real se utilizan autómatas temporizados tanto para el modelo abstracto del sistema como para la representación de la fórmula de la propiedad. Los relojes pueden reiniciar en cualquier transición predeterminada y se tienen también restricciones tanto para habilitar transiciones como para permanecer en los vértices, llamados localidades. Con el autómata temporizado del modelo del sistema y el autómata temporizado de la formula se obtiene el producto paralelo temporizado, y se determina si el lenguaje temporizado que acepta es o no es vacío.

El problema de vacuidad temporizado se traduce a uno de vacuidad clásico no temporizado. Para lograrlo, se realiza una abstracción del tiempo, llamada *autómata de región* [2], en el producto paralelo temporizado antes obtenido; en seguida, se procede a resolver el problema de vacuidad clásico no temporizado. Un contraejemplo encontrado corresponderá justamente a una trayectoria que indica la secuencia de estados donde la propiedad temporizada no se cumple.

Para el caso de comprobación de modelos con lógicas temporales ramificadas de tiempo real, como lo es TCTL, también se realiza una abstracción del tiempo. Sin embargo la abstracción sólo se aplica al modelo del sistema. De manera semejante a la comprobación de modelos con CTL, el algoritmo de etiquetado procede a etiquetar el autómata de región con subfórmulas de la fórmula de la propiedad. Si al final del algoritmo el estado inicial del autómata de región está etiquetado con la fórmula de la propiedad, se concluye que la propiedad se cumple en el modelo del sistema [4].

## 1.5. Objetivo

Existen trabajos científicos que han aplicado la comprobación de modelos para la verificación de sistemas de tiempo real, [8].

Para definir las especificaciones de sistema con tiempos cuantificados se han creado lógicas temporales cuantitativas las cuales extienden las lógicas temporales cualitativas con el propósito de considerar tiempos exactos, duraciones o intervalos de tiempo.

Para modelar los sistemas de tiempo real existen los autómatas temporizados para sistemas de tiempo lineal y los grafos temporizados para sistemas de tiempo ramificado.

Por tanto, también han surgido nuevos algoritmos en la comprobación de modelos, para poder considerar las lógicas de tiempo real y los modelos de tiempo real, así como también la manera de determinar si las especificaciones de tiempo real, ahora como formulas de lógica de tiempo real, se cumplen en los modelos de tiempo real.

Esta tesis se enfoca a dos lógicas lineales extendidas para expresar propiedades de sistemas de tiempo real. Específicamente, se trata de una lógica lineal temporizada discreta, la LTPT, y una lógica lineal temporizada continua, la LTIM.

El objetivo de este trabajo es presentar un estudio tanto de la LTPT como de la LTIM y proponer algoritmos para construir autómatas asociados a fórmulas escritas ya sea en LTPT o en LTIM. Un autómata asociado a una fórmula, la cual puede representar a una propiedad a verificar, es fundamental para determinar si dicha fórmula se cumple o no en el modelo de un sistema dado. También se busca que los algoritmos propuestos para LTPT y LTIM extiendan el algoritmo propuesto en [9], el cual construye un autómata asociado a una fórmula escrita en LTLP.

## 1.6. Contenido de la tesis

En el capítulo 2 se presenta la lógica temporal proposicional temporizada (LTPT), la cual se utiliza para expresar propiedades con tiempos cuantificados, donde el tiempo es lineal y de dominio discreto.

En el capítulo 3 se presenta la lógica temporal de intervalos métricos (LTIM), la cual se utiliza para expresar propiedades con tiempos cuantificados. El tiempo se considera lineal y de dominio continuo.

El capítulo 4 presenta los casos de estudio: se verifica el buen funcionamiento del controlador de compuerta del cruce de tren, de un controlador de semáforos, del protocolo de exclusión mutua de Fischer y del protocolo de comunicaciones CSMA/CD.

El capítulo 5 expone las conclusiones de este trabajo, así como también se proponen trabajos a futuro.

El apéndice A muestra las capturas de los modelos y propiedades de los casos de estudio, presentados en el capítulo 4, en las herramientas Kronos y Uppaal.

# Capítulo 2

# Lógica Temporal Proposicional Temporizada LTPT

## 2.1. Contenido del capítulo

En este capítulo se presenta la Lógica Temporal Proposicional Temporizada (LTPT). La LTPT es una lógica que contiene a la LTLP y también es decidible [3], esto es, se pueden crear algoritmos que logran determinar si una fórmula de la LTPT es satisfacible o no. La LTPT expresa propiedades a verificar en sistemas de tiempo real. Es una lógica que considera el tiempo lineal y los tiempos cuantificados están en un dominio discreto.

En la sección 2.6 se propone un algoritmo que determina si una fórmula LTPT es satisfacible. Para ello, se consideran los conceptos que definen a la LTPT en [3] para extender el trabajo de B. Casillas [9], construyendo un autómata asociado a una fórmula LTPT.

## 2.2. Secuencia temporizada de estados

Sea PA un conjunto de proposiciones atómicas (*p, q, r,* ...) y sea $\mathbb{N}$ el conjunto de enteros no negativos. El conjunto de tiempos es una copia T de N.

Un estado $\sigma$ es un subconjunto de PA.

Una *secuencia de estados* $\sigma = \sigma_0\sigma_1\sigma_2...$ es una secuencia infinita de estados $\sigma_i \subseteq PA$, $i \geq 0$

Una *secuencia de tiempos* $\tau = \tau_0\tau_1\tau_2...$ es una secuencia infinita de tiempos $\tau_i \in T$, $i \geq 0$, que cumple lo siguiente:

- *Monotonía* - $\tau_i \leq \tau_{i+1}$ para toda $i \geq 0$, y
- *Progreso* - para todo $t \in T$ hay un $i \geq 0$ tal que $\tau_i > t$

**Definición 2.2.1. (Secuencia temporizada de estados)**

Una *secuencia temporizada de estados* es un par $\rho = (\sigma, \tau)$ que consiste de una secuencia de estados $\sigma$ y una secuencia de tiempos $\tau$.

Se denota $\sigma^i$ (o $\tau^i$) a una secuencia de estados (o tiempos) que resulta de recortar los primeros i elementos de $\sigma$ (o $\tau$). También se define $\rho^i = (\sigma^i \ . \ \tau^i)$.

## 2.3. Sintaxis y semántica

Sea $V := (x, y, z, ...)$ un conjunto infinito de variables.

**Definición 2.3.1. (Sintaxis de la LTPT)**

Los términos $\pi$ y las fórmulas $\phi$ de la LTPT son definidos inductivamente de la siguiente manera:

$\pi := x + c \mid c;$

$\phi := p \mid \pi_1 \leq \pi_2 \mid \pi_1 \equiv_d \pi_2 \mid \bot$ (falsedad) $\mid \phi_1 \rightarrow \phi_2 \mid \bigcirc\phi \mid \phi_1 \, \mathcal{U} \, \phi_2 \mid x.\phi$

donde $x \in V$, $p \in PA$, y $c, d \in \mathbb{N}$, $d \neq 0$.

La afirmación $\pi_1 \equiv_d \pi_2$ significa que el tiempo $\pi_1$ es congruente con el tiempo $\pi_2$ módulo la constante d.

Las abreviaturas $=, <, >, \geq$, T (verdad), $\neg$, $\wedge$ y $\vee$ son definidas de manera usual.

**Operadores adicionales**

Otros operadores y modalidades temporales se pueden definir a partir de los anteriores.

- *Eventualmente* $\phi$, $\Diamond\phi$, es lo mismo que $\mathrm{T}\,\mathcal{U}\phi$;
- *Siempre* $\phi$, $\Box\phi$, es lo mismo que $\neg\Diamond\neg\phi$;
- $\phi_1$ *libera* a $\phi_2$, $(\phi_1\ \mathcal{V}\ \phi_2)$, es lo mismo que $\neg(\neg\phi_1\ \mathcal{U}\ \neg\phi_2)$;
- *No se cumple* $\phi$ en el tiempo 'x'. $\neg x.\phi$, es lo mismo que $x.\neg\phi$. En el tiempo 'x' *se cumple la negación* de $\phi$.

**Ejemplo 2.3.2.**

La afirmación "Cada petición $p$ es seguida por una respuesta $q$", se expresa en LTLP de la siguiente manera:

$$\Box(p \rightarrow \Diamond q)$$

La afirmación "Cada petición $p$ es seguida por una respuesta $q$ en a lo más 10 unidades de tiempo", se expresa en LTPT de la siguiente manera:

$$\Box x.(p \rightarrow \Diamond y.(q \wedge y \leq x + 10))$$

Esto es, siempre que haya una petición $p$, y la variable $x$ sea congelada en ese instante, la petición es seguida por una respuesta $q$, en el tiempo $y$, tal que $y$ es a lo más $x$+10.

**Definición 2.3.3. (Interpretación)**

Las *interpretaciones(o los ambientes)* son funciones $V \rightarrow$ T. Los ambientes son usados para determinar los valores de los relojes.

Sea $\varepsilon: V \rightarrow$ T un ambiente; entonces, $\varepsilon[x := t]$ denota el ambiente que está acorde con $\varepsilon$ sobre todas las variables excepto $x$, y asigna $t$ a $x$, donde $t \in$ T.

Para cualquier ambiente $\varepsilon$ se define $\varepsilon(x+c) := \varepsilon(x)+\varepsilon(c)$, $\varepsilon(c) := c$ donde $x \in V$ y $c \in \mathbb{N}$.

Las fórmulas LTPT se evalúan sobre secuencias temporizadas de estados.

**Definición 2.3.4. (Semántica de LTPT)**

Sea $\rho = (\sigma, \tau)$ una secuencia temporizada de estados y $\varepsilon$ un ambiente. La afirmación "el par $(\rho, \varepsilon)$ *satisface* la fórmula LTPT $\phi$" se denota por $\rho \models \varepsilon\ \phi$ y se define inductivamente de la siguiente manera:

$\rho \models \varepsilon\ p$ sii $p \in \sigma_0$;

$\rho \models \varepsilon\ \pi_1 \leq \pi_2$ sii $\varepsilon(\pi_1) \leq \varepsilon(\pi_2)$;

$\rho \models \varepsilon\ \pi_1 \equiv_d \pi_2$ sii $\varepsilon(\pi_1) \equiv_d \varepsilon(\pi_2)$;

$\rho \not\models \varepsilon\ \bot$;

$\rho \models \varepsilon\ \phi_1 \rightarrow \phi_2$ sii ( si $\rho \models \varepsilon\ \phi_1$ entonces $\rho \models \varepsilon\ \phi_2$ );

$\rho \models \varepsilon\ \bigcirc\phi$ sii $\rho^1 \models \varepsilon\ \phi$;

$\rho \models \varepsilon\ \phi_1 \mathcal{U} \phi_2$ sii para algún $i \geq 0$ $\rho^i \models \varepsilon\ \phi_2$, y para todo $0 \leq j < i$, $\rho^j \models \varepsilon\ \phi_1$;

$\rho \models \varepsilon\ x.\phi$ sii $\rho \models \varepsilon[x := \tau_0]\ \phi$ ;

Una fórmula LTPT es *cerrada* si cada ocurrencia de la variable $x$ está dentro del alcance de un cuantificador de congelamiento "$x$." De aquí en adelante sólo se consideran fórmulas LTPT cerradas.

La secuencia temporizada de estados $\rho$ es un *modelo* de la fórmula LTPT $\phi$, denotado por $\rho \models \phi$, si el par $(\rho, \varepsilon)$ satisface $\phi$ para cualquier ambiente $\varepsilon$.

La fórmula $\phi$ es *satisfacible (válida)* si $\rho \models \phi$ para alguna (todas) secuencia temporizada de estados $\rho$.

Dos fórmulas LTPT son *equivalentes* si tienen los mismos modelos.

# 2.4. Avance del tiempo

A continuación, se explica como el avance de tiempo actualiza las restricciones de tiempo de una fórmula LTPT.

Hágase la siguiente suposición:

Las restricciones de tiempo son de la forma $x \leq y+c$, $x+c \leq y$, ó $x \equiv_d y+c$, para enteros no negativos $d > c \geq 0$; $x, y \in V$.

Una secuencia temporizada de estados $\rho = (\sigma, \tau)$ es *Δ-acotada*, para una constante $\Delta \in \mathbb{N}$, si $\tau_i \leq \tau_{i-1} + \Delta$ para toda $i \geq 0$.

El enfoque de este trabajo es revisar la satisfacibilidad de fórmulas LTPT en modelos Δ-acotados.

El tiempo que está asociado a un estado puede ser modelado con una proposición de diferencia de tiempo $Prev_\delta$ , $0 \leq \delta \leq \Delta$.

Se puede capturar la información de tiempos y estados de una $\rho = (\sigma, \tau)$ mediante una secuencia de estados $^\sigma_i := \sigma_i \cup \{ Prev_{\delta_i} \}$, donde $\delta_i = \tau_i - \tau_{i-1}$ para toda $i \geq 0$. Se adopta la convención $\tau_{-1} := 0$.

## Actualizando restricciones de tiempo

Se procede a definir la fórmula $x.\psi(x)^\delta$ que resulta de actualizar todas las referencias en $\psi$ al tiempo inicial x por la diferencia de tiempo δ.

**Ejemplo 2.4.1.**

Si $x.\psi$ es la fórmula

$$x.\Box y.(p \rightarrow y.y \leq x + 7 ) .$$

entonces $x.\psi(x)^2$ , $x.\psi(x)^4$ $x.\psi(x)^7$ $x.\psi(x)^8$ son respectivamente iguales a

$$x.\Box y.(p \rightarrow y.y \leq x + 5 )$$

$$x.\Box y.(p \rightarrow y.y \leq x + 3 )$$

$$x.\Box y.(p \rightarrow y.y \leq x )$$

$$x.\Box y.(p \rightarrow \bot )$$

**Definición 2.4.2. (Actualizando restricciones de tiempo)**

Dada una fórmula LTPT $x.\psi$ y una $\delta \in \mathbb{N}$, la fórmula LTPT $x.\psi^{\delta}$ es definida inductivamente como sigue:

- $x.\psi^{0} := x.\psi$
- $x.\psi(x)^{\delta+1}$ es igual al reemplazo, en $x.\psi(x)^{\delta}$

  de los términos de la forma $x+(c+1)$ por $x+c$,

  y cada subfórmula (restricciones de tiempo) de la forma:

  $x \leq y+c$ con $\top$,

  $y+c+1 \leq x$ con $\bot$,

  $x \equiv_d y+c$ con $x \equiv_d y+(c+1)$,

  donde $x,y \in V$ y $c,d \in \mathbb{N}$ tal que $0 \leq c < d$.

El lema siguiente confirma que esta transformación actualiza todas las referencias de tiempo; esto es, que la fórmula $x.\psi(x)^{\delta}$ expresa la condición "$x.\psi(x-\delta)$"

**Lema 2.4.3. (Avance del tiempo)** [3]

Sean $\rho = (\sigma, \tau)$ una secuencia temporizada de estados, $\varepsilon$ un ambiente, y $\delta \in \mathbb{N}$, tal que $\delta \leq \tau_0$ . Entonces, para cada fórmula LTPT $x.\psi$,

$\rho \vDash \varepsilon\ x.\psi^{\delta}$ si y sólo si $\rho \vDash \varepsilon[x:=\tau_0-\delta]\psi$

## 2.5. Autómata asociado a una fórmula

En esta sección se definen la clausura de una fórmula LTPT, partículas temporizadas, transiciones y otros elementos que permiten construir el autómata correspondiente a una fórmula LTPT.

### Clasificación de fórmulas

Algunas fórmulas LTPT se clasifican como α o β-fórmulas de acuerdo a las tablas 2.1 y 2.2 respectivamente.

Una fórmula es llamada *α-fórmula* o *β-fórmula* si ella aparece en la columna izquierda de la tabla 2.1 o 2.2 respectivamente.

Para cada α-fórmula $\phi$, la tabla 2.1 contiene un conjunto de fórmulas $K(\phi)$, las cuales pueden ser vistas como las consecuencias de $\phi$. Para cada β-fórmula $\psi$, la tabla 2.2 contiene un conjunto de fórmulas $K_1(\psi)$ y un conjunto de fórmulas $K_2(\psi)$, las cuales pueden ser vista como consecuencias alternativas de $\psi$.

El significado intencionado de una fórmula que pertenece a una de estas tablas es como sigue. Una α-fórmula $\phi$ se cumple en la fila j si y sólo si todas las $K(\phi)$-fórmulas se cumplen en la fila j. Una β-fórmula $\psi$ se cumple en la fila j si y sólo si ya sea todas las $K_1(\psi)$-fórmulas, o todas las $K_2(\psi)$-fórmulas (o ambas) se cumplen en la fila j.

| $\alpha$ | $K(\alpha)$ |
|---|---|
| $z.(\psi \wedge \tau)$ | $z.\psi,\ z.\tau$ |
| $z.\Box\psi$ | $z.\psi,\ z.\bigcirc\Box\psi$ |
| $z.\neg\Diamond\psi$ | $z.\neg\psi,\ z.\bigcirc\neg\Diamond\psi$ |
| $z.x.\psi$ | $z.\psi[x:=z]$ |

Tabla 2.1. Definición de las α-fórmulas para la LTPT.

| $\beta$ | $K_1(\beta)$ | $K_2(\beta)$ |
|---|---|---|
| $z.(\psi \vee \tau)$ | $z.\psi$ | $z.\tau$ |
| $z.\Diamond\psi$ | $z.\psi$ | $z.\bigcirc\Diamond\psi$ |
| $z.\neg\Box\psi$ | $z.\neg\psi$ | $z.\bigcirc\neg\Box\psi$ |
| $z.(\psi \mathcal{U} \tau)$ | $z.\tau$ | $z.\psi$, $z.\bigcirc(\psi \mathcal{U} \tau)$ |
| $z.(\psi \mathcal{V} \tau)$ | $z.\psi$, $z.\tau$ | $z.\tau$, $z.\bigcirc(\psi \mathcal{V} \tau)$ |

Tabla 2.2. Definición de las β-fórmulas para la LTPT.

El algoritmo para obtener un autómata asociado a una fórmula LTPT, descrito en la sección 2.6, supone que la fórmula está en fnp.

**Definición 2.5.1. (Forma normal positiva)**

Una fórmula en *forma normal positiva* (fnp) es aquella en la cual las negaciones están aplicadas únicamente a las proposiciones atómicas.

En B. Casillas [9] se ha demostrado que toda fórmula LTLP se puede traducir a su fnp, esto se logra con la aplicación recursiva de reglas de equivalencia. Para traducir una fórmula LTPT a su fnp se proponen las siguientes reglas de equivalencia.

1) $\neg z.\psi \equiv z.\neg\psi$
2) Para las restricciones de tiempo $\neg(x < y+c)$ , $\neg(x \leq y+c)$, $\neg(x \equiv_d y+c)$ reemplazarlas por $(x \geq y+c)$ , $(x > y+c)$ o $(x \not\equiv_d y+c)$ respectivamente; donde $x,y \in V$ y $c,d \in \mathbb{N}$ tal que $0 \leq c < d$.
3) $\neg\neg z.\ \psi \equiv z.\psi$
4) $z.\neg(\psi \vee \tau) \equiv z.(\neg\psi \wedge \neg\tau)$
5) $z.\neg(\psi \wedge \tau) \equiv z.(\neg\psi \vee \neg\tau)$
6) $z.\neg\bigcirc\psi \equiv z.\bigcirc\neg\psi$

7) $z.\neg\Diamond\psi \equiv z.\Box\neg\psi$

8) $z.\neg\Box\psi \equiv z.\Diamond\neg\psi$

9) $z.\neg(\psi \mathcal{U} \tau) \equiv z.(\neg\psi \mathcal{V} \neg\tau)$

10) $z.\neg(\psi \mathcal{V} \tau) \equiv z.(\neg\psi \mathcal{U} \neg\tau)$

La complejidad en la construcción de un autómata asociado a una fórmula LTPT depende en gran medida del producto de las constantes que ocurren en la fórmula.

**Definición 2.5.2. (Producto de las constantes de $\phi$)**

Una constante *c ocurre* en una fórmula LTPT $\phi$ si $\phi$ contiene una subfórmula de la forma $x \leq y + (c\text{-}1)$ o $x + (c\text{-}1) \leq y$, o $\phi$ contiene el símbolo de predicado $\equiv_c$ .

Se define inductivamente $K_\phi$ como el *producto de todas las constantes* que ocurren en una fórmula LTPT $\phi$ como sigue:

$K_{\pi 1 \leq \pi 2} := K_{\pi 1} \; K_{\pi 2}$ ; $\quad K_{\pi_1 \equiv_c \pi_2} := c \; K_{\pi_1} \; K_{\pi_2}$ ;

$K_\perp := 1$ ; $\quad K_{\phi_1 \to \phi_2} := K_{\phi_1} \; K_{\phi_2}$ ;

$K_{\bigcirc\phi} := K_\phi$ ; $\quad K_{\phi_1 \mathcal{U} \phi_2} := K_{\phi_1} \; K_{\phi_2}$ ; $\quad K_{x.\phi'} := K_{\phi'}$ ;

$K_{x+c} := c+1$; $\quad K_c := c+1$; $\quad K_p := 1$;

donde $p \in PA$ y $c \in \mathbb{N}$ tal que $c \geq 0$.

**Lema 2.5.3. (Incremento de tiempo acotado)** [3]

Si la fórmula LTPT $\phi$ es satisfacible, entonces $\phi$ tiene un modelo $K_\phi$ -acotado.

Los modelos que satisfacen la fórmula LTPT $\phi$ se pueden obtener del autómata asociado a $\phi$, el cual se define a continuación. Los estados que conforman el autómata asociado a $\phi$ se construyen a partir de un conjunto finito de fórmulas LTPT llamado

clausura positiva de $\phi$. La clausura positiva de $\phi$ expresa todas las fórmulas LTPT que dividen recursivamente el presente y futuro de $\phi$.

De aquí en adelante, y sin perdida de generalidad, sólo se consideran fórmulas LTPT de la forma $z.\varphi$.

**Definición 2.5.4. (Clausura positiva)**

Dada una fórmula LTPT $z.\varphi$, la *clausura positiva* de $z.\varphi$ que se denota por $CL^+(z.\varphi)$, es el menor conjunto de fórmulas LTPT expresadas en fnp que satisfacen las siguientes condiciones:

- $z.\varphi \in CL^+(z.\varphi)$.
- Para toda $p \in PA$, si $z.p \in CL^+(z.\varphi)$, entonces $z.\neg p \in CL^+(z.\varphi)$.
- Para toda $z.(\lambda \rightarrow \psi) \in CL^+(z.\varphi)$, $z.\lambda \in CL^+(z.\varphi)$ y $z.\psi \in CL^+(z.\varphi)$.
- Para toda $z.\mu$ donde $\mu \in \{\lambda \mathcal{U} \psi, \lambda \mathcal{V} \psi\}$, si $z.\mu \in CL^+(z.\varphi)$, entonces $z.\bigcirc\mu \in CL^+(z.\varphi)$.
- Para toda fórmula $z.\bigcirc\psi \in CL^+(z.\varphi)$ y para cada $0 \leq \delta \leq K_{z.\varphi}$ $z.\psi^{\delta} \in CL^+(z.\varphi)$.
- Para toda fórmula $z.x.\psi \in CL^+(z.\varphi)$, $z.\psi[x:= z] \in CL^+(z.\varphi)$.

A los estados del autómata asociado a una fórmula LTPT $z.\varphi$ se le llaman partículas temporizadas. Estas son conjuntos de fórmulas de $CL^+(z.\varphi)$ que cumplen con las propiedades introducidas a continuación.

**Definición 2.5.5. (Partícula temporizada)**

Un subconjunto $P$ de $CL^*(\phi) = CL^+(\phi) \cup \{Prev_\delta \mid 0 \leq \delta \leq \Delta = K_\phi\}$ es *maximalmente consistente* si satisface las siguientes condiciones:

- $Prev_\delta$ está en $P$ para precisamente una $\delta$, $0 \leq \delta \leq \Delta$; esta $\delta$ es referida como $\delta_P$
- $z.(z \sim z+c)$ está en $P$ si y sólo si se cumple $0 \sim c$ (donde $\sim$ puede ser $\geq$, $\leq$ ó $\equiv_d$ ).
- $z.\perp$ no está en $P$

- Si z.$p \in P$, entonces z.$\neg p \notin P$.
- Para toda α-fórmula $\psi \in CL^*(\phi)$, $\psi \in P$ y $K(\psi) \subseteq P$
- Para toda β-fórmula $\psi \in CL^*(\phi)$, $\psi \in P$, $K_1(\psi) \subseteq P$ ó $K_2(\psi) \subseteq P$.

Una *partícula temporizada* sobre una fórmula LTPT $\phi$ es un subconjunto maximalmente consistente sobre $CL^*(\phi)$.

La partícula temporizada donde no hay fórmulas, y sólo muestra el avance del tiempo es la partícula temporizada vacía.

**Definición 2.5.6. (Partícula temporizada vacía)**

La partícula temporizada que no contiene fórmulas y sólo contiene la proposición $Prev_\delta$, para cualquier $0 < \delta \leq K_\phi$, es la *partícula temporizada vacía*, que se denota por $P_\varnothing$.

**Definición 2.5.7. (Transición)**

Existe la *transición* de la partícula temporizada $P$ a la partícula temporizada $Q$ si y sólo si para toda fórmula z.$\bigcirc\varphi$ que está en $CL^*(\phi)$,

$$z.\bigcirc\varphi \in P \text{ si y sólo si } z.\varphi^{\delta_Q} \in Q$$

**Definición 2.5.8. (Autómata $A_\phi$)**

El *autómata* $A_\phi$, asociado a la fórmula LTPT $\phi$, es una tripleta $(S, S_I, \rightarrow)$, donde:

- S es un conjunto de partículas temporizadas sobre $\phi$.
- $S_I \subseteq S$ donde cada partícula temporizada de $S_I$ contiene a $\phi$, a estas se le llaman *partículas temporizadas iniciales*.
- $\rightarrow \subseteq S \times S$ tal que $(P, Q) \in \rightarrow$ si y sólo si existe la transición de $P$ a $Q$.

El significado del autómata $A_\phi$ para una fórmula $\phi$ es que cada modelo de $\phi$ corresponde a un camino infinito en $A_\phi$ donde todas las eventualidades son satisfechas a tiempo.

