Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Análisis automático de propiedades de calidad en sistemas distribuidos basados en servicios

Tesis presentada para optar por el título de Doctor de la Universidad de Buenos Aires en el área Ciencias de la Computación

Agustín Eloy Martinez Suñé

Director de tesis: Dr. Carlos Gustavo Lopez Pombo

Consejero de estudios: Dr. Juan Pablo Galeotti

Lugar de trabajo: Laboratorio de Fundamentos y Herramientas para la Ingeniería de Software (LaFHIS), Departamento de Computación (DC) e Instituto UBA-CONICET de Ciencias de la Computación (ICC), Facultad de Ciencias Exactas y Naturales (FCEyN), Universidad de Buenos Aires (UBA).

Buenos Aires, 2023

Análisis automático de propiedades de calidad en sistemas distribuidos basados en servicios

Los sistemas distribuidos basados en servicios responden a paradigmas de cómputo que ponen a los *servicios* como los elementos fundamentales de la arquitectura y del proceso de desarrollo de las aplicaciones de software. Bajo estos paradigmas, entre los que se encuentra la Computación Orientada a Servicios (SOC), los sistemas de software ya no son piezas monolíticas de código que se ejecutan dentro de los límites de una organización. En cambio, están constituidos a partir de la ejecución coordinada y distribuida de múltiples servicios que proveen funcionalidades específicas. Estos sistemas se sustentan sobre lo que se conoce como *cómputo global*, que comprende los recursos computacionales e infraestructuras de comunicación disponibles a nivel mundial, y dan lugar a la llamada *economía de las API*. Esta idea sostiene que es posible construir artefactos de software a partir de componer, en tiempo de ejecución, servicios provistos por terceros y registrados previamente en repositorios.

En este contexto, la *calidad de servicio* (QoS) es un aspecto fundamental a tener en cuenta en el proceso de desarrollo y despliegue de aplicaciones de software, ya que está directamente influenciada por la calidad de los servicios que componen el sistema. Por lo tanto, los aspectos de calidad se convierten en elementos centrales en la selección de dichos servicios y, como resultado, en la negociación de *acuerdos de nivel de servicio* (SLA). El término calidad de servicio se utiliza de manera amplia en este contexto para referirse a lo que comúnmente se conoce como *requisitos de calidad*, en contraposición a los *requisitos funcionales*. Esto significa que se refiere a aquellos requisitos que expresan propiedades de calidad del software sin condicionar la funcionalidad que proporciona.

En esta tesis, nuestro objetivo es contribuir al campo del análisis automático de propiedades de calidad en sistemas distribuidos basados en servicios. Nuestras contribuciones se basan en el enfoque de los *métodos formales* aplicados a la ingeniería de software, es decir, métodos de análisis de artefactos de software cuyas técnicas pueden explicarse en términos lógico-matemáticos. Desde este enfoque, nos concentraremos en el análisis de propiedades que predican sobre *atributos cuantitativos*, esto es, aquellos atributos de calidad que admiten algún tipo de medición. Este conjunto incluye atributos de naturaleza diversa, como el tiempo de respuesta, el consumo de memoria, el costo monetario y la reputación, entre otros. En suma, las contribuciones de esta tesis se centran en dos aspectos fundamentales. Por un lado, en el desarrollo de lenguajes formales capaces de expresar contratos de calidad de servicio, tanto de provisión como de requisito, desde una mirada general que no hace suposiciones sobre la naturaleza de los atributos de calidad, excepto por la condición de que sean cuantitativos. Por otro lado, en el diseño de técnicas de análisis para estos lenguajes, que permiten la verificación automática de la compatibilidad entre un contrato de provisión y uno de requisito.

Palabras claves: Sistemas distribuidos, Computación Orientada a Servicios (SOC), Calidad de Servicio (QoS), Acuerdos de nivel de servicio (SLA), Economía de las API, Métodos formales, Ingeniería de Software.

Automatic analysis of quality properties in service-based distributed systems

Service-based distributed systems are designed based on computing paradigms that prioritize the concept of *services* as the core elements of software application architecture and development processes. These paradigms, such as Service-Oriented Computing (SOC), have transformed software systems from monolithic entities confined within organizational boundaries to dynamic collections of coordinated and distributed services, each providing specific functionalities.

These systems are built upon what is known as *global computing*, encompassing computational resources and communication infrastructures available worldwide, thereby giving rise to the so-called *API economy*. This idea states that it is possible to construct software artifacts by dynamically composing services provided by third parties, if they have been previously registered in repositories.

In this context, the consideration of *quality of service* (QoS) becomes crucial in the software development and deployment processes, as it is directly influenced by the quality of the services comprising the system. Consequently, quality aspects become central elements in the selection of these services and, consequently, in the negotiation of *service level agreements* (SLAs). Here, the term quality of service is used in a broad sense to refer to what is commonly known as *quality requirements*, in contrast to *functional requirements*. This means that it encompasses requirements expressing quality properties of the software without conditioning its functionality.

The goal of this thesis is to contribute to the field of automatic analysis of quality properties in service-based distributed systems. Our contributions are based on the approach of *formal methods* for software engineering, which involve analysis methods for software artifacts whose techniques can be explained in terms of logic and mathematics. With this approach, our focus will be on analyzing properties that predicate over *quantitative attributes*, that is, quality attributes that can be measured. This set encompasses attributes of various natures, such as response time, memory consumption, monetary cost, reputation, and others. In summary, this thesis's contributions focus on two fundamental aspects. On one hand, the development of formal languages capable of expressing quality of service contracts, both for provision and requirement, from a general perspective that does not make assumptions about the nature of the quality attributes, except that they are quantitative. On the other hand, the design of analysis techniques for these languages, enabling the automatic verification of compatibility between a provision contract and a requirements contract.

Keywords: Distributed systems, Service-Oriented Computing (SOC), Quality of Service (QoS), Service Level Agreements (SLA), API economy, Formal methods, Software Engineering.

Agradecimientos

Son muchas las personas que formaron parte de este camino y que me ayudaron a llegar hasta aquí. Seguramente no mencione a todas ellas, pero mi agradecimiento es para todos los que de alguna manera me acompañaron en este proceso.

Este logro no hubiera sido posible sin la mentoría, no solo académica sino también humana, de Charlie, quien comenzó siendo mi director de tesis y terminó siendo un amigo. Gracias por formarme como investigador y por estar siempre pendiente de abrirme todas las puertas que necesité para avanzar en mi carrera profesional.

Gracias a toda mi familia. A mi vieja, que siempre me apoyó en todo lo que hice sin cuestionar el camino que elegí. A mi viejo, quien fue uno de los impulsores iniciales para que me interesara por la computación como disciplina científica y que siempre me alentó a seguir adelante. A mis hermanos, por todos los momentos que compartidos y todas las charlas que tuvimos.

Gracias a Mai, mi compañera de vida, por apoyarme y creer en mí, incluso en los momentos en que yo no creía en mí mismo. Gracias por ser mi compañera de aventuras y por estar siempre a mi lado.

Gracias a mis amigos de la Facultad, a mis amigos de LaFHIS y a mis amigos de toda la vida. Gracias por estar ahí para tomar una birra y charlar de la vida.

Gracias a Emilio por su disciplina y dedicación, que me ayudaron a llevar mi tesis a un nivel de calidad que me enorgullece. Gracias por abrirme las puertas de tu casa y ayudarme a dar mis primeros pasos en la difusión internacional de mi trabajo.

Gracias a mi consejero de estudios, Juan Pablo, quien estuvo disponible para escucharme y aconsejarme en los momentos que lo necesité.

Gracias a los miembros del jurado por aceptar evaluar mi tesis y por sus comentarios y sugerencias, que me ayudaron a mejorarla.

Finalmente, quiero agradecer a las políticas públicas del Estado Argentino que me permitieron acceder a la educación superior y formarme como investigador en lo que me apasiona.

Índice general

1.. Introducción 1
1.1. Estructura de esta tesis 4

I Análisis formal de atributos de calidad 5

2.. Problema de selección de servicios 6
2.1. Posicionamiento dentro del estado del arte 7

3.. Especificacion formal de atributos de calidad 11

4.. Especificaciones convexas para describir formalmente atributos cuantitativos . . 16
4.1. Contratos de calidad como especificaciones convexas 16
4.2. Resultados experimentales 22
4.3. Conclusiones 30

5.. Satisfacción parcial de requisitos aplicada al rankeo de servicios 32
5.1. La *tasa de inclusión* como medio para cuantificar el cumplimiento parcial de contratos de calidad 33
5.2. Resultados experimentales 37
5.3. Conclusiones y trabajo futuro 44

6.. Descripción y análisis de contratos de calidad probabilísticos 46
6.1. Nuestro enfoque 46
6.2. Acuerdo de nivel de servicio (SLA) para atributos probabilísticos 48
6.3. Análisis automático de satisfacción de requisitos de calidad 52
6.4. Construyendo automáticamente contratos de calidad a partir de datos . . . 56
6.5. Conclusiones 65

II Propiedades de calidad en modelos de sistemas distribuidos 67

7.. Hacia una visión de los servicios con estructura de cómputo explícita 68

8.. Análisis de propiedades de calidad en modelos coreográficos de composición asincrónica 70
8.1. Trabajo relacionado 71
8.2. Preliminares 72
8.3. Especificaciones de calidad de sistemas de comunicación 74
8.4. $\mathcal{QL}$: Una lógica dinámica para propiedades de calidad 76
8.5. Un procedimiento de semidecisión para $\mathcal{QL}$ 79

8.6. Consideraciones sobre modelos de composición sincrónica 84
8.7. Conclusiones 84

9.. **MoCheQoS**: un verificador acotado de modelos para calidad de servicio 86
9.1. Trabajo relacionado 86
9.2. Arquitectura y funcionamiento de **MoCheQoS** 87
9.3. Evaluación experimental 92
9.4. Conclusiones y trabajo futuro 97

10..Conclusiones generales 99

Apéndices 115
A. Cuerpos real-cerrados 115
B. Semántica de G-coreografías basada en pomsets 117
C. Evaluación experimental de **MoCheQoS** 119

1

Introducción

Son múltiples las líneas de desarrollo científico-tecnológico, teóricas y aplicadas, que contribuyeron a la computación como la conocemos hoy en día. Si nos restringimos a las maneras en que se concibe, desarrolla y distribuye software, un recorte particularmente útil es observar el recorrido de los conceptos de modularidad y abstracción desde el surgimiento de los lenguajes de programación estructurados [131, Capítulos 8, 9] hasta el presente. Tal recorrido fue respondiendo a la necesidad de manejar satisfactoriamente la creciente complejidad de los sistemas de software a partir de descomponerlos en partes más pequeñas (modularizar) y posibilitar el uso de esas partes sólo a través de su interfaz pública sin preocuparse por sus detalles de implementación (abstraer) [101]. Por ejemplo, un hito en este recorrido fue la introducción de primitivas en los lenguajes de programación para definir *tipos abstractos de datos* [102]. Desde esta mirada, el paradigma de *Computación Orientada a Servicios* – SOC se puede entender como un hito mucho más reciente en ese recorrido. La definición que proponen los autores en [87] echa luz sobre el paso de abstracción que supuso la introducción de este paradigma:

> SOA aims to abstract by concentrating on the business aspects of a problem. The fundamental term introduced here is service. In essence, a service is an IT representation of some business functionality. The goal of SOA is to structure large distributed systems based on the abstractions of business rules and functions.

Una definición complementaria, presentada en [20], pone el enfásis en el rol que cumplen los servicios como componentes elementales en la construcción de este tipo de sistemas:

> Service-Oriented Computing (SOC) is the computing paradigm that utilizes software services as fundamental elements for developing and deploying distributed software applications. Services are self-describing, platform-agnostic computational elements that support rapid, low-cost composition of distributed applications.

Los sistemas distribuidos basados en servicios responden a este tipo de paradigmas de cómputo, que ponen a los servicios como los elementos fundamentales de la arquitectura y del proceso de desarrollo de las aplicaciones de software. Bajo estos paradigma, los sistemas de software ya no son piezas monolíticas de código que se ejecutan dentro de los límites de una organización. En cambio, están constituídos, por un lado, por aplicaciones que manejan la lógica de negocios buscada por la organización y, por otro lado, por servicios que proveen funcionalidades específicas. De la ejecución coordinada y distribuída de estos diversos componentes emergerá el comportamiento global del sistema [56].

Otro aspecto fundamental que explica el recorrido de las ciencias de la computación hasta el surgimiento de estos paradigmas es el aumento sostenido en el poder de cómputo [121, 117]. Aquello que en su momento fue uno de los principales motivos por los que la complejidad de los sistemas de software se volvió extremadamente difícil de manejar durante la "crisis del software" [46] [1] es hoy el principal habilitador de la computación distribuida y de la computación en la nube. De esta manera, los sistemas de software modernos que responden a estos paradigmas se sostienen sobre lo que se denomina *cómputo global* [57]:

> Global computing refers to computation over "global computers" , i.e., computational infrastructures available globally and able to provide uniform services with variable guarantees for communication, co-operation and mobility, resource usage, security policies and mechanisms, etc., with particular regard to exploiting their universal scale and the programmability of their services.

Es decir, esta nueva generación de aplicaciones se ejecuta sobre los recursos computacionales y la infraestructura de comunicación disponibles globalmente.

Por último, un elemento central de los paradigmas basados en servicios, que los diferencia de otros paradigmas de cómputo distribuido, es la capacidad de reconfigurarse dinámicamente ante nuevas necesidades, aprovechando el ecosistema de servicios disponibles para ser consumidos en tiempo de ejecución. Bajo estos paradigmas, la reconfiguración dinámica se lleva a cabo de manera transparente mediante la intervención de un *middleware* que vincula una aplicación en ejecución con los servicios que cumplen sus requisitos. Esto se logra a través de la negociación de un *Acuerdo de Nivel de Servicio* – SLA, que permite que los servicios cumplan conjuntamente con un objetivo de negocio específico [56].

En este contexto, la *calidad de servicio* (QoS) es un aspecto fundamental a tener en cuenta en el proceso de desarrollo y despliegue de aplicaciones de software, ya que las propiedades de calidad del conjunto del sistema estarán directamente condicionadas por las propiedades de calidad de los servicios que lo componen. Por lo tanto, los aspectos de calidad se convierten en elementos centrales en el proceso de selección dinámica de dichos servicios. Para ser más precisos, entenderemos las propiedades de calidad tomando el concepto de *requisito* del campo de la *ingeniería de software*. Un requisito es una *condición* o capacidad que debe poseer un sistema, o componente, para satisfacer una especificación previamente documentada [83, 127]. En general, la literatura categoriza los requisitos en, al menos, dos grandes tipos: *requisitos funcionales* y *requisitos de calidad* [127, Capítulo 2]. Los primeros, como su nombre lo indica, especifican la funcionalidad que el sistema debe proveer a sus usuarios, es decir, el comportamiento que debe tener ante diversos escenarios y datos de entrada. En cambio, los segundos especifican propiedades de calidad que el sistema debe cumplir sin condicionar la funcionalidad que provee. Eficiencia, mantenibilidad, confiabilidad, costo son algunos ejemplos de propiedades de calidad comunmente utilizadas en el ámbito de la ingeniería de software.

En esta tesis, nuestro objetivo es contribuir al campo del análisis automático de propiedades de calidad en sistemas distribuidos basados en servicios. Nuestras contribuciones se basan en el enfoque de los *métodos formales* aplicados a la ingeniería de software, también enmarcado por algunos autores bajo el término *modelado, análisis y verificación* [95]. En

[1] Crisis que, a su vez, fue uno de los principales catalizadores del surgimiento de la *ingeniería de software* [122] como disciplina.

ámbos casos nos referimos a métodos de análisis de artefactos de software cuyas técnicas pueden explicarse en términos lógico-matemáticos [15]. El rigor matemático de estas técnicas permite brindar garantías fuertes sobre sus resultados. Desde este enfoque, nos concentraremos en el análisis de propiedades que predican sobre *atributos cuantitativos*, esto es, aquellos atributos de calidad que admiten algún tipo de medición. Este conjunto incluye atributos de naturaleza diversa, como el tiempo de respuesta, el consumo de memoria, el costo monetario y la reputación de un servicio, entre otros.

Las principales contribuciones de esta tesis a los objetivos planteados son: (1) una formalización inicial de los contratos de calidad como fórmulas proposicionales cuyos átomos son restricciones convexas sobre atributos cuantitativos, junto con un algoritmo de análisis en dos etapas que permite reducir el tiempo de ejecución al preprocesar los contratos mediante técnicas de optimización convexa, (2) una interpretación geométrica de los contratos de calidad que permite introducir un criterio de satisfacción parcial para construir un ranking de servicios según el grado de satisfacción, el cual puede calcularse utilizando técnicas de geometría computacional, (3) un enfoque para la especificación de contratos de calidad probabilísticos que admite un análisis automático y preciso de la probabilidad de que un contrato satisfaga un requisito dado, y (4) un enfoque para la especificación y análisis de propiedades de calidad para sistemas distribuidos en los que se cuenta con un modelo explícito de su estructura de cómputo, junto con una herramienta que implementa el enfoque propuesto.

Trabajos publicados relevantes

- Lopez Pombo, Carlos G. and Martinez Suñé, Agustín E. and Tuosto, Emilio. *A Dynamic Temporal Logic for Quality of Service in Choreographic Models*. En: ICTAC 2023 (Best Paper Award). Los resultados de este trabajo están contenidos en el Capítulo 8.

- Martinez Suñé, Agustín E. and Lopez Pombo, Carlos G. *Quality of Service Ranking by Quantifying Partial Compliance of Requirements*. En: COORDINATION 2020. Los resultados de este trabajo están contenidos en el Capítulo 5.

- Martinez Suñé, Agustín E. and Lopez Pombo, Carlos G. *Automatic Quality-of-Service Evaluation in Service-Oriented Computing*. En: COORDINATION 2019. Los resultados de este trabajo están contenidos en el Capítulo 4.

- Martinez Suñé, Agustín E. *Formal Quality of Service Analysis in the Service Selection Problem*. En: ICSOC 2020 Workshops. En este trabajo se presentó una visión general del proyecto de investigación abordado en esta tesis.

- Martinez Suñé, Agustín E. *Formalization and Analysis of Quantitative Attributes of Distributed Systems*. En: ICSE '20 Companion Proceedings. En este trabajo se presentó una visión general del proyecto de investigación abordado en esta tesis.

- Agustín E. Martinez Suñé y Carlos G Lopez Pombo. *Probabilistic Quality of Service Aware Service Selection*. Mayo de 2022. DOI: 10.48550/arXiv.2205.09877. arXiv: 2205.09877 [cs] Los resultados de este trabajo están contenidos en el Capítulo 6.

- Carlos G. Lopez Pombo, Agustín E. Martinez Suñé y Emilio Tuosto. *MoCheQoS: Automated Analysis of Quality of Service Properties of Communicating Systems*.

Nov. de 2023. DOI: 10.48550/arXiv.2311.01415. arXiv: 2311.01415 [cs] Los resultados de este trabajo están contenidos en el Capítulo 9.

1.1. Estructura de esta tesis

Esta tesis se encuentra dividida en dos partes. La primera de ellas dedicada a la especificación y el análisis de atributos de calidad en el contexto de paradigmas orientados a servicios donde se conoce a los servicios únicamente por sus propiedades de calidad. La segunda dedicada a la especificación y el análisis de atributos de calidad para sistemas distribuidos en los que se cuenta con la estructura de cómputo de todos los servicios que componen el sistema.

La Parte I comienza en el Capítulo 2 con una introducción al problema de *selección de servicios*, que será el hilo conductor de la primera mitad de la tesis. En el Capítulo 3 brindamos un marco formal inicial para la resolución de este problema, en el que presentamos una primera formalización de los contratos de calidad junto con un algoritmo de decisión asociado. En los capítulos siguientes, nos alejaremos de este marco inicial al introducir diferentes supuestos que nos permitirán abordar el problema de selección de servicios desde diversas perspectivas. En el Capítulo 4 tomamos inspiración en técnicas de verificación de sistemas híbridos para proponer un enfoque basado en restricciones convexas que admite un algoritmo de análisis eficiente. En el Capítulo 5 abordamos el problema de cuantificar el grado de satisfacción parcial de un contrato de requisitos de calidad para los escenarios en que el contrato no se satisface, lo cual a su vez permite dar una respuesta al problema de rankeo de servicios. En el Capítulo 6 nos desprenderemos de la hipótesis implícita de que los valores de los atributos de calidad son equiprobables. Esto da lugar a un enfoque que admite la descripción de contratos de calidad probabilísticos y el análisis automático de la probabilidad de que un contrato satisfaga un requisito dado.

En la Parte II estudiamos la especificación y el análisis de atributos de calidad para sistemas distribuidos en los que contemos con la estructura de cómputo de los componentes de manera explícita. Esto se distingue de los modelos de servicios como cajas negras presentados en la Parte I, en los que se conoce únicamente su perfil de calidad en términos de los valores de sus atributos. Primero introducimos la visión y motivación de esta perspectiva en el Capítulo 7. Luego, en el Capítulo 8 abordamos el problema de analizar propiedades de calidad de servicio en modelos de sistemas distribuidos de composición asincrónica basados en intercambio de mensajes. Por último, en el Capítulo 9 presentamos una herramienta que implementa el enfoque propuesto en el Capítulo 8.

Finalmente, en el Capítulo 10 damos un resumen de las contribuciones de esta tesis junto con conclusiones y posibles líneas de trabajo futuro. Se incluye material adicional en los apéndices.

Parte I:

Análisis formal de atributos de calidad

2

Problema de selección de servicios

Como mencionamos anteriormente, una de las piezas fundamentales de cualquier modelo de cómputo basado en servicios es la determinación de un acuerdo de nivel de servicio (SLA) [26, 45, 44, 56] entre el consumidor de servicios y el proveedor. La posibilidad de automatizar el SLA es lo que sustenta la visión a largo plazo impulsada por los paradigmas basados en servicios. En esta visión, los proveedores ofrecen sus servicios a través de un mercado global, donde los consumidores de servicios pueden elegir entre múltiples opciones para satisfacer sus necesidades. A lo largo de los próximos capítulos desarrollaremos nuestras contribuciones a esta visión, centrándonos en el problema de selección de servicios [20, pt. II], que definimos a continuación.

Definición 1 (Problema de selección de servicios)**.** *Sea un conjunto* $S = \{Pr_1, Pr_2, \ldots, Pr_n\}$ *de* contratos de provisión *que describen las propiedades de calidad ofrecidas por cada uno de los* n *servicios disponibles, y* Rq *un* contrato de requisitos *que describe las propiedades de calidad que la aplicación cliente solicita. El problema de selección de servicios consiste en determinar automáticamente cuál o cuáles* $Pr \in S$ *se adaptan mejor a las necesidades del cliente establecidas en* Rq*.*

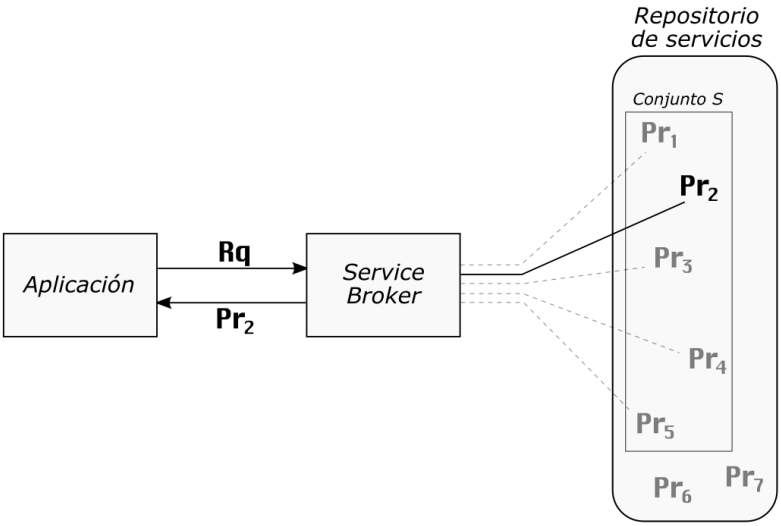


Fig. 2.1: Una vista esquemática del problema de selección de servicios.

En la Figura 2.1 se muestra una vista esquemática del problema. En esta, el mercado de servicios está representado por un *repositorio de servicios* en el que están registrados todos los servicios disponibles para ser consumidos, junto con sus contratos de provisión. El conjunto S del que se realizará la selección está ejemplificado por los servicios $\{Pr_1, Pr_2, Pr_3, Pr_4, Pr_5\}$, dado que todos ellos son compatibles con la aplicación cliente en términos funcionales y de interoperabilidad, solo diferenciándose por sus propiedades de calidad. El *service broker* cumple la función de intermediario entre la aplicación cliente y el repositorio de servicios, y es el encargado de efectuar el proceso de selección de servicio. Recibe el contrato de requisitos Rq de la aplicación cliente y, luego de analizar el conjunto S, determina qué servicio es el más adecuado para satisfacer las necesidades del cliente. En la Figura 2.1, el servicio elegido está ejemplificado por Pr_2. A partir de esta selección, la aplicación cliente puede proceder a consumir el servicio Pr_2, usualmente a través de una *Application Programming Interface (API)* que el proveedor pone a disposición de los clientes.

Como mencionamos en el Capítulo 1, nuestras contribuciones se basan en el enfoque de los métodos formales aplicados a la ingeniería de software. Desde esta perspectiva, el problema de selección de servicios se reduce al problema de verificar para qué servicios $Pr \in S$ se cumple que $Pr \models Rq$, donde $\models$ es una relación de satisfacción formal entre contratos de provisión y contratos de requisitos. Esto requiere, por un lado, modelar formalmente los contratos Pr y Rq y, por otro lado, definir un procedimiento de análisis que implemente la relación de satisfacción $\models$. Como veremos más adelante, este segundo punto a su vez requerirá del uso de técnicas del área de *razonamiento automático* [73] para llegar a un veredicto de manera automatizada. A partir de esta formulación del problema, surgen las siguientes preguntas que guiarán las constribuciones presentadas en esta Parte I:

- ¿Cuál es el lenguaje formal adecuado para modelar los contratos de provisión Pr y de requisitos Rq?
- ¿Cuál es el procedimiento de análisis adecuado para encontrar automáticamente el servicio $Pr \in S$ correcto? ¿Se puede realizar esta selección de manera computacionalmente eficiente?
- ¿Quién es responsable de suministrar los contratos de provisión Pr y de requisitos Rq? ¿De qué manera se construyen estos contratos?

Exploraremos estas preguntas desde distintos ángulos, intentando capturar familias de propiedades de calidad relevantes a ser analizadas, mientras que procuraremos mantenernos dentro de lo que es posible analizar de manera automática. Además del tratamiento formal de los atributos de calidad mantendremos un enfoque práctico en el que para cada una de nuestras propuestas brindaremos un algoritmo de análisis, una implementación prototipo de dicho algoritmo y una evaluación experimental del mismo.

2.1. Posicionamiento dentro del estado del arte

El análisis de propiedades funcionales y de interoperabilidad que permite determinar, dado un repositorio de servicios, cuál es el conjunto de los que son compatibles con la aplicación cliente pero que solo difieren en sus propiedades de calidad es un problema en sí mismo y ha sido ampliamente estudiado en la literatura [9, 30, 59, 148]. A diferencia de estos trabajos, en esta tesis asumiremos que este conjunto, denominado S en la Definición 1

y la Figura 2.1, ya ha sido determinado y nos concentraremos en la selección de servicios basada en propiedades de calidad.

Requisitos de calidad Se ha discutido extensamente sobre lo que constituyen un requisito no funcional y un requisito de calidad, tanto en el contexto general de la ingeniería de software como en el contexto específico de los paradigmas basados en servicios [66, 124]. Sin embargo, la mayoría de la investigación en la selección de servicios se ha centrado en atributos de calidad de servicio que admiten algún tipo de medición [118, 74]. Seguimos este supuesto, ya que es una consecuencia de tener como objetivos el tratamiento formal y el análisis automático y, de acuerdo con la clasificación de requisitos presentada en [66], llamamos a este subconjunto *atributos cuantitativos*. Como bien se indica en [118], este conjunto puede a su vez clasificarse en las siguientes tres categorías:

Categoría	**Ejemplos**
Atributos técnicos independientes del dominio	tiempo de respuesta, disponibilidad, robustez
Atributos no técnicos independientes del dominio	precio/costo monetario, reputación
Atributos dependientes del dominio	tasa de cambio (dominio: finanzas), precisión (dominio: clasificación de imágenes)

Nuestro enfoque es agnóstico de la naturaleza del atributo más allá del hecho de que es cuantitativo y, por lo tanto, será aplicable a las tres categorías anteriores.

Relevancia del problema La relevancia del problema de selección de servicios basado en propiedades de calidad ha sido establecida por distintas comunidades de investigación. La *International Conference on Service Oriented Computing (ICSOC)*, desde su primera edición en 2003 [125] hasta la última edición registrada en 2022 [141], ha consistentemente publicado trabajos que abordan el problema de modelado de atributos de calidad o el problema de selección de servicios. En esta misma sintonía, el libro *Web Services Foundations* [20], publicado en 2014 por investigadores de esa comunidad, lo presenta como un tema central en *Part II: Service Selection and Assisted Composition*. Por otro lado, el proyecto europeo *SENSORIA* que reunió investigadores de 13 universidades durante el período 2005-2010, colocó a la determinación de acuerdos de nivel de servicio y el análisis de propiedades de calidad de servicio como desafíos de primer orden en sus esfuerzos de investigación [151].

Comparación con la literatura existente Vamos a utilizar la revisión sistemática de la literatura realizada en [74] y la revisión comparativa de enfoques para abordar el problema de selección de servicios realizada en [118] para comparar nuestras contribuciones con el estado del arte. En [74] los autores presentan una revisión sistemática que contempla trabajos publicados hasta el año 2017. Específicamente, el estudio se focaliza en el problema selección y composición de servicios basado en propiedades de calidad, en el contexto de la computación en la nube. Los autores observan que, si se clasifican los trabajos según la forma en que estos modelan los atributos de calidad, se encuentran tres categorías: modelos con valores puntuales para cada atributo [29, 153, 152], modelos con múltiples valores para cada atributo, usualmente un segmento de valores entre un mínimo

y un máximo [109], y modelos donde los atributos toman distribuciones de probabilidad [154, 93, 132, 81, 80]. Las contribuciones presentadas en esta tesis se enmarcan en la segunda y tercera categoría, ya que en el Capítulo 4 proponemos un lenguaje que permite modelar el espacio de valores de un conjunto de atributos a partir de utilizar restricciones convexas y, en el Capítulo 6, proponemos un enfoque en el que el espacio de valores de un conjunto de atributos se modela como una distribucion de probabilidad multivariada. En ambos casos, un rasgo distintivo es la capacidad de modelar interdependencias entre los distintos atributos en lugar de tratarlos como entidades independientes. Por otro lado, en esta revisión sistemática se observa que algunos trabajos contemplan la especificación explícita de restricciones de calidad del cliente para guiar el proceso de selección de servicios [149] mientras que otros no lo hacen [78]. En esta tesis nos enfocamos en el primer caso ya que consideramos que las necesidades específicas de la aplicación cliente son un aspecto fundamental del problema. Esto se ve reflejado en el estatus que le damos a los contratos de requisitos Rq en la Definición 1. Finalmente, el estudio observa además que ninguno de los trabajos analizados aborda por sí solo todos los atributos de calidad relevantes para los autores sino que se cada uno se enfoca en algún subconjunto de estos. En cambio, nuestras contribuciones son aplicables a cualquier atributo que admita una interpretación cuantitativa. En [118] los autores presentan una revisión comparativa de distintos enfoques para abordar el problema de selección de servicios, en el contexto de la composición de servicios web. En particular, enmarcan la selección como una de varias etapas en el ciclo de vida de la composición de servicios web. Entre otros elementos, este ciclo de vida involucra los requisitos de calidad de tareas individuales y también los requisitos de calidad de un *business process* [150] global que especifica la forma en que dichas tareas deben componerse. Desde esta perspectiva, el proceso de selección debe asignar un servicio a cada tarea de manera que tanto los requisitos individuales como los globales sean satisfechos. Como se deduce de la Definición 1, esta Parte I se posiciona en la etapa de selección, enfocándose en la asignación de un servicio a una tarea individual. Esto significa que, por un lado, nuestras contribuciones son aplicables de manera directa a la asignación de servicios a tareas individuales contemplando los requisitos de calidad de estas tareas y, por otro lado, podrían ser utilizadas como base para el desarrollo de procesos de selección más complejos que contemplen los requisitos de calidad de un *business process*. Hemos trabajado en esta segunda dirección en [91], donde presentamos una herramienta que utiliza *soft-constraint solving* [115] para determinar automáticamente una asignación de servicios a tareas individuales que maximice la satisfacción de los requisitos de calidad impuestos sobre un *business process* global. Sin embargo, esta línea de investigación queda fuera del alcance de esta tesis. Por otro lado, los autores de la revisión comparativa posicionan los enfoques dentro de un espectro con dos tendencias marcadas. En un extremo, enfoques basados en optimización [1, 52] y, en el otro, enfoques basados en negociación [34, 158]. Los primeros, como su nombre lo indica, modelan el problema como uno de optimización, donde la técnica dominante suele ser asignar un puntaje a cada servicio usando métricas de utilidad. Los enfoques de negociación automatizada se basan en un proceso dinámico de intercambio de información y compromisos entre el proveedor y el cliente. Su objetivo es llegar a un acuerdo que beneficie a ambas partes. Las contribuciones presentadas en esta tesis se diferencian de ambos enfoques. A diferencia de aquellos basados en optimización, no contamos con una función de utilidad sino que son los requisitos de calidad de la aplicación cliente los que guían la selección. A diferencia de los enfoques de negociación, no contemplamos un proceso dinámico de intercambio de información entre el proveedor

y el cliente sino que asumimos que los contratos de provisión y de requisitos están fijos durante todo el proceso de selección. Nuestras contribuciones se podrían enmarcar en lo que los autores llaman "enfoques híbridos", específicamente dentro de los enfoques de "optimización y configuración". Esto es porque los contratos de provisión y de requisitos se interpretan respectivamente como los conjuntos de valores que cada proveedor puede satisfacer y el conjunto de valores que el cliente está dispuesto a aceptar. Esto significa que la información obtenida por el algoritmo de evaluación de los contratos podría ser eventualmente utilizada, luego de la selección, para un último paso de configuración de un acuerdo de valores específicos entre las partes.

Como se mencionó previamente, uno de los puntos distintivos del enfoque impulsado en esta tesis es el tratamiento homogéneo de los atributos de calidad. En lugar de estudiar el comportamiento de atributos de calidad específicos como el *tiempo*, el *consumo de memoria* o la *disponibilidad*, nuestro enfoque no realiza suposiciones sobre los atributos, aparte de que se interpretan como valores. Tal tratamiento homogéneo permite la modelización de interdependencias entre ellos y un tratamiento más holístico de su espacio de valores. A esto se le suma una visión teórico-práctica que se refleja en la implementación de prototipos de los algoritmos de análisis propuestos y en la evaluación experimental de los mismos. No conocemos trabajos previos que compartan estas características y que logren modelar el tipo de propiedades que se logran modelar con las contribuciones presentadas en esta tesis.

3

Especificacion formal de atributos de calidad

Para nosotros, los atributos de calidad son la unidad básica con la que se describen los requisitos de calidad para un servicio. Un atributo de calidad es cualquier aspecto de un artefacto de software que pueda ser medido y que sea relevante para determinar si el artefacto cumple o no con requisitos de calidad de interés para quien lo utiliza. Por esta razón, los números reales se vuelven candidatos naturales para expresar atributos cuantitativos de un artefacto de software. En este capítulo presentaremos una primera formalización de contratos de calidad de servicio como una descripción lógica de los comportamientos esperados de los atributos cuantitativos de un servicio.

Como nuestro objetivo es poder razonar de manera sistemática sobre los contratos de calidad, utilizaremos un lenguaje formal para describirlos. Un sistema lógico se determina generalmente por:

a) Una *gramática* paramétrica en una signatura Σ que declara los símbolos extalógicos rígidos y un conjunto $\mathcal{X}$ que declara los símbolos flexibles que serán utilizados como variables de primer orden, lo que da lugar a un conjunto de fórmulas $Form_{\mathcal{X},\Sigma}$,

b) Una *clase de modelos* formada por: I) una interpretación de los símbolos rígidos de Σ, y II) una valuación $v : \mathcal{X} \to S$ que permite interpretar los símbolos flexibles, donde S es el dominio de discurso de la interpretación de Σ, dando lugar a una clase de modelos $Mod_{\mathcal{X},\Sigma}$,

c) Una *relación de consecuencia* $\vdash \subseteq 2^{Form_{\mathcal{X},\Sigma}} \times Form_{\mathcal{X},\Sigma}$, y

d) Una *relación de satisfacción* $\models \subseteq Mod_{\mathcal{X},\Sigma} \times Form_{\mathcal{X},\Sigma}$.

En particular, debido a su expresividad, utilizaremos el lenguaje de la lógica de primer orden como base para describir los contratos de calidad.

Definición 2 (Lógica de primer orden con igualdad)**.** *Sea $\mathcal{X}$ un conjunto de símbolos de variable, $\Sigma = \langle \mathcal{C}, \mathcal{F}, \mathcal{P} \rangle$ una signatura que declara los símbolos extralógicos rígidos de primer orden para representar constantes, funciones y predicados:*

- *La gramática se define en BNF mediante las siguientes producciones:*

$$
\begin{array}{lcl}
Term_{\mathcal{X},\Sigma} & ::= & x \in \mathcal{X} \mid c \in \mathcal{C} \mid f(Term_{\mathcal{X},\Sigma}, \ldots, Term_{\mathcal{X},\Sigma}),\ con\ f \in \mathcal{F} \\
Form_{\mathcal{X},\Sigma} & ::= & P(Term_{\mathcal{X},\Sigma}, \ldots, Term_{\mathcal{X},\Sigma}),\ con\ P \in \mathcal{P} \mid \\
 & & Term_{\mathcal{X},\Sigma} = Term_{\mathcal{X},\Sigma} \mid \\
 & & \neg Form_{\mathcal{X},\Sigma} \mid Form_{\mathcal{X},\Sigma} \implies Form_{\mathcal{X},\Sigma} \mid \\
 & & (\forall x) Form_{\mathcal{X},\Sigma},\ con\ x \in \mathcal{X}
\end{array}
$$

- *Una Σ-interpretación es una estructura $\mathcal{I} = \langle S, \{\overline{c}\}_{c\in\mathcal{C}}, \{\overline{f}\}_{f\in\mathcal{F}}, \{\overline{p}\}_{p\in\mathcal{P}}\rangle$ tal que: a) para todo $c \in \mathcal{C}$, $\overline{c} \in S$, b) para todo $f \in \mathcal{F}$, $\overline{f} : S^{arity(f)} \to S$, and c) para todo $p \in \mathcal{P}$, $\overline{p} \subseteq S^{arity(p)}$; una valuación es una función $v : \mathcal{X} \to S$, y un Σ-modelo es una estructura $\mathcal{M} = \langle \mathcal{I}, v\rangle$, donde $\mathcal{I}$ es una Σ-interpretación y v es una valuación. El conjunto de Σ-modelos se denota como $Mod_{\mathcal{X},\Sigma}$.*

- *Dados $\Gamma \subseteq Form_{\mathcal{X},\Sigma}$ y $\varphi \in Form_{\mathcal{X},\Sigma}$, la relación clásica de consecuencia para la lógica de primer orden (denotada por $\Gamma \vdash \varphi$) se caracteriza por los axiomas y reglas de cualquier cálculo completo de elección. A modo de ejemplo, reproducimos a continuación el cálculo presentado en [53, Sec. 2.4]. Considere el siguiente conjunto de esquemas axiomáticos:*

  1. *Esquemas tautológicos de la lógica proposicional,*
  2. $(\forall x)\alpha \implies \alpha[x \mapsto t]$, *donde t es un término cerrado,*
  3. $(\forall x)(\alpha \implies \beta) \implies ((\forall x)\alpha \implies (\forall x)\beta)$,
  4. $\alpha \implies (\forall x)\alpha$, *donde x no ocurre libre en α.*
  5. $x = x$,
  6. $x = y \implies (\alpha \implies \alpha')$, *donde α es atómica y α' es el resultado de reemplazar cero o más ocurrencias de x por y en α.*

  El conjunto de axiomas Λ es el conjunto de todas generalizaciones de las fórmulas que se pueden obtener a partir de los esquemas anteriores, donde una generalización de una fórmula α es una fórmula de la forma $(\forall x_1)\ldots(\forall x_n)\ \alpha$ para algún $n \geq 0$. La única regla de inferencia en este caso es la regla de modus ponens:

$$\frac{\alpha \quad \alpha \implies \beta}{\beta}$$

  De esta manera, $\Gamma \vdash \varphi$ si y solo si φ está en la clausura de $\Gamma \cup \Lambda$ bajo la regla de modus ponens.

- *Dados $\mathcal{M} \in Mod_{\mathcal{X},\Sigma}$ y $\varphi \in Form_{\mathcal{X},\Sigma}$, la relación clásica de satisfacción para la lógica de primer orden (denotada por $\mathcal{M} \models \varphi$) caracteriza la noción de que la fórmula φ sea verdadera en $\mathcal{M}$:*

$$\begin{array}{rcl}
\mathcal{M} \models t_1 = t_2 & sii & m_{\mathcal{M}}(t_1) = m_{\mathcal{M}}(t_2), \textit{para } t_1, t_2 \in Term_{\mathcal{X},\Sigma} \\
\mathcal{M} \models p(t_1, \ldots, t_n) & sii & \langle m_{\mathcal{M}}(t_1), \ldots, m_{\mathcal{M}}(t_n)\rangle \in \overline{p}, \textit{para } p \in \mathcal{P} \\
\mathcal{M} \models \neg\alpha & sii & \mathcal{M} \not\models \alpha \\
\mathcal{M} \models \alpha \implies \beta & sii & \mathcal{M} \not\models \alpha \textit{ o } \mathcal{M} \models \beta \\
\mathcal{M} \models (\forall x)\alpha & sii & \textit{para todo } s \in S \textit{ se cumple que } \mathcal{M} \models \alpha[x \mapsto s]
\end{array}$$

  donde $m_{\mathcal{M}} : Term_{\mathcal{X},\Sigma} \to S$ es la función definida inductivamente como:

$$\begin{array}{rcll}
m_{\mathcal{M}}(x) & = & v(x) & , x \in \mathcal{X} \\
m_{\mathcal{M}}(c) & = & \overline{c} & , c \in \mathcal{C} \\
m_{\mathcal{M}}(f(t_1, \ldots, t_n)) & = & \overline{f}(m_{\mathcal{M}}(t_1), \ldots, m_{\mathcal{M}}(t_n)) & , f \in \mathcal{F}
\end{array}$$

Dada una signatura Σ, un conjunto de símbolos de variable $\mathcal{X}$ y $\Gamma \subseteq Form_{\mathcal{X},\Sigma}$, diremos que $\langle \Sigma, \Gamma\rangle$ es una presentación de teoría. La relación de satisfacción entre un modelo y

una presentaciones de teoría se define de la siguiente manera: $\mathcal{M} \models \langle \Sigma, \Gamma \rangle$ *si y solo si* $\mathcal{M} \in Mod_\Sigma$ *y* $\mathcal{M} \models \varphi$ *para todo* $\varphi \in \Gamma$*. La relación de satisfacción entre una presentacion de teoría y una fórmula se define de la siguiente manera:* $\langle \Sigma, \Gamma \rangle \models \varphi$ *si y solo si para todo* $\mathcal{M} \models \langle \Sigma, \Gamma \rangle$ *se cumple que* $\mathcal{M} \models \varphi$*. La relación de consecuencia lógica para presentaciones de teoría se define de la siguiente manera:* $\langle \Sigma, \Gamma \rangle \vdash \varphi$ *si y solo si* $\Gamma \vdash \varphi$*.*

A lo largo de esta tesis, priorizaremos la posibilidad de implementar técnicas automáticas para analizar los contratos de calidad, por lo tanto la lógica que utilicemos debe ser decidible. En consecuencia, como base para los valores de calidad de servicio adoptaremos los *cuerpos real-cerrados* (RCFs), que es una formalización de la teoría de primer orden de los números reales donde las operaciones elementales son la suma y el producto.

Definición 3. *Un* cuerpo real-cerrado F *es un* cuerpo *que es* elementalmente equivalente *a los números reales, es decir, que satisface las mismas propiedades de primer orden que los números reales. Más precisamente, un cuerpo real-cerrado es un modelo de la presentación de teoría* $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle$*, donde* Σ_{RCF} *es la signatura* $\langle \{-1, 0, 1\}, \{+, \cdot\}, \{<\} \rangle$ *y* Γ_{RCF} *es el conjunto de fórmulas de primer orden de la axiomatización de Tarski de los cuerpos real-cerrados (ver Sección A para más detalles o [140, Nota 9] para la presentación original).*

La razón por la que la teoría de los cuerpos real-cerrados resulta particularmente útil para nuestros objetivos es que Tarski demostró la existencia de un procedimiento de decisión para esta [140, Teorema 16], lo que nos permitirá, a su vez, decidir si una especificación de calidad satisface o no un conjunto de requisitos de calidad.

Ahora sí estamos en condiciones de definir una especificación de calidad como una teoría de primer orden que extiende $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle$ con nuevos símbolos de constante que representan los atributos cuantitativos de interés y nuevas fórmulas de primer orden que formalizan las restricciones sobre los mismos.

Definición 4. *Una* especificación de calidad $\langle \Sigma, \Gamma \rangle$ *es una presentación de teoría de primer orden que extiende* $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle$ *de la siguiente manera:*

1. $\Sigma = \langle \{-1, 0, 1\} \cup \mathsf{Q}, \{+, \cdot\}, \{<\} \rangle$*, donde* Q *es un conjunto finito de símbolos de constante (distintos de* -1*,* 0 *y* 1*) representando los* atributos cuantitativos*, y*

2. $\Gamma = \Gamma_{\mathrm{RCF}} \cup \Gamma'$*, donde* Γ' *es un conjunto finito de fórmulas de primer orden declarando condiciones sobre los atributos cuantitativos en* Q*.*

Ejemplo 1 (Especificación formal de calidad)**.** *Consideremos un contexto de una aplicación de software basada en API que requiere un servicio al que se accede y se paga por el tiempo que se utiliza. En ese contexto, consideramos como atributos relevantes el costo y el tiempo de ejecución. El costo está formado por a) perSec: el costo monetario del servicio por segundo transcurrido mientras la sesión de comunicación está abierta, y b) costMb: el costo monetario por megabyte de información procesada. El tiempo de ejecución está formado por a) maxWait: el tiempo máximo de espera en segundos para que el servidor atienda efectivamente una solicitud, b) maxTimeGb: la cantidad máxima de tiempo en segundos que el servicio ejecutará para procesar un gigabyte, y c) netSpeedMb: un límite superior en segundos para transferir un megabyte.*

Entonces, una especificación de dicho servicio se puede formalizar como una presentación de teoría con signatura $\langle \{-1, 0, 1\} \cup \mathsf{Q}, \{+, \cdot\}, \{<\} \rangle$ *donde* Q *es el conjunto de*

símbolos constantes $\{perSec, costMb, maxWait, maxTimexGb, netSpeedMb\}$ *y con axiomas* $\Gamma_{\mathrm{RCF}} \cup \Gamma'$, *donde* Γ' *es:*

$$\begin{array}{l} 0 < maxWait \leq 100 \\ 1000 \leq maxTimexGb \leq 3000 \\ 0{,}1 \leq perSec \leq 0{,}3 \\ 0{,}5 \cdot (perSec/netSpeedMb) \leq costMb \leq perSec/netSpeedMb \\ 0{,}05 \leq netSpeedMb \leq 0{,}15 \end{array}$$

En el contexto del problema de selección de servicios (ver Definición 1), un contrato de requisitos se puede formalizar como un conjunto de formulas Δ sobre la signatura Σ, estableciendo propiedades que deben cumplir los atributos cuantitativos en Q para que el servicio sea aceptado.

