Índice general

Agradecimientos

A Dios y a la Virgen de Guadalupe porque simpre han guiado mis pasos y me han ayudado a llegar hasta este punto.

A mis padres, José Luis Chavarría y Lourdes Báez porque siempre me han apoyado y animado a salir adelante, a superarme día con día y a ser una mejor persona. Porque siempre he recibido de ellos ejemplos de superación y de calidad humana.

A mis hermanos, Ivonne y Juan Franciso, por su comprensión, apoyo, ayuda y por todos los momentos que hemos pasado juntos.

A mi asesora, la Dra. Xiaoou Li, por toda la paciencia que me tuvo durante el desarrollo de la tesis, por todas las correcciones en mi investigación y porque siempre ha sido muy amable.

A los revisores de mi trabajo, el Dr. Sergio Chapa y el Dr. Pedro Mejía, por sus comentarios ya que estos contribuyeron a mejorar el contenido de la tesis.

A mis super amigos, Lalo y Adry, por su amistad, apoyo y solidaridad cuando teníamos que estudiar mucho, pero sobretodo, por su cariño.

A Miguel porque siempre estuvo a mi lado en estos dos años llenos de trabajo, porque se alegraba conmigo cuando salian las cosas y me animaba cuando otras iban algo mal.

A Joselito por toda su colaboración y buena disposición.

Con especial cariño a las dos personas que estuvieron conmigo por casi 20 años y que estarían orgullosos de lo que he logrado.

A todos los Doctores de la Sección de Computación porque compartieron sus conocimientos.

A Sofi porque sin ella no habría podido hacer ningún trámite administrativo pero, además, por su preocupación y cariño hacia nosotros.

A toda mi generación porque estuvimos juntos en las buenas y en las malas.

Al CINVESTAV y a CONACYT por apoyarme con la beca de maestría.

Resumen

Tradicionalmente, los sistemas de base de datos (BD) han sido vistos como simples repositorios de información. Son pasivos en el sentido de que sólo ejecutan acciones cuando el usuario o un programa se los pide explícitamente. Sin embargo, existen aplicaciones para las cuales este comportamiento no es suficiente. Algunas aplicaciones representan procesos que necesitan ser monitoreados y controlados de forma automática. Para este tipo de aplicaciones, las BD activas (BDAs) ofrecen una solución adecuada porque manejan reglas Evento – Condición - Acción (ECA). Las reglas ECA permiten especificar acciones que se ejecutan automáticamente cuando sucede un evento y se cumplen algunas condiciones.

Las BDAs solucionan eficazmente algunas aplicaciones que necesitan ser controladas automáticamente. Sin embargo, su mantenimiento, particularmente de las reglas ECA, es un reto, aún con un conjunto pequeño de reglas. Durante el procesamiento de una regla otras reglas se pueden activar y/o desactivar, como consecuencia de un flujo de eventos previo, sin que el usuario esté consciente de ello. La dificultad del mantenimiento de las reglas ECA es uno de los mayores obstáculos para que las BDAs sean ampliamente usadas.

Esta tesis presenta una evaluación de la complejidad de la interacción entre las reglas ECA. El propósito es identificar aquellas reglas que, por su dependencia con algunas otras, pueden causar problemas en su mantenimiento, tales como inconsistencia de funcionamiento. En esta evaluación se consideran eventos compuestos, los cuales involucran tiempo o más de un evento primitivo para poder disparar una regla. Para evaluar la complejidad de la interacción entre reglas ECA, se emplearon los atributos: número de anclas, distancia y potencial de disparo de una regla. Además, se empleó la Red de Petri Coloreada Condicional para modelar las reglas ECA. La evaluación se validó con la teoría de la medición. Asimismo, se desarrolló un sistema de software que permite calcular automáticamente el valor de los atributos. Este software se integró a una herramienta más robusta, mediante la cual es posible generar una CCPN a partir de reglas

ECA. Por otra parte, se analizó un caso de estudio con el software desarrollado. Los resultados muestran que el método desarrollado es viable para identificar aquellas reglas que pueden causar problemas en su mantenimiento y, como consecuencia, en el mantenimiento total del sistema.

Abstract

Traditionally, database systems have been used as simple information repositories. They are passive in the sense that they only execute actions when a user or a program request them. However, there are applications which cannot be effectively modeled by this pattern. For this kind of applications, active databases (ADBs) offer a suitable solution because they handle event - condition - action rules (ECA). ECA rules allow specify actions which execute automatically when an event happens and some conditions are met.

ADBs resolve effectively some applications which need to be controlled automatically. However, its maintenance, specifically of the ECA rules, is a challenge, even with a small set of rules. The difficulty of the maintenance of ECA rules is one of the greatest obstacles for ADBs to become widely used.

This thesis presents an evaluation of triggering interaction complexity among ECA rules. The purpose is to identify those rules which, due to dependency with some ones, can cause troubles in its maintenance, such as operation inconsistency. In this evaluation we consider not only primitive events but also composite events which involve time specification and/or more than one primitive event. In order to evaluate the triggering interaction complexity between ECA rules, we define three parameters: number of anchors, distance and triggering potential, based on the Conditional Colored Petri Net (CCPN) model. Furthermore the evaluation was validated using the measurement theory. Also, we developed a software system ECAComplex, which allows to calculate automatically the parameter values. This software has been integrated into ECAPNSim which is an ECA rule simulator based on CCPN. On the other hand, a case study was analyzed with ECAComplex. The results show that the proposed method is feasible to identify those rules that can cause problems in rule base maintenance, and as consequence in the system maintenance.

Capítulo 1

Introducción

Los Sistemas Gestores de Base de Datos (DBMS, por sus siglas en inglés) son pasivos en el sentido que ejecutan acciones cuando el usuario o una aplicación se los pide. Este comportamiento pasivo no es suficiente para cubrir las necesidades de algunos sistemas que necesitan ser monitoreados y controlados de forma automática. Para satisfacer las necesidades de tales sistemas, las BD han evolucionado y se han creado diferentes áreas de investigación. Una de esas áreas son las Bases de Datos Activas (BDAs).

Las BDAs permiten ejecutar acciones de forma automática cuando sucede un evento y se cumplen algunas condiciones. El mecanismo mediante el cual exhiben el comportamiento activo son las reglas evento - condición - acción (ECA). Utilizando estas reglas, las BDAs solucionan eficazmente aplicaciones que necesitan ser controladas automáticamente. Sin embargo, el mantenimiento de las reglas ECA en una BDA es un reto, incluso con un conjunto pequeño. La interacción implícita entre las reglas y la dependencia que pueden llegar a tener unas con otras, dificulta hacer un seguimiento del disparo de las reglas, así como tratar de determinar el evento que originó su disparo.

Para resolver el problema que representa el mantenimiento de las reglas ECA se hace uso de las métricas de software. Dichas métricas permiten determinar el grado de complejidad de una regla a través de la medición de diferentes parámetros. La información que proporcionan las métricas ayuda a simplificar el mantenimiento de las reglas ECA.

1.1. Estado del arte

Para satisfacer los requisitos especiales de cierto tipo de aplicaciones (manejo de grandes volúmenes de datos, alertas para la administración, manejo de datos espaciales y temporales, transacciones grandes y complejas) se está desarrollando investigación sobre BD. Las siguientes son algunas de la aplicaciones no tradicionales más importantes que la tecnología de BD se esfuerza en resolver [8]:

- *Sistemas de información geográfica (GIS)*. Los GIS manejan datos geográficos/espaciales, por ejemplo, mapas para investigación ambiental, militar, planeación de ciudades, etc.
- *Educación*. En el proceso de aprendizaje a distancia, los cursos multimedia requieren de datos en tiempo real ya sea por Internet o por una intranet.
- *Sistemas estadísticos*. Este tipo de sistemas tiene que trabajar con grandes volúmenes de datos con procesos costosos de agregación, manejo de tiempo y dimensiones espaciales.
- *Comercio electrónico*. Las aplicaciones relacionadas con Internet se incrementan día a día. La tendencia es poner toda la información en Internet para hacerla accesible a una mayor cantidad de personas.

Desde la concepción de las BD, se ha visto una tendencia hacia transferir toda la semántica posible de los programas hacia la BD y almacenarla junto con los datos. La migración de la semántica y otras capacidades han evidenciado ventajas, a tal grado que su centralización libera a las aplicaciones de tener que verificar restricciones de integridad. De esta forma, todos los programas pueden compartir los datos sin tener que preocuparse de aspectos que el DBMS mantiene unificados, forzando su verificación, sin importar qué programa acceda a la BD. Este comportamiento es el que exhiben las BDAs, las cuales aparecieron a principios de 1990. En los DBMS activos (ADBMS), además de la descripción de los datos y las restricciones, parte de la información de control se almacena en la BD. De esta forma, las BDA pueden ejecutar aplicaciones sin la intervención del usuario apoyándose en la ejecución de disparos (*triggers*), reglas, alertas, entre otros.

Las ventajas de los ADBMS son numerosas:

1. Promueven la reusabilidad de código. Más que replicar el código en distintas aplicaciones, el código reside en un solo lugar desde el cual se invoca implícitamente.
2. El mantenimiento se simplifica. Los cambios en las políticas se localizan en una única pieza de código.

3. En un ambiente cliente/servidor, el centralizar el comportamiento reactivo reduce el tráfico de la red.

El mecanismo en el cual los ADBMS se apoyan para tener un comportamiento reactivo, son las reglas activas. Las reglas activas tienen tres componentes: un evento, una condición y una acción, por lo tanto, también se conocen como reglas ECA. El evento describe lo que tiene que suceder para que la regla pueda responder. La condición es una expresión booleana o una consulta que examina el contexto en el cual el evento tomó lugar. La acción especifica la tarea que se debe llevar a cabo si un evento ha tomado lugar y la evaluación de la condición ha sido verdadera. El hecho de que el evento haya ocurrido no es suficiente para que la acción se lleve a cabo, se necesita, además, que la condición se cumpla. La acción, a menudo, lleva a cabo operaciones de actualización de la BD o de comunicación de información al usuario de la BD.

Las reglas ECA son un mecanismo eficiente para describir el comportamiento reactivo de una BDA. Sin embargo, a diferencia de lenguajes de programación tradicionales, donde el control secuencial se especifica explícita y estáticamente por el programador, las reglas ECA se disparan dinámicamente basadas en un flujo de eventos previo. La dependencia implícita entre las reglas y la forma en la que interactúan, hace difícil no solamente preveer, sino también determinar cuáles fueron las causas que hicieron que una regla se disparara.

La dificultad en el mantenimiento de las reglas ECA ha sido abordada en [10]. En ese trabajo, los autores proponen la utilización de métricas como una herramienta para mejorar el mantenimiento de los proyectos de BD, en particular, de las reglas ECA. Basándose en la dificultad de comprobar las causas que originan el disparo de una regla, los autores proponen tres diferentes métricas para medir la complejidad de las reglas ECA. Tales métricas son: el número de anclas, la distancia y el potencial de disparo. Para representar el contexto de las reglas, hacen uso de la noción de gráfica de disparo y la modifican en dos aspectos:

1. Se les asigna un peso a los arcos. Este peso indica el número de eventos potenciales producidos por el disparo de la regla.

2. Los nodos se extienden con el conjunto de transacciones *T*. Una transacción es un conjunto atómico de acciones de BD donde cada una de esas acciones puede corresponder a un evento que dispara una o más reglas.

Un ejemplo de una gráfica de disparo modificada se muestra en la figura 1.1, donde T0 puede producir un evento significativo para S1 y cuatro para S2.

Las métricas propuestas se validaron usando la teoría de la medición. Sin embargo, la desventaja de ese trabajo es que los autores únicamente consideran eventos primitivos ya que su gráfica de disparo modificada

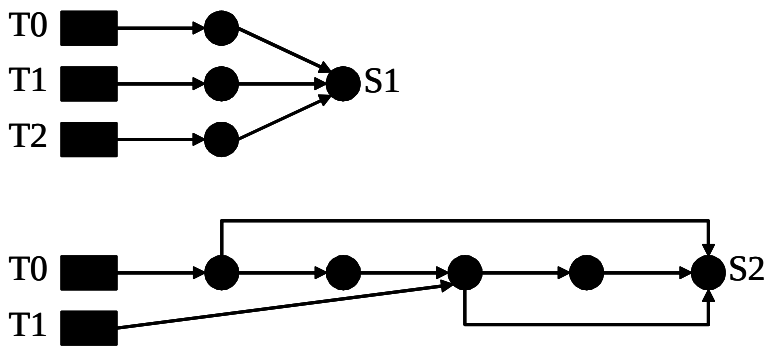

Figura 1.1: Gráfica de disparo modificada

no proporciona un mecanismo para poder representar eventos compuestos.

Un modelo que permite obtener mayor información acerca de las reglas ECA es la Red de Petri Coloreada Condicional (CCPN). La relación lógica de las tres partes de una regla ECA puede ser modelada como una estructura de CCPN [11]. La condición se modela como una transición. El evento y la acción se modelan como lugares de entrada y salida de la transición, respectivamente. Además, las operaciones del evento y la acción se pueden modelar como colores de la CCPN. Esta extensión de PN fue desarrollada específicamente para BDA. Cuenta con una interfaz gráfica denominada ECAPNSim, la cual puede generar, automáticamente, una CCPN a partir de un conjunto de reglas ECA almacenadas en un archivo.

1.2. Motivación

Actualmente, existe un mercado competitivo en el cual la calidad es un factor crítico para alcanzar el éxito en aspectos organizacionales y económicos. El área de sistemas de información no está exenta de esta competitividad. Desarrollar aplicaciones de software con calidad es importante, así como evaluar aspectos involucrados con la misma, por ejemplo, mantenimiento o complejidad de los sistemas usando métricas válidas.

Las métricas de software son ampliamente reconocidas como una forma efectiva de entender, monitorear, controlar, predecir y mejorar el desarrollo y mantenimiento de los proyectos.

Siendo las BDA un área importante dentro de las BD, es lógico pensar en tener métricas que puedan

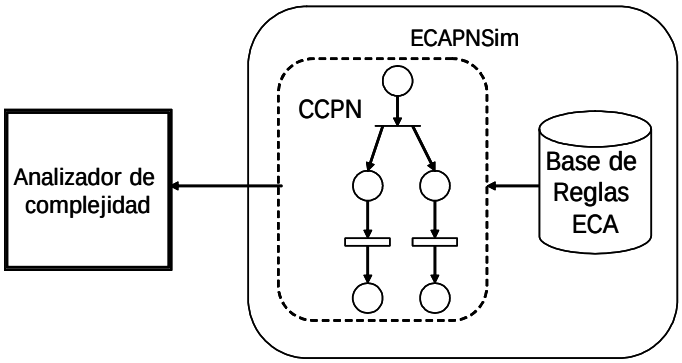

Figura 1.2: Ubicación del módulo de análisis de complejidad

evaluar su desempeño, sobre todo, porque el mantenimiento de las reglas ECA representa un reto cuando se tiene, incluso, un conjunto pequeño de reglas que interactúan mucho entre sí.

Dado el problema que representa el mantenimiento de reglas ECA en una BDA y que tal vez éste sea el mayor obstáculo para que los sistemas activos sean ampliamente usados, el establecimiento de una métrica para la medición de la complejidad de la interacción entre las reglas ayudará a los desarrolladores de software a decidir entre diferentes cursos de acción para el diseño e implementación de los mismos, de forma que sean útiles y fáciles de mantener.

Existen pocos trabajos que tratan sobre el tema de medición de complejidad de reglas ECA en BDAs, y aún hay aspectos que detallar antes de poder tener una métrica bien establecida. Es por eso que este proyecto se enfoca en el aspecto de medir la complejidad de las interacciones de las reglas en una BDA.

Otra motivación para el desarrollo de este proyecto radica en el hecho de que existe una base para el modelado de reglas ECA que permite realizar un análisis de las reglas empleando para ello la CCPN. En [11], se describe el modelo de CCPN así como el sistema de software ECAPNSim. ECAPNSim permite generar automáticamente una CCPN a partir de un archivo de texto que contenga la descripción de las reglas. Además de la representación gráfica de la CCPN, esta interfaz puede generar la representación matemática de la CCPN.

En la figura 1.2 se muestra la interacción que tendrá el módulo desarrollado en este trabajo de tesis y el trabajo desarrollado en [11].

Hay que hacer notar que el trabajo en [11] contiene más módulos de los que se muestran en la figura

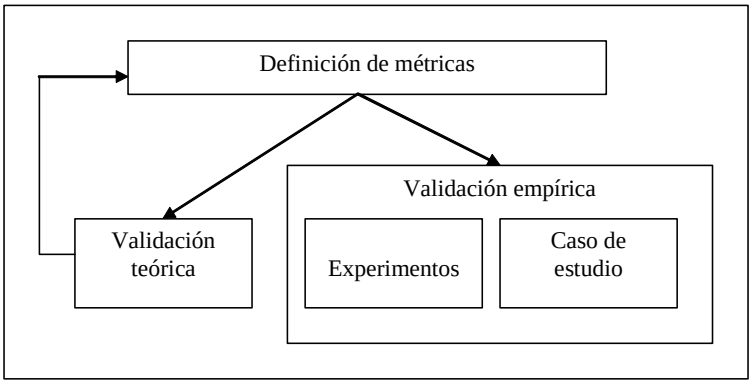

Figura 1.3: Proceso para la propuesta y validación de métricas

1.2, pero el trabajo desarrollado aquí se apoyará fuertemente sólo en el módulo de la CCPN.

1.3. Planteamiento del problema

El planteamiento del problema se puede describir con ayuda de la figura 1.3.

A continuación se detalla cada una de las etapas mostradas en la figura 1.3:

- **Definición de métricas:** El primer paso es proponer las métricas. Una metodología para proponerlas es la aproximación GQM (Goal - Question - Metric).
- **Validación teórica:** El segundo paso es la validación formal de las métricas. Esta validación formal ayuda para saber cómo y cuando aplicar las métricas.
- **Validación empírica:** El objetivo de este paso es demostrar la utilidad práctica de las métricas propuestas. Existen dos formas de lograr este objetivo, que son: experimentación y caso de estudio. Las primeras consisten en realizar experimentos controlados y los casos de estudio trabajan sobre datos reales.

Siguiendo los pasos anteriores, para este trabajo de tesis se tienen las siguientes propuestas:
Definición de métricas.

Para definir las métricas se utilizó la aproximación GQM.

Donde:

Goal: Medir la complejidad de la interacción de las reglas en una BDA.

Question: ¿Cómo medir la complejidad de la interacción de las reglas ECA en una BDA?

Metric: Para responder la pregunta y cumplir así el objetivo, se propusieron las siguientes métricas, basados en [10], distancia, potencial de disparo y número de anclas sobre el modelo de CCPN.

Validación teórica.

Para la validación teórica, se revisó la teoría de la medición [12] de forma general para entender los principios básicos. Posteriormente, se revisaron marcos de trabajo para medidas de software existentes en la literatura. Específicamente se revisó el marco de trabajo de Zuse [2].

Validación empírica.

Para la parte de experimentos, se compararon los resultados obtenidos con este trabajo de tesis con los resultados obtenidos en [10]. También se realizaron comparaciones entre los resultados obtenidos utilizando un número variable de reglas.

1.4. Descripción del trabajo

Para responder la pregunta ¿cómo medir la complejidad de la interacción de los disparos de las reglas ECA en una BDA? lo primero que establecimos fue el modelo para representar las reglas ECA. El modelo que utilizamos fue la CCPN debido a las siguientes razones:

- Cuenta con una representación gráfica y matemática.
- Proporciona información más detallada acerca de las reglas.
- Permite el manejo de eventos compuestos.

Sobre la estructura gráfica de la CCPN modelamos los parámetros de medición: número de anclas, distancia y potencial de disparo, los cuales se propusieron en [10]. Fue necesario ampliar la definición de los parámetros para poder considerar eventos compuestos. Dado que el proceso de medición sobre la estructura gráfica de CCPN es tedioso cuando se tiene una CCPN muy grande, empleamos la representación matemática de la misma para desarrollar algoritmos que permitieran determinar el valor de los parámetros.

La representación matemática de la CCPN está dada por la matriz de incidencia. Cada entrada de la matriz contiene la información del peso del arco que conecta una transición con un lugar y viceversa.

Teniendo la definición de los parámetros sobre el modelo de CCPN aplicamos la teoría de la medición, particularmente el trabajo desarrollado por el Dr. Horst Zuse, para verificar la validez de los parámetros. La validación de los parámetros nos permitió ubicarlos en el nivel de la escala ordinal.

Posteriormente, se implementaron los algoritmos desarrollados en el lenguaje de programación Java. Esto se hizo para mantener la compatibilidad con la interfaz ECAPNSim.

El sistema de software desarrollado permite obtener, de forma automática, el valor de cada uno de los parámetros de medición de complejidad.

Para verificar la utilidad práctica de este sistema, realizamos diversos experimentos sobre diferentes conjuntos de reglas ECA. Asimismo, realizamos comparaciones entre los resultados que obtuvimos con nuestro sistema y los resultados presentados en [10]. El resultado de estas comparaciones es que nuestro sistema permite realizar un análisis de complejidad más completo al contemplar eventos compuestos.

Las aportaciones principales de este trabajo de tesis son:

1. Utilizar la CCPN para modelar las métricas de complejidad propuestas en [10].
2. Ampliar la definición de las métricas para considerar eventos compuestos.
3. Modelar los parámetros de medición de complejidad sobre la estructura gráfica de la CCPN.
4. Utilizar la matriz de incidencia para el cálculo de los paramétros.
5. Desarrollar algoritmos basados en la matriz de incidencia para calcular de forma automática el valor de los parámetros.
6. Desarrollar una aplicación de software para obtener el valor de las métricas.
7. Desarrollar ejemplos para mostrar la factibilidad del sistema que implementamos.

El documento de tesis se estructura de la siguiente forma: en el Capítulo 2 se presenta el papel de las reglas ECA en las BDAs. El Capítulo 3 exhibe los fundamentos de la teoría de la medición clásica, asimismo, se muestra el marco de trabajo para mediciones de software desarrollado por el Dr. H. Zuse. El Capítulo 4 expone los fundamentos de la CCPN, se muestra su modelo gráfico y su modelo matemático. En el Capítulo

5 se desarrolla la modelación de los parámetros de medición de complejidad sobre la estructura gráfica de la CCPN, se muestra el mapeo de los parámetros al modelo matemático de la CCPN y los algoritmos desarrollados. Finalmente, se muestra la verificación teórica de los parámetros sobre el marco de trabajo del Dr. Zuse y algunas conclusiones al respecto. El Capítulo 6 presenta la implementación del sistema de software que desarrollamos. El Capítulo 7 expone algunos casos de estudio. Finalmente, el Capítulo 8 expone los resultados que se obtuvieron y se listan los trabajos futuros a fin de realizar mejoras a los resultados que se tienen hasta el momento.

Capítulo 2

Reglas ECA en Bases de Datos Activas

Normalmente, las bases de datos (BD) han sido vistas como un medio para almacenar información. El usuario puede interactuar con ellas a través de interfaces o programas de usuario. Sin embargo, las BD han empezado a incorporarse a una gama más amplia de aplicaciones que están asociadas con un procesamiento de información más complejo. Esto ha traído como consecuencia el desarrollo de investigaciones sobre BD con el fin de introducir, dentro de la BD misma, funcionalidad adicional requerida por las aplicaciones. Entre las áreas que han recibido especial atención en años recientes se encuentran: BD temporales, espaciales, multimedia, deductivas y activas [13].

Los *Sistemas Manejadores de BD* (DBMS, por sus siglas en inglés) consisten de una colección de datos interrelacionados - la BD - y un conjunto de programas para acceder a dichos datos [9]. Son pasivos en el sentido que los comandos son ejecutados por la BD cuando el usuario, o alguna aplicación, lo solicitan. Sin embargo, existen situaciones que no pueden ser modeladas eficazmente por los sistemas pasivos. Como ejemplo, consideremos una BD de reservaciones en donde se almacena información acerca de horarios, lugares disponibles, precios, etc., la cual puede ser accedida por diferentes terminales. En algunos casos puede ser benéfico agregar programas especiales para controlar que la cantidad de lugares disponibles no se encuentre por debajo de un cierto límite. Para tratar esta situación, se pueden implementar dos soluciones. La primera de ellas consiste en agregar una capacidad de monitoreo a todos los programas de reservaciones que acceden a la BD. Sin embargo, esta opción implica que la semántica de la tarea de monitoreo sea distribuida, replicada y ocultada entre diferentes programas de aplicación. La segunda opción se basa en un mecanismo de interrogación que periódicamente verifica el número de lugares disponibles. En este caso la semántica

de la aplicación se representa en un solo lugar. La principal desventaja de este método es la dificultad de determinar la frecuencia de interrogación. Si es muy baja, la reacción puede darse demasiado tarde.

Las BDAs solucionan una aplicación como la descrita porque incorporan el comportamiento activo al DBMS. Esto implica que el DBMS activo (ADBMS) tiene que proveer al usuario mecanismos para describir el comportamiento reactivo (modelo de conocimiento), para monitorear y reaccionar ante circunstancias relevantes (modelo de ejecución) y para mantener y depurar el comportamiento reactivo [13].

Una aproximación común al modelo de conocimiento utiliza reglas que tienen tres componentes: un *evento*, una *condición* y una *acción*. Tales reglas se denominan reglas evento - condición - acción (ECA). El evento de la regla describe un suceso para el cual la regla es capaz de responder. La condición examina el contexto en el cual el evento tomó lugar. La acción describe la tarea que la regla debe llevar a cabo si un evento relevante ha tomado lugar y la evaluación de la condición resultó verdadera.

El modelo de ejecución determina la forma en que un conjunto de reglas se procesa en tiempo de ejecución para responder a eventos que están tomando lugar.

Además de proporcionar el soporte para el modelo de conocimiento y para el modelo de ejecución, una BDA debe proveer facilidades para administrar la base de reglas ECA. Por ejemplo, mecanismos para activar o desactivar reglas específicas, para visualizar conjuntos grandes de reglas y para apoyar la programación de las mismas.

Existen diversas aplicaciones que pueden resolverse satisfactoriamente mediante el uso de reglas activas [13], [14]. Entre esas aplicaciones se pueden distinguir las siguientes:

- *Extensiones a los sistemas de base de datos*. Las reglas activas pueden ser vistas como un mecanismo para ayudar en la implementación de otras partes del sistema de BD.
- *Aplicaciones de base de datos cerradas*. Involucra el uso de la funcionalidad activa para describir parte del comportamiento que es manifestado por un sistema de software sin hacer referencia a dispositivos o sistemas externos.
- *Aplicaciones de base de datos abiertas*. En esta categoría la BD se usa junto con un dispositivo de monitoreo para registrar y responder a situaciones externas a la BD.

En este capítulo analizamos las principales características de las reglas ECA. Para ello, la sección 2.1 describe de forma general a las reglas ECA así como su modelo de conocimiento y de ejecución. La sección 2.2 presenta el manejo de reglas ECA. En la sección 2.3 se muestran algunas propiedades de las reglas. La

sección 2.4 presenta la complejidad de la interacción de las reglas. Finalmente, la sección 2.5 expone los comentarios finales del capítulo.

2.1. Reglas Evento-Condición-Acción

Una parte importante de las BDAs son las reglas ECA. Para mostrar las características de estas reglas es necesario describir su modelo de conocimiento y su modelo de ejecución. El modelo de conocimiento describe las características estructurales de las reglas ECA de forma individual. El modelo de ejecución describe la evaluación, en tiempo de ejecución, de conjuntos de reglas. En esta sección describimos ambos modelos.

2.1.1. Modelo de Conocimiento

El modelo de conocimiento se utiliza para describir la funcionalidad activa a través de las reglas ECA e indica *qué* se puede decir acerca de las mismas [13], [14]. Cada elemento que conforma una regla ECA se describe a continuación.

Evento

Un evento es algo que sucede en un instante de tiempo dado. El poder detectarlo depende, en gran medida, de su *fuente* o *generador*. Un evento puede originarse por cualquiera de las siguientes fuentes [13]:

- *Operación de estructura*. El evento sucede por una operación sobre alguna parte de la estructura de la BD, por ejemplo, insertar una tupla o actualizar un atributo.
- *Invocación de comportamiento*. El evento sucede por la ejecución de operaciones definidas por el usuario.
- *Transacción*. El evento sucede por comandos de transacciones, por ejemplo, el comando abort.
- *Definidos por el usuario*. Se utiliza un mecanismo de programación para permitir que un programa de aplicación señale la ocurrencia de un evento, por ejemplo, como respuesta a alguna información introducida por el usuario.

- *Excepción*. El evento sucede como resultado de una excepción, por ejemplo, intentar acceder a los datos sin tener autorización.

- *Reloj*. El evento sucede en un instante de tiempo que puede ser absoluto, relativo a otro acontecimiento o periódico.

- *Externo*. El evento sucede por una situación fuera de la BD, por ejemplo, la temperatura subió por encima de 30 grados.

La *granularidad* de un evento indica si un evento está definido para todos los objetos en un conjunto, para un cierto subconjunto o para miembros específicos del conjunto.

Un evento puede ser de dos *tipos*: **primitivo** o **compuesto** [14]. El evento es primitivo cuando sucede por una ocurrencia de cualquiera de las fuentes descritas. El evento es compuesto cuando sucede por una combinación de eventos primitivos o compuestos usando un rango de operadores que constituyen el *álgebra de eventos*.

Los operadores de eventos más comunes son los siguientes: *disyunción, <E_1 o E_2>* sucede cuando cualquiera de los eventos E_1 ó E_2 ocurre; *conjunción, <E_1 y E_2>* sucede cuando ambos eventos E_1 y E_2 ocurren en cualquier orden; *secuencia, <sec(E_1, E_2)>* sucede cuando E_1 ocurre antes que E_2; *cerradura, <cerradura(E) en Int>* sucede la primera vez que *E* ocurre en el intervalo de tiempo *Int*, sin importar las posteriores ocurrencias de *E*; *historia, <veces(n, E) en Int>* se señala cuando el evento *E* ocurre *n* veces durante el intervalo de tiempo *Int*; *negación, <negación(E) en Int>* detecta la no ocurrencia del evento *E* en el intervalo de tiempo *Int*; *último, <last(E) en Int>*, toma la última ocurrencia del evento *E* en el intervalo de tiempo *Int*; *simultáneo, <sim(e1, e2)>* ocurre cuando suceden al mismo tiempo los eventos *E1* y *E2*; *alguno, <any(E1,E2,...,E_n,E_m)>* ocurre cuando han sucedido *m* eventos de *n* posibles. Cabe mencionar que se han propuesto diversas álgebras de eventos para determinados sistemas [15], sin embargo, los que aquí presentamos son los más comunes.

Cuando se detecta un evento compuesto, pudieron haber ocurrido un número de instancias de eventos, tal vez del mismo tipo, para formar el evento compuesto. Para detectar la forma en que esos eventos se combinaron para formar el evento compuesto se usan *políticas de consumo*. Las cuatro políticas de consumo más comunes son: **contexto reciente**, en donde se considera el conjunto de eventos más reciente que puede ser usado para formar la composición; **contexto cronológico**, el cual consume un evento en orden cronológico; **contexto continuo**, el cual define una ventana deslizante e inicia una nueva composición cada vez que se

detecta un evento primitivo; y **contexto acumulativo**, el cual acumula todos los eventos primitivos hasta formar el evento compuesto.

El *rol* de un evento indica si el evento siempre se debe especificar para las reglas activas o si puede omitirse. Si el rol es **opcional** y no se especifica el evento, se obtienen reglas de la forma condición - acción. Cuando el rol es **ninguno**, entonces no se pueden especificar eventos y todas las reglas son condición - acción. Si el rol es **obligatorio**, todas las reglas serán reglas ECA porque el evento siempre debe ser especificado.

Condición

La condición de una regla evalúa el contexto en el cual el evento tomó lugar.

El *rol* de una condición indica si ésta siempre debe especificarse [13]. En las reglas ECA la condición puede ser **opcional**. Si la condición no se especifica o el rol es **ninguno**, entonces se obtienen reglas de la forma evento - acción. Sin embargo, si el evento y la condición son opcionales, uno de ellos debe ser especificado.

El *contexto* indica el marco en el cual será evaluada la condición. Cada uno de los componentes de las reglas ECA no se evalúa por separado ni de forma independiente de la BD. El procesamiento de una regla ECA puede estar asociado con cuatro estados de la base de datos: DB_T - el estado de la BD al inicio de la transacción actual; DB_E - la BD cuando ocurre un evento; DB_C - la BD cuando la condición es evaluada; y DB_A - la BD cuando la acción es ejecutada.

Acción

El rango de tareas que pueden ser llevadas a cabo por una acción se especifica como sus *opciones*. Las acciones pueden ser [16]: actualizar la estructura de la BD o de un conjunto de reglas, realizar la invocación de comportamiento dentro de la base de datos o una llamada externa, informar al usuario o al administrador del sistema de alguna situación anómala, interrumpir una transacción, o tomar un curso de acción alternativo usando la cláusula **hacer – en_lugar_de**.

El *contexto* de una acción indica la información que está disponible para la acción.

Una vez definido el modelo de conocimiento mostramos las características del modelo de ejecución.

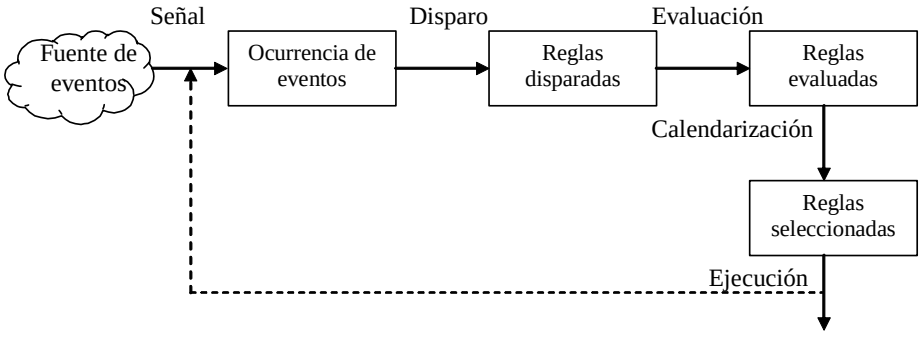

Figura 2.1: Principales pasos que toman lugar durante la ejecución de una regla

2.1.2. Modelo de ejecución

El modelo de ejecución indica la forma en la que un conjunto de reglas se trata en tiempo de ejecución.

Aunque este modelo está estrechamente relacionado con aspectos esenciales del DBMS, existen fases en la evaluación de una regla que merecen atención [13]:

- *Fase de señal*, se refiere a la ocurrencia de un evento debido a una fuente de evento.
- *Fase de disparo*, toma los eventos producidos hasta el momento y activa las reglas correspondientes. Cuando se asocia una regla con la ocurrencia de un evento se instancia una regla.
- *Fase de evaluación*, evalúa la condición de la regla que ha sido activada. Si existen varias instancias de reglas y de todas ellas se satisface su condición entonces se tiene un conjunto de reglas en conflicto.
- *Fase de calendarización*, indica la forma de resolver los conflictos en el conjunto anterior.
- *Fase de ejecución*, realiza las acciones de la instancia de regla escogida. Durante la ejecución de la acción otros eventos pueden ser señalados lo que puede producir el disparo de reglas en cascada.

La figura 2.1 ilustra las fases descritas.

Todas esas fases pueden no ser ejecutadas de forma secuencial, todo depende del modo de acoplamiento de los pares evento - condición y condición - acción. Los modos de acoplamiento más comunes son [13]:

- *Inmediato*, en este caso la condición (acción) se evalúa (ejecuta) inmediatamente después del evento (condición).

- *Diferido*, en este caso la condición (acción) se evalúa (ejecuta) dentro de la misma transacción que el evento (condición) de la regla, pero no necesariamente en la primera oportunidad.
- *Separado*, en este caso la condición (acción) se evalúa (ejecuta) en una transacción diferente de la del evento (condición).

La relación entre los eventos y las reglas que disparan se captura por la *granularidad de la transición*. Cuando la granularidad de la transición es **tupla**, la ocurrencia de un único evento dispara una única regla. Cuando la granularidad es de tipo **conjunto**, se agregan los eventos a una colección que se usa para disparar una regla. Otra característica de las relaciones entre eventos y reglas disparadas, es la *política de efecto total*. Esta política indica si debe considerarse el efecto total de una cierta cantidad de eventos o deben considerarse cada uno por separado. Por ejemplo, si una instancia es actualizada y luego borrada, el efecto total es borrar la instancia original; si una instancia se inserta y luego se actualiza, el efecto total es insertar la instancia actualizada.

Cuando los eventos son señalados por la evaluación de la condición o acción de una regla se emplea una *política de ciclo*. En general, hay dos opciones. Si la política de ciclo es **iterativa**, la evaluación de la condición y la ejecución de la acción no se suspenden para responder a los eventos señalados. Si la política de ciclo es **recursiva**, los eventos señalados por la evaluación de la condición o acción provocan que la condición o acción se suspenda.

La fase de calendarización de evaluación de una regla determina qué sucede cuando múltiples o varias reglas se disparan al mismo tiempo. Los dos puntos principales de esta etapa son:

- *La selección de la siguiente regla a ser disparada*. El orden de las reglas puede influenciar fuertemente el resultado final y reflejar el tipo de razonamiento empleado por el sistema. Las reglas se pueden seleccionar por *aproximaciones dinámicas*, que son aquellas que dan prioridad a las reglas basándose, ya sea, en actualizaciones recientes o en la complejidad de la condición. Las *aproximaciones estáticas*, a menudo, son determinadas por el sistema o por el usuario como un atributo de la regla. En este último caso, una regla se selecciona de un conjunto de reglas disparadas simultáneamente usando un mecanismo de asignación de prioridades. Las reglas pueden ser colocadas en orden usando un esquema numérico, en el cual cada regla tiene un valor absoluto que es su prioridad.
- *El número de reglas a ser disparadas*. Entre las opciones se tiene: (1) disparar todas las reglas de forma secuencial; (2) disparar todas las reglas en paralelo; (3) disparar una regla por saturación y (4)

disparar una o sólo algunas reglas.

El *manejo de errores* es un aspecto importante que debe ser considerado. La mayoría de los sistemas interrumpen la transacción. Sin embargo, otras alternativas pueden ser más convenientes, por ejemplo, ignorar la regla que ha provocado el error y continuar procesando otras reglas; hacer un seguimiento hacia atrás hasta el estado cuando inició el procesamiento de la regla y tratar de restablecer el procesamiento de las reglas o continuar con la transacción; o bien, adoptar planes de contingencia para poderse recuperar del estado de error.

Además de los modelos de conocimiento y ejecución se deben tener facilidades para manejar la base de reglas ECA. En la siguiente sección presentamos las características del manejo de reglas.

2.2. Manejo de reglas

El manejo de reglas indica cómo se pueden representar las reglas, qué operaciones se pueden aplicar sobre ellas y el soporte de programación para las mismas [13].

La *descripción* de las reglas se refiere a la forma en que las reglas se expresan normalmente, ya sea empleando lenguajes de programación de BD, un lenguaje de consultas o como objetos en una BD orientada a objetos.

Otras operaciones comunes que se pueden aplicar a las reglas, aparte de las de creación y borrado, son activación, desactivación y señalización. La activación (desactivación) de reglas hace que el sistema inicie (detenga) el monitoreo del evento o la condición de la regla. Esto ayuda al administrador de la BD a activar o desactivar algunas reglas sin borrarlas. Este mecanismo es útil porque se pueden prevenir ciclos y mejorar la eficiencia. La operación de señalización se requiere para soportar eventos abstractos y es invocada explícitamente por la aplicación para notificar al sistema de reglas de ocurrencias externas.

La *adaptabilidad* se refiere a poder cambiar las reglas. En algunos sistemas sólo es posible cambiar las reglas asociadas con una aplicación recompilando el código de la aplicación. De esta forma, las reglas se pueden cambiar en tiempo de compilación. Otros sistemas permiten la modificación de las reglas en tiempo de ejecución.

2.2.1. Arquitectura del Modelo de Ejecución

En este apartado exponemos la forma de implementar la funcionalidad mostrada en la figura 2.1. Para ello, vamos a describir los principales procesos (rectángulos) de la figura 2.2.

Los principales procesos son los siguientes:

El *detector de eventos* comprueba qué eventos de interés para el sistema de reglas han tomado lugar, si algún evento primitivo se notifica desde la BD o desde una fuente externa, los eventos compuestos se forman a partir de los eventos primitivos y de información acerca de eventos pasados que pueden ser obtenidos de la historia.

Hay dos aspectos principales en la implementación de la detección de eventos: el *monitoreo de eventos primitivos* y la acumulación de información relevante para los *eventos compuestos*.

La detección de eventos primitivos normalmente involucra alguna forma de verificar dentro del núcleo del sistema de la BD.

La detección de eventos compuestos involucra registrar información sobre todos los eventos compuestos detectados parcialmente que pueden ser detectados completamente en el futuro.

El *monitor de condiciones* evalúa las condiciones de reglas asociadas con eventos que han sido identificados con el detector de eventos. En sistemas que soportan reglas condición - acción no hay una cláusula explícita de los eventos que deben ser monitoreados.