**Definición 2.5.9. ($\phi$-camino)**

Sea $A_\phi = (S, S_I, \rightarrow)$ el autómata asociado a $\phi$. Un camino infinito a lo largo de $A_\phi$

$\Phi : \Phi_0, \Phi_1, \Phi_2, \ldots$ , donde $\Phi_i \in S$ tal que $(\Phi_i , \Phi_{i+1}) \in \rightarrow$ , para toda $i \geq 0$,

es un $\phi$-*camino* si satisface las tres condiciones siguientes:

1) *Inicio* - $\phi \in \Phi_0$
2) *Legalidad* - para toda fórmula $z.(\psi_1 \mathcal{U} \psi_2) \in CL^*(\phi)$ con $i \geq 0$,

$$z.(\psi_1 \mathcal{U} \psi_2) \in \Phi_i \text{ implica } z.\psi_2^{\delta} \in \Phi_j \text{ para alguna } j \geq i, \text{ con } \delta = \sum_{i<k\leq j} \delta_{\Phi_k}$$

y $z.\psi_1 \in \Phi_k$ para todo $i \leq k \leq j-1$.

3) *Progreso* - $\delta_{\Phi_i} > 0$ para una infinidad de $i \geq 0$.

Un $\phi$-camino de $A_\phi$ , el autómata asociado a la fórmula $\phi$, corresponde justamente con un modelo que satisface a $\phi$ y viceversa. De esta manera, el problema de encontrar un modelo de $\phi$ se traduce en un problema de construcción de un autómata y búsqueda de un $\phi$-camino en dicho autómata.

**Lema 2.5.10. (Satisfacibilidad para una fórmula de la LTPT)** [3]

(1) *Correctud* - Si el autómata $A_\phi$ para la fórmula LTPT $\phi$ contiene un $\phi$-camino, entonces $\phi$ es satisfacible.

(2) *Completud* - Si $\phi$ tiene un modelo $\Delta$-acotado, entonces $A_\phi$ contiene un $\phi$-camino.

En R. Alur [3] se demostró que construir $A_\phi$ y buscar un $\phi$-camino sobre $A_\phi$ es decidible; es decir, existe un algoritmo que en un tiempo finito realiza la construcción y búsqueda. Por lo tanto, la LTPT es decidible.

**Teorema 2.5.11. (Dedicidibilidad para la LTPT)** [3]

El problema de validez para una fórmula $\phi$ de la LTPT puede ser decidido en tiempo exponencial sobre *nk*, donde *n*-1 es el número de operadores booleanos, temporales y de congelamiento presentes en $\phi$, y *k* es el producto de todas las constantes que ocurren en $\phi$.

## 2.6. Construcción incremental del autómata asociado $A_\phi$

En este trabajo se propone un algoritmo incremental; una extensión del algoritmo de B. Casillas [9] para la LTLP Este algoritmo evita crear al inicio todas las partículas temporizadas sobre $\phi$; pues solamente crea las partículas temporizadas iniciales, después construye las partículas temporizadas sucesoras a las existentes, y así recursivamente se irán construyendo partículas temporizadas sucesoras hasta no haber más.

La manera en que se construyen las partículas temporizadas sucesoras se basa en la consideración de las fórmulas de estado siguiente. Por eso se introduce la noción de fórmulas implicadas por una partícula temporizada.

**Definición 2.6.1. (Fórmulas implicadas)**

Si *P* es una partícula temporizada sobre una fórmula LTPT $\phi$, entonces para una $\delta$ en $0 < \delta \leq K_\phi$, $\psi^\delta$ es una *fórmula implicada* por *P* si y sólo si $\bigcirc\psi \in P$;

Se denota como *imps*(*P*) al conjunto de fórmulas implicadas por la partícula temporizada *P*; e *imps*(*P*, $\delta$) es igual a *imps*(*P*) $\cup$ {Prev$_\delta$} donde $0 < \delta \leq K_\phi$.

Las fórmulas implicadas dan la base para construir las partículas temporizadas sucesoras. A continuación, se construyen todos los posibles subconjuntos maximalmente consistentes sobre las fórmulas implicadas, esto es, todos aquellos subconjuntos que efectivamente son partículas temporizadas. Esto se realiza aplicando la función Cubierta de partículas temporizadas de las fórmulas implicadas; antes se define un conjunto atómicamente consistente, un conjunto $\alpha$-cerrado y un conjunto $\beta$-cerrado

**Definición 2.6.2.( Conjunto atómicamente consistente, conjunto $\alpha$-cerrado, conjunto $\beta$-cerrado)**

Sea B un subconjunto de CL*($\phi$). Se dice que B es:

i. *Atómicamente consistente*, si no contiene la constante $\perp$ (o $\neg\mathsf{T}$ ), ni contiene una proposición atómica y su negación.

ii. $\alpha$-*cerrado*, si para toda $\alpha$-fórmula $\psi$ en CL*($\phi$), $\psi \in B$ si y sólo si $K(\psi) \subseteq B$.

iii. $\beta$-*cerrado*, si para toda $\beta$-fórmula $\psi$ en CL*($\phi$), $\psi \in B$ si y sólo si

$$K_1(\psi) \subseteq B \text{ ó } K_2(\psi) \subseteq B .$$

Para obtener un conjunto $\beta$-cerrado, que contenga un conjunto B dado, se aplica de ser necesario una de las operaciones siguientes a cada $\beta$-fórmula $\psi$ en CL*($\phi$):

iv. $\beta$-*expansión*, en la cual se añade a B los elementos de uno de los conjuntos $K_1(\psi)$ ó $K_2(\psi)$, siempre que $\psi \in B$;

v. $\beta^{-1}$-*expansión*, en la cual se añade $\psi$ a B, si al menos uno de los conjuntos correspondientes $K_1(\psi)$ y $K_2(\psi)$ está contenido en B.

vi. La $\alpha$-*cerradura* de B es el menor subconjunto $\alpha$-*cerrado* de CL*($\phi$) que contiene a B y se denota con $\alpha(B)$.

Para construir la $\alpha$-cerradura de un conjunto B se aplican dos tipos de operaciones a cada $\alpha$-fórmula $\psi$ en CL*($\phi$):

vii. $\alpha$-*expansión*, en la cual se añaden a B los elementos del conjunto $K(\psi)$, si $\psi \in B$;

viii. $\alpha^{-1}$-*expansión*, en la cual se añade $\psi$ a B, si su conjunto correspondiente $K(\psi)$ está contenido en B.

ix. Una $\beta$-fórmula $\psi$ está *apoyada por un conjunto* X (contenido en CL*($\phi$)) si y sólo si $K_1(\psi) \subseteq X$ ó $K_2(\psi) \subseteq X$.

**Definición 2.6.3. (Función Cubierta$_P$)**

La *función Cubierta$_P$*, de la colección $2^{CL^*(\phi)}$ de los subconjuntos de $CL^*(\phi)$ en la colección $2^{2^{CL^*(\phi)}}$ de colecciones de subconjuntos de $CL^*(\phi)$, está dada recurrentemente por las reglas siguientes:

- Cubierta$_P(\varnothing)$ := Cubierta$_P(\{\top\})$:= $\{P_\varnothing\}$;
- Cubierta$_P(X)$ := $\varnothing$, si X no es atómicamente consistente;
- Cubierta$_P(X)$ := Cubierta$_P(\alpha(X))$, si X no es $\alpha$-cerrado;
- Cubierta$_P(X)$ := Cubierta$_P(X \cup \{\psi\})$, si existe una $\beta$-fórmula $\psi$ en $CL^*(\phi)$ tal que $\psi \notin X$, pero $K_1(\psi) \subseteq X$ ó $K_2(\psi) \subseteq X$, ($\beta^{-1}$-expansión);
- Cubierta$_P(X)$ := Cubierta$_P(X \cup K_1(\psi)) \cup$ Cubierta$_P(X \cup K_2(\psi))$, si existe una $\beta$-fórmula $\psi$ en X que no está apoyada por X, ($\beta$-expansión);
- Cubierta$_P(X)$ := $\{ P_\varnothing \}$ Si X sólo contiene la proposión $Prev_\delta$, donde $0 < \delta \leq K_\phi$ ;
- Cubierta$_P(X)$ := Cubierta$_P(X \cup \{Prev_{\delta=0}\})$ Si X no contiene algún $Prev_\delta$, donde $0 < \delta \leq K_\phi$;
- Cubierta$_P(X)$ := $\{X\}$ de otro modo.

Se propone el siguiente algoritmo incremental para construir el autómata asociado $A_\phi = (S, S_I, \rightarrow)$ a una fórmula LTPT $\phi$ en fnp.

**Algoritmo 2.6.4**

1) Se obtiene $CL^*(\phi)$.
2) Se obtiene Cubierta$_P(\{\phi\})$:= Cubierta$_P(\{\phi, Prev_{\delta=0}\})$; las partículas temporizadas resultantes se agregan al conjunto S, al igual que al conjunto de partículas temporizadas iniciales $S_I$.
3) Para cada $P \in S$ se obtienen las partículas sucesoras de $P$, de acuerdo a la función *Sucesores*, definida por la regla siguiente:

$$Sucesores(P) := \bigcup_{0<\delta\leq K_\phi} \text{Cubierta}_P\ (\ imps(P, \delta)\ )$$

4) Si Q ∈ *Sucesores(P)* y Q ∉ S, entonces Q se agrega al conjunto S. Además, si $\phi \in Q$ y Q ∉ $S_I$, entonces Q se agrega al conjunto $S_I$.
5) Se agregan las transiciones de la forma (*P*, *Q*) a la relación de transición →, para cada Q ∈ *Sucesores(P)*.
6) Ir al paso 3) en caso de haber agregado nuevas partículas a S en el paso 4); de lo contrario, continuar al paso 7).
7) Se ha terminado la construcción de $A_\phi$.

**Ejemplo 2.6.5.**

Aplicar el algoritmo incremental 2.6.1. para obtener el autómata $A_\phi = (S, S_I, \rightarrow)$, asociado a la fórmula $\varphi := z.\Diamond x.(p \wedge (x \leq z+2))$. La fórmula dice que desde que inicia el sistema en el tiempo 'z' eventualmente ocurrirá el evento 'p' en el tiempo 'x' y habrán pasado a lo más dos unidades de tiempo.

Paso 1.

$K_\phi = 2+1 = 3$.

$CL^+(\varphi) = \{ z.\Diamond x.(p \wedge (x \leq z+2)), z.x.(p \wedge (x \leq z+2)), z.\bigcirc\Diamond x.(p \wedge (x \leq z+2)),$
$z.(p \wedge (z \leq z+2)), z.\Diamond x.(p \wedge (x \leq z+1)), z.\Diamond x.(p \wedge (x \leq z)), z.\bigcirc\Diamond x.(p \wedge \bot) = z.\bot,$
$z.p, z.(z \leq z+2), z.x.(p \wedge (x \leq z+1)), z.\bigcirc\Diamond x.(p \wedge (x \leq z+1)),$
$z.\neg p, z.(p \wedge (z \leq z+1)), z.(z \leq z+1),$
$z.x.(p \wedge (x \leq z)), z.\bigcirc\Diamond x.(p \wedge (x \leq z)),$
$z.(p \wedge (z \leq z)), z.(z \leq z) \}$

$CL^*(\varphi) = CL^+(\varphi) \cup \{Prev_0, Prev_1, Prev_2, Prev_3\}$

Paso2.

Se calcula $Cubierta_P(\{\varphi\}) := Cubierta_P(\{\varphi, Prev_{\delta=0}\})$, creando el árbol semántico para φ mostrado en la figura 2.1.

Luego, Cubierta$_P$({φ, Prev$_{\delta=0}$}) = {P$_1$, P$_2$}, donde cada P$_i$ contiene todas las fórmulas que están sobre la misma rama del árbol, la cual va de la raíz al nodo terminal indicado por P$_i$.

P$_1$ = { z.◇x.(p∧(x≤z+2), z.x.(p∧(x≤z+2)), z.(p∧(z≤z+2)), z.p, z.(z≤z+2), Prev$_0$ }

P$_2$ = { z.◇x.(p∧(x≤z+2), z.○◇x.(p∧(x≤z+2)), Prev$_0$ }

Hacer S:= {P$_1$, P$_2$} y S$_1$ = {P$_1$, P$_2$}.

Pasos 3 a 6.

Sucesores(P$_1$)= Cubierta$_P$(*imps*(P$_1$, 1)) ∪ Cubierta$_P$(*imps*(P$_1$, 2)) ∪ Cubierta$_P$(*imps*(P$_1$, 3));

Cubierta$_P$(*imps*(P$_1$, 1)) = Cubierta$_P$({ Prev$_1$ })= { $P_\varnothing$ };

Cubierta$_P$(*imps*(P$_1$, 2)) = Cubierta$_P$({ Prev$_2$ })= { $P_\varnothing$ };

Cubierta$_P$(*imps*(P$_1$, 3)) = Cubierta$_P$({ Prev$_3$ })= { $P_\varnothing$ };

Sucesores(P$_1$) = { $P_\varnothing$ };

Agregar a la relación → la transición (P$_1$, P$_\varnothing$).

Sucesores(P$_2$)= Cubierta$_P$(*imps*(P$_2$, 1)) ∪ Cubierta$_P$(*imps*(P$_2$, 2)) ∪ Cubierta$_P$(*imps*(P$_3$, 3));

Cubierta$_P$(*imps*(P$_2$, 1)) = Cubierta$_P$({z.◇x.(p∧(x≤z+1), Prev$_1$}) ;

Cubierta$_P$(*imps*(P$_2$, 2)) = Cubierta$_P$({z.◇x.(p∧(x≤z), Prev$_2$}) ;

Cubierta$_P$(*imps*(P$_2$, 3)) = Cubierta$_P$({ Prev$_3$ })= { $P_\varnothing$ };

Para calcular Cubierta$_P$({ z.◇x.(p∧(x≤z+1), Prev$_1$ }) se construye el árbol semántico mostrado en la figura 2.2.

Cubierta$_P$({ z.◇x.(p∧(x≤z+1), Prev$_1$ } = {P$_3$, P$_4$}, donde

P$_3$ = { z.◇x.(p∧(x≤z+1), z.x.(p∧(x≤z+1)), z.(p∧(z≤z+1)), z.p, z.(z≤z+1), Prev$_1$ }

P$_4$ = { z.◇x.(p∧(x≤z+1), z.○◇x.(p∧(x≤z+1)), Prev$_1$ }

Para calcular Cubierta$_P$({z.◇x.(p∧(x≤z), Prev$_2$}) se construye el árbol semántico mostrado en la figura 2.3.

Cubierta$_P$({z.◇x.(p∧(x≤z), Prev$_2$}) = {P$_5$, P$_6$}, donde

$P_5$ = { z.◇x.(p∧(x≤z), z.x.(p∧(x≤z)), z.(p∧(z≤z)), z.p, z.(z≤z), $Prev_2$ },

$P_6$ = { z.◇x.(p∧(x≤z), z.○◇x.(p∧(x≤z)), $Prev_2$ }.

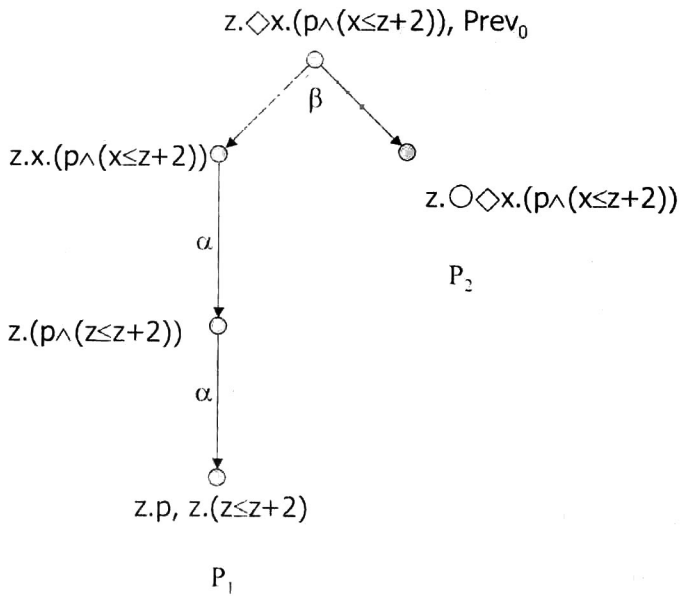


Figura 2.1. Árbol semántico para calcular $Cubierta_P(\{\varphi\})$.

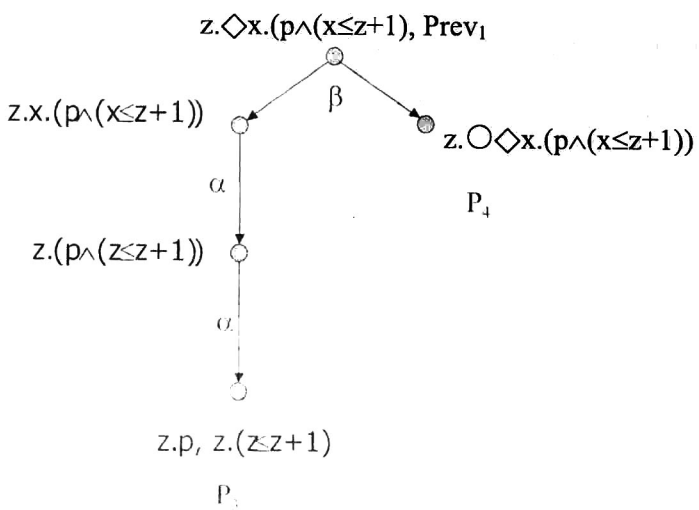


Figura 2.2. Árbol semántico para calcular $Cubierta_P(\{z.\Diamond x.(p \wedge (x \leq z+1), Prev_1\})$

Entonces, Sucesores($P_2$) = {$P_3$, $P_4$} U {$P_5$, $P_6$} = {$P_3$, $P_4$, $P_5$, $P_6$}.

Agregar a la relación → las transiciones ($P_2$, $P_3$), ($P_2$, $P_4$), ($P_2$, $P_5$) y ($P_2$, $P_6$).

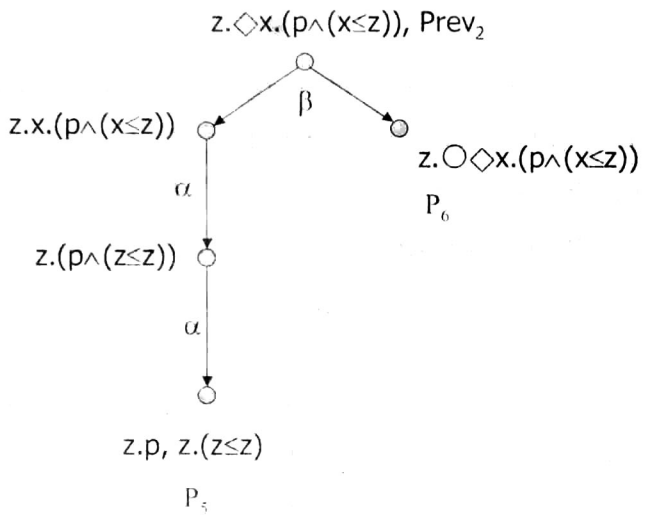


Figura 2.3. Árbol semántico para calcular Cubierta$_P$({z.◇x.(p∧(x≤z), Prev2})

Ahora se calculan los sucesores para cada partícula temporizada de {$P_3$, $P_4$, $P_5$, $P_6$}.

Sucesores($P_3$) = Cubierta$_P$(*imps*($P_3$, 1)) U Cubierta$_P$(*imps*($P_3$, 2)) U Cubierta$_P$(*imps*($P_3$, 3));

Cubierta$_P$(*imps*($P_3$, 1)) = Cubierta$_P$({ Prev$_1$}) = { $P_\varnothing$ };

Cubierta$_P$(*imps*($P_3$, 2)) = Cubierta$_P$({ Prev$_2$}) = { $P_\varnothing$ };

Cubierta$_P$(*imps*($P_3$, 3)) = Cubierta$_P$({ Prev$_3$}) = { $P_\varnothing$ }.

Sucesores($P_3$) = { $P_\varnothing$ }.

Agregar a la relación → la transición ($P_3$, $P_\varnothing$).

Sucesores($P_4$) = Cubierta$_P$(*imps*($P_4$, 1)) U Cubierta$_P$(*imps*($P_4$, 2)) U Cubierta$_P$(*imps*($P_4$, 3));

Cubierta$_P$(*imps*($P_4$, 1)) = Cubierta$_P$({z.◇x.(p∧(x≤z)), Prev$_1$}) = {$P_7$, $P_8$}.

Donde, $P_7 = \{ z.\Diamond x.(p \wedge (x \leq z), z.x.(p \wedge (x \leq z)), z.(p \wedge (z \leq z)), z.p, z.(z \leq z), Prev_1 \}$ y

$P_8 = \{ z.\Diamond x.(p \wedge (x \leq z), z.\bigcirc\Diamond x.(p \wedge (x \leq z)), Prev_1 \}$;

$Cubierta_P(imps(P_4, 2)) = Cubierta_P(\{ z.\bot \}) = \varnothing$;

$Cubierta_P(imps(P_4, 3)) = Cubierta_P(\{ z.\bot \}) = \varnothing$.

$Sucesores(P_4) = \{P_7, P_8\}$.

Agregar a la relación $\rightarrow$ las transiciones $(P_4, P_7)$ y $(P_4, P_8)$.

$Sucesores(P_5) = Cubierta_P(imps(P_5, 1)) \cup Cubierta_P(imps(P_5, 2)) \cup Cubierta_P(imps(P_5, 3))$;

$Cubierta_P(imps(P_5, 1)) = Cubierta_P(\{ Prev_1 \}) = \{ P_\varnothing \}$;

$Cubierta_P(imps(P_5, 2)) = Cubierta_P(\{ Prev_2 \}) = \{ P_\varnothing \}$;

$Cubierta_P(imps(P_5, 3)) = Cubierta_P(\{ Prev_3 \}) = \{ P_\varnothing \}$.

Agregar a la relación $\rightarrow$ la transición $(P_5, P_\varnothing)$.

$Sucesores(P_6) = Cubierta_P(imps(P_6, 1)) \cup Cubierta_P(imps(P_6, 2)) \cup Cubierta_P(imps(P_6, 3))$;

$Cubierta_P(imps(P_6, 1)) = Cubierta_P(\{z.\bot\}) = \varnothing$;

$Cubierta_P(imps(P_6, 2)) = Cubierta_P(\{z.\bot\}) = \varnothing$;

$Cubierta_P(imps(P_6, 3)) = Cubierta_P(\{z.\bot\}) = \varnothing$.

No se agrega ninguna transición a la relación $\rightarrow$.

$Sucesores(P_7) = Cubierta_P(imps(P_7, 1)) \cup Cubierta_P(imps(P_7, 2)) \cup Cubierta_P(imps(P_7, 3))$;

$Cubierta_P(imps(P_7, 1)) = Cubierta_P(\{ Prev_1 \}) = \{ P_\varnothing \}$;

$Cubierta_P(imps(P_7, 2)) = Cubierta_P(\{ Prev_2 \}) = \{ P_\varnothing \}$;

$Cubierta_P(imps(P_7, 3)) = Cubierta_P(\{ Prev_3 \}) = \{ P_\varnothing \}$.

Agregar a la relación $\rightarrow$ la transición $(P_7, P_\varnothing)$.

$Sucesores(P_8) = Cubierta_P(imps(P_8, 1)) \cup Cubierta_P(imps(P_8, 2)) \cup Cubierta_P(imps(P_8, 3))$;

$Cubierta_P(imps(P_8, 1)) = Cubierta_P(\{z.\bot\}) = \varnothing$.

$Cubierta_P(imps(P_8, 2)) = Cubierta_P(\{z.\bot\}) = \varnothing$.

$Cubierta_P(imps(P_8, 3)) = Cubierta_P(\{z.\bot\}) = \varnothing$.

No se agrega ninguna transición a la relación →.

Paso 7.

Se ha terminado la construcción de $A_\phi$. Ver figura 2.4.

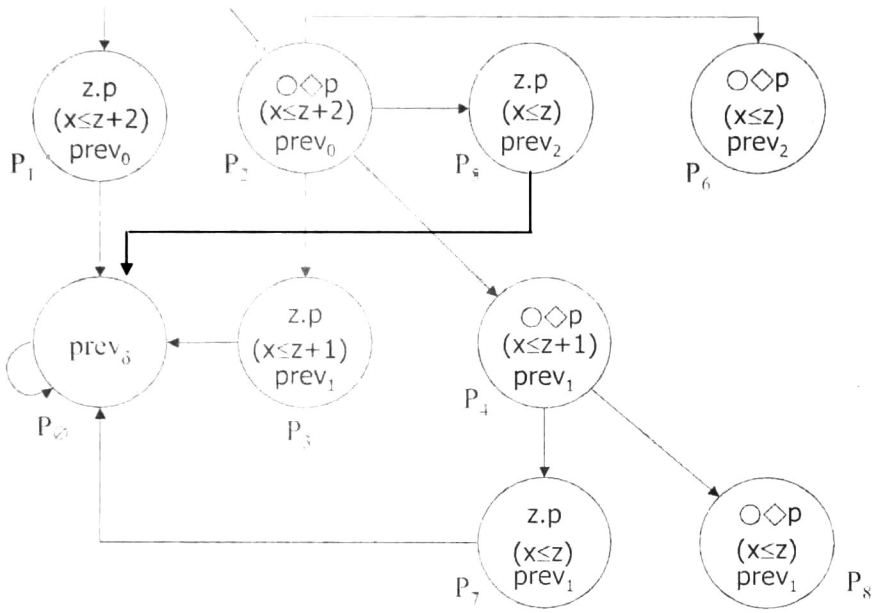


Figura 2.4. Autómata asociado a φ:= z.◇x.(p ∧ (x ≤ z+2)).

## 2.7 Comprobación de modelos utilizando LTPT

Si se desea verificar una propiedad temporizada, escrita en LTPT, en un sistema de tiempo real, se puede utilizar la comprobación de modelos, [3]. El sistema de tiempo real se modela con un *grafo temporizado de estados* [3].