Ejemplo 2 (Contrato de requisitos)**.** *Continuando con el escenario planteado en Ejemplo 1, un contrato de requisitos podría ser el siguiente:*

$$(maxWait + 500 \cdot maxTimexGb/1024 < 2000) \wedge \tag{3.1}$$

$$(500 \cdot costMb \leq 4000) \wedge \tag{3.2}$$

$$((maxWait + 500 \cdot netSpeedMb) \cdot 0{,}7 < 500 \cdot maxTimexGb/1024 \cdot 0{,}3) \wedge \tag{3.3}$$

$$(perSec \cdot maxWait < 600) \tag{3.4}$$

La subfórmula 3.1 expresa un límite superior para el tiempo requerido para procesar 500 *megabytes; la subfórmula 3.2 expresa que el costo adicional por megabyte no puede superar los* 4000*. La subfórmula 3.3 expresa que el tiempo invertido en la transferencia de los* 500 *megabytes es, como máximo, el* 30 % *del tiempo total (es decir,* $(maxWait + 500 * netSpeedMb) + (500 \cdot maxTimexGb/1024)$*). Finalmente, la subfórmula 3.4 impone un límite superior a la cantidad de dinero pagada mientras se espera a ser atendido.*

El problema de analizar si un servicio con especificación de calidad $\langle \Sigma, \Gamma \rangle$ cumple con requisitos de calidad Δ se reduce a verificar si $\langle \Sigma, \Gamma \rangle \models \varphi$, para todo $\varphi \in \Delta$. Como mencionamos previamente, Tarski dio un procedimiento automático para decidir si una fórmula de primer orden φ es verdadera en la teoría de primer orden de los números reales $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle$. El procedimiento se basa en la transformación de φ en una fórmula equivalente que no tiene cuantificadores y, luego, la transformación de esta en alguna de las dos fórmulas del conjunto $\{0 = 0, 1 = 0\}$ preservando equivalencia semántica. Esto permite decidir automáticamente si $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle \models \varphi$. Si bien no cualquier extensión de $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle$ es decidible, la extensión particular que presentamos en Definición 4 sí lo es. De la siguiente proposicion se desprende un procedimiento para reducir el problema de decisión para especificaciones de calidad a un problema de decisión para cuerpos real-cerrados. En esencia, lo que se hace es construir una fórmula de primer órden en los números reales que codifica la relación de satisfacción entre la especificación de calidad y el conjunto de requisitos. Esto se logra utilizando el símbolo de la implicación y introduciendo un cuantificador universal por cada símbolo de constante en la especificación de calidad.

Proposición 1. *Sea* $\langle \Sigma, \Gamma \rangle$ *una especificación de calidad, donde la signatura* Σ *es* $\langle \{-1, 0, 1\} \cup \mathsf{Q}, \{+, \cdot\}, \{<\} \rangle$ *con* $\mathsf{Q} = \{q_1, \ldots, q_n\}$ *un conjunto finito de símbolos de constante y* $\Gamma = \Gamma_{\mathrm{RCF}} \cup \Gamma'$*. Entonces,*

$$\langle \Sigma, \Gamma \rangle \models \varphi \text{ si y sólo si } \langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle \models (\forall \widehat{q_1}) \ldots (\forall \widehat{q_n}) \bigwedge_{\gamma \in \Gamma'} \widehat{\gamma} \implies \widehat{\varphi}$$

donde $\widehat{\gamma}$ y $\widehat{\varphi}$ son las fórmulas obtenidas luego de reemplazar cada ocurrencia de q_i en γ y φ por $\widehat{q_i}$, respectivamente, y $\widehat{q_1}, \ldots, \widehat{q_n}$ son símbolos de variable no utilizados en Γ' y φ.

Demostración.

$$
\begin{aligned}
& \langle \Sigma, \Gamma \rangle \models \varphi \\
[\ \textit{por definición de } \Gamma\ ] \quad & \textit{sii} \\
& \langle \Sigma,\ \Gamma_{\mathrm{RCF}}\ \cup\ \Gamma' \rangle \models \varphi \\
[\ \textit{ya que } \Gamma' \textit{ es un conjunto finito}\ ] \quad & \textit{sii} \\
& \langle \Sigma,\ \Gamma_{\mathrm{RCF}}\ \cup\ \{ \bigwedge_{\gamma \in \Gamma'} \gamma \} \rangle \models \varphi \\
[\ \textit{por el teorema de la deducción}^{1}\ ] \quad & \textit{sii} \\
& \langle \Sigma,\ \Gamma_{\mathrm{RCF}} \rangle \models \bigwedge_{\gamma \in \Gamma'} \gamma \implies \varphi \\
[\ \textit{por definición de } \models\ ] \quad & \textit{sii} \\
& \mathcal{M} \models \langle \Sigma,\ \Gamma_{\mathrm{RCF}} \rangle \text{ implica } \mathcal{M} \models \bigwedge_{\gamma \in \Gamma'} \gamma \implies \varphi, \textit{ para todo } \mathcal{M} \in \mathit{Mod}_{\Sigma} \\
[\ \textit{ya que ningún } q_i \in \mathsf{Q} \textit{ aparece en } \Gamma_{\mathrm{RCF}}\ ] \quad & \textit{sii} \\
& \mathcal{M}' \models \langle \Sigma_{\mathrm{RCF}},\ \Gamma_{\mathrm{RCF}} \rangle \text{ implica } \mathcal{M}' \models (\forall \widehat{q_1}) \ldots (\forall \widehat{q_n}) \bigwedge_{\gamma \in \Gamma'} \widehat{\gamma} \implies \widehat{\varphi}, \\
& \qquad \textit{para todo } \mathcal{M}' \in \mathit{Mod}_{\Sigma_{\mathrm{RCF}}} \\
[\ \textit{por definición de } \models\ ] \quad & \textit{sii} \\
& \langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle \models (\forall \widehat{q_1}) \ldots (\forall \widehat{q_n}) \bigwedge_{\gamma \in \Gamma'} \widehat{\gamma} \implies \widehat{\varphi}
\end{aligned}
$$

□

En este capítulo hemos presentado una primera formalización de contratos de calidad de servicio como una descripción lógica de los comportamientos esperados de los atributos cuantitativos de un servicio. Esta descripción lógica se basa en el lenguaje de la lógica de primer orden y en la teoría de los cuerpos real-cerrados. La Proposición 1 nos permite reducir el problema de decisión para especificaciones de calidad a un problema de decisión para cuerpos real-cerrados, lo que muestra que el problema de decidir si una especificación de calidad satisface un conjunto de requisitos de calidad es decidible. Esto nos permite sentar las bases para el desarrollo de técnicas automáticas para analizar contratos de calidad de servicio. En el próximo capítulo, tomaremos inspiración en técnicas de verificación para sistemas híbridos para desarrollar un método de verificación eficiente.

[1] El teorema de la deducción establece que si $\Gamma \cup \{\gamma\} \vdash \varphi$ entonces $\Gamma \vdash \gamma \implies \varphi$. Para más detalles, ver [53, Sec. 2.4].

4

Especificaciones convexas para describir formalmente atributos cuantitativos

Identificar y describir formalmente propiedades cuantitativas de un sistema no es una necesidad exclusiva del campo del análisis de propiedades de calidad. También puede encontrarse en otros campos de la ciencia de la computación, como por ejemplo en el análisis de sistemas híbridos [76], donde se busca verificar propiedades de sistemas que exhiben tanto comportamiento discreto como continuo. En general, los aspectos continuos de dichos sistemas se formalizan como restricciones sobre variables que toman sus valores de los números reales. Existe una gran cantidad de técnicas de análisis que se han propuesto para este tipo de sistemas, con la gran mayoría enfocándose en aquellos aspectos que se encuentran dentro del límite de lo que es decidible [77]. Un enfoque relativamente nuevo para el análisis de especificaciones de sistemas híbridos, presentado por Shoukry y otros [138], integra la resolución *Satisfiability Modulo Theories* (SMT) [41] con restricciones convexas [69], bajo el nombre de *Satisfiability Modulo Convex Optimization*(SMC).

En este capítulo desarrollamos un procedimiento eficiente de dos fases para evaluar SLA cuantitativos basado en SMC, pero adaptado para aprovechar el hecho de que los contratos (tanto de provisión Pr como de requisito Rq) pueden ser minimizados en una fase de preprocesamiento, a la que llamamos *Minimización a través de Optimización Convexa (MCO)*, cuando el servicio es registrado en el repositorio. La expectativa es que dicho preprocesamiento pueda producir una ganancia de eficiencia al momento de verificar si se cumple $Pr \models Rq$. Además, proponemos un enfoque en el que la minimización de contratos se puede realizar de manera incremental y preservando la semántica. Esto se vuelve relevante ya que, como veremos, la minimización completa es altamente costosa.

En la Sección 4.1 nos enfocamos en la definición del marco formal que utilizaremos en este capítulo para especificar contratos de calidad y en la presentación nuestra propuesta para su análisis. Se podrá observar que esta formalización se aparta de la presentada en el Capítulo 3 al estar basada en un lenguaje de restricciones convexas. En la Sección 4.2 presentamos el diseño experimental y los resultados que respaldan las respuestas a cada una de nuestras preguntas de investigación. Finalmente, en la Sección 4.3 sacamos algunas conclusiones y señalamos algunas líneas de investigación futuras. Los resultados presentados en este capítulo fueron previamente publicados en [111].

4.1. Contratos de calidad como especificaciones convexas

En [138], Shoukry y otros adoptan las *fórmulas SMC monótonas* como lenguaje de especificación; nos referiremos a estas especificaciones como *especificaciones convexas*. Las

fórmulas SMC monótonas se definen como fórmulas libres de cuantificadores en forma normal conjuntiva, donde las fórmulas atómicas son variables proposicionales o restricciones convexas sobre variables reales.

Definición 5 (Especificación convexa [138]). *Sea $\mathcal{X}$ un conjunto de variables reales y $\mathcal{P}$ un conjunto de variables proposicionales, entonces una* fórmula SMC monótona *es cualquier fórmula que pueda ser producida por la siguiente gramática:*

$$
\begin{array}{rcl}
\mathit{formula} & ::= & \{\mathit{clause}\ \wedge\}^{*}\mathit{clause} \\
\mathit{clause} & ::= & (\{\mathit{literal}\ \vee\}^{*}\mathit{literal}) \\
\mathit{literal} & ::= & p \mid \neg p \mid \top \mid \bot \mid \mathit{conv_constraint}\ ,\ \mathit{donde}\ p \in \mathcal{P} \\
\mathit{conv_constraint} & ::= & \mathit{equation} \mid \mathit{inequality} \\
\mathit{equation} & ::= & f(x_0, \ldots, x_{arity(f)}) = 0 \\
 & & \quad ,\ \mathit{donde}\ f\ \mathit{es\ una\ función\ afín\ y}\ x_0, \ldots, x_{arity(f)} \in \mathcal{X}. \\
\mathit{inequality} & ::= & f(x_0, \ldots, x_{arity(f)})\ \mathit{relation}\ 0 \\
 & & \quad ,\ \mathit{donde}\ f\ \mathit{es\ una\ función\ convexa\ y}\ x_0, \ldots, x_{arity(f)} \in \mathcal{X}. \\
\mathit{relation} & ::= & < \ \mid\ \leq
\end{array}
$$

Entonces, una especificación convexa *sobre $\mathcal{X}$ y $\mathcal{P}$ es una tupla $\langle\langle \mathcal{X}, \mathcal{P}\rangle, \alpha\rangle$, donde α es una fórmula SMC monótona sobre $\mathcal{X}$ y $\mathcal{P}$. Denotaremos como $mSMCf(\mathcal{X}, \mathcal{P})$ al conjunto de todas las fórmulas SMC monótonas sobre $\mathcal{X}$ y $\mathcal{P}$.*

En la gramática anterior, *affine_function* y *convex_function* denotan, como se podría suponer, funciones afines y convexas, respectivamente. Las fórmulas SMC monótonas solo admiten restricciones convexas como átomos, a diferencia de las fórmulas SMC genéricas sobre restricciones convexas [123], no se permiten restricciones convexas inversas[1]. Como se menciona en [138], la monotonía (es decir, la falta de negación a las restricciones convexas) es fundamental para garantizar que siempre que una especificación convexa tenga un modelo, este se pueda encontrar resolviendo uno (o más) problemas de optimización convexa. El siguiente ejemplo ilustra una especificación escrita en el lenguaje presentado en la Def. 5.

Ejemplo 3 (Contrato de requisitos). *Consideremos una modificación del Ejemplo 2 en la que los costos pueden cambiar si el sistema que requiere el servicio tiene algún tipo de código promocional. En este nuevo escenario, los atributos pueden ser formalizados por una especificación convexa $\langle\langle \mathcal{X}, \mathcal{P}\rangle, \alpha\rangle$ donde $\mathcal{X} = \{perSec, costMb, maxWait, maxTimexGb, netSpeedMb\}$, $\mathcal{P} = \{promotionMode\}$, y donde α es la conjunción de las siguientes fórmulas[2]:*

$$
\begin{array}{l}
0 < maxWait \leq 100 \\
(1000 \leq maxTimexGb \leq 3000) \wedge (0{,}05 \leq netSpeedMb \leq 0{,}15) \\
promotionMode \implies 0{,}0 \leq perSec < 0{,}1 \\
\neg promotionMode \implies 0{,}1 \leq perSec \leq 0{,}3 \\
costMb < 0{,}5
\end{array}
$$

Por otro lado, los autores definen la *expansión convexa monótona* [138, Def. 3.4] de una especificación convexa como una nueva especificación convexa donde cada restricción convexa en la especificación original se reemplaza por una variable proposicional que la representa.

[1] Nótese que las restricciones convexas lineales podrían admitir negaciones, pero como la negación de una restricción convexa lineal se puede reescribir como una restricción convexa lineal en sí misma, no es necesario ningún tratamiento especial.

[2] No hay impedimento en traducir la especificación a una fórmula que respete la gramática presentada en la Def. 5.

Definición 6 (Expansión convexa monótona [138])**.** *La* expansión convexa monótona *de una especificación convexa* $\langle\langle\mathcal{X},\mathcal{P}\rangle,\alpha\rangle$ *como una nueva especificación convexa* $\langle\langle\mathcal{X},\widehat{\mathcal{P}}\rangle,\widehat{\alpha}\rangle$ *donde:*

- $\widehat{\mathcal{P}} = \mathcal{P}\cup\mathcal{V}$*, donde* $\mathcal{V} = \{v_C \mid C$ *es una restricción convexa en* $\alpha\}$*, y*
- $\widehat{\alpha} = \alpha_B \wedge \left[\bigwedge_{v_C\in\mathcal{V}}(\neg v_C \vee C)\right]$*, donde* $\alpha_B = \alpha\,|^{v_C}_{C}$ *es la* abstracción proposicional *de* α*, es el resultado de sustituir cada ocurrencia de una restricción convexa* C *en la fórmula SMC monótona* α *por la nueva variable proposicional* $v_C \in \mathcal{V}$ *asociada a* C*.*

Cuando resulte conveniente, denotaremos a la expansión convexa monótona de $\langle\langle\mathcal{X},\mathcal{P}\rangle,\alpha\rangle$ *como* $\langle\langle\mathcal{X},\mathcal{P}\cup\mathcal{V}\rangle,\alpha_B\rangle$*, con* $\delta:\mathcal{V}\to$ *conv_constraint la función que mapea cada variable proposicional* $v_C\in\mathcal{V}$ *a su correspondiente restricción convexa* C*.*

Posteriormente, en [138, Prop. 3.5], los autores demuestran que: 1) para cada valuación satisfacible de $\widehat{\alpha}$, existe una valuación satisfacible para α y, si $\widehat{\alpha}$ es insatisfacible, entonces también lo es α, 2) cualquier valuación proposicional satisfacible de α_B reduce el problema de satisfacibilidad de $\widehat{\alpha}$ a una conjunción de restricciones convexas, y 3) el problema de satisfacibilidad de $\widehat{\alpha}$ se reduce a una disyunción finita (una para cada valuación proposicional satisfacible para α_B) de conjunciones finitas de restricciones convexas.

En la Fig. 4.1 presentamos un pseudocódigo del método de análisis propuesto por Shoukry y otros [138], mientras que en la Fig. 4.2 mostramos un diagrama de su ejecución. En términos generales, el método se reduce a enumerar las valuaciones proposicionales satisfacibles $\gamma:\widehat{\mathcal{P}}\to\{0,1\}$ para α_B y usar un *convex constraint solver* para determinar si del conjunto de restricciones convexas $\{C \mid \gamma(v_C)=1\}$ es factible o no. Es decir, determinar si existe una valuación de las variables reales que satisface todas las restricciones convexas en el conjunto o, dicho de otra manera, si el conjunto de restricciones convexas es vacío o no. Para evitar confusiones de terminología, hablaremos de *satisfacibilidad* para referirnos a la satisfacibilidad de una valuación proposicional y de *factibilidad* para referirnos a la factibilidad de un conjunto de restricciones convexas.

Input: especificación convexa $\langle\langle\mathcal{X},\mathcal{P}\rangle,\alpha\rangle$
Result: veredicto de satisfacibilidad de α
Construir α_B, la abstracción proposicional de α
Sea $\mathcal{V}$ el conjunto de variables proposicionales en α_B que corresponden a restricciones convexas en α
Sea δ el mapeo de variables en $\mathcal{V}$ a sus restricciones convexas en α
Inicializar un SAT-solver con α_B
while *el SAT-solver devuelve una valuación satisfacible* γ **do**
  Ejecutar un *convex solver* con $\{\ \delta(v) \mid v\in\mathcal{V}\wedge\gamma(v)=1\ \}$
  if *el* convex solver *indica que el conjunto de restricciones es factible* **then**
    return *verdadero*
  else
    Sea $cert = \{v \mid v\in\mathcal{V}\wedge\gamma(v)=1\}$
    Actualizar el estado del SAT-solver con $\neg(\bigwedge_{v\in minimise(cert)} v)$
return *falso*

Fig. 4.1: Pseudocódigo del procedimiento de análisis SMC

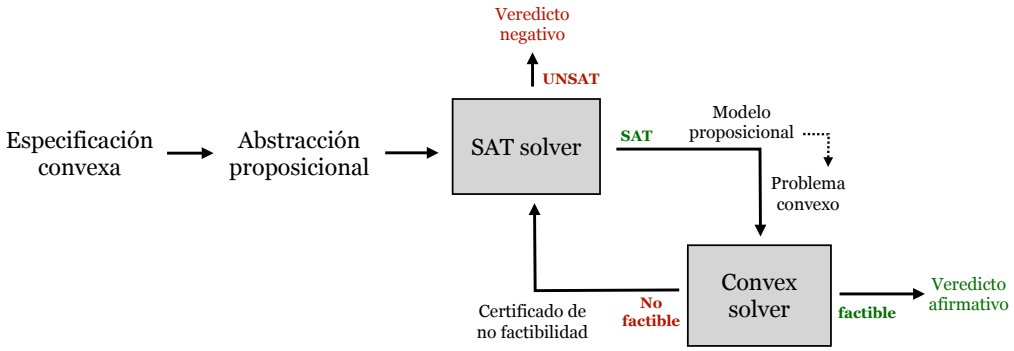


Fig. 4.2: Diagrama del procedimiento de análisis SMC.

Por cada modelo proposicional que se le pide al SAT-solver en la Línea 5, se utiliza un convex solver para determinar si el conjunto de restricciones convexas correspondientes es factible o no. En caso de no ser factible, el certificado de no factibilidad de la Línea 10 se utiliza para incorporar una nueva cláusula en la base de datos del SAT-solver (ver Línea 11) que descarte la valuación proposicional actual y así continuar la enumeración de modelos. La minimización del certificado *cert*, como se utiliza también en la Línea 11, produce un nuevo conjunto de variables $minimise(cert)$. Este es un subconjunto de *cert* que mantiene la no factibilidad de las restricciones convexas correspondientes pero que, además, es minimal en el sentido de que cualquier subconjunto propio de $minimise(cert)$ denota un conjunto factible de restricciones convexas. Es razonablemente evidente que cuanto menor sea la cantidad de variables involucradas en $minimise(cert)$, mayor será la poda del espacio de búsqueda, porque ninguna valuación proposicional que contenga tal asignación parcial se considerará más adelante en la enumeración de valuaciones proposicionales satisfacibles. Volveremos a este punto en la Sección 4.2 donde discutimos algunas notas de implementación. Vale la pena señalar que la minimización de certificados es un problema independiente y ortogonal a la contribución principal de este capítulo sobre la minimización de contratos de calidad.

4.1.1. Hacia una determinación eficiente de SLA

La Computación Orientada a Servicios se basa en una noción de sistema de software como una entidad dinámica construida a partir de servicios descubiertos y vinculados en tiempo de ejecución. Para hacer esto posible, los servicios deben registrarse previamente en repositorios públicos desde donde pueden ser adquiridos por un *Service broker* según lo requerido por la aplicación en ejecución, como esquematizamos previamente en Capítulo 2. Este proceso de reconfiguración dinámica se activa automáticamente cuando una aplicación alcanza un punto en su ejecución en el que se depende exclusivamente de la intervención de un servicio externo para continuar. Consideremos el caso de un contrato (satisfactible) de requisito de calidad Rq. Un servicio con un contrato de provisión Pr será un buen candidato solo si es cierto que $Pr \implies Rq$; o de manera equivalente, si la fórmula $Pr \wedge \neg Rq$ no es satisfactible. Suponiendo que dicha fórmula puede ser vista como una especificación convexa (es decir, una fórmula que puede ser producida por la gramática en la Def. 5), entonces es posible realizar la verificación de interoperabilidad ejecutando el algoritmo de la Fig. 4.1 para determinar el cumplimiento entre el contrato de requisito de

calidad de la aplicación y el contrato de provisión de calidad del servicio. Si bien desde un punto de vista teórico no hay objeción para aplicar este procedimiento para determinar el SLA, desde una perspectiva práctica se espera que esta verificación de interoperabilidad se realice sobre muchos candidatos, para garantizar que el servicio elegido por el broker de servicios sea el adecuado según el estado del repositorio.

Un contexto de uso como este impone fuertes consideraciones de eficiencia sobre este tipo de análisis, y para hacer frente a ellas es necesario reducir el tiempo de ejecución para realizar el análisis tanto como sea posible, incluso a expensas de invertir una cantidad mayor de tiempo en el preprocesamiento de los contratos cuando los servicios se registran en el repositorio. Volviendo a nuestro ejemplo, supongamos que Pr y $\neg Rq$ son especificaciones convexas satisfacibles que denotan un contrato de provisión y la negación de un contrato de requisito, respectivamente; tanto Pr_B como $\neg Rq_B$ podrían tener muchas valuaciones proposicionales satisfacibles γ_{Pr} (resp. $\gamma_{\neg Rq}$) que conducen a conjuntos no factibles de restricciones convexas $\{C \mid \gamma_{Pr}(v_C) = 1\}$ (resp. $\{C \mid \gamma_{\neg Rq}(v_C) = 1\}$), lo que sugiere que los contratos admiten cierta minimización relativa que puede ser realizada actualizando la base de datos de cláusulas del *SAT solver* mediante la inyección de una nueva cláusula que caracterice dicha información de no factibilidad. Idealmente, este proceso de *Minimización a través de la Optimización Convexa* – MCO debe recorrer todo el espacio de valuaciones proposicionales de la abstracción proposicional del contrato, determinando cuáles de ellas conduce a un conjunto factible de restricciones convexas y cuáles no. Esta observación motiva la idea de un enfoque de análisis de dos fases basado en el algoritmo de la Fig. 4.1, en el que la primera fase tiene como objetivo la minimización de los contratos de calidad de requisito y de provisión, realizada solo una vez cuando un servicio se registra en un repositorio, y la segunda fase es el análisis de la conformidad de los contratos de calidad, ejecutado para la negociación del SLA.

Un algoritmo de análisis para especificaciones convexas en dos fases

Nuestra propuesta para el análisis de la conformidad de contratos de calidad está motivada por el escenario de uso específico planteado en Capítulo 2, donde se espera que las especificaciones se reutilicen en muchos análisis. Este contexto impone que el consumo de tiempo para verificar si $Pr \wedge \neg Rq$ es satisfacible debe ser lo más ajustado posible debido al hecho de que dicha verificación debe realizarse sobre cada candidato que sea compatible en términos funcionales. Para hacer frente a esta demanda, diseñamos un preproceso (denominado *fase de minimización*) que tiene como objetivo la minimización de los contratos de calidad, representados por una especificación convexa, y una segunda fase (denominada *fase de verificación*) que implementa el mismo análisis que el algoritmo de la Fig. 4.1.

A continuación presentamos y evaluamos la principal contribución de este capítulo, que es la implementación de un procedimiento de minimización de contratos de calidad, con el objetivo de preprocesar los contratos para podar una parte significativa del espacio de valuaciones proposicionales satisfacibles de su abstracción proposicional. Este procedimiento de minimización está motivado por dos observaciones sobre el algoritmo de la Fig. 4.1: 1) el procedimiento de análisis se basa en enumerar todos los posibles modelos de la abstracción proposicional del contrato de calidad, y 2) la línea 11 del algoritmo de la Fig. 4.1 altera el proceso de enumeración utilizando la información de no factibilidad obtenida del análisis convexo. La Figura 4.3 muestra el pseudocódigo del algoritmo para la minimización de contratos de calidad, mientras que la Figura 4.4 muestra un diagrama

de su ejecución.

Input: especificación convexa $\langle\langle \mathcal{X}, \mathcal{P} \rangle, \alpha \rangle$
Result: especificación convexa minimizada
1 Construir la expansión monótona convexa $\langle\langle \mathcal{X}, \mathcal{P} \cup \mathcal{V} \rangle, \alpha_B \rangle$ (ver Def. 6)
2 Sea δ el mapeo de variables en $\mathcal{V}$ a sus restricciones convexas en α
3 Inicializar un SAT-solver con α_B
4 **foreach** *valuación satisfacible* γ *de* α_B **do**
5 Ejecutar un *convex solver* con $\{ \delta(v) \mid v \in \mathcal{V} \wedge \gamma(v) = 1 \}$
6 **if** *el* convex solver *indica que el conjunto de restricciones NO es factible* **then**
7 Sea $cert = \{v \mid v \in \mathcal{V} \wedge \gamma(v) = 1\}$
8 Añadir la cláusula $\neg(\bigwedge_{v \in minimise(cert)} v)$ a la abstracción proposicional α_B
9 Actualizar el estado del SAT-solver con $\neg(\bigwedge_{v \in minimise(cert)} v)$
10 **return** $\langle\langle \mathcal{X}, \mathcal{P} \cup \mathcal{V} \rangle, \alpha_B \rangle$

Fig. 4.3: Pseudocódigo del algoritmo de *Minimización de a través de Optimización Convexa*

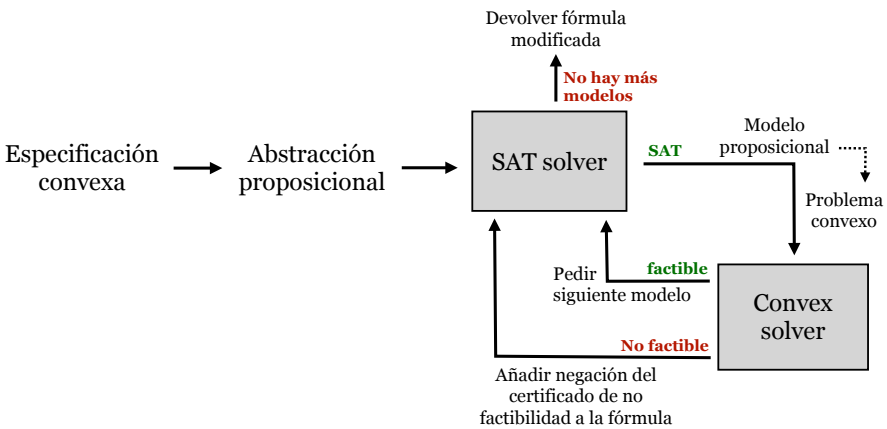


Fig. 4.4: Diagrama del procedimiento de *Minimización de a través de Optimización Convexa*

El algoritmo para la minimización de contratos de calidad se basa en realizar análisis convexo para probar la factibilidad del conjunto de restricciones convexas determinado por cada valuación proposicional satisfacible. La única diferencia con el procedimiento de análisis SMC es que, mientras que en la línea 8 del algoritmo de la Fig. 4.1 se informa que el problema es satisfacible y la búsqueda de más valuaciones proposicionales termina, MCO descarta esa valuación proposicional satisfacible para volver a la enumeración y continuar la búsqueda de valuaciones proposicionales que conduzcan a conjuntos de restricciones no factibles.

Dado un contrato de calidad, que consiste en una fórmula SMC monótona α, se puede notar que el proceso de minimización visita todas las valuaciones proposicionales satisfacibles de la abstracción proposicional α_B para determinar cuáles de ellas deben permanecer como valuaciones satisfacibles, al ser declaradas factibles por el *convex solver*, y cuáles de ellas deben ser eliminadas del espacio de modelos de la especificación. Entonces podemos diseñar un algoritmo de análisis de dos fases considerando: 1) una aplicación única del preproceso de contratos de calidad de la fase MCO, tanto para un contrato de provisión

Pr como para la negación de un contrato de requisito $\neg Rq$, obteniendo $\widehat{Pr}$ y $\widehat{\neg Rq}$ respectivamente, y 2) una segunda fase de búsqueda de una valuación proposicional satisfacible para la abstracción proposicional de $\widehat{Pr} \wedge \widehat{\neg Rq}$ que conduzca a un conjunto factible de restricciones convexas (es decir, el resultado de aplicar el algoritmo mostrado en la Fig. 4.1).

Una preocupación importante con respecto al procedimiento MCO es que, como mencionamos anteriormente, minimizar un contrato de calidad requiere visitar todas las posibles valuaciones proposicionales que satisfacen la abstracción proposicional de cada fórmula Pr y Rq. Es inmediato ver que este procedimiento tiene una naturaleza exponencial debido al hecho de que el tamaño del espacio de valuaciones proposicionales que satisfacen una fórmula booleana es (potencialmente) exponencial con respecto a la cantidad de variables proposicionales[3]. Las mejoras de eficiencia resultantes de posibles optimizaciones del proceso no pueden cambiar este límite de complejidad tan alto. Sin embargo, más adelante en este capítulo propondremos un enfoque incremental como forma de mitigar este problema.

4.2. Resultados experimentales

En esta sección presentaremos nuestra implementación de los algoritmos presentados en la Sección 4.1, presentaremos nuestras preguntas de investigación y discutiremos los resultados experimentales que respaldan las respuestas a las mismas.

A continuación, proporcionaremos algunos detalles sobre la implementación de los algoritmos, la configuración del hardware del entorno experimental y el conjunto de datos utilizado para la evaluación experimental.

4.2.1. Implementación y configuración experimental

Detalles de implementación En esta sección discutiremos brevemente algunos aspectos de la implementación de los algoritmos presentados en la Sección 4.1. La implementación desarrollada por Shoukry y otros en [138] del algoritmo de la Figura 4.1 utiliza Z3 [42] como SAT-solver. Dicho esto, nosotros decidimos desarrollar dos versiones de nuestros algoritmos, una también utilizando Z3 y otra utilizando Minisat [51], un conocido SAT-solver cuya minimalidad lo ha convertido en uno de los más eficientes disponibles. Nuestra implementación utilizando Z3 tiene por objetivo verificar si la reimplementación del algoritmo no introduce ninguna diferencia significativa en el rendimiento. La comprobación de la factibilidad de conjuntos de restricciones convexas se implementó utilizando IBM ILOG CPLEX Optimization Studio [82], ya que es uno de los software de optimización convexa más potentes disponibles para fines de investigación y educación. El algoritmo que orquesta la enumeración de valuaciones proposicionales satisfacibles de la abstracción proposicional de los contratos con la resolución convexa del conjunto de restricciones convexas correspondiente, así como todo lo necesario para la generación y ejecución sistemática de instancias según el diseño experimental, se desarrollaron en Python 2.7 (`https://www.python.org`).

Finalmente, es necesario hacer una observación sobre cómo se obtiene la minimización del certificado de no factibilidad $minimise(cert)$ (ver la Línea 10 del algoritmo de la Figura 4.1 y las líneas correspondientes en la Figura 4.3). En [138, Sec. 5], los autores discuten muchos enfoques para minimizar la cláusula que determina un conjunto de restricciones

[3] Para ser precisos, el problema de enumerar todas las posibles asignaciones que satisfacen una fórmula SAT está en la clase de complejidad **#P-complete** [144].

no factible y señalan la tensión existente entre el costo computacional asociado con la obtención de un certificado de no factibilidad y la longitud del propio certificado. Como se mencionó anteriormente, cuantas menos variables estén involucradas en el certificado, mayor será la poda del espacio de búsqueda y, por lo tanto, se esperará un mejor rendimiento del algoritmo y, en consecuencia, del análisis en general. En la práctica, después de haber implementado los métodos propuestos por los autores, la forma más efectiva y eficiente que encontramos para obtener un certificado de no factibilidad minimal es mediante la obtención del conjunto mínimo de conflictos proporcionado por CPLEX cuando se declara que el problema convexo no es factible, lo cual coincide con la implementación más reciente de SatEx [136], el nombre bajo el cual los autores distribuyen su implementación.

Configuración de hardware Los experimentos se llevaron a cabo en un procesador de arquitectura x86_64, un Intel(R) Core(TM) i5 CPU 750 a 2.67 GHz (CPU MHz: 2661 - CPU max MHz: 2661 - CPU min MHz: 1197) con 3 niveles de caché (caché L1d: 32K, caché L1i: 32K - caché L2: 256K - caché L3: 8192K), 8 GB de SDRAM a 1333 MHz y se ejecutaron en SMP Debian Linux 4.9.88-1+deb9u1 (2018-05-07). Cada instancia individual se ejecutó durante un máximo de 3 horas, a menos que se indique lo contrario, ya que creemos que lo que es factible dentro de ese marco de tiempo proporciona suficiente información para validar nuestras respuestas. Siempre que el análisis de una instancia del problema o la construcción de la infraestructura de resolución excediera el límite de tiempo, la celda correspondiente en las tablas se marcó con "TO", indicando que el proceso alcanzó el tiempo límite, y cuando el límite se debió a la agotamiento de la memoria del sistema que llevó a la máquina a un estado de trashing, se marcó con "OoM", indicando que el sistema se quedó sin memoria.

Notas sobre la configuración experimental Una aclaración necesaria sobre el diseño experimental es que no utilizamos el sistema de experimentación distribuido por los autores de [138] porque en ese trabajo: 1) las especificaciones se construyen *hardcodeando* la instancia del *solver* y no se brinda ninguna interfaz que permita a los usuarios proporcionar una especificación en un lenguaje estándar, lo que dificulta probar la herramienta con diferentes conjuntos de datos, y 2) no se informa sobre el tiempo necesario para construir la instancia del *solver*, lo que, después de evaluar las implementaciones basadas en Z3, hemos visto que no es despreciable debido a que parte de la resolución se realiza durante la adición de las cláusulas, a veces consumiendo más tiempo que la invocación de la resolución en sí misma (ver la columna "Tiempo de inicialización" de la Tabla 4.1).

A continuación, presentamos nuestras preguntas de investigación. Estas tienen como objetivo evaluar el rendimiento de nuestra propuesta de un procedimiento de dos fases para verificar la conformidad de contratos de calidad. Las respuestas serán presentadas en la sección subsiguiente.

4.2.2. Preguntas de investigación

Nuestra primera pregunta de investigación tiene como objetivo relacionar el rendimiento del algoritmo en la Fig. 4.1 tal como está implementado por Shoukry y otros con nuestra reimplementación cuando se iteran las valuaciones proposicionales satisfacibles utilizando, por un lado, Z3 y, por otro lado, Minisat como SAT-solvers.

RQ1: ¿Existe alguna mejora en el rendimiento en nuestra reimplementación basada en Z3 de la fase de verificación (Fig. 4.1) en comparación con la presentada en [138]? ¿Y

cuál es la diferencia entre una implementación basada en Minisat en comparación con la implementación basada en Z3?

Las preguntas de investigación restantes tienen como objetivo evaluar el rendimiento de nuestra propuesta de un procedimiento de dos fases para verificar el cumplimiento del contrato de calidad. Para lograr esto, la comparación de las tres implementaciones diferentes del procedimiento de análisis SMC realizadas para responder a **RQ1** servirá para establecer un *baseline* para la evaluación experimental de la contribución principal de este capítulo, que es la comparación del tiempo requerido para verificar el contrato de calidad después de realizar una fase de *Minimización a través de la Optimización Convexa.*

La segunda pregunta de investigación tiene como objetivo identificar si existe un límite en el tamaño de los contratos de calidad que pueden minimizados de manera completa.

RQ2: ¿Cuál es el tamaño de los contratos de calidad que se pueden minimizar completamente dentro de un límite de 3 horas?

Para la próxima pregunta de investigación, primero introduciremos la noción de minimización parcial de un contrato de calidad. Una inspección detallada del procedimiento de minimización propuesto muestra que, si este se detiene en cualquier iteración justo antes de continuar la enumeración de valuaciones proposicionales satisfacibles, el contrato de calidad resultante comparte todos los modelos factibles con el original.

Proposición 2. *Sea* $\langle\langle\mathcal{X},\mathcal{P}\rangle,\alpha\rangle$ *una especificación convexa y sea* $\langle\langle\mathcal{X},\mathcal{P}\cup\mathcal{V}\rangle,\alpha_B\rangle$ *su expansión monótona convexa (ver Def. 6), donde* $\mathcal{V}$ *es el conjunto de variables proposicionales que corresponden a las restricciones convexas,* α_B *es la abstracción proposicional de* α*, y* δ *la función que asigna variables en* $\mathcal{V}$ *a sus correspondientes restricciones convexas que aparecen en* α*. Sea* $[\alpha_{Bi}]_{0\leq i\leq n}$ *una secuencia de fórmulas proposicionals donde* $\alpha_{B0}=\alpha_B$*,* α_{Bi+1} *es el resultado de una sola iteración del algoritmo de minimización aplicado a* α_{Bi}*, para todo* $i<n-1$*, y* α_{Bn} *es el resultado final del algoritmo.*

Entonces, todas las especificaciones convexas $\langle\langle\mathcal{X},\mathcal{P}\cup\mathcal{V}\rangle,\alpha_{Bi}\rangle$ *comparten los mismos modelos factibles, con* $0\leq i\leq n$*.*

Demostración. Observar que la cantidad de valuaciones proposicionales satisfacibles de α_{Bi+1} solo puede ser menor o igual a la cantidad de valuaciones proposicionales satisfacibles de α_{Bi} como consecuencia de que la actualización realizada sobre la especificación convexa es no decreciente, lo que significa que solo se agregan cláusulas. Supongamos entonces que existe $0<i\leq n$ tal que el algoritmo enumera la valuación proposicional satisfacible γ para α_{Bi} y que α_{Bi+1} tiene menos modelos factibles.

Si α_{Bi+1} tiene menos modelos factibles que α_{Bi}, entonces eso significa que alguno de los modelos eliminados por la cláusula $\neg(\bigwedge_{v\in minimise(cert)} v)$ en la Línea 8 de la Figura 4.3 era factible. Sin embargo, sabemos que el conjunto de restricciones convexas $\{\ \delta(v)\ \mid\ v\in minimise(cert)\ \}$ es no factible ya que *cert* es un certificado de no factibilidad del problema ejecutado en la Línea 6 de la Figura 4.3 y $minimise(cert)$ es un subconjunto de *cert* que preserva la no factibilidad. Cualquier valuaciones proposicionales eliminada por la cláusula $\neg(\bigwedge_{v\in minimise(cert)} v)$ denota una extensión del conjunto de restricciones convexas $\{\ \delta(v)\ \mid\ v\in minimise(cert)\ \}$. Por definición, cualquier extensión de dicho conjunto de restricciones convexas es no factible. Llegando a un absurdo. □

Un hecho interesante derivado de la proposición anterior es que el proceso de minimización del contrato de calidad se puede realizar incrementalmente, una iteración a la vez, lo que conduce a una sucesión de contratos de calidad parcialmente minimizados, lo que nos lleva a nuestra tercera pregunta de investigación.

RQ3: ¿Cómo evoluciona la dificultad computacional del problema cuando se consideran minimizaciones parciales sucesivas de una especificación dada?

4.2.3. Evaluación experimental

En esta sección evaluamos las preguntas de investigación planteadas previamente y presentamos datos experimentales que respaldan nuestras respuestas.

RQ1: ¿Existe alguna mejora en el rendimiento en nuestra reimplementación basada en Z3 de la fase de verificación (Fig. 4.1) en comparación con la presentada en [138]? ¿Y cuál es la diferencia entre una implementación basada en Minisat en comparación con la implementación basada en Z3?

Diseño experimental. Los contratos de provisión se obtuvieron generando primero instancias aleatorias de problemas SAT satisfacibles, Pr_B. El conjunto de datos está formado por contratos con una cantidad de variables proposicionales que van desde 50 hasta 350. El número de cláusulas de cada contrato es 80 veces el número de variables proposicionales. De la cantidad total de variables proposicionales, elegimos aleatoriamente un 50 % al que asociamos restricciones convexas lineales generadas al azar. La cantidad de variables reales en las restricciones convexas va desde 5 a 30. Los contratos $\neg Rq_B$ (satisfacibles) se obtienen de: 1) negar Pr_B, 2) empujar las negaciones a los átomos, y 3) revertir las restricciones lineales (produciendo también restricciones lineales). De esta manera, $Pr_B \wedge \neg Rq_B$ resulta satisfacible desde el punto de vista proposicional, pero no tiene ningún modelo convexo factible. Esta falta de contraejemplos para $Pr \implies Rq$ tiene como objetivo estresar el procedimiento de verificación obligándolo a recorrer todo el espacio de soluciones.

El límite superior en el número de variables proposicionales responde al hecho de que generar una fórmula proposicional en forma normal conjuntiva (CNF) $\neg Pr_B$ como contrato de requisitos de calidad, utilizando la transformación de Tseitin [142], produce una fórmula proposicional cuadráticamente mayor que Pr_B, tanto en el número de cláusulas como de variables. Si consideramos que una abstracción proposicional de un contrato de provisión Pr_B de 400 variables proposicionales y 32 000 cláusulas resulta en un archivo de 35 Mb, la construcción de un contrato de calidad equivalente a $\neg Pr_B$ produce una fórmula CNF de aproximadamente 30 000 variables proposicionales y 9 500 000 cláusulas, consumiendo aproximadamente 1 Gb de almacenamiento. El análisis se realizó sobre 8 instancias de cada combinación de cantidad de variables proposicionales y de variables reales para tratar de paliar la varianza entre los casos. El tiempo se dividió en dos: por un lado el tiempo necesario para configurar la infraestructura de verificación y por otro lado el tiempo de análisis en sí mismo, ya que el uso de Z3 mostró que una parte significativa del análisis tiene lugar durante la inicialización del *SMT-solver* con las cláusulas.

Resultados experimentales. La Tabla 4.1 muestra la comparación de tiempo de ejecución (expresado en segundos) entre la implementación original del algoritmo SMC presentado

en [138], que presentamos en la Fig. 4.1, y nuestras implementaciones basadas en Z3 y en Minisat. El diseño de la tabla es el siguiente: 1) la primera columna muestra la cantidad de variables proposicionales en el contrato de provisión, 2) la segunda muestra el *solver* utilizado, 3) la tercera columna muestra el tiempo requerido para inicializar el solucionador hasta el momento exacto antes de que esté listo para resolver el problema, y 4) las columnas cuarta a nueve muestran el promedio de tiempo en segundos requerido para resolver las instancias según el número de variables reales. La Figura 4.5 muestra gráficos de

# prop. vars en provisión	Solver	Initialisation (s)	# variables reales 5	10	15	20	25	30
50	SMC	150	157.48	176.71	176.71	178.28	176.86	178.22
	Z3 check	170	167.82	177.82	177.55	177.01	176.84	177.55
	Minisat check	4	3.21	2.23	2.13	2.12	2.12	2.12
100	SMC	630	1 575.70	1 802.20	1 756.17	1 812.62	1 854.42	1 852.64
	Z3 check	640	1 589.20	1 819.81	1 843.83	1 954.84	1 925.89	1 931.48
	Minisat check	17	16.47	19.89	11.90	10.01	9.11	8.70
150	SMC	1 680	4 243.10	TO	TO	TO	TO	TO
	Z3 check	1 820	4 580.25	TO	TO	TO	TO	TO
	Minisat check	45	21.32	174.12	79.92	47.76	26.66	23.60
200	SMC	3 100	TO	TO	TO	TO	TO	TO
	Z3 check	3 120	TO	TO	TO	TO	TO	TO
	Minisat check	77	35.53	236.32	906.17	488.88	102.78	60.34
250	SMC	4 920	TO	TO	TO	TO	TO	TO
	Z3 check	5 010	TO	TO	TO	TO	TO	TO
	Minisat check	127	45.93	321.73	1 231.22	2 929.81	1 052.29	405.59
300	SMC	OoM	–	–	–	–	–	–
	Z3 check	7 300	TO	TO	TO	TO	TO	TO
	Minisat check	186	60.05	431.87	1 714.84	4 423.42	8 259.39	4 720.92
350	SMC	OoM	–	–	–	–	–	–
	Z3 check	OoM	–	–	–	–	–	–
	Minisat check	OoM	–	–	–	–	–	–

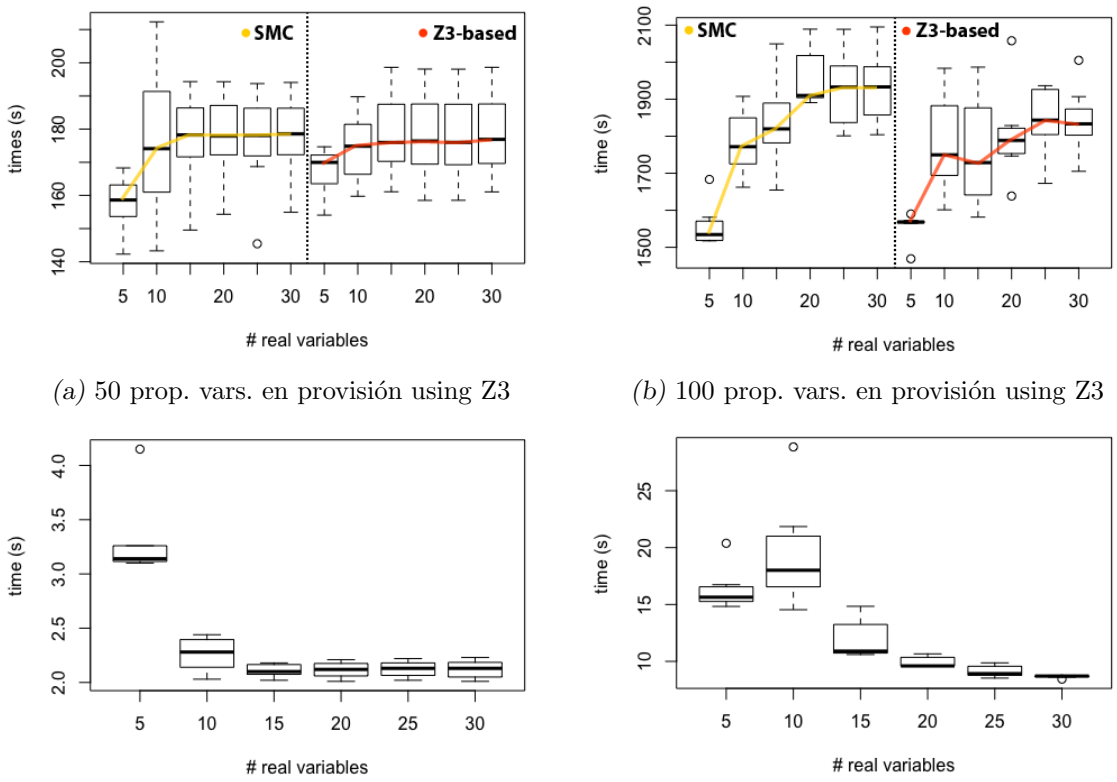
Tab. 4.1: Tiempo de ejecución en segundos de diferentes implementaciones del algoritmo para verificar una especificación convexa no factible.

diagramas de caja (*boxplots*) que contienen información sobre el tiempo de ejecución para las dos primeras filas de la Tabla 4.1, exponiendo la estabilidad relativa de los algoritmos. Las Figuras 4.5a y 4.5b muestran el tiempo de ejecución de la herramienta utilizando la implementación de [138] y nuestra reimplementación basada en Z3. Las Figuras 4.5c y 4.5d muestran el tiempo de ejecución utilizando nuestra reimplementación basada en Minisat.

