Autómatas estocásticos para sistemas concurrentes y tolerantes a fallas.

por

Raúl Enrique Monti

Presentado ante la Facultad de Matemática, Astronomía, Física y Computación como parte de los requerimientos para la obtención del grado de Doctor/a en Ciencias de la Computación de la

UNIVERSIDAD NACIONAL DE CÓRDOBA

Diciembre, 2018

Director: Pedro Ruben D'Argenio
Co-director: Holger Hermanns

Tribunal Especial:

Dr. Pablo Castro	Universidad Nacional de Río Cuarto
Dr. Diego Garbervetsky	Universidad de Buenos Aires
Dr. Nicolás Wolovik	Universidad Nacional de Córdoba
Dr. Nazareno Aguirre	Universidad Nacional de Río Cuarto (Suplente)

Resumen

El modelado por autómatas otorga un estructura matemática rigurosa al invaluable análisis de sistemas altamente *dependibles*. Se han definido muchas clases de autómatas, desde simples sistemas de transiciones hasta los complejos autómatas híbridos, pasando por otros como autómatas probabilistas, redes de Petry, etc. El principal propósito de un autómata es el dar una representación matemática de un sistema real con el fin de analizarlo. En este sentido, ser capaces de representar elecciones probabilistas es una característica de modelado deseable, dado que la gran mayoría de los modelos reales presentan algún tipo de comportamiento probabilista. Mucho de este comportamiento ocurre en el dominio continuo, cuando se modelan medidas físicas como el tiempo, la temperatura, energía, etc. Usualmente es técnicamente imposible analizar por medio del Model Checking Sistemas que presentan este tipo de complejidad, debido principalmente a la explosión de estados y la complejidad de los cálculos probabilistícos. Otras técnicas, como la simulación (también conocida como Model Checking Estadístico) se presentan como una alternativa, dado que evitan la construcción del espacio completo de estados. No se puede simular sistemas no-deterministas sin más. El no-determinismo debe ser resuelto previamente.

En esta tésis introducimos una nueva clase de automáta llamada Input/Output Stochastic Automata (IOSA), una restricción de Stochastic Automata con transiciones de entrada y salida. Definimos una primera versión de IOSA y le otorgamos semántica a partir de Non-deterministic Labeled Markov Process (NLMP). Definimos un operador de composición paralela y una noción de bisimulación. Demostramos que la composición paralela es una congruencia con respecto a dicha bisimulación. Finalmente demostramos que los modelos IOSA cerrados (modelos completamente generadores, i.e. sin acciones de entrada) son completamente deterministas y por lo tanto aptos para simulación por eventos discretos.

Una segunda versión de IOSA introduce al modelo acciones urgentes, con el fin de mejorar la composicionalidad del modelo, una característica muy apreciada en cualquier formalísmo de modelado. Esta extensión introduce no-determinismo también el los modelos cerrados. Logramos distinguir no-determinismo espurio producido por acciones confluentes. Primero demostramos que los modelos confluentes son débilmente deterministas en el sentido que sin importar la resolución del no-determinismo, el comportamiento estocástico es el mismo. Sumado a esto,

proveemos condiciones suficientes para asegurar que una red de IOSAs que interactúan, constituída posiblemente por componentes no confluentes, es confluente, sin la necesidad de analizar el modelo compuesto. Al hacer esto, abordamos las complicaciones de definir una forma débil de transición en un entorno continuo el cual es comúnmente evitado.

Finalmente, usamos IOSA con acciones urgentes para definir una semántica formal para Repairable Fault Trees (RFT), una técnica muy efectiva para el análisis de modelos industriales. De nuestro conocimiento, este es el primer trabajo sobre semántica para árboles de fallas dinámicos con reparaciones complejas que permiten distribuciones generales de fallas y reparaciones. Mas aún, demostramos que nuestros modelos de RFT son deterministas en ausencia de Spare gates, y aún también en presencia de un subconjunto de combinaciones de Spare gates y Spare Basic Elements, permitiendo su análisis por medio de simulación de eventos discretos. Un ejemplo de simulación de eventos raros en RFT usando la herramienta de simulación de eventos raros FIG (desarrollada en el grupo de sistemas dependibles de FAMAF) concluye este trabajo.

Abstract

Automata modeling gives a rigorous mathematical structure to the invaluable analysis of highly dependable systems. Many kinds of automata have been defined, ranging from the simple transition systems to the complexity of hybrid automata, all the way through probabilistic automata, Petri nets and many others. The main purpose of automata is to give a mathematical representation of a real system for the purpose of its analysis. Being able to represent probabilistic election is a desired modeling feature since most real life models include some kind of probabilistic behavior. Much of this behavior happens on the continuous domain, when modeling for physical measures such as time, temperature, energy, etc. Complex systems that present such behaviour are usually impossible to analyze by Model Checking techniques given the state space explosion. Other techniques, such as simulation (also known as Statistical Model Checking), become an alternative, given that they avoid the construction of the full state space. No real simulation can be taken over a non-deterministic model as it is. Non-determinism has to be resolved.

In this thesis we introduce a new class of automata named Input/Output Stochastic Automata (IOSA), a restriction of Stochastic Automata with input and output transitions. We define a first version of IOSA and we give it semantic on Non-deterministic Labeled Markov Process (NLMP). We define a parallel composition operator and a notion of bisimulation. We show that parallel composition is a congruence with respect to bisimulation. Finally we demonstrate that closed IOSA models (fully generative models, i.e. without input actions) are fully deterministic and thus amenable to discrete event simulation.

A second version of IOSA introduces urgent actions to the model in order to enhance compositional modeling, a most desired characteristic in any modeling formalism. This extension introduces non-determinism even into the closed model. We are able to tell apart spurious non-determinism produced by confluent actions. We first show that confluent models are weakly deterministic in the sense that, regardless the resolution of the non-determinism, the stochastic behaviour is the same. In addition, we provide sufficient conditions to ensure that a network of interacting IOSAs, constructed with possibly non-confluent components, is confluent, without the need to analyse the larger composed IOSA. In doing so, we address the complications of defining a particular form of weak transition on a continuous setting that is normally elusive.

Finally we use IOSA with urgency to define a formal semantics for Repairable Fault Trees (RFT), a prominent technique for analyzing industrial models. From what we know, this is the first work on semantics for Dynamic Fault Trees with complex repairs that allows for general distributions of failure and repairs. Furthermore our RFT models are shown to be deterministic in the absence of spare gates, and even in the presence of a subset of combinations of spare gates and spare basic elements, making them amenable to discrete event simulation. An example of rare event simulation on Repairable Fault Trees using the rare event simulation tool FIG (developed in the Dependable Systems group at FAMAF) closes this work.

A Flor, Tere y Cati. A Mamá y Papá.
A María Luz, Pancho, Roli y Vero.
A Dios y María.

Agradecimientos

Quiero agradecer a muchos y recordar –volver a pasar por el corazón– a muchos otros, sin un orden en particular, que me acompañaron y formaron parte de mi camino de aprendizaje en la FAMAF que eventualmente llevó a la escritura de esta tesis.

Llegué a mis primeras clases en la facultad acompañado de dos grandes amigos, primos entre ellos, Pablo y Nico. Con ellos formamos nuestros primeros grupos de laboratorio y nos juntamos a estudiar muchas veces. Recuerdo algunas tarde que se volvían noches programando juntos para las materias de Redes y de Sistemas Operativos. Con ellos fuimos conociendo muchos nuevos amigos, y el grupo se fue ampliando de a poco. En un momento éramos suficientes para armar dos equipos de futbol y jugar en las canchitas de la universidad, atrás de Economía, o en Don Balón, un poco mas lejos. Después de un par de años, el grupo se fue achicando y la amistad creciendo. Las anécdotas son muchas, pero mejor no escribir un libro acá. Quiero nombrarlos a todos eso si: Charlos, Agu, Mariano, Adriano, Juli, Ramón, Pana, Cancu, Mallku, Jairo, y Pablito.

Recuerdo el último tiempo de mi licenciatura. De repente me di cuenta que necesitaba preparar el trabajo final, y lo encontré a Pedro para sacarle ideas. Arranqué el trabajo, pero a mitad de camino aparecieron mis ex ayudantes alumnos Mati, Gastón y Eric y me endulzaron con la idea de la ICPC. Dejé a un lado la defensa del trabajo, entrenamos, y llegamos a la final en Varsovia. Recuerdo que una persona muy especial, Leopoldo, nos dio un empujoncito con una carta al director de la ICPC.

Volví, terminé la licenciatura, y, unos días antes del cierre a presentación de proyectos de doctorado, toqué la puerta de la oficina de Pedro :D Carlos ya estaba haciendo su doctorado con Pedro, así que seguimos aprendiendo juntos. Recuerdo nuestras escuelas de verano en Rio Cuarto, las reuniones de FACAS, algunos asados. Compartimos mucho con Chun, con Silvia, Laura, Raúl, Damián. Con Chun también trabajamos, y paseamos por Madrid para alguna conferencia.

Viajé en tres ocaciones a Saarbrueken, Alemania, con Carlos y con Silvia.

Allá nos recibieron Holger y su grupo, el DSG de la UNI de Saarland. Fuimos muy bien recibidos, hicimos muchas amistades, y recuerdo con cariño a mucha gente allá. Arnd fue a buscarnos a la terminal. Christa nos solucionó la vida más de una vez. Conocimos a Luis, el argentino del grupo. Nos hicimos grandes amigos fuera del grupo con Koni y Maru.

La facu me dio una oficina, la 272, y a dos compañeros de oficina: Mariano y José. Lucho y Carde se sumaron como anexos y nos visitaban seguido. Compartimos grandes locros, y juegos de rol. Me mudé a Nueva Córdoba junto a la Dra. Luz Monti, al lado de un cheboli, cosa difícil de esquivar en Nva. Cba. En algún momento conocí a Flor, que por ese entonces estudiaba su licenciatura en química cerquita de FAMAF. Nos casamos, y un 16 de Diciembre de 2018 defendí mi tesis en el auditorio de nuestra facultad.

Por supuesto, quiero agradecer a los excelentes docentes que tuve en mi carrera, su enseñanza y dedicación es impagable. Algunos nombres, de aquellos que sentí mas cercanos, pero sin desvalorar a los demás: Nico, Naza, Pedro x 2, Laura x 2, Damián, Daniel, Javier. El agradecimiento mas grande aquí se lo lleva Pedro, por supuesto, por ser no solo un gran director sino también un gran amigo. Con él seguimos compartiendo trabajo y encuentros. Espero sigamos así por mucho tiempo más. Tampoco puedo dejar de agradecer a mi país, Argentina, que apuesta por una universidad pública y gratuita, que me dio la posibilidad de prepararme con el más alto estándar en educación.

El agradecimiento final va a Mamá y Papá, por todo lo que entregaron, toda su vida, para que podamos tener salud, educación y sobre todo amor.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Los sistemas embebidos reactivos están presentes en la mayoría de nuestras actividades diarias. Desde actividades relativamente simples como controlar las luces en nuestras calles, regar nuestros jardines, lavar nuestra ropa, o prepararnos café, hasta actividades mucho más complejas como controlar el vuelo de los aviones, manejar los satélites, permitir el uso de sistemas médicos complejos, enrutar nuestro internet al resto del mundo, asegurar nuestros autos y edificios, controlar plantas de energía, etc. El software embebido esta en el centro de muchos sistemas importantes en los cuales confiamos nuestra seguridad, salud y dinero. Por lo tanto, podemos decir que existe una gran necesidad de asegurar su correcto funcionamiento.

El uso de robots en la industria ya ha significado una revolución en cómo se producen las cosas. El llamado "Internet de las cosas" nos presenta un futuro en el que objetos de uso común, como lámparas, termómetros, secadores de pelo y otros electrodomésticos están conectados a Internet. No es difícil encontrar estos objetos de esta clase ya en el mercado. Este no era el caso hace un tiempo cuando la mayoría de estas actividades eran realizadas por circuitos electrónicos más simples, que no presentaban mucho interés para el análisis dada su simplicidad, o no se realizaban en absoluto. Hoy en día, una parte importante del mercado de microprocesadores se llena con microcontroladores que son el núcleo programable de los sistemas integrados. Un automóvil de rango medio típico tiene aproximadamente 40 microcontroladores, mientras que los automóviles de lujo pueden tener hasta 150. La mayoría de los aparatos electrónicos para el hogar llevan uno o más microcontroladores. Estos microcontroladores son generalmente más simples que la mayoría de los procesadores conocidos que se encuentran en nuestros teléfonos celu-

lares o computadoras, ya que usualmente tienen una actividad específica que realizar en contraste con la utilidad general de una PC.

Uno de los principales desafíos en el análisis de software integrado es la interacción continua de estos sistemas con el mundo físico. Las computadoras clásicas tenían como objetivo interactuar con un teclado y un monitor como objetos de entrada y salida. Este no es el caso de la mayoría de los sistemas integrados que, en cambio, están conectados a sensores a través de los cuales perciben datos de entrada, y responden al mundo físico por medio de diferentes y heterogéneos actuadores. Además, los sistemas integrados son mucho más complejos y exigentes un mayor análisis que los circuitos electrónicos que solían realizar su actividad de una manera más ingenua. Algunas características especiales de estos sistemas nos ayudarán a introducir las necesidades, y luego los medios, para analizar su comportamiento:

No es esperable que los sistemas integrados puedan actualizarse ni ajustarse una vez que estén siendo usados. No es el caso que podamos actualizar el software en nuestro horno de microondas, o que podamos tomar nuestro refrigerador y hacer que haga algo nuevo con lo que tiene. Por lo general, es difícil detener una central nuclear y actualizar los sensores y microcontroladores encargados de controlar su correcto funcionamiento, o cambiar el comportamiento de los microcontroladores en un satélite. Los sistemas integrados deben ser correctos y estar bien desde el principio, y permanecer sin cambios hasta el final de su funcionamiento.

Los sistemas integrados están en constante e interminable interacción con su entorno. Pertenecen a lo que Pnueli describe como *sistemas reactivos* [78]. Durante su interacción con el mundo físico, las restricciones en tiempo real y la concurrencia se convierten en un tema muy necesario de abordar. La representación habitual de este tipo de comportamiento de los sistemas es la de un ciclo sin fin, en el que el sistema escucha las entradas del entorno físico, de comportamiento concurrente, y reacciona respondiendo en consecuencia y escuchando nuevamente. Esto contrasta con otro tipo de sistemas que se detienen inmediatamente después de transformar los datos de entrada en un resultado, conocidos como sistemas de transformación. Esta es también una de las razones por las que la medición del tiempo es importante y medir la disponibilidad del sistema es de gran importancia, ya que el sistema tiene que mantenerse al día con la velocidad y las solicitudes de un entorno formado por un mundo físico concurrente y muy heterogéneo.

Los sistemas integrados deben ser altamente confiables. No hay duda de que el riesgo que implica el mal comportamiento de estos sistemas es muy alto. Imagine los sensores de aterrizaje de un avión que fallan, o los sensores de calor de una planta de energía fallando. Imagine las pérdidas de dinero si los microcontroladores de una línea de autos nuevos están defectuosos y todos los autos vendidos deben ser devueltos para ser reparados. Estos sistemas requieren que se verifiquen no solo la corrección, sino también la disponibilidad y la eficacia. No es el caso que un cajero automático le haga esperar media hora para darle el dinero que solicitó. No debería ser el caso de que un solo microcontrolador en un satélite consuma en promedio la mitad de la energía disponible de la batería. Por lo tanto, no solo se requieren aspectos cualitativos relacionados con la corrección, sino que también se desean cumplir con medidas cuantitativas relacionadas con la efectividad y la disponibilidad.

1.2. Necesidad y medios para el análisis formal

Hemos dejado en claro la necesidad de un análisis riguroso y eficiente del software integrado. Se puede deducir, por supuesto, que cuanto antes detectemos el problema, más reduciremos los costos de resolverlo. Por lo tanto, la fase de diseño de un producto de software es de gran interés para la verificación. Se imponen altos requisitos en el rendimiento y la confiabilidad, lo que hace que sea necesario encontrar los métodos correctos y efectivos para analizar este tipo de software, teniendo en cuenta no solo su naturaleza, sino también su entorno de competencia para la comunicación. Se han propuesto muchos formalismos para este propósito, que abarcan procesos álgebras de precesos estocásticos [66, 40, 65], variaciones de Petri Nets [83, 82], o extensions propias de autómatas tal como autómatas temporizados [1, 63], autómatas probabilistas [84] y autómatas híbridos [62]. La teoría de autómatas nos ha proporcionado una manera efectiva y eficiente de verificar formalmente la confiabilidad y la corrección del diseño del software. El análisis formal se ha centrado históricamente en el análisis cualitativo, es decir, determinar si una propiedad de corrección se satisface o no mediante un modelo abstracto del sistema. Sin embargo, para que el sistema proporcione un servicio eficiente y confiable, se requiere el análisis de propiedades cuantitativas como la confiabilidad, el rendimiento, el tiempo medio hasta el fallo, etc. Por ejemplo, no solo queremos saber si un sistema está libre de errores, sino también si puede responder en un lapso de tiempo adecuado mientras consume una cantidad

limitada de energía.

El tiempo es un requisito importante cuando se modela el comportamiento de los sistemas físicos. Es interesante modelar con precisión, tal vez, la tasa de falla de un componente del tren para analizar el tiempo promedio hasta la falla del tren en sí. La confiabilidad generalmente exige alcanzar objetivos de probabilidad que, en muchos casos, se cuantifican en un espacio continuo que responde a la naturaleza física del entorno de los sistemas.

Model Checking [33, 79, 6] ha sido durante mucho tiempo una importante técnica rigurosa para verificar exhaustivamente el correcto comportamiento de un modelo. Sin embargo, la creciente complejidad de los modelos de interés expone las limitaciones del Model Checking.

Simulation [74, 106, 32], also known as statistical model checking, offers an alternative by giving tight probability bounds without the need to construct nor visit the whole state space, avoiding the state space explosion problem. Regardless which of this techniques we decide to use, a common pattern has to be followed: a model has to be obtained as a formal specification of the design, plus a formal specification of the desired properties of the model should be written down using some kind of logic. The model has to be analyzed against this properties using one of the mentioned methods. The expressiveness of the modeling language is critical for the analysis of the desired properties.

En general, no es posible analizar sistemas estocásticos con distribuciones continuas generales mediante el uso de Model Checking. La simulación [74, 106, 32], también conocida como Model Checking estadístico, presenta una alternativa al ofrecer límites de probabilidad arbitrariamente ajustados sin la necesidad de construir ni visitar todo el espacio de estados, evitando el problema de la explosión de estados. Independientemente de cuál de estas técnicas decidamos utilizar, se debe seguir un patrón común: se debe obtener un modelo como una especificación formal del diseño, además de una especificación formal de las propiedades deseadas sobre el modelo, las cuales deben ser expresadas usando algún tipo de la lógica. Luego, el modelo debe ser analizado contra estas propiedades usando uno de los métodos mencionados. La expresividad del lenguaje de modelado es crítica para el análisis de las propiedades deseadas.

Para mantenerse al día con las necesidades actuales de análisis de confiabilidad de sistemas, se vuelve crucial unir estas dos técnicas al darle a estos lenguajes livianos las bases matemáticas firmes de la verificación y simulación de modelos.

1.3. Contribución

En esta tesis contribuimos en el modelado y verificación de sistemas estocásticos con distribuciones generales. Históricamente, el modelado para la simulación y la verificación de modelos se ha centrado en modelos Markovianos. Los algoritmos se han estudiado y optimizado para este tipo de modelos en los que, en el caso continuo, el tiempo y otras medidas se rigen por la distribución exponencial. Sin embargo, es más frecuente que encontremos que el mundo físico no responde a este tipo de distribución. Stochastic Automata (SA) [40, 43] presenta un marco para el modelado composicional de sistemas estocásticos de distribución general, eliminando la restricción Markoviana. Extendiendo SA, proponemos un nuevo lenguaje de modelado formal para describir sistemas donde los eventos se rigen por distribuciones de probabilidad continuas generales. El lenguaje se llama *Input/Output Stochastic Automata* (IOSA) y su semántica se construye sobre las rigurosas bases matemáticas de Non-deterministic Labeled Markov Process (NLMP) [46, 102] . Su propósito es servir como lenguaje de modelado para el análisis de sistemas utilizando técnicas de simulación. Por esta razón, es importante asegurarse de que el modelo bajo análisis sea determinista (solo los sistemas deterministas están sujetos a simulación real) en una primera versión, mientras que se dan las condiciones suficientes para un determinismo débil en su segunda versión.

IOSA ha sido diseñado como un lenguaje de descripción composicional. Esto atiende a varias preocupaciones, tales como evitar los problemas de explosión de estado que reducen en gran medida la eficiencia e incluso la posibilidad de alcanzar el análisis deseado. Además, la composicionalidad permite la reutilización de componentes y modelos, al disociar el comportamiento del sistema en componentes independientes. Además, ayuda en todo el proceso de ingeniería al ofrecer una comprensión más intuitiva y clara del sistema, al tiempo que lo modela como componentes independientes arbitrariamente simples con puntos de comunicación precisos. Esto contrasta con los enormes modelos monolíticos, donde la comprensión del flujo de control se vuelve difícil y los errores en el diseño se vuelven muy comunes.

Finalmente, hacemos contribuimos a cerrar la brecha con la ingeniería de análisis y la verificación del rendimientoa a nivel industrial , mediante la definición de una semántica rigurosa para los árboles con fallas dinámicas en la reparación (RFT) en términos de nuestra lenguage IOSA. Probamos que los modelos escritos en el formalismo RFT son débilmente deterministas en el sentido de que el único no determinismo presente en ellos es espurio y, por lo tanto, no importa cómo decidamos resolverlo, no afectará el resultado final de las propiedades a analizar. Este determinismo nos permite simular

sobre los modelos IOSA de RFT. Hoy en día, los sistemas deben tener un alto grado de resiliencia y confiabilidad. La evaluación de propiedades que fallan con una probabilidad extremadamente pequeña en modelos complejos puede ser computacionalmente muy exigente. La simulación de eventos raros es una alternativa eficaz a la simulación de Monte Carlo la cual es más ingenua al realizar este tipo de análisis exigente. Analizamos un caso de estudio utilizando la herramienta de simulación de eventos raros FIG [31, 28, 25], que permite modelar y analizar efectivamente los sistemas tolerantes a fallas modelados en lenguaje IOSA.

1.4. Trabajo relacionado

Existen variados lenguajes de modelado formal para sistemas estocásticos con distribuciones continuas generales. De hecho, la mayor parte de este trabajo es una extensión de Stochastic Automata (SA) [40, 42, 43], que fueron motivados como una alternativa a los autómatas probabilísticos para ser capaces de representar simbólicamente la naturaleza infinita de los sistemas de comportamiento estocástico. Inspirado por Generalized Semi-Markov Process y Timed Automata, SA introduce variables aleatorias llamadas relojes que determinan en qué momento ocurrirá un evento mediante el muestreo de una distribución de probabilidad continua arbitraria. En [42] el modelo se presenta como un framework composicional con un comportamiento abierto y cerrado, y se sugieren diferentes tipos de bisimulación. Los SA no son deterministas, por lo que es un inconveniente para la simulación, ya que el no determinismo debe resolverse interviniendo el modelo o mediante el uso de schedulers definidos por un experto en el modelo. En cualquier caso, esta es una actividad propensa a errores y, por lo tanto, es necesario pensar y modelar los sistemas de una manera totalmente probabilística desde su construcción. IOSA es un intento de hacerlo. Otros trabajos en esta dirección son la tesis de Bravetti sobre la especificación y el análisis de sistemas temporizados estocásticos a través de las álgebras de proceso [21] y su trabajo con D'Argenio [22] donde el comportamiento estocástico se generaliza a distribuciones continuas arbitrarias. También tomamos varias ideas del trabajo [103] para la definición y análisis del comportamiento composicional asíncrono de IOSA.

Contemporáneos a SA, encontramos las Interactive Markov Chains (IMC) [64] y su versión con entradas y salidas Input/Output Interactive Markov Chains (I/O-IMC) [36], de las cuales inspiramos nuestro trabajo sobre confluencia y el determinismo débil para la versión urgente de IOSAİ/O-IMC también está orientado al modelado composicional, como una forma eficiente

de evitar el problema de explosión de estado y como una práctica de ingeniería limpia. La principal diferencia con nuestro trabajo es que I/O-IMC solo permite distribuciones Markovianas en sus transiciones, es decir, solo distribuciones exponenciales en el caso de espacios continuos (aunque sugiere utilizar distribuciones de fase para aproximar otras clases de distribuciones), en contraste con el soporte nativo para distribuciones generales continuas de IOSA.

Para definir la bisimulación en IOSA tomamos ideas de [45, 46, 10, 48]. Más precisamente utilizamos una de las bisimulaciones sobre NLMP definidas en [45]. El operador de composición paralela en IOSA se define en base a ideas de [52, 45, 46, 76]. De hecho, se parece mucho al operador de composición paralela definido para Stochastic Automata en [40], pero toma el enfoque generativo-reactivo, así como el concepto de input-enableness, de [76].

Existen muchos intentos de dar una semántica formal a una de las muchas variantes de Fault Trees [51, 69, 12, 86, 9, 17], y muchas otras. La principal diferencia entre nuestro trabajo en el capítulo 5 y estos otros trabajos, es que o bien no consideran el modelo de reparación, o bien se limitan a los modelos Markovianos, o bien no aseguran un modelado determinista. Uno de los trabajos más completos es [18]. Presenta la formalización de un marco completo de Árbol de Fallas que incluye un modelo de Reparación, lo hace de forma agregativa para reducir el tamaño de la cadena de Markov resultante, pero basándose en I/O-IMC como su semántica y por lo tanto concierne sólo a modelos Markovianos. Otro modelo que comprende una versión reparable de árboles de falla es [12] Nuevamente, se restringe a modelos Markovianos utilizando Redes de Petri como su semántica, resultando exitoso en reducir el espacio de estado del modelo en comparación con un enfoque de Cadena Markov pura. Un trabajo menos completo pero similar es [17], donde los Árboles Dinámicos de Fallas se formalizan por medio de I/O-IMC con las restricciones que ya hemos mencionado. Algunos otros trabajos recientes hacen un intento efectivo de formalizar Árboles de Fallas. Este es el caso de [70] que presenta una unificación de varios formalismos conocidos de Árboles de Fallas en la literatura, que pueden obtenerse seleccionando adecuadamente las prioridades y el tratamiento no determinista. En las Redes de Petri estocásticas generalizadas se da una semántica de composición, pero no trata el modelo reparable, ni probabilidades continuas generales: sólo se permite la distribución exponencial. En nuestro trabajo, formalizamos los Árboles de Fallas y definimos una semántica clara para ellos hasta la extensión de Fault Trees reparables, lo que cambia completamente la lógica de los modelos e introduce varias fuentes nuevas de no determinismo y subespecificación, en contraste con versiones mas simples de Fault Trees. Es importante destacar que en ninguna de los trabajos mencionados se han asegurado modelos de-

terministas. En cambio, mostramos que nuestra semántica asegura modelos RFT débilmente deterministas, en el sentido de que el único no-determinismo presente en el modelo es espurio y no afecta el comportamiento estocástico y por lo tanto la evaluación de la probabilidad de los eventos a evaluar.

Furthermore we present a compositional semantic for the FT models, and we symbolically execute simulations on the IOSA level avoiding the state space explosion. Good surveys on fault trees and their formal specification can be found at [92]. Finally we have not treated maintenance models as [90, 91] do. Maintenance models are an interesting approach to fault tolerance treatment reason why we are living this topic as a future work.

Además presentamos una semántica de composición para los modelos FT, y ejecutamos simulaciones simbólicas a nivel IOSA evitando la explosión del espacio de estados. Se pueden encontrar una buena revisiones sobre árboles de fallas y sus especificaciones formales en [92]. Por último, no hemos tratado los modelos con mantenimiento como lo hacen [90, 91]. Los modelos con mantenimiento son un enfoque interesante para el tratamiento de la tolerancia a fallos, por lo que tomamos este tema como un trabajo a futuro.

1.5. Organización de la tésis

Además de la introducción y las discusiones finales, esta tesis está organizada como sigue:

El capítulo 2 presenta los antecedentes generales sobre *autómatas* y *teoría de la medida*, que son la base de la mayor parte de los resultados teóricos de la tesis. Más precisamente: Sección 2.1 presenta algunos principios de la teoría de la medida y probabilidades, junto con algunas definiciones importantes sobre determinados σ-algebras y medidas de probabilidad. La sección 2.2 presenta los *Non Deterministic Labeled Markov Process*, que se utilizan como la semántica concreta de IOSA y IOSA$_u$. También revisamos la noción de bisimulación en estos procesos.

El capítulo 3 introduce una primera versión de Input/Output Stochastic Automata. Las secciones 3.1 y 3.2 recuerdan trabajos y conceptos anteriores sobre modelado y análisis de sistemas concurrentes estocásticos, tales como *concurrencia*, *composicionalidad*, y modelado de eventos temporales arbitrariamente distribuidos en modelos concurrentes. Estos conceptos motivan el trabajo en el nuevo modelo de autómatas IOSA. Se define IOSA en la Sección 3.3. La semántica de IOSA en términos de NLMP en la Sección 3.4. En la Sección 3.5 definimos la composición paralela y la bisimulación de IOSA.

También demostramos que la bisimulación es una congruencia para la composición paralela. En la sección 3.6 probamos que los IOSAs cerrados son deterministas y por lo tanto aptos para la simulación por eventos discretos.

Capítulo 4. En este capítulo extendemos IOSA con acciones urgentes. La sección 4.1 define a IOSA$_u$La sección 4.2 define la semántica de IOSA$_u$ en términos de NLMP. La sección 4.3 define la composición paralela en IOSA$_u$. Los IOSA$_u$ cerrados son no deterministas en el caso general. Confluencia sobre el IOSA$_u$ se define en la Sección 4.4. En la Sección 4.5, definimos el concepto de determinismo débil y mostramos que todo IOSA$_u$ confluente y cerrado es débilmente determinista. Finalmente, en la Sección 4.6 encontramos las condiciones suficientes para asegurar que una red de componentes posiblemente no confluentes sea en verdad confluente. También presentamos un algoritmo de tiempo polinomial para verificar estas condiciones.

Capíutlo 5. En este capítulo damos semántica a los Árboles de Fallas con Reparación (RFT) en términos de IOSA$_u$. El capítulo comienza con una discusión sobre el estado del arte en el análisis de árboles de fallas. También motiva la necesidad de permitir distribuciones continuas arbitrarias junto con reparaciones complejas en Fault Trees. La sección 5.3 presenta un lenguaje simbólico para describir modelos IOSA$_u$En la sección 5.4 damos una definición formal de RFT. Utilizamos el lenguaje simbólico de IOSA$_u$ para proporcionar una semántica formal para RFT en la Sección 5.5. En la Sección 5.6, mostramos que cualquier modelo RFT induce un IOSA$_u$ débilmente determinista. Extendemos la semántica RFT con cajas de reparación en la Sección 5.7. En la Sección 5.8, presentamos un pequeño caso de estudio que analizamos utilizando el simulador FIG.

Capítulo 2

Preliminares

2.1. Probabilidad y teoría de la medida

En esta sección presentamos brevemente los principales conceptos de probabilidad probabilidad y teoría de la medida que utilizaremos para sustentar el desarrollo de nuestros resultados de mayor importancia en la tesis. La Teoría de la Medida nos permite analizar el comportamiento de nuestros autómatas que evolucionan en un espacio continuo y toman decisiones probabilísticas sobre el camino a seguir. Es una base matemática sólida sobre la que podemos definir y justificar nuestros resultados. La mayor parte del contenido de esta sección sigue [3].

2.1.1. σ-algebras (o σ-fields)

La Teoría de la Medida trata el problema de cuantificar los posibles resultados de un experimento. Dado el conjunto de posibles resultados S, una medida μ es una función establecida en los subconjuntos de S, es decir, $\mu : S^2 \to \mathbb{R}^+$. No todos los subconjuntos pueden ser medidos, y llamaremos eventos a aquellos subconjuntos que puedan. Un σ-algebra da una estructura a los conjuntos de eventos:

Definición 2.1. Sea Σ una colección de subconjuntos de un conjunto S. Se dice que Σ es una σ-algebra si $S \in \Sigma$ y Σ es cerrada bajo complemento y unión contable. Es decir:

- $S \in \Sigma$.
- si $Q \in \Sigma$, entonces $Q^c \in \Sigma$.
- si $Q_1, Q_2, ... \in \Sigma$, entonces $\bigcup_{i=1}^{\infty} Q_i \in \Sigma$.

La intuición sobre esta definición es que si un conjunto de Q es medible entonces para cualquier resultado del experimento en $\omega \in S$ deberíamos ser capaces de saber si ω está en Q. Si es así, entonces podemos decir si ω está en Q^c. Más aún, la respuesta a "¿está ω en S?" es siempre Verdadera y por lo tanto S es un evento. Por último, si podemos responder si ω está en Q_i para $i \geq 1$, entonces podemos responder si ω está en $\bigcup_i Q_i$, y lo mismo para su intersección.

Sea Q un subconjunto propio no vacío de S. Entonces se puede probar fácilmente que $\{\varnothing, S, Q, Q^c\}$ es el más pequeño σ-algebra que contiene Q. Para un conjunto $\mathcal{G} \subseteq 2^S$ de subconjuntos de S, escribimos $\sigma(\mathcal{G})$ para denotar el mínimo σ-algebra que contiene $\mathcal{G}$. Llamamos a cada elemento en $\mathcal{G}$ un generador, llamamos $\mathcal{G}$ al grupo generador, y $\sigma(\mathcal{G})$ el σ-algebragenerado.

Un σ-algebra muy útil es la σ-algebra de Borel, que es generada por el conjunto de todos los conjuntos abiertos en una topología. En particular, la σ-algebra de Borel en la línea real es $\mathscr{B}(\mathbb{R}) = \sigma(\{(a, b)|a, b \in \mathbb{R} \text{ y } a < b\})$. De manera similar, $\mathscr{B}([0, 1])$ es la σ-algebra de Borel en el intervalo $[0, 1]$ generado por los conjuntos abiertos en el intervalo $[0, 1]$. Una construcción equivalente se obtiene utilizando el conjunto de todos los intervalos semicerrados a la derecha o el conjunto de todos los intervalos semicerrados a la izquierda como generadores.

Llamaremos a cada $Q \in \Sigma$ un conjunto medible, y generalmente adjuntaremos la σ-algebra a sus espacios bases correspondientes y llamaremos a la tupla (S, Σ) un espacio medible, siendo $(S, \{\varnothing, S\})$ el más pequeño y $(S, \mathcal{P}(S))$ el más grande.

Sean (L, Λ) y (S, Σ) espacios medibles. Un *rectángulo medible* es un conjunto $Q \times B$ con $Q \in \Lambda$ y $B \in \Sigma$. La *σ-algebra prodcuto* en $L \times S$ es el más pequeño σ-algebra que contiene todos los rectángulos medibles, y se denota por $\Lambda \otimes \Sigma$. La *σ-algebra coproducto* $\Lambda \oplus \Sigma$ de L y S se define como la unión disjunta $L \uplus S$ y se genera mediante el conjunto $\Lambda \cup \Sigma$. Si una colección de espacios medibles (S_i, Σ_i) para $i = 1...n$ coinciden en su σ-algebra digamos Σ, entonces denotamos su σ-algebra producto con Σ^n.

2.1.2. Medidas de probabilidad

Una medida en un σ-algebra Σ es una función no negativa, con valores reales μ en Σ, de manera que es σ-aditiva, es decir,

$$\mu(\bigcup_{i \in \mathbb{N}} Q_i) = \sum_{i \in \mathbb{N}} \mu(Q_i)$$

para toda la familia contable de conjuntos medibles de pares separados $\{Q_i \mid i \in \mathbb{N}\} \subseteq \Sigma$. Si además $\mu(S) = 1$ entonces μ se llama *medida de probabilidad*,

en cuyo caso $\mu : \Sigma \to [0, 1]$. A menudo usaremos la medida de probabilidad de Dirac concentrada en un punto $a \in S$, definida para cada $Q \in \Sigma$ como:

$$\delta_a(Q) = \begin{cases} 1 & \text{si } a \in Q \\ 0 & \text{si } a \notin Q \end{cases}$$

La siguiente construcción será útil para definir y analizar medidas en álgebras sigma del producto. Dadas las medidas μ y μ' en (L, Λ) y (S, Σ) respectivamente, el producto mide $\mu \times \mu'$ en el espacio del producto $(L \times S, \Lambda \otimes \Sigma)$ se define como la medida única tal que $(\mu \times \mu')(Q \times B) = \mu(Q) \cdot \mu'(B)$ para todos $Q \in \Lambda$ y $B \in \Sigma$. Además, cualquier medida μ en (L, Λ) se puede extender naturalmente a una medida $\hat{\mu}$ en el espacio de coproducto $(L \uplus S, \Lambda \oplus \Sigma)$ tomando $\hat{\mu}(Q) = \mu(Q \backslash S)$, y de manera similar para las medidas en (S, Σ).

Sea $\Delta(S)$ tal que denote el conjunto de todas las medidas de probabilidad sobre el espacio medible (S, Σ). Permitimos que μ, μ', μ_1, ... denoten elementos en $\Delta(S)$. Será conveniente dotar a $\Delta(S)$ con un σ-algebra. Para esto usaremos una construcción estándar de Giry [56], donde $\Delta(\Sigma)$ se define como el σ-algebra generado por los conjuntos de medidas de probabilidad $\Delta^{\geq p}(Q) \doteq \{\mu \mid \mu(Q) \geq p\}$, con $Q \in \Sigma$ y $p \in [0, 1]$. Permitimos que ξ y ζ varíen sobre $\Delta(\Sigma)$. Vale la pena señalar que, si el σ-algebra subyacente es generado por un sistema π innumerable, cada singleton $\{\mu\}$ es medible en la σ-algebra de Giry . Además, se garantiza que $\Delta(\Sigma)$ separa los puntos, es decir, si hay dos medidas $\mu \neq \mu'$ entonces hay un generador que los distingue, o equivalentemente, existe $Q \in \Delta(\Sigma)$ tal que $\mu \in Q$ y $\mu' \notin Q$.

2.1.3. Funciones medibles e integrales de Lebesgue

Sean (S_1, Σ_1) y (S_2, Σ_2) dos espacios medibles. Se dice que una función $f : S_1 \to S_2$ es *medible* si para cualquier $Q_2 \in \Sigma_2$, $f^{-1}(Q_2) \in \Sigma_1$, es decir, su imagen inversa asigna conjuntos medibles a conjuntos medibles. Es estándar escribir $f : (S_1,\ sigmaA_1) \to (S_2, \Sigma_2)$ para denotar que la función f es medible. Tenga en cuenta que las funciones $f : S \to S'$ conserva los complementos y las uniones arbitrarias (en particular contables),

$$f^{-1}(X^c) = (f^{-1}(X))^c$$

$$f^{-1}(\bigcup_i X_i) = \bigcup_i f^{-1}(X_i)$$

Por lo tanto, para probar que una función se puede medir desde (S_1, Σ_1) a (S_2, Σ_2), es suficiente demostrar que su inverso asigna cada conjunto de $\mathcal{G}$ a un conjunto medible en Σ_1, donde $\mathcal{G}$ es un conjunto generador de Σ_2.

A menudo será útil construir medidas de probabilidad a partir de otras medidas de probabilidad. Dado que estamos trabajando en terrenos continuos, haremos uso de integrales. Además, dada su generalidad, utilizaremos la integral de Lebesgue en lugar de la integral de Riemann.

Definición 2.2 (Integral de Lebesgue de funciones simples)**.** Sea $h : (S, \Sigma) \to (\mathbb{R}^+, \mathscr{B}(\mathbb{R}^+))$ una función simple, sean $h = \sum_{i=1}^{n} x_i A_i$, para A_i conjuntos disjuntos en Σ. Entonces definimos la integral de Lebesgue de h con respecto a μ:

$$\int_S h \ \delta\mu = \sum_{i=1}^{n} x_i \mu(A_i)$$

Dado que cada función medible puede considerarse como el límite para el aumento de funciones simples, la integral de las funciones medibles se puede definir de la siguiente manera.

Definición 2.3 (Integral de Lebesgue para medidas de probabilidad)**.** Sea h uan función medible de Borel no-negativa, es decir, $h : (S, \Sigma) \to (\mathbb{R}^+, \mathscr{B}(\mathbb{R}^+))$, definimos la integral de Lebesgue de h con respecto a μ como sigue

$$\int_S h \ \delta\mu = sup \left\{ \int_S s \ \delta\mu : s \ simple, 0 \leq s \leq h \right\}$$

Proposición 2.1. Sea (S, Σ, μ) un espacio medible y $h : (S, \Sigma) \to (\mathbb{R}^+, \mathscr{B}(\mathbb{R}^+))$ una función medible. Entonces $\nu = \int_S h \ \delta\mu$ es una medida.

2.2. Non-deterministic Labeled Markov Process

Labeled Markov Processes [10, 47, 48] son una clase probabilística de sistemas de transición etiquetados. Se distinguen de otros enfoques al estar basados en los sólidos fundamentos de la teoría de la medida y la teoría del Proceso de Markov. Las LMP están motivadas por las necesidades de modelar correctamente los sistemas físicos (también conocidos como sistemas híbridos) que evolucionan en un espacio de estado continuo, involucrando parámetros continuos como la distancia, el tiempo, la temperatura, la presión. Tales sistemas usualmente combinan este espacio continuo con conjuntos de acciones discretas y comúnmente pero no siempre finitas. Los modelos resultantes son probados contra acciones tomadas por el ambiente. En este sentido, decimos que son sistemas reactivos y están destinados a funcionar simultáneamente. Al no modelar el entorno, el trabajo de LMP sobre el no

determinismo externo, es decir, el comportamiento del entorno no es conocido (es decir, no es determinista) del modelo. La bisimulación en LMP se extiende a partir de Larsen y Skou [72], definida para el caso más simple de los sistemas de espacio discreto. Ideas de Joyal, Nielsen y Winskel [68], permiten adaptarlo al espacio continuo.

Definición 2.4 (Labeled Markov Processes)**.** Un Labeled Markov Process es una tupla $(S, \Sigma, \{\mathcal{T}_a \mid a \in L\})$ donde Σ es una σ-algebra en el conjunto de estados S ,y para cada etiqueta $a \in L$, la función probabilística de transición $\mathcal{T}_a : S \to \Delta(S) \cup \{\bar{\varnothing}\}$ es una medida de probabilidad. Aquí, dejamos que $0 : \Sigma \to [0, 1]$ sea la medida nula tal que $\bar{\varnothing}(Q) = 0$ para todo $Q \in \Sigma$.

Para cada estado $s \in S$, $\mathcal{T}_a(s)(Q) \in [0, 1]$ representa la probabilidad condicional de alcanzar cualquier estado en $Q \in \Sigma$ dado que estamos en el estado S y hacemos una etiqueta de transición a. La notación de la función nula y esta definición de LMP se toman de [29] que difiere ligeramente de la definición original de [47]. Aquí, para modelar casos donde ciertas transiciones se rechazan para ser tomadas de ciertos estados, usamos la función nula que da una probabilidad cero a cada equivalencia establecida en Σ, es decir, $\bar{\varnothing}(Q) = 0$ para todos $Q \in \Sigma$. En [47] la función de transición puede ser una función de sub-probabilidad, en su lugar.

Non-deterministic Labeled Markov Process surge como una extensión del Proceso de Markov etiquetado que introduce el no determinismo interno [45]. También pueden verse como una extensión continua de Probabilistic Automata [93] para darles una base de robusta en teoría de la medida sobre el espacio de estados continuo. Esta base sólida en la teoría de la medida es una distinción clave de otros enfoques del tema. Encontramos dos características distinguibles en NLMP: la función de transición $\mathcal{T}_a$ mapea los estados en conjuntos medibles de medidas de probabilidad en lugar de conjuntos arbitrarios. Esto permite resolver el no determinismo mediante la introducción de planificadores más adelante. Permitir conjuntos arbitrarios podría hacer que el sistema sufra de no medición en presencia de ciertos conjuntos continuos no medibles. Una clase de NLMPs [29] estructurados también otorga a las etiquetas un σ-algebra. Una segunda característica permite, como en LMP, tener operadores modales bien definidos de una lógica probabilística de Hennessy-Milner. Esto se logra al restringir $\mathcal{T}_a$ a funciones medibles por cada $a \in L$.

Definición 2.5. Un *Non-deterministic Labeled Markov Process* (NLMP) es una estructura $(\mathbf{S}, \Sigma, \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ donde Σ es una σ-algebra en el conjunto de estados $\mathbf{S}$, y para cada etiqueta $a \in \mathcal{L}$ tenemos que $\mathcal{T}_a : \mathbf{S} \to \Delta(\Sigma)$ es medible desde Σ a la hit σ-algebra $H(\Delta(\Sigma))$.

El requisito de medibilidad $\mathcal{T}_a$ requiere una definición de una σ-algebra sobre su codominio, la σ-algebra de Giry $\Delta(\Sigma)$. Más aún, para poder mapear sobre conjuntos medibles en S, $\Delta(\Sigma)$, es dotado de una σ-algebra denominada hit σ-algebra $H(\Delta(\Sigma))$ [46].

Definición 2.6 (Hit σ-algebra)**.** Sea (S, Σ) un espacio medible, entonces $H(\Delta(\Sigma))$ es la mínima σ-algebra que contiene todos los conjuntos

$$H_\xi = \{\zeta \in \Delta(\Sigma) | \zeta \cap \xi \neq \varnothing\}$$

para los conjuntos medibles $\xi \in \Delta(\Sigma)$.

Para cada etiqueta $a \in L$ la función de transición no-determinística T_a debe ser medible desde la σ-algebra de estados a la hit σ-algebra de medidas, es decir, $T_a : (S, \Sigma) \to (\Delta(\Sigma), H(\Delta(\Sigma)))$. Para probar esto, es suficiente revisar los generadores de la hit σ-algebra $H(\Delta(\Sigma))$, esto es, es suficiente mostrar que para cada $\xi \in \Delta(\Sigma), T_a^{-1}(H_\xi) \in \Sigma$. Nótece que $T_a^{-1}(H_\xi) = \{s \in S | T(s) \cap \xi \neq \varnothing\}$ es el conjunto de todos los estados s tal que, a través de la etiqueta a, la función de transición "pega" en el conjunto de medidas en ξ.

En este trabajo usamos NLMP para definir la semántica que sostiene Input/Output Stochastic Automata.

La bisimulación se ha convertido en la principal forma de analizar la equivalencia de comportamiento de los sistemas de transición en concurrencia. Se basa en el principio de que dos agentes solo deben distinguirse entre sí, si se puede observar que se comportan de manera diferente mediante un tercer agente externo que interactúa [81].

En [45] se presentan tres variedades de bisimulación para NLMP. La llamada *traditional bisimulation* se basa en la definición original de Milner y se construye elevando la bisimulación de estado al espacio de probabilidades, y verificando que el conjunto de medidas de probabilidad salientes de estados equivalentes coincida con este levantamiento. Nos parecemos a esta definición al definir la bisimulación en IOSA. De hecho, esta definición es una adaptación de *probabilistic bisimulation*, introducida por Larsen y Skou [72] en una configuración discreta y adaptada a una configuración continua como NLMP en [48, 45]. La idea detrás de la equivalencia de bisimulación es que a partir de dos estados equivalentes, una transición de a debería conducir con igual probabilidad a cualquier agregado medible de clases de equivalencia (hablando correctamente, a cualquier conjunto medible que resulte de una unión arbitraria de clases de equivalencia).

Dada una relación $R \subseteq \mathbf{S} \times \mathbf{S}$, un conjunto $Q \subseteq \mathbf{S}$ es *R-closed* si $R(Q) \subseteq Q$. Si R es simétrica, Q es R-closed sii para toda $s, t \in S$ tal que $s \ R \ t$, $s \in Q \Leftrightarrow t \in Q$. Usando esta definición, una relación simétrica R puede ser

elevada a una relación de equivalencia en $\Delta(\mathbf{S})$ como sigue: $\mu\ R\ \mu'$ sii para todo R-closed $Q \in \Sigma$, $\mu(Q) = \mu'(Q)$.