Los pasos para verificar una propiedad temporizada $\phi$, escrita en LTPT, en un sistema de tiempo real por comprobación de modelos son los siguientes:

1. Obtener la forma normal positiva de la negación de $\phi$, la cual representa la propiedad a verificar.
2. Obtener el autómata temporizado asociado a la negación de la fórmula $\phi$ en fnp. Esto es, se calcula $A_{\neg\phi}$.
3. Calcular el producto síncrono temporizado [3], de los autómatas de la negación de la propiedad y del modelo abstracto del sistema de tiempo real, esto es, un grafo temporizado de estados *G*. Por tanto, se calcula el autómata $G \times A_{\neg\phi}$.
4. Se realiza la búsqueda de algún ciclo de aceptación en $G \times A_{\neg\phi}$.
5. El sistema *G* cumple la propiedad $\phi$ si y sólo si no se encuentra un ciclo de aceptación en $G \times A_{\neg\phi}$.

El problema de validez para una fórmula $\phi$ de la LTPT con respecto a un grafo temporizado de estados *G*, puede ser decidido en tiempo lineal determinístico con respecto al tamaño de *G* y doblemente exponencial con respecto a la longitud de $\phi$, donde la longitud de $\phi$ es el número de operadores booleanos, temporales y de congelamiento presentes en $\phi$ [3].

# Capítulo 3

# Lógica Temporal de Intervalos Métricos LTIM

## 3.1. Contenido del capítulo

Este capítulo presenta la Lógica Temporal de Intervalos Métricos (LTIM). La LTIM es una lógica decidible [5], esto es, se pueden crear algoritmos que logran determinar si una fórmula de la LTIM es satisfacible o no. La LTIM expresa propiedades a verificar en sistemas de tiempo real. LTIM es una lógica que considera el tiempo lineal y los tiempos cuantificados están en el dominio de los números reales no negativos.

Se propone un algoritmo para construir un autómata asociado a una fórmula LTIM; la fórmula es satisfacible siempre y cuando el lenguaje aceptado por su autómata asociado no es vacío. Este trabajo se basa en los conceptos que definen a la LTIM según [5] y [13] y

extiende el trabajo de B. Casillas [9], el cual construye un autómata asociado a una fórmula LTLP.

El contenido de éste capítulo es el siguiente: en la sección 3.2 se definen los intervalos de tiempo de la LTIM; la sección 3.3 explica las secuencias temporizadas de estados como modelos que satisfacen fórmulas LTIM; la sección 3.4 define la sintaxis y semántica de LTIM; la sección 3.5 define el refinamiento de modelos, el cual permite desarrollar algoritmos incrementales para LTIM; la sección 3.6 define la $LTIM_{\leq}$ cuyo problema de satisfacibilidad tiene menor complejidad que el de LTIM; la sección 3.7 define los autómatas temporizados, que son autómatas finitos extendidos con variables de reloj; finalmente, en la sección 3.8 se propone un algoritmo incremental el cual construye un autómata temporizado asociado una fórmula $LTIM_{\leq}$

## 3.2. Intervalos de tiempo

Sean $\mathbb{N}$ el conjunto de enteros no negativos y $R_{\geq 0}$ el conjunto de los números reales no negativos.

**Definición 3.2.1. (Intervalo de tiempo)**

Un *intervalo de tiempo* es un subconjunto convexo no vacío de $R_{\geq 0}$.

Los intervalos pueden ser abiertos, semi-abiertos, ó cerrados; acotados ó no acotados. Cada intervalo tiene una de las siguientes formas:

$(a, b), [a, b), (a, b], [a, b], [a, \infty), (a, \infty)$, donde $a \leq b$ para $a, b \in R_{\geq 0}$.

Para estos intervalos, 'a' es el extremo izquierdo, y 'b' el extremo derecho. El extremo izquierdo para un intervalo I se denota con Izq(I), y el extremo derecho se denota con Der(I).

Un intervalo I es *singular* si y sólo si es de la forma [a, a].

Dos intervalos I e I' son *adyacentes* si y sólo si

(2) Der(I) = Izq(I') e

(3) I está abierto por la derecha y I' está cerrado por la izquierda, ó I es cerrado por la derecha y I' abierto por la izquierda.

Notaciones:

- $\leq b$ y $>a$, denotan los intervalos $[0, b]$ y $(a, \infty)$, respectivamente;
- $<I$ denota el intervalo $\{t \mid$ para toda $t' \in I, 0 \leq t < t'\}$;

para $t \in R_{\geq 0}$

- $t+I$ denota el intervalo $\{t+t' \mid t' \in I\}$ ;
- $I-t$ denota el intervalo $\{t'-t \mid t' \in I$ y $t' \geq t\}$ ;
- $t \cdot I$ denota el intervalo $\{tt' \mid t' \in I\}$.

## 3.3. Secuencias temporizadas de estados

Sea PA un conjunto finito de proposiciones atómicas.

Un *estado* es un subconjunto de PA. Si $s \subseteq$ PA es un estado y $p \in s$, se escribe $s \vDash p$ y se dice que $s$ es un *p-estado*.

Una *secuencia de estados* $\underline{s} = s_0 s_1 s_2 \ldots$ es una secuencia infinita de estados $s_i \subseteq$ PA, para toda $i \geq 0$.

Una *secuencia de intervalos* $\underline{I} = I_0 I_1 I_2 \ldots$ es una secuencia infinita de intervalos de tiempo tal que cumple:

- *Inicio* - $I_0$ es cerrado en su extremo izquierdo e $Izq(I_0) = 0$;
- *Adyacencia* - para toda $i \geq 0$, los intervalos $I_i$ e $I_{i+1}$ son adyacentes;
- *Progreso* - cada tiempo $t \in R_{\geq 0}$ pertenece a algún intervalo $I_i$.

**Definición 3.3.1. (Secuencia temporizada de estados STE)**

Una *secuencia temporizada de estados* (STE) $\tau = (\underline{s}, \underline{I})$ es un par que consiste de una secuencia de estados $\underline{s}$ y una secuencia de intervalos $\underline{I}$.

La STE $\tau = (\underline{s}, \underline{I})$ se puede representar como:

$$\tau: (s_0, I_0) \rightarrow (s_1, I_1) \rightarrow (s_2, I_2) \rightarrow \cdots$$

Sea $\tau = (\underline{s}, \underline{I})$ una STE. Para toda $i \geq 0$ y $t \in I_i$, el estado $\tau^*(t)$ en el tiempo t es $s_i$.

Entonces, la STE $\tau$ puede ser vista como una función $\tau^*$ que va del tiempo con dominio $R_{\geq 0}$ a los estados $2^{PA}$, la cual provee un estado del sistema a cada instante de tiempo.

**Definición 3.3.2. (STEs equivalentes)**

Dos STEs $\tau_1$ y $\tau_2$ son *equivalentes* si y sólo si para todo $t \in R_{\geq 0}$, $\tau_1^*(t) = \tau_2^*(t)$.

**Definición 3.3.3. (Sufijo)**

Sea $\tau = (\underline{s}, \underline{I})$ una STE. Dado $t \in I_i$, el *sufijo* $\tau^t$ en el tiempo t es la STE:

$$\tau^t: (s_i, I_i - t) \rightarrow (s_{i+1}, I_{i+1} - t) \rightarrow (s_{i+2}, I_{i+2} - t) \rightarrow \cdots$$

**Ejemplo 3.3.4.**

Sea $\tau$ la siguiente STE:

$$\tau: \langle \varnothing, [0, 13) \rangle \rightarrow \langle \{p\}, [13, 13] \rangle \rightarrow \langle \varnothing, (13, 15) \rangle \rightarrow \langle \{q\}, [15, 20) \rangle \rightarrow$$
$$\langle \varnothing, [20, 40) \rangle \rightarrow \langle \{q\}, [40, 41) \rangle \rightarrow \langle \{q\}, [41, 42) \rangle \rightarrow \ldots$$

Entonces el sufijo en el tiempo 13.5 es

$$\tau^{13.5}: \langle \varnothing, [0, 1.5) \rangle \rightarrow \langle \{q\}, [1.5, 6.5) \rangle \rightarrow \langle \varnothing, [6.5, 26.5) \rangle \rightarrow$$
$$\langle \{q\}, [26.5, 27.5) \rangle \rightarrow \langle \{q\}, [27.5, 28.5) \rangle \rightarrow \ldots$$

## 3.4. Sintaxis y semántica

**Definición 3.4.1. (Sintaxis de LTIM)**

Las fórmulas de LTIM son definidas inductivamente por la siguiente gramática:

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \, \mathcal{U}_I \, \phi_2$$

Donde $p \in PA$, I no es un intervalo singular y tiene extremos enteros (I puede ser no acotado).

**Definición 3.4.2. (Semántica de LTIM)**

Para una fórmula LTIM $\phi$ y una STE $\tau=(\underline{s}, \underline{I})$, la relación de satisfacción $\tau \models \phi$ es definida inductivamente de la siguiente forma:

- $\tau \models p$ sii $s_0 \models p$, donde $p \in PA$;
- $\tau \models \neg\phi$ sii $\tau \not\models \phi$;
- $\tau \models \phi_1 \wedge \phi_2$ sii $\tau \models \phi_1$ y $\tau \models \phi_2$;
- $\tau \models \phi_1 \, \mathcal{U}_I \, \phi_2$ sii para algún $t \in I$, $\tau^t \models \phi_2$ , y para todo $t' \in (0, t)$, $\tau^{t'} \models \phi_1$.

La STE $\tau$ es un *modelo* de la fórmula $\phi$, ó $\tau$ *satisface* $\phi$, sii $\tau \models \phi$. Se escribe $L(\phi)$ para representar el conjunto de modelos de $\phi$ $\phi$ es *satisfacible* sii $L(\phi) \neq \varnothing$. Dos fórmulas $\phi$ y $\phi'$ son *equivalentes* sii $L(\phi) = L(\phi')$. El *problema de satisfacibilidad* para LTIM es decidir si una fórmula LTIM dada es satisfacible.

**Observación 3.4.3.**

Sea $\phi$ una fórmula LTIM. Si dos STEs $\tau_1$ y $\tau_2$ son equivalentes, entonces

$$\tau_1 \models \phi \text{ sii } \tau_2 \models \phi.$$

### 3.4.1. Operadores adicionales

- El operador $\Diamond_I \phi$ (*eventualmente* con tiempo restringido) es igual a $\top \mathcal{U}_I \phi$.
- $\Box_I \phi$ (*siempre* con tiempo acotado) es igual a $\neg \Diamond_I \neg \phi$.
- La fórmula con operador *libera*, $\phi_1 \mathcal{V}_I \phi_2$ es igual a $\neg(\neg\phi_1 \mathcal{U}_I \neg \phi_2)$.

Usualmente no se escribe el intervalo $(0, \infty)$ como subíndice. En tal caso, se emplean los operadores no estrictos siguientes:

- $\Diamond_{\geq 0} \phi := \phi \vee \Diamond \phi$;
- $\Box_{\geq 0} \phi := \phi \wedge \Box \phi$;
- $\phi_1 \mathcal{U}_{\geq 0} \phi_2 := \phi_2 \vee (\phi_1 \mathcal{U} \phi_2)$.

**Ejemplo 3.4.4.**

i) "Cada p-estado es seguido por un q-estado en a lo más 3 unidades de tiempo"

$$\Box_{\geq 0} (p \rightarrow \Diamond_{(0,3]} q)$$

ii) "La proposición p es verdadera en una infinidad de estados singulares y sólo en ellos"

$$\Diamond_{\geq 0} p \wedge \Box_{\geq 0} ( p \rightarrow \neg p \mathcal{U} p)$$

### 3.4.2. Decidibilidad para la LTIM

El problema de satisfacibilidad para la LTIM es decidible [5]. Esto es, se pueden crear algoritmos que logran determinar si una fórmula de la LTIM es satisfacible o no.

**Definición 3.4.5. (Constante que aparece en una fórmula)**

Decimos que una constante *c aparece* en la fórmula LTIM $\phi$ si y sólo si *c* es extremo de algún intervalo que aparece en $\phi$ , como subíndice de un operador $\mathcal{U}$.

**Teorema 3.4.6. (Decidibilidad para la LTIM) [5]**

El problema de satisfacibilidad para una fórmula $\phi$ de la LTIM puede ser decidido en tiempo $O(2^{NK \log(NK)})$, donde K-1 es la mayor constante que aparece en $\phi$, y N es el número de proposiciones, conectivos booleanos y operadores temporales de $\phi$.

### 3.4.3. LTIM$_=$

LTIM$_=$ es la extensión de LTIM que admite intervalos singulares. Por ejemplo, la propiedad: "Cada p-estado va seguido por un q-estado en 5 unidades de tiempo exactamente" se puede expresar con la fórmula LTIM$_=$

$$\Box(p \rightarrow \Diamond_{=5} q)$$

donde '=5' representa el intervalo singular [5, 5].

**Teorema 3.4.7. (Decidibilidad para LTIM$_=$ )** [5]

El problema de satisfacibilidad para LTIM$_=$ es indecidible.

**Observación 3.4.8.**

Aunque LTIM$_=$ es indecidible, para las fórmulas del tipo $(\neg\phi)\mathcal{U}_{=c}\phi$ se puede decidir si son satisfacibles. Esto se logra sustituyéndolas por sus equivalentes en LTIM:

$$(\neg\phi)\mathcal{U}_{=c}\phi \equiv (\Box_{(0,c)} \neg\phi) \wedge (\Diamond_{(0,c]}\phi)$$

# 3.5. Refinamiento de modelos

En la sección 3.8.5 se propone un algoritmo incremental para construir un autómata asociado a una fórmula LTIM; la fórmula es satisfacible siempre y cuando el lenguaje aceptado por su autómata asociado no es vacío. Este algoritmo extiende el trabajo de Brenda Casillas [9], el cual construye de manera incremental un autómata asociado a una fórmula LTLP

Lo esencial en un algoritmo incremental de la LTLP es la separación de restricciones del estado actual de las restricciones del resto de la secuencia de estados usando el operador $\bigcirc$. De igual manera en LTIM se introduce un operador $\bigcirc$. Para lograrlo se introduce el concepto de *refinamiento de modelos* explicado a continuación.

Todas las STEs cumplen con la condición de *variabilidad finita*: entre cualesquiera dos puntos del tiempo hay sólo una cantidad finita de cambios de estado [5]. Las STEs no sólo satisfacen la condición de variabilidad con respecto al valor de verdad de proposiciones, sino también con respecto al valor de verdad de toda fórmula LTIM. Esto se logra dividiendo los intervalos de un modelo $\tau$ en intervalos mas pequeños hasta que el valor de una fórmula $\phi$ se mantenga invariante en todos los intervalos. Este proceso es llamado refinamiento de $\tau$.

**Definición 3.5.1. (STE $\phi$-fina)**

Dada una fórmula LTIM $\phi$, la STE $\tau=(\underline{s}, \underline{I})$ es $\phi$-*fina* si y sólo si para todas las subfórmulas $\psi$ de $\phi$, para todo $I_i$ y para todos los t, t' $\in I_i$

$$\tau^t \vDash \psi \text{ si y sólo si } \tau^{t'} \vDash \psi.$$

Para una fórmula LTIM $\phi$, entonces, el valor de verdad de toda subfórmula de $\phi$ se mantiene invariante sobre todo intervalo de una STE $\phi$-fina.

**Lema 3.5.2. (STE $\phi$-fina equivalente)** [5]

Sea $\phi$ una fórmula LTIM. Para toda STE $\tau$, existe una STE $\phi$-fina equivalente a $\tau$.

De aquí en adelante y sin perdida de generalidad sólo se consideran STEs $\phi$-finas.

**Definición 3.5.3. (Sintaxis de LTIM$_{\bigcirc}$)**

Las fórmulas de LTIM$_{\bigcirc}$ son definidas inductivamente por la gramática:

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U}_I \phi_2 \mid \bigcirc\phi_1$$

donde $p \in PA$ e I no es un intervalo singular, tiene extremos enteros y puede ser no acotado.

**Definición 3.5.4. (Semántica de LTIM$_{\bigcirc}$ )**

Para una fórmula LTIM$_{\bigcirc}$ $\phi$ y una STE $\phi$-fina $\tau = (\underline{s}, \underline{I})$, la relación de satisfacción

$\tau \models \phi$ es definida inductivamente de la siguiente forma:

- $\tau \models p$ sii $s_0 \models p$;
- $\tau \models \neg \phi$ sii $\tau \not\models \phi$;
- $\tau \models \phi_1 \wedge \phi_2$ sii $\tau \models \phi_1$ y $\tau \models \phi_2$;
- $\tau \models \phi_1 \mathcal{U}_I \phi_2$ sii para algún $t \in I$ $\tau^t \models \phi_2$ y para toda $t' \in (0, t)$, $\tau^{t'} \models \phi_1$;
- $\tau \models \bigcirc\phi_1$ sii $\tau^{|I_0|} \models \phi_1$, donde $|I_0|$ es el valor de la diferencia $Der(I_0) - Izq(I_0)$.

## 3.6. $LTIM_{\leq}$

El problema de satisfacibilidad para una fórmula LTIM tiene complejidad temporal en el orden de dos exponenciales: una exponencial en el tamaño de la fórmula y otra exponencial en la mayor constante presente en la fórmula en cuestión [5].

Una restricción en la sintaxis de LTIM permite disminuir la complejidad del problema de satisfacibilidad para LTIM.

**Definición 3.6.1. (Sintaxis de $LTIM_{\leq}$)**

Las fórmulas de $LTIM_{\leq}$ son definidas inductivamente por la gramática:

$$\phi := p \mid \neg\phi_1 \mid \phi_1 \wedge \phi_2 \mid \phi_1 \mathcal{U}_{\leq d} \phi_2$$

Donde $p \in PA$, $d \in \mathbb{N}$ y $\leq d$ representa al intervalo $[0, d]$.

**Teorema 3.6.2. (Decidibilidad para la $LTIM_{\leq}$)** [5]

El problema de validez para una fórmula $\phi$ de la $LTIM_{\leq}$ puede ser decidido en tiempo exponencial sobre $N \cdot \log(N \cdot K)$, donde K-1 es la mayor constante que aparece en $\phi$, y N es el número de proposiciones, conectivos booleanos y operadores temporales de $\phi$.

## 3.7 Autómatas temporizados

Los *autómatas temporizados* son una extensión de los autómatas finitos no-deterministas sobre cadenas infinitas. Mientras los ω-autómatas aceptan secuencias de estados infinitas, los autómatas temporizados son adicionalmente restringidos por requerimientos de tiempo, para poder aceptar secuencias temporizadas de estados.

Un autómata temporizado opera con un conjunto finito de localidades y un conjunto finito de relojes (valuados sobre $R_{\geq 0}$). Cada localidad del autómata impone restricciones

sobre los valores de las proposiciones y sobre los valores de los relojes. El control del autómata puede residir en una localidad si y sólo si los valores de las proposiciones y relojes satisfacen la restricción correspondiente.

### 3.7.1. Valuaciones de reloj

Una *variable de reloj* ó simplemente reloj es una variable a la cual se le puede asignar algún valor sobre $R_{\geq 0}$, y siempre incrementa su valor para medir el tiempo que transcurre.

Una *restricción proposicional* es un conjunto de estados.

Una *restricción de reloj* $I \subseteq R_{\geq 0}$ es una unión de intervalos (posiblemente no acotados) con extremos enteros. Usualmente se denotan las restricciones de reloj para el reloj x como una combinación de expresiones aritméticas que contienen a x. Por ejemplo, se escribe $1 \leq x < 3 \vee x = 4 \vee x > 5$ para las restricciones de reloj $[1,3) \cup [4,4] \cup (5, 8)$.

La colección de restricciones de reloj sobre un conjunto de relojes X se denota con R(X).

Los valores de reloj son dados por *valuaciones de reloj*, las cuales son funciones de un conjunto finito de relojes X a $R_{\geq 0}$.

Dada una valuación de reloj $\nu$ y $t \in R_{\geq 0}$, se escribe $\nu + t$ para la valuación de reloj que asigna a cada reloj x de un conjunto de relojes X el valor $\nu(x) + t$.

Para un subconjunto $\gamma$ de un conjunto de relojes X, $\nu[\gamma:=0]$ denota la valuación de reloj que asigna cero a todos los relojes de $\gamma$, y $\nu(x)$ para los otros relojes $x \notin \gamma$.

$\mathcal{V}(X)$ denota el conjunto de valuaciones de reloj para los relojes en X.

**Definición 3.7.1. (Satisfacibilidad de restricciones de reloj)**

Se dice que una valuación de reloj $\nu$ para los relojes en X *satisface* la restricción de reloj *R*, y se escribe $\nu \models R$, si y sólo si para todos los relojes $x \in X$, $\nu(x) \in R$.

## 3.7.2. Corridas de autómatas temporizados

**Definición 3.7.2. (Autómata temporizado)**

Un *autómata temporizado* $\mathcal{A}$ es una tupla (V, $V^0$, α, X, β, E) con los siguientes componentes :

- V es un conjunto finito de localidades (de control).
- $V^0 \subseteq V$ es el conjunto inicial de localidades.
- α es la función de etiquetado, la cual asigna a cada localidad v una restricción proposicional $\alpha(v) \subseteq 2^{PA}$
- X es un conjunto finito de relojes.
- β es una función de etiquetado de localidades, la cual asigna a cada localidad v y a cada reloj x una restricción de reloj $\beta(v, x) \subseteq R_{\geq 0}$. También se usa β(v) para denotar a la familia de restricciones de reloj para una localidad v.
- $E \subseteq V \times V \times 2^X$ es un conjunto de transiciones. Cada transición (v, v' γ ) ∈ E, también denotada como $v \rightarrow^{\gamma} v'$ consiste de una localidad origen v, una localidad destino v' y un conjunto de relojes γ, los cuales son reiniciados con la transición.

**Ejemplo 3.7.3.**

La figura 3.1. muestra un ejemplo de autómata temporizado, con las localidades $v_i$, $0 \leq i \leq 4$, PA={p, q} y el conjunto de relojes X={x, y}.

**Definición 3.7.4. (Corrida de un autómata temporizado)**

Una *corrida* ρ de un autómata temporizado $\mathcal{A}$ =(V, $V^0$, α, X, β, E) es una secuencia infinita:

$$\xrightarrow[\sigma 0]{} (v_0, I_0) \xrightarrow[\sigma 1]{\gamma 1} (v_1, I_1) \xrightarrow[\sigma 2]{\gamma 2} (v_2, I_2) \xrightarrow[\sigma 3]{\gamma 3} \ldots$$

de localidades $v_i \in V$, intervalos $I_i$ que forman una secuencia de intervalos $\underline{I}_\rho = I_0 I_1 I_2 \ldots$ conjuntos de relojes $\gamma_i \subseteq X$, y valuaciones de relojes $\sigma_i \in \mathcal{V}(X)$, que satisface las propiedades siguientes:

- Inicial - $v_0 \in V^0$
- Consecutiva - Para toda $i \geq 0$ , $(v_i , v_{i+1} \gamma_{i+1}) \in E$,

  $\sigma_{i+1} = (\sigma_i + (Der(I_i) - Izq(I_i)))[\gamma_{i+1} := 0]$.

  Para toda $i \geq 0$ y $t \in I_i$ , la localidad $v_\rho(t)$ en el tiempo t es $v_i$ , y la valuación de reloj $\sigma_\rho(t)$ en el tiempo t es $\sigma_i + (t - Izq(I_i))$.
- Temporizada - Para toda $t \in R_{\geq 0}$ , $\sigma_\rho(t)$ satisface $\beta(v_\rho(t))$.

La corrida $\rho$ de un autómata temporizado $\mathcal{A}$ genera todas las STEs $\tau$ de la forma $(\underline{s}, \underline{I}_\rho)$ si para toda $t \in R_{\geq 0}$ , $\tau^*(t) \in \alpha(v_\rho(t))$.

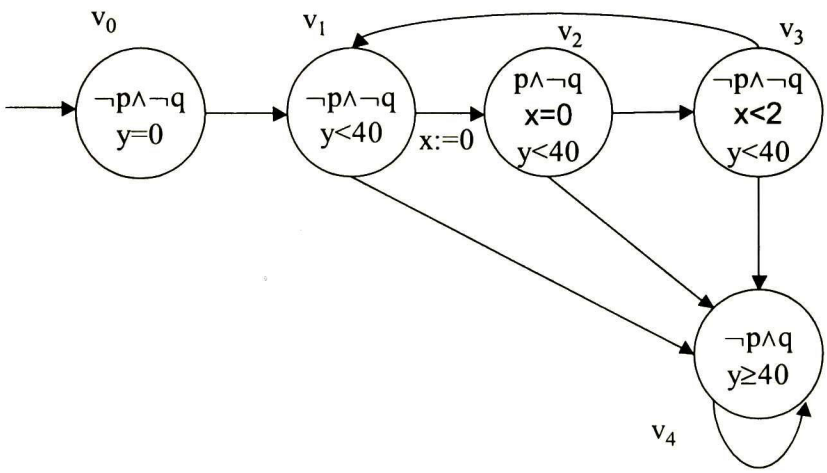


Figura 3.1. Ejemplo de un autómata temporizado.

El autómata $\mathcal{A}$ *acepta* la STE $\tau$ si y sólo si $\tau$ es equivalente a una STE que es generada por una corrida de $\mathcal{A}$. Se escribe $L(\mathcal{A})$ para representar el conjunto de STEs

aceptadas por $\mathcal{A}$. El *problema de vacuidad* para un autómata temporizado es decidir si acepta alguna STE.

# 3.8. Autómata temporizado asociado a una fórmula de la LTIM

En ésta sección se definen la LTIM$_{\leq}$ extendida propuesta por [13], la clausura de una fórmula, partículas temporizadas, aristas y otros elementos que permiten construir el autómata temporizado $\mathcal{A}_\phi$ correspondiente a una fórmula LTIM$_{\leq}$ (no extendida) $\phi$. El significado del autómata $\mathcal{A}_\phi$ para una fórmula $\phi$ es que cada modelo de $\phi$ corresponde a una corrida de $\mathcal{A}_\phi$.

## 3.8.1. LTIM$_{\leq}$ extendida

Se define la LTIM$_{\leq}$ extendida la cual permite expresar, además de fórmulas LTIM$_{\leq}$, reinicio de relojes y restricciones de reloj.