Discusión. Observando los tiempos de ejecución obtenidos de las tres implementaciones del algoritmo SMC para verificar una fórmula no factible (Tabla 4.1), derivamos las siguientes conclusiones:

1) El tiempo de ejecución crece exponencialmente a una tasa de 2,37 en función del número de variables proposicionales.

2) Observando la evolución del tiempo a lo largo de una fila, es posible apreciar que el tiempo requerido para analizar la satisfacibilidad de las instancias crece hasta que encuentra un máximo, luego disminuye hasta estabilizarse en lo que parece ser un *plateau*. Para cada fila, la fórmula CNF y la cantidad de restricciones convexas utilizadas es la misma, por lo tanto, este fenómeno parece exponer una relación entre el número de restricciones y el número de variables reales sobre las cuales se expresan esas restricciones. También es observable que la disminución y posterior estabilización del tiempo es



(a) 50 prop. vars. en provisión using Z3

(b) 100 prop. vars. en provisión using Z3

(c) 50 prop. vars. en provisión using Minisat

(d) 100 prop. vars. en provisión using Minisat

Fig. 4.5: Análisis de cumplimiento de contratos de calidad sobre 50 y 100 variables proposicionales en el contrato de provision (el tamaño total del problema es de +4000 y +8000 variables proposicionales respectivamente)

acompañada por el número de iteraciones del algoritmo, lo que sugiere que el fenómeno tiene su origen en una relación entre el número de variables reales y la capacidad de poda del certificado de no factibilidad emitido por CPLEX en cada iteración. Dicho esto, creemos que entender este fenómeno particular no tiene un impacto relevante en el experimento realizado para responder a esta pregunta de investigación.

3) No hay una diferencia consistente y significativa entre el rendimiento de la implementación original de SMC de Shoukry et al. y nuestra reimplementación basada en Z3 como *SAT-solver*. Los datos experimentales muestran una diferencia en el tiempo de ejecución no mayor al 10 %. Esta diferencia podría ser una consecuencia de haber desarrollado una implementación más abstracta de la especificación convexa que encapsula un iterador para las valuaciones proposicionales satisfacibles de la abstracción proposicional.

4) La implementación basada en Minisat supera ampliamente a ambas implementaciones basadas en Z3, tomando solo aproximadamente el 2 % del tiempo requerido por estas últimas, considerando la inicialización de la infraestructura y el análisis en conjunto.

Una observación importante sobre la respuesta dada a **RQ1** es que solo la implementación basada en Minisat del algoritmo de la fase de verificación (Figura 4.1) puede producir

una cantidad significativa de datos experimentales útiles para evaluar nuestra propuesta de procedimiento de análisis en dos fases; por lo tanto, **RQ2** y **RQ3** solo se responderán evaluando el rendimiento de la implementación basada en Minisat de ambas fases del procedimiento.

RQ2: ¿Cuál es el tamaño de los contratos de calidad que se pueden minimizar completamente dentro de un límite de 3 horas?

Diseño experimental. El conjunto de datos utilizado para este experimento fue generado con los mismos criterios con los que se generó el conjunto de datos para responder a **RQ1**, pero considerando una granularidad más fina en el eje de variables proposicionales (cada 5) y comenzando desde 30, ya que el tiempo de ejecución de la fase de minimización de instancias con menos de 30 variables fue despreciable. En aquellos casos en los que el proceso de minimización no encontró ningún modelo convexo infactible, y por lo tanto no produjo ninguna minimización, se informa el tiempo de ejecución etiquetado con **(nm)**, indicando "sin minimización".

Resultados experimentales. La Tabla 4.2 muestra el tiempo de ejecución de la fase de minimización (algoritmo de Figura 4.3). Una comparación lado a lado del tiempo requerido

	# variables reales					
# vars. prop.	5	10	15	20	25	30
30	186.68	**(nm)** 200.40	**(nm)** 203.18	**(nm)** 206.43	**(nm)** 209.94	**(nm)** 212.82
35	691.26	**(nm)** 852.93	**(nm)** 921.20	**(nm)** 930.37	**(nm)** 944.27	**(nm)** 953.85
40	5 606.34	**(nm)** 9 519.08	**(nm)** 9 682.34	**(nm)** 9 814.39	**(nm)** 1 0004.27	**(nm)** 7 234.67
45	TO	TO	TO	TO	TO	TO

Tab. 4.2: Tiempo de ejecución en segundos del algoritmo de minimización basado en Minisat

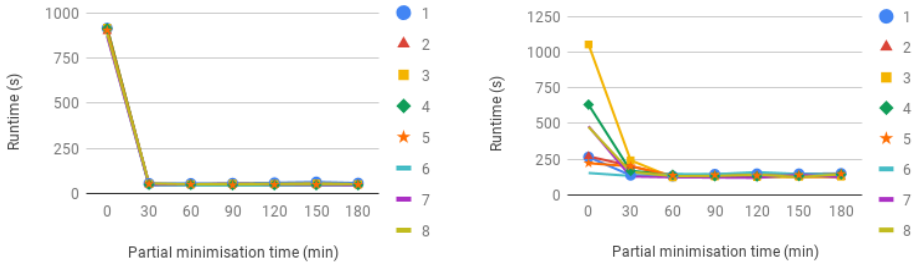
por el algoritmo de análisis, sobre instancias minimizadas y no minimizadas, no es de interés en este caso ya que el costo asociado al proceso de minimización completa de contratos calidad es tan alto que incluso la instancia más pequeña de la Tabla 4.1 de 50 variables proposicionales y 5 variables reales no pudo ser minimizada dentro del límite de tiempo de 3 hs.

Discusión. Los resultados mostrados en la Tabla 4.2 revelan que un enfoque ingenuo para la minimización no es viable, ya que el costo de la minimización completa podría ser demasiado alto, incluso para contratos muy pequeños. Es notable que los datos muestran que las instancias donde la cantidad de variables reales es mayor o igual a 10 no tenían modelos convexos infactibles. Nuestra hipótesis al respecto es que, para estas instancias generadas de manera aleatoria, si queda fija la cantidad de restricciones convexas, la probabilidad de que exista un modelo convexo infactible disminuye a medida que aumenta la cantidad de variables reales. Debido a que en la Tabla 4.2 quisimos mostrar casos de estudio que puedan minimizarse por completo, eso limitó la cantidad de variables proposicionales que fueron consideradas. Sin embargo, como se verá en el siguiente experimento, tomando una cantidad de variables proposicionales más grande, el algoritmo de minimización logra encontrar modelos convexos no factibles.

RQ3: ¿Cómo cambia la naturaleza del problema considerando minimizaciones parciales sucesivas de una especificación dada?

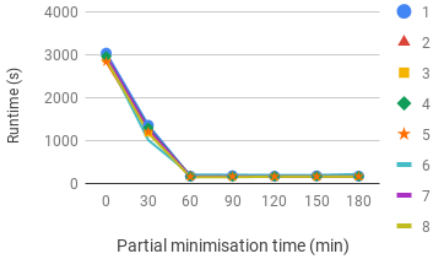
Diseño experimental. Para responder a esta pregunta de investigación, ejecutamos el algoritmo de minimización durante 3 horas y tomamos una instantánea de la minimización parcial obtenida cada 30 minutos. Hacemos esto para cada par de contratos de calidad de las celdas de las filas con cantidad de variables proposicionales 200, 250 y 300 de la Tabla 4.1, donde el número de variables reales varía entre el 5 % y el 10 % del número de variables proposicionales. Luego, llevamos a cabo la fase de verificación para identificar cómo disminuye el tiempo de análisis a medida que se invierte más tiempo en el procedimiento de minimización.

Resultados. La Figura 4.6 muestra gráficos de la evolución del tiempo requerido por la fase de verificación sobre contratos de calidad minimizados parcialmente, considerando las instantáneas tomadas cada 30 minutos de minimización. La tabla 4.3 muestra la compa-

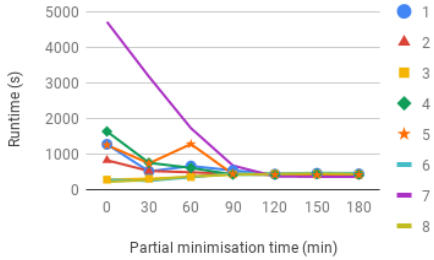


(a) 200 vars. prop. / 15 vars. reales

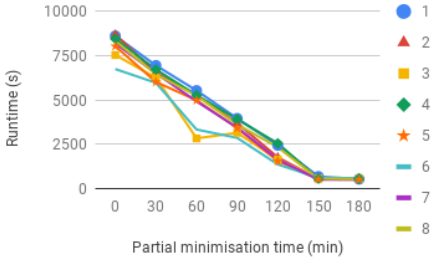
(b) 200 vars. prop. / 20 vars. reales



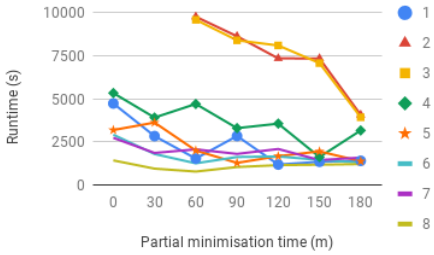
(c) 250 vars. prop. / 20 vars. reales



(d) 250 vars. prop. / 25 vars. reales



(e) 300 vars. prop. / 25 vars. reales



(f) 300 vars. prop. / 30 vars. reales

Fig. 4.6: Tiempos de verificación para minimizaciones parciales (desde 0 minutos hasta 3 horas con intervalos de 30 minutos) de contratos de calidad.

ración del tiempo de resolución promedio requerido por contratos de calidad minimizados parcialmente hasta 3 horas.

# vars. prop. en provisión		Initialisation time (s)	# variables reales 10	15	20	25	30
150	sin minimización	43	171.35	94.01	–	–	–
	minimización de 3hs	44	15.13	38.59	–	–	–
	percentage	102.3 %	8.8 %	41 %	–	–	–
200	sin minimización	77	–	896.46	444.53	–	–
	minimización de 3hs	80	–	50.31	139.34	–	–
	percentage	103.90 %	–	5.6 %	31.3 %	–	–
250	sin minimización	127	–	–	2 930.41	1 310.06	–
	minimización de 3hs	130	–	–	173.81	429.67	–
	percentage	102.36 %	–	–	5.9 %	32.7 %	–
300	sin minimización	186	–	–	–	8 083.04	3 400.33 / **TO**
	minimización de 3hs	189	–	–	–	560.94	2 265.39
	percentage	101.61 %	–	–	–	6.9 %	66.6 %

Tab. 4.3: Comparación del tiempo de resolución promedio necesario para contratos no minimizados y minimizados (3 horas)

Discusion. Observando los gráficos en la Fig. 4.6 se derivan las siguientes conclusiones:

1) El tiempo de ejecución requerido por la fase de verificación disminuye a medida que aumenta la cantidad de tiempo invertido en la minimización. Para contratos con más de 200 variables proposicionales, el tiempo requerido por la fase de verificación disminuye drásticamente hasta alcanzar un *plateau* después de sólo 30 minutos de minimización. En el caso de los contratos con más de 250 variables proposicionales, el patrón es exactamente el mismo pero requiere entre 60 y 90 minutos de minimización. Finalmente, para contratos con más de 300 variables proposicionales, este fenómeno solo se puede observar para instancias con más de 25 variables reales después de 150 minutos, pero en el caso de 30 variables reales podemos observar casos que no se podían verificar dentro de las 3 horas para los contratos no minimizados y que sí se pueden verificar después de 60 minutos de la fase de minimización.

2) El crecimiento en el número de variables reales para instancias con el mismo número de variables proposicionales muestra más dispersión (en el tiempo requerido para la fase de verificación) sobre instancias minimizadas por períodos cortos de tiempo, pero que rápidamente colapsan en el *plateau*.

Los resultados en la Tabla 4.3 muestran cómo 3 horas de la fase de minimización reducen drásticamente el costo de la fase de verificación. La reducción depende naturalmente del tamaño del problema bajo análisis; es fácil ver que los problemas más grandes muestran tasas de reducción más pequeñas con la misma cantidad de tiempo invertido en la fase de minimización.

4.3. Conclusiones

Propusimos el uso de un lenguaje formal para la especificación de contratos de calidad de servicio junto con un procedimiento de verificación de compatibilidad de dos fases, basado en el algoritmo propuesto por Shoukry y otros en [138] para la verificación de

sistemas híbridos, adaptado al escenario concreto de negociación de acuerdos de nivel de servicio (SLA) para la reconfiguración automática de sistemas de software basados en servicios. Se formularon preguntas de investigación y se llevaron a cabo experimentos para responderlas. El conjunto de datos se diseñó para estresar la técnica al forzar que las instancias del problema sean no factibles, lo que requiere agotar el espacio de posibles soluciones; la evaluación de la herramienta bajo instancias factibles queda pendiente para el futuro. Los resultados experimentales mostrados en la respuesta a **RQ2** evidenciaron que la minimización completa de contratos podría ser inalcanzable, ya que el consumo de tiempo y memoria podría resultar demasiado alto. Para mitigar esta demanda, propusimos el uso de la minimización incremental, y los resultados experimentales mostrados en la respuesta a **RQ3** evidenciaron que una cantidad relativamente pequeña de tiempo invertido en una primera fase de *Minimización a través de Optimización Convexa* – MCO de contratos de calidad reduce drásticamente el tiempo requerido por una segunda fase dedicada a verificar las versiones minimizadas de los contratos de calidad.

5

Satisfacción parcial de requisitos aplicada al rankeo de servicios

En el Capítulo 4 presentamos un procedimiento de análisis eficiente y completamente automático para establecer SLA a través de la verificación de un juicio de la forma $Pr \models Rq$. Esta visión garantiza la selección de un servicio que satisfaga los requisitos de la aplicación en ejecución, pero no proporciona ninguna información cuando no hay ningún servicio disponible cuyo contrato de provisión cumpla completamente con esos requisitos. En este escenario, una aplicación se vería obligada a abortar su ejecución porque no hay un SLA posible sobre los parámetros de calidad requeridos. Una interpretación geométrica del juicio $Pr \models Rq$ expresa que todos los valores que satisfacen las variables reales de Pr también satisfacen las variables reales de Rq; por otro lado, una respuesta negativa simplemente significa que existe al menos una asignación de valores a variables reales que satisface Pr pero que no satisface Rq. En este capítulo, proponemos un procedimiento automático para evaluar el cumplimiento parcial de los contratos de calidad mediante la estimación del tamaño de la intersección entre el conjunto de valores que satisfacen el contrato de provisión y el conjunto de valores que satisfacen el contrato de requisitos. Cuando esta métrica se interpreta en relación con el tamaño del conjunto de valores que satisfacen el contrato de provisión, obtenemos lo que llamamos la *tasa de inclusión*.

Existen en la literatura otras nociones de *ranking* de servicios de acuerdo con sus atributos de calidad. Estos enfoques abordan el problema de clasificar servicios de acuerdo con alguna noción de preferencia y, en general, se basan en monitorear el comportamiento de atributos específicos en tiempo de ejecución [104, 113] o mediante el intercambio social de la opinión de los usuarios [156, 108]. Otros enfoques, como [94], se basan en modelos probabilísticos utilizados para verificar formalmente las garantías de calidad de despliegues de sistemas concretos. Hasta donde sabemos, no hay ninguno enfocado en el problema del *rankeo* de servicios en un escenario en el que los valores de calidad provistos y requeridos están caracterizados por contratos especificados formalmente.

En la Sec. 5.1 presentamos una definición de satisfacción parcial como la *tasa de inclusión* de un contrato de calidad dentro de otro; en la Sec. 5.2 presentamos un estudio experimental inicial del comportamiento del algoritmo para calcular dichos indicadores. Finalmente, en la Sec. 5.3, sacamos algunas conclusiones y señalamos algunas líneas de investigación adicionales. Los resultados presentados en este capítulo fueron previamente publicados en [112].

5.1. La *tasa de inclusión* como medio para cuantificar el cumplimiento parcial de contratos de calidad

El procedimiento presentado en Capítulo 4 garantiza la selección de cualquier servicio cuyo contrato de calidad satisfaga los requisitos de la aplicación que se está ejecutando, pero no proporciona ninguna ayuda sobre qué hacer cuando ningún servicio registrado en el repositorio satisface completamente los requisitos. Tal escenario puede entenderse intuitivamente como una aplicación que desencadena el proceso de selección de servicios dentro de un repositorio donde cada servicio registrado que es compatible en términos funcionales tiene un contrato de provisión que no cumple de manera estricta con lo estipulado por el contrato de requisitos. Es decir, el contrato de provisión admite al menos una asignación de valores para los atributos cuantitativos que no es aceptada por el contrato de requisitos. Esto puede entenderse como un servicio que tiene al menos una ejecución en la que los valores adoptados por los atributos cuantitativos no satisfacen los requisitos de la aplicación. Denominaremos a este fenómeno como *cumplimiento parcial*.

Una interpretación geométrica de la fórmula $Pr \implies Rq$ lleva a una visión de la noción de satisfacción diferente a la que veníamos presentando en capítulos anteriores, en la que será posible medir diferentes grados de cumplimiento parcial. Estos grados van desde el cumplimiento total, cuando todas las asignaciones de valores cuantitativos admitidas por Pr son aceptadas por Rq, hasta cumplimiento nulo, cuando ninguna de las asignaciones admitidas por Pr es aceptada por Rq. Para mostrar esto, primero consideramos una formulación diferente, pero equivalente, del problema. Para simplificar la presentación de la principal contribución del capítulo, nos enfocamos en especificaciones convexas cuyo conjunto de variables proposicionales está vacío. Sin embargo, los resultados pueden extenderse a especificaciones convexas con variables proposicionales.

Observando la Definición 5, se puede ver que siempre que el conjunto de variables proposicionales esté vacío, el conjunto $mSMCf(\mathcal{X}, \emptyset)$ estará formado por fórmulas de la forma

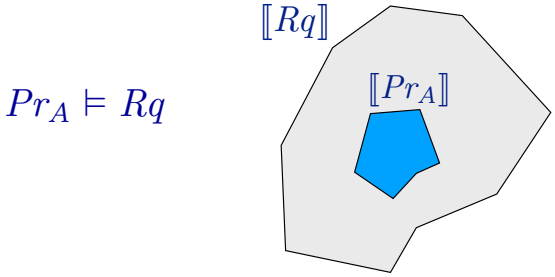
$$\bigwedge_{0 \leq i < n} \bigvee_{0 \leq j < m_i} f_{i,j}(\overrightarrow{x}) \ \mathcal{R} \ 0$$

con $\mathcal{R} \in \{<, \leq\}$ y $f_{i,j}$ una función convexa.

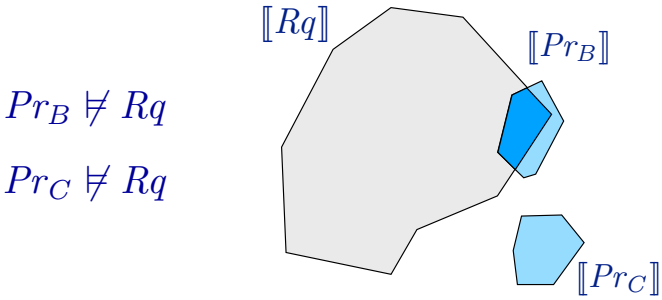
Veamos de manera precisa qué características tiene una valuación que satisfaga una fórmula de este tipo.

Definición 7. *Sea $\langle \langle \mathcal{X}, \emptyset \rangle, \alpha \rangle$ una especificación convexa. Definimos una valuación para α como una función $v : \mathcal{X} \to \mathbb{R}$. Por simplicidad, la denotaremos como el vector $\overrightarrow{v} = \{v_0, \ldots, v_{|\mathcal{X}|-1}\}$. La noción de satisfacción es una relación $\models \subseteq [\mathcal{X} \to \mathbb{R}] \times mSMCf(\mathcal{X}, \emptyset)$ definida como $\overrightarrow{v} \models \alpha$ si y solo si para todo $0 \leq i < n$, existe $0 \leq j < m_i$, tal que $f_{i,j}(\overrightarrow{v}) \ \mathcal{R} \ 0$. Luego, definimos el conjunto de valores que satisfacen una fórmula como $[\![\alpha]\!] = \left\{ \overrightarrow{v} \ \middle| \ \overrightarrow{v} \models \alpha \right\}$.*

Recordemos que nuestro objetivo es verificar compatibilidad entre un contrato de requisitos Rq y un contrato de provisión Pr, es decir, verificar si $Pr \models Rq$. Basándonos en la definición anterior, podemos reescribir esta relación como $[\![Pr]\!] \subseteq [\![Rq]\!]$. Esto es, el conjunto de valores que satisfacen Pr debe ser un subconjunto del conjunto de valores que satisfacen Rq, como se muestra en la Figura 5.1a. Mientras que un escenario en el que $Pr \not\models Rq$ se puede entender como $[\![Pr]\!] \not\subseteq [\![Rq]\!]$, como se muestra en la Figura 5.1b.



(a) Escenario en que se cumple la relación de satisfacción.



(b) Escenario que no cumple la relación de satisfacción.

Fig. 5.1: Interpretación geométrica de la relación de satisfacción entre dos contratos de calidad.

A continuación presentamos una serie de observaciones que nos ayudarán a caracterizar de manera geométrica el conjunto de valores que satisfacen una especificación convexa.

Proposición 3. *Sea* $\langle\langle\mathcal{X},\emptyset\rangle,\alpha\rangle$ *una especificación convexa monótona, entonces existe una fórmula* β *con forma* $\bigvee_{0\leq i<m}\bigwedge_{0\leq j<n_i} f_{i,j}(\overrightarrow{x})\ \mathcal{R}\ 0$ *(es decir, una fórmula en* forma normal disyuntiva*) tal que* $[\![\alpha]\!]=[\![\beta]\!]$.

Demostración. Sea C el conjunto de restricciones convexas que aparecen en α y α_B la abstracción proposicional de α donde se sustituyó cada restricción convexa $C\in\alpha$ por una nueva variable proposicional $v_C\in\mathcal{V}$ (ver Definición 6). Se puede obtener una fórmula $\widehat{\alpha_B}$ en forma normal disyuntiva, equivalente a α_B, a partir de listar todas las valuaciones satisfacibles y escribir una cláusula por cada una. Luego, se obtiene la fórmula β a partir de reemplazar cada aparición de v_C en $\widehat{\alpha_B}$ por C y cada aparición de $\neg v_C$ por la negación de C. La negación de una inecuación $f(\overrightarrow{x})\ <\ 0$ se puede expresar como $-f(\overrightarrow{x})\ \leq\ 0$ y la negación de una inecuación $f(\overrightarrow{x})\ \leq\ 0$ se puede expresar como $-f(\overrightarrow{x})\ <\ 0$. Nótese que si f es una función lineal convexa entonces $-f$ también lo es. □

Si nos restringimos a especificaciones convexas donde todas las funciones son lineales, entonces el hecho de que la fórmula β introducida por la Proposición 3 está en forma normal conjuntiva nos permite caracterizar su conjunto de valores como politopos convexos. A continuación presentamos la definición formal de politopo convexo y una proposición que nos permite caracterizar el conjunto de valores de una especificación convexa como la unión de politopos convexos.

Definición 8 (Región poliédrica)**.** *Una* región poliédrica *(o* politopo convexo*) en* $\mathbb{R}^n$ *es un conjunto de puntos* $\{\, \mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n, A\mathbf{x} \leq \mathbf{b} \,\}$*, con* $A \in \mathbb{R}^{m \times n}$ *y* $\mathbf{b} \in \mathbb{R}^m$*. Es decir, es el conjunto de puntos que satisfacen un sistema de desigualdades lineales.*

Proposición 4. *Sea una fórmula* β *con forma* $\bigvee_{0 \leq i < m} \bigwedge_{0 \leq j < n_i} f_{i,j}(\overrightarrow{x}) \; \mathcal{R} \; 0$*. Si la función* $f_{i,j}$ *es lineal para todo* $0 \leq i < m$ *y* $0 \leq j < n_i$*, entonces* $[\![\beta]\!] = \bigcup_{0 \leq i < m} [\![\mathtt{P}_i]\!]$ *donde* $\mathtt{P}_i$ *es el politopo convexo definido por el conjunto de restricciones* $\{f_{i,j}(\overrightarrow{x}) \; \mathcal{R} \; 0\}_{0 \leq j < n_i}$*.*

Demostración. Se deriva directamente del hecho que un politopo convexo se define como la intersección de un número finito de semiespacios convexos, caracterizados por restricciones lineales. □

Basándonos en esta observación hemos propuesto calcular el volumen de la intersección entre $[\![Pr]\!]$ y $[\![Rq]\!]$ (es decir, aquellos valores de los atributos cuantitativos de las ejecuciones del servicio que también satisfacen los requisitos de la aplicación en ejecución) en relación al volumen de $[\![Pr]\!]$, a lo que llamamos *tasa de inclusión.*

Definición 9 (tasa de inclusión)**.** *Sean Pr y Rq dos especificaciones convexas, definimos la tasa de inclusión como:*

$$\frac{vol\,([\![Pr]\!] \cap [\![Rq]\!])}{vol\,([\![Pr]\!])}$$

Este indicador esencialmente cuantifica el porcentaje de los valores de calidad permitidos por el contrato de provisión que son aceptados por el contrato de requisitos. Bajo esta interpretación, argumentamos que dicho indicador sirve para cuantificar el cumplimiento parcial de *Rq* por parte de *Pr* y, por lo tanto, los servicios funcionalmente compatibles se pueden *rankear* por calidad ordenándolos según la tasa de inclusión. Cabe remarcar que esta métrica sólo tendra sentido cuando se cumplen las siguientes condiciones:

- $vol\,([\![Pr]\!] \cap [\![Rq]\!])$ es finito,
- $vol\,([\![Pr]\!])$ es finito, y
- $vol\,([\![Pr]\!])$ es distinto de cero.

Para las primeras dos condiciones es suficiente con que los politopos convexos que caracterizan *Pr* (Proposición 4) sean acotados. Para la tercera condicion es suficiente con que *Pr* sea satisfacible, es decir, que la propia especificación del servicio no sea contradictoria.

5.1.1. Sobre la complejidad del cálculo del volumen

En la literatura, el volumen de objectos geométricos de estas características se define invariablemente en términos de la suma de los volúmenes de una cierta descomposición en una familia de politopos convexos [68]. Afortunadamente, la Proposición 3 nos ayuda ya que expresa el conjunto de valores de un contrato como la unión de los conjuntos de valores de cada miembro de una familia de politopos convexos, expresada como $Pr = \bigvee_{i \in \mathcal{I}} \mathtt{P}_i$ y $Rq = \bigvee_{j \in \mathcal{J}} \mathtt{R}_j$, donde $\mathcal{I}$ y $\mathcal{J}$ son conjuntos finitos de índices. Como ninguna de las dos familias de politopos convexos, $\{\mathtt{P}_i\}_{i \in \mathcal{I}}$ y $\{\mathtt{R}_j\}_{j \in \mathcal{J}}$, se garantiza que sea una partición, podría suceder que dos politopos convexos en la misma familia tengan intersección no vacía. Por lo tanto, los volúmenes deben calcularse siguiendo el *Principio de inclusión-exclusión* [103, Chap. 4] para evitar contabilizar múltiples veces la misma región. Entonces, la tasa de inclusión se calcula como se muestra en la Figura 5.2.

Primero: Calcular el volumen de la intersección entre Pr y Rq:

$$\begin{aligned} vol\left(\bigcup_{i\in\mathcal{I}}[\![\mathtt{P}_i]\!]\cap\bigcup_{j\in\mathcal{J}}[\![\mathtt{R}_j]\!]\right) &= vol\left(\bigcup_{i\in\mathcal{I}}\left([\![\mathtt{P}_i]\!]\cap\bigcup_{j\in\mathcal{J}}[\![\mathtt{R}_j]\!]\right)\right) \\ &= vol\left(\bigcup_{i\in\mathcal{I}}\bigcup_{j\in\mathcal{J}}([\![\mathtt{P}_i]\!]\cap[\![\mathtt{R}_j]\!])\right) \\ &= vol\left(\bigcup_{(i,j)\in\mathcal{I}\times\mathcal{J}}([\![\mathtt{P}_i]\!]\cap[\![\mathtt{R}_j]\!])\right) \\ &= \sum_{\emptyset\neq\mathcal{K}\subseteq\mathcal{I}\times\mathcal{J}}(-1)^{|\mathcal{K}|-1}vol\left(\bigcap_{(i,j)\in\mathcal{K}}([\![\mathtt{P}_i]\!]\cap[\![\mathtt{R}_j]\!])\right) \end{aligned}$$

Segundo: Calcular el volumen de Pr:

$$vol\left(\bigcup_{i\in\mathcal{I}}[\![\mathtt{P}_i]\!]\right)=\sum_{\emptyset\neq\mathcal{K}\subseteq\mathcal{I}}(-1)^{|\mathcal{K}|-1}vol\left(\bigcap_{k\in\mathcal{K}}[\![\mathtt{P}_k]\!]\right)$$

Tercero: Calcular la tasa de inclusión de Pr en Rq como: $\frac{vol(\bigcup_{i\in\mathcal{I}}[\![\mathtt{P}_i]\!]\cap\bigcup_{j\in\mathcal{J}}[\![\mathtt{R}_j]\!])}{vol(\bigcup_{i\in\mathcal{I}}[\![\mathtt{P}_i]\!])}$.

Fig. 5.2: Algoritmo para el cálculo de la tasa de inclusión

En el algoritmo anterior hay dos fuentes principales de complejidad. Por un lado, está el costo asociado a la computación del operador de volumen para un solo politopo convexo (es decir, $vol(\cdot)$) y, por otro lado, está el costo asociado al número de invocaciones a dicho operador. Calcular el volumen de un politopo convexo ha sido un problema relevante desde finales de los años 70, cuando la geometría computacional estaba emergiendo [116]. A fines de los años 80, Dyer et. al. [50] y Khachiyan [92] estudiaron la complejidad algorítmica y demostraron que era **# P-Hard**, por lo que cualquier algoritmo que calcule el volumen exacto de un politopo en una dimensión arbitraria tendrá un alto costo computacional (al lector interesado se le señala hacia [25] para un análisis práctico de distintas implementaciones). Naturalmente, este alto costo computacional asociado a proporcionar una solución exacta motivó el desarrollo de algoritmos más eficientes para estimar volúmenes, hasta cierto nivel de precisión; el primero de los cuales fue presentado por Kannan et. al. en [88]. En [62] se propone un algoritmo de estimación basado en el *algoritmo Monte Carlo multifase*, construido sobre mejoras de los métodos previos. Los autores muestran, a través de una extensa experimentación, que puede manejar eficientemente instancias de docenas de dimensiones con alta precisión. La complejidad del algoritmo se muestra como $O^*(m\cdot n^3)$, donde n es la dimensión y m es el número de restricciones, y se mide en términos de operaciones básicas, en lugar de depender de consultas costosas a un oráculo, que generalmente dominan la complejidad de este tipo de algoritmos.[1] El método de estimación se distribuye bajo el nombre `PolyVest` (disponible en `http://lcs.ios.ac.cn/~zj/polyvest.html`) y demuestra que para instancias aleatorias de politopos es más rápido que las herramientas existentes de cálculo exacto de volumen. Los resultados experimentales muestran una

[1] La notación Soft-O (es decir, O^*) indica la omisión de factores logarítmicos, así como factores que dependen de otros parámetros como la limitación de error.

desviación promedio de menos del 20 % y un error de menos del 1 % en un método de validación de resultados basado en verificar el invariante $vol(P) = vol(P_1) + vol(P_2)$, donde P es la unión disjunta de P_1 y P_2.

Con respecto al costo asociado al número de invocaciones al operador de volumen; la implementación del algoritmo de la Figura 5.2 para calcular la tasa de inclusión requiere $\sum_{0<i\leq|\mathcal{I}|\cdot|\mathcal{J}|}\binom{|\mathcal{I}|\cdot|\mathcal{J}|}{i} + \sum_{0<i\leq|\mathcal{I}|}\binom{|\mathcal{I}|}{i} = 2^{|\mathcal{I}|\cdot|\mathcal{J}|} + 2^{|\mathcal{I}|}$ invocaciones a la función que calcula el volumen de un politopo convexo. En este caso, una forma natural de realizar una estimación será considerar solo algunas de las intersecciones involucradas en el cálculo, es decir, considerar sólo algunos de los términos de las sumatorias en el primer y segundo paso del algoritmo de la Figura 5.2. Esta estimación puede llevar a una subestimación o una sobreestimación según el signo y la magnitud de los términos faltantes. Las propuestas para acotar el número de invocaciones al operador de volumen se presentarán y estudiarán empíricamente en la siguiente sección.

5.2. Resultados experimentales

Los experimentos que presentaremos en esta sección fueron ejecutados sobre un procesador de arquitectura x86_64 Intel(R) Core(TM) i5 CPU 750 a 2.67 GHz (CPU MHz: 2661 - CPU max MHz: 2661 - CPU min MHz: 1197) con una caché de 3 niveles (caché L1d: 32 Kb, caché L1i: 32 Kb - caché L2: 256 Kb - caché L3: 8192 Kb), 8 Gb de SDRAM a 1333 MHz, y con la distribución SMP Debian Linux 4.9.88-1+deb9u1 (2018-05-07).

Nuestra primera pregunta de investigación está relacionada con las consideraciones de complejidad realizadas al final de la sección anterior, respecto al cómputo del volumen de un contrato $\mathcal{P}$, descompuesto como la unión de politopos convexos $\bigcup_{i\in\mathcal{I}}\mathcal{P}_i$, recurriendo al principio de inclusión-exclusión. Si bien el cálculo la tasa de inclusión requiere este cómputo de volumen dos veces, la pregunta de investigación se planteará en términos de un solo contrato descompuesto en una familia de politopos convexos. Esto nos permitirá entender los límites impuestos por la complejidad de este cómputo.

RQ1: ¿Cómo se refleja la complejidad del cómputo de $vol\left(\bigcup_{i\in\mathcal{I}}\mathcal{P}_i\right)$ en la evolución de su tiempo de ejecución?

La Tabla 5.1 muestra el tiempo de ejecución del cómputo de volumen para distintas familias de politopos. El experimento se llevó a cabo sobre casos de estudio sintéticos, generados automáticamente de manera aleatoria, que consisten en familias de 5 a 11 politopos convexos sobre 10, 15, 20, 25, 30 y 35 atributos cuantitativos. Cada politopo de dimensión n se construye primero generando un cubo n-dimensional cuyo tamaño es aleatorio y luego, para evitar trivializar el problema, inclinando sus caras un ángulo al azar. La posición del politopo garantiza incluir el 0 n-dimensional, con lo que se garantizar que el volumen de cada término en la suma

$$\sum_{\emptyset\neq\mathcal{K}\subseteq\mathcal{I}}(-1)^{|\mathcal{K}|-1}vol\left(\bigcap_{k\in\mathcal{K}}[\![\mathrm{P}_k]\!]\right) \tag{5.1}$$

no es cero. Para cada combinación de dimensión y tamaño de la familia de politopos, el procedimiento se ejecutó sobre 20 instancias diferentes, como máximo durante 1 hora. Los

resultados se muestran ya sea indicando el tiempo promedio de ejecución en segundos o colocando **TO** en aquellos casos que no se terminaron dentro del presupuesto de tiempo.

$\lvert\mathcal{P}\rvert$	Cantidad de atributos					
	10	15	20	25	30	35
5	27.45	155.12	535.53	1 388.15	3017.07	**TO**
6	64.73	369.74	1 274.30	3 305.37	**TO**	**TO**
7	149.85	857.45	2 952.59	**TO**	**TO**	**TO**
8	341.75	1 953.10	**TO**	**TO**	**TO**	**TO**
9	765.22	**TO**	**TO**	**TO**	**TO**	**TO**
10	1 701.66	**TO**	**TO**	**TO**	**TO**	**TO**
11	**TO**	**TO**	**TO**	**TO**	**TO**	**TO**

Tab. 5.1: Tiempo de ejecución del cálculo del volumen de un politopo $\mathcal{P} = \bigcup_{i\in\mathcal{I}} \mathcal{P}_i$

A primera vista, es posible identificar que: *1*) la forma de los datos explicita la naturaleza exponencial del cálculo, y *2*) el tamaño de la familia de politopos en los casos en los que el volumen se calculó con éxito dentro de la cota de 1 hora de tiempo es pequeña.

En este escenario, la aplicabilidad de una técnica que depende del cálculo del volumen de una familia de politopos convexos depende de la posibilidad de calcular una aproximación buena y eficiente de dicho volumen. Una forma natural de hacerlo es considerando solamente una cantidad limitada de términos de la suma en la Ec. 5.1, ya que sabemos que la cantidad total de términos es $\sum_{0<i\leq|\mathcal{I}||} \binom{|\mathcal{I}|}{i} = 2^{|\mathcal{I}|}$. A continuación, consideramos y evaluamos las implicaciones de aproximar el volumen de un contrato.

Si analizamos la estructura de los términos en la suma de la Ec. 5.1, podemos observar algunas propiedades acerca de la manera en que las sumas parciales convergen al valor del volumen. La siguiente proposición establece una relación entre los subconjuntos de la familia de politopos y la intersección de estos subconjuntos, lo que nos ayudará a comparar los valores de las sumas parciales consecutivas en el contexto del principio de inclusión-exclusión.

Proposición 5. *Sea $\mathcal{P} = \{\mathcal{P}_i\}_{i\in\mathcal{I}}$ y $\mathcal{K} \subseteq \mathcal{I}$ tal que $1 < |\mathcal{K}| \leq |\mathcal{I}|$, entonces para todo $\mathcal{K}' \subset \mathcal{K}$, con $|\mathcal{K}'| = |\mathcal{K}| - 1$, se tiene que $\bigcap_{k\in\mathcal{K}}[\![\mathcal{P}_k]\!] \subseteq \bigcap_{k\in\mathcal{K}'}[\![\mathcal{P}_k]\!]$. Es decir, quitar un elemento de la intersección de un conjunto de politopos convexos $\mathcal{K}$ siempre produce una intersección que contiene a la intersección original.*

Demostración. Por hipótesis, $\bigcap_{k\in\mathcal{K}}[\![\mathcal{P}_k]\!]$ se puede escribir como $(\bigcap_{k\in\mathcal{K}'}[\![\mathcal{P}_k]\!]) \bigcap [\![\mathcal{P}_j]\!]$, con $j \in \mathcal{K} \setminus \mathcal{K}'$. Luego, $(\bigcap_{k\in\mathcal{K}'}[\![\mathcal{P}_k]\!]) \bigcap [\![\mathcal{P}_j]\!] \subseteq \bigcap_{k\in\mathcal{K}'}[\![\mathcal{P}_k]\!]$ por propiedades básicas de la intersección de conjuntos. □

Esta relación se transfiere a las sumas parciales de la siguiente manera:

Proposición 6. *Sea $\mathcal{P} = \{\mathcal{P}_i\}_{i\in\mathcal{I}}$ y $0 < n < |\mathcal{I}|$, con $n \equiv 0 \pmod 2$ entonces*

$$\sum_{\substack{\emptyset \neq \mathcal{K} \subseteq \mathcal{I} \\ 0 < |\mathcal{K}| \leq n}} (-1)^{|\mathcal{K}|-1} vol\left(\bigcap_{k\in\mathcal{K}} [\![\mathcal{P}_k]\!]\right) \leq vol(\mathcal{P}) \leq \sum_{\substack{\emptyset \neq \mathcal{K} \subseteq \mathcal{I} \\ 0 < |\mathcal{K}| \leq n+1}} (-1)^{|\mathcal{K}|-1} vol\left(\bigcap_{k\in\mathcal{K}} [\![\mathcal{P}_k]\!]\right) .$$

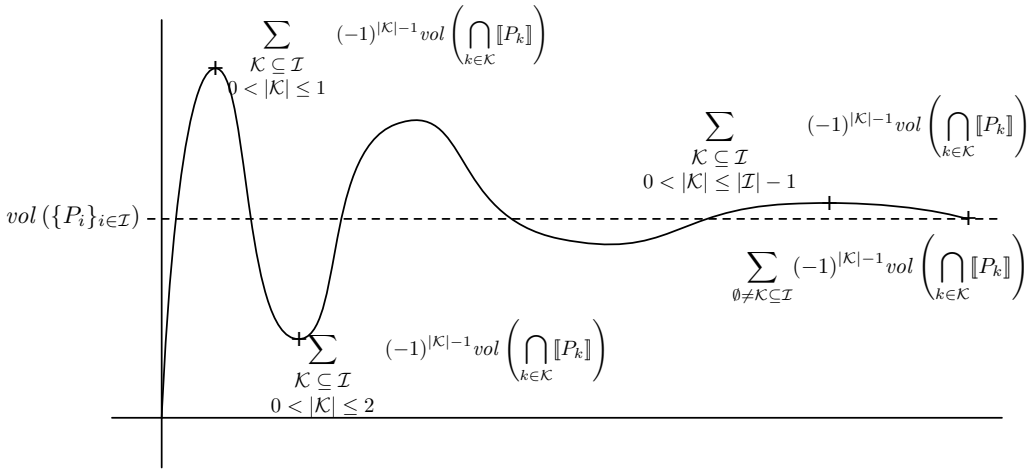


Fig. 5.3: Diagrama ilustrativo que muestra de qué manera las sumas parciales convergen hacia el valor del volumen.

De manera intuitiva, las sumas parciales subsiguientes representan una cota superior y una cota inferior para el valor exacto del volumen, como se muestra en el diagrama ilustrativo de la Fig. 5.3.

Como se mostró en la evaluación experimental de **RQ1**, no es razonable calcular el valor exacto del volumen de un contrato debido a la complejidad exponencial del procedimiento. Calcular una aproximación de una magnitud requiere la introducción de una métrica para medir la distancia entre el valor obtenido como aproximación y el valor exacto. Como es habitual, el error relativo entre la magnitud x y un valor aproximado $\widehat{x}$ se define como $\frac{|\widehat{x}-x|}{|x|}$, donde $|\widehat{x}-x|$ es el error absoluto.

La siguiente proposición establece límites superiores para el error relativo que no dependen del valor exacto x del volumen de un politopo, sino de los valores de dos aproximaciones consecutivas: $\{x_{k-1}, x_k\}$ o $\{x_k, x_{k+1}\}$.

Proposición 7. *Sea* $x, x_{k-1}, x_k, x_{k+1} \in \mathbb{R}$, *entonces:*

- *si* $0 < x_{k-1} \leq x \leq x_k$, *entonces* $\frac{x_k - x}{x} \leq \frac{x_k - x_{k-1}}{x_{k-1}}$ *y*
- *si* $x_k \geq x \geq x_{k+1} > 0$, *entonces* $\frac{x_k - x}{x} \leq \frac{x_k - x_{k+1}}{x_{k+1}}$.
- *si* $0 < x_{k-1} \leq x_{k+1}$, *entonces* $\frac{x_k - x_{k-1}}{x_{k-1}} \geq \frac{x_k - x_{k+1}}{x_{k+1}}$.

En esencia, lo que nos dice la Proposición 7 es que si tenemos dos aproximaciones consecutivas, entonces sabemos una cota superior para el error relativo de la aproximación. No sólo eso, sino que esta cota superior es decreciente a medida que tomamos aproximaciones sucesivas. Usando estas cotas superiores del error relativo, y considerando la Proposición 6, podemos diseñar un algoritmo para calcular el volumen aproximado de un contrato calculando aproximaciones sucesivas. Una cota superior deseable para error relativo, junto con la establecido en la Proposición 7, funcionarán como criterio de convergencia para el algoritmo. Las cotas establecidas en Proposición 7 no garantizan en sí mismas cierta complejidad computacional del algoritmo, más allá de que: a) en el peor caso, cuando se requiere un error relativo igual a 0, la complejidad es $O(2^n)$, b) en el mejor caso, cuando la cota es lo suficientemente grande como para permitir que el algoritmo converja en la

segunda aproximación, la complejidad es $O(n^2)$ (con n términos correspondientes al volumen de cada politopo convexo, y $\frac{n(n-1)}{2}$ términos correspondientes a las $\binom{n}{2}$ posibles intersecciones de 2 de los n politopos convexos).

Nuestra segunda pregunta de investigación analiza el impacto de calcular el volumen de un contrato hasta cierta cota del error relativo, utilizando la Proposición 7.

RQ2: ¿Cómo convergen las sumas parciales de los términos de la sumatoria de la Ec. 5.1 al valor $vol\left(\bigcup_{i\in\mathcal{I}}\mathcal{P}_i\right)$ con una tolerancia al error relativo de 1 % y 5 %?

La Tabla 5.2 muestra los resultados de calcular el volumen de los mismos casos de estudio de la configuración experimental utilizada para responder a **RQ1**, considerando 1 % y 5 % como cotas del error relativo. En la columna etiquetada *Ap.* se muestra el número promedio de aproximaciones necesarias para alcanzar la cota correspondiente.