Los eventos combinados con la condición describen la situación que una regla ECA está monitoreando. Esto hace que haya una asociación estrecha entre el detector de eventos y el monitor de condiciones - el detector de eventos le informa al monitor de condiciones de los eventos que han ocurrido. Cuando el evento es compuesto la información relevante acerca de éste es también más compleja.

Una vez que las reglas que están asociadas con los eventos detectados han sido recuperadas de la base de reglas deben de pasarse al evaluador de consultas para establecer qué reglas satisfacen las condiciones. El procesador de consultas es una extensión que es usada con una BD pasiva.

El *calendarizador* compara las reglas disparadas recientemente con aquellas que han sido disparadas previamente, actualiza el conjunto de reglas en conflicto y dispara algunas reglas que están calendarizadas para procesamiento inmediato.

El *evaluador de consultas* ejecuta consultas o acciones en la BD. Para ello, necesita accesar al estado actual de la BD así como a los estados pasados para poder soportar el monitoreo de la evolución de la BD.

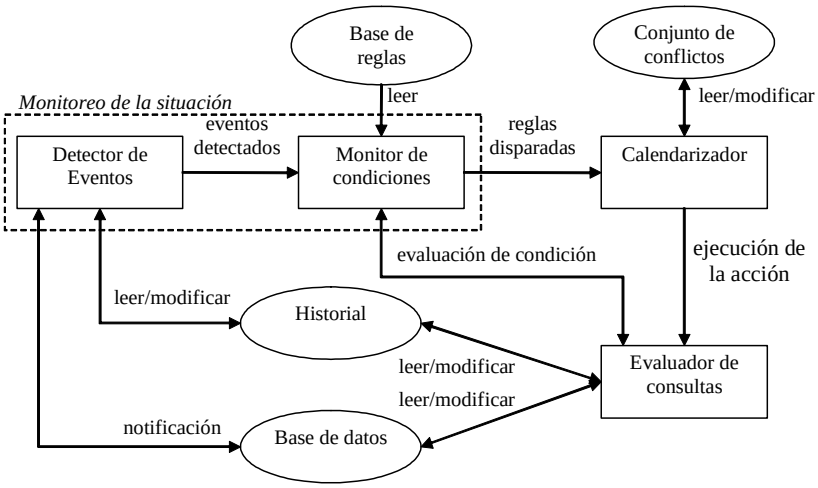

Figura 2.2: Arquitectura de un sistema de BDA abstracto

La funcionalidad de cada uno de los componentes anteriores depende en gran parte de los modelos de conocimiento y de ejecución del sistema de BDA del que se trate. De acuerdo a su arquitectura, se pueden definir dos categorías principales de BDAs:

Por capas. El componente activo es desarrollado como una capa de software de un sistema de BD pasivo. Esta aproximación tiene la ventaja de que no se requiere acceder al código fuente del sistema de BD pasiva y que el sistema activo resultante es fácilmente portable para usarse con diferentes sistemas pasivos.

Integrados. El componente activo se desarrolla cambiando el código fuente de un sistema de BD pasivo existente.

Existen diversos sistemas que incorporan el comportamiento activo. A continuación se muestran algunos ejemplos.

2.2.2. Sistemas de bases de datos activas

La forma general de una regla ECA (también llamada *trigger*) es [9]:

```
ON evento
IF condición
THEN acción
```

Algunos sistemas comerciales de BD proveen facilidades para implementar reglas ECA y, de forma

general, podemos clasificarlos en: sistemas relacionales y sistemas orientados a objetos.

Dentro de los sistemas relacionales, se tienen cuatro propuestas principales que son: Starburst, POSTGRES, Ariel y SQL-3.

Enseguida se muestran las características más importantes de cada uno de ellos.

En el modelo de ejecución de *Starburst*, las reglas son disparadas por el efecto total de un conjunto de cambios a los datos almacenados en la BD.

La característica distintiva del sistema *POSTGRES* [16] es que es orientado a tuplas y las consecuencias de que un evento esté siendo invocado toman efecto inmediatamente.

Una característica importante de *Ariel*, que lo distingue de Starburst o POSTGRES, es que los eventos son opcionales. La conveniencia de las reglas condición – acción depende del contexto. Hay algunas circunstancias en las cuales es necesario hacer explícito la parte del evento de una regla para capturar la semántica de la aplicación.

El estándar *SQL-3* soporta los disparos a nivel de filas y disparos a nivel de cláusula. Estos últimos son ejecutados una vez en respuesta a una actualización de tabla, sin tomar en cuenta que otras tuplas pueden verse afectadas por esta operación.

Por otro lado, las BD orientadas a objetos (BDOO) tienen una estrecha asociación del comportamiento definido por el usuario y los datos de la BD. El comportamiento definido por el usuario se expresa generalmente como métodos que se agregan a las clases que estructuran los datos almacenados en la BD. Esto oculta ciertos aspectos de la estructura de un objeto usando encapsulación y ciertas tareas que pueden ser llevadas a cabo por el comportamiento activo en BD relacionales son soportadas usando métodos en sistemas orientados a objetos. A pesar de esto, existen varias propuestas para extensiones activas de BDOO. Entre las más significativas se encuentran: *HiPAC, EXACT, NAOS, Chimera, Ode, SAMOS, Sentinel* y *REACH.*

El proyecto *HiPAC* introdujo ideas en las BDAs como los modos de acoplamiento y eventos compuestos, pero no fue desarrollado completamente.

EXACT, agregó servicios activos a la BDOO ADAM en la cual instancias, clases, reglas y eventos se representan uniformemente como objetos de la BD.

NAOS (Native Active Object System) es un sistema de reglas activas para la BDOO comercial O_2. Está integrado con la arquitectura modular de O_2 y se puede considerar como un nuevo componente. El componente NAOS se puede usar para definir reglas ECA que serán ejecutadas cuando se procesen aplicaciones de O_2. EL modelo de NAOS soporta eventos primitivos y compuestos [14].

Chimera es un lenguaje de BD conceptual basado sobre un modelo de datos orientado a objetos y ofrece funcionalidad activa y deductiva [20]. El sistema Ode se define y manipula usando el lenguaje de programación de BD O++, el cual extiende C++ con servicios para BDs. Provee dos categorías de reglas que están semánticamente divididas en restricciones y disparos, cada una con diferente sintaxis y modelo de ejecución [21].

El principal objetivo del proyecto *SAMOS* fue el desarrollo de un sistema manejador de BDOO activo. Junto con los lenguajes propuestos en los proyectos Sentinel y Ode, el lenguaje de SAMOS fue de los primeros lenguajes de definición de eventos [14]. Provee servicios activos para la BDOO comercial ObjectStore. La característica más importante de este sistema consiste en el detector de eventos, la semántica de éste se basa en redes de Petri y se implementa usando una estructura gráfica que refleja la estructura de la red de Petri [13].

Sentinel, es una extensión activa de C++ basado sobre el sistema OpenOODB de Texas Instruments. El enfoque de este sistema está en suministrar mecanismos de especificación de eventos detallados, representación de reglas como objetos de la BD e integración del sistema de reglas con un manejo de transacciones sofisticadas [13].

El proyecto *REACH* se concentró en construir un sistema manejador de BDOO con una funcionalidad activa completa. Se consideró importante puesto que proyectos previos habían especificado un amplio rango de funcionalidad, pero no un sistema robusto con funcionalidad activa completa que pudiera ser utilizado por las aplicaciones que estaban disponibles [14].

2.3. Propiedades de las reglas ECA

Existen ciertas características del comportamiento de una regla que pueden ser analizadas. Aquí presentamos dos de ellas.

1. **Terminación**.

   El procesamiento de una regla puede no terminar siempre que exista un ciclo en una gráfica en la cual los nodos representan reglas y los arcos representan relaciones que pueden dispararse. Por ejemplo, en la figura 2.3 el disparo de las reglas R1 ó R3 puede iniciar una serie de disparos de reglas que no termina.

Un análisis estático de una base de reglas puede indicar si el conjunto de reglas puede no terminar.

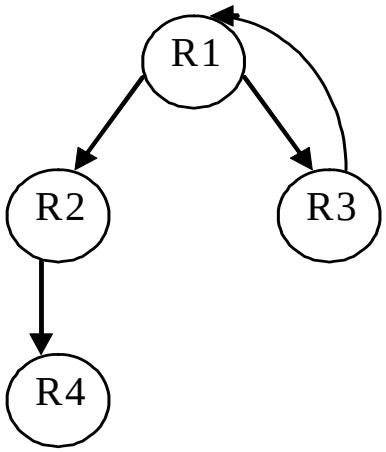

Figura 2.3: Gráfica de no terminación

2. **Confluencia**.

Puede entenderse considerando que el disparo de una regla en un estado de la BD puede llevar a la creación de un nuevo estado. Si más de una regla se dispara al mismo tiempo, entonces existe más de un estado sucesor potencial, como se ilustra en la figura 2.4. Los estados S2 y S3 son los sucesores de S1 que resultan del disparo de las reglas R_i y R_j, respectivamente. Una base de reglas es confluente si para dos reglas R_i y R_j disparadas en un estado inicial S1, se garantiza que se llegará a un solo estado final sin importar el orden en el que alguna secuencia simultánea de disparo de reglas se seleccione para dispararse (representado como * en la figura 2.4).

2.4. Complejidad de la interacción de las reglas ECA

Aún cuando existe un soporte para los mecanismos activos no se puede tener la certeza de que éstos vayan a ser utilizados. Este problema se produce por alguna de las dificultades siguientes [13]:

1. No es fácil decidir qué partes de la aplicación deben tener soporte usando mecanismos activos y qué partes deben emplear otras técnicas.

2. Qué precio hay que pagar en el funcionamiento del sistema por el uso de las reglas.

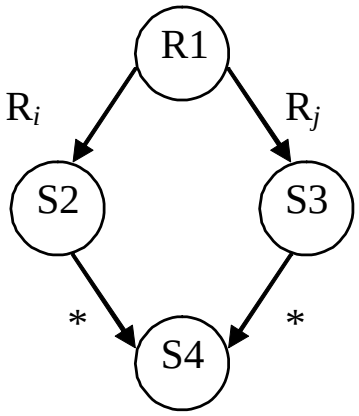

Figura 2.4: Gráfica de Confluencia

3. La funcionalidad de una base de reglas muy grande puede ser difícil de entender.

4. Las herramientas asociadas con un sistema de reglas activo pueden ser mínimas y/o con poco soporte.

Estos puntos reflejan la necesidad de diseñar metodologías, técnicas de análisis de reglas y herramientas para eliminar errores. En particular, es importante tener en cuenta la funcionalidad de la base de reglas cuando existe una alta interacción entre las reglas, ya que la dependencia implícita entre ellas puede ocasionar problemas en el mantenimiento general de la base de reglas.

Para solventar este problema, debemos analizar la complejidad de la interacción de las reglas ECA.

La complejidad de una regla ECA se puede dividir en: complejidad *inter - regla* y complejidad *intra - regla* [10]. La primera se refiere a la complejidad de la regla misma, mientras que la segunda depende de la interacción implícita entre las reglas. El grado de interacción determina la complejidad más que las reglas por separado. De forma más concreta, dos aspectos tienen especial significado en la interacción de las reglas:

- La profundidad del razonamiento. Refleja la intuición de que entre más grande sea el flujo de ocurrencias de eventos que conforman el contexto de una regla, más difícil será entender esa regla. Las reglas que se disparan por eventos compuestos serán más difíciles de entender que aquellas reglas que se disparan por eventos primitivos.

- La amplitud del razonamiento. Refleja la complejidad de la línea de razonamiento que conecta la regla con el contexto donde se produjo.

2.5. Comentarios finales

Las BDA, como una mejora de las BD tradicionales, permiten especificar acciones que se ejecutan automáticamente cuando sucede un evento y se cumplen algunas condiciones. Este mecanismo se denomina reglas evento - condición - acción o, simplemente, reglas ECA.

Las características de las reglas ECA se pueden describir a través de los modelos de conocimiento y de ejecución. El modelo de conocimiento indica qué se puede decir acerca de las reglas ECA de forma individual. El modelo de ejecución indica la forma en la que un conjunto de reglas se trata en tiempo de ejecución.

La utilidad del comportamiento activo ha sido aprovechada por sistemas relacionales y sistemas orientados a objetos. Ejemplos de sistemas relacionales con comportamiento activo son: Starburst, Postgres, Ariel y SQL-3. Sistemas orientados a objetos que han incorporado las reglas activas son: HiPAC, EXACT, NAOS, Chimera y Ode.

De forma general, para implementar un sistema activo son necesarias las siguientes etapas: detector de eventos, monitor de condiciones, calendarizador y evaluador de consultas.

El desarrollo de aplicaciones activas tiene algunas limitantes. Aún cuando existe un soporte para los mecanismos activos no se puede tener la certeza de que vayan a ser utilizados. El diseño de reglas todavía no tiene una técnica definida y se deben detectar problemas de no terminación y confluencia. Un aspecto importante en el mantenimiento de la base de reglas es la complejidad de la interacción de los disparos, un conjunto pequeño de reglas puede hacer que el sistema sea inmanejable si interactúan mucho entre sí.

Para medir la complejidad de la interacción de las reglas ECA se utilizan las métricas de software. Estas métricas deben adherirse a la ciencia de la medición para que sean válidas y aceptadas. Una forma de validar las métricas es usando la teoría de la medición.

Capítulo 3

Fundamentos de la Teoría de la Medición

Los seres humanos emplean la medición de forma cotidiana. Por ejemplo, los precios actúan como una medida del valor de un artículo en una tienda y la talla de una persona se usa para adquirir prendas del tamaño correcto. Estos ejemplos presentan una descripción de situaciones en las que es necesario una medida. Podemos observar que todos ellos tienen un denominador común: en cada caso un aspecto de un objeto se asigna como una descripción del mismo, lo que permite comparar un objeto con algún otro. En la tienda se pueden comparar diferentes productos por medio de su precio y en la tienda de ropa se pueden comparar prendas usando la talla.

En el caso de la ingeniería de software, el poder medir la calidad de los sistemas de software se ha convertido en un aspecto esencial. Algunos desarrolladores miden las características del software para verificar si los requerimientos son consistentes y se han cumplido, si el diseño es de alta calidad y si el código está listo para ser examinado.

Sin embargo, aún cuando es importante la medición en la ingeniería de software la definición de criterios (medidas) razonables, su análisis estadístico y la búsqueda de patrones entre variables no son actividades sencillas. En años recientes, la teoría de la medición se ha propuesto como un medio para evaluar las medidas de software planteadas en la literatura, para establecer criterios para las técnicas estadísticas que se usan en el análisis de los datos y en la búsqueda de patrones [1]. La teoría de la medición es un marco de trabajo teórico conveniente para definir explícitamente las teorías fundamentales sobre las cuales se basan las mediciones de la ingeniería de software. Esto significa que las medidas de software no se definen fuera de un contexto y que las teorías sobre las cuales se basan se pueden discutir, adaptar y refinar.

En este capítulo se presentan los fundamentos de la medición para tener una noción general de los aspectos involucrados con la misma. La estructura del capítulo es la siguiente: la sección 3.1 muestra, de forma general, conceptos básicos de medición así como razones para realizarla; la sección 3.2 expone los fundamentos de la teoría de la medición; la sección 3.3 muestra el marco de trabajo para mediciones de software desarrollado por el Dr. Zuse. Finalmente, la sección 3.4 expone los comentarios finales del capítulo.

3.1. Introducción a la Medición

En esta sección mostramos conceptos importantes acerca de la medición así como las razones por las que es importante realizarla en el ámbito del software.

3.1.1. Medición: ¿Qué es? y ¿Por qué hacerla?

En las primeras etapas del desarrollo científico, la medición se realizaba en un nivel básico: únicamente se clasificaban los objetos [2]. Sin embargo, la clasificación no es la forma más efectiva de medir. Cuando se lleva a cabo una medición que va más allá de una simple clasificación se pueden empezar a diferenciar más características de los objetos medidos [3]. Por ejemplo, se puede hacer algo más que distinguir entre objetos calientes y fríos; se pueden asignar grados de temperatura.

Para capturar la noción de medición, se define el concepto de la siguiente manera [1]:

Definición 3.1 *Medición es el proceso mediante el cual se asignan números o símbolos a atributos de entidades en el mundo real, de forma que los describan de acuerdo a reglas claramente definidas.*

La medición capta la información acerca de atributos de entidades. Una **entidad** es un objeto (tal como una persona o una habitación) o un evento (tal como un viaje o la fase de pruebas de un proyecto de software) en el mundo real [1], [4]. Se puede describir una entidad identificando aquellas características que son relevantes y que permiten distinguirla de otra. Las características o propiedades particulares de una entidad se denominan **atributos**.

Frecuentemente se habla de entidades y atributos de manera indistinta, por ejemplo, la frase “ella es más alta que él” en realidad significa que “la estatura de ella es mayor que la de él”. Es ambigüo decir, por ejemplo, que se “mide una habitación” puesto que se puede medir su longitud o área. En otras palabras, no es correcto decir que se miden objetos o se miden atributos porque, de hecho, se miden atributos de objetos.

Cuando se describen entidades por medio de sus atributos, a menudo se emplean números o símbolos para preservar las relaciones que se observan entre las entidades. Esos números y símbolos son abstracciones que se usan para reflejar la percepción que se tiene del mundo real. De esta forma, la altura se puede definir en términos de centímetros o pulgadas.

En la ingeniería de software la medición se considera un “lujo” ya que es difícil establecer los objetos de medición de los productos de software. Por ejemplo, un desarrollador promete que su producto será amigable con el usuario, confiable y de fácil mantenimiento, sin especificar clara y objetivamente lo que significan esos términos. Como resultado, cuando el proyecto se completa, no se puede decir si se han cumplido los objetivos.

Para solventar este problema, una de las primeras etapas que se debe llevar a cabo cuando se intenta realizar una medición, es identificar aquellos objetos que pueden ser medidos. Una vez identificados se puede medir un atributo. En la ingeniería de software, los objetos se pueden agrupar en cuatro tipos: *procesos*, *proyectos*, *productos* y *recursos* [5]. Cada clase de objeto tiene dos tipos de atributos:

1. Internos. Aquellos atributos que pueden ser medidos únicamente en términos del objeto mismo. Por ejemplo, la longitud es un atributo interno de un documento de software [4].

2. Externos. Aquellos atributos que sólo pueden ser medidos con respecto a la forma en que el objeto se relaciona con su entorno. Por ejemplo, la confiabilidad de un programa depende no sólo del programa mismo, sino del compilador, las características de la computadora en la que se ejecute y el usuario [4].

En el desarrollo de un producto de software intervienen diferentes personas. Cada una tiene diferentes necesidades de información así como diversas formas de utilizarla. Algunos usos comunes que las personas les dan a los datos son los siguientes: estimación, predicción, evaluación, comparación e investigación.

La medición es importante para tres actividades básicas en la ingeniería de software [1]. Primero, hay medidas que ayudan a *entender* qué sucede durante el desarrollo y mantenimiento de los sistemas.

Segundo, la medición permite *controlar* lo que sucede en el proyecto. Usando los objetivos establecidos y entendiendo las relaciones entre entidades se pueden predecir algunos eventos y hacer cambios a los procesos y productos de forma que se puedan cumplir los objetivos. Por ejemplo, se puede monitorear la complejidad de módulos de código, haciendo una revisión minuciosa de aquellos que excedan los límites establecidos.

Tercero, la medición ayuda a *mejorar* los procesos y productos.

3.1.2. Modelos de Medición

En general, un **modelo** es una abstracción de la realidad. Permite obtener un mayor detalle y ver a una entidad desde una perspectiva particular. Por ejemplo, los modelos de costos permiten examinar sólo aquellos aspectos del proyecto que contribuyen al costo final del mismo. Los modelos se presentan en diferentes formas: como ecuaciones o diagramas, por ejemplo. Cada una de ellas muestra cómo se relacionan los componentes y así se pueden examinar y entender tales relaciones y emitir juicios acerca de ellas.

Por ejemplo, cuando se mide la estatura de las personas se deben entender y enunciar las suposiciones acerca del atributo para evitar mediciones ambigüas, es decir, se tiene que especificar una cierta postura así como si se permite el uso de zapatos. En este sentido, se está definiendo, de hecho, un modelo de una persona más que la persona en sí misma.

Cuando se mide, existe siempre el riesgo de enfocarse mucho más en el sistema formal (matemático) y no lo suficiente en el sistema empírico. Se tiende a apresurarse en crear mapeos y manipular números, sin tener cuidado de las relaciones entre entidades y sus atributos en el mundo real.

En muchas situaciones se usan mapeos directos de un atributo en un número y se usa este número para dar respuestas o verificar datos. Sin embargo, cuando existen relaciones complejas entre atributos o cuando un atributo debe ser medido mediante la combinación de diversos aspectos, entonces se necesita un modelo que combine medidas relacionadas. Esta es la razón para diferenciar entre **medición directa** y **medición indirecta**.

La medición directa del atributo de una entidad no involucra ningún otro atributo o entidad. Por ejemplo, la longitud de un objeto físico se puede medir sin hacer referencia a otro objeto o atributo. Por otro lado, la densidad de un objeto físico puede medirse indirectamente en términos de la masa y el volumen; entonces se usa un modelo para mostrar que la relación entre los tres aspectos es:

$$densidad = \frac{masa}{volumen}$$

Las siguientes medidas directas son comunes en la ingeniería de software:

- *Longitud* del código fuente (medida por líneas de código)
- *Duración* del proceso de pruebas (medido por tiempo transcurrido en horas)
- *Número de defectos descubiertos* durante el proceso de prueba (medido por conteo de defectos)

- *Tiempo* que un programador emplea en un proyecto (medido por meses trabajados)

La tabla 3.1 muestra algunos ejemplos de medidas indirectas que se emplean comunmente en la ingeniería de software. La más común de todas, y la más controversial, es la medida para la productividad del programador debido a que no toma en cuenta la funcionalidad o complejidad del código.

La medición indirecta es útil para visualizar las interacciones entre mediciones directas, es decir, algunas veces es más fácil observar lo que sucede con un proyecto usando combinaciones de medidas.

Productividad del programador	$\frac{\text{Líneas de código}}{\text{Meses de esfuerzo}}$
Densidad de defectos de un módulo	$\frac{\text{Número de defectos}}{\text{Tamaño del módulo}}$
Eficiencia de detección de defectos	$\frac{\text{Número de defectos detectados}}{\text{Número total de defectos}}$

Tabla 3.1 Ejemplos de medidas indirectas comunes usadas en la ingeniería de software

3.2. Teoría de la medición

De forma ordinaria, cuando se miden objetos, no se piensa acerca del principio científico que se está aplicando. Para medir un atributo de un objeto se utilizan herramientas y principios que ahora se dan por sentados. Sin embargo, se han desarrollado modernos dispositivos y técnicas de medición a lo largo del tiempo, basándose en un mayor entendimiento de los atributos que se miden.

Desafortunadamente, en el área de software no se tiene un entendimiento profundo de las cualidades que lo caracterizan ni se cuenta con herramientas de medición sofisticadas. Por lo tanto, las preguntas que son relativamente fáciles de contestar para entidades que no son de software son difíciles de contestar para entidades de software. Ejemplo de ello son las siguientes preguntas:

1. ¿Cuánto se debe saber acerca de un atributo antes de considerar medirlo?
2. ¿Cómo se puede determinar si realmente se ha medido el atributo que se quiere medir?
3. Usando la medición, ¿qué enunciados significativos se pueden hacer acerca de un atributo y de las entidades que lo poseen?
4. ¿Qué operaciones significativas se pueden llevar a cabo sobre las medidas?

Para contestar esas preguntas, se deben establecer los fundamentos de la teoría de la medición. Se debe examinar la teoría de la medición formal, desarrollada como una disciplina clásica de las ciencias físicas. Asímismo, hay que plantear cómo se aplican los conceptos de la teoría de la medición a la ingeniería de software.

3.2.1. La representación de la teoría de la medición

La representación de la teoría de la medición formaliza la intuición que se tiene acerca de la manera en que funciona el mundo. Esto es, los datos que se obtienen como resultado de una medición deben representar atributos de las entidades observadas y la manipulación de los datos debe preservar las relaciones que se observan entre las entidades. Precisamente, la intuición es el punto de inicio para todas las mediciones.

Si se considera la forma en que se percibe el mundo real, las personas tienden a entender las cosas comparándolas sin asignarles números. Por ejemplo, en la figura 3.1 podemos observar que ciertas personas son más altas que otras sin haberlas medido (en la figura 3.1 las personas se representan como barras). Así, podemos decir que la persona A (PA) es más alta que la persona B (PB) quien, a su vez, es más alta que la persona C (PC). La relación “más alto que” observada entre las personas se denomina **relación empírica**.

Cuando dos personas se miden y sus estaturas correspondientes son muy similares, se puede encontrar una diferencia de opinión. Algunas personas pueden pensar que un cierto individuo A es más alto que un individuo B, mientras que otras personas están convencidas de lo contrario. Las relaciones empíricas permiten estas diferencias pero necesitan un consenso de opinión acerca de las relaciones en el mundo real. Una relación empírica es aquella para la cual existe un consenso razonable acerca de los elementos en la relación.

Se puede definir más de una relación empírica sobre el mismo conjunto. Por ejemplo, la figura 3.1 también muestra la relación “mucho más alto que”. Puede observarse que PA y PB son mucho más altos que PC.

Las relaciones empíricas no son necesariamente binarias. Esto es, se puede definir una relación sobre un elemento único del conjunto o una colección de elementos. La relación “es alto” es un ejemplo de una relación unaria. En la figura 3.1 se puede ver que PA es alto pero PC no lo es. De forma similar, se puede definir una relación ternaria, la figura 3.1 muestra que PC sentado sobre los hombros de PB debería ser más alto que PA.

Podemos pensar en las relaciones anteriores como mapeos del mundo real a un mundo matemático for-

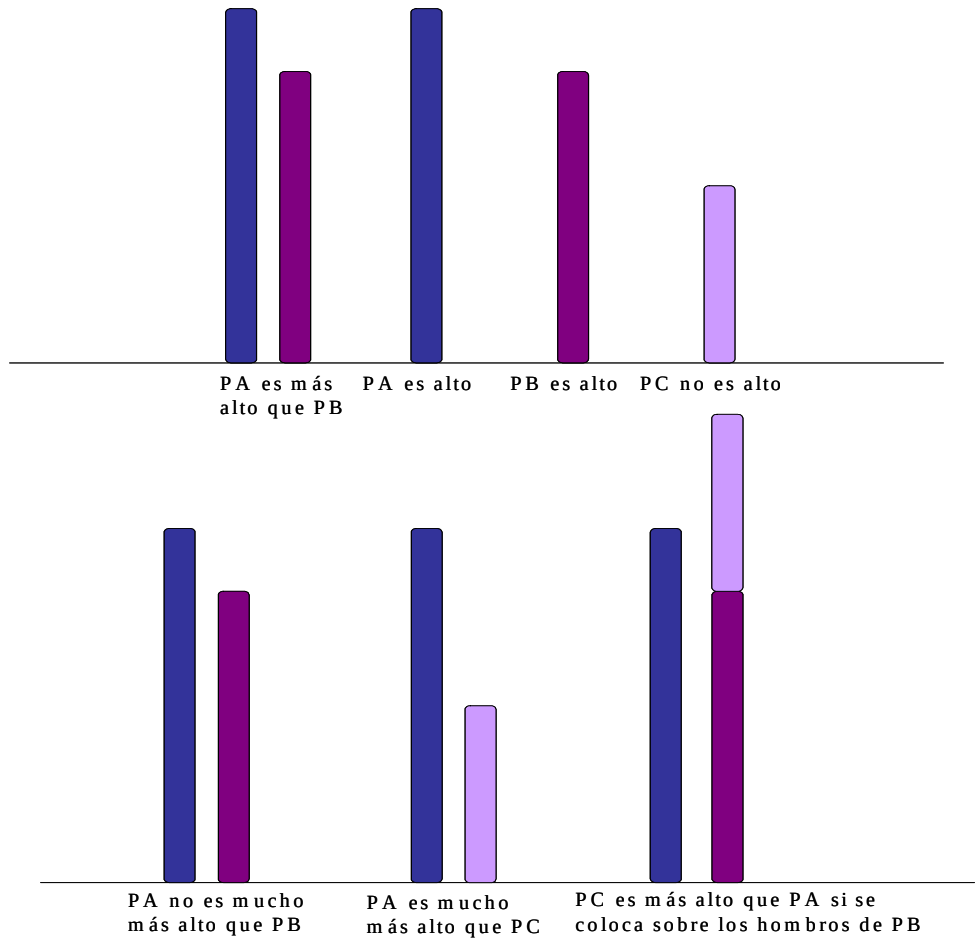

Figura 3.1: Algunas relaciones empíricas para el atributo “estatura”

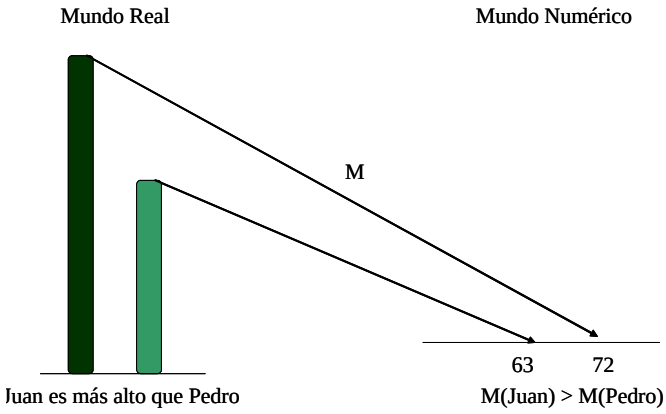

Figura 3.2: Condición de la representación

mal. Tenemos entidades y sus atributos en el mundo real y definimos un mapeo matemático que preserva las relaciones observadas. De esta forma, la altura (estatura) se puede considerar como un mapeo del conjunto de personas al conjunto de los números reales. Si estamos de acuerdo con que PA es "más alto" que PC, entonces la medición de la altura debe asignar un número mayor a PA que a PC.

Cada relación en un sistema empírico corresponde, vía medición, a un elemento en un sistema numérico. Se pretende que el comportamiento de las medidas en el sistema numérico sea el mismo que presentan los elementos en el mundo real, de forma que por medio del estudio de los números se aprenda del mundo real. Esta regla se llama la *condición de representación*, y se ilustra en la figura 3.2. La representación de la condición afirma que un mapeo *M* debe mapear entidades en números y relaciones empíricas en relaciones numéricas de forma que las relaciones empíricas se preserven y sean preservadas por las relaciones numéricas. En la figura 3.2 se observa que la relación empírica "más alto que" se mapea a la relación numérica "$>$".

En particular, podemos decir que:

$$Juan\ es\ más\ alto\ que\ Pedro\ si\ y\ sólo\ si\ M\left(Juan\right) > M\left(Pedro\right)$$

Este enunciado implica que:

- Siempre que Juan sea más alto que Pedro, entonces $M\left(Juan\right)$ debe ser un número más grande que $M\left(Pedro\right)$

- Se puede mapear a Juan a un número más alto que Pedro solamente si Juan es más alto que Pedro

Ya hemos mencionado que pueden existir varias relaciones sobre un conjunto dado. El siguiente ejemplo muestra otras relaciones que se pueden establecer entre los elementos del conjunto de personas.

Ejemplo 3.1 *Para la relación empírica (binaria) "más alto que", se tiene la relación numérica*

$$x > y$$

Entonces, la condición de la representación requiere que para una medida M,

$$\textit{A es más alto que B si y sólo si } M(A) > M(B)$$

Para la relación empírica (unaria) "es alto", se podría tener la relación numérica

$$x > 70$$

La condición de la representación requiere que para una medida M,

$$\textit{A es alto si y sólo si } M(A) > 70$$

Para la relación empírica (binaria) "mucho más alto que", se podría tener la relación numérica

$$x > y + 15$$

La condición de la representación requiere que para alguna medida M,

$$\textit{A es mucho más alto que B si y sólo si } M(A) > M(B) + 15$$

Para la relación empírica (ternaria) "x es más alto que y si está sentado sobre los hombros de z", se podría tener la relación numérica

$$0{,}7x + 0{,}8z > y$$

La representación de la condición requiere que para alguna medida M,

$$\textit{A es más alto que B sentado sobre los hombros de C si y sólo si } 0.7M(A) + 0{,}8M(C) > M(B)$$

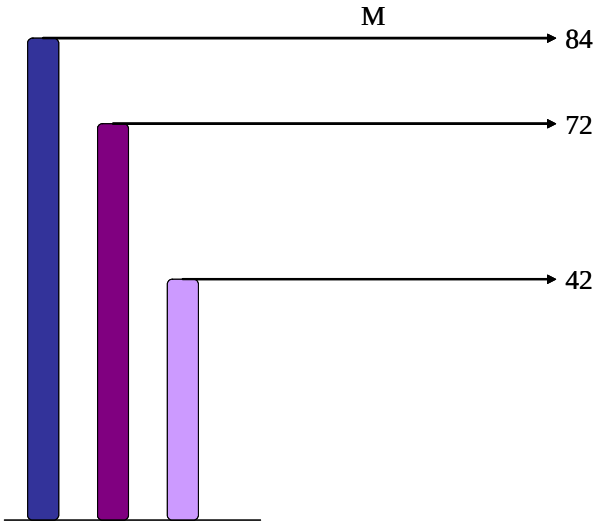

Figura 3.3: Mapeo del mundo real al mundo numérico

Si se considera la asignación de números *M* de la figura 3.3. PB se mapea al número real 72 (esto es, $M\,(PB) = 72$), PA al número 84 ($M\,(PA) = 84$) y PC al número 42 ($M\,(PC) = 42$). Con este mapeo particular *M*, las cuatro relaciones numéricas anteriores se cumplen siempre que las cuatro relaciones empíricas se cumplan. Por ejemplo:

- PA es más alto que PB, y $M\,(PA) > M\,(PB)$
- PB es alto, y $M\,(PB) = 72 > 70$
- PA es mucho más alto que PC, y $M\,(PA) = 84 > 57 = M\,(PC) + 15$. De forma similar, PB es mucho más alto que PC, y $M\,(PB) = 72 > 57 = M\,(PC) + 15$
- PC, cuando se sienta sobre los hombros de PB, es más alto que PA, y $0{,}7M\,(PC) + 0{,}8M\,(PB) = 87 > M\,(PA)$

Debido a que todas las relaciones se preservan de esta forma por el mapeo, entonces se puede definir el mapeo como una medida para el atributo.

No todas las asignaciones satisfacen la condición de la representación. Por ejemplo, si se define el mapeo de la siguiente manera:

$M(PB) = 72$
$M(PA) = 84$
$M(PC) = 60$

entonces tres de las relaciones anteriores se satisfacen, sin embargo, la relación “mucho más alto que” no se cumple. Esto se debe a que la cláusula “PB es mucho más alto que PC” no es verdadera bajo este mapeo.

El mapeo del mundo real al mundo numérico algunas veces se denomina *representación* u *homomorfismo*, debido a que la medida representa el atributo en el mundo numérico.

Una vez que hemos introducido de manera intuitiva algunos aspectos importantes de la medición podemos trasladarlos a un plano más formal. De esta forma, decimos que un *sistema de relación empírico* se conforma por [2]: $\mathbf{A} = (A, \cdot \geqslant, o)$ donde **A** es un conjunto no vacío de objetos, $\cdot \geqslant$ es una relación empírica sobre A y o es una operación binaria cerrada sobre A. Desde luego, son posibles más de una relación y operación binaria sobre *A*. De acuerdo a [12], se supone que para un sistema de relación empírica **A** “existe una interpretación empírica bien establecida para los elementos de *A* y para cada relación $\cdot \geqslant$ de *A*”. Se hace la misma suposición para la(s) operacion(es) de concatenación.

Además del sistema de relación empírico necesitamos la definición del sistema de relación numérico. Un *sistema de relación numérico* se conforma por [2]: $\mathbf{B} = (\Re, \geqslant, +)$ donde $\Re$ es el conjunto de los números reales, $\geqslant$ es una relación sobre $\Re$ y + es una operación binaria cerrada sobre $\Re$. Desde luego, son posibles más de una relación y operación binaria sobre B. También se incluye el caso en el que no existen relaciones u operaciones. El signo + denota adición. La operación numérica + es un caso especial y se puede reemplazar por alguna operación numérica arbitraria $\otimes$. En ese caso, el sistema de relación numérica se escribiría como: $\mathbf{B} = (\Re, \geqslant, \otimes)$.

Una vez definidos los dos sistemas anteriores, podemos decir que una medida [2] es un mapeo u: $A \rightarrow \Re$ tal que, para todos los elementos a, b $\in A$, se cumple:

$$a \cdot \geqslant b \Longleftrightarrow u(a) \geqslant u(b)$$

El mapeo entre objetos del mundo real y objetos del mundo numérico está restringido por dos tipos de teoremas: la representación del teorema y la unicidad del teorema.

La representación del teorema [5] (o representación de la condición) afirma que si un sistema de relación dado, satisface ciertos axiomas, entonces se puede construir un homomorfismo. La unicidad del teorema [1] (o unicidad de la condición) define las transformaciones matemáticas que mapean un homomorfismo en

otro. Dicho de otra forma, la unicidad del teorema define las relaciones entre diferentes mapeos o representaciones. Por ejemplo, si PA es más alto que PB, entonces $M\,(PA) > M\,(PB)$ sin importar si la medida está en pulgadas, pies o centímetros. Sin embargo, hay dos representaciones *M* y *M'* que se relacionan de forma específica: existe una constante $c > 0$ tal que *M = cM'* (donde *M* es la representación de la estatura en pulgadas y *M'* es la representación de la estatura en centímetros, *c = 2.54*). La transformación de una representación válida a otra se llama *transformación admisible [4]*.

La mayoría de las estructuras de relación involucran una relación de orden. El orden implica las nociones de completitud y transitividad. Estas nociones son capturadas por el orden débil. Acerca del orden débil decimos que dado un conjunto *A* y una relación binaria $\succcurlyeq$ sobre *A*, el sistema de relación $(A, \succcurlyeq)$ es un órden débil si y sólo si para todos los elementos a, b $\in$ *A*, se satisfacen los siguientes axiomas:

1. Completo: a $\succcurlyeq$ b ó b $\succcurlyeq$ a
2. Transitivo: Si a $\succcurlyeq$ b y b $\succcurlyeq$ c, entonces a $\succcurlyeq$ c

3.2.2. Medición extensiva

En el ejemplo de la medición de las personas, cuando PC se coloca sobre los hombros de PB, se lleva a cabo un proceso de *medición extensiva [12], [2]*. Este tipo de medición se basa en un procedimiento que se denomina *conteo de unidades [5], [12]*, el cual se bosqueja a continuación.

Supongamos que se tienen tres barras - *a*, *b* y *c* -. Se toman dos barras, *a* y *b*, y se mide su altura. Puede suceder que *a* sea más larga que *b* ($a \succ b$) ó que *b* sea más larga que *a* ($b \succ a$). La *operación de concatenación* consiste en tomar dos barras y colocarlas una al final de la otra, de manera que se forme una línea. Esta operación se denota por $a \circ b$. Se construye una *secuencia estándar* seleccionando una de las barras como unidad de medida. Entonces se crea un número de "copias perfectas" de la barra seleccionada y se miden las otras barras concatenando las copias de la unidad de medida (barra seleccionada) hasta que se haya aproximado la longitud de la barra que se está midiendo. Si tres copias de la barra *c* son más pequeñas que la barra *b*, y cuatro copias de la barra *c* son más grandes que la barra *b*, se dice que la longitud de la barra *b* está entre 3*c* y 4*c*. Si se le da a la longitud de la barra *c* un nombre, por ejemplo, un pie, se puede decir que la longitud de la barra *b* está entre 3 y 4 pies. Este procedimiento es la medición extensiva.

De forma general, la medición extensiva es un conjunto de suposiciones o axiomas, formulados en términos de un ordenamiento $\succcurlyeq$ de objetos con respecto a un atributo y a una operación de concatenación $\circ$ entre objetos que permiten la construcción de una escala ϕ que satisface [12]:

1. $a \succcurlyeq b$ si y sólo si $\phi(a) \geqslant \phi(b)$,
2. $\phi(a \circ b) = \phi(a) + \phi(b)$.

El siguiente conjunto de axiomas son necesarios y suficientes para la medición extensiva cuando la operación de concatenación es cerrada, es decir, cuando dos objetos se pueden concatenar. Antes es necesario aclarar que $a \sim b$ si y sólo si $a \succcurlyeq b$ y $b \succcurlyeq a$; y $a \succ b$ si y sólo si $a \succcurlyeq b$ y no$(b \succcurlyeq a)$

Definición 3.2 *Dado un conjunto no vacío A, una relación binaria $\succcurlyeq$ sobre A, y una operación binaria cerrada $\circ$ sobre A, la tercia $\langle A, \succcurlyeq, \circ \rangle$ es una estructura extensiva cerrada si y sólo si se cumplen los siguientes axiomas para todos los elementos a, b, c, d ∈ A:*

1. Orden débil: $\langle A, \succcurlyeq \rangle$ es un orden débil, es decir, $\succcurlyeq$ es una relación reflexiva, transitiva y completa
2. Asociación débil: $a \circ (b \circ c) \sim (a \circ b) \circ c$
3. Monotonía: $a \succcurlyeq b$ si y sólo si $a \circ c \succcurlyeq b \circ c$ si y sólo si $c \circ a \succcurlyeq c \circ b$
4. Arquímedes: Si $a \succ b$, entonces para cualquier c, d ∈ A,
existe un entero positivo n tal que $na \circ c \succcurlyeq nb \circ d$, donde na se define inductivamente como $1a = a$, $(n+1)\,a = na \circ a$

La estructura se llama positiva si, además, se satisface:

5. Positiva: $a \circ b \succ a$

En el contexto del axioma de Arquímedes, se define la propiedad de idempotencia:

Definición 3.3 *Una operación de concatenación es idempotente, si para todos los objetos $a \in A$ se cumple:*

$$a \circ a = a$$

Siempre que una operación de concatenación sea idempotente, no hay forma de que el axioma de Arquímedes se pueda cumplir. Como consecuencia, no existe una estructura extensiva.