Definición 2.7 (Traditional Bisimulation)**.** Una relación $R \subseteq \mathbf{S}\times\mathbf{S}$ es una *bisimulation* en el NLMP $\mathcal{P} = (\mathbf{S}, \Sigma, \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ si es simétrica y para toda $a \in \mathcal{L}$, $s\ R\ t$ inmplica que para toda $\mu \in \mathcal{T}_a(s)$, hay una $\mu' \in \mathcal{T}_a(t)$ tal que $\mu\ R\ \mu'$. Decimos que $s, t \in S$ son *bisimilares*, denotado por $s \sim t$, si existe una bisimulación R tal que $s\ R\ t$.

Sabemos que $\sim$ es una relación de equivalencia [45].

Una segunda noción de bisimulación se define tal como para LMP in [38], y se la conoce como *event bisimulation*. La tercera bisimulación presentada es original al trabajo y se basa en la también original hit σ-algebras (ver [45]). Nos referimos a ella como *state bisimulation* y establece que si $s\ R\ t$ entonces ambas $\mathcal{T}_a(s)$ y $\mathcal{T}_a(t)$ golpean el mismo conjunto medible de medidas.

Definición 2.8 (State bisimulation)**.** Una relación $R \subseteq S \times S$ es una State Bisimulation si es simétrica y para toda $a \in L$, tenemos que sRt implica que

$$\forall \xi \in \Delta(\Sigma(R)) : \mathcal{T}_a(s) \cap \xi \neq \varnothing \Leftrightarrow \mathcal{T}_a(t) \cap \xi \neq \varnothing,$$

donde $\Sigma(R)$ es la sub-σ-algebra de Σ que contiene todos los conjuntos R-closed Σ-medibles, esto es, todo $Q \in \Sigma$ tal que $R(Q) \subseteq Q$.

Se ha demostrado que si R es simétrico, R es una State Bisimulation si y solo si $\Sigma(R)$ es una Event Bisimulation. Además, se ha demostrado que si el NLMP denumerable en su imágen, entonces R es una Traditional Bisimulation si y solo si es una State Bisimulation. Este es precisamente el caso de IOSAs, ya que su semántica está dada por una imagen NLMP denumerable, por lo tanto, los tres conceptos de bisimulación coinciden.

Capítulo 3

Input/Output Stochastic Automata

Se han propuesto muchos formalismos para modelar sistemas con el fin de analizar su fiabilidad y rendimiento. El campo de los procesos estocásticos es uno de los más destacados en este tema. Modelar no es una tarea fácil, ya que no solo se pretende una buena representación del sistema real, sino también una representación eficiente para evitar el fenómeno de la explosión de estados que crece exponencialmente con respecto al tamaño de los sistemas. Este fenómeno es especialmente problemático en la verificación de modelos, donde la exploración exhaustiva del espacio de estados es la base de la técnica. En el caso de técnicas de simulación, aunque no se necesita inspeccionar el espacio de estados completo, la explosión de estados sigue siendo un problema en este tipo de análisis, ya que representa un inconveniente en la eficiencia de los algoritmos y aumenta el tiempo para obtener resultados estadísticos suficientemente precisos.

Además de considerar los asuntos de eficiencia, también se debe considerar los asuntos de calidad al modelar un sistema para el análisis de rendimiento y confiabilidad. Siempre desearemos un modelo tan fiel al sistema real como sea posible. Dos aspectos principales del lenguaje de modelado son determinantes en este sentido: debe ser lo suficientemente complejo como para capturar todas aquellas características interesantes del sistema real relevantes para el análisis previsto, y además debe permitir ofrecer modelos mantenibles con un significado claramente comprensible y sólido, para evitar en lo posible la introducción de errores durante el modelado.

Siendo así, no hay duda de que el modelado puede beneficiarse enormemente de los enfoques composicionales. Estos enfoques facilitan el diseño sistemático y el intercambio de componentes, permiten el análisis composicional y ayudan a la representación compacta de espacios de estado, junto con

otras formas de atacar el problema de explosión de estado, como las técnicas de agregación y minimización [5, 17]. El modelado composicional permite al diseñador enfocarse en el modelado del comportamiento operacional de los componentes, bastante discernible, y la evidente sincronización entre ellos (en comparación con la dificultad de descubrir el comportamiento completo en un modelo monolítico). El salto de un tipo monolítico de modelado a uno composicional no es directo, y requiere construir mecanismos formales de interacción entre componentes y analizar la equivalencia en el comportamiento, por ejemplo mediante la bisimulación.

Existe una clara necesidad de considerar distribuciones continuas generales al modelar varios aspectos de tiempo de un sistema concurrente, como podrían ser las duraciones de actividades o la ocurrencia de eventos en el mundo real. De hecho, es casi obligatorio si el sistema está destinado a ser analizado en función del rendimiento. Sin embargo, esto se debe a la certeza de que gran parte de la corrección funcional de los sistemas depende en gran medida de estas cantidades en tiempo real, y que, además, la expresión correcta de estos aspectos del tiempo permite analizar y estimar los problemas de rendimiento. Aunque tradicionalmente la distribución exponencial negativa sin memoria ha ofrecido un marco bastante útil para modelar y analizar sistemas en dominios continuos, y ha proporcionado modelos analíticamente analizables (es decir, CTMCs), no es capaz de modelar realmente muchos fenómenos. Fenómenos como los tiempos de espera en los protocolos de comunicación, los plazos estrictos en los sistemas en tiempo real, los tiempos de respuesta humanos, la variabilidad de la demora de los cuadros de sonido y video (llamada fluctuación de fase) en los modernos sistemas de comunicación multimedia, la información censal en los sistemas integrados, o fenómenos naturales, se describen típicamente mediante distribuciones sin memoria, como distribuciones uniformes, log-normales o Weibull.

Algunos trabajos se han dirigido en esta dirección, que introducen un tipo de modelado composicional en sistemas estocásticos de distribución general [22, 14], pero introducen no-determinismo, lo que no es deseable si queremos simular un modelo. Ninguna simulación real puede realizarse en un modelo no determinista sin más. Ciertos mecanismos artificiales como la introducción de programadores o la intervención de una decisión humana deben implementarse para resolver el no-determinismo. Además, se debe tener cuidado al hacerlo de una manera imparcial, que no es una tarea simple. Esto tiene dos inconvenientes principales: la técnica de simulación deja de ser una técnica push-button automática, y decisiones artificiales, que no se consideran parte del sistema real a modelar, se introducen en el análisis. En general, no es posible analizar los procesos estocásticos de distribución general, y mucho menos si no son también deterministas. Sin embargo, los

procesos estocásticos deterministas se pueden simular mediante la simulación de eventos discretos. Aunque existen enfoques para simular el MDP ya sea reconociendo el no determinismo no esencial [13, 36] o mediante el muestreo a partir de schedulers [39], no está claro cómo estas técnicas se ajustan a las configuraciones continuas.

En este capítulo presentamos una extensión de Autómatas Estocásticos denominada Autómatas Estocásticos de Entrada/Salida (Input/Output Stochastic Automata). Los Autómatas Estocásticos [37, 42, 40] fueron presentados como un modelo para representar simbólicamente sistemas de transición probabilísticos que, debido a los espacios de probabilidad continua, son infinitos por naturaleza. Este es, por ejemplo, el caso cuando queremos modelar sistemas con comportamiento de tiempo estocástico, es decir, modelos en los que el tiempo de ocurrencia de eventos puede responder a cualquier variable aleatoria continua. Los autómatas estocásticos (SA) proporcionan una manera de representar su comportamiento de manera finita. Los SA están inspiradas en los Generalized semi-Markov Processes (GSMP) y los Timed Automata, y están equipadas con composición paralela y una versión probabilística de bisimulación, heredada de su semántica construida sobre sistemas de transición probabilísticos.

En un primer intento de producir un modelo que logre ser capaz de representar eventos de tiempo distribuidos en general, y ser adecuado para la simulación, es decir, ser completamente determinista, introdujimos en este capítulo el modelo de Input/Output Stochastic Automata. Como sugiere su nombre, traemos aquí una variante de entrada/salida de autómatas estocásticos que, una vez que se cierra el modelo, es decir, se resuelven todas las sincronizaciones, el autómata resultante no contiene opciones no deterministas. Esto lo hace susceptible de simulación en el caso general y de un análisis mucho más eficiente si se restringe a los modelos de Markov. Comenzamos el capítulo presentando una introducción teórica a IOSA, para la cual proporcionamos una semántica concreta en términos de Proceso de Markov etiquetado no deterministas (NLMP). Luego, probamos que la bisimulación es una congruencia para la composición paralela tanto en NLMP como en IOSA, mostramos que la composición paralela conmuta en el nivel simbólico y concreto, y proporcionamos una prueba de que los IOSA cerrados son deterministas.

3.1. Relojes

Los autómatas estocásticos [40, 42] usan variables de reloj para controlar y observar el paso del tiempo. Dado que en el contexto de IOSA el momento

en que ocurren los eventos es aleatorio, los relojes son de hecho variables aleatorias. Cada vez que se configura un reloj, toma un valor aleatorio cuya probabilidad depende de la función de distribución asociada al reloj. A medida que el tiempo evoluciona, los relojes cuentan de manera sincronizada, es decir, todos lo hacen a la misma velocidad. Cuando un reloj alcanza el valor cero, "el reloj expira" y esto puede habilitar algunos eventos. De hecho, en nuestro contexto, la expiración de un reloj permitirá tomar una transición. En este sentido, la evolución del sistema se determinará mediante el ajuste y la caducidad continuos de los relojes. En cada paso verificaremos el reloj de valor más bajo y las transiciones que puede habilitar. Un factor clave en nuestro trabajo será garantizar que la expiración de un reloj no permita más de una transición, lo que de lo contrario induciría a una situación no determinista. Si x es una variable de reloj aleatoria en $\mathcal{C}$, entonces μ_x denotará su distribución de probabilidad. La notación

$$s \xrightarrow{C,a,C'} s'$$

representará una transición de un estado s a un estado s' produciendo la acción a que se realizará tan pronto como todos Los relojes en el conjunto C han caducado, e instantáneamente establecerán todos los relojes en C', cada uno con un valor muestreado de sus correspondientes distribuciones de probabilidad. Dado que los relojes son variables aleatorias, probablemente se les asignará un valor diferente cada uno, lo que representará un factor clave al analizar la determinación en los modelos IOSA. Por lo general, desearemos imponer un pedido en los relojes del modelo, por simplicidad. Para un conjunto C de relojes, indicaremos $\vec{C}$ a este pedido y, dado el conjunto de todas las valoraciones $V : C \to \mathbb{R}^n$, $\vec{v}$, denotarán una valoración para cada reloj en $\vec{C}$.

3.2. Modelo abierto vs modelo cerrado

A partir de la noción de autómatas estocásticos, restringimos este marco para obtener IOSA. Basándonos en los autómatas probabilísticos de entrada/salida [103], dividimos las acciones en entradas y salidas y dejamos que se comporten de manera reactiva y generativa, respectivamente. Es decir, las acciones de entrada permanecen pasivas y su aparición depende solo de su interacción con las acciones de salida. En contraste, la ocurrencia de acciones de salida depende de la expiración de los relojes y se toman tan pronto como se habilitan (vea [96] para los conceptos de transiciones generativas y reactivas). La sincronización de una acción de entrada y salida da como resultado

una acción de salida que nuevamente está disponible para la sincronización con otras acciones de entrada. Por lo tanto, una salida se transmite a todos los componentes capaces de escucharla a través de acciones de entrada. Este no es el caso en SA, donde la sincronización entre componentes se logra mediante un acuerdo y no se hace distinción entre los tipos de acciones. El acuerdo permite bloquear componentes al no sincronizarse con ellos. Por el contrario, nuestros modelos tienen entrada habilitada y, por lo tanto, no modelan el bloqueo de transiciones mediante la sincronización. En nuestra opinión, esto logra un mayor desacoplamiento de los componentes y un tratamiento mucho más claro del determinismo y la composicionalidad.

También podríamos pensar que las entradas son acciones controladas externamente y las salidas son acciones controladas localmente. Precisamente debido a esto, el tiempo de aparición de las acciones de salida está controlado por una variable aleatoria, mientras que las entradas son pasivas y, por lo tanto, su tiempo de aparición solo puede depender de su interacción con las salidas. En este sentido, llamaremos modelos abiertos a aquellos modelos reactivos donde las acciones de entrada aún están presentes, y de alguna manera falta información para determinar el comportamiento completo de ese modelo. Por otro lado, llamaremos modelos cerrados a aquellos modelos generativos donde no hay acción de entrada, y por lo tanto su comportamiento está completamente determinado por el propio modelo. Un conjunto de restricciones, que explicaremos más adelante, garantiza que, casi con seguridad, no se habiliten dos acciones de salida al mismo tiempo. El objetivo es asegurar el determinismo en los modelos cerrados para habilitar la simulación de eventos discretos en los modelos IOSA. Un modelo abierto se convertirá en un modelo cerrado una vez que todas sus acciones de entrada se hayan sincronizado a través de la composición paralela con otros modelos, y se hayan convertido en acciones de salida (generativas).

Al analizar si un modelo es determinista, nos centraremos en los modelos cerrados, ya que los modelos abiertos son intrínsecamente no deterministas, dado que su comportamiento depende de las decisiones tomadas por el entorno aún no sincronizado y potencialmente no determinista. Sin embargo, dado que nuestro análisis del determinismo se realiza sobre los componentes del modelo, encontraremos muchos modelos abiertos en el camino.

Ejemplo 3.2.1 (Modelo abierto vs cerrado de un negocio de venta de huevos)**.** En la Figura 3.1 encontramos un modelo de negocio de venta de huevos. En el modelo abierto (Figura 3.1b), encontramos que conocemos la tasa de producción de los huevos junto con la tasa de empaque de los huevos. Las transiciones de entrada aún están presentes en el modelo ya que no tenemos información sobre las tarifas de envío. Esto no nos permitirá hacer un

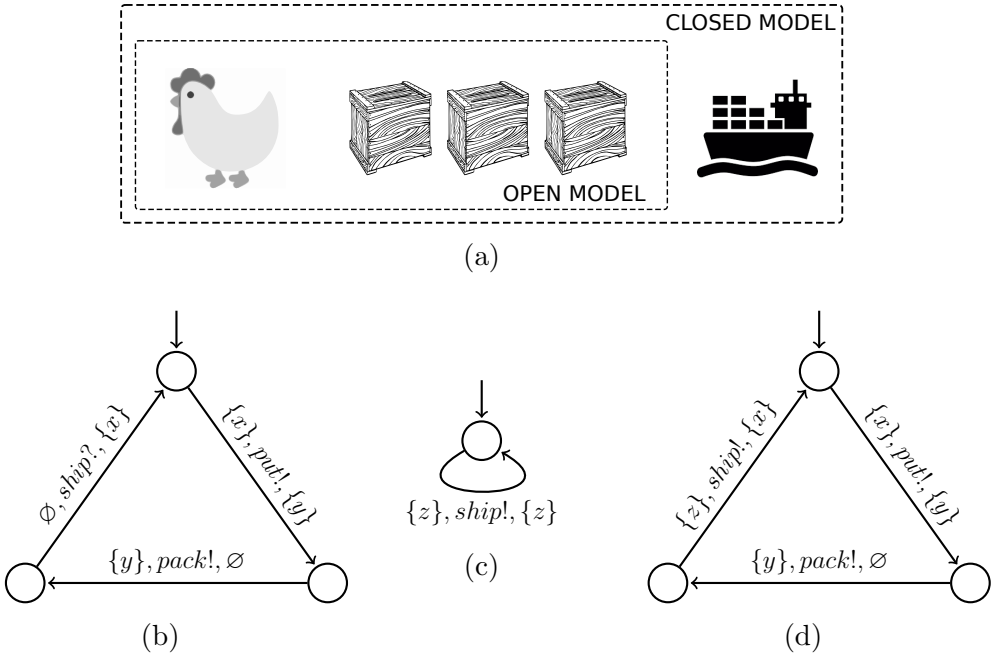


Figura 3.1: Modelo abierto vs cerrado. (a) Representacion de un negocio de ventas de huevos. (b) IOSA del modelo abierto de (a). (c) IOSA del modelo de transporte. (d) IOSA del modelo cerrado despues de componer (b) y (c).

análisis completo del sistema real, ya que dependerá de las tasas de tiempo en que se realiza el envío. Estos tiempos no se conocen a priori y, en lo que nos concierne, introducen elecciones no deterministas en el momento en que ocurren. El modelo cerrado (Figura 3.1d) completa el modelo al sincronizar el modelo abierto con el modelo de envío faltante (Figura 3.1c) . De esta manera, las transiciones de entrada se convierten en transiciones de salida, las tarifas se proporcionan según el modelo de envío. El modelo abierto nos da una pista sobre uno de los beneficios de la composicionalidad, que es la reutilización. De hecho, muchos modelos se pueden obtener reutilizando el modelo abierto 3.1b y componiéndolo con diferentes modelos de envío como 3.1c. Se puede llevar a cabo un análisis comparativo entre estos diferentes ajustes, sin la necesidad de remodelar todo el sistema cada vez, lo que sería el caso en un estilo monolítico de modelado.

3.3. Input/Output Stochastic Automata

Presentamos una primera versión de Input/Output Stochastic Automata. La definición formal de IOSA consiste en una estructura y un conjunto de restricciones que eventualmente garantizarán que los modelos cerrados sean deterministas.

Definición 3.1. Un input/output stochastic automaton (IOSA) es una estructura $(\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, \mathcal{C}_0, s_0)$, donde $\mathcal{S}$ es un conjunto (numerable) de estados, $\mathcal{A}$ es un conjunto (numerable) de etiquetas particionado en partes disjuntas de etiquetas de entrada (input) $\mathcal{A}^{\mathsf{i}}$, y de salida (output) $\mathcal{A}^{\circ}$, $\mathcal{C}$ es un conjunto (finito) de relojes tal que cada $x \in \mathcal{C}$ tiene asociado una medida de probabilidad continua μ_x en $\mathbb{R}$ (por lo tanto $\mu_x(d) = 0$ para cualquier $d \in \mathbb{R}$) que también satisface que $\mu_x(\mathbb{R}_{>0}) = 1$, $\rightarrow \subseteq \mathcal{S} \times \mathcal{C} \times \mathcal{A} \times \mathcal{C} \times S$ es una función de transición, $\mathcal{C}_0$ es el conjunto de relojes que son inicializados en el estado inicial, y $s_0 \in \mathcal{S}$ es el estado inicial. Además, un IOSA debe satisfacer las siguientes condiciones:

(a) Si $s \xrightarrow{C,a,C'} s'$ y $a \in \mathcal{A}^{\mathsf{i}}$, entonces $C = \varnothing$.

(b) Si $s \xrightarrow{C,a,C'} s'$ y $a \in \mathcal{A}^{\circ}$, entonces C es un singletón.

(c) Si $s \xrightarrow{\{x\},a_1,C_1} s_1$ y $s \xrightarrow{\{x\},a_2,C_2} s_2$ entonces $a_1 = a_2$, $C_1 = C_2$ y $s_1 = s_2$.

(d) Si $s \xrightarrow{\{x\},a,C} s'$ entonces, para toda transición $t \xrightarrow{C_1,b,C_2} s$, ya sea $x \in C_2$, o $x \notin C_1$ y existe una transición $t \xrightarrow{\{x\},c,C_3} t'$.

(e) Si $s_0 \xrightarrow{\{x\},a,C} s$ entonces $x \in C_0$.

(f) Para toda $a \in \mathcal{A}^{\mathsf{i}}$ y estado s, existe una transición $s \xrightarrow{\varnothing,a,C} s'$.

(g) Para toda $a \in \mathcal{A}^{\mathsf{i}}$, si $s \xrightarrow{\varnothing,a,C_1} s_1$ y $s \xrightarrow{\varnothing,a,C_2} s_2$, entonces $C_1 = C_2$ and $s_1 = s_2$.

La ocurrencia de una acción es controlada por la expiración de los relojes. Por lo tanto, siempre que $s \xrightarrow{\{x\},a,C} s'$ y el sistema se encuentre en el estado s, la acción de salida a ocurrirá una vez que el valor del reloj x alcance 0. En este punto, el sistema pasa al estado s' configurando los valores de cada reloj $y \in C$ a un valor muestreado según la distribución μ_y. Para las transiciones de entrada $s \xrightarrow{\varnothing,a,C} s'$, el comportamiento es similar, solo que su aparición

puede ocurrir en cualquier momento, lo que se volverá definitivo una vez que la acción interactúe con una salida.

Dado que pretendemos que osa sea determinista, se han hecho algunas restricciones al modelo, de modo que no se habilitan dos transiciones al mismo tiempo. La restricción (a) establece que cada entrada es reactiva y, por lo tanto, su presencia está controlada por el entorno. Por lo tanto, ningún reloj controla su aparición. La restricción (b) establece que cada salida es generativa (o está localmente controlada), por lo que tiene asociado un reloj que determina su tiempo de ocurrencia. También limitamos el conjunto a exactamente un reloj, para tener una definición limpia.

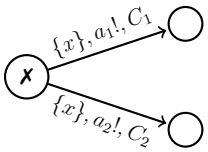


Figura 3.2: Restriccion (c)

La restricción (c) prohíbe que un solo reloj habilite dos transiciones diferentes; de lo contrario, dos acciones de salida se habilitarían simultáneamente y la elección de cuál tomar sería no determinista (ver Figura 3.2 donde ✗ denota un estado prohibido). Además, tenga en cuenta que si se usan relojes

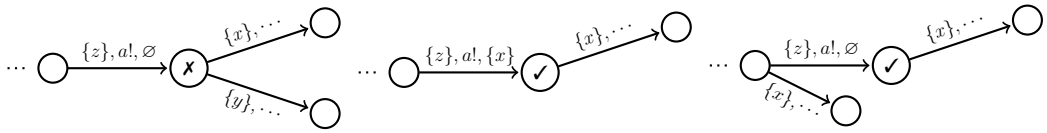


Figura 3.3: Restriccion (d)

cuando ya han expirado, se habilitará inmediatamente la transición de salida respectiva. Esto puede llevar a salidas habilitadas simultáneamente si el sistema llega a un estado con dos relojes caducados que permiten dos transiciones diferentes. Las restricciones (d) y (e) aseguran que nunca se usará un reloj cuando ya haya expirado. En particular, (d) declara que un reloj habilitador x en el estado s debe configurarse a la llegada ($x \in C_2$) o no ha sido usado inmediatamente antes ($x \notin C_1$) pero también debe estar habilitado en el estado inmediatamente anterior (vea las Figuras 3.3 y 3.4 donde donde ✗ denota un estado prohibido y ✓ estados permitidos). Dado que los relojes se configuran mediante el muestreo de una variable aleatoria continua, la probabilidad de que los valores de dos relojes diferentes sean iguales es 0. Este último hecho, junto con las restricciones (c), (d) y (e), garantiza que

casi nunca se activen dos transiciones de salida diferentes al mismo tiempo. Las restricciones (f) y (g) son restricciones usuales en I/O-like automata: (f)

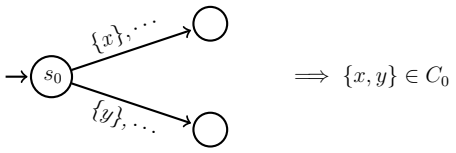


Figura 3.4: Restriction (e)

asegura que las salidas son No bloqueado en una composición. Esto tiene un doble efecto al garantizar que se conserva el comportamiento original de los autómatas y, por otro lado, al evitar situaciones no deterministas como la que se muestra en la Figura 3.5. Allí, el estado gris en el componente más a la izquierda carece de habilitación de entrada. Esto introduce el no determinismo después de la sincronización durante la composición paralela (ver Definición 3.4 para las reglas de composición), y finalmente viola la restricción (d) en el estado marcado con un ✗. La restricción (g), también ayuda a asegurar que el

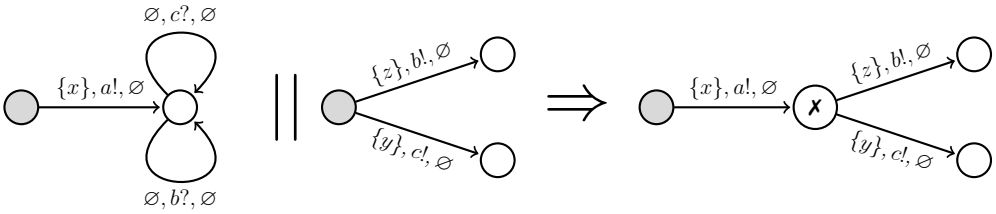


Figura 3.5: Restriction (f)

determinismo se conserve después de la composición, ya que de lo contrario podrían surgir situaciones como la que se muestra en la Figura 3.6. Esto se debe a la composición, donde la sincronización entre las acciones de entrada y salida genera acciones de salida, como veremos en Definición 3.4.

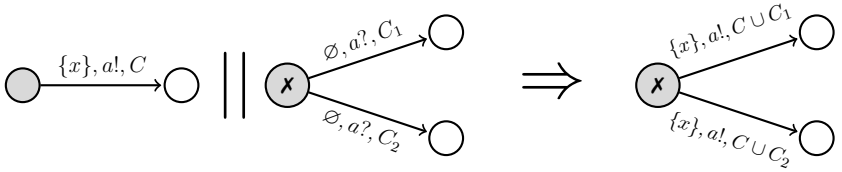


Figura 3.6: Restricción (g)

3.4. Semántica

La semántica de IOSA se define en términos de NLMP(consulte la sección 2.2), una generalización de los sistemas de transición probabilísticos con dominio continuo. La semántica formal de una IOSA se define por un NLMP con dos clases de transiciones: una que codifica los pasos discretos y contiene toda la información probabilística introducida por el muestreo de relojes, y otra que describe los pasos de tiempo, que solo registra El paso del tiempo disminuye sincrónicamente el valor de todos los relojes. Para simplificar la definición, asumimos que el conjunto de relojes tiene un orden particular y sus valores actuales siguen el mismo orden en un vector. Esto es, $\vec{C}$ denotará este pedido en el conjunto de relojes C de cardinalidad n, y una valoración $\vec{v}$ del conjunto de valoraciones $V : \mathcal{C} \to \mathbb{R}^n$, indicará los valores de cada reloj en cualquier momento.

Definición 3.2. Given an IOSA $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \to, \mathcal{C}_0, s_0)$ with $\mathcal{C} = \{x_1, \ldots, x_N\}$, its semantics is defined by the NLMP $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ where

- $\mathbf{S} = (\mathcal{S} \cup \{\mathsf{init}\}) \times \mathbb{R}^N$, $\mathcal{L} = \mathcal{A} \cup \mathbb{R}_{>0} \cup \{\mathsf{init}\}$, with $\mathsf{init} \notin \mathcal{S} \cup \mathcal{A} \cup \mathbb{R}_{>0}$
- $\mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v}) = \{\delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}\}$
- $\mathcal{T}_a(s, \vec{v}) = \{\mu_{\vec{v},C',s'} \mid s \xrightarrow{C,a,C'} s', \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\}$,

  for all $a \in \mathcal{A}$, where $\mu_{\vec{v},C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$ with

$$\overline{\mu}_{x_i} = \begin{cases} \mu_{x_i} & \text{if} x_i \in C' \\ \delta_{\vec{v}(i)} & \text{otherwise} \end{cases}$$

- $\mathcal{T}_d(s, \vec{v}) =$
  $\{\delta^{-d}_{(s,\vec{v})} \mid 0 < d \leq \text{mín}\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\circ}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}\}$

  for all $d \in \mathbb{R}_{\geq 0}$, where $\delta^{-d}_{(s,\vec{v})} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$.

El espacio de estado es el espacio de producto de los estados de la IOSA con todas las posibles valuaciones de reloj. Se agrega un estado inicial distinguido init para codificar la inicialización aleatoria de todos los relojes (sería suficiente inicializar los relojes en $\mathcal{C}_0$ pero decidimos esta simplificación). Dicha codificación se realiza mediante la transición $\mathcal{T}_{\mathsf{init}}$. El espacio de estado está estructurado en el habitual Borel σ -algebra. El paso discreto está codificado por $\mathcal{T}_a$, con $a \in \mathcal{A}$. Tenga en cuenta que, en el estado $(s, \vec{v})$, la transición $s \xrightarrow{C,a,C'} s'$ solo tendrá lugar si $\bigwedge_{x_i \in C} \vec{v}(i) \leq 0$, es decir, si los

valores actuales de todos los relojes en C no son positivos. Para el caso particular de las acciones de entrada, esto siempre será cierto. El siguiente estado real se determinaría al azar de la siguiente manera: el estado simbólico será s' (esto corresponde a $\delta_{s'}$ in $\mu_{\vec{v},C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$), cualquier reloj que no esté en C' conserva el valor actual (por lo tanto, $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ si $x_i \notin C'$), y cualquier reloj en C' se configura al azar de acuerdo con su respectiva distribución asociada (por lo tanto, $\overline{\mu}_{x_i} = \mu_{x_i}$ si $x_i \in C'$). El paso de tiempo está codificado por $\mathcal{T}_d(s, \vec{v})$ con $d \in \mathbb{R}_{\geq 0}$. Solo puede tener lugar en d unidades de tiempo si no hay una transición de salida habilitada en el estado actual dentro de las siguientes unidades de tiempo de d (esto se verifica por la condición $0 < d \leq \text{mín}\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\circ}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$). Es decir, el salto de tiempo máximo posible es igual al tiempo mínimo requerido para transcurrir hasta que se habilite una transición de salida en el estado actual). En este caso, el sistema permanece en el mismo estado simbólico (esto corresponde a δ_s in $\delta_{(s,\vec{v})}^{-d} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$), y todos los valores de reloj se reducen en d unidades de tiempo (representadas por $\delta_{\vec{v}(i)-d}$ en la misma fórmula. La figura 3.7 ilustra los dos tipos de transiciones definidas en la semántica de IOSA: la transición correspondiente a un paso discreto en el lado izquierdo y la transición correspondiente al salto de tiempo en el lado derecho.

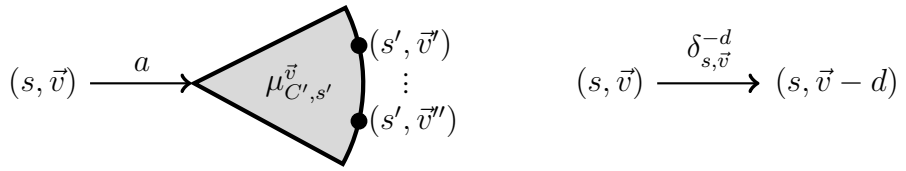


Figura 3.7: Transiciones en la semántica NLMP de IOSA.

Todavía tenemos que mostrar que $\mathcal{P}(\mathcal{I})$ es en verdad un NLMP. Para esto tenemos que probar que $\mathcal{T}_a$ mapea en conjuntos medibles en $\Delta(\mathscr{B}(\mathbf{S}))$ (Lemma 3.1), y que $\mathcal{T}_a$ es una función medible para $a \in \mathcal{L}$ (Lemma 3.2).

Lema 3.1. $\mathcal{T}_a(s, \vec{v}) \in \Delta(\mathscr{B}(\mathbf{S}))$ para todo $a \in \mathcal{L}$ y $(s, \vec{v}) \in \mathbf{S}$.

Demostración. La demostración hace uso del Lema 3.1 en [45], a partir del cual sabemos que para toda $\mu \in \Delta(\mathbf{S})$, $\{\mu\} \in \Delta(\mathscr{B}(\mathbf{S}))$ (dado que $\mathscr{B}(\mathbf{S})$ es generado por un π-sistema discreto).

Observer que para cualquier $\vec{v} \in \mathbb{R}^N$, $\mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v})$ es un conjunto singleton y por lo tanto mensurable. De manera similar, observe que por cada $d \in \mathbb{R}_{>0}$,

$s \in \mathcal{S}$, y $\vec{v} \in \mathbb{R}^N$, $\mathcal{T}_d(s, \vec{v})$ es ya sea un conjunto singleton o el conjunto vacío, y por lo tanto medible. Finalmente, dado que solo hay un número numerable de transiciones en un IOSA, por cada $a \in \mathcal{A}$, $s \in \mathcal{S}$, y $\vec{v} \in \mathbb{R}^N$, $\mathcal{T}_a(s, \vec{v})$ es una unión numerable de conjuntos singleton, y por lo tanto también medible. □

El siguiente lema usa las *hit σ-algebra* introducidas en los preliminares de esta tesis (sección 2.6).

Lema 3.2. Para todo $a \in \mathcal{L}$, $\mathcal{T}_a$ es medible desde $\mathscr{B}(\mathbf{S})$ a $H(\Delta(\mathscr{B}(\mathbf{S})))$.

Demostración. Necesitamos mostrar que para cada $a \in \mathcal{L}$ y cada $\xi \in \Delta(\mathscr{B}(\mathbf{S}))$, $\mathcal{T}_a^{-1}(H(\xi)) = \{(s, \vec{v}) \mid \mathcal{T}_a(s, \vec{v}) \cap \xi \neq \varnothing\}$ es medible.

Dividimos la prueba en tres casos dependiendo de la naturaleza de la etiqueta en la función de transición. Primero, note que $\mathcal{T}_{\mathsf{init}}^{-1}(H(\xi)) = \{\mathsf{init}\} \times \mathbb{R}^N$ si $\delta_{s_0} \times \prod_{i=1}^N \mu_{x_i} \in \xi$ y $\mathcal{T}_{\mathsf{init}}^{-1}(H(\xi)) = \varnothing$ de lo contrario, y ambos conjuntos son medibles.

Ahora analizamos el caso de $a \in \mathcal{A}$, para el cual podemos calcular

$$\begin{aligned}\mathcal{T}_a^{-1}(H(\xi)) &= \{(s, \vec{v}) \mid \{\mu_{\vec{v},C',s'} \mid s \xrightarrow{C,a,C'} s', \textstyle\bigwedge_{x_i \in C} \vec{v}(i) \leq 0\} \cap \xi \neq \varnothing\} \\ &= \textstyle\bigcup_{s \xrightarrow{C,a,C'} s'} \{(s, \vec{v}) \mid \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\} \cap \{(s, \vec{v}) \mid \mu_{\vec{v},C',s'} \in \xi\}\end{aligned}$$

Dado que la unión es numerable, basta con probar que los dos conjuntos que se intersecan son medibles. Primero, note que $\{(s, \vec{v}) \mid \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\} = \{s\} \times \prod_{i=1}^N V_i$ donde $V_i = (-\infty, 0]$ si $x_i \in C$ y $V_i = \mathbb{R}$ en otro caso. Por lo tanto, es medible.

Para el segundo caso, defina $f_{C',s'} : \mathbb{R}^N \to \Delta(\mathbf{S})$ por $f_{C',s'}(\vec{v}) = \mu_{\vec{v},C',s'}$. Luego $\{(s, \vec{v}) \mid \mu_{\vec{v},C',s'} \in \xi\} = \{(s, \vec{v}) \mid f_{C',s'}(\vec{v}) \in \xi\} = \{s\} \times f_{C',s'}^{-1}(\xi)$. Entonces, solo queda probar que $f_{C',s'}$ es una función medible. Usando [99, Lema 3.6], solo necesitamos probar que $f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \prod_{i=1}^N V_i))$ con $A \subseteq \mathbf{S}$ y $V_i \in \mathscr{B}(\mathbb{R})$, $1 \leq i \leq N$, es medible, para lo cual podemos calcular

$$\begin{aligned}f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \textstyle\prod_{i=1}^N V_i)) &= \{\vec{v} \mid \mu_{\vec{v},C',s'}(A \times \textstyle\prod_{i=1}^N V_i) \geq q\} \\ &= \{\vec{v} \mid s' \in A, (\textstyle\prod_{x_i \in C'} \mu_{x_i})(\prod_{x_i \in C'} V_i) \geq q, \forall x_i \notin C' : \vec{v}(i) \in V_i\}\end{aligned}$$

Luego, si $s' \in A$ and $(\prod_{x_i \in C'})(\prod_{x_i \in C'} V_i) \geq q$, $f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \prod_{i=1}^N V_i)) = \prod_{i=1}^N \overline{V_i}$ con $\overline{V_i} = \mathbb{R}$ si $x_i \in C'$, $\overline{V_i} = V_i$ si $x_i \notin C'$, o $f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \prod_{i=1}^N V_i)) = \varnothing$ en caso contrario, y en ambos casos los conjuntos son medibles.

Para el caso de $d \in \mathbb{R}$, observe que

$$\begin{aligned}\mathcal{T}_d^{-1}(H(\xi)) = \{(s, \vec{v}) \mid \delta_{(s,\vec{v})}^{-d} \in \xi\} \cap \\ \{(s, \vec{v}) \mid 0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^\circ, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}\end{aligned}$$

El segundo conjuntos es igual a $\mathbf{S} \times \prod_{i=1}^{N} V_i$ donde $V_i = [d, \infty)$ si $s \xrightarrow{\{x_i\},a,C'} s'$, y $V_i = \mathbb{R}$ en caso contrario. Luego es medible. Para el primer conjunto, defina $f_d : \mathbf{S} \to \Delta(\mathbf{S})$ por $f_d(s, \vec{v}) = \delta^{-d}_{(s,\vec{v})}$. Luego $\{(s, \vec{v}) \mid \delta^{-d}_{(s,\vec{v})} \in \xi\} = f_d^{-1}(\xi)$ y por lo tanto es suficiente mostrar que f_d es medible. Entonces, debemos probar que $f_d^{-1}(\Delta^{\geq q}(Q))$ es medible para cualquier $Q \in \mathscr{B}(\mathbf{S})$. Pero $f_d^{-1}(\Delta^{\geq q}(Q)) = \{(s, \vec{v}) \mid \delta^{-d}_{(s,\vec{v})}(Q) \geq q\}$, y por lo tanto $f_d^{-1}(\Delta^{\geq q}(Q)) = \{(s, \vec{v}) \mid (s, \vec{v} - d) \in Q\}$ if $q > 0$ or $f_d^{-1}(\Delta^{\geq q}(Q)) = \mathcal{S}$ if $q = 0$, y en ambos casos los conjuntos son medibles. □

3.5. Composición y bisimulación como congruencia

La composición paralela es una herramienta fundamental para la construcción modular de grandes modelos. La composición paralela en IOSA modela la concurrencia a través del intercalado de acciones independientes, pero sincronizando acciones con el mismo nombre tal como sucede en CSP [67] y LOTOS [15].

La bisimulación es una noción de equivalencia entre agentes. Se basa en la idea de que solo queremos distinguir entre agentes si su comportamiento observable se puede distinguir por un tercer agente [81].

En el caso de IOSA, la composición conserva la relación de bisimulación. Es decir, la bisimulación es una congruencia con respecto al operador de composición. Esto nos permite reemplazar componentes en nuestros modelos por otros componentes que se comportan equivalentemente, por lo tanto, sin alterar el comportamiento del modelo compuesto.

En esta sección definimos la composición paralela para IOSA y demostramos que IOSA es cerrado para la composición paralela. También mostramos que la bisimulación es una congruencia con respecto a la composición paralela y lo logramos definiendo composición paralela en NLMP. Como pretendemos que las salidas sean autónomas (o controladas localmente), no permitimos la sincronización entre salidas. Además, necesitamos evitar conflictos de nombres en los relojes, de modo que el el comportamiento de cada componente se conserve y, además, para garantizar que el autómata compuesto resultante es de hecho un IOSA. Por lo tanto requerimos componer solo IOSAs *compatibles*.

Definición 3.3. Dos IOSAs $\mathcal{I}_1$ e $\mathcal{I}_2$ se dicen compatibles si no comparten acciones de salida ni relokes, i.e. $\mathcal{A}_1^O \cap \mathcal{A}_2^O = \varnothing$ y $\mathcal{C}_1 \cap \mathcal{C}_2 = \varnothing$.

Definición 3.4. Dado dos IOSAs compatibles $\mathcal{I}_1$ e $\mathcal{I}_2$, la composición paralela $\mathcal{I}_1||\mathcal{I}_2$ es un nuevo IOSA $(\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{A}, \mathcal{C}, \rightarrow, \mathcal{C}_0, s_0^1||s_0^2)$ donde

(I) $\mathcal{A}^{\circ} = \mathcal{A}_1^O \cup \mathcal{A}_2^O$

(II) $\mathcal{A}^{\mathsf{i}} = (\mathcal{A}_1^I \cup \mathcal{A}_2^I) \setminus \mathcal{A}^{\circ}$

(III) $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$

(IV) $\mathcal{C}_0 = \mathcal{C}_0^1 \cup \mathcal{C}_0^2$

y $\rightarrow$ es la menor relación definida por las reglas in la Tabla 3.1 donde escribimos $s||t$ en lugar de (s, t).

Cuadro 3.1: Composición paralela en IOSA

$$\frac{s_1 \xrightarrow{C,a,C'}_1 s_1'}{s_1||s_2 \xrightarrow{C,a,C'} s_1'||s_2} \; a \in \mathcal{A}_1 \backslash \mathcal{A}_2 \tag{R1}$$

$$\frac{s_2 \xrightarrow{C,a,C'}_2 s_2'}{s_1||s_2 \xrightarrow{C,a,C'} s_1||s_2'} \; a \in \mathcal{A}_2 \backslash \mathcal{A}_1 \tag{R2}$$

$$\frac{s_1 \xrightarrow{C_1,a,C_1'}_1 s_1' \quad s_2 \xrightarrow{C_2,a,C_2'}_2 s_2'}{s_1||s_2 \xrightarrow{C_1 \cup C_2,a,C_1' \cup C_2'} s_1'||s_2'} \; a \in \mathcal{A}_1 \cap \mathcal{A}_2 \tag{R3}$$

Las reglas R1 y R2 de la Tabla 3.1 se refieren al intercalado de transiciones. Este será el caso de todas las transiciones para las que la acción no pertenece a la intersección de $\mathcal{A}_1$ y $\mathcal{A}_2$. Por otro lado, la regla R3 define la sincronización para todas las acciones en la intersección de los alfabetos. Dado que ambos autómatas son compatibles, sus acciones de salida son disjuntas, de lo que deducimos que la acción a debe ser una acción de entrada en al menos uno de entre $\mathcal{I}_1$ e $\mathcal{I}_2$. En caso de que esté en ambos, la transición resultante será una transición de entrada. De lo contrario, sería una transición de salida (observe el elemento (II) en la definición 3.4). Dadas las restricciones para IOSA (a) y (b), la transición resultante estará bien formada.

La definición anterior es solo estructural. Necesitamos mostrar que las siete restricciones que definen a los IOSAs también se preservan en los autómatas compuestos.

Teorema 3.1. Sean $\mathcal{I}_1$ e $\mathcal{I}_2$ dos IOSAs compatibles. Entonces $\mathcal{I}_1||\mathcal{I}_2$ es en efecto un IOSA.

Demostración. La demostración de las restricciones (a), (b), (f), (e), y (g) derivan de una inspección directa de las reglas, considerando que $\mathcal{I}_1$ y $\mathcal{I}_2$ también satisfacen la respectiva restricción, y haciendo un análisis por casos. Como $\mathcal{I}_1$ e $\mathcal{I}_2$ son compatibles, la restricción (c) también resulta de inspeccionar las reglas, teniendo en cuenta, además, que $\mathcal{I}_1$ e $\mathcal{I}_2$ también satisfacen la restricción (g).

Por lo tanto, solo nos enfocamos en (d). Supongamos que $s_1||s_2 \xrightarrow{\{x\},a,C} s_1'||s_2'$. Analizamos el caso en el que $a \in \mathcal{A}_1$ y $x \in \mathcal{C}_1$. El otro caso es simétrico. Es más, solo consideramos el caso en que $a \in \mathcal{A}_1 \cap \mathcal{A}_2$ dado que el caso $a \in \mathcal{A}_1 \setminus \mathcal{A}_2$ sigue similar razonamiento.

En este caso, tenemos que $s_1 \xrightarrow{\{x\},a,C_1}_1 s_1'$, $s_2 \xrightarrow{\varnothing,a,C_2}_2 s_2'$, y $C = C_1 \cup C_2$. Sea $t_1||t_2 \xrightarrow{C',b,C''} s_1||s_2$. Distinguimos tres casos:

(I) Supongamos $b \in \mathcal{A}_1 \setminus \mathcal{A}_2$. Luego $t_1 \xrightarrow{C',b,C''} s_1$ and $t_2 = s_2$. Dado que $\mathcal{I}_1$ satisface (d), entonces ya sea $x \in C''$, o $x \notin C'$ y existe $t_1 \xrightarrow{\{x\},c,C_3}_1 t_1'$. Por lo tanto $x \in C''$, or $x \notin C'$ y existen t_2' y C_3' tal que $t_1||t_2 \xrightarrow{\{x\},c,C_3'} t_1'||t_2'$ (lo cual ocurre ya sea por la regla (R1) o (R3) si $c \in \mathcal{A}_1 \cap \mathcal{A}_2$).

(II) Si $b \in \mathcal{A}_2 \setminus \mathcal{A}_1$, entonces $t_2 \xrightarrow{C',b,C''}_2 s_2$ and $t_1 = s_1$. Observe que $C', C'' \subseteq \mathcal{C}_2$ y por lo tanto $x \notin C'$ y $x \notin C''$. Más aún, dado que $\mathcal{I}_2$ es input-enabled (restricción (f)), $t_2 \xrightarrow{\varnothing,a,C_3}_2 t_2'$ para ciertos C_3 y t_2'. Luego, por la regla (R3), $s_1||t_2 \xrightarrow{\{x\},a,C_1\cup C_3} s_1'||t_2'$ lo cual demuestra este caso.

(III) Si $b \in \mathcal{A}_1 \cap \mathcal{A}_2$, entonces, por la regla (R3), $t_1 \xrightarrow{C_1',b,C_1''}_1 s_1$, $t_2 \xrightarrow{C_2',b,C_2''}_2 s_2$, $C' = C_1' \cup C_2'$ y $C'' = C_1'' \cup C_2''$. Dado que $\mathcal{I}_1$ satisface (d), luego ya sea $x \in C_1''$, or $x \notin C_1'$ y existe $t_1 \xrightarrow{\{x\},c,C_3}_1 t_1'$. Si $x \in C_1''$, entonces $x \in C'$ lo cual demuestra parcialmente este caso. En cambio si $x \notin C_1'$ y existe $t_1 \xrightarrow{\{x\},c,C_3}_1 t_1'$, entonces $x \notin C''$ (dado que $x \notin C_2''$ por compatibilidad), y existen t_2' y C_3' tal que $t_1||t_2 \xrightarrow{\{x\},c,C_3'} t_1'||t_2'$ (lo cual puede ocurrir por la regla (R1) o (R3) si $c \in \mathcal{A}_1 \cap \mathcal{A}_2$), demostrando finalmente este caso. □

Para probar que la bisimulación es una congruencia en IOSA, primero definimos composición paralela en NLMPs, probamos la congruencia en este escenario y luego mostramos que la semántica de la composición paralela de dos IOSAs es isomorfa a la composición paralela de la semántica de cada IOSA. De esto se sigue que la bisimulación es también una congruencia para

la composición paralela de IOSAs. Una consideración importante es que los NLMPs no son cerrados para composición paralela [54] en general, aunque sí lo son en nuestra configuración, es decir, cuando corresponden a la semántica de IOSAs. Por ello, requerimos que la composición paralela de NLMPs también sea NLMP como hipótesis del teorema de congruencia en NLMP (Teorema 3.2).