**Definición 3.8.1. (Sintaxis de LTIM$_{\leq}$ extendida)**

Las fórmulas de LTIM$_{\leq}$ son definidas inductivamente por la gramática:

$$\phi := \varphi \mid \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \phi_1 \, \mathcal{U}_{x \leq d} \, \phi_2 \mid \bigcirc\phi_1 \mid R \mid [\gamma{:=}0].\phi_1$$

donde $\varphi \in$ LTIM$_{\leq}$, $d \in \mathbb{N}$, $R \in R(X)$ y x pertenece a un conjunto de relojes X.

**Definición 3.8.2. (Semántica de LTIM$_{\leq}$ extendida)**

Para una fórmula LTIM$_{\leq}$ extendida $\phi$ y una STE $\phi$-fina $\tau=(\underline{s}, \underline{I})$ la relación $\tau \vDash_{\nu} \phi$ expresa cuando una STE $\tau$ satisface $\phi$ en el contexto de una valuación de reloj $\nu$:

- $\tau \models \nu\, \varphi$ sii $\tau \models \varphi$ ;
- $\tau \models \nu\, \neg\phi_1$ sii $\tau \not\models \nu\, \phi_1$ ;
- $\tau \models \nu\, \phi_1 \wedge \phi_2$ sii $\tau \models \nu\, \phi_1$ y $\tau \models \nu\, \phi_2$ ;
- $\tau \models \nu\, \phi_1\ \mathcal{U}_{x \leq d}\ \phi_2$ sii $\tau \models \nu\, \phi_2$ ó existe $t \in [0,d]$, tal que $\tau^t \models_{\nu(x)+t} \phi_2$ , y para toda $t' \in [0, t)$, $\tau^{t'} \models_{\nu(x)+t'} \phi_1$ ;
- $\tau \models \nu\, \bigcirc\phi_1$ sii $\tau^{|I_0|} \models_{\nu+|I_0|} \phi_1$ , donde $|I_0|$ es el valor de la diferencia $Der(I_0) - Izq(I_0)$ ;
- $\tau \models \nu\, R$ sii $\nu \models R$ ;
- $\tau \models \nu\, [\gamma:=0].\phi_1$ sii $\tau \models \nu_{[\gamma:=0]}\, \phi_1$

Las restricciones de reloj $R$ son interpretadas como la substitución de los relojes que ocurren en $R$ por sus valores según el ambiente temporizado $\nu$. La fórmula $[\gamma:=0].\phi_1$ asigna la valuación $\nu[\gamma:=0]$ a todo reloj de $\gamma$ que ocurre libremente en $\phi_1$.

### 3.8.2. Operadores adicionales

Al igual que en la LTIM, se introducen operadores eventual, siempre y libera para LTIM extendida:

- En a lo más 'd' unidades de tiempo, *eventualmente* $\phi$, $\Diamond_{x \leq d}\ \phi$, la cual es igual a $T\,\mathcal{U}_{x \leq d}\ \phi$;
- Durante 'd' unidades de tiempo, *siempre* $\phi$ sólo, $\Box_{x \leq d}\, \phi$, que es igual a $\neg\Diamond_{x \leq d}\ \neg\phi$;
- En a lo más 'd' unidades de tiempo, $\phi_1$ *libera* a $\phi_2$, $\phi_1\, \mathcal{V}_{x \leq d}\ \phi_2$, que es igual a $\neg(\neg\phi_1\ \mathcal{U}_{x \leq d}\ \neg\ \phi_2)$.

Para escribir una fórmula $LTIM_{\leq}$ extendida en forma normal positiva (fnp) se utilizan las siguientes reglas de reescritura:

1) $\neg\neg\phi \equiv \phi$;

2) $\neg(\phi_1 \vee \phi_2) \equiv \neg\phi_1 \wedge \neg\phi_2$;

3) $\neg(\phi_1 \wedge \phi_2) \equiv \neg\phi_1 \vee \neg\phi_2$;

4) $\neg\bigcirc\phi \equiv \bigcirc\neg\phi$ ;

5) $\neg(\phi_1 \, \mathcal{U}_{x\leq d} \, \phi_2) \equiv \neg\phi_1 \, \mathcal{V}_{x\leq d} \, \neg\phi_2$;

6) $\neg(\phi_1 \, \mathcal{V}_{x\leq d} \, \phi_2) \equiv \neg\phi_1 \, \mathcal{U}_{x\leq d} \, \neg\phi_2$;

7) $\neg\Diamond_{x\leq d}\,\phi \equiv \Box_{x\leq d}\,\neg\phi$;

8) $\neg\Box_{x\leq d}\,\phi \equiv \Diamond_{x\leq d}\,\neg\phi$;

9) $\neg[\gamma:=0].\phi_1 \equiv [\gamma:=0].\neg\phi_1$.

### 3.8.3. α-fórmulas y β-fórmulas de la $LTIM_{\leq}$ extendida

En las tablas 3.1. y 3.2. se muestran las α-fórmulas y β-fórmulas de la $LTIM_{\leq}$ extendida.

| α | K(α) |
|---|---|
| $\phi_1 \wedge \phi_2$ | $\phi_1, \phi_2$ |
| $[\gamma:=0].(\phi_1 \wedge \phi_2)$ | $[\gamma:=0].\phi_1 \quad [\gamma:=0].\phi_2$ |
| $[\gamma_1:=0].[\{x\}:=0].\phi$ | $[(\gamma_1 \cup \{x\}):=0].\phi$ |

Tabla 3.1. Definición de las α-fórmulas para la LTIM.

### 3.8.4. Clausura, partículas temporizadas y transiciones

**Definición 3.8.3. (Clausura)**

La *clausura* cl($\phi$) de una fórmula LTIM$_{\leq}$ extendida $\phi$ es el menor conjunto $\Phi$, tal que

- $\phi \in \Phi$ ;
- Para $p \in$ PA, si $[\gamma:=0].p \in \Phi$, entonces $[\gamma:=0].\neg p \in \Phi$ ;
- Si $[\gamma:=0].(\phi_1 \ \mathcal{U}_{x\leq d} \ \phi_2) \in \Phi$ entonces

$$\{ [\gamma:=0].(\phi_1 \wedge \bigcirc(\phi_1 \mathcal{U}_{x\leq d} \ \phi_2) \wedge (x<d)), [\gamma:=0].\phi_2 \} \subseteq \Phi ;$$

- Si $[\gamma:=0].(\phi_1 \ \mathcal{V}_{x\leq d} \ \phi_2) \in \Phi$ entonces

$$\{ [\gamma:=0].(\phi_2 \wedge \bigcirc(\phi_1 \ \mathcal{V}_{x\leq d} \ \phi_2)), [\gamma:=0].((\phi_2 \wedge \phi_1) \ \vee (x>d)) \} \subseteq \Phi ;$$

- Si $[\gamma:=0].( \phi_1 \vee \phi_2) \in \Phi$ entonces $\{ [\gamma:=0].\phi_1 \ \ [\gamma:=0].\phi_2 \} \subseteq \Phi$ ;
- Si $[\gamma:=0].( \phi_1 \wedge \phi_2) \in \Phi$ entonces $\{ [\gamma:=0].\phi_1 \ \ [\gamma:=0].\phi_2 \} \subseteq \Phi$ ;
- Si $[\gamma_1:=0].[\gamma_2:=0].\phi_1 \in \Phi$ entonces $[\gamma_1 \cup \gamma_2:=0].\phi_1 \in \Phi$, donde $\gamma_1$ y $\gamma_2$ son disjuntos.

| $\beta$ | $K_1(\beta)$ | $K_2(\beta)$ |
|---|---|---|
| $\phi_1 \vee \phi_2$ | $\phi_1$ | $\phi_2$ |
| $[\gamma:=0].(\phi_1 \vee \phi_2)$ | $[\gamma:=0].\phi_1$ | $[\gamma:=0].\phi_2$ |
| $[\gamma:=0].(\phi_1 \ \mathcal{U}_{x\leq d} \ \phi_2)$ | $[\gamma:=0].(\phi_1 \wedge \bigcirc(\phi_1 \mathcal{U}_{x\leq d} \ \phi_2) \wedge (x<d))$ | $[\gamma:=0].\phi_2$ |
| $[\gamma:=0].(\phi_1 \ \mathcal{V}_{x\leq d} \ \phi_2)$ | $[\gamma:=0].(\phi_2 \wedge \bigcirc(\phi_1 \ \mathcal{V}_{x\leq d} \ \phi_2))$ | $[\gamma:=0].((\phi_2 \wedge \phi_1) \vee (x>d))$ |
| $[\gamma:=0].\Diamond_{x\leq d} \ \phi$ | $[\gamma:=0].(\bigcirc\Diamond_{x\leq d} \ \phi \wedge (x<d))$ | $[\gamma:=0].\phi$ |
| $[\gamma:=0].\Box_{x\leq d} \ \phi$ | $[\gamma:=0].(\phi \wedge \bigcirc\Box_{x\leq d} \ \phi)$ | $[\gamma:=0].(x>d)$ |

Tabla 3.2. Definición de las $\beta$-fórmulas para la LTIM.

**Definición 3.8.4. (Partícula temporizada)**

Una *partícula temporizada* sobre una fórmula $LTIM_{\leq}$ extendida $\phi$ es un subconjunto maximalmente consistente de $cl(\phi)$.

Un subconjunto $P$ de $cl(\phi)$ es *maximalmente consistente* si satisface las siguientes condiciones:

- $[\gamma:=0].\bot \notin P$ ;
- Para toda $p \in PA$ , si $[\gamma:=0].p \in P$ entonces $[\gamma:=0].\neg p \notin P$ ;
- Para toda $\alpha$-fórmula $\psi \in cl(\phi)$, $\psi \in P$ sii $K(\psi) \subseteq P$ ;
- Para toda $\beta$-fórmula $\psi \in cl(\phi)$, $\psi \in P$ sii $K_1(\psi) \subseteq P$ ó $K_2(\psi) \subseteq P$.

**Definición 3.8.5. (Relojes activos)**

Para una partícula temporizada $P$ sobre la fórmula $LTIM_{\leq}$ extendida $\phi$, $\gamma(P)$ denota el menor conjunto de relojes $r$, tal que si $[r':=0].\varphi \in P$ entonces $r' \subseteq r$, para alguna subfórmula $\varphi$ de $\phi$. Se llama a $\gamma(P)$ el conjunto de *relojes activos* en $P$

**Definición 3.8.6. (Partícula temporizada vacía)**

La *partícula temporizada vacía* $P_{\varnothing}$ es la partícula temporizada que no contiene fórmulas.

**Definición 3.8.7. (Transición)**

Existe una *transición* de la partícula temporizada $\Phi$ a la partícula temporizada $\Psi$, si y sólo si para toda fórmula $[\gamma:=0].\bigcirc\varphi$ que está en $cl(\phi)$

$$[\gamma:=0].\bigcirc\varphi \in \Phi \text{ sii } [\gamma:=0].\varphi \in \Psi.$$

**Definición 3.8.8. (Fórmulas implicadas)**

Si $P$ es una partícula temporizada, entonces $[\gamma:=0].\varphi$ es una *fórmula implicada* por $P$ si y sólo si $[\gamma:=0].\bigcirc\varphi \in P$

Se denota como *imps*($P$) al conjunto de fórmulas implicadas por la partícula temporizada $P$.

**Definición 3.8.9. (Conjunto atómicamente consistente, conjunto α-cerrado, conjunto β-cerrado)**

Sea B un subconjunto de cl($\phi$). Se dice que B es:

i. *Atómicamente consistente*, si no contiene la constante $\perp$ (o $\neg\top$ ), ni contiene una proposición atómica y su negación.

ii. α-*cerrado*, si para toda α-fórmula $\psi$ en cl($\phi$), $\psi \in B$ sii $K(\psi) \subseteq B$.

iii. β-*cerrado*, si para toda fórmula $\psi$ en cl($\phi$), $\psi \in B$ sii $K_1(\psi) \subseteq B$ ó $K_2(\psi) \subseteq B$ .

Para obtener un conjunto β-cerrado, que contenga un conjunto B dado, se aplica de ser necesario una de las operaciones siguientes a cada β-fórmula $\psi$ en cl($\phi$):

iv. β-*expansión*, en la cual se añade a B los elementos de uno de los conjuntos $K_1(\psi)$ ó $K_2(\psi)$, siempre que $\psi \in B$;

v. $\beta^{-1}$-*expansión*, en la cual se añade $\psi$ a B, si al menos uno de los conjuntos correspondientes $K_1(\psi)$ y $K_2(\psi)$ está contenido en B.

vi. La α-*cerradura* de B es el menor subconjunto α-*cerrado* de cl($\phi$) que contiene a B y se denota con $\alpha(B)$.

Para construir la α-cerradura de un conjunto B se aplican dos tipos de operaciones a cada α-fórmula $\psi$ en cl($\phi$):

vii. α-*expansión*, en la cual se añaden a B los elementos del conjunto $K(\psi)$, si $\psi \in B$;

viii. $\alpha^{-1}$-*expansión*, en la cual se añade $\psi$ a B, si su conjunto correspondiente $K(\psi)$ está contenido en B.

ix. Una β-fórmula $\psi$ está *apoyada por un conjunto* X (contenido en cl($\phi$)) sii

$$K_1(\psi) \subseteq X \text{ ó } K_2(\psi) \subseteq X$$

**Definición 3.8.10. (Función Cubierta$_P$)**

La *función Cubierta$_P$*, de la colección $2^{cl(\phi)}$ de los subconjuntos de $cl(\phi)$ en la colección $2^{2^{cl(\phi)}}$ de colecciones de subconjuntos de $cl(\phi)$, está dada recurrentemente por las reglas siguientes:

- Cubierta$_P(\varnothing)$ := Cubierta$_P(\{T\})$:= $\{P_\varnothing\}$;
- Cubierta$_P(X)$ := $\varnothing$, si X no es atómicamente consistente;
- Cubierta$_P(X)$ := Cubierta$_P(\alpha(X))$, si X no es $\alpha$-cerrado;
- Cubierta$_P(X)$ := Cubierta$_P(X \cup \{\psi\})$, si existe una $\beta$-fórmula $\psi$ en $cl(\phi)$ talque $\psi \notin X$, pero $K_1(\psi) \subseteq X$ ó $K_2(\psi) \subseteq X$, ($\beta^{-1}$-expansión);
- Cubierta$_P(X)$ := Cubierta$_P(X \cup K_1(\psi)) \cup$ Cubierta$_P(X \cup K_2(\psi))$, si existe una $\beta$-fórmula $\psi$ en X que no está apoyada por X , ($\beta$-expansión);
- Cubierta$_P(X)$ := $\{X\}$, de otro modo.

**Definición 3.8.11. (Introducción de relojes)**

Al *introducir el reloj* $x_\phi$ en la fórmula $LTIM_{\leq}$ $\phi$ que sea de la forma $\phi_1 \mathcal{U}_{\leq d} \phi_2$, $\phi_1 \mathcal{V}_{\leq d} \phi_2$, $\Diamond_{\leq d} \phi$ ó $\Box_{\leq d} \phi$, se obtiene la fórmula $LTIM_{\leq}$ extendida $[\{x_\phi\}:=0].(\phi_1 \mathcal{U}_{x\leq d} \phi_2)$, $[\{x_\phi\}:=0].(\phi_1 \mathcal{V}_{x\leq d} \phi_2)$, $[\{x_\phi\}:=0].\Diamond_{x\leq d} \phi$ ó $[\{x_\phi\}:=0].\Box_{x\leq d} \phi$ respectivamente, y para otro tipo de fórmulas se obtiene la misma fórmula $\phi$.

## 3.8.5. Construcción incremental del autómata temporizado $A_\phi$

Se propone el siguiente algoritmo incremental para construir el autómata temporizado asociado $A_\phi = (V, V^0, \alpha, X, \beta, E)$ para una fórmula $LTIM_{\leq}$ (no extendida) $\phi$ en fnp.

**Algoritmo 3.8.12.**

1) Calcular $\phi$', talque $\phi$' resulta de introducir relojes en toda subfórmula de $\phi$.
2) Se obtiene cl($\phi$').
3) Se obtiene Cubierta$_P$({$\phi$'}), las partículas temporizadas resultantes, se agregan al conjunto V, al igual que al conjunto de partículas temporizadas iniciales $V^0$
4) Para cada partícula temporizada inicial $P$, se agrega la transición inicial
   $(\varnothing, P, \gamma(P))$ al conjunto de transiciones E.
5) Para cada $P \in V$ a la que no se haya calculado sus partículas temporizadas sucesoras, entonces calcular dichas partículas temporizadas sucesoras, de acuerdo a la siguiente función:
   $$Sucesores(P) = \text{Cubierta}_P\ (\ imps(P)\ )$$
6) Si $Q \in Sucesores(P)$ y $Q \notin V$, entonces Q se agrega al conjunto V. Además, si $\phi' \in Q$ y $Q \notin V^0$, entonces Q se agrega al conjunto $V^0$
7) Para cada $Q \in Sucesores(P)$ agregar la transición $(P, Q, \gamma_2)$ a E, tal que
   $\gamma_2 = \gamma(Q) - \gamma(P)$.
8) Si existe alguna partícula temporizada de V a la cual no se le hayan calculado sus partículas temporizadas sucesoras, entonces ir al paso 5), de lo contrario, ir al paso 9).
9) Se ha terminado la construcción de $A_\phi$.

**Ejemplo 3.8.13.**

Se aplica el algoritmo incremental para obtener el autómata $A_\phi = (V, V^0, \alpha, X, \beta, E)$ asociado a la fórmula:

$$\phi = \Box_{\leq 9}(p \rightarrow \Diamond_{\leq 2}\ q) \text{ donde } p, q \in PA.$$

Paso 1. $\phi = \phi' = [\{x\}:=0].\Box_{x\leq 9}(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\ q)$.

Paso 2. cl($\phi'$) = { $\phi'$ , $p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\ q$, p, $[\{y\}:=0].\Diamond_{y\leq 2}\ q$, q , $\neg p$, $\neg q$}

Paso 3. Se calcula Cubierta$_P$({$\phi'$}), creando el árbol semántico para $\phi'$ mostrado en la figura 3.2.

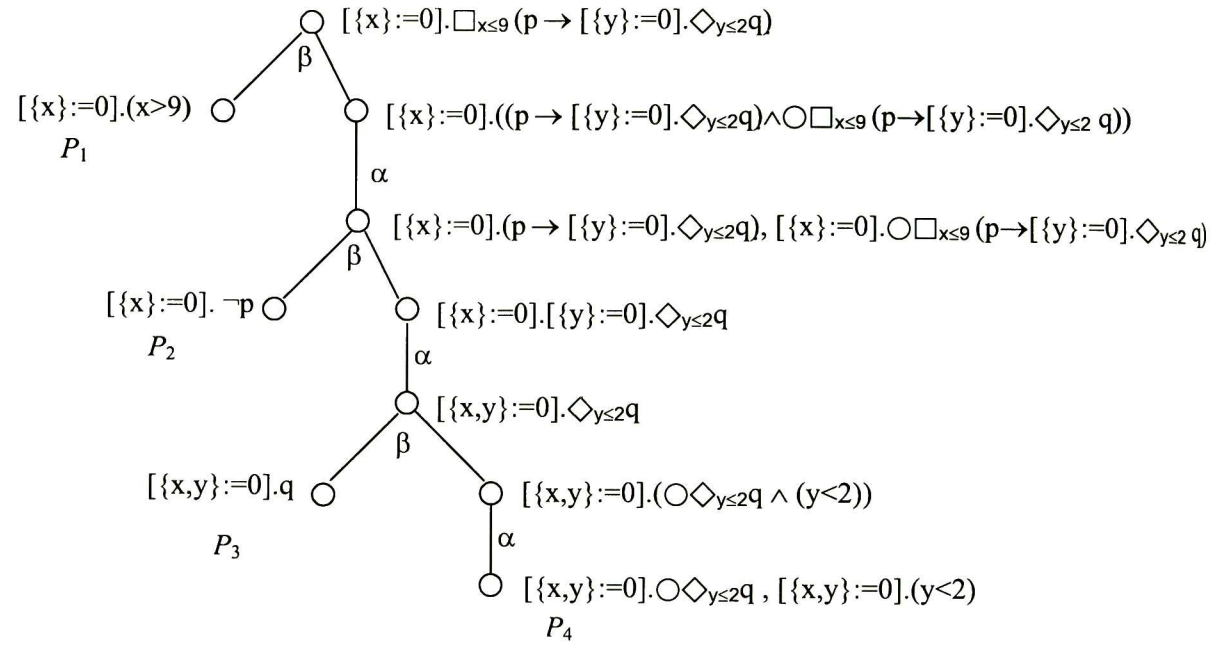


Figura 3.2. Árbol semántico para calcular Cubierta$_P$({ϕ'})

Luego, Cubierta$_P$({ϕ'}) = {$P_1$, $P_2$, $P_3$, $P_4$}, donde cada $P_i$ contiene todas las fórmulas que están sobre la misma rama del árbol, la cual va de la raíz al nodo terminal indicado por $P_i$

$P_1$ = { [{x}:=0].□x≤9 (p → [{y}:=0].◇y≤2q) , [{x}:=0].(x>9) };

$P_2$ = { [{x}:=0].□x≤9 (p → [{y}:=0].◇y≤2q) , [{x}:=0].((p → [{y}:=0].◇y≤2q)∧○□x≤9 (p→[{y}:=0].◇y≤2 q)) [{x}:=0].(p → [{y}:=0].◇y≤2q), [{x}:=0].○□x≤9 (p→[{y}:=0].◇y≤2 q) [{x}:=0]. ¬p } ;

$P_3$ = { [{x}:=0].□x≤9 (p → [{y}:=0].◇y≤2q) , [{x}:=0].((p → [{y}:=0].◇y≤2q)∧○□x≤9 (p→[{y}:=0].◇y≤2 q)) , [{x}:=0].(p → [{y}:=0].◇y≤2q), [{x}:=0].○□x≤9 (p→[{y}:=0].◇y≤2 q) , [{x}:=0].[{y}:=0].◇y≤2q , [{x,y}:=0].◇y≤2q , [{x,y}:=0].q } ;

$P_4$ = { $[\{x\}:=0].\Box_{x\leq 9}(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}q)$ . $[\{x\}:=0].((p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}q) \wedge \bigcirc\Box_{x\leq 9}$
$(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\, q))$ , $[\{x\}:=0].(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}q)$, $[\{x\}:=0].\bigcirc\Box_{x\leq 9}$
$(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\, q)$ , $[\{x\}:=0].[\{y\}:=0].\Diamond_{y\leq 2}q$ $[\{x,y\}:=0].\Diamond_{y\leq 2}q$
$[\{x,y\}:=0].(\bigcirc\Diamond_{y\leq 2}q \wedge (y<2))$ . $[\{x,y\}:=0].\bigcirc\Diamond_{y\leq 2}q$ , $[\{x,y\}:=0].(y<2)\}$ ;

Hacer V:= { $P_1$, $P_2$, $P_3$, $P_4$ } y $V^0$:= V.

Paso 4. Agregar a E las transiciones iniciales:

E:= $(\varnothing, P_1, \{x\})$, $(\varnothing, P_2, \{x\})$, $(\varnothing, P_3, \{x,y\})$ , $(\varnothing, P_4, \{x,y\})$

Pasos 5,6 y 7. Calcular los sucesores de cada partícula de V=$\{P_1, P_2, P_3, P_4\}$.

Sucesores($P_1$):= Cubierta*p*(*Imps*($P_1$)) = Cubierta*p*($\varnothing$) = { $P_\varnothing$ }.

Agregar $P_\varnothing$ a V; agregar la transición ($P_1$, $P_\varnothing$, {}) a E.

Sucesores($P_2$):= Cubierta*p*(*Imps*($P_2$)) =

Cubierta*p*({$[\{x\}:=0].\Box_{x\leq 9}(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\, q)$ }) = $\{P_1, P_2, P_3, P_4\}$.

Agregar a E las transiciones:

($P_2$, $P_1$, {}), ($P_2$, $P_2$, {}), ($P_2$, $P_3$, {y}) y ($P_2$, $P_4$, {y}).

Sucesores($P_3$):= Cubierta*p*(*Imps*($P_3$)) =

Cubierta*p*({$[\{x\}:=0].\Box_{x\leq 9}(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\, q)$ }) = $\{P_1, P_2, P_3, P_4\}$.

Agregar a E las transiciones:

($P_3$, $P_1$, {}), ($P_3$, $P_2$, {}), ($P_3$, $P_3$, {}) y ($P_3$, $P_4$, {}).

Sucesores($P_4$):= Cubierta*p*(*Imps*($P_4$)) =

Cubierta*p*({$[\{x\}:=0].\Box_{x\leq 9}(p \rightarrow [\{y\}:=0].\Diamond_{y\leq 2}\, q)$, $[\{x,y\}:=0].\Diamond_{y\leq 2}q$ }) =

$\{P_3, P_4, P_5, P_6\}$;

Se generaron algunas nuevas partículas temporizadas, agregar $P_5$ y $P_6$ a V donde

$P_5 = P_1 \cup \{[\{x,y\}:=0].q\}$ ;

- Para cada reloj $x_1 \in X_1$ la restricción de reloj $\beta((v_1, v_2), x_1)$ es $\beta_1(v_1, x_1)$ y para cada reloj $x_2 \in X_2$ la restricción de reloj $\beta((v_1, v_2), x_2)$ es $\beta_2(v_2, x_2)$ ;
- Para cada par de transiciones $v \xrightarrow{\gamma_1} v'$ y $w \xrightarrow{\gamma_2} w'$ de $\mathcal{A}_1$ y $\mathcal{A}_2$ respectivamente, $\mathcal{A}_1 \times \mathcal{A}_2$ tiene tres transiciones: $(v,w) \xrightarrow{\gamma_1 \cup \gamma_2} (v',w')$, $(v,w) \xrightarrow{\gamma_1} (v',w)$ y $(v,w) \xrightarrow{\gamma_2} (v,w')$. Así, el conjunto de transiciones E simula la ejecución conjunta de los dos autómatas componentes.

**Ejemplo 3.9.2.**

La figura 3.4. Muestra un ejemplo del producto de dos autómatas temporizados con PA={p,q}.