3.2.3. Escalas de medición

El propósito de llevar a cabo el mapeo de entidades empíricas a entidades numéricas es poder manipular los datos en el sistema numérico y usar los resultados para bosquejar resultados acerca de los atributos en

el sistema empírico. Sin embargo, no todos los mapeos son el mismo. Las diferencias entre los mapeos restringen el tipo de análisis que se puede realizar. Para entender tales diferencias, existe la noción de escala de medición.

Una escala de medición se encuentra formada por el triple $(\mathbf{A}, \mathbf{B}, u)$ [6], donde:

A es un sistema de relación empírica

B es un sistema de relación numérica

u es un mapeo de $A \rightarrow B$

Las escalas de medición se clasifican en cinco tipos principales:

- Nominal
- Ordinal
- Intervalo
- Razón
- Absoluta

1. Escala Nominal

En esta escala se definen clases o categorías y se coloca cada entidad en una clase o categoría, basándose en el valor del atributo. Esta clasificación es la base para la forma más primitiva de medición. La escala nominal tiene las siguientes características:

- El sistema de relación empírico consiste de diferentes clases; no existe noción de ordenamiento entre las clases
- Cualquier enumeración distinta o representación simbólica de las clases es una medida aceptable, pero no hay noción de la magnitud asociada a los números o a los símbolos

En otras palabras, la escala nominal coloca a los elementos en un esquema de clasificación. Las clases no están ordenadas. Incluso si las clases se numeran de 1 a *n* para identificarlas, eso no implica un ordenamiento de las clases.

2. Escala Ordinal

   La escala ordinal incorpora información acerca del ordenamiento de las clases o categorías. El ordenamiento conduce a análisis que no son posibles con mediciones nominales. Este tipo de escala tiene las siguientes propiedades:

- El sistema de relación empírica consiste de clases que se ordenan con respecto a un atributo
- Un mapeo que preserva el ordenamiento (esto es, una función monótona) es aceptable
- Los números representan una clasificación, entonces, la suma, resta y otras operaciones aritméticas no tienen significado.

Debido a que el mapeo para una escala ordinal conserva el orden de las clases, el conjunto de clases ordenadas $\langle C_1, C_2, ..., C_n \rangle$ se mapea a una serie creciente de números $\langle a_1, a_2, ..., a_n \rangle$ donde a_i es mayor que a_j cuando *i* es mayor que *j*.

3. Escala de Intervalo

Esta escala captura la información acerca del tamaño de los intervalos que separan las clases. La escala de intervalo se caracteriza por lo siguiente:

- Conserva el orden
- Conserva la diferencia pero no los cocientes. Esto es, se conoce la diferencia entre cualesquiera dos clases pedidas en el rango del mapeo, pero no tiene sentido calcular el cociente de dos clases en el rango del mapeo.
- Suma y resta son aceptables en la escala de intervalo, no así la multiplicación y división

Si un atributo es medible sobre una escala de intervalo y *M* y *M'* son mapeos que satisfacen la condición de la representación, entonces, se pueden encontrar números *a* y *b* tales que:

$$M = aM' + b$$

Este tipo de transformación se denomina **transformación afín**. Todas las transformaciones afines son permisibles para la escala de intervalo. Un ejemplo de transformación afín es la conversión de grados Celsius a grados Fahrenheit:

$$F = \frac{9}{5}C + 32$$

4. Escala de Razón

La escala de razón tiene las siguientes características:

- Conserva el orden, el tamaño de los intervalos entre entidades y los cocientes entre entidades
- Existe el elemento cero, el cual representa una carencia total del atributo
- Debe comenzar en cero e incrementarse en intervalos iguales, conocidos como unidades
- Toda la aritmética se puede aplicar significativamente a las clases en el rango del mapeo

La característica distintiva de esta escala es la existencia de relaciones empíricas que capturan cocientes. En general, cualquier transformación aceptable para una escala de razón es un mapeo de la forma:

$$M = aM'$$

Donde *a* es un escalar positivo.

5. Escala Absoluta

La escala absoluta es la más restrictiva de todos los tipos de escalas. Para dos medidas, M y M', sólo existe una transformación admisible: la transformación identidad. Las características de este tipo de escala son las siguientes:

- La medición para una escala absoluta se hace por conteo del número de elementos en el conjunto de entidades
- El atributo siempre toma la forma "número de ocurrencias de *x* en la entidad"
- Todos los análisis aritméticos del resultado del conteo son significativos

Se tienen algunos ejemplos de la escala absoluta en la ingeniería de software. Por ejemplo, el número de fallas observadas durante la prueba de integración solamente se puede medir en una forma: mediante el conteo del número de fallas observadas. Asímismo, el número de personas trabajando sobre un proyecto de software se puede medir solamente contando el número de personas.

La tabla 3.2 resume los tipos de escalas y sus transformaciones admisibles.

Tipo de escala	**Transformaciones Admisibles**
Nominal	Mapeo 1 a 1 de *M* a *M'*
Ordinal	Función creciente de *M* a *M'*, esto es, $M(x) \geqslant M(y)$ implica $M'(x) \geqslant M(y)$
Intervalo	$M' = aM + b\ (a > 0)$
Razón	$M' = aM\ (a > 0)$
Absoluta	$M' = M$

Tabla 3.2 Tipos de escalas y sus transformaciones admisibles

El entender los tipos de escalas permite determinar cuándo las declaraciones acerca de una medición tienen sentido. Por ejemplo, no es apropiado calcular cocientes con escalas nominales, ordinales y de intervalo. En general, las medidas mapean atributos a números reales y se intenta manipular éstos en una forma familiar, es decir, sumando, promediando y llevando a cabo procedimientos estadísticos. Sin embargo, se debe recordar que el análisis está restringido por el tipo de escala. En otras palabras, el conocimiento del tipo de escala proporciona las limitaciones sobre el tipo de manipulación matemática que se puede realizar. La tabla 3.3 muestra los tipos de escalas y los cálculos estadísticos que son permisibles.

Tipo de escala	**Relaciones definidas**	**Ejemplos de estadísticas apropiadas**
Nominal	Equivalencia	Modo Frecuencia
Ordinal	Equivalencia Mayor que	Mediana Porcentaje
Intervalo	Equivalencia Mayor que	Desviación estándar
Razón	Equivalencia Mayor que	Coeficiente de variación

Tabla 3.3 Tipos de escalas y cálculos estadísticos permisibles

En la literatura existen marcos de trabajo para mediciones de software [39], [2] que contemplan las características que hemos mencionado. Uno de los marcos principales es el propuesto por el Dr. H. Zuse. En la siguiente sección presentamos sus características principales.

3.3. Marco de Trabajo del Dr. H. Zuse

En el área de ingeniería de software, los atributos de calidad de software se formulan con declaraciones verbales. Ejemplo de ello son los términos productividad, estimación de costos, mantenibilidad, etc. El objetivo de medir estos aspectos es mapearlos a términos cuantitativos y viceversa. Por esta razón es necesario saber cuáles son las condiciones cualitativas detrás de las medidas de software. La estructura extensiva es una de las más importantes estructuras de medición [2].

El marco de trabajo desarrollado por el Dr. H. Zuse [7] captura el ciclo de vida del software e incluye medición de procesos, estimación de costos, diseño de medidas, entre otros aspectos. Está basado en la teoría de la medición. Esta teoría es adecuada para dar respuestas a cuestiones de medición de software, tales como propiedades de las medidas, escalas de medición, validación de medidas y modelos de predicción. El marco de trabajo del Dr. Zuse incorpora extensiones a la teoría de medición para cubrir necesidades especiales en el área de medición del software. A continuación se muestra la definición de la *estructura extensiva modificada* que pertenece a este marco de trabajo.

Definición 3.4 *Dado un conjunto no vacío **P**, una relación* $\cdot \geqslant$ *sobre **P** y una operación binaria cerrada* $\circ$ *sobre P, el sistema de relación* $(\mathbf{P}, \cdot \geqslant, \circ)$ *es una estructura extensiva si y sólo si se cumplen los siguientes axiomas para todos los elementos* $P_1, ..., P_n \in \mathbf{P}$.

1. $(P, \cdot \geqslant)$ *es un orden débil*
2. $P_1 \circ (P_2 \circ P_3) \sim (P_1 \circ P_2) \circ P_3$ *axioma de asociación débil*
3. $P_1 \circ P_2 \sim P_2 \circ P_1$ *axioma de conmutatividad débil*
4. $P_1 \cdot \geqslant P_2 \Rightarrow P_1 \circ P_3 \cdot \geqslant P_2 \circ P_3$ *axioma de monotonía débil*
5. *Si* $P_3 \cdot > P_4$ *entonces para cualquiera* P_1, P_2 *existe un número natural n, tal que,* $P_1 \circ nP_3 \cdot > P_2 \circ nP_4$ *axioma de Arquímedes*

La estructura, además, se denomina positiva si se satisface:

$$P_1 \circ P_2 \cdot > P_1$$

Los axiomas de la estructura extensiva modificada describen condiciones empíricas relacionadas a la operación de concatenación entre objetos.

La estructura extensiva describe las condiciones empíricas para la medición del software y estas medidas dependen de la operación de concatenación seleccionada. Si, por ejemplo, la relación empírica fuera $\cdot \geqslant$ significara "igual o más difícil de mantener", entonces los axiomas de la estructura extensiva proporcionan condiciones intuitivas para hablar acerca del significado de la relación empírica $\cdot \geqslant$.

Otro aspecto importante de este marco de trabajo son las *reglas de combinación* y las *condiciones de independencia*.

Una regla de combinación se define como:

Definición 3.5 *Si una medida u es una escala ordinal ((A, $\cdot \geqslant$),($\Re$, $\geqslant$),u) entonces, una regla de combinación f se define:*

$$\mu(a \circ b) = f(\mu(a), u(b))$$

donde a, b $\in$ A; A es un conjunto de objetos, $\circ$ es una operación binaria de concatenación y u es una medida.

El que una regla de combinación, *f*, exista o no, depende de las condiciones empíricas de independencia. Las condiciones de independencia describen, precisamente, el grado de independencia de dos componentes concatenados. Las condiciones de independencia del marco de trabajo del Dr. Zuse son las siguientes:

Condición C1: $a \sim b \Rightarrow a \circ c \sim b \circ c$, y $a \sim b \Rightarrow c \circ a \sim c \circ b$
Condición C2: $a \sim b \Longleftrightarrow a \circ c \sim b \circ c$, y $a \sim b \Longleftrightarrow c \circ a \sim c \circ b$
Condición C3: $a\cdot \geqslant b \Rightarrow a \circ c\cdot \geqslant b \circ c$, y $a\cdot \geqslant b \Rightarrow c \circ a\cdot \geqslant c \circ b$
Condición C4: $a\cdot \geqslant b \Longleftrightarrow a \circ c\cdot \geqslant b \circ c$, y $a\cdot \geqslant b \Longleftrightarrow c \circ a\cdot \geqslant c \circ b$

Las condiciones de independencia son importantes para la medición del software. Algunas de las razones son las siguientes:

1. Una regla de combinación *f* describe las propiedades de las medidas de software
2. Las condiciones de independencia caracterizan la existencia de una regla de combinación
3. La condición de independencia C1 es la condición más débil para la existencia de una regla de combinación *f*.

4. Si la condición de independencia C1 se viola, no hay forma de llegar a la estructura extensiva.

5. Las condiciones de independencia son importantes para las operaciones de sustitución de componentes en un programa, diagrama de flujo o gráfica de estructura. Para tener propiedades claras con operaciones de sustitución, una medida tiene que asumir la condición de independencia C4.

6. Las condiciones de independencia son importantes para la validación de medidas de software y para modelos de predicción.

7. Si una medida asume las condiciones de independencia C1 - C4, pero no logra una estructura extensiva, entonces el tipo de escala es, aún, ordinal.

Las funciones de confianza proveen una herramienta útil para la cuantificación de juicios subjetivos o personales. Una función de confianza describe la confianza en una hipótesis y la confianza en la combinación de las hipótesis. Este tipo de funciones se utilizan, entre otras cosas, en sistemas expertos.

El Dr. Zuse define la relación y la función modificada de confianza de la siguiente forma:

Dados un conjunto contable X, un conjunto $\Im$ de subconjuntos finitos de X y una relación $\cdot \geqslant$ sobre $\Im$, $\cdot \geqslant$ es una relación modificada de confianza si y sólo si se cumplen las siguientes relaciones:

MRB1: $\forall A, B \in \Im : A\cdot \geqslant B$ ó $B\cdot \geqslant A$
MRB2: $\forall A, B, C \in \Im : A\cdot \geqslant B$ y $B\cdot \geqslant C \Rightarrow A\cdot \geqslant C$
MRB3: $\forall A \supseteq B \Rightarrow A\cdot \geqslant B$ axioma de dominancia
MRB4: $\forall (A \supset B, A \cap C = \varnothing) \Rightarrow (A\cdot > B \Rightarrow A \cup C\cdot > B \cup C)$ mononicidad parcial
MRB5: $\forall A \in \Im : A\cdot \geqslant 0$ positivo

Además, existe una función modificada de confianza tal que:

$$A\cdot \geqslant B \Leftrightarrow u(A) \geqslant u(B)$$

si y sólo si cumple los axiomas de la relación modificada de confianza: MRB1- MRB5.

3.4. Comentarios Finales

El proceso de medición es esencial en la vida cotidiana. Las personas lo utilizan al comparar productos por medio de algún atributo, por ejemplo, el precio o la talla de una prenda. En el ámbito de la ingeniería

de software, la medición es útil para verificar que se han cumplido los requerimientos, para demostrar que el diseño tiene una buena calidad, etc. Sin embargo, la actividad de medición se complica cuando se trata de determinar los atributos que se pueden medir. Una vez cumplida esta etapa, los atributos (medidas) de software se deben evaluar sobre un marco de trabajo que permita darles un sustento formal. Recientemente, se ha propuesto la teoría de la medición para evaluar medidas de software, así como para establecer criterios para las técnicas estadísticas que se usan en el análisis de los datos.

En este trabajo de tesis utilizamos los parámetros de medición de complejidad: número de anclas, distancia y potencial de disparo. Estos parámetros deben ser válidos, es decir, deben cumplir con ciertas características que nos permitan clasificarlos en un tipo de escala y obtener información acerca de las operaciones que podemos realizar. Por lo tanto, nos apoyamos en la teoría de la medición, específicamente en el trabajo del Dr. Zuse, para verificar la validez de los parámetros de medición.

Previo al trabajo de validación, proponemos la utilización de la Red de Petri Coloreada Condicional para la modelación de los parámetros de medición.

Capítulo 4

Red de Petri Coloreada Condicional

Las *Redes de Petri* (PN por sus siglas en inglés) son una herramienta gráfica y matemática para modelar diversos sistemas. Con ellas se pueden describir sistemas concurrentes, asíncronos, distribuidos, paralelos, no determinísticos y/o estocásticos [22].

Como herramienta gráfica, las PN se pueden utilizar como ayuda visual, similar a los diagramas de flujo y diagramas de bloque. Además, los tokens pueden simular las actividades dinámicas y concurrentes de los sistemas.

Como herramienta matemática, es posible establecer ecuaciones de estado, ecuaciones algebraicas y otros modelos matemáticos que describan el comportamiento de los sistemas.

Las PN se clasifican en tres tipos principales [23]: *PN ordinarias*, *PN abreviadas* y *PN extendidas*. En una PN ordinaria todos los arcos tienen peso 1, existe un único tipo de token, la capacidad de un lugar es infinita, no involucran tiempo y el disparo de una transición puede ocurrir si y sólo si cada uno de sus lugares de entrada contiene al menos un token. Las PN abreviadas corresponden a representaciones simplificadas, útiles para hacer más compacta la representación gráfica. Las extensiones corresponden a modelos a los cuales se les han agregado reglas de funcionamiento para enriquecer el modelo inicial haciendo posible que un mayor número de aplicaciones se puedan tratar. Una extensión de las PN, especialmente desarrollada para BDA, es la *Red de Petri Coloreada Condicional* (CCPN).

Este capítulo se encuentra dividido de la siguiente forma: la sección 4.1 muestra algunas definiciones preliminares; la sección 4.2 presenta la definición de la CCPN; la sección 4.3 muestra la modelación y simulación de reglas ECA con la CCPN. En la sección 4.4 se muestra una descripción del sistema ECAPNSim.

Finalmente, la sección 4.5 presenta los comentarios finales del capítulo.

4.1. Conceptos Preliminares

Una PN es un tipo particular de grafo dirigido, junto con un estado inicial llamado la *marca inicial* M_0. La gráfica básica de una PN consiste de dos tipos de elementos, llamados *lugares* y *transiciones*. Ambos elementos se conectan mediante arcos, ya sea de un lugar a una transición o viceversa. En la gráfica, los lugares se dibujan como círculos y las transiciones como barras o rectángulos. Los arcos se etiquetan con un entero positivo, que representa su *peso*. Un arco con peso *k* indica un conjunto de *k* arcos paralelos. Si un arco no tiene etiqueta, su peso es uno. Una *marca* (estado) asigna a cada lugar un entero no negativo que representa el número de tokens colocados en ese lugar. Gráficamente, se colocan *k* puntos negros (tokens) en el lugar *p*. Una marca se denota por M, un vector de *m* componentes, donde *m* es el número total de lugares. El *n*-ésimo componente de M, denotado por $M\,(p)$, es el número de tokens en el lugar *n*.

La definición formal de una PN se da en la tabla 4.1.

Una PN es una 5-tupla, PN = (P, T, F, W, M_0) donde:
$P = \{p_1, p_2, \ldots, p_m\}$ es un conjunto finito de lugares
$T = \{t_1, t_2, \ldots, t_m\}$ es un conjunto finito de transiciones
$F \subseteq (P \times T) \cup (T \times P)$ es un conjunto de arcos
$W : F \longrightarrow \{1, 2, 3, \cdots\}$ es una función de peso
$M_0 : P \longrightarrow \{0, 1, 2, 3, \cdots\}$ es la marca inicial
$P \cap T = \varnothing$ y $P \cup T \neq \varnothing$
Una estructura de Red de Petri $N = (P, T, F, W)$ sin marca inicial se denota por N
Una Red de Petri con una marca inicial dada se denota por (N, M_0)

Tabla 4.1 Definición formal de una PN

Los métodos de análisis para las PN se pueden clasificar en tres grupos: 1) el árbol de cobertura; 2) la matriz de incidencia y ecuación de estado; y 3) técnicas de reducción o descomposición. El primer método involucra, esencialmente, la enumeración de todas las marcas alcanzables [22]. Debería ser aplicable a todos los tipos de redes, pero está limitada a redes “pequeñas”. Las técnicas de reducción son poderosas pero en algunos casos son aplicables sólo a subclases especiales de PN o situaciones especiales. Las ecuaciones de la matriz de incidencia gobiernan el comportamiento dinámico de sistemas concurrentes modelados por PN.

Sin embargo, esas ecuaciones están un poco limitadas debido a la naturaleza no determinista inherente en los modelos de PN.

Para una PN con *n* transiciones y *m* lugares, la *matriz de incidencia* A =$[a_{ij}]$ es una matriz de $n \times m$ enteros y sus entradas están dadas por:

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-}$$

donde $a_{ij}^{+} = w\,(i,j)$ es el peso del arco de una transición i a su lugar de salida j y $a_{ij}^{-} = w\,(j,i)$ es el peso del arco de la transición i a su lugar de entrada j. Las entradas de la matriz de incidencia a_{ij}^{-}, a_{ij}^{+} y a_{ij} representan el número de tokens removidos, agregados y cambiados en el lugar *j* cuando la transición *i* se dispara una vez.

4.2. Definición de la CCPN

En esta sección mostramos la definición formal de la CCPN así como de su regla de disparo de transiciones.

4.2.1. Estructura de la CCPN

Para dar una definición formal de la CCPN es necesario definir algunos conceptos.

Definición 4.1 *Tipo de dato. Un tipo de dato se compone de uno o más elementos y al conjunto de todos esos elementos se le conoce con el mismo nombre que el del tipo de dato. El tipo de dato para una variable* **v** *se expresa por*

$$Type(v)$$

El tipo de datos de una expresión ***expr*** *se expresa por*

$$Type(expr)$$

La notación Type(v) es extendida a

$$Type(A) = \{Type(v) \mid v \in A\}$$

donde A es un conjunto de variables

Otros conceptos que son necesarios son el de conjunto de lugares de entrada y conjunto de lugares de salida. Enseguida presentamos ambos.

Definición 4.2 *El conjunto de los lugares de entrada $p \in P$ de una transición $t \in T$ se representa por ${}^{\cdot}t$*

$${}^{\cdot}t = \{p \mid p \in P, t \in T, p \text{ es un lugar de entrada de } t\}$$

Definición 4.3 *El conjunto de los lugares de salida $p \in P$ de una transición $t \in T$ se representa por $t^{\cdot}$*

$$t^{\cdot} = \{p \mid p \in P, t \in T, p \text{ es un lugar de salida de } t\}$$

Una vez definidos los conceptos anteriores es posible dar la definición de la CCPN.

Definición 4.4 *Una Red de Petri Coloreada Condicional con Tiempo (CCPN) es una 12-tupla*

$$CCPN = \{\Sigma, P, T, A, W, N, C, Con, Acción, D, \tau, I\}$$

donde

(i) Σ es un conjunto finito de tipos, llamado conjunto de colores. Un elemento del conjunto puede ser información que toma un evento o una acción.

(ii) P es un conjunto finito de lugares. Un lugar puede ser uno de los siguientes cuatro tipos:

1. *Primitivo,* el cual representa eventos primitivos;
2. *Compuesto,* el cual representa eventos compuestos;
3. C*opy,* el cual se usa cuando un mismo evento dispara dos o más reglas. Un evento puede participar en dos o más reglas, sin embargo, en las PN, se necesita que un token se duplique para poderse compartir. Un lugar de tipo copy toma la misma información que su lugar original.
4. *Virtual,* los cuales se usan para acumular diferentes eventos que disparan la misma regla. Por ejemplo, cuando el evento de una regla es un evento compuesto OR.

(iii) T es un conjunto finito de transiciones. Una transición puede tener uno de los siguientes tres tipos:

1. *Rule,* las cuales representan una regla;

2. *Compuesta*, las cuales representan la generación de eventos compuestos;

3. *Copy*, las cuales duplican un evento para cada regla disparada.

(iv) A es un conjunto finito de arcos entre lugares y transiciones, tal que $P \cap T = P \cap A = T \cap A = \Phi$.

(v) W es una función que asigna a cada arco un peso. Si el peso no se especifica, el valor es 1.

(vi) N es una función *nodo*. Está definida de A a $P \times T \cup T \times P$.

(vii) C es una función *color*. Está definida de P a Σ.

(viii) Con es una función *condición*. Está definida de T_{rule} o T_{comp} a una expresión tal que :

$$\begin{aligned} \forall t \quad &\in \quad T_{rule} : [Type(Con(t)) = \mathcal{B}], \\ &\quad \text{donde la función } Con \text{ evalúa la condición de la regla.} \\ \forall t \quad &\in \quad T_{comp} : [Type(Con(t)) = \mathcal{B}], \\ &\quad \text{donde la función } Con \text{ evalúa el intervalo de tiempo} \\ &\quad \text{de } t \text{ contra la estampa de tiempo del token.} \end{aligned}$$

(ix) $Acción$ es una función *acción*. Está definida de T_{rule} a expresiones tales que:

$$\forall t \in T_{rule},\ p \in t^{\cdot} : [Type(Action(t)) = C(p)_{MS}]$$

(x) D es una función *intervalo de tiempo*. Está definida de T_{comp} a un intervalo de tiempo $[d1(t), d2(t)]$, donde $t \in T_{comp}$, y $d1(t)$, $d2(t)$ son el tiempo inicial y final del intervalo, respectivamente.

(xi) τ es una función *estampa de tiempo*. Está definida de $M(p)$ a $\mathbb{R}^+$, la cual asigna a cada token en un lugar p una estampa de tiempo correspondiente a la hora natural, con la forma $año : mes : día - hora - minuto- : segundo$. Por ejemplo, $2003 : 11 : 10 - 11 : 16 : 46$.

(xii) I es una función de *inicialización*. Está definida de P en una expresión tal que:

$$\forall p \in P : [Type(I(p)) = (C(p)_{MS}, \tau(C(p)_{MS})]$$

Gráficamente, una CCPN para una sola regla se muestra en la figura 4.1(a), donde el lugar de entrada de la transición representa al evento de la regla ECA, la transición almacena la parte condicional y el lugar de salida de la transición representa la acción de la regla. La figura 4.1(b) muestra una CCPN de dos reglas cuyo evento activador es el mismo para ambas. La transición $t \in T_{copy}$ duplica el token del evento colocándolo

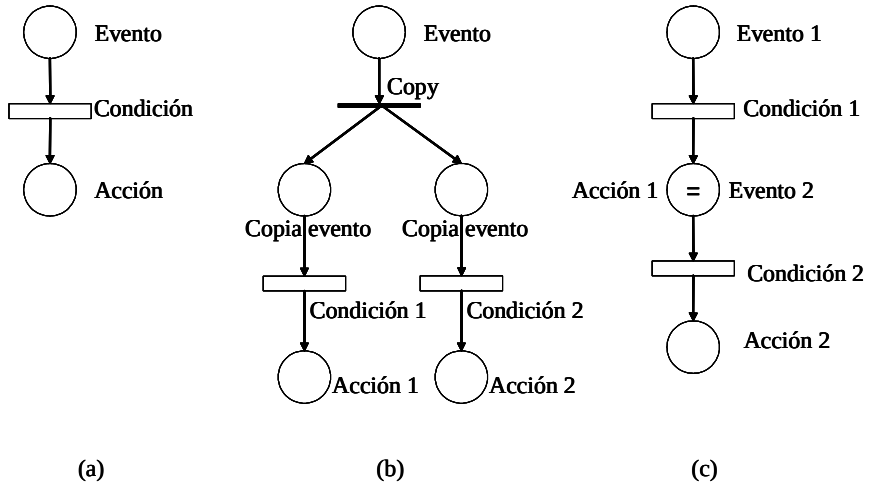

Figura 4.1: Modalidades que pueden presentarse en la creación de una CCPN

en lugares de salida que a la vez son los lugares de entrada de transiciones que guardan la condición de las reglas. La figura 4.1(c) muestra una CCPN de dos reglas, donde la acción ejecutada por una de ellas es el evento activador de la otra.

Además de la representación gráfica, la CCPN cuenta con una representación matemática dada por la matriz de incidencia. La definición de la matriz de incidencia para la CCPN es la siguiente:

Definición 4.5 *Matriz de incidencia de una CCPN. Para una CCPN N, con n transiciones y m lugares, la matriz de incidencia A = [a_{ij}] es una matriz de números enteros de n x m. El valor para cada elemento de la matriz está dado por:*

1. $-w\,(j,i)$ *Cuando el lugar* $p_j \in P$ *es un lugar de entrada a la transición* $t_i \in T$. $w\,(j,i)$ *es el peso del arco que conecta a la transición* t_i *con su lugar de entrada* $p_j \in P$.
2. 0 *Cuando no existe un arco que conecta al lugar* $p_j \in P$ *con la transición* $t_i \in T$ *y viceversa.*
3. $w\,(i,j)$ *Cuando el lugar* $p_j \in P$ *es un lugar de salida a la transición* $t_i \in T$. $w\,(i,j)$ *es el peso del arco que conecta a la transición* t_i *con su lugar de salida* $p_j \in P$.

Ahora vamos a presentar la regla de disparo de transiciones.

4.2.2. Regla de disparo de transiciones

Un elemento fundamental en el comportamiento de la CCPN es la transición. Para describir las condiciones bajo las cuales una transición se dispara y cómo se refleja en el cambio de estado de la CCPN es necesario definir previamente el concepto de token.

Uno de los elementos más importantes que rigen el comportamiento de la PN son los tokens. En la CCPN, los tokens tienen la capacidad de almacenar información de un cierto tipo de dato. Su definición es la siguiente:

Definición 4.6 *Un elemento token consta de tres elementos (p, c, t) donde p* $\in P$*, c* $\in C(p)$ *y t es la estampa de tiempo en la que el token fue generado. El conjunto de todos los elementos token está denotado por TE. Una marca M es un multi-conjunto sobre TE. La marca inicial* M_0 *es obtenida al evaluar las expresiones de inicialización:*

$$\forall (p, c, t) \in TE : M_0 (p, c, t) = I (p)$$

El conjunto de todas las marcas es expresado por M.

Una vez definido el concepto de token podemos mostrar la regla de disparo de transiciones en la CCPN.

Una transición $t \in T_{copy}$ se activa en una marca M si y sólo si:

$$\forall p \in {}^{\cdot}t : M (p) > 0$$

Una transición $t \in T_{rule}$ se activa en una marca M si y sólo si se satisface:

$$\forall p \in {}^{\cdot}t : M (p) > 0 \; y \; Type (Cond (t)) = verdadero$$

Si $t \in T$ está activada, se necesita una función $C_{enabled}$ para establecer la activación, así como los tokens que la originaron. La función de activación $C_{enabled}$ asocia cada par ordenado $(p, t) \in \{P \times T\}$ con el multi-conjunto de un tipo de dato $C(p)$. Es decir,

$$\forall t \in T, p \in {}^{\cdot}t : [Type (C_{enabled} (p, t)) = C (p)_{MS}]$$

De forma resumida podemos decir que cuando una transición *t* está activada en una marca M_1, puede cambiar de M_1 a M_2, respetando:

$$
\begin{aligned}
&(i)\,\text{Si}\, t \in T_{copy}, \forall p_1,\ p_2 \in P,\ p_1 \in\ {}^{\cdot}t,\ p_2 \in t^{\cdot}: \\
&\quad M_2\left(p_1\right) = M_1\left(p_1\right) - C_{enabled}\left(p_1, t\right) \\
&\quad M_2\left(p_2\right) = M_1\left(p_2\right) + C_{enabled}\left(p_1, t\right) \\
&(ii)\,\text{Si}\, t \in T_{rule}, \forall p \in P: \\
&\quad M_2\left(p_1\right) = M_1\left(p_1\right) - C_{enabled}\left(p_1, t\right) \\
&\quad M_2\left(p_2\right) = M_1\left(p_2\right) + Action\left(t, p\right)
\end{aligned}
$$

4.3. Modelación de reglas ECA con CCPN

En esta sección presentamos el algoritmo de conversión de reglas ECA a una estructura de CCPN.

Algoritmo de conversión de reglas ECA a CCPN

Las reglas ECA se modelan en la CCPN considerando los siguientes pasos:

1. Los eventos y acciones se modelan como lugares.
2. Las reglas ECA en sí mismas se mapean en transiciones.
3. El disparo de la regla corresponde al disparo de la transición.
4. La detección de eventos se modela como tokens iniciales.
5. Las condiciones se modelan como condiciones agregadas a las transiciones.

Los lugares en la CCPN se dividen cuatro tipos: lugares primitivos, lugares compuestos, lugares virtuales y lugares copy. Los lugares copy se representan en la CCPN por medio de un doble círculo, con el círculo interno punteado. La información que almacena este tipo de lugar es la misma que la del evento del cual se originó la copia. Los lugares virtuales se representan con un círculo punteado y se emplean como almacenes de tokens. Los lugares primitivos son un conjunto de instrucciones de BD y cualquiera de esas instrucciones puede ocasionar la activación de la regla. Se dibujan con un círculo (con línea continua). Los lugares compuestos se utilizan para formar eventos compuestos, en la CCPN se dibujan como dos círculos concéntricos.

Las transiciones de la CCPN se clasifican en tres tipos: transiciones T_{rule}, transiciones T_{copy} y transiciones T_{comp}. En la CCPN, cada transición de tipo T_{rule} se identifica con un rectángulo. Cada una de

estas transiciones almacena la condición de una regla ECA. Las transiciones T_{copy} se identifican por una barra. Sirven para hacer duplicados de un evento que participa en dos o más reglas. Si este duplicado de lugares no se realizara, lo que se obtiene es que sólo una de las dos reglas se activa cuando, de hecho, ambas deben activarse. Para este tipo de transiciones, su condición siempre es verdadera. Las transiciones T_{comp} se identifican por una doble barra. Se emplean para esperar por la ocurrencia de eventos compuestos.

El algoritmo que mapea reglas ECA en CCPN se muestra en la figura 4.2.

La figura 4.3 muestra el significado de cada elemento en la CCPN.

Utilizando este algorimo vamos a presentar un ejemplo de conversión de reglas ECA a CCPN.

Ejemplo de conversión de reglas ECA a CCPN

Vamos a ver un ejemplo de reglas ECA en un sistema de BDA.

La BD usada en [40] se basa en las siguientes tablas:

```
EMP(emp_id, nombre, rango, salario)
BONO(emp_id, cantidad)
VENTAS(emp_id, mes, número)
```

La descripción de cada regla es la siguiente:

Regla 1: Cuando el bono de un empleado se incrementa en más de 100, entonces el rango del empleado se incrementa en 1

Regla 2: Cuando se actualiza el rango del empleado, entonces el bono del empleado se incrementa 10 veces el nuevo rango

Regla 3: Cuando las ventas del empleado son mayores a 50 y su rango es menor de 15, su bono se decrementa en 100

Regla 4: Cuando el rango del empleado llega a 15, incrementar el salario del empleado un 10 %

Para ver claramente las relaciones de las reglas, analizamos el evento, la condición y la acción de cada una y las reescribimos en la forma ON - IF - THEN.

Regla 1:

ON `update of cantidad on BONO`

IF `update.BONO_cantidad-BONO_cantidad` $>$ `100`

THEN `update EMP set rango = update.rango + 1`

`where emp_id = update.emp_id`;

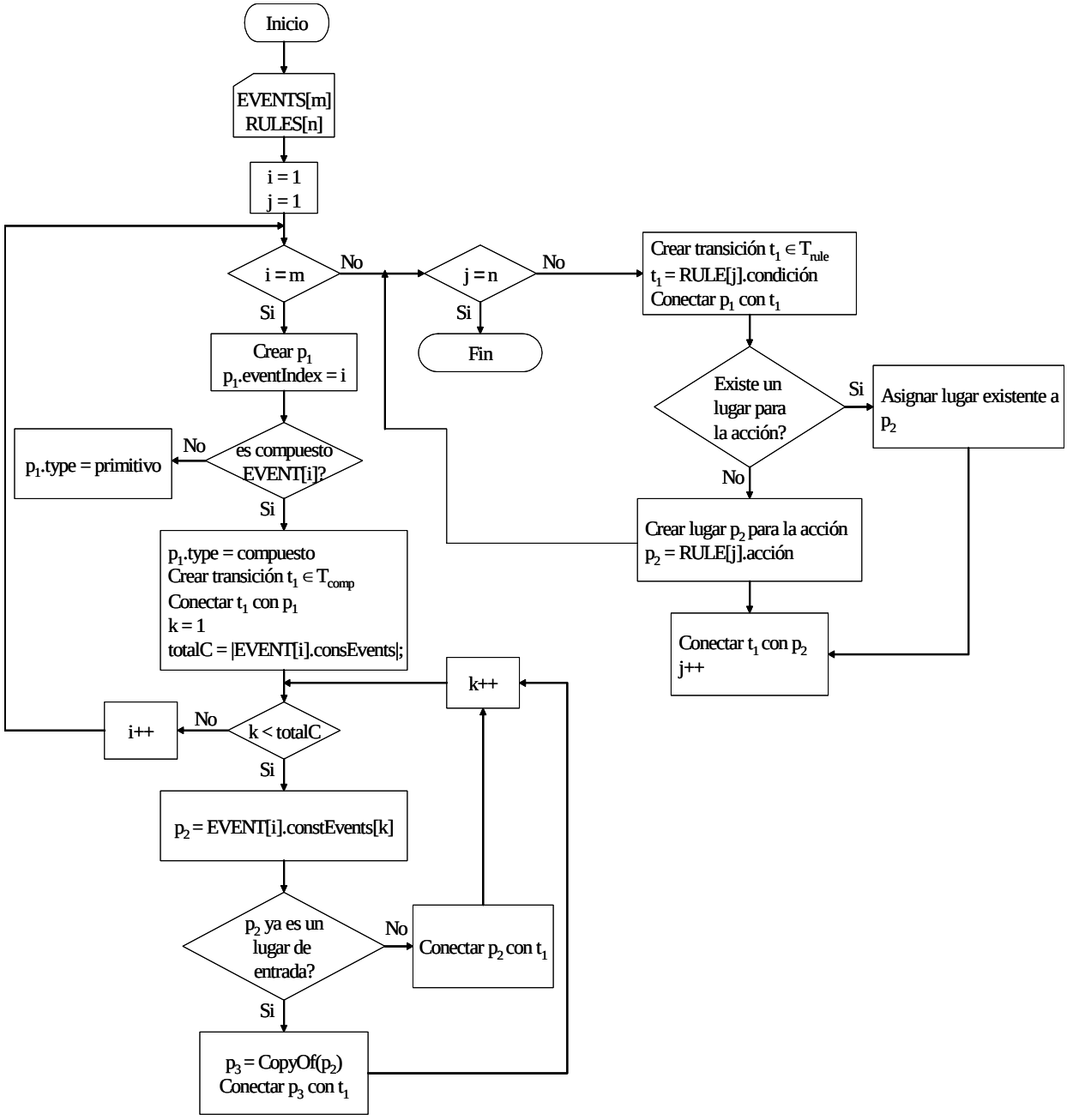

Figura 4.2: Algoritmo de conversión de reglas ECA a CCPN

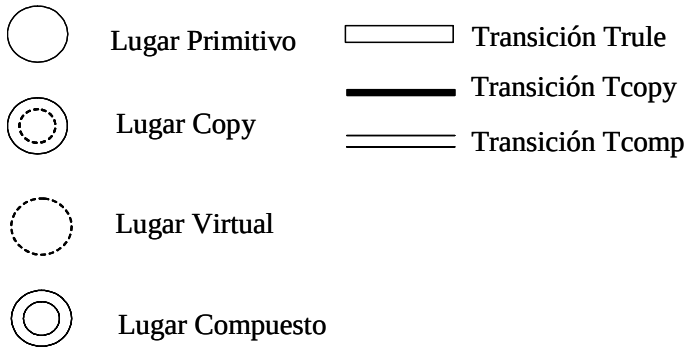

Figura 4.3: Descripción de cada elemento de la CCPN

Regla 2:

ON `update of rango on EMP`

IF `EMP_rango` $>$ `5`

THEN `update BONO set mes = old.cantidad + rango * 10`

`where emp_id = update.emp_id;`

Regla 3:

ON `insert on VENTAS`

IF `VENTAS_número` $>$ `50`

THEN `update EMP set rango = old.rango+1 where emp_id = insert.emp_id;`

Regla 4:

ON `update of rango on EMP`

IF `rango = 15`

THEN `update EMP set salario = old.salario * 1.1`

`where emp_id = insert.emp_id;`

La figura 4.4 muestra la CCPN que representa al conjunto de reglas anterior. Para obtenerla se aplicó el algoritmo descrito en la figura 4.2.

En la figura 4.4, hay cuatro transiciones *T10*, *T11*, *T12* y *T14* las cuales corresponden a las cuatro reglas descritas. También existe una transición T_{copy} para el evento que activa tanto a la Regla 2 como a la Regla 4.

En la siguiente sección se muestra la implementación del algoritmo.

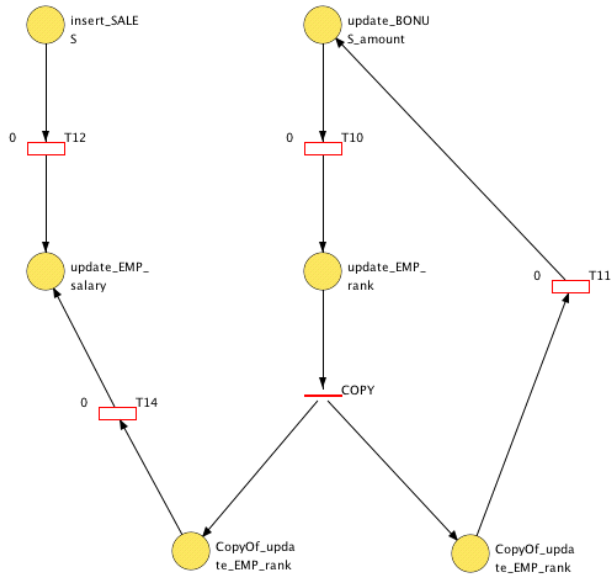

Figura 4.4: CCPN de la base de reglas anterior generada con ECAPNSim

4.4. Simulación de Reglas ECA con ECAPNSim

La CCPN se propuso especialmente para la especificación y simulación de reglas ECA. Para mostrar su uso, se desarrolló una aplicación de software denominada ECAPNSim (ECA Petri Nets SIMulator) sobre un sistema de BDA. ECAPNSim se implementó en Java y puede trabajar con Postgress a través del driver JDBC.