Definición 3.5. Sean $\mathcal{P}_i = (\mathbf{S}_i, \Sigma_i, \{\mathcal{T}_a^i \mid a \in \mathcal{L}_i\})$, $i \in \{1,2\}$, dos NLMPs. Definimos la composición paralela como $\mathcal{P}_1||\mathcal{P}_2 = (\mathbf{S}_1 \times \mathbf{S}_2, \Sigma_1 \otimes \Sigma_2, \{\mathcal{T}_a \mid a \in \mathcal{L}_1 \cup \mathcal{L}_2\})$ donde, escribiendo $s_1||s_2$ en lugar de (s_1, s_2),

(I) $\mathcal{T}_a(s_1||s_2) = \{\mu_1 \times \delta_{s_2} \mid \mu_1 \in \mathcal{T}_a^1(s_1)\}$, if $a \in \mathcal{L}_1 \setminus \mathcal{L}_2$,

(II) $\mathcal{T}_a(s_1||s_2) = \{\delta_{s_1} \times \mu_2 \mid \mu_2 \in \mathcal{T}_a^2(s_2)\}$, if $a \in \mathcal{L}_2 \setminus \mathcal{L}_1$, and

(III) $\mathcal{T}_a(s_1||s_2) = \{\mu_1 \times \mu_2 \mid \mu_1 \in \mathcal{T}_a^1(s_1), \mu_2 \in \mathcal{T}_a^2(s_2)\}$, if $a \in \mathcal{L}_1 \cap \mathcal{L}_2$.

El siguiente teorema establece que $\sim$ es una congruencia para composición paralela siempre que la composición resultante sea de hecho un NLMP. Para la definición de $\sim$ remitimos al lector a los Preliminares 2.7.

Teorema 3.2. Sean $\mathcal{P}_i = (\mathbf{S}_i, \Sigma_i, \{\mathcal{T}_a^i \mid a \in \mathcal{L}_i\})$ $i \in \{1,2\}$, dos NLMPs. Si $\mathcal{P}_1||\mathcal{P}_2$ es un NLMP, entonces para todo $s_1, s_1' \in \mathbf{S}_1$ y $s_2 \in \mathbf{S}_2$, si $s_1 \sim s_1'$, entonces $s_1||s_2 \sim s_1'||s_2$ y $s_2||s_1 \sim s_2||s_1'$.

Demostración. Solo probamos que $s_1||s_2 \sim s_1'||s_2$. El caso restante es simétrico. Sea $R \subseteq \mathbf{S}_1 \times \mathbf{S}_1$ una relación de bisimulación. Definimos $R' \subseteq (\mathbf{S}_1 \times \mathbf{S}_2) \times (\mathbf{S}_1 \times \mathbf{S}_2)$ como $R' = \{(s_1||s_2, s_1'||s_2) \mid (s_1, s_1') \in R, s_2 \in \mathbf{S}_2\}$.

Probamos que R' es una bisimulación haciendo análisis por casos sobre la definición de la relación de transición en la composición paralela.

Supongamos en general que $s_1||s_2 \; R' \; s_1'||s_2$, y consideremos el caso en que $\mathcal{T}_a(s_1||s_2)$ resulta de (i) en la Definición 3.5. Sea $\mu_1 \times \delta_{s_2} \in \mathcal{T}_a(s_1||s_2)$ con $\mu_1 \in \mathcal{T}_a^1(s_1)$. Dado que $s_1 \; R \; s_1'$, entonces existe $\mu_1' \in \mathcal{T}_a^1(s_1')$ tal que $\mu_1 \; R \; \mu_1'$. Sea $Q \in \Sigma_1 \otimes \Sigma_2$ sea R'-closed y definamos $Q|_{s_2} = \{s_1 \mid s_1||s_2 \in Q\}$. $Q|_{s_2}$ es medible en Σ_1 [3], y se puede demostrar fácilmente que es R-closed. Ahora calculamos:

$$\begin{aligned}(\mu_1 \times \delta_{s_2})(Q) &= (\mu_1 \times \delta_{s_2})(Q|_{s_2} \times \{s_2\}) = \mu_1(Q|_{s_2}) \\ &\overset{(*)}{=} \mu_1'(Q|_{s_2}) = (\mu_1' \times \delta_{s_2})(Q|_{s_2} \times \{s_2\}) = (\mu_1' \times \delta_{s_2})(Q)\end{aligned}$$

donde la igualdad $(*)$ se sigue de $\mu_1 \; R \; \mu_1'$, y por lo tanto $(\mu_1 \times \delta_{s_2}) \; R' \; (\mu_1' \times \delta_{s_2})$.

El caso (ii) en la Definición 3.5 se prueba con un análisis similar, por lo que nos enfocamos en el caso (iii). Sea $\mu_1 \times \mu_2 \in \mathcal{T}_a(s_1||s_2)$ con $\mu_1 \in \mathcal{T}_a^1(s_1)$. Dado

que $s_1\ R\ s_1'$, entonces existe $\mu_1' \in \mathcal{T}_a^1(s_1')$ tal que $\mu_1\ R\ \mu_1'$. Sea $Q \in \Sigma_1 \otimes \Sigma_2$ R'-closed. Usando el teorema de Fubini [3], calculamos:

$$\begin{aligned}
(\mu_1 \times \mu_2)(Q) &= \int_{\mathbf{S}_2}\int_{\mathbf{S}_1} 1_Q(x,y)\ d\mu_1(x)\ d\mu_2(y) \\
&= \int_{\mathbf{S}_2}\int_{\mathbf{S}_1} 1_{Q|_y}(x)\ d\mu_1(x)\ d\mu_2(y) \\
&= \int_{\mathbf{S}_2} \mu_1(Q|_y)\ d\mu_2(y) \\
&\overset{(*)}{=} \int_{\mathbf{S}_2} \mu_1'(Q|_y)\ d\mu_2(y) \\
&= (\mu_1' \times \mu_2)(Q)
\end{aligned}$$

donde 1_Q es la función característica usual, y $(*)$ se sigue de $\mu_1\ R\ \mu_1'$. Por lo tanto $(\mu_1 \times \mu_2)\ R'\ (\mu_1' \times \mu_2)$. □

A continuación, demostramos que la interpretación semántica de IOSAs y la composición paralela conmuta, es decir, que el NLMP que resulta de interpretar una composición paralela de dos IOSAs es isomorfo a la composición paralela de los dos NLMPs interpretando cada uno de los IOSAs.

Teorema 3.3. Dados dos IOSAs $\mathcal{I}_1$ y $\mathcal{I}_2$, existe un isomorfismo entre (las partes alcanzables de) $\mathcal{P}(\mathcal{I}_1||\mathcal{I}_2)$ y $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$.

Demostración. Sean N y M el número de relojes en $\mathcal{I}_1$ y $\mathcal{I}_2$, respectivamente. Sean $\mathbf{S} = ((\mathcal{S}_1 \times \mathcal{S}_2) \cup \{\mathsf{init}\}) \times \mathbb{R}^{N+M}$ y $\mathbf{S}' = ((\mathcal{S}_1 \times \mathbb{R}^N) \times (\mathcal{S}_2 \times \mathbb{R}^M)) \cup ((\{\mathsf{init}\} \times \mathbb{R}^N) \times (\{\mathsf{init}\} \times \mathbb{R}^M))$ los estados de $\mathcal{P}(\mathcal{I}_1||\mathcal{I}_2)$ y $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$, respectivamente [1]. El isomorfismo está dado por la función $f : \mathbf{S} \to \mathbf{S}'$ definida como $f(\mathsf{init}, \vec{v}_1\vec{v}_2) = (\mathsf{init}, \vec{v}_1)||(\mathsf{init}, \vec{v}_2)$, y $f((s_1||s_2), \vec{v}_1\vec{v}_2) = (s_1, \vec{v}_1)||(s_2, \vec{v}_2)$ para todo $s_1 \in \mathcal{S}_1$, $s_2 \in \mathcal{S}_2$, y vectores $\vec{v}_1$ y $\vec{v}_2$ que representan valuaciones en los conjuntos de relojes $\mathcal{C}_1$ y $\mathcal{C}_2$ respectivamente. f es claramente biyectiva, y se puede demostrar directamente que tanto f como f^{-1} son medibles (es decir, f es *bimedible*). De esto se deduce que los espacios medibles $(\mathbf{S}, \mathscr{B}(\mathbf{S}))$ y $(\mathbf{S}', \mathscr{B}(\mathbf{S}'))$ son isomorfos.

Siguiendo [50], f induce un mapa $\Delta f : \Delta(\mathbf{S}) \to \Delta(\mathbf{S}')$ definido por $\Delta f(\mu) = \mu \circ f^{-1}$. No es difícil probar que Δf es biyectiva y bimedible. Por lo tanto, $(\Delta(\mathbf{S}), \Delta(\mathscr{B}(\mathbf{S})))$ y $(\Delta(\mathbf{S}'), \Delta(\mathscr{B}(\mathbf{S}')))$ son isomorfos.

[1] Estrictamente hablando, $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$ debería también contener estados de la forma $(s, \vec{v}_1)||(\mathsf{init}, \vec{v}_2)$ y $(\mathsf{init}, \vec{v}_1)||(s, \vec{v}_2)$ con $s \neq \mathsf{init}$. Sin embargo, estos estados no son alcanzables. Por lo tanto, no los consideramos ya que de lo contrario el resultado no sería estrictamente un isomorfismo y solo agregaría problemas técnicos irrelevantes a la demostración.

Podemos elevar f por segunda vez para obtener un isomorfismo en σ-álgebras. Definamos [2] $Hf : \Delta(\mathscr{B}(\mathbf{S}')) \to \Delta(\mathscr{B}(\mathbf{S}))$ por $Hf = (\Delta f)^{-1}$. De nuevo se puede probar que Hf es biyectiva y bimesurable y por lo tanto , $(\Delta(\mathscr{B}(\mathbf{S})), H(\Delta(\mathscr{B}(\mathbf{S}))))$ y $(\Delta(\mathscr{B}(\mathbf{S}')), H(\Delta(\mathscr{B}(\mathbf{S}'))))$ son isomorfos.

Ahora, no es difícil ver que para todo $a \in \mathcal{L}$, $\mathcal{T}_a(r) = Hf(\mathcal{T}'_a(f(r)))$ para todo $r \in \mathbf{S}$ donde $\mathcal{T}_a$ and $\mathcal{T}'_a$ son las funciones de transición en $\mathcal{P}(\mathcal{I}_1||\mathcal{I}_2)$ y $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$, respectivamente. Esto prueba que los dos NLMPs son isomorfos. □

Dados dos NLMPs $\mathcal{P}_1$ y $\mathcal{P}_2$ con el mismo conjunto de etiquetas, la definición de bisimulación puede extenderse a estados en los diferentes NLMPs construyendo el NLMP inducido por el coproducto σ-álgebra. El NLMP $\mathcal{P}_1 \oplus \mathcal{P}_2$ está definido por la estructura $(\mathbf{S}_1 \uplus \mathbf{S}_2, \Sigma_1 \oplus \Sigma_2, \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ donde, para todo $s \in \mathbf{S}_1 \uplus \mathbf{S}_2$ y $a \in \mathcal{L}$, $\mathcal{T}_a(s) = \mathcal{T}^1_a(s)$ if $s \in \mathbf{S}_1$ y $\mathcal{T}_a(s) = \mathcal{T}^2_a(s)$ if $s \in \mathbf{S}_2$. Por lo tanto, si s_1 y s_2 son estados de $\mathcal{P}_1$ y $\mathcal{P}_2$ respectivamente, $s_1 \sim s_2$ siempre que sean bisimilares en $\mathcal{P}_1 \oplus \mathcal{P}_2$.

Por [50, Prop. 3.6], el siguiente corolario se sigue inmediatamente del Teorema 3.3.

Corolario 3.1. Para cualquier $\vec{v}_1$ y $\vec{v}_2$ que representen valuaciones de relojes en $\mathcal{I}_1$ y $\mathcal{I}_2$, resp., $(\mathsf{init}, \vec{v}_1\vec{v}_2) \sim (\mathsf{init}, \vec{v}_1)||(\mathsf{init}, \vec{v}_2)$ y $((s_1||s_2), \vec{v}_1\vec{v}_2) \sim (s_1, \vec{v}_1)||(s_2, \vec{v}_2)$.

We say that two IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$ are bisimilar, notation $\mathcal{I}_1 \sim \mathcal{I}_2$ whenever $(\mathsf{init}, \vec{v}_1) \sim (\mathsf{init}, \vec{v}_2)$ for any vectors $\vec{v}_1$ and $\vec{v}_2$ representing the valuations of clocks in $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively.

Then, the fact that bisimulation equivalence is a congruence on IOSAs follows from Theorem 3.2 and Corollary 3.1 and it is stated in the following theorem.

Decimos que dos IOSAs $\mathcal{I}_1$ y $\mathcal{I}_2$ son bisimilares, notación $\mathcal{I}_1 \sim \mathcal{I}_2$ siempre que $(\mathsf{init}, \vec{v}_1) \sim (\mathsf{init}, \vec{v}_2)$ para cualquier vector $\vec{v}_1$ y $\vec{v}_2$ que representen las valuaciones de los relojes en $\mathcal{I}_1$ y $\mathcal{I}_2$, respectivamente.

Entonces, el hecho de que la equivalencia de bisimulación es una congruencia en IOSAs se sigue del Teorema 3.2 y Corolario 3.1 y se establece en el siguiente teorema.

Teorema 3.4. Sean $\mathcal{I}_1$ y $\mathcal{I}_2$ dos IOSAs tal que $\mathcal{I}_1 \sim \mathcal{I}_2$. Entonces, para cualquier IOSA $\mathcal{I}_3$, $\mathcal{I}_1||\mathcal{I}_3 \sim \mathcal{I}_2||\mathcal{I}_3$ y $\mathcal{I}_3||\mathcal{I}_1 \sim \mathcal{I}_3||\mathcal{I}_2$.

[2] Observe que el dominio y la imagen de Hf aparecen aparentemente invertidos. Esto es necesario en [50] ya que solo tratan con morfismos y estamos siguiendo sus definiciones. En nuestro caso, también podríamos haber definido un mapa directo desde $\Delta(\mathscr{B}(\mathbf{S}))$ a $\Delta(\mathscr{B}(\mathbf{S}'))$ dado que Δf es bimesurable, es decir $H(f^{-1}) = (\Delta(f^{-1}))^{-1}$.

3.6. Determinismo

Un IOSA *cerrado* es un IOSA en el que todas las sincronizaciones, si alguna vez existieron en el modelo, han sido resueltas a través de composición paralela. Por lo tanto, no tiene acciones de entrada (es decir, $\mathcal{A}^{\mathsf{i}} = \varnothing$) y, por lo tanto, representa un modelo totalmente generativo. Recién entonces es razonable hablar de determinismo en el modelo.

En esta sección mostramos que un IOSA cerrado es determinista. Los IOSA deterministas son aptos para la simulación de eventos discretos o, en caso de que todos sus relojes sean variables aleatorias distribuidas exponencialmente, también aptos para el análisis como una cadena de Markov de tiempo continuo. Diremos que un IOSA es determinista si es casi seguro que se habilita como máximo una transición discreta en cada punto de tiempo. Para evitar referirnos explícitamente al tiempo, decimos en cambio que un IOSA es determinista si casi nunca alcanza un estado en el que se habilitan dos transiciones discretas diferentes.

Definición 3.6. Un IOSA $\mathcal{I}$ es *determinístico* siempre que en $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$, un estado $(s, \vec{v}) \in \mathbf{S}$ tal que $\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ contiene al menos dos medidas de probabilidad diferentes, casi nunca se alcanza desde ningún $(\mathsf{init}, \vec{v}') \in \mathbf{S}$.

Por "casi nunca" queremos decir que la medida del conjunto de todos los caminos que conducen a un estado $(s, \vec{v}) \in \mathbf{S}$ tal que $\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ contiene al menos dos elementos es 0. Una definición estrictamente formal de esto requiere una serie de definiciones relacionadas con schedulers y medidas en paths en NLMPs que no es crucial para el desarrollo del resultado. (Para obtener una definición formal de scheduler y medidas de probabilidad en paths en NLMPs, consulte [102, Cap. 7]).

Por supuesto, para garantizar determinismo, también se deben tener en cuenta las transiciones temporales. Por lo tanto, la definiciń anterior solo tiene sentido si $\mathcal{P}(\mathcal{I})$ satisface *aditividad de tiempo*, *determinismo de tiempo* y *máximo progreso* [105]. En particular, por máximo progreso entendemos que el tiempo no puede avanzar si se habilita una transición de salida.

Teorema 3.5. Para un IOSA $\mathcal{I}$, su semántica $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ satisface, para todo $(s, \vec{v}) \in \mathbf{S}$, $a \in \mathcal{A}^{\circ}$ y $d, d' \in \mathbb{R}_{>0}$,

progreso máximo: $\mathcal{T}_a(s, \vec{v}) \neq \varnothing \;\Rightarrow\; \mathcal{T}_d(s, \vec{v}) = \varnothing$

determinismo del tiempo: $\mu, \mu' \in \mathcal{T}_d(s, \vec{v}) \;\Rightarrow\; \mu = \mu'$, y

aditividad de tiempo: $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s, \vec{v}) \wedge \delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s, \vec{v} - d) \;\Leftrightarrow\; \delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s, \vec{v})$.

Demostración. Observe que si $\mathcal{T}_a(s,\vec{v}) \neq \varnothing$, con $a \in \mathcal{A}^\circ$, entonces existe una transición $s \xrightarrow{\{x_j\},a,C'} s'$ tal que $\vec{v}(j) \leq 0$. Supongamos por contradicción que $\mathcal{T}_d(s,\vec{v}) \neq \varnothing$, entonces $0 < d \leq \min\{\vec{v}(i) \mid \exists a{\in}\mathcal{A}^\circ, C'{\subseteq}\mathcal{C}, s'{\in}S : s \xrightarrow{\{x_i\},a,C'} s'\} \leq \vec{v}(j) \leq 0$, lo cual es una contradicción.

El determinismo temporal es inmediato por definición 3.2 ya que ya sea $\mathcal{T}_d(s,\vec{v}) = \{\delta^{-d}_{(s,\vec{v})}\}$ o $\mathcal{T}_d(s,\vec{v}) = \varnothing$.

Para la aditividad del tiempo, sea $\hat{d} = \min\{\vec{v}(i) \mid \exists a{\in}\mathcal{A}^\circ, C{\subseteq}\mathcal{C}, s'{\in}\mathcal{S} : s \xrightarrow{\{x_i\},a,C} s'\}$. Supongamos que $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s,\vec{v})$ y $\delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s,\vec{v}-d)$. Por definición 3.2, $0 < d \leq \hat{d}$ y $0 < d' \leq \hat{d} - d$, es decir, $0 < d + d' \leq \hat{d}$. De este modo $\delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s,\vec{v})$. Supongamos ahora que $\delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s,\vec{v})$. Después $0 < d + d' \leq \hat{d}$ y por lo tanto $0 < d \leq \hat{d}$ y $0 < d' \leq \hat{d} - d$, lo que implica que $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s,\vec{v})$ y $\delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s,\vec{v}-d)$. □

Presentamos ahora el teorema principal de esta sección.

Teorema 3.6. Todo IOSA cerrado es determinista.

El resto de la sección está dedicada a probar este teorema. A partir de ahora, trabajamos con un IOSA $\mathcal{I} = (\mathcal{S},\mathcal{C},\mathcal{A},\rightarrow,s_0,\mathcal{C}_0)$ cerrado, con $|\mathcal{C}| = N$, y su semántica $\mathcal{P}(\mathcal{I}) = (\mathbf{S},\mathscr{B}(\mathbf{S}),\{\mathcal{T}_a \mid a \in \mathcal{L}\})$. Recordemos que IOSAs sólo admiten muestreo de valores de reloj de variables aleatorias continuas, lo cual es esencial para la validez del Teorema 3.6.

Para cada estado $s \in \mathcal{S}$, sea $\mathit{active}(s) = \{x \mid s \xrightarrow{\{x\},a,C} s'\}$ el conjunto de relojes activos en el estado s. Por definición 3.1(d) se deduce que $\mathit{active}(s') \subseteq (\mathit{active}(s)\backslash\{x\}) \cup C$ siempre que $s \xrightarrow{\{x\},a,C} s'$.

La idea de la prueba del Teorema 3.6 es mostrar que la propiedad de que todos los relojes activos tienen valores no negativos y son diferentes entre sí es casi con certeza una invariante de $\mathcal{I}$, y que a lo sumo se habilita una transición en cada estado que satisface tal invariante. Formalmente, la invariante es el conjunto

$$\begin{aligned}\mathsf{Inv} = \{(s,\vec{v}) \mid s \in \mathcal{S}, \vec{v}(i) \neq \vec{v}(j), \text{ and } \vec{v}(i) \geq 0 \\ \text{for all } x_i, x_j \in \mathit{active}(s) \text{ with } i \neq j\} \cup (\{\mathsf{init}\}{\times}\mathbb{R}^N)\end{aligned} \tag{3.1}$$

por lo tanto, su complemento es

$$\begin{aligned}\mathsf{Inv}^c = \{(s,\vec{w}) \mid s \in \mathcal{S}, \vec{w}(i) = \vec{w}(j) \text{ for some } x_i, x_j \in \mathit{active}(s) \text{ with } i \neq j\} \\ \cup\ \{(s,\vec{w}) \mid s \in \mathcal{S}, \vec{w}(i) < 0 \text{ for some } x_i \in \mathit{active}(s)\}\end{aligned} \tag{3.2}$$

El siguiente lema establece que Inv^c casi nunca se alcanza en un solo paso desde un estado que satisface la invariante.

Lema 3.3. Para todo $(s, \vec{v}) \in \mathsf{Inv}$, $a \in \mathcal{L}$, y $\mu \in \mathcal{T}_a(s, \vec{v})$, $\mu(\mathsf{Inv}^c) = 0$.

Demostración. Procedemos analizando por casos, según a es init, está en $\mathcal{A}$, o en $\mathbb{R}_{>0}$.

Para $a = \mathsf{init}$, solo consideramos estados de la forma $(\mathsf{init}, \vec{v})$ desde $\mathcal{T}_{\mathsf{init}}(s, \vec{v}) \neq \varnothing$ si y solo si $s = \mathsf{init}$. Entonces, sea $\mu \in \mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v})$. Luego $\mu = \delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}$. Desde cada uno μ_{x_i} es una medida de probabilidad continua (de ahí la probabilidad de que dos relojes posean el mismo valor es 0) y $\mu_{x_i}(\mathbb{R}_{>0}) = 1$, luego $\mu(\mathsf{Inv}^c) = 0$.

Para $a \in \mathcal{A}$, tomemos $\mu \in \mathcal{T}_a(s, \vec{v})$ con $(s, \vec{v}) \in \mathsf{Inv}$. Observe que $s \in \mathcal{S}$. Por definición 3.2 y porque $\mathcal{I}$ es cerrado, existe $s \xrightarrow{\{x\},a,C} s'$ con $\vec{v}(i) \leq 0$ y $\mu = \mu_{\vec{v},C,s'} = \delta_{s'} \times \prod_{i \in I} \mu_{x_i} \times \prod_{j \in J} \delta_{\vec{v}(j)}$ donde $I = \{i \mid x_i \in C\}$ y $J = \{j \mid x_j \notin C\}$.

Para cada $x_i, x_j \in \mathit{active}(s')$ definamos $\mathsf{Inv}_{ij}^c = \{(s'', \vec{w}) \mid s'' \in \mathcal{S}, \vec{w}(i) = \vec{w}(j)\}$ siempre que $i \neq j$, y $\mathsf{Inv}_i^c = \{(s'', \vec{w}) \mid s'' \in \mathcal{S}, \vec{w}(i) < 0\}$. Observe que $\mathsf{Inv}^c = \bigcup \mathsf{Inv}_{ij}^c \cup \bigcup \mathsf{Inv}_i^c$ y, dado que las uniones son finitas, $\mu(\mathsf{Inv}^c) = 0$ sii $\mu(\mathsf{Inv}_{ij}^c) = 0$ y $\mu(\mathsf{Inv}_i^c) = 0$, por cada i, j. A continuación demostramos esta última afirmación.

Sea $x_i \in \mathit{active}(s')$. Entonces $x_i \in (\mathit{active}(s) \backslash \{x\}) \cup C$. Si $x_i \in C$, entonces $\mu(\mathsf{Inv}_i^c) = 0$ porque $\mu_i(\mathbb{R}_{\geq 0}) = 1$. Si en cambio $x_i \in \mathit{active}(s) \backslash \{x\}$, entonces $\mu(\mathsf{Inv}_i^c) = 0$ porque $\delta_{\vec{v}(i)}(\mathbb{R}_{\geq 0}) = 1$, ya que $(s, \vec{v}) \in \mathsf{Inv}$ y por lo tanto $\vec{v}(i) \geq 0$.

Sea $x_i, x_j \in \mathit{active}(s')$ con $i \neq j$. Después $x_i, x_j \in (\mathit{active}(s) \backslash \{x\}) \cup C$. Si $x_i \in C$ entonces μ_i es una medida de probabilidad continua y por lo tanto $\mu(\mathsf{Inv}_{ij}^c) = 0$. Del mismo modo, si $x_j \in C$. Si en cambio $x_i, x_j \in \mathit{active}(s) \backslash \{x\}$, entonces $\delta_{\vec{v}(i)} \neq \delta_{\vec{v}(j)}$ porque $(s, \vec{v}) \in \mathsf{Inv}$ y por lo tanto $\vec{v}(i) \neq \vec{v}(j)$. Por lo tanto $\mu(\mathsf{Inv}_{ij}^c) = 0$. Esto prueba que $\mu(\mathsf{Inv}^c) = 0$ para este caso.

Finalmente, tome $d \in \mathbb{R}_{>0}$ y suponga que $\mathcal{T}_d(s, \vec{v}) = \{\delta_{(s,\vec{v})}^{-d}\}$ con $(s, \vec{v}) \in \mathsf{Inv}$. Observe que $s \in \mathcal{S}$. Por definición 3.2, $0 < d \leq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\},a,C'} s', a \in \mathcal{A}^\circ\}$ y $\delta_{(s,\vec{v})}^{-d} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$. Tomamos los conjuntos Inv_{ij}^c y Inv_i^c como antes y seguimos un razonamiento parecido. Para $x_i \in \mathit{active}(s)$, $\vec{v}(i) - d \geq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\},a,C'} s', a \in \mathcal{A}^\circ\} - d \geq 0$ y por lo tanto $\delta_{\vec{v}(i)-d}(\mathbb{R}_{\geq 0}) = 1$. Por lo tanto $\mu(\mathsf{Inv}_i^c) = 0$. Para $x_i, x_j \in \mathit{active}(s)$ con $i \neq j$, $\delta_{\vec{v}(i)-d} \neq \delta_{\vec{v}(j)-d}$ porque $(s, \vec{v}) \in \mathsf{Inv}$ y por lo tanto $\vec{v}(i) \neq \vec{v}(j)$. Luego $\mu(\mathsf{Inv}_{ij}^c) = 0$. Esto prueba que $\mu(\mathsf{Inv}^c) = 0$ para este caso, y por lo tanto el lema. □

Por Lemma 3.3 obtenemos el siguiente corolario.

Corolario 3.2. El conjunto Inv^c es casi nunca alcanzado en $\mathcal{P}(\mathcal{I})$.

La prueba del corolario requiere, nuevamente, las definiciones relacionadas con schedulers y medidas en paths en NLMPs. Lo omitimos aquí ya que la

prueba finalmente se reduce a aplicar directamente Lemma 3.3 y ver que la medida de todos los caminos que conducen a un estado en Inv^c es 0 para todos los schedulers posibles.

El siguiente lema establece que cualquier estado en la invariante Inv tiene como máximo una transición discreta habilitada.

Lema 3.4. Para todo $(s, \vec{v}) \in \mathsf{Inv}$, el conjunto $\mathit{enabled}(s, \vec{v}) = \bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ is un singletón o el conjunto vacío.

Demostración. Por Def 3.2, $\mathit{enabled}(\mathsf{init}, \vec{v}) = \mathcal{T}_{\mathsf{init}}(s, \vec{v}) = \{\delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}\}$, lo cual prueba este caso. Por lo tanto, sea $(s, \vec{v}) \in \mathsf{Inv}$ with $s \in \mathcal{S}$ y supongamos que $\mathit{enabled}(s, \vec{v}) \neq \varnothing$. Por Def 3.2, existe al menos una transición $s \xrightarrow{\{x_i\}, a, C} s'$ tal que $\vec{v}(i) \leq 0$. Porque, $(s, \vec{v}) \in \mathsf{Inv}$ y $x_i \in \mathit{active}(s)$, entonces $\vec{v}(i) = 0$ y para todo $x_j \in \mathit{active}(s)$ con $i \neq j$, $\vec{v}(j) > 0$. La condición (c) en la Definición 3.1 asegura que no hay otra transición $s \xrightarrow{\{x_i\}, b, C'} s''$ y, como consecuencia, $\mathit{enabled}(s, \vec{v})$ es un singletón. □

Finalmente, la demostración del Teorema 3.6 es una consecuencia directa del Corolario 3.2 y el Lema 3.4.

Prueba del Teorema 3.6. Sea $\mathsf{En}_{\geq 2} = \{(s, \vec{v}) \in \mathsf{S} \mid \ |\mathit{enabled}(s, \vec{v})| \geq 2\}$. Por Corolario 3.2, $\mathsf{En}_{\geq 2} \subseteq \mathsf{Inv}^c$. Por lo tanto, por Lema 3.4, $\mathsf{En}_{\geq 2}$ es casi nunca alcanzable. □

3.7. Conclusión

En este capítulo hemos definido Input/Output Stochastic Automata. IOSA disfruta de las siguientes características importantes:

- IOSA es composicional, lo que permite la reutilización de componentes, reduce significativamente el problema de explosión del espacio de estado y facilita la ingeniería de modelado de sistemas.
- IOSA permite modelar eventos que ocurren según medidas generales de probabilidad continua y no solo memoryless.
- Se garantiza que los modelos IOSA cerrados son totalmente deterministas, es decir, todas las opciones se resuelven probabilísticamente.

Estas características permiten utilizar IOSA como un modelo eficiente y fiel a la realidad para la simulación de eventos discretos. Se ha utilizado como lenguaje de especificación de entrada en la primera versión de la herramienta de simulación de eventos raros FIG (consulte la Sección 5.8.1).

En este capítulo se han tratado todas las formalidades necesarias para respaldar el lenguaje: su semántica se ha dado en términos de NLMP. Se ha definido una relación de bisimulación (sobre NLMP y luego elevada a IOSAs) así como un operador de composición paralelo. Además, se ha demostrado que la bisimulación en IOSAs es una congruencia con respecto a la composición paralela. Finalmente, hemos demostrado que los modelos IOSA cerrados son completamente deterministas y, por lo tanto, se pueden simular eventos discretos sin ninguna intervención adicional de schedulers ni experiencia humana para resolver el no determinismo.

Capítulo 4

IOSA con urgencia

En el capítulo anterior presentamos IOSA, un lenguaje de modelado para sistemas temporizados estocásticos con distribuciones generales. De sus características destacamos (i) la posibilidad de definir la ocurrencia de eventos distribuida en manera general , lo que permite una mejor representación de los sistemas reales a modelar en comparación con otros formalismos que solo permiten eventos distribuidos sin memoria; (ii) que un modelo IOSA cerrado es determinista y, por lo tanto, se puede someter a simulación de eventos discretos; (iii) que la composicionalidad en IOSA ataca el crecimiento exponencial del espacio de estado, dando como resultado un marco de análisis más eficiente y proporcionando modelos más limpios y reutilizables.

Este capítulo y, más específicamente, la introducción de transiciones urgentes en IOSA encuentra motivación en el hecho de que la composicionalidad en IOSA está muy limitada por la sincronización que se toma solo al expirar un reloj (ver regla de paralelización R3). Esto restringe nuestras capacidades de modularización y hace un uso limitado de los beneficios de la composicionalidad.

De hecho, al experimentar con la herramienta de simulación de eventos raros FIG [28, 25, 41], a menudo hemos experimentado dificultades en el modelado composicional y esto finalmente nos lleva a modelar un gran componente monolítico en lugar de un modelo composicional más natural que surgiría de forma más natural en otros lenguajes como Modest [14, 60, 61].

Para ilustrar este problema, en la Figura 4.1 encontramos un modelo gráfico de un sumador completo de dos bits. Al igual que otros dispositivos electrónicos, se describe mediante compuertas, que producen señales de salida como una combinación lógica de señales de entrada. Una forma intuitiva y reutilizable de modelar un dispositivo de este tipo debería tener en cuenta que todas las puertas de un tipo dado funcionan de la misma manera. Por lo tanto, un buen diseño debería, en principio, modelar cada tipo de puer-

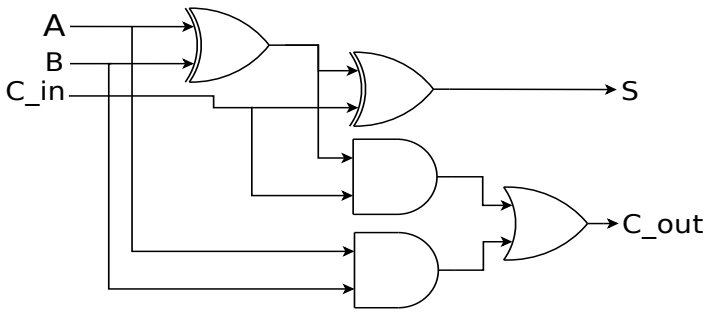


Figura 4.1: Sumador completo de dos bits. El sumador suma los bits de entrada A y B, tomando en cuenta el bit the acarreo C_{in}. El resultado se señala en S y si hay algún acarreo, entonces se coloca en C_{out}.

ta como un componente separado y combinar sus entradas y salidas para sincronizarse entre sí y producir el resultado deseado. Una abstracción habitual al analizar este tipo de sistemas, considera que las compuertas funcionan instantáneamente, y se supone que la comunicación entre ellas también es instantánea. Por lo tanto, un enfoque composicional como el descrito se vuelve imposible de obtener con el modelo IOSA presentado en el capítulo anterior, ya que la sincronización entre los componentes de la puerta está obligada a introducir relojes. Por lo tanto, nos vemos obligados a producir un modelo monolítico en IOSA.

En este capítulo presentamos *Input/Output Stochastic Automata con Urgencia* (IOSA_u), una extensión de IOSA con acciones urgentes. La ocurrencia de transiciones urgentes de salida, es decir, aquellas etiquetadas por acciones urgentes de salida, no se rigen por la expiración de un reloj. En su lugar, se realiza una transición urgente inmediatamente, tan pronto como se alcanza un estado en el que está habilitada. Las salidas urgentes todavía se consideran generativas y las entradas urgentes reactivas, pero ninguna de ellas tiene restricciones de reloj en sus guardas.

Con la introducción de este nuevo tipo de transiciones, mejoramos una de las características más útiles de IOSA, como es su composicionalidad, pero por otro lado tenemos un inconveniente en otra de sus interesantes características: esta extensión introduce no-determinismo incluso en los modelos cerrados. No obstante, comprobaremos que este no-determinismo se produce entre acciones urgentes y que, habitualmente, se introduce por el entrecruzamiento de acciones urgentes *confluentes*. Este no determinismo resulta ser espurio en el sentido de que no cambia los resultados estocásticos del comportamiento del sistema.

Basándonos en [36], definimos una noción de determinismo débil para

IOSA$_u$s confluentes, demostrando que estos autómatas mantienen su comportamiento estocástico independientemente de la resolución del no determinismo introducido por acciones urgentes. También basándonos en el trabajo de Crouzen [36], proporcionamos condiciones suficientes para asegurar que una red de IOSA$_u$s que interactúan sea confluente y, por lo tanto, débilmente determinista, sin la necesidad de obtener el modelo compuesto, evitando así el problema habitual de explosión de estado. Tales condiciones se pueden verificar en tiempo polinomial.

4.1. Input/Output Stochastic Automata with Urgency (IOSA$_u$)

La definición formal de IOSA$_u$ es similar a la de IOSA. Al igual que antes, se dan condiciones con el fin de asegurar que la estructura definida sea determinista en ausencia de acciones urgentes. En el caso de IOSA$_u$, no sólo las acciones de entrada sino también las acciones urgentes carecen de un reloj habilitador. Esto corresponde al hecho de que se activarán tan pronto como se alcance el estado.

Definición 4.1. Un *Input/Output Stochastic Automaton with Urgency* es una estructura $(\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0)$, donde

- $\mathcal{S}$ es un conjunto (numerable) de estados,
- $\mathcal{A}$ es un conjunto (numerable) the etiquetas particionado en conjuntos disjuntos de etiquetas de *input* $\mathcal{A}^{\mathsf{i}}$ y *ouput* $\mathcal{A}^{\mathsf{o}}$, de las cuales un subconjunto $\mathcal{A}^{\mathsf{u}} \subseteq \mathcal{A}$ es marcado como *urgent*,
- $\mathcal{C}$ es un conjunto (finito) de relojes tal que cada $x \in \mathcal{C}$ tiene una medida de probabilidad continua asociada μ_x en $\mathbb{R}$ tal que $\mu_x(\mathbb{R}_{>0}) = 1$,
- $\rightarrow \subseteq \mathcal{S} \times \mathcal{C} \times \mathcal{A} \times \mathcal{C} \times S$ es una función de transición,
- C_0 es el conjunto de relokes que están inicializados en el estado inicial, y
- $s_0 \in \mathcal{S}$ es el estado inicial.

Además, un IOSA$_u$ debe satisfacer las siguientes restricciones:

(a) Si $s \xrightarrow{C,a,C'} s'$ y $a \in \mathcal{A}^{\mathsf{i}} \cup \mathcal{A}^{\mathsf{u}}$, entonces $C = \varnothing$.

(b) Si $s \xrightarrow{C,a,C'} s'$ y $a \in \mathcal{A}^{\mathsf{o}} \setminus \mathcal{A}^{\mathsf{u}}$, entonces C es un singletón.

(c) Si $s \xrightarrow{\{x\},a_1,C_1} s_1$ y $s \xrightarrow{\{x\},a_2,C_2} s_2$ entonces $a_1 = a_2$, $C_1 = C_2$ y $s_1 = s_2$.

(d) Para toda $a \in \mathcal{A}^{\mathsf{i}}$ y estado s, existe una transición $s \xrightarrow{\varnothing,a,C} s'$.

(e) Para toda $a \in \mathcal{A}^{\mathsf{i}}$, si $s \xrightarrow{\varnothing,a,C'_1} s_1$ y $s \xrightarrow{\varnothing,a,C'_2} s_2$, $C'_1 = C'_2$ y $s_1 = s_2$.

(f) Existe una función $\mathit{active} : \mathcal{S} \to 2^{\mathcal{C}}$ tal que:

(I) $\mathit{active}(s_0) \subseteq C_0$,

(II) $\mathit{enabling}(s) \subseteq \mathit{active}(s)$,

(III) si s es estable, $\mathit{active}(s) = \mathit{enabling}(s)$, y

(IV) si $t \xrightarrow{C,a,C'} s$ entonces $\mathit{active}(s) \subseteq (\mathit{active}(t) \setminus C) \cup C'$.

donde $\mathit{enabling}(s) = \{y \mid s \xrightarrow{\{y\},\_,\_} \_\}$, y s es *estable* si no hay $a \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ tal que $s \xrightarrow{\varnothing,a,\_} \_$. ($\_$ indica la cuantificación existencial de un parámetro.)

La ocurrencia de una transición de salida está controlada por la expiración de los relojes. Si $a \in \mathcal{A}^{\circ}$, $s \xrightarrow{C,a,C'} s'$ indica que hay una transición del estado s al estado s' que se puede tomar solo cuando todos los relojes en C han expirado y, cuando se toman, desencadenan la acción a y configuran todos los relojes en C' a un valor muestreado según su distribución de probabilidad asociada. Tenga en cuenta que si $C = \varnothing$ (que significa $a \in \mathcal{A}^{\circ} \cap \mathcal{A}^{\mathsf{u}}$) $s \xrightarrow{C,a,C'} s'$ se activa inmediatamente.

Las restricciones (a) a (f) aseguran que cualquier *cerrado* IOSA$_u$ sin acciones urgentes es determinista, al igual que en el IOSAoriginal. Un IOSA$_u$ se cierra si todas sus sincronizaciones han sido resueltas, es decir, el IOSA$_u$ resultante de una composición no tiene acciones de entrada ($\mathcal{A}^{\mathsf{i}} = \varnothing$). Dado que estas restricciones son muy similares a las de IOSA (3.1), solo analizamos las diferencias. La restricción (a) es doble: por un lado, especifica que las acciones urgentes de salida deben ocurrir tan pronto como se alcance el estado habilitador; por otro lado, como las acciones de entrada son reactivas y su ocurrencia temporal solo puede depender de la interacción con una salida, ningún reloj puede controlar su habilitación.

Finalmente, (f) asegura que los relojes que permiten alguna transición de salida no hayan expirado antes, es decir, no hayan sido utilizados antes por otra transición de salida (sin ser reiniciados en medio) ni llegó inadvertidamente a cero. La función *active* recopila todos los relojes que deben estar activos (es decir, que se han configurado pero aún no han expirado) en cada estado. Tenga en cuenta que se requiere que los relojes habilitados estén activos (condiciones (f)(II) y (f)(III)). También tenga en cuenta que los relojes

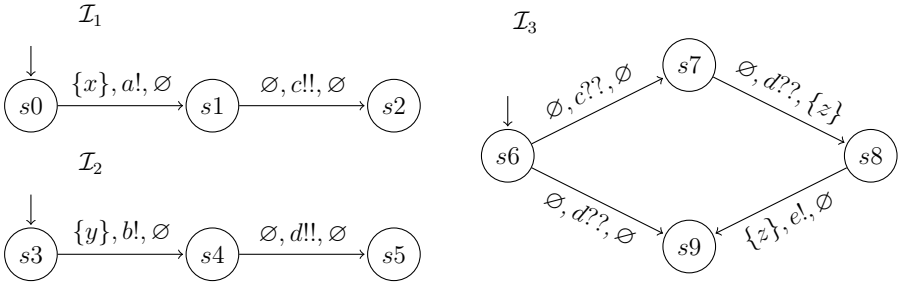


Figura 4.2: Ejemplos de IOSA$_u$s.

que están activos en un estado pueden permanecer activos en un estado sucesor siempre que el reloj no se haya utilizado, y los relojes que se acaban de configurar pueden volverse activos en el estado sucesor (condición (f)(IV)). Además, esta condición asegura que todos los relojes habilitados inicialmente estén incluidos en el conjunto de relojes inicial. Tenga en cuenta que esta condición reemplaza las condiciones (d) y (e) de la definición 3.1. En este caso, debemos tener en cuenta los relojes que se reinician mediante una serie de transiciones urgentes que finalmente llegan al estado estable.

La figura 4.2 muestra tres ejemplos simples de IOSA$_u$s. Aunque los IOSA$_u$s son input-enabled, hemos omitido los bucles de entrada que permiten las transiciones en aras de la legibilidad. En la figura, representamos acciones de salida con el sufijo '!' y con '!!' cuando son urgentes, y acciones de entrada con el sufijo '?' y con '??' cuando son urgentes.

4.2. Semántica de IOSA$_u$

Al igual que con IOSA, la semántica de IOSA$_u$ se define en términos de Non-deterministic Labeled Markov Processes (NLMP) que extiende LMP con no determinismo *interno* (ver Sección 2.2) .

La semántica formal de un IOSA$_u$ está definida por un NLMP con dos clases de transiciones: una que codifica los pasos discretos y contiene toda la información probabilística introducida por el muestreo de relojes, y otra que describe los pasos de tiempo que solo registra el paso del tiempo disminuyendo sincrónicamente el valor de todos los relojes. Nuevamente, por simplicidad, asumimos que el conjunto de relojes tiene un orden particular y sus valores corrientes siguen el mismo orden en un vector.

Definición 4.2. Dado un IOSA$_u$ $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0)$ with $\mathcal{C} = \{x_1, \ldots, x_N\}$, su semántica está definida por el NLMP $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ donde

- $\mathbf{S} = (\mathcal{S} \cup \{\mathsf{init}\}) \times \mathbb{R}^N$, $\mathcal{L} = \mathcal{A} \cup \mathbb{R}_{>0} \cup \{\mathsf{init}\}$, con $\mathsf{init} \notin \mathcal{S} \cup \mathcal{A} \cup \mathbb{R}_{>0}$
- $\mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v}) = \{\delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}\}$,
- $\mathcal{T}_a(s, \vec{v}) = \{\mu^{\vec{v}}_{C',s'} \mid s \xrightarrow{C,a,C'} s', \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\}$, para toda $a \in \mathcal{A}$, donde $\mu^{\vec{v}}_{C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$ con $\overline{\mu}_{x_i} = \mu_{x_i}$ si $x_i \in C'$ y $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ de lo contrario, y
- $\mathcal{T}_d(s, \vec{v}) = \{\delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}\}$ si no hay $b \in \mathcal{A}^{\circ} \cap \mathcal{A}^{\mathsf{u}}$ urgente para la cual $s \xrightarrow{\_,b,\_} \_$ y $0 < d \leq \mín\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\circ}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$, y $\mathcal{T}_d(s, \vec{v}) = \varnothing$ de lo contrario, para todo $d \in \mathbb{R}_{\geq 0}$.

El espacio de estados es el espacio producto de los estados del IOSA$_u$ con todas las posibles valuaciones de reloj. Se agrega un estado inicial distinguido init para codificar la inicialización aleatoria de todos los relojes (sería suficiente inicializar los relojes en C_0 pero decidimos por esta simplificación). Dicha codificación se realiza mediante la transición $\mathcal{T}_{\mathsf{init}}$. El espacio de estados está estructurado con el habitual σ-álgebra de Borel. El paso discreto está codificado por $\mathcal{T}_a$, con $a \in \mathcal{A}$. Observe que, en el estado $(s, \vec{v})$, la transición $s \xrightarrow{C,a,C'} s'$ solo tendrá lugar si $\bigwedge_{x_i \in C} \ vecv(i) \leq 0$, es decir, si los valores actuales de todos los relojes en C no son positivos. Para el caso particular de las entradas o acciones urgentes esto siempre será cierto. El siguiente estado real se determinaría aleatoriamente de la siguiente manera: el estado simbólico será s' (esto corresponde a $\delta_{s'}$ en $\mu^{\vec{v}}_{C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$), cualquier reloj que no esté en C' conserva el valor actual (por lo tanto, $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ si $x_i \notin C'$), y cualquier reloj en C' se setea aleatoriamente de acuerdo con su respectiva distribución asociada (por lo tanto, $\overline{\mu}_{x_i} = \mu_{x_i}$ si $x_i \in C'$). El paso de tiempo está codificado por $\mathcal{T}_d(s, \vec{v})$ con $d \in \mathbb{R}_{\geq 0}$. Solo puede tener lugar en d unidades de tiempo si no hay una transición de salida habilitada en el estado actual dentro de las próximas d unidades de tiempo (esto se verifica mediante la condición $0 < d \leq \mín\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\circ}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$). En este caso, el sistema permanece en el mismo estado simbólico (esto corresponde a δ_s en $\delta^{-d}_{(s,\vec{v})} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$), y todos los valores del reloj disminuyen en d unidades de tiempo (representadas por $\delta_{\vec{v}(i)-d}$ en la misma fórmula). Nótese la diferencia con la semántica de transiciones temporizadas de IOSA puro. Esto se debe a la suposición de progreso máximo, que obliga a realizar una transición urgente tan pronto como se habiliten. Codificamos esto al no permitir hacer transiciones de tiempo en presencia de acciones urgentes, es decir, verificamos que no haya $b \in \mathcal{A}^{\circ} \cap \mathcal{A}^{\mathsf{u}}$ urgentes para las cuales $s \xrightarrow{\_,b,\_} \_$ (en cuyo caso $\mathcal{T}_d(s, \vec{v}) = \varnothing$.) En su lugar, observe la naturaleza *paciente* de un estado

$(s, \vec{v})$ que no tiene salida habilitada. Es decir, $\mathcal{T}_d(s, \vec{v}) = \{\delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}\}$ para todo $d > 0$ siempre que no haya una acción de salida $b \in \mathcal{A}^{\circ}$ tal que $s \xrightarrow{-,b,-} \_$.