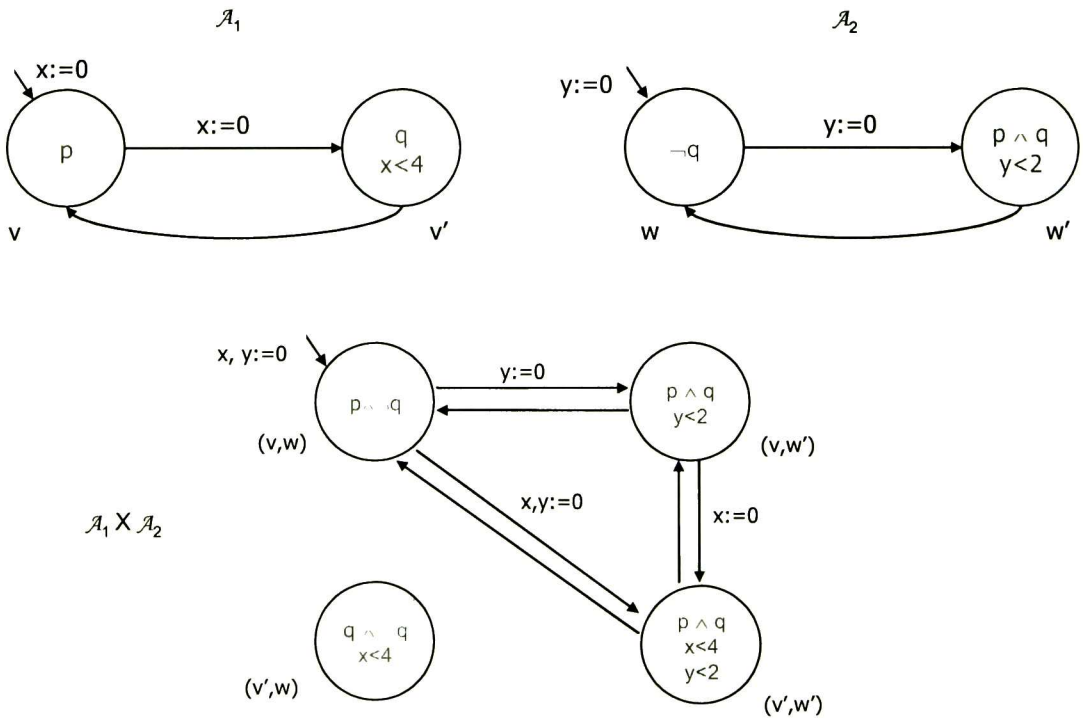


Figura 3.4. Ejemplo del producto de dos autómatas temporizados.

### 3.9.1. Autómata de región

En cada punto del tiempo el comportamiento de un autómata temporizado está determinado por su estado y los valores de sus relojes. Esto motiva a la siguiente definición.

**Definición 3.9.3. (Estado extendido)**

Para un autómata temporizado $\mathcal{A}$ = (V, $V^0$, α, X, β, E) un *estado extendido* es una pareja <s,*v*>, donde s∈V y *v* es una valuación de reloj.

Para un autómata temporizado $\mathcal{A}$ no se podría construir otro autómata tal que sus estados sean estados extendidos de $\mathcal{A}$ ya que el número de tales estados extendidos es infinito. Pero, si dos estados extendidos <s,*v*> y <s,*v*'> concuerdan en las partes enteras de los valores de los relojes de *v* y *v*' y también en el orden de las partes fraccionarias de los valores de los relojes, entonces las corridas que inicien desde esos dos estados extendidos serán muy similares. Por esta razón, se pueden agrupar estados extendidos similares en una sola clase de equivalencia.

**Definición 3.9.4. (Región de reloj)**

Sea $\mathcal{A}$ = (V, $V^0$, α, X, β, E) un autómata temporizado. Para cualquier $t \in R_{\geq 0}$, fracc(*t*) denota la parte fraccionaria de *t*, y $\lfloor t \rfloor$ denota la parte entera de *t*. Para cada x∈X, sea $c_x$ la constante más grande *c* tal que *x* es comparado con *c* en alguna restricción de reloj que aparezca en β.

La relación de equivalencia ~ se define sobre el conjunto de valuaciones de reloj $\mathcal{V}$(X):

Para cualesquiera v, v' ∈ $\mathcal{V}$(X), v ~ v' si y solo si se cumplen todas las condiciones siguientes:

1. Para todo $x \in X$, ya sea que $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ o ambas son mayores a $c_x$.
2. Para todos los x, $y \in X$ con $v(x) \leq c_x$ y $v(y) \leq c_y$,

   $$fracc(v(x)) \leq fracc(v(y)) \text{ sii } fracc(v'(x)) \leq fracc(v'(y)).$$

3. Para todo $x \in X$ con $v(x) \leq c_x$, $fracc(v(x)) = 0$ sii $fracc(v'(x)) = 0$.

Una *región de reloj* para $\mathcal{A}$ es una clase de equivalencia de las valuaciones de reloj inducidas por ~. Se denota con $[v]$ la región de reloj a la cual pertenece la valuación de reloj $v$.

Se observa que hay un número finito de regiones para un autómata temporizado. También se observa que para una restricción de reloj $\delta$ para $\mathcal{A}$, si $v \sim v'$ entonces v satisface $\delta$ sii v' satisface $\delta$. Cada región puede ser representada por:

(1) Para cada reloj x, se tiene una restricción de reloj del conjunto

$$\{x = c \mid c = 0, 1, \ldots c_x\} \cup \{c-1 < x < c \mid c = 1, \ldots c_x\} \cup \{x > c_x\},$$

(2) Para cada par de relojes x e y tales que $c-1 < x < c$ y $d-1 < y < d$ que aparezcan en (1), para algunos c, d, ya sea que fracc(x) es menor, igual o mayor a fracc(y).

**Definición 3.9.5. (Transición extendida)**

Sean $\langle s, v \rangle$, $\langle s' \ v' \rangle$ dos estados extendidos de un autómata temporizado $\mathcal{A} = (V, V^0, \alpha, X, \beta, E)$. Existe la transición extendida $\langle s, v \rangle \rightarrow \langle s' \ v' \rangle$ si y sólo si existe un incremento $t \in R_{\geq 0}$ y una transición $(s, s' \ \gamma) \in E$ tal que $v' = (v+t)[\gamma := 0]$

**Definición 3.9.6. (Región sucesora)**

Sean $\alpha$ y $\beta$ dos regiones de reloj distintas sobre los estados extendidos de un autómata temporizado. La región $\beta$ se dice que es *sucesora* de $\alpha$, se denota sucesora($\alpha$)=$\beta$, si y sólo si para cada $v \in \alpha$, existe un $t \in R_{\geq 0}$ tal que $v + t \in \beta$, y $v + t' \in \alpha \cup \beta$ para todo $t' < t$.

**Definición 3.9.7. (Autómata de región)**

Sea $\mathcal{A} = (V, V^0, \alpha, X, \beta, E)$ un autómata temporizado, el correspondiente autómata de región $R(\mathcal{A})$ es el autómata de estados finito tal que:

- Los estados de R($\mathcal{A}$) son de la forma <s, $\alpha$> donde s $\in$ V y $\alpha$ es una región de reloj.
- Los estados iniciales son de la forma <$s^0$, [$v^0$]>, donde $s^0 \in V^0$ y $v^0(x)=0$ para $x \in X$.
- R($\mathcal{A}$) tiene dos tipos de aristas:
  1. Las que representan el paso del tiempo <s, $\alpha$> $\rightarrow$ <s, $\beta$>, donde sucesora($\alpha$) = $\beta$.
  2. Las que representan las transiciones extendidas de $\mathcal{A}$:
     <s, $\alpha$> $\rightarrow$ <s', $\alpha$'> sii existe <s, $v$> $\rightarrow$ <s' $v$'>, tal que $v \in \alpha$ y $v' \in \alpha'$

**Ejemplo 3.9.8.**

En la figura 3.5. se tiene un autómata temporizado $\mathcal{A}$, y en la figura 3.6. se tiene el autómata de región R($\mathcal{A}$) correspondiente.

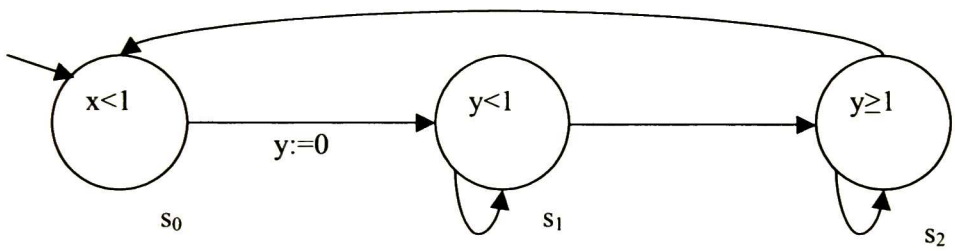


Figura 3.5. Autómata temporizado $\mathcal{A}$.

El algoritmo para determinar la vacuidad del lenguaje del autómata de región R($\mathcal{A}$), para el autómata temporizado $\mathcal{A}$ = (V, $V^0$, $\alpha$, X, $\beta$, E), busca un ciclo de aceptación tal que,

(1) sea accesible desde algún estado inicial de R($\mathcal{A}$) y

(2) satisface la condición de progreso: para todo x $\in$ X, el ciclo contiene al menos una región que satisface [(x = 0) $\vee$ (x > $c_x$)].

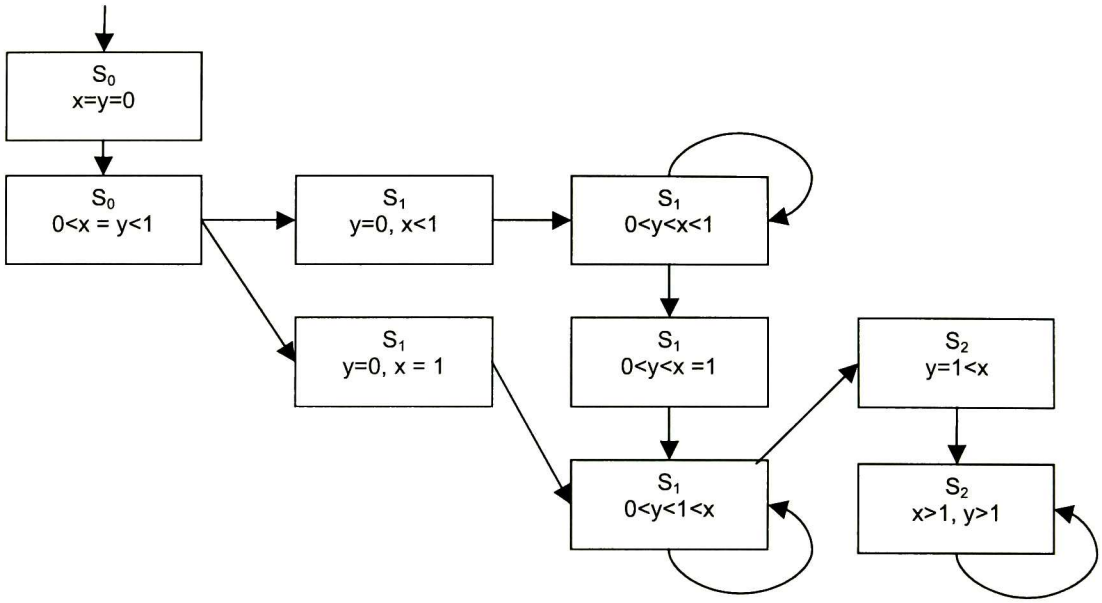


Figura 3.6. Autómata de región R($\mathcal{A}$).

Los pasos para verificar una propiedad temporizada, escrita en LTIM, en un sistema de tiempo real por comprobación de modelos temporizada son los siguientes:

1. Obtener el autómata temporizado asociado a la negación de la fórmula $\phi$, la cual representa a la propiedad. Esto es, se calcula $A_{\neg\phi}$.
2. Calcular el producto de los autómatas de la negación de la propiedad y del modelo abstracto del sistema de tiempo real *S*. Esto es, se calcula el autómata temporizado $S \times A_{\neg\phi}$.
3. Se realiza una abstracción temporizada del autómata $S \times A_{\neg\phi}$, es decir, se obtiene el autómata de región $R(S \times A_{\neg\phi})$.
4. Se realiza la búsqueda de ciclos de aceptación en $R(S \times A_{\neg\phi})$.
5. Si se encuentra un ciclo de aceptación, se tiene un contraejemplo. Si no, entonces se afirma que la negación de la propiedad no se cumple en el sistema.

Cabe destacar que, obtener el autómata de región de un autómata temporizado tiene una complejidad exponencial en la cantidad de relojes presentes en el autómata temporizado [2]. Otra manera de hacer una abstracción temporizada, de un autómata temporizado, es calculando el autómata de zonas. Este tipo de autómata tiende a ser más pequeño que el autómata de región, esto es, posee menos estados [2].

# Capítulo 4

# Casos de Estudio

## 4.1. Contenido del capítulo

Este capítulo presenta cuatro casos de estudio sobre la verificación de sistemas de tiempo real. La sección 4.2 expone la sintaxis de la TCTL. La sección 4.3 expone el caso de estudio de un controlador de la compuerta de un tren. La sección 4.4 expone el caso de estudio de un controlador de semáforos. La sección 4.5 expone el caso de estudio del protocolo de exclusión mutua de Fischer. Por último, la sección 4.6 expone el caso de estudio del Protocolo de comunicaciones CSMA/CD.

En estos cuatro casos de estudio se utiliza la herramienta de verificación Kronos [11, 26, 8, 15]. El protocolo de exclusión mutua de Fischer se verifica con las herramientas Kronos y Uppaal [27, 17, 14, 6, 8]. Kronos permite capturar autómatas temporizados, los

cuales representan los modelos de los sistemas de tiempo real, y calcular la composición paralela de dos ó más autómatas temporizados.

En cada caso de estudio presentado en este capítulo las propiedades temporizadas a verificar se expresan como fórmulas LTIM, sin embargo, se convierten a fórmulas equivalentes de la Lógica Temporal Ramificada Temporizada TCTL[4]; esto se debe a que las herramientas existentes de verificación automática de sistemas de tiempo real sólo aceptan TCTL como lenguaje de especificación. Se observará que es más fácil e intuitivamente más claro el proceso de formular propiedades temporizadas con la LTIM.

## 4.2. Sintaxis de la TCTL

Sea $\mathbb{N}$ el conjunto de constantes {0, 1, 2, ...} denotando a los números naturales.

Se utiliza el símbolo ~ para expresar alguna de las relaciones binarias $<, \geq, =, \leq$, ó $>$.

**Definición 4.2.1 (Sintaxis TCTL)**

Las fórmulas $\phi$ de TCTL son definidas inductivamente como sigue:

$$\phi := p \mid \bot \mid \phi_1 \rightarrow \phi_2 \mid \exists(\phi_1 \, U_{\sim c} \, \phi_2) \mid \forall(\phi_1 \, U_{\sim c} \, \phi_2)$$

donde $p \in PA$ y $c \in \mathbb{N}$.

Informalmente, $\exists(\phi_1 \, U_{\sim c} \, \phi_2)$ significa que en algún cómputo, que parte de algún estado inicial, se cumple $\phi_1 \, U \, \phi_2$ en a lo más $c$ unidades de tiempo; es decir, la trayectoria tiene un estado donde se cumple $\phi_2$ y mientras tanto $\phi_1$ se cumple en todos los estados intermedios. $\forall(\phi_1 \, U_{\sim c} \, \phi_2)$ significa que en todos los cómputos, que parten de algún estado inicial, se cumple $\phi_1 \, U \, \phi_2$ en a lo más $c$ unidades de tiempo.

### 4.2.1. Fórmulas LTIM equivalentes a fórmulas TCTL

Si se considera que una fórmula LTLP se cumple en todas las trayectorias sobre estructuras ramificadas. Entonces, muchas fórmulas LTLP tienen una fórmula equivalente en CTL [12].

LTIM y TCTL son extensiones de tiempo cuantificado, sobre un dominio continuo, de LTLP y CTL respectivamente; la extensión a tiempo cuantificado de LTIM y TCTL es idéntica pues ambas incluyen en sus sintaxis la restricción de tiempo en el operador $\mathcal{U}$ (hasta que).

Se observa entonces, que muchas fórmulas LTIM son equivalentes a fórmulas TCTL.

## 4.3. Controlador de la compuerta del tren

La figura 4.1 muestra un cruce en una vía de tren, se tiene una compuerta para detener el tráfico de automóviles mientras el tren pasa por el cruce. También se tienen dos sensores *aprox* y *salida* que respectivamente le indican al controlador de la compuerta que el tren se está aproximando y que el tren ya está lejos del cruce.

Para simplificar, se asume que la unidad de tiempo es un minuto. Se sabe que cuando el sensor *aprox* detecta al tren, este llegará al cruce en mas de dos minutos. Una vez que llega al cruce, el tren tarda en recorrer la sección *dentro,* a lo más en tres minutos.

Cuando el sensor *aprox* detecta al tren, se envía una señal de aproximación al controlador, el controlador espera un minuto y envía la orden de cerrar a la compuerta.

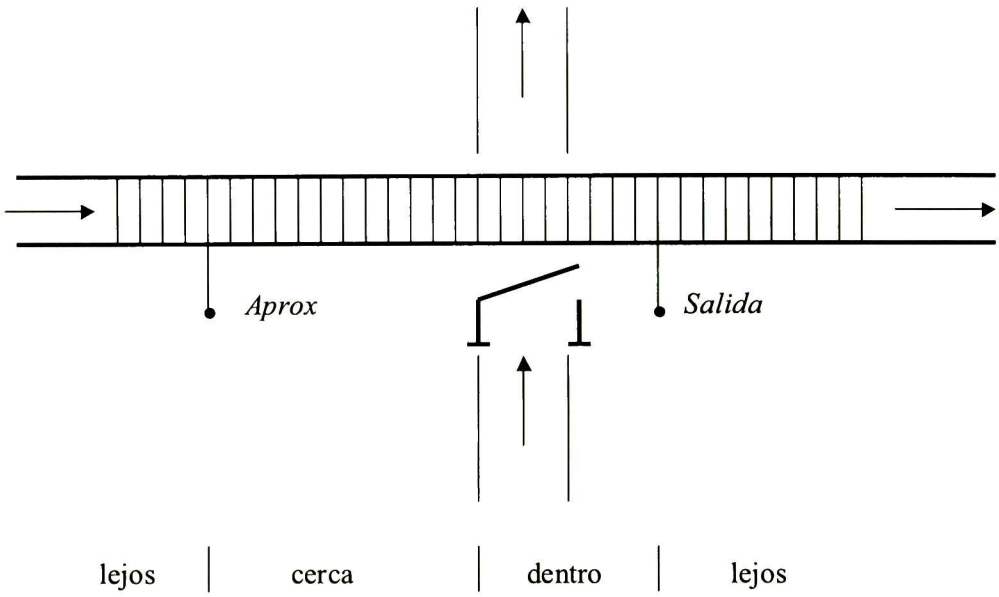


Figura 4.1. Cruce a una vía de tren.

Cuando la compuerta recibe la orden de cerrarse, ésta tarda menos de un minuto en cerrarse, y si recibe orden de abrirse tardará de uno a dos minutos.

Una vez que el sensor salida detecta que el tren está lejos se envía señal de aviso al controlador, el cual espera un minuto y envía orden de abrir a la compuerta.

Este sistema de tiempo real se modela con tres autómatas temporizados sincronizados. Véase figura 4.2. El modelo de este caso de estudio también se puede consultar en [2]. Cada uno de los tres autómatas temporizados modela el comportamiento del tren, el controlador y la compuerta respectivamente. El autómata del tren y el autómata del controlador están sincronizados con las transiciones marcadas con las señales *aprox* y *salida*. A su vez, los autómatas de la compuerta y del controlador se sincronizan con las transiciones marcadas con las señales *abrir* y *cerrar*.

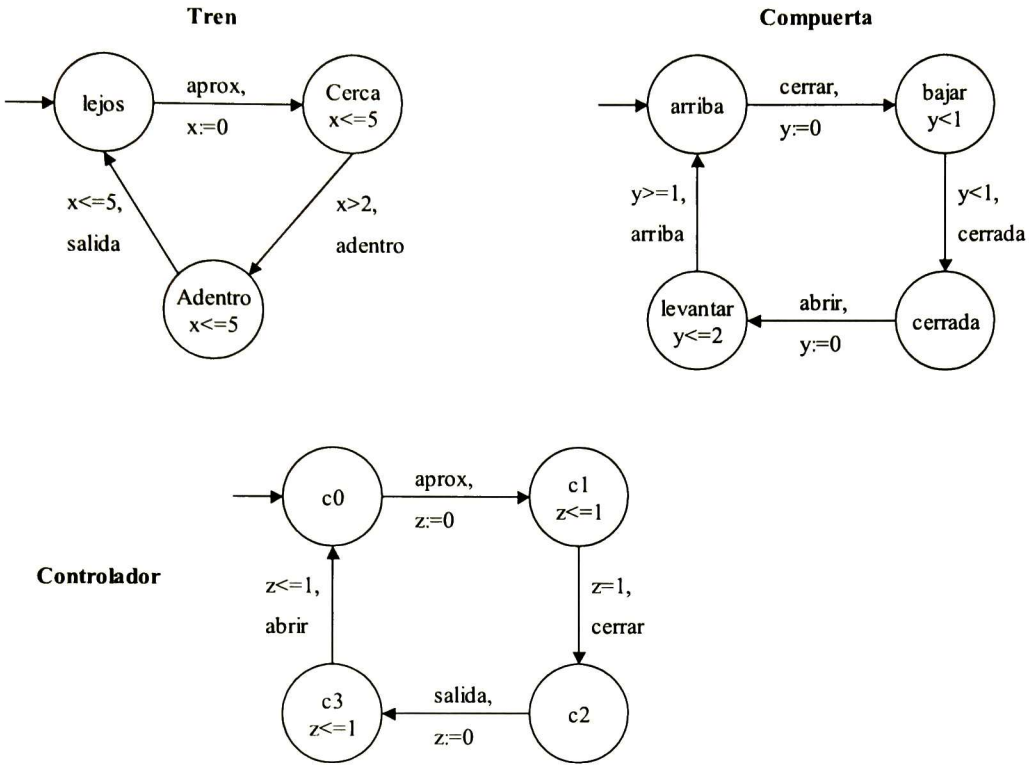


Figura 4.2. Modelos de comportamiento para controlador, compuerta y tren.

### 4.3.1. Verificación del modelo

Se verifican algunas propiedades temporizadas en el modelo del controlador de la compuerta del tren. La verificación se realiza en la herramientas Kronos. Véanse en los apéndices la captura del modelo, propiedades y cálculo de la composición paralela de los autómatas temporizados en Kronos.

**Propiedad 1**

"Durante el intervalo [0, ∞), siempre ocurre verdad"

LTIM: $\Box_{\geq 0}\, T$

TCTL: Inicio $\rightarrow$ AG($EF_{\geq 1}\, T$ ).

"A partir de que inicie el sistema, en todas las trayectorias siempre ocurre que existe una trayectoria donde eventualmente se alcanza alguna localidad en al menos una unidad de tiempo". Esta es la propiedad de no bloqueo. Donde Inicio es una proposición especial que se cumple en la localidad inicial del sistema.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 2**

"Siempre que el tren esté cerca, entonces eventualmente estará en la sección dentro"

LTIM: $\Box_{\geq 0}$ (cerca $\rightarrow$ $\Diamond_{\geq 0}$ dentro).

TCTL: AG(cerca $\rightarrow$ AF dentro).

"En todas las trayectorias siempre se cumple que si el tren está cerca, entonces eventualmente el tren estará en la sección dentro" Esta es una propiedad cualitativa, pues no expresa cuanto tiempo tardará el tren en llegar a la sección dentro.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 3**

"Siempre que el tren esté cerca, entonces eventualmente, y en a lo más 5 unidades de tiempo, el tren estará en la sección dentro"

LTIM: $\Box_{\geq 0}$ (cerca $\rightarrow$ $\Diamond_{\leq 5}$ dentro).

TCTL: AG(cerca $\rightarrow$ $AF_{\leq 5}$ dentro).

La propiedad expresa lo mismo que la propiedad 2, pero ahora si está cuantificado el tiempo que tardará a lo más el tren en llegar a la sección dentro. La propiedad dice "En todas las trayectorias siempre ocurre que, cuando el tren está cerca entonces en todas las trayectorias eventualmente el tren llegará a la sección dentro en a lo más 5 unidadades de tiempo"

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 4**

"En todas las trayectorias siempre ocurre que cuando esté cerca el tren, entonces existe una trayectoria donde eventualmente en menos de una unidad de tiempo la compuerta ya estará cerrada"

LTIM: No existe fórmula de la LTIM equivalente a la siguiente fórmula en TCTL.

TCTL: AG(cerca $\rightarrow$ $EF_{<1}$ cerrada).

Esto no es posible, porque se sabe que cuando el tren está cerca el controlador se espera un minuto para enviar orden de cerrar a la compuerta.

*Resultado de la verificación*: la invariante es falsa.

**Propiedad 5**

"En alguna trayectoria eventualmente si el tren está cerca, entonces existe una trayectoria donde eventualmente en menos de una unidad de tiempo la compuerta estará cerrada"

LTIM: No existe fórmula de la LTIM equivalente a la siguiente fórmula en TCTL.

TCTL: EF(cerca $\rightarrow$ $EF_{<1}$ cerrada).

Similar a la propiedad 4, pero ahora solo se expresa la existencia de una trayectoria.

Esta propiedad debe ser verdadera, pues cuando el tren está a solo un minuto de llegar al cruce la compuerta empezará a cerrarse, y terminar de cerrarse lleva menos de un minuto.

*Resultado de la verificación*: alcanzabilidad verdadera.

**Propiedad 6**

"Siempre que el tren esté cerca, entonces eventualmente, y en a lo más dos unidades de tiempo, la compuerta estará cerrada".

LTIM: $\Box_{\geq 0}$ (cerca $\rightarrow$ $\Diamond_{\leq 2}$ cerrada).

TCTL: AG(cerca $\rightarrow$ $AF_{\leq 2}$ cerrada).