La forma en la que funciona ECAPNSim es la siguiente: primero se detecta el evento, luego se ejecuta el simulador. Posteriormente, si alguna(s) regla(s) se disparan, ECAPNSim envía la acción de la(s) regla(s) disparadas a la BD. Finalmente, la BD realiza la operación indicada por ECAPNSim.

La arquitectura de ECAPNSim se muestra a continuación.

ECAPNSim consta de dos bloques principales: el kernel y las herramientas del ambiente. Ambos bloques se ilustran en la figura 4.5

El kernel de ECAPNSim provee la funcionalidad activa a una BD pasiva. Consiste del Administrador de CCPN, la base de CCPN, el detector de eventos compuestos y el componente de ejecución de reglas.

ECAPNSim tiene un conjunto de herramientas usadas por el desarrollador de reglas ECA. Las herra-

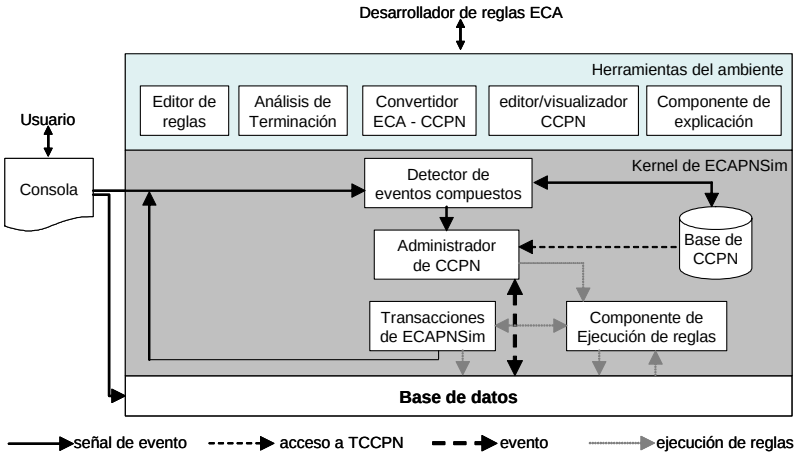

Figura 4.5: Arquitectura de ECAPNSim

mientas de ECAPNSim están compuestas por: el editor de reglas, el módulo de análisis de terminación, el convertidor de reglas ECA a CCPN, el editor/visualizador de CCPN y el componente de explicación.

Esta herramienta permite modelar a las reglas ECA como una Red de Petri pero le otorga más información acerca de la regla.

4.5. Comentarios Finales

Las PNs son una herramienta poderosa para modelar y simular sistemas manejados por eventos. Cuentan con una representación gráfica y una representación matemática, las cuales permiten verificar la consistencia de las observaciones que se hagan sobre el sistema que se modela. Los componentes básicos del modelo gráfico son lugares (que se dibujan como círculos) y transiciones (que se dibujan como rectángulos pequeños). Los elementos de la PN se conectan mediante arcos dirigidos de lugares a transiciones y viceversa. El modelo gráfico es útil como los diagramas de flujo y los diagramas de bloque. Sin embargo, para sistemas muy grandes, el modelo gráfico tiene algunos inconvenientes. En estos casos el modelo matemático, representado por la matriz de incidencia, es más útil para obtener información.

De forma general, las PNs se pueden dividir en tres categorías: PNs ordinarias, PNs abreviadas y PNs extendidas. Las PNs ordinarias son aquellas en las que el peso de los arcos es 1 y no involucran tiempo.

Las PNs abreviadas son aquellas que utilizan una representación simplificada del modelo gráfico. Las PNs extendidas son aquellas a las que se les han agregado reglas de funcionamiento para enriquecer el modelo inicial lo cual hace posible que un mayor número de aplicaciones puedan ser modeladas. La Red de Petri Coloreada Condicional (CCPN) es un ejemplo de PNs extendidas.

La CCPN fue desarrollada específicamente para BDA. A las transiciones normales de PN se les añadió la capacidad de evaluar expresiones lógicas y a los lugares la capacidad de almacenar información acerca del evento con el que están relacionados. Existe una gran variedad de sistemas, tanto relacionales como orientados a objetos, que incorporan el comportamiento activo. Sin embargo, los sistemas desarrollados no son compatibles entre sí porque cada sistema tiene su manera particular para implementar las reglas ECA. La forma general de una regla ECA es:

ON evento

IF condición

THEN acción

Esta forma general es la que toma la CCPN. Así como las PN normales, la CCPN cuenta con su modelo gráfico y su modelo matemático. El modelo matemático es la matriz de incidencia.

Para convertir un conjunto de reglas ECA en una estructura de CCPN se desarrolló el sistema de software denominado ECAPNSim.

Capítulo 5

Definición y Validación de los Parámetros de Medición

A través de los capítulos anteriores ha sido posible establecer un panorama acerca del estado de desarrollo de las BDAs. Se ha visto que se pueden emplear en una gran cantidad de aplicaciones no tradicionales, por ejemplo, control de procesos, control de tráfico aéreo, automatización de procesos en una compañía, etc. Estas aplicaciones requieren que la BD pueda reaccionar ante eventos específicos y verificar condiciones definidas sobre el estado de la BD para poder ejecutar acciones. Las reglas ECA son el mecanismo que las BDAs utilizan para proporcionar el comportamiento reactivo.

Cuando ocurre un evento se pueden activar una o varias reglas. Una vez activada la regla, se comprueba la condición de la misma. Si se satisface la condición, se ejecuta la acción de la regla.

Las ventajas de las BDAs son diversas, entre ellas se pueden distinguir las siguientes:

1. Promueven la reusabilidad de código, ya que en lugar de replicar el código en distintas aplicaciones, el código reside en un solo lugar.

2. Incrementan la consistencia de las aplicaciones porque, al centralizar el comportamiento reactivo, no se pueden ignorar ciertos procedimientos que todas las aplicaciones tienen en común y que se deben verificar.

3. Reducen el tráfico en la red en un ambiente cliente/servidor, ya que la acción asociada a la aparición de un evento se ejecuta localmente.

Sin embargo, las bondades de las BDAs se ven reducidas por el reto que representa el mantenimiento de las reglas ECA. La dependencia implícita entre ellas y la complejidad de su interacción hacen difícil determinar las causas que provocaron el disparo de una regla. A diferencia de los lenguajes de programación tradicionales, donde el control secuencial se especifica explícita y estáticamente, las reglas ECA se disparan dinámicamente dependiendo del flujo de eventos previo. Aún con un conjunto pequeño de reglas ECA, si éstas tienen una interacción significativa entre sí, el sistema puede llegar a ser inmanejable. La dificultad del mantenimiento de las reglas ECA es el principal obstáculo para que las BDAs lleguen a ser ampliamente usadas.

Para solventar el problema del mantenimiento de las reglas ECA se deben identificar y medir aquellos aspectos que la afectan. De esta forma, se han identificado los siguientes tres aspectos: el entendimiento, la modificabilidad y la verificabilidad de las reglas ECA. Por lo tanto, la complejidad de la interacción de las reglas ECA debe ser medida con métricas objetivas, confiables y válidas.

El trabajo presentado en [10] es el único relacionado directamente con la medición de la complejidad de la interacción de las reglas ECA en una BDA. Los autores proponen tres parámetros para realizar sus mediciones, tales parámetros son: el *número de anclas*, la *distancia* y el *potencial de disparo*. Sin embargo, utilizan una gráfica de disparo para modelar las reglas ECA y esta representación no permite el manejo de eventos compuestos.

En el trabajo que se presenta en esta tesis se adoptaron las métricas presentadas en [10] y se modelaron sobre una estructura de CCPN. La CCPN permite manejar eventos compuestos, por lo tanto, se realizaron modificaciones a los parámetros de medición propuestos en [10] para adecuarlas al manejo de tales eventos. Una vez modelados los parámetros de medición fue necesario verificar su validez teórica por medio del cumplimiento de los axiomas que forman el marco de trabajo del Dr. H. Zuse.

Para describir adecuadamente el proceso de modelación, definición, y validación de los parámetros de medición sobre la CCPN, este capítulo se estructura de la siguiente forma: en la sección 5.1 se presenta la modelación de los parámetros de medición sobre la estructura gráfica de la CCPN; la sección 5.2 muestra la modelación de los parámetros sobre la estructura matemática de la CCPN; en la sección 5.3 se presenta la validación de los parámetros de medición y la conclusión a la que se llegó después de realizada. Finalmente, en la sección 5.4 se presentan los comentarios finales.

5.1. Conceptos Preliminares

Para hacer un análisis de la complejidad de la interacción de los disparos en una BDA es necesario contar con un modelo que represente las reglas ECA de forma detallada. La CCPN proporciona esta representación, por lo que se eligió para modelar sobre ella los parámetros de medición propuestos en [10].

La figura 5.1 muestra un conjunto de reglas y su representación con la CCPN. La CCPN se obtuvo mediante la interfaz ECAPNSim. Esta interfaz recibió como entrada un archivo de texto que contenía la descripción del conjunto de reglas ECA y, automáticamente, generó la estructura de la CCPN que se muestra en la figura 5.1. El evento, la condición y la acción de una regla ECA se almacenan, en la CCPN, en un lugar de entrada, una transición tipo T_{rule} y un lugar de salida, respectivamente. Por lo tanto, se pueden intercambiar los términos evento y lugar de entrada, regla y transición T_{rule} y acción y lugar de salida.

En el capítulo III se dio una descripción del modelo de la CCPN así como de la interfaz ECAPNSim.

Antes de exponer la modelación de los parámetros de medición sobre la CCPN es necesario dar los conceptos de:

1. Lugar inicial.
2. Lugar final.
3. Ruta.
4. Ruta de Transiciones.
5. Longitud de una ruta de transiciones.
6. Causas del lugar de entrada de una transición.
7. Cardinalidad del lugar de entrada de una transición.

La figura 5.1 se utilizará para ilustrar algunos ejemplos.

1. Lugar Inicial (Ancla).

Dada una CCPN, un lugar es inicial si:

$$\forall p \in P, \dot{} \; p = \varnothing$$

Regla 0
on delete to Dep
if true
then delete to Emp
where Emp.SuDep = Dep.ElDep
Regla 1
on delete to Trabaja
if true
then delete to Proyecto
where Proyecto.ElProy = Trabaja.ElProy
Regla 2
on delete to Emp
if true
then delete to Trabaja
where Emp.ElEmp = Trabaja.ElEmp
Regla 3
on(or(insert Emp), (update Emp.Salario))
if (Emp.Salario > 15,000)
then delete to Emp
where Emp.ElEmp = new.Emp.ElEmp
Regla 4
on(and(update Dep.prod), (insert Emp))
if(Dep.Prod > 90 & Emp.SuDep = Dep.ElDep)
then update Emp.Bono = 100
where Emp.ElEmp = new.Emp.ElEmp
Regla 5
on(or(update Dep.Presupuesto), (insert Dep))
if true
then update Proyecto.Presupuesto = 0.1 * Proyecto.Presupuesto
where Proyecto.ElProy = Trabaja.ElProy
and Trabaja.ElEmp = Emp.ElEmp
and Emp.SuDep = Dep.ElDep
Regla 6
on(update Proyecto.Presupuesto)
if true
then update Emp.Salario = 0.01 * Proyecto.Presupuesto
where Proyecto.ElProy = Trabaja.ElProy
and Trabaja.ElEmp = Emp.ElEmp

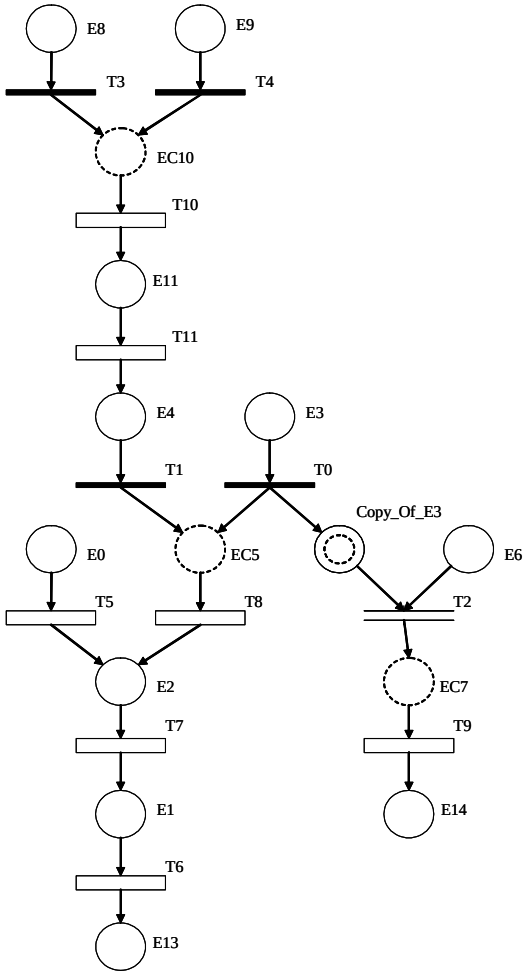

Figura 5.1: Conjunto de reglas ECA y su representación con CCPN

En otras palabras, un lugar es inicial, si sólo tiene arcos de salida y ningún arco de entrada. En la figura 5.1 se puede observar que los lugares iniciales son los lugares *E0, E3, E6, E8* y *E9*.

2. Lugar Final.

Dada una CCPN, un lugar es final si:

$$\forall p \in P, p^{\cdot} = \varnothing$$

Un lugar es final, si sólo tiene arcos de entrada y ningún arco de salida. En la figura 5.1 se observa que los lugares finales son *E13* y *E14*.

3. Ruta.

Dados un lugar inicial p_i y una transición $t_j \in T_{rule}$ en la CCPN, una ruta R, de p_i a t_j es una secuencia finita de pares, los cuales relacionan lugares y transiciones y viceversa mediante arcos de entrada y salida, respectivamente. Así, una ruta $R\left(p_i, t_j\right)$ se representa por:

$$R\left(p_i, t_j\right) = \left(p_i, t_k\right) \rightarrow \left(t_k, p_n\right) \rightarrow ... \rightarrow \left(t_m, p_s\right) \rightarrow \left(p_s, t_j\right)$$

donde:

p_i es un lugar inicial

$t_j \in T_{rule}$

$p_r \in P, \quad r = n, ...s$

$t_d \in T, \quad d = k, ...m$

$\left(p_r, t_d\right)$ es un arco de entrada

$\left(t_d, p_r\right)$ es un arco de salida

En esta definición, una ruta se finaliza cuando se alcanza una transición $t_j \in T_{rule}$ porque en la CCPN cada transición T_{rule} almacena la información de su lugar de entrada y su lugar de salida. Por lo tanto, no es necesario llegar hasta el lugar de salida de una T_{rule} para establecer una ruta.

Por ejemplo, una ruta del lugar inicial *E0* a la transición *T6* $\left(T6 \in T_{rule}\right)$ es:

$$R\left(E0, T6\right) = \left(E0, T6\right) \rightarrow \left(T5, E2\right) \rightarrow \left(E2, T7\right) \rightarrow \left(T7, E1\right) \rightarrow \left(E1, T6\right)$$

En la ruta anterior se puede identificar que *E0* es el lugar inicial; $T6 \in T_{rule}$ es la transición que se quiere alcanzar; *E0, E2,* y *E1* $\in P$; *T5, T7* y *T6* $\in T$; los pares $\left(E0, T5\right), \left(E2, T7\right)$ y $\left(E1, T6\right)$ se relacionan por arcos de entrada y los pares $\left(T5, E2\right), \left(T7, E1\right)$ se relacionan por arcos de salida.

Es posible que a partir de diferentes lugares iniciales se alcance una misma transición, es decir, puede existir más de una ruta para una transición $t_j \in T_{rule}$. Todas las rutas que, a partir de diferentes lugares iniciales, llegan a la transición t_j, forman el *conjunto de rutas* para la transición t_j. Por ejemplo, el conjunto de rutas para la transición *T11* de la figura 5.1 es:

$$\begin{aligned} CR\left(T11\right) = \quad & \{(E8,T3) \rightarrow (T3,EC10) \rightarrow (EC10,T10) \rightarrow (T10,E11) \rightarrow (E11,T11)\,, \\ & (E9,T4) \rightarrow (T4,EC10) \rightarrow (EC10,T10) \rightarrow (T10,E11) \rightarrow (E11,T11)\} \end{aligned}$$

4. Ruta de Transiciones.

Una ruta de transiciones para una transición $t_j \in T_{rule}$, $RT\left(t_j\right)$, se forma por todas las transiciones $t \in \{T_{rule} \cup T_{comp}\}$, de una ruta $R\left(p_i, t_j\right)$. De esta forma, una ruta de transiciones para la transición t_j, se representa de la siguiente manera:

$$RT\left(t_j\right) = t_d \rightarrow t_g \rightarrow ... \rightarrow t_j$$

donde:

$t_h \in \{T_{rule} \cup T_{comp}\} \quad h = d, ..., j$

$t_e \neq t_i \quad \forall e \neq i$

Por ejemplo, la ruta de transiciones para la ruta $R\left(E8, T11\right)$ es: $RT\left(T11\right) = T10 \rightarrow T11$. Si se observa, en la ruta de transiciones además de no incluirse los lugares, tampoco se incluyen las transiciones T_{copy}, ya que únicamente se utilizan para hacer el duplicado de tokens.

5. Longitud de una ruta de transiciones.

La longitud de una ruta de transiciones de una transición $t_j \in T_{rule}$, $Long\left(t_j\right)$, se define como:

$$\begin{aligned} & \forall t \in RT\left(t_j\right) \\ & Long\left(RT\left(tj\right)\right) = \sum_{\forall p_k \in \cdot t} w\left(p_k, t\right) \end{aligned}$$

Esto es, la suma de los pesos de los arcos de entrada a cada una de las transiciones en la ruta de transiciones.

Por medio de algunos ejemplos se puede clarificar la definición de longitud.

Consideremos la ruta $R\left(E3, T6\right) = (E3,T0) \rightarrow (T0,EC5) \rightarrow (EC5,T8) \rightarrow (T8,E2) \rightarrow (E2,T7) \rightarrow (T7,E1) \rightarrow (E1,T6)$. La ruta de transiciones es $R\left(T6\right) = T8 \rightarrow T7 \rightarrow T6$. La distancia de esta ruta se obtiene mediante:

$$Long\left(RT\left(T11\right)\right) = w\left(EC5,T8\right) + w\left(E2,T7\right) + w\left(E1,T6\right) = 1 + 1 + 1 = 3$$

En el ejemplo, todas las transiciones de la ruta de transiciones son del tipo T_{rule}. Este es el caso más sencillo que se puede tener, ya que cada transición tiene un único arco de entrada y sólo se tiene que verificar su peso. Sin embargo, puede darse el caso en el que a lo largo de la ruta de transiciones también se encuentren involucradas transiciones compuestas (hay que recordar que una transición compuesta se utiliza para formar eventos compuestos). En este caso se deben considerar todos los eventos que forman el evento compuesto. La razón es que estos eventos también son causas que se requieren para conectar un lugar inicial con el disparo de una regla. El siguiente ejemplo muestra una situación similar a la descrita.

Consideremos la ruta $R\left(E6,T9\right)=\left(E6,T2\right)\rightarrow\left(T2,EC7\right)\rightarrow\left(EC7,T9\right)$. Para esta ruta, su ruta de transiciones es $R\left(T9\right)=T2\rightarrow T9$. Para el cálculo de la longitud se tiene:

$$Long\left(RT\left(T9\right)\right)=w\left(E6,T2\right)+w\left(Copy\_Of\_E3,T2\right)+w\left(E7,T9\right)=1+1+1=3$$

Se puede observar que aparecen dos términos relacionados con la transición *T2*. Esto se debe a que la transición *T2* es compuesta y cada una de sus entradas debe ser considerada.

6. Causas del lugar de entrada de una transición.

El lugar de entrada (evento) de una transición $t_j\in T_{rule}$ puede ocurrir como consecuencia de la interacción del usuario/aplicación con la BD, o bien, como resultado del disparo de otras transiciones (reglas). De esta forma, las causas del lugar de entrada de una transición, $causE\left(t_j\right)$, son todas aquellas circunstancias que pueden generar la ocurrencia del lugar de entrada, es decir, la activación de la transición. Dado que en la CCPN se cuenta con diferentes tipos de lugares, es necesario tener un algoritmo para calcular las causas del lugar de entrada de una transición. El siguiente algoritmo realiza esta función:

$\forall p\in{}^{\cdot}t_j$

Caso I: p es primitivo

Si ${}^{\cdot}p=\varnothing$

causa = causa + 1

De otra forma,

causa = causa + $\sum_{t\in{}^{\cdot}p}w\left(t,p\right)+1$

Caso II: p es compuesto

causa = causa +1

Caso III: p es virtual

causa = causa + $\sum_{t\in{}^{\cdot}p}w\left(t,p\right)$

En el caso en el que el lugar p sea de tipo copy basta verificar cuál es su lugar origen para saber el tipo de lugar y aplicar al algoritmo anterior.

En la primera parte del algoritmo se consideran aquellos eventos que son primitivos y, además, son iniciales. Este tipo de eventos pueden ocurrir, únicamente, por la interacción del usuario/aplicación con la BD. Debido a esto, su causa tiene un valor de 1. El evento de la transición *T5* exhibe tales características.

Por otro lado, existen eventos que son primitivos y no son iniciales, es decir, son la acción de alguna otra regla. Es natural suponer que este tipo de eventos tienen un mayor número de causas, dado que varias reglas pueden producir el mismo resultado. Además, al ser primitivos, también pueden generarse por la interacción del usuario/aplicación con la BD. El evento de la transición *T7 - E2 -* es un ejemplo claro. Este evento es primitivo, por lo cual, tiene una causa. Sin embargo, aún cuando no suceda por la interacción del usuario/aplicación con la BD, el disparo de las reglas *T5* ó *T8* pueden generarlo. Como consecuencia, el evento *E2* puede ser originado por tres causas.

Los eventos compuestos se consideran como una unidad porque, aún cuando son varios eventos los que intervienen, es la combinación de tales eventos lo que genera la activación de una regla. Como ejemplo, tomemos el evento *EC7* de la regla *T9*. Para generar este evento se combinan los eventos *E3* y *E6*, sin embargo, es realmente el evento *EC7* el necesario para activar la regla. Por lo tanto, la causa del evento de la regla *T9* es uno.

Finalmente, los lugares virtuales presentan una similitud con los lugares primitivos ya que ambos pueden ser generados por la interacción de otras transiciones. La diferencia radica en que los lugares virtuales no se pueden generar por la interacción del usuario/aplicación con la BD. Tomando como ejemplo el lugar *EC5* de la transición *T8* se puede observar que tal evento se puede generar cuando se disparan las transiciones *T0* ó *T1*.

7. Cardinalidad del lugar de entrada de una transición.

La cardinalidad del lugar de entrada de una transición $t_j \in T_{rule}$, $cardE\left(t_j\right)$, se define como el número de lugares primitivos que intervienen en la generación del lugar de entrada de la transición t_j. Se calcula la cardinalidad mediante el siguiente algoritmo:

$\forall p_i \in {}^{\cdot}t_j$

Caso I: Si p_i es primitivo

cardinalidad = cardinalidad + 1

Caso II: Si p_i es compuesto o p_i es virtual

$\forall t \in \cdot p_i$

$\forall p_j \in \cdot t$

Repetir desde el Caso I

Una regla se puede activar por la ocurrencia de un evento primitivo o un evento compuesto. Cuando el evento de la regla es primitivo, su cardinalidad es uno, porque su sola ocurrencia activa la regla. Cuando el evento de la regla es compuesto, se necesitan considerar los eventos involucrados en la formación del evento. Por lo tanto, la cardinalidad de un evento compuesto será el número de eventos primitivos que intervienen en la generación del evento compuesto. Por ejemplo, consideremos las transiciones *T5* y *T9* de la figura 5.1. Se puede observar que la transición *T5* se activa por la ocurrencia del lugar primitivo *E0*, por lo tanto, la cardinalidad del lugar de entrada de la regla *T0* es uno. El lugar de entrada de la transición *T9* es el lugar virtual que representa al evento compuesto *and*. Este lugar necesita que sucedan los lugares primitivos *E3* y *E6*, por lo tanto, la cardinalidad del lugar de entrada de la regla *T9* es dos.

Una vez definidos los conceptos anteriores, se pueden dar las definiciones de los parámetros de medición de la complejidad de la interacción de las reglas ECA modelados sobre la CCPN.

5.2. Definición de los parámetros de medición sobre la CCPN

En esta sección vamos a utilizar los conceptos que hemos definido para poder mostrar cada uno de los parámetros de medición.

Iniciamos con la definición del parámetro Número de Anclas.

Definición 5.1 *Número de Anclas. Dada una transición $t_j \in T_{rule}$ en la CCPN, el número de anclas de la transición t_j, $NA(t_j)$, es el número de lugares iniciales que se necesitan para abarcar todas las causas del lugar de entrada de la transición t_j.*

Por ejemplo, en la figura 5.1 la transición *T6* se activa por la ocurrencia del lugar *E1*. El lugar *E1* ocurre como consecuencia del disparo de la transición *T7*. Los lugares iniciales que, potencialmente, pueden provocar el disparo de la transición *T7* son *E0*, *E3*, *E8* y *E9*. Así, el número de anclas para la transición *T6* es cuatro, $NA(T6) = 4$.

Consideremos ahora la transición *T8*. Esta transición se activa por la ocurrencia del lugar virtual *EC5*. Este lugar se origina, a su vez, por el disparo de la transición *T0* ó *T1*, por lo tanto, tiene dos causas. Indagando en la CCPN sobre la transición *T0*, se puede observar que se activa por la ocurrencia del lugar

inicial *E3*. Analizando la transición *T1*, los lugares iniciales que pueden provocar su activación son *E8* y *E9*. Estos eventos, aunados al lugar inicial de la transición *T0* engloban todas las causas que pueden hacer que la transición *T8* se active, por lo tanto, el valor del número de anclas para la transición *T8* es $NA\left(T8\right) = 3$.

Dada una transición $t_j \in T_{rule}$ en la CCPN, el valor $NA\left(t_j\right)$ refleja el número de formas en las que es posible activar a t_j. Por lo tanto, un mayor número de anclas indica una mayor dificultad para entender una regla, ya que cada ancla representa una línea de razonamiento que se debe seguir hacia la transición.

Por ejemplo, el número de anclas para la transición *T6* es cuatro, lo cual significa que cuando se quiere entender, verificar o modificar la regla correspondiente a la transición *T6* se deben considerar las cuatro rutas por las cuales se llega a *T6*.

A continuación se muestra la definición del parámetro de Distancia.

Definición 5.2 *Distancia. Dada una transición $t_j \in T_{rule}$ en la CCPN, la distancia de la transición t_j, $D\left(t_j\right)$, se define como la longitud de la ruta de transiciones más larga que conecta a la regla t_j con un lugar inicial.*

En esta definición se dice que la ruta de transiciones conecta a la transición $t_j \in T_{rule}$ con un lugar inicial porque la primera transición de la ruta de transiciones está relacionada, mediante un arco de entrada, con un lugar inicial.

Para ilustrar la definición de distancia, tomemos la transición *T6*. Las rutas desde un lugar inicial hasta la transición *T6* se listan a continuación:

$R\left(E0, T7\right) = \left(E0, T5\right) \rightarrow \left(T5, E3\right) \rightarrow \left(E2, T7\right)$ $R\left(E3, T7\right) = \left(E3, T0\right) \rightarrow \left(T0, EC5\right) \rightarrow \left(EC5, T8\right) \rightarrow \left(T8, E2\right) \rightarrow \left(E2, T7\right)$

$R\left(E8, T7\right) = \left(E8, T3\right) \rightarrow \left(T3, EC10\right) \rightarrow \left(EC10, T10\right) \rightarrow \left(T10, E11\right) \rightarrow \left(E11, T11\right) \rightarrow$
$\left(T11, E4\right) \rightarrow \left(E4, T1\right) \rightarrow \left(T1, EC5\right) \rightarrow \left(EC5, T8\right) \rightarrow \left(T8, E2\right) \rightarrow \left(E2, T7\right)$

$R\left(E9, T7\right) = \left(E9, T4\right) \rightarrow \left(T4, EC10\right) \rightarrow \left(EC10, T10\right) \rightarrow \left(T10, E11\right) \rightarrow \left(E11, T11\right) \rightarrow$
$\left(T11, E4\right) \rightarrow \left(E4, T1\right) \rightarrow \left(T1, EC5\right) \rightarrow \left(EC5, T8\right) \rightarrow \left(T8, E2\right) \rightarrow \left(E2, T7\right)$

De este conjunto de rutas se obtuvieron las siguientes rutas de transiciones:

$RT\left(T7\right) = T5 \rightarrow T7$

$RT\left(T7\right) = T8 \rightarrow T7$

$RT\left(T7\right) = T10 \rightarrow T11 \rightarrow T8 \rightarrow T7$

Las rutas que inician en los lugares *E8* y *E9* son las mismas, por tal motivo se puede omitir una de ellas. Una vez que se tienen las rutas de transiciones es necesario calcular la longitud de cada una de ellas.

$Long\,(RT\,(T7)) = w\,(E0, T5) + w\,(E2, T7) = 1 + 1 = 2$

$Long\,(RT\,(T7)) = w\,(EC5, T8) + w\,(E2, T7) = 1 + 1 = 2$

$Long\,(RT\,(T7)) = w\,(EC10, T10) + w\,(E11, T11) + w\,(EC5, T8) + w\,(E2, T7) = 1+1+1+1 = 4$

Según los datos obtenidos, la distancia de la transición *T7* es 4, $D\,(T7) = 4$, debido a que es el máximo valor de todos los calculados.

Para mostrar otro ejemplo, tomemos la transición *T11*. Todas las rutas hacia *T11* desde un lugar inicial son las siguientes:

$R\,(E8, T11) = (E8, T3) \rightarrow (T3, EC10) \rightarrow (EC10, T10) \rightarrow (T10, E11) \rightarrow (E11, T11)$

$R\,(E9, T11) = (E9, T4) \rightarrow (T4, EC10) \rightarrow (EC10, T10) \rightarrow (T10, E11) \rightarrow (E11, T11)$

En ambas rutas, cuando se forma la ruta de transiciones se obtiene el mismo resultado, por lo tanto, existe una única ruta de transiciones que es la siguiente:

$RT\,(T11) = T10 \rightarrow T11$

Calculamos la longitud de la ruta.

$Long\,(RT\,(T11)) = w\,(EC10, T10) + w\,(E11, T11) = 1 + 1 = 2$

En este caso, la distancia es la longitud de la ruta porque existe un único valor.

Dada una transición $t_j \in T_{rule}$ en la CCPN, el valor $D\,(t_j)$ permite saber el número de inferencias necesarias, a lo largo de una ruta, para activar a la transición t_j. Un valor de distancia mayor significa una mayor cantidad de inferencias que se necesitan para que una regla se active, por lo tanto, será más difícil entender y verificar la regla.

Finalmente, definimos el parámetro del Potencial de Disparo.

Definición 5.3 *Potencial de Disparo. El potencial de disparo de una regla $t_j \in T_{rule}, TP\,(t_j)$, se define como el cociente de las causas del lugar de entrada de t_j entre la cardinalidad del lugar de entrada de t_j.*

$$TP\,(t_j) = \frac{causE\,(t_j)}{cardE\,(t_j)}$$

Para dar ejemplos del potencial de disparo de una regla, consideremos las transiciones *T0* y *T2* de la CCPN de la figura 5.2.

Para ambas transiciones calculamos el valor de las causas y la cardinalidad de su lugar de entrada. El valor de la causa del lugar de entrada de la transición *T0* es uno, porque el lugar *E3* es primitivo e inicial. Además, por ser el lugar *E3* primitivo, su cardinalidad es uno. Para el lugar de entrada de la transición *T2*, el valor de sus causas es tres, porque puede generarse por el disparo de las transiciones *T1* o *T3* y, además, por

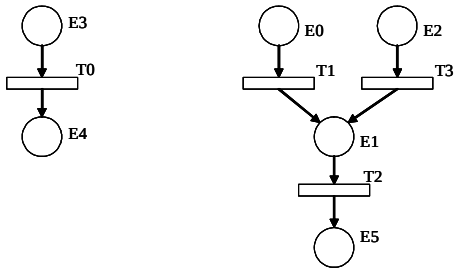

Figura 5.2: Reglas para el ejemplo del potencial de disparo

la interacción del usuario/aplicación con la BD. Dado que el lugar *E1* es primitivo, el valor de su cardinalidad es uno. Con estos valores se puede calcular el potencial de disparo para ambas reglas.

$$TP\,(T0) = \tfrac{1}{1} \qquad TP\,(T2) = \tfrac{3}{1}$$

Como se puede observar, el potencial de disparo de la transición *T2* es mayor que el potencial de disparo de la transición *T0*. Esto significa que para entender la regla representada en la CCPN por la transición *T2* se realiza un mayor análisis que para entender la regla representada por la transición *T0*. Esto se debe a que cuando ocurre el lugar *E1* ó se dispara la transición *T1* ó se dispara la transición *T3* se activa la transición *T2*. En cambio, para entender la regla que representa la transición *T0* únicamente se tiene que analizar la ocurrencia del evento *E3*.

Los lugares de entrada de las transiciones *T0* y *T2* son primitivos. Sin embargo, los eventos compuestos OR y NEGACIÓN, en cierto sentido, caen en el caso de los eventos primitivos. Por ejemplo, veamos las reglas de la figura 5.3.

Si calculamos las causas y la cardinalidad de los eventos *E3* y *E1* obtenemos: para el evento *E3* su causa y su cardinalidad es uno porque solamente interviene un evento primitivo en la formación del evento compuesto NEGACIÓN; para el evento *E1* sus causas son dos porque puede ocurrir por el disparo de las transiciones *T1* ó *T3*, la cardinalidad de *E1* también es dos porque intervienen dos eventos compuestos. Si las causas y la cardinalidad de un lugar de entrada de una transición t_j son las mismas, se puede inferir que cada causa es producida por un lugar primitivo. Si esto ocurre, se asigna una pseudocardinalidad (con valor uno) al lugar de entrada de t_j dado que, si sustituímos el lugar virtual por cualquiera de los lugares que lo componen, se sustituirá por un lugar primitivo. No obstante, el lugar virtual sigue correspondiendo

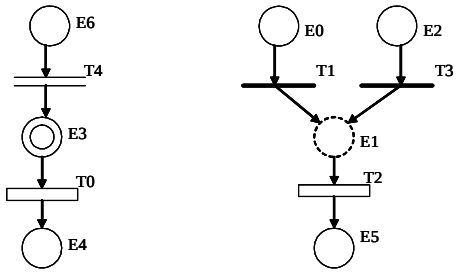

Figura 5.3: Reglas de ejemplo para el potencial de disparo para los eventos compuestos OR y NEGACIÓN

a un evento compuesto y la verdadera cardinalidad del evento corresponde al número de lugares primitivos necesarios para formarlo. De esta forma, el potencial de disparo para las transiciones *T0* y *T2* es:

$$TP\,(T0) = \tfrac{1}{1} \qquad TP\,(T2) = \tfrac{2}{1}$$

Como ya se mencionó, la cardinalidad de la transición *T2* es uno porque el valor de sus causas es igual al valor de su cardinalidad. Observando los valores del potencial de disparo, la transición *T2* de la figura 5.2 aún es más difícil de entender que la regla *T2* de la figura 5.3 ya que en la primera, además de analizar qué sucede cuando se disparan las transiciones *T0* ó *T1*, hay que analizar qué sucede cuando ocurre el lugar *E1*.

Este es un primer caso acerca del evento que activa una regla. Un segundo caso consiste en considerar los eventos compuestos (sin incluir los eventos OR y NEGACIÓN). Para formar los eventos compuestos se combinan eventos primitivos y compuestos utilizando algún operador del álgebra de eventos. Por esta razón, la cardinalidad de una regla puede ser mayor que sus causas. Como consecuencia, el potencial de disparo de una regla que involucre eventos compuestos disminuirá.

Para analizar este caso, vamos a tomar como ejemplo las reglas de la figura 5.4.

El valor de la causa del lugar *E3* es uno y el valor de su cardinalidad es tres porque son tres eventos primitivos los que intervienen en el evento compuesto. Por lo tanto, el valor de su potencial de disparo es:

$$TP\,(T0) = \frac{1}{3}$$

El lugar de entrada de la transición *T2* tiene el mismo valor de causa que el lugar de entrada de la transición *T0*. La cardinalidad del lugar *E1* es dos porque intervienen dos eventos primitivos en su formación. Por lo tanto, su potencial de disparo es:

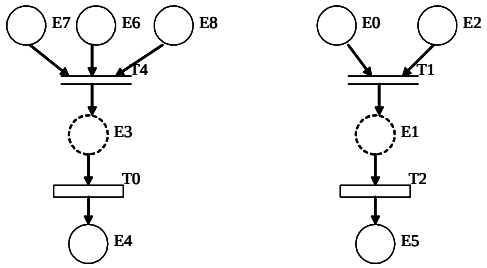

Figura 5.4: CCPN con eventos compuestos

$$TP\,(T2) = \frac{1}{2}$$

El potencial de disparo de la transición *T0* es menor que el potencial de disparo de la transición *T2*. Esto significa que una mayor cantidad de lugares primitivos se combinan para formar el lugar compuesto de *T0* y, como consecuencia, hay que esperar por la ocurrencia de más eventos para que la transición *T0* se pueda disparar. Desde este punto de vista, la regla representada por la transición *T0* es más difícil de entender que la regla representada por la transición *T2*.

En este caso un potencial de disparo menor indica que una regla es más difícil de entender. Esto se debe a que se involucran eventos compuestos.

A partir de la exposición de los casos, el cálculo del potencial de disparo se puede dividir en:

Caso I: El lugar de entrada de la transición $t_j \in T_{rule}$ es primitivo ó virtual (que represente el evento compuesto OR) ó compuesto (que represente el evento compuesto NEGACIÓN)

En este caso, un potencial de disparo mayor indica una mayor dificultad para entender una regla.

Caso II: El lugar de entrada de la transición $t_j \in T_{rule}$ es virtual (que represente a los eventos compuestos AND y ANY) ó es compuesto (que represente a los eventos compuestos con tiempo sin incluir la NEGACIÓN).

En este caso, un potencial de disparo menor indica mayor dificultad para entender una regla, debido a que hay una mayor cantidad de eventos primitivos que se combinan para formar un evento compuesto. Probablemente, existan menos causas que analizar, pero cada una de esas causas es más difícil de entender puesto que incluyen eventos compuestos.

Esta modelación se hizo sobre la estructura gráfica de la CCPN. Es útil porque permite visualizar cada

uno de los parámetros, no obstante, es impráctica cuando la CCPN es muy grande. Es un proceso tedioso para quien lo realiza el estar identificando, para cada una de las reglas, los parámetros descritos.

Ya que la CCPN también cuenta con un modelo matemático se puede aprovechar éste para definir sobre él los parámetros de medición y, por medio de algoritmos, automatizar el proceso de medición. La siguiente sección muestra la representación de los parámetros de medición sobre el modelo matemático de la CCPN.

5.3. Cálculo de los parámetros de medición sobre la matriz de incidencia

La representación matemática de la CCPN está dada por la matriz de incidencia. La matriz de incidencia de la figura 5.1 se muestra a continuación.

$$A = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{array} \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Al igual que la CCPN de la figura 5.1, la matriz de incidencia se obtuvo mediante la interfaz ECAPNSim. Los renglones de la matriz corresponden a las transiciones y las columnas representan los lugares. El valor absoluto del elemento $A_{i,j}$ representa el peso del arco que conecta la transición i con el lugar j. Si el valor de la entrada $A_{i,j}$ es cero, significa que no existe un arco de entrada entre la transición $t_i \in T$ y el lugar $p_j \in P$. Una entrada $A_{i,j}$ de la matriz con valor negativo, significa que el lugar $p_j \in P$ es un lugar de entrada a la transición $t_i \in T$. Una entrada $A_{i,j}$ de la matriz con un valor positivo significa que el lugar $p_j \in P$ es un lugar de salida de la transición $t_i \in T$. Para verificar la correspondencia entre el modelo gráfico y el modelo matemático de la CCPN, tomemos la columna 2 (la numeración de las columnas, al igual que la de

los renglones, inicia en cero). Esta columna tiene valores positivos en los renglones 5 y 8 y un valor negativo en el renglón 7. La interpretación de estos valores es que el lugar *E2* tiene un arco de entrada proveniente de la transición *T5*, un arco de entrada proveniente de la transición *T8* y un arco de salida hacia la transición *T7*, lo que coincide con el modelo gráfico de la figura 5.1.