De manera similar a la Sección 3.4, es posible mostrar que $\mathcal{P}(\mathcal{I})$ es de hecho un NLMP, es decir, que $\mathcal{T}_a$ se mapea en conjuntos medibles en $\Delta(\mathscr{B}(\mathbf{S}))$, y que $\mathcal{T}_a$ es una función medible para cada $a \in \mathcal{L}$. La prueba sigue exactamente como para los lemas 3.1 y 3.2

4.3. Composición paralela

En esta sección definimos la composición paralela de IOSA$_u$. Al hacerlo, debemos evitar los conflictos de nombres en los relojes, de modo que nos aseguremos de que se conserve el comportamiento previsto de cada componente. Además, no permitimos la sincronización entre salidas para garantizar que sean autónomas y controladas localmente. Finalmente, solo debemos permitir sincronizar IOSA$_u$s que concuerden en sus acciones urgentes para asegurar su ocurrencia inmediata. Más precisamente, solo deberíamos permitir la sincronización de IOSA$_u$s compatibles como se define a continuación:

Definición 4.3. Dos IOSA$_u$s $\mathcal{I}_1$ y $\mathcal{I}_2$ son *compatibles* si no comparten acciones de salida ni relojes, es decir, $\mathcal{A}_1^O \cap \mathcal{A}_2^O = \varnothing$ y $\mathcal{C}_1 \cap \mathcal{C}_2 = \varnothing$ y, además, acuerdan acciones urgentes, es decir, $\mathcal{A}_1 \cap \mathcal{A}_2^{\mathsf{u}} = \mathcal{A}_2 \cap \mathcal{A}_1^{\mathsf{u}}$.

Definición 4.4. Dados dos IOSA$_u$s compatibles $\mathcal{I}_1$ y $\mathcal{I}_2$, la composición paralela $\mathcal{I}_1||\mathcal{I}_2$ es un nuevo IOSA$_u$ $(\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0^1||s_0^2)$ donde

(I) $\mathcal{A}^{\circ} = \mathcal{A}_1^{\circ} \cup \mathcal{A}_2^{\circ}$

(II) $\mathcal{A}^{\mathsf{i}} = (\mathcal{A}_1^{\mathsf{i}} \cup \mathcal{A}_2^{\mathsf{i}}) \setminus \mathcal{A}^{\circ}$

(III) $\mathcal{A}^{\mathsf{u}} = \mathcal{A}_1^{\mathsf{u}} \cup \mathcal{A}_2^{\mathsf{u}}$

(IV) $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$

(V) $C_0 = C_0^1 \cup C_0^2$

y $\rightarrow$ está definida por las mismas reglas en la Tabla 3.1 donde escribimos $s||t$ en vez de (s, t).

Def. 4.4 no asegura *a priori* que la estructura resultante satisfaga las condiciones (a)–(f) en Def. 4.1. Esto sólo está garantizado por la siguiente proposición.

Proposición 4.1. Sean $\mathcal{I}_1$ y $\mathcal{I}_2$ dos IOSA$_u$s compatibles. Entonces $\mathcal{I}_1||\mathcal{I}_2$ es en efecto un IOSA$_u$.

Demostración. La prueba de las restricciones (a), (b), (d), y (e) se obtienen mediante una inspección directa de las reglas, considerando que $\mathcal{I}_1$ y $\mathcal{I}_2$ también satisfacen la restricción respectiva, y haciendo un análisis de casos. Dado que $\mathcal{I}_1$ y $\mathcal{I}_2$ son compatibles, la restricción (c) también se obtiene al inspeccionar las reglas teniendo en cuenta, además, que $\mathcal{I}_1$ y $\mathcal{I}_2$ satisfacen la restricción (e).

Para probar (f), debemos tener en cuenta que

$$enabling(s_1) \cap enabling(s_2) = \varnothing$$

lo cual está garantizado por la compatibilidad, y que

$$enabling(s_1||s_2) = enabling(s_1) \cup enabling(s_2)$$

lo cual está garantizado por input enabling.

Tomamos $active(s_1||s_2) = active_1(s_1) \cup active_2(s_2)$ y demostramos que satisface las condiciones (i)–(iv) en (f).

(I) $active(s_0^1||s_0^2) = active_1(s_0^1) \cup active_2(s_0^2) \subseteq C_0^1 \cup C_0^2 = C_0$.

(II) $enabling(s_1||s_2) = enabling(s_1) \cup enabling(s_2) \subseteq active_1(s_1) \cup active_2(s_2) = active(s_1||s_2)$.

(III) Sea $s_1||s_2$ estable, entonces s_1 y s_2 son estables también (garantizado por input enabledness). Entonces $active(s_1||s_2) = active_1(s_1) \cup active_2(s_2) = enabling(s_1) \cup enabling(s_2) = enabling(s_1||s_2)$.

(IV) Sea $t_1||t_2 \xrightarrow{C,a,C'} s_1||s_2$. Probamos por casos de acuerdo con las reglas en la Tabla 3.1

(R1) Sea $a \in \mathcal{A}_1 \setminus \mathcal{A}_2$. Entonces $t_1 \xrightarrow{C,a,C'} s_1$ and $s_2 = t_2$, y calculamos: $active(s_1||s_2) = active_1(s_1) \cup active_2(s_2) = active_1(s_1) \cup active_2(t_2) \subseteq (active_1(t_1) \setminus C) \cup C' \cup active_2(t_2) = ((active_1(t_1) \cup active_2(t_2)) \setminus C) \cup C' = (active(t_1||t_2) \setminus C) \cup C'$. En particular, la penúltima igualdad sigue por compatibilidad.

(R2) Similar al caso anterior si $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$.

(R3) Sea $a \in \mathcal{A}_1 \cup \mathcal{A}_2$. Entonces $t_1 \xrightarrow{C_1,a,C_1'} s_1$ y $t_2 \xrightarrow{C_2,a,C_2'} s_2$, con $C = C_1 \cup C_2$ y $C' = C_1' \cup C_2'$, y podemos calcular: $active(s_1||s_2) = active_1(s_1) \cup active_2(s_2) \subseteq ((active_1(t_1) \setminus C_1) \cup C_1') \cup ((active_2(t_2) \setminus$

$C_2) \cup C_2') = ((active_1(t_1) \cup active_2(t_2)) \setminus C_1 \cup C_2) \cup C_1' \cup C_2' = (active(t_1||t_2) \setminus C) \cup C'$. La penúltima igualdad sigue a la compatibilidad.

□

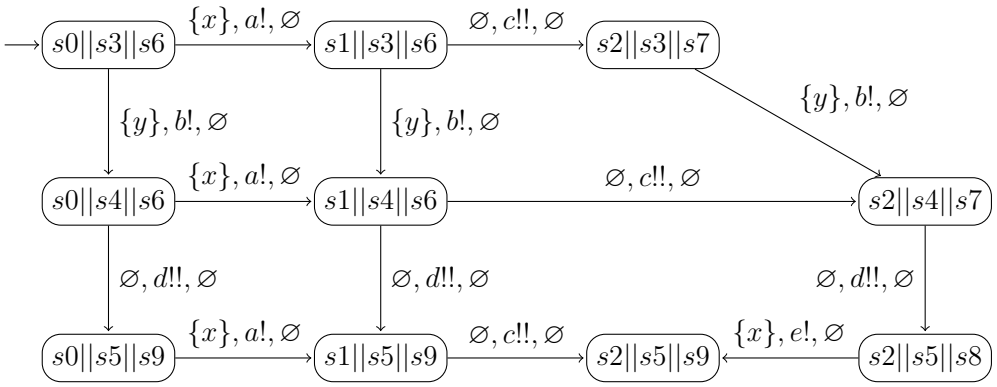


Figura 4.3: Composición $\mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$.

La figura 4.3 muestra el resultado de componer los componentes $\mathcal{I}_1$, $\mathcal{I}_2$ y $\mathcal{I}_3$ del ejemplo 4.2.

Se puede demostrar que la bisimulación es una congruencia para la composición paralela de IOSA$_u$. De hecho, ya hemos demostrado esto para IOSA sin urgencia en el corolario 3.1, y dado que las características de urgencia no juegan ningún papel en la demostración del corolario 3.1 el resultado se extiende inmediatamente a esta nueva configuración. Formalizamos esto en el siguiente teorema.

Teorema 4.1. Sea $\sim$ la relación de bisimimulación en los NLMP de la definición 2.7 correctamente elevada a IOSA$_u$, y sean $\mathcal{I}_1$, $\mathcal{I}_1'$, $\mathcal{I}_2$, $\mathcal{I}_2'$ IOSA$_u$s tal que $\mathcal{I}_1 \sim \mathcal{I}_1'$ y $\mathcal{I}_2 \sim \mathcal{I}_2'$. Entonces, $\mathcal{I}_1||\mathcal{I}_2 \sim \mathcal{I}_1'||\mathcal{I}_2'$.

4.4. Confluencia

En esta sección presentamos una noción de *confluencia*, introducida por primera vez por Robert Milner en su libro *Communication and Concurrency* [81]. Como noción de equivalencia de comportamiento, la confluencia nos ayudará a eliminar ciertos tipos de no determinismo introducidos por acciones urgentes. Decimos que tales casos de no determinismo son espurios, ya que representan situaciones donde cualquier posible decisión sobre qué camino seguir no cambia el comportamiento estocástico del autómata. Llamamos

débilmente determinista a la clase de IOSA$_u$s que sólo presentan tal tipo de no determinismo.

Como se ejemplifica en $\mathcal{I}_3$ en la Figura 4.2, IOSA$_u$s con urgencia pueden ser no determinista (a priori no se indica qué acción tomar en el estado $s6$). Además, incluso los IOSA$_u$s cerrados pueden ser no deterministas. Considere, por ejemplo, la composición de $\mathcal{I}_3$ con $\mathcal{I}_4$ en la Figura 4.4 que da como resultado el IOSA$_u$ compuesto $\mathcal{I}_5$ a la derecha de la Figura 4.4, tenga en cuenta que $\mathcal{I}_5$ está cerrado pero, sin embargo, no es determinista.

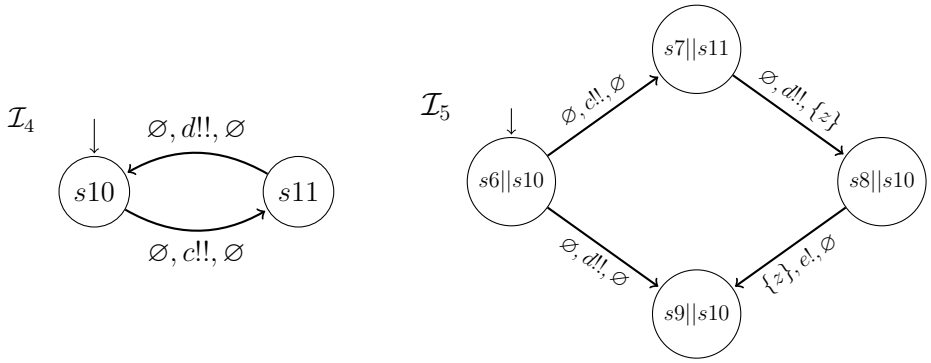


Figura 4.4: $\mathcal{I}_5$ es el resultado de la composición $\mathcal{I}_3||\mathcal{I}_4$.

Using the concept of confluence [81, Chapter 11.3], we proceed to identify an important set of IOSA$_u$s that are weakly-deterministic. In our case we will eventually consider that the urgent actions in a closed IOSA$_u$ are silent, since they do not delay the behaviour of the model. Furthermore, confluent urgent actions can be interchanged while not modifying the stochastic behaviour of the model. Thus, we will focus on the study of confluence over our urgent actions only.

Utilizando el concepto de confluencia [81, Chapter 11.3], procedemos a identificar un conjunto importante de IOSA$_u$s que son débilmente deterministas. En nuestro caso consideraremos eventualmente que las acciones urgentes en un IOSA$_u$ cerrado son silenciosas, ya que no retrasan el comportamiento del modelo. Además, se pueden intercambiar acciones urgentes confluentes sin modificar el comportamiento estocástico del modelo. Por lo tanto, nos centraremos en el estudio de la confluencia sobre nuestras acciones urgentes únicamente.

Más precisamente, un IOSA$_u$ $\mathcal{I}$ es confluente con respecto a las acciones urgentes a y b en $\mathcal{A}^{\mathsf{u}}$ si para cada estado s en $\mathcal{S}$ podemos completar el diagrama de Figura 4.5. Tenga en cuenta que estamos pidiendo converger en un solo estado, que es más fuerte que la confluencia fuerte de Milner, donde la convergencia tiene lugar en estados bisimilares pero potencialmente

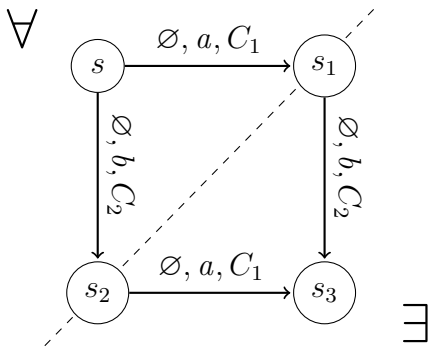


Figura 4.5: Confluencia en IOSA.

distintos. Formalmente:

Definición 4.5. Un IOSA$_u$ $\mathcal{I}$ es confluente con respecto a acciones $a, b \in \mathcal{A}^{\mathsf{u}}$, si para cada estado $s \in \mathcal{S}$ satisface que:

$$s \xrightarrow{\varnothing,a,C_1} s_1 \wedge s \xrightarrow{\varnothing,b,C_2} s_2 \implies \exists s_3 \in \mathcal{S} \cdot s_1 \xrightarrow{\varnothing,b,C_2} s_3 \wedge s_2 \xrightarrow{\varnothing,a,C_1} s_3$$

$\mathcal{I}$ es *confluente* si es confluente con respecto a cada par de acciones urgentes.

La confluencia es preservada por la composición paralela como se demuestra en la siguiente proposición.

Proposición 4.2. Si tanto $\mathcal{I}_1$ como $\mathcal{I}_2$ son confluentes con respecto a las acciones urgentes a y b, entonces también lo es $\mathcal{I}_1||\mathcal{I}_2$. Por lo tanto, si $\mathcal{I}_1$ y $\mathcal{I}_2$ son confluentes, $\mathcal{I}_1||\mathcal{I}_2$ también lo es.

Demostración. Sea $s_1||s_2$ en $\mathcal{S}_{\mathcal{I}_1||\mathcal{I}_2}$, tal que $s_1||s_2 \xrightarrow{\varnothing,a,C'} s_1'||s_2'$ y $s_1||s_2 \xrightarrow{\varnothing,b,C''} s_1''||s_2''$ con $a, b \in \mathcal{A}_{\mathcal{I}_1||\mathcal{I}_2}$. Procedemos mediante análisis por casos sobre cada una de las posibles combinaciones de las reglas de la Tabla 3.1 que originan las transiciones. Probamos el caso en el que $s_1||s_2 \xrightarrow{\varnothing,a,C'} s_1'||s_2'$ se produce por la regla (R1), por lo tanto $a \in \mathcal{A}_{\mathcal{I}_1} \setminus \mathcal{A}_{\mathcal{I}_2}$. El resto de los casos se obtienen de manera similar. Entonces $s_2' = s_2$ y $s_1 \xrightarrow{\varnothing,a,C'} s_1'$. Tenemos entonces tres subcasos dada la naturaleza de b:

- Si $b \in \mathcal{A}_{\mathcal{I}_1} \setminus \mathcal{A}_{\mathcal{I}_2}$, corresponde la regla (R1) y por lo tanto $s_2'' = s_2$ y $s_1 \xrightarrow{\varnothing,b,C'} s_1''$. Dado que $\mathcal{I}_1$ es confluente, existe s_1''' tal que $s_1' \xrightarrow{\varnothing,a,C'} s_1'''$ y $s_1'' \xrightarrow{\varnothing,b,C''} s_1'''$. Usando (R1) en ambos casos, $s_1'||s_2 \xrightarrow{\varnothing,a,C'} s_1'''||s_2$ y $s_1''||s_2 \xrightarrow{\varnothing,b,C''} s_1'''||s_2$, lo cual prueba este caso.

- Si $b \in \mathcal{A}_{\mathcal{I}_2} \backslash \mathcal{A}_{\mathcal{I}_1}$, corresponde (R2) y por lo tanto $s_1 = s_1''$ y $s_2 \xrightarrow{\varnothing, b, C''} s_2''$. Por (R1), $s_1 || s_2'' \xrightarrow{\varnothing, a, C'} s_1' || s_2''$, y por (R2), $s_1' || s_2 \xrightarrow{\varnothing, b, C''} s_1' || s_2''$ lo cual prueba este caso.
- Si $b \in \mathcal{A}_{\mathcal{I}_1} \cap \mathcal{A}_{\mathcal{I}_2}$, corresponde (R3). Por lo tanto hay C_1'' y C_2'' tal que $C'' = C_1'' \cup C_2''$, $s_1 \xrightarrow{\varnothing, b, C_1''} s_1''$ y $s_2 \xrightarrow{\varnothing, b, C_2''} s_2''$. Más aún, dado que $\mathcal{I}_1$ es confluente, existe s_1''' tal que $s_1' \xrightarrow{\varnothing, b, C_1''} s_1'''$ y $s_1'' \xrightarrow{\varnothing, a, C'} s_1'''$. Luego, por (R3), $s_1' || s_2 \xrightarrow{\varnothing, b, C''} s_1''' || s_2''$, y por (R1), $s_1'' || s_2'' \xrightarrow{\varnothing, a, C'} s_1''' || s_2''$, lo cual concluye la prueba.

□

La proposición 4.2 implica que no importa cuán grande y complejo sea nuestro sistema, si se puede descomponer en componentes confluentes, entonces el modelo es confluente. Además, al componer el modelo con otros componentes confluentes se obtiene un nuevo modelo confluente.

Sin embargo, este no es el caso para la no confluencia. De hecho, podría suceder que varios componentes no confluentes se compongan dando como resultado un IOSA$_u$confluente. Más adelante, siguiendo [36], proporcionaremos condiciones suficientes sobre componentes posiblemente no confluentes para garantizar que el IOSA$_u$ compuesto sea, sin embargo, confluente.

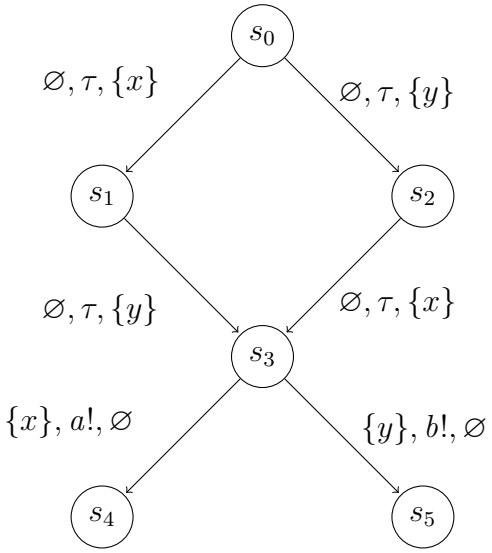


Figura 4.6: La confluencia es weakly determinista

Al observar el IOSA$_u$ de la Fig. 4.6, uno puede notar que el no determinismo introducido por las acciones de salida urgentes confluentes es falso

en el sentido de que no cambia el comportamiento estocástico del modelo. (Aquí $\tau \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$). De hecho, dado que el tiempo no avanza, es lo mismo muestrear primero el reloj x y luego el reloj y pasando por el estado s_1, o primero y y luego x pasando por s_2, o incluso muestreando ambos relojes simultáneamente a través de una transición $s_1 \xrightarrow{\varnothing,\tau,\{x,y\}} s_3$. En cualquiera de estos casos, la resolución estocástica de la ejecución de a o b en el estado estable s_3 es la misma. Esto podría generalizarse a cualquier número de transiciones confluentes.

En este sentido, será conveniente utilizar técnicas de rewriting de términos para recopilar todos los relojes que se encuentran activos en el estado estable convergente y han sido activados a través de un camino de acciones urgentes. El *sistema de reducción* resultante se usará para probar el weak determinismo en IOSA$_u$s confluentes (Sección 4.5). Por ello, recordamos algunas nociones básicas de rewriting de sistemas.

Un *abstract reduction system* [4] es un par $(\mathcal{E}, \rightarrowtail)$, donde la reducción $\rightarrowtail$ es una relación binaria sobre el conjunto $\mathcal{E}$, es decir $\rightarrowtail \subseteq \mathcal{E} \times \mathcal{E}$. Escribimos $a \rightarrowtail b$ para $(a, b) \in \rightarrowtail$. También escribimos $a \overset{*}{\rightarrowtail} b$ para denotar que existe un camino $a_0 \rightarrowtail a_1 \ldots \rightarrowtail a_n$ con $n \geq 0$, $a_0 = a$ y $a_n = b$. Un elemento $a \in \mathcal{E}$ se encuentra en *forma normal* si no hay b tal que $a \rightarrowtail b$. Decimos que b es una forma normal de a si $a \overset{*}{\rightarrowtail} b$ y b se encuentra en forma normal. Un reduction system $(\mathcal{E}, \rightarrowtail)$ es *confluente* si para todos $a, b, c \in \mathcal{E}$ $a \overset{*}{\leftarrowtail} c \overset{*}{\rightarrowtail} b$ implica $a \overset{*}{\rightarrowtail} d \overset{*}{\leftarrowtail} b$ para algún $d \in \mathcal{E}$. Esta noción de confluencia deriva de la siguiente declaración: para todos $a, b, c \in \mathcal{E}$, $a \leftarrowtail c \rightarrowtail b$ implica que o bien $a \rightarrowtail d \leftarrowtail b$ para algún $d \in \mathcal{E}$, o $a = b$.

Un reduction system es *normal* si cada elemento tiene una forma normal, y es *terminal* si no hay una cadena infinita $a_0 \rightarrowtail a_1 \rightarrowtail \cdots$. Un reduction system terminal también es normal. En un reduction system confluente cada elemento tiene como máximo una forma normal. Si además también es normal, entonces tal forma normal siempre existe y es única.

Definimos ahora el abstract reduction system introducido por las transiciones urgentes de un IOSA$_u$. La idea es introducir una reducción en el reduction system cada vez que encontremos una transición urgente en el modelo IOSA$_u$ y dejar que los estados estables sean la forma normal. En la reducción acumulamos todos los relojes seteados a cero y el número de pasos que hemos dado para llegar.

Definición 4.6. Dado un IOSA$_u$ $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow_{\mathcal{I}}, C_0, s_0)$, definase el abstract reduction system $\mathcal{U}_I$ como $(\mathcal{S} \times \mathcal{P}(\mathcal{C}) \times \mathbb{N}_0, \rightarrowtail)$ donde $(s, C, n) \rightarrowtail (s', C \cup C', n+1)$ si y solo si existe $a \in \mathcal{A}^{\mathsf{u}}$ tal que $s \xrightarrow{\varnothing,a,C'} s'$.

Un IOSA$_u$ es *non-Zeno* si no hay ciclo de acciones urgentes. Este concepto

en realidad está relacionado con Zenoness en autómatas temporizados, donde una acción infinita tiene lugar en una cantidad de tiempo finita (ver, por ejemplo, [6]). El siguiente resultado se puede probar directamente.

Proposición 4.3. Sea el IOSA$_u$ $\mathcal{I}$ cerrado y confluente. Entonces $\mathcal{U}_\mathcal{I}$ es confluente, y por lo tanto cada elemento tiene como máximo una forma normal. Además, un elemento (s, C, n) está en forma normal si y solo si s es estable en $\mathcal{I}$. Si además $\mathcal{I}$ no es Zeno, $\mathcal{U}_\mathcal{I}$ también es terminal y, por lo tanto, cada elemento tiene una forma normal única.

El siguiente es el corte interesante del sistema de reducción abstracta para el IOSA$_u$ de la figura 4.6.

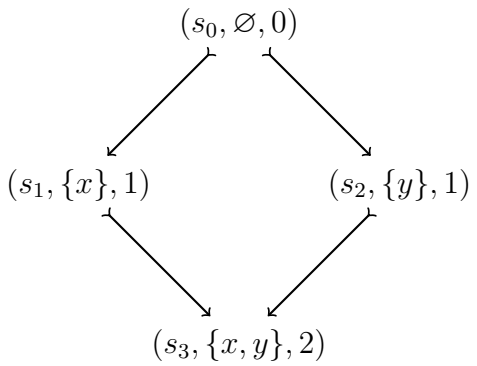


Figura 4.7: Abstract reduction system de Fig. 4.6

El sistema de reducción abstracto de un IOSA$_u$ será útil en la siguiente sección, donde demostramos que los IOSA$_u$s confluentes cerrados son weakly deterministas.

4.5. Determinismo débil

Llamamos a un IOSA$_u$ *cerrado* cuando todas sus sincronizaciones se han resuelto a través de la composición y no quedan acciones de entrada, es decir, $\mathcal{A}^{\mathrm{i}} = \varnothing$ (ver Sección 3.2). En esta sección mostramos que los IOSA$_u$s confluentes cerrados son deterministas. Tenga en cuenta que, en el caso más general, los IOSA$_u$s cerrados pueden no ser deterministas, como lo ejemplifica IOSA$_u$ $\mathcal{I}_3$ en la Figura 4.2.

Un IOSA$_u$ determinista es apto para simulación de eventos discretos o, en caso de que todos sus relojes sean variables aleatorias distribuidas exponencialmente, también apto para el análisis como una cadena de Markov de tiempo continuo. En realidad, mostramos que los IOSA$_u$s confluentes cerrados se

comportan de forma determinista en el sentido de que el comportamiento estocástico del modelo es el mismo, independientemente de la forma en que se resuelva el no determinismo. Así, decimos que un IOSA$_u$ es *débilmente determinista* si

(I) es casi seguro que como máximo se habilita una transición discreta no urgente en cada punto de tiempo,

(II) la elección sobre las transiciones urgentes habilitadas no afecta el comportamiento no urgente del modelo, y

(III) ninguna salida no urgente y salida urgente están habilitadas simultáneamente.

Para evitar referirnos explícitamente al tiempo en (i), decimos en cambio que un IOSA$_u$ es débilmente determinista si casi nunca alcanza un estado en el que se habilitan dos transiciones discretas no urgentes diferentes. Además, para garantizar (ii), definimos la siguiente transición débil, donde la notación $\mathrm{st}(s)$ (léase "s es estable") indica que el estado s no tiene transiciones urgentes habilitadas.

Definición 4.7. Para un estado s no estable, y $v \in \mathbb{R}^N$, definimos $(s, \vec{v}) \stackrel{C}{\Longrightarrow}_n \mu$ inductivamente por las siguientes reglas:

$$\frac{s \xrightarrow{\varnothing, a, C} s' \qquad \mathrm{st}(s')}{(s, \vec{v}) \stackrel{C}{\Longrightarrow}_1 \mu^{\vec{v}}_{C,s'}} \tag{T1}$$

$$\frac{s \xrightarrow{\varnothing, a, C'} s' \qquad \forall \vec{v}' \in \mathbb{R}^N : \exists C'', \mu' : (s', \vec{v}') \stackrel{C''}{\Longrightarrow}_n \mu'}{(s, \vec{v}) \xrightarrow{C' \cup C''}_{n+1} \hat{\mu}} \tag{T2}$$

donde $a \in \mathcal{A}^{\mathsf{u}}$, $\mu^{\vec{v}}_{C,s}$ esta definido como en la Def. 3.2, y $\hat{\mu} = \int_{\mathcal{S} \times \mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$, con $f_n^{C''}(t, \vec{w}) = \nu$, if $(t, \vec{w}) \stackrel{C''}{\Longrightarrow}_n \nu$, y $f_n^{C''}(t, \vec{w}) = \bar{\varnothing}$ en caso contrario.

Definimos las weak *weak transition* $(s, \vec{v}) \Rightarrow \mu$ si $(s, \vec{v}) \stackrel{C}{\Longrightarrow}_n \mu$ para algún $n \geq 1$ y $C \subseteq \mathcal{C}$.

Encontramos aquí una definición inductiva. En esta definición, la función $f_n^{C''}$ actúa como acumulador de las medidas hasta el paso $n-1$. Una integral de Lebesgue define la medida final, dado que $f_n^{C''}$ es medible. Dada de esta manera, no hay garantía de que $\stackrel{C}{\Longrightarrow}_n$ esté bien definida. En particular, no hay garantía de que $f_n^{C''}$ sea una función medible bien definida. Posponemos esto, más abajo, al Lemma 4.1.

Con esta definición, podemos introducir el concepto de determinismo débil:

Definición 4.8. Un IOSA cerrado $\mathcal{I}$ is *weakly determinista* si $\Rightarrow$ está bien definida en $\mathcal{I}$ y, en $P(\mathcal{I})$, cualquier estado $(s, v) \in \mathbf{S}$ que satisfaga una de las siguientes condiciones casi nunca se alcanza desde ningún $(\mathsf{init}, v_0) \in \mathbf{S}$:

(a) s es estable y $\cup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, v)$ contiene al menos dos diferentes medidas de probabilidad,

(b) s es no estable, $(s, v) \Rightarrow \mu$, $(s, v) \Rightarrow \mu'$ y $\mu \neq \mu'$, o

(c) s es no estable y $(s, v) \xrightarrow{a} \mu$ para alguna $a \in \mathcal{A}^{\circ} \setminus \mathcal{A}^{\mathsf{u}}$.

Por "casi nunca" queremos decir que la medida del conjunto de todos los caminos que conducen a cualquier conjunto medible en $\mathscr{B}(\mathbf{S})$ que contiene solo estados que satisfacen (a), (b) o (c) es cero . Así, la definición Def. 4.8 establece que, en un IOSA$_u$ débilmente determinista, una situación en la que una acción de salida no urgente se habilita con otra acción de salida, ya sea urgente el (caso (c)), o caso no urgente (caso (a)), o en el que secuencias de transiciones urgentes conducen a diferentes situaciones estables (caso (b)), casi nunca se alcanza.

Para que la definición anterior tenga sentido, necesitamos que $\mathcal{P}(\mathcal{I})$ satisfaga *time aditivity*, *time determinism* y *maximal progress* [105]. Esto se afirma en el siguiente teorema cuya demostración es muy similar a la del Teorema 3.5.

Teorema 4.2. Sea $\mathcal{I}$ un IOSA$_u$. Su semántica $\mathcal{P}(\mathcal{I})$ satisface, para toda $(s, \vec{v}) \in \mathbf{S}$, $a \in \mathcal{A}^{\circ}$ y $d, d' \in \mathbb{R}_{>0}$, los siguientes tres items:

(I) $\mathcal{T}_a(s, \vec{v}) \neq \varnothing \;\Rightarrow\; \mathcal{T}_d(s, \vec{v}) = \varnothing$ (maximal progress),

(II) $\mu, \mu' \in \mathcal{T}_d(s, \vec{v}) \;\Rightarrow\; \mu = \mu'$ (time determinism), y

(III) $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s, \vec{v}) \wedge \delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s, \vec{v} - d) \;\Leftrightarrow\; \delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s, \vec{v})$ (time additivity).

En el siguiente lema demostramos que, bajo la hipótesis de que IOSA es cerrado y confluente, $\overset{C}{\Rightarrow}_n$ está bien definida. Simultáneamente, demostramos que $\overset{C}{\Rightarrow}_n$ es determinista.

Lema 4.1. Sea $\mathcal{I}$ un IOSA cerrado y confluente. Entonces, para todo $n \geq 1$, se cumple lo siguiente:

1. Si $(s, \vec{v}) \overset{C}{\Rightarrow}_n \mu$ entonces hay un estado estable s' tal que

   (I) $\mu = \mu^{\vec{v}}_{C,s'}$,

(II) $(s, C', m) \overset{*}{\rightarrowtail} (s', C' \cup C, m+n)$ para todo $C' \subseteq \mathcal{C}$ y $m \geq 0$, y

(III) si $(s, \vec{v}') \overset{C'}{\Longrightarrow}_n \mu'$ entonces $C' = C$ y más aún, si $\vec{v}' = \vec{v}$, también $\mu' = \mu$; y

2. f_n^C es una función medible.

Demostración. Procedemos por inducción sobre n demostrando el primer elemento 1 y usándolo para demostrar el 2.

Entonces, supongamos que $n = 1$ y $(s, \vec{v}) \overset{C}{\Longrightarrow}_1 \mu$. Por la regla (T1) en Definition 4.7, existe s' estable tal que $s \xrightarrow{\varnothing, a, C} s'$ para algún $a \in \mathcal{A}^{\mathsf{u}}$ con $\mu = \mu^{\vec{v}}_{C,s'}$, lo que prueba (i). De aquí y Definición 4.6, $(s, C', m) \overset{*}{\rightarrowtail} (s', C' \cup C, m+1)$, demostrando (ii). Para demostrar (iii), suponga $(s, \vec{v}') \overset{C'}{\Longrightarrow}_1 \mu'$. Aplicando (i) y (ii) a esta otra transición, existe un s'' estable tal que $\mu' = \mu^{\vec{v}'}_{C',s''}$ y $(s, \varnothing, 0) \overset{*}{\rightarrowtail} (s'', C', 1)$. Pero también $(s, \varnothing, 0) \overset{*}{\rightarrowtail} (s', C, 1)$ como se demostró antes. Dado que s' y s'' son estables, entonces, por Prop. 4.3, tanto $(s', C, 1)$ como $(s'', C', 1)$ están en forma normal, que también debe ser única. Entonces $s' = s''$ y $C' = C''$. Además, si $\vec{v}' = \vec{v}$ entonces $\mu' = \mu^{\vec{v}'}_{C',s''} = \mu^{\vec{v}}_{C,s'} = \mu$.

Para probar el punto 2 para $n = 1$, observe primero que, por (iii), f_1^C es de hecho una función. Por (i), $f_1^C(t, \vec{w}) = \mu^{\vec{w}}_{C,t'}$ siempre que $(t, \vec{w}) \overset{C}{\Longrightarrow}_1 \mu^{\vec{w}}_{C,t'}$ para algún t' estable que se concede que exista, y $f_1^C(t, \vec{w}) = \bar{\varnothing}$ de lo contrario. Para mostrar que f_1^C es medible, mediante [98, Lemma 3.6], basta probar que $(f_1^C)^{-1}(\Delta^q(A \; times \prod_{i=1}^N V_i))$ es medible para todos los $A \subseteq \mathcal{S}$ y $V_i \in \mathscr{B}(\mathbb{R})$. Note que

$$
\begin{aligned}
&(f_1^C)^{-1}(\Delta^q(A \times \textstyle\prod_{i=1}^N V_i)) = \\
&= \{(t, \vec{w}) \mid \exists t' : (t, \vec{w}) \overset{C}{\Longrightarrow}_1 \mu^{\vec{w}}_{C,t'} \wedge \mu^{\vec{w}}_{C,t'}(A \times \textstyle\prod_{i=1}^N V_i) \geq q\} \\
&= \{(t, \vec{w}) \mid \exists t' \in A : (t, \vec{w}) \overset{C}{\Longrightarrow}_1 \mu^{\vec{w}}_{C,t'} \wedge \textstyle\prod_{x_i \in C} \mu_{x_i}(\prod_{x_i \in C} V_i) \geq q \wedge \forall x_i \notin C : \vec{w}(i) \in V_i\}
\end{aligned}
$$

$$
= \bigcup_{t' \in A} \underbrace{\{(t, \vec{w}) \mid (t, \vec{w}) \overset{C}{\Longrightarrow}_1 \mu^{\vec{w}}_{C,t'} \wedge \textstyle\prod_{x_i \in C} \mu_{x_i}(\prod_{x_i \in C} V_i) \geq q \wedge \forall x_i \notin C : \vec{w}(i) \in V_i\}}_{= X_t}
$$

Observe que, si $\prod_{x_i \in C} \mu_{x_i}(\prod_{x_i \in C} V_i) \geq q$, entonces $X_t = \{t\} \times \prod_{i=1}^N \overline{V}_i$, con $\overline{V}_i = \mathbb{R}$ si $x_i \in C$ y $\overline{V}_i = V_i$ si $x_i \notin C$, y $X_t = \varnothing$ en caso contrario. En ambos casos X_t es medible. Dado que $\mathcal{S}$ es finito, la unión también es finita y, por lo tanto, f_1^C es medible, lo que prueba el caso base.

Para el caso inductivo, sea $n \geq 1$ y suponga $(s, \vec{v}) \stackrel{C}{\Longrightarrow}_{n+1} \mu$. Por (T2), hay C' y C'' tales que $C = C' \cup C''$, $s \xrightarrow{\varnothing, a, C'} s'$, $\forall \vec{v'} \in \mathbb{R}^{\mathbb{N}} : (s', \vec{v'}) \stackrel{C''}{\Longrightarrow}_n \mu'$, y $\mu = \int_{\mathcal{S}\times\mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$. Por inducción, C'' es único (por 1.(iii)), $(s', \vec{v'}) \stackrel{C''}{\Longrightarrow}_n \mu^{\vec{v'}}_{C'',s''}$ para todo $\vec{v'}$ y estado estable único s'' (por 1.(i) y 1.(ii)), y $f_n^{C''}$ es medible (por 2). Por lo tanto, $\int_{\mathcal{S}\times\mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$ está bien definido. Además, observe que $f_n^{C''}(s', \vec{v'}) = \mu^{\vec{v'}}_{C'',s''}$ para todo $vecv'$.

Nos enfocamos en 1.(i) y mostramos que $\mu = \mu^{\vec{v}}_{C'\cup C'',s''}$. Primero, observe que $\mu = \int_{\{s'\}\times\mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'} + \int_{(\mathcal{S}\setminus\{s'\})\times\mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$ y dado que $\mu^{\vec{v}}_{C',s'} = \delta_{s'} \times \prod_{i=1}^N \overline{\mu}^{v_i}_{x_i}$ with $\overline{\mu}^{v_i}_{x_i} = \mu_{x_i}$ if $x_i \in C'$ y $\overline{\mu}^{v_i}_{x_i} = \delta_{v_i}$ caso contrario (escribimos v_i para $\vec{v}(i)$), entonces el segundo sumando es la función nula $\bar{\varnothing}$. Ahora, para $A \subseteq \mathcal{S}$ and $Q_i \in \mathbb{R}$, $1 \leq i \leq N$, calculamos

$$
\begin{aligned}
&\mu(A \times Q_1 \times \cdots \times Q_N) = \\
&= \int_{\{s'\}\times\mathbb{R}^N} f_n^{C''}(t, \vec{w})(A \times Q_1 \times \cdots \times Q_N)\ d\mu^{\vec{v}}_{C',s'}(t, \vec{w}) \\
&= \int_{\mathbb{R}^N} f_n^{C''}(s', \vec{w})(A \times Q_1 \times \cdots \times Q_N)\ d(\textstyle\prod_{i=1}^N \overline{\mu}^{v_i}_{x_i})(\vec{w}) \\
&= \int_{\mathbb{R}^N} \mu^{\vec{w}}_{C'',s''}(A \times Q_1 \times \cdots \times Q_N)\ d(\textstyle\prod_{i=1}^N \overline{\mu}^{v_i}_{x_i})(\vec{w}) = (\dagger)
\end{aligned}
$$

Por definición, $\mu^{\vec{w}}_{C'',s''} = \delta_{s''} \times \prod_{i=1}^N \overline{\mu}^{w_i}_{x_i}$ con $\overline{\mu}^{w_i}_{x_i} = \mu_{x_i}$ if $x_i \in C''$ y $\overline{\mu}^{w_i}_{x_i} = \delta_{v_i}$ caso contrario. Entonces (a continuación omitimos el dominio $\mathbb{R}$ para cada integral), usando el teorema de Fubini, obtenemos:

$$
\begin{aligned}
(\dagger) &= \int\cdots\int \delta_{s''}(A) \cdot \overline{\mu}^{w_1}_{x_1}(Q_1)\cdots\overline{\mu}^{w_N}_{x_N}(Q_N)\ d\overline{\mu}^{v_1}_{x_1}(w_1)...d\overline{\mu}^{v_N}_{x_N}(w_N) \\
&= \delta_{s''}(A) \int\cdots\int \overline{\mu}^{w_2}_{x_2}(Q_2)\cdots\overline{\mu}^{w_N}_{x_N}(Q_N) \Big(\underbrace{\int \overline{\mu}^{w_1}_{x_1}(Q_1)\, d\overline{\mu}^{v_1}_{x_1}(w_1)}_{(*)} \Big) d\overline{\mu}^{v_2}_{x_2}(w_2)...d\overline{\mu}^{v_N}_{x_N}(w_N)
\end{aligned}
$$

nos concentramos en $(*)$. Pueden presentarse tres casos, Si $x_1 \in C''$, entonces

$$(*) = \int \mu_{x_1}(Q_1)\, d\overline{\mu}^{v_1}_{x_1}(w_1) = \mu_{x_1}(Q_1) \int d\overline{\mu}^{v_1}_{x_1}(w_1) = \mu_{x_1}(Q_1)$$

dado que $\int d\overline{\mu}^{v_1}_{x_1}(w_1) = 1$. Si $x_1 \in C' \setminus C''$,

$$(*) = \int \delta_{w_1}(Q_1)\, d\mu_{x_1}(w_1) = \int \chi_{Q_1}(w_1)\, d\mu_{x_1}(w_1) = \mu_{x_1}(Q_1)$$

donde χ_{Q_1} es la función característica usual. Finalmente, si $x_1 \notin C \cup C''$,

$$(*) = \int \delta_{w_1}(Q_1)\, d\delta_{v_1}(w_1) = \int \chi_{Q_1}(w_1)\, d\delta_{v_1}(w_1) = \delta_{v_1}(Q_1).$$

Por lo tanto $(*) = \overline{\mu}_{x_1}(Q_1)$ con $\overline{\mu}_{x_1} = \mu_{x_1}$ if $x_1 \in C' \cup C''$ y $\overline{\mu}_{x_1} = \delta_{v_1}$ caso contrario. Luego, procediendo de la misma manera para todos los índices, continuamos,

$$\begin{aligned}
&= \delta_{s''}(A)\overline{\mu}_{x_1}(Q_1)\int\cdots\int \overline{\mu}^{w_2}_{x_2}(Q_2)\cdots\overline{\mu}^{w_N}_{x_N}(Q_N)\, d\overline{\mu}^{v_2}_{x_2}(w_2)...d\overline{\mu}^{v_N}_{x_N}(w_N)\\
&= \delta_{s''}(A)\cdot\overline{\mu}_{x_1}(Q_1)\cdots\overline{\mu}_{x_N}(Q_N) = (\delta_{s''}\times\textstyle\prod_{i=1}^{N}\overline{\mu}_{x_i})(A\times Q_1\times\cdots\times Q_N)\\
&= \mu^{\vec{v}}_{C\cup C'',s''}(A\times Q_1\times\cdots\times Q_N)
\end{aligned}$$

lo cual prueba 1.(i).

Para probar 1.(ii), por Def. 4.6, $(s, C^*, m) \rightarrowtail (s', C^*\cup C', m{+}1)$ dado que $s \xrightarrow{\varnothing, a, C'} s'$. Por inducción, $(s', \vec{v}') \overset{C''}{\Longrightarrow}_n \mu'$ implica $(s', C^*\cup C', m{+}1) \overset{*}{\rightarrowtail} (s'', C^*\cup C'\cup C'', m{+}1{+}n)$. Luego $(s, C^*, m) \overset{*}{\rightarrowtail} (s'', C^*\cup C'\cup C'', m{+}1{+}n)$, lo cual prueba 1.(ii).

Las pruebas para 1.(iii) y 2 siguen como para el caso base.

□

El siguiente corolario sigue al ítem 1 del Lema 4.1. Establece que $\Rightarrow$ es determinista.

Corolario 4.1. Sea $\mathcal{I}$ un IOSA$_u$ cerrado y confluente. Entonces, para todo $(s, \vec{v})$, si $(s, \vec{v}) \Rightarrow \mu_1$ y $(s, \vec{v}) \Rightarrow \mu_2$, entonces $\mu_1 = \mu_2$.

Este corolario ya muestra que los IOSA$_u$s cerrados y confluentes satisfacen la parte (b) de la Definición 4.8. En general, podemos afirmar:

Teorema 4.3. Cada IOSA confluente cerrada es débilmente determinista.

El resto de la sección está dedicada a probar este teorema. Como ya hemos mostrado que los IOSA$_u$s confluentes cerrados satisfacen (b) de la definición de determinismo débil (Definición 4.8), ahora nos centramos en los puntos (a) y (c). A partir de ahora, trabajamos con el IOSA confluente y cerrado $\mathcal{I} = (\mathcal{S}, \mathcal{C}, \mathcal{A}, \rightarrow, s_0, C_0)$, con $|\mathcal{C}| = N$, y su semántica $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$.

La idea de la prueba del Teorema 4.3 es mostrar que la propiedad de que todos los relojes activos tienen valores no negativos y son diferentes entre sí es casi con certeza una invariante de $\mathcal{I}$, y que a lo sumo se habilita una transición no urgente en cada estado que satisface tal invariante. Además,

queremos mostrar que, para estados inestables, los relojes activos tienen valores estrictamente positivos, lo que implica que las transiciones no urgentes nunca están habilitadas en estos estados. Formalmente, el invariante es el conjunto

$$\begin{aligned}\mathsf{Inv} = \quad & \{(s,\vec{v}) \mid \mathrm{st}(s) \text{ and } \forall x_i, x_j \in \mathit{active}(s) : i \neq j \Rightarrow \vec{v}(i) \neq \vec{v}(j) \wedge \vec{v}(i) \geq 0\} \\ & \cup \{(s,\vec{v}) \mid \neg\mathrm{st}(s) \text{ and } \forall x_i, x_j \in \mathit{active}(s) : i \neq j \Rightarrow \vec{v}(i) \neq \vec{v}(j)\vec{v}(i) > 0\} \\ & \cup (\{\mathsf{init}\} \times \mathbb{R}^N) \end{aligned} \tag{4.1}$$

con *active* como en Def. 4.1. Note que su complemente es:

$$\begin{aligned}\mathsf{Inv}^c = \quad & \{(s,\vec{v}) \mid \exists x_i, x_j \in \mathit{active}(s) : i \neq j \wedge \vec{v}(i) = \vec{v}(j)\} \\ & \cup \{(s,\vec{v}) \mid \mathrm{st}(s) \text{ and } \exists x_i \in \mathit{active}(s) : \vec{v}(i) < 0\} \\ & \cup \{(s,\vec{v}) \mid \neg\mathrm{st}(s) \text{ and } \exists x_i \in \mathit{active}(s) : \vec{v}(i) \leq 0\} \end{aligned} \tag{4.2}$$

No es difícil demostrar que Inv^c es medible y, en consecuencia, también lo es Inv. El siguiente lema establece que Inv^c casi nunca se alcanza en un paso desde un estado que satisface el invariante.

Lema 4.2. Si $(s,\vec{v}) \in \mathsf{Inv}$, $a \in \mathcal{L}$, y $\mu \in \mathcal{T}_a(s,\vec{v})$, luego $\mu(\mathsf{Inv}^c) = 0$.

Demostración. Procedemos analizando por casos según a es init, en $\mathcal{A}$, o en $\mathbb{R}_{>0}$.

Si a es init, solo consideramos los casos donde $s = \mathsf{init}$, ya que $\mathcal{T}_{\mathsf{init}}(s,v) = \varnothing$ de lo contrario. Si $\mu \in \mathcal{T}_{\mathsf{init}}(\mathsf{init},v)$, entonces $\mu = \delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}$. Desde cada uno μ_{x_i} es una medida de probabilidad continua, la probabilidad de dos relojes que se configuran con el mismo valor es 0 y $\mu_{x_i}(\mathbb{R}_{>0}) = 1$. Entonces $\mu(\mathsf{Inv}^c) = 0$. Esto prueba el primer caso.

Para los demás casos introducimos la siguiente notación. Para cada $x_i, x_j \in \mathit{active}(s')$, defina $\mathsf{Inv}^c_{ij} = \{(s'',\vec{w}) \mid \vec{w}(i) = \vec{w}(j)\}$ siempre que $i \neq j$, $\mathsf{Inv}^c_{i,\mathrm{st}} = \{(s'',\vec{w}) \mid \mathrm{st}(s''), \vec{w}(i) < 0\}$, y $\mathsf{Inv}^c_{i,\mathrm{nst}} = \{(s'',\vec{w}) \mid \neg\mathrm{st}(s''), \vec{w}(i) \leq 0\}$. No es difícil probar que cada uno de este tipo de conjuntos es mensurable. Note que $\mathsf{Inv}^c = \bigcup \mathsf{Inv}^c_{ij} \cup \bigcup \mathsf{Inv}^c_{i,\mathrm{st}} \cup \bigcup \mathsf{Inv}^c_{i,\mathrm{nst}}$ y, dado que las uniones son finitas, $\mu(\mathsf{Inv}^c) = 0$ si y sólo si $\mu(\mathsf{Inv}^c_{ij}) = 0$, $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = 0$, y $\mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$, por cada i,j. Por lo tanto, para los dos casos restantes nos centramos en probar estas tres últimas igualdades.