Esta es una propiedad de seguridad crítica. "En todas las trayectorias siempre ocurre que si el tren está cerca, entonces en todas las trayectorias eventualmente en a lo más dos minutos la compuerta debe estar cerrada"

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 7**

"Eventualmente ocurrirá que el tren esté en la sección dentro y la compuerta no esté cerrada"

LTIM: $\Diamond_{\geq 0}$ (dentro $\wedge$ $\neg$cerrada).

TCTL: EF(dentro $\wedge$ $\neg$cerrada).

Esta es una propiedad de seguridad crítica. "Existe una trayectoria donde eventualmente el tren esté en la sección dentro y la compuerta no esté cerrada"

*Resultado de la verificación*: alcanzabilidad falsa.

**Propiedad 8**

"Siempre que la compuerta esté cerrada, entonces eventualmente, y en a lo más 7 unidades de tiempo, la compuerta estará abierta"

LTIM: $\Box_{\geq 0}$ (cerrada $\rightarrow$ $\Diamond_{\leq 7}$ abierta).

TCTL: AG(cerrada $\rightarrow$ $AF_{\leq 7}$ abierta).

"En todas las trayectorias siempre ocurre que si la compuerta esta cerrada, entonces en todas las trayectorias eventualmente en a lo más siete minutos estará de nuevo abierta" Esta es una propiedad de vivacidad.

*Resultado de la verificación*: la invariante es verdadera.

## 4.4. Controlador de semáforos

La figura 4.3 muestra los semáforos instalados en la intersección de una avenida y una calle. El controlador de semáforos asigna un lapso de 40 segundos de luz verde al semáforo de la avenida y 20 segundos de luz verde al de la calle. Ambos semáforos tardan con luz amarilla un lapso de 5 segundos.

Se tiene un sensor que detecta si hay o no hay autos en la calle. Si se tiene luz verde en la calle y se detecta que ya no hay autos, entonces el controlador ordena inmediatamente al semáforo de la calle que ponga luz amarilla.

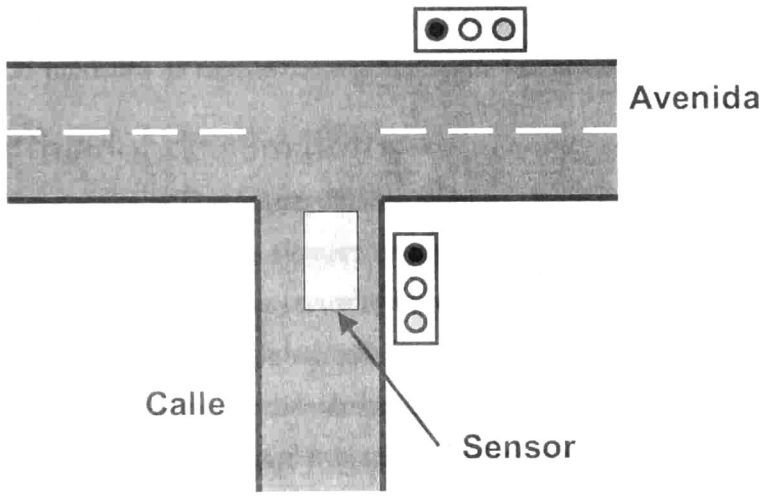


Figura 4.3. Semáforos instalados en la intersección de una avenida y una calle.

Si al terminar el tiempo de luz verde de la avenida, el controlador detecta que no hay autos en la calle, vuelve a asignar un nuevo lapso de tiempo de luz verde a la avenida.

La figura 4.4 muestra el modelo abstracto del controlador de semáforos. Tres autómatas temporizados modelan los comportamientos del semáforo de la avenida, del semáforo de la calle y del sensor de presencia de autos.

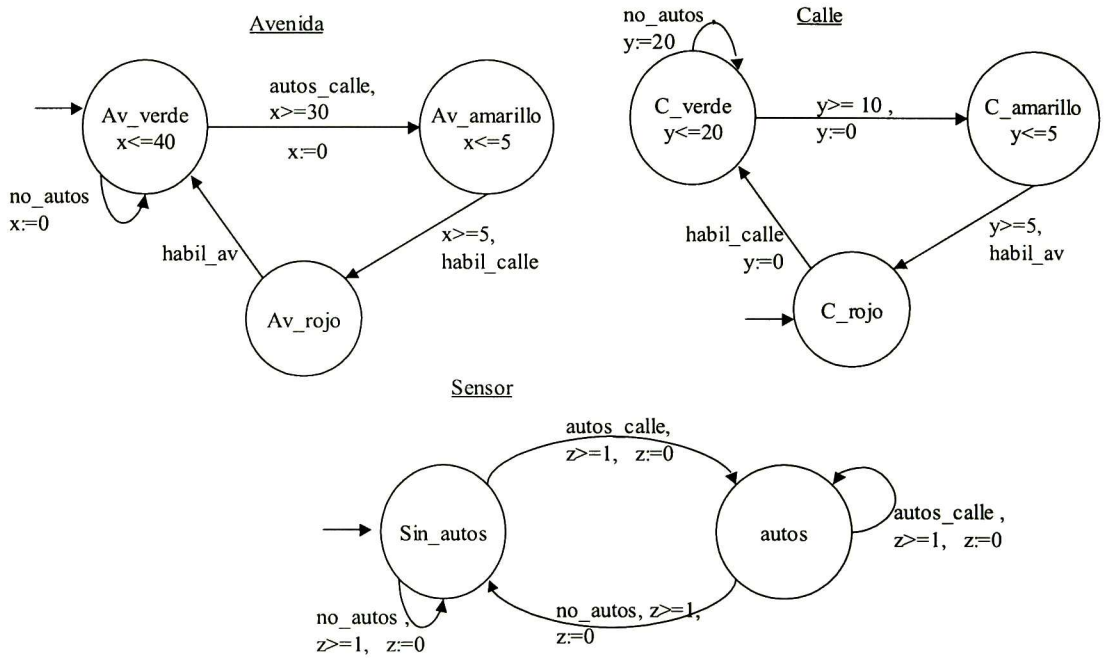


Figura 4.4. Modelos de comportamiento para semáforos y sensor.

El autómata de la avenida está sincronizado con el autómata de la calle según las transiciones sincronizadas marcadas con las señales "habil_calle" y "habil_av"

Tanto el autómata de la avenida como el de la calle, están sincronizados con el autómata del sensor con las transiciones marcadas con la señal "no_autos"

### 4.4.1. Verificación del modelo

Se verifican algunas propiedades temporizadas en el modelo del controlador de semáforos. La verificación se realiza en la herramientas Kronos. Véanse en los apéndices la captura del modelo, propiedades y cálculo de la composición paralela de los autómatas temporizados en Kronos.

**Propiedad 1**

"Durante el intervalo [0, ∞), siempre ocurre verdad"

LTIM: $\Box_{\geq 0}$ T.

TCTL: Inicio → AG($EF_{\geq 1}$ T ).

"A partir de que inicie el sistema, en todas las trayectorias siempre ocurre que existe una trayectoria donde eventualmente se alcanza alguna localidad en al menos una unidad de tiempo" Esta es la propiedad de no bloqueo. Donde Inicio es una proposición especial que se cumple en la localidad inicial del sistema.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 2**

"Siempre ocurre que, si la avenida tiene luz roja, entonces eventualmente, y en a lo más 25 unidades de tiempo, la avenida tendrá luz verde"

LTIM: $\Box_{\geq 0}$ (Av_rojo → $\Diamond_{\leq 25}$ Av_verde).

TCTL: AG(Av_rojo → $AF_{\leq 25}$ Av_verde).

"En todas las trayectorias siempre ocurre que, si el semáforo de la avenida está con luz roja, entonces en todas las trayectorias eventualmente en a lo más 25 segundos la luz cambiará a verde" Se espera sea verdad la propiedad, pues si la avenida tiene luz roja implica que la calle tiene luz verde o amarilla, y como sabemos que el tiempo máximo de luz verde en la calle es 20 segundos y de luz amarilla 5 segundos, o sea, se tiene un total de 25 segundos. Por tanto, a lo más hay que esperar 25 segundos para dar de nuevo luz verde a la avenida.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 3**

"En todas las trayectorias siempre ocurre que, si la calle tiene luz verde, entonces existe una trayectoria donde eventualmente, y a lo más en 5 segundos, la calle tendrá luz roja y por tanto la avenida tendrá luz verde"

LTIM: No existe fórmula de la LTIM equivalente a la siguiente fórmula en TCTL.

TCTL: AG(C_verde $\rightarrow$ $EF_{\leq 5}$ (C_rojo $\wedge$ Av_verde)).

Esta situación se da cuando el semáforo de la calle esta a punto de pasar de luz verde a luz amarilla, y como sabemos que la luz amarilla tarda solo 5 segundos. Una vez que termine el lapso de luz amarilla, el semáforo de la calle cambia a luz roja.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 4**

"En todas las trayectorias siempre ocurre que, si la avenida tiene luz verde, entonces existe una trayecoria donde en a lo más 400 unidaddes de tiempo, eventualmente la venida no tendrá luz roja"

LTIM: No existe fórmula de la LTIM equivalente a la siguiente fórmula en TCTL.

TCTL: AG(Av_verde $\rightarrow$ $EF_{\leq 400}$ $\neg$ Av_rojo).

Esta propiedad verifica que si no hay autos en la calle por un lapso grande de tiempo, digamos 400 segundos, el semáforo de la avenida permanecerá con luz verde. La propiedad debe cumplirse pues los autómatas de la avenida y del sensor están sincronizados para renovar el lapso de tiempo de luz verde en la avenida cuando no hay autos en la calle.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 5**

"Eventualmente la avenida y la calle tendrán al mismo tiempo luz roja"

LTIM: $\Diamond_{\geq 0}$ (Av_rojo $\wedge$ C_rojo).

TCTL: EF(Av_rojo $\wedge$ C_rojo)

"Existe una trayectoria donde eventualmente, la avenida y la calle tienen luz roja"

Analizar si se tiene una propiedad de bloqueo. Verificar que ambos semáforos no estén con luz roja al mismo tiempo.

*Resultado de la verificación*: alcanzabilidad falsa.

**Propiedad 6**

"Eventualmente la avenida y la calle tendrán al mismo tiempo luz roja"

LTIM: $\Diamond_{\geq 0}$ (Av_rojo $\wedge$ C_rojo).

TCTL: EF(Av_verde $\wedge$ C_verde).

"Existe una trayectoria donde eventualmente, la avenida y la calle tienen luz verde" Esta es una propiedad de seguridad crítica. Verificar que ambos semáforos no estén con luz verde al mismo tiempo.

*Resultado de la verificación*: alcanzabilidad falsa.

# 4.5. Protocolo de exclusión mutua de Fischer

El protocolo de exclusión mutua de Fischer [20] se verifica con Kronos y Uppaal. Uppaal tiene la ventaja de modelar el sistema en un ambiente gráfico, y por tanto no necesita capturar el modelo en archivos de texto como lo hace Kronos.

El protocolo resuelve la asignación de un recurso compartido a un solo proceso de entre varios que compiten por el recurso. Para simplificar, modelamos el protocolo con dos procesos, las figuras 4.5 y 4.6 muestran el modelo del protocolo en Uppaal y en Kronos respectivamente.

## 4.5.1. Modelado del protocolo en Uppaal

Primero se analiza el modelo en Uppaal, véase figura 4.5, cada proceso se modela con un autómata temporizado, los autómatas pueden asignar o leer el valor de la variable global Id, la cual representa el recurso compartido del protocolo. La asignación "Id:=0" significa que el recurso ha sido liberado e "Id=0?" pregunta si el recurso esta desocupado.

Cuando un proceso *i* requiere utilizar el recurso Id, primero debe solicitarlo, si es el caso, el proceso *i* pasa de la localidad *libre_i* a la localidad *solicita_i* siempre que el recurso este desocupado. En la localidad *solicita_i* se espera un tiempo aleatorio, de a lo más $\Delta$ unidades de tiempo, para pasar a la localidad *espera_i*, y hace la asignación Id:=i, la cual significa que el proceso *i* es de momento el candidato a utilizar el recurso Id.

Estando en la localidad *espera_i* el proceso *i* espera un tiempo aleatorio mayor a δ unidades de tiempo, para δ<Δ, y si todavía sigue siendo el candidato a entrar, entonces pasa a la localidad *sección_crítica_i*, esto es, el proceso *i* está utilizando el recurso Id y ningún otro proceso puede utilizar el recurso, sino hasta que lo liberen.

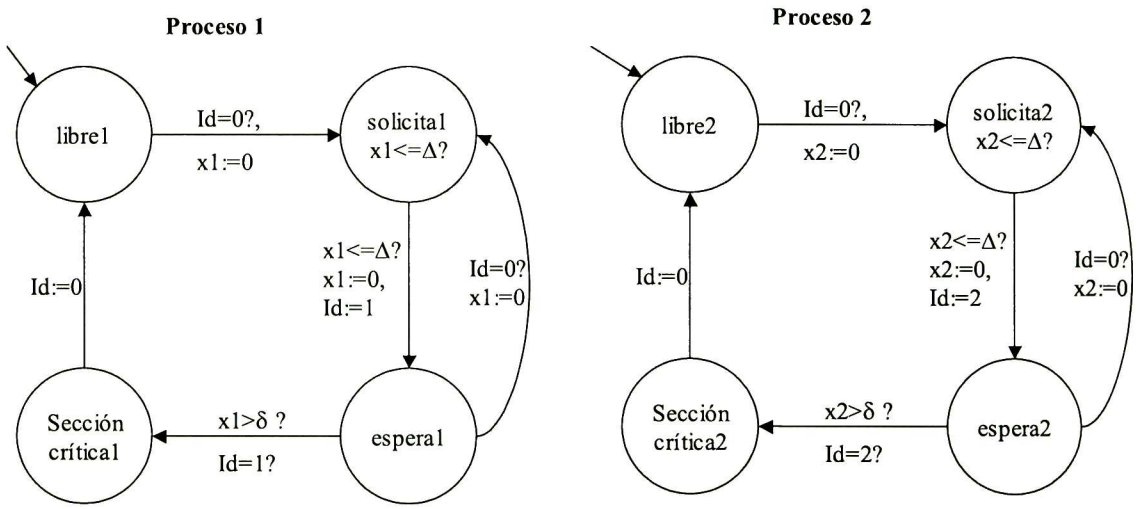


Figura 4.5. Modelo del protocolo de exclusión mutua de Fischer para dos procesos en Uppaal.

Sin embargo, si el proceso *i* ya esperó un tiempo aleatorio mayor a δ unidades de tiempo en la localidad *espera_i* y ya no es el candidato a utilizar el recurso Id, entonces tendrá que esperar aún más tiempo en esa localidad. La espera se prolonga hasta que desocupen el recurso. Ya que desocupen el recurso, el proceso *i* podrá regresar a la localidad *solicita_i* para que de nuevo entre en la competencia.

### 4.5.2. Modelado del protocolo en Kronos

La figura 4.6 muestra el modelo del protocolo capturado en Kronos. La herramienta Kronos no permite usar variables globales en los modelos. Por tanto, hay que hacer ciertas modificaciones al modelo capturado en Uppaal.

El autómata temporizado para modelar un proceso es similar al capturado en Uppaal, la diferencia es que ahora ninguna transición de los autómatas tiene asignaciones o lecturas a la variable global Id. El problema de la asignación del recurso entre los procesos competidores se resuelve introduciendo un nuevo autómata temporizado llamado *exclusión.*

Los autómatas de los procesos del modelo se sincronizan con el autómata *exclusión* según las transiciones marcadas con las señales entra_1, entra_2, id01 e id02.

Cuando los procesos 1 y 2 están en las localidades *espera1* y *espera2* respectivamente y ambos procesos han esperado un tiempo aleatorio mayor a $\delta$, la única forma de pasar a la localidad *sección_crítica_i*, para *i* igual a 1 y 2, es que se sincronice la señal *entra_i* la cual en forma no determinista seleccionará el autómata *exclusión.*

Si por ejemplo, el proceso 1 gana el recurso, entonces el autómata del proceso 1 pasa a la localidad sección_crítica1, y el autómata *exclusión* pasa a la localidad 1. En ésta localidad 1, el autómata *exclusión* no puede asignar el recurso al proceso 2, lo único que puede hacer es esperar a que se habilite la transición con la señal id01, esto es, esperar que el proceso 1 libere el recurso.

### 4.5.3. Verificación del modelo

Se verifican algunas propiedades temporizadas en el modelo del protocolo de exclusión mutua de Fischer. Esta verificación se realiza en ambas herramientas Kronos y Uppaal, considérese $\Delta$=10 y $\delta$=4 unidades de tiempo, véanse en los apéndices la captura del modelo y propiedades en las herramientas. Después de calcular la composición paralela de los autómatas temporizados pertenecientes al modelo, ya podemos iniciar la verificación de las propiedades temporizadas siguientes:

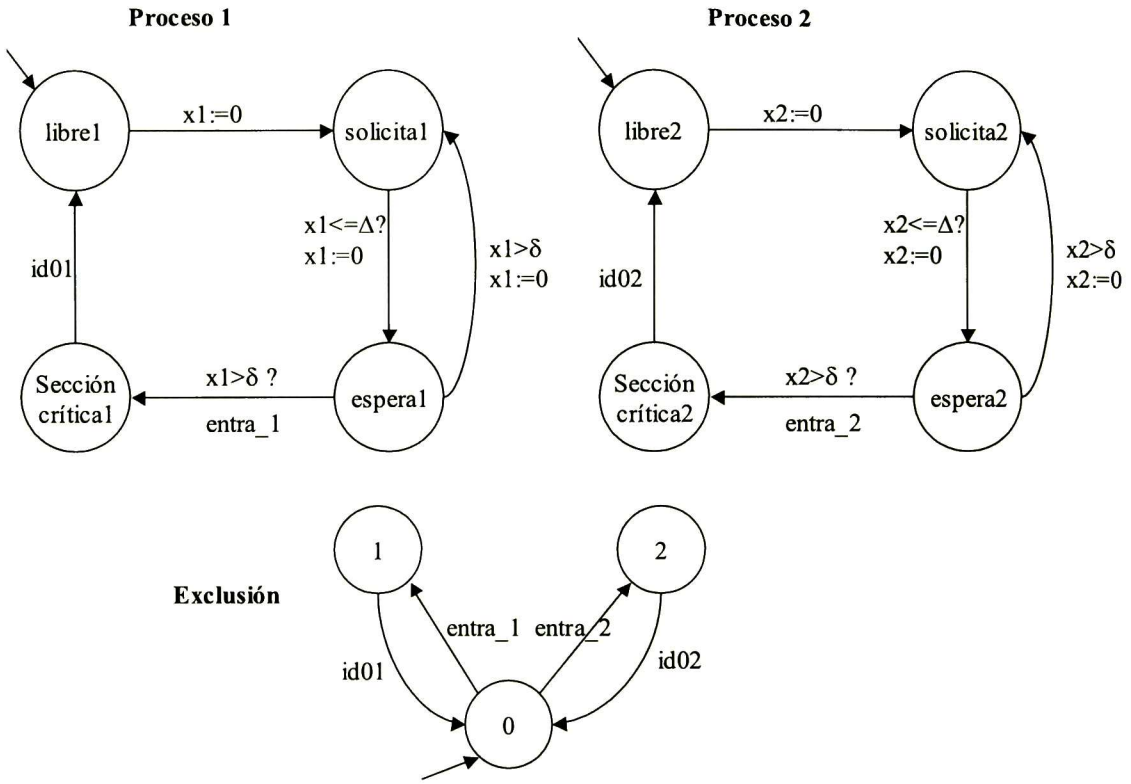


Figura 4.6. Modelo del protocolo de exclusión mutua de Fischer para dos procesos en Kronos.

**Propiedad 1**

"Durante el intervalo [0, ∞), siempre ocurre verdad"

LTIM: $\Box_{\geq 0}\, \mathsf{T}$.

TCTL: Inicio → $AG(EF_{\geq 1}\, \mathsf{T})$.

"A partir de que inicie el sistema, en todas las trayectorias siempre ocurre que existe una trayectoria donde eventualmente se alcanza alguna localidad en al menos una unidad de tiempo". Esta es la propiedad de no bloqueo. Donde Inicio es una proposición especial que se cumple en la localidad inicial del sistema.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 2**

"Siempre ocurre que, si el proceso 1 solicita el recurso, entonces eventualmente, y en a lo más 100 unidades de tiempo, el proceso 1 tendrá el recurso".

LTIM: $\Box_{\geq 0}$ (Solicita1 $\rightarrow \Diamond_{\leq 100}$ Sección_Crítica_1).

TCTL: AG(Solicita1 $\rightarrow$ AF$_{\leq 100}$ Sección_Critica_1).

"En todas las trayectorias siempre ocurre que, si el proceso 1 solicita el recurso, entonces en todas las trayectorias eventualmente, y en a lo más 100 unidades de tiempo, el proceso 1 tendrá el recurso". Esto no siempre se cumple, pues a pesar de que el proceso 1 solicite el recurso, podría pasar mucho tiempo, mas de 100 unidades de tiempo incluso, sin ganar la contienda por el recurso.

*Resultado de la verificación*: la invariante es falsa.

**Propiedad 3**

"En todas las trayectorias siempre ocurre que, si el proceso 1 solicita el recurso, entonces existe una trayectoria donde eventualmente, y en a lo más 100 unidades de tiempo, el proceso entrará en la sección crítica"

LTIM: No existe fórmula de la LTIM equivalente a la siguiente fórmula en TCTL.

TCTL: AG(Solicita1 $\rightarrow$ EF$_{\leq 5}$ Sección_Critica_1)

Similar a la propiedad 2, pero ahora solo exige la existencia de una trayectoria. Cuando el proceso 1 es candidato a entrar a la sección crítica y ha esperado mas de $\delta$=4 unidades de tiempo, entonces cabe la posibilidad de que entre a la sección crítica.

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 4**

"Eventualmente los dos procesos estarán al mismo tiempo utilizando el recurso compartido"

LTIM: $\Diamond_{\geq 0}$ (Sección_Critica_1 $\wedge$ Sección_Critica_2).

TCTL: EF (Sección_Critica_1 $\wedge$ Sección_Critica_2).

"Existe alguna trayectoria donde eventualmente los dos procesos estén al mismo tiempo utilizando el recurso compartido"

*Resultado de la verificación*: alcanzabilidad falsa.

### 4.5.4. Comparación entre Kronos y Uppaal

**Las ventajas de modelar con Uppaal son:**

- La captura del modelo del sistema se realiza de manera gráfica, y por tanto, no se capturan archivos de texto como en Kronos.
- Una vez capturado un modelo de un sistema, se puede generar en automático otras copias del mismo modelo. Por ejemplo, en el protocolo de Fischer, se construye un autómata de un proceso y luego se puede pedir a Uppaal crear tantas copias de ese proceso según sean necesarias.
- Permite utilizar variables globales en el modelo del sistema, las cuales como el caso del protocolo de Fischer evitan introducir nuevos autómatas que representen a dichas variables.
- Resultados experimentales muestran que Uppaal es substancialmente más rápido para verificar sistemas de tiempo real comparado con Kronos, [17].

**La ventaja de modelar con Kronos es:**

- Kronos cubre todo el poder expresivo de TCTL [26]. En cambio Uppaal, solo cubre un subconjunto de TCTL y es aquel que permite especificar únicamente propiedades de seguridad [18, 19]. Las sintaxis para especificar propiedades en Uppaal se define como:

$$\phi := \mathrm{AG}\ \psi \mid \mathrm{EF}\ \psi \quad ; \quad \psi := a \mid \alpha \mid \psi_1 \wedge \psi_2 \mid \neg\ \psi_1$$

Donde 'a' es una localidad de algún autómata temporizado del sistema, $\alpha$ es una restricción lineal simple de reloj de la forma $x \sim n$, para un operador de comparación $\sim$, $x$ un reloj y $n$ un entero. Nótese la restricción del uso de los operadores AG y EF: solo se permiten al inicio de la fórmula.

# 4.6. Protocolo CSMA/CD

En cualquier red de difusión con un solo canal compartido de comunicaciones, el cual es referido algunas veces como un canal de multi-acceso, el problema central es como asignar el uso del canal cuando varias estaciones compiten por él. Existen varios protocolos para resolver este problema. Uno de esos protocolos es el protocolo de acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD, por sus siglas en inglés) [24].

Una descripción simplificada del protocolo es la siguiente. Cuando una estación tiene datos para enviar, primero escucha el canal. Si el canal esta libre (ninguna otra estación esta transmitiendo), la estación comienza a enviar su mensaje. Sin embargo, si detecta que el canal esta ocupado, la estación espera un lapso de tiempo aleatorio y repite la operación. Cuando ocurre una colisión, debido a que muchas estaciones transmiten en forma simultanea, todas ellas detectan la colisión, abortan sus transmisiones inmediatamente y esperan un tiempo aleatorio para iniciar la operación otra vez. Si dos mensajes colisionan entonces se pierden ambos mensajes.

Formalmente se modela el protocolo CSMA/CD para dos procesos transmisores localizados en dos estaciones comunicadas por un canal bidireccional. Supongamos que el canal es tipo Ethernet de 10Mbps con un retardo de propagación $\sigma$ de 26$\mu$s. Los mensajes tienen una longitud fija de 1024 bytes, y por tanto el tiempo $\lambda$ para enviar un mensaje completo, incluyendo el tiempo de propagación, es de 808$\mu$s. Para simplificar, se hacen las siguientes suposiciones: el canal esta libre de errores, solo se modela la transmisión del mensaje, y no se permite el almacenamiento de mensajes de entradas.

## 4.6.1. Estructura del sistema

La estructura del sistema se muestra en la figura 4.7. Cada caja representa un componente del sistema. Las líneas representan las sincronizaciones entre los componentes. Emisor_i y el canal se sincronizan con los siguientes eventos:

ini*i*, el emisor_i empieza a transmitir el mensaje;

ocup*i*, el emisor_i detectó canal ocupado;

fin*i*, el emisor_i completó la transmisión;

colisión*i*, el emisor_i detectó una colisión.