Dada la correspondencia que existe entre los modelos gráfico y matemático de la CCPN, los conceptos que se definieron en la sección anterior (sobre el modelo gráfico) se pueden trasladar al modelo matemático. Este cambio entre modelos de la CCPN obedece a la facilidad que se tiene para manipular la matriz de incidencia mediante algoritmos. En la sección previa, se expuso que el proceso de identificar rutas, lugares iniciales, calcular distancias, etc. sobre la estructura gráfica es un proceso tedioso si la CCPN es muy grande. La ventaja de la matriz de incidencia, es que se pueden desarrollar algoritmos de forma que, sin importar las dimensiones de la matriz, los cálculos se realicen automáticamente.

En esta sección, se muestran los algoritmos desarrollados sobre la matriz de incidencia a fin de calcular los conceptos presentados en la sección anterior. Para todos los algoritmos se considera que:

1. La matriz de incidencia se encuentra almacenada en “matriz”.
2. Las transiciones tipo rule, copy y compuestas se encuentran almacenadas en “trule”, “tcopy” y “tcomp”, respectivamente.
3. Los lugares primitivos, compuestos, virtules y copy se encuentran almacenados en “lprim”, “lcomp”, “lvirt” y “lcopy”, respectivamente.
4. El número de renglones se almacena en “no_renglones”.
5. El número de columnas se almacena en “no_columnas”.

En la matriz de incidencia, un lugar es *inicial* si en su columna sólo existen valores de cero y un único valor negativo. El siguiente algoritmo permite encontrar todos los lugares iniciales de una matriz de incidencia.

1. `Desde i = 0 hasta i < no_columnas-1`
2. `    contador = 0`
3. `    Desde j = 0 hasta j < no_renglones-1`

```
4.        Si matriz[j][i] > 0
5.            contador = contador + 1
6.    Si contador == 0
7.        Agrega el índice 'i' a los 'l_iniciales'
```

Analizando la matriz de incidencia con este algoritmo, se encontró que las columnas 0, 3, 6, 9 y 10 cumplen con la definición de lugar inicial. Esas columnas corresponden con los lugares iniciales que se obtuvieron en el modelo gráfico, esto es, los lugares *E0*, *E3*, *E6*, *E8* y *E9*. Puede generar confusión que los lugares E8 y E9 "no correspondan" con las columnas 9 y 10 de la matriz de incidencia. Esto se debe a que en la CCPN se manejan dos índices: el índice de lugares y el índice de eventos. En la estructura gráfica se muestra en índice de eventos y el índice de lugares corresponde con las columnas de la matriz de incidencia.

Una *ruta* para una transición $t_j \in T_{rule}$ se construye alternando la búsqueda de valores negativos (en los renglones) y valores positivos (en las columnas) hasta llegar a un lugar inicial. Esto es, se localiza en el renglón j-ésimo un valor negativo. Sobre la columna donde se encontró el valor negativo, se busca un valor positivo. Nuevamente, se toma el renglón donde se encontró el valor positivo y se repite el proceso hasta que se alcanza un lugar inicial.

Por ejemplo, el proceso para identificar una ruta hacia la transición *T6* se inicia localizando el renglón 6 en la matriz de incidencia. En este renglón se busca un valor negativo. Ese valor se encuentra en la columna 1, lo que significa que existe un arco de entrada del lugar *E1* a la transición *T6*. En la columna uno, se busca un valor positivo, el cual se encuentra en el renglón 7, lo que muestra que hay un arco de salida de la transición *T7* al lugar *E1*. Procediendo de esta forma, en el renglón 7 hay un valor negativo en la columna 2. En la columna 2 existe un valor positivo en el renglón 5. En el renglón 5 hay un valor negativo en la columna 0. Como la columna 0 representa un lugar inicial, el proceso se detiene. La ruta que se obtiene es:

$$R\,(E0, T6) = E0 \rightarrow T5 \rightarrow E2 \rightarrow T7 \rightarrow E1 \rightarrow T6$$

La ruta anterior es una versión simplificada de la ruta que se obtiene en el modelo gráfico, no obstante, ambas rutas consideran los mismos lugares y transiciones para conectar el lugar *E0* con la transición *T6*.

La *ruta de transiciones* se obtiene extrayendo todas las transiciones $t \in \{T_{rule} \cup T_{comp}\}$ de una ruta. El siguiente algoritmo calcula todas las rutas de transiciones para todas las $t_j \in T_{rule}$, previamente se encuentran los lugares iniciales con el algoritmo correspondiente y se almacenan en 'l_iniciales'.

```
1.  Desde i  = 0 hasta N-1, donde N es el número de elementos de trule
2.      renglón = trule[i]
3.      ruta = 0
4.      Si renglón se encuentra en tcomp ó renglón se encuentra en trule
5.          Agrega renglón a 'ruta'
6.      Desde j = 0 hasta no_columnas-1
7.          Si matriz[renglón] [j] < 0
8.              lugar = j
9.              Si lugar se encuentra listado en 'l_iniciales'
10.                 Si Rutas[i] no contiene a ruta
11.                     Agrega 'ruta' a Rutas[i]
12.             Si no
13.                 Desde k = 0 hasta no_renglones-1
14.                     Si matriz[k] [lugar] > 0
15.                         renglón = k
16.                         Se repite el algoritmo desde el paso 4
```

La salida del algoritmo se encuentra en la variable Rutas.

Una vez que se tiene la ruta de transiciones, se puede calcular su *longitud*. Para tal efecto, se considera el peso del arco que conecta un lugar con una transición y viceversa. Cada una de las entradas de la matriz proporcionan la información necesaria, ya que, además de indicar el peso del arco, el signo indica el tipo. El siguiente algoritmo calcula la longitud de una ruta de transiciones, la ruta se encuentra almacenada en 'ruta'.

```
1. Desde i = 0 hasta N-1, donde N es el número de elementos de ruta
2.     elemento = ruta[i]
3.     Desde j = 0 hasta no_columnas-1
4.         Si matriz[elemento][j] < 0
5.             longitud = longitud + abs(matriz[elemento][j])
```

La función *abs* obtiene el valor absoluto de la entrada $A_{i,j}$. Esto se debe a que algunos elementos de la matriz tienen signo negativo para indicar el tipo de arco que conecta a la transición i con el lugar j. El resultado final se almacena en la variable longitud.

Las *causas del lugar de entrada de una transición* se calculan utilizando el siguiente algoritmo:

```
1. Desde i = 0 hasta N-1, donde N es el número de elementos de trule
2.     causas = 0
3.     Para todos los lugares p ∈ ·t
4.         Si p es copy
5.         Identifica su lugar de origen
6.         Si p es primitivo
7.             Si p es inicial
8.                 causas = causas + 1
9.             De otra forma
10.                Para todas las t ∈ ·p
11.                    causas = causas + w(t,p)
12.            causas = causas + 1
13.        Si p es virtual
```

```
14.            causas = 0
15.            Para todas las t ∈ ·p
16.                causas = causas + w(t,p)
17.        Si p es compuesto
18.            causas = 0
19.            causas = causas + 1
```

La cardinalidad del lugar de entrada de una transición $t_j \in T_{rule}$ se puede calcular mediante el algoritmo siguiente:

```
1.  Desde i = 0 hasta N, donde N es el número de elementos de trule
2.     cardinalidad = 0
3.     Desde j = 0 hasta no_columnas
4.        Si matriz[i][j] < 0
5.           l_entrada = j
6.           Si l_entrada se encuentra en lcopy
7.              Localizar su lugar de origen
8.              Repetir el algoritmo desde el paso 6
9.           Si l_entrada se encuentra en lprim
10.             cardinalidad = cardinalidad + 1
11.          Si l_entrada se encuentra en lvirtual ó l_entrada se
             encuentra en lcomp
12.             Desde k = 0 hasta no_renglones
```

13. Si matriz$[k]\,[l_entrada] > 0$
14. i = l_entrada
15. Repetir el algoritmo desde el paso 3

Para calcular el número de anclas de una transición $t_j \in T_{rule}$, se deben encontrar todas las rutas de transiciones hacia t_j. Una vez que se ha realizado este cálculo, para cada ruta se localiza la transición que inicia la ruta y se buscan sus lugares iniciales. El siguiente algoritmo describe detalladamente el proceso, todas las rutas para una transición $t_j \in T_{rule}$ se encuentran almacenadas en Rutas.

1. Desde i = 0 hasta N, donde N es el número de rutas para t_j
2. ruta = Rutas$[i]$
3. t_ini = ruta$[final]$, donde final es el total de elementos de ruta
4. Desde j = 0 hasta no_columnas
5. Si matriz$[t_ini]\,[j] < 0$
6. l_entrada = j
7. Si l_entrada se encuentra en l_iniciales y no se encuentra en ‘anclas’
8. Agrega l_entrada a anclas
9. Si no
10. Desde k = 0 hasta no_renglones
11. Si matriz$[k]\,[l_entrada] > 0$
12. Repite el algoritmo desde el paso 4

El número de anclas para t_j es el número de elementos que se almacenaron en ‘anclas’.

La *distancia* de una transición se obtiene calculando el valor máximo de todas las rutas que conectan un lugar inicial con la transición $t_j \in T_{rule}$. Utilizamos el algoritmo siguiente para calcular este parámetro de medición, las rutas para t_j se encuentran en Rutas:

```
1. ruta = Rutas[0]
2. max = Longitud de ruta. Se utiliza el algoritmo correspondiente
   para obtener este valor
3. Desde i = 1 hasta N, donde N es el número de rutas para tj
4.     ruta = Rutas[i]
5.     max1 = Longitud de ruta. Se utiliza el algoritmo correspondiente
            para obtener este valor
6.     Si max1 > max
7.         max = max1
```

El resultado se almacena en la variable max.

El potencial de disparo se obtiene calculando el cociente de las causas del lugar de entrada de t_j entre la cardinalidad del lugar de entrada de t_j. Mediante el siguiente algoritmo se puede obtener el valor de este parámetro:

```
1. pseudocardinalidad = 1
2. vcausas = Valor de las causas para el lugar de entrada de tj. Se
   obtiene este valor utilizando el algoritmo correspondiente
3. vcardin = Valor de la cardinalidad para el lugar de entrada de tj.
   Se obtiene este valor utilizando el algoritmo correspondiente
4. Si vcausas == vcardina
5.     vcardin = pseudocardinalidad
6. tp = vcausas / vcardin
```

El resultado se almacena en la variable vcardin.

5.4. Validación de los parámetros de medición

Una vez definidos los parámetros para la medición de la complejidad de la interacción de las reglas ECA, es necesario verificar que sean parámetros válidos. Para esto, se deben de ubicar en un marco de trabajo que proporcione las condiciones necesarias para analizar su validez. El marco de trabajo propuesto por el Dr. H. Zuse se enfoca a la validación de medidas de software. Este marco se compone por un conjunto de axiomas que dan las condiciones para la traducción de enunciados empíricos a enunciados formales y se apoya fuertemente sobre la teoría de la medición clásica. En este trabajo de tesis se eligió el marco de trabajo del Dr. Zuse porque cuenta con un apoyo matemático y existe una amplia documentación acerca de él. El capítulo II presenta una descripción de los axiomas que componen el marco de trabajo.

Para realizar la verificación de los parámetros de medición es necesario establecer:

1. A es un conjunto no vacío de reglas (también se le denomina $\Im$)
2. $\cdot \geqslant$ es la relación empírica “más o igual compleja que” sobre A
3. $\circ$ es una operación (concatenación) cerrada binaria sobre A tal que la concatenación de reglas $A_1\,(E_1, C_1, Ac_1)$ y $A_2\,(E_2, C_2, Ac_2)$ produce la regla A_3, donde:
   - *a*) el evento E_3 se obtiene como $E_1 OR\, E_2$
   - *b*) la condición C_3 se obtiene como $(C_1\ AND\ e_1)\ OR\ (C_2\ AND\ e_2)$ donde e_i es una variable booleana donde true(false) indica la ocurrencia (ausencia) del evento E_i
   - *c*) la acción Ac_3 se puede definir como: $IF\ e_1\ THEN\ Ac_1, IF\ e_2\ THEN\ Ac_2$
4. El signo $\approx$ representa la relación empírica “equivalente en complejidad”. Significa que: $A_1 \approx A_2 \Leftrightarrow A_1 \cdot \geqslant A_2\ \&\ A_2 \cdot \geqslant A_1$
5. Para dos reglas $A_1\,(E1, C1, Ac1)$ y $A_2\,(E2, C2, Ac2)\,;\ A_1\ \supseteq\ A_2$ significa que todos los eventos simples que componen E_2 están presentes en E_1. Lo mismo para las condiciones y las acciones.

La exposición, en secciones anteriores, de los parámetros de medición se formaliza para poder realizar la verificación sobre el marco de trabajo. En cada uno de los apartados siguientes se muestra la definición formal de cada parámetro así como la verificación de cada uno de los axiomas.

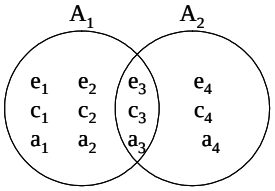

Figura 5.5: Intersección de la regla A_1 y la regla A_2

5.4.1. Número de Anclas

El número de anclas de una transición $t_j \in T_{rule}$, $NA\left(t_j\right)$, es el número de lugares iniciales que se necesitan para abarcar todas las causas del lugar de entrada de la transición t_j.

Formalmente, el parámetro NA es un mapeo, $NA : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow NA\left(A_i\right) \geqslant NA\left(A_j\right); \forall A_i, A_j \in A \tag{5.1}$$

Su operación de concatenación $\circ$ se define como:

$$NA\left(A_i \circ A_j\right) = NA\left(A_i\right) + NA\left(A_j\right) - NA\left(A_i \cap A_j\right) \tag{5.2}$$

donde $NA\left(A_i \cap A_j\right)$ es el número de anclas que tienen en común la regla A_i y la regla A_j a lo largo de la intersección.

El símbolo $\cap$ denota la intersección entre los elementos que forman la regla A_i con los elementos que forman la regla A_j. Por ejemplo, la figura 5.5 muestra que la regla A_1 se forma por los eventos e_1, e_2 y e_3, las condiciones c_1, c_2 y las acciones a_1 y a_2. La regla A_2 se forma por los eventos e_3 y e_4, las condiciones c_3 y c_4 y las acciones a_3 y a_4. Entonces, los elementos que tienen en común son e_3, c_3 y a_3. De esta intersección es de donde se calcula el número de anclas, $NA\left(A_1 \cap A_2\right)$.

Primero se verifica si el parámetro NA es una estructura extensiva modificada mostrando si cumple los axiomas de esta estructura.

Axiomas de la estructura extensiva modificada

Axioma 1. Orden débil $(A, \cdot \geqslant)$

Un orden débil tiene las siguientes características:

(i) $A_1 \cdot \geqslant A_2, A_2 \cdot \geqslant A_3 \Rightarrow A_1 \cdot \geqslant A_3$

(ii) $A_1 \cdot \geqslant A_2$ ó $A_2 \cdot \geqslant A_1$

Para demostrar (i) , utilizando la definición 5.1 se tiene que:

$A_1 \cdot \geqslant A_2, A_2 \cdot \geqslant A_3 \Longrightarrow A_1 \cdot \geqslant A_3 \Leftrightarrow NA(A_1) \geqslant NA(A_2), NA(A_2) \geqslant NA(A_3) \Rightarrow NA(A_1) \geqslant NA(A_3)$

Consideremos $NA(A_1) \geqslant NA(A_2), NA(A_2) \geqslant NA(A_3) \Rightarrow NA(A_1) \geqslant NA(A_3)$

Las hipótesis del enunciado son: $NA(A_1) \geqslant NA(A_2), NA(A_2) \geqslant NA(A_3)$

La conclusión del enunciado es: $NA(A_1) \geqslant NA(A_3)$

En los números reales, sabemos que si $a \geqslant 0$ y $b \geqslant 0$ entonces $a + b \geqslant 0$. Y que $a \geqslant b$ si $a - b \geqslant 0$. Utilizando estas propiedades, tenemos:

$NA(A_1) \geqslant NA(A_2)$ si $NA(A_1) - NA(A_2) \geqslant 0$

$NA(A_2) \geqslant NA(A_3)$ si $NA(A_2) - NA(A_3) \geqslant 0$

Como $NA(A_1) - NA(A_2) \geqslant 0$ y $NA(A_2) - NA(A_3) \geqslant 0$ entonces $[NA(A_1) - NA(A_2)] + [NA(A_2) - NA(A_3)] \geqslant 0$

Los términos $-NA(A_2) + NA(A_2)$ se cancelan y se obtiene $NA(A_1) - NA(A_3) \geqslant 0$. Por lo tanto, $NA(A_1) \geqslant NA(A_3)$

Entonces (i) se cumple.

Para la parte (ii) , por la definición 5.1:

$A_1 \cdot \geqslant A_2$ ó $A_2 \cdot \geqslant A_1 \Leftrightarrow NA(A_1) \geqslant NA(A_2)$ ó $NA(A_2) \geqslant NA(A_1)$

Consideremos $NA(A_1) \geqslant NA(A_2)$ ó $NA(A_2) \geqslant NA(A_1)$

$NA(A_1) \geqslant NA(A_2)$ significa que $NA(A_1) > NA(A_2)$ ó $NA(A_1) = NA(A_2)$

Supongamos que $NA(A_1) > NA(A_2)$ es falsa. Entonces si $no\,(NA(A_1) > NA(A_2))$ se obtiene que $NA(A_2) \geqslant NA(A_1)$

Lo que demuestra que este axioma se cumple.

Entonces:

$$A_1 \cdot \geqslant A_2 \text{ ó } A_2 \cdot \geqslant A_1 \Leftrightarrow NA(A_1) \geqslant NA(A_2) \text{ ó } NA(A_2) \geqslant NA(A_1)$$

Axioma 2. Positividad. $A_1 \circ A_2 \cdot \geqslant A_1$

De la definición 5.1 se tiene:

$A_1 \circ A_2 \cdot \geqslant A_1 \Leftrightarrow NA(A_1 \circ A_2) \geqslant NA(A_1)$

Consideremos $NA(A_1 \circ A_2) \geqslant NA(A_1)$

De la definición 5.2, sabemos que $NA(A_1 \circ A_2) = NA(A_1) + NA(A_2) - NA(A_1 \cap A_2)$

Entonces $NA(A_1) + NA(A_2) - NA(A_1 \cap A_2) \geqslant NA(A_1)$

El valor de $NA(A_1 \cap A_2)$ se encuentra en el intervalo $[0, \text{mín}(NA(A_1), NA(A_2))]$ donde

$\text{mín}(NA(A_1), NA(A_2))$ representa el valor mínimo de los valores $NA(A_1)$ y $NA(A_2)$. Esto significa que:

(i) Cuando el valor de $NA(NA(A_1) \cap NA(A_2))$ es 0, las reglas A_1 y A_2 se originan por anclas completamente distintas. Debido a esto y tomando en cuenta que cada regla tiene siempre, al menos, un ancla, se tiene:

$$\begin{array}{ccccccc} NA(A_1) & + & NA(A_2) & - & NA(A_1 \cap A_2) & \geqslant & NA(A_1) \\ 1 & + & 1 & - & 0 & > & 1 \end{array}$$

(ii) Cuando el valor de $NA(NA(A_1) \cap NA(A_2))$ es $\text{mín}(NA(A_1), NA(A_2))$ pueden suceder tres cosas:

(a) Las reglas A_1 y A_2 se originan por las mismas anclas. Entonces, el valor de $NA(A_1 \cap A_2) = NA(A_1)$

$$\begin{array}{ccccccc} NA(A_1) & + & NA(A_2) & - & NA(A_1 \cap A_2) & \geqslant & NA(A_1) \\ 1 & + & 1 & - & 1 & = & 1 \end{array}$$

(b) El conjunto de anclas de la regla A_2 es un subconjunto propio del conjunto de anclas de la regla A_1. Por lo que $NA(A_1) > NA(A_2)$ y $NA(A_1 \cap A_2) = NA(A_2)$

$$\begin{array}{ccccccc} NA(A_1) & + & NA(A_2) & - & NA(A_1 \cap A_2) & \geqslant & NA(A_1) \\ 2 & + & 1 & - & 1 & > & 1 \end{array}$$

(c) El conjunto de anclas de la regla A_1 es un subconjunto propio del conjunto de anclas de la regla A_2. Por lo que $NA(A_2) > NA(A_1)$ y $NA(A_1 \cap A_2) = NA(A_1)$

$$\begin{array}{ccccccc} NA(A_1) & + & NA(A_2) & - & NA(A_1 \cap A_2) & \geqslant & NA(A_1) \\ 1 & + & 2 & - & 1 & > & 1 \end{array}$$

De esta forma, en todos los casos se cumple $NA(A_1 \cap A_2) \geqslant NA(A_1)$.

Finalmente,

$$A_1 \circ A_2 \cdot \geqslant A_1 \Leftrightarrow NA(A_1 \circ A_2) \geqslant NA(A_1)$$

Axioma 3. Asociatividad débil. $A_1 \circ (A_2 \circ A_3) \approx (A_1 \circ A_2) \circ A_3$

El signo $\approx$ significa:

$A_1 \approx A_2 \Leftrightarrow A_1 \cdot \geqslant A_2 \;\&\; A_2 \cdot \geqslant A_1$

Entonces $A_1 \approx A_2 \Leftrightarrow NA(A_1) \geqslant NA(A_2) \;\&\; NA(A_2) \geqslant NA(A_1)$

De la definición 5.1 se tiene:

$A_1 \circ (A_2 \circ A_3) \approx (A_1 \circ A_2) \circ A_3 \Leftrightarrow NA(A_1 \circ (A_2 \circ A_3)) \geqslant NA((A_1 \circ A_2) \circ A_3) \;\&$

$NA((A_1 \circ A_2) \circ A_3) \geqslant NA(A_1 \circ (A_2 \circ A_3))$

Consideremos $NA(A_1 \circ (A_2 \circ A_3)) \geqslant NA((A_1 \circ A_2) \circ A_3)$

Tomando los paréntesis más internos y la definición 5.2, se tiene que:

$NA(A_2 \circ A_3) = NA(A_2) + NA(A_3) - NA(A_2 \cap A_3)$

$NA(A_1 \circ A_2) = NA(A_1) + NA(A_2) - NA(A_1 \cap A_2)$

Aplicando el axioma anterior se obtiene:

$NA(A_2 \circ A_3) \geqslant NA(A_2)$ y

$NA(A_1 \circ A_2) \geqslant NA(A_1)$

Por simplicidad, vamos a considerar el caso de igualdad:

$NA(A_2 \circ A_3) = NA(A_2)$

$NA(A_1 \circ A_2) = NA(A_1)$

Por la definición 5.1:

$NA(A_2) \Leftrightarrow A_1$

$NA(A_1) \Leftrightarrow A_2$

Con este resultado, podemos sustituir:

$NA(A_1 \circ (A_2 \circ A_3)) = NA(A_1 \circ A_2) = NA(A_1) + NA(A_2) - NA(A_1 \cap A_2)$

$NA((A_1 \circ A_2) \circ A_3) = NA(A_1 \circ A_3) = NA(A_1) + NA(A_3) - NA(A_1 \cap A_3)$

Nuevamente, utilizando el axioma anterior y tomando el caso de igualdad:

$NA(A_1 \circ A_2) = NA(A_1)$

$NA(A_1 \circ A_3) = NA(A_1)$

Tenemos que: $NA(A_1) = NA(A_1)$

Para el caso $NA((A_1 \circ A_2) \circ A_3) \geqslant NA(A_1 \circ (A_2 \circ A_3))$ es un análisis similar.

Finalmente,

$$A_1 \circ (A_2 \circ A_3) \approx (A_1 \circ A_2) \circ A_3 \Leftrightarrow$$
$$NA(A_1 \circ (A_2 \circ A_3)) \geqslant NA((A_1 \circ A_2) \circ A_3) \ \& \ NA((A_1 \circ A_2) \circ A_3) \geqslant NA(A_1 \circ (A_2 \circ A_3))$$

Axioma 4. Conmutatividad débil. $A_1 \circ A_2 \approx A_2 \circ A_1$

El signo $\approx$ significa:

$A_1 \approx A_2 \Leftrightarrow A_1 \cdot \geqslant A_2 \ \& \ A_2 \cdot \geqslant A_1$

Entonces $A_1 \approx A_2 \Leftrightarrow NA(A_1) \geqslant NA(A_2) \ \& \ NA(A_2) \geqslant NA(A_1)$

De la definición 5.1 se tiene:

$A_1 \circ A_2 \approx A_2 \circ A_1 \Leftrightarrow NA(A_1 \circ A_2) \geqslant NA(A_2 \circ A_1) \ \& \ NA(A_2 \circ A_1) \geqslant NA(A_1 \circ A_2)$

Consideremos $NA(A_1 \circ A_2) \geqslant NA(A_2 \circ A_1)$

De la definición 5.1 se tiene que:

$NA(A_1 \circ A_2) = NA(A_1) + NA(A_2) - NA(A_1 \cap A_2)$

$NA(A_2 \circ A_1) = NA(A_2) + NA(A_1) - NA(A_2 \cap A_1)$

Por lo que:

$NA(A_1) + NA(A_2) - NA(A_1 \cap A_2) \geqslant NA(A_2) + NA(A_1) - NA(A_2 \cap A_1)$

Reacomodando términos:

$NA(A_1) + NA(A_2) - NA(A_1 \cap A_2) \geqslant NA(A_1) + NA(A_2) - NA(A_2 \cap A_1)$

Como el resultado de $NA(A_1 \cap A_2) = NA(A_2 \cap A_1)$ se cumple la primera parte.

Consideremos ahora $NA(A_2 \circ A_1) \geqslant NA(A_1 \circ A_2)$

De la definición de concatenación:

$NA(A_2) + NA(A_1) - NA(A_2 \cap A_1) \geqslant NA(A_1) + NA(A_2) - NA(A_1 \cap A_2)$

Reacomodando términos:

$NA(A_2) + NA(A_1) - NA(A_2 \cap A_1) \geqslant NA(A_2) + NA(A_1) - NA(A_1 \cap A_2)$

Como el resultado de $NA(A_2 \cap A_1) = NA(A_1 \cap A_2)$ se cumple la segunda parte.

Finalmente,

$$A_1 \circ A_2 \approx A_2 \circ A_1 \Leftrightarrow NA(A_1 \circ A_2) \geqslant NA(A_2 \circ A_1) \,\&\, NA(A_2 \circ A_1) \geqslant NA(A_1 \circ A_2)$$

Axioma 5. Monotonicidad débil. $A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A$

De la definición 5.1 se tiene:

$A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow NA(A_1) \geqslant NA(A_2) \Rightarrow NA(A_1 \circ A) \geqslant NA(A_2 \circ A)$

Consideremos $NA(A_1) \geqslant NA(A_2) \Rightarrow NA(A_1 \circ A) \geqslant NA(A_2 \circ A)$

Podemos aplicar la operación de concatenación a ambos lados de $A_1 \cdot \geqslant A_2$ y obtenemos:

$A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow NA(A_1 \circ A) \geqslant NA(A_2 \circ A)$

Consideremos $NA(A_1 \circ A) \geqslant NA(A_2 \circ A)$, entonces:

$(a)\ NA(A_1 \circ A) = NA(A_1) + NA(A) - NA(A_1 \cap A)$

$(b)\ NA(A_2 \circ A) = NA(A_2) + NA(A) - NA(A_2 \cap A)$

$(c)\ NA(A_1) + NA(A) - NA(A_1 \cap A) \geqslant NA(A_2) + NA(A) - NA(A_2 \cap A)$

Nada impide que la regla A_1 y la regla A tengan las mismas anclas. Entonces, $NA(A_1 \cap A) = NA(A_1)$. Si esto sucede, en el lado izquierdo del signo $\geqslant$ de la desigualdad (c) el valor máximo que se obtendrá es $NA(A)$. Por lo que:

$NA(A) \geqslant NA(A_2) + NA(A) - NA(A_2 \cap A)$ y esta desigualdad no se cumple. De forma general, se puede tener:

$NA(A_1) \geqslant NA(A_2)$ y $no\,(NA(A_1 \circ A) \geqslant NA(A_2 \circ A))$

Por lo que el axioma no se cumple.

Como ejemplo, supongamos que tenemos los siguientes valores de NA para las reglas A_1, A y A_2:

$NA(A_1) = 3$

$NA(A) = 3$

$NA(A_1 \cap A) = 3$

$NA(A_2) = 2$

$NA(A_2 \cap A) = 0$

Entonces:

$$\begin{array}{ccccccc} NA(A) & \geqslant & NA(A_2) & \Rightarrow & NA(A_1 \circ A) & \geqslant & NA(A_2 \circ A) \\ 3 & > & 2 & \Rightarrow & 3+3-3 & \geqslant & 2+3-0 \\ 3 & > & 2 & \Rightarrow & 3 & \not> & 5 \end{array}$$

Axioma 6. Axioma de Arquímedes. Si $A_3 \cdot > A_4$ entonces para cualquier A_1, A_2 existe un número natual n, tal que $A_1 \circ nA_3 \cdot > A_2 \circ nA_4$

$A_3 \cdot > A_4 \Rightarrow A_1 \circ nA_3 \cdot > A_2 \circ nA_4$

Esto es como si se tuviera:

$A_3 \cdot > A_4 \Rightarrow A_1 \circ A_3 \circ A_3 \circ ... \circ A_3 \cdot > A_2 \circ A_4 \circ ... \circ A_4 \Leftrightarrow$

$NA\,(A_3) > NA\,(A_4) \Rightarrow NA\,(A_1 \circ A_3 \circ ... \circ A_3) > NA\,(A_2 \circ A_4 \circ ... \circ A_4)$

En este caso, $NA\,(A_3 \circ A_3) = NA\,(A_3)$, por lo tanto, el parámetro NA es una operación idempotente. El axioma de Arquímedes no se puede cumplir si la operación es idempotente, por lo tanto, no se cumple este axioma.

Axiomas de las condiciones de independencia

Condición de Independencia 1. $A_1 \approx A_2 \Rightarrow A_1 \circ A \approx A_2 \circ A$ y $A_1 \approx A_2 \Rightarrow A \circ A_1 \approx A \circ A_2$

Por el punto 4 definido al inicio de esta sección, tenemos:

$A_1 \approx A_2 \Leftrightarrow A_1{\cdot} \geqslant A_2$ & $A_2{\cdot} \geqslant A_1$

Entonces $A_1 \approx A_2 \Leftrightarrow NA\,(A_1) \geqslant NA\,(A_2)$ & $NA\,(A_2) \geqslant NA\,(A_1)$

De la definición 5.1 se tiene:

$A_1 \circ A \approx A_2 \circ A \Leftrightarrow NA\,(A_1 \circ A) \geqslant NA\,(A_2 \circ A)$ & $NA\,(A_2 \circ A) \geqslant NA\,(A_1 \circ A)$

Consideremos $NA\,(A_1 \circ A) \geqslant NA\,(A_2 \circ A)$

Se puede observar que este es el axioma de monotonicidad de la estructura extensiva modificada. Como se mostró en ese axioma, puede suceder que si la regla A_1 y la regla A tienen las mismas causas, entonces: $NA\,(A_1 \circ A) \ngeq NA\,(A_2 \circ A)$. Por lo tanto, este axioma no se cumple.

Condición de Independencia 2. $A_1 \approx A_2 \Leftrightarrow A_1 \circ A \approx A_2 \circ A$ y $A_1 \approx A_2 \Leftrightarrow A \circ A_1 \approx A \circ A_2$

Este axioma tampoco se cumple porque no se cumple el axioma de monotonicidad de la estructura extensiva modificada.

Condición de Independencia 3. $A_1{\cdot} \geqslant A_2 \Rightarrow A_1 \circ A{\cdot} \geqslant A_2 \circ A$ y $A_1{\cdot} \geqslant A_2 \Rightarrow A \circ A_1{\cdot} \geqslant A \circ A_2$

Este axioma es similar a los anteriores porque involucra la propiedad de monotonicidad. Como tal propiedad no se cumple, tampoco se cumple este axioma.

Condición de Independencia 4. $A_1{\cdot} \geqslant A_2 \Leftrightarrow A_1 \circ A{\cdot} \geqslant A_2 \circ A$ y $A_1{\cdot} \geqslant A_2 \Leftrightarrow A \circ A_1{\cdot} \geqslant A \circ A_2$

Este punto no se cumple porque no cumple con la propiedad de monotonicidad.

Axiomas de la relación modificada de confianza

Relación Modificada de Confianza 1. $\forall A, B \in \Im : A{\cdot} \geqslant B$ ó $B{\cdot} \geqslant A$

Este axioma se cumple porque en los números reales se cumple la propiedad de completitud.

Relación Modificada de Confianza 2. $\forall A, B, C \in \Im : A{\cdot} \geqslant B$ y $B{\cdot} \geqslant C \Rightarrow A{\cdot} \geqslant C$

Este axioma se cumple porque en los números reales se cumple la propiedad de transitividad.

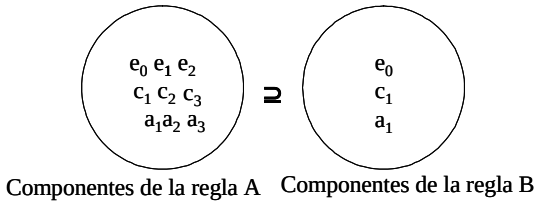

Figura 5.6: Ejemplo para la relación modificada de confianza 3

Relación Modificada de Confianza 3. $\forall A \supseteq B \Rightarrow A\cdot \geqslant B$

Hay que recordar que una regla está formada por tres partes: evento, condición y acción. Entonces la regla $A\left(E_1, C_1, A_1\right) \supseteq B\left(E_2, C_2, A_2\right)$ si todos los eventos que componen E_2 están presentes en E_1, de forma análoga para la condición C_2 y las instrucciones de la acción A_2.

Consideremos la figura 5.6, en ella se muestran los componentes de la regla A y de la regla B. Como puede observarse $A \supseteq B$.

Dado que NA representa la cardinalidad del conjunto de anclas, también se cumple $NA\left(A\right) \geqslant NA\left(B\right)$ siempre que $A \supseteq B$.

Por lo tanto,

$$\forall A \supseteq B \Rightarrow A\cdot \geqslant B \Leftrightarrow NA\left(A\right) \geqslant NA\left(B\right)$$

Relación Modificada de Confianza 4. $(\forall A \supset B, A \cap C = \varnothing) \Rightarrow (A\cdot > B \Rightarrow A \cup C\cdot > B \cup C)$

Del axioma anterior podemos decir que si $A \supset B$ entonces $A\cdot > B$ y $NA\left(A\right) > NA\left(B\right)$. Por otro lado, $A \cap C$ significa la intersección entre los elementos de la regla A y los elementos de la regla C. Entonces $A \cap C = \varnothing$ quiere decir que las causas de la regla A son diferentes de las causas de la regla C, de forma similar con las condiciones y las acciones. Sin embargo, aún cuando la intesección puede ser vacía, las causas de A y las causas de C se pueden generar por las mismas anclas. Por lo tanto, se puede tener $A \cap C = \varnothing$ y $NA\left(A \cap C\right) \neq 0$. Por esta razón, cuando se unen la regla A con la regla C el resultado, en cuanto a las anclas, es que son las mismas, es decir, $A \cup C = A$ y esto no es mayor que las anclas de $B \cup C$. De esta forma, este axioma no se cumple.

Por ejemplo, supongamos que el evento de la regla A es el evento compuesto *and(E0, E1)* y que el evento de la regla B es *E0*. De esta forma, $A \supset B$. El evento de la regla C es el evento compuesto *secuencia(E0, E1)*, entonces $A \cap C = \varnothing$ porque el evento de la regla A (and) es diferente del evento de la regla C (secuencia). Sin embargo, las anclas del evento de la regla A son las mismas que las anclas de la regla C

(*E0, E1*), la diferencia es el operador de eventos. Cuando se hace $A \cup C$ el resultado es el conjunto {*E0, E1*} y cuando se hace $B \cup C$ el resultado es el conjunto {*E0, E1*}. Como se puede observar, es el mismo. Entonces $NA\left(A \cup C\right) = 2$ y $NA\left(B \cup C\right) = 2$, y como $2 \not> 2$, entonces el axioma no se cumple.

Relación Modificada de Confianza 5.1 $\forall A \in \Im : A \cdot \geqslant 0$

Esta relación modificada de confianza se cumple porque una regla tiene siempre, al menos, un ancla.

5.4.2. Distancia

Dada una transición $t_j \in T_{rule}$ en la CCPN, la distancia de la transición t_j, $D\left(t_j\right)$, se define como la longitud de la ruta de transiciones más larga que conecta a la regla t_j con un lugar inicial.

Formalmente, el parámetro D es un mapeo $D : S \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow D\left(A_i\right) \geqslant D\left(A_j\right) \forall A_i, A_j \in A \tag{5.3}$$

Su operación de concatenación $\circ$ se define de la siguiente manera:

$$D\left(A_i \circ A_j\right) = MAX\left(D\left(A_i\right), D\left(A_j\right)\right) \tag{5.4}$$

Axiomas de la estructura extensiva modificada

Axioma 1. Orden débil. $(A, \cdot \geqslant)$

Este axioma se cumple porque las propiedades de transitividad y completitud se cumplen en los números reales. La demostración de tales propiedades para este parámetro de medición es similar a la del axioma de orden débil para el número de anclas.

Axioma 2. Positividad. $A_1 \circ A_2 \cdot \geqslant A_1$

De la definición 5.3 se tiene:

$A_1 \circ A_2 \cdot \geqslant A_1 \Leftrightarrow D\left(A_1 \circ A_2\right) \geqslant D\left(A_1\right)$

Consideremos $D\left(A_1 \circ A_2\right) \geqslant D\left(A_1\right)$

De la definición 5.4, sabemos que $D\left(A_1 \circ A_2\right) = MAX\left(D\left(A_1\right), D\left(A_2\right)\right)$

Entonces, $MAX\left(D\left(A_1\right), D\left(A_2\right)\right) \geqslant D\left(A_1\right)$

Se puede observar que si el valor máximo es $D\left(A_1\right)$ entonces se cumple el caso de igualdad

$MAX\left(D\left(A_1\right), D\left(A_2\right)\right) = D\left(A_1\right)$. Si el valor máximo es $D\left(A_2\right)$ entonces se cumplirá el caso $>$. De esta forma, este axioma se cumple.

$$A_1 \circ A_2 \cdot \geqslant A_1 \Leftrightarrow D\left(A_1 \circ A_2\right) \geqslant D\left(A_1\right)$$

Axioma 3. Asociatividad débil. $A_1 \circ \left(A_2 \circ A_3\right) \approx \left(A_1 \circ A_2\right) \circ A_3$

Como sabemos lo que significa el símbolo $\approx$, podemos tener:

$A_1 \approx A_2 \Leftrightarrow D\left(A_1\right) \geqslant D\left(A_2\right) \,\&\, D\left(A_2\right) \geqslant D\left(A_1\right)$

De la definición 5.3 se tiene:

$A_1 \circ \left(A_2 \circ A_3\right) \approx \left(A_1 \circ A_2\right) \circ A_3 \Leftrightarrow D\left(A_1 \circ \left(A_2 \circ A_3\right)\right) \geqslant D\left(\left(A_1 \circ A_2\right) \circ A_3\right) \,\&\, D\left(\left(A_1 \circ A_2\right) \circ A_3\right) \geqslant D\left(A_1 \circ \left(A_2 \circ A_3\right)\right)$

Consideremos $D\left(A_1 \circ \left(A_2 \circ A_3\right)\right) \geqslant D\left(\left(A_1 \circ A_2\right) \circ A_3\right)$

Tomando los paréntesis más internos y la definición 5.4, se tiene que:

$D\left(A_2 \circ A_3\right) = MAX\left(D\left(A_2\right), D\left(A_3\right)\right)$

$D\left(A_1 \circ A_2\right) = MAX\left(D\left(A_1\right), D\left(A_2\right)\right)$

Aplicando el axioma anterior se sabe:

$D\left(A_2 \circ A_3\right) \geqslant D\left(A_2\right)$ y

$D\left(A_1 \circ A_2\right) \geqslant D\left(A_1\right)$

Por simplicidad, tomamos el caso de igualdad:

$D\left(A_2 \circ A_3\right) = D\left(A_2\right)$

$D\left(A_1 \circ A_2\right) = D\left(A_1\right)$

Por la definición 5.3:

$D\left(A_2\right) \Leftrightarrow A_1$

$D\left(A_1\right) \Leftrightarrow A_2$

Con este resultado, podemos sustituir:

$D\left(A_1 \circ \left(A_2 \circ A_3\right)\right) = D\left(A_1 \circ A_2\right) = MAX\left(D\left(A_1\right), D\left(A_2\right)\right)$

$D\left(\left(A_1 \circ A_2\right) \circ A_3\right) = D\left(A_1 \circ A_3\right) = MAX\left(D\left(A_1\right), D\left(A_3\right)\right)$

Nuevamente, utilizando el axioma anterior, tomamos el caso de igualdad:

$D\left(A_1 \circ A_2\right) = D\left(A_1\right)$

$D\left(A_1 \circ A_3\right) = D\left(A_1\right)$

Tenemos que: $D\left(A_1\right) = D\left(A_1\right)$

Para el caso $D((A_1 \circ A_2) \circ A_3) \geqslant D(A_1 \circ (A_2 \circ A_3))$ es un análisis similar.