Sean $a \in \mathcal{A}$, $\mu \in \mathcal{T}_a(s,\vec{v})$ y $(s,\vec{v}) \in \mathsf{Inv}$. Entonces $s \neq \mathsf{init}$ y por lo tanto, por Definición 3.2, existe $s \xrightarrow{C,a,C'} s'$ tal que $\bigwedge_{x_i \in C} \vec{v}(i) \leq 0$, y $\mu = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$ con $\overline{\mu}_{x_i} = \mu_{x_i}$ si $x_i \in C$, $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ de lo contrario.

Sea $x_i \in \mathit{active}(s')$, entonces $x_i \in (\mathit{active}(s) \setminus C) \cup C'$. Si $x_i \in C'$, entonces $\mu_{x_i}(\mathbb{R}_{>0}) = 1$ y por lo tanto $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = \mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$. Si $x_i \in (\mathit{active}(s) \setminus C) \setminus$

C' consideramos dos subcasos: ya sea $C = \varnothing$ o $C = \{x_j\}$. En el primer caso, $a \in \mathcal{A}^{\mathsf{u}}$ y por lo tanto s no es estable. Entonces $\vec{v}(i) > 0$ (ya que $(s, \vec{v}) \in \mathsf{Inv}$) y por lo tanto $\delta_{\vec{v}(i)}(\mathbb{R}_{>0}) = 1$, lo que implica $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = \mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$. Si en cambio $C = \{x_j\}$, $i \neq j$ y, por Def. 3.2, $\vec{v}(j) = 0$. Como s es estable y $(s, \vec{v}) \in \mathsf{Inv}$, luego $\vec{v}(i) \geq 0$ y $\vec{v}(i) \neq \vec{v}(j)$, por lo tanto $\vec{v}(i) > 0$ y, como antes, $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = \mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$.

Supongamos ahora $x_i, x_j \in \mathit{active}(s')$ con $i \neq j$, entonces $x_i, x_j \in (\mathit{active}(s) \setminus C) \cup C'$. Si $x_i \in C$ entonces μ_{x_i} es una medida de probabilidad continua y por lo tanto $\mu(\mathsf{Inv}^c_{ij}) = 0$. Del mismo modo, si $x_j \in C$. Si en cambio $x_i, x_j \in \mathit{active}(s) \setminus C$, entonces $\vec{v}(i) \neq \vec{v}(j)$ porque $(s, \vec{v}) \in \mathsf{Inv}$ y por tanto $\delta_{\vec{v}(i)} \neq \delta_{\vec{v}(j)}$. Por lo tanto $\mu(\mathsf{Inv}^c_{ij}) = 0$. Esto prueba que $\mu(\mathsf{Inv}^c) = 0$ para este caso.

Finalmente, tome $d \in \mathbb{R}_{>0}$ y suponga que $\mathcal{T}_d(s, \vec{v}) = \{\mu\}$ con $(s, \vec{v}) \in \mathsf{Inv}$. Por Def. 3.2, s debe ser estable, $0 < d \leq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\}, a, C'} s', a \in \mathcal{A}^{\circ}\}$, y $\mu = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$. Dado que s es estable, $\mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$. Para $x_i \in \mathit{active}(s)$, $\vec{v}(i) - d \geq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\}, a, C'} s', a \in \mathcal{A}^{O}\} - d \geq 0$, ya que $\mathit{active}(s) = \mathit{enabling}(s)$ (s es estable). Por eso $\delta_{\vec{v}(i)-d}(\mathbb{R}_{\geq 0}) = 1$. Por lo tanto $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = 0$. Para $x_i, x_j \in \mathit{active}(s)$ con $i \neq j$, $\vec{v}(i) \neq \vec{v}(j)$ porque $(s, \vec{v}) \in \mathsf{Inv}$. Por eso $\delta_{\vec{v}(i)-d} \neq \delta_{\vec{v}(j)-d}$. Así que $\mu(\mathsf{Inv}^c_{ij}) = 0$. Esto prueba que $\mu(\mathsf{Inv}^c) = 0$ para este caso, y por lo tanto el lema.

□

De este lema se desprende el siguiente corolario

Corolario 4.2. El conjunto Inv^c casi nunca se alcanza en $\mathcal{P}(\mathcal{I})$.

La prueba del corolario requiere definiciones relacionadas con schedulers y medidas sobre caminos en NLMP (ver [102, Cap. 7] para una definición formal de scheduler y medidas de probabilidad sobre caminos en NLMP).

Omitimos la prueba del corolario ya que finalmente se reduce a una aplicación inductiva del Lema 4.2.

El siguiente lema establece que cualquier estado estable en el invariante Inv tiene como máximo una transición discreta habilitada. Su demostración es la misma que la del Lema 3.4.

Lema 4.3. Para todos los $(s, \vec{v}) \in \mathsf{Inv}$ con s estable o $s = \mathsf{init}$, el conjunto $\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ es un conjunto singletón o un conjunto vacío.

Así, sólo nos queda el punto (c) de la Definición 4.8 para probar el Teorema 4.3. El siguiente lema establece que cualquier estado inestable en el invariante Inv solo puede producir acciones urgentes.

Lema 4.4. Para cada estado $(s, \vec{v}) \in \mathsf{Inv}$, si $\neg\mathrm{st}(s)$ y $(s, \vec{v}) \xrightarrow{a} \mu$, luego $a \in \mathcal{A}^{\mathsf{u}}$.

Demostración. Primero, recuerde que $\mathcal{I}$ es cerrado, por lo tanto, $\mathcal{A}^{\mathsf{i}} = \varnothing$. Si $(s, \vec{v}) \in \mathsf{Inv}$ y $\neg\mathrm{st}(s)$ entonces $\vec{v}_i > 0$ para todos los $x_i \in \mathit{enabling}(s) \subseteq \mathit{active}(s)$. Por lo tanto, por Def. 3.2, $\mathcal{T}_a(s, \vec{v}) = \varnothing$ si $a \in \mathcal{A}^{\circ} \setminus \mathcal{A}^{\mathsf{u}}$. Además, para cualquier $d \in \mathbb{R}_{>0}$, $\mathcal{T}_d(s, \vec{v}) = \varnothing$ ya que s no es estable y por lo tanto $s \xrightarrow{\_,\ actb,\_} \_$ por algunas $b \in \mathcal{A}^{\circ} \cup \mathcal{A}^{\mathsf{u}}$. □

Finalmente, Teo. 4.3 es una consecuencia de Lema 4.3, Lema 4.4, Cor. 4.2, y Cor. 4.1.

Prueba del teorema 4.3. Tenemos que demostrar que todo conjunto medible $B \in \mathscr{B}(\mathbf{S})$ de estados que satisfacen las condiciones (a), (b) o (c) en Def. 4.8 casi nunca se alcanza en $\mathcal{P}(\mathcal{I})$. Sea $B_{\mathrm{st}} = B \cap ((\{s \mid \mathrm{st}(s)\} \cup \{\mathsf{init}\}) \times \mathbb{R}^N)$ y $B_{\ negst} = B \cap (\{s \mid \neg\mathrm{st}(s)\} \times \mathbb{R}^N)$. Entonces $B = B_{\mathrm{st}} \cup B_{\neg\mathrm{st}}$, y B_{st} y $B_{\neg\mathrm{st}}$ son medibles. Por lo tanto, B casi nunca se alcanza si y solo si B_{st} y $B_{\neg\mathrm{st}}$ casi nunca se alcanzan.

Sea $\mathsf{En}_{\geq 2} = \{(s, \vec{v}) \in \mathbf{S} \mid (\mathrm{st}(s) \vee s = \mathsf{init}) \wedge |\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})| \geq 2\}$. Por Lema 4.3, $\mathsf{En}_{\geq 2} \subseteq \mathsf{Inv}^c$, y por (a) en Def. 4.8, $B_{\mathrm{st}} \subseteq \mathsf{En}_{\geq 2}$. Entonces, por Cor. 4.2, casi nunca se alcanza B_{st}. Además, Cor. 4.1, asegura que ningún $(s, \vec{v}) \in B_{\neg\mathrm{st}}$ satisface (b). Por lo tanto cada $(s, \vec{v}) \in B_{\neg\mathrm{st}}$ satisface (c). Por lo tanto, por Lema 4.4 $B_{\neg\mathrm{st}} \subseteq \mathsf{Inv}^c$. Entonces, por Cor. 4.2, casi nunca se alcanza $B_{\neg\mathrm{st}}$, lo que prueba el teorema. □

4.6. Condiciones suficientes para determinismo débil

Hasta ahora, insistimos en señalar que estábamos interesados en construir modelos de forma compositiva, de manera que resultaran susceptibles de simulación real, es decir, modelos totalmente estocásticos. Suponiendo que hemos llegado a un modelo cerrado al componer solo componentes confluentes, podemos aplicar el Teorema 4.3 para garantizar que la composición sea débilmente determinista, ya que la composición paralela conserva la confluencia (Prop. 4.2). Sin embargo, tener un componente no confluente no implica que el modelo compuesto también sea no confluente. Por lo tanto, sería interesante encontrar condiciones bajo las cuales los componentes potencialmente no confluentes puedan componerse en un modelo confluente.

Fig. 4.3 muestra un ejemplo en el que el IOSA compuesto es weak determinista a pesar de que algunos de sus componentes no son confluentes. El no

determinismo potencial introducido por el estado $s_2||s_4||s_6$ nunca se alcanza ya que las acciones urgentes en los estados $s_0||s_4||s_6$ y $s_1||s_3||s_6$ impiden la ejecución de una acción no urgente que conduzca a tal estado. Decimos que el estado $s_2||s_4||s_6$ no es *potencialmente alcanzable*. El concepto de potencialmente alcanzable se puede definir de la siguiente manera.

Definición 4.9. Dado un IOSA$_u$ $\mathcal{I}$, un estado s es *potencialmente alcanzable* si hay un camino $s_0 \xrightarrow{\_,a_0,\_} s_1 \ldots, s_{n-1} \xrightarrow{\_,a_{n-1},\_} s_n = s$ desde el estado inicial, con $n \geq 0$, tal que para todo $0 \leq i < n$, si $s_i \xrightarrow{\_,b,\_} \_$ para algunas $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ luego $a_i \in \mathcal{A}^{\mathsf{u}}$. En tal caso llamamos al camino *plausible*.

Observe que ninguno de los caminos que conducen a $s_2||s_4||s_6$ en Fig. 4.3 son plausibles. Además, observe que un IOSA$_u$ es bisimilar al mismo IOSA$_u$ cuando su conjunto de estados está restringido solo a estados potencialmente alcanzables.

Proposición 4.4. Sea $\mathcal{I}$ un IOSA$_u$ cerrado con un conjunto de estados $\mathcal{S}$ y sea $\overline{\mathcal{I}}$ el mismo IOSA$_u$ que $\mathcal{I}$ restringido al conjunto de estados $\overline{\mathcal{S}} = \{s \in \mathcal{S} \mid$ es potencialmente alcanzable en $\mathcal{I}\}$. Entonces $\mathcal{I} \sim \overline{\mathcal{I}}$.

Debe quedar claro que ambas semánticas son bisimilares a través de la relación de identidad ya que un $s \xrightarrow{\{x\},a,C} s'$ con s inestable no introduce ninguna transición concreta. (Recuerde que IOSA$_u$ está cerrado, por lo que no hay acción de entrada en $\mathcal{I}$).

Para que un estado en un IOSA$_u$ compuesto sea potencialmente alcanzable, necesariamente cada uno de los estados componentes tiene que ser potencialmente alcanzable en su respectivo componente IOSA$_u$.

Lema 4.5. Si un estado $s_1|| \cdots ||s_n$ es potencialmente alcanzable en $\mathcal{I}_1|| \cdots ||\mathcal{I}_n$, entonces s_i es potencialmente alcanzable en $\mathcal{I}_i$ para todos los $i = 1, \ldots, N$.

Demostración. Solo lo demostramos para $\mathcal{I}_1||\mathcal{I}_2$. La generalización a cualquier n sigue fácilmente. Lo demostramos por inducción sobre la longitud del camino plausible σ que lleva a $s_1||s_2$. Si $|\sigma| = 0$ entonces $\sigma = s_1^0||s_2^0$, donde cada s_i^0 es inicial en cada $\mathcal{I}_i$ y, por lo tanto, potencialmente accesible. Para el caso inductivo, sea $\sigma = \sigma' \cdot (s_1'||s_2') \xrightarrow{C,a,C'} (s_1||s_2)$. S.p.d.g. y, por contradicción, supongamos que s_1 no es potencialmente alcanzable en $\mathcal{I}_1$. Necesariamente, $s_1 \neq s_1'$ ya que s_1' es potencialmente alcanzable por inducción ($|\sigma| = |\sigma'| + 1$). Así $s_1'||s_2' \xrightarrow{C,a,C'} s_1||s_2$ es el resultado de aplicar (R1) o (R3). El resto de la prueba sigue de manera similar para ambos casos. Supongamos que se aplicó (R3). Entonces $s_1' \xrightarrow{C_1,a,C_1'} s_1$ para algunos $C_1 \subseteq C$ y $C_1' \subseteq C'$. Dado que s_1 no es potencialmente alcanzable pero s_1' sí, entonces $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$

y hay un $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ tal que $s_1'\ trans[\_, b, \_]\_$. Entonces $s_1'||s_2' \xrightarrow{\_,b,\_} \_$, ya sea por (R1) o por (R3) (siendo $\mathcal{I}_2$ entrada habilitada) dando σ no plausible y por lo tanto una contradicción. □

En esta sección, y siguiendo las ideas presentadas en [36], nos basamos en una teoría que nos permite asegurar que un IOSA$_u$ compuesto cerrado es confluente (y por lo tanto débilmente determinista) en una forma composicional, incluso cuando sus componentes pueden no ser confluentes. El teorema 4.5 proporciona tales condiciones suficientes para garantizar que el IOSA$_u$ compuesto sea confluente. Debido a la Proposición 4.2, basta comprobar si se alcanzan potencialmente dos acciones urgentes que no son confluentes en un solo componente. Dado que la alcanzabilidad potencial depende de la composición, la idea es sobreaproximar inspeccionando los componentes para evitar la explosión del espacio de estado. El resto de la sección se basa en conceptos que son esenciales para construir tal sobreaproximación.

Identificamos los conjuntos de acciones urgentes que pueden habilitarse al mismo tiempo. Para hacerlo, observamos los eventos que pueden desencadenar que esta acción se habilite. Hay tres tipos de tales eventos, que ocurren en la habilitación de acciones habilitadas espontáneamente, acciones inicialmente habilitadas y acciones activadas. También definimos formalmente los conjuntos habilitados y mostramos que los eventos propuestos son los únicos capaces de habilitar estas acciones urgentes.

Definamos el conjunto $\text{uen}(s) = \{a \in \mathcal{A}^{\mathsf{u}} \mid s \xrightarrow{\_,a,\_} \_\}$ como el conjunto de acciones urgentes habilitadas en un estado s. Decimos que un conjunto B de acciones urgentes de output se habilita espontáneamente mediante una acción b si se habilita justo después de una transición no urgente etiquetada como b.

Definición 4.10. Un conjunto $B \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ es *habilitado espontáneamente* por $b \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ en $\mathcal{I}$, si $B = \varnothing$ o hay estados potencialmente alcanzables s y s' tales que s es estable, $s \xrightarrow{\_,b,\_} s'$, y $B \subseteq \text{uen}(s')$. B es *maximal* si para cualquier B' habilitado espontáneamente por b en $\mathcal{I}$ tal que $B \subseteq B'$, $B = B'$.

Un conjunto que se habilita espontáneamente en un IOSA$_u$compuesto, puede construirse como la unión de conjuntos habilitados espontáneamente en cada uno de los componentes como lo establece la siguiente proposición. Por lo tanto, los conjuntos habilitados espontáneamente en un IOSA$_u$ compuesto pueden sobreaproximarse mediante uniones de conjuntos habilitados espontáneamente de sus componentes.

Proposición 4.5. Sea B habilitado espontáneamente por a en $\mathcal{I}_1||\ldots||\mathcal{I}_n$. Entonces, hay $B_1, \ldots, B_n$ tales que cada B_i es habilitado espontáneamente

por a en $\mathcal{I}_i$, y $B = \bigcup_{i=1}^{n} B_i$. Si además B es máximo, hay $B_1, \ldots, B_n$ tales que cada B_i es un máximo habilitado espontáneamente por a en $\mathcal{I}_i$, y $B \subseteq \bigcup_{i=1}^{n} B_i$.

Demostración. Solo lo demostramos para $\mathcal{I}_1||\mathcal{I}_2$. La generalización a cualquier n sigue fácilmente. Sea $\bar{B}_i = B \cap \mathcal{A}_i$ para $i = 1, 2$ y tenga en cuenta que $B = \bar{B}_1 \cup \bar{B}_2$. Mostramos que $\bar{B}_1$ es habilitado espontáneamente por a en $\mathcal{I}_1$. El caso de $\bar{B}_2$ sigue de manera similar. Dado que B es habilitado espontáneamente por a en $\mathcal{I}_1||\mathcal{I}_2$, existen estados potencialmente alcanzables $s_1||s_2$ y $s_1'||s_2'$, tales que $s_1||s_2$ es estable, $s_1||s_2 \xrightarrow{-,a,-} s_1'||s_2'$, y $B \subseteq \text{uen}(s_1'||s_2')$. Primero observe que $\bar{B}_1 \subseteq \text{uen}(s_1)$. Además, supongamos que $\bar{B}_1 \neq \varnothing$, de lo contrario, $\bar{B}_1$ se habilita espontáneamente mediante a de manera trivial. Considere primero el caso de que $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$. Por (R2), $s_1 = s_1'$, pero, como hay algo de $b \in \bar{B}_1$, $s_1 \xrightarrow{-,b,-} \_$ y por lo tanto $s_1||s_2 \xrightarrow{-,b,-} \_$ haciendo que $s_1||s_2$ sea inestable, lo cual es una contradicción. Entonces $a \in \mathcal{A}_1$ y $s_1 \xrightarrow{-,a,-} s_1'$. Por Lemma 4.5, s_1 y s_1' son potencialmente alcanzables y, necesariamente, s_1 es estable (de lo contrario, $s_1||s_2$ tiene que ser inestable como se muestra antes). Por lo tanto $\bar{B}_1$ es habilitado espontáneamente por a en $\mathcal{I}_1$. La segunda parte de la proposición es inmediata a la primera. □

La proposición 4.5 nos permite sobreaproximar el conjunto de acciones habilitantes espontáneas en el IOSA$_u$ compuesto:

$$\{B \mid B \text{ spontaneously enabled by } a \text{ in } \mathcal{I}_1||\ldots||\mathcal{I}_n\} \subseteq \left\{\bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n \cdot B_i \text{ spontaneously enabled by } a \text{ in } \mathcal{I}_i\right\} \tag{4.3}$$

Los conjuntos habilitados espontáneamente se refieren a conjuntos de acciones de salida urgentes que se habilitan después de algunos pasos de ejecución. Las acciones de salida urgentes también se pueden habilitar en el estado inicial.

Definición 4.11. Un conjunto $B \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ es *inicial* en un IOSA$_u$ $\mathcal{I}$ si $B \subseteq \text{uen}(s_0)$, con s_0 el estado inicial de $\mathcal{I}$. B es *maximal* si $B = \text{uen}(s_0) \cap \mathcal{A}^{\circ}$.

Un conjunto inicial de un IOSA$_u$ compuesto se puede construir como la unión de conjuntos iniciales de sus componentes. En particular el conjunto inicial maximal es la unión de todos los conjuntos maximales de sus componentes. La prueba se deriva directamente de la definición de composición paralela teniendo en cuenta que IOSA$_u$s están habilitados para la entrada.

Proposición 4.6. Sea B inicial en $\mathcal{I} = (\mathcal{I}_1 || \ldots || \mathcal{I}_n)$. Entonces, hay $B_1, \ldots, B_2$, con B_i inicial en $\mathcal{I}_i$, $1 \leq i \leq n$ y $B = \bigcup_{i=1}^{n} B_i$. Mas aún, $\text{uen}(s_0) \cap \mathcal{A}^{\circ}_{\mathcal{I}} = \bigcup_{i=1}^{n} \text{uen}(s_i^0) \cap \mathcal{A}^{\circ}_i$.

La proposición 4.6 nos permite identificar los conjuntos de acciones inicialmente habilitadas en el IOSA$_u$ compuesto considerando el conjunto de acciones inicialmente habilitadas en sus componentes:

$$\{B \mid B \text{ initial in } \mathcal{I}_1 || \ldots || \mathcal{I}_n\} = \left\{ \bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n \cdot B_i \text{ initial in } \mathcal{I}_i \right\} \tag{4.4}$$

Lo mismo puede deducirse para el conjunto máximo de acciones iniciales:

$$\{B \mid B \text{ maximally initial in } \mathcal{I}_1 || \ldots || \mathcal{I}_n\} = \left\{ \bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n \cdot B_i \text{ maximally initial in } \mathcal{I}_i \right\}. \tag{4.5}$$

Decimos que una acción urgente desencadena (triggers) una acción de salida urgente si la primera habilita la ocurrencia de la segunda que antes no estaba habilitada. Con esto, identificamos no solo las acciones de salida urgentes que se habilitan después de una transición no urgente (conjuntos espontáneos), sino también aquellas que se habilitan después de una transición urgente.

Definición 4.12. Sea $a \in \mathcal{A}^{\mathsf{u}}$ y $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$. *a triggers b* en un IOSA$_u$ $\mathcal{I}$ si hay estados potencialmente alcanzables s_1, s_2, y s_3 tal que $s_1 \xrightarrow{\_,a,\_} s_2 \xrightarrow{\_,b,\_} s_3$ y, si $a \neq b$, $b \notin \text{uen}(s_1)$.

Nótese que, para el caso particular en el que $a = b$, no se requiere $b \notin \text{uen}(s_1)$, ya que sería una contradicción. La siguiente proposición establece que si una acción desencadena otra en un IOSA$_u$compuesto, entonces ocurre la misma activación en un componente particular.

Proposición 4.7. Sea $a \in \mathcal{A}^{\mathsf{u}}$ y $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ tal que a triggers b en $\mathcal{I}_1 || \ldots || \mathcal{I}_n$. Entonces hay una componente $\mathcal{I}_i$ tal que $b \in \mathcal{A}^{\circ}_i$ y a triggers b en $\mathcal{I}_i$.

Demostración. Solo lo demostramos para $\mathcal{I}_1 || \mathcal{I}_2$. La generalización a cualquier n sigue fácilmente. Dado que $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ necesariamente $b \in \mathcal{A}^{\circ}_1$ or $b \in \mathcal{A}^{\circ}_2$. S.p.d.g. supongamos $b \in \mathcal{A}^{\circ}_1$. Dado que a triggers b en $\mathcal{I}_1 || \mathcal{I}_1$,

$s_1||s_2 \xrightarrow{-,a,-} s_1'||s_2' \xrightarrow{-,b,-} s_1''||s_2''$ con $s_1||s_2$, $s_1'||s_2'$, y $s_1''||s_2''$ siendo potencialmente alcanzable. Supongamos primero que $a \neq b$. Entonces $b \notin \text{uen}(s_1||s_2)$. Recuerde que, por Lemma 4.5, s_1, s_1' y s_1'' son potencialmente accesibles en $\mathcal{I}_1$. Desde $b \in \mathcal{A}_1^\circ$, $s_1' \xrightarrow{-,b,-} s_1''$. Supongamos que $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$. Entonces, necesariamente, $s_1 = s_1'$ que da $b \in \text{uen}(s_1) \cap \mathcal{A}^\circ \subseteq \text{uen}(s_1||s_2)$, produciendo una contradicción. Así, necesariamente $a \in \mathcal{A}_1^{\mathsf{u}}$ y por lo tanto $s_1 \xrightarrow{-,a,-} s_1'$, por la definición de composición paralela. Queda por demostrar que $b \notin \text{uen}(s_1)$, pero esto es inmediato ya que $\text{uen}(s_1) \cap \mathcal{A}^\circ \subseteq \text{uen}(s_1||s_2)$ y $b \notin \text{uen}(s_1||s_2)$. Por lo tanto, a activa b en $\mathcal{I}_1$ en este caso. Si en cambio $a = b$, por la definición de composición paralela tenemos inmediatamente que $s_1 \xrightarrow{-,b,-} s_1' \xrightarrow{-,b,-} s_1''$, probando así la proposición.

□

Proposition 4.7 tells us that the triggering relation of a composed IOSA$_u$ can be overapproximated by the union of the triggering relations of its components. Thus we define:

La proposición 4.7 nos dice que la relación de triggering de un IOSA$_u$ compuesto puede sobreaproximarse mediante la unión de las relaciones de triggering de sus componentes. Así definimos:

Definición 4.13. La *approximate triggering relation* de $\mathcal{I}_1||\ldots||\mathcal{I}_n$ se define por $\rightsquigarrow = \bigcup_{i=1}^n \{(a,b) \mid a \text{ triggers } b \text{ in } \mathcal{I}_i\}$. Su clausura reflexo transitiva $\rightsquigarrow^*$ se llama *approximate indirect triggering relation.*

Hemos pasado por las tres formas de habilitar acciones de salida urgentes. Estos son por conjuntos espontáneos, por conjuntos iniciales, por conjuntos desencadenados. Ahora definiremos formalmente los conjuntos habilitados y probaremos que estas son, de hecho, las únicas tres formas de generarlos. La siguiente definición caracteriza todos los conjuntos de acciones de salida urgentes que se habilitan simultáneamente en cualquier estado potencialmente alcanzable de un IOSA$_u$ dado.

Definición 4.14. A set $B \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^\circ$ is an *enabled set* in an IOSA$_u$ $\mathcal{I}$ if there is a potentially reachable state s such that $B \subseteq \text{uen}(s)$.If $a \in B$, we say that a is *enabled* in s. Let $\mathsf{ES}_\mathcal{I}$ be the set of all enable sets in $\mathcal{I}$.

If an urgent output action is enabled in a potentially reachable state of IOSA$_u$, then it is is either initial, spontaneously enabled, or triggered by some action, as proved by the following Theorem.

Teorema 4.4. Sea $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^\circ$ habilitado en algún estado potencialmente alcanzable del IOSA$_u$ $\mathcal{I}$. Entonces existe un conjunto B con $b \in B$ que es o bien incial, o bien espontáneamente habilitado por alguna acción $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$, o b o es desencadenado por alguna acción $a \in \mathcal{A}^{\mathsf{u}}$.

Demostración. Sea s potencialmente alcanzable en $\mathcal{I}$ tal que $b \in \text{uen}(s) \cap \mathcal{A}^{\circ}$. Demostramos el teorema para b por inducción en el camino plausible σ que conduce a s. Si $|\sigma| = 0$, entonces $\sigma = s$ y s es el estado inicial. Entonces el conjunto $\text{uen}(s) \cap \mathcal{A}^{\circ}$ es inicial y hemos terminado en este caso. Si $|\sigma| > 0$, entonces $\sigma = \sigma' \cdot (s' \xrightarrow{-,a,-} s)$ para algunos s', a y plausible σ'. Si $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ entonces s' es estable (ya que σ es plausible) y, por lo tanto, $\text{uen}(s) \cap \mathcal{A}^{\circ}$ se habilita espontáneamente por a. Si en cambio $a \in \mathcal{A}^{\mathsf{u}}$, tenemos dos posibilidades. Si $b \notin \text{uen}(s')$, entonces b es activado por a. Si $b \in \text{uen}(s')$, las condiciones se cumplen por inducción ya que $|\sigma'| = |\sigma| - 1$. □

Ahora presentamos una forma de construir los conjuntos habilitados de un IOSA$_u$ cerrado analizando sus componentes. De hecho, sobreaproximamos estos conjuntos mirando sus componentes en lugar de mirar el modelo compuesto, evitando así el problema de la explosión de estado. Para esto hacemos uso de la Definición 4.13, junto con la sobreaproximación estudiada en las Proposiciones 4.5 y 4.6. La siguiente definición es auxiliar para probar el teorema principal de esta sección. Construye un grafo a partir de un IOSA$_u$ cerrado y compuesto cuyos vértices son conjuntos de acciones de salida urgentes. Tiene la propiedad de que, si hay un camino de un vértice a otro, todas las acciones en el segundo vértice son provocadas indirectamente por acciones en el primer vértice (Lemma 4.7). Esto permite mostrar que cualquier conjunto de acciones de salida urgentes habilitadas simultáneamente se desencadena de manera indirecta por acciones iniciales o un conjunto habilitado espontáneamente (Lemma 4.8).

Definición 4.15. Sea $\mathcal{I} = (\mathcal{I}_1 || \ldots || \mathcal{I}_n)$ una IOSAcerrada. El *grafo habilitado* de $\mathcal{I}$ está definido por el grafo etiquetado $\mathsf{EG}_{\mathcal{I}} = (V, E)$, donde $V \subseteq 2^{\mathcal{A}^{\circ} \cap \mathcal{A}^{\mathsf{u}}}$ y $E \subseteq V \times (\mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}) \times V$, con $V = \bigcup_{k \geq 0} V_k$ y $E = \bigcup_{k \geq 0} E_k$, y, para todo $k \in \mathbb{N}$, V_k y E_k se definen inductivamente por

$$
\begin{aligned}
V_0 = {} & \textstyle\bigcup_{a \in \mathcal{A}} \{ \bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n : \\
& \qquad\qquad B_i \textit{ is spontaneously enabled by } a \textit{ and maximal in } \mathcal{I}_i \} \\
& \cup \{ \textstyle\bigcup_{i=1}^{n} \text{uen}(s_i^0) \cap \mathcal{A}_i^{\circ} \mid \forall 1 \leq i \leq n : s_i^0 \textit{ is the initial state in } \mathcal{I}_i \} \\
E_k = {} & \{ (v, a, (v \backslash \{a\}) \cup \{b \mid a \rightsquigarrow b\}) \mid v \in V_k, a \in v \} \\
V_{k+1} = {} & \{ v' \mid v \in V_i, (v, v') \in E_k, v' \notin \textstyle\bigcup_{j=0}^{k} V_j \}
\end{aligned}
$$

Observe que V_0 contiene el conjunto inicial máximo de $\mathcal{I}$ y una sobreaproximación de todos sus conjuntos habilitados espontáneamente máximos. Observe también que, por construcción, existe un camino desde cualquier vértice en V hasta algún vértice en V_0.

El conjunto clausura de V en $\mathsf{EG}_{\mathcal{I}}$, definido por

$$\overline{\mathsf{ES}}_{\mathcal{I}} = \{B \mid B \subseteq v, v \in V\}$$

resulta ser una sobreaproximación del conjunto real $\mathsf{ES}_{\mathcal{I}}$ de todos los conjuntos habilitados en $\mathcal{I}$, como lo demuestra el siguiente lema.

Lema 4.6. Para cualquier IOSA cerrado $\mathcal{I} = (\mathcal{I}_1 || \cdots || \mathcal{I}_n)$, $\mathsf{ES}_{\mathcal{I}} \subseteq \overline{\mathsf{ES}}_{\mathcal{I}}$.

Demostración. Sea $B \in \mathsf{ES}_{\mathcal{I}}$. Procedemos por inducción sobre la longitud del camino plausible σ que conduce al estado s tal que $B \subseteq \text{uen}(s)$. Si $|\sigma| = 0$ entonces s es el estado inicial y por lo tanto B es inicial en $\mathcal{I}$. Así, por Definición 4.11, Prop. 4.6, y Definición 4.15, $B \subseteq (\text{uen}(s_0) \cap \mathcal{A}^{\circ}_{\mathcal{I}}) = (\bigcup_{i=1}^{n} \text{uen}(s_i^0) \cap \mathcal{A}_i^{\circ}) \in V_0 \subseteq \overline{\mathsf{ES}}_{\mathcal{I}}$. Como consecuencia $B \in \overline{\mathsf{ES}}_{\mathcal{I}}$.

Si $|\sigma| > 0$ entonces $\sigma = \sigma' \cdot (s' \xrightarrow{-,a,-} s)$, para algunos s', a, y plausible σ'. Si $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ entonces s' es estable (ya que σ es plausible) y, por lo tanto, B se habilita espontáneamente por a. Por Prop. 4.5, hay $B_1, \ldots, B_n$ tales que cada B_i es habilitado espontáneamente por a y máximo en $\mathcal{I}_i$, y $B \subseteq \bigcup_{i=1}^{n} B_i$. Dado que $\bigcup_{i=1}^{n} B_i \in V_0 \subseteq \overline{\mathsf{ES}}_{\mathcal{I}}$, entonces $B \in \overline{\mathsf{ES}}_{\mathcal{I}}$. Si en cambio $a \in \mathcal{A}^{\mathsf{u}}$, sea $B' = \{a\} \cup (B \cap \text{uen}(s'))$. Observe que $B' \subseteq \text{uen}(s') \cap \mathcal{A}^{\circ}$. Dado que s' es el último estado en σ' y $|\sigma'| = |\sigma| - 1$, $B' \in \overline{\mathsf{ES}}_{\mathcal{I}}$ por inducción. Por tanto, existe un vértice $v' \in V$ en $\mathsf{EG}_{\mathcal{I}}$ tal que $B' \subseteq v$ y, por Def 4.15, $v' \in V_k$ para unos $k \geq 0$. Sea $v = (v' \backslash \{a\}) \cup \{b \mid a \rightsquigarrow b\}$, entonces $(v', a, v) \in E_k$ y por lo tanto $v \in V_{k+1}$. Mostramos que $B \subseteq v$. Sea $b \in B$. Si $b = a$, entonces $a \in \text{uen}(s) \cap \mathcal{A}^{\circ}$ y, por lo tanto, a activa a en $\mathcal{I}$. Por Prop. 4.7, $a \rightsquigarrow a$ lo que implica $a \in v$. Supongamos, en cambio, que $b \neq a$. Si $b \in \text{uen}(s')$, entonces $b \in B' \backslash \{a\} \subseteq v' \backslash \{a\} \subseteq v$. Si $b \notin \text{uen}(s')$, entonces a activa b en $\mathcal{I}$, y por Prop. 4.7, $a \rightsquigarrow b$ que implica $b \in v$. Esto prueba $B \subseteq v \in \overline{\mathsf{ES}}_{\mathcal{I}}$ y por lo tanto $B \in \overline{\mathsf{ES}}_{\mathcal{I}}$. □

El siguiente lema establece que si hay un camino desde un vértice de $\mathsf{EG}_{\mathcal{I}}$ a otro vértice, cada acción en el segundo vértice se desencadena de manera indirecta por alguna acción en el primer vértice.

Lema 4.7. Sea $\mathcal{I}$ un IOSAcerrado, sean $v, v' \in V$ vértices de $\mathsf{EG}_{\mathcal{I}}$ y sea ρ un camino que sigue a E de v a v'. Entonces por cada $b \in v'$ hay una acción $a \in v$ tal que $a \rightsquigarrow^* b$.

Demostración. Procedemos por inducción en la longitud de ρ. Si $|\rho| = 0$ entonces $v = v'$ y el lema se cumple ya que $\rightsquigarrow^*$ es reflexivo. Si $|\rho| > 0$, hay una ruta ρ', $v'' \in V$ y $c \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ tal que $\rho = \rho' \cdot (v'', c, v')$. Por inducción, para cada acción $d \in v''$ hay algo de $a \in v$ tal que $a \rightsquigarrow^* d$. Debido a la definición de E en Definición 4.15, ya sea $b \in v''$ o $c \rightsquigarrow b$ y $c \in v''$. El primer caso se sigue por inducción. En el segundo caso, también por inducción, $a \rightsquigarrow^* c$ para algún $a \in v$ y por lo tanto $a \rightsquigarrow^* b$. □

Dado que el vértice inicial del grafo habilitado de un IOSA$_u$ consiste en conjuntos iniciales y conjuntos espontáneamente habilitados, y dado que por construcción cada vértice del grafo tiene un camino hacia el vértice inicial, el último lema nos permite deducir el siguiente . El siguiente lema establece que cada conjunto habilitado B en un IOSA$_u$ compuesto se desencadena aproximadamente por un conjunto de acciones iniciales de los componentes del IOSA$_u$ o por un subconjunto de la unión de conjuntos habilitados espontáneamente en cada componente donde tales conjuntos son habilitados espontáneamente por el mismo evento.

Lema 4.8. Sea $\mathcal{I} = (\mathcal{I}_1 || \ldots || \mathcal{I}_n)$ un IOSA$_u$ cerrado y sean $\{b_1, \ldots, b_m\} \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ conjuntos habilitados en $\mathcal{I}$. Entonces, hay (no necesariamente diferentes) $a_1, \ldots, a_m$ tales que $a_j \rightsquigarrow^* b_j$, para todo $1 \leq j \leq m$, y (I) $\{a_1, \ldots, a_m\} \subseteq \bigcup_{i=1}^{n} \mathrm{uen}(s_i^0) \cap \mathcal{A}_i^{\circ}$, o (II) existe $e \in$ *actions* y conjuntos (posiblemente vacíos) $B_1, \ldots, B_n$ activados espontáneamente por e en $\mathcal{I}_1, \ldots, \mathcal{I}_n$ respectivamente, de modo que $\{a_1, \ldots, a_m\} \subseteq \bigcup_{i=1}^{n} B_i$.

Demostración. Debido al Lema 4.6 hay un vértice v de $\mathsf{EG}_{\mathcal{I}}$ tal que $\{b_1, \ldots, b_n\} \subseteq v$. Debido a la construcción inductiva de E y V, existe un camino desde algún $v' \in V_0$ hasta v en $\mathsf{EG}_{\mathcal{I}}$. De Lemma 4.7, para cada $1 \leq j \leq m$, hay un $a_j \in v'$ tal que $a_j \rightsquigarrow^* b_j$. Porque $v' \in V_0$, entonces $v' = \bigcup_{i=1}^{n} \mathrm{uen}(s_i^0) \cap \mathcal{A}_i^{\circ}$ o hay algunas $e \in \mathcal{A}$ como que $v' = \bigcup_{i=1}^{n} B_i$ con B_i habilitado espontáneamente por e en $\mathcal{I}_i$ □

La proposición 4.2 y el lema 4.8 nos dan los ingredientes para el teorema principal de esta sección que proporciona condiciones suficientes para garantizar que un IOSA$_u$ compuesto cerrado es confluente o, como establecidas en el teorema, condiciones necesarias para que la IOSA sea no confluente.

Teorema 4.5. Sea $\mathcal{I} = (\mathcal{I}_1 || \cdots || \mathcal{I}_n)$ un IOSAcerrado. Si $\mathcal{I}$ alcanza potencialmente un estado no confluente, entonces hay acciones $a, b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ tales que algunos $\mathcal{I}_i$ no son confluentes w.r.t. a y b, y hay son c y d tales que $c \rightsquigarrow^* a$, $d \rightsquigarrow^* b$, y

(I) c y d son acciones iniciales en cualquier componente, o

(II) hay $e \in \mathcal{A}$ y (posiblemente vacío) conjuntos $B_1, \ldots, B_n$ habilitados espontáneamente por e en $\mathcal{I}_1, \ldots, \mathcal{I}_n$ respectivamente, tal que $c, d \in \bigcup_{i=1}^{n} B_i$.

Demostración. Supongamos que $\mathcal{I}$ alcanza potencialmente un estado no confluente s. Entonces necesariamente hay $a, b \in \mathrm{uen}(s)$ que lo muestran y por lo tanto $\mathcal{I}$ no es confluente w.r.t. a y b. Por Prop. 4.2, necesariamente hay un componente $\mathcal{I}_i$ que no es confluente w.r.t. a y b. Dado que $\{a, b\}$ es un conjunto habilitado en $\mathcal{I}$, el resto de la prueba sigue el Lema 4.8. □

Debido a la Prop. 4.4 y al Teorema 4.3, si todos los estados potencialmente alcanzables en un IOSA $\mathcal{I}$ cerrado son confluentes, entonces $\mathcal{I}$ es débilmente determinista. Por lo tanto, si no se encuentran en $\mathcal{I}$ ningún par de acciones que satisfagan las condiciones del Teorema 4.5, entonces $\mathcal{I}$ es débilmente determinista.

Note que el IOSA $\mathcal{I} = \mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$ de Fig 4.3 (ver también Fig. 4.2) es un ejemplo que no no cumple las condiciones del Teorema 4.5, y por lo tanto se detecta como confluente. Las únicas acciones no confluentes potenciales son c y d que no son confluentes en el estado s_6 de $\mathcal{I}_3$. La relación de activación indirecta aproximada se puede calcular en $\rightsquigarrow^* = \{(c,c),(d,d)\}$. Además, $\{c\}$ se activa espontáneamente con a en $\mathcal{I}_1$ y $\{d\}$ se activa espontáneamente con b en $\mathcal{I}_2$. Dado que ambos conjuntos se habilitan espontáneamente por *diferentes* acciones y c y d no son iniciales, el conjunto $\{c,d\}$ no aparece en V_0 de $\mathsf{EG}_{\mathcal{I}}$ que sería necesario para cumplir las condiciones del teorema.

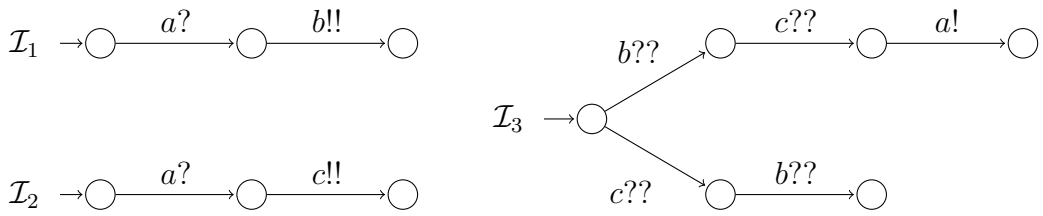


Figura 4.8: $\mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$ cumple condiciones en Teorema 4.5

Por otro lado, dado que las condiciones en el Teorema 4.5 no son suficientes, también IOSAs confluentes pueden satisfacerlas. Considere los IOSAs en la Fig. 4.8. $\mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$ es una IOSA cerrado con un solo estado y sin transición saliente. Por lo tanto, es confluente. Sin embargo, $\mathcal{I}_3$ no es confluente w.r.t. b y c, $\rightsquigarrow^* = \{(b,b),(c,c)\}$, $B_1 = \{b\}$ se habilita espontáneamente por a en $\mathcal{I}_1$, y $B_2 = \{c\}$ se habilita espontáneamente por a en $\mathcal{I}_2$. Por lo tanto $b, c \in \bigcup_{i=1}^{n} B_i$, cumpliendo así las condiciones del Teorema 4.5.

Evitar la explosión del espacio estatal tiene sus beneficios. De hecho, existe un algoritmo para verificar las condiciones del teorema 4.5 en tiempo y espacio polinomial. El algoritmo 1 es una revisión del de I/O-IMC en [36, Chapter 8.6.1]. Para un modelo cerrado dado $\mathcal{I} = \mathcal{I}_1|| \ldots ||\mathcal{I}_n$, verifica si satisface las condiciones del Teorema 4.5. Lo hace en tiempo y espacio polinomial. Para cada componente calcula los conjuntos de todas las acciones iniciales $A^{(init)}$, de tamaño $\mathcal{O}(|\mathcal{A}|)$, y el conjunto de acciones no confluentes que tiene tamaño $\mathcal{O}(|\mathcal{A}|^2)$. Calcular todos los conjuntos espontáneos requeriría un tamaño exponencial para mantener. Ya que solo estamos interesados en saber si dos acciones están en el mismo conjunto espontáneo, calculamos

la relación

$$R^{sp} = \{(a, b) \mid \text{a y b pertenecen a un mismo conjunto espontáneo}\}$$

Note que R^{sp} tiene tamaño $\mathcal{O}(|\mathcal{A}|^2)$.

En términos generales, para calcular R^{sp} debemos calcular n profundidad primero buscar cada par de acciones $(e, a) \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}} \times \mathcal{A}^{\mathsf{u}}$, para marcar cada acción b que comparte con a un mismo conjunto espontáneo habilitado por e. Esto llevará $\mathcal{O}(|\mathcal{A}|^2 \cdot \sum_{i=1}^{n} |S_i|^2)$ tiempo. El cálculo de las acciones iniciales, la relación de activación y los pares de acciones no confluentes se puede realizar aplicando la búsqueda en profundidad a cada componente, lo que requiere $\mathcal{O}(\sum_{i=1}^{n} |\mathcal{S}_i|^2)$ time . La relación de activación aproximada para $\mathcal{I}$ es la unión de las relaciones de activación de sus componentes y nuevamente tiene un tamaño de $\mathcal{O}(|\mathcal{A}|^2)$. Calcular el cierre reflexivo y transitivo de esta relación tiene una complejidad de tiempo $\mathcal{O}(|\mathcal{A}|^3)$ [35].

Después de construir estos conjuntos queda buscar acciones no confluentes a y b, en un conjunto de tamaño $\mathcal{O}(|\mathcal{A}|^2)$. Luego debemos buscar un par de acciones c y d, que sean iniciales o estén contenidas en un conjunto espontáneo, también $\mathcal{O}(|\mathcal{A}|^2)$, que desencadenen aproximadamente indirectamente a y b. Esto da una complejidad de tiempo de $\mathcal{O}(|\mathcal{A}|^4)$ para verificar las condiciones en el bucle 6-13 del algoritmo. Entonces, la complejidad temporal total es

$$\mathcal{O}(|A|^2 \cdot \sum_{i=1}^{n} |S_i|^2 + |A|^4),$$

mientras que la complejidad espacial es

$$\mathcal{O}(\sum_{i=1}^{n} |S_i|^2 + |A|^2).$$

4.7. Conclusiones

En este capítulo hemos introducido una nueva versión de Input/Output Stochastic Automata, que amplía los IOSAs anteriores con la posibilidad de modelar transiciones instantáneas. Llamamos a estas transiciones *urgentes*. A diferencia de las transiciones no urgentes, carecen de un reloj de habilitación y, por lo tanto, su ocurrencia se da inmediatamente, tan pronto como se alcanza el estado de habilitación. Esto da como resultado un marco mucho más flexible para modelar la composicionalidad en comparación con los IOSA originales donde muchas veces experimentamos la imposibilidad de modularizar

Algorithm 1 Verifica si un IOSA$_u$ cerrado $\mathcal{I} = (\mathcal{I}_1 || \ldots || \mathcal{I}_n)$ satisface las condiciones del Teorema 4.5. Si el algoritmo devuelve "True" entonces C puede ser no-determinista, de lo contrario $\mathcal{I}$ es determinista.

```
1: Computar R^sp.
2: for 1 ≤ i ≤ n do
3:     Computar A_i^init, y la triggering relation de I_i.
4:     Computar todos los pares de acciones no confluentes para I_i.
5: Computar la approximate triggering relation para C.
6: Computar la clausura reflexo-transitiva de la approximate triggering relation.
7: for todos los pares a, b de acciones no confluentes do
8:     for all acciones iniciales c que aproximadamente indirectamente habilitan a a do
9:         for all acciones iniciales d do
10:            if d habilita indirectamente a b then
11:                return True
12:    for all acciones espontáneas c que aproximadamente indirectamente habilitan a a do
13:        for all acciones d en el mismo conjunto espontáneo que c do
14:            if d habilita indirectamente a b then
15:                return True
16: return False
```

un modelo y en su lugar necesitábamos construir un gran modelo monolítico, haciendo un uso muy pobre de la composicionalidad.