$\|P\|$	Número de atributos											
	10		15		20		25		30		35	
	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo
5	3	19.04	2.15	58.45	2	172.52	2	444.01	2	963.33	2	1 885.34
6	3.5	42.28	2.3	106.22	2	248.08	2	639.84	2	1 387.02	2	2705.40
7	3.75	86.03	2.6	218.97	2	337.45	2	870.69	2	1 885.73	–	**TO**
8	3.85	158.17	2.65	337.98	2	440.65	2	1 138.22	2	2 464.32	–	**TO**
9	4.2	328.51	2.6	452.73	2	557.20	2	1 439.46	2	3 120.49	–	**TO**
10	5.1	886.75	2.85	787.08	2.05	806.05	2	1 776.29	–	**TO**	–	**TO**
11	4.95	1 452.44	2.6	811.80	2	831.83	2	2 149.33	–	**TO**	–	**TO**

(a) Tiempo de ejecución con tolerancia 1 % para el error relativo

$\|P\|$	Número de atributos											
	10		15		20		25		30		35	
	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo	Ap.	Tiempo
5	2.25	11.74	2	49.76	2	172.52	2	444.01	2	963.33	2	1 885.34
6	2.55	23.94	2	71.47	2	248.08	2	639.84	2	1 387.02	2	2 705.40
7	2.7	42.35	2	97.07	2	337.45	2	870.69	2	1 885.73	–	**TO**
8	2.75	66.33	2.05	143.15	2	440.65	2	1 138.22	2	2 464.32	–	**TO**
9	3	119.96	2	160.16	2	557.20	2	1 439.46	2	3 120.49	–	**TO**
10	3.3	261.83	2	197.01	2	687.05	2	1 776.29	–	**TO**	–	**TO**
11	3.2	308.42	2	238.46	2	831.83	2	2 149.33	–	**TO**	–	**TO**

(b) Tiempo de ejecución con toleracia 5 % para el error relativo

Tab. 5.2: Cálculo aproximado de los volúmenes de los politopos de **RQ1**

Se debe tener en cuenta que, como mencionamos antes, una vez que la cota superior para el error relativo sea lo suficientemente grande como para hacer que el algoritmo se detenga después de calcular las dos primeras aproximaciones, no habrá más ganancia de rendimiento al aumentar la cota (ver, por ejemplo, la celda correspondiente a contratos formados por 5 politopos convexos en 20 atributos, en las Tablas 5.2a y 5.2b).

Una forma natural de extender los resultados presentados en la Tabla 5.2 es explorar la escalabilidad del procedimiento hacia instancias más grandes. La Tabla 5.3 muestra el tamaño máximo de la familia de politopos cuyo volumen aproximado se puede calcular en una hora, considerando errores relativos con límites establecidos en el 1 % y el 5 %. Para las familias de politopos de más de 30 y 35 atributos cuantitativos, la cota superior ya se muestra en las Tablas 5.2a y 5.2b.

En la tabla 5.3 mostramos, para cada cantidad de atributos cuantitativos, el tamaño

Tol.	Número de atributos											
	10			15			20			25		
	$\|P\|$	Ap.	Tiempo	$\|P\|$	Ap.	Tiempo	$\|P\|$	Ap.	Tiempo	$\|P\|$	Ap.	Tiempo
1 %	11	4.95	1452.42	15	3	3067.82	22	2	3320.67	14	2	3481.29
5 %	15	3.72	1857.42	24.92	2	1218.55	22	2	3320.67	14	2	3481.29

Tab. 5.3: Límite superior para el tamaño de la familia de politopos cuyo volumen se puede calcular dentro de una hora con la tolerancia para el error relativo en 1 % y 5 %

máximo promedio de la familia de politopos ($|P|$) cuyo volumen se puede calcular dentro de una hora con la tolerancia (Tol.) para el error relativo en 1 % y 5 %. Esto nos muestra cómo el cálculo de un valor aproximado para el volumen de los contratos ayuda a obtener escalabilidad relativa como consecuencia de la disminución de la complejidad algorítmica, desde exponencial (en el caso de la computación exacta del volumen) a polinomial (cuyo grado depende del número de aproximaciones necesarias para satisfacer la cota superior, en el caso de la computación aproximada del volumen). Una comparación entre las Tablas 5.1 y 5.3 muestra la escalabilidad que se obtiene al calcular el volumen de un contrato con una tolerancia del 1 % para el error relativo. Por ejemplo, para contratos con 20 atributos cuantitativos, el tamaño máximo de la familia de politopos cuyo volumen se puede calcular de manera exacta en una hora es 7, mientras que el tamaño máximo de la familia de politopos cuyo volumen se puede calcular con una tolerancia del 1 % para el error relativo es 22.

Como introdujimos en la Sección 5.1, el objetivo de calcular eficientemente el volumen de este tipo de familias de politopos es cuantificar el cumplimiento parcial de contratos de calidad. La siguiente pregunta de investigación apunta a identificar cómo impactan los resultados obtenidos en el cómputo de la tasa de inclusión.

RQ3: ¿Cómo se transfiere la ganancia de rendimiento obtenida aproximando el volumen de los politopos al cómputo de la tasa de inclusión?

Las Tablas 5.4, 5.5 y 5.6 muestran la evaluación experimental del cómputo de la *tasa de inclusión* con volúmenes computados tanto exactos como aproximados con cotas superiores para los errores relativos establecidos en 1 % y 5 %, respectivamente. El experimento se llevó a cabo sobre contratos sintéticos generados automáticamente sobre 20 atributos cuantitativos, y con un número de politopos en Pr (denominado $|Pr|$) y Rq (denominado $|Rq|$) que iba de 1 a 6. Los casos de estudio se generaron siguiendo las mismas pautas que se utilizaron en los experimentos realizados para **RQ2** y para cada combinación de tamaños de $|Pr|$ y $|Rq|$, se calculó la *tasa de inclusión* sobre 10 instancias diferentes. Como cada caso de tasa de inclusión requiere necesariamente más potencia de cómputo que un sólo cómputo de volumen, estos casos se ejecutaron durante un máximo de 10 horas, marcando como **TO** las ejecuciones que no finalicen dentro de ese tiempo. Una celda presentada en color gris indica que solo algunos casos de ese tamaño agotaron el tiempo, y se indica el tiempo promedio de ejecución para los casos que sí terminaron.

Cada celda de la Tabla 5.4 muestra los tiempos promedio de ejecución en segundos para el cálculo exacto del volumen. Las Tablas 5.5 y 5.6 muestran el tiempo promedio de ejecución en segundos para el cálculo del volumen con una cota para el error relativo establecido en 1 % y 5 %, respectivamente. En estas también se presenta una tabla que

muestra el porcentaje de tiempo de ejecución en comparación con el tiempo de ejecución para el cálculo exacto del volumen, lo que nos indica la ganancia relativa en tiempo de ejecución.

$\|Pr\|$	$\|Rq\|$					
	1	2	3	4	5	6
1	20.8	60.1	162.3	413.5	1007.2	2 378.4
2	80.9	436.5	2 416.0	12 561.2	**TO**	**TO**
3	237.4	2 469.8	28 150.1	**TO**	**TO**	**TO**
4	622.7	12 745.2	**TO**	**TO**	**TO**	**TO**
5	1 535.1	**TO**	**TO**	**TO**	**TO**	**TO**
6	3 643.9	**TO**	**TO**	**TO**	**TO**	**TO**

Fig. 5.4: Tiempo de cómputo de la tasa de inclusión exacta para contratos con 20 atributos cuantitativos

$\|Pr\|$	$\|Rq\|$					
	1	2	3	4	5	6
1	20.8	60.1	136.3	320.1	654.0	1 067.9
2	80.9	358.1	949.5	2 890.9	13 600.6	14 908.1
3	191.5	984.1	3 973.7	11 866.9	15 055.9	35 264.7
4	423.6	2 974.1	11 916.1	18 126.8	**TO**	**TO**
5	839.3	13 764.3	15 237.3	**TO**	**TO**	**TO**
6	1 308.7	15 087.6	35 434.7	**TO**	**TO**	**TO**

$\|Pr\|$	$\|Rq\|$					
	1	2	3	4	5	6
1	100.0 %	100.0 %	84.0 %	77.4 %	64.9 %	44.9 %
2	100.0 %	82.0 %	39.3 %	23.0 %	-	-
3	80.7 %	39.8 %	14.1 %	-	-	-
4	68.0 %	23.3 %	-	-	-	-
5	54.7 %	-	-	-	-	-
6	35.9 %	-	-	-	-	-

Fig. 5.5: Tiempo de ejecución de la tasa de inclusión para contratos con 20 atributos, con un error relativo de hasta 1 %

Como puede verse, a medida que aumenta el tamaño de los contratos la ganancia en eficiencia es dramáticamente mejor. Hay casos en los cuales el cálculo aproximado se realiza en el 4 % del tiempo de cálculo del volumen exacto y varios casos en los que la tasa de inclusión exacta no se puede calcular dentro del presupuesto de tiempo de 10 horas, pero se vuelve factible cuando se introduce un límite para el error relativo.

Teniendo en cuenta que la tasa de inclusión sirve para proporcionar una forma de *rankear* los servicios según el grado de satisfacción parcial, vale la pena saber cómo se propaga el error relativo en el cálculo de la tasa de inclusión al proceso de selección de un servicio entre varios candidatos registrados en un repositorio. La siguiente proposición establece límites inferiores y superiores de cómo se aleja la tasa de inclusión aproximada

$\|Pr\|$	$\|Rq\|$					
	1	2	3	4	5	6
1	20.8	60.1	128.9	245.7	480.6	623.1
2	80.9	268.5	650.0	1 706.0	11 069.8	8 472.2
3	184.1	684.5	1 120.8	11 034.2	4 920.0	10 476.7
4	334.4	1 788.8	2 822.3	14 065.3	5 343.1	7 651.2
5	646.0	4 190.3	5 030.0	5 405.2	8 357.8	11 971.0
6	864.0	8 696.2	10 665.5	7 784.7	12 042.5	17 256.7

$\|Pr\|$	$\|Rq\|$					
	1	2	3	4	5	6
1	100.0 %	100.0 %	79.4 %	59.4 %	47.7 %	26.2 %
2	100.0 %	61.5 %	26.9 %	13.6 %	-	-
3	77.5 %s	27.7 %	4.0 %	-	-	-
4	53.7 %	14.0 %	-	-	-	-
5	42.1 %	-	-	-	-	-
6	23.7 %	-	-	-	-	-

Fig. 5.6: Tiempo de cómputo de la tasa de inclusión para contratos con 20 atributos, con un error relativo de hasta 5 %

de la tasa de inclusión exacta.

Proposición 8. *Sea* $\frac{|v-v_k|}{|v|} \leq e$ *y* $\frac{|w-w_k|}{|w|} \leq e$*, entonces:*

1. $(1-e) \cdot v \leq v_k \leq (1+e) \cdot v$ *(respectivamente* $(1-e) \cdot w \leq w_k \leq (1+e) \cdot w$*), y*
2. $\frac{(1-e) \cdot v}{(1+e) \cdot w} \leq \frac{v_k}{w_k} \leq \frac{(1+e) \cdot v}{(1-e) \cdot w}$ *(alternativamente* $\frac{1-e}{1+e} \cdot \frac{v}{w} \leq \frac{v_k}{w_k} \leq \frac{1+e}{1-e} \cdot \frac{v}{w}$*).*

Considere, como ejemplo, los límites obtenidos en la Prop. 8 para la aproximación de la tasa de inclusión, cuando los volúmenes aproximados se computan con un límite de error relativo menor al 5 % y al 1 %, como se hizo en experimentos previos. Sea eir la tasa de inclusión exacta, $ir5$ la tasa de inclusión computada con un error relativo menor al 5 % y $ir1$ la tasa de inclusión computada con un error relativo menor al 1 %; entonces, obtenemos que $0{,}905 \cdot eir \approx \frac{0{,}95}{1{,}05} \cdot eir \leq ir5 \leq \frac{1{,}05}{0{,}95} \cdot eir \approx 1{,}105 \cdot eir$ y $0{,}98 \cdot eir \approx \frac{0{,}99}{1{,}01} \cdot eir \leq ir1 \leq \frac{1{,}01}{0{,}99} \cdot eir \approx 1{,}02 \cdot eir$, respectivamente.

Nuestra última pregunta de investigación (**RQ4**) se centra en *rankear* diferentes contratos de provisión utilizando la tasa de inclusión como métrica para cuantificar la satisfacción parcial. Para este experimento, mostramos 3 ejemplos de clasificación de 20 contratos de provisión por satisfacción parcial de un único contrato de requisitos. Los conjuntos de datos fueron generados sintéticamente siguiendo las pautas utilizadas en la generación de los casos de estudio utilizados en **RQ3**. Cada clasificación corresponde a un tamaño diferente de familias de politopos según indicamos a continuación:

a) $|Pr| = 2$, $|Rq| = 4$,

b) $|Pr| = 3$, $|Rq| = 3$ y,

c) $|Pr| = 4$, $|Rq| = 2$.

La Tabla 5.4 muestra los resultados del experimento. Para cada caso, comparamos el ordenamiento obtenido utilizando el cálculo de volumen exacto junto con las aproximaciones obtenidas al aumentar el límite del error relativo. Es notable que en todos los ejemplos el ordenamiento óptimo se conserva incluso para los cálculos aproximados de volumen con todas las tolerancias de error relativo. Exhibimos la cantidad total de tiempo en segundos requerida para el cálculo de cada ordenamiento y el porcentaje sobre el tiempo de cálculo del ordenamiento óptimo.

Tolerancia error rel.	$\|Pr\| = 2$, $\|Rq\| = 4$ Time (s)	%	$\|Pr\| = 3$, $\|Rq\| = 3$ Time (s)	%	$\|Pr\| = 4$, $\|Rq\| = 2$ Time (s)	%
0 %	211 363	100 %	483 529	100 %	212 660	100 %
1 %	46 777	22 %	53 755	11 %	39 221	18 %
5 %	17 122	8 %	20 604	4 %	18 297	9 %
10 %	13 716	6 %	17 999	4 %	14 845	7 %
20 %	13 716	6 %	17 999	4 %	14 845	7 %

Tab. 5.4: Información resumida de los rankings basados en la tasa de inclusión para contratos con 20 atributos con un límite del error relativo del 0 %, 1 %, 5 %, 10 %, 20 %.

5.3. Conclusiones y trabajo futuro

En este capítulo hemos extendido el trabajo presentado en el Capítulo 4 en el que propusimos una formalización de los contratos de calidad de servicio como especificaciones convexas [138]. Bajo una interpretación geométrica, un contrato puede ser visto como una familia de politopos convexos acotados que caracterizan los valores admisibles para los atributos cuantitativos y, por lo tanto, un juicio de la forma $Pr \models Rq$ puede ser visto como la inclusión entre estos conjuntos de valores. Desde esta perspectiva, hemos propuesto una forma de *rankear* servicios cuantificando el volumen de la intersección entre $[\![Pr]\!]$ y $[\![Rq]\!]$, bajo el nombre de *tasa de inclusión*. Dado que este indicador cuantifica el porcentaje de valores permitidos por el contrato de provisión que son aceptados por el contrato de requisito, sostenemos que sirve al propósito de cuantificar el grado de satisfacción parcial. De la misma forma que la complejidad de computar el volumen exacto de un politopo convexo nos llevó a adoptar una solución aproximada a través de un método basado en Monte Carlo [62], la naturaleza exponencial de computar el volumen de una familia de politopos convexos nos obligó a proponer una técnica de aproximación de volumen basada en la imposición de un límite superior al error relativo. Hemos evaluado el tiempo de ejecución del cómputo de volumen aproximado con el objetivo de mostrar qué tan buena es la aproximación en relación con la cantidad de tiempo necesaria para computar el volumen exacto. También evaluamos la escalabilidad del cómputo de volumen aproximado observando el tamaño de los estudios de caso que pueden ser resueltos en un tiempo fijo de una hora. Usando este cómputo de volumen aproximado, evaluamos el tiempo de ejecución del cómputo de la tasa de inclusión con respecto al cómputo exacto, para aquellos casos en los que el valor exacto pudo ser obtenido en un tiempo fijo de diez horas. Finalmente, mostramos ejemplos del impacto de la tasa de inclusión aproximada en la clasificación de servicios comparándo la clasificación ideal, obtenida mediante el uso de la tasa de inclusión exacta, y la obtenida mediante el uso de la tasa de inclusión aproximada, para diferentes límites al error relativo. De manera notable, para los ejemplos que utilizamos en nuestros

experimentos, la tasa de inclusión aproximada es capaz de clasificar servicios preservando la clasificación ideal. Esta técnica para evaluar la satisfacción parcial de los contratos de calidad de servicio dispara una pregunta de investigación relevante con respecto a la automatización del *service broker*. Bajo la satisfacción parcial de los contratos de calidad de servicio, nos gustaría establecer preferencias entre candidatos, por ejemplo, si consideramos dos servicios con la misma tasa de inclusión, una aplicación de bajo presupuesto podría querer expresar su preferencia por un servicio más baratos y con menor rendimiento que por uno más caro y eficiente. El trabajo presentado en este capítulo establece una base para abordar este problema en el futuro.

Por último, nos gustaría hacer notar que la forma en que calculamos la tasa de inclusión esconde una suposición implícita que no hemos visibilizado aún. Se trata de que todos los valores de los atributos de calidad caracterizados por el contrato de provisión son equiprobables. Esto se manifiesta en el hecho de que la fórmula de la tasa equivale a integrar la función $\frac{1}{Vol(\llbracket Pr \rrbracket)}$ dentro de la región $\llbracket Pr \rrbracket \cap \llbracket Rq \rrbracket$, asumiendo efectivamente que cada valor caracterizado por el contrato de provisión tiene probabilidad $\frac{1}{Vol(\llbracket Pr \rrbracket)}$. En el próximo capítulo soltaremos esta suposición de uniformidad y consideraremos la naturaleza probabilística de los atributos de calidad.

6

Descripción y análisis de contratos de calidad probabilísticos

Como venimos desarrollando a lo largo de esta tesis, el contrato de calidad de provisión de un servicio puede considerarse como una especificación que describe los valores adoptados por sus atributos cuantitativos (*tiempo de respuesta*, *uso de memoria*, *precio*, *reputación*, etc.) a lo largo de sus posibles ejecuciones. En escenarios de ejecución realistas, dichos valores suelen estar sujetos a eventos externos no controlables, como fallos de hardware, errores de comunicación, congestión en la red, etc. Por lo tanto, resulta extremadamente difícil garantizar que los valores caracterizados por un contrato de calidad siempre se cumplan con absoluta certeza, lo que conduce naturalmente a una interpretación probabilística de estos contratos, en la que una ejecución produce una valuación de los atributos cuantitativos con cierta probabilidad. Del mismo modo, necesitamos una interpretación probabilística de los requisitos de calidad; en ella, un cliente se vincula a un proveedor de servicios específico solo si la probabilidad de que sus atributos cuantitativos adopten ciertos valores se encuentra dentro de límites aceptables para el cliente.

6.1. Nuestro enfoque

Cualquier enfoque para abordar el problema de selección de servicios tal como lo concebimos en Capítulo 2 debe abordar tres aspectos clave: 1) el modelo de *provisión de calidad* para describir cada servicio, 2) el modelo de *requisitos de calidad* para describir las necesidades del cliente, y 3) el *procedimiento de decisión* para seleccionar automáticamente el servicio adecuado. A continuación, delineamos estas dimensiones para el enfoque probabilístico que proponemos y lo comparamos con los enfoques existentes en la literatura.

Modelo de contrato de provisión El contrato de calidad de cada servicio se modela como una distribución de probabilidad continua multivariada donde cada variable aleatoria denota un atributo de calidad específico. Proponemos una metodología para aprender el contrato de calidad de un servicio a partir de datos muestreados del comportamiento de dicho servicio en el mundo real.

Modelo de contrato de requisitos Los requisitos de calidad son dados por la aplicación cliente para guiar la selección de servicios. Se trata de combinaciones proposicionales de restricciones de calidad que, a su vez, especifican cotas para la probabilidad de que los atributos de calidad tomen valores dentro de una región determinada por un politopo convexo acotado.

Procedimiento de decisión Dado un modelo de contrato de calidad y una fórmula de requisitos de calidad, nuestro algoritmo determina si el contrato de calidad satisface los requisitos. Se apoya en *SAT-solving* para manejar la estructura proposicional de la fórmula de requisitos y en integración de Monte Carlo para determinar si las cotas de probabilidad prescriptas por las restricciones de calidad se cumplen.

En sintonía con la noción de atributos de calidad que venimos desarrollando a lo largo de esta tesis, todo esto se hace en un entorno general donde no hay suposición *a priori* sobre la naturaleza específica de los atributos cuantitativos. Es decir, cualquier característica cuantificable del servicio de software puede considerarse un atributo cuantitativo.

6.1.1. Trabajo relacionado

Modelo de contrato de provisión Situamos las contribuciones de este capítulo como parte de un número creciente de investigaciones interesadas en capturar la naturaleza probabilística de contratos de provisión de calidad [93, 132, 81, 80]. En [93, 132], los autores proponen enfoques en los que los atributos de calidad se modelan como distribuciones de probabilidad separadas e independientes. En [81], los autores proponen un modelo probabilístico para describir calidad de servicio, pero solo consideran distribuciones de probabilidad discretas y restringen su enfoque a cinco atributos específicos. En [80], los autores proponen que cada atributo de calidad se modele como una variable aleatoria discreta con una función de masa de probabilidad, dejando de lado los modelos de probabilidad continua.

Modelo de contrato de requisitos En la mayoría de las contribuciones para el modelado de calidad en el contexto del problema de selección de servicios la noción de requisito de calidad se relega a un criterio de optimización implícito [118, Sección 5.1], donde el objetivo es minimizar o maximizar el valor de cada atributo según su naturaleza. En cambio, consideramos que los requisitos de calidad son elementos clave ya que permiten que la selección de servicios sea específica para las necesidades de cada cliente. Bajo esta visión, nuestra propuesta de lenguaje para describir requisitos de calidad probabilísticos se vuelve relevante.

Otras contribuciones relevantes para el problema que abordamos en este capítulo se pueden encontrar en el campo de la verificación cuantitativa probabilística de sistemas basados en servicios [27]. Allí se propone un marco más clásico de verificación donde los sistemas se representan con modelos de Markov y se busca determinar la validez de fórmulas lógicas temporales probabilísticas que representan las propiedades de calidad de servicio. En [28], el enfoque general se enriquece con una noción relativa de validez dentro de un intervalo de confianza. La principal diferencia entre estos enfoques y el nuestro se basa en el modelo del sistema que se está analizando. Mientras que los enfoques de verificación de modelos probabilísticos requieren un modelo basado en estados del comportamiento del sistema, nuestro enfoque solo necesita un modelo de los valores tomados por los atributos de calidad de dicho sistema. Esto es una consecuencia de abordar el problema desde la perspectiva de la selección de servicios en computación orientada a servicios, donde los servicios se consideran entidades opacas desde la perspectiva del consumidor (es decir, el consumidor no tiene conocimientos de su comportamiento interno).

Hay tres ingredientes clave en nuestro enfoque y, según nuestro conocimiento, nuestra

propuesta es la primera en abordar simultáneamente todos estos aspectos: 1) proponemos un lenguaje de modelado capaz de manejar un número ilimitado de atributos de calidad, de los cuales no hay una interpretación implícita (es decir, solo requerimos que puedan tengan una interpretación métrica en cada ejecución concreta del servicio), 2) nos centramos en distribuciones de probabilidad continuas, capaces de capturar una caracterización más granular del comportamiento de los servicios, y 3) proponemos el uso de distribuciones de probabilidad multivariada, lo que permite la caracterización de las interdependencias entre el comportamiento probabilístico de los atributos de calidad.

El resto del capítulo está organizado de la siguiente manera. En la sección 6.2 presentamos las definiciones los contratos de provisión y de requisitos calidad, junto con una relación de satisfacción; también enmarcamos el problema de selección de servicios en términos de estos tres elementos. En la sección 6.3 proponemos un algoritmo que implementa la relación de satisfacción propuesta en la sección 6.2. En la sección 6.4 proponemos una metodología para aprender contratos de calidad probabilísticos a partir de datos muestreados del comportamiento del mundo real de los servicios, sacando del proveedor de servicios la responsabilidad de producirlo. Finalmente, en la sección 6.5 sacamos algunas conclusiones y proponemos direcciones de investigación adicionales. Los resultados presentados en este capítulo fueron previamente publicados como *preprint* en [110].

6.2. Acuerdo de nivel de servicio (SLA) para atributos probabilísticos

En esta sección presentamos las principales contribuciones de este capítulo. Comenzamos proporcionando una definición probabilística de contratos de provisión y de requisitos, junto con una definición de la relación de satisfacción para determinar compatibilidad entre estos.

6.2.1. Contrato de calidad de servicio probabilístico

Comencemos discutiendo cómo se modela el comportamiento de calidad de servicio probabilístico de un servicio de software. Consideremos el ejemplo presentado en [113], donde se tiene un repositorio con un conjunto de servicios cuya funcionalidad es validar una dirección de correo electrónico. Al llamar a uno de estos servicios, el cliente proporciona una dirección de correo electrónico y el servicio indica si es una dirección válida y existente. Diferentes servicios en este conjunto cuentan con diferentes valores de velocidad de procesamiento (desde ahora TP, abreviando del inglés *throughput*) y tiempo de respuesta (desde ahora RT, abreviando del inglés *response time*). Estamos interesados en la naturaleza probabilística de los atributos cuantitativos. Dado un servicio candidato, nos gustaría poder capturar una situación como la que se muestra en la Figura 6.1, donde el valor de TP ofrecido por un servicio dado se sabe que sigue una distribución *Gaussiana* con una *media* de 50 Mbps, y el valor de RT se sabe que sigue una distribución *Gamma* con una media de 300 ms.

A partir de ahora, las definiciones y resultados se presentarán asumiendo un conjunto fijo, pero arbitrario, de atributos cuantitativos $A = \{a_1, \ldots, a_n\}$.

Definición 10 (Contrato de provisión probabilístico)**.** *Un* contrato de provisión probabilístico *es un vector aleatorio* $\mathbf{X} = (X_1, \ldots, X_n)$*, donde cada variable aleatoria* X_i *se interpreta como el valor del atributo cuantitativo* a_i *y se caracteriza mediante una función de distribución acumulada conjunta:* $F_{\mathbf{X}}(x_1, \ldots, x_n) = \mathbb{P}(X_1 \leq x_1, \ldots, X_n \leq x_n)$*.*

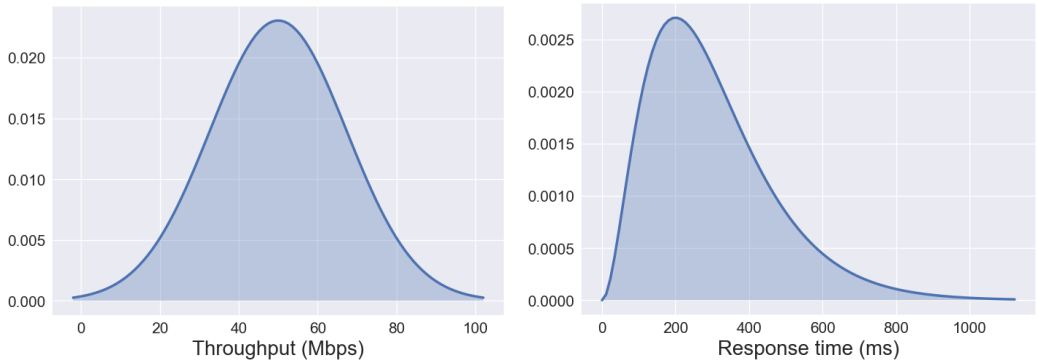


Fig. 6.1: Contrato de calidad de servicio probabilístico donde TP sigue una distribución Gaussiana con $\mu = 50$ y $\sigma^2 = 300$ y RT sigue una distribución Gamma con $\alpha = 3$ y $\beta = \frac{1}{100}$.

Además, si $F_{\mathbf{X}}$ es diferenciable, entonces $\mathbf{X}$ se puede caracterizar mediante una función de densidad de probabilidad (PDF, por sus siglas en inglés) conjunta: $f_{\mathbf{X}}(x_1, \ldots, x_n) = \frac{\partial^n F(x_1,\ldots,x_n)}{\partial_{x_1} \ldots \partial_{x_n}}$

Considerando los atributos TP y RT, mencionados en el ejemplo de la Figura 6.1, en ese caso el contrato de provisión es el vector aleatorio $\mathbf{X} = (X_1, X_2)$, donde X_1 y X_2 representan el valor de TP y RT, respectivamente. Entonces, utilizando las definiciones de las distribuciones gaussiana y gamma, la PDF se define como

$$f_{\mathbf{X}}(x_1, x_2) = f_{X_1}(x_1) \cdot f_{X_2}(x_2)$$
$$\text{donde } f_{X_1}(x_1) = \frac{1}{\sigma\sqrt{2\pi}} \exp[-\frac{1}{2}\left(\frac{x_1 - \mu}{\sigma}\right)^2] \text{ con } \mu = 50, \sigma^2 = 300$$
$$\text{y } f_{X_2}(x_2) = \frac{\beta^\alpha}{\Gamma(\alpha)} {x_2}^{\alpha-1} \exp(-\beta x_2) \text{ con } \alpha = 3, \beta = \frac{1}{100}.$$

Nótese que, como se mostró en la Figura 6.1, y como se ve reflejado en el hecho de que pudimos definir la PDF conjunta como el producto de las PDF marginales, el comportamiento de TP es independiente del comportamiento de RT en este modelo. Sin embargo, en general, este no es el caso; es bien sabido que, en la práctica, TP y RT tienen una correlación negativa (es decir, un alto RT implica un bajo TP, y viceversa). Este es sólo un ejemplo de situaciones que requieren necesariamente un enfoque que pueda de manejar una distribución de probabilidad multivariada compuesta, capaz de modelar la correlación entre los atributos cuantitativos. En la Figura 6.2 se presenta un ejemplo donde TP y RT siguen las distribuciones descritas en la Figura 6.1, pero están negativamente correlacionados. En este caso, tenemos una distribución bivariada $\mathbf{X}' = (X_1, X_2)$, donde

$$X_1 \sim \mathit{Gaussian}(\mu, \sigma^2) \text{ con } \mu = 50, \sigma^2 = 300 \text{ y}$$
$$X_2 \mid X_1 \sim \mathit{Gamma}(\alpha - \frac{X_1 - \mu}{\mu}, \beta), \text{ con } \alpha = 3, \beta = \frac{1}{100}.$$

Se observa que el primer parámetro de la distribución *Gamma*, que gobierna la media de X_2, está negativamente correlacionado con el resultado de X_1. La Definición 10 permite modelar tanto el ejemplo anterior donde los atributos cuantitativos son independientes, como este ejemplo donde los atributos cuantitativos están correlacionados.

En resumen, un contrato de provisión se modela con un vector aleatorio cuya dimensión coincide con el número (y orden) de la lista de atributos cuantitativos de interés A.

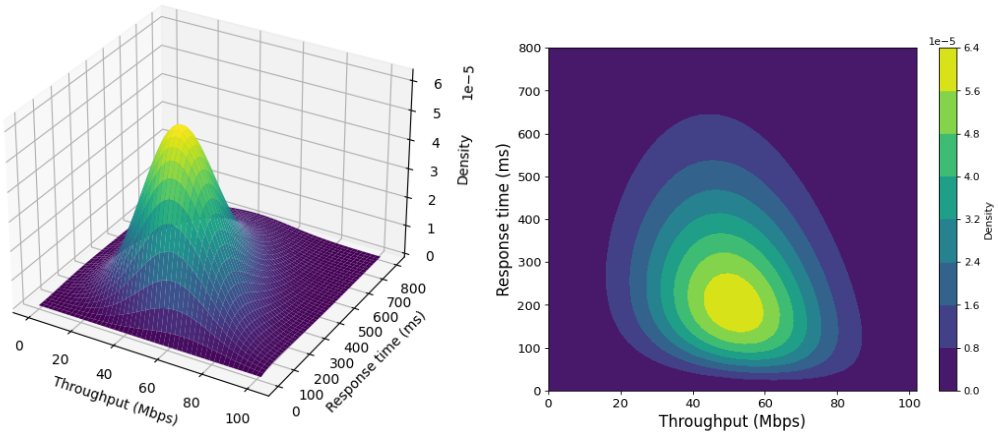


Fig. 6.2: Representación de un perfil de calidad donde los atributos de la velocidad de procesamiento y el tiempo de respuesta están negativamente correlacionados.

6.2.2. Requisitos de calidad probabilístico

Cuando un perfil de calidad de un servicio se caracteriza por una PDF conjunta, es posible calcular la probabilidad de que la ejecución de un servicio adopte valores para los atributos dentro de una región específica, integrando la PDF conjunta sobre esa región. Esto sostiene la idea de brindar una definición de requisito de calidad que admita determinar su satisfacción por un servicio mediante consultas de este tipo. Primero presentamos restricciones básicas sobre los atributos cuantitativos y luego definimos un lenguaje para expresar requisitos de calidad de servicio.

Definición 11 (Restricción de calidad probabilística)**.** *Una* restricción de calidad probabilística *sobre el conjunto de atributos A es una 3-tupla que contiene:*

1) una región de valores $R \subseteq \mathbb{R}^n$,

2) una cota inferior $p_{min} \in [0, 1]$, y

3) una cota superior $p_{max} \in [0, 1]$.

El conjunto de restricciones de calidad sobre A se denotará como $QoSCstr(A)$.

La interpretación intuitiva es que R caracteriza una región de valores de calidad de interés, donde cada punto $\mathbf{x} \in R$ representa una combinación específica de valores de los atributos cuantitativos, y p_{min} y p_{max} son los límites para la probabilidad de que los atributos cuantitativos tomen valores en R. Entonces, podemos considerar un escenario como el mostrado en la Figura 6.3, donde un cliente está interesado en un servicio con alto valor de TP y bajo valor de RT.

La región hacia la derecha de Figura 6.3 muestra una región R_{buena} de interés para el cliente. R_{buena} impone un alto TP (entre 60 y 100 Mbps), un RT rápido (por debajo de 300 ms) y, además, excluye la sección de baja calidad dentro de esos valores (la línea diagonal excluye la esquina superior izquierda de baja calidad en RT y TP lento). La siguiente restricción de calidad expresa que el cliente está interesado en un servicio que tenga una alta probabilidad de caer en R_{buena}:

$$\langle R_{\text{buena}},\ p_{min} : 0.6,\ p_{max} : 1 \rangle$$

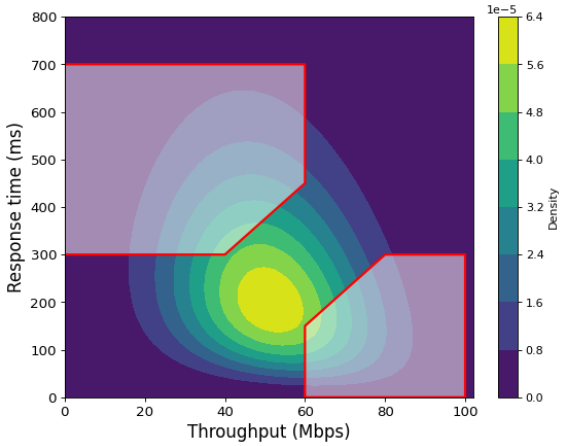


Fig. 6.3: Las regiones resaltadas en rojo indican TP alto con RT bajo, y TP bajo con RT alto, respectivamente.

Por simplicidad, omitiremos escribir p_{min} y p_{max} cuando sean 0 y 1, respectivamente. De manera similar, la región hacia la izquierda de Figura 6.3, que denominaremos R_{mala}, caracteriza un TP bajo y un RT lento. El cliente podría estar interesado establecer una cota superior baja para su probabilidad esperada.

A continuación, definimos la noción de satisfacción de una restricción de calidad probabilística por un contrato de provisión.

Definición 12 (Satisfacción de restricción de calidad probabilística)**.** *Dado un contrato de provisión* $\mathbf{X}$ *y una restricción de calidad* $\varphi = \langle R,\ p_{min},\ p_{max}\rangle$*, decimos que* $\mathbf{X}$ *satisface* φ *(denotado* $\mathbf{X} \models \varphi$*) si y solo si* $p_{min} \leq \mathbb{P}(\mathbf{X} \in R) \leq p_{max}$*. Si el contrato de provisión* $\mathbf{X}$ *está descripto en términos de una PDF conjunta* $f_{\mathbf{X}}$*, entonces el problema de satisfacción* $\mathbf{X} \models \varphi$ *es equivalente a decidir si la integral de la PDF conjunta sobre la región* R *cae dentro de los límites* p_{min} *y* p_{max}*:*

$$p_{min} \leq \int_R f_{\mathbf{X}}(x_1, \ldots, x_n)\ dx_1 \ldots dx_n \leq p_{max}$$

A continuación, utilizamos las restricciones de calidad probabilísticas como elementos básicos para la definición de un lenguaje más expresivo para caracterizar los requisitos de calidad de servicio.

Definición 13 (Requisito de calidad probabilístico)**.** *Dado un conjunto de variables proposicionales* $\mathcal{P}$*, un* requisito de calidad probabilístico *es una fórmula en el lenguaje de la siguiente gramática,*

$$\Phi ::= \top \mid p \mid \varphi \mid \neg\Phi \mid \Phi \vee \Phi$$

donde $\varphi \in QoSCstr(A)$ *y* $p \in \mathcal{P}$*. Denotaremos el conjunto de requisitos de calidad probabilísticos sobre* A *y* $\mathcal{P}$ *como* $QoSReq(A, \mathcal{P})$*.*

La interpretación intuitiva es que los requisitos de calidad son combinaciones proposicionales de restricciones de calidad y variables proposicionales puras. Si bien es natural construir requisitos de calidad a partir de restricciones de calidad porque son el elemento clave en el modelado del comportamiento de la calidad de servicio, las variables proposicionales puras desempeñan un papel importante para facilitar la tarea de diseño de contratos

que expresan además aspectos discretos de los modelos. El siguiente requisito de calidad Φ_1 expresa una probabilidad alta de que los valores caigan en R_{buena} y una probabilidad baja de que caigan en R_{mala},

$$\Phi_1 \equiv \langle R_{\text{buena}},\ p_{min} : 0{,}6\rangle \wedge \langle R_{\text{mala}},\ p_{max} : 0{,}3\rangle$$

Un cliente distinto podría estar dispuesto a aceptar cualquiera de las dos condiciones, siempre y cuando se pueda identificar cuál de ellas se cumplió. Esto se puede hacer recurriendo a variables proposicionales de la siguiente manera:

$$\begin{gathered}(\langle R_{\text{buena}},\ p_{min} : 0{,}6\rangle \vee \langle R_{\text{mala}},\ p_{max} : 0{,}3\rangle)\\ \wedge\ (p_1 \Leftrightarrow \langle R_{\text{buena}},\ p_{min} : 0{,}6\rangle) \wedge (p_2 \Leftrightarrow \langle R_{\text{mala}},\ p_{max} : 0{,}3\rangle)\end{gathered}$$

De esta manera, se limita el valor de verdad de cada variable proposicional a la satisfacción de cada restricción de calidad. Las variables proposicionales p_1, p_2 pueden luego utilizarse como variables de decisión en otro fragmento de la misma fórmula, o pueden ser leídas por el cliente si el procedimiento de verificación devuelve una asignación satisfactoria.

La siguiente definición extiende la noción de satisfacción a requisitos de calidad.

Definición 14 (Satisfacción de requisito de calidad probabilístico)**.** *Un contrato de provisión* $\mathbf{X}$ *satisface un requisito de calidad probabilístico* $\Phi \in QoSReq(A, \mathcal{P})$ *si y solo si existe una valuación* $v : \mathcal{P} \to \{\top, \bot\}$ *(es decir, una valuación de las variables proposicionales) tal que* $\mathbf{X}, v \models \Phi$*. Con la relación de satisfacción* $\models$ *definida de la siguiente manera:*

$\mathbf{X}, v$	$\models$	$\top$	*se cumple siempre*
$\mathbf{X}, v$	$\models$	p	*si y solo si* $v(p) = \top$
$\mathbf{X}, v$	$\models$	φ	*si y solo si* $\mathbf{X} \models \varphi$ *(Definición 12), donde* $\varphi \in QoSCstr(A)$.
$\mathbf{X}, v$	$\models$	$\neg\Phi$	*si y solo si* $\mathbf{X}, v \models \Phi$ *no se cumple*
$\mathbf{X}, v$	$\models$	$\Phi \vee \Phi'$	*si y solo si* $\mathbf{X}, v \models \Phi$ *o* $\mathbf{X}, v \models \Phi'$ *donde* $\Phi, \Phi' \in QoSReq(A, \mathcal{P})$.

Considere el primer ejemplo de requisito de calidad Φ_1 definido previamente y el contrato de provisión $\mathbf{X}'$ que grafica la Figura 6.2. Bajo esta definición, $\mathbf{X}' \models \Phi_1$ no se cumple porque, aunque $\int_{R_{\text{mala}}} f_{\mathbf{X}'} \approx 0{,}24$ satisface la segunda restricción, $\int_{R_{\text{buena}}} f_{\mathbf{X}'} \approx 0{,}13$ no satisface la primera restricción, lo que hace que la conjunción no sea satisfactoria.

6.3. Análisis automático de satisfacción de requisitos de calidad

Un algoritmo para decidir la relación de satisfacción de la Definición 14 resolvería efectivamente el problema de selección de servicios, al proporcionar un criterio de decisión para cada servicio del repositorio. En esta sección damos tal algoritmo, que consta de dos componentes: la integración numérica de PDF para determinar la satisfacción de las restricciones de calidad y la satisfacción algorítmica de la estructura proposicional del requisito de calidad. Comenzamos definiendo una simple transformación de los requisitos de calidad que permitirá el uso de *SAT-solvers* para tratar eficientemente con la estructura proposicional de la fórmula. El espíritu de esta transformación es similar a la abstracción proposicional de las especificaciones convexas presentada en el Capítulo 4.

Definición 15 (Abstracción proposicional de requisito de calidad)**.** *Sea A un conjunto de atributos cuantitativos, $\mathcal{P}$ un conjunto de variables proposicionales y $\Phi \in QoSReq(A, \mathcal{P})$ un requisito de calidad probabilístico. Sea C el conjunto de restricciones de calidad probabilísticas presentes en Φ. Definimos la* abstracción proposicional *de Φ como una fórmula proposicional Φ_B, obtenida de Φ reemplazando cada restricción de calidad probabilística con una nueva variable proposicional $c_i, i \in \{1, \ldots, |C|\}$, y una función $\delta : \{c_1, \ldots, c_{|C|}\} \to C$ que mapea cada variable proposicional a la restricción de calidad que representa.*

El algoritmo 1 describe el procedimiento para decidir si se cumple o no $\mathbf{X} \models \Phi$. Un diagrama de ejecución del algoritmo se muestra en la Figura 6.4. Esencialmente, hay tres etapas: (1) se construye la abstracción proposicional de Φ, (2) se evalúa cada restricción de calidad para decidir el valor de verdad de la variable proposicional correspondiente y (3) se comprueba si la fórmula proposicional resultante es satisfactible. El algoritmo se basa en dos procedimientos externos: SAT y ComputeIntegral. Por un lado, existen distintas implementaciones modernas y eficientes del primero [51, 63, 100] que se pueden emplear fácilmente para resolver el problema de satisfacibilidad proposicional.

Data: Función de densidad $f_{\mathbf{X}}$ del contrato de provisión, requisito de calidad Φ.
Result: Valor de verdad que indica si se cumple $\mathbf{X} \models \Phi$.
1 Sea C el conjunto de restricciones de calidad que aparecen en Φ.
2 Sea δ un diccionario de variables proposicionales a restricciones de calidad.
3 $\Phi_B \leftarrow \Phi$
4 **foreach** *restricción de calidad φ en C* **do**
5 | Sea c una variable proposicional nueva.
6 | Reemplazar las apariciones de φ en Φ_B por c.
7 | $\delta[c] \leftarrow \varphi$
8 **foreach** *variable proposicional c en el diccionario δ* **do**
9 | Sea $\langle R,\ p_{min},\ p_{max}\rangle$ la restricción de calidad $\delta[c]$.
10 | **if** $p_{min} \leq$ `ComputeIntegral`$(f_{\mathbf{X}}, R) \leq p_{max}$ **then**
11 | | $\Phi_B \leftarrow \Phi_B \wedge c$
12 | **else**
13 | | $\Phi_B \leftarrow \Phi_B \wedge \neg c$
14 **return** `SAT`(Φ_B)

Algorithm 1: Procedimiento de decisión para la satisfacción de requisitos de calidad

Por otro lado, dada una función de densidad de probabilidad conjunta $f_{\mathbf{X}}$ y una región $R \subseteq \mathbb{R}^n$, el procedimiento ComputeIntegral$(f_{\mathbf{X}}, R)$ debe calcular $\int_R f_{\mathbf{X}}$. La implementación de este procedimiento requiere una discusión más detallada que dareamos a continuación.

6.3.1. Integración numérica de un contrato de calidad $f_{\mathbf{X}}$ en la región R

A continuación haremos un breve repaso del problema de integración numérica en múltiples dimensiones y distintos enfoques para resolverlo. Como quedará claro más adelante, en este trabajo nos decidiremos por un enfoque utilizando aproximación Monte Carlo.

Obsérvese que, dado que los límites de integración ya están especificados por R, no estamos interesados en el problema de integración indefinida o simbólica [23]. En cambio, el

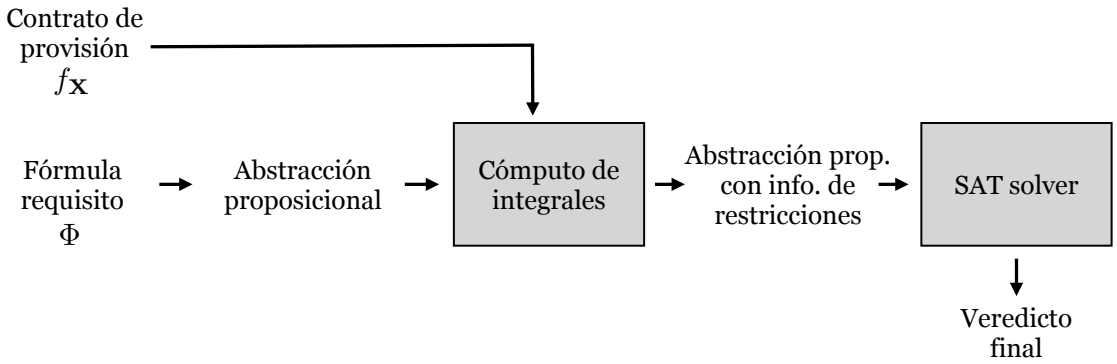


Fig. 6.4: Diagrama de ejecución del algoritmo 1.

problema al que nos enfrentamos es la evaluación numérica de una integral definida [38]. En general, los algoritmos para integrar numéricamente una función dada se clasifican según el tipo de función y el tipo de regiones que permiten integrar.