Finalmente,

$$A_1 \circ (A_2 \circ A_3) \approx (A_1 \circ A_2) \circ A_3 \Leftrightarrow$$
$$D(A_1 \circ (A_2 \circ A_3)) \geqslant D((A_1 \circ A_2) \circ A_3) \ \& \ D((A_1 \circ A_2) \circ A_3) \geqslant D(A_1 \circ (A_2 \circ A_3))$$

Axioma 4. Conmutatividad débil. $A_1 \circ A_2 \approx A_2 \circ A_1$

Tomando el punto 4. del inicio de esta sección, sabemos que:

$A_1 \approx A_2 \Leftrightarrow D(A_1) \geqslant D(A_2) \ \& \ D(A_2) \geqslant D(A_1)$

De la definición 5.3 se tiene:

$A_1 \circ A_2 \approx A_2 \circ A_1 \Leftrightarrow D(A_1 \circ A_2) \geqslant D(A_2 \circ A_1) \ \& \ D(A_2 \circ A_1) \geqslant D(A_1 \circ A_2)$

Consideremos $D(A_1 \circ A_2) \geqslant D(A_2 \circ A_1)$

De la definición 5.3 se tiene que:

$D(A_1 \circ A_2) = MAX(D(A_1), D(A_2))$

$D(A_2 \circ A_1) = MAX(D(A_2), D(A_1))$

Por lo que:

$MAX(D(A_1), D(A_2)) \geqslant MAX(D(A_2), D(A_1))$

La función MAX siempre obtiene el máximo valor sin importar el orden de sus argumentos. Por esta razón, en el lado izquierdo del símbolo $\geqslant$ cualquiera que sea el valor máximo será el mismo que se obtenga en la parte derecha.

Para la parte $D(A_2 \circ A_1) \geqslant D(A_1 \circ A_2)$ es un análisis similar.

Finalmente, el axioma se cumple.

$$A_1 \circ A_2 \approx A_2 \circ A_1 \Leftrightarrow D(A_1 \circ A_2) \geqslant D(A_2 \circ A_1) \,\&\, D(A_2 \circ A_1) \geqslant D(A_1 \circ A_2)$$

Axioma 5. Monotonicidad débil. $A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A$

Por la definición 5.3, se sabe que:

$A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$

Consideremos $D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$

Podemos concatenar una tercer regla A a ambos lados de $A_1 \cdot \geqslant A_2$. Entonces:

$A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow NA(A_1 \circ A) \geqslant NA(A_2 \circ A)$

De la definición de concatenación 5.4, sabemos que:

$D(A_1 \circ A) = MAX(D(A_1), D(A))$

$D(A_2 \circ A) = MAX(D(A_2), D(A))$

Entonces,

$MAX(D(A_1), D(A)) \geqslant MAX(D(A_2), D(A))$

En este caso, si $D(A)$ es el valor máximo, la relación entre $D(A_1)$ y $D(A_2)$ no se ve alterada. Por lo tanto, el axioma se cumple.

$$A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$$

Axioma 6. Axioma de Arquímedes. Si $A_3 \cdot > A_4$ entonces para cualquier A_1, A_2 existe un número natual n, tal que $A_1 \circ nA_3 \cdot > A_2 \circ nA_4$

$A_3 \cdot > A_4 \Rightarrow A_1 \circ nA_3 \cdot > A_2 \circ nA_4$

Esto es como si se tuviera:

$A_3 \cdot > A_4 \Rightarrow A_1 \circ A_3 \circ A_3 \circ ... \circ A_3 \cdot > A_2 \circ A_4 \circ ... \circ A_4$

Se verifica el valor de $D(A_3 \circ A_3)$

$D(A_3 \circ A_3) = MAX(D(A_3), D(A_3)) = D(A_3)$ sin importar cuántas veces se realice la concatenación de una regla consigo misma. Debido que D es idempotente, este axioma no se cumple.

Entonces la medida D no cumple con los axiomas de la estructura extensiva modificada.

Axiomas de las condiciones de independencia

Condición de Independencia 1. $A_1 \approx A_2 \Rightarrow A_1 \circ A \approx A_2 \circ A$ y $A_1 \approx A_2 \Rightarrow A \circ A_1 \approx A \circ A_2$

Considerando el punto 4 definido al inicio de esta sección: $A_1 \approx A_2 \Leftrightarrow D(A_1) \geqslant D(A_2)$ & $D(A_2) \geqslant D(A_1)$

De la definición 5.3 se tiene:

$A_1 \circ A \approx A_2 \circ A \Leftrightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$ & $D(A_2 \circ A) \geqslant D(A_1 \circ A)$

Consideremos $D(A_1 \circ A) \geqslant D(A_2 \circ A)$

Se puede observar que éste es el axioma de monotonicidad de la estructura extensiva modificada. Como se mostró en ese axioma, se puede agregar una regla A sin que se altere la relación entre A_1 y A_2.

Como también se cumple la propiedad de conmutatividad, también se cumple la parte $D(D_2 \circ A) \geqslant D(A_1 \circ A)$.

Como consecuencia, se cumple el axioma.

$$A_1 \circ A \approx A_2 \circ A \Leftrightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A) \,\&\, D(A_2 \circ A) \geqslant D(A_1 \circ A)$$

Condición de Independencia 2. $A_1 \approx A_2 \Leftrightarrow A_1 \circ A \approx A_2 \circ A$ y $A_1 \approx A_2 \Leftrightarrow A \circ A_1 \approx A \circ A_2$

Para la dirección $\Rightarrow$ se ha demostrado en el axioma anterior.

Para la dirección $\Leftarrow$ consideremos:

$D(A_1 \circ A) \geqslant D(A_2 \circ A)$ y $D(A_2 \circ A) \geqslant D(A_1 \circ A)$

Según la definición de concatenación 5.4

$MAX(D(A_1), D(A)) \geqslant MAX(D(A_2), D(A))$ y

$MAX(D(A_2), D(A)) \geqslant MAX(D(A_1), D(A))$

Se puede observar que si $D(A)$ es el valor máximo, aún se conservará la relación que existe entre $D(A_1)$ y $D(A_2)$. Sin embargo, esa relación puede ser mayor que, menor que ó igual. Por lo tanto, este axioma no se cumple.

Condición de Independencia 3. $A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A$ y $A_1 \cdot \geqslant A_2 \Rightarrow A \circ A_1 \cdot \geqslant A \circ A_2$

Por la definición 5.3, se tiene:

$A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$ y

$A_1 \cdot \geqslant A_2 \Rightarrow A \circ A_1 \cdot \geqslant A \circ A_2 \Leftrightarrow D(A_1) \geqslant D(A_2) \Rightarrow D(A \circ A_1) \geqslant D(A \circ A_2)$

Consideremos $D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$

En los axiomas anteriores se ha demostrado que se cumple la propiedad de monotonicidad. Por lo tanto esta parte se cumple

Consideremos ahora $D(A_1) \geqslant D(A_2) \Rightarrow D(A \circ A_1) \geqslant D(A \circ A_2)$

También se cumple la propiedad de conmutatividad.

Entonces este axioma se cumple.

$$A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A) \text{ y}$$
$$A_1 \cdot \geqslant A_2 \Rightarrow A \circ A_1 \cdot \geqslant A \circ A_2 \Leftrightarrow D(A_1) \geqslant D(A_2) \Rightarrow D(A \circ A_1) \geqslant D(A \circ A_2)$$

Condición de Independencia 4. $A_1 \cdot \geqslant A_2 \Leftrightarrow A_1 \circ A \cdot \geqslant A_2 \circ A$ y $A_1 \cdot \geqslant A_2 \Leftrightarrow A \circ A_1 \cdot \geqslant A \circ A_2$

Por la definición 5.3, se tiene:

$A_1 \cdot \geqslant A_2 \Leftrightarrow A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow D(A_1) \geqslant D(A_2) \Leftrightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$ y

$A_1 \cdot \geqslant A_2 \Leftrightarrow A \circ A_1 \cdot \geqslant A \circ A_2 \Leftrightarrow D(A_1) \geqslant D(A_2) \Leftrightarrow D(A \circ A_1) \geqslant D(A \circ A_2)$

La parte $D(A_1) \geqslant D(A_2) \Rightarrow D(A_1 \circ A) \geqslant D(A_2 \circ A)$ y $D(A_1) \geqslant D(A_2) \Rightarrow D(A \circ A_1) \geqslant D(A \circ A_2)$ se demostró en la condición de independencia anterior.

Consideremos ahora $D(A_1 \circ A) \geqslant D(A_2 \circ A)$

$D(A_1 \circ A) \geqslant D(A_2 \circ A) = MAX(D(A_1), D(A)) \geqslant MAX(D(A_2), D(A))$, puede suceder que el valor máximo sea $D(A)$ y ello no provoca que se modifique la relación que existe entre $D(A_1)$ y $D(A_2)$. Sin embargo, no podemos tener la certeza de que esta relación sea $D(A_1) \geqslant D(A_2)$. Puede darse el caso que $D(A_1)$ y $D(A_2)$ se encuentren relacionadas por $D(A_1) < D(A_2)$. Por lo tanto esta condición de independencia no se cumple.

Como conclusión, la medida D no cumple las condiciones de independencia.

Axiomas de la relación modificada de confianza

Relación Modificada de Confianza 1. $\forall A, B \in \Im : A\cdot \geqslant B$ ó $B\cdot \geqslant A$

Este axioma se cumple porque en los números reales se cumple la propiedad de completitud.

Relación Modificada de Confianza 2. $\forall A, B, C \in \Im : A\cdot \geqslant B$ y $B\cdot \geqslant C \Rightarrow A\cdot \geqslant C$

Esta propiedad también se cumple porque en los números reales existe la transitividad.

Relación Modificada de Confianza 3. $\forall A \supseteq B \Rightarrow A\cdot \geqslant B$

Hay que recordar que una regla está formada por tres partes: evento, condición y acción. Entonces la regla $A(E_1, C_1, A_1) \supseteq B(E_2, C_2, A_2)$ si todos los eventos que componen E_2 están presentes en E_1, de forma análoga para la condición C_2 y las instrucciones de la acción A_2. Por esta razón, si una regla contiene a otra, el número de inferencias necesarias (distancia) para disparar la regla contenedora será mayor o igual que la distancia de la regla contenida. De esta forma, se cumple esta relación modificada de confianza.

Relación Modificada de Confianza 4. $(\forall A \supset B, A \cap C = \varnothing) \Rightarrow (A\cdot > B \Rightarrow A \cup C\cdot > B \cup C)$

Supongamos que una regla A contiene a una regla B, entonces, $A \supset B$ y por el axioma anterior, $A\cdot > B$. Ahora si ambas reglas se combinan con una tercera, se obtiene que $D(A) > D(B) \Rightarrow D(A \circ C) > D(B \circ C)$. Si el valor de $D(C)$ es mayor que el valor de $D(A)$ y el valor de $D(B)$ entonces en ambos lados del signo $>$ se obtendrá el mismo resultado. Por lo tanto, este axioma no se cumple.

Relación Modificada de Confianza 5.2 $\forall A \in \Im : A\cdot \geqslant 0$

Esta relación modificada de confianza se cumple porque la distancia mínima de una regla es uno.

5.4.3. Potencial de Disparo

En secciones anteriores se mostraron los dos casos que se pueden tener para el parámetro potencial de disparo. La definición formal del parámetro se divide entonces en:

Caso I: El potencial de disparo $\forall A_i, A_j \in A$ cuyo evento sea primitivo ó los eventos compuestos OR y NEGACIÓN se define formalmente como un mapeo $TP : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow TP\left(A_i\right) \geqslant TP\left(A_j\right) \ \forall \ A_i, \ A_j \in A \tag{5.5}$$

Caso II: El potencial de disparo $\forall A_i, A_j \in A$ cuyo evento sea compuesto, excluyendo a los eventos compuestos OR y NEGACIÓN, se define formalmente como un mapeo $TP : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow TP\left(A_i\right) \leqslant TP\left(A_j\right) \ \forall \ A_i, \ A_j \in A \tag{5.6}$$

Como TP se define como:

$$TP = \frac{causE\left(A\right)}{cardE\left(A\right)} \tag{5.7}$$

La operación de concatenación $\circ$ se define como:

$$TP\left(A_i \circ A_j\right) = \frac{causE\left(A_i\right) + causE\left(A_j\right)}{cardE\left(A_i\right) + cardE\left(A_j\right) - cardE\left(A_i \cap A_j\right)} \tag{5.8}$$

Axiomas de la estructura extensiva modificada

En este apartado se verifica si el parámetro potencial de disparo cumple con los axiomas de la estructura extensiva modificada.

A lo largo de la verificación del parámetro de medición *potencial de disparo* se emplea la concatenación de reglas. Hay que notar que, aunque se tienen dos interpretaciones diferentes para el potencial de disparo, la concatenación entre dos reglas puede darse independientemente del tipo de evento que las active. Por otro lado, para calcular el potencial de disparo de dos reglas concatenadas, se consideran los valores de las causas y las cardinalidades no el valor del potencial de disparo, por lo tanto, se tiene que tomar el valor real de la cardinalidad del lugar de entrada de una transición. Recordemos que en el cálculo del potencial de disparo se utiliza el valor de una pseudocardinalidad cuando el lugar de entrada a una transición es virtual y su número de causas es igual al valor de su cardinalidad. Ello no implica que cambie la cardinalidad del lugar virtual sino que, simplemente, cada causa que lo origina es provocada por un lugar primitivo.

Axioma 1. Orden débil. $(A,\cdot \geqslant)$

Para ambos casos del potencial de disparo se cumplen las propiedades de transitividad y completitud porque tales propiedades se cumplen en los números reales.

Axioma 2. Positividad. $A_1 \circ A_2 \cdot \geqslant A_1$

Para verificar este axioma supongamos que se concatenan una regla activada por un evento primitivo y una regla activada por un evento compuesto. El axioma establece que:

$A_1 \circ A_2 \cdot \geqslant A_1 \Leftrightarrow TP\left(A_1 \circ A_2\right) \geqslant TP\left(A_1\right)$

De la definición 5.8 sabemos que:

$$TP\left(A_1 \circ A_2\right) = \frac{causE\left(A_1\right) + causE\left(A_2\right)}{cardE\left(A_1\right) + cardE\left(A_2\right) - cardE\left(A_1 \cap A_2\right)}$$

Como se conoce la definición de la operación de concatenación, se puede sustituir:

$$\frac{causE\left(A_1\right) + causE\left(A_2\right)}{cardE\left(A_1\right) + cardE\left(A_2\right) - cardE\left(A_1 \cap A_2\right)} \geqslant \frac{causE\left(A_1\right)}{cardE\left(A_1\right)}$$

Supongamos que las reglas A_1 y A_2 son independientes, entonces el valor de $cardE\left(A_1 \cap A_2\right) = 0$. En este caso puede suceder que la cardinalidad de los eventos de las reglas A_1 y A_2 sea mayor que sus causas lo que origina que el cociente del lado derecho del símbolo $\geqslant$ disminuya. Incluso puede llegar a ser menor que el cociente del lado derecho del símbolo $\geqslant$.

Por ejemplo, supongamos los valores $TP\left(A_2\right) = \frac{1}{2}$, $TP\left(A_1\right) = \frac{2}{2}$ y $cardE\left(A_1 \cap A_2\right) = 0$. Estos valores se pueden obtener cuando se concatena una regla cuyo evento es compuesto con una regla cuyo evento es primitivo

Las concatenamos y obtenemos:

$TP\left(A_1 \circ A_2\right) = \frac{2+1}{2+2} = \frac{3}{4}$ lo que no cumple con $\frac{3}{4} \geqslant 1$

Ahora supongamos que $TP\left(A_1\right) = \frac{2}{3}$, $TP\left(A_2\right) = \frac{1}{2}$ y el valor de $cardE\left(A_1 \cap A_2\right) = 2$. Entonces, cuando realizamos la concatenación:

$$\frac{causE\left(A_1\right) + causE\left(A_2\right)}{cardE\left(A_1\right) + cardE\left(A_2\right) - cardE\left(A_1 \cap A_2\right)} \leqslant \frac{causE\left(A_1\right)}{cardE\left(A_1\right)}$$

$$\frac{2+1}{3+2-2} = \frac{3}{3} \leqslant \frac{2}{3}$$

Como $\frac{3}{3}$ no es menor que $\frac{2}{3}$ este axioma no se cumple.

Axioma 3. Asociatividad débil. $A_1 \circ \left(A_2 \circ A_3\right) \approx \left(A_1 \circ A_2\right) \circ A_3$

Del punto 4 definido al inicio de la sección, sabemos:

$A_1 \approx A_2 \Leftrightarrow A_1 \cdot \geqslant A_2$ y $A_2 \cdot \geqslant A_1$

$A_1 \cdot \geqslant A_2 \Leftrightarrow TP(A_1) \geqslant TP(A_2)$ y $A_2 \cdot \geqslant A_1 \Leftrightarrow TP(A_2) \geqslant TP(A_1)$

Consideremos $A_2 \circ A_3$

$$TP(A_2 \circ A_3) = \frac{causE(A_2) + causE(A_3)}{cardE(A_2) + cardE(A_3) - cardE(A_2 \cap A_3)}$$

Al hacer la concatenación con una tercera regla A_1 se obtiene:

$$\frac{causE(A_2) + causE(A_3) + causE(A_1)}{cardE(A_2) + cardE(A_3) + cardE(A_1) - cardE(A_2 \cap A_3 \cap A_1)}$$

Del otro lado, al hacer la concatenación de $TP(A_1 \circ A_2)$ se obtiene:

$$\frac{causE(A_1) + causE(A_2)}{cardE(A_1) + cardE(A_2) - cardE(A_1 \cap A_2)}$$

Cuando concatenamos con una tercera regla A_3 se obtiene:

$$\frac{causE(A_1) + causE(A_2) + causE(A_3)}{cardE(A_1) + cardE(A_2) + cardE(A_3) - cardE(A_1 \cap A_2 \cap A_3)}$$

Por lo que:

$$\begin{aligned} & \frac{causE(A_2) + causE(A_3) + causE(A_1)}{cardE(A_2) + cardE(A_3) + cardE(A_1) - cardE(A_2 \cap A_3 \cap A_1)} \\ = \quad & \frac{causE(A_1) + causE(A_2) + causE(A_3)}{cardE(A_1) + cardE(A_2) + cardE(A_3) - cardE(A_1 \cap A_2 \cap A_3)} \end{aligned}$$

Este axioma se cumple porque las operaciones involucradas en el TP son conmutativas.

$$A_1 \circ (A_2 \circ A_3) \approx (A_1 \circ A_2) \circ A_3 \Leftrightarrow TP(A_1 \circ (A_2 \circ A_3)) = TP((A_1 \circ A_2) \circ A_3)$$

Axioma 4. Conmutatividad débil. $A_1 \circ A_2 \approx A_2 \circ A_1$

De la definición de concatenación 5.8 se tiene:

$$TP(A_1 \circ A_2) = \frac{causE(A_1) + causE(A_2)}{cardE(A_1) + cardE(A_2) - cardE(A_1 \cap A_2)}$$

$$TP(A_2 \circ A_1) = \frac{causE(A_2) + causE(A_1)}{cardE(A_2) + cardE(A_1) - cardE(A_2 \cap A_1)}$$

Reacomodando los términos del numerador y denominador, se obtiene:

$$\frac{causE\left(A_1\right)+causE\left(A_2\right)}{cardE\left(A_1\right)+cardE\left(A_2\right)-cardE\left(A_1\cap A_2\right)}\geqslant\frac{causE\left(A_1\right)+causE\left(A_2\right)}{cardE\left(A_1\right)+cardE\left(A_2\right)-cardE\left(A_2\cap A_1\right)}$$

Como el valor de $cardE\left(A_1\cap A_2\right)=cardE\left(A_2\cap A_1\right)$, se obtiene el mismo valor en ambos lados del símbolo $\geqslant$.

Por lo tanto,

$$\frac{causE\left(A_1\right)+causE\left(A_2\right)}{cardE\left(A_1\right)+cardE\left(A_2\right)-cardE\left(A_1\cap A_2\right)}=\frac{causE\left(A_1\right)+causE\left(A_2\right)}{cardE\left(A_1\right)+cardE\left(A_2\right)-cardE\left(A_2\cap A_1\right)}$$

Para el caso II es una situación similar.

Finalmente,

$$A_1\circ A_2\approx A_2\circ A_1\Leftrightarrow TP\left(A_1\circ A_2\right)=TP\left(A_2\circ A_1\right)\ \&\ TP\left(A_2\circ A_1\right)=TP\left(A_1\circ A_2\right)$$

Axioma 5. Monotonicidad débil. $A_1\cdot\geqslant A_2\Rightarrow A_1\circ A\cdot\geqslant A_2\circ A$

Este axioma no se cumple porque se puede tener una situación como la siguiente:

$TP\left(A_1\right)=\frac{2}{2},TP\left(A_2\right)=\frac{1}{1},TP\left(A\right)=\frac{1}{2}$, $cardE\left(A_1\cap A\right)=0,\ cardE\left(A_2\cap A\right)=1.$

Entonces:

$TP\left(A_1\circ A\right)=\frac{2+1}{2+2-0}=\frac{3}{4}$

$TP\left(A_2\circ A\right)=\frac{1+1}{1+2-1}=\frac{2}{2}=1$

Entonces, $TP\left(A_1\right)\geqslant TP\left(A_2\right)\Longrightarrow TP\left(A_1\circ A\right)\geqslant TP\left(A_2\circ A\right)$ no se cumple.

Para el caso II podemos tener una situación como la siguiente:

$TP\left(A_1\right)=\frac{1}{2},\ TP\left(A_2\right)=\frac{1}{3},\ TP\left(A\right)=\frac{1}{2},\ card\left(A_1\circ A\right)=2$ y $card\left(A_2\circ A\right)=0$

Entonces:

$$TP\left(A_1\circ A\right)=\frac{1+1}{2+2-2}=\frac{2}{2}$$

$$TP\left(A_2\circ A\right)=\frac{1+1}{2+3}=\frac{2}{5}$$

Por lo que:

$TP(A_1) \geqslant TP(A_2) \Rightarrow TP(A_1 \circ A) \leqslant TP(A_2 \circ A)$ no se cumple.

Axioma 6. Axioma de Arquímedes. Si $A_3 \cdot > A_4$ entonces para cualquier A_1, A_2 existe un número natual n, tal que $A_1 \circ nA_3 \cdot > A_2 \circ nA_4$

$A_3 \cdot > A_4 \Rightarrow A_1 \circ nA_3 \cdot > A_2 \circ nA_4$

Supongamos, por ejemplo, que tenemos $TP(A_3) = \frac{1}{1}$, $TP(A_4) = \frac{1}{2}$; $TP(A_1) = \frac{1}{1}$; $TP(A_2) = \frac{1}{1}$; $cardE(A_1 \cap A_2) = 0$ y $cardE(A_2 \cap A_4) = 1$

En este caso, $TP(A_3) > TP(A_4)$ pero:

$$TP(A_1 \circ A_2) = \frac{1 + n * 1}{1 + n * 1} = \frac{n+1}{n+1}$$

y

$$TP(A_2 \circ A_4) = \frac{1 + 1 + *n}{1 + 2 * (n-1) * n} = \frac{n+1}{n+1}$$

De forma que $TP(A_1 \circ A_2) = TP(A_2 \circ A_4)$ y este axioma no se cumple.

Entonces TP no es una estructura extensiva.

Axiomas de las condiciones de independencia

Condición de independencia 1. $A_1 \approx A_2 \Rightarrow A_1 \circ A \approx A_2 \circ A$ y $A_1 \approx A_2 \Rightarrow A \circ A_1 \approx A \circ A_2$

En esta condición de independencia se involucran el axioma de monotonicidad y el axioma de conmutatividad de la estructura extensiva modificada. Como se mostró en esos axiomas, el parámetro de medición potencial de disparo cumple el axioma de conmutatividad pero no cumple el axioma de monotonicidad. Por lo tanto, esta condición de independencia no se cumple.

Condición de independencia 2. $A_1 \approx A_2 \Leftrightarrow A_1 \circ A \approx A_2 \circ A$ y $A_1 \approx A_2 \Leftrightarrow A \circ A_1 \approx A \circ A_2$

La dirección $\Rightarrow$ se verificó en la condición de independencia 1.

Para la dirección $\Leftarrow$ tenemos:

$A_1 \circ A \approx A_2 \circ A \Leftrightarrow TP(A_1 \circ A) \geqslant TP(A_2 \circ A)$ & $TP(A_2 \circ A) \geqslant TP(A_1 \circ A)$

Consideremos $TP(A_1 \circ A) \geqslant TP(A_2 \circ A)$

Utilizando la definición 5.8 tenemos:

$$\frac{causE(A_1) + causE(A)}{cardE(A_1) + cardE(A) - cardE(A_1 \cap A)} \geqslant \frac{causE(A_2) + causE(A)}{cardE(A_2) + cardE(A) - cardE(A_2 \cap A)}$$

Esta relación se puede cumplir aún cuando $TP(A_1) < TP(A_2)$. Por ejemplo, considerando los valores:

$TP(A_1) = \frac{3}{1},\ TP(A_2) = \frac{4}{1},\ TP(A) = \frac{1}{2},\ cardE(A_1 \cap A) = 1$ y $TP(A_2 \cap A) = 0$

$\frac{3+1}{1+2-1} \geqslant \frac{4+1}{1+2} = \frac{4}{2} \geqslant \frac{5}{3} = 2 \geqslant 1{,}6$ pero $TP(A_1) = \frac{3}{1}$ es menor que $TP(A_2) = \frac{4}{1}$

Por lo tanto, esta condición de independencia no se cumple.

Se puede presentar una situación similar para el caso II.

Condición de independencia 3. $A_1 \cdot \geqslant A_2 \Rightarrow A_1 \circ A \cdot \geqslant A_2 \circ A$ y $A_1 \cdot \geqslant A_2 \Rightarrow A \circ A_1 \cdot \geqslant A \circ A_2$

Esta condición de independencia no se cumple porque no se cumple el axioma de monotonicidad de la estructura extensiva modificada.

Condición de independencia 4. $A_1 \cdot \geqslant A_2 \Leftrightarrow A_1 \circ A \cdot \geqslant A_2 \circ A$ y $A_1 \cdot \geqslant A_2 \Leftrightarrow A \circ A_1 \cdot \geqslant A \circ A_2$

La dirección $\Rightarrow$ se verificó en la condición de independencia 3.

Para la dirección $\Leftarrow$ tenemos:

$A_1 \circ A \cdot \geqslant A_2 \circ A \Leftrightarrow TP(A_1 \circ A) \geqslant TP(A_2 \circ A)$

Consideremos $TP(A_1 \circ A) \geqslant TP(A_2 \circ A)$

Utilizando la definición 5.8 tenemos:

$$\frac{causE(A_1) + causE(A)}{cardE(A_1) + cardE(A_2) - cardE(A_1 \cap A)} \geqslant \frac{causE(A_2) + causE(A)}{cardE(A_2) + cardE(A) - cardE(A_2 \cap A)}$$

Supongamos que tenemos los valores siguientes:

$TP(A_1) = \frac{2}{1}, TP(A_2) = \frac{3}{1}, TP(A) = \frac{1}{2}, cardE(A_1 \cap A) = 1$ y $cardE(A_2 \cap A) = 0$

Entonces obtenemos:

$$\frac{2+1}{1+2-1} \geqslant \frac{3+1}{1+2} = \frac{3}{2} \geqslant \frac{3}{3}$$

pero $TP(A_1) < TP(A_2)$

Por lo tanto esta condición de independencia no se cumple.

El parámetro de medición potencial de disparo no cumple las condiciones de independencia.

Axiomas de la relación modificada de confianza

Relación modificada de confianza 1. $\forall A, B \in \Im : A \cdot \geqslant B$ ó $B \cdot \geqslant A$

En ambos casos del potencial de disparo se cumple esta propiedad porque en los números reales se cumple la propiedad de completitud.

Relación modificada de confianza 2. $\forall A, B, C \in \Im : A\cdot \geqslant B$ y $B\cdot \geqslant C \Rightarrow A\cdot \geqslant C$

La propiedad de transitividad se cumple para ambos casos del potencial de disparo porque en los números reales también se cumple.

Relación modificada de confianza 3. $\forall A \supseteq B \Rightarrow A\cdot \geqslant B$

Para el caso I.

Sabemos que $A\cdot \geqslant B \Leftrightarrow TP(A) \geqslant TP(B)$

Consideremos $TP(A) \geqslant TP(B)$. Entonces

$$\frac{causE(A_1)}{cardE(A_1)} \geqslant \frac{causE(A_2)}{cardE(A_2)}$$

En este caso, se cumple la relación porque la cardinalidad es uno ó es igual que las causas. Entonces en ambos lados del símbolo $\geqslant$ se obtiene el mismo resultado.

Para el caso II.

Sabemos que $A\cdot \geqslant B \Leftrightarrow TP(A) \leqslant TP(B)$. Entonces:

$$\frac{causE(A)}{cardE(A)} \geqslant \frac{causE(B)}{cardE(B)}$$

Puede suceder que los eventos primitivos involucrados en el evento de la regla B sean un subconjunto de los eventos primitivos que forman el evento compuesto de la regla A, de esta forma, $cardE(A) \geqslant cardE(B)$. Sin embargo, la regla A y la regla B pueden tener las mismas causas. Por ejemplo, supongamos que $TP(A)$ = $\frac{1}{3}$ y $TP(B) = \frac{1}{2}$. Se puede ver que $\frac{1}{3} < \frac{1}{2}$. Entonces, de forma general, esta relación modificada de confianza no se cumple.

Relación modificada de confianza 4. $(\forall A \supset B, A \cap C = \varnothing) \Rightarrow (A\cdot \geqslant B \Rightarrow A \cup C\cdot > B \cup C)$

Esta relación modificada de confianza no se cumple porque se puede presentar una situación similar a la descrita en la relación modificada de confianza 3. Es decir, que aún cuando una regla A contiene a una regla B, el potencial de disparo de A puede ser menor que el potencial de disparo de B.

Relación Modificada de Confianza 5.3 $\forall A \in \Im : A\cdot \geqslant 0$

Esto es cierto porque $causE(A_1) > 0$ y $cardE(A_1) > 0$

Los parámetros no cumplen todos los axiomas, no obstante, se pueden clasificar sobre el nivel de escala ordinal porque todos ellos cumplen las propiedades de transitividad, completitud y orden.

5.5. Comentarios Finales

En este trabajo de tesis, adoptamos las métricas número de anclas, distancia y potencial de disparo propuestas en [10]. Ya que en en ese trabajo los autores únicamente consideran eventos primitivos, extendimos la definición de los parámetros para considerar a los eventos compuestos. Al hacer esta modificación a la definición de los parámetros necesitamos un modelo más completo que la gráfica de disparo para representar a las reglas ECA. El modelo que elegimos fue el de la CCPN porque proporciona más información acerca de la regla y cuenta con una representación gráfica y matemática, lo cual permite verificar la consistencia de nuestras mediciones.

Sobre la estructura gráfica de la CCPN modelamos los parámetros de medición y obtuvimos su definición formal. En el caso particular del parámetro potencial de disparo, obtuvimos una definición en dos casos para abarcar a eventos primitivos y eventos compuestos.

Cada una de las definiciones de los parámetros las utilizamos para realizar la validación de los mismos sobre el marco de trabajo desarrollado por el Dr. Zuse. Este marco de trabajo consta de un conjunto de axiomas que dan las condiciones para la traducción de enunciados empíricos a enunciados formales y se apoya fuertemente sobre la teoría de la medición clásica. El resultado que obtuvimos con la verificación de los parámetros es que cumplen con las propiedades de transitividad, completitud y orden, lo cual permite ubicarlos en el nivel de escala ordinal.

Por otro lado, ya que la CCPN cuenta con una representación matemática, dada por la matriz de incidencia, aprovechamos ésta para desarrollar algoritmos a fin de calcular el valor de los parámetros automáticamente.

Capítulo 6

Implementación

En el capítulo IV se presentaron los parámetros de medición número de anclas, distancia y potencial de disparo para determinar la complejidad de la interacción de las reglas ECA en una BDA. Tomando en cuentra el valor de tales parámetros se puede modificar, si es necesario, el diseño de las reglas, de forma que se obtenga un balance entre la complejidad de su interacción y la funcionalidad que deben cumplir. Asimismo, los parámetros se modelaron sobre la estructura gráfica y matemática de la CCPN. Aprovechando las ventajas que ofrece la estructura matemática, se desarrollaron algoritmos para obtener el valor de los parámetros de manera sencilla. En este capítulo se presenta la implementación de los algoritmos, además de la integración que se realizó con la interfaz ECAPNSim.

Para mostrar el desarrollo de nuestro sistema de software, al que hemos denominado Módulo de Complejidad, este capítulo se divide en las siguientes secciones: la sección 6.1 muestra el diseño del sistema desarrollado; la sección 6.2 presenta la implementación de los algoritmos; en la sección 6.3 se expone el uso del Módulo de Complejidad; en la sección 6.4 se exhiben los comentarios finales.

6.1. Diseño del sistema

En esta sección se describe el diseño del Módulo de Complejidad. Se presenta una descripción del mismo, así como la arquitectura y el diagrama de clases.

6.1.1. Descripción del sistema

El Módulo de Complejidad que se desarrolló en esta tesis ofrece un medio para determinar la complejidad de la interacción de las reglas ECA en una BDA. Para ello se utilizó la estructura matemática de la CCPN y los parámetros de medición número de anclas, distancia y potencial de disparo. El funcionamiento del Módulo de Complejidad se puede describir en los siguientes pasos: primero, se lee el archivo que contiene la matriz de incidencia de la CCPN que representa a las reglas ECA; segundo, con esta información se calculan todas las rutas que, desde un lugar inicial, llegan hasta una transición $t_j \in T_{rule}$; tercero, a partir de las rutas calculadas, se obtiene el valor de los parámetros número de anclas y distancia; cuarto, para el parámetro potencial de disparo, se calcula el valor de sus causas y su cardinalidad y se obtiene el valor del parámetro; y quinto, se muestran los resultados al usuario.

A través de la interpretación de los resultados, se pueden identificar aquellas reglas que presentan una interacción significativa con algunas otras. La identificación de estas reglas permite centrar la atención en el mejoramiento de reglas particulares. Si se mejora el diseño de estas reglas, como consecuencia, se mejora el diseño y el mantenimiento de todo el conjunto de reglas. Este procedimiento se puede repetir hasta que se obtenga un equilibrio entre la complejidad de la interacción de las reglas y su funcionamiento.

En el capítulo IV, después de hacer la verificación sobre el marco de trabajo del Dr. Zuse se concluyó que los parámetros se pueden medir sobre la escala ordinal. Una de las operaciones estadísticas que se pueden realizar sobre la escala ordinal es la mediana. Por lo que, además de realizar el cálculo de los parámetros de medición, el Módulo de Complejidad brinda la opción de obtener la mediana de cada uno de los parámetros.

Para la implementación de los algoritmos desarrollados en el capítulo IV se empleó el lenguaje de programación Java. Esto se hizo para mantener la compatibilidad con la interfaz ECAPNSim, ya que el Módulo de Complejidad se integró a esta interfaz. Con la integración del Módulo de Complejidad a ECAPNSim se obtuvo una herramienta más robusta. No obstante, el Módulo de Complejidad también puede funcionar de manera independiente a ECAPNSim especificando el nombre del archivo donde se almacena la matriz de incidencia.

6.1.2. Arquitectura del sistema

La arquitectura del Módulo de Complejidad se muestra en la figura 6.1. El sistema se encuentra dividido en diferentes módulos los cuales cumplen una función específica. Algunos de estos módulos se agrupan en subsistemas para realizar el cálculo de los parámetros así como de la mediana y la conversión de la CCPN

en una gráfica de disparo modificada. Algunos otros módulos son comunes a los subsistemas, tales módulos son: Identificador de lugares iniciales, Identificador de rutas, Clasificador de transiciones, Clasificador de lugares e Identificador de lugares de entrada y salida de una transición.

El subsistema denominado Número de Anclas necesita como entrada a los módulos Identificador de lugares iniciales e Identificador de rutas. Dentro del subsistema Número de Anclas, a partir de las rutas para una transición se identifican los lugares iniciales y se obtiene el número total para cada transición. Finalmente, el resultado se presenta en una interfaz gráfica que contiene el número de la transición, su número de lugares iniciales y el conjunto de lugares iniciales para esa transición. De esta forma, el usuario puede conocer cuántos y cuáles son los lugares inciales para cada transición y se le evita el proceso de identificar las rutas y localizar los lugares iniciales.

En el subsistema Distancia, las entradas requeridas se toman de los módulos Generador de matriz de incidencia e Identificador de rutas. Para todas las rutas de una transición se calcula su longitud utilizando el valor de los arcos que conectan transiciones con lugares. Este valor de los arcos lo proporciona la matriz de incidencia. Ya que se tienen los valores de las longitudes, se calcula la máxima, es decir, el valor de la distancia. Por último, se presentan los resultados en una interfaz gráfica.

Los módulos Clasificador de transiciones, Clasificador de lugares e Identificador de lugares de entrada y salida son las entradas al subsistema Potencial de Disparo. En este subsistema, según la definición que se dió del parámetro del mismo nombre, es necesario calcular previamente las causas y la cardinalidad de la transición. Para realizar esta función es necesario identificar los lugares de entrada de las transiciones así como su tipo. Una vez obtenidos estos valores, se calcula el cociente de las causas entre la cardinalidad y se presenta el resultado.

Para calcular el valor de la mediana se toman las salidas de los subsistemas que calculan los parámetros, se ordenan de manera creciente y se calcula la mediana. Finalmente, se presentan los resultados al usuario.

Adicionalmente, se le muestran al usuario todas las rutas para todas las transiciones $t_j \in T_{rule}$. Para el usuario el contar con esta opción es una ventaja porque se le evita el trabajo de estar buscando manualmente las rutas en la representación gráfica de la CCPN. Otra característica adicional del Módulo de Complejidad es la conversión de la CCPN en una gráfica de disparo modificada. Esta opción permite visualizar el conjunto de reglas ECA en una representación alternativa, con las restricciones que tiene esta representación.

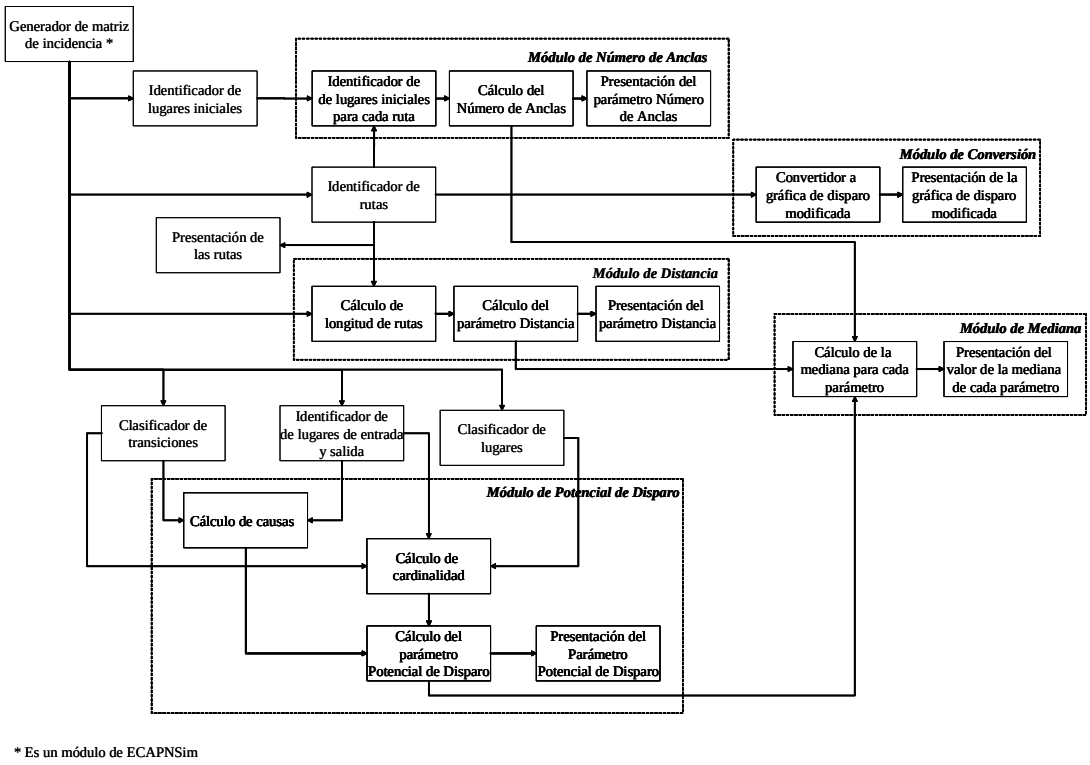

Figura 6.1: Arquitectura del Módulo de Complejidad

6.1.3. Diagrama de clases

Algunos sistemas se construyen a partir de clases y objetos, cuya colaboración permite lograr el comportamiento deseado en el sistema. Los diagramas de clases muestran la estructura de un sistema en términos de clases y objetos, incluyendo la forma en que los objetos y clases se relacionan. La colaboración entre objetos impone la existencia de relaciones entre ellos, una de esas relaciones es la agregación. La agregación es una asociación especializada en la cual un todo se relaciona con sus partes. También se suele denominar como la relación “parte de”.