Mostramos cómo la introducción de la urgencia hizo que nuestros modelos fueran no deterministas incluso en su comportamiento cerrado. Sin embargo, apuntábamos que muchas veces este indeterminismo es introducido por acciones confluentes. Esto resulta ser un no determinismo espurio en el sentido de que no cambia el comportamiento estocástico del modelo. Luego definimos una noción de determinismo débil basada en la confluencia entre acciones urgentes y demostramos que los modelos confluentes (aquellos en los que todas las acciones urgentes son confluentes) son débilmente deterministas. Mostramos que si los componentes son confluentes entonces la composición es confluente.

Tomando como base [36], obtuvimos condiciones suficientes para garantizar que una red de componentes IOSA$_u$ posiblemente no confluentes sea, sin embargo, confluente. Además, propusimos una metodología que descubre las condiciones desencadenantes de acciones urgentes, que es capaz de determinar si una red de IOSA$_u$s cumple con estas condiciones de confluen-

cia. Finalmente desarrollamos un algoritmo que sigue esta metodología para determinar si se cumplen las condiciones. El algoritmo se ejecuta en tiempo polinomial, trabajando directamente sobre los componentes. Observe que si la composición es confluente, entonces estamos seguros de que es débilmente determinista. Observe también que el algoritmo puede devolver *falsos negativos*, dado que las condiciones son suficientes pero no necesarias.

El formalismo IOSA_u entrega un marco para modelar y analizar sistemas estocásticos con distribuciones generales por medio de simulación de eventos raros. Su naturaleza compositiva, permite modelar sistemas de tamaño industrial de una manera cómoda y robusta al concentrarse en el comportamiento claro de cada componente y la sincronización intuitiva entre ellos. Esto contrasta con los enfoques monolíticos en los que el tamaño y la complejidad de los modelos convierten rápidamente el trabajo de ingeniería en una actividad propensa a errores. Los modelos con comportamiento estocástico continuo general no pueden ser tratados mediante model checking en el caso general. La simulación de eventos discretos es la principal alternativa para analizar este tipo de modelos. Probamos que IOSA_u es débilmente determinista y, por lo tanto, susceptible de simulación de eventos discretos.

Capítulo 5

Repairable Fault Trees

El análisis de árbol de fallas (FTA, por Fault Tree Analysis) es una destacada técnica para analizar las propiedades de confiabilidad y seguridad en sistemas industriales. Las características atractivas de FTA se basan en su notación gráfica aparentemente fácil de entender. Fue introducido por primera vez por Bell Laboratories en 1962, como un medio para analizar un sistema balístico para el ejército de EE.UU. Desde entonces, se ha utilizado en muchos campos, especialmente donde el análisis de riesgos es muy importante, como en la aviación y el espacio (NASA, SPACE-X, Boeing, etc.), la industria automotriz, farmacéutica y otras industrias de alto riesgo [49, 73, 97]. También es una técnica prometedora para la evaluación de problemas de disponibilidad y ganancias en la industria ferroviaria (con árboles de fallas de mantenimiento) [90], para la evaluación de seguridad en ciberseguridad (con árboles de ataque) y otros. El análisis de árbol de fallas es una técnica top-down, donde partiendo de un top-event (un fallo que interesa analizar en el sistema), vamos bajando conectando las posibles causas de dicho fallo mediante compuertas lógicas hasta llegar a a fallas básicas (aquellas que no podemos, o no queremos descomponer más) conocidas como eventos básicos (BE). Estos eventos básicos tienen una tasa de falla conocida, generalmente descrita por una distribución de probabilidad. El cálculo de la probabilidad de falla del evento superior depende de la relación de causalidad existente entre la falla superior y los eventos básicos, que viene dada por las compuertas del árbol.

Desde sus orígenes, los árboles de fallas se han extendido de muchas maneras diferentes y se han utilizado para muchos propósitos diferentes. Informalmente podemos decir que los árboles de fallas originales, llamados *Standard o Static Fault Tress* (SFT) [59], son DAGs cuyas hojas se denominan Basic Events (BE), aunque en este trabajo los llamaremos convenientemente *Elementos básicos*. Los BE generalmente representan la falla de un componente

del sistema atómico. Cada hoja está equipada con una tasa de falla, que indica la frecuencia con la que se rompe el componente, generalmente descrita por una distribución exponencial (sin memoria). El resto de los nodos en el DAG se denominan compuertas y modelan cómo las fallas de eventos básicos se combinan para inducir fallas más complejas del sistema. Las compuertas estándar AND, OR y Voting representan combinadores lógicos simples. Otras variedades de FT permiten capacidades de modelado adicionales, por ejemplo, introduciendo nuevas compuertas o un comportamiento más complejo en los BE.

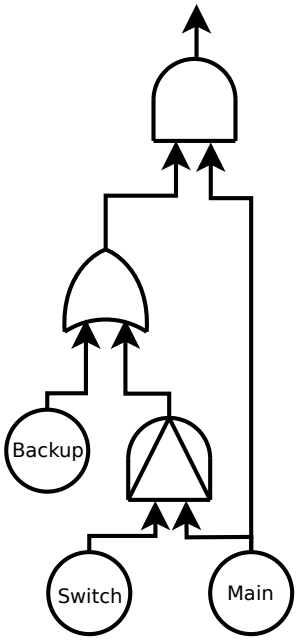


Figura 5.1: Energy backup.

Una de las extensiones más comunes son *Dynamic Fault Trees* (DFT) [51, 69], que introducen compuertas AND de prioridad para modelar el orden en la ocurrencia de fallas, compuertas de repuesto para administrar piezas de repuesto para componentes rotos y compuertas de dependencia funcional que modelan la dependencia funcional entre los BE. Un ejemplo de uso típico para estas nuevas compuertas se puede encontrar en la Figura 5.1, donde modelamos un sistema de respaldo de electricidad compuesto por un interruptor que detecta la energía de la línea principal y enciende el respaldo de energía, y una fuente de alimentación principal . Nos interesa medir la probabilidad de quedarse sin energía (evento superior). Esta falla ocurrirá si el interruptor se rompe y luego la línea de alimentación principal deja de suministrar

electricidad, o si tanto la principal como la de respaldo se rompen. En el primer caso, el acondicionamiento de tiempo entre el interruptor y la alimentación principal lo establece la compuerta PAND. Tenga en cuenta que, en un ejemplo del mundo real, estos BE generalmente se descompondrían en subárboles más complejos, que servirían para analizar en profundidad las causas del evento principal y, por lo tanto, brindar sugerencias sobre dónde buscar una mejora en la confiabilidad de el sistema modelado.

Árboles de fallas reparables (RFT) [85, 8] aumentan la expresividad de FTs al introducir la posibilidad de representar eventos de reparación además de eventos de falla. Se han propuesto varias formas de introducir esta capacidad de modelado. Nos centraremos en el modelo Repair Box (RBOX) [12, 85]. Una compuerta RBOX modela una unidad de reparación encargada de reparar un determinado conjunto de elementos básicos cuando fallan, siguiendo una determinada política de reparación. Las diferentes políticas de reparación, como el orden de llegada, el servicio prioritario, la elección aleatoria o no determinista de las cajas de reparación, permiten a los usuarios analizar el impacto de tomar estas decisiones en el sistema real. La introducción de estas cajas cambia mucho la dinámica del árbol, ya que el cálculo de la probabilidad de falla ya no es un cálculo ascendente. Además, permite estudiar la disponibilidad en un modelo dado. En este modelo mejorado, no solo señalamos fallas sino también reparaciones.

Tanto el análisis cualitativo como el cuantitativo se pueden llevar a cabo en Fault Trees. Tradicionalmente, para las SFT, el análisis cualitativo se ha llevado a cabo descubriendo los conjuntos de corte mínimo (MCS) [95]. Además, muchas de las técnicas de análisis existentes sobre SFT se basan en estos conjuntos. Un conjunto de cortes es un conjunto de eventos básicos que provoca la ocurrencia del evento superior del árbol. Un conjunto cortado es mínimo si no tiene un subconjunto propio que también sea un conjunto cortado. En una SFT, si ocurre el evento top es porque fallaron todos los BEs en uno de sus MCS. El descubrimiento de instancias de un conjunto de corte mínimo para el cual las probabilidades de falla de sus BEs son altas, podría indicar la necesidad de fortalecer esa parte del sistema. Aunque parezca un análisis simple, suele ocurrir que ciertos MCS pueden escapar a una simple inspección visual del modelo, por lo que se han propuesto varias técnicas para llevar a cabo un descubrimiento más preciso de estos conjuntos, como la manipulación booleana o Técnicas de BDD [95].

Las restricciones de tiempo introducidas por las compuertas dinámicas hacen que MCS no sea muy útil para los DFT. Tome como ejemplo un modelo de una sola compuerta PAND con dos entradas BE_1 y BE_2 como en la Figura 5.2. Si BE_1 y BE_2 fallan en ese orden, entonces podemos pensar en ellos como

un MCS, aunque no es el caso que cada vez que fallan, el sistema falla. Los MCS no cubren todo el comportamiento de las compuertas dinámicas y no son suficientes para analizar las restricciones de fallas introducidas por las DFT. Una alternativa a MCS son los conjuntos de secuencias de corte [75]. Las secuencias de corte funcionan separando los aspectos combinatorios y temporales de las compuertas dinámicas para extraer los MCS e imponer un orden dentro de ellos después. [69] analiza algunos problemas importantes relacionados con las secuencias de corte, que pueden desalentar su uso en el caso general de las DFT.

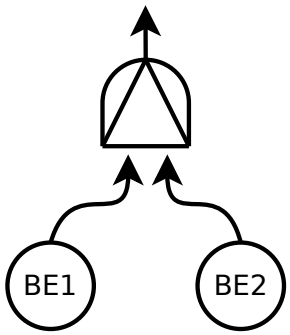


Figura 5.2: Pand gate.

Ningún MCS ni secuencias de corte son suficientemente interesantes en el caso de RFT ya que no tienen en cuenta el proceso de reparación. Esto es, la probabilidad de falla de un conjunto de cortes no puede tomarse directamente sin un análisis de espacio de estados dados los complejos mecanismos de reparación interdependientes. Por lo tanto, no hay mucha importancia en la obtención de los conjuntos de corte. Además, por la misma razón, no son tan relevantes en el análisis cuantitativo como lo son para las OFV.

Por otro lado, el análisis cuantitativo puede ser de gran interés en el análisis de las RFT. Las medidas típicas de análisis cuantitativo de interés son *confiabilidad* y *disponibilidad*. La fiabilidad de un modelo se define como la probabilidad de que el sistema que representa no falle durante un tiempo determinado, mientras que la disponibilidad de un sistema se define como la probabilidad de que el sistema esté funcionando. En el caso de la confiabilidad, generalmente sucede que nos gustaría calcular lo contrario, es decir, la probabilidad de falla a largo plazo de un árbol de fallas y esperamos que sea lo más pequeña posible. Si bien la confiabilidad suele ser una medida interesante para tomar en una SFT (que tiene una sola ejecución), la disponibilidad suele ser la medida interesante para tomar en una RFT donde el sistema puede repararse y, por lo tanto, una probabilidad a largo plazo tendrá un significado más fuerte.

La técnica de análisis más eficiente en SFT no estocásticos consiste en construir un Diagrama de decisión binaria (BDD) que represente la misma fórmula que el FT y luego resolver el estudio de confiabilidad requerido utilizando varios algoritmos optimizados. Para el caso de DFT, se introdujeron nuevas técnicas de análisis para capturar los requisitos temporales, como secuencias de corte, traducción a modelos de Markov [51, 51, 17], Sequence Binary Decision Diagrams [55, 87, 104], enfoques algebraicos [80, 2], simulación y combinación y optimizaciones de estos métodos [11, 58]. El comportamiento cíclico introducido por la reparación de fallas de interdependencia en las RFT no permite la mayoría de estas técnicas, y se debe considerar un enfoque basado en el estado, como la simulación de eventos discretos o la verificación de modelos.

Encontramos dos enfoques posibles para analizar las RFT. Un primer enfoque sería traducir el modelo a un modelo de Markov, tal vez aplicando tantas optimizaciones durante el modelado y el análisis para aliviar el problema de la explosión de estado. Este es el enfoque seguido por muchos trabajos como [9, 11, 12]. Se pueden señalar dos inconvenientes principales en este enfoque. La primera es que no importa qué métodos de optimización existentes se utilicen, no hay garantía de que habrá una reducción significativa del espacio de estado en los modelos generales. Esta es una situación especialmente difícil en el análisis de sistemas industriales grandes y complejos que involucran reparación. Un segundo inconveniente es la restricción a eventos distribuidos exponencialmente, lo que no captura adecuadamente el comportamiento correcto de los eventos donde el tiempo está gobernado por otras distribuciones continuas. Este es el caso, por ejemplo, de fenómenos como los tiempos de espera en los protocolos de comunicación, los plazos estrictos en los sistemas en tiempo real, los tiempos de respuesta humanos o la variabilidad del retardo de los fotogramas de sonido y vídeo (el llamado jitter) en los modernos sistemas de comunicación multimedia. , que normalmente se describen mediante distribuciones sin memoria, como distribuciones uniformes, logarítmicas normales o de Weibull [44]. Un segundo enfoque para el análisis RFT sería recurrir a la simulación, que no necesita construir el espacio de estado completo del modelo, y no impone *per se* la restricción a ningún tipo particular de distribuciones probabilísticas. El principal problema a la hora de enfrentarse a la simulación es la gran cantidad de cómputo necesaria para llegar a un resultado suficientemente preciso si se trata de eventos muy bajos posibles. Este es un tema muy relevante cuando se analizan sistemas altamente confiables o tolerantes a fallas, donde la probabilidad de falla es muy pequeña y la simulación plana de Monte Carlo se vuelve inviable. Para enfrentar este problema, se pueden utilizar técnicas de simulación de eventos raros, como división de importancia o muestreo de

importancia [101, 26, 27, 88].

En esta sección presentamos una definición formal de árboles de fallas reparables, junto con su semántica dada en términos de autómatas estocásticos de entrada/salida (IOSA). También analizamos el determinismo en los modelos RFT, con el fin de obtener modelos adecuados para la simulación de eventos discretos. Muchos trabajos abordan el problema de definir una sintaxis y una semántica rigurosas para FT, DFT y RFT, siendo algunos de ellos [34, 12, 18, 8]. Suelen diferir, por ejemplo, en los tipos y el significado de las compuertas, el poder de expresividad, cómo se reclaman los elementos de repuesto y cómo se resuelven las carreras de reparación. La presencia de situaciones no deterministas también es un tema discordante principal. Se han abordado otros temas en la literatura, como la agregación compositiva y el modelado modular [17]. Se pueden encontrar buenas encuestas sobre el estado del arte de FT y herramientas en [92], [69] y [7] . Aunque se usan ampliamente, los árboles de fallas todavía tienen varias limitaciones. En particular, nos centramos en los siguientes:

(a) El poder de expresividad de los RFT viene acompañado de una vaga definición formal del comportamiento de sus componentes. Muchos trabajos abordan o al menos advierten sobre este tema. Es entonces un interés principal definir completamente el comportamiento de cada componente, de manera que ninguna ambigüedad pueda perjudicar el modelado y análisis de un modelo, intentando, al mismo tiempo, ser lo más permisivo posible, para permitir modelar tantas características de sistemas reales como sea posible. Nos interesa prestar especial atención a aquellas subespecificaciones que podrían conducir a un comportamiento no determinista que haría que los modelos no fueran aptos para la simulación de eventos discretos.

(b) A medida que el software y los sistemas integrados se vuelven más y más grandes, las técnicas analíticas y numéricas se vuelven inviables dado el enorme tamaño del espacio de estado para construir y explorar. Otras técnicas, siendo la simulación la más popular, presentan una alternativa que, aunque resulta en una respuesta aproximada, presenta una confianza sintonizable y adecuada sobre el resultado dado. La simulación no se puede llevar a cabo en sistemas no deterministas. Entonces, es un asunto principal asegurar que los modelos sean deterministas, es decir, que en cada paso del cálculo podamos determinar un siguiente paso probabilístico único. En el caso de las RFTs, el no determinismo es un problema muy extendido, dada la subespecificación sobre el comportamiento de sus componentes. Esta subespecificación muchas veces tiene el propósito

de permitir libertad al modelador. Otras veces, es un mecanismo efectivo para analizar diferentes soluciones posibles a un hecho desconocido en el sistema modelado.

(c) La distribución exponencial no captura muchos de los comportamientos probabilísticos de los sistemas de la vida real que los FT pretenden modelar y analizar. Desafortunadamente, la mayoría de las herramientas de análisis DFT actuales solo permiten distribuciones exponenciales, por lo que se restringen solo a los modelos markovianos. Además, los pocos casos en los que permiten distribuciones no markovianas restringen el modelo a un pequeño rango de compuertas y no ofrecen soporte para el modelo de reparación [7]. Entonces, existe la necesidad de conferir una semántica más general a los modelos RFT que permita adoptar distribuciones continuas arbitrarias como tasas de falla y reparación.

En la sección 5.5 definimos formalmente la sintaxis de las RFT siguiendo las ideas de [16]. Además, para definir la semántica determinista composicional usando IOSA$_u$ discutimos diferentes preocupaciones sobre el determinismo en las RFT. Nuestra principal contribución en este capítulo consiste en permitir distribuciones de probabilidad generales para las tasas de falla y reparación. Con esto cubrimos puntos (**??**) y (**??**) de nuestras preocupaciones. Esta capacidad viene dada por nuestro lenguaje de modelado IOSA$_u$ que al mismo tiempo genera modelos RFT débilmente deterministas, centrándose también en el punto (c). Además, podemos simular estos modelos deterministas utilizando la herramienta de simulación de eventos raros FIG, lo que aumenta considerablemente la eficiencia al analizar sistemas altamente confiables [31, 28, 89, 101], y si el modelo es compatible, también en otras herramientas vía Jani [30]. Recientemente, [91] abordó el tema del uso de simulación de eventos raros para analizar árboles de fallas, aunque se restringen a distribuciones exponenciales y Erlang. Por el momento, no conocemos ningún otro enfoque que aplique la simulación de eventos raros específicamente a los árboles de fallas.

5.1. Repairable Fault Trees

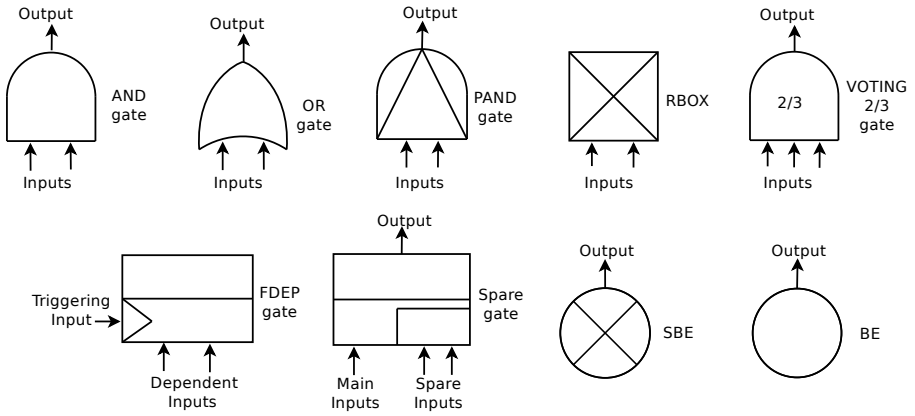


Figura 5.3: RFT elements

En la Figura 5.3 presentamos la representación gráfica de cada posible compuerta en una RFT. Cada elemento tiene un conjunto de entradas donde conectar sus subárboles y una salida para propagar las señales de falla, reparación y otras. La propagación de una falla y su posterior reparación comienza en las hojas del árbol de fallas. Solo los elementos básicos pueden ser una hoja en un árbol de fallas.

Elementos básicos (BE) son las hojas de las RFT. Modelan unidades atómicas de falla y también la posibilidad de reparación. Están representados gráficamente por un círculo, y van acompañados de una distribución de tiempo de falla, así como una distribución de tiempo de reparación. Por lo general, describen un componente básico del sistema modelado, para el cual se conocen los tiempos de falla y reparación. Los BE en árboles de fallas reparables señalarán un evento de falla cuando fallen, como en los árboles de fallas originales y, además, señalarán un evento de reparación cuando sean reparados. Cuando un BE falla o se repara, propaga instantáneamente el evento a las compuertas a las que está conectado. Podemos pensar que el estado de una compuerta cambia en función de las señales que recibe de sus entradas. Por lo general, una señal de falla puede cambiar el estado de la compuerta a fallar y una señal de reparación puede cambiarlo a un estado de funcionamiento. Al mismo tiempo, las compuertas pueden emitir una señal cuando cambia su estado. Una vez más, normalmente será el caso de que emitirán una señal de falla cuando su estado cambie de funcionamiento a falla, y una señal de reparación cuando cambie al revés. También se pueden producir otras señales, como puede ser el caso de las cajas de reparación,

que pueden enviar una señal de "comenzar a reparar" a cualquiera de los BE que están a cargo de reparar. La intuición sobre el comportamiento de cada compuerta es la siguiente.

Una *compuerta AND* falla cuando todas sus entradas fallan, y se repara cuando al menos una de sus entradas falla. Las compuertas AND corresponden a una conjunción lógica con respecto a las señales de falla.

Una *compuerta OR* falla cuando al menos una de sus entradas falla y deja de fallar cuando todas sus entradas se arreglan. Las compuertas OR corresponden a una disyunción lógica con respecto a las señales de falla.

Una *compuerta VOT* falla cada vez que al menos k de sus n entradas fallan y deja de fallar si, en un estado en el que exactamente k de sus entradas están fallando, una de ellas es reparada. Este tipo de compuerta se puede reemplazar por una combinación adecuada de compuertas AND y OR.

Tenga en cuenta que las tres compuertas que presentamos hasta ahora reaccionan solo en función de los cambios en la combinación de sus entradas. Estos se denominan *compuertas estáticas* y ya están presentes en los árboles de fallas estáticas. Por el contrario, las siguientes compuertas se llaman *compuertas dinámicas* y reaccionan a las señales de sus entradas teniendo en cuenta otros aspectos como los tiempos y la dependencia.

Una *compuerta PAND* falla cuando todas sus entradas fallan y lo hacen de izquierda a derecha, imponiendo una condición de orden en la ocurrencia de la falla. Se repara cada vez que se repara su última entrada. Tenga en cuenta esta condición de reparación. No se ha escrito mucho sobre la reparación de una compuerta PAND. Decidimos esta posibilidad basándonos en los casos más utilizados, como el de la Figura 5.1, donde arreglar la primera entrada no representa arreglar el sistema representado, mientras que arreglar la segunda entrada sí lo hace. Si estamos de acuerdo en que una compuerta PAND de n entradas se puede modelar con una secuencia de compuertas PAND de 2 entradas $n-1$ conectadas en cascada, entonces nuestro punto sigue siendo consistente y razonable para compuertas PAND más grandes. Otra posibilidad que no seguimos, pero que aún tiene sentido, es esperar hasta que se arreglen todas las entradas de la compuerta PAND. Finalmente, no encontramos ningún significado razonable en el caso restante, es decir, arreglar la compuerta PAND tan pronto como la primera entrada (u otra que no sea la última) se arregla. Es por eso que nuestra elección en el comportamiento de reparación de la compuerta PAND.

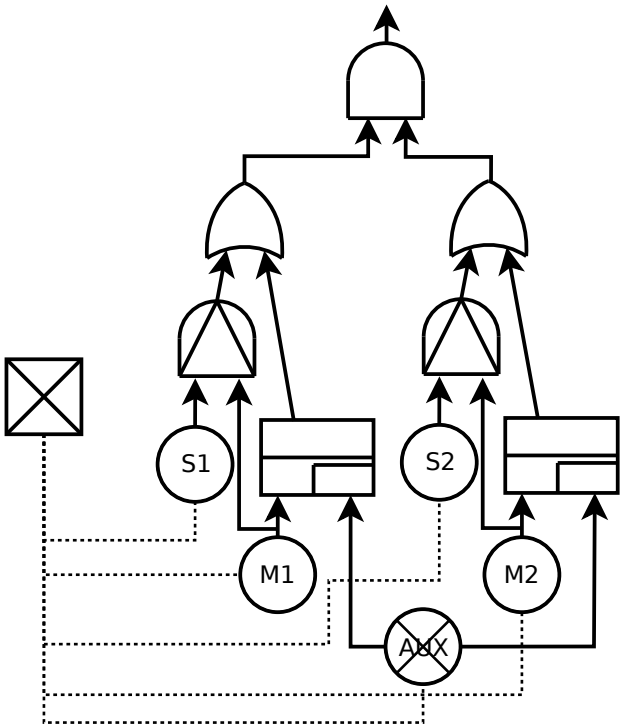


Figura 5.4: Un sistema de refrigeración tolerante a fallas.

Una *Functional dependency gate (FDEP)*, o *Compuerta de Dependencia Funcional*, tiene $n + 1$ entradas. La señal de falla de una de sus entradas (la de activación) hace que todas las demás entradas sean inaccesibles para el resto del sistema. Tenga en cuenta que las entradas dependientes no fallan y estarán accesibles nuevamente tan pronto como se repare el componente desencadenante (observe la diferencia con [16, 91] donde los BE dependientes fallar). Esta compuerta es un azúcar sintáctico para un sistema de compuertas OR como se muestra en la Figura 5.5. Consulte la Sección 5.2 para obtener más información sobre estas compuertas.

Un *Elemento básico de repuesto (SBE)* es un caso especial de BE que se puede habilitar y deshabilitar, y se puede usar como repuesto para otros BE por medio de compuertas de repuesto. Un SBE puede ser compartido por varias compuertas de repuesto y se introducen diferentes políticas de uso compartido para este propósito. Es común distinguir tres tipos de SBE, dependiendo de si no fallan cuando están deshabilitados, llamados *cold SBE*, o fallan pero con una tasa menor que cuando están habilitados, llamados *warm SBE*, o fallan con la misma frecuencia que lo hacen cuando están habilitados, llamados *hot SBE*.

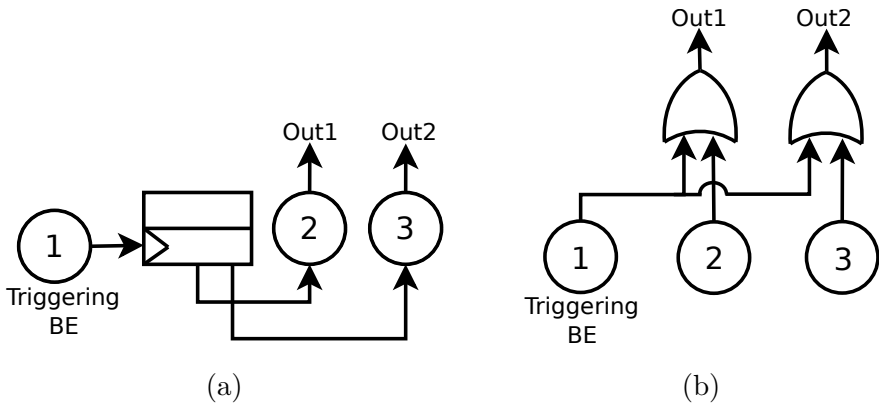


Figura 5.5: Un sistema de compuerta FDEP (a) y una construcción equivalente (b).

Una *compuerta de repuesto (SG)* permite respaldar un elemento básico principal con varios elementos básicos de repuesto en caso de que falle el principal. Cada compuerta de repuesto tiene una entrada principal y n entradas de repuestos. Una entrada principal solo puede ser un BE. Las entradas de repuesto solo pueden ser SBE. Tan pronto como falla la entrada principal, el SG selecciona el siguiente elemento de repuesto disponible para reemplazarlo. El SG fallará siempre que no obtenga un reemplazo, e informará a la reparación cada vez que se repare la entrada principal o se obtenga una entrada de repuesto. Si un reemplazo en uso falla, el SG buscará uno nuevo. Si se repara la entrada principal, el SG liberará su entrada de repuesto adquirida, en caso de que exista. Puede encontrar un ejemplo del uso de compuertas de repuesto y SBE en la Figura 5.4. Muestra un modelo RFT de un sistema de dos bombas de agua (*M1* y *M2*) con una bomba de respaldo de repuesto (AUX) en caso de que alguna de ellas falle. Si ambos subsistemas fallan, todo el sistema falla (observe la compuerta AND en la parte superior). Ambos subsistemas comparten una bomba de repuesto que está conectada a sus correspondientes compuertas de repuesto. Cuando se rompe una bomba principal, la compuerta de repuesto correspondiente solicita la bomba Auxiliar. Si la bomba de repuesto no está rota y disponible, entonces la compuerta de repuesto la habilita para que comience a funcionar. Se puede encontrar una descripción más detallada del modelo y su comportamiento en la Sección 5.8.2.

Una *Repair Boxs (RBOX)* es la unidad encargada de gestionar la reparación de los BE y SBE averiados. Tiene n entradas, correspondientes a cada uno de los elementos básicos que debe reparar, y una salida ficticia. Una política de RBOX determina en qué orden se repararán los BE (o SBE) de-

fectuosos. El tiempo de reparación de cada entrada se define en el propio BE (o SBE). Creemos que este es un mejor enfoque que definir un solo tiempo de reparación en el cuadro de reparación, ya que la naturaleza de las entradas varía y puede influir en la complejidad de la reparación. Tenga en cuenta que el simple hecho de cambiar las políticas en las Cajas de reparación ya abre la posibilidad de comparar diferentes escenarios que ciertamente influirán en la confiabilidad del sistema modelado.

5.2. Discusiones sobre diseño

En esta sección nos gustaría comentar algunas alternativas a lo que hemos decidido sobre el diseño de los elementos de un Árbol de Fallas Reparable. Tenemos la intención de aclarar y justificar nuestras elecciones sobre la semántica de las diferentes compuertas.

Es notable que en los árboles de fallas estándar no se consideró ningún razonamiento sobre la reparación de un componente del modelo. En efecto, sólo se consideró la propagación y combinación lógica de fallas. De hecho, los elementos básicos en SFT son las fallas de los componentes del sistema que luego se propagan y combinan, construyendo fallas más complejas hasta llegar al evento principal. En este contexto, las hojas de los árboles representan toda la información que necesitamos sobre estos componentes atómicos, es decir, las tasas de falla. Con la introducción de las unidades de reparación (RB) y la posibilidad de reparar componentes rotos del sistema, toda la lógica cambia. En primer lugar, existe la necesidad de introducir un nuevo tipo de evento, que es el evento de reparación. Además, ahora las hojas necesitan representar más información para modelar completamente un componente atómico. El significado anterior de un elemento básico no cubre estos eventos de reparación, y por su simplicidad se denominan *Eventos Básicos*. Una hoja ahora tiene información sobre ambos tipos de eventos para un componente, su evento de falla y su evento de reparación, que en su lugar están comprendidos por el nombre *Elemento básico*.

El cambio de dinámica alcanza también a las distintas compuertas del árbol. En algunos casos, como las compuertas AND y OR, se logra una definición clara y simple del comportamiento bajo la reparación de componentes sin mucha dificultad y sin introducir el no determinismo que es de mayor interés. Este no es el caso de la compuerta PAND, por ejemplo, como hemos visto anteriormente, donde surgen muchas preguntas y situaciones que involucran el no determinismo obligan a un análisis más profundo.

Otra compuerta que merece nuestra atención es la compuerta de dependencia funcional (FDEP). En trabajos anteriores como [51] esta compuerta se

definía para deshabilitar el acceso a varios componentes como consecuencia de la falla de un componente activador. Tenga en cuenta que no hubo una mención explícita sobre la falla de los componentes dependientes, la única consecuencia fue no poder acceder a ellos más. En el contexto de un Árbol de Fallas Estándar, la falla de los componentes dependientes tenía el mismo efecto que no haber podido acceder a ellos, ya que no existía la noción de reparar después de la falla. En nuestro contexto, por otro lado, estas dos situaciones diferentes tienen un impacto severo en el significado de la compuerta. Decidimos mantener el significado de [51] en contraste con otros trabajos como [34, 91] donde la falla del elemento básico desencadenante provoca la falla de los elementos básicos dependientes. Es decir, los elementos básicos dependientes no fallarán como consecuencia de la falla del elemento básico desencadenante, sino que se volverán inaccesibles. Elegir un enfoque adecuado requeriría analizar el no determinismo causado por las fallas simultáneas. Nuestra decisión de modelado asegura que esta compuerta no introduzca ningún no determinismo. El trabajo [17] menciona que no es difícil generalizar la compuerta FDEP para aceptar otras compuertas o subárboles como entradas dependientes. Esto también podría hacerse en nuestro marco, pero hemos decidido no trabajar en ello todavía.

La compuerta de repuesto y el elemento básico de repuesto también merecen una discusión. Si bien algunos trabajos permiten ampliar las posibilidades de estas construcciones para permitir que un subárbol general funcione como parte de repuesto de un subárbol defectuoso [18], esto sin duda requeriría restringir el repuesto permitido subárboles para evitar el no determinismo. Dejamos esta posibilidad para próximos trabajos.

En general, muchas extensiones y soporte adicional para compuertas pueden analizarse e introducirse en nuestro RFT. Sin embargo, como primer paso, tenemos la intención de dar aquí un marco estricto pero robusto que se puede asegurar que sea determinista, y dejar esas posibilidades para trabajo futuro.

5.3. El lenguaje simbólico de IOSA

Presentamos un lenguaje simbólico para describir modelos de IOSA_u que facilitará el modelado al evitar la descripción extensa del espacio de estado discreto y numerosas transiciones. Dado que nuestro marco (IOSA_u) es compositivo, esto se refleja también en el lenguaje, donde cada componente se modela por separado mediante lo que llamamos un *módulo*. Un módulo está compuesto por un conjunto de variables, cuya valoración representa el estado actual del componente, un conjunto de relojes correspondientes a

los relojes de habilitación para transiciones no urgentes, y un conjunto de transiciones que describen simbólicamente los posibles saltos entre estados (cambios de valoraciones y puesta a cero de relojes). Figure 5.6 modela un elemento básico como ejemplo. Las variables pueden ser de tipo entero (con rango finito) o booleano. Como veremos más adelante, también los arreglos se pueden definir como variables.

```
1  module BE
2    fc, rc : clock;
3    signal : [0..2] init 0;
4    broken : [0..2] init 0;
5
6    [fl!] broken=0 @ fc -> (signal'=1) & (broken'=1);
7    [r??] broken=1 -> (broken'=2) & (rc'=γ);
8    [up!] broken=2 @ rc -> (signal'=2) &
9                           (broken'=0) & (fc'=μ);
10
11   [f!!] signal=1 -> (signal'=0);
12   [u!!] signal=2 -> (signal'=0);
13 endmodule
```

Figura 5.6: Modelo simbólico IOSA de un Basic Element.

El valor inicial para cada variable se determina después de la palabra clave `init`. Las distribuciones de relojes se definen en las transiciones donde se reinician los relojes. Una transición se describe por el nombre de la acción que tiene lugar (o sin nombre si no es necesario), una condición que restringe los estados de origen, un reloj de habilitación (solo para el caso de transiciones de salida no urgentes), una condición que describe el destino estados, y el conjunto de relojes que se restablecerán. Una visión general rápida de la figura 5.6 ayudará a comprender mejor nuestro lenguaje simbólico: dos relojes, `fc` y `rc`, se definen en la línea 2. estos relojes se utilizarán como relojes habilitadores para las transiciones en las líneas 6 y 8, y se restablecerán en las transiciones en las líneas 7 y 8. Las líneas 3 y 4 definen las variables `signal` y `broken`, ambas de tipo entero entre 0 y 2, e inicializadas con valor 0. La línea 6 define un conjunto de transiciones no urgentes de salida, que producen la acción de salida `fl`. Más precisamente, esta línea define el conjunto de transiciones no urgentes $s \xrightarrow{\{fc\},fl!,\varnothing} s'$, donde s cumple la condición `broken=0`, y s' es el resultado de cambiar los valores en el estado s de las variables `signal` y `broken` a `1`. El símbolo `@` precede al reloj habilitador

para la transición, mientras que el símbolo → distingue entre las condiciones para el estado de origen y el estado de destino. Las condiciones en el estado objetivo se expresan como asignaciones a los siguientes valores de estado de las variables, indicados con un apóstrofe. La línea 7 define una transición de entrada urgente con la etiqueta `r`. Los signos de interrogación dobles después del nombre indican que describe transiciones de entrada urgentes. Las transiciones de salida urgente se indican con signos de exclamación dobles (`!!`), las transiciones de entrada no urgentes con un solo signo de interrogación y las transiciones de salida no urgentes con un solo signo de exclamación o sin marca si no se proporciona una etiqueta. Al final de la línea encontramos el reinicio del reloj `rc` a un valor muestreado con una distribución de probabilidad γ. Esta línea luego define las transiciones $s \xrightarrow{\varnothing, r??, \{rc\}} s'$, donde s cumple con la condición `broken=1`, s' es idéntico a s excepto por la variable `roto` que tiene valor `2`, y el reloj `rc` tiene distribución γ. En la línea 10, se define una transición de salida urgente, lo que indica la falla de este componente. Usualmente usaremos estas transiciones urgentes para sincronizar y comunicarnos con otros módulos. Como nota final sobre el asunto, debemos señalar que un modelo como el descrito en la Figura 5.6 no es legalmente un IOSA ya que no está habilitado para entrada. De hecho, le falta una transición:

```
[r??] broken != 1 -> ;
```

No obstante, en aras de la simplicidad, evitaremos escribir este tipo de transiciones de "self loop", aunque supondremos que existen en el modelo.

La especificación completa del lenguaje de modelado simbólico IOSA se puede encontrar en el Apéndice A. Un modelo IOSA se describe mediante un conjunto de módulos, cada uno de los cuales describe un componente concurrente del sistema a modelar. El cuerpo de un módulo se puede dividir claramente en tres partes: las declaraciones de variables, las declaraciones de relojes y la especificación de transición. Los arreglos se declaran junto con las variables, con el requisito adicional de definir el rango del arreglo entre paréntesis. Los guardias de transición son fórmulas booleanas que describen los estados de origen de la transición simbólica. En este caso, el símbolo & representa el operador de conjunción proposicional, mientras que | representa el operador de disyunción proposicional y ! la negación. Las asignaciones (o postcondiciones) por otro lado, describen los cambios de los valores. Cada asignación está entre paréntesis y el nombre de la variable va seguido de un apóstrofe para indicar que corresponde al valor de la variable en el estado alcanzado después de realizar la transición. Un ampersand ('&') separa cada asignación. Observe la similitud con la sintaxis PRISM [71] para describir las transiciones. Junto a la asignación de valores a futuras variables, encontramos el reseteo de relojes. A un reloj se le asigna una distribución de probabilidad

$(clock' = \gamma)$ para indicar que se establecerá en un valor de esa distribución de probabilidad inmediatamente antes de alcanzar el nuevo estado. Para un reloj c, su distribución de probabilidad debe ser siempre la misma, por lo que debe coincidir en todas las líneas donde se encuentra configurado. Cada línea que define una variable o matriz, describe una transición o declara un reloj, termina en punto y coma y se puede colocar en cualquier lugar dentro del alcance de un módulo.

5.4. Una definición formal de RFT

En esta sección presentamos una definición formal de RFT junto con su semántica dada en términos de IOSA$_u$. Extendemos la formalización DFT dada por [17] con nuevas funciones como cuadros de reparación y algunas otras modificaciones como condiciones para garantizar el determinismo. Cada elemento de RFT se caracteriza por una tupla que consta de su tipo, su aridad, es decir, el número de entradas y posiblemente otros parámetros, como las distribuciones de probabilidad para los eventos de falla y reparación en BE.

Definición 5.1. Sean n, m y k pertenecientes a $\mathbb{N}^+$, y sean μ, ν y γ distribuciones de probabilidad continuas. Definimos el conjunto $\mathcal{E}$ de elementos de un RFT para que esté compuesto por las siguientes tuplas:

- $(\mathsf{be}, 0, \mu, \gamma)$ y $(\mathsf{sbe}, 0, \mu, \nu, \gamma)$, que representa el elemento básico y básico de repuesto, sin entradas, con una distribución de fallas activa μ, una distribución de falla latente ν y una distribución de reparación γ.
- (and, n), (or, n) and (pand, n), que representan compuertas AND, OR and PAND con n entradas, respectivamente,
- (vot, n, k), que representan compuertas VOT k en n,
- (fdep, n), que representa una compuerta de dependencia funcional, con una entrada de activación y $n-1$ entradas dependientes. Por convención la primera entradas es la de activación,
- (sg, n), que representa una compuerta de repuestos con una entrada principal y $n-1$ entradas de repuesto. Por convención, la primera entrada es la principal.
- (rbox, n), que representa una RBOX para n BEs (o SBEs).

Un RFT es un grafo dirigido acíclico, para el cual cada vértice v está etiquetado con un elemento $l(v) \in \mathcal{E}$. Un borde de v a w significa que la salida de v está conectada a una entrada de w. Dado que el orden de las entradas es relevante, las damos en términos de una lista $i(w)$ en lugar de un conjunto. De manera similar, $si(v)$ enumerará todas las puertas de repuesto a las que se conecta un elemento básico de repuesto v como entrada. Sea $\mathrm{t}(v)$ el tipo de v. Es decir, $\mathrm{t}(v)$ es la primera proyección de $l(v)$. Sea $\#(v)$ el número de entradas de v, es decir, es la segunda proyección $l(v)$.

Definición 5.2. Un Repairable Fault Tree es una 4-upla $T = (V, i, si, l\ )$, donde V es un conjunto de vértices, $l\colon V \to \mathcal{E}$ es una función que etiqueta cada vértice con un elemento RFT, $i\colon V \to V^*$ es una función que asigna $\#(v)$ entradas a cada elemento v en V, y $si\colon V \to V^*$ que indican qué puertas de repuesto administran cada SBE. El conjunto de aristas $\mathrm{E} = \{(v, w) \in V^2 \mid \exists j \cdot v = (i(w))[j]\}$ es el conjunto de pares (v, w) tal que v es una entrada de w. Si tal borde existe, diremos que v está conectado a w y w a v.

Para que una RFT T esté *bien formado*, se deben cumplir las siguientes condiciones:

(I) La tupla (V, E) es un grafo dirigido acíclico (DAG en inglés).

(II) T tiene un elemento superior único, es decir, un elemento único cuya salida no ficticia no está conectada a otra puerta. Es decir, existe un único vértice $v \in V$ tal que para todo $w \in V$, $(v, w) \notin E$ y $\mathrm{t}(v) \notin \{\mathsf{fdep}, \mathsf{rbox}\}$. Una salida no puede ser más de una vez entrada de una misma puerta. Es decir, para todo $1 \leq j, k \leq |i(w)|$ con $i(w)[j] = i(w)[k]$, tenemos $j = k$.

(III) Dado que las salidas de FDEP y RBOX son ficticias, si $(v, w) \in E$ entonces $\mathrm{t}(v) \notin \{\mathsf{fdep}, \mathsf{rbox}\}$.

(IV) Las entradas de un RBOX solo pueden ser elementos básicos. Es decir, si $(v, w) \in E$ y $\mathrm{t}(w) = \mathsf{rbox}$ entonces $\mathrm{t}(v) = \mathsf{be}$ o $\mathrm{t}(v) = \mathsf{sbe}$.

(V) Cada elemento básico (de repuesto) se puede conectar como máximo a una RBOX. Esto es si $(v, w) \in E$ y $(v, w') \in E$ y $\mathrm{t}(w) = \mathrm{t}(w') = \mathsf{rbox}$, entonces $w = w'$.

(VI) Las entradas de repuesto de una puerta de repuesto solo pueden ser SBE, mientras que su entrada principal solo puede ser un BE. Es decir, si $(v, w) \in E$ y $\mathrm{t}(w) = \mathsf{sg}$ entonces $\mathrm{t}(i(v)[0]) = \mathsf{be}$ y para $j > 0, \mathrm{t}(i(v)[j]) = \mathsf{sbe}$. Además, un SBE solo se puede conectar a una puerta de repuesto o a un RBOX, es decir, si $(v, w) \in E$ y $\mathrm{t}(v) = \mathsf{sbe}$ entonces $\mathrm{t}(w) \in \{\mathsf{sg}, \mathsf{rbox}\}$.

(VII) Un elemento básico de repuesto es una entrada de una puerta de repuesto, si y solo si esa puerta de repuesto es una entrada de repuesto del elemento básico de repuesto, es decir, para v y v' tal que $l(v') = (\ SP, 0, \mu, \nu, \gamma)$ y $l(v) = (\mathsf{sg}, n)$, $(v', v) \in E$ si y solo si existe j tal que $v = si(v')[j]$.

(VIII) Un elemento básico se puede conectar como máximo a una puerta de repuesto, es decir, si $(v, w) \in E$ y $(v, w') \in E$ con $\mathrm{t}(w) = \mathrm{t}(w') = \mathsf{sg}$ y $\mathrm{t}(v) = \mathsf{be}$ luego $w = w'$.

(IX) Si un elemento básico está conectado a una puerta de repuesto, entonces no puede conectarse a una puerta FDEP, es decir, si $(v, w) \in E$ y $\mathrm{t}(v) = \mathsf{be}$ y $\mathrm{t}(w') = \mathsf{sg}$, entonces no hay $(v, w') \in E$ tal que $\mathrm{t}(w') = \mathsf{fdep}$.

Muchas de las condiciones que hemos impuesto a los árboles de fallas reparables obedecen a dos razones principales: el RFT no debe tener bucles y debe ser determinista. Las condiciones (I), (II) y (II) son condiciones habituales para los FT, ya que aseguran que la estructura corresponde a un árbol. . La condición (III) asegura que las salidas ficticias permanezcan libres. Las condiciones (IV) y (V) aseguran que RBOX funcione correctamente. En particular (V) evita situaciones no deterministas en las que, después de que falla un elemento básico, hay más de un RBOX disponible para solucionarlo.

La condición (VI) garantiza que los SBE actúen solo como repuesto. Es decir, se conectan como repuestos de una puerta de repuesto y se pueden reparar como otros elementos básicos. Además, garantiza que las puertas de repuesto solo gestionen BE y SBE, ya que la gestión de otras puertas requeriría un análisis más amplio sobre el determinismo que no forma parte de este trabajo.

Los siguientes elementos determinan, para un SBE determinado, qué puertas de repuesto pueden solicitarlo. Definir correctamente la función *si* sería esencial para esto, por lo que la condición (VII) está ahí para asegurarlo. La condición (VIII) controla que el elemento principal de una puerta de repuesto solo sea administrado por esta misma puerta de repuesto. Finalmente condition (IX) pretende evitar situaciones no deterministas como las que se muestran en la Figura 5.7. Estas situaciones no deterministas son el resultado de las condiciones de carrera para adquirir un reemplazo entre eventos principales que se vuelven inaccesibles al mismo tiempo. Tenga en cuenta que dado que FDEP puede ser reemplazado por un sistema de puertas OR (como mostramos en 5.5), conectar los elementos básicos principales al FDEP tampoco cumpliría la condición (VI).

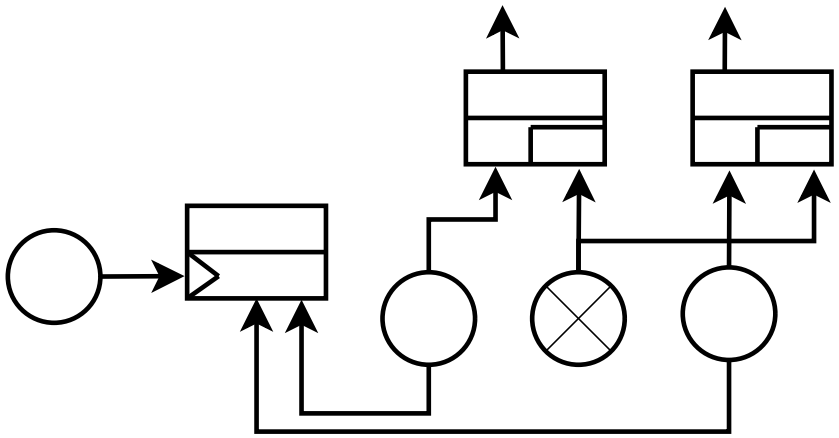


Figura 5.7: Condición de carrera entre puertas de repuesto por falla simultánea de componentes principales dada una conexión FDEP.

Para el resto de este trabajo solo consideraremos RFTs bien formados.