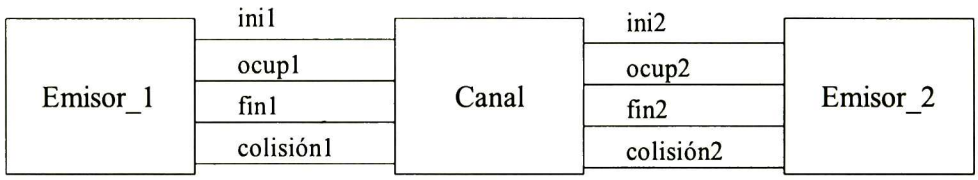


Figura 4.7. Estructura del protocolo CSMA/CD.

### 4.6.2. Autómatas temporizados de los emisores

El comportamiento del emisor emisor_i se muestra en la figura 4.8. Un emisor puede estar en uno de las siguientes localidades:

espera*i*, emisor_i está desocupado, esto, es no hay mensaje para enviar;

transm*i*, emisor_i está enviando un mensaje;

reintenta*i*, emisor_i esta esperando para reintentar transmisión, esto después de haber detectado canal ocupado ó una colisión.

Inicialmente, emisor_i está en la localidad espera*i*. Cuando se tiene un mensaje para enviar, se ejecuta alguna de las siguientes transiciones. Si el canal esta desocupado, el emisor inicia la transmisión. De otro modo, si el canal esta ocupado debido a que otro emisor esta también transmitiendo, o debido a alguna colisión, entonces emisor_i se espera para después reintentar la operación. Por otro lado, si emisor_i no tiene mensaje que transmitir y detecta una colisión, entonces emisor_i se mantiene en espera*i*.

Emisor_i permanece en la localidad transm*i* a lo más un lapso de tiempo de $\lambda$. Si se detecta una colisión antes de un lapso de tiempo de $\sigma$, entonces emisor_i va a la localidad reintenta*i*. De otro modo, emisor_i termina de enviar su mensaje y va a la localidad espera*i*.

En la localidad reintenta*i*, el emisor_i intenta transmitir en forma no determinista antes de $2\sigma$ desde del último intento.

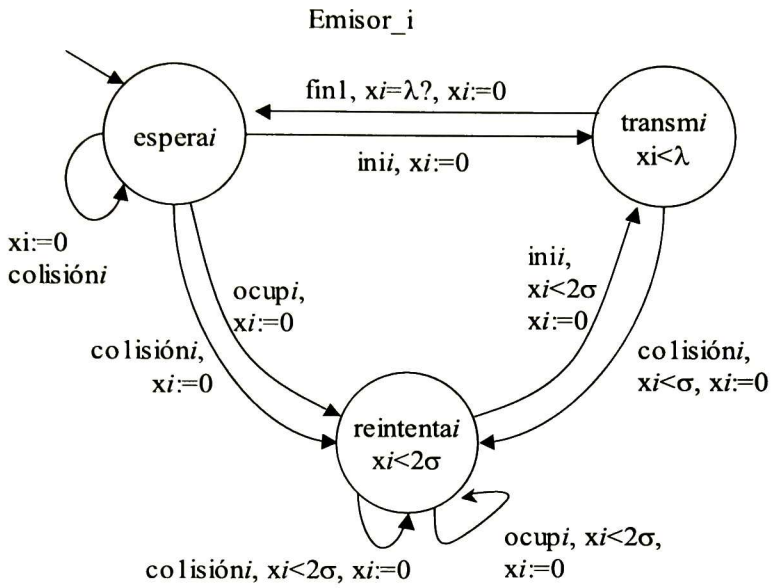


Figura 4.8. Modelo de comportamiento del emisor_i.

### 4.6.3. Autómatas temporizados del canal de comunicación

El comportamiento del canal de comunicación se muestra en la figura 4.9. El canal puede estar en alguna de las cuatro localidades siguientes:

libre, ninguna estación está transmitiendo;

activo, alguna estación está transmitiendo;

colisión, se propaga en el canal un ruido por colisión de mensajes.

Inicialmente, el canal esta en la localidad libre. Cuando una de las estaciones inicia la transmisión de su mensaje, el canal se mueve a la localidad activo. Después de que el canal esté activo por al menos $\sigma$, responderá ocupado para nuevos intentos de transmisión, esto

modela el hecho que la cabecera del mensaje se ha propagado por el canal. Sin embargo, si también otra estación empieza a transmitir antes de σ, el canal se mueve a la localidad de colisión, donde se toma un tiempo menos de σ para propagar la colisión a todas las estaciones. La transición etiquetada con los eventos colisión1 y colisión2, sincroniza a ambas estaciones y el canal. Esto significa que la colisión será detectada por simultáneamente ambas estaciones.

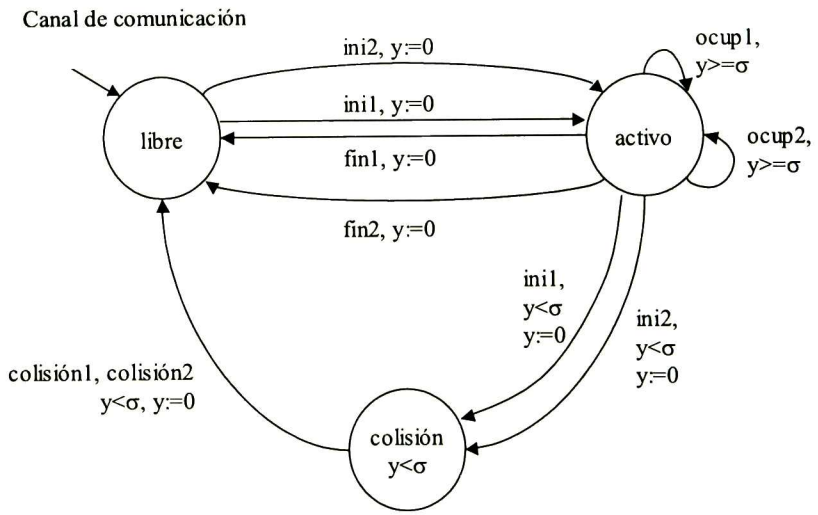


Figura 4.9. Modelo de comportamiento del canal de comunicación.

### 4.6.4. Verificación de propiedades

Se verifican algunas propiedades temporizadas en el modelo del protocolo de comunicaciones CSMA/CD. Esta verificación se realiza en la herramienta Kronos, véanse en los apéndices la captura del modelo y propiedades en la herramienta. Después de calcular la composición paralela de los autómatas temporizados pertenecientes al modelo, se puede iniciar la verificación de las propiedades temporizadas siguientes. Se considera que la unidad de tiempo es en microsegundos.

**Propiedad 1**

"Durante el intervalo [0, ∞), siempre ocurre verdad"

LTIM: $\Box_{\geq 0}$ T.

TCTL: Inicio → AG($EF_{\geq 1}$ T ).

"A partir de que inicie el sistema, en todas las trayectorias siempre ocurre que existe una trayectoria donde eventualmente se alcanza alguna localidad en al menos una unidad de tiempo" Esta es la propiedad de no bloqueo. Donde Inicio es una proposición especial que se cumple en la localidad inicial del sistema.

Resultado de la verificación: la invariante es falsa.

La figura 4.10 muestra el contraejemplo que arroja Kronos: se trata de un camino en el autómata que resulta de la composición paralela. Este camino va desde la localidad inicial hasta la localidad donde queda bloqueado el protocolo.

El contraejemplo demuestra que se puede llegar a una localidad, de hecho es la localidad 3 mostrado en la figura 4.10, donde no se tiene otra localidad a donde avanzar.

En la localidad 3 se cumple el conjunto de proposiciones {TRANSM1, TRANSM2, COLISION}, esto significa que ambas estaciones están transmitiendo simultáneamente y el canal en consecuencia está difundiendo una colisión de mensajes. Lo ideal es que estando en la localidad 3, se pueda ir a una localidad donde se cumpla {REINTENTA1, REINTENTA2, LIBRE}, esto es, una localidad donde el canal quede libre y las estaciones reintenten sus transmisiones en tiempos aleatorios.

Desde la localidad 3 no se puede avanzar a otra localidad, ni permanecer en ella para siempre, debido a que en la localidad se puede cumplir que 26<=X1 y, según el modelo del emisor_i, estando en la localidad transm*i* puede avanzar a reintenta*i* si cumple x*i*<σ=26. Por tanto, es claro que el error está en dar muy poco tiempo a las estaciones para detectar la colisión. Sin embargo, queda el problema de determinar cual es la cota máxima de x*i*, ya que no puede ser σ=26. Las mismas restricciones de la localidad 3 nos dicen que se debe cumplir "25<=Y and Y<26 and X1<X2+26 and X2=Y"; por tanto, la cota de X1 es X1<26+26.

```
location: 0
propositions: LIBRE ESPERA_2 ESPERA_1
constraint: X1=0 and X2=0 and Y=0

location: 0
propositions: LIBRE ESPERA_2 ESPERA_1
constraint: TRUE
transition: TRUE => INI1 ENVIA1   RESET{ X1 Y } ; goto 1

location: 1
propositions: ESPERA2 ACTIVO TRANSM1
constraint: X1=0 and Y=0

location: 1
propositions: ESPERA2 ACTIVO TRANSM1
constraint: 0<X1 and X1<26 and X1<=X2 and X1=Y
transition: X1<=808 and Y<26 => INI2 ENVIA2 ; RESET{ X2 Y } ; goto 3

location: 3
propositions: TRANSM1 TRANSM2 COLISION
constraint: X1<26 and X2=0 and Y=0 and X2<X1 and Y<X1

location: 3
propositions: TRANSM1 TRANSM2 COLISION
constraint: 26<=X1 and 25<=Y and Y<26 and X1<X2+26 and X2=Y
```

Figura 4.10. Contraejemplo que exhibe bloqueo del protocolo

Entonces, se debe corregir el modelo del emisor_i. Hay que corregir la restricción $x_i<\sigma=26$ de la transición que va de la localidad transm*i* a reintenta*i* por la restricción $x_i<2\sigma$. En la figura 4.11 se muestra el modelo del emisor corregido.

La razón del bloqueo era debido a que el emisor (sin corrección de cota) al transmitir su mensaje esperaba un tiempo $\sigma$ para determinar si hubo colisión, pero $\sigma$ es sólo el retardo de propagación de la cabecera del mensaje por todo el canal, si hubiera colisión, el canal necesitaba otro lapso de tiempo igual a $\sigma$ para difundir un mensaje de colisión.

Al hacer la corrección antes mencionada en el modelo del protocolo CSMA/CD, la propiedad de no bloqueo ya se cumple.

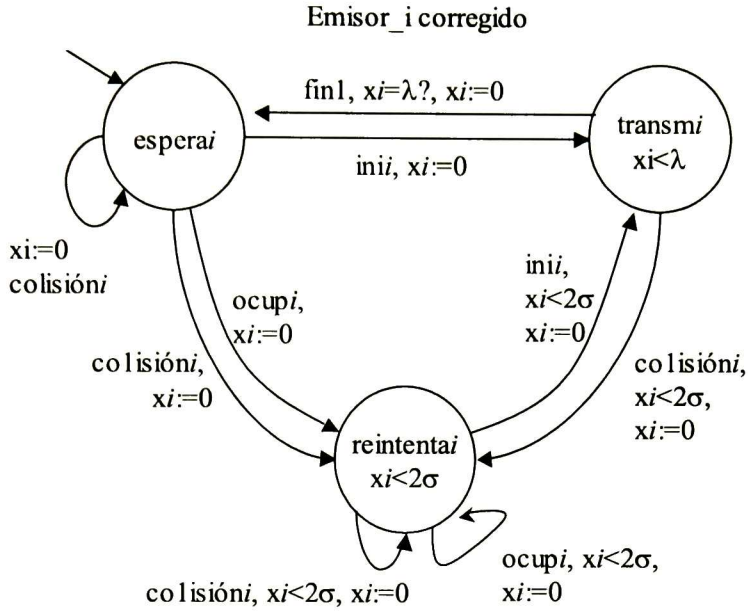


Figura 4.11. Modelo del emisor corregido.

**Propiedad 2**

"Siempre ocurre que, cuando ambas estaciones estén transmitiendo, entonces eventualmente, y en a lo más 52 μs, las estaciones dejarán de transmitir para reintentarlo después"

LTIM: $\Box_{\geq 0}$ ((transm1 $\wedge$ transm2) $\rightarrow$ $\Diamond_{\leq 52}$ (reintenta1 $\wedge$ reintenta2)).

TCTL: AG((transm1 $\wedge$ transm2) $\rightarrow$ $AF_{\leq 52}$ (reintenta1 $\wedge$ reintenta2)).

"En todas las trayectorias siempre ocurre que cuando ambas estaciones estén transmitiendo, entonces en todas las trayectorias eventualmente, y en a lo más 52 μs, las estaciones dejarán de transmitir para reintentarlo después"

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 3**

"Siempre ocurre que, si el canal detecta colisión, entonces eventualmente, y en a lo más 26 μs, el canal quedará libre"

LTIM: $\Box_{\geq 0}$ (colisión $\rightarrow \Diamond_{\leq 26}$ libre).

TCTL: AG(colisión $\rightarrow AF_{\leq 26}$ libre).

"En todas las trayectorias siempre ocurre que si el canal detecta colisión, entonces en todas las trayectorias eventualmente, y en a lo más 26 μs, el canal quedará libre"

*Resultado de la verificación*: la invariante es verdadera.

**Propiedad 4**

"En todas las trayectorias siempre ocurre que si emisor_1 está transmitiendo y no hay colisión, entonces existe una trayectoria donde eventualmente a las 26 μs, se cumple que en todas las trayectorias eventualmente a las 782 regresará a la localidad espera1"

LTIM: No existe fórmula de la LTIM equivalente a la siguiente fórmula en TCTL.

TCTL: AG((transm1 $\wedge \neg$colisión) $\rightarrow EF_{=26} AF_{=782}$ espera1).

Sí existe esta posibilidad, debido que a partir de las 26 μs solo faltarían 808-26=782 μs restantes para terminar de transmitir el mensaje.

*Resultado de la verificación*: la invariante es verdadera.

# Capítulo 5

# Conclusiones

## 5.1. Contenido del capítulo

En este capítulo se muestran las conclusiones obtenidas de este trabajo y una lista de posibles líneas de investigación.

## 5.2. Conclusiones

En la realización de este trabajo se logran obtener las siguientes conclusiones:

- Se propuso la Lógica Temporal Proposicional Temporizada (LTPT) [3] como un lenguaje formal para especificar propiedades con tiempos cuantificados en un dominio discreto.

- Se propone una teoría que establece las bases para resolver el problema de satisfacibilidad de una fórmula de la LTPT. Esto es, una teoría que define un autómata asociado a una fórmula de la LTPT.

- Se propone un algoritmo incremental para construir un autómata asociado a una fórmula de la LTPT. Este algoritmo resulta de una extensión del algoritmo incremental de [9] para construir autómatas asociados a fórmulas de la LTLP.

- Se propuso la Lógica Temporal de Intervalos Métricos (LTIM) [5] como un lenguaje formal para especificar propiedades con tiempos cuantificados en un dominio continuo.

- Se propone una teoría que establece las bases para resolver el problema de satisfacibilidad de una fórmula de la LTIM. Esto es, una teoría que define un autómata temporizado asociado a una fórmula de la LTIM.

- Se propone un algoritmo incremental para construir un autómata temporizado asociado a una fórmula de la LTIM. Este algoritmo resulta de una extensión del algoritmo incremental de [9] para construir autómatas asociados a fórmulas de la LTLP.

## 5.3. Trabajo futuro

El desarrollo de este trabajo deja distintos puntos para investigar, y también se propone implementar algoritmos:

- Realizar la demostración formal de la teoría propuesta para resolver el problema de satisfacibilidad de fórmulas de la LTLP.

- Implementar el algoritmo incremental propuesto para construir autómatas asociados a fórmulas de la LTPT en la herramienta de verificación del Cinvestav [9, 23, 16].

- Realizar la demostración formal de la teoría propuesta para resolver el problema de satisfacibilidad de fórmulas de la LTIM.

- Implementar el algoritmo incremental propuesto para construir autómatas temporizados asociados a fórmulas de la LTIM en la herramienta de verificación del Cinvestav [9, 23, 16].

- Proponer una sintaxis alternativa para la LTPT con el fin de disminuir la complejidad en la solución del problema de satisfacibilidad de la LTPT. Una sugerencia es tratar con una sintaxis similar de la LTIM.

- Investigar cuales técnicas de reducción del espacio de estados aplicadas a la LTLP se pueden también aplicar a la LTIM.

- Proponer lógicas matemáticas para especificar propiedades de sistemas híbridos [8], aquellos donde las variables no solo miden el tiempo, sino también miden otros factores como temperatura, presión, velocidad, niveles, etc.

# Bibliografía


[1] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183-235, 1994.

[2] R. Alur and D.L. Dill. Automata-theoretic verification of real-time systems. In *Formal Methods for Real-Time Computing, Trends in Software Series, John Wiley & Sons Publishers*, pp. 55-82, 1996.

[3] R. Alur and T.A. Henzinger. A really temporal logic. *Journal of the ACM* 41:181-204, 1994.

[4] R. Alur, C. Courcoubetis, and D. Dill. Model checking in dense real time. *Information and Computation*, 104(1):2-34, 1993.

[5] R. Alur, T. Feder, T.A. Henzinger. The benefits of relaxing punctuality. *Journal of the ACM* 43:116-146, 1996.

[6] Gerd Behrmann, Johan Bengtsson, Alexandre David, Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal Implementation Secrets. In *Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems* (FTRTFT'02).

[7] M. Ben-Ari, Z.Manna, and Pnueli. The temporal logic of branching time. *Acta Informática*, 207-226. 1983.

[8] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen, and P McKenzie. *Systems and Software Verification.* Springer-Verlag, 2001.

[9] Brenda Casillas. Representación de una fórmula de lógica temporal lineal propoposicional como un autómata de Büchi generalizado etiquetado. Tesis de Maestría, CINVESTAV del IPN, Unidad Guadalajara, 2003.

[10] E. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press. 2000.

[11] C. Daws, A. Olivero, and S. Yovine. The tool Kronos. In *Hybrid Systems III, Verification and Control*, pages 208-219. LNCS 1066, Springer-Verlag, 1996.

[12] E.A. Emerson and J.Y. Halpern. "Sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM* 33(1):151-178, 1986.

[13] M. Geilen. Formal Techniques for Verification of Complex Real-time Systems. Ph.D. Thesis, Eindhoven University of Technology, 2002.

[14] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A Tutorial on Uppaal. In *proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems* (SFM-RT'04). LNCS 3185. 2004.

[15] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193-244, 1994.

[16] César Hernández. Modelado y verificación por comprobación de modelos de una clase de sistemas de procesamiento industrial. Tesis de Maestría, CINVESTAV del IPN, Unidad Guadalajara, 2004.

[17] Johan Bengtsson, Kim G. Larsen, Fredrik Larsson, Paul Pettersson and Wang Yi. Uppaal a Tool Suite for Automatic Verification of Real-Time Systems. In *Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems, New Bruns*

[18] Joost-Pieter Katoen. *Concepts, Algorithms, and Tools for Model Checking.* Lecture Notes. Institut für Mathematische Maschinen und Datenverarbeitung, 1999.

[19] Kim G. Larsen, Paul Pettersson and Wang Yi. Uppaal in a Nutshell. In *Springer International Journal of Software Tools for Technology Transfer* 1(1+2), 1997.

[20] L. Lamport. A fast mutual exclusion algorithm. In *ACM trans. on Computer Systems*, vol. 5, pp. 1-11, Feb. 1987.

[21] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems (Safety).* Springer-Verlag, 1995.

[22] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems.* Specification Springer-Verlag, 1991.

[23] G. Pérez Banderas. Comprobación de modelos explícita basada en autómatas de Büchi y lógica temporal lineal proposicional. Tesis de Maestría, CINVESTAV del IPN, Unidad Guadalajara, 2003.

[24] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, 1989.

[25] S. Tripakis. The analysis of timed systems in practice. Ph.D. Thesis, Universite Joseph Fourier, Grenoble, France, 1998.

[26] S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, Vol. 1, Issue 1/2, pages 123-133. Springer-Verlag. October 1997.

[27] S. Yovine. Model checking timed automata. In *Embedded Systems*, LNCS 1494, 1998.

# Apéndice A

# Verificación con Kronos y Uppaal

## A.1. Contenido del apéndice

En este apéndice se presenta la verificación de los casos de estudio del capítulo 4 en las herramientas Kronos y Uppaal. Para los casos de estudio verificados en Kronos se muestran los autómatas temporizados capturados en el formato Kronos, composición paralela de autómatas temporizados del modelo y captura de propiedades a verificar. Para los casos de estudio verificados en Uppaal se muestra el diseño del modelo del sistema en Uppaal y captura de propiedades a verificar en el modelo. La sección A.2 presenta el caso de estudio del controlador de la compuerta del tren en Kronos. La sección A.3 presenta el caso de estudio del controlador de semáforos en Kronos. La sección A.4 presenta el caso de estudio del protocolo de exclusión mutua de Fischer tanto en Kronos como Uppaal. Por último, la sección A.5. presenta el caso de estudio del protocolo de comunicaciones CSMA/CD.

Un tutorial de Uppaal se puede consultar en [14]. Para Kronos no se tiene algún tutorial, sin embargo en [11, 26, 25] se pueden ver varios ejemplos de cómo utilizar Kronos. Las herramientas Uppaal y Kronos son gratuitas y se pueden obtener de sus sitios en Internet, http://www.uppaal.com/ y http://www-verimag.imag.fr/TEMPORISE/kronos/.

## A.2. Captura del controlador de la compuerta del tren

Este caso de estudio se verifica en Kronos. Las figuras A.1, A.2 y A.3 muestran los autómatas temporizados para los modelos del tren, compuerta y controlador respectivamente capturados en archivos de texto.

```
/* tren.tg */
#states 3
#trans 3
#clocks 1 X
#sync aprox salida

state: 0
prop: lejos
invar: true
trans:
true => aprox; reset{X}; goto 1

state: 1
prop: cerca
invar: X<=5
trans:
X>2 => adentro; reset{}; goto 2

state: 2
prop: adentro
invar: X<=5
trans:
X<=5 => salida; reset{}; goto 0
```

Figura A.1. Archivo "tren.tg"

```
/* compuerta.tg */
#states 4
#trans 4
#clocks 1 Y
#sync cerrar abrir  /*up down*/

state: 0
prop: arriba
invar: true
trans:
true =>cerrar; reset{Y}; goto 1

state: 1
prop: bajar
invar: Y<1
trans:
Y<1 => cerrada; reset{}; goto 2

state: 2
prop: cerrada
invar: true
trans:
true => abrir; reset{Y}; goto 3

state: 3
prop: levantar
invar: Y<=2
trans:
Y>=1 => arriba; reset{}; goto 0
```

Figura A.2. Archivo "compuerta.tg"

```
/* controlador.tg */

#states 4
#trans 4
#clocks 1 Z
#sync aprox cerrar salida abrir

state: 0
prop: c0
invar: true
trans:
true => aprox; reset{Z}; goto 1

state: 1
prop: c1
invar: Z<=1
trans:
Z=1 => cerrar; reset{}; goto 2

state: 2
prop: c2
invar: true
trans:
true => salida; reset{Z}; goto 3

state: 3
prop: c3
invar: Z<=1
trans:
Z<=1 => abrir; reset{}; goto 0
```

Figura A.3. Archivo "controlador.tg"

Al calcular con Kronos la composición paralela de los autómatas 'tren' 'compuerta' y 'controlador' se obtiene el autómata mostrado en las figuras A.4 y A.5.

En la tabla A.1 se muestra la captura de las propiedades para este caso de estudio, propuestas en el capítulo 4, en formato Kronos. Los resultados obtenidos, de la verificación de estas propiedades en el modelo, se explican en el capítulo 4.

```
/* composicion.tg */
#states 12
#trans  17
#clocks  3
        X /* tren */
        Y /* compuerta */
        Z /* controlador */

#sync
        APROX /* tren controlador */
        SALIDA /* tren controlador */
        CERRAR /* compuerta controlador */
        ABRIR /* compuerta controlador */


state: 0  /* vector state: < 0, 0, 0 > */
prop: LEJOS ARRIBA C0
invar: TRUE
trans:
L: TRUE =C> APROX ; RESET{ X Z }; goto 1

state: 1  /* vector state: < 1, 0, 1 > */
prop: CERCA ARRIBA C1
invar: X<=5 and Z<=1
trans:
L: 2<X and X<=5 =C> ADENTRO ; reset{};
goto 2
L: Z=1 =C> CERRAR ; RESET{ Y }; goto 3

state: 2  /* vector state: < 2, 0, 1 > */
prop: ADENTRO ARRIBA C1
invar: X<=5 and Z<=1
trans:
L: Z=1 =C> CERRAR ; RESET{ Y }; goto 4

state: 3  /* vector state: < 1, 1, 2 > */
prop: CERCA BAJAR C2
invar: X<=5 and Y<1
trans:
L: 2<X and X<=5 =C> ADENTRO ; reset{};
goto 4
L: Y<1 =C> CERRADA ; reset{}; goto 5

state: 4  /* vector state: < 2, 1, 2 > */
prop: ADENTRO BAJAR C2
invar: X<=5 and Y<1
```

Figura A.4. Parte 1 del archivo "composición tren.tg"

```
trans:
L: Y<1 =C> CERRADA ; reset{}; goto 6
L: X<=5 =C> SALIDA ; RESET{ Z }; goto 7

state: 5  /* vector state: < 1, 2, 2 > */
prop: CERCA CERRADA C2
invar: X<=5
trans:
L: 2<X and X<=5 =C> ADENTRO ; reset{}; goto 6

state: 6  /* vector state: < 2, 2, 2 > */
prop: ADENTRO CERRADA C2
invar: X<=5
trans:
L: X<=5 =C> SALIDA ; RESET{ Z }; goto 8

state: 7  /* vector state: < 0, 1, 3 > */
prop: LEJOS BAJAR C3
invar: Y<1 and Z<=1
trans:
L: Y<1 =C> CERRADA ; reset{}; goto 8

state: 8  /* vector state: < 0, 2, 3 > */
prop: LEJOS CERRADA C3
invar: Z<=1
trans:
L: Z<=1 =C> ABRIR ; RESET{ Y }; goto 9

state: 9  /* vector state: < 0, 3, 0 > */
prop: LEJOS LEVANTAR C0
invar: Y<=2
trans:
L: 1<=Y and Y<=2 =C> ARRIBA ; reset{}; goto 0
L: TRUE =C> APROX ; RESET{ X Z }; goto 10

state: 10 /* vector state: < 1, 3, 1 > */
prop: CERCA LEVANTAR C1
invar: X<=5 and Y<=2 and Z<=1
trans:
L: 2<X and X<=5 =C> ADENTRO ; reset{}; goto 11
L: 1<=Y and Y<=2 =C> ARRIBA ; reset{}; goto 1

state: 11 /* vector state: < 2, 3, 1 > */
prop: ADENTRO LEVANTAR C1
invar: X<=5 and Y<=2 and Z<=1
trans:
L: 1<=Y and Y<=2 =C> ARRIBA ; reset{}; goto 2
```

Figura A.5. Parte 2 del archivo "composición tren.tg"

| Num | TCTL | Kronos |
|---|---|---|
| 1 | Inicio → AG($EF_{\geq 1}$ T ) | init impl ab (ed{>=1} true ) |
| 2 | AG(cerca → AF(dentro)) | ab(cerca impl ad (adentro) ) |
| 3 | AG(cerca → $AF_{\leq 5}$(dentro)) | ab(cerca impl ad{<=5}(adentro) ) |
| 4 | AG(cerca → $EF_{<1}$(cerrada)) | ab(cerca impl ed{<1}(cerrada) ) |
| 5 | EF(cerca → $EF_{<1}$(cerrada)) | ed( cerca impl ed{<1}(cerrada) ) |
| 6 | AG(cerca → $AF_{\leq 2}$(cerrada)) | ab( cerca impl ad{<=2}(cerrada) ) |
| 7 | EF(dentro ∧ ¬cerrada) | ed(adentro and not cerrada) |
| 8 | AG(cerrada → $AF_{\leq 7}$(abierta)) | ab( cerrada impl ad{<=7}(arriba) ) |

Tabla A.1. Propiedades del controlador de la compuerta del tren.