El concepto de herencia define otro tipo de relación entre clases, donde una clase comparte estructura y/o comportamiento con una o más clases. El término superclase se refiere a la clase que guarda la información común, mientras que el término subclase se refiere a cada uno de los descendientes de la superclase. Una subclase hereda todos los atributos, operaciones y relaciones definidos en alguna de sus superclases, de forma que los atributos y operaciones comunes se definen en el nivel más alto de la jerarquía. Las subclases se pueden ampliar (especializar) con atributos y operaciones adicionales, que se aplican sólo a ese nivel de la jerarquía. Además una subclase puede tener su propia implementación de una operación de la superclase.

La figura 6.2 muestra el diagrama de clases del Módulo de Complejidad. En él se puede observar que la superclase es Object. Las clases que son descendientes directas de Object son la clase General y la clase JPanel. La clase General contiene métodos y atributos que son comunes a las clases Rutas, PotDisparo y Mediana. A su vez, las clases NumAnclas y Distancia heredan de la clase Rutas ya que ambas clases necesitan las rutas que se generan en la clase Rutas para obtener sus propios resultados. La clase Mediana, además de heredad de la clase General, se relaciona mediante una agregación con las clases NumAnclas, Distancia y PotDisparo ya que necesita de los resultados generados por esas clases para obtener sus propios resultados. Las clases cuyo nombre inicia con Vistas heredan los métodos y atributos de la clase JPanel. Esto les permite manipular los componentes de dibujo para presentar los datos al usuario. La clase VistaRutas se relaciona mediante una agregación con la clase Rutas, para mostrar en la pantalla el resultado generado por la clase Rutas. Lo mismo sucede con la clase VistaNumAnclas y la clase NumAnclas, la clase VistaDistancia y la clase Distancia, la clase VistaPotDisparo y la clase PotDisparo y, finalmente, la clase VistaMediana y la clase Mediana.

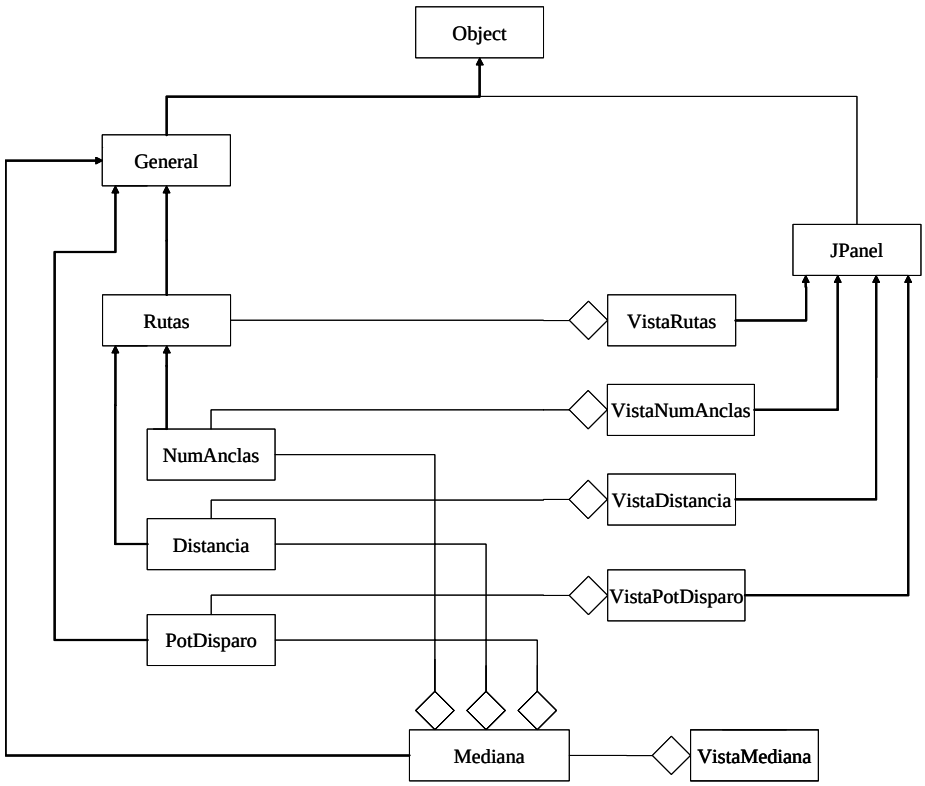

Figura 6.2: Diagrama de clases del Módulo de Complejidad

6.2. Implementación del sistema

El Módulo de Complejidad se desarrolló tratando de seguir el paradigma Modelo - Vista - Controlador (MVC). Este paradigma es un patrón de diseño aportado originariamente por el lenguaje SmallTalk a la Ingeniería del Software. El paradigma MVC consiste en dividir las aplicaciones en tres partes: el modelo, la vista y el controlador.

El modelo es el objeto de la aplicación. La vista es la presentación en pantalla. El controlador define la manera en que la interfaz de usuario reacciona a la entrada del usuario.

En el Módulo de Complejidad, el modelo lo conforman las clases General, Rutas, NumAnclas, Distancia, PotDisparo, Mediana y Conversión. Todas estas clases pueden funcionar de forma independiente y su salida se muestra en la consola de usuario. La vista está compuesta por las clases VistaRutas, VistaNumAnclas, VistaDistancia, VistaPotDisparo, VistaMediana y VistaConversión. El control se establece en el menú de ECAPNSim.

A continuación se muestran todas las clases que forman el Módulo de Complejidad, asimismo, se presenta una breve descripción de los métodos que tiene cada una de ellas.

1. La Clase General

Esta clase contiene atributos y métodos comunes a todas las demás clases. Se implementó para evitar replicar cierto código en otras clases. La función principal de la clase General es leer el archivo donde se guarda la matriz de incidencia. Una vez que se lee el archivo, esta clase cuenta con métodos que le permiten hacer una clasificación de los tipos de lugares y transiciones, encontrar los lugares iniciales, las entradas y salidas de cada transición, entre otras cosas. A continuación se listan los métodos principales de la clase:

public Vector TransFinales( ): Este método obtiene la primer transición de la ruta de transiciones.

public Vector buscaEntradasT( ): Con este método se obtienen los lugares de entrada de todas las transiciones T_{rule}. La manera en que se hace es que para todos los renglones de la matriz de incidencia que representan las transiciones T_{rule} se van recorriendo las columnas, cuando se encuentra un valor positivo, el índice de la columna se va almacenando y de esta forma se van obteniendo una lista de los lugares de entrada.

public Vector encuentraLugaresIniciales( ): Este método encuentra los lugares iniciales a partir de la matriz de incidencia. Para cada columna de la matriz se van revisando todos los renglones y si al final de la revisión existe un único valor positivo y el resto de los valores es cero, significa que es un lugar inicial. El valor del índice de la columna con tales características se va almacenando en un Vector. Este método es

muy utilizado por otras clases, por ejemplo, la clase PotDisparo lo utiliza para contabilizar las causas de sus lugares de entrada.

public Vector encuentraSalidasL( ): Este método encuentra los lugares de salida de las transiciones T_{rule} a partir de la matriz de incidencia.

public void leerMatriz( ): Mediante este método se lee el archivo que contiene la matriz de incidencia de la CCPN que se está modelando.

public boolean perteneceA( ): Este método verifica si un lugar pertenece a un cierto tipo. Regresa un valor verdadero si el lugar sí pertenece al tipo por el que se está preguntando, es decir, si el lugar l_i pertenece a los lugares copy, regresa verdadero.

public void separaPorTiposLugar( ): Este método clasifica a los lugares de la CCPN según su tipo. Los tipos posibles de los lugares son: primitivo, compuesto, virtual y copy.

public void separaPorTiposTrans( ): Este método clasifica a las transiciones de la CCPN según su tipo. Los tipos posibles de las transiciones son: ruel, copy y compuesta.

public boolean tieneLugaresIniciales( ): Con este método se verifica si una transición tiene lugares iniciales. Es particularment útil para la identificación de rutas.

2. La Clase Rutas

Esta clase permite generar todas las rutas que conectan un lugar inicial con una transición. Para lograrlo se auxilia de los siguientes métodos:

public void analizaLasNDef( ): Este método analiza las rutas para las cuales no se ha llegado a un lugar inicial.

public void calculaDefyNDef( ): Con este método se encuentran las rutas definitivas y las no definitivas. Una ruta es definitiva cuando una transición está conectada directamente, mediante un arco de entrada, a un lugar inicial. Este tipo de rutas se identifican en un solo paso. Las rutas no definitivas son aquellas en las que se requiere más de un paso para encontrarlas.

public Vector concatenarse( ): Conforme se van encontrando nuevos elementos de una ruta es necesario concatenar estos nuevos elementos con los que ya se habían calculado. Este método permite realizar esta función.

public Vector getRutas( ): Con este método se pueden obtener las rutas calculadas.

public void llenaBien( ): Este método es auxiliar en la generación de rutas. Se utiliza para ir colocando un indicador que muestra si ya se han encontrado todas las rutas para una transición. Cuando se han encontrado

todas las rutas para una transición, se coloca el identificador “si”, de otra forma, se coloca el identificador “no”.

public void llenaDef( ): Con este método se inicializa el vector que contendrá las rutas definitivas.

public Vector rutas( ): Este método encuentra todas las rutas para todas las transiciones T_{rule}.

public void setRutas( ): Con este método se colocan las rutas calculadas en una variable de forma que puedan ser accedidas por otra clase.

3. La Clase NumAnclas

Mediante la clase NumAnclas es posible calcular el parámetro número de anclas de todas las transiciones T_{rule}. Se utiliza el método getRutas( ) de la clase Rutas para recuperar las rutas encontradas mediante el método rutas( ) de la misma clase. Los métodos disponibles en la clase NumAnclas se listan enseguida.

public Vector ConjuntoAnclas( ): Este método encuentra el conjunto de anclas para las transiciones T_{rule}. Utiliza las rutas previamente calculadas.

public Vector getCA( ): Con este método se recupera el conjunto de anclas para cada una de las transiciones T_{rule}. Es necesario para poder mostrarle al usuario cuáles son los lugares iniciales de cada transición.

public void numeroAnclas( ): Este método realiza el conteo del número de anclas que tiene cada transición, es decir, obtiene la cardinalidad del conjunto de anclas para cada transición T_{rule}.

public Vector getNA( ): Este método permite recuperar el número de anclas para cada transición T_{rule}.

public void setCA( ): Con este método se coloca el conjunto de anclas para las transiciones T_{rule} en una variable, de forma que sean accesibles para otras clases.

public void setNA( ): Con este método se coloca el número de anclas para cada transición T_{rule} en una variable para que todas las demás clases puedan recuperar dicho valor.

4. La Clase Distancia

La clase Distancia permite calcular el valor del parámetro del mismo nombre para cada transición T_{rule}. Cuenta con los métodos siguientes:

public void Distancias( ): Este método encuentra la longitud de cada ruta para una transición.

public Vector MaxD( ): Este método obtiene la longitud máxima de todas las longitudes del conjunto de rutas para una transición.

public void distancia( ): En este método se invocan los métodos anteriores para encontrar el valor del parámetro distancia para todas las transiciones T_{rule}.

public Vector quitaRutasRepetidas( ): Como su nombre lo indica, este método elimina las rutas dupli-

cadas en un conjunto de rutas para una transición.

public Vector quitaTCopy( ): Este método elimina de las rutas las de transiciones aquellas transiciones que sean del tipo T_{copy}.

public void setDistancia( ): Este método coloca el valor del parámetro distancia para que pueda ser usado por otras clases.

public Vector getDistancia( ): Este método permite a otras clases recuperar el valor del parámetro distancia.

5. La Clase PotDisparo

La clase PotDisparo permite calcular el valor del parámetro potencial de disparo para cada transición T_{rule}. Esta clase cuenta con los siguientes métodos:

public Vector TP( ): Este método calcula el cociente entre las causas y la cardinalidad de una transición.

public Vector cardinalidad( ): Este método calcula la cardinalidad de las transiciones T_{rule}. Para ello necesita conocer el tipo de lugar de entrada de una transición, por lo tanto, hace uso del método separaPorTiposLugar( ) de la clase General.

public Vector causas( ): Con este método se pueden calcular las causas del lugar de entrada de una transición T_{rule}.

public void potencialDisparo( ): Este método invoca los métodos anteriores para cada una de las transiciones T_{rule} de la CCPN.

public int verificaLugar( ): Con este método se verifica el tipo del lugar de entrada de una transición.

public void setPotencialDisparo( ): Mediante este método se coloca el valor del potencial de disparo para todas las transiciones T_{rule} en una variable que puede ser leída por otras clases.

public Vector getPotencialDisparo( ): Este método permite recuperar el valor del potencial de disparo para las transiciones T_{rule} de la CCPN.

5. La Clase Mediana

Esta clase calcula el valor de la mediana para cada parámetro de medición. Al interpretar este resultado se puede saber qué reglas tienen una complejidad mayor. Cuenta con los siguientes métodos:

public double calculaMediana( ): Este método calcula el valor de la mediana. Para calcular dicho valor, los elementos deben estar ordenados de forma creciente. Si el total de elementos es impar, el valor de la mediana será el elemento que se encuentre en la posición $\frac{n+1}{2}$ donde n es el total de elementos. Por ejemplo, si se tienen los siguientes elementos ordenados: $5, 7, 9, 45, 67$; el total de elementos es $n = 5$ y el valor de la

mediana es 9 porque es el elemento que se encuentra en la posición $\frac{5+1}{2} = 3$. Si el número de elementos es par, entonces el valor de la mediana es el promedio de los dos elementos que se encuentren en las posiciones $\frac{n}{2}$ y $\frac{n}{2} + 1$, donde n es el total de elementos. Por ejemplo, si se tienen los elementos: $5, 7, 9, 45, 67, 89$ el total de elementos es $n = 6$, por lo tanto, el valor de la mediana será el promedio de los elementos en las posiciones $\frac{6}{2} = 3$ y $\frac{6}{2} + 1 = 4$, es decir, $\frac{9+45}{2} = \frac{54}{2} = 27$.

public void medianaDistancia( ): Este método calcula el valor de la mediana para el parámetro distancia.

public void medianaNumAnclas( ): Este método calcula el valor de la mediana para el parámetro número de anclas.

public void medianaPotDisparo( ): Este método calcula el valor de la mediana para el parámetro potencial de disparo.

public float [] ordenaCreciente( ): Este método ordena de forma creciente los valores obtenidos en cada parámetro.

public void setMediana( ): Este método coloca el valor de la mediana en una variable para que pueda ser accedado por otros valores.

public Vector getMediana( ): Este método recupera el valor de la mediana.

6. La Clase Conversión

Esta clase permite realizar una conversión de la CCPN a la gráfica de disparo modificada propuesta en [10]. Consta de los siguientes métodos.

public Vector quitaTComp( ): Este método elimina las transiciones T_{rule} que contengan en su ruta transiciones T_{comp}. Esto se hace porque la gráfica de disparo modificada no puede representar eventos compuestos.

public Vector separaEnPares( ): Este método toma cada ruta y forma pares con los elementos para que, posteriormente, se presenten en la pantalla.

public void clasifica( ): Este método se encarga de separar las respectivas entradas de cada transición.

public void conversion( ): Dentro de este método se invoca a los métodos anteriores.

7. La Clase VistaRutas

Esta clase presenta al usuario las rutas que se generaron para cada transición. Cuenta con los siguientes métodos:

public void generaTexto( ): Este método propociona formato a los datos que se van a presentar al usuario.

public Vector invierte( ): Este método invierte el orden de las rutas. Esto se hace porque en el proceso de identificación de rutas, se inicia desde una transición hasta encontrar sus lugares iniciales.

public void marco( ): Este método divide la pantalla donde se muestran los resultados para presentar el número de regla, la transición que representa a esa regla y las rutas.

public void muestraRutas( ): Este método muestra todas las rutas para cada una de las transiciones T_{rule} de la CCPN.

public void presenta( ): Este método invoca a los métodos anteriores para presentar la información completa al usuario acerca de las rutas.

8. La Clase VistaNumAnclas

Esta clase muestra al usuario la información acerca del número de anclas de cada transición T_{rule} de la CCPN. Para ello utiliza los siguientes métodos:

public void muestraAnclas( ): Este método va dibujando en la pantalla el conjunto y número de anclas para cada transición T_{rule}.

public void obtieneAnclas( ): Con este método se establece la comunicación con la clase NumAnclas, a fin de recuperar los valores del número de anclas.

9. La Clase VistaDistancia

Esta clase presenta al usuario el valor de la distancia para cada transición T_{rule}. Consta de los siguientes métodos:

public void formaCadena( ): Este método permite darle el formato adecuado a los datos que se van a presentar al usuario.

public void muestraDistancia( ): Este método va dibujando en la pantalla el valor de la distancia para cada transición, así como la ruta más larga a la que corresponde el valor de distancia.

public void obtieneDistancia( ): Con este método se establece la comunicación con la clase Distancia para recurerar los datos que esta clase generó para cada transición.

public void obtieneRutaMasLarga( ): Mediante este método se obtiene la ruta más larga para cada transición.

10. La Clase VistaPotDisparo

La clase VistaPotDisparo muestra al usuario los resultados que se obtuvieron para el parámetro potencial de disparo de cada una de las transiciones T_{rule}.

public void muestraPotDisparo( ): Este método va mostrando en la pantalla el valor del parámetro po-

tencial de disparo para cada transición T_{rule}.

public void obtienePotDisparo( ): Este método se comunica con la clase PotDisparo para obtener los valores que dicha clase calculó para cada transición T_{rule}.

11. La Clase VistaMediana

Esta clase se implementó para mostrarle al usuario el resultado del cálculo de la mediana para cada parámetro de medición. Consta de los siguientes métodos:

public void muestraMediana( ): Este método dibuja sobre la pantalla el valor de la mediana de cada parámetro.

public void obtieneMediana( ): Con este método se puede comunicar la clase VistaMediana con la clase Mediana para recuperar el valor los valores calculados por ésta última clase.

12. La Clase VistaConversión

Esta clase muestra en la pantalla, la gráfica de disparo modificada de una CCPN. Para ello se auxilia de los siguientes métodos.

public void dibujaCirculo( ): Este método dibuja los círculos que representan a las reglas en la gráfica de disparo médificada.

public void dibujaCuadrado( ): Este método dibuja los cuadrados que representan las anclas.

public void dibujaLinea( ): Con este método se unen los elementos de la gráfica de disparo modificada.

public void dibujaPares( ): Este método tiene como objetivo recorrer cada uno de los pares para identificar el elemento que se va a dibujar. Verifica si ese elemento ya ha sido dibujado para ajustar las coordenadas de los elementos posteriores.

public void obtienePares( ): Este método se encarga de comunicarse con la clase Conversión para obtener los pares generados por ésta.

6.3. Uso del Módulo de Complejidad

En esta sección se presenta el funcionamiento del Módulo de Complejidad integrado a ECAPNSim. Para mostrar el funcionamiento vamos a considerar las siguientes reglas:

Regla 0

```
on delete DEP
if true
Then delete from EMP
```

```
       where EMP.SuDep = DEP.ElDep
```

Regla 1

```
on delete EMP
if true
Then delete from TRABAJA
       where EMP.ElEmp = TRABAJA.ElEmp
```

Regla 2

```
on insert EMP
if EMP.Salario > 15000
Then delete from EMP
       where EMP.ElEmp = new.EMP.ElEmp
```

Regla 3

```
on update DEP.Presupuesto
if true
Then update PROYECTO set value Presupuesto = 0.1 *
PROYECTO.Presupuesto where PROYECTO.ElProy = TRABAJA.ElProy
and TRABAJA.ElEmp = EMP.ElEmp and EMP.SuDep = DEP.ElDep
```

Estas reglas se capturan en un archivo de texto y se compilan utilizando el programa Automata.java incluído con el conjunto de programas de ECAPNSim. El resultado de la compilación es un archivo con extensión obj.

Para iniciar la interfaz ECAPNSim es necesario abrir una ventana de MS-DOS y teclear la siguiente instrucción:

c:\>java ECAPNSim

Dado que el lenguaje de programación Java es por, la manera de ejecutarlo en otros ambientes es similar.

Una vez ejecutada la aplicación y compilado el archivo de reglas, se importa este archivo. Esto se hace con la opción ImportAR de ECAPNSim. La figura 6.3 muestra la representación de las reglas ECA anteriores en la CCPN.

En la barra de menús de ECAPNSim se encuentra la opción Complexity. Haciendo click sobre esta opción se despliega un conjunto de opciones que incluyen el cálculo de las rutas, el número de anclas, la distancia, el potencial de disparo, la mediana y la conversión a la gráfica de disparo modificada. A continua-

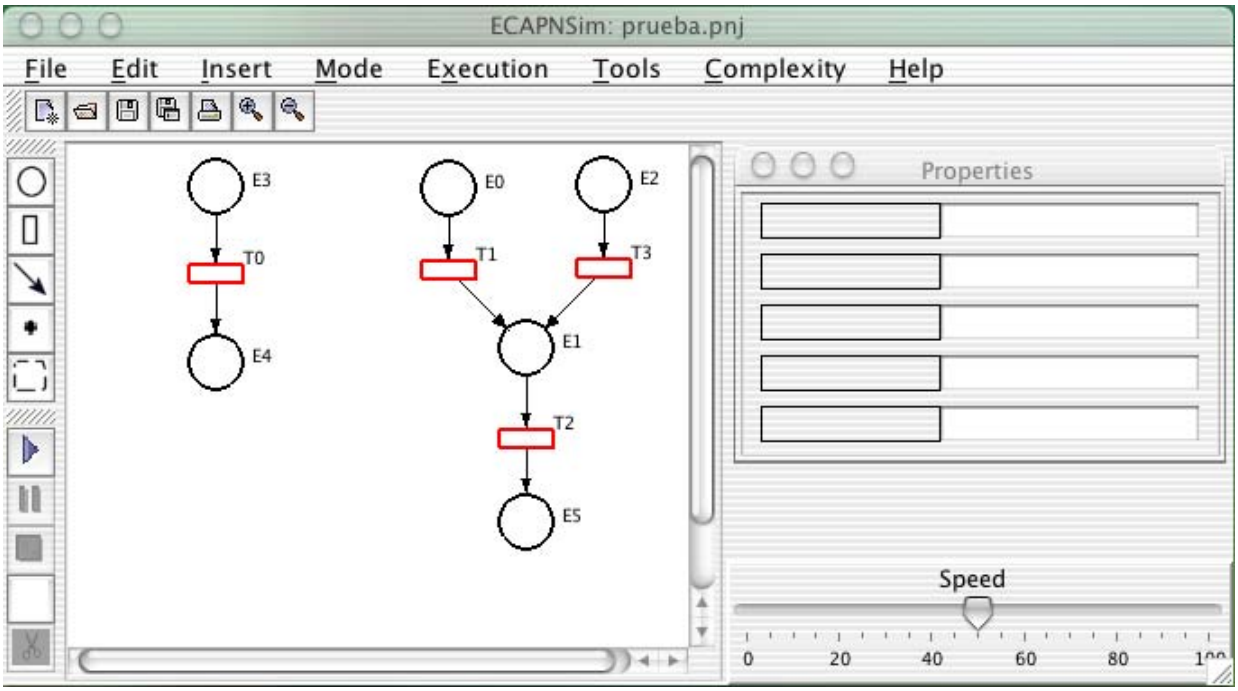

Figura 6.3: CCPN que representa el conjunto de reglas anterior

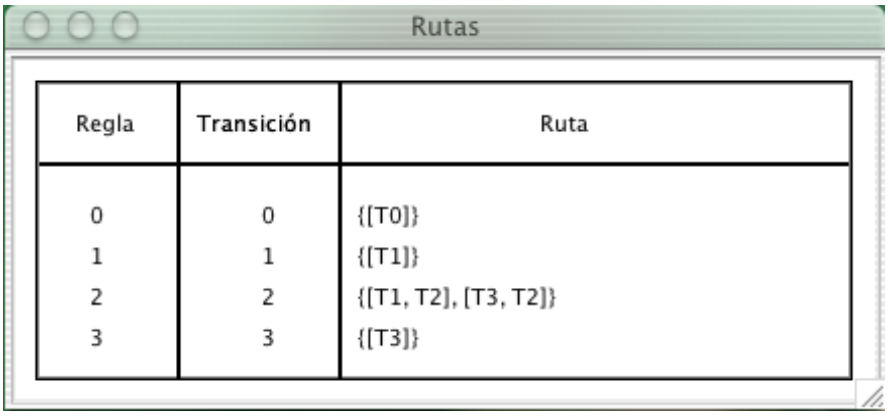

Figura 6.4: Rutas para las transiciones Trule de la CCPN

ción se muestra cada una de las ventanas en donde se visualizan esos parámetros.

Las dos primeras columnas de la figura 6.4 muestran la correspondencia entre las reglas y las transiciones de la CCPN. La tercer columna muestra todas las rutas de transiciones para cada transición T_{rule}.

Se puede ver que para la transición *T1* se tienen las rutas $[T0, T1]$ y $[T2, T1]$. Estas rutas se pueden verificar en el conjunto de reglas ECA, ya que la acción de la Regla 0 produce el evento de la Regla 1. Lo mismo sucede con la acción de la Regla 2. De esta forma, la Regla 1 se puede activar si se dispara la Regla 0 o si se dispara la Regla 2. No obstante, identificar de forma manual la dependencia que existe entre las reglas es tardado y complicado.

La figura 6.5 muestra el valor del parámetro número de anclas para cada transición. Adicionalmente, se muestran los lugares de entrada que forman las anclas. Así, el usuario no tiene que buscar cúales son los lugares iniciales, únicamente tiene que identificarlos sobre la CCPN.

La figura 6.6 refleja el valor de la distancia para cada transición. Este valor se obtiene a través de los algoritmos descritos en capítulos anteriores. Además, se muestra al usuario la ruta de transiciones más larga.

La figura 6.7 presenta el valor del potencial de disparo para cada transición. Se muestra el valor de las causas y la cardinalidad para que el usuario pueda corroborar el valor obtenido.

La figura 6.8 muestra el valor de la mediana para cada parámetro de medición.

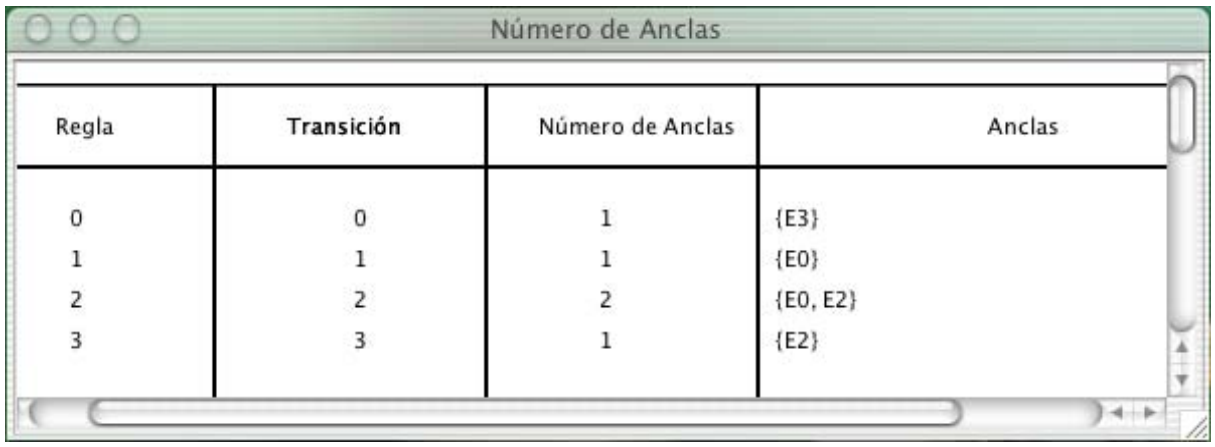

Figura 6.5: Valor del parámetro Número de Anclas

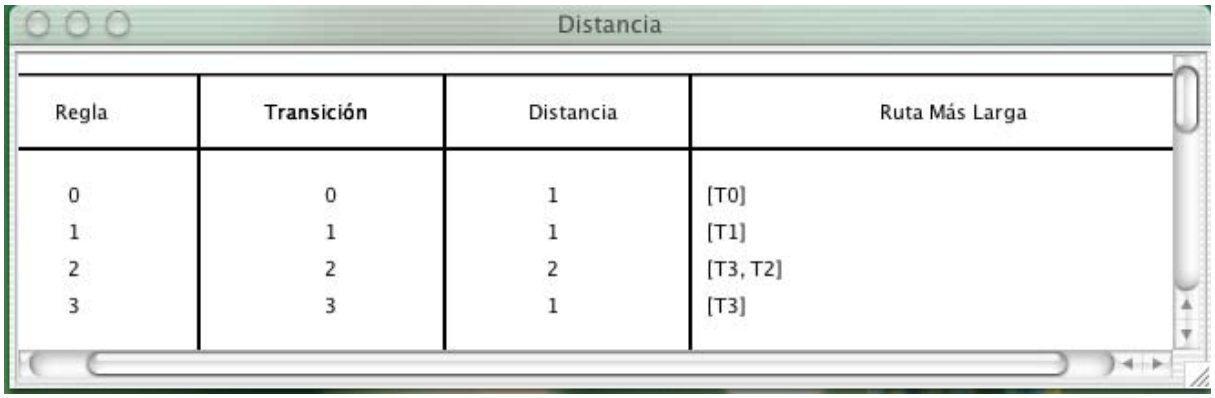

Figura 6.6: Valor del parámetro Distancia

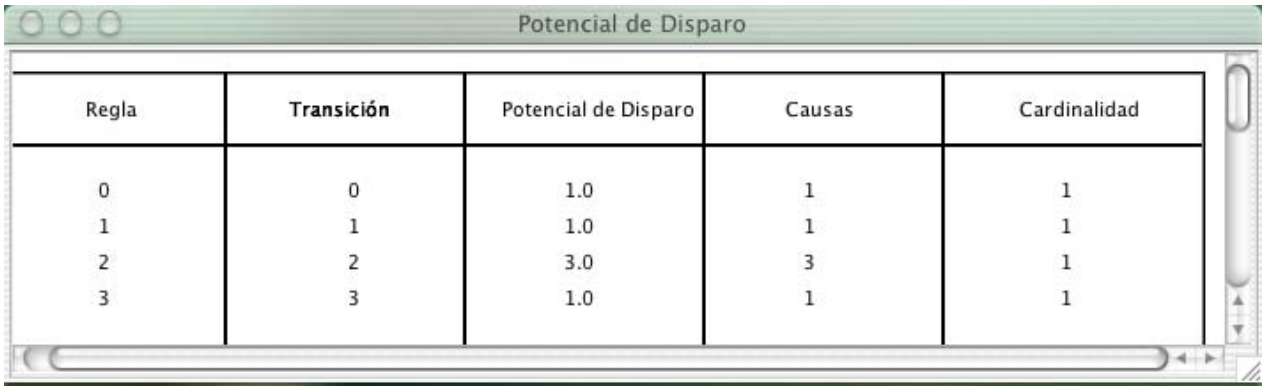

Figura 6.7: Valor del parámetro Potencial de Disparo

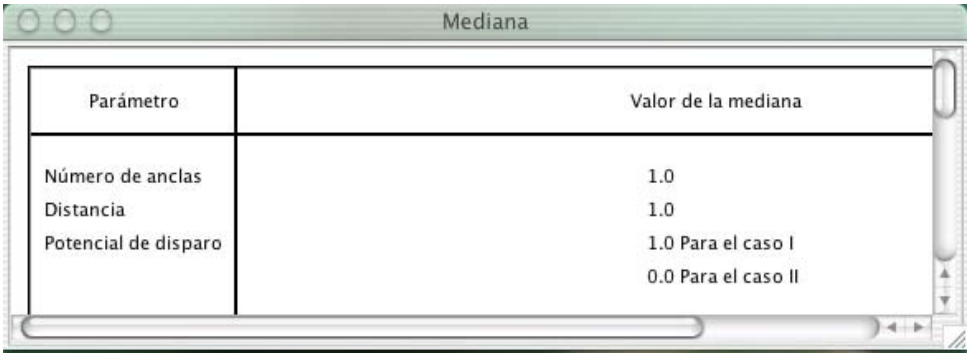

Figura 6.8: Valor de la mediana para cada parámetro de medición

6.4. Comentarios Finales

Es importante el desarrollo tanto de métricas como de sistemas que puedan aplicarlas automáticamente a las reglas ECA de una BDA a fin de analizar el comportamiento de los sistemas antes de su puesta en operación. La importancia de este sistema de software es que puede determinar la complejidad de un conjunto de reglas ECA y detectar aquellas reglas que, por su dependencia con algunas otras, pueden causar problemas en el manteniemiento.

La integración del Módulo de Complejidad a la interfaz gráfica ECAPNSim permitió obtener una herramienta más completa. Su desarrollo completamente en Java permite que este sistema funcione en diferentes sistemas operativos

Capítulo 7

Casos de estudio

En este capítulo presentamos el análisis de complejidad, utilizando los parámetros de medición propuestos, para algunos ejemplos de bases de reglas ECA. Tales bases de reglas fueron modeladas como CCPN mediante la interfaz ECAPNSim. El capítulo se estructura de la siguiente manera: la sección 7.1 muestra la descripción y el análisis de complejidad del primer caso de estudio, la sección 7.2 presenta la descripción y el análisis de complejidad para el segundo caso de estudio, la sección 7.3 muestra una comparación con el trabajo propuesto en [10] y la sección 7.4 expone los comentarios finales del capítulo.

7.1. Una BD de empleados

En esta sección presentamos el análisis de complejidad para un conjunto de siete reglas que actúan sobre las tablas de la BD de una compañía.

7.1.1. Descripción del sistema

El conjunto de reglas de este caso de estudio se tomaron de [38]. Se trata de una BD que almacena información acerca de empleados, los cuales trabajan en ciertos proyectos de un determinado departamento. Las tablas que forman parte de la BD son las siguientes:

Tablas

```
EMP( SuDep, ElEmp, Salario, Bono )
DEP( ElDep, Prod, Presupuesto )
```

```
PROYECTO( ElProy, Presupuesto )
TRABAJA( ElProy, ElEmp )
```

La tabla EMP almacena información acerca de los empleados que trabajan en una compañía. Tal información incluye el departamento en el que trabajan (SuDep), el número de empleado (ElEmp), su salario (Salario) y un bono por productividad (Bono). La información de los departamentos se almacena en la tabla DEP. Los campos de esta tabla contienen el departamento (ElDep), la producción del departamento (Prod) y el presupuesto asignado (Presupuesto). La tabla PROYECTO almacena información acerca de los proyectos (ElProy) desarrollados en la compañía y el presupuesto (Presupuesto) asignado a cada uno. La tabla TRABAJA almacena los datos de un empleado (ElEmp) asignado a un proyecto (ElProy).

El siguiente conjunto de reglas se asociada con las tablas anteriores.

Regla 0

```
on delete to Dep
if true
then delete to Emp
       where Emp.SuDep = Dep.ElDep
```

Regla 1

```
on delete to Trabaja
if true
then delete to Proyecto
       where Proyecto.ElProy = Trabaja.ElProy
```

Regla 2

```
on delete to Emp
if true
then delete to Trabaja
       where Emp.ElEmp = Trabaja.ElEmp
```

Regla 3

```
on( or( insert Emp ), ( update Emp.Salario ) )
if ( Emp.Salario > 15,000 )
then delete to Emp
       where Emp.ElEmp = new.Emp.ElEmp
```

Regla 4

```
on( and( update Dep.prod ), ( insert Emp ) )
if( Dep.Prod > 90 & Emp.SuDep = Dep.ElDep )
then update Emp.Bono = 100
       where Emp.ElEmp = new.Emp.ElEmp
```

Regla 5

```
on( or( update Dep.Presupuesto ), ( insert Dep ) )
if true
then update Proyecto.Presupuesto = 0.1 * Proyecto.Presupuesto
       where Proyecto.ElProy = Trabaja.ElProy
       and Trabaja.ElEmp = Emp.ElEmp
       and Emp.SuDep = Dep.ElDep
```

Regla 6

```
on update Proyecto.Presupuesto
if true
then update Emp.Salario = 0.01 * Proyecto.Presupuesto
       where Proyecto.ElProy = Trabaja.ElProy
           and Trabaja.ElEmp = Emp.ElEmp
```

7.1.2. Análisis de complejidad

La CCPN que representa el conjunto de reglas completo mostrado al inicio de esta sección se presentó en la figura 5.1.

La figura 7.1 muestra las rutas para las transiciones T_{rule} de la CCPN.

La figura 7.2 presenta los resultados obtenidos para todas las transiciones T_{rule} del parámetro número de anclas.

Analizando las reglas en cuanto al parámetro número de anclas, vemos que las transiciones *T6* y *T7* son más complejas que cualquiera de las otras transiciones. Recordemos que, formalmente, el parámetro NA es un mapeo, $NA : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow NA\left(A_i\right) \geqslant NA\left(A_j\right); \forall A_i, A_j \in A$$

Rutas

Regla	Transición	Ruta
0	5	{[T5]}
1	6	{[T5, T7, T6], [T0, T8, T7, T6], [T3, T10, T11, T1, T8, T7, T6], [T4, T10, T11, T1, T8, T7, T6]}
2	7	{[T5, T7], [T0, T8, T7], [T3, T10, T11, T1, T8, T7], [T4, T10, T11, T1, T8, T7]}
3	8	{[T0, T8], [T3, T10, T11, T1, T8], [T4, T10, T11, T1, T8]}
4	9	{[T2, T9], [T0, T2, T9]}
5	10	{[T3, T10], [T4, T10]}
6	11	{[T3, T10, T11], [T4, T10, T11]}

Figura 7.1: Rutas para las transiciones Trule del Ejemplo 3

Número de Anclas

Regla	Transición	Número de Anclas	Anclas
0	5	1	{E0}
1	6	4	{E0, E3, E9, E10}
2	7	4	{E0, E3, E9, E10}
3	8	3	{E3, E9, E10}
4	9	2	{E6, E3}
5	10	2	{E9, E10}
6	11	2	{E9, E10}

Figura 7.2: Valor del parámetro Número de Anclas

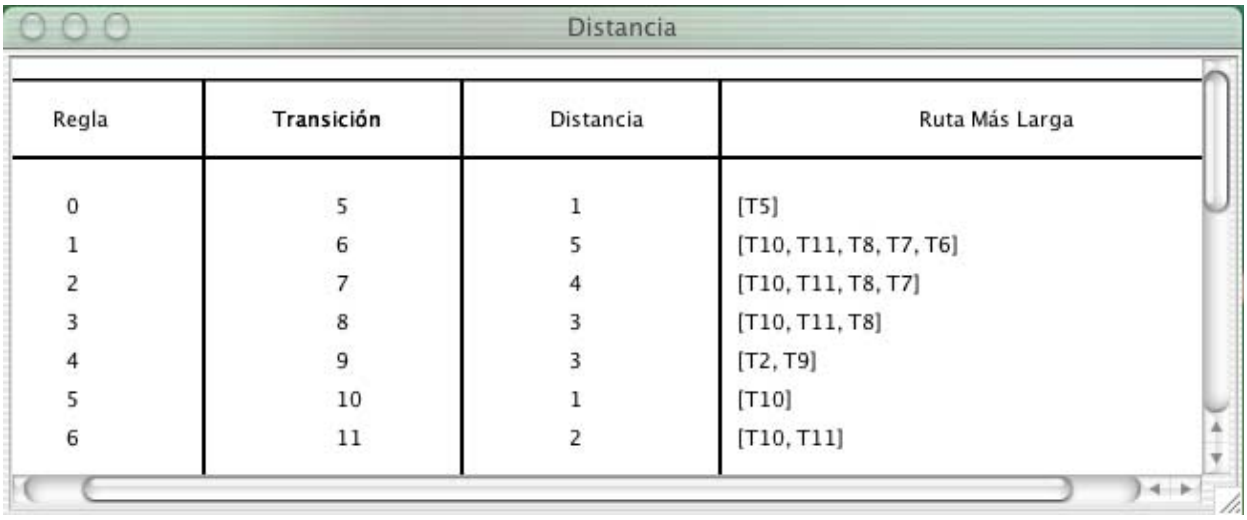

Figura 7.3: Valor del parámetro Distancia

donde A es el conjunto de reglas y $\cdot \geqslant$ es la relación empírica más o igual compleja que.

Entonces, si tomamos las Reglas 0 y 1 podemos decir que la Regla $1 \cdot >$ Regla 0 porque NA(Regla 1) $> NA$(Regla 0), es decir, $4 > 1$.

El valor numérico que obtuvimos a través de la CCPN lo podemos corroborar en el conjunto de reglas. Para analizar la Regla 1 tenemos que seguir las rutas que inician en los lugares *E0*, *E3*, *E9* y *E10* mientras que para analizar o entender el contexto de la Regla 0 sólo tenemos que analizar la ruta que inicia en el lugar *E0*. Por lo tanto, tenemos que realizar un mayor trabajo para entender la Regla 1 que para entender la Regla 0.

Ahora vamos a analizar qué sucede con el parámetro distancia.