En nuestro enfoque, las funciones son PDFs arbitrarias que surgen del modelado probabilístico del contrato de provisión (el comportamiento de los atributos cuantitativos de un servicio). Basándonos en esta interpretación general de los contratos de provisión, desestimamos aquellos métodos de integración diseñados para clases específicas de funciones, como los adaptados específicamente para funciones polinomiales [40]. En segundo lugar, las regiones resultan de las restricciones presentes en los requisitos de calidad del cliente, lo que lleva al problema de elegir un lenguaje para describirlas. Esto requiere considerar dos aspectos importantes; por un lado, la expresividad del lenguaje, relativizada por su utilidad (tanto desde el punto de vista de la usabilidad del lenguaje, como desde la relevancia práctica de las regiones que puede describir) y, por otro lado, la existencia de algoritmos eficientes para resolver la integración numérica sobre regiones descriptas en el lenguaje.

El enfoque más básico para la integración numérica en múltiples dimensiones es encontrar una forma de reducirla a una iteración de integrales unidimensionales, lo que por lo general asume que la región R es un rectángulo de n dimensiones [145]. La integración sobre un símplice de n dimensiones también se ha estudiado exhaustivamente [70]. Partiendo de esto, se han desarrollado una multitud de métodos para regiones más generales que se pueden descomponer en aquellas elementales [38, Cap. 5.5]. Esto se debe a que el valor de la integral sobre la unión de regiones disjuntas es igual a la suma de los valores de las integrales sobre cada un de las regiones. En este contexto, los politopos convexos, que ya hemos introducido en la Definición 8 del Capítulo 5, son probablemente el tipo de regiones más estudiado [12, 25, 157], y se han utilizado para muchos propósitos en el campo de la verificación de software [7]. Los politopos convexos son atractivos porque proporcionan una forma de delinear un conjunto de valores de interés y, además, las regiones no convexas se pueden descomponer en una unión de politopos convexos disjuntos. Estas observaciones nos llevan a elegir una familia finita y disjunta de politopos acotados [1] para caracterizar las regiones que definen las restricciones de calidad en la Definición 11.

Existen muchas definiciones diferentes para el politopo [157], pero como se mostró en la Definición 8, adoptamos la basada en la intersección de semiespacios. Cuando el espacio de

[1] Se debe tener en cuenta que se podría admitir el uso de familias de politopos no disjuntas, ya que se pueden descomponer en una familia de politopos disjuntos mediante el uso del *Principio de inclusión-exclusión* como mostramos en el Capítulo 5.

valores de un politopo está acotado tenemos un *politopo acotado* que, como se establece en [157, Thm. 2.15], se puede representar de forma equivalente como un conjunto de símplices que no tienen puntos interiores en común. La definición precisa de símplice no es relevante para nuestro propósito (ver [157, Cap. 0]), se puede pensar como la generalización de un triángulo a n dimensiones. El proceso de construcción de esta representación simplicial a partir de un politopo se denomina *triangulación* o *descomposición* y existen varios algoritmos para lograrlo [68, Sec. 3.1] (alternativamente [25]). Un algoritmo para realizar esta descomposición, junto con un algoritmo para integrar sobre símplices, constituye un procedimiento efectivo para integrar numéricamente funciones sobre regiones caracterizadas como politopos acotados. La complejidad del procedimiento está dominada principalmente por la cantidad de símplices obtenidos por la descomposición, ya que todos ellos deben integrarse por separado. El número de símplices resultantes de la descomposición de un politopo acotado depende en gran medida de la geometría específica del politopo que se está descomponiendo, llegando a magnitudes en $O(2^n)$ o $O(n!)$, donde n es el número de dimensiones (en nuestro contexto, el número de atributos cuantitativos). Si el número de atributos cuantitativos es pequeño, el enfoque es aceptable, pero si es grande, la complejidad amenaza su aplicabilidad.

En este trabajo decidimos superar esta limitación apoyándonos en el cálculo de las integrales mediante un algoritmo de aproximación de Monte Carlo [55]. Para que sea posible el cálculo de la integral $\int_R f_{\mathbf{X}}$ mediante este método, es necesario poder realizar las siguientes operaciones:

a) calcular el volumen $vol(R)$,

b) muestrear uniformemente k puntos $\{\mathbf{x_1}, \dots \mathbf{x_k}\} \in R$, y

c) dado un punto $\mathbf{x} \in \mathbb{R}^n$, calcular $f_{\mathbf{X}}(\mathbf{x})$.

Luego, podemos calcular un estimador de la integral, $Q_k = V\frac{1}{k}\sum_{i=1}^{k} f_{\mathbf{X}}(\mathbf{x_i})$ que, según la ley fuerte de los grandes números [49, Chap. 2], se sabe que es una aproximación del valor de la integral, en el sentido de que $\mathbb{P}(\lim_{k\to\infty} Q_k = \int_R f_{\mathbf{X}}) = 1$. Una presentación general de esta técnica puede encontrarse en [55, Sec. 6.2].

Revisemos ahora cómo vamos a satisfacer los supuestos mencionados para la implementación del algoritmo de aproximación de Monte Carlo:

a) **Cálculo del volumen de la región poliédrica:** la complejidad de calcular el volumen exacto de un politopo acotado está demostrado que es **#P-Hard** [50], pero se pueden encontrar implementaciones prácticas en [25]. No obstante, se han desarrollado algoritmos de estimación eficientes, como el presentado en [62] que ya utilizamos en el Capítulo 5. Como vimos, este algoritmo de estimación se basa en un *algoritmo de monte carlo multiphase* y se demuestra que tiene una complejidad algorítmica de $O^*(mn^3)$, donde n es la dimensión y m es el número de facetas (es decir, el número de desigualdades lineales).

b) **Muestreo uniforme de puntos en la región poliédrica:** se pueden utilizar varias estrategias, por ejemplo, algoritmos de *Monte Carlo de Cadena de Markov* como la *caminata de Dikin*, introducida en [89], con complejidad algorítmica en $O(mn)$, donde n es la dimensión del politopo y m es el número de restricciones. Nuevos enfoques [31], también basados en algoritmos MCMC, presentan diferentes estrategias de muestreo con mejor complejidad bajo condiciones específicas, por ejemplo, la *caminata de Vaidya*

tiene una complejidad en $O(m^{0,5}n^{1,5})$, lo cual es significativamente mejor cuando $m \gg n$.

c) **Evaluación de la función de densidad:** asumimos a lo largo de este trabajo que la función de densidad $f_{\mathbf{X}}$ tiene una representación finita, como una *expresión algebraica* o un *programa probabilístico* [67], que proporciona pasos computacionales precisos para su evaluación.

Bajo estas condiciones, por lo presentado en [55, Thm. 6.1], sabemos que el error $|Q_k - \int_R f_{\mathbf{X}}|$ es de orden $1/\sqrt{k}$. El factor $\sqrt{k}$ proviene de la distribución dada por el teorema central del límite aplicado al estimador Q_k. El hecho de que esta tasa de convergencia no dependa de la dimensión n es una de las ventajas del método sobre otros métodos numéricos de integración.

De esta manera queda establecida la implementación del procedimiento ComputeIntegral($f_{\mathbf{X}}, R$) requerido por el algoritmo 1. Así, el algoritmo 1 proporciona un procedimiento efectivo para decidir la satisfacción de un requisito de calidad por un contrato de provisión. En la siguiente sección, proponemos un enfoque para la construcción automática del contrato de provisión $f_{\mathbf{X}}$ a partir de datos muestreados de la ejecución real del servicio que está siendo modelado.

6.4. Construyendo automáticamente contratos de calidad a partir de datos

Una parte significativa de la literatura sobre sistemas de software basados en servicios se dedica a supervisar el comportamiento en tiempo de ejecución de los servicios y, casi siempre, esta información se utiliza para abordar el problema de las violaciones de contrato. En ese enfoque se confía inicialmente en que los proveedores de servicio actúen de acuerdo con el contrato que prometen y se los castiga si incumplen el mismo. En cambio, en esta sección tomamos un punto de vista opuesto en el que el contrato de provisión debe reflejar el comportamiento real del servicio. Es decir, debe ser el resultado del aprendizaje a partir de su comportamiento en el mundo real, mediante el monitoreo de los atributos en tiempo de ejecución.

Dado el conjunto de atributos cuantitativos $A = \{a_1, \ldots, a_n\}$ de un servicio, un *registro de valores de calidad* será un vector $\mathbf{x} \in \mathbb{R}^n$. La idea es que un registro de valores de calidad formaliza los valores tomados por los atributos cuantitativos en una única ejecución del servicio. El enfoque de esta sección será estimar el contrato de provisión de un servicio mediante la recopilación de sus sucesivos registros de calidad y la estimación de una PDF conjunta a partir de ellos. En el campo de la probabilidad y la estadística, esto es un problema conocido como *estimación de densidad de probabilidad* [139] y existen varios métodos para estimar funciones de densidad de probabilidad a partir de una muestra de la distribución subyacente. Ilustramos el enfoque recurriendo a la técnica bien conocida de *estimación de densidad por núcleos* (*kernel density estimation*) [58, 133].

Definición 16 (Estimación de densidad por núcleos)**.** *Dado un conjunto de observaciones* $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ *en* $\mathbb{R}^n$*, el estimador de densidad por núcleos (KDE)* $\hat{f}(\mathbf{x})$ *se define como:*

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{m} K_h(\mathbf{x} - \mathbf{x}_i)$$

donde K_h *es una función de núcleo que satisface* $\int_{\mathbb{R}^n} K_h(\mathbf{x})\ d\mathbf{x} = 1$*, siendo* h *el* ancho de banda *o* parámetro de suavización.

Existen muchos tipos de funciones de núcleo que se pueden utilizar en KDE y cada una de ellas representa una interpretación diferente de cómo se distribuye el peso asociado a una observación en su vecindario. Una vez que el núcleo está fijo, el ancho de banda juega un papel crucial en cómo ese KDE estima la distribución de las observaciones. Si el ancho de banda es demasiado bajo, el KDE sobreajustará las observaciones y estará demasiado influenciado por los valores concretos; en cambio, si es demasiado alto, no podrá estimar adecuadamente y adaptarse al peso de las observaciones. Si bien no hay forma de conocer de antemano el ancho de banda óptimo, existen varias estrategias para identificar buenos candidatos. Los métodos más rápidos para aproximar el ancho de banda adecuado se conocen como reglas generales (*"rules of thumb"*). Ejemplos de dichas reglas para núcleos Gaussianos son la *regla de Silverman* para KDEs univariados [139, pp. 48, eq. 3.31], su versión multivariada [137, pp. 152, eq. 6.41], y la *regla de Scott* [137, pp. 152, eq. 6.42]. Para otros núcleos, [137, tabla 6.3] proporciona constantes por las cuales se deben multiplicar los anchos de banda obtenidos con las reglas anteriores. La validación cruzada completa [135] es una técnica para obtener una combinación óptima de un tipo de kernel y un ancho de banda a partir de los datos disponibles. Este método requiere significativamente más recursos computacionales, pero su uso es aceptable si la actualización de los parámetros se realiza selectivamente.

La efectividad de un enfoque de estimación de contratos de calidad, en el contexto de la determinación de satisfacción de un requisito de calidad por parte de un servicio, depende de la capacidad de la PDF estimada para caracterizar adecuadamente el comportamiento del servicio. Conceptualmente podemos pensar que existe, por un lado, una distribución de probabilidad exacta que caracteriza el comportamiento del servicio, pero que tal vez no podemos conocer, y, por otro lado, una distribución de probabilidad estimada a partir de los datos disponibles. Idealmente, nos gustaría que el Algoritmo 1 aplicado a un contrato de calidad estimado $\widehat{f_{\mathbf{X}}}$ y un requisito de calidad Φ propocione el mismo veredicto que si se aplicara a la distribución de probabilidad ideal $f_{\mathbf{X}}$ y el requisito de calidad Φ, al menos con una probabilidad alta. A continuación, presentamos una evaluación preliminar de esta pregunta para el ejemplo que venimos desarrollando en este capítulo (Figura 6.2) y, en la próxima sección, presentamos una evaluación más completa de este enfoque.

En la Tabla 6.1 mostramos una comparación entre el cálculo aproximado del volumen de las regiones representadas en la Figura 6.3 sobre el contrato de provisión exacto de la Figura 6.2 y sobre KDEs entrenados con diferentes tamaños de muestra, basados en combinaciones óptimas de kernel y ancho de banda obtenidos mediante validación cruzada completa. En la Figura 6.5 mostramos un gráfico de la PDF estimada por KDE utilizando 1000 nuestras de la PDF exacta presentada en la Figura 6.2 y utilizando una validación cruzada completa para optimizar la elección del kernel y el ancho de banda.

A continuación, presentamos una evaluación exhaustiva de la viabilidad de aprender contratos de provisión a partir de datos empíricos.

6.4.1. Resultados experimentales

Nos interesa evaluar experimentalmente cuán bien aproxima la integral de la estimación KDE a la integral de la distribución real y, en consecuencia, cómo afecta la estimación KDE

KDE sample size	(kernel, bandwidth, time)	Región R_{buena}		Región R_{mala}	
		Integ.	Rel. err.	Integ.	Rel. err.
100	(exp., 13.107, 41.1s)	0.1729	(23.57 %)	0.2021	(11.45 %)
1000	(exp., 6.903, 7m 59,3s)	0.1546	(10.49 %)	0.2694	(18.03 %)
10000	(exp., 4.145, 135m 4.1s)	0.1312	(6.20 %)	0.2400	(5.16 %)
Distribución exacta		0.1399	(100 %)	0.2282	(100 %)

Tab. 6.1: Comparación entre el cálculo de la integral de las regiones de la Figura 6.3 utilizando la PDF exacta y las estimaciones KDE.

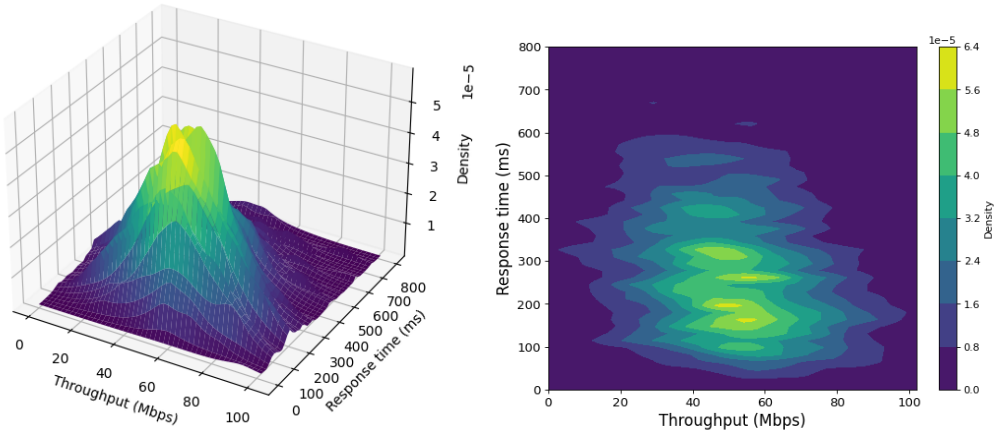


Fig. 6.5: Contrato de calidad estimado por KDE con un tamaño de muestra de 1000 y validación cruzada completa para la optimización de kernel y ancho de banda.

al procedimiento de decisión delineado en el Algoritmo 1. Para responder estas preguntas hemos generado automaticamente casos de estudio de manera aleatoria.

Configuración experimental. Las distribuciones de probabilidad generadas son normales multivariadas de la forma $\mathcal{N}(\mu, \Sigma)$, donde la media μ es un vector aleatorio dentro del hipercubo $[0, 10]^n$ y los valores de la matriz de covarianza Σ son aleatorios, garantizando correlación e interdependencia entre los atributos. Estos casos de estudio fueron generados mediante *programación probabilística* utilizando la biblioteca *TensorFlow Probability* [47]. Las regiones se generan a partir de muestrear aleatoriamente 2^n puntos dentro del hipercubo $[0, 10]^n$ y luego aplicar el algoritmo de *convex hull* de la biblioteca *SciPy* [147] para obtener la cápsula convexa de los puntos, efectivamente generando un politopo convexo. En todos los casos hemos fijado la cantidad de muestras para la integración Monte Carlo en 10000, ya que, por un lado, ha sido suficiente para obtener los valores de error relativo que buscabamos y, por otro lado, el objetivo de esta experimentación no es evaluar la precisión del método de integración Monte Carlo sino el impacto de la estimación KDE en el procedimiento de decisión delineado en el Algoritmo 1 cuando el método de cómputo de la integral está fijo. Para el cómputo de la estimación KDE hemos utilizado la implementación de la biblioteca *scikit-learn*.

Infraestructura de ejecución. 8-cores MacBook Pro (Apple M1) con 16GB de memoria. Python 3.10.8, TensorFlow 2.23, TensorFlow Probability 0.21, SciPy 1.11.2, scikit-learn 1.2.2.

Las preguntas de investigación que nos interesan responder son las siguientes:

- **RQ1:** ¿Cómo evoluciona el tiempo de ejecución del cálculo de la integral, en promedio, cuando aumenta la cantidad de muestras utilizadas para construir el KDE?
- **RQ2:** ¿Cómo evoluciona el error relativo de la estimación de la integral cuando aumenta la cantidad de muestras utilizadas para construir el KDE?
- **RQ3:** ¿Cómo evoluciona el error relativo de la estimación de la integral cuando aumenta la cantidad de atributos?
- **RQ4:** ¿Cuánto crece el tamaño de la muestra necesario para construir el KDE a medida que aumenta la cantidad de atributos, con el fin de mantener el error relativo por debajo del 5 %?
- **RQ5:** ¿Cuál es el impacto de un error relativo aproximado del 5 % en el cálculo del valor de verdad de fórmulas atómicas generadas aleatoriamente?

A continuación, presentamos los resultados de los experimentos realizados para responder estas preguntas.

RQ1: ¿Cómo evoluciona el tiempo de ejecución del cálculo de la integral, en promedio, cuando aumenta la cantidad de muestras utilizadas para construir el KDE?

Para responder esta pregunta, hemos fijado la cantidad de atributos en 5 y generado 100 casos de estudio aleatorios (pares de distribución y región). Para cada caso de estudio, hemos calculado la estimación KDE de la distribución utilizando $10^2, 10^3, 10^4, 10^5, 10^6$ y 10^7 muestras y luego hemos calculado la integral de la estimación KDE utilizando integración Monte Carlo. El tiempo de ejecución promedio para el cálculo de la integral se muestra en la Figura 6.6, donde ambos ejes están en escala logarítmica. El tiempo de construcción del KDE es despreciable (menor a 0,1 segundos) en todos los casos y por lo tanto no se muestra en la figura.

Como se puede observar, el tiempo de ejecución crece linealmente con la cantidad de muestras utilizadas para construir el KDE. Esto es esperable, ya que el cálculo de la integral se basa en la evaluación de la estimación KDE en un conjunto de puntos de tamaño fijo y el costo de evaluar la estimación KDE en un punto crece linealmente con la cantidad de muestras utilizadas para construir el KDE. Esto se debe a que la cantidad de términos de la sumatoria en la función a evaluar (ver la Definición 16) es igual a la cantidad de muestras utilizadas para construir el KDE.

RQ2: ¿Cómo evoluciona el error relativo de la estimación de la integral cuando aumenta la cantidad de muestras utilizadas para construir el KDE?

Para responer esta pregunta, utilizaremos las mismas distribuciones de probabilidad y regiones generadas para responder la RQ1. Los experimentos que realizaremos consisten en calcular el error relativo de la integral de la estimación KDE, con respecto a la integral

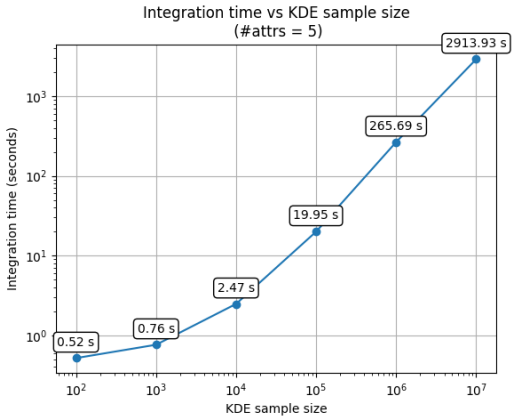


Fig. 6.6: Tiempo promedio de ejecución del cálculo de la integral de la estimación KDE en función de la cantidad de muestras utilizadas para construir el KDE, para 100 casos de estudio con 5 atributos.

de la misma región sobre la distribución real, en función de la cantidad de muestras utilizadas para construir el KDE. Realizaremos este experimento bajo las siguientes tres configuraciones:

1. fijar una distribución de probabilidad aleatoria y tomar 100 regiones aleatorias,
2. fijar una región aleatoria y tomar 100 distribuciones de probabilidad aleatorias, y
3. tomar 100 casos de estudio aleatorios (pares de distribución y región).

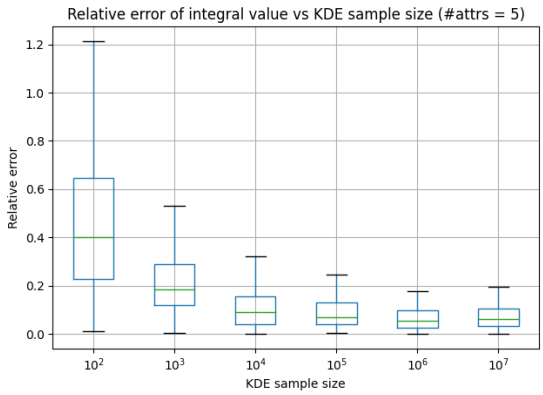
Los primeros dos experimentos nos permiten independizar el efecto de la distribución de probabilidad y la región en el error relativo de la estimación de la integral. El tercer experimento nos permite confirmar que los resultados de los dos primeros experimentos se mantienen cuando se varían tanto la distribución de probabilidad como la región. Los resultados de estos experimentos se muestran como diagramas de caja en las Figuras 6.7a, 6.7b y 6.7c, respectivamente. En este caso, el eje x está en escala logarítmica y el eje y está en escala lineal.

Se puede observar que el error relativo de la estimación de la integral disminuye a medida que aumenta la cantidad de muestras utilizadas para construir el KDE. No sólo eso, sino que la varianza del error relativo también disminuye a medida que aumenta la cantidad de muestras. Sin embargo, es notable que a medida que aumenta la cantidad de muestras utilizadas para construir el KDE, el error relativo de la integral disminuye a una tasa cada vez menor.

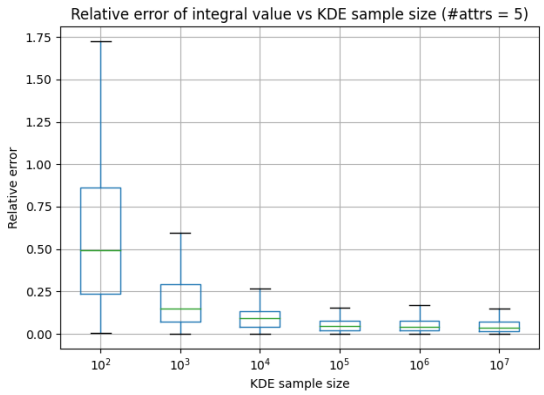
Hasta el momento, hemos fijado la cantidad de atributos en 5. La siguiente pregunta apunta a evaluar cómo afecta la cantidad de atributos al error relativo de la estimación de la integral.

RQ3: ¿Cómo evoluciona el error relativo de la estimación de la integral cuando aumenta la cantidad de atributos?

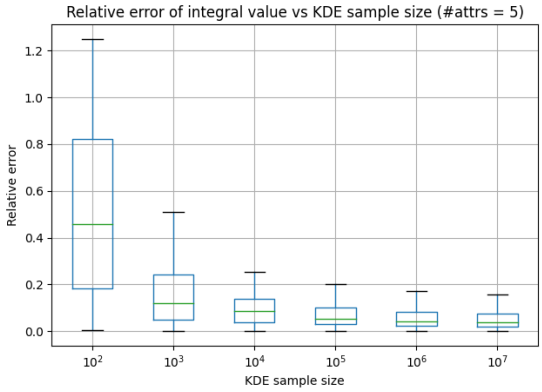
Para responder esta pregunta fijaremos la cantidad de muestras utilizadas para construir el KDE en 10^5 y llevaremos a cabo un experimento análogo a la configuración 3



(a) Distribución de probabilidad fija



(b) Región fija



(c) Distribución y región aleatorias

Fig. 6.7: Error relativo de la estimación de la integral de la estimación KDE en función de la cantidad de muestras utilizadas para construir el KDE, para 100 casos de estudio con 5 atributos.

del experimento realizado para RQ2, pero donde el número de atributos toma varlores en el rango $3, 4, 5, 6$ y 7. Es decir, tomaremos 100 casos de estudio aleatorios (pares de distribución y región) y calcularemos el error relativo de la integral de la estimación KDE con respecto a la integral de la misma región sobre la distribución real, en función de la cantidad de atributos. El resultado de este experimento se muestra en la Figura 6.8.

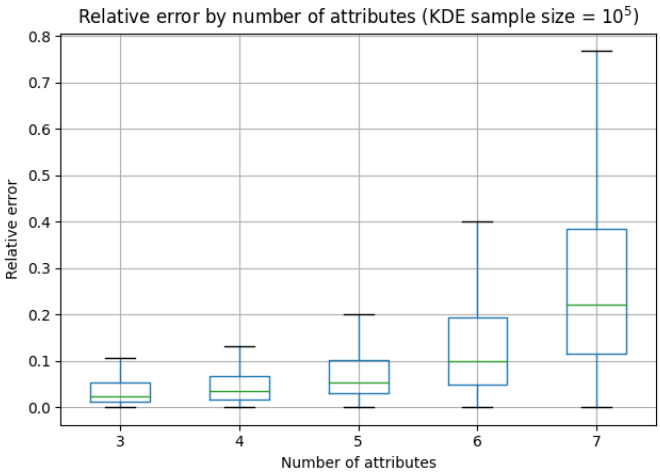


Fig. 6.8: Error relativo de la integral de la estimación KDE en función de la cantidad de atributos.

Se puede observar que el error relativo de la integral aumenta a medida que aumenta la cantidad de atributos, así como también su varianza. Esto sugiere que a mayor cantidad de atributos se necesita una mayor cantidad de muestras para construir el KDE si se quiere obtener una integral con un error relativo bajo. En la próxima pregunta evaluaremos esto más detalladamente.

RQ4: ¿Cuánto crece el tamaño de la muestra necesario para construir el KDE a medida que aumenta la cantidad de atributos, con el fin de mantener el error relativo por debajo del 5 %?

Para este experimento tomamos los resultados de la configuración número 3 del experimento realizado para la RQ2 (Figura 6.7c) y, además, realizamos un experimento de las mismas características para cantidad de atributos igual a $3, 4, 6$ y 7. Con estos datos seleccionamos, para cada cantidad de atributos, la cantidad de muestras necesarias para construir el KDE de manera tal que la mediana del error relativo de la integral sea menor al 5 %, es decir, menor a 0.05. Los resultados de este experimento se muestran en la Tabla 6.2. Una celda pintada de rojo indica que, para esa cantidad de atributos, la mediana del error relativo de la integral fue mayor al 0.05 en todas las cantidades de muestras KDE utilizadas para el experimento; en ese caso se muestra el error relativo más bajo que se pudo obtener.

Cantidad de atributos	3	4	5	6	7
Cantidad de muestras KDE necesarias	10^4	10^4	10^5	10^7	10^7
Error relativo (mediana)	0.04	0.04	0.05	0.05	0.12

Tab. 6.2: Cantidad de muestras necesarias para construir el KDE para mantener el error relativo de la integral por debajo del 5 %.

Los resultados obtenidos confirman la necesidad de utilizar una mayor cantidad de muestras para construir el KDE a medida que aumenta la cantidad de atributos, si se desea

obtener una integral con un error relativo bajo. Además, se observa que este incremento en la cantidad de muestras necesarias para construir el KDE se vuelve cada vez mayor a medida que la cantidad de atributos aumenta. Por ejemplo, para pasar de 4 a 5 atributos se necesita multiplicar por 10 la cantidad de muestras necesarias para construir el KDE, mientras que para pasar de 5 a 6 atributos se necesita multiplicar por 100.

Ahora que entendemos las condiciones bajo las cuales se puede mantener el error relativo de la integral por debajo del 5 %, queda evaluar el impacto de un error relativo aproximado del 5 % en el cálculo del valor de verdad de fórmulas atómicas generadas aleatoriamente. Lo abordaremos a continuación.

RQ5: ¿Cuál es el impacto de un error relativo aproximado del 5 % en el cálculo del valor de verdad de fórmulas atómicas generadas aleatoriamente?

En última instancia, lo que nos gustaría saber es, dada una restricción de calidad $\varphi = \langle R,\ p_{min},\ p_{max} \rangle$ y una distribución de probabilidad $\mathbf{X}$, cuál es la probabilidad de que el veredicto calculado por

$$p_{min} \leq \textsc{ComputeIntegral}(f_{\mathbf{X}}, R) \leq p_{max}$$

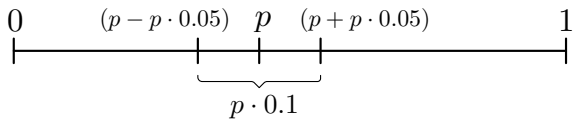
sea el mismo veredicto que el calculado por

$$p_{min} \leq \textsc{ComputeIntegral}(\widehat{f_{\mathbf{X}}}, R) \leq p_{max}$$

donde $\widehat{f_{\mathbf{X}}}$ es la estimación KDE de $f_{\mathbf{X}}$. Por simplicidad, a partir de ahora denotaremos los valores de las integrales como $p = \textsc{ComputeIntegral}(f_{\mathbf{X}}, R)$ y $\widehat{p} = \textsc{ComputeIntegral}(\widehat{f_{\mathbf{X}}}, R)$. Dicho esto, la probabilidad que queremos calcular es

$$\mathbb{P}\left(p_{min} \leq p \leq p_{max}\ \ sii\ \ p_{min} \leq \widehat{p} \leq p_{max}\right)$$

Entendamos primero la relación de las dos integrales p y $\widehat{p}$. En el siguiente diagrama vemos el margen de error de la integral de la estimación KDE, para el caso de un error relativo del 5 %:



La integral sobre la estimación KDE, que llamamos $\widehat{p}$, está contenida dentro de un intervalo de tamaño $0.1 \cdot p$ centrado en p. Si tanto p_{max} como p_{min}, las cotas de la restricción de calidad φ, están fuera de este intervalo, entonces el veredicto de la restricción de calidad no se ve afectado por el error relativo de la estimación KDE. En cambio, si alguna de las cotas está dentro del intervalo, entonces el veredicto de la restricción de calidad puede verse afectado por el error relativo de la estimación KDE. Esto nos permite descomponer la probabilidad que queremos calcular en dos partes, una condicionada a que p_{min} y p_{max} estén fuera del intervalo y otra condicionada a que al menos una de las cotas esté dentro del intervalo. Lo que nos lleva a la siguiente fórmula, en la que denotaremos Out al evento de que las cotas estén fuera del intervalo y $\overline{Out}$ a su complemento, es decir, el evento de que al menos una de las cotas esté dentro del intervalo:

$$\mathbb{P}\left(Out\right) + \mathbb{P}\left(\overline{Out}\right) \cdot \mathbb{P}\left(p_{min} \leq p \leq p_{max} \ sii \ p_{min} \leq \widehat{p} \leq p_{max} \mid \overline{Out}\right) \tag{6.1}$$

Veamos a continuación cómo calcular la probabilidad de que ambas cotas estén fuera del intervalo.

$$\begin{aligned}
\mathbb{P}\left(Out\right) &= \mathbb{P}\left(\{p_{min}, p_{max}\} \cap [p - 0.05 \cdot p,\ p + 0.05 \cdot p] = \emptyset\right) \\
&= \mathbb{P}\left(p_{max} \notin [p - 0.05 \cdot p,\ p + 0.05 \cdot p]\right) \cdot \\
&\quad \mathbb{P}\left(p_{min} \notin [p - 0.05 \cdot p,\ p + 0.05 \cdot p]\right) \\
&= (1 - p \cdot 0.1) \cdot (1 - p \cdot 0.1) \\
&= (1 - p \cdot 0.1)^2
\end{aligned}$$

Como vemos, la probabilidad de que las cotas estén fuera del intervalo depende del valor de la integral p, sin embargo, nos gustaría obtener resultados independientes del valor de la integral. Si p fuese 0, entonces la probabilidad de que las cotas estén fuera del intervalo sería 1 y, por lo tanto, la probabilidad de que el veredicto de la restricción de calidad sea el mismo que el veredicto de la restricción de calidad con la estimación KDE sería 1. Pero tomemos el "peor caso" posible, donde p es 1, es decir, donde la integral de la región sobre la distribución real es 1. En este caso, la probabilidad de que las cotas estén fuera del intervalo es $(1 - 0.1)^2 = 0.81$ y la Ecuación (6.1) se reduce a:

$$0.81 + 0.19 \cdot \mathbb{P}\left(p_{min} \leq p \leq p_{max} \ sii \ p_{min} \leq \widehat{p} \leq p_{max} \mid \overline{Out}\right) \tag{6.2}$$

Unícamente nos queda calcular la probabilidad de que el veredicto de la restricción de calidad sea el mismo que el veredicto de la restricción de calidad con la estimación KDE, condicionada a que al menos una de las cotas esté dentro del intervalo. Decidimos calcular esta probabilidad de manera experimental, aprovechando los casos de estudio generados para la configuración 3 del experimento realizado para la RQ2 (Figura 6.7c). En particular, tomaremos las estimaciones KDE que utilizan 10^5 muestras, ya que, como vimos en la RQ4, es la cantidad de muestras necesarias para mantener el error relativo de la integral por debajo del 5 % para el caso de 5 atributos. Para cada uno de los 100 casos de estudio, que son pares de distribución y región, generamos de manera aleatoria cotas p_{min} y p_{max} tales que $p_{min} \leq p_{max}$ y al menos una de las cotas está dentro del intervalo $[p - 0.05 \cdot p,\ p + 0.05 \cdot p]$. Luego, calculamos el porcentaje de casos donde los veredictos coinciden, es decir, donde vale que $p_{min} \leq p \leq p_{max}$ *sii* $p_{min} \leq \widehat{p} \leq p_{max}$. Debido a que el porcentaje de casos donde se preserva el veredicto podría depender de si el veredicto original es *Verdadero* o *Falso*, realizamos el mismo experimento para cada uno de estos dos casos por separado, obteniendo así dos porcentajes distintos. Los resultados de este experimento se muestran en la Tabla 6.3.

Esto completa el cálculo de la probabilidad de que se preserve el veredicto de la restricción de calidad y así completa la respuesta a la RQ5. Para el caso en que el veredicto original es *Verdadero*, obtenemos:

$$\begin{aligned}
\mathbb{P}\left(p_{min} \leq p \leq p_{max} \ sii \ p_{min} \leq \widehat{p} \leq p_{max}\right) &= 0.81 + 0.19 \cdot 0.5 \\
&= 0.9
\end{aligned}$$

Veredicto original	**Verdadero**	**Falso**
% de preservación del veredicto	0.5	0.73

Tab. 6.3: Porcentaje de casos donde el veredicto de la restricción de calidad es el mismo que el veredicto de la restricción de calidad con la estimación KDE, condicionada a que al menos una de las cotas esté dentro del intervalo.

mientras que para el caso en que el veredicto original es *Falso*, obtenemos:

$$\begin{aligned} \mathbb{P}\left(p_{min} \leq p \leq p_{max} \ sii \ p_{min} \leq \widehat{p} \leq p_{max}\right) &= 0.81 + 0.19 \cdot 0.73 \\ &= 0.95 \end{aligned}$$

Esto significa que, en promedio, el veredicto de la restricción de calidad se preserva en un 90 % de los casos cuando el veredicto original es *Verdadero* y en un 95 % de los casos cuando el veredicto original es *Falso*, para los datos experimentales que hemos utilizado.

6.5. Conclusiones

En este capítulo abordamos el problema de la selección de servicios con una interpretación probabilística de los atributos de calidad. Identificamos algunos elementos de interés que la literatura existente ha tratado de forma parcial y separada: 1) Los atributos de calidad son solo atributos medibles que no asumen ninguna interpretación o comportamiento específico prescrito, y 2) adoptan valores de una distribución de probabilidad continua. A eso agregamos la necesidad de adoptar una visión multivariada con respecto a la distribución de probabilidad de los valores, para permitir la representación del comportamiento probabilístico interdependiente entre los atributos. Esta necesidad se justificó recurriendo a la conocida relación existente entre la velocidad de procesamiento y el tiempo de respuesta de un servicio.

Para abordar el problema, propusimos un enfoque unificado en el que: 1) los atributos de calidad son solo variables reales sin interpretación adicional o comportamiento intrínseco asumido, 2) los contratos de provisión se definen como funciones de densidad describiendo vectores aleatorios, cuyas componentes representan los atributos de calidad, y 3) los requisitos de calidad son combinaciones proposicionales de restricciones probabilísticas. Además, proporcionamos: 1) un algoritmo para decidir si un requisito de calidad se satisface en un contrato de calidad, basado en un procedimiento eficiente de integración de Montecarlo y el uso de SAT-solvers modernos, y 2) los fundamentos matemáticos para un enfoque de estimación de contratos de provisión a partir de datos empíricos muestreados de ejecuciones, eliminando la carga de crear contratos de provisión de los proveedores de servicios. Usando PDFs estimadas, los contratos de provisión proporcionan la información que mejor se ajusta al comportamiento del servicio ya observado, evitando: 1) requerir al proveedor de servicios que proporcione una formalización, potencialmente compleja, de las propiedades estadísticas de los atributos cuantitativos (especialmente considerando el caso de aquellos cuyo comportamiento está fuera del control de los proveedores, como la latencia o el ancho de banda de la infraestructura de comunicación) y 2) establecer esquemas de penalización complejos que, debido a la complejidad de proporcionar una PDF realista, podrían ser injustos o excesivamente severos con un proveedor de servicios. Una idea similar ha sido estudiada, con resultados experimentales prometedores, en [155]. Allí,

los autores proponen el uso de métodos estadísticos para estimar la PDF univariante de cada atributo específico de forma aislada. Nuestra contribución puede considerarse como una generalización de su enfoque al caso multivariado, que creemos es más fiel a la realidad. Hemos presentamos una extensa evaluación experimental de la viabilidad de estimar contratos de provisión a partir de datos empíricos, que permitió identificar las condiciones en las que el enfoque propuesto es viable tanto en términos de precisión como de eficiencia.

Si bien hacia el final de este capítulo nos hemos enfocado en la estimación de contratos de provisión a partir de datos empíricos, una dirección de investigación complementaria es la de estudiar una metodología de diseño de contratos probabilísticos utilizando un lenguaje de programación probabilística [67]. Esto podría dar lugar a un enfoque híbrido en el que los proveedores de servicios específican en este lenguaje los aspectos probabilísticos de los que tienen control o conocimiento certero, mientras que los aspectos que escapan a su control se estiman a partir de datos empíricos.

Parte II:

Propiedades de calidad en modelos de sistemas distribuidos

7

Hacia una visión de los servicios con estructura de cómputo explícita

Durante las últimas dos décadas, el software ha cambiado de forma constante de aplicaciones monolíticas a componentes cooperativos distribuidos. Todas las contribuciones que hemos presentado en la Parte I se concentran puntualmente en el problema de selección de servicios (Definición 1) como un elemento clave para la construcción de aplicaciones distribuidas. Desde esa mirada, las contribuciones presentadas hasta el momento concibieron al modelo de un servicio como una caja negra de la que solo se conoce una especificación que caracteriza los valores que pueden tomar los atributos de calidad a lo largo de sus ejecuciones. Esto nos permitió investigar, con un nivel de profundidad considerable, el modelado formal de distintos aspectos de los atributos de calidad junto con la definición de algoritmos eficientes para determinar si un contrato de calidad satisface un requisito de calidad.

En lo que resta de esta tesis, en cambio, soltaremos el supuesto de que sólo se conoce a los servicios a partir de los valores que pueden tomar sus atributos de calidad y nos enfocaremos en la formalización y el análisis de propiedades de calidad en sistemas distribuidos para los cuales se conocemos la estructura de cómputo de cada uno de sus componentes. Para el modelado de sistemas distribuidos, adoptaremos los enfoques coreográficos; estos están ganando ímpetu en la industria (por ejemplo [21, 19, 61, 4]) que, cada vez más, concibe las aplicaciones como componentes interactuando sobre las infraestructuras de comunicación existentes. Entre otros modelos, las coreografías destacan por una separación nítida de responsabilidades: los modelos coreográficos abstraen los cómputos locales de las comunicaciones entre los participantes. De hecho, desde su introducción [35], las coreografías abogan por una separación entre una *vista global* y una *vista local* de la comunicación. La primera es una descripción de alto nivel de interacciones (distribuidas) mientras que la segunda es una descripción de cada componente de manera aislada. Esta es la característica distintiva de las coreografías que aprovechamos en lo que resta de esta tesis para razonar sobre propiedades cuantitativas de las aplicaciones. La idea básica es especificar los valores de los atributos de calidad de los estados locales de los componentos y luego *acumular* esos atributos a lo largo de las ejecuciones que involucran comunicaciones. A modo de ejemplo, supongamos que un componente A envía a otro componente B un mensaje m y que queremos considerar dos atributos de calidad: costo monetario (c) y consumo de memoria (mem). Este comportamiento puede ser abstraído con las siguientes

máquinas de estados finitos

$$\begin{array}{ll} \text{comportamiento de } \mathsf{A}: & \{c \leq 5, mem = 0\} \quad q_0 \xrightarrow{\mathsf{A\,B!m}} q_1 \quad \{5 \leq c \leq 10,\ mem < 3\} \\ \text{comportamiento de } \mathsf{B}: & \{c = 0, mem = 0\} \quad q_0{}' \xrightarrow{\mathsf{A\,B?m}} q_1' \quad \{10 \leq mem \leq 50,\ c = 0{,}01 \cdot mem\} \end{array} \tag{7.1}$$

donde A B!m y A B?m denotan respectivamente las acciones de comunicación de salida y entrada, y cada estado está decorado con una especificación que predica sobre los atributos de calidad en los estados locales de A y B. Por ejemplo, tanto A como B no asignan memoria en sus estados iniciales, el cómputo en A puede costar hasta cinco unidades monetarias antes de enviar un mensaje, y B no tiene costo ya que simplemente está esperando para recibir el mensaje, como refleja la especificación ($c = 0$). De manera similar, después de las acciones de comunicación, los cómputos locales de A y B se especifican con las fórmulas asociadas a los estados q_1 y q_1'.

La interacción entre A y B depende de la infraestructura de comunicación. Nostros adoptaremos modelos asincrónicos de envío de mensajes, lo que produce una ejecución como la siguiente

$$\pi : \quad s_0 \xrightarrow{\mathsf{A\,B!m}} s_1 \xrightarrow{\mathsf{A\,B?m}} s_2 \tag{7.2}$$

donde primero el mensaje es enviado por A y luego es eventualmente consumido por B.

En este contexto, estamos interesados en analizar propiedades de atributos cuantitativos. Específicamente, nos interesa analizar cómo las propiedades cuantitativas de los cómputos locales influyen en las propiedades globales de todo el sistema. Por lo tanto, concebimos las especificaciones de calidad de los estados locales como *contratos* que los cómputos locales de los componentes deben respetar. Para ser claros, las especificaciones en (7.1) indican el costo de los cómputos locales en A y B, y no predican sobre las propiedades de calidad de la infraestructura de comunicación.

Una vez que estas restricciones de calidad en los cómputos locales se establecen, las preguntas naturales a hacer son, por ejemplo, "¿se encuentra el consumo de memoria a lo largo de la ejecución π dentro de un rango dado?" o "¿el costo total está por debajo de un umbral dado?". Las respuestas a tales preguntas requieren verificar que la *acumulación* de los valores de los atributos de calidad a lo largo de la ejecución π implica las propiedades de calidad deseadas. Es interesante observar que la forma en que se acumulan esos valores depende de los atributos de calidad. Por ejemplo, la acumulación del consumo de memoria se puede calcular tomando el máximo, mientras que la acumulación del costo monetario se puede calcular como la suma. Trabajamos bajo la hipótesis de que los desarrolladores no tienen control sobre la infraestructura de comunicación. Más precisamente, los aspectos de calidad relacionados con cómo se realizan las comunicaciones no están bajo el control de los diseñadores de aplicaciones. En su lugar, los diseñadores tienen control sobre los cómputos locales, lo que sugiere que las restricciones de calidad están naturalmente asociadas a los estados de los componentes.

En el Capítulo 8 presentamos las contribuciones teóricas que sustentan este enfoque. Luego, en el Capítulo 9 presentamos una herramienta de verificación de modelos que implementa el procedimiento de decisión que presentamos en el Capítulo 8 junto con una evaluación experimental de su rendimiento y aplicabilidad.

8

Análisis de propiedades de calidad en modelos coreográficos de composición asincrónica

En este capítulo proponemos un marco para el diseño y análisis de propiedades de calidad de sistemas distribuidos, que se sustenta en las siguientes contribuciones técnicas:

Modelos para atributos de calidad La Sección 8.3 presenta una extensión directa de máquinas de comunicación de estados finitos (CFSMs [22]; revisado en Sección 8.2) para expresar aspectos calidad de los componentes. Básicamente, asignamos a cada estado de una CFSM una especificación calidad como mostramos en el capítulo anterior. Adoptamos las especificaciones basadas en cuerpos real-cerrados que presentamos en Capítulo 3 para representar abstractamente los valores de calidad. Sin embargo, el enfoque podría ser adaptado para considerar otro tipo de especificaciones, como por ejemplo las especificaciones probabilísticas que presentamos en Capítulo 6. Dejamos esta extensión para trabajos futuros.

Una lógica temporal dinámica para propiedades de calidad. La Sección 8.4 introduce una lógica, que llamamos $\mathcal{QL}$, para expresar propiedades de calidad. Tomando inspiración de la *Lógica Temporal Lineal Dinámica Proposicional* (DLTL) [75], $\mathcal{QL}$ indexa modalidades temporales con *coreografías globales* [143] (g-coreografías, Sección 8.2), un modelo de vistas globales de coreografías, para predicar sobre las propiedades de calidad del sistema completo. Esta es una característica distintiva de $\mathcal{QL}$ que comentamos en Sección 8.1.

Un procedimiento de semi-decisión para $\mathcal{QL}$. La Sección 8.5 demuestra que $\mathcal{QL}$ es semi-decidible al proporcionar un procedimiento de semi-decisión k-acotado y sustentarse en la decidibilidad de la teoría de los cuerpos real-cerrados [140] para verificar las restricciones de calidad en fórmulas atómicas. Un aspecto distintivo del procedimiento es que puede ser utilizado como un procedimiento de verificación acotada de modelos de las fórmulas de $\mathcal{QL}$.

En la Sección 8.7 damos conclusiones y proponemos algunas líneas de investigación futuras. Los resultados presentados en este capítulo fueron previamente publicados en [106] obteniendo el premio al mejor artículo de la conferencia.