## A.3. Captura del controlador de semáforos

Este caso de estudio se verifica con Kronos. Las figuras A.6, A.7 y A.8 muestran los autómatas temporizados para los modelos del semáforo en la avenida, semáforo en la calle y sensor de autos, respectivamente capturados en archivos de texto.

Al calcular con Kronos la composición paralela de los autómatas 'avenida', 'calle' y 'sensor' se obtiene el autómata mostrado en las figuras A.9 (primera parte) y A.10 (segunda parte).

En la tabla A.2 se muestra la captura de las propiedades para este caso de estudio, propuestas en el capítulo 4, en formato Kronos. Los resultados obtenidos, de la verificación de estas propiedades en el modelo, se explican en el capítulo 4.

```
/* avenida.tg */

#states 3
#trans 4
#clocks 1 x
#sync carros_calle no_carros_calle
habil_calle habil_av

state: 0
prop: av_verde
invar: x<=40
trans:
true => no_carros_calle; reset{x}; goto 0
x>=30 => carros_calle; reset{x}; goto 1


state: 1
prop: av_amarillo
invar: x<=5
trans:
x>=5 => habil_calle; reset{}; goto 2

state: 2
prop: av_rojo
invar: true
trans:
true => habil_av; reset{x}; goto 0
```

Figura A.6. Archivo "avenida.tg"

```
/* calle.tg */

#states 3
#trans 4
#clocks 1 y
#sync no_carros_calle habil_calle habil_av


state: 0
prop: calle_rojo
invar: true
trans:
true => habil_calle; reset{y}; goto 1


state: 1
prop: calle_verde
invar: y<=20
trans:
 true => no_carros_calle; y:=20; goto 1
  /* ya no necesario verde*/
y>=10 => ini_amarillo; reset{y}; goto 2
/*expira verde*/

state: 2
prop: calle_amarillo
invar: y<=5
trans:
y>=5 => habil_av; reset{}; goto 0
```

Figura A.7. Archivo "calle.tg"

```
/* sensor.tg */

#states 2
#trans 4
#clocks 1 z
#sync no_carros_calle carros_calle

state: 0
prop: s0
invar: true
trans:
z>=1 => no_carros_calle; reset{z}; goto 0
true => carros_calle; reset{z}; goto 1

state: 1
prop: s1
invar: true
trans:
z>=1 => carros_calle; reset{z}; goto 1
true => no_carros_calle; reset{z}; goto 0
```

Figura A.8. Archivo "sensor.tg"

```
/* Timed graph generated for the parallel
composition of:
        automaton 0: avenida.tg
        automaton 1: calle.tg
        automaton 2: sensor.tg
*/
#states 5
#trans  8
#clocks  3
        X /* avenida */
        Y /* calle */
        Z /* sensor */

#sync
        CARROS_CALLE /* avenida
sensor */
        NO_CARROS_CALLE /* avenida
calle */
        HABIL_CALLE /* avenida calle */
        HABIL_AV /* avenida calle */
        NO_CARROS_CALLE /* avenida
calle sensor */


state: 0  /* vector state: < 0, 0, 0 > */
prop: AV_VERDE CALLE_ROJO S0
invar: X<=40
trans:
L: X<=40 =C> NO_CARROS_CALLE ;
RESET{ X }; goto 0
L: 30<=X and X<=40 =C>
CARROS_CALLE ; RESET{ X Z }; goto 1
```

Figura A.9. Parte 1 del archivo "composición semáforos.tg"

```
state: 1  /* vector state: < 1, 0, 1 > */
prop: AV_AMARILLO CALLE_ROJO S1
invar: X<=5
trans:
L: X=5 =C> HABIL_CALLE ; RESET{ Y };
goto 2

state: 2  /* vector state: < 2, 1, 1 > */
prop: AV_ROJO CALLE_VERDE S1
invar: Y<=20
trans:
L: Y<=20 =C> NO_CARROS_CALLE ;
RESET{ Y }; goto 3
L: Y=20 =C> INI_AMARILLO ; RESET{ Y };
goto 3

state: 3  /* vector state: < 2, 2, 1 > */
prop: AV_ROJO CALLE_AMARILLO S1
invar: Y<=5
trans:
L: Y=5 =C> HABIL_AV ; RESET{ X }; goto 4

state: 4  /* vector state: < 0, 0, 1 > */
prop: AV_VERDE CALLE_ROJO S1
invar: X<=40
trans:
L: X<=40 =C> NO_CARROS_CALLE ;
RESET{ X }; goto 4
L: 30<=X and X<=40 and 1<=Z =C>
CARROS_CALLE ; RESET{ X Z }; goto 1
```

Figura A.10. Parte 2 del archivo "composición semáforos.tg"

| Num | TCTL | Kronos |
|---|---|---|
| 1 | Inicio → AG($EF_{\geq 1}$ T ) | init impl ab (ed{>=1} true ) |
| 2 | AG(Av_rojo → $AF_{\leq 25}$ Av_verde) | ab( av_rojo impl ed{<=25}(av_verde) ) |
| 3 | AG(C_verde → $EF_{\leq 5}$ (C_rojo ∧ Av_verde)) | ab( calle_verde impl ed{<=5}(calle_rojo and av_verde) ) |
| 4 | AG(Av_verde → $EF_{\leq 400}$ ¬ Av_rojo) | ab(av_verde impl ed{<=400}(not av_rojo) ) |
| 5 | EF(Av_rojo ∧ C_rojo) | ed(av_rojo and calle_rojo) |
| 6 | EF(Av_verde ∧ C_verde) | ed(av_verde and calle_verde) |

Tabla A.2. Propiedades del controlador semáforos.

# A.4. Captura del protocolo de exclusión mutua de Fischer

Este caso de estudio se verifica con Uppaal y Kronos. Con Uppaal se muestra el diseño gráfico del protocolo y con Kronos los archivos de texto que capturan el protocolo.

## A.4.1. Captura en Uppaal

La figura A.11 muestra la captura del modelo en Uppaal. La localidad con doble circulo indica que es la localidad inicial. La variable 'x' es un reloj sobre un dominio continuo, 'id' es una variable que representa el recurso compartido en exclusión. La etiqueta 'pid' es el identificador del proceso *i*.

La tabla A.3 muestra tres propiedades verificadas con Uppaal. La propiedad 3 no se puede verificar en Uppaal debido a que la sintaxis de Uppaal no permite capturar ese tipo de propiedades. Véase el capítulo 4, donde se define la sintaxis de Uppaal.

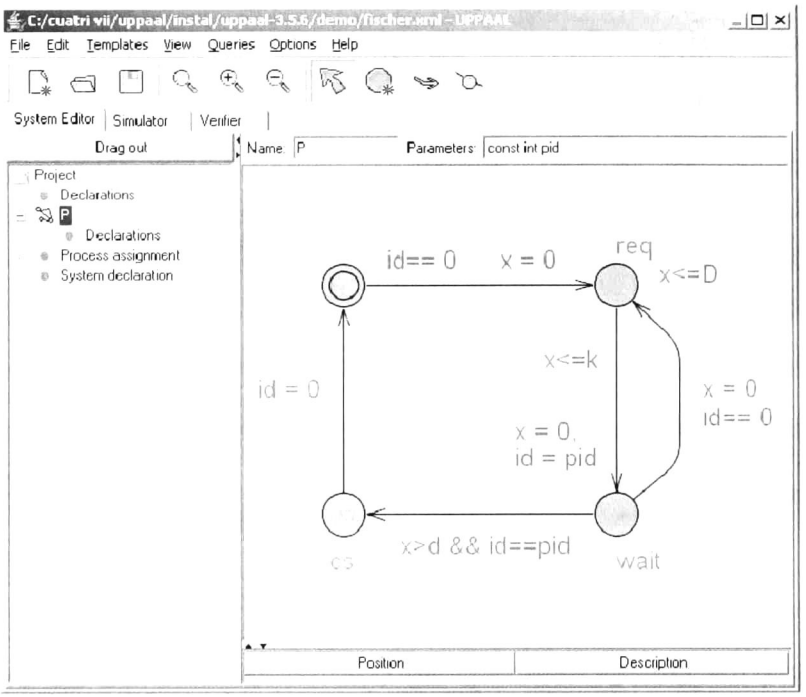


Figura A.11. Diseño de un proceso 'pid' del protocolo de exclusión mutua de Fischer en Uppaal.

| Num | TCTL | Uppaal |
|---|---|---|
| 1 | Inicio $\rightarrow$ AG($EF_{\geq 1}$ T ) | A[] not deadlock |
| 2 | AG(Solicita1 $\rightarrow$ $AF_{\leq 100}$ Sección_Critica_1) | P1.req --> ( P1.cs and P1.x<=100) |
| 3 | AG(Solicita1 $\rightarrow$ $EF_{\leq 5}$ Sección_Critica_1) | No permitido por la sintaxis de Uppaal |
| 4 | EF (Sección_Critica_1 $\wedge$ Sección_Critica_2) | E<> ( P1.cs and P2.cs ) |

Tabla A.3. Propiedades verificadas en el protocolo de exclusión mutua de Fischer con Uppaal.

### A.4.2. Captura en Kronos

Las figuras A.12 y A.13 muestran los autómatas temporizados para los modelos de los procesos 1 y 2 respectivamente capturados en archivos de texto.

```
/* fisher1.tg */

#states 4
#trans 5
#clocks 1 x1
#sync setx0_1 enter_1

state: 0
prop: idle_1
invar: true
trans:
true => try_1; reset{x1}; goto 1

state: 1
prop: req_1
invar: x1<=10
trans:
x1<=10 => setx_1; reset{x1}; goto 2

state: 2
prop: wait_1
invar: true
trans:
x1>=4 => enter_1; reset{}; goto 3
x1>=4 => try_1;  reset{x1}; goto 1

state: 3
prop: critical_1
invar: true
trans:
true => setx0_1; reset{}; goto 0
```

Figura A.12. Archivo "fischer1.tg"

```
/* fisher2.tg */

#states 4
#trans 5
#clocks 1 x2
#sync setx0_2 enter_2

state: 0
prop: idle_2
invar: true
trans:
true => try_2; reset{x2}; goto 1

state: 1
prop: req_1
invar: x2<=10
trans:
x2<=10 => setx_2; reset{x2}; goto 2

state: 2
prop: wait_2
invar: true
trans:
x2>=4 => enter_2; reset{}; goto 3
x2>=4 => try_2;  reset{x2}; goto 1

state: 3
prop: critical_2
invar: true
trans:
true => setx0_2; reset{}; goto 0
```

Figura A.13. Archivo "fischer2.tg"

La figuras A.14 muestra el autómata temporizado 'exclusión' capturado en un archivo de texto.

```
/* exclusión.tg */

#states 3
#trans 7
#clocks
#sync  setx0_1 setx0_2 enter_1 enter_2

state: 0
prop: m0
invar: true
trans:
true => enter_1; reset{}; goto 1
true => enter_2; reset{}; goto 1


state: 1
prop: m1
invar: true
trans:
true => setx0_1 ; reset{}; goto 0
true => setx0_2; reset{}; goto 0
true => enter_1 ; reset{}; goto 2
true => enter_2; reset{}; goto 2


state: 2
prop: m2
invar: true
 trans:
 true => enter_1; reset{}; goto 2
```

Figura A.14. Archivo "exclusión.tg"

Al calcular con Kronos la composición paralela de los autómatas 'proceso 1' 'proceso 2' y 'exclusión' se obtiene un autómata de 15 estados y 36 transiciones. El autómata resultante de la composición se muestra parcialmente, solo los primeros 7 estados, en las figuras A.15 (primera parte) y A.16 (segunda parte).

```
/* Timed graph generated for the parallel
composition of:
        automaton 0: fisher1.tg
        automaton 1: fisher2.tg
        automaton 2: exclusion.tg */
#states 15
#trans 36
#clocks 2
        X1 /* fisher1 */
        X2 /* fisher2 */

#sync
        SETX0_1 /* fisher1 exclusion */
        ENTER_1 /* fisher1 exclusion */
        SETX0_2 /* fisher2 exclusion */
        ENTER_2 /* fisher2 exclusion */


state: 0  /* vector state: < 0, 0, 0 > */
prop: IDLE_1 IDLE_2 E0
invar: TRUE
trans:
L: TRUE =C> TRY_1 ; RESET{ X1 }; goto 1
L: TRUE =C> TRY_2 ; RESET{ X2 }; goto 2

state: 1  /* vector state: < 1, 0, 0 > */
prop: REQ_1 IDLE_2 E0
invar: X1<=10
trans:
L: X1<=10 =C> SETX_1 ; RESET{ X1 };
goto 3
L: TRUE =C> TRY_2 ; RESET{ X2 }; goto 4

state: 2  /* vector state: < 0, 1, 0 > */
prop: IDLE_1 REQ_1 E0
invar: X2<=10
trans:
L: TRUE =C> TRY_1 ; RESET{ X1 }; goto 4
L: X2<=10 =C> SETX_2 ; RESET{ X2 };
goto 5
```

Figura A.15. Parte 1 del archivo "composición fischer.tg"

```
state: 3  /* vector state: < 2, 0, 0 > */
prop: WAIT_1 IDLE_2 E0
invar: TRUE
trans:
L: 4<=X1 =C> TRY_1 ; RESET{ X1 }; goto 1
L: TRUE =C> TRY_2 ; RESET{ X2 }; goto 6
L: 4<=X1 =C> ENTER_1 ; reset{}; goto 7


state: 4  /* vector state: < 1, 1, 0 > */
prop: REQ_1 E0
invar: X1<=10 and X2<=10
trans:
L: X1<=10 =C> SETX_1 ; RESET{ X1 }; goto 6
L: X2<=10 =C> SETX_2 ; RESET{ X2 }; goto 8

state: 5  /* vector state: < 0, 2, 0 > */
prop: IDLE_1 WAIT_2 E0
invar: TRUE
trans:
L: TRUE =C> TRY_1 ; RESET{ X1 }; goto 8
L: 4<=X2 =C> TRY_2 ; RESET{ X2 }; goto 2
L: 4<=X2 =C> ENTER_2 ; reset{}; goto 9

state: 6  /* vector state: < 2, 1, 0 > */
prop: WAIT_1 REQ_1 E0
invar: X2<=10
trans:
L: 4<=X1 =C> TRY_1 ; RESET{ X1 }; goto 4
L: X2<=10 =C> SETX_2 ; RESET{ X2 }; goto 10
L: 4<=X1 =C> ENTER_1 ; reset{}; goto 11

state: 7  /* vector state: < 3, 0, 1 > */
prop: CRITICAL_1 IDLE_2 E1
invar: TRUE
trans:
L: TRUE =C> TRY_2 ; RESET{ X2 }; goto 11
L: TRUE =C> SETX0_1 ; reset{}; goto 0
```

Figura A.16. Parte 2 del archivo "composición fischer.tg"

En la tabla A.4 se muestra la captura de las propiedades para este caso de estudio, propuestas en el capítulo 4, en formato Kronos. Los resultados obtenidos, de la verificación de estas propiedades en el modelo, se explican en el capítulo 4.

| Num | TCTL | Kronos |
|---|---|---|
| 1 | Inicio $\rightarrow$ AG($EF_{\geq 1}$ T ) | init impl ab (ed{=1} true ) |
| 2 | AG(Solicita1 $\rightarrow$ $AF_{\leq 100}$ Sección_Critica_1) | ab( req_1 impl ad{<=100}(critical_1) ) |
| 3 | AG(Solicita1 $\rightarrow$ $EF_{\leq 5}$ Sección_Critica_1) | ab( req_1 impl ed{<=5}(critical_1) ) |
| 4 | EF (Sección_Critica_1 $\wedge$ Sección_Critica_2) | ed( critical_1 and critical_2) |

Tabla A.4. Propiedades del protocolo de exclusión mutua de Fischer.

## A.5. Captura del protocolo de comunicaciones CSMA/CD

Este caso de estudio se verifica en Kronos. Las figuras A.17, A.18 y A.19 muestran los autómatas temporizados para los modelos del emisor1, emisor2 y el canal de comunicación respectivamente capturados en archivos de texto.

Al calcular con Kronos la composición paralela de los autómatas 'emisor1, 'emisor2' y 'canal' se obtiene el autómata mostrado en las figuras A.20 y A.21.

En la tabla A.5 se muestra la captura de las propiedades para este caso de estudio, propuestas en el capítulo 4, en formato Kronos. Los resultados obtenidos, de la verificación de estas propiedades en el modelo, se explican en el capítulo 4.

```
/* emisor1.tg */

#locs 3
#trans 9
#clocks x1
#sync begin1 end1 busy1 cd1

loc: 0
prop: wait_1
invar: true
trans:
true => send1 begin1; reset{x1}; goto 1
true => send1 busy1; reset{x1}; goto 2
true => send1 cd1;   reset{x1}; goto 2
true => cd1;         reset{x1}; goto 0


loc: 1
prop: transm_1
invar: x1<=808
trans:
x1=808 => end1 ; reset{x1}; goto 0
x1<52  => cd1  ; reset{x1}; goto 2


loc: 2
prop: retry_1
invar: x1<=52
trans:
x1<=52 => begin1; reset{x1}; goto 1
x1<=52 => busy1;  reset{x1}; goto 2
x1<=52 => cd1;    reset{x1}; goto 2
```

Figura A.17. Archivo "emisor_1.tg"

```
/* emisor2.tg */

#locs 3
#trans 9
#clocks x2
#sync begin2 end2 busy2 cd2

loc: 0
prop: wait_2
invar: true
trans:
true => send2 begin2; reset{x2}; goto 1
true => send2 busy2; reset{x2}; goto 2
true => send2 cd2;   reset{x2}; goto 2
true => cd2;         reset{x2}; goto 0


loc: 1
prop: transm_2
invar: x2<=808
trans:
x2=808 => end2 ; reset{x2}; goto 0
x2<52  => cd2  ; reset{x2}; goto 2


loc: 2
prop: retry_2
invar: x2<=52
trans:
x2<=52 => begin2; reset{x2}; goto 1
x2<=52 => busy2;  reset{x2}; goto 2
x2<=52 => cd2;    reset{x2}; goto 2
```

Figura A.18. Archivo "emisor_2.tg"

```
/* canal.tg */

#locs 3
#trans 9
#clocks y
#sync  begin1 begin2 end1 end2
       busy1  busy2  cd1 cd2

loc: 0
prop: idle
invar: true
trans:
true => begin1; y:=0 ; goto 1
true => begin2; y:=0 ; goto 1

loc: 1
prop: active
invar: true
trans:
y<26  => begin1 ; y:=0 ; goto 2
y<26  => begin2 ; y:=0 ; goto 2

y>=26 => busy1 ;  ; goto 1
y>=26 => busy2 ;  ; goto 1

true => end1 ; y:=0 ; goto 0
true => end2 ; y:=0 ; goto 0

loc: 2
prop: collision
invar: y<26
trans:
y<26 => cd1 cd2 ; ; goto 0
```

Figura A.19. Archivo "canal de comunicación.tg"

```
/* Timed graph generated for the parallel
composition of:
        sender1.tg , sender2.tg , bus.tg  */

#states 9
#trans  21
#clocks 3  X1    X2   Y

#sync
        BEGIN1 /* sender1 bus */
        END1 /* sender1 bus */
        BUSY1 /* sender1 bus */
        CD1 /* sender1 bus */
        BEGIN2 /* sender2 bus */
        END2 /* sender2 bus */
        BUSY2 /* sender2 bus */
        CD2 /* sender2 bus */


state: 0  /* vector state: < 0, 0, 0 > */
prop: WAIT_1 WAIT_2 IDLE
invar: TRUE
trans:
L: TRUE =C> BEGIN1 SEND1 ; RESET{ X1 Y
}; goto 1
L: TRUE =C> BEGIN2 SEND2 ; RESET{ X2 Y
}; goto 2

state: 1  /* vector state: < 1, 0, 1 > */
prop: TRANSM_1 WAIT_2 ACTIVE
invar: X1<=808
trans:
L: Y<26 =C> BEGIN2 SEND2 ; RESET{ X2 Y
}; goto 3
L: 26<=Y =C> BUSY2 SEND2 ; RESET{ X2 };
goto 4
L: X1=808 =C> END1 ; RESET{ X1 Y }; goto 0
state: 2  /* vector state: < 0, 1, 1 > */
prop: WAIT_1 TRANSM_2 ACTIVE
invar: X2<=808
trans:
L: Y<26 =C> BEGIN1 SEND1 ; RESET{ X1 Y
}; goto 3
L: 26<=Y =C> BUSY1 SEND1 ; RESET{ X1 };
goto 5
L: X2=808 =C> END2 ; RESET{ X2 Y }; goto 0
state: 3  /* vector state: < 1, 1, 2 > */
prop: TRANSM_1 TRANSM_2 COLLISION
```

Figura A.20. Parte 1 del archivo "composición csma.tg"

```
invar: X1<=808 and X2<=808 and Y<26
trans:
L: X1<52 and X2<52 and Y<26 =C> CD1 CD2 ;
RESET{ X1 X2 }; goto 6

state: 4  /* vector state: < 1, 2, 1 > */
prop: TRANSM_1 RETRY_2 ACTIVE
invar: X1<=808 and X2<=52
trans:
L: X2<=52 and Y<26 =C> BEGIN2 ; RESET{ X2 Y
}; goto 3
L: X2<=52 and 26<=Y =C> BUSY2 ; RESET{ X2 };
goto 4
L: X1=808 =C> END1 ; RESET{ X1 Y }; goto 7

state: 5  /* vector state: < 2, 1, 1 > */
prop: RETRY_1 TRANSM_2 ACTIVE
invar: X1<=52 and X2<=808
trans:
L: X1<=52 and Y<26 =C> BEGIN1 ; RESET{ X1 Y
}; goto 3
L: X1<=52 and 26<=Y =C> BUSY1 ; RESET{ X1 };
goto 5
L: X2=808 =C> END2 ; RESET{ X2 Y }; goto 8

state: 6  /* vector state: < 2, 2, 0 > */
prop: RETRY_1 RETRY_2 IDLE
invar: X1<=52 and X2<=52
trans:
L: X1<=52 =C> BEGIN1 ; RESET{ X1 Y }; goto 4
L: X2<=52 =C> BEGIN2 ; RESET{ X2 Y }; goto 5

state: 7  /* vector state: < 0, 2, 0 > */
prop: WAIT_1 RETRY_2 IDLE
invar: X2<=52
trans:
L: TRUE =C> BEGIN1 SEND1 ; RESET{ X1 Y };
goto 4
L: X2<=52 =C> BEGIN2 ; RESET{ X2 Y }; goto 2

state: 8  /* vector state: < 2, 0, 0 > */
prop: RETRY_1 WAIT_2 IDLE
invar: X1<=52
trans:
L: X1<=52 =C> BEGIN1 ; RESET{ X1 Y }; goto 1
L: TRUE =C> BEGIN2 SEND2 ; RESET{ X2 Y };

goto 5
```

Figura A.21. Parte 2 del archivo "composición csma.tg"

| Núm. | TCTL | Kronos |
|---|---|---|
| 1 | Inicio $\rightarrow$ AG($EF_{\geq 1}$ T ) | init impl ab (ed{>=1} true ) |
| 2 | AG((transm1 $\wedge$ transm2) $\rightarrow$ $AF_{\leq 52}$ (reintenta1 $\wedge$ reintenta2)) | ab( (transm_1 and transm_2 ) impl ad{<=52 } (retry_1 and retry_2)) |
| 3 | AG(colisión $\rightarrow$ $AF_{\leq 26}$ libre) | ab( collision impl ad{<=26}(idle) ) |
| 4 | AG((transm1 $\wedge$ $\neg$colisión) $\rightarrow$ $EF_{=26}$ $AF_{=782}$ espera1) | ab( (transm_1 and not collision) impl (ed{=26} ad {=782} wait_1 ) ) |

Tabla A.5. Propiedades del protocolo de comunicaciones CSMA/CD.

# CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
# UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Lógicas Temporales Lineales para la Verificación Formal de Sistemas de Tiempo Real

del (la) C.

Francisco MANZANO PINZÓN

el día 03 de Septiembre de 2007.

Dr. Arturo del Sagrado Corazón Sánchez Carmona
Investigador CINVESTAV 3B
CINVESTAV Unidad Guadalajara

Dr. Deni Librado Torres Román
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

Dr. Raúl Ernesto González Torres
Investigador CINVESTAV 2C
CINVESTAV Unidad Guadalajara