La figura 7.3 muestra el valor de la distancia para las transiciones T_{rule}.

Recordemos que la definición de este parámetro es: D es un mapeo, $D : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow D\left(A_i\right) \geqslant D\left(A_j\right); \forall A_i, A_j \in A$$

En este caso, la transición *T6* es más compleja que cualquiera de las otras reglas, ya que tiene un valor de distancia mayor. Tomemos las transiciones *T6* (la cual representa a la Regla 1) y *T9* (la cual representa a la Regla 4) para analizar el comportamiento de este parámetro. De acuerdo a los resultados que obtuvimos,

Figura 7.4: Valor del parámetro Potencial de Disparo

podemos decir que la Regla $1 \cdot >$ Regla 4 porque $D(\text{Regla } 1) > D(\text{Regla } 4)$, esto es, $5 > 3$. Los resultados que obtuvimos en el sistema numérico deben reflejar las observaciones que hacemos en el conjunto de reglas, por lo tanto, vamos a verificar la consistencia de los resultados.

En el conjunto de reglas, la Regla 1 necesita para activarse (en el peor de los casos) del disparo previo de las Reglas 5, 6, 3, 2 y 1 mientras que la Regla 4 sólo necesita que ocurra su evento. En este caso, el evento de la Regla 4 es compuesto (and) y esta es la razón de que aún cuando la transición *T2* no representa una regla, es necesaria para formar el evento compuesto and.

El último análisis que presentamos es el del potencial de disparo.

La figura 7.4 muestra el valor del potencial de disparo para cada transición T_{rule}.

En cuanto al potencial de disparo, la definición de este parámetro está dada en dos casos:

Caso I: El potencial de disparo $\forall A_i, A_j \in A$ cuyo evento sea primitivo ó los eventos compuestos OR y NEGACIÓN se define formalmente como un mapeo $TP : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow TP\left(A_i\right) \geqslant TP\left(A_j\right)\ \forall A_i, A_j \in A$$

Caso II: El potencial de disparo $\forall A_i, A_j \in A$ cuyo evento sea compuesto, excluyendo a los eventos compuestos OR y NEGACIÓN, se define formalmente como un mapeo $TP : A \rightarrow \Re$ tal que:

$$A_i \cdot \geqslant A_j \Leftrightarrow TP\left(A_i\right) \leqslant TP\left(A_j\right)\ \forall A_i, A_j \in A$$

Tomemos las transiciones *T7* y *T6* para comparar su complejidad. Ya que el potencial de disparo de la transición *T7* (la cual representa a la Regla 2) es mayor que el potencial de disparo de la transición *T6* (la cual representa a la Regla 1) podemos decir que la Regla 2 es más compleja que la Regla 1 (aplicamos el caso I de la definición de potencial de disparo porque el evento que activa ambas reglas es primitivo). Vamos a corroborar este resultado en el conjunto de reglas para verificar si, efectivamente, los valores numéricos que obtuvimos reflejan lo que sucede en el mundo real.

Podemos ver que la Regla 2 se puede activar por la ocurrencia de su evento, sin embargo, el disparo de la Regla 0 o el disparo de la Regla 3 también pueden originar que se active. Por lo tanto, cuando queremos darle mantenimiento a esta regla debemos verificar, además, la Regla 0 y la Regla 3.

Por otro lado, la Regla 1 sólo se puede activar por la ocurrencia de su evento o por el disparo de la Regla 2. Como vemos, al darle mantenimiento a esta regla sólo tenemos que verificar dos causas.

Ahora vamos a ilustrar el caso II de la definición de potencial de disparo. Para ello, tomemos las transiciones *T5* (la cual representa a la Regla 0) y *T9* (la cual representa a la Regla 4). Como el valor de la causa es mayor que el valor de la cardinalidad para el lugar de entrada de la transición *T9*, podemos inferir que se trata de un evento compuesto. Como el valor del potencial de disparo para *T9* es menor que el valor del potencial de disparo para *T5*, entonces se cumple que la Regla 9 es más compleja que la Regla 0. Esto corrobora que las reglas activadas por eventos compuestos son más complejas que las reglas activadas por eventos primitivos.

Como se había mencionado, la validación de los parámetros sobre el marco de trabajo del Dr. Zuse nos permitió concluir que la escala de medición es la ordinal. Una de las operaciones estadísticas que se pueden efectuar en la escala ordinal es la mediana. El valor de la mediana ayuda a identifica las reglas que sobrepasan un cierto valor. En la figura 7.5 se muestra el valor de la mediana para cada parámetro.

Utilizando esta información podemos centrar nuesta atención en aquellas reglas que se encuentren por arriba de la mediana para tratar de bajar su complejidad, de forma que su mantenimiento se simplifique.

En la siguiente sección vamos a mostrar un caso de estudio con un conjunto más grande de reglas.

Figura 7.5: Valor de la Mediana para cada parámetro

7.2. Una BD de un sistema bancario

En esta sección presentamos la descripción y el análisis de complejidad para una base de reglas que actúa sobre las tablas de una BD de un sistema bancario.

7.2.1. Descripción del sistema

Este caso de estudio consta de 30 reglas ECA pertenecientes a un sistema bancario. Las tablas sobre las que actúan estas reglas son las siguientes:

```
CLIENTE( num_cliente, nombre, domicilio, fecha_alta )
CUENTA(numero, num_cliente, saldo, límite, tipo, estado,num_empleado)
PENDIENTES( clave, tipo, cuenta_origen, cuenta_destino, cantidad,
          periodo )
DEPÓSITO( fecha, cuenta, cantidad, lugar, moneda )
RETIRO( fecha, cuenta, cantidad, lugar, ubicación )
PAGO_POR_SERVICIO( fecha, cuenta_origen, cuenta_destino, modo_acceso,
cantidad, concepto_pago, ubicación )
EXPIDE_COMPROBANTE( descripcion )
IMPRIME_TIPO_1( descripcion )
```

```
ACCESO_EXTERNO_CUENTA( descripcion )
REEXPEDICIÓN_ESTADO_CUENTA( descripcion )
IMPRIME_TIPO_2( descripcion )
GENERAR_NIP( descripcion )
IMPRIME_TIPO_3( descripcion )
IMPRIME_TIPO_4( tipo, descripcion, cliente, cuenta, lugar, estado )
REPORTE ( tipo, descripcion, cliente, cuenta, lugar, estado )
REPOSICIÓN_TARJETA( descripcion )
MENSAJES( mensaje );
EXPEDICIÓN_TC( numero, cuenta, con_fotografia, adicional )
EXPEDICIÓN_CHEQUE( numero, cuenta, cantidad )
```

En las tablas anteriores se almacena información acerca de los clientes del banco y de las diferentes operaciones que pueden efectuar.

Las reglas asociadas con estas tablas son las siguientes:

Regla 0

```
on or( insert_DEPOSITO:insert_RETIRO:insert_PAGO_POR_SERVICIO )
if true
Then insert into expide_comprobante values (’expide_comprobante’)
```

Regla 1

```
on insert_EXPIDE_COMPROBANTE
if DEPOSITO.lugar = ’Sucursal’ OR RETIRO.lugar = ’Sucursal’
Then insert into imprime_tipo_1 values (’imprime_tipo_1’)
```

Regla 2

```
on insert_RETIRO
if CUENTA.saldo > RETIRO.cantidad
Then update CUENTA set value saldo = saldo - RETIRO.cantidad
```

Regla 3

```
on insert_RETIRO
if CUENTA.saldo <= RETIRO.cantidad
Then insert into imprime_tipo_3 values (’imprime_tipo_3’)
```

Regla 4

```
on insert_PAGO_POR_SERVICIO
if CUENTA.saldo > PAGO_POR_SERVICIO.cantidad
Then insert into RETIRO values ( today, CUENTA.numero,
PAGO_POR_SERVICIO.cantidad, 'Sucursal', PAGO_POR_SERVICIO.ubicacion)
```

Regla 5

```
on insert_RETIRO
if CUENTA.saldo <= PAGO_POR_SERVICIO.cantidad
Then insert into imprime_tipo_3 values ('imprime_tipo_3')
```

Regla 6

```
on insert_DEPOSITO
if DEPOSITO.fecha < 16:00
Then update CUENTA set value saldo = saldo + DEPOSITO.cantidad
```

Regla 7

```
on insert_DEPOSITO
if DEPOSITO.fecha >= 16:00
Then insert into PENDIENTES values (CUENTA.numero, 'deposito', NULL,
DEPOSITO.cuenta, DEPOSITO.cantidad, 2 )
```

Regla 8

```
on insert_PAGO_POR_SERVICIO
if PAGO_POR_SERVICIO.modo_acceso = 'Internet'
Then insert into RETIRO values (today, CUENTA.numero, 25.00,
'INTERNET', PAGO_POR_SERVICIO.ubicacion )
```

Regla 9

```
on insert_PAGO_POR_SERVICIO
if PAGO_POR_SERVICIO.modo_acceso = 'Telefono'
Then insert into RETIRO values (today, CUENTA.numero, 20.00,
'TELEFONO', PAGO_POR_SERVICIO.ubicacion )
```

Regla 10

```
on insert_PAGO_POR_SERVICIO
```

```
if true
Then insert into RETIRO values (today, CUENTA.numero, 20.00,
'SUCURSAL', PAGO_POR_SERVICIO.ubicacion )
```

Regla 11

```
on insert_EXPIDE_COMPROBANTE
if DEPOSITO.lugar = 'Cajero automatico'
Then insert into imprime_tipo_2 values ( ' imprime_tipo_2 ' )
```

Regla 12

```
on insert_CUENTA
if true
Then insert into generar_nip values ( ' generar_nip ' )
```

Regla 13

```
on insert_DEPOSITO
if DEPOSITO.moneda <> 'PESO' & DEPOSITO.fecha < 1600
Then update CUENTA set value cantidad = cantidad - DEPOSITO.cantidad
+ DEPOSITO.cantidad * tipomoneda
```

Regla 14

```
on insert_DEPOSITO
if DEPOSITO.moneda <> 'PESO' & DEPOSITO.fecha >= 1600
Then insert into PENDIENTES values (1, 'deposito', null,
DEPOSITO.cuenta, DEPOSITO.cantidad, 2) )
```

Regla 15

```
on seq(insert_REPORTE:insert_REPOSICIÓN_TARJETA)
if CUENTA.saldo > 150
then insert into PAGO_POR_SERVICIO values (today, NULL,
CUENTA.numero, REPORTE.lugar, 150, 'Servicios_varios', 'Nacional' )
```

Regla 16

```
on insert_ACCESO_EXTERNO_CUENTA
if CUENTA.estado = 'Bloqueda' OR CUENTA.estado = 'Cancelada' OR
CUENTA.estado = 'No existe'
```

```
then insert into MENSAJES values (’Cuenta inactiva’)
```

Regla 17

```
on update_REPORTE_estado
if REPORTE.tipo = ’Reclamacion’ & REPORTE.estado = ’No procede’
then insert into PAGO_POR_SERVICIO values ( today, NULL,
CUENTA.numero, REPORTE.lugar, 200, ’Servicios Varios’, ’Nacional’)
```

Regla 18

```
on insert_EXPEDICIÓN_TC
if expedicion_TC.con_fotografia = 1
Then insert into PAGO_POR_SERVICIO values (today, NULL,
expedicion_TC.cuenta, ’Automatico’, 200, ’Servicios varios’,
’Nacional’)
```

Regla 19

```
on insert_EXPEDICIÓN_CHEQUE
if CUENTA.saldo < expedicion_cheque.cantidad
Then insert into PAGO_POR_SERVICIO values (today, NULL,
expedicion_cheque.cuenta, ’Automatico’, 900, ’Servicios varios’,
’Nacional’)
```

Regla 20

```
on insert_EXPEDICIÓN_TC
if expedicion_TC.adicional = 1
Then insert into PAGO_POR_SERVICIO values (today, NULL,
expedicion_TC.cuenta, ’Automatico’, 100, ’Servicios varios’,
’Nacional’)
```

Regla 21

```
on insert_EMPLEADO
if true
Then insert into CUENTA values (numero, num_cliente,
EMPLEADO.salario, NULL, ’Nómina’, ’No existe’, NULL)
```

Regla 22

```
on insert_EMPLEADO
if EMPLEADO.puesto = 'Gerente'
Then insert into CUENTA values (numero, num_cliente, NULL, 10000.00,
'Crédito', 'No existe', NULL)
```

Regla 23

```
on insert_EMPLEADO
if EMPLEADO.puesto = 'Subgerente'
Then insert into CUENTA values (numero, num_cliente, NULL, 8000.00,
'Crédito', 'No existe', NULL)
```

Regla 24

```
on insert_EMPLEADO
if EMPLEADO.puesto = 'Ejecutivo de cuenta'
Then insert into CUENTA values (numero, num_cliente, NULL, 5000.00,
'Crédito', 'No existe', NULL)
```

Regla 25

```
on insert_EMPLEADO
if EMPLEADO.puesto = 'Cajero'
Then insert into CUENTA values (numero, num_cliente, NULL, 3000.00,
'Crédito', 'No existe', NULL)
```

Regla 26

```
on delete_EMPLEADO
if true
Then update CUENTA set value tipo = 'Ahorro' where CUENTA.num_cliente
= EMPLEADO.num_empleado
```

Regla 27

```
on update_CUENTA_saldo
if EMPLEADO.aumento > 1000
Then update CUENTA set value limite = limite + EMPLEADO.aumento where
CUENTA.num_cliente = EMPLEADO.num_empleado
```

Regla 28

```
on insert_CUENTA
if true
Then update EMPLEADO set value puntos = puntos + 1 where
CUENTA.num_empleado = EMPLEADO.num_empleado
```

Regla 29

```
on insert_CUENTA
if CUENTA.tipo = 'Crédito'
Then insert into PAGO_POR_SERVICIO values ( today, CUENTA.numero,
NULL, 'Automatico' , 350.00, 'Servicios varios' , 'Sucursal' )
```

Para este conjunto de reglas vamos a realizar su análisis de complejidad. El resultado del análisis lo presentamos en el siguiente apartado.

7.2.2. Análisis de complejidad

Utilizando el Módulo de Complejidad que integramos a ECAPNSim mostramos los valores de los parámetros de medición para el conjunto de reglas anterior.

La figura 7.6 muestra la CCPN que representa al conjunto de reglas ECA del sistema bancario.

Como puede observarse, la CCPN resultante es de un tamaño considerable. Es esta razón la que nos condujo a la utilización de la matriz de incidencia como medio para calcular los parámetros de medición.

Los valores para el parámetro de medición número de anclas se muestran en la figura 7.7.

Podemos observar que las transiciones *T4* (Regla 0) y *T5* (Regla 1) tienen un valor de número de anclas mayor, por lo tanto, las Reglas 0 y 1 son las más complejas (en cuanto a número de anclas) de todo el conjunto.

La figura 7.8 muestra el valor del parámetro distancia.

En lo que respecta al parámetro de distancia, las transiciones para las cuales se obtuvo un valor mayor son *T5* y *T15*. Esto significa que las reglas representadas por esas transiciones (Regla 1 y Regla 11, respectivamente) son las más complejas.

La figura 7.9 muestra el valor del parámetro potencial de disparo.

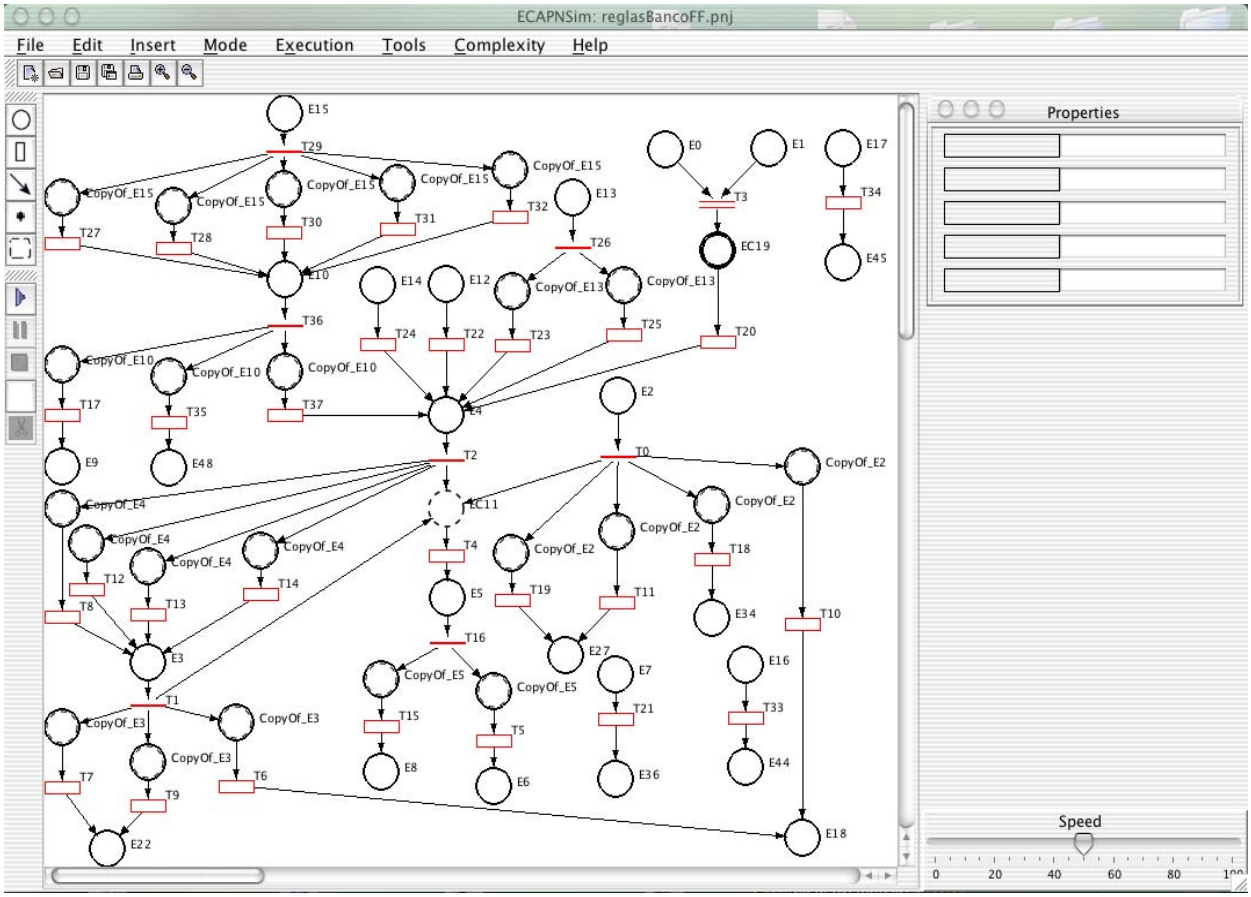

Figura 7.6: CCPN que representa las reglas del sistema bancario

Número de Anclas

Regla	Transición	Número de Anclas	Anclas
0	4	7	{E2, E0, E1, E12, E13, E14, E15}
1	5	7	{E2, E0, E1, E12, E13, E14, E15}
2	6	6	{E0, E1, E12, E13, E14, E15}
3	7	6	{E0, E1, E12, E13, E14, E15}
4	8	6	{E0, E1, E12, E13, E14, E15}
5	9	6	{E0, E1, E12, E13, E14, E15}
6	10	1	{E2}
7	11	1	{E2}
8	12	6	{E0, E1, E12, E13, E14, E15}
9	13	6	{E0, E1, E12, E13, E14, E15}
10	14	6	{E0, E1, E12, E13, E14, E15}
11	15	7	{E2, E0, E1, E12, E13, E14, E15}
12	17	1	{E15}
13	18	1	{E2}
14	19	1	{E2}
15	20	2	{E0, E1}
16	21	1	{E7}
17	22	1	{E12}
18	23	1	{E13}
19	24	1	{E14}
20	25	1	{E13}
21	27	1	{E15}
22	28	1	{E15}
23	30	1	{E15}
24	31	1	{E15}
25	32	1	{E15}
26	33	1	{E16}
27	34	1	{E17}
28	35	1	{E15}
29	37	1	{E15}

Figura 7.7: Valores del parámetro Número de Anclas

Distancia

Regla	Transición	Distancia	Ruta Más Larga
0	4	5	[T3, T20, T8, T4]
1	5	6	[T3, T20, T8, T4, T5]
2	6	5	[T3, T20, T12, T6]
3	7	5	[T3, T20, T12, T7]
4	8	4	[T3, T20, T8]
5	9	5	[T3, T20, T12, T9]
6	10	1	[T10]
7	11	1	[T11]
8	12	4	[T3, T20, T12]
9	13	4	[T3, T20, T13]
10	14	4	[T3, T20, T14]
11	15	6	[T3, T20, T8, T4, T15]
12	17	2	[T28, T17]
13	18	1	[T18]
14	19	1	[T19]
15	20	3	[T3, T20]
16	21	1	[T21]
17	22	1	[T22]
18	23	1	[T23]
19	24	1	[T24]
20	25	1	[T25]
21	27	1	[T27]
22	28	1	[T28]
23	30	1	[T30]
24	31	1	[T31]
25	32	1	[T32]
26	33	1	[T33]
27	34	1	[T34]
28	35	2	[T28, T35]
29	37	2	[T28, T37]

Figura 7.8: Valor del parámetro Distancia

Potencial de Disparo

Regla	Transición	Potencial de Disparo	Causas	Cardinalidad
0	4	3.0	3	3
1	5	2.0	2	1
2	6	5.0	5	1
3	7	5.0	5	1
4	8	7.0	7	1
5	9	5.0	5	1
6	10	1.0	1	1
7	11	1.0	1	1
8	12	7.0	7	1
9	13	7.0	7	1
10	14	7.0	7	1
11	15	2.0	2	1
12	17	6.0	6	1
13	18	1.0	1	1
14	19	1.0	1	1
15	20	0.5	1	2
16	21	1.0	1	1
17	22	1.0	1	1
18	23	1.0	1	1
19	24	1.0	1	1
20	25	1.0	1	1
21	27	1.0	1	1
22	28	1.0	1	1
23	30	1.0	1	1
24	31	1.0	1	1
25	32	1.0	1	1
26	33	1.0	1	1
27	34	1.0	1	1
28	35	6.0	6	1
29	37	6.0	6	1

Figura 7.9: Valor del Potencial de Disparo

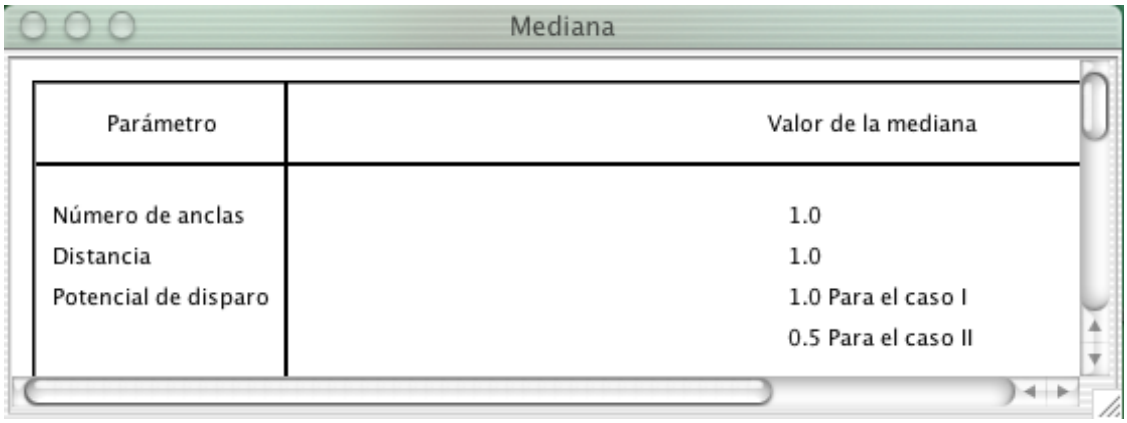

Figura 7.10: Valor de la Mediana

Dentro de los valores obtenidos para el potencial de disparo podemos ubicar a la transición *T20* (Regla 15) como la más compleja de todas ya que tiene el menor potencial de disparo. Si verificamos en el conjunto de reglas, podemos ver que el evento de esta regla es compuesto e involucra tiempo. Cabe mencionar que en el análisis que realizamos consideramos la parte estática de la CCPN ya que no podemos decir nada acerca de la frecuencia de ocurrencia de los eventos.

Finalmente, mostramos el valor de la mediana para cada parámetro de medición.

La figura 7.10 muestra el valor de la mediana para cada parámetro de medición.

Además de realizar el análisis de complejidad para los dos casos anteriores, desarrollamos una comparación con el trabajo [10] para mostrar las ventajas de nuestras mediciones. La comparación se muestra en la siguiente sección.

7.3. Comparaciones con trabajos relacionados

En esta sección mostramos una comparación entre el trabajo que realizamos en esta tesis y el trabajo desarrollado en [10]. Para ello, desarrollamos un módulo que realiza la conversión de la gráfica de CCPN a la gráfica de disparo modificada propuesta en [10]. El conjunto de reglas que utilizamos es el que mostramos al inicio de este capítulo.

La CCPN que representa a esas reglas se muestra en la figura 5.1.

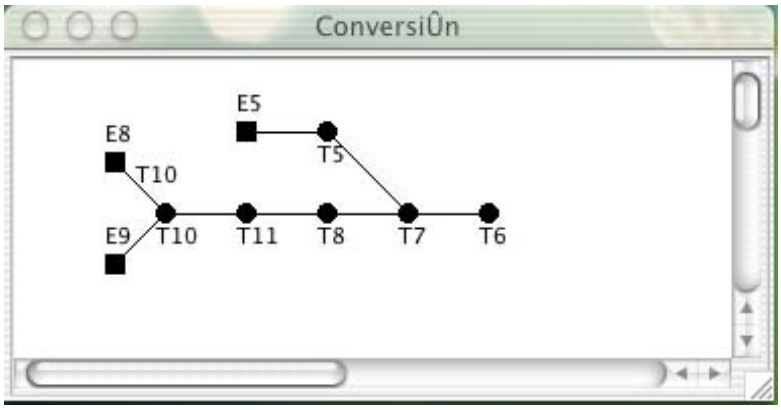

Figura 7.11: Gráfica de disparo modificada

En la figura 7.11 se muestra la gráfica de disparo modificada que representa al conjunto de reglas. Utilizando esta gráfica no se pueden representar eventos compuestos, por lo tanto, se han omitido algunas reglas y sólo se han conservado aquellas que se pueden representar.

El primer aspecto que vamos a comparar es el modelo para representar a las reglas ECA.

En [10] se utiliza una gráfica de disparo modificada. Una gráfica de disparo es un par $<S, L>$ donde S representa la conjunto de reglas ECA y L es un conjunto de arcos dirigidos, donde un arco se dibuja de S_1 a S_j si la acción de S_i provoca la ocurrencia de alguno de los eventos que participan en S_j. Los autores modifican esta definición agregando un peso a cada arco. El peso representa número de ocurrencias de eventos potenciales producidos por el disparo de la regla que pueden afectar a la regla disparada. Una segunda modificación consiste en extender el conjunto de nodos S con el conjunto de transacciones T. Una transacción es un conjunto atómico de acciones de BD, donde cualquiera de esas acciones puede corresponder a un evento que dispara una o más reglas. Un ejemplo de gráfica de disparo modificada es el de la figura 7.11.

En este trabajo de tesis, proponemos el modelo de CCPN para representar a las reglas ECA. Este modelo fue desarrollado específicamente para BDAs, por lo que se consideran todas las partes de la regla ECA. Cada elemento de la regla se mapea a la CCPN de la forma siguiente: el evento y la acción se mapean como lugares de entrada y salida, respectivamente. La condición se representa como una transición. Además, el disparo de la regla corresponde al disparo de la transición y la detección de eventos se modela como tokens iniciales.

Por otro lado, además de contar con una representación gráfica, la CCPN está dotada de una repre-

sentación matemática dada por la matriz de incidencia.

De la descripción de ambos modelos obtenemos las siguientes conclusiones:

1. La gráfica de disparo modificada es una representación compacta para las reglas ECA ya que encapsula todos sus elementos en un solo nodo. Sin embargo, esta representación no es la más adecuada para realizar mediciones puesto que oculta los detalles de la formación de eventos.

2. La gráfica de disparo modificada consta de un único modelo gráfico, el cual funciona adecuadamente para conjuntos de reglas pequeños. No obstante, cuando el conjunto de reglas es grande, el proceso de medición es tedioso y complicado.

3. La CCPN proporciona un mayor detalle en la información de la regla ECA porque considera cada uno de sus elementos. Esto provoca que la representación gráfica sea más grande que la representación por gráfica de disparo modificada.

4. La CCPN cuenta con una representación gráfica y una representación matemática, lo cual asegura la consistencia de nuestras mediciones.

5. En la CCPN el proceso de medición se facilita ya que se aprovecha su representación por matriz de incidencia para desarrollar algoritmos que puedan obtener automáticamente el valor de los parámetros.

Un segundo aspecto que vamos a comparar es la forma de realizar la medición.

En [10], al contar únicamente con la representación gráfica el proceso de medición se realiza visualmente. Para conjuntos pequeños es adecuado, sin embargo, al trabajar con conjuntos grandes el proceso se dificulta.

Con la CCPN modelamos los parámetros de medición sobre su estructura gráfica, no obstante, el proceso de medición lo realizamos sobre la estructura matemática ya que es adecuada para manipularla por medio de algoritmos. Los algoritmos desarrollados se pueden implementar en un lenguaje de programación y así, no importando el tamaño del conjunto de reglas, la medición la realizará la computadora.

El tercer aspecto a considerar son el resultado de las mediciones, para ello vamos a considerar la tabla 7.1 que muestra el valor de cada parámetro para la figura 7.11.

Regla	Nodo	Número de Anclas	Distancia	Potencial de Disparo
0	*T5*	1	1	1
1	*T6*	2	5	1
2	*T7*	2	4	2
3	*T8*	2	3	1
5	*T10*	2	1	2
6	*T11*	2	2	1

Tabla 7.1 Resultado de la medición para los elementos de la figura 7.11

Comparando estos valores con lo que obtuvimos con nuestra medición, podemos ver que algunas de ellas difieren. Sobre todo en aquellas reglas que estan relacionadas con eventos compuestos. Esto se debe a que la gráfica de disparo modificada no puede representar eventos compuestos. Por lo tanto, nuestra medición es más precisa.

7.4. Comentarios Finales

Con el desarrollo de esta tesis fue posible obtener un conjunto de parámetros de medición de complejidad de la interacción de las reglas ECA. La modelación de los mismos sobre la CCPN y la definición formal que obtuvimos, nos permitieron realizar su validación sobre el marco de trabajo para mediciones de software del Dr. Zuse. El resultado de la validación nos permitió realizar nuestras mediciones sobre la escala ordinal.

Aprovechando la representación matemática de la CCPN, desarrollamos algoritmos para obtener el valor de los parámetros de una manera más sencilla.

Implementamos tales algoritmos en el lenguaje de programación Java para que fuera un sistema portable y para mantener la compatibilidad con la interfaz ECAPNSim, ya que los algoritmos que implementamos se agregaron como un Módulo de Complejidad a dicha interfaz.

Para demostrar la utilidad de un sistema como el Módulo de Complejidad es necesario generar diversas pruebas con diferentes bases de reglas ECA. En esta tesis utilizamos sólo dos conjuntos de reglas, sin embargo, ambos conjuntos contienen un número suficiente de reglas para mostrar la aplicación de los parámetros de medición de complejidad.

En ambos casos obtuvimos resultados que permiten centrar la atención en aquellas reglas con una complejidad alta. De esta forma, se puede modificar el diseño de las mismas a fin de encontrar un equilibrio

entre la complejidad y la funcionalidad del conjunto de reglas.

Por otro lado, al realizar la comparación con el trabajo presentado en [10] mostramos que nuestra medición es más exacta puesto que ampliamos la definición de los parámetros de medición para considerar eventos compuestos.

Capítulo 8

Conclusiones

En este capítulo presentamos de forma general los resultados que obtuvimos con el desarrollo de este trabajo de tesis. Asimismo, describimos las actividades que se pueden seguir a fin de mejorar el trabajo.

El capítulo se estructura de la siguiente forma: la sección 8.1 presenta los resultados obtenidos así como las aportaciones principales de la investigación y la sección 8.2 expone las actividades propuestas para perfeccionar los parámetros de medición.

8.1. Resultados Obtenidos

El mantenimiento de la base de reglas ECA en una BDA puede ser difícil de realizar si las reglas interactúan mucho entre sí. Para tratar de resolver este problema se pueden aplicar métricas que permitan establecer criterios de complejidad. Sin embargo, en la literatura no existe un trabajo abundante sobre este tema. El único trabajo relacionado es [10]. En este trabajo, los autores proponen tres parámetros de medición: número de anclas, distancia y potencial de disparo para medir la complejidad de la interacción de las reglas ECA en una BDA. No obstante, este trabajo no considera eventos compuestos ya que utiliza una gráfica de disparo para representar a las reglas ECA.

En esta tesis utilizamos la CCPN para modelar reglas ECA. La CCPN proporciona más información acerca de las reglas e integra una representación gráfica y matemática en un mismo modelo. Estas características nos permitieron extender los parámetros de medición propuestos en [10] para considerar eventos compuestos. Por lo tanto, las mediciones que realizamos son más precisas.

Por otro lado, también verificamos la validez de los parámetros de medición sobre el marco de trabajo del Dr. Zuse. A través de esta validación concluímos que se puede medir sobre el nivel de escala ordinal.

Para mostrar la utilidad práctica de los parámetros de medición, desarrollamos un sistema de software al que denominamos Módulo de Complejidad. Este sistema se basó en la matriz de incidencia de la CCPN que representa al conjunto de reglas que se quiere analizar.

8.2. Trabajo Futuro

Las siguientes actividades muestran las mejoras que se pueden hacer a este trabajo de tesis:

- Refinar los parámetros de medición.

En este aspecto se pueden proponer nuevos parámetros de medición para considerar con mayor detalle a los eventos compuestos que involucran tiempo. La propuesta de los nuevos parámetros se puede basar en un análisis estádístico de la ocurrencia de eventos. Desde luego, el uso que se haga de cada sistema activo es diferente, sin embargo, se puede tratar de realizar un análisis lo más general posible.

Una característica importante en la complejidad de las reglas ECA son las condiciones. Una regla cuya condición es verdadera será menos compleja que aquella regla que tiene que evaluar una condición. Por lo tanto, también se pueden desarrollar parámetros de medición de complejidad para las condiciones.

- Mejorar la implementación del Módulo de Complejidad para que sea más eficiente.

Dado que nuestra medición de los parámetros de complejidad está basada en la matriz de incidencia se puede buscar mejorar la eficiencia de los algoritmos a través del paralelismo. Las operaciones involucradas en el cálculo de los parámetros no dependen unas de otras, por lo tanto, se pueden ejecutar simultáneamente.

- Desarrollar ejemplos con un mayor número de reglas y eventos más complejos.

Para realizar ajustes al Módulo de Complejidad es necesario examinar más bases de reglas. También es necesario que estas bases de reglas involucren eventos compuestos. Por lo tanto, tenemos las opciones de: realizar experimentos y analizar casos de estudio. Para realizar experimentos necesitamos desarrollar ejemplos de bases de reglas para tener un ambiente más controlado. Una vez que hemos verificado la funcionalidad del Módulo de Complejidad podemos analizar bases de reglas de sistemas reales.

Bibliografía

[1] Fenton N. y Pfleeger L., *Software Metrics: a rigorous and practical approach*, PWS Pub., 1997

[2] Zuse H., *A framework of software measurement*, Walter de Gruyter, 1998

[3] F. Roberts. *Measurement Theory with Applications to Decisionmaking, Utility, and the Social Sciences*. Addison Wesley, 1979

[4] Fenton N., Software Measurement: A necessary Scientific Basis. *IEEE Transactions on Software Engineering*, 20(3):199-206 (1994)

[5] Whitmire S., *Object Oriented Design Measurement*. Wiley Computer Publishing, 1997

[6] Zuse H. y Bollman P., Software Metrics: Using Measurement Theory to Describe the Properties and Scales of Static Software Complexity Metrics, *SIGPLAN Notices*, Vol. 24, No. 8 24(1989)8, pp. 23-33

[7] http://irb.cs.tu-berlin.de/~zuse/index.html

[8] Piattini M. and Díaz O. *Advanced Database Technology and Design*, Artech House, 2000

[9] Silberschatz A., *Fundamentos de Bases de Datos*, Mc Graw Hill, 1998

[10] Díaz O., Piattini M. and Calero C. Measuring Triggering-Interaction Complexity on Active Databases, *Information Systems*, vol. 26, 2001, pp 15-34

[11] Medina J., *Red de Petri Coloreada Condicional (CCPN) y su aplicación en bases de datos activas*, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México. Septiembre de 2002

[12] Krantz D., *Foundations of measurement*, Academic Press, 1971

[13] Paton N. and Díaz O. Active Database System. *ACM Computer Surveys*, 31(1):63-103 (1999)

[14] Paton N., *Active Rules in Database Systems*, Springer, 1999

[15] Baralis E. y Widom J. .An algebraic approach to static analysis of active database rules", *ACM Transactions On Database System*, vol. 25, no. 3, September 2000, pp. 269-332

[16] Widom J. and Ceri S., *Active Database Systems*, Morgan Kaufmann Publishers, 1996

[17] Hanson E. *The Ariel Project* in Active Database Systems Triggers and Rules For Advanced Database Processing, J. Widom, Ed. Morgan Kaufmann Publishers, San Francisco, California, 1996

[18] Kulkarni, K., Mattos N. and Cochrane R., *Active database features in SQL3,* in Active Database Systems, N. Paton, Ed. Springer Verlag, Berlin, Germany, 1998

[19] Dayal U., Buchmann A. P., and Chakravarthy S., *The HiPAC Project* in Active Database Systems Triggers and Rules For Advanced Database Processing, J. Widom, Ed. Morgan Kaufmann Publishers, San Francisco, California, 1996

[20] Ceri, S., Fraternali, P., *et al*, *Active Rule Management in Chimera*, in Active Database Systems Triggers and Rules For Advanced Database Processing, J. Widom, Ed. Morgan Kaufmann Publishers, San Francisco, California, 1996

[21] Gehani N. and Jagadish H. V., *Active Database Facilities in Ode* in Active Database Systems Triggers and Rules For Advanced Database Processing, J. Widom, Ed. Morgan Kaufmann Publishers, San Francisco, California, 1996

[22] Murata T., Petri Nets: Properties, analisys and applications, *Proceedings of the IEEE*, 77(4):541-580, 1989

[23] David R. and Alla H., Petri Nets for Modeling of Dynamic Systems - A Survey, *Automatica*, Vol. 30, No. 2, pp.175 - 202, 1994

[24] Gatziu, S. y Dittrich R. K., *SAMOS.* In Active Database Systems, N. Paton, Ed. Springer Verlag, Berlin, Germany, 1998

[25] Collet, C., *"NAOS"*, in Active Database Systems, N. Paton, Ed. Springer Verlag, Berlin, Germany, 1998

[26] Potamianos, S. y Stonebraker, M., *The Postgres Rule System*, in Active Database Systems Triggers and Rules For Advanced Database Processing, J. Widom, Ed. Morgan Kaufmann Publishers, San Francisco, California, 1996

[27] Widom, J., *"The Starburst Rule System"*, in Active Database Systems Triggers and Rules For Advanced Database Processing, J. Widom, Ed. Morgan Kaufmann Publishers, San Francisco, California, 1996

[28] Eckel B., Thinking in Java, 2nd Edition, Prentice Hall, June 2000

[29] Date, C. J., *An Introduction to Database Systems*, Addison-Wesley, Sixth Edition, System Programming Series, Reading, (MA) 1995

[30] www.daimi.au.dk/PetriNets/tools/quick.html

[31] Piattini M., Calero C. and Genero M., Table Oriented Metrics for Relational Databases, *Software Quality Journal*, vol. 9, 2001, pp 79-97

[32] Elmasri R. and Navathe S., *Sistemas de base de datos*, Addison-Wesley, 1997

[33] Zuse H., *Software Complexity Metrics/Analysis* in Encyclopedia of Software Engineering, Vol. I, Marciniak John, Ed. John Wiley and Sons, 1994

[34] Zuse H., *Validation of Measures and Prediction Models*, International Workshop on Software Measurement (IWSM'99) - September 8-10, 1999 Lac Supérieur, Canadá

[35] Briand L., El Emam K. and Morasca S., On the Application of Measurement Theory in Software Engineering, International Software Ingeneering Research Network technical report #ISERN-95-04

[36] Briand L. and Morasca S., Property-Based Software Engineering Measurement, *IEEE Transactions on Software Engineering*, Vol. 2, No. 1, January 1996

[37] Baralis E., *Rule Analysis*, in Active Database Systems, N. Paton, Ed. Springer Verlag, Berlin, Germany, 1998

[38] http://alarcos.inf-cr.uclm.es

[39] Weyuker, E., Evaluating Software Complexity Metrics, *IEEE Transactions on Software Engineering*, Vo. 14, Issue 9, pp. 1357-1365, September 1988

[40] Guisheng Y., Qun L., Jianpei Z., Jie L., Daxin L., Petri Based Analysis Method For Active Database Rules, *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 1996, pp. 858-863