8.1. Trabajo relacionado

Efectivamente, nuestro enfoque se sustenta sobre los *tipos comportamentales* (como [71, 5, 16, 10, 79]) que se abstraen de los detalles de bajo nivel. La relevancia del problema de analizar las propiedades de calidad en modelos coreográficos ya ha sido destacada por otros investigadores [86, 90]. La combinación de cuerpos real-cerrados y nuestros tipos comportamentales tiene como objetivo capturar aspectos esenciales para el análisis cuantitativo de las aplicaciones mientras se busca la generalidad. En esta línea, se ha propuesto una metodología de prueba de concepto basada en tipos comportamentales en [65] para sistemas cliente-servidor. Hasta donde tenemos conocimiento, el nuestro es el primer trabajo que combina tipos comportamentales con propiedades de calidad y ofrece un procedimiento de decisión para protocolos multiparte (de múltiples participantes).

Para capturar abstractamente los atributos de calidad en lugar de enfocarnos en atributos específicos adoptamos los cuerpos real-cerrados. Otros modelos abstractos de calidad como cuantales [134] o c-semirings [26, 105, 43] han sido propuestos. Optamos por los RCFs debido a su inherente decidibilidad, como fue discutido en el Capítulo 3, que es crucial para garantizar la (semi) decidibilidad de nuestra lógica. Además, los RCFs ofrecen ventajas prácticas ya que las fórmulas escritas en este lenguaje se pueden emplear fácilmente en los *SMT-solvers* modernos (satisfacibilidad módulo teorías) [13, Capítulo 33], como veremos en el Capítulo 9.

Las especificaciones convexas que presentamos en el Capítulo 4 para permitir el análisis eficiente de las propiedades de calidad solo considera politopos convexos; esta restricción no está presente en el lenguaje utilizado en este capítulo.

El uso de métodos coreográficos para el análisis no funcional ofrece otras ventajas. Por ejemplo, los contratos de calidad de los componentes se pueden derivar de especificaciones globales [86]. Estos contratos pueden utilizarse luego para predicción en tiempo de ejecución, para composición adaptativa, o para verificación de cumplimiento, de manera similar a lo que se hace en [90]. Este enfoque *top-down* (de arriba hacia abajo) se puede transferir a tipos comportamentales de manera similar a lo que se ha hecho en [16, 17] para propiedades cualitativas.

El enfoque propuesto en [148] utiliza CFSMs como un mecanismo de vinculación dinámica de servicios, en un contexto muy similar al problema de selección de servicios que abordamos en el Capítulo 2. Sin embargo solo considera los aspectos comunicacionales de los componentes de software. Tal enfoque podría extenderse para incluir también atributos de calidad aprovechando los resultados presentados en este capítulo y sería una forma de integrar los resultados de esta parte de la tesis con el enfoque de la Parte I.

Nuestra lógica $\mathcal{QL}$ toma inspiración de la *lógica temporal lineal dinámica* (*DLTL*) [75] que combina semántica de trazas (semejante a la *lógica temporal lineal* [126]) y expresiones regulares sobre un conjunto de acciones atómicas (semejantes a programas en *lógica dinámica proposicional* [130]). Una distinción fundamental es que, a diferencia de *DLTL*, $\mathcal{QL}$ no predica sobre el comportamiento de los programas secuenciales sino que describe propiedades de sistemas de envío de mensajes asíncronos. Esto requiere una modificación de la sintaxis de *DLTL*; de hecho, la sintaxis de $\mathcal{QL}$ es esencialmente la misma que *DLTL* salvo por los índices de las modalidades, que se convierten en coreografías de interacciones. Esta modificación tiene un impacto profundo en la semántica que requiere una redefinición completa (ver Sección 8.4 para más detalles). Otra diferencia clave es que, mientras que *DLTL* es proposicional, las fórmulas atómicas de $\mathcal{QL}$ son fórmulas de primer orden sobre

atributos de calidad. Como consecuencia, puede expresar propiedades temporales que restringen el valor de los atributos de calidad. Estos puntos de comparación con *DLTL* se aplican de la misma manera a una lógica similar llamada *lógica dinámica lineal* (LDL), introducida por primera vez en [146] y más tarde formalizada para trazas finitas en [39].

8.2. Preliminares

Esta sección examina resultados preliminares que sustentan nuestro trabajo. Primero describimos el protocolo utilizado como un ejemplo durante todo el capítulo y luego revisamos el modelo coreográfico que utilizamos para describir el comportamiento de los componentes.

Un ejemplo. A lo largo del capítulo utilizaremos una (variante simplificada) del protocolo POP [128]. Este protocolo permite a los clientes de correo acceder a un buzón remoto y recuperar correos electrónicos. En el protocolo POP, un cliente inicia la comunicación enviando un mensaje de tipo helo a un servidor POP (nótese que las especificaciones del protocolo ignoran el contenido de los mensajes).[1] El servidor responde con el número de mensajes no leídos en el buzón de correo usando un mensaje de tipo int. En este punto, el cliente puede detener el protocolo o leer uno de los correos electrónicos. Estas opciones se seleccionan enviando un mensaje de tipo quit o de tipo read respectivamente. En el primer caso, el servidor responde con un mensaje de tipo bye y el protocolo termina. En el segundo caso, el servidor envía al cliente el número de bytes del correo actual no leído en un mensaje de tipo size. A continuación, el cliente tiene nuevamente la opción de abandonar el protocolo (como antes) o de recibir el correo electrónico (seleccionado en el mensaje read) enviando al servidor un mensaje de tipo retr. En este segundo caso, el servidor envía el correo electrónico con un mensaje de tipo msg, el cliente responde con un mensaje de tipo ack y el proceso de lectura comienza nuevamente.

Un modelo coreográfico. Utilizamos *coreografías globales* [143] para especificar la vista global de sistemas de comunicación cuya vista local se representa como *máquinas de comunicación de estados finitos* [22]. A partir de ahora, fijamos un conjunto $\mathcal{P}$ de *participantes* y un conjunto $\mathcal{M}$ de (tipos de) *mensajes* tales que $\mathcal{P} \cap \mathcal{M} = \emptyset$. Comenzamos estudiando la definición de g-coreografías.

Definición 17 (Coreografías globales [143])**.** *Una* coreografía global sobre $\mathcal{P}$ y $\mathcal{M}$ *(*g-coreografía *para abreviar) es un término* G *que respeta la siguiente gramática:*

$$\mathsf{G} \ ::= \ \mathbf{0} \ \mid \ \mathsf{A}{\rightarrow}\mathsf{B}\colon \mathsf{m} \ \mid \ \mathsf{G};\mathsf{G}' \ \mid \ \mathsf{G} \mid \mathsf{G}' \ \mid \ \mathsf{G} + \mathsf{G}' \ \mid \ \mathsf{G}^*$$

donde $\mathsf{A}, \mathsf{B} \in \mathcal{P}$, $\mathsf{A} \neq \mathsf{B}$ *y* $\mathsf{m} \in \mathcal{M}$.

Intuitivamente, una g-coreografía especifica el protocolo de comunicación de los participantes. La g-coreografía básica es la vacía $\mathbf{0}$, que especifica que no deben ocurrir comunicaciones. Una *interacción* $\mathsf{A}{\rightarrow}\mathsf{B}\colon \mathsf{m}$ especifica que los participantes A y B (se espera que) intercambien un mensaje de tipo m; vale la pena volver a señalar que asumimos una comunicación asíncrona donde el emisor A no espera que B consuma m para continuar su

[1] Nuestro enfoque puede manejar protocolos multiparte; sin embargo, nuestros ejemplos son de dos participantes por simplicidad. Además, nos limitamos a los tipos de mensajes tal como se describen cuidadosamente en las especificaciones del protocolo [128].

ejecución. Además, las g-coreografías pueden ser compuestas secuencialmente ($\mathsf{G}; \mathsf{G}'$), en paralelo ($\mathsf{G} \mid \mathsf{G}'$), y en elecciones no deterministas ($\mathsf{G} + \mathsf{G}'$); asumimos que $\mathbf{0}$ es el elemento neutro de $_;_$, $_ \mid _$, y $_ + _$. Es preciso notar que, debido a la asincronía en la comunicación, en una composición secuencial $\mathsf{G}; \mathsf{G}'$, las salidas en G' pueden ocurrir antes de que G sea completamente ejecutado; por ejemplo, la ejecución distribuida de $\mathsf{A}{\to}\mathsf{B}\colon \mathsf{m}; \mathsf{C}{\to}\mathsf{B}\colon \mathsf{m}'$ permite que la salida de C ocurra antes de la de A. Finalmente, una g-coreografía puede ser iterada (G^*).

Ejemplo 4 (Una g-coreografía para POP)**.** *Nuestro ejemplo puede ser expresado como la g-coreografía* $\mathsf{G}_{\text{pop}} = \mathsf{C}{\to}\mathsf{S}\colon \mathsf{helo}; \mathsf{G}_{\text{start}} + \mathsf{G}_{\text{quit}}$ *donde*

$$\mathsf{G}_{\text{start}} = \mathsf{S}{\to}\mathsf{C}\colon \mathsf{int}; (\mathsf{G}_{\text{read}} + \mathsf{G}_{\text{read}}; \mathsf{G}_{\text{retr}})^*; \mathsf{G}_{\text{quit}} \qquad \mathsf{G}_{\text{read}} = \mathsf{C}{\to}\mathsf{S}\colon \mathsf{read}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{size}$$
$$\mathsf{G}_{\text{retr}} = \mathsf{C}{\to}\mathsf{S}\colon \mathsf{retr}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{msg}; \mathsf{C}{\to}\mathsf{S}\colon \mathsf{ack} \qquad \mathsf{G}_{\text{quit}} = \mathsf{C}{\to}\mathsf{S}\colon \mathsf{quit}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{bye}$$

($_;_$ tiene precedencia sobre $_ + _$). ◇

Los participantes de un sistema de comunicación interactúan a través de *canales* tomados del conjunto $\mathcal{C} = \{ (\mathsf{A}, \mathsf{B}) \in \mathcal{P} \times \mathcal{P} \mid \mathsf{A} \neq \mathsf{B} \}$. Un canal $(\mathsf{A}, \mathsf{B}) \in \mathcal{C}$ (escrito $\mathsf{A}\,\mathsf{B}$ para abreviar) permite que A envíe mensajes de manera asíncrona a B a través de un búfer FIFO ilimitado asociado a $\mathsf{A}\,\mathsf{B}$. El conjunto de *acciones de comunicación* es $\mathcal{L} = \mathcal{L}^! \cup \mathcal{L}^?$ donde $\mathcal{L}^! = \{\mathsf{A}\,\mathsf{B}!\mathsf{m} \mid \mathsf{A}\,\mathsf{B} \in \mathcal{C} \text{ y } \mathsf{m} \in \mathcal{M}\}$ y $\mathcal{L}^? = \{\mathsf{A}\,\mathsf{B}?\mathsf{m} \mid \mathsf{A}\,\mathsf{B} \in \mathcal{C} \text{ y } \mathsf{m} \in \mathcal{M}\}$ son respectivamente el conjunto de *acciones de salida* y *acciones de entrada*. El *lenguaje* $\mathcal{L}[\mathsf{G}]$ de una g-coreografía G es esencialmente el conjunto de todas las secuencias posibles en $\mathcal{L}$ compatibles con la relación causal inducida por G, es decir, aquellas que pueden ser generadas por una ejecución válida de G. Dado que $\mathcal{L}[\mathsf{G}]$ es cerrado por prefijos, escribimos $\hat{\mathcal{L}}[\mathsf{G}]$ para el conjunto de secuencias en $\mathcal{L}[\mathsf{G}]$ que no son prefijos estrictos de ninguna otra secuencia en $\mathcal{L}[\mathsf{G}]$. La definición técnica detallada de $\mathcal{L}[\mathsf{G}]$ no es necesaria para comprender las contribuciones de este capítulo pero se puede encontrar en el Sección B y de manera más detallada en [143].

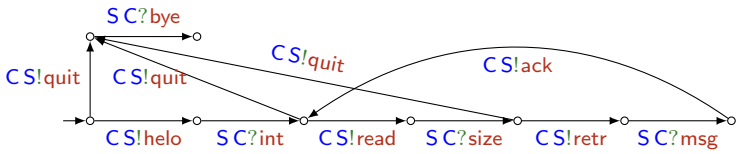
A continuación, introducimos las CFSM [22] para modelar la *vista local* del sistema distribuido que se quiere analizar.

Definición 18 (Sistemas de comunicación [22])**.** *Una* máquina de comunicación de estados finitos *(*CFSM*) es un sistema de transición finito* $M = (Q, q_0, \to)$ *donde*

- Q *es un conjunto finito de* estados *donde* $q_0 \in Q$ *es el estado* inicial*, y*
- $\to \subseteq Q \times \mathcal{L} \times Q$*; escribimos* $q \xrightarrow{\ell} q'$ *para referirnos a* $(q, \ell, q') \in \to$.

Para $\mathsf{A}\,\mathsf{B}!\mathsf{m} \in \mathcal{L}$*, denotaremos el sujeto de* $\mathsf{A}\,\mathsf{B}!\mathsf{m}$ *como* $\mathit{sbj}(\mathsf{A}\,\mathsf{B}!\mathsf{m}) = \mathsf{A}$*. Respectivamente, para* $\mathsf{A}\,\mathsf{B}?\mathsf{m} \in \mathcal{L}$*, denotaremos* $\mathit{sbj}(\mathsf{A}\,\mathsf{B}?\mathsf{m}) = \mathsf{B}$*. Dado* $\mathsf{A} \in \mathcal{P}$*, una máquina* M *es* A-local *si* $\mathit{sbj}(\ell) = \mathsf{A}$ *para cada* $q \xrightarrow{\ell} q'$*. Un* sistema de comunicación *es un map* $S = (M_{\mathsf{A}})_{\mathsf{A} \in \mathcal{P}}$ *que asigna una CFSM* A*-local* M_{A} *a cada* $\mathsf{A} \in \mathcal{P}$*.*

Ejemplo 5 (Sistema de comunicación para POP)**.** *La siguiente CFSM exhibe un comportamiento de un cliente POP compatible con el protocolo en Ejemplo 4, ya que sus ejecuciones producen un subconjunto del comportamiento del cliente especificado allí.*



Para modelar un servidor POP, al ser un protocolo de dos participantes, podemos usar una CFSM dual obtenida reemplazando las acciones de envío con las acciones de recepción correspondientes y viceversa. ◇

La comunicación asíncrona entre los participantes se formaliza mediante un sistema de transición etiquetado (LTS) que lleva registro del estado (local) de cada CFSM y el contenido de cada buffer (es decir, canal de comunicación) en el sistema. Una *configuración* de un sistema de comunicación S es un par $s = \langle \mathfrak{q} \ ; \ \mathfrak{b} \rangle$ donde $\mathfrak{q}$ y $\mathfrak{b}$ mapean respectivamente los participantes a estados y los canales a secuencias de mensajes; el estado $\mathfrak{q}(\mathsf{A})$ mantiene un registro del estado de la máquina M_{A} y el búfer $\mathfrak{b}(\mathsf{A}\,\mathsf{B})$ mantiene registro de los mensajes enviados desde A a B que aún no fueron consumidos. La configuración *inicial* s_0 es aquella en la que, para todos $\mathsf{A} \in \mathcal{P}$, $\mathfrak{q}(\mathsf{A})$ es el estado inicial del CFSM correspondiente y $\mathfrak{b}(\mathsf{A}\,\mathsf{B})$ es la secuencia vacía para todos $\mathsf{A}\,\mathsf{B} \in \mathcal{C}$.

Una configuración $s' = \langle \mathfrak{q}' \ ; \ \mathfrak{b}' \rangle$ es *alcanzable* desde otra configuración $s = \langle \mathfrak{q} \ ; \ \mathfrak{b} \rangle$ mediante el *disparo de una transición* ℓ, escrito $s \overset{\ell}{\Longrightarrow} s'$, si hay un mensaje $\mathsf{m} \in \mathcal{M}$ tal que se o bien (1) o bien (2) a continuación:

1. $\ell = \mathsf{A}\,\mathsf{B}!\mathsf{m}$ donde $\mathfrak{q}(\mathsf{A}) \xrightarrow{\ell}_{\mathsf{A}} q'$ y
   a. $\mathfrak{q}' = \mathfrak{q}[\mathsf{A} \mapsto q']$
   b. $\mathfrak{b}' = \mathfrak{b}[\mathsf{A}\,\mathsf{B} \mapsto \mathfrak{b}(\mathsf{A}\,\mathsf{B}).\mathsf{m}]$
2. $\ell = \mathsf{A}\,\mathsf{B}?\mathsf{m}$ donde $\mathfrak{q}(\mathsf{B}) \xrightarrow{\ell}_{\mathsf{B}} q'$ y
   a. $\mathfrak{q}' = \mathfrak{q}[\mathsf{B} \mapsto q']$
   b. $\mathfrak{b} = \mathfrak{b}'[\mathsf{A}\,\mathsf{B} \mapsto \mathsf{m}.\mathfrak{b}'(\mathsf{A}\,\mathsf{B})]$.

La condición (1) pone m en el canal AB, mientras que (2) obtiene m del canal AB. En ambos casos, cualquier máquina o buffer no involucrado en la transición se deja sin cambios en la nueva configuración s'.

Ejemplo 6 (Semántica de CFSMs)**.** *Para la ejecución π del sistema de comunicación en el ejemplo (7.1) del Capítulo 7 tenemos, para $i \in \{0, 1, 2\}$, $s_i = \langle \mathfrak{q}_i \ ; \ \mathfrak{b}_i \rangle$, donde $\mathfrak{q}_0 = \{\mathsf{A} \mapsto q_0, \mathsf{B} \mapsto q_0{}'\}$, $\mathfrak{b}_0 = \{\mathsf{A}\,\mathsf{B} \mapsto \epsilon, \mathsf{B}\,\mathsf{A} \mapsto \epsilon\}$, $\mathfrak{q}_1 = \{\mathsf{A} \mapsto q_1, \mathsf{B} \mapsto q_0{}'\}$, $\mathfrak{b}_1 = \{\mathsf{A}\,\mathsf{B} \mapsto \mathsf{m}, \mathsf{B}\,\mathsf{A} \mapsto \epsilon\}$, y $\mathfrak{q}_2 = \{\mathsf{A} \mapsto q_1, \mathsf{B} \mapsto q_1'\}$, $\mathfrak{b}_2 = \mathfrak{b}_0$.* ◇

Sea S un sistema de comunicación. Una secuencia $\pi = (s_i, \ell_i, s_{i+1})_{i \in I}$, donde I es un segmento inicial de números naturales[2], es una ejecución de S si $s_i \overset{\ell_i}{\Longrightarrow} s_{i+1}$ es una transición de S para todos $i \in I$. El conjunto de ejecuciones de S se denota como Δ_S^∞ y el conjunto de ejecuciones de longitud k se denota como Δ_S^k. Notar que Δ_S^∞ puede contener ejecuciones de longitud infinita, el conjunto de ejecuciones finitas de S es la unión de todos los Δ_S^k y se denotará como Δ_S. Dada una ejecución π, definimos $\mathcal{L}[\pi]$ como la secuencia de etiquetas $(\ell_i)_{i \in I}$. El *lenguaje* de S es el conjunto $\mathcal{L}[S] = \{\mathcal{L}[\pi] \mid \pi \in \Delta_S^\infty\}$. Finalmente, $prf : \Delta_S^\infty \to 2^{\Delta_S}$ mapea cada ejecución $\pi \in \Delta_S^\infty$ a su conjunto de prefijos finitos. Como es usual, para todos los $\pi \in \Delta_S^\infty$, el prefijo vacío ϵ pertenece a $prf(\pi)$. Por comodidad, ocasionalmente escribiremos $s_0 \overset{\ell_0}{\Longrightarrow} s_1 \ \ldots \ s_n \overset{\ell_n}{\Longrightarrow} s_{n+1}$ para denotar secuencias finitas.

8.3. Especificaciones de calidad de sistemas de comunicación

En esta sección extendemos CFSMs con especificaciones de calidad para expresar contratos de calidad de componentes en sistemas distribuidos basados en envío de mensajes. Básicamente, a cada estado de CFSMs se le asigna un contrato de calidad que especifica

[2] Es decir, $i - 1 \in I$ para cada $0 < i \in I$

el uso de recursos computacionales. Tomaremos la formalización de contratos de calidad de la Definición 4 que presentamos en el Capítulo 3 y la extenderemos con *operadores de acumulación*, denotados como $\oplus^{\mathsf{a}}$, para definir cómo cada atributo de calidad debe ser acumulado al analizar una ejecución del sistema. Estos operadores de acumulación serán esenciales para definir formalmente la noción de acumulación a lo largo de una ejecución, como se muestra más adelante en la Definición 22.

Definición 19. *Una* especificación de calidad $\langle \Sigma, \Gamma \rangle$ *es una presentación de teoría de primer orden extendiendo* $\langle \Sigma_{\mathrm{RCF}}, \Gamma_{\mathrm{RCF}} \rangle$ *de la siguiente manera:*

1. $\Sigma = \langle \{-1, 0, 1\} \cup \mathsf{Q}, \{+, \cdot\} \cup \{\oplus^{\mathsf{a}}\}_{\mathsf{a} \in \mathsf{Q}}, \{<\} \rangle$, *donde* Q *es un conjunto finito de símbolos de constante (distintos de* -1, 0 *y* 1*) que representan los* atributos cuantitativos *y, para cada* $\mathsf{a} \in \mathsf{Q}$, $\oplus^{\mathsf{a}}$ *es un operador binario asociativo, y*

2. $\Gamma = \Gamma_{\mathrm{RCF}} \cup \Gamma'$, *donde* Γ' *es un conjunto finito de fórmulas de primer órden formalizando condiciones específicas sobre los atributos de calidad en* Q.

Denotaremos la clase de especificaciones de calidad como $\mathcal{C}(\mathsf{Q})$.

Cabe señalar que los operadores de acumulación dependen fuertemente de la naturaleza de cada atributo específico; por ejemplo, los operadores de acumulación naturales para la memoria y el tiempo son el máximo y la suma respectivamente.

Ejemplo 7 (Especificación de calidad)**.** *Con referencia al Ejemplo 5, posibles atributos cuantitativos de interés en una implementación de POP son* $\mathsf{Q} = \{t, c, m\}$ *que representan el tiempo de CPU, el costo monetario y el uso de memoria, respectivamente. Entonces, una especificación de calidad que caracteriza bajos costos computacionales, donde ningún proceso interno consume una cantidad significativa de recursos, puede escribirse de acuerdo con la Definición 4 de la siguiente manera:*

$$\Sigma = \langle \{-1, 0, 1\} \cup \{t, c, m\}, \{+, \cdot\} \cup \{\oplus^t, \oplus^c, \oplus^m\}, \{<\} \rangle$$
$$\Gamma = \Gamma_{\mathrm{RCF}} \cup \Gamma'_{\mathrm{Low}}$$

donde $\oplus^t = +$, $\oplus^c = +$, $\oplus^m = max$, *y* $\Gamma'_{\mathrm{Low}} = \{t \leq ,01, c \leq ,01, m \leq ,01\}$ ◇

A partir de ahora, fijamos un conjunto de símbolos constantes Q, con sus respectivos operadores de acumulación, que omitimos del conjunto de *especificaciones de calidad*, al que se hará referencia simplemente como $\mathcal{C}$. Vale la pena señalar que, cuando Q está fijo, una especificación de calidad $\langle \Sigma, \Gamma_{\mathrm{RCF}} \cup \Gamma' \rangle$, está completamente determinada por Γ'. Por lo tanto, podemos referirnos sin ambigüedad a una especificación de calidad utilizando su conjunto de fórmulas Γ'. Así, la especificación de calidad en Ejemplo 7 es Γ'_{Low}.

Ejemplo 8 (Especificaciones de calidad para POP)**.** *Las siguientes especificaciones de calidad formalizan los costos asociados a diferentes actividades en el protocolo POP del Ejemplo 5.*

$$\Gamma'_{\mathrm{Chk}} = \{t \leq 5, c = 0{,}5, m = 0\}$$
$$\Gamma'_{\mathrm{Mem}} = \{1 \leq t \leq 6, c = 0, m \leq 64\}$$
$$\Gamma'_{\mathrm{DB}} = \{t \leq 3 \implies (\exists x)(0{,}5 \leq x \leq 1 \wedge c = t \cdot x), t > 3 \implies c = 10, m \leq 5\}$$

Básicamente, Γ'_{Chk} *formaliza los costos asociados a la actividad de verificar la integridad de un mensaje,* Γ'_{Mem} *a la actividad de un servidor reservando memoria para recibir un mensaje, y* Γ'_{DB} *a establecer que el costo monetario es fijo si la inserción toma más de tres unidades de tiempo y es una fracción del tiempo de ejecución en caso contrario.* ◇

Ahora extendemos los sistemas de comunicación (cf. Sección 8.2) con especificaciones de calidad.

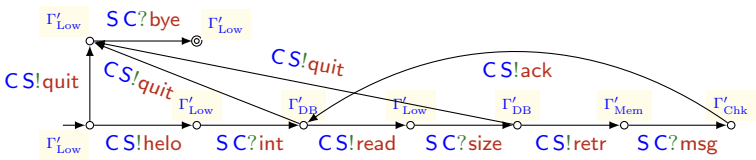
Definición 20 (CFSMs extendidas con calidad)**.** *Una* CFSM extendida con calidad*, que llamamos* qCFSM*, es una tupla* $M^{QoS} = \langle M, F, \mathsf{qos}\rangle$ *donde:*

- $M = \langle Q, q_0, \rightarrow\rangle$ *es una CFSM,*
- $F \subseteq Q$ *es un conjunto de* estados finales *de* M*, y*
- $\mathsf{qos} : Q \rightarrow \mathcal{C}$ *asigna a cada estado de* M *una especificacion de calidad.*

Un sistema de comunicación extendido con calidad S^{QoS} *es un map* $(M_{\mathsf{A}}^{QoS})_{\mathsf{A}\in\mathcal{P}}$ *que asigna una CFSM extendida con calidad* M_{A}^{QoS} A*-local para cada* $\mathsf{A} \in \mathcal{P}$*. Una configuración* $\langle \mathfrak{q} \ ; \ \mathfrak{b}\rangle$ *de* S^{QoS} *es una* configuración final *si* $\mathfrak{q}(\mathsf{A}) \in F_{\mathsf{A}}$ *para cada* $\mathsf{A} \in \mathcal{P}$*.*

La semántica de los sistemas de comunicación extendidos con especificaciones de calidad se define de la misma manera que la semántica de los sistemas de comunicación. Esto es una consecuencia de que las especificaciones de calidad no tienen ningún efecto en las comunicaciones. A continuación, presentamos un ejemplo de una CFSM extendida con calidad para el cliente POP en Ejemplo 5.

Ejemplo 9 (CFSM extendida con calidad)**.** *Una CFSM del cliente POP en Ejemplo 5 extendida con las especificaciones de calidad en Ejemplo 8 es la siguiente:*



donde el estado doblemente rodeado es el único estado final. A cada estado se le asigna una especificación de calidad según se indica en el Ejemplo 8 de acuerdo con la siguiente idea. A los estados donde el cliente realiza cómputo despreciables se les asigna la especificación de calidad Γ'_{Low}*. Los estados locales donde* C *realiza una inserción en la base de datos (justo después de recibir un mensaje* int *o* size*) y aquellos donde* C *accede a la memoria (justo antes de recibir un correo electrónico no leído) están especificados respectivamente por* Γ'_{DB} *y* Γ'_{Mem}*. Finalmente,* Γ'_{Chk} *especifica los estados donde* C *realiza una verificación de integridad (justo después de recibir un correo electrónico no leído).* ◇

Observe que la Definición 20 requiere que a cada estado de una CFSM se le asigne una especificación de calidad. Sin embargo, es posible que en muchos casos, la mayoría de los estados tenga la misma especificación calidad, como es el caso de Γ'_{Low} en Ejemplo 9; típicamente alcanza con identificar los costos de calidad específicos para unos pocos estados.

A continuación, presentamos una lógica temporal dinámica para describir propiedades de calidad de interés en sistemas de comunicación extendidos con calidad.

8.4. $\mathcal{QL}$: Una lógica dinámica para propiedades de calidad

Para describir las propiedades de calidad introducimos $\mathcal{QL}$, un lenguaje lógico similar a *DLTL*.

Definición 21 (Fórmula de calidad)**.** *La lógica $\mathcal{QL}$ consiste en el conjunto de fórmulas que se pueden obtener de la siguiente gramática:*

$$\Phi \ ::= \ \top \ \mid \ \psi \ \mid \ \neg\Phi \ \mid \ \Phi \vee \Phi \ \mid \ \Phi \, \mathcal{U}^{\mathsf{G}} \Phi$$

donde ψ es una fórmula en una presentación de teoría en $\mathcal{C}$, es decir, una fórmula de primer orden sobre atributos de calidad, y G es una g-coreografía sobre $\mathcal{P}$ y $\mathcal{M}$ (ver Definición 17).

Las fórmulas atómicas expresan condiciones sobre atributos cuantitativos. Similares a *DLTL*, las propiedades de las ejecuciones son fórmulas temporales lineales donde el operador *until* está indexado con una coreografía global G. En esencia, el papel de G es restringir el conjunto de ejecuciones a considerar para la satisfacción del *until*. Las coreografías globales son adecuadas para este propósito porque son una forma declarativa y compacta de caracterizar el comportamiento de los sistemas de envío de mensajes asíncronos. La modalidad de posibilidad $\langle \mathsf{G} \rangle \Phi$ se define como $\top \mathcal{U}^{\mathsf{G}} \Phi$ y la modalidad de necesidad $[\mathsf{G}]\Phi$ se define (dualmente) como $\neg\langle \mathsf{G} \rangle \neg\Phi$. Finalmente, los conectivos proposicionales $\wedge$ y $\implies$ se definen como es usual.

El siguiente ejemplo muestra cómo utilizar $\mathcal{QL}$ para expresar propiedades de calidad de ejecuciones específicas del sistema.

Ejemplo 10 (Propiedades de calidad del protocolo POP)**.** *Podemos usar las g-coreografías y la fórmula $\mathcal{QL}$ que mostramos a continuación para indicar que, a menos que el costo sea cero para los primeros tres correos electrónicos leídos, el costo está acotado por 10 veces el tiempo de CPU y el consumo de memoria está acotado por 5. Definimos*

$$\begin{aligned}
\Phi &\equiv [\mathsf{G}_3](c > 0) \implies [\mathsf{G}_3; \mathsf{G}_{\mathsf{msg}}{}^{*}]\big((c \leq t \cdot 10) \wedge (m \leq 5)\big) \qquad \textit{donde} \\
\mathsf{G}_3 &= \mathsf{C}{\to}\mathsf{S}\colon \mathsf{helo}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{int}; \mathsf{G}_{\mathsf{msg}}; \mathsf{G}_{\mathsf{msg}}; \mathsf{G}_{\mathsf{msg}} \qquad \textit{y} \\
\mathsf{G}_{\mathsf{msg}} &= \mathsf{C}{\to}\mathsf{S}\colon \mathsf{read}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{size}; \mathsf{C}{\to}\mathsf{S}\colon \mathsf{retr}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{msg}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{ack}
\end{aligned}$$

Intuitivamente, para que Φ valga en una ejecución, o bien las primeras tres recepciones de correo deben tener un costo cero, o bien en cada recepción posterior, el costo total y el consumo de memoria deben caer dentro de los límites especificados. ◇

Una fórmula $\mathcal{QL}$ (como Φ en Ejemplo 10) puede ser utilizada en análisis cuantitativos al *acumular* los valores de los atributos de calidad a lo largo de las ejecuciones del sistema. Más precisamente, dada una ejecución π, nuestra interpretación es que, para cada transición $s_i \xRightarrow{\ell_i} s_{i+1}$ de π, las obligaciones indicadas en las especificaciones de calidad de s_i se cumplen después de acumular todas las obligaciones de calidad a lo largo de π desde el estado s_0 hasta el estado s_i. Por lo tanto, una noción central en nuestro enfoque es la de *función de acumulación*. Dado S un sistema de comunicación extendido con especificaciones de calidad, una *función de acumulación* $\mathsf{agg}_S : \Delta_S \to \mathcal{C}$ produce una especificación de calidad que captura los valores acumulados de los atributos de calidad a lo largo de una ejecución $\pi \in \Delta_S$ mediante "la suma" de las especificaciones de calidad de los estados locales de los participantes.

Ejemplo 11 (Acumulación)**.** *A modo ilustrativo, retomemos el ejemplo de ejecución π en*

el capítulo anterior (7.1)*:*

$$
\begin{array}{lcccc}
 & \{c \leq 5, m = 0\} & & \{5 \leq c \leq 10,\ m < 3\} & \\
\mathsf{A}: & q_0 & \xrightarrow{\mathsf{A\,B!m}} & q_1 & \\
 & \{c = 0, m = 0\} & & \{10 \leq m \leq 50,\ c = 0{,}01 \cdot m\} & \\
\mathsf{B}: & q_0{}' & \xrightarrow{\mathsf{A\,B?m}} & q_1' & \\
 & & & & \\
\pi: & s_0 & \xrightarrow{\mathsf{A\,B!m}} & s_1 \xrightarrow{\mathsf{A\,B?m}} & s_2
\end{array}
$$

Denotaremos con c^q_{A} *(resp.* c^q_{B}*) el valor del atributo* c *en el estado* q *del participante* A *(resp.* B*) y de manera similar para el atributo* m*. Después de la ejecución de* π*, esperamos que* $\max\{m^{q_0}_{\mathsf{A}}, m^{q_1}_{\mathsf{A}}, m^{q_0'}_{\mathsf{B}}, m^{q_1'}_{\mathsf{B}}\}$ *y* $c^{q_0}_{\mathsf{A}} + c^{q_1}_{\mathsf{A}} + c^{q_0'}_{\mathsf{B}} + c^{q_1'}_{\mathsf{B}}$ *sean respectivamente el consumo de memoria y el costo monetario acumulado en* s_2*. Esto se reduce a acumular los atributos de calidad* c *y* m *utilizando las operaciones de máximo y suma, respectivamente.* ◇

Esencialmente, la acumulación en este caso se obtiene por (1) recolectar las especificaciones asociadas a cada uno de los estados locales de los participantes (renombrando los atributos como en Ejemplo 11); y (2) añadir una ecuación que combina todas las instancias de los atributos, utilizando el operador de acumulación correspondiente. La siguiente fórmula captura esta intuición y ejemplifica una manera en la que la función de acumulación podría ser definida.

Definición 22 (Función de acumulación)**.** *Sea* $S = (\langle M_{\mathsf{A}}, \mathit{qos}_{\mathsf{A}}\rangle)_{\mathsf{A}\in\mathcal{P}}$*, definimos la función de acumulación* $\mathit{agg}_S : \Delta_S \to \mathcal{C}$ *como* $\mathit{agg}_S(\pi) = f(\pi) \cup g(\pi)$ *donde*

$$
f(\pi) = \bigcup_{\substack{\mathsf{A}\in\mathcal{P}\\ 0\leq i\leq n}} \mathit{qos}_{\mathsf{A}}(\mathfrak{q}_i(\mathsf{A}))^{\mathfrak{q}_i(\mathsf{A})}_{\mathsf{A}} \qquad y \qquad g(\epsilon) = \left\{ \mathsf{a} = \left(\bigoplus_{\mathsf{A}\in\mathcal{P}}^{\mathsf{a}} \mathsf{a}^{\mathfrak{q}_0(\mathsf{A})}_{\mathsf{A}} \right) \;\middle|\; \mathsf{a} \in \mathsf{Q} \right\}
$$

$$
g(\pi) = \left\{ \mathsf{a} = \left(\bigoplus_{\substack{0\leq i<n\\ \mathsf{A}=\mathit{sbj}(\ell_i)}}^{\mathsf{a}} \mathsf{a}^{\mathfrak{q}_i(\mathsf{A})}_{\mathsf{A}} \right) \oplus^{\mathsf{a}} \left(\bigoplus_{\mathsf{A}\in\mathcal{P}}^{\mathsf{a}} \mathsf{a}^{\mathfrak{q}_n(\mathsf{A})}_{\mathsf{A}} \right) \;\middle|\; \mathsf{a} \in \mathsf{Q} \right\} \qquad si\ \pi \neq \epsilon
$$

donde

- $\pi = \langle \mathfrak{q}_0 \,;\, \mathfrak{b}_0\rangle \xrightarrow{\ell_0} \ldots \xrightarrow{\ell_{n-1}} \langle \mathfrak{q}_n \,;\, \mathfrak{b}_n\rangle \in \Delta_S$,
- *dado un conjunto* Π *de fórmulas* $\mathcal{QL}$*, definimos* $\Pi^q_{\mathsf{A}} = \{\psi^q_{\mathsf{A}} \mid \psi \in \Pi\}$*, y*
- ψ^q_{A} *se obtiene al reemplazar cada atributo de calidad* c *con el símbolo* c^q_{A} *en la fórmula atómica* ψ*.*

La intuición es que $f(\pi)$ *recolecta todas las especificaciones de los estados locales de los participantes a lo largo de la ejecución* π*, y* $g(\pi)$ *usa los operadores de acumulación para calcular los valores acumulados de los atributos calidad en la ejecución de* π*.*

Si aplicamos esta función de acumulación a la ejecución π *en Ejemplo 11, obtenemos lo siguiente:*

$$
\begin{aligned}
f(\pi) = {} & \{c^{q_0}_{\mathsf{A}} \leq 5, m^{q_0}_{\mathsf{A}} = 0\} \cup \{5 \leq c^{q_1}_{\mathsf{A}} \leq 10,\ m^{q_1}_{\mathsf{A}} < 3\} \\
& \cup \{c^{q_0'}_{\mathsf{B}} = 0, m^{q_0'}_{\mathsf{B}} = 0\} \cup \{10 \leq m^{q_1'}_{\mathsf{B}} \leq 50,\ c^{q_1'}_{\mathsf{B}} = 0{,}01 \cdot m^{q_1'}_{\mathsf{B}}\} \\
g(\pi) = {} & \left\{ m = \max\{m^{q_0}_{\mathsf{A}}, m^{q_1}_{\mathsf{A}}, m^{q_0'}_{\mathsf{B}}, m^{q_1'}_{\mathsf{B}}\}, c = c^{q_0}_{\mathsf{A}} + c^{q_1}_{\mathsf{A}} + c^{q_0'}_{\mathsf{B}} + c^{q_1'}_{\mathsf{B}} \right\}
\end{aligned}
$$

Es importante observar que, en nuestra concepción, una función de acumulación toma una ejecución del sistema como su entrada. Esta ejecución comprende inherentemente un orden secuencial específico de las acciones llevadas a cabo por los participantes. Los operadores de acumulación *max* y + utilizados en Ejemplo 11 siguen esta interpretación. Como se hará evidente en la Definición 23, esta interpretación es suficiente para los propósitos de este capítulo, ya que permite a $\mathcal{QL}$ especificar propiedades de calidad temporales sobre las ejecuciones del sistema. Sin embargo, podría ser de interés un tipo diferente de acumulación que incorpore el hecho de que hay estados locales que se ejecutan en paralelo. Esto puede requerir posiblemente explotar modelos verdaderamente concurrentes, como *pomsets* [129], pero lo dejamos para trabajos futuros.

A continuación, presentamos la semántica de nuestra lógica que se define en términos de sistemas de comunicación extendidos con especificaciones de calidad.

Definición 23 (Semántica de $\mathcal{QL}$)**.** *Dado S un sistema de comunicación extendido con especificaciones de calidad, un S-*modelo para una propiedad de calidad Φ *es un par $\langle \pi, \pi' \rangle$, donde $\pi \in \Delta_S^\infty$ contiene una configuración final (ver Definición 20) y $\pi' \in prf(\pi)$ hasta esa configuración, tal que $\langle \pi, \pi' \rangle \models_S \Phi$ donde la relación $\models_S$ se define de la siguiente manera:*

$$
\begin{array}{rcl}
\langle \pi, \pi' \rangle \models_S \psi & \textit{sii} & \mathsf{agg}_S(\pi') \models_{RCF} \psi \quad \textit{si } \psi \textit{ es una fórmula atómica} \\
\langle \pi, \pi' \rangle \models_S \neg\Phi & \textit{sii} & \langle \pi, \pi' \rangle \models_S \Phi \textit{ no se cumple} \\
\langle \pi, \pi' \rangle \models_S \Phi_1 \vee \Phi_2 & \textit{sii} & \langle \pi, \pi' \rangle \models_S \Phi_1 \textit{ o } \langle \pi, \pi' \rangle \models_S \Phi_2 \\
\langle \pi, \pi' \rangle \models_S \Phi_1 \, \mathcal{U}^{\mathsf{G}} \, \Phi_2 & \textit{sii} & \textit{existe } \pi'' \textit{tal que } \mathcal{L}[\pi''] \in \hat{\mathcal{L}}[\mathsf{G}], \\
 & & \pi'\pi'' \in prf(\pi) \textit{ hasta una configuración final en } \pi, \\
 & & \langle \pi, \pi'\pi'' \rangle \models_S \Phi_2 \textit{ y, para todos } \pi''' \in prf(\pi''), \\
 & & \textit{si } \pi''' \neq \pi'' \textit{ entonces } \langle \pi, \pi'\pi''' \rangle \models_S \Phi_1.
\end{array}
$$

Una propiedad de calidad Φ es satisfacible *en S si existe una ejecución $\pi \in \Delta_S^\infty$ tal que $\langle \pi, \epsilon \rangle \models_S \Phi$, y es* válida *(denotado como $\models_S \Phi$) si, para todas las ejecuciones $\pi \in \Delta_S^\infty$ que contienen una configuración final, se cumple $\langle \pi, \epsilon \rangle \models_S \Phi$.*

Las fórmulas atómicas se manejan obteniendo la especificación de calidad acumulada de la ejecución π' y utilizando la relación de satisfacibildad de RCFs. La definición del operador *until* es similar al operador estándar: Φ_2 debe cumplirse en algún momento en el futuro, es decir, $\pi'\pi''$ y Φ_1 deben cumplirse hasta ese punto; la diferencia clave es que la satisfacción de Φ_2 está restringida a las ejecuciones donde la extensión π'' está en el lenguaje caracterizado por la g-coreografía G.

8.5. Un procedimiento de semidecisión para $\mathcal{QL}$

Ahora establecemos la semidecidibilidad de $\mathcal{QL}$ presentando un procedimiento de semidecisión k-acotado que se basa en tres algoritmos: `qSat`, `qModels`, y `qUntil`. El algoritmo `qSat` es el algoritmo principal del procedimiento y determina si una fórmula dada es satisfacible en un sistema dado. Se apoya en `qModels` para verificar si hay alguna ejecución que satisfaga la fórmula y este, a su vez, utiliza `qUntil` para manejar el operador $\mathcal{U}$. Comencemos observando `qSat`, definido como:

```
1  QSAT(Φ, S, k):
2      i = 0
3      while i ≤ k do
4          foreach π ∈ Δ^i_S do
5              if la última configuración de π es final y QMODELS(Φ, S, π, ε) then
6                  return verdadero
7          i = i + 1
8      return falso
```

En esencia, QSAT enumera todas las ejecuciones de S hasta una longitud límite dada k y verifica si alguna de ellas satisface Φ (vale recordar que Δ^i_S es el conjunto de todas las ejecuciones de S con longitud i). Centremos ahora nuestra atención en el algoritmo QMODELS:

```
1  QMODELS(Φ, S, π, π'):
2      switch Φ do
3          case ⊤ do
4              return verdadero
5          case ψ do
6              return si agg_S(π') ⊨_RCF ψ
7          case ¬Φ_1 do
8              return no QMODELS(Φ_1, S, π, π')
9          case Φ_1 ∨ Φ_2 do
10             return QMODELS(Φ_1, S, π, π') o QMODELS(Φ_2, S, π, π')
11         case Φ_1 𝒰^G Φ_2 do
12             return QUNTIL(Φ_1, G, Φ_2, S, π, π', ε)
```

Siguiendo Definición 23, QMODELS inspecciona recursivamente la fórmula $\mathcal{QL}$. Invoca QUNTIL para manejar el operador $\mathcal{U}$ y al procedimiento de decisión de la teoría de los cuerpos real-cerrados para verificar las fórmulas atómicas. Ahora veamos el algoritmo QUNTIL:

```
1  QUNTIL(Φ_1, G, Φ_2, S, π, π', π''):
2      if ℒ[π''] ∈ ℒ̂[G] y QMODELS(Φ_2, S, π, π'π'') then
3          return verdadero
4      else if no QMODELS(Φ_1, S, π, π'π'') then
5          return falso
6      else
7          Sea ⇒^ℓ q la transición tal que π'π'' ⇒^ℓ q ∈ prf(π)
           (toma la primera transición en π si π'π'' = ε,
           y no está definida si π'π'' = π)
8          if π'π'' = π o ℒ[π'' ⇒^ℓ q] ∉ ℒ[G] then
9              return falso
10         else
11             return QUNTIL(Φ_1, G, Φ_2, S, π, π', π'' ⇒^ℓ q)
```

Este procedimiento se encarga de buscar un testigo para el existencial de la semántica del operador $\mathcal{U}$ comenzando en el prefijo actual π' y siguiendo las transiciones de π. De acuerdo con la Definición 23, QUNTIL busca un testigo de la parte existencial de $\mathcal{U}$. Toma como parámetros la ejecución completa π, el prefijo π' en el que se está evaluando el operador $\mathcal{U}$, y la extensión actual π'' que se usa para buscar el testigo. Si π'' es suficiente para alcanzar

un veredicto, el algoritmo devuelve verdadero o falso en consecuencia (Líneas 3,5). De lo contrario, intenta extender π'' tomando la siguiente transición de π (Línea 7). Si tal extensión existe y es un candidato para estar en el lenguaje de G, el algoritmo se llama a sí mismo recursivamente con el prefijo extendido (Línea 11).

A continuación, probamos que `qSat` es correcto y k-acotado completo. Para eso, a partir de ahora fijaremos un sistema de comunicación S extendido con especificaciones de calidad. También asumiremos que los operadores de acumulación definidos para los atributos de calidad se pueden expresar con operaciones elementales de la teoría de los cuerpos real-cerrados, lo que garantiza la decidibilidad de la relación de satisfacción para fórmulas atómicas de $\mathcal{QL}$, como mostramos en la Proposición 1 del Capítulo 3

Teorema 1 (`qSat` es correcto y k-acotado completo)**.** *Dada una fórmula de calidad $\Phi \in \mathcal{QL}$ y una cota k, $\texttt{qSat}(\Phi, S, k)$ devuelve **verdadero** si y solo si existe $\pi \in \Delta_S^i$ tal que $\langle \pi, \epsilon \rangle \models_S \Phi$, para algún $i \leq k$.*

La corrección y completitud de `qSat` se deduce inmediatamente de la corrección y completitud de `qModels` (establecida en Lema 1 a continuación) que, a su vez, depende de la corrección y completitud de `qUntil` (cf. Lemas 2 y 3, respectivamente). Esto garantiza que la llamada a `qModels` en Línea 5 de `qSat` devuelva verdadero si y solo si la ejecución π satisface Φ. Es importante tener en cuenta que sólo se garantiza que `qSat` sea completo dentro de la cota k.