5.5. Semántica de RFT

Presentamos ahora una semántica paramétrica para cada elemento en $\mathcal{E}$. Esto se usará más adelante para definir la semántica de cada vértice de RFT, y la consecuente semántica del modelo completo como una composición paralela de la de sus componentes. Solo proporcionamos la semántica para BE, compuertas AND, compuertas OR, compuertas PAND y RBOX. Recuerde que las puertas FDEP son un azúcar de sintaxis y se pueden reemplazar usando puertas OR. De manera similar, las puertas de votación se pueden modelar mediante una serie de puertas AND y una puerta OR, aunque aquí se presenta un modelo más simple. Las puertas de repuesto y los SBE se presentarán más adelante en la Sección 5.7.

En el diseño de los módulos de IOSA debemos tener en cuenta la comunicación entre cada elemento de un RFT y sus hijos y padres. Por ejemplo, un elemento básico tiene que comunicar su falla y reparación a aquellas puertas para las que es una entrada. De manera similar, un RBOX debe comunicar a sus entradas una señal de "comenzar a reparar". Para ello, la semántica de cada elemento vendrá dada por una función, que toma acciones como parámetros. Vamos a llamar a estas acciones señales, ya que representarán fallas, reparaciones y otras señales de los elementos RFTĖn este sentido, para un índice de entrada i de un elemento $e \in \mathcal{E}$ llamaremos f_i a la señal que falla de esa entrada, u_i a la señal de up (reparado), y r a la señal que envían

las cajas de reparación a sus elementos básicos para iniciar el proceso de reparación. En consecuencia, nombraremos las señales de salida con f para fallas, u para reparación y r para "comenzar a reparar".

Elemento Básico.

Para un *BE* $e \in \mathcal{E}$, su semántica es una función $[\![(\mathsf{be}, 0, \mu, \gamma)]\!] : \mathcal{A}^5 \rightarrow$ IOSA, donde $[\![(\mathsf{be}, 0, \mu, \gamma)]\!](fl, up, f, u, r)$ resulta en el IOSA de la Figura 5.8. El estado de un elemento básico está definido por el reloj de falla `fc`, el

```
 1 module BE
 2   fc, rc : clock;
 3   signal : [0..2] init 0;
 4   broken : [0..2] init 0;
 5
 6   [fl!] broken=0 @ fc -> (signal'=1) & (broken'=1);
 7   [r??] broken=1 -> (broken'=2) & (rc'=γ);
 8   [up!] broken=2 @ rc -> (signal'=2) &
 9                          (broken'=0) & (fc'=μ);
10
11   [f!!] signal=1 -> (signal'=0);
12   [u!!] signal=2 -> (signal'=0);
13 endmodule
```

Figura 5.8: Basic Element IOSA symbolic model.

reloj de reparación `rc`, una variable `signal` que indica cuándo señalar la falla o reparación, y la variable `broken` para distinguir entre estados rotos y normales. Un elemento básico falla cuando expira el reloj `fc` (línea 6) e inmediatamente lo informa con la señal urgente `f!!` en la línea 11. Tan pronto como comienza la reparación por la correspondiente reparación conectada (línea 7), el reloj `rc` está configurado. Cuando caduca, el componente se repara. Por lo tanto, `fc` se establece de nuevo en la línea 8, y la reparación se señala con la acción urgente `u!!` en la línea 11. En el estado inicial de un módulo IOSA todos sus relojes se establecen aleatoriamente de acuerdo con sus distribuciones asociadas. Por lo tanto, `rc` se establece en el estado inicial y eventualmente podría expirar sin haber sido establecido por una transición de reparación. Es por esto que tenemos que distinguir entre los casos en que el BE está siendo reparado (`broken=2`) y cuando no lo está.

Compuerta AND.

Pra una *compuerta AND* con dos entradas, su semántica es una función $[\![(\mathsf{and}, 2)]\!] : \mathcal{A}^6 \to$ IOSA, donde $[\![(\mathsf{and}, 2)]\!](f, u, f_1, u_1, f_2, u_2)$ resulta en el siguiente IOSA:

```
 1 module AND
 2  singalf: bool init false;
 3  signalu: bool init false;
 4  count: [0..2] init 0;
 5
 6  [ f1??] count=1 -> (count'=2) & (signalf'=true);
 7  [ f1??] count=0 -> (count'=1);
 8  [ f1??] count=2 -> ;
 9  [ f2??] count=1 -> (count'=2) & (signalf'=true);
10  [ f2??] count=0 -> (count'=1);
11  [ f2??] count=2 -> ;
12
13  [ u1??] count=2 -> (count'=1) & (signalu'=true);
14  [ u1??] count=1 -> (count'=0);
15  [ u1??] count=0 -> ;
16  [ u2??] count=2 -> (count'=1) & (signalu'=true);
17  [ u2??] count=1 -> (count'=0);
18  [ u2??] count=0 -> ;
19
20  [ f!!] signalf & count=2 -> (signalf'=false);
21  [ u!!] signalu & count!=2 -> (signalu'=false);
22 endmodule
```

En las líneas 6 a 11, se informa a la compuerta AND del fallo de cualquiera de sus entradas. Cuando es así, distinguimos entre el caso en el que la otra entrada ya ha fallado (`count=1`) y el caso en el que no lo ha hecho (`count=0`). En el primer caso, tenemos que mostrar la falla de esta puerta, para lo cual configuramos la variable `signalf` para habilitar la transición en la línea 20. Además, en ambos casos aumentamos el valor de `count` para que tomamos nota del fallo de una entrada. Se hace un razonamiento similar para el caso de la reparación de una entrada en las líneas 13 a 18. En este caso, tenemos que configurar el módulo para señalar una reparación cuando una entrada se repara en un estado en el que ambas entradas estaban fallando (líneas 13 y 16), habilitando la transición en la línea 21. En otros modelos de puertas, omitiremos escribir los bucles propios originados por la habilitación de entrada de IOSA, como las líneas $8, 11, 15$ y 18 del modelo de puerta AND. No

obstante, remarcamos que es necesario tenerlos en cuenta al analizar la confluencia de los módulos en la sección 5.6. Si bien hemos descrito una puerta AND con solo dos entradas, es fácil imaginar una generalización a un mayor número de entradas. Además, es fácil modelar una puerta AND de entradas N componiendo varias puertas AND en topologías de cascada o pirámide.

Compuerta OR

Para una compuerta OR $e \in \mathcal{E}$ con dos entradas, su semántica es una función $[\![(\mathsf{or}, 2)]\!] : \mathcal{A}^6 \to \text{IOSA}$, donde $[\![(\mathsf{or}, 2)]\!](f, u, f_1, u_1, f_2, u_2)$ resulta en el siguiente IOSA:

```
1  module OR
2    signalf: bool init false;
3    signalu: bool init false;
4    count: [0..2] init 0;
5
6    [f1??] count=0 -> (count'=1) & (signalf'=true);
7    [f1??] count=1 -> (count'=2);
8    [f2??] count=0 -> (count'=1) & (signalf'=true);
9    [f2??] count=1 -> (count'=2);
10
11   [u1??] count=2 -> (count'=1);
12   [u1??] count=1 -> (count'=0) & (signalu'=true);
13   [u2??] count=2 -> (count'=1);
14   [u2??] count=1 -> (count'=0) & (signalu'=true);
15
16   [f!!] signalf & count!=0 -> (signalf'=false);
17   [u!!] signalu & count=0 -> (signalu'=false);
18 endmodule
```

Observe que el modelo de compuerta OR es muy similar a la compuerta AND. Tomamos como premisa para estos modelos que una entrada no se romperá dos veces seguidas sin ser reparada a la mitad, ni se reparará si no ha fallado. Al igual que para las compuertas AND, las compuertas OR con más de 2 entradas se pueden modelar o reemplazar fácilmente por una combinación de compuertas OR de entrada 2.

Compuerta VOT

El siguiente modelo IOSA corresponde a una compuerta de votación 2 de 3. Se puede obtener fácilmente una generalización a otros valores de N y K. Aunque se puede obtener un modelado alternativo de estas puertas mediante

una combinación de puertas OR y AND, es posible que desee reducir la complejidad del modelado del sistema utilizando la interpretación presentada aquí, que también resulta ser débilmente determinista.

Para una compuerta VOT 2 de 3 $e \in \mathcal{E}$, su semántica es una función $[\![(\mathsf{vot}, 3, 2)]\!] : \mathcal{A}^8 \to \text{IOSA}_u$, donde $[\![(\mathsf{vot}, 3, 2)]\!](f, u, f_0, u_0, f_1, u_1, f_2, u_2)$ resulta en el siguiente IOSA_u:

```
1  module VOTING
2    count: [0..3] init 0;
3    inform: bool init false;
4
5    [ f0??] -> (count'=count+1) & (inform'=(count+1=2));
6    [ f1??] -> (count'=count+1) & (inform'=(count+1=2));
7    [ f2??] -> (count'=count+1) & (inform'=(count+1=2));
8
9    [ u0??] -> (count'=count-1) & (inform'=(count=2));
10   [ u1??] -> (count'=count-1) & (inform'=(count=2));
11   [ u2??] -> (count'=count-1) & (inform'=(count=2));
12
13   [ f!!] inform & count >= 2 -> (inform'=false);
14   [ u!!] inform & count < 2 -> (inform'=false);
15 endmodule
```

Las puertas de votación se modelan utilizando un contador que cuenta cuántas entradas han fallado. Esto se hace escuchando las señales de falla correspondientes en las líneas 5 a 7, y las señales de reparación en las líneas 9 a 11. En estas mismas líneas tenemos en cuenta si acabamos de alcanzar el valor K (2 en nuestro ejemplo) o si acaba de descender este valor, que son las circunstancias bajo las cuales se informa la avería y la reparación respectivamente, que finalmente se realiza en las líneas 13 y 14.

Compuerta PAND

La semántica de una *Priority AND gate* con 2 entradas está definida por $[\![(\mathsf{pand}, 2)]\!] : \mathcal{A}^6 \to \text{IOSA}$, donde $[\![(\mathsf{pand}, 2)]\!](f, u, f_0, u_0, f_1, u_1)$ resulta en el siguiente IOSA_u:

```
1  module PAND
2
3   f1: bool init false;
4   f2: bool init false;
5   st: [0..4] init 0; // 0:up, 1:inform fail, 2:failed,
6                      // 3:inform up, 4:unbreakable
```

```
7
8   [_?] st=0 & f1 & !f0 -> (st'=4);
9
10  [ f0??] st=0 & !f0 & !f1-> (f0'=true);
11  [ f0??] st=0 & !f0 & f1 -> (st'=1) & (f0'=true);
12  [ f0??] st!=0 & !f0 -> (f0'=true);
13  [ f0??] f0 -> ;
14
15  [ f1??] st=0 & !f0 & !f1 -> (f1'=true);
16  [ f1??] st=0 & f0 & !f1 -> (st'=1) & (f1'=true);
17  [ f1??] st=3 & !f1 -> (st'=2) & (f1'=true);
18  [ f1??] (st==1|st==2|st=4) & !f1 -> (f1'=true);
19  [ f1??] f1 -> ;
20
21  [ u0??] st!=1 & f0 -> (f0'=false);
22  [ u0??] st=1 & f0 -> (st'=0) & (f0'=false);
23  [ u0??] !f0 -> ;
24
25  [ u1??] (st=0|st=3) & f1 -> (f1'=false);
26  [ u1??] (st=1|st=4) & f1 -> (st'=0) & (f1'=false);
27  [ u1??] st=2 & f1 -> (st'=3) & (f1'=false);
28
29  [ f!!] st=1 -> (st'=2);
30  [ u!!] st=3 -> (st'=0);
31
32 endmodule
```

Las compuertas PAND fallan solo cuando todas sus entradas fallan y lo hacen de izquierda a derecha. Esto permite condicionar la falla de un sistema no solo a la falla de los subsistemas sino también a la ordenación en el tiempo en que estos ocurren. Observe que una compuerta PAND de n entradas se puede modelar mediante un sistema de compuertas PAND de dos entradas $n-1$ conectadas en una topología en cascada. La literatura no siempre es clara o incluso discrepa sobre cuál debería ser el comportamiento de la puerta PAND en caso de que ambas entradas fallen al mismo tiempo [77, 34, 69]. Esta situación se da en algunas construcciones con compuertas AND y OR, o cuando las entradas de una compuerta PAND están conectadas a un mismo FDEP (ver Figura 5.9), ya que no hay un orden establecido en el fallo de las dependientes SER. Incluso debería cuestionarse si esto representa un comportamiento no determinista. Algunos trabajos no permiten estas situaciones y las descartan durante las primeras comprobaciones sintácticas

[91]. Algunos otros encuentran que el no determinismo es importante para analizar escenarios reales donde el comportamiento es de hecho desconocido [17]. Otros trabajos decidieron que la puerta PAND no se romperá a menos que sus entradas se rompan estrictamente de izquierda a derecha [16, 12]. Algunos trabajos permiten que las puertas PAND se rompan cuando ambas entradas fallan al mismo tiempo [34, 20, 19]. En nuestro caso estamos de acuerdo con esta última opción y decidimos modelar nuestra puerta para poder identificar si ha pasado tiempo entre la ocurrencia de las fallas y actuar en consecuencia. En el caso particular donde no transcurre tiempo entre la falla de las entradas, consideramos que el orden en que fallan los BE dependientes realmente no importa y, por lo tanto, el no determinismo es espurio.

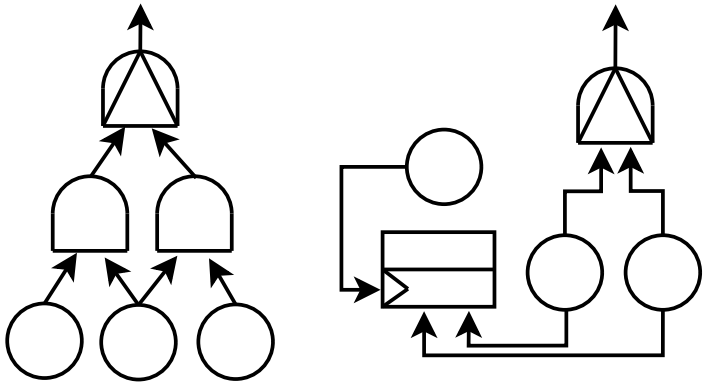


Figura 5.9: no-determinismo espurio.

Para identificar si ha pasado tiempo entre la ocurrencia de las fallas de entrada, nuestro modelo debe escuchar las acciones de salida que indican que un reloj ha expirado. Esto se hace mediante la acción de entrada especial en la línea 8, que se sincronizará con todas las salidas urgentes, sin importar qué acción activen. Tenga en cuenta que solo hay un escenario que queremos descartar, que es cuando la segunda entrada falla y luego pasa el tiempo sin que la primera entrada también falle. De hecho, este es el caso descrito por line 8 guard. Además, esta transición nos lleva al estado "a prueba de fallas" (`st=4`), desde el cual solo podemos volver arreglando la última entrada. En consecuencia, la falla de nuestra puerta ocurre si ambas entradas fallan al mismo tiempo o si falla la primera entrada, luego pasó el tiempo y luego falla la segunda entrada. Una variable `st` nos permite distinguir entre los estados donde la puerta PAND está funcionando (valor 0), desde cuando necesita señalar una falla (valor 1), o cuando falla y ya no hay necesidad de señalar (valor 2), o si necesita señalar que ha sido reparado (valor 3), o finalmente

se encuentra en estado de seguridad, como consecuencia del orden inverso de ocurrencia de las fallas (valor 4).

Repair Box con Prioridad

La semántica de una *RBOX con prioridad de n entradas*, es una función $[\![(\mathsf{rbox}, n)]\!] : \mathcal{A}^{3*n} \rightarrow$ IOSA, donde $[\![(\mathsf{rbox}, n)]\!](\mathit{fl}_0, \mathit{up}_0, r_0, ..., \mathit{fl}_{n-1}, \mathit{up}_{n-1}, r_{n-1})$ es el siguiente IOSA:

```
 1  module RBOX
 2   broken[n]: bool init false;
 3   busy: bool init false;
 4
 5   [ fl_0?] -> (broken[0]'=true);
 6   ...
 7   [ fl_n-1?] -> (broken[n-1]'=true);
 8
 9   [ r_0!!] !busy & broken[0] -> (busy'=true);
10   ...
11   [ r_n-1!!] !busy & broken[n-1] & !broken[n-2]
12                & ... & !broken[0] -> (busy'=true);
13
14   [ up_0?] -> (broken[0]'=false) & (busy'=false);
15   ...
16   [ up_n-1?] -> (broken[n-1]'=false) & (busy'=false);
17  endmodule
```

El cuadro de reparación prioritaria usa una matriz para realizar un seguimiento de las entradas fallidas (`broken[n]`), actualizándolo cuando recibe sus señales de falla (líneas 5 a 7) y señales de aumento (líneas 13 a 15). Al mismo tiempo, cuando no está ocupado, continúa enviando señales de reparación a las entradas rotas (líneas 9 a 12). Tenga en cuenta que en lugar de escuchar las señales de salida urgentes de los BE de entrada, escucha las acciones no urgentes de las transiciones que desencadenan la falla o la reparación. Esto se hace con el único propósito de facilitar el análisis de confluencia sobre este módulo.

Repair Box con Política *First Come First Serve*

La semántica de una *Repair Box con política first come first serve* de n entradas es una función $[\![(\mathsf{rbox}, n)]\!] : \mathcal{A}^{3*n} \rightarrow$ IOSA, where $[\![(\mathsf{rbox}, n)]\!](\mathit{fl}_0, \mathit{up}_0, r_0, ..., \mathit{fl}_{n-1}, \mathit{up}_{n-1}, r_{n-1})$ es el siguiente IOSA:

```
1  module RBOX % with first come first serve policy
2    queue[n]: [0..n] init 0;
3    busy: bool init false;
4    r: [0..n] init n;
5    dummy: [0..0] init 0;
6
7    [ fl_0?] -> (dummy'=broken(queue,0));
8    ...
9    [ fl_{n-1}?] -> (dummy'=broken(queue,n-1));
10
11   [!!] some(queue,0) & r=n -> (r'=maxfrom(queue,0));
12
13   [ r_0!!] !busy & r=0 -> (busy'=true) & (queue[0]'=0);
14   ...
15   [ r_{n-1}!!] !busy & r=n-1 -> (busy'=true) & (queue[n-1]'=0);
16
17   [ up_0?] -> (queue[0]'=false) & (busy'=false) & (r' = n);
18   ...
19   [ up_{n-1}?] -> (queue[n-1]'=false) & (busy'=false) & (r' = n);
20 endmodule
```

El modelo para una caja de reparación con política de orden de llegada utiliza una matriz para marcar cada entrada rota. Observe que cada posición en la cola corresponde a cada entrada. Un valor 0 en un índice i significa que la entrada i no ha fallado, mientras que un valor mayor en esa posición indica "cuánto tiempo" ha estado rota. Los cuadros de reparación usan algunos elementos sintácticos presentes en la herramienta de simulación FIG [25]. Estos elementos no introducen un nuevo comportamiento semántico y están ahí solo para reducir la complejidad y ofuscación que representaría modelar esto usando solo la gramática presentada en el Apéndice A. Ejemplos de esto son la función `broken` en la línea 7, que dada una matriz, en este caso `queue`, y un índice, en este caso `0`, aumenta en uno el valor en ese índice y cualquier otro valor superior a 0 en la matriz. De esta forma podemos comprobar el orden en que fallaron las entradas comparando los valores en el índice correspondiente. Cuanto mayor era el valor, antes se rompían. La función sintáctica `some`, por otro lado, devuelve un valor booleano que indica si hay algún valor diferente a cero en la matriz. En este caso lo usamos para comprobar si hay alguna entrada fallida. Si hay al menos uno, entonces la función `maxfrom` devolverá el índice del valor más alto en `queue`, que corresponde a la entrada que rompió primero entre todos los rotos. La variable `dummy` se usa solo para cumplir con las restricciones sintácticas de las asignaciones de

IOSAs. Para un análisis rápido de determinismo, señalamos que todos los `broken`, `fstexclude` y `maxfrom` son deterministas. Además, todos los pares de transiciones urgentes en el modelo son confluentes dado que sus guardias son mutuamente excluyentes debido al valor de la variable `r`.

Un Modelo General para RBOX

El siguiente modelo se puede usar para modelar muchas cajas de reparación diferentes con políticas arbitrarias:

```
1  module RBOX % general policy
2    queue[n]: [0..n] init 0;
3    busy: bool init false;
4    r: [0..n] init n;
5
6    [ fl_0?] -> (dummy'=broken(queue,0));
7    ...
8    [ fl_{n-1}?] -> (dummy'=broken(queue,n-1));
9
10   [!!] some(queue) & r=n -> (r'=policy(queue));
11
12   [ r_0!!] !busy & r=0 -> (busy'=true) & (queue[0]'=0);
13   ...
14   [ r_{n-1}!!] !busy & r=n-1 -> (busy'=true) & (queue[n-1]'=0);
15
16   [ up_0?] -> (queue[0]'=false) & (busy'=false) & (r' = n);
17   ...
18   [ up_{n-1}?] -> (queue[n-1]'=false) & (busy'=false) & (r' = n);
19 endmodule
```

Tenga en cuenta que en este modelo general de RBOX tomamos nota no solo de las entradas fallidas sino también de su orden de falla. Esto se hace usando la función `broken` ya explicada, junto con una matriz compatible (`queue`). Se puede obtener un modelo más simple si no nos interesa el orden en que fallaron. Definir cómo `policy` selecciona la siguiente entrada para reparar de las entradas rotas marcadas en `queue` producirá el RBOX deseado. En nuestro caso, por supuesto, solo nos interesan las políticas deterministas; de lo contrario, esta función haría que el modelo no fuera determinista.

Semántica para RFT

La semántica de un RFT es la de la composición paralela de la semántica de sus componentes, estando convenientemente sincronizados.

Definición 5.3. Dado un RFT $T = (V, i, si, l)$ definimos la semántica de T como $[\![T]\!] = ||_{v \in V} [\![v]\!]$ donde $[\![v]\!]$ esta definido por:

$$[\![v]\!] = \begin{cases} [\![l(v)]\!](\mathtt{fl}_v, \mathtt{up}_v, \mathtt{f}_v, \mathtt{u}_v, \mathtt{r}_v) & \text{if } l(v) = (\mathsf{be}, 0, \mu, \gamma) \\ [\![l(v)]\!](\mathtt{f}_v, \mathtt{u}_v, \mathtt{f}_{i(v)[0]}, \mathtt{u}_{i(v)[0]}, ..., \mathtt{f}_{i(v)[n-1]}, \mathtt{u}_{i(v)[n-1]}) & \\ & \text{if } l(v) \in \{(\mathsf{and}, n), (\mathsf{or}, n)\} \\ [\![l(v)]\!](\mathtt{f}_v, \mathtt{u}_v, \mathtt{f}_{i(v)[0]}, \mathtt{u}_{i(v)[0]}, \mathtt{f}_{i(v)[1]}, \mathtt{u}_{i(v)[1]}) & \text{if } l(v) = (\mathsf{pand}, 2) \\ [\![l(v)]\!](\mathtt{fl}_{i(v)[0]}, \mathtt{up}_{i(v)[0]}, \mathtt{r}_{i(v)[0]}, ..., \mathtt{fl}_{i(v)[n-1]}, \mathtt{up}_{i(v)[n-1]}, \mathtt{r}_{i(v)[n-1]}) & \\ & \text{if } l(v) = (\mathsf{rbox}, n) \end{cases}$$

Note que la función i se usa para sincronizar convenientemente cada entrada. De hecho, $i(v)$ es una lista ordenada de las entradas de v. Supongamos como ejemplo que v es una puerta AND, y que el BE w es la primera entrada de v. Entonces el nombre asignado a la señal de falla de w será f_w, ya que $l(w) = (\mathsf{be}, 0, \mu, \gamma)$ (ver Definición **??**). Convenientemente, el nombre de la primera señal de entrada fallida de v se llamará f_w, ya que $lv = (\mathsf{and}, n)$ y $i(v) = [w, ...]$ y por lo tanto $i(v)[0] = w$. En la Sección 5.7, ampliamos la semántica a puertas de repuesto y SBE.

5.6. Los RFTs son deterministas

En esta sección mostramos que RFTs compuestas solo por BEs, compuertas AND, compuertas OR, compuertas PAND y RBOX, son débilmente deterministas. Hacemos uso de los resultados sobre IOSA dados en la Sección 4.5. Dado que las puertas de votación y FDEP se pueden construir usando puertas OR y AND, también estamos demostrando que se pueden modelar de manera determinista. Trabajaremos bajo la premisa de que podemos construir el espacio de estados alcanzables para cada componente en un tiempo y espacio razonables. Esta no es una suposición extraña ya que la composicionalidad nos permite mantener los componentes lo suficientemente pequeños. En las siguientes proposiciones enumeramos los conjuntos de acciones habilitadas inicial y espontáneamente en una RFT. Posteriormente analizamos los posibles casos de acciones no confluentes en un RFT, y además describimos su relación aproximada de desencadenamiento indirecto. Con todos estos ingredientes podemos finalmente probar que RFTs son débilmente deterministas aplicando el Teorema 4.5.

Proposición 5.1. Sea T un RFT. $[\![T]\!]$ no tiene acciones habilitadas inicialmente. Además, los únicos conjuntos espontáneos de acciones son singletones con forma $\{f_v\}$ y $\{u_v\}$, para $\mathrm{t}(v) = \mathsf{be}$, que se habilitan espontáneamente por fl_v y up_v, respectivamente.

Demostración. Como consecuencia de la Proposición 4.6, las acciones inicialmente habilitadas de $[\![T]\!]$ están contenidas en la unión de los conjuntos de acciones inicialmente habilitadas de sus componentes $[\![v]\!]$, $v \in V$, y las acciones espontáneamente habilitadas de $[\![T]\!]$ están contenidas en la unión de los conjuntos espontáneamente habilitados de $[\![v]\!]$ ps Es directo ver que, para cualquier elemento $e \in \mathcal{E}$, ninguna de las salidas urgentes está habilitada en el estado inicial de $[\![e]\!]$, ya que sus guardias son inicialmente falsas. Además, la única transición de salida no urgente en nuestros modelos se encuentra en las líneas 6 y 8 del BE (Figura 5.6). Sea $v \in V$ tal que $\mathrm{t}(v) = \mathsf{be}$. Luego, después de tomar la transición en la línea 6, la única salida urgente habilitada es $\mathtt{f}_v$ (en la instancia $[\![v]\!]$), mientras que después de tomar la transición en la línea 8, la única salida es $\mathtt{u}_v$, por lo que estas son las únicas acciones habilitadas espontáneas posibles. □

Proposición 5.2. Sea T un RFT. Los únicos pares posibles de acciones no confluentes en $[\![T]\!]$ son:

- $\{(\mathtt{f}_v, \mathtt{u}_{v'}) \mid v, v' \in i(w), t(w) \in \{\mathsf{and}, \mathsf{or}, \mathsf{pand}\}\}$, y
- $\{(\mathtt{f}_w, \mathtt{u}_v), (\mathtt{u}_w, \mathtt{f}_v) \mid v \in i(w), t(w) \in \{\mathsf{and}, \mathsf{or}\}\}$.

Demostración. La composición paralela no introduce nuevos pares de acciones no confluentes y, además, preserva la confluencia de sus componentes (Proposición 4.2). Por lo tanto, miramos los componentes de forma aislada. Primero observe que las transiciones en un módulo IOSA se definen simbólicamente. Cada transición simbólica en un módulo describe, de hecho, un conjunto de transiciones IOSA que se concretan cuando la transición simbólica se evalúa en un estado que satisface la guardia. Observe también que un estado en un módulo está definido por los valores actuales de sus variables. Al analizar que dos acciones urgentes a y b son confluentes en un módulo, por cada transición simbólica t_a y t_b definida para esas acciones en ese módulo, buscamos un *testigo de no confluencia* , es decir, un estado que satisface las protecciones de t_a y t_b y muestra que a y b no son confluentes (es decir, el par no satisface Def. 4.5). Tenga en cuenta que al verificar solo los estados alcanzables en el componente, ya estamos sobreaproximando los estados alcanzables en la composición.

Para esta prueba solo analizamos el caso de la puerta AND. Para otros elementos RFT la demostración es similar. Sea v un vértice en un RFT tal que $l(v) = (\mathsf{and}, 2)$.

Analizamos $\mathtt{f}_1$ contra $\mathtt{u}_1$ en $[\![(\mathsf{and}, 2)]\!]$ (ver la semántica de AND) y demostramos que no son confluentes . Tomemos, por ejemplo, el estado s definido por `count=1`, `signalf=false` y `signalu=false`, que se puede verificar fácilmente para que sea accesible. Allí, encontramos que permite transiciones simbólicas en las líneas 6 (con etiqueta $\mathtt{f}_1$) y 14 (con etiqueta $\mathtt{u}_1$). Por un lado, la transición en la línea 6 se mueve al estado donde se alcanza `count=2`, `signalf=true` y `signalu=false`. En este punto, la acción $\mathtt{u}_1$ solo se puede realizar a través de la transición en la línea 13, lo que produce el estado s' definido por `count=1`, `signalf=true` y `signalu=verdadero`. Por otro lado, la transición en la línea 14 se mueve al estado donde `count=0`, `signalf=false` y `signalu=false`. Este estado solo habilita $\mathtt{f}_1$ en la línea 7, lo que produce el estado s'' definido por `count=1`, `signalf=false` y `signalu=false` . Dado que s' y s'' son dos estados diferentes, hemos demostrado que $\mathtt{f}_1$ y $\mathtt{u}_1$ no son confluentes. De manera similar, podemos mostrar que los pares $(\mathtt{f}, \mathtt{u}_i)$ y $(\mathtt{u}, \mathtt{f}_i)$, para $i = 1, 2$, no son confluentes.

Todos los demás pares son confluentes. Tomemos como ejemplo las transiciones en las líneas 7 y 10 que están definidas para las acciones $\mathtt{f}_1$ y $\mathtt{f}_2$ respectivamente, y el estado s definido por `count=0` , `signalf=false` y `signalu=false`. Por un lado, la línea 7 conduce al estado donde `count=1`, `signalf=false` y `signalu=false` que a su vez habilita $\mathtt{f}_2$ solo en línea 9 que produce el estado s' definido por `count=2`, `signalf=true` y `signalu=false`. Por otro lado, la línea 10 en el estado s se mueve al estado donde `count=1`, `signalf=false` y `signalu=false` que solo habilita $\mathtt{f}_1$ en la línea 6 dando el mismo estado s'. La prueba se sigue de manera similar desde cualquier otro estado alcanzable que permita $\mathtt{f}_1$ y $\mathtt{f}_2$ mostrando, por lo tanto, que $\mathtt{f}_1$ y $\mathtt{f}_2$ son confluente. En algunos otros casos, la prueba de confluencia se deriva del hecho de que el par de acciones nunca se habilitan simultáneamente, como es el caso, por ejemplo, de `f` y `u` (nótese que los guardias habilitan cada uno de ellos son mutuamente excluyentes). □

Proposición 5.3. Sea T un RFT. Para cada $v \in V$, la triggering relation de $[\![v]\!]$ está dada por:

- $\{\}$, if $l(v) \in \{(\mathsf{be}, 0, \mu, \gamma), (\mathsf{rbox}, n)\}$,
- $\{(\mathtt{f}_w, \mathtt{f}_v) \mid w \in i(v)\} \cup \{(\mathtt{u}_w, \mathtt{u}_v) \mid w \in i(v)\}$, si $l(v) \in \{(\mathsf{and}, n), (\mathsf{or}, n)\}$, y
- $\{(\mathtt{u}_w, \mathtt{u}_v) \mid w = i(v)[1]\} \cup \{(\mathtt{f}_w, \mathtt{f}_v) \mid w \in i(v)\}$, si $l(v) = (\mathsf{pand}, 2)$.

Proof (sketch). Basta con hacer un análisis de satisfacibilidad sobre las guardas y postcondiciones de cada par (t_a, t_b) con t_b una transición simbólica

urgente de salida y t_a cualquier transición simbólica urgente, teniendo en cuenta solo los estados alcanzables. □

Teorema 5.1. Sea T un RFT. Entonces $[\![T]\!]$ es débilmente determinista.

Demostración. Buscamos a, b, c, d y e, así como conjuntos B_i con $i = 1 \dots n$ como sugiere el Teorema 4.5. Dado que Prop. 5.1 garantiza que no haya acciones habilitadas inicialmente en $[\![T]\!]$, c y d deberían ser acciones habilitadas espontáneamente. Por la misma proposición, e tiene la forma $\mathtt{fl}_v$ para algunos v y luego $\bigcup_{i=1}^{1} B_i = B_1 = \{\mathtt{f}_v\}$, o e tiene la forma $\mathtt{up}_v$ para algunos v y luego $\bigcup_{i=1}^{1} B_i = B_1 = \{\mathtt{u}_v\}$. En el primer caso, obtenemos $c = d = \mathtt{f}_v$ por unos v, y en el segundo caso $c = d = \mathtt{u}_v$. Además, por Prop. 5.2, a tiene la forma $\mathtt{f}_w$ para algunos w y b tiene la forma $\mathtt{u}_{w'}$ por algo de w' o al revés. Como se muestra en Prop. 5.3, las acciones fallidas ($\mathtt{f}_v$ para algunos v) solo desencadenan acciones fallidas y acciones ascendentes ($\mathtt{u}_v$ para algunos v) solo activan acciones, por lo que es imposible que c y d activen indirectamente a y b respectivamente. Por lo tanto, no es posible encontrar acciones a, b, c, d y e que satisfagan las condiciones 1 a 3 del Teorema 4.5 , y por lo tanto $[\![T]\!]$ es confluente. Dado que $[\![T]\!]$ también es cerrado, entonces es débilmente determinista. □

5.7. Seántica extendida

En esta sección ampliamos la semántica de RFTs introduciendo puertas de repuesto y elementos básicos de repuesto. Como antes, nuestro objetivo es garantizar que los modelos de IOSA derivados de RFT sean débilmente deterministas. Para hacerlo, debemos prestar especial atención a dos escenarios particulares que podrían introducir el no determinismo.

El primer escenario surge cuando un elemento básico principal falla en una puerta de repuesto que se sirve con varios elementos básicos de repuesto. Llegados a este punto, nos queda la duda de cuál de los elementos básicos de repuesto disponibles debe llevarse la puerta de repuesto. Tradicionalmente, los elementos de repuesto se seleccionan en orden de un conjunto ordenado. Para generalizar este mecanismo de selección de repuestos pretendemos permitir políticas más complejas de participación estatal. Debe darse siempre el caso de que esta política opte de manera determinista. El segundo escenario surge cuando varias puertas de repuesto solicitadas están disponibles SBE, y están rotas u ocupadas por otra solicitud de repuesto. La situación no determinista surgirá cuando el SBE sea reparado o liberado por la puerta de reserva de retención, respectivamente. En este punto, no está claro cuál de las puertas de repuesto solicitantes tomará el SBE recientemente disponible.

Para ello, definimos políticas de compartición en el SBE. Por lo tanto, para proporcionar semántica a un SBE, en realidad introducimos dos módulos IOSA uno que extiende el comportamiento de un BE con la posibilidad de cambiar de estado inactivo a estado habilitado y viceversa, y otro, el llamado *módulo multiplexor*, que gestiona la compartición del SBE.

El Spare Basic Element (SBE)

La semántica de un *SBE* es una función $[\![(\mathsf{sbe}, n, \mu, \nu, \gamma)]\!] : \mathcal{A}^{7+5*n} \rightarrow$ IOSA, donde $[\![(\mathsf{sbe}, n, \mu, \nu, \gamma)]\!](\mathit{fl}, \mathit{up}, f, u, r, e, d, \mathit{rq}_0, \mathit{asg}_0, \mathit{rel}_0, \mathit{acc}_0, \mathit{rj}_0, ..., \mathit{rq}_{n-1}, \mathit{asg}_{n-1}, \mathit{rel}_{n-1}, \mathit{acc}_{n-1}, \mathit{rj}_{n-1})$ resulta en el siguiente par de módulos IOSA:

```
1  module SBE
2   fc, dfc, rc : clock;
3   inform : [0..2] init 0;
4   active : bool init false;
5   broken : [0..2] init 0;
6
7   [ e??] !active -> (active'=true) & (fc'=μ);
8   [ d??] active -> (active'=false) & (dfc'=ν);
9
10  [ fl!] active & broken=0 @ fc -> (inform'=1) & (broken'=1);
11  [ fl!] !active & broken=0 @ dfc -> (inform'=1) & (broken'=1);
12  [ r??] -> (broken'=2) & (rc'=γ);
13  [ up!] active & broken=2 @ rc -> (inform'=2) & (broken'=0) & (fc'=μ);
14  [ up!] !active & broken=2 @ rc -> (inform'=2) & (broken'=0) & (dfc'=μ);
15
16  [ f!!] inform=1 -> (inform'=0);
17  [ u!!] inform=2 -> (inform'=0);
18 endmodule
```

```
1  module MUX
2   queue[n]: [0..3] init 0; % idle, requesting, reject, using
3   avail: bool init true;
4   broken: bool init false;
5   enable: [0..2] init 0;
6
7   [ fl?] -> (broken'=true);
8   [ up?] -> (broken'=false);
9
10  [ e!!] enable=1 -> (enable'=0);
11  [ d!!] enable=2 -> (enable'=0);
12
13  [ rq0??] queue[0]=0 & (broken | !avail) -> (queue[0]'=2);
14  [ rq0??] queue[0]=0 & !broken & avail -> (queue[0]'=1);
15  [ asg0!!] queue[0]=1 & !broken & avail -> (queue[0]'=3) & (avail'=false);
16  [ rj0!!] queue[0]=2 -> (queue[0]'=1);
17  [ rel0??] queue[0]=3 -> (queue[0]'=0) & (avail'=true) & (enable'=2);
18  [ acc0??] -> (enable'=1);
19  ...
20  [ rqn−1??] queue[n-1]=0 & (broken | !avail) -> (queue[n-1]'=2);
```

```
21   [ rq_{n-1}??] queue[n-1]=0 & !broken & avail -> (queue[n-1]'=1);
22   [ asg_{n-1}!!] queue[n-1]=1 & queue[n-2]=0 & ... & queue[0]=0 & !broken & avail
23                  -> (queue[n-1]'=3) & (avail'=false);
24   [ rj_{n-1}!!] queue[n-1]=2 -> (queue[n-1]'=1);
25   [ rel_{n-1}??] queue[n-1]=3 -> (queue[n-1]'=0) & (avail'=true) & (enable'=2);
26   [ acc_{n-1}??] -> (enable'=1);
27
28 endmodule
```

Además del nuevo modelo MUX, el modelo de SBE difiere del modelo de BE dado anteriormente en la introducción de un nuevo reloj de fallas (`dfc`) que controla las fallas en el modo deshabilitado, una nueva variable (`enabled`) que distingue entre estados habilitados y deshabilitados, y algunas líneas reflejadas de alguna manera para distinguir entre estados activos y deshabilitados y actuar en consecuencia (líneas 11 y 12, 14 y 16). Además se introducen dos nuevas acciones y sus correspondientes transiciones para poder habilitar o deshabilitar el SBE cuando sea necesario (líneas 8 y 9).

En el caso del multiplexor, decidimos modelarlo con una política de prioridad, que prioriza las puertas de repuesto de entrada de índice más bajo a las de índice más alto (observe las transiciones de asignación en la línea 16 y 25 del módulo multiplexor). Otros tipos de políticas pueden definirse como para puertas de caja de reparación. En el modelo, las acciones rq_i indican que la entrada de puerta de repuesto i está solicitando el repuesto. acc_i indica que la entrada i acepta el repuesto que se le ha asignado previamente a través de la acción asg_i. Por otro lado la acción rj_i indica que lo rechaza. La acción rel_i indica que la entrada i está liberando el repuesto que previamente se le asignó a dicha entrada. Finalmente, las acciones e y d habilitan y deshabilitan el elemento básico de repuesto cuando sea necesario.

Tenga en cuenta que no se habría necesitado ningún multiplexor en ausencia de cajas de reparación. Dado que en tales casos, los SBE no están disponibles después de que se toman o fallan, no habría necesidad de resolver ningún no determinismo. No habría surgido ningún no determinismo si los elementos de repuesto no fueran compartidos por diferentes puertas de repuesto [12, 11]. Otra posible situación de no determinismo habría sido las condiciones de carrera entre dos puertas de repuesto que fallan al mismo tiempo, como se estudió en [69]. Sin embargo, hemos descartado estas condiciones de carrera de nuestra semántica introduciendo las dos últimas condiciones de Definición 5.2 junto con el hecho de que dos fallos simultáneos de elementos básicos no son posibles en la semántica determinista IOSA.

La Spare Gate (SG)

La semántica de una *spare gate con política de prioridad* es una función $[\![(\mathsf{sg}, n)]\!] : \mathcal{A}^{4+7*n} \rightarrow$ IOSA, donde $[\![(\mathsf{sg}, n)]\!](f, u, fl_0, up_0, fl_1, up_1, rq_1, asg_1, acc_1, rj_1, rel_1 ..., fl_n, up_n, rq_n, asg_n, acc_n, rj_n, rel_n)$ es el siguiente IOSA:

```
1  module SPAREGATE
2    state: [0..4] init 0; // on main, request, wait, on spare, broken
3    inform: [0..2] init 0;
4    release: [-n..n] init 0;
5    idx: [1..n] init 1;
6
7    [ fl0?] state=0 -> (state'=1) & (idx'=1);
8    [ up0?] state=4 -> (state'=0) & (inform'=2);
9    [ up0?] state=3 & idx=1 -> (state'=0) & (idx'=1) & (release'=1);
10   ...
11   [ up0?] state=3 & idx=n -> (state'=0) & (idx'=1) & (release'=n);
12
13   [ fl1?] state=3 & idx=1 -> (release'=1);
14   ...
15   [ fln?] state=3 & idx=n -> (release'=n);
16
17   [ rq1!!] state=1 & idx=1 -> (state'=2);
18   ...
19   [ rqn!!] state=1 & idx=n -> (state'=2);
20
21   [ asg1??] state=0 | state=1 | state=3 -> (release'=1);
22   [ asg1??] state=2 & idx=1 -> (release'=-1) & (state'=3);
23   [ asg1??] state=4 -> (release'=-1) & (state'=3) & (idx'=1) & (inform'=2);
24   ...
25   [ asgn??] state=0 | state=1 | state=3 -> (release'=n);
26   [ asgn??] state=2 & idx=n -> (release'=-n) & (state'=3);
27   [ asgn??] state=4 -> (release'=-n) & (state'=3) & (idx'=n) & (inform'=2);
28
29   [ rj1??] state=2 & idx=1 -> (idx'=2) & (state'=1);
30   [ rj2??] state=2 & idx=2 -> (idx'=3) & (state'=1);
31   ...
32   [ rjn??] state=2 & idx=n -> (state'=4) & (idx'=1) & (inform'=1);
33
34   [ rel1!!] release=1 & !(state=3 & idx=1) -> (release'= 0);
35   [ rel1!!] release=1 & state=3 & idx=1 -> (release'= 0) & (state'=1) & (idx'=1);
36   ...
37
38   [ reln!!] release=n & !(state=3 & idx=n) -> (release'=0);
39   [ reln!!] release=n & state=3 & idx=n -> (release'= 0) & (state'=1) & (idx'=1);
40
41   [ acc1!!] release=-1 -> (release'= 0);
42   ...
43   [ accn!!] release=-n -> (release'=0);
44
45   [ f!!] inform = 1 -> (inform'=0);
46   [ u!!] inform = 2 -> (inform'=0);
47 endmodule
```

El modelo de la puerta de repuesto utiliza una política de prioridad sobre los BE de repuesto disponibles. Esto significa que al buscar un SBE, comenzará a pedirlo desde la entrada de índice más bajo hasta la entrada

de índice más alto hasta obtener un reemplazo. También se pueden definir otras políticas en la puerta de repuesto, al igual que con el multiplexor y la caja de reparación. En el modelo de un SG, la variable `state` distingue si el SG está trabajando con su BE principal, solicitando un SBE, esperando una respuesta de sus entradas, trabajando en un SBE o roto. El vector `release` indica para cada entrada de SBE i cuando el SG tiene que liberar (valor i) o aceptar (valor $-i$) la asignación de ese SBE. La variable `idx` indica cuál de las entradas solicitar a continuación. La línea 7 define la transición que comienza con el protocolo de adquisición SBE cada vez que falla el BE principal. Las siguientes transiciones hasta la línea 15 liberan los SBE adquiridos cada vez que fallan o se repara el BE principal. Las transiciones de las líneas 17 a 19 solicitan el siguiente SBE posible. Después de hacerlo, debemos esperar una respuesta del multiplexor correspondiente (`state'=2`). La solicitud puede ser rechazada (líneas 29 a 32), y se procede solicitando el siguiente SBE configurando `idx` al valor correspondiente si lo hay, o fallando en caso de que ninguno de los SBE esté disponible (códigoestado'=4 en la línea 32). Se puede asignar un SBE al SG cuando ya no se necesite (líneas 21 y 25), o cuando el SG lo haya solicitado para evitar fallar (líneas 22 y 26), o cuando el SG ya haya fallado y así se repara usándolo (líneas 23 y 27). I SG puede querer liberar un SBE cuando se le asigna pero no necesita el SBE (líneas 34 y 38) o cuando el SBE falla mientras el SG lo está usando (líneas 35 y 39). El SG puede aceptar SBE asignados en las líneas 41 a 43. Finalmente, el SG señala falla en la línea 45 y reparación en la línea 46. Para comprender mejor el significado y la intuición de cada transición, remitimos al lector a la descripción de SBE.

Semántica RFTExtendida

Extendemos la semántica de RFT con los elementos SBE y SG como sigue.

Definición 5.4. Dado un RFT $T = (V, E)$, extendemos la Definición 5.3

con los siguientes casos:

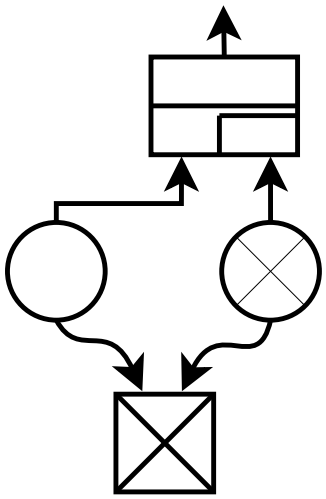
$$[\![v]\!] = \begin{cases} \cdots \\ [\![l(v)]\!](\mathtt{fl}_v, \mathtt{up}_v, \mathtt{f}_v, \mathtt{u}_v, \mathtt{r}_v, \mathtt{e}_v, \mathtt{d}_v, \mathtt{rq}_{(si(v)[0],v)}, \mathtt{asg}_{(v,si(v)[0])}, \\ \qquad \mathtt{rel}_{(si(v)[0],v)}, \mathtt{acc}_{(si(v)[0],v)}, \mathtt{rj}_{(v,si(v)[0])}, .., \mathtt{rj}_{(v,si(v)[n-1])}) \\ \qquad\qquad \text{if } l(v) = (\mathsf{sbe}, n, \mu, \nu, \gamma) \\ [\![l(v)]\!](\mathtt{f}_v, \mathtt{u}_v, \mathtt{fl}_{i(v)[0]}, \mathtt{up}_{i(v)[0]}, \mathtt{fl}_{i(v)[1]}, \mathtt{up}_{i(v)[1]}, \mathtt{rq}_{(v,i(v)[1])}, \mathtt{asg}_{(i(v)[1],v)}, \\ \qquad \mathtt{acc}_{(v,i(v)[1])}, \mathtt{rj}_{(i(v)[1],v)}, \mathtt{rel}_{(v,i(v)[1])}, ..., \mathtt{rel}_{(v,i(v)[n-1])}) \\ \qquad\qquad \text{if } l(v) = (\mathsf{sg}, n) \end{cases}$$

Observe que en el caso de SBE y SG, varias señales están indexadas por un par de elementos. Este par indica quién realiza la acción y quién la escucha en la sincronización. Como ejemplo, $as_{(v,i(v)[0])}$ indicará que el multiplexor que administra v, asigna su elemento básico de repuesto a su primera puerta de repuesto conectada ($i(v)[0]$).