Lema 1 (`qModels` es correcto y completo)**.** *Dada una fórmula de calidad $\Phi \in \mathcal{QL}$ y las ejecuciones $\pi, \pi' \in \Delta_S$, donde $\pi' \in prf(\pi)$, $\texttt{qModels}(\Phi, S, \pi, \pi')$ devuelve **verdadero** si y solo si $\langle \pi, \pi' \rangle \models_S \Phi$.*

Demostración. Por inducción estructural en Φ. Si Φ es $\top$, el resultado se sigue trivialmente. Si Φ es una fórmula atómica, el algoritmo calcula la acumulación sobre la ejecución π' (Línea 6) e invoca el procedimiento de decisión de RCFs para comprobar si $\mathsf{agg}_S(\pi')$ satisface Φ en la teoría de los cuerpos real-cerrados. Si Φ es $\Phi_1 \vee \Phi_2$, el algoritmo realiza dos llamadas recursivas y devuelve verdadero si y solo si $\langle \pi, \pi' \rangle \models_S \Phi_1$ o $\langle \pi, \pi' \rangle \models_S \Phi_2$. Si Φ es $\Phi_1 \, \mathcal{U}^{\mathsf{G}} \, \Phi_2$, el algoritmo devuelve verdadero si $\texttt{qUntil}(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \epsilon)$ devuelve verdadero. Por Lemas 2 y 3 esto es equivalente a $\langle \pi, \pi' \rangle \models_S \Phi_1 \, \mathcal{U}^{\mathsf{G}} \, \Phi_2$. □

Ahora probamos la corrección y completitud de `qUntil`.

Lema 2 (`qUntil` es correcto)**.** *Dada una fórmula de calidad $\Phi_1, \Phi_2 \in \mathcal{QL}$, una g-coreografía G, y las ejecuciones π, π', π'' tales que*

1. *$\pi'\pi'' \in prf(\pi)$ y $\pi \in \Delta_S$, y*
2. *para todo $\pi''' \in prf(\pi'')$, si $\pi''' \neq \pi''$ entonces $\langle \pi, \pi'\pi''' \rangle \models_S \Phi_1$*

*si $\texttt{qUntil}(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \pi'')$ devuelve **verdadero** entonces $\langle \pi, \pi' \rangle \models_S \Phi_1 \, \mathcal{U}^{\mathsf{G}} \, \Phi_2$.*

Demostración. La llamada a $\texttt{qUntil}(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \pi'')$ o bien llega a la Línea 3 o bien alcanza la Línea 11 y la llamada recursiva devuelve verdadero. En el primer caso, sabemos que $\mathcal{L}[\pi''] \in \hat{\mathcal{L}}[\mathsf{G}]$ y que $\texttt{qModels}(\Phi_2, S, \pi, \pi'\pi'')$ devolvió verdadero. Por Lema 1 se deduce que $\langle \pi, \pi'\pi'' \rangle \models_S \Phi_2$. Junto con las hipótesis 1 y 2, se cumplen las condiciones de la semántica de la fórmula $\Phi_1 \, \mathcal{U}^{\mathsf{G}} \, \Phi_2$ (ver Definición 23). En el caso de alcanzar la Línea 11, sabemos que la llamada recursiva $\texttt{qUntil}(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \pi'' \overset{\ell}{\Rightarrow} q)$ devolvió

verdadero. Las condiciones 1 y 2 aplicadas a la entrada de la llamada recursiva se cumplen debido a la forma en que se eligió la transición $\overset{\ell}{\Rightarrow} q$ y al hecho de que la condición en la Línea 4 devolvió falso. Por lo tanto, podemos tomar que la salida de la llamada recursiva satisface el Lema 2 como hipótesis inductiva y concluir que $\langle \pi, \pi' \rangle \models_S \Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$. □

Lema 3 (QUNTIL es completo)**.** *Dadas fórmulas de calidad $\Phi_1, \Phi_2 \in \mathcal{QL}$, una g-coreografía G, y las ejecuciones π, π', π'' de manera que*

1. *$\pi'\pi'' \in prf(\pi)$ y $\pi \in \Delta_S$, y*
2. *para todo $\pi''' \in prf(\pi'')$, si $\pi''' \neq \pi''$ entonces, o bien $\mathcal{L}[\pi'''] \notin \hat{\mathcal{L}}[\mathsf{G}]$ o bien $\langle \pi, \pi'\pi''' \rangle \not\models_S \Phi_2$*

si QUNTIL$(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \pi'')$ *devuelve* ***falso*** *entonces $\langle \pi, \pi' \rangle \not\models_S \Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$*

Demostración. La llamada a QUNTIL$(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \pi'')$ alcanza la Línea 5, la Línea 9 o la Línea 11 y la llamada recursiva devuelve falso. En todos los casos, por la condición 2 sabemos que ningún prefijo de π'' podría ser testigo del existencial en la semántica de $\Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$ (ver Definición 23) porque tendría que estar en $\hat{\mathcal{L}}[\mathsf{G}]$ y también satisfacer Φ_2. La propia ejecución π'' no puede ser el testigo por las mismas razones, debido al hecho de que la condición en la Línea 2 no se cumplió. Lo que significa que o bien $\mathcal{L}[\pi''] \notin \hat{\mathcal{L}}[\mathsf{G}]$ o bien QMODELS$(\Phi_2, S, \pi, \pi'\pi'')$ devolvió falso; por lo tanto, usando el Lema 1, sabemos que o bien $\mathcal{L}[\pi''] \notin \hat{\mathcal{L}}[\mathsf{G}]$ o bien $\langle \pi, \pi'\pi'' \rangle \not\models_S \Phi_2$. La única posibilidad restante es que el testigo sea un $\pi^\star$ tal que $\pi'' \in prf(\pi^\star)$ y $\pi^\star \neq \pi''$. En el caso de que se haya alcanzado la Línea 5, sabemos que QMODELS$(\Phi_1, S, \pi, \pi'\pi'')$ devolvió falso. Por el Lema 1 se deduce que $\langle \pi, \pi'\pi'' \rangle \not\models_S \Phi_1$. Por lo tanto, la extensión $\pi^\star$ no pudo ser un testigo para la existencial en la semántica de $\Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$. En el caso de que se haya alcanzado la Línea 9, la candidata a extensión $\pi^\star$ o bien no existe o bien no está en $\hat{\mathcal{L}}[\mathsf{G}]$. En el caso de que se haya alcanzado la Línea 11, sabemos que QUNTIL$(\Phi_1, \mathsf{G}, \Phi_2, S, \pi, \pi', \pi'' \overset{\ell}{\Rightarrow} q)$ devolvió falso. Notemos que las condiciones 1 y 2 aplicadas a la entrada de las llamadas recursivas se cumplen por la forma en que se eligió la transición $\overset{\ell}{\Rightarrow} q$ y que la condición en la Línea 2 no se cumplió. Por lo tanto, podemos tomar que la salida de las llamadas recursivas satisfacen el Lema 3 como hipótesis inductiva y concluir que $\langle \pi, \pi' \rangle \not\models \Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$. □

Notemos que la prueba para Lema 1 usa Lema 3 y Lema 2, y que las pruebas para Lema 3 y Lema 2 usan Lema 1. Esto no socava la integridad de las pruebas porque los lemas son siempre (inductivamente) aplicados en fórmulas $\mathcal{QL}$ más pequeñas. Ahora que la completitud y corrección de QMODELS y QUNTIL está establecida, queda por mostrar su terminación. La terminación sigue del hecho de que tanto el número de operadores lógicos en Φ como el número de transiciones en π son finitos. El primero garantiza que QMODELS eventualmente llega a un caso base y el segundo garantiza que QUNTIL eventualmente llega a un caso base. Finalmente, el caso base en QMODELS, que calcula la acumulación y verifica la relación de satisfacción en la teoría de los cuerpos real-cerrados, termina debido a la decidibilidad de los RCFs [140].

8.5.1. Un enfoque de verificación acotada de modelos para $\mathcal{QL}$

Los resultados anteriores permiten un enfoque sencillo de verificación acotada de modelos para $\mathcal{QL}$. Al igual que con otros procedimientos de verificación de modelos para un

lenguaje que admite negación, QSat se puede utilizar para comprobar la validez de una fórmula $\mathcal{QL}$ en un sistema S comprobando la satisfactibilidad de la fórmula negada. Esto constituye un procedimiento de búsqueda de contraejemplos para $\mathcal{QL}$. La única advertencia a tener en cuenta es que QSat es un procedimiento de semidecisión acotado por k en lugar de un procedimiento de decisión. Sin embargo, podemos estudiar la restricción del lenguaje en la que las fórmulas no contienen el operador $*$ en sus coreografías, que denominaremos $\mathcal{QL}^-$. En este caso sí podemos encontrar modelos finitos de fórmulas satisfacibles de $\mathcal{QL}^-$ (cf. Teorema 2).

Teorema 2 (Propiedad de modelo finito de $\mathcal{QL}^-$)**.** *Dados una fórmula $\Phi \in \mathcal{QL}^-$, y las ejecuciones $\pi \in \Delta_S^\infty$, $\pi' \in \Delta_S$ tal que $\pi' \in prf(\pi)$. Si $\langle \pi, \pi' \rangle \models_S \Phi$ entonces existe una ejecución finita $\pi^- \in \Delta_S$ tal que $\pi^- \in prf(\pi)$ y $\langle \pi^-, \pi' \rangle \models_S \Phi$.*

Demostración. Por inducción estructural en Φ. Si Φ es $\top$ o una fórmula atómica, tomamos $\pi^- = \pi'$. Si Φ es $\Phi_1 \vee \Phi_2$, tenemos que o bien $\langle \pi, \pi' \rangle \models_S \Phi_1$ o bien $\langle \pi, \pi' \rangle \models_S \Phi_2$. Por hipótesis inductiva, o bien $\langle \pi_1^-, \pi' \rangle \models_S \Phi_1$ o $\langle \pi_2^-, \pi' \rangle \models_S \Phi_2$ para algunos $\pi_1^-, \pi_2^- \in prf(\pi)$ finitos. Por lo tanto, sabemos que $\langle \pi_1^-, \pi' \rangle \models_S \Phi_1 \vee \Phi_2$ o $\langle \pi_2^-, \pi' \rangle \models_S \Phi_1 \vee \Phi_2$.

Si Φ es $\Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$, por la Definición 23 tenemos que existe π'' tal que $\mathcal{L}[\pi''] \in \mathcal{L}[\mathsf{G}]$, $\pi'\pi'' \in prf(\pi)$ hasta una configuración final con $\langle \pi, \pi'\pi'' \rangle \models_S \Phi_2$, y para todos $\pi''' \in prf(\pi'')$, si $\pi''' \neq \pi''$ entonces $\langle \pi, \pi'\pi''' \rangle \models_S \Phi_1$. Si aplicamos la hipótesis inductiva a Φ_1 y Φ_2, tenemos que existe π'' tal que $\mathcal{L}[\pi''] \in \mathcal{L}[\mathsf{G}]$, $\pi'\pi'' \in prf(\pi)$ hasta una configuración final con $\langle \pi_2^-, \pi'\pi'' \rangle \models_S \Phi_2$ para algún $\pi_2^- \in \Delta_S$ tal que $\pi_2^- \in prf(\pi)$, y para todos $\pi''' \in prf(\pi'')$, si $\pi''' \neq \pi''$ entonces $\langle \pi_1^-, \pi'\pi''' \rangle \models_S \Phi_1$ para algún $\pi_1^- \in \Delta_S$ tal que $\pi_1^- \in prf(\pi)$. Observemos que dado que G no contiene $*$, la ejecución π'' en el lenguaje $\mathcal{L}[\mathsf{G}]$ es necesariamente finita y también lo es el número de ejecuciones cuantificadas π'''. Por lo tanto, el número de ejecuciones π_1^- involucradas en la afirmación anterior es finito, y hay un máximo entre sus longitudes, así que podemos tomar π^- como el más largo entre π_2^- y las ejecuciones π_1^-. Dado que π_2^- y todas las π_1^- son prefijos de π, entonces también serán prefijos de π^-, y por lo tanto tenemos las condiciones para concluir $\langle \pi^-, \pi' \rangle \models_S \Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$.

Si el operador más externo en Φ es $\neg$, necesitamos considerar todos los posibles casos para la subfórmula inmediata de Φ. Si Φ es $\neg\psi$ con ψ una fórmula atómica, tenemos que $\langle \pi, \pi' \rangle \not\models_S \psi$. Tomamos $\pi^- \in prf(\pi)$ una extensión de π' cuya última configuración sea final. Si Φ es $\neg(\Phi_1 \vee \Phi_2)$, tenemos que $\langle \pi, \pi' \rangle \not\models_S \Phi_1 \vee \Phi_2$. Se deduce que $\langle \pi, \pi' \rangle \not\models_S \Phi_1$ y $\langle \pi, \pi' \rangle \not\models_S \Phi_2$. Por hipótesis inductiva, existe $\pi_1^- \in \Delta_S$ tal que $\pi_1^- \in prf(\pi)$ y $\langle \pi_1^-, \pi' \rangle \not\models_S \Phi_1$ y existe $\pi_2^- \in \Delta_S$ tal que $\pi_2^- \in prf(\pi)$ y $\langle \pi_2^-, \pi' \rangle \not\models_S \Phi_2$. Es suficiente tomar π^- como el más largo entre π_1^- y π_2^-. Si Φ es $\neg(\Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2)$, tenemos que $\langle \pi, \pi' \rangle \not\models_S \Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$. Por lo tanto, para todos π'' tal que $\pi'\pi'' \in prf(\pi)$ hasta una configuración final de π, si $\mathcal{L}[\pi''] \in \mathcal{L}[\mathsf{G}]$, y $\langle \pi, \pi'\pi'' \rangle \models_S \Phi_2$, entonces existe $\pi''' \in prf(\pi'')$, con $\pi''' \neq \pi''$ tal que $\langle \pi, \pi'\pi''' \rangle \models_S \neg\Phi_1$. Si aplicamos la hipótesis inductiva a Φ_2 y $\neg\Phi_1$, tenemos que para todos π'' tal que $\pi'\pi'' \in prf(\pi)$ hasta una configuración final de π, si $\mathcal{L}[\pi''] \in \mathcal{L}[\mathsf{G}]$, y $\langle \pi_2^-, \pi'\pi'' \rangle \models_S \Phi_2$ para algún $\pi_2^- \in \Delta_S$ con $\pi_2^- \in prf(\pi)$, entonces existe $\pi''' \in prf(\pi'')$, con $\pi''' \neq \pi''$ tal que $\langle \pi_1^-, \pi'\pi''' \rangle \models_S \neg\Phi_1$ para algún $\pi_1^- \in \Delta_S$ con $\pi_1^- \in prf(\pi)$. Observemos que dado que G no contiene $*$, cualquier ejecución en el lenguaje $\mathcal{L}[\mathsf{G}]$ es necesariamente finita. Por lo tanto, hay un máximo entre las longitudes de las ejecuciones π_2^-, y podemos tomar π^- como el más largo entre π_2^- y π_1^-. □

La prueba del Teorema 2 sugiere que la longitud de la ejecución π^- constituye una cota adecuada para QSat; lo cual convertiría a QSat en un procedimiento de decisión para $\mathcal{QL}^-$ si uno pudiera calcular dicha cota. Buscando contraejemplos de una fórmula

arbitraria $\Phi \in \mathcal{QL}$ hasta un número limitado de *unfoldings* de $*$ es equivalente a buscar contraejemplos en una fórmula $\hat{\Phi}$ en $\mathcal{QL}^-$ donde cada $*$ ha sido reemplazado por un número finito de *unfoldings*. Lo que significa que el procedimiento acotado para buscar modelos de fórmulas en $\mathcal{QL}^-$ podría usarse para buscar contraejemplos de fórmulas en $\mathcal{QL}$. Observe que `QSAT` se puede extender fácilmente para devolver la ejecución que satisface la fórmula, si es que hay una. Dicha ejecución se puede utilizar para identificar la fuente de violaciones de la fórmula de calidad cuando `QSAT` se utiliza como un procedimiento de búsqueda de contraejemplos.

8.6. Consideraciones sobre modelos de composición sincrónica

Si bien nosotros hemos decidido presentar nuestro enfoque basándonos en sistemas de envío de mensajes asincrónicos, no hay ningún obstáculo para definir un enfoque análogo en modelos de sistemas distribuidos basados en composición sincrónica. A continuación identificamos los cambios que serían necesarios para adaptar nuestro enfoque a este tipo de sistemas.

Modelo de sistema distribuido extendido con especificaciones de calidad. En la Definición 20 hemos extendido las CFSMs con especificaciones de calidad. Se debería dar una definición análoga utilizando un modelo de componente de composición sincrónica, como por ejemplo, los autómatas finitos. Notar que esto es casi directo debido a que la extensión de las CFSMs con especificaciones de calidad es independiente de la semántica de ejecución de las mismas.

Índice del operador $\mathcal{U}$ en la lógica $\mathcal{QL}$. En la Definición 21 hemos definido la lógica $\mathcal{QL}$ utilizando g-coreografías indexando el operador $\mathcal{U}$. Sin embargo, el único propósito de este índice es caracterizar las ejecuciones sobre las que se debe evaluar la semántica de este operador y, por lo tanto, se puede pensar de manera abstracta como un conjunto de ejecuciones del sistema. Esto independiza la semántica de $\mathcal{QL}$ de la semántica asincrónica de las g-coreografías y, por lo tanto, permitiría redefinir $\mathcal{QL}$ utilizando cualquier modelo de ejecución de sistemas distribuidos, incluyendo modelos de composición sincrónica.

Enumeración de ejecuciones. El algoritmo `QSAT` se basa en la enumeración incremental de todas las ejecuciones del sistema, esta enumeración debe ser adaptada a la semántica de ejecución del modelo de sistema distribuido utilizado.

Fuera de estos cambios, el resto del enfoque y los resultados teóricos acerca de la lógica y el algoritmo de decisión se deberían poder aplicar de manera análoga a sistemas distribuidos basados en composición sincrónica.

8.7. Conclusiones

Presentamos un enfoque y una técnica para el diseño y análisis de propiedades de calidad de sistemas distribuidos basados en envío de mensajes, utilizando coreografías y un modelo general de atributos de calidad. Abordamos este problema al: 1) representar abstractamente los atributos de calidad como símbolos que denotan valores reales, cuyo

comportamiento es capturado completamente por una teoría de RCFs decidible, 2) extender el modelo coreográfico de CFSMs asociando especificaciones de calidad a cada estado de las máquinas, 3) introduciendo $\mathcal{QL}$, una lógica basada en *DLTL*, para expresar propiedades de calidad con una relación de satisfacción basada en ejecuciones de sistemas de comunicación, y 4) proporcionando un procedimiento de semi-decisión para $\mathcal{QL}$ y definiendo un fragmento $\mathcal{QL}^-$ que nos permitió proporcionar un procedimiento de verificación acotada de modelos para la lógica completa. Un subproducto interesante de nuestro enfoque es que se podría utilizar para el monitoreo de cómputos locales para verificar, en tiempo de ejecución, si estos se mantienen dentro de la restricción de las especificaciones de calidad, lo que se puede hacer adaptando técnicas existentes para generación de monitores a partir de tipos de comportamentales [60, 18].

Identificamos dos direcciones principales de investigación a futuro. Por un lado, está la cuestión teórica de si $\mathcal{QL}$ es decidible o no. En este sentido, la similitud de $\mathcal{QL}$ con *DLTL* (cf. Sección 8.1) insinúa una respuesta afirmativa sugiriendo que el problema se puede traducir a verificar la vacuidad de los autómatas de Büchi [24] correspondientes a las fórmulas $\mathcal{QL}$. Sin embargo, la decidibilidad de $\mathcal{QL}$ no es tan fácil de lograr. En general, un sistema de comunicación puede producir un espacio de estados infinito debido a muchas razones por lo que la decisión de satisfacibilidad podría no ser posible en un número finito de pasos. Por ejemplo, debido a las potencialmente infinitas instancias de valores acumulados los atributos de calidad o a que ninguna configuración final podría ser alcanzable. Por otro lado, la usabilidad del enfoque podría mejorarse a través de dos extensiones de $\mathcal{QL}$ y una forma menos exigente de modelar sistemas de comunicación extendidos con especificaciones de calidad. La primera extensión de $\mathcal{QL}$ son acumulaciones *selectivas*, que permiten la acumulación de atributos de calidad solo para algunos estados específicos de ejecuciones. Esto se puede hacer extendiendo la gramática de g-coreografías dada en la Definición 17 con una producción extra de la forma $\mathsf{G} ::= \cdots \mid \lceil \mathsf{G} \rfloor$, identificando la parte de la coreografía relevante para la acumulación.Una segunda extensión de $\mathcal{QL}$ es la introducción de *comodines* como un mecanismo para "ignorar" una subcoreografía. Sintácticamente, se puede representar, una vez más, extendiendo la gramática dada en la Definición 17 con una producción extra, con la forma $\mathsf{G} ::= \cdots \mid _$, donde $_$ se interpreta como un comodín y desempeña el papel de coincidir con cualquier posible g-corerografía. En este caso, se ignora la estructura de la parte de la ejecución que coincide con el comodín pero los atributos se acumulan a lo largo de toda la ejecución. Finalmente, una forma menos exigente de modelar los sistemas extendidos con especificaciones de calidad podría lograrse extendiendo las g-coreografías con especificaciones de calidad que anotan interacciones específicas y extendiendo la función de proyección de las g-coreografías a CFSMs teniendo en cuenta dichas anotaciones. Esta última extensión forma parte de la implementación que presentamos en el siguiente capítulo.

En el siguiente capítulo, presentamos implementación de nuestro procedimiento, que se sustenta sobre el *SMT-solver* Z3 [42] para la satisfacción de las fórmulas atómicas y en ChorGram [36, 32] para la semántica de g-coreografías y CFSMs.

9

MoCheQoS: un verificador acotado de modelos para calidad de servicio

En este capítulo presentamos **MoCheQoS**, un Model-Checker acotado para propiedades de QoS, que implementa el algoritmo que presentamos en Capítulo 8 al aprovechar las características de **ChorGram** [32, 98], un set de herramientas que da soporte al desarrollo coreográfico. A diferencia de otras herramientas que se centran en la verificación en tiempo de ejecución de componentes individuales, **MoCheQoS** permite el análisis estático de propiedades de calidad de sistemas construidos a partir de la composición de servicios de comunicación, basándose en el SLA de cada componente. Además de la descripción de **MoCheQoS**, en este capítulo evaluamos empíricamente su rendimiento a través de: (*a*) el ejemplo presentado en el Capítulo 8 (*b*) casos de estudio tomados de la literatura, incluyendo protocolos de comunicación extraídos automáticamente del código, y (*c*) ejemplos sintéticos construídos para evaluar la escalabilidad de **MoCheQoS**. El desarrollo de **MoCheQoS** requirió extender **ChorGram** con nuevas características: (I) para soportar qCFSM (CFSMs con estados finales y especificaciones de calidad), y (II) para proyectar qCFSMs desde g-coreografías extendidas con especificaciones de calidad. **MoCheQoS**, junto con un tutorial para reproducir los experimentos presentados en este capítulo, está disponible para la comunidad en `https://bitbucket.org/aemartinez/chorgram/` y `https://zenodo.org/records/10139309`.

La estructura del capítulo es la siguiente. La Sección 9.1 revisa trabajos relacionados. La Sección 9.2 discute aspectos de implementación de **MoCheQoS**. La Sección 9.3 presenta una evaluación empírica de **MoCheQoS**. La Sección 9.4 presenta conclusiones y sugiere trabajos futuros. Los resultados presentados en este capítulo fueron previamente publicados como *preprint* en [107].

9.1. Trabajo relacionado

Ubicamos a **MoCheQoS** en la categoría de analizadores estáticos de propiedades de calidad a nivel de sistema. Existe una vasta literatura sobre calidad de servicio, que abarca una amplia gama de contextos y métodos [2, 74] así como modelos formales y procedimientos de análisis que han sido propuestos sin soporte de herramientas.

Se utilizan funciones métricas en [65] para verificar SLAs de sistemas cliente-servidor mediante el demostrador interactivo de teoremas KeY [48]. Nosotros podemos lidiar con sistemas multiparte y el análisis de las propiedades de calidad de **MoCheQoS** es totalmente automático. Otros modelos abstractos de atributos de calidad como quantales [134] o c-semirings [26, 105, 43] han sido propuestos. Del último, [26] propone un cálculo de procesos

capaz de expresar SLAs pero no propone ninguna técnica de análisis automático, en el caso de [105], los autores proponen una variante de μ-cálculo equipado con la capacidad de expresar propiedades de calidad y un algoritmo de análisis asociado, pero no proporcionan una implementación y, en el último caso, [43], los autores también proponen un cálculo de procesos para el razonamiento de atributos de calidad pero también carece de un procedimiento de análisis automático.

La extracción automática de contratos de calidad locales a partir de especificaciones globales de calidad está definida e implementada en [86]; el artículo propone aplicaciones que incluyen el uso de los contratos derivados para el monitoreo pero no se propone ningún procedimiento de análisis estático del comportamiento de los atributos de calidad de los sistemas. En esa dirección, se presentaron algoritmos de monitoreo en [90] donde los contratos se utilizan para la predicción en tiempo real, la composición adaptativa o la verificación de compatibilidad.

Otro enfoque para el análisis automatizado de propiedades cuantitativas de los sistemas es la verificación cuantitativa y probabilística de modelos (*Probabilistic Model Checking* – PMC) [96, 8], implementado en herramientas como PRISM [97]. Las principales diferencias con respecto a nuestro trabajo son el lenguaje de modelado y las propiedades que pueden ser verificadas. Primero, mientras que los modelos de PMC se expresan generalmente como cadenas de Markov, **MoCheQoS** es capaz de analizar sistemas de envío de mensajes asíncronos donde no hay información probabilística disponible y, segundo, nuestro lenguaje para expresar información cuantitativa de estados es más expresivo que las funciones de recompensa adoptadas en [96] ya que permite expresar fórmulas de primer orden sobre atributos de calidad. Por ejemplo, las especificaciones de calidad que mostramos en Ejemplo 9 (en la página 76) no pueden ser expresadas con las funciones de recompensa de PRISM. Finalmente, mientras que en PMC las propiedades se expresan como fórmulas temporales sobre cotas en el valor acumulado esperado de una recompensa, **MoCheQoS** puede verificar fórmulas temporales dinámicas donde los átomos son fórmulas de primer orden sobre atributos de calidad y las modalidades temporales están indexadas con g-coreografías. Como vimos en el Capítulo 8, nuestro enfoque requiere el cómputo de una función de acumulación que recolecta especificaciones de calidad de estados a lo largo de una ejecución, lo cual no es así en el caso de PMC. Por otro lado, los autómatas temporizados [3] se utilizan en UPPAAL [99] para verificar sistemas en tiempo real. Nuestras especificaciones de calidad pueden predicar sobre un atributo de calidad que denote el tiempo pero, a diferencia de en UPPAAL, el comportamiento de los sistemas es independiente de él. Para resumir, no es sencillo comparar **MoCheQoS** con herramientas como PMC o UPPAAL ya que están diseñadas para diferentes propósitos. Extender **MoCheQoS** con tiempo y probabilidades es, de hecho, una dirección interesante para trabajo futuro.

9.2. Arquitectura y funcionamiento de MoCheQoS

A continuación describimos **MoCheQoS** presentando su arquitectura y algunos detalles de implementación. Todos los componentes de **MoCheQoS** están implementados en Haskell; una representación gráfica de la arquitectura de **MoCheQoS** se puede ver en la Figura 9.1 donde las cajas inclinadas representan archivos u objetos de datos, las cajas rectangulares representan módulos o funciones, mientras que las flechas representan flujos de control y datos. Las cajas sombreadas identifican los módulos desarrollados para este trabajo, mientras que las cajas delgadas identifican los módulos de **ChorGram** y otras bi-

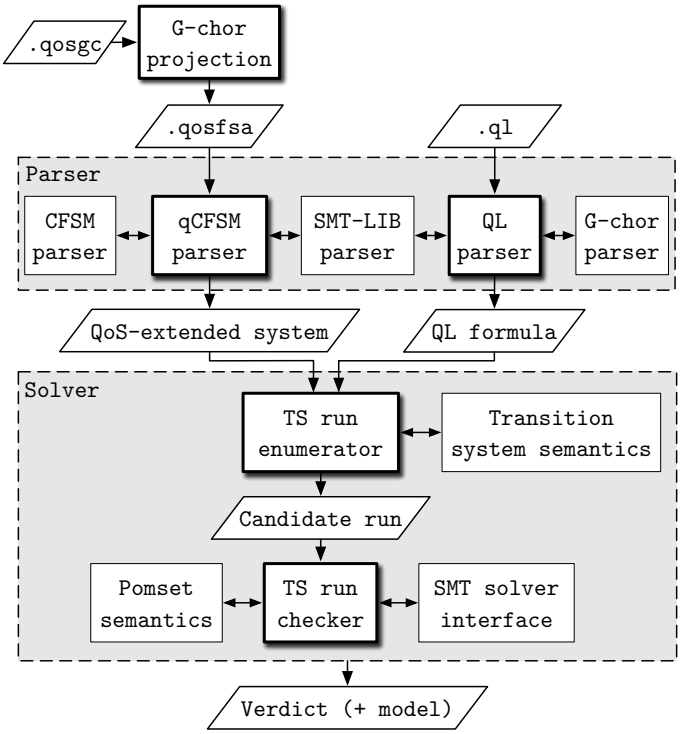


Fig. 9.1: Arquitectura de **MoCheQoS**

bliotecas de código abierto utilizadas por **MoCheQoS**. Los dos elementos principales son el `Parser` y el `Solver` (cajas punteadas grises). El primero transforma las representaciones textuales de las entradas de **MoCheQoS** en representaciones internas de Haskell utilizadas por el segundo. Más precisamente, el `qCFSM parser` aprovecha el `CFSM parser` de **ChorGram** para leer un archivo `.qosfsa` que contiene la descripción textual del sistema que se va a analizar. De manera similar, `QL parser` aprovecha `G-chor parser` (el analizador de g-coreografías de **ChorGram**) para leer el archivo `.ql` que contiene una descripción textual de la fórmula $\mathcal{QL}$ a verificar. Ambos módulos dependen del `SMT-LIB parser` (paquete `smt-lib` de Haskell).

El formato de los archivos `.qosfsa` es una extensión del formato `.fsa` (autómata de estado finito) de **ChorGram**. Esta extensión permite la especificación del conjunto de atributos de calidad de interés con sus operadores de acumulación correspondientes, las especificaciones de calidad asociadas a cada estado de las máquinas y el conjunto de estados finales de cada máquina. Alternativamente, la herramienta admite un archivo `.qosgc`, que es una extensión del formato `.gc` (global choreography) de **ChorGram**. Esta extensión permite especificar la información de los atributos de calidad directamente en la vista global del sistema y luego proyectarlo en las qCFSMs, en lugar de tener que especificarlo en las CFSMs. Esto requirió adaptar los modulos `G-chor projection` y `G-chor parser` de **ChorGram** para soportar especificaciones de calidad.

El formato `.ql`, una representación textual simple de las fórmulas $\mathcal{QL}$, toma prestada la sintaxis de g-coreografías de **ChorGram** para el caso de la modalidad 'until'. (Se puede encontrar un resumen detallado de ambos formatos `.qosfsa` y `.ql` más adelante en la Sección 9.2.1). Después de que el `Parser` procese los archivos de entrada, el `QoS-extended system` y la `QL formula` son la entrada del `Solver`, que consta de los módulos `TS run`

`enumerator` y `TS run checker` (TS aquí denota sistema de transición por sus siglas en inglés).

La combinación de estos módulos implementa el algoritmo de verificación acotada de modelos presentado en el Capítulo 8. El módulo `TS run enumerator` invoca a `Transition system semantics` (el módulo `TS` de **ChorGram**) para generar ejecuciones del sistema a partir de su configuración inicial. Enumera de forma incremental las ejecuciones del sistema dentro del límite dado en el parámetro de entrada `k` y verifica si la ejecución es un modelo de la fórmula $\mathcal{QL}$ al llamar al `TS run checker`. Las fórmulas que contienen subfórmulas de la forma $\Phi\ \mathcal{U}^{\mathsf{G}}\ \Phi'$ requieren que `TS run checker` invoque el módulo `PomsetSemantics` de **ChorGram** para computar el lenguaje de G y para verificar la pertenencia de ejecuciones en ese lenguaje. Para calcular dicho lenguaje, todas las iteraciones G'^{*} en G se despliegan[1] un número finito de veces hasta el límite establecido por el parámetro de entrada `u` (que por defecto es `k` cuando no se especifica explícitamente). Fórmulas involucrando restricciones sobre atributos de calidad requieren que `TS run checker` invoque al `SMT solver interface` para producir y verificar consultas SMT-LIB. El módulo `SMT solver interface` se compone de una versión modificada del paquete `smt-lib` de Haskell para construir la consulta SMT-LIB y del paquete `SimpleSMT` de Haskell para llamar al *smt-solver* Z3. La consulta SMT-LIB para una fórmula atómica ψ se construye para verificar si existe un contraejemplo para el juicio de satisfacción $\mathsf{agg}_S(\pi') \models_{RCF} \psi$. Por lo cual, modificamos el paquete `smt-lib` de Haskell añadiendo soporte para el comando `declare-const` introducido en SMT-LIB 2.5 y para el renombramiento de símbolos en términos de SMT-LIB; estas características son utilizadas por **MoCheQoS** para declarar los atributos de calidad y para implementar la función de acumulación agg_S, respectivamente. La consulta SMT-LIB se envía luego a Z3 usando el paquete `SimpleSMT` de Haskell. Finalmente, si se encuentra una ejecución que satisface la fórmula (`QL formula`), el `Solver` devuelve un veredicto (`Verdict`) positivo y el `modelo` (la ejecución en sí). Si no, el veredicto devuelto indica que no se encontró ningún modelo dentro de los límites dados. Para evitar computar repetidas veces los resultados de las siguientes operaciones sobre las mismas entradas, **MoCheQoS** las memoriza: (I) el cálculo del lenguaje de g-coreografías indexando los operadores 'until', (II) el cálculo del veredicto de la satisfacción de fórmulas atómicas ($\mathsf{agg}_S(\pi') \models_{RCF} \psi$), y (III) la comprobación de pertenencia de una ejecución al lenguaje de una g-coreografía. Los resultados de las operaciones (I) y (III) se almacenan mediante una tabla hash, mientras que los resultados de la operación (II) se almacenan utilizando un árbol binario de búsqueda balanceado.

9.2.1. Formatos de archivos de entrada

Ahora describimos el formato de los archivos de entrada de **MoCheQoS** a través de ejemplos simples de un sistema de comunicación extendido con especificaciones de calidad y una fórmula $\mathcal{QL}$. Considere el siguiente sistema extendido con especificaciones de calidad:

donde $\Gamma_{\mathrm{A}} = \{5 \leq mem \leq 10, cost = 0{,}2 \cdot mem\}$ y $\Gamma_{\mathrm{B}} = \{mem = 0, cost = 1\}$. El contenido del archivo `.qosfsa` que describe el sistema se muestra en la Figura 9.2. El texto dentro del bloque `fsa` (líneas 2–18) es la representación textual de **ChorGram** para

[1] El n-unfolding de G'^{*} es la composición secuencial de G' consigo mismo n veces.

```
1  fsa{
2      .outputs A
3      .state graph
4      0 1 ! x 1
5      1 1 ? y 2
6      2 1 ! z1 1
7      2 1 ! z2 3
8      .marking 0
9      .end
10
11     .outputs B
12     .state graph
13     0 0 ? x 1
14     1 0 ! y 2
15     2 0 ? z1 1
16     2 0 ? z2 3
17     .marking 0
18     .end
19 }
20 qos_attributes{cost:+, mem:max}
21 qos_specifications{
22     A@1 : (and (<= 5 mem) (<= mem 10)
23                (= cost (* 0.2 mem))),
24     B@3 : (and (= cost 1) (= mem 0))
25 }
26 final_states{A: [3], B: [3]}
```

Fig. 9.2: Especificación de un sistema en formato `.qosfsa`

CFSMs (archivos `.fsa`). Las líneas 4–7 y 13–16 corresponden a las transiciones de A y B respectivamente. Cada transición está representada por una línea de la forma `src m act msg tgt`, donde `src` y `tgt` son el estado origen y el estado destino, `m` es el índice, en el sistema, de la otra máquina involucrada (`0` para A y `1` para B), `act` es una acción de entrada (`?`) o de salida (`!`), y `msg` es el mensaje. Los atributos de calidad se dan en la línea 20 como una lista de pares `attr:op` donde `attr` es el nombre del atributo y `op` es su operador de acumulación. El operador de acumulación debe ser un operador binario válido en SMT-LIB. Las líneas 21–25 definen las especificaciones de calidad asociadas a los estados de las máquinas, separadas por una coma. Cada una sigue la sintaxis `ptp@estado : condición` donde `ptp` es el nombre de la CFSM, `estado` es el identificador del estado, y `condición` es un término válido de SMT-LIB donde las únicas variables libres permitidas son los nombres de los atributos de calidad. Los estados que no aparecen en la lista de especificaciones de calidad no tienen ninguna restricción de calidad asociada a ellos (por lo tanto, los valores de sus atributos de calidad no se considerarán durante el análisis). Finalmente, la línea 26 especifica el conjunto de estados finales de cada máquina. Para cada participante tenemos `ptp : [estados]`, donde `ptp` es el nombre de la máquina y `estados` es una lista separada por comas de identificadores de los estados finales de la máquina.

Sea $\mathsf{G} = \mathsf{A}\rightarrow\mathsf{B}: \mathsf{x}; \mathsf{B}\rightarrow\mathsf{A}: \mathsf{y}; \mathsf{A}\rightarrow\mathsf{B}: \mathsf{z2}$, supongamos que estamos interesados en verificar si la fórmula $\langle \mathsf{G} \rangle$ $(2 \leq \mathit{cost} \leq 3)$ se cumple en el sistema descripto en la Figura 9.2.

La representación textual de esta fórmula (archivo `.ql`) se muestra en la Figura 9.3. El índice del operador 'until' se representa en las líneas 3–5 utilizando la sintaxis de g-coreografías de **ChorGram**. La palabra clave `qos` en la línea 7 denota una fórmula atómica con restricciones para los atributos de calidad, que se especifica como un término válido de SMT-LIB donde las únicas variables libres permitidas son los nombres de los atributos

```
True Until
    [
        A -> B: x;
        B -> A: y;
        A -> B: z2
    ]
qos{(and (<= 2 cost) (<= cost 3))}
```

Fig. 9.3: Especificación de una fórmula en formato `.ql`

de calidad.

9.2.2. Usando MoCheQoS

Como se ve, **MoCheQoS** requiere la descripción del sistema de comunicación extendido con especificaciones de calidad. Esto se reduce a asignar una especificación de calidad a los estados de las máquinas de comunicación en el sistema. Cabe señalar que estas máquinas no necesitan ser especificadas a mano; de hecho, se puede aprovechar la operación de *proyección* de **ChorGram** para generar automáticamente los qCFSM a partir de las g-coreografías extendidas con especificaciones de calidad (archivos `.qosgc` introducidos anteriormente), que son el punto de partida del diseño coreográfico *top-down*. Este es uno de los motivos principales para aprovechar las g-coreografías, ya que eliminan la carga de especificar manualmente los modelos locales. Alternativamente, al analizar sistemas existentes, los CFSM pueden extraerse automáticamente del código. De hecho, existen herramientas disponibles para esta tarea, adaptadas a varios lenguajes de programación [84, 85, 64]. Si el sistema que se está analizando fue construido componiendo servicios existentes, las especificaciones de calidad se pueden extraer de los SLA de esos servicios. Una vez que se especifican el modelo del sistema y la propiedad, el usuario puede invocar **MoCheQoS** a través de su interfaz de línea de comandos de la siguiente manera:

```
mocheqos <cmd> <qcfsms> <prop> <k> [opciones]
```

donde

- `<cmd>` es el (nombre de la) operación a aplicar; puede ser `satisfiability` o `validity` para comprobar la satisfacibilidad o validez de la propiedad `prop` en el sistema de comunicación `qcfsms`; en el caso de `validity`, **MoCheQoS** negará `prop` y buscará un contraejemplo,
- `<qcfsms>` es la ruta a un archivo `.qosfsa` que contiene una descripción textual del sistema,
- `<prop>` es la ruta a un archivo `.ql` que contiene una descripción textual de la fórmula $\mathcal{QL}$,
- `<k>` especifica la cota en la longitud de las ejecuciones a considerar, y
- `[opciones]` especifica solicitudes opcionales a **MoCheQoS**:
  - `--unfoldings <u>` especifica el número máximo de *unfoldings* de los operadores $*$ en las g-coreografías en `<prop>` (`u` toma por defecto el valor `k`.),
  - `--show-model` imprime una ejecución que satisface la propiedad, si la hay, y

- `--verbose` ejecuta el análisis en modo detallado, lo que imprime la longitud de las ejecuciones que se están considerando en cada momento y el número total de ejecuciones con dicha longitud.

Además, la opción `--help` imprimirá un mensaje de ayuda.

9.3. Evaluación experimental

Nuestro estudio empírico está destinado tanto a evaluar la escalabilidad de **MoCheQoS** como a mostrar la aplicabilidad de nuestro enfoque. Para nuestros experimentos utilizamos la versión 4.10.2 de Z3 y un MacBook Pro de 8 núcleos (Apple M1) con 16GB de memoria. El rendimiento de **MoCheQoS** depende del costo de comprobar si una fórmula Φ se cumple en alguna de las ejecuciones del sistema S cuya longitud es como máximo `k`. El costo de verificar si una ejecución satisface Φ se ve afectado principalmente por el costo de verificar si los operadores 'until' en Φ están satisfechos y por el costo de resolver consultas SMT-LIB para las restricciones de calidad en Φ. Nuestra evaluación no se enfoca en el costo de ejecutar Z3 para resolver las consultas SMT-LIB (que podría aumentar con la complejidad de las restricciones calidad) ya que usamos Z3 en forma de caja negra que no podemos controlar. Por otro lado, el costo de verificar si los operadores 'until' se satisfacen depende de la complejidad de las g-coreografías que los indexan y de la complejidad de las subfórmulas que acompañan a cada operador 'until', razón por la cual utilizamos fórmulas de la forma $\Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$ para nuestra evaluación. En resumen, nuestro objetivo es evaluar cómo la complejidad de las g-coreografías que indexan el 'until' afecta el rendimiento de **MoCheQoS** y evaluar su aplicabilidad analizando propiedades de sistemas de comunicación extraídos automáticamente de código. Nuestros experimentos fueron diseñados para abordar las siguientes preguntas sobre el rendimiento y la efectividad de **MoCheQoS**:

Unfoldings *de ciclos* ¿Cómo evoluciona el rendimiento, en el peor y mejor escenario, a medida que aumenta el número de *unfoldings* de ciclos en la g-coreografía del operador 'until'?

Elecciones anidadas ¿Cómo evoluciona el rendimiento promedio a medida que aumenta el número de elecciones anidadas en la g-coreografía del operador 'until'?

Extracción de modelo ¿Cuál es la efectividad en el análisis de propiedades de sistemas de comunicación extraídos automáticamente del código?

Para la evaluación de escalabilidad, generamos propiedades sintéticas siguiendo el patrón $\Phi \equiv \Phi_1 \; \mathcal{U}^{\mathsf{G}} \; \Phi_2$ y variando el tamaño y la complejidad de G. Las fórmulas Φ_1 y Φ_2 se crean para cubrir (i) el mejor caso (cualquier ejecución que coincida con el lenguaje de G satisface Φ), (ii) el peor caso (ninguna ejecución que coincida con el lenguaje de G satisface Φ), y (iii) el caso 'promedio' (solo una de las ejecuciones que coincide con el lenguaje de G satisface Φ). Para la evaluación de la aplicabilidad, consideramos el sistema de comunicación de un caso de estudio presentado en [84], que fue extraído automáticamente de código OCaml. Escribimos manualmente las especificaciones de calidad para el sistema, proponemos las fórmulas $\mathcal{QL}$ para ser verificadas en el modelo, y las verificamos usando **MoCheQoS**.

9.3.1. *Unfoldings* de ciclos

Utilizamos una versión de tres participantes del protocolo POP introducido en el Ejemplo 9 con un servidor de autenticación A (modelado a partir del protocolo de autenticación *OAuth* [72]). Los participantes siguen la g-coreografía $\mathsf{G}_{\mathsf{auth}}$ definida como:

$$\mathsf{G}_{\mathsf{auth}} = \mathsf{C}\to\mathsf{A}\colon \mathsf{cred};\ (\mathsf{G}_{\mathsf{token}} + \mathsf{A}\to\mathsf{C}\colon \mathsf{error})$$
$$\mathsf{G}_{\mathsf{token}} = \mathsf{A}\to\mathsf{C}\colon \mathsf{token};\ \mathsf{C}\to\mathsf{S}\colon \mathsf{token};\ (\mathsf{S}\to\mathsf{C}\colon \mathsf{fail} + \mathsf{S}\to\mathsf{C}\colon \mathsf{ok};\ \mathsf{G}_{\mathsf{POP}})$$

donde $\mathsf{G}_{\mathsf{pop}}$ es la g-coreografía del protocolo POP tal como fue definida en el Ejemplo 4. La máquina del cliente contiene 15 estados y 17 transiciones, la máquina del servidor consta de 12 estados 14 transiciones, y el servidor de autenticación 4 estados y 3 transiciones. El LTS del sistema completo consta de 34 estados y 38 transiciones. (Los CFSMs para C, S y A así como el LTS, se muestran en Sección C.1.) Si bien estos números pueden no parecer particularmente altos, debido al intercalamiento de las transiciones en la comunicación asincrónica, el número de ejecuciones del sistema crece exponencialmente cuando el número de *unfoldings* de ciclos en el protocolo POP aumenta, como se muestra en la Figura 9.4. Además, vale la pena recordar que los CFSMs abstraen las cómputos locales y

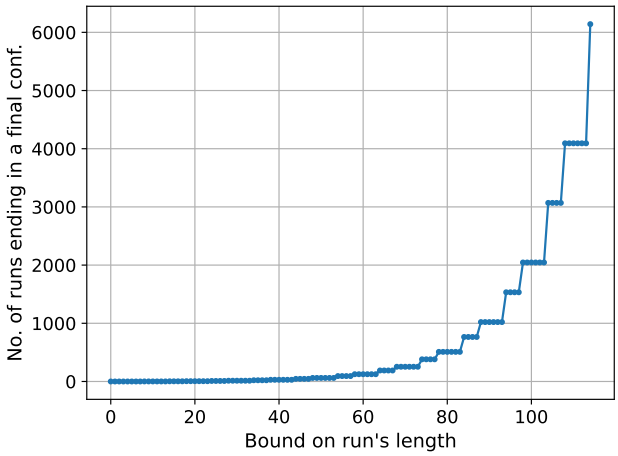


Fig. 9.4: Tamaño del espacio de búsqueda versus k

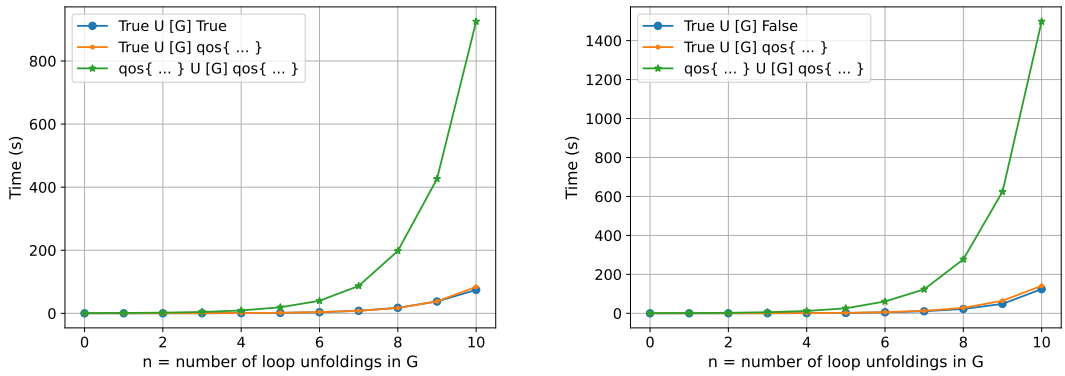
se enfocan solamente en las acciones de comunicación. Por lo tanto, el número de estados y transiciones solo reflejan el tamaño del protocolo de comunicación y no necesariamente el tamaño de la implementación.