Determinismo

Desafortunadamente, no pudimos encontrar una forma directa de demostrar que esta extensión es determinista débil, como hicimos con el RFT sin repuestos. Si bien se puede demostrar fácilmente que el módulo SBE es confluente, este no es el caso de los modelos del multiplexor y la puerta de repuesto. De hecho, los modelos con puertas de repuesto y SBE son casos de falso positivo para el teorema 4.5 con respecto al determinismo débil. Supongamos, por ejemplo, un modelo simple de una puerta de repuesto de una sola entrada conectada a su BE correspondiente y a un SBE de una sola entrada, con ambos elementos básicos conectados al mismo RBOX (Figura ref fig: falso positivo). En tal modelo, las acciones $\mathtt{asg}_1$ y $\mathtt{rj}_1$ no son confluentes en el modelo de puerta de repuesto, y $\mathtt{rq}_1$ se habilita espontáneamente por $\mathtt{fl}_0$ en el mismo modelo. Además, $\mathtt{rq}_1$ activa de forma aproximadamente indirecta tanto $\mathtt{asg}_1$ como $\mathtt{rj}_1$ en el modelo de multiplexor de la primera entrada a dicha puerta de repuesto. Por lo tanto, no se cumplen las condiciones del teorema 4.5. Sin embargo, se da el caso de que, después de la composición, dicho modelo cerrado no tiene acciones no conflictivas y, por lo tanto, es débilmente determinista.

Aunque como se ha dicho, no hemos podido probar un determinismo débil en la generalidad de las combinaciones entre puertas de repuesto y SBE, hemos probado la confluencia para algunas composiciones particulares de estos modelos por medio de un programa simple en Python. Enumeramos estos resultados:



$(V = \{v_0, v_1, v_2, v_3\}$
$, i = \{(v_0, []), (v_1, []), (v_2, [v_0, v_1]), (v_3, [v_0, v_1])\}$
$, si = \{(v_1, [v_2])\}$
$, l = \{(v_0, \mathsf{be}), (v_1, \mathsf{sbe}), (v_2, \mathsf{sg}), (v_3, \mathsf{rbox})\})$

Figura 5.10: Modelo con falso positivo para Teo. 4.5

- Todas las combinaciones de hasta 3 puertas de repuesto conectadas a hasta 3 SBE son confluentes (ejemplos en la Figura 5.11).
- Un modelo con una única puerta de repuesto conectada a hasta 8 SBE es confluente (Figura 5.12).
- Un modelo con un único SBE compartido por hasta 8 puertas de repuesto es confluente (Figura 5.12).

La explosión del espacio de estados limita nuestras posibilidades de buscar combinaciones más grandes. La herramienta de simulación de eventos raros FIG (http://dsg.famaf.unc.edu.ar/fig) también admite la verificación de confluencia y, por lo tanto, se puede usar para verificar otras combinaciones, que luego se pueden usar de manera segura como partes de modelos más grandes

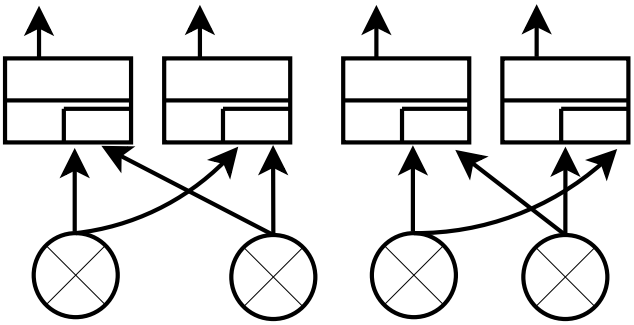


Figura 5.11: Configuraciones confluentes

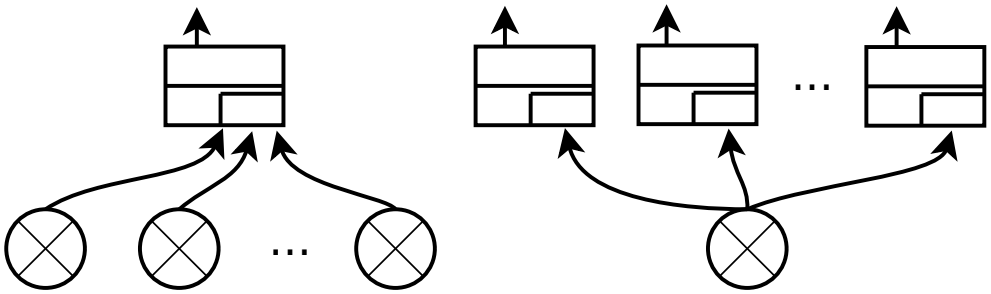


Figura 5.12: Configuraciones confluentes

5.8. Análisis de RFT con el simulador FIG

IOSA$_u$ pretende ser un medio útil y riguroso para modelar sistemas distribuidos continuos generales con el propósito de realizar análisis formales mediante técnicas de simulación. En este capítulo presentamos un ejemplo de aplicación. Para ello primero presentaremos brevemente el simulador FIG, desarrollado en FAMAF-UNC [25, 41] (http://dsg.famaf.unc.edu.ar/fig). Luego presentamos un ejemplo de juguete de un sistema de enfriamiento y repasamos todos los pasos desde la definición del modelo y las propiedades de interés hasta el examen de los resultados proporcionados por la herramienta. El simulador FIG está especialmente diseñado para analizar sistemas de eventos raros, es decir, sistemas donde la probabilidad de ocurrencia de la propiedad de interés es muy pequeña. Por lo tanto, configuramos nuestro modelo de ejemplo RFT para que la probabilidad de su evento principal sea muy pequeña.

5.8.1. Simulación de Eventos Raros y el Simulador FIG

FIG significa *Finite Improbability Generator* como un homenaje a la obra maestra de Douglas Adam. Es una herramienta de simulación de eventos discretos adaptada a las propiedades de eventos raros y está disponible gratuitamente en (http://dsg.famaf.unc.edu.ar/fig). La herramienta ha sido desarrollada por el grupo de sistemas confiables de FAMAF UNC.

La alta resiliencia y confiabilidad requerida por el sistema actual se traduce en analizar propiedades que fallan con una probabilidad extremadamente pequeña. La complejidad de los modelos hace que el análisis sea computacionalmente muy exigente. La simulación estándar de Monte Carlo requiere una enorme cantidad de muestreo para adquirir un nivel de confianza significativo en la probabilidad estimada, a fin de compensar la alta varianza inducida por las raras ocurrencias de tal evento. Esto hace que esta técnica sea extremadamente ineficiente. Se han estudiado algunas optimizaciones sobre la simulación Monte Carlos para tratar eventos raros. Una de estas técnicas, la implementada por FIG, se llama *Importance Splitting*, más precisamente el llamado método RESTART [100, 101]. La división de importancia (IS para abreviar) tiene como objetivo acelerar la ocurrencia de un evento raro sin modificaciones en la dinámica del sistema [53, 31]. La idea general de IS es favorecer las "corridas prometedora" que se acercan al evento raro guardando los estados que visitan en ciertos puntos de control predefinidos. Para hacerlo, IS divide el espacio de estados en niveles ascendentes, donde idealmente a medida que la corrida sube de un nivel a otro, la probabilidad de alcanzar el evento raro aumenta considerablemente. La estimación de la probabilidad rara se obtiene como el producto de las estimaciones de las probabilidades condicionales (no tan raras) de subir un nivel. La eficacia de la técnica depende en gran medida de una óptima agrupación de los estados. La *función de importancia* se encarga de dicha tarea. Antiguamente, la tarea de definir una función de importancia se le daba al ingeniero que modelaba el sistema oa cualquier experto capaz de hacerlo. La herramienta FIG es capaz de construir una función de importancia automáticamente a partir del modelo [31, 28], convirtiendo todo el proceso de verificación en una técnica de botón, una vez que se han descrito el modelo y las propiedades de interés. Al hacerlo, FIG combina la función de impor-

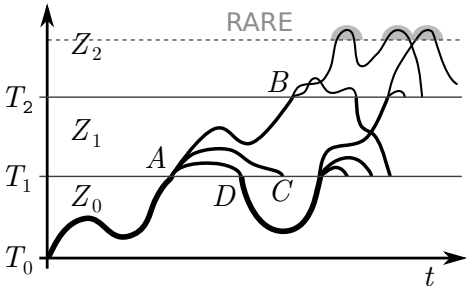


Figura 5.13: Importance Splitting

tancia de la función de importancia local para cada componente, superando los problemas surgidos de la explosión del espacio de estado.

5.8.2. Caso de Estudio de Sistema de Refrigeración por Agua

En esta sección se presenta un modelo de un Sistema de Refrigeración por Agua. Este sistema se encarga de enfriar una cámara de alta presión haciendo circular agua a su alrededor. El sistema consta de dos subsistemas replicados, cada uno compuesto por un sistema de refrigeración por agua principal y un interruptor. Ambos subsistemas comparten una bomba de agua auxiliar de repuesto como mecanismo para aumentar su tolerancia a fallos mediante la replicación. Si ambos subsistemas fallan, entonces el sistema de enfriamiento falla ya que no pasará agua alrededor de la cámara. En un escenario favorable, si una bomba de agua principal falla, la bomba auxiliar la reemplazaría automáticamente, mientras que el interruptor cambiaría las tuberías de agua para redirigir el agua de la tubería auxiliar a la sección de enfriamiento y el subsistema seguiría funcionando. Sin embargo, se pueden encontrar algunos escenarios menos favorables, donde la bomba auxiliar está rota o ya está ocupada por el otro subsistema, o el interruptor está roto y, por lo tanto, el subsistema en cuestión no tiene forma de dirigir las tuberías.

La figura 5.4 muestra la descripción gráfica del modelo RFT para el sistema de refrigeración por agua. Nos interesa saber qué tan tolerante a fallas es este sistema, dadas las correspondientes probabilidades de falla y reparación de sus componentes. Una sola caja de reparación está conectada a todos los elementos básicos (por líneas discontinuas para facilitar la comprensión de la imagen), que son las bombas y los interruptores. Estudiamos el comportamiento de nuestro sistema en una situación de estado estacionario, es decir, la porción de tiempo que pasa en estados fallidos sobre la cantidad de tiempo de todo el largo plazo. Como se explica en [28], sea `SYS_FAIL` la proposición que describe estos estados fallidos, es decir, la falla de nuestro evento principal en nuestro caso, luego la fórmula CSL `S(SYS_FAIL)` describe la propiedad a analizar.

En lugar de describir el modelo de forma gráfica, podemos hacerlo en un lenguaje de descripción textual para árboles de fallas. Esto facilitará la compilación a los módulos IOSALa sintaxis para el lenguaje de descripción de árboles de fallas de Kepler se puede encontrar en el Apéndice B. Este lenguaje está inspirado principalmente en el lenguaje de descripción textual de Galileo [94]. La descripción en lenguaje Kepler del modelo Water Cooling se encuentra en la Figura 5.14. Hemos desarrollado un compilador que tra-

duce los modelos de Kepler en modelos IOSA$_u$ siguiendo la teoría presentada en este capítulo. Hemos compilado nuestro modelo de Kepler y el modelo IOSA$_u$ obtenido se introdujo como entrada en FIG junto con la propiedad antes mencionada para analizar automáticamente el modelo de refrigeración por agua sin necesidad de más intervención del usuario.

```
 1 toplevel "FAIL";
 2 "FAIL" and "S1" "S2";
 3 "S1" or "SS1" "PS1";
 4 "S2" or "SS2" "PS2";
 5 "SS1" pand "SW1" "M1";
 6 "PS1" sg "M1" "AUX";
 7 "SS2" pand "SW2" "M2";
 8 "PS2" sg "M2" "AUX";
 9 "M1" exponential(0.01) uniform(1,5);
10 "M2" exponential(0.01) uniform(1,5);
11 "AUX" exponential(0.01) exponential(0.0025) uniform(1,5);
12 "SW1" exponential(0.003) uniform(1,2);
13 "SW2" exponential(0.003) uniform(1,2);
14 "RBOX" priority_rbox "M1" "M2" "SW1" "SW2" "AUX";
```

Figura 5.14: Descripción Kepler del Water Cooling System

Escribimos un programa simple en Python para traducir estos modelos escritos en lenguaje Kepler a un modelo IOSA en el formalismo RFT descrito en este capítulo. Realizamos un análisis de Montecarlo en dicho modelo utilizando la herramienta FIG. La propiedad bajo análisis que elegimos es `S(SYS_FAIL)`, para cuantificar la disponibilidad del sistema de enfriamiento. Los resultados entregados por la herramienta FIG se muestran en la Tabla 5.1. El límite de tiempo para el experimento fue de 5 minutos. El valor obtenido para el evento superior fue 2,39e-08. El experimento se ejecutó en una máquina Intel i5-4200M de 2,5 GHz.

5.9. Conclusiones

En este capítulo, proporcionamos una definición formal de una versión reparable de los árboles de fallas. También lo incrustamos con una semántica formal cuyo dominio son IOSA con urgencia. Además, demostramos que cualquier modelo RFT es determinista débil si el modelo no utiliza puertas

Estimated value: **2.39e-08**		
Confidence	Precision	Confidence Interval
80 %	1.49e-08	[1.64e-08, 3.14e-08]
85 %	2.04e-08	[1.37e-08, 3.41e-08]
90 %	2.60e-08	[1.09e-08, 3.69e-08]
99 %	4.08e-08	[3.49e-09, 4.43e-08]

Cuadro 5.1

de repuesto y elementos básicos de repuesto. Desafortunadamente, no pudimos proporcionar un resultado tan general cuando se utilizan sistemas de repuesto en el modelo. Sin embargo, hemos demostrado la confluencia en varias configuraciones de puertas de repuesto y elementos básicos de repuesto, e insinuamos cómo se puede verificar la confluencia de cualquier configuración posible. El capítulo termina con un estudio de caso que hace uso de la herramienta FIG para el análisis de una propiedad rara. Por lo que sabemos, este es el primer estudio sobre el análisis de simulación de eventos raros de árboles de fallas reparables con distribuciones de probabilidad de reparación y falla general.

La introducción del modelo de reparación junto con distribuciones generales para el análisis de árboles de fallas, convierte los modelos en lo que Christos Cassandras [32] llama sistemas del "mundo real". Para este tipo de modelos, se deben abandonar las suposiciones y abstracciones habituales realizadas para facilitar el análisis, y ya no es posible buscar una solución analítica. Otro trabajo que investiga el uso de la simulación de eventos raros aplicado al caso específico de los Fault Trees es [91]. Incluye un modelo de reparación, que involucra estrategias de reparación complejas como la inspección [90], pero restringe las medidas de probabilidad a Exponencial y Erlang.

Una posible dirección para el trabajo futuro podría ser la introducción de puertas de dependencia de fallas (como en [16, 91]). Esto debe hacerse de manera que no produzcan indeterminismo. Nótese que por el momento no se ha definido ningún orden en las fallas dependientes y por lo tanto el no determinismo es intrínseco a la definición. Otra línea de trabajo sería definir una traducción automática de una herramienta de modelado gráfico de árboles de fallas a los modelos IOSA, con el fin de automatizar y facilitar la ingeniería de modelado y análisis de RFTs. Además, se podrían introducir mecanismos de reparación más complejos con el modelo de mantenimiento de [90].

Capítulo 6

Discuciones finales

Un viaje desde la matemática rigurosa a la aplicación industrial

En esta tesis presentamos un marco que nos permite recorrer todo el camino desde los fundamentos matemáticos esenciales, hasta finalmente dar rigor matemático al análisis de sistemas industriales complejos. Para ello definimos una especialización de Autómatas Estocásticos, a la que llamamos *Input/Output Stochastic Automata* (IOSA), adaptada al análisis de sistemas estocásticos con distribuciones generales mediante simulación de eventos discretos. El análisis a través de la simulación de eventos discretos requiere que los modelos sean deterministas. Por lo tanto, una parte considerable de esta tesis trabaja en desarrollar y unir las herramientas matemáticas para probar que un modelo en IOSA es determinista. Finalmente, usamos nuestro lenguaje de modelado determinista para elaborar una versión determinista de Repairable Fault Trees (RFT), una técnica destacada para el análisis de sistemas industriales tolerantes a fallas. IOSA permite que los modelos RFT proporcionen distribuciones continuas arbitrarias a eventos de falla y reparación. Los modelos RFT se pueden analizar, por ejemplo, utilizando la herramienta de simulación de eventos raros FIG.

6.1. Logros

La figura 6.1 resume gráficamente los logros de esta tesis. IOSA y IOSA_u construyen su semántica sobre NLMP, tal como lo hacen los Autómatas Estocásticos (SA) (las flechas que van hacia NLMP). IOSA es una especialización de SA con transiciones de entrada y salida que resulta ser determinista (flecha de SA a IOSA). IOSA_u se obtiene extendiendo IOSA con transiciones

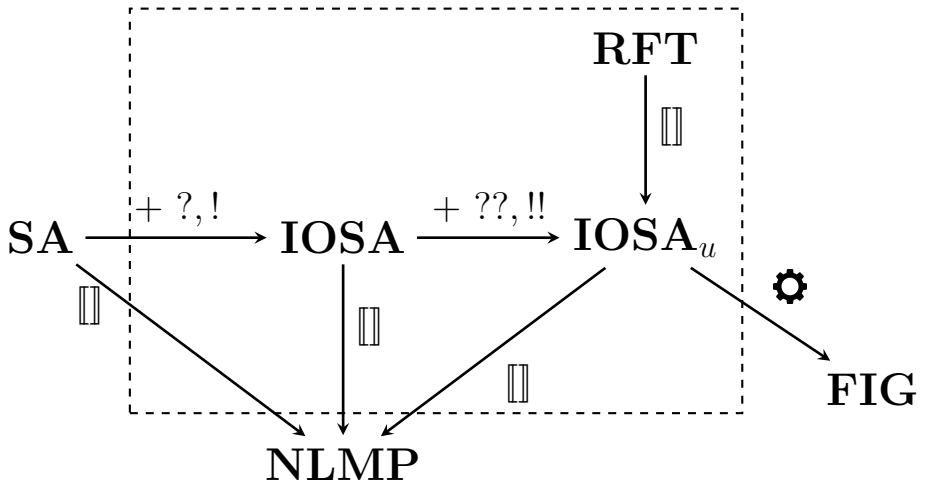


Figura 6.1: Síntesis Gráfica del Aporte de Esta Tesis (Enmarcado en Lineas Punteadas).

urgentes (flecha de IOSA a IOSA$_u$). En este trabajo se formalizó RFT y se les dio una semántica determinista débil en términos de IOSA$_u$ (flecha de RFT a IOSA$_u$). Finalmente, IOSA$_u$ es el lenguaje de entrada para la herramienta de simulación de eventos raros de FIG (flecha de IOSA$_u$ a FIG). El rectángulo punteado en la Figura 6.1 encierra la contribución de esta tesis.

Presentamos la primera versión de autómatas estocásticos con entrada / salida (IOSA) en el capítulo 3. IOSA es composicional y admite distribuciones de probabilidad continuas arbitrarias para modelar el comportamiento temporal estocástico de un sistema. Definimos su semántica en términos de NLMP, imponiendo un conjunto de condiciones en la definición. Estas condiciones finalmente aseguraron que un IOSA cerrado, es decir, un modelo donde todas las sincronizaciones se han resuelto y no quedan entradas, es determinista. La posibilidad de utilizar distribuciones continuas arbitrarias hace que IOSA sea muy adecuado para modelar y simular sistemas con resultados más realistas que los modelos de Markov como los CTMC. Además, en caso de que el modelo use solo distribuciones exponenciales, el IOSA cerrado es susceptible al análisis numérico ya que se reduce a una CTMC. La naturaleza compositiva de IOSA nos permite concentrarnos en el comportamiento claro de los componentes y la comunicación intuitiva entre ellos, en contraste con las técnicas de modelado monolítico propensas a errores. La reutilización y la mantenibilidad se mencionan como beneficios adicionales del modelado compositivo. Todas estas características son de gran ventaja cuando se modelan modelos industriales extensos y complejos.

También ampliamos IOSA con acciones urgentes. Llamamos a este nuevo

modelo IOSA$_u$. Se introdujeron acciones urgentes como solución a las limitaciones de modelado compositivo de la IOSA original, que instaba a introducir un retraso a la hora de sincronizar componentes. La sincronización a través de acciones urgentes elimina esta condición. Aunque tal extensión introduce el no determinismo incluso si el IOSA$_u$ está cerrado, lo hace de manera limitada. Pudimos caracterizar cuándo un IOSA es determinista débil, lo cual es un concepto importante ya que los IOSAs deterministas débiles son susceptibles de simulación de eventos discretos. En particular, mostramos que las IOSAs cerradas y confluentes son deterministas débiles. Proporcionamos condiciones para verificar composicionalmente si una IOSA cerrada es confluente.

Finalmente formalizamos árboles de fallas reparables (RFT) y los dotamos de una semántica en términos de IOSA$_u$. Aunque los árboles de fallas son un formalismo omnipresente en el análisis de riesgo de sistemas de tamaño industrial, hasta donde sabemos, no hay otro trabajo que involucre distribuciones continuas arbitrarias para fallas y tiempos de reparación y reparaciones interdependientes complejas. Además, hemos demostrado que nuestra semántica produce modelos deterministas débiles que, por lo tanto, son aptos para la simulación por eventos discretos.

6.2. Trabajo Futuro

La complejidad y necesidades de los sistemas industriales actuales representan un enorme desafío para la verificación formal. Por un lado, no solo se desea un análisis cualitativo de fallas, sino que también se requiere generalmente un análisis cuantitativo, como costos y rendimiento. Algunos trabajos futuros en esta dirección serían actualizar IOSA para respaldar los costos, las recompensas y las probabilidades internas. En la misma dirección, podríamos modificar nuestro formalismo RFT para involucrar el mantenimiento y la degradación de fase como sugiere [23, 24, 57, 90]. La degradación de fase parece ser una mejora simple que no afectaría el determinismo débil del modelo. Por otro lado, se debe tener especial cuidado al actualizar a mantenibilidad, ya que involucra la dinámica de las reparaciones complejas interdependientes. Por lo tanto, esta extensión es más propensa a introducir no determinismo en el modelo.

Demostrar que el modelo de puerta de repuesto es confluente y, por lo tanto, débilmente determinista en el caso general es una deuda de este trabajo. Los repuestos se utilizan para modelar la redundancia, que es de gran importancia en los diseños de sistemas tolerantes a fallos.

Destacamos que la representación gráfica intuitiva de Fault Trees es una

de sus características más atractivas. Un trabajo interesante sería desarrollar una interfaz gráfica para el modelado de RFT y un compilador en la semántica IOSA. Luego, el simulador FIG podría usarse para analizar el modelo compilado. Una interfaz gráfica para modelar y diseñar el análisis sería un gran paso hacia las posibilidades de ofrecer una herramienta completa para uso industrial.

La experimentación con RFT en FIG ha evidenciado la necesidad de mejorar las construcciones automáticas de funciones de importancia para el método RESTART. Si bien se esperaba una mayor eficiencia al usar el método RESTART, los experimentos mostraron que los tiempos para obtener los resultados no diferían mucho de los de Montecarlo simple. Analizar las razones detrás de este fenómeno podría ayudar a construir mejores funciones de importancia automática para otros sistemas que comparten características similares con los RFT.

Bibliografía

[1] Rajeev Alur and David L. Dill. The theory of timed automata. In J. W. de Bakker, Cornelis Huizing, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer, 1991.

[2] Suprasad Amari, Glenn Dill, and Eileen Howald. A new approach to solve dynamic fault trees. In *Reliability and Maintainability Symposium, 2003. Annual*, pages 374–379. IEEE, 2003.

[3] R.B. Ash and C. Doléans-Dade. *Probability and Measure Theory*. Harcourt/Academic Press, 2000.

[4] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[5] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[6] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[7] Anis Baklouti, Nga Nguyen, Jean-Yves Choley, Faïda Mhenni, and Abdelfattah Mlika. Free and open source fault tree analysis tools survey. In *2017 Annual IEEE International Systems Conference, SysCon 2017, Montreal, QC, Canada, April 24-27, 2017*, pages 1–8. IEEE, 2017.

[8] Marco Beccuti, Daniele Codetta-Raiteri, Giuliana Franceschinis, and Serge Haddad. Non deterministic repairable fault trees for computing optimal repair strategy. In *Proceedings of the 3rd international conference on performance evaluation methodologies and tools*, page 56. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[9] Marco Beccuti, Daniele Codetta Raiteri, Giuliana Franceschinis, and Serge Haddad. Non deterministic repairable fault trees for computing optimal repair strategy. In John S. Baras and Costas Courcoubetis, editors, *3rd International ICST Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2008, Athens, Greece, October 20-24, 2008*, page 56. ICST/ACM, 2008.

[10] Richard Blute, Josee Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 149–158. IEEE Computer Society, 1997.

[11] Andrea Bobbio, Giuliana Franceschinis, Rossano Gaeta, and Luigi Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level petri net semantics. *IEEE Trans. Software Eng.*, 29(3):270–287, 2003.

[12] Andrea Bobbio and D Codetta Raiteri. Parametric fault trees with dynamic gates and repair boxes. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS*, pages 459–465. IEEE, 2004.

[13] Jonathan Bogdoll, Luis María Ferrer Fioriti, Arnd Hartmanns, and Holger Hermanns. *Partial Order Methods for Statistical Model Checking and Simulation*, pages 59–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[14] Henrik C. Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Software Eng.*, 32(10):812–830, 2006.

[15] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14:25–59, 1987.

[16] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A compositional semantics for dynamic fault trees in terms of interactive markov chains. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *Automated Technology for Verification and Analysis, 5th International Symposium, ATVA 2007, Tokyo, Japan, October 22-25, 2007, Proceedings*, volume 4762 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2007.

[17] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. Dynamic fault tree analysis using input/output interactive markov chains. In *The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25-28 June 2007, Edinburgh, UK, Proceedings*, pages 708–717. IEEE Computer Society, 2007.

[18] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE Trans. Dependable Sec. Comput.*, 7(2):128–143, 2010.

[19] Hichem Boudali and Joanne Bechta Dugan. A discrete-time bayesian network reliability modeling and analysis framework. *Reliability Engineering & System Safety*, 87(3):337–349, 2005.

[20] Hichem Boudali and Joanne Bechta Dugan. A continuous-time bayesian network reliability modeling, and analysis framework. *IEEE transactions on reliability*, 55(1):86–97, 2006.

[21] Mario Bravetti. *Specification and analysis of stochastic real-time systems*. PhD thesis, PhD thesis, Dottorato di Ricerca in Informatica. Universita di Bologna, Padova, Venezia, 2002.

[22] Mario Bravetti and Pedro R. D'Argenio. *Tutte le Algebre Insieme: Concepts, Discussions and Relations of Stochastic Process Algebras with General Distributions*, pages 44–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[23] K. Buchacker. Combining fault trees and petri nets to model safety-critical systems. *High Performance Computing 1999*, pages 439–444, 1999. Cited By :17.

[24] K. Buchacker. Modeling with extended fault trees. In *Proceedings. Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE 2000)*, pages 238–246, Nov 2000.

[25] Carlos E. Budde. *Automation of Importance Splitting Techniques for Rare Event Simulation*. PhD thesis, Universidad Nacional de Córdoba, Argentina, 2017.

[26] Carlos E. Budde. *Automation of Importance Splitting Techniques for Rare Event Simulation*. PhD thesis, Universidad Nacional de Córdoba, 2017.

[27] Carlos E. Budde, Pedro R. D'Argenio, and Holger Hermanns. Rare event simulation with fully automated importance splitting. In Marta Beltrán, William J. Knottenbelt, and Jeremy T. Bradley, editors, *EPEW 2015*, volume 9272 of *LNCS*, pages 275–290. Springer, 2015.

[28] Carlos E. Budde, Pedro R. D'Argenio, and Raúl E. Monti. Compositional construction of importance functions in fully automated importance splitting. In Antonio Puliafito, Kishor S. Trivedi, Bruno Tuffin, Marco Scarpa, Fumio Machida, and Javier Alonso, editors, *Procs. of VALUETOOLS 2016*. ACM, 2017.

[29] Carlos E. Budde, Pedro R. D'Argenio, Pedro Sánchez Terraf, and Nicolás Wolovick. *A Theory for the Semantics of Stochastic and Nondeterministic Continuous Systems*, pages 67–86. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[30] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: quantitative model and tool interaction. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, volume 10206 of *Lecture Notes in Computer Science*, pages 151–168, 2017.

[31] Carlos E Budde, Pedro R D'Argenio, and Holger Hermanns. Rare event simulation with fully automated importance splitting. In *European Workshop on Performance Engineering*, pages 275–290. Springer, 2015.

[32] Edwin K. P. Chong. Discrete event systems: Modeling and performance analysis - by christos g. cassandras, richard d. irwin, inc., and aksen associates, inc., homewood, il, 1993. xix + 790 pp. ISBN 0-256-11212-6. *Discrete Event Dynamic Systems*, 4(1):113–116, 1994.

[33] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.

[34] David Coppit, Kevin J Sullivan, and Joanne Bechta Dugan. Formal semantics of models for computational engineering: A case study on dynamic fault trees. In *Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on*, pages 270–282. IEEE, 2000.

[35] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[36] Pepijn Crouzen. *Modularity and Determinism in Compositional Markov Models*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2014.

[37] Pedro R. D' Argenio, Joost P. Katoen, and Hendrik Brinksma. *A Stochastic Automata Model and its Algebraic Approach*, pages 1–16. Technical Report. Centre for Telematics and Information Technology (CTIT), Netherlands, 1997.

[38] Vincent Danos, Josée Desharnais, François Laviolette, and Prakash Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation*, 204(4):503 – 523, 2006. Seventh Workshop on Coalgebraic Methods in Computer Science 2004.

[39] Pedro D'Argenio, Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. Smart sampling for lightweight verification of markov decision processes. *International Journal on Software Tools for Technology Transfer*, 17(4):469–484, Aug 2015.

[40] Pedro R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, Enschede, 1999.

[41] Pedro R. D'Argenio, Carlos E. Budde, Matias David Lee, Raúl E. Monti, Leonardo Rodríguez, and Nicolás Wolovick. The road from stochastic automata to the simulation of rare events. In Joost-Pieter Katoen, Rom Langerak, and Arend Rensink, editors, *ModelEd, TestEd, TrustEd - Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday*, volume 10500 of *Lecture Notes in Computer Science*, pages 276–294. Springer, 2017.

[42] Pedro R. D'Argenio and Joost-Pieter Katoen. A theory of stochastic systems part I: Stochastic automata. *Inf. Comput.*, 203(1):1–38, 2005.

[43] Pedro R. D'Argenio, Joost-Pieter Katoen, and Ed Brinksma. An algebraic approach to the specification of stochastic systems. In David Gries and Willem P. de Roever, editors, *Programming Concepts and Methods, IFIP TC2/WG2.2,2.3 International Conference on Programming Concepts and Methods (PROCOMET '98) 8-12 June 1998, Shelter Island, New York, USA*, volume 125 of *IFIP Conference Proceedings*, pages 126–147. Chapman & Hall, 1998.

[44] Pedro R. D'Argenio and Raúl E. Monti. Input/output stochastic automata with urgency – confluence and weak determinism, 2018. In preparation.

[45] Pedro R. D'Argenio, Pedro Sánchez Terraf, and Nicolás Wolovick. Bisimulations for non-deterministic labelled markov processes. *Mathematical Structures in Computer Science*, 22(1):43–68, 2012.

[46] Pedro R. D'Argenio, Nicolás Wolovick, Pedro Sánchez Terraf, and Pablo Celayes. Nondeterministic labeled markov processes: Bisimulations and logical characterization. In *QEST 2009, Sixth International Conference on the Quantitative Evaluation of Systems, Budapest, Hungary, 13-16 September 2009*, pages 11–20. IEEE Computer Society, 2009.

[47] Josée Desharnais. *Labelled markov processes.* PhD thesis, McGill University, Montréal, 1999.

[48] Josee Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. *Inf. Comput.*, 179(2):163–193, 2002.

[49] E. S. DIAMANT and L. M. HEROLD. Thermal performance of cork insulation on minuteman missiles. *Journal of Spacecraft and Rockets*, 3(5):679–684, May 1966.

[50] Ernst-Erich Doberkat and Pedro Sánchez Terraf. Stochastic nondeterminism and effectivity functions. *J. Log. Comput.*, 27(1):357–394, 2017.

[51] J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, Sep 1992.

[52] Christian Eisentraut, Holger Hermanns, Julia Krämer, Andrea Turrini, and Lijun Zhang. Deciding bisimilarities on distributions. In Kaustubh R. Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio, editors, *Quantitative Evaluation of Systems - 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8054 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 2013.

[53] Marnix Joseph Johann Garvels. *The splitting method in rare event simulation.* PhD thesis, University of Twente, Enschede, Netherlands, 2000.

[54] Daniel Gburek, Christel Baier, and Sascha Klüppelholz. Composition of stochastic transition systems based on spans and couplings. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 102:1–102:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[55] Daochuan Ge, Meng Lin, Yanhua Yang, Ruoxing Zhang, and Qiang Chou. Quantitative analysis of dynamic fault trees using improved sequential binary decision diagrams. *Rel. Eng. & Sys. Safety*, 142:289–299, 2015.

[56] Michèle Giry. A categorical approach to probability theory. In *Categorical aspects of topology and analysis (Ottawa, Ont., 1980)*, volume 915 of *Lecture Notes in Mathematics*, pages 68–85. Springer, Berlin, 1982.

[57] Dennis Guck, Joost-Pieter Katoen, Mariëlle IA Stoelinga, Ted Luiten, and Judi Romijn. Smart railroad maintenance engineering with stochastic model checking. *Proceedings of RAILWAYS. Saxe-Coburg Publications*, pages 950–953, 2014.

[58] Rohit Gulati and Joanne Bechta Dugan. A modular approach for analyzing static and dynamic fault trees. In *Reliability and Maintainability Symposium. 1997 Proceedings, Annual*, pages 57–63. IEEE, 1997.

[59] David F Haasl, NH Roberts, WE Vesely, and FF Goldberg. Fault tree handbook. Technical report, Nuclear Regulatory Commission, Washington, DC (USA). Office of Nuclear Regulatory Research, 1981.

[60] Ernst Moritz Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods in System Design*, 43(2):191–232, 2013.

[61] Arnd Hartmanns. *On the analysis of stochastic timed systems*. PhD thesis, Saarland University, 2015.

[62] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 278–292. IEEE Computer Society, 1996.

[63] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.

[64] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002.

[65] Holger Hermanns, Ulrich Herzog, and Vassilis Mertsiotakis. Stochastic process algebras - between LOTOS and markov chains. *Computer Networks*, 30(9-10):901–924, 1998.

[66] Ulrich Herzog. Formal description, time and performance analysis. A framework. In Theo Härder, Hartmut Wedekind, and Gerhard Zimmermann, editors, *Entwurf und Betrieb verteilter Systeme, Fachtagung des Sonderforschungsbereiche 124 und 182, Dagstuhl, 19.-21. September 1990, Proceedings*, volume 264 of *Informatik-Fachberichte*, pages 172–190. Springer, 1990.

[67] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[68] André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Inf. Comput.*, 127(2):164–185, 1996.

[69] Sebastian Junges, Dennis Guck, Joost-Pieter Katoen, and Mariëlle Stoelinga. Uncovering dynamic fault trees. In *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016, Toulouse, France, June 28 - July 1, 2016*, pages 299–310. IEEE Computer Society, 2016.

[70] Sebastian Junges, Joost-Pieter Katoen, Mariëlle Stoelinga, and Matthias Volk. One net fits all - A unifying semantics of dynamic fault trees using gspns. In Victor Khomenko and Olivier H. Roux, editors, *Application and Theory of Petri Nets and Concurrency - 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia, June 24-29, 2018, Proceedings*, volume 10877 of *Lecture Notes in Computer Science*, pages 272–293. Springer, 2018.

[71] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In P. Kemper, editor, *Proc. Tools Session of Aachen 2001 International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems*, pages 7–12, September 2001.

[72] Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[73] Wen-Shing Lee, Doris L Grosh, Frank A Tillman, and Chang H Lie. Fault tree analysis, methods, and applications - a review. *IEEE transactions on reliability*, 34(3):194–203, 1985.

[74] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In Howard Barringer, Yliès Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon J. Pace, Grigore Rosu, Oleg Sokolsky, and Nikolai Tillmann, editors, *Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, volume 6418 of *Lecture Notes in Computer Science*, pages 122–135. Springer, 2010.

[75] Dong Liu, Weiyan Xing, Chunyuan Zhang, Rui Li, and Haiyan Li. Cut sequence set generation for fault tree analysis. In Yann-Hang Lee, Heung-Nam Kim, Jong Kim, Yongwan Park, Laurence Tianruo Yang, and Sung Won Kim, editors, *Embedded Software and Systems, [Third] International Conference, ICESS 2007, Daegu, Korea, May 14-16, 2007, Proceedings*, volume 4523 of *Lecture Notes in Computer Science*, pages 592–603. Springer, 2007.

[76] Nancy A. Lynch and Mark R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In Fred B. Schneider, editor, *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1987*, pages 137–151. ACM, 1987.

[77] Ragavan Manian, David W Coppit, Kevin J Sullivan, and J Bechta Dugan. Bridging the gap between systems and dynamic fault tree models. In *Reliability and Maintainability Symposium, 1999. Proceedings. Annual*, pages 105–111. IEEE, 1999.

[78] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems - specification*. Springer, 1992.

[79] Kenneth L. McMillan. *Symbolic model checking*. Kluwer, 1993.

[80] Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage, and Andrea Bobbio. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. *IEEE Trans. Reliability*, 59(1):250–261, 2010.

[81] R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[82] Michael K. Molloy. Performance analysis using stochastic petri nets. *IEEE Trans. Computers*, 31(9):913–917, 1982.

[83] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[84] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[85] Daniele Codetta Raiteri, Giuliana Franceschinis, Mauro Iacono, and Valeria Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *Dependable Systems and Networks, 2004 International Conference on*, pages 659–668. IEEE, 2004.

[86] Daniele Codetta Raiteri, Mauro Iacono, Giuliana Franceschinis, and Valeria Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *DSN 2004*, pages 659–668. IEEE Computer Society, 2004.

[87] Antoine Rauzy. Sequence algebra, sequence decision diagrams and dynamic fault trees. *Rel. Eng. & Sys. Safety*, 96(7):785–792, 2011.

[88] Gerardo Rubino and Bruno Tuffin. *Rare Event Simulation Using Monte Carlo Methods*. Wiley Publishing, 2009.

[89] Gerardo Rubino and Bruno Tuffin. *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

[90] E. Ruijters, D. Guck, P. Drolenga, and M. Stoelinga. Fault maintenance trees: Reliability centered maintenance via statistical model checking. In *2016 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6, Jan 2016.

[91] Enno Ruijters, Daniël Reijsbergen, Pieter-Tjerk de Boer, and Mariëlle Stoelinga. Rare event simulation for dynamic fault trees. In Stefano Tonetta, Erwin Schoitsch, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security - 36th International Conference, SAFECOMP 2017, Trento, Italy, September 13-15, 2017, Proceedings*, volume 10488 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2017.

[92] Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15:29–62, 2015.

[93] Roberto Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[94] K. J. Sullivan, J. B. Dugan, and D. Coppit. The galileo fault tree analysis tool. In *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No.99CB36352)*, pages 232–235, June 1999.

[95] Zhihua Tang and J. B. Dugan. Minimal cut set/sequence generation for dynamic fault trees. In *Annual Symposium Reliability and Maintainability, 2004 - RAMS*, pages 207–213, Jan 2004.

[96] Rob J. van Glabbeek, Scott A. Smolka, and Bernhard Steffen. Reactive, generative and stratified models of probabilistic processes. *Inf. Comput.*, 121(1):59–80, 1995.

[97] W. Vesely, J. Dugan, J. Fragola, Minarick, and J. Railsback. Fault Tree Handbook with Aerospace Applications. Handbook, National Aeronautics and Space Administration, Washington, DC, 2002.

[98] Ignacio Viglizzo. *Coalgebras on Measurable Spaces*. PhD thesis, Indiana University, USA, 2005.

[99] Ignacio Dario Viglizzo. Coalgebras on measurable spaces. 2010.

[100] Manuel Villen-Altamirano and Jose Villen-Altamirano. Restart: A method for accelerating rare event simulations. *Analysis*, 3(3), 1991.

[101] Manuel Villén-Altamirano and José Villén-Altamirano. The rare event simulation method RESTART: efficiency analysis and guidelines for its application. In Demetres D. Kouvatsos, editor, *Network Performance Engineering - A Handbook on Convergent Multi-Service Networks and Next Generation Internet*, volume 5233 of *LNCS*, pages 509–547. Springer, 2011.

[102] Nicolás Wolovick. Continuous probability and nondeterminism in labeled transaction systems. Phd, Universidad Nacional de Córdoba, Córdoba, 2012.

[103] Sue-Hwey Wu, Scott A. Smolka, and Eugene W. Stark. Composition and behaviors of probabilistic I/O automata. *Theor. Comput. Sci.*, 176(1-2):1–38, 1997.

[104] Liudong Xing, Akhilesh Shrestha, and Yuanshun Dai. Exact combinatorial reliability analysis of dynamic systems with sequence-dependent failures. *Rel. Eng. & Sys. Safety*, 96(10):1375–1385, 2011.

[105] Wang Yi. Real-time behaviour of asynchronous agents. In Jos C. M. Baeten and Jan Willem Klop, editors, *CONCUR '90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 27-30, 1990, Proceedings*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer, 1990.

[106] Håkan LS Younes and Reid G Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *International Conference on Computer Aided Verification*, pages 223–235. Springer, 2002.

Apéndice A

IOSA Syntax

La gramática libre de contexto de la figura A.1 define el lenguaje completo de modelado simbólico IOSA$_u$Àquí * significa "tantas veces como quieras", + "al menos una vez", ? significa opcional, | separa opciones y entre paréntesis agrupa producciones y elementos.

$$
\begin{aligned}
\text{MODEL} &= (\text{MODULE})+ \\
\text{MODULE} &= (\text{VARIABLE} \mid \text{ARRAY} \mid \text{CLOCK} \mid \text{TRANSITION})+ \\
\text{VARIABLE} &= \text{NAME} \texttt{ : } \text{TYPE} \texttt{ init } \text{VALUE} \texttt{ ;} \\
\text{ARRAY} &= \text{NAME}\texttt{[}\text{INT}\texttt{]:} \text{ TYPE } \texttt{init } \text{VALUE} \texttt{ ;} \\
\text{CLOCK} &= \text{NAME} \texttt{ : clock ;} \\
\text{TRANSITION} &= \texttt{[} (\text{NAME } (\texttt{?}\mid\texttt{??}\mid\texttt{!}\mid\texttt{!!})?)? \texttt{]} \text{ PRE } (\texttt{@ } \text{NAME})? \rightarrow \text{POS} \texttt{ ;} \\
\text{PRE} &= ((\text{NAME} = \text{EXPR})(\texttt{\&} \text{ NAME} = \text{EXPR})*)? \\
\text{POS} &= (\texttt{(} \text{ NAME'} = \text{EXPR } \texttt{)}(\texttt{\&} \texttt{ (} \text{ NAME'} = \text{EXPR } \texttt{)})*)? \\
\text{EXPR} &= \text{VALUE} \mid \text{NAME} \mid \text{EXPR OP EXPR} \mid \texttt{(} \text{ EXPR } \texttt{)} \mid \texttt{!} \text{ EXPR} \mid \\
&\quad \text{DISTR} \\
\text{OP} &= \texttt{| } \mid \texttt{ \& } \mid \texttt{ + } \mid \texttt{ - } \mid \texttt{ * } \mid \texttt{ / } \mid \texttt{ =} \\
\text{NAME} &= (\texttt{a}\mid\texttt{b}\mid\ldots\mid\texttt{z}\mid\texttt{A}\mid\texttt{B}\mid\ldots\mid\texttt{Z})(\texttt{a}\mid\texttt{b}\mid\ldots\mid\texttt{z}\mid\texttt{A}\mid\texttt{B}\mid\ldots\mid\texttt{Z}\mid\texttt{1}\mid\ldots\mid\texttt{9}\mid\texttt{\_}\mid\texttt{-})* \\
\text{TYPE} &= \texttt{boolean} \mid \texttt{[} \text{INT } \texttt{..} \text{ INT } \texttt{]}
\end{aligned}
$$

$$
\begin{aligned}
\text{VALUE} &= \texttt{true} \mid \texttt{false} \mid \text{INT} \\
\text{INT} &= (\texttt{1}|\texttt{2}|...|\texttt{9})(\texttt{0}|\texttt{1}|...|\texttt{9})* \\
\text{FLOAT} &= (\texttt{0}|\texttt{1}|...|\texttt{9}) + (\texttt{.}(\texttt{0}|\texttt{1}|...|\texttt{9})+)? \\
\text{DISTR} &= \texttt{normal(}\text{FLOAT}\texttt{,}\text{FLOAT}\texttt{)} \mid \texttt{exponential(}\text{FLOAT}\texttt{)} \mid \\
&\quad\ \texttt{uniform(}\text{FLOAT}\texttt{,}\text{FLOAT}\texttt{)} \mid ...
\end{aligned}
$$

Figura A.1: Gramática del Lenguaje Simbólico IOSA$_u$

Apéndice B

Kepler Syntax

La siguiente gramática define la sintaxis del lenguaje de descripción del árbol de fallas Kepler. La gramática se define en la notación de gramática de expresión de análisis (PEG). Las producciones se describen con mayúsculas y minúsculas, mientras que cualquier otra palabra alfabética es una cadena que va tal cual, así como otros tipos de palabras y caracteres encerrados entre comillas simples. Paréntesis agrupa producciones y barras separan opciones. El símbolo + indica que el patrón anterior debe producirse en la lista una vez, * indica una o más veces en el mismo sentido, mientras que ? indica opcional.

El evento principal se escribe en la línea superior. Cada línea sucesiva describirá una puerta, un evento básico o un cuadro de reparación. Un BE comienza con su nombre, luego la palabra be para indicar que es un evento básico y finalmente la distribución de probabilidad para sus relojes de falla y reparación. Un be de repuesto es similar, excepto que la segunda distribución de probabilidad corresponde a su reloj de falla en modo inactivo y una tercera corresponde a la distribución de falla. Solo hay siete tipos de distribuciones definidas, que son las implementadas actualmente en el simulador de eventos raros de FIG. Sin embargo, el lenguaje puede extenderse simplemente a cualquier otra distribución continua y aún será posible compilarlo en modelos IOSA. Finalmente, las puertas se definen por un nombre, una cadena que determina el tipo de puerta y una lista de sus nombres de entrada. En particular, las puertas de repuesto y las cajas de reparación requieren especificar su modo operativo, que puede ser “primero en llegar, primero en ser atendido”, o “prioritario”.

```
 1 KEPLER = TOPLEVEL (GATE / MODEGATE / BE / SBE)+
 2 TOPLEVEL = toplevel NAME ';'
 3 BE = NAME be DIST DIST ';'
 4 SBE = NAME sbe DIST DIST DIST ';'
 5 GATE = NAME ( and / or / pand / vot / fdep) NAME* ';'
 6 MODEGATE = ( sg / rbox ) MODE NAME* ';'
 7 INT = 0 / (1/.../9)(0/.../9)*
 8 FLOAT = INT+ (. INT+( (e/E) -? INT+)? )?
 9 NAME = (a / ... / z / A / ... / Z)+ ( _ / INT / NAME)*
10 MODE = fcfs / priority
11 DIST = exponential '(' FLOAT ')' /
12        normal '(' FLOAT , FLOAT ')' /
13        erlang '(' FLOAT , FLOAT ')'/
14        uniform '(' FLOAT , FLOAT ')'/
15        lognormal '(' FLOAT , FLOAT ')'/
16        weibull '(' FLOAT , FLOAT ')'/
17        rayleigh '(' FLOAT ')'/
18        gamma '(' FLOAT , FLOAT ')'/
```

Figura B.1: Sintaxis FTDL