Adjuntamos una especificación de calidad simple a cada estado de cada máquina en el protocolo POP, determinando el valor de cinco atributos de calidad. Luego, generamos sintéticamente seis familias de fórmulas $\mathcal{QL}$ con la forma $\Phi_1\ \mathcal{U}^{\mathsf{G}}\ \Phi_2$ donde $\mathsf{G} = \mathsf{G}_{\mathsf{init}}; {\mathsf{G}_{\mathsf{msg}}}^{n}$, para $1 \leq n \leq 10$, donde $\mathsf{G}_{\mathsf{init}}$ describe las interacciones iniciales antes de cualquier recepción de un correo[2] y $\mathsf{G}_{\mathsf{msg}}$ caracteriza la lectura de un mensaje del buzón[3]. Las primeras tres familias de fórmulas se construyen para ser satisfacibles mientras que las últimas tres para ser insatisfacibles. Las fórmulas insatisfacibles se construyen garantizando que ninguna ejecución compatible con G satisface Φ_2. Para cada fórmula ejecutamos **MoCheQoS** con una cota $\mathtt{k} = 16 + 10n$, que garantiza que las ejecuciones del sistema que cumplen con

[2] $\mathsf{G}_{\mathsf{init}} = \mathsf{C}\to\mathsf{A}\colon \mathsf{cred}; \mathsf{A}\to\mathsf{C}\colon \mathsf{token}; \mathsf{C}\to\mathsf{S}\colon \mathsf{token}; \mathsf{S}\to\mathsf{C}\colon \mathsf{ok}; \mathsf{C}\to\mathsf{S}\colon \mathsf{helo}; \mathsf{S}\to\mathsf{C}\colon \mathsf{int}$

[3] $\mathsf{G}_{\mathsf{msg}} = \mathsf{C}\to\mathsf{S}\colon \mathsf{read}; \mathsf{S}\to\mathsf{C}\colon \mathsf{size}; \mathsf{C}\to\mathsf{S}\colon \mathsf{retr}; \mathsf{S}\to\mathsf{C}\colon \mathsf{msg}; \mathsf{C}\to\mathsf{S}\colon \mathsf{ack}$

G son alcanzadas. De esta manera, k varía entre 16 y 116, y el tamaño del espacio de búsqueda para cada k es precisamente el que se muestra en la Figura 9.4. En la Figura 9.5a graficamos el tiempo que tarda **MoCheQoS** para encontrar un modelo para las tres familias de fórmulas satisfacibles en función de n. Las tres familias difieren en cómo se definen Φ_1 y Φ_2: (I) ambas como valores de verdad atómicos, (II) Φ_1 como un valor de verdad atómico y Φ_2 como una restricción de calidad, o (III) ambas como restricciones de calidad. Finalmente, en la Figura 9.5b mostramos el tiempo que le toma a **MoCheQoS** informar que no se encontró ningún modelo para las tres familias de fórmulas insatisfacibles en función de n. Los resultados muestran que la principal fuente de carga computacional, a medida que aumenta el número de *unfoldings* de ciclos, es la verificación de la restricción de calidad en Φ_1, el primer operando del operador 'until'. En la Figura 9.5a y en la Figura 9.5b, esto se manifiesta por la curva verde creciendo significativamente más rápido que las otras dos curvas. La explicación de esto es que, debido a la semántica de 'until', la verificación de Φ_1 debe realizarse en cada prefijo de la ejecución y, cuando Φ_1 es una restricción de calidad, cada verificación se realiza llamando a Z3 con una consulta SMT-LIB diferente.



(a) Instancias satisfacibles *(b)* Instancias insatisfacibles

Fig. 9.5: Tiempo de ejecución para analizar fórmulas 'until' (in)satisfacibles

9.3.2. Elecciones anidadas

Consideramos un sistema de dos participantes que se turnan para enviarse un mensaje el uno al otro; en cada turno el remitente hace una elección no determinístca entre dos mensajes distintos. Debido a la naturaleza ramificada de este comportamiento, el número de ejecuciones del sistema crece como la función 2^n, donde n es el número de elecciones anidadas (es decir, el número de turnos). Generamos sintéticamente sistemas con este comportamiento variando n de 1 a 10. Vale la pena notar que las elecciones anidadas corresponden a condicionales anidados en un programa y las métricas de software aceptadas recomiendan evitar grandes números de condicionales anidados. En particular, un límite superior aceptado de *complejidad ciclomática*[4] es 15, lo que corresponde a menos de 4 condicionales anidados. Para generar estos sistemas, elaboramos una g-coreografía extendida con especificaciones de calidad en formato `.qosgc` y luego aprovechamos la `G-chor`

[4] La complejidad ciclomática [114] mide la complejidad de los programas de acuerdo al número de caminos independientes representados en el código fuente.

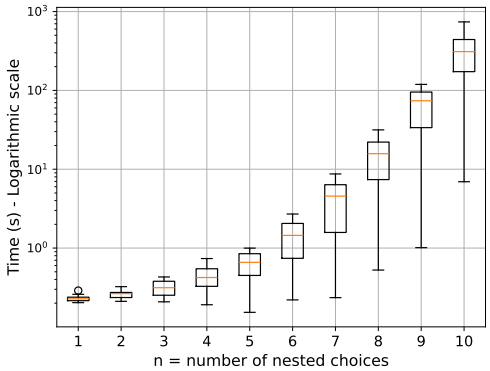
`projection` de **ChorGram** para obtener las qCFSMs del sistema. Las especificaciones de calidad comprenden cinco atributos de calidad y determinan valores únicos para ellos en cada estado final, lo que permite la construcción de fórmulas $\mathcal{QL}$ que son satisfechas por solo una ejecución del sistema. De esta manera, podemos usar estos casos generados para evaluar el rendimiento de **MoCheQoS** en encontrar la única ejecución que satisface una fórmula en un espacio de búsqueda de tamaño exponencial en n. La fórmula se genera

```
Bob → Alice: m0 ; {
  Alice → Bob: m0; Bob → Alice: leaf1
  +
  Alice → Bob: m1; Bob → Alice: leaf2
}

+

Bob → Alice: m1; {
  Alice → Bob: m0; Bob → Alice: leaf3
  +
  Alice → Bob: m1; Bob → Alice: leaf4
}
```

(a) Coreografía generada para $n = 2$



(b) Tiempo de ejecución promedio versus n

Fig. 9.6: Rendimiento en el caso 'promedio' para fórmulas con elecciones anidadas

siguiendo el patrón $\top \ \mathcal{U}^{\mathsf{G}} \ \psi$, donde G coincide con cada ejecución del sistema y ψ es una restricción de calidad, determinando el valor de los cinco atributos, que solo es satisfecha por una única ejecución. En Figura 9.6a mostramos la g-coreografía G generada para dos elecciones anidadas. Consulte Sección C.2 para una vista detallada de los archivos utilizados en este caso de estudio. El límite `k` está establecido lo suficientemente alto para garantizar que todas las ejecuciones del sistema son alcanzadas por el análisis. Para cada valor de n, generamos 100 instancias aleatorias diferentes de la fórmula $\mathcal{QL}$, donde la única ejecución que satisface la fórmula se elige al azar, y ejecutamos **MoCheQoS** en cada instancia. En la Figura 9.6b mostramos los resultados como un *boxplot* por número de elecciones anidadas donde el eje y tiene escala logarítmica. Vale la pena observar que tanto el tiempo medio de ejecución como su varianza aumentan a medida que n aumenta. Esto se debe al hecho de que la diferencia de tiempo entre los escenarios de mejor y peor caso aumenta con n. Estos escenarios se corresponden respectivamente con encontrar el modelo en la primera ejecución enumerada que coincide con G o en la última. El sesgo aparente en la dispersión de cada *boxplot* hacia tiempos de ejecución más bajos es solo un efecto visual debido a la escala logarítmica del eje y.

9.3.3. Extracción de modelo

En [84] se presenta una biblioteca que verifica la compatibilidad de programas concurrentes basados en envío de mensajes. Esta biblioteca presenta un algoritmo de inferencia de tipos que extrae CFSMs directamente de código OCaml. Para evaluar la aplicabilidad de **MoCheQoS** tomamos prestadas las CFSMs inferidas del código OCaml del caso de estudio presentado en [84], las reproducimos en la Figura 9.7. En el sistema de Figura 9.7 el User solicita al Master resolver un problema computacional (cuya naturaleza es irrelevante aquí). El Master divide el problema en dos tareas y las envía al Worker, quien devuelve

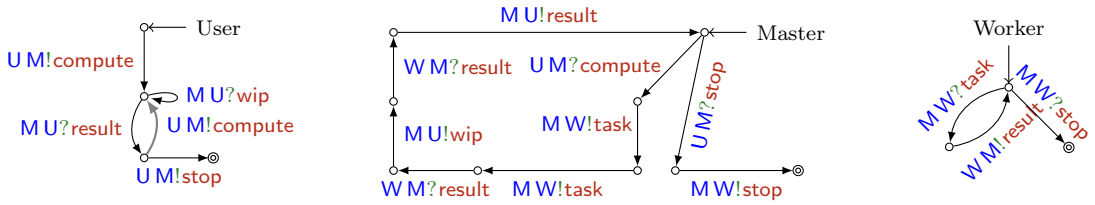


Fig. 9.7: Sistema de comunicación del caso de estudio presentado en [84]

al Master las soluciones de cada tarea. Antes de recoger el segundo resultado parcial del Worker, el Master envía un mensaje de trabajo en progreso ('wip') al User. Finalmente, tras recibir el segundo resultado parcial, el Master combina las soluciones y envía el resultado final al User. La flecha gris gruesa en la Figura 9.7 es la única adición que hicimos al caso de estudio original para permitir al User decidir o bien hacer nuevas solicitudes de forma iterativa o bien parar.

Las especificaciones de calidad involucran p (precio), t (número de tareas computadas) y $pmem$ (memoria asignada); han sido especificadas y asignadas manualmente a estados en las máquinas donde estos son relevantes. Por ejemplo, modelamos la instancia del problema para consumir como máximo 5 unidades de memoria, por lo tanto, especificamos consistentemente cotas para $pmem$ en los estados donde una máquina necesita asignar memoria para el problema. Del mismo modo, limitamos a una unidad la memoria necesaria para el resultado, a diez unidades el precio cobrado por el Master una vez que se realiza el cómputo, mientras que el Worker cobra un precio que depende del tamaño de la tarea. Finalmente, nos interesa verificar las siguientes propiedades $\mathcal{QL}$ del modelo:

$$\begin{array}{lcl@{\qquad}lcl} \Phi_1 & \equiv & [\mathsf{G}](t \cdot 6 \leq p < 12{,}5) & \Phi_3 & \equiv & [\mathsf{G}^*](1 \leq pmem < 10) \\ \Phi_2 & \equiv & [\mathsf{G}^*](t \cdot 6 \leq p < 12{,}5) & \Phi_4 & \equiv & (p \leq t \cdot 12{,}5)\ \mathcal{U}^{\mathsf{G}}\ ([\mathsf{G}^*]\ p \leq 25) \end{array}$$

donde G describe el proceso de resolver una instancia del problema, comenzando en U→M: compute y terminando en M→U: result. (Consultar Sección C.3 para una vista detallada de los archivos utilizados en este caso de estudio.) La fórmula Φ_1 utiliza la modalidad 'necesidad' de la lógica temporal para expresar límites en el precio de la resolución de una instancia de problema. Las fórmulas Φ_2 y Φ_3 usan la modalidad 'necesidad' para expresar límites en el precio y la memoria utilizados después de resolver cualquier número de instancias de problema.

Ejecutamos **MoCheQoS** en estas tres fórmulas y obtenemos un modelo usando como cota `k` $= 18$, que es la longitud de las ejecuciones que primero coinciden con G y luego avanzan hacia una configuración final. En cada caso, **MoCheQoS** es capaz de encontrar el modelo en menos de 0.4 segundos. Al usar el comando `validity` también podemos usar **MoCheQoS** como un procedimiento de búsqueda de contraejemplos al verificar la satisfactibilidad de la negación de la fórmula. Con una cota `k` $= 32$ toma, respectivamente, 35, 206 y 208 segundos para informar que no se encontraron contraejemplos para Φ_1, Φ_2 y Φ_3.

La fórmula Φ_4 establece que 1) hasta la resolución de la primera instancia del problema, el precio se encuentra por debajo de un límite que depende del número de tareas calculadas, y 2) después, el precio siempre está acotado por 25 justo después de que la resolución de alguna instancia de un problema se haya completado. Usando una cota `k` $= 18$, **MoCheQoS** tarda 0.7 segundos en encontrar un modelo para Φ_4. Si usamos **MoCheQoS** como un

procedimiento para encontrar contraejemplos, la cota no es suficiente para encontrar un contraejemplo: tarda 7 segundos en informar que no se encontraron contraejemplos. Sin embargo, si aumentamos la cota a $\mathtt{k} = 32$, **MoCheQoS** encuentra un contraejemplo en 35 segundos. Lo que muestra que, a diferencia de las fórmulas anteriores, Φ_4 es satisfacible pero no es válida.

9.4. Conclusiones y trabajo futuro

Presentamos **MoCheQoS**, una herramienta para verificar propiedades de calidad en modelos coreográficos de sistemas asíncronos basados en envío de mensajes, que se basa en la lógica y el procedimiento de semi-decisión que presentamos en Capítulo 8. Nuestra herramienta proporciona soporte para la descripción textual de sistemas de comunicación extendidos con especificaciones de calidad, la especificación de las propiedades de calidad a verificar, y la verificación automatizada de dichas propiedades a través de un algoritmo de verificación acotada de modelos. La satisfacibilidad de las restricciones de calidad en fórmulas atómicas se delega al *smt-solver* Z3 mientras que **ChorGram** se utiliza para manejar los modelos coreográficos y su semántica. Evaluamos **MoCheQoS** a través de una serie de casos de estudio, para evaluar su escalabilidad y aplicabilidad. Demostramos el rendimiento de la herramienta con respecto al número de *unfoldings* de ciclos y el número de elecciones anidadas en las coreografías que indexan el operador 'until' de la lógica. Además, demostramos la efectividad de la herramienta en el manejo de sistemas de comunicación extraídos automáticamente del código. Hasta donde sabemos, **MoCheQoS** es la primera herramienta que soporta el análisis automático de propiedades de calidad para modelos coreográficos de sistemas basado en envío de mensajes. Notablemente, **MoCheQoS** puede manejar cualquier atributo de calidad que tome valores en los números reales (si está equipado con un operador de acumulación), lo que lo hace altamente versátil.

Optimizar **MoCheQoS** es una dirección natural para trabajos futuros ya que, como se discutió en Sección 9.3, hay dos factores principales en los que se puede mejorar el rendimiento de la herramienta; uno es la asincronía de la comunicación y el otro es que el número de ejecuciones crece exponencialmente con el número de elecciones anidadas de la g-coreografía. Estamos considerando semánticas abstractas donde las ejecuciones se dividen en clases de equivalencia de modo que solo tendríamos que verificar las ejecuciones representativas de dichas clases. También, planeamos hacer más eficiente la comprobación de pertenencia de ejecuciones dentro del lenguaje de una g-coreografía reemplazando la semántica de *pomsets* implementada en **ChorGram** con una basada en *estructuras de eventos*. De hecho, las *estructuras de eventos*, similares a los *tries*, representan ejecuciones con prefijos comunes con estructuras en forma de árbol. De esta manera, evitamos duplicar las comprobaciones en prefijos comunes como se hace actualmente con el uso de *pomsets*. Sin embargo, argumentamos que, en la práctica, este es un problema menor dado que no esperamos tener un alto número de elecciones anidadas en g-coreografías realistas.

Aunque son alentadores, nuestros experimentos son preliminares y planeamos considerar más casos de estudio de nivel industrial. Específicamente, planeamos analizar sistemas sustentados por la nube de *AWS* construidos mediante la composición de servicios tomados de un gran catálogo. Esos servicios están equipados con SLAs individuales centrados en atributos de calidad como tiempo de actividad, disponibilidad y costo (al igual que los ejemplos que usamos en este capítulo). Por ejemplo, los SLA para servicios de persistencia y almacenamiento (como el servicio *DynamoDB*, *S3*, etc.) establecen el precio de acuerdo

con, por ejemplo, el tamaño (en Gb) así como la frecuencia de acceso a los datos almacenados. Además, en los SLA para servicios de *serverless computing* (por ejemplo, *AWS Lambda*), el precio puede depender de atributos cuantitativos, incluyendo la arquitectura de ejecución específica, la cantidad de solicitudes y la duración de la ejecución. Esto requeriría extender **MoCheQoS** con interacciones sincrónicas para adaptarse a la arquitectura de *AWS* donde los servicios interactúan de forma sincrónica por defecto [6].

10

Conclusiones generales

En esta tesis nos propusimos contribuir al campo del análisis automático de propiedades de calidad en sistemas distribuidos basados en servicios. Nuestras contribuciones se basan en el enfoque de los métodos formales aplicados a la ingeniería de software, es decir, métodos de análisis de artefactos de software cuyas técnicas pueden explicarse en términos lógico-matemáticos. Desde este enfoque, nos concentraremos en el análisis de propiedades que predican sobre atributos cuantitativos, esto es, aquellos atributos de calidad que admiten algún tipo de medición. Esto nos permitió brindar un tratamiento general y homogéneo a los atributos de calidad, sin hacer suposiciones sobre su naturaleza, excepto por la condición de que sean cuantitativos.

Todas nuestras contribuciones se basan en el desarrollo de lenguajes formales para la especificación de aspectos de calidad de sistemas distribuidos basados en servicios y en el diseño de técnicas de análisis automático para estos lenguajes. En particular, a lo largo de la Parte I desarrollamos técnicas para modelar y analizar contratos de calidad desde distintas perspectivas. Primero, propusimos lenguajes para caracterizar el espacio de valores de los atributos de calidad de un contrato a partir de restricciones convexas y técnicas de análisis que no sólo permiten verificar la compatibilidad entre un contrato de provisión y uno de requisito, sino también calcular el grado de satisfacción de un contrato de provisión con respecto a un contrato de requisito. Luego, propusimos un modelo formal de contratos de calidad que contemple la índole probabilística de los atributos de calidad junto con una técnica de análisis que permite verificar si un contrato de esta naturaleza satisface una propiedad determinada. Finalmente, en la Parte II nos concentramos, en cambio, en la extensión de modelos de composición asincrónica de sistemas distribuidos con información acerca de los valores de los atributos de calidad de los servicios que los componen y en el diseño de una técnica de análisis para estos modelos. Esto nos llevó, por un lado, al desarrollo de una lógica temporal para describir las propiedades a verificar y, por otro lado, a la implementación de **MoCheQoS**, una herramienta que permite verificar estas propiedades en modelos coreográficos basados en envío de mensajes.

Consideramos que esta tesis demuestra la viabilidad de un enfoque formal para abordar los aspectos de calidad en sistemas distribuidos basados en servicios, de manera análoga a como se abordan los aspectos funcionales. En particular, este enfoque formal facilita el desarrollo de técnicas de análisis automático aplicables en el proceso de diseño y desarrollo de este tipo de sistemas. Esperamos que esta tesis contribuya a consolidar los métodos formales como una herramienta valiosa en el desarrollo de sistemas distribuidos basados en servicios. No obstante, es importante destacar que aún existen preguntas abiertas sobre la metodología adecuada para el desarrollo de especificaciones y modelos formales de atributos de calidad mediante los lenguajes propuestos en esta tesis. Aunque hemos tomado

medidas para reducir la carga de trabajo asociada al desarrollo de estos modelos, como proponer técnicas estadísticas para estimar especificaciones de calidad a partir de datos de ejecución y mostrar cómo integrar nuestra técnica de análisis de sistemas distribuidos con herramientas de extracción automática de modelos a partir de código, este no ha sido el enfoque principal de esta tesis y consideramos que es un tema que merece ser investigado en el futuro.

Bibliografía

[1] Vikas Agarwal y Pankaj Jalote. «From Specification to Adaptation: An Integrated QoS-driven Approach for Dynamic Adaptation of Web Service Compositions». En: *2010 IEEE International Conference on Web Services*. Jul. de 2010, págs. 275-282. DOI: 10.1109/ICWS.2010.39.

[2] Aldeida Aleti y col. «Software Architecture Optimization Methods: A Systematic Literature Review». En: *IEEE Transactions on Software Engineering* 39.5 (mayo de 2013), págs. 658-683. ISSN: 1939-3520. DOI: 10.1109/TSE.2012.64.

[3] Rajeev Alur y David L. Dill. «A Theory of Timed Automata». En: *Theoretical Computer Science* 126.2 (abr. de 1994), págs. 183-235. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)90010-8.

[4] Marco Autili, Paola Inverardi y Massimo Tivoli. «Automated Synthesis of Service Choreographies». En: *IEEE Software* 32.1 (ene. de 2015), págs. 50-57. ISSN: 1937-4194. DOI: 10.1109/MS.2014.131.

[5] Marco Autili y col. «CHOReVOLUTION: Automating the Realization of Highly–Collaborative Distributed Applications». En: *Coordination Models and Languages*. Ed. por Hanne Riis Nielson y Emilio Tuosto. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, págs. 92-108. ISBN: 978-3-030-22397-7. DOI: 10.1007/978-3-030-22397-7_6.

[6] Amazon. *Set up asynchronous invocation of the backend Lambda function*. https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-integration-async.html.

[7] Roberto Bagnara, Patricia M. Hill y Enea Zaffanella. «Applications of Polyhedral Computations to the Analysis and Verification of Hardware and Software Systems». En: *Theoretical Computer Science* 410.46 (nov. de 2009), págs. 4672-4691. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.07.033.

[8] Christel Baier y Joost-Pieter Katoen. *Principles of Model Checking*. Cambridge, Mass: The MIT Press, 2008. ISBN: 978-0-262-02649-9.

[9] Massimo Bartoletti y col. «Call-by-Contract for Service Discovery, Orchestration and Recovery». En: *Rigorous Software Engineering for Service-Oriented Systems: Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing*. Ed. por Martin Wirsing y Matthias Hölzl. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, págs. 232-261. ISBN: 978-3-642-20401-2. DOI: 10.1007/978-3-642-20401-2_11.

[10] Samik Basu, Tevfik Bultan y Meriem Ouederni. «Deciding Choreography Realizability». En: *Proceedings of the 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '12. New York, NY, USA: Association for Computing Machinery, ene. de 2012, págs. 191-202. ISBN: 978-1-4503-1083-3. DOI: 10.1145/2103656.2103680.

[11] Michael Ben-Or, Dexter Kozen y John Reif. «The Complexity of Elementary Algebra and Geometry». En: *Journal of Computer and System Sciences* 32.2 (abr. de 1986), págs. 251-264. ISSN: 0022-0000. DOI: 10.1016/0022-0000(86)90029-2.

[12] Fausto Bernardini. «Integration of Polynomials over N-Dimensional Polyhedra». En: *Computer-Aided Design* 23.11 (ene. de 1991), págs. 51-58. ISSN: 0010-4485. DOI: 10.1016/0010-4485(91)90099-I.

[13] Armin Biere y col., eds. *Handbook of Satisfiability: Second Edition*. Vol. 336. Frontiers in Artificial Intelligence and Applications. IOS Press, feb. de 2021. ISBN: 978-1-64368-160-3 978-1-64368-161-0. DOI: 10.3233/FAIA336.

[14] Garrett Birkhoff y Saunders Mac Lane. *A Survey of Modern Algebra*. AKP Classics. Wellesley, Mass: A.K. Peters, 1997. ISBN: 978-1-56881-068-3.

[15] Dines Bjørner y Klaus Havelund. «40 Years of Formal Methods». En: *FM 2014: Formal Methods*. Ed. por Cliff Jones, Pekka Pihlajasaari y Jun Sun. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, págs. 42-61. ISBN: 978-3-319-06410-9. DOI: 10.1007/978-3-319-06410-9_4.

[16] Laura Bocchi, Hernan Melgratti y Emilio Tuosto. «On Resolving Non-determinism in Choreographies». En: *Logical Methods in Computer Science* 16.3 (sep. de 2020). ISSN: 1860-5974. DOI: 10.23638/LMCS-16(3:18)2020.

[17] Laura Bocchi y col. «A Theory of Design-by-Contract for Distributed Multiparty Interactions». En: *CONCUR 2010 - Concurrency Theory*. Ed. por Paul Gastin y François Laroussinie. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, págs. 162-176. ISBN: 978-3-642-15375-4. DOI: 10.1007/978-3-642-15375-4_12.

[18] Laura Bocchi y col. «Monitoring Networks through Multiparty Session Types». En: *Theoretical Computer Science* 669 (2017), págs. 33-58. DOI: 10.1016/j.tcs.2017.02.009.

[19] Jonas Bonér. *Reactive Microsystems - the Evolution of Microservices at Scale*. O'Reilly, 2018. ISBN: 978-1-4919-9435-1.

[20] Athman Bouguettaya, Quan Z. Sheng y Florian Daniel, eds. *Web Services Foundations*. New York, NY: Springer New York, 2014. ISBN: 978-1-4614-7517-0. DOI: 10.1007/978-1-4614-7518-7.

[21] Object Management Group. *Business Process Model and Notation*.

[22] Daniel Brand y Pitro Zafiropulo. «On Communicating Finite-State Machines». En: *Journal of the ACM* 30.2 (abr. de 1983), págs. 323-342. ISSN: 0004-5411. DOI: 10.1145/322374.322380.

[23] Manuel Bronstein. *Symbolic Integration I: Transcendental Functions*. 2.ª ed. Algorithms and Computation in Mathematics, Algorithms,Computation Math. Vol 1 Bronstein,M.:Symbolic Integrat. Berlin Heidelberg: Springer-Verlag, 2005. ISBN: 978-3-540-21493-9. DOI: 10.1007/b138171.

[24] J. Richard Büchi. «On a Decision Method in Restricted Second Order Arithmetic». En: *The Collected Works of J. Richard Büchi*. Ed. por Saunders Mac Lane y Dirk Siefkes. New York, NY: Springer, 1990, págs. 425-435. ISBN: 978-1-4613-8928-6. DOI: 10.1007/978-1-4613-8928-6_23.

[25] Benno Büeler, Andreas Enge y Komei Fukuda. «Exact Volume Computation for Polytopes: A Practical Study». En: *Polytopes — Combinatorics and Computation*. Ed. por Gil Kalai y Günter M. Ziegler. DMV Seminar. Basel: Birkhäuser, 2000, págs. 131-154. ISBN: 978-3-0348-8438-9. DOI: 10.1007/978-3-0348-8438-9_6.

[26] Maria Grazia Buscemi y Ugo Montanari. «CC-Pi: A Constraint-Based Language for Specifying Service Level Agreements». En: *Programming Languages and Systems*. Ed. por Rocco De Nicola. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, págs. 18-32. ISBN: 978-3-540-71316-6. DOI: 10.1007/978-3-540-71316-6_3.

[27] Radu Calinescu y col. «Dynamic QoS Management and Optimization in Service-Based Systems». En: *IEEE Transactions on Software Engineering* 37.3 (mayo de 2011), págs. 387-409. ISSN: 1939-3520. DOI: 10.1109/TSE.2010.92.

[28] Radu Calinescu y col. «Formal Verification With Confidence Intervals to Establish Quality of Service Properties of Software Systems». En: *IEEE Transactions on Reliability* 65.1 (mar. de 2016), págs. 107-125. ISSN: 1558-1721. DOI: 10.1109/TR.2015.2452931.

[29] Fabio Casati y col. «Adaptive and Dynamic Service Composition in eFlow». En: *Advanced Information Systems Engineering*. Ed. por Benkt Wangler y Lars Bergman. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2000, págs. 13-31. ISBN: 978-3-540-45140-2. DOI: 10.1007/3-540-45140-4_3.

[30] Manman Chen y col. «Verification of Functional and Non-functional Requirements of Web Service Composition». En: *Formal Methods and Software Engineering*. Ed. por Lindsay Groves y Jing Sun. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, págs. 313-328. ISBN: 978-3-642-41202-8. DOI: 10.1007/978-3-642-41202-8_21.

[31] Yuansi Chen y col. «Fast MCMC Sampling Algorithms on Polytopes». En: *The Journal of Machine Learning Research* 19.1 (ene. de 2018), págs. 2146-2231. ISSN: 1532-4435.

[32] Alex Coto y col. *ChorGram: Tool Support for Choreographic Deveelopment*. Available at https://bitbucket.org/emlio_tuosto/chorgram/wiki/Home. 2015.

[33] George E. Collins. «Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition». En: *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Ed. por Bob F. Caviness y Jeremy R. Johnson. Texts and Monographs in Symbolic Computation. Vienna: Springer, 1998, págs. 85-121. ISBN: 978-3-7091-9459-1. DOI: 10.1007/978-3-7091-9459-1_4.

[34] M. Comuzzi y B. Pernici. «An Architecture for Flexible Web Service QoS Negotiation». En: *Ninth IEEE International EDOC Enterprise Computing Conference (EDOC'05)*. Sep. de 2005, págs. 70-79. DOI: 10.1109/EDOC.2005.4.

[35] World Wide Web Consortium. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. On-line.

[36] Alex Coto, Roberto Guanciale y Emilio Tuosto. «Choreographic Development of Message-Passing Applications». En: *Coordination Models and Languages*. Ed. por Simon Bliudze y Laura Bocchi. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, págs. 20-36. ISBN: 978-3-030-50029-0. DOI: 10.1007/978-3-030-50029-0_2.

[37] James H. Davenport y Joos Heintz. «Real Quantifier Elimination Is Doubly Exponential». En: *Journal of Symbolic Computation* 5.1 (feb. de 1988), págs. 29-35. ISSN: 0747-7171. DOI: 10.1016/S0747-7171(88)80004-X.

[38] Philip J. Davis y Philip Rabinowitz. *Methods of Numerical Integration*. 2nd ed. Computer Science and Applied Mathematics. Orlando: Academic Press, 1984. ISBN: 978-0-12-206360-2.

[39] Giuseppe De Giacomo y Moshe Y. Vardi. «Linear Temporal Logic and Linear Dynamic Logic on Finite Traces». En: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. IJCAI '13. Beijing, China: AAAI Press, ago. de 2013, págs. 854-860. ISBN: 978-1-57735-633-2.

[40] J. A. De Loera y col. «Software for Exact Integration of Polynomials over Polyhedra». En: *Computational Geometry* 46.3 (abr. de 2013), págs. 232-252. ISSN: 0925-7721. DOI: 10.1016/j.comgeo.2012.09.001.

[41] Leonardo De Moura y Nikolaj Bjørner. «Satisfiability modulo Theories: Introduction and Applications». En: *Communications of the ACM* 54.9 (sep. de 2011), págs. 69-77. ISSN: 0001-0782. DOI: 10.1145/1995376.1995394.

[42] Leonardo de Moura y Nikolaj Bjørner. «Z3: An Efficient SMT Solver». En: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. por C. R. Ramakrishnan y Jakob Rehof. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, págs. 337-340. ISBN: 978-3-540-78800-3. DOI: 10.1007/978-3-540-78800-3_24.

[43] Rocco De Nicola y col. «A Process Calculus for QoS-Aware Applications». En: *Coordination Models and Languages*. Ed. por Jean-Marie Jacquet y Gian Pietro Picco. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, págs. 33-48. ISBN: 978-3-540-32006-7. DOI: 10.1007/11417019_3.

[44] Flavio De Paoli y col. «Non-Functional Properties and Service Level Agreements in Service-Oriented Computing (NFPSLA-SOC '07) – Organizers' Workshop Summary». En: *Service-Oriented Computing - ICSOC 2007 Workshops*. Ed. por Elisabetta Di Nitto y Matei Ripeanu. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, págs. 43-44. ISBN: 978-3-540-93851-4. DOI: 10.1007/978-3-540-93851-4_5.

[45] Elisabetta Di Nitto y col. «Negotiation of Service Level Agreements: An Architecture and a Search-Based Approach». En: *Service-Oriented Computing – ICSOC 2007*. Ed. por Bernd J. Krämer, Kwei-Jay Lin y Priya Narasimhan. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, págs. 295-306. ISBN: 978-3-540-74974-5. DOI: 10.1007/978-3-540-74974-5_24.

[46] Edsger W. Dijkstra. «The Humble Programmer». En: *Communications of the ACM* 15.10 (oct. de 1972), págs. 859-866. ISSN: 0001-0782. DOI: 10.1145/355604.361591.

[47] Joshua V. Dillon y col. *TensorFlow Distributions*. Nov. de 2017. DOI: 10.48550/arXiv.1711.10604. arXiv: 1711.10604 [cs, stat].

[48] Crystal Chang Din, Richard Bubel y Reiner Hähnle. «KeY-ABS: A Deductive Verification Tool for the Concurrent Modelling Language ABS». En: *Automated Deduction - CADE-25*. Ed. por Amy P. Felty y Aart Middeldorp. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, págs. 517-526. ISBN: 978-3-319-21401-6. DOI: 10.1007/978-3-319-21401-6_35.

[49] Rick Durrett. *Probability: Theory and Examples*. 5.ª ed. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge: Cambridge University Press, 2019. ISBN: 978-1-108-47368-2. DOI: 10.1017/9781108591034.

[50] M. E. Dyer y A. M. Frieze. «On the Complexity of Computing the Volume of a Polyhedron». En: *SIAM Journal on Computing* 17.5 (oct. de 1988), págs. 967-974. ISSN: 0097-5397. DOI: 10.1137/0217060.

[51] Niklas Eén y Niklas Sörensson. «An Extensible SAT-solver». En: *Theory and Applications of Satisfiability Testing, SAT 2003*. Ed. por Enrico Giunchiglia y Armando Tacchella. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, págs. 502-518. ISBN: 978-3-540-24605-3. DOI: 10.1007/978-3-540-24605-3_37.

[52] Joyce El Hadad, Maude Manouvrier y Marta Rukoz. «TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition». En: *IEEE Transactions on Services Computing* 3.1 (ene. de 2010), págs. 73-85. ISSN: 1939-1374. DOI: 10.1109/TSC.2010.5.

[53] Herbert B. Enderton. *A Mathematical Introduction to Logic*. 2nd ed. San Diego: Harcourt/Academic Press, 2001. ISBN: 978-0-12-238452-3.

[54] P. Erdos, L. Gillman y M. Henriksen. «An Isomorphism Theorem for Real-Closed Fields». En: *Annals of Mathematics* 61.3 (1955), págs. 542-554. ISSN: 0003-486X. DOI: 10.2307/1969812. JSTOR: 1969812.

[55] Michael Evans y T. Swartz. *Approximating Integrals via Monte Carlo and Deterministic Methods*. Oxford Statistical Science Series 20. Oxford ; New York: Oxford University Press, 2000. ISBN: 978-0-19-850278-4.

[56] José Luiz Fiadeiro, Antónia Lopes y Laura Bocchi. «An Abstract Model of Service Discovery and Binding». En: *Formal Aspects of Computing* 23.4 (jul. de 2011), págs. 433-463. ISSN: 1433-299X. DOI: 10.1007/s00165-010-0166-z.

[57] José Luiz Fiadeiro, Ugo Montanari y Martin Wirsing. «05081 Abstracts Collection – Foundations of Global Computing». En: *Foundations of Global Computing*. Ed. por José Luiz Fiadeiro, Ugo Montanari y Martin Wirsing. Vol. 5081. Dagstuhl Seminar Proceedings (DagSemProc). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2006, págs. 1-16. DOI: 10.4230/DagSemProc.05081.1.

[58] Evelyn Fix y J. L. Hodges. «Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties». En: *International Statistical Review* 57.3 (1989), págs. 238-247. ISSN: 0306-7734. DOI: 10.2307/1403797. JSTOR: 1403797.

[59] Howard Foster y col. «WS-Engineer: A Model-Based Approach to Engineering Web Service Compositions and Choreography». En: *Test and Analysis of Web Services*. Ed. por Luciano Baresi y Elisabetta Di Nitto. Berlin, Heidelberg: Springer, 2007, págs. 87-119. ISBN: 978-3-540-72912-9. DOI: 10.1007/978-3-540-72912-9_4.

[60] Adrian Francalanza, Claudio Antares Mezzina y Emilio Tuosto. «Towards Choreographic-Based Monitoring». En: *Reversible Computation: Extending Horizons of Computing - Selected Results of the COST Action IC1405*. Vol. 12070. Lecture Notes in Computer Science. Springer, 2020, págs. 128-150. DOI: 10.1007/978-3-030-47361-7_6.

[61] Leonardo Frittelli y col. «A Choreography-Driven Approach to APIs: The OpenDXL Case Study». En: *Coordination Models and Languages*. Ed. por Simon Bliudze y Laura Bocchi. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, págs. 107-124. ISBN: 978-3-030-50029-0. DOI: 10.1007/978-3-030-50029-0_7.

[62] Cunjing Ge y Feifei Ma. «A Fast and Practical Method to Estimate Volumes of Convex Polytopes». En: *Frontiers in Algorithmics*. Ed. por Jianxin Wang y Chee Yap. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, págs. 52-65. ISBN: 978-3-319-19647-3. DOI: 10.1007/978-3-319-19647-3_6.

[63] Martin Gebser y col. «Clasp: A Conflict-Driven Answer Set Solver». En: *Logic Programming and Nonmonotonic Reasoning, LPNMR 2007*. Ed. por Chitta Baral, Gerhard Brewka y John Schlipf. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, págs. 260-265. ISBN: 978-3-540-72200-7. DOI: 10.1007/978-3-540-72200-7_23.

[64] Gabrile Genovese. *Choreography Automata from Erlang Code*. https://github.com/gabrielegenovese/chorer/blob/master/README.md.

[65] Elena Giachino y col. «Statically and Dynamically Verifiable SLA Metrics». En: *Theory and Practice of Formal Methods: Essays Dedicated to Frank de Boer on the Occasion of His 60th Birthday*. Ed. por Erika Ábrahám, Marcello Bonsangue y Einar Broch Johnsen. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, págs. 211-225. ISBN: 978-3-319-30734-3. DOI: 10.1007/978-3-319-30734-3_15.

[66] Martin Glinz. «On Non-Functional Requirements». En: *15th IEEE International Requirements Engineering Conference (RE 2007)*. Oct. de 2007, págs. 21-26. DOI: 10.1109/RE.2007.45.

[67] Andrew D. Gordon y col. «Probabilistic Programming». En: *Future of Software Engineering Proceedings*. FOSE 2014. New York, NY, USA: Association for Computing Machinery, mayo de 2014, págs. 167-181. ISBN: 978-1-4503-2865-4. DOI: 10.1145/2593882.2593900.

[68] Peter Gritzmann y Victor Klee. «On the Complexity of Some Basic Problems in Computational Convexity». En: *Polytopes: Abstract, Convex and Computational*. Ed. por T. Bisztriczky y col. NATO ASI Series. Dordrecht: Springer Netherlands, 1994, págs. 373-466. ISBN: 978-94-011-0924-6. DOI: 10.1007/978-94-011-0924-6_17.

[69] Branko Grünbaum. *Convex Polytopes*. Ed. por Günter M. Ziegler. 2.ª ed. Graduate Texts in Mathematics. New York: Springer-Verlag, 2003. ISBN: 978-0-387-00424-2. DOI: 10.1007/978-1-4613-0019-9.

[70] Axel Grundmann y H. M. Möller. «Invariant Integration Formulas for the N-Simplex by Combinatorial Methods». En: *SIAM Journal on Numerical Analysis* 15.2 (abr. de 1978), págs. 282-290. ISSN: 0036-1429, 1095-7170. DOI: 10.1137/0715019.

[71] Matthias Güdemann y col. «VerChor: A Framework for the Design and Verification of Choreographies». En: *IEEE Transactions on Services Computing* 9.4 (jul. de 2016), págs. 647-660. ISSN: 1939-1374. DOI: 10.1109/TSC.2015.2413401.

[72] Dick Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. de 2012. DOI: 10.17487/RFC6749.

[73] John Harrison. *Handbook of Practical Logic and Automated Reasoning*. 1.ª ed. Cambridge University Press, mar. de 2009. ISBN: 978-0-521-89957-4 978-0-511-57643-0. DOI: 10.1017/CBO9780511576430.

[74] Vahideh Hayyolalam y Ali Asghar Pourhaji Kazem. «A Systematic Literature Review on QoS-aware Service Composition and Selection in Cloud Environment». En: *Journal of Network and Computer Applications* 110 (mayo de 2018), págs. 52-74. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2018.03.003.

[75] Jesper G. Henriksen y P. S. Thiagarajan. «Dynamic Linear Time Temporal Logic». En: *Annals of Pure and Applied Logic* 96.1 (mar. de 1999), págs. 187-207. ISSN: 0168-0072. DOI: 10.1016/S0168-0072(98)00039-6.

[76] T.A. Henzinger. «The Theory of Hybrid Automata». En: *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*. Jul. de 1996, págs. 278-292. DOI: 10.1109/LICS.1996.561342.

[77] Thomas A. Henzinger y col. «What's Decidable about Hybrid Automata?» En: *Journal of Computer and System Sciences* 57.1 (ago. de 1998), págs. 94-124. ISSN: 0022-0000. DOI: 10.1006/jcss.1998.1581.

[78] Xu Hongzhen y col. «Evolution of Service Composition Based on QoS under the Cloud Computing Environment». En: *2016 IEEE International Conference of Online Analysis and Computing Science (ICOACS)*. Mayo de 2016, págs. 66-69. DOI: 10.1109/ICOACS.2016.7563050.

[79] Hans Hüttel y col. «Foundations of Session Types and Behavioural Contracts». En: *ACM Computing Surveys* 49.1 (abr. de 2016), 3:1-3:36. ISSN: 0360-0300. DOI: 10.1145/2873052.

[80] San-Yih Hwang, Chien-Ching Hsu y Chien-Hsiang Lee. «Service Selection for Web Services with Probabilistic QoS». En: *IEEE Transactions on Services Computing* 8.3 (mayo de 2015), págs. 467-480. ISSN: 1939-1374. DOI: 10.1109/TSC.2014.2338851.

[81] San-Yih Hwang y col. «A Probabilistic Approach to Modeling and Estimating the QoS of Web-Services-Based Workflows». En: *Information Sciences*. Including: Mathematics of Uncertainty 177.23 (dic. de 2007), págs. 5484-5503. ISSN: 0020-0255. DOI: 10.1016/j.ins.2007.07.011.

[82] IBM. *IBM ILOG CPLEX Optimization Studio*. Software. 2004.

[83] «IEEE Standard Glossary of Software Engineering Terminology». En: *IEEE Std 610.12-1990* (dic. de 1990), págs. 1-84. DOI: 10.1109/IEEESTD.1990.101064.

[84] Keigo Imai, Julien Lange y Rumyana Neykova. «Kmclib: Automated Inference and Verification of Session Types from OCaml Programs». En: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. por Dana Fisman y Grigore Rosu. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2022, págs. 379-386. ISBN: 978-3-030-99524-9. DOI: 10.1007/978-3-030-99524-9_20.

[85] Keigo Imai, Shoji Yuen y Kiyoshi Agusa. «Session Type Inference in Haskell». En: *Proceedings Third Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software, PLACES 2010, Paphos, Cyprus, 21st March 2010*. Ed. por Kohei Honda y Alan Mycroft. Vol. 69. EPTCS. 2010, págs. 74-91. DOI: 10.4204/EPTCS.69.6.

[86] Dragan Ivanović, Manuel Carro y Manuel V. Hermenegildo. «A Constraint-Based Approach to Quality Assurance in Service Choreographies». En: *Service-Oriented Computing*. Ed. por Chengfei Liu y col. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, págs. 252-267. ISBN: 978-3-642-34321-6. DOI: 10.1007/978-3-642-34321-6_17.

[87] Nicolai M. Josuttis. *SOA in Practice: The Art of Distributed System Design*. 1st edition. Beijing ; Sebastopol: O'Reilly Media, sep. de 2007. ISBN: 978-0-596-52955-0.

[88] Ravi Kannan, László Lovász y Miklós Simonovits. «Random Walks and an $O^*(n^5)$ Volume Algorithm for Convex Bodies». En: *Random Structures & Algorithms* 11.1 (1997), págs. 1-50. ISSN: 1098-2418. DOI: 10.1002/(SICI)1098-2418(199708)11:1<1::AID-RSA1>3.0.CO;2-X.

[89] Ravindran Kannan y Hariharan Narayanan. «Random Walks on Polytopes and an Affine Interior Point Method for Linear Programming». En: *Mathematics of Operations Research* 37.1 (feb. de 2012), págs. 1-20. ISSN: 0364-765X, 1526-5471. DOI: 10.1287/moor.1110.0519.

[90] Ajay Kattepur, Nikolaos Georgantas y Valérie Issarny. «QoS Analysis in Heterogeneous Choreography Interactions». En: *Service-Oriented Computing*. Ed. por Samik Basu y col. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, págs. 23-38. ISBN: 978-3-642-45005-1. DOI: 10.1007/978-3-642-45005-1_3.

[91] Elias Keis y col. «Automated QoS-Aware Service Selection Based on Soft Constraints». En: *Recent Trends in Algebraic Development Techniques*. Ed. por Alexandre Madeira y Manuel A. Martins. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, págs. 121-140. ISBN: 978-3-031-43345-0. DOI: 10.1007/978-3-031-43345-0_6.

[92] L. G. Khachiyan. «The Problem of Calculating the Volume of a Polyhedron Is Enumerably Hard». En: *Russian Mathematical Surveys* 44.3 (jun. de 1989), pág. 199. ISSN: 0036-0279. DOI: 10.1070/RM1989v044n03ABEH002136.

[93] Adrian Klein, Fuyuki Ishikawa y Bernhard Bauer. «A Probabilistic Approach to Service Selection with Conditional Contracts and Usage Patterns». En: *Service-Oriented Computing*. Ed. por Luciano Baresi, Chi-Hung Chi y Jun Suzuki. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, págs. 253-268. ISBN: 978-3-642-10383-4. DOI: 10.1007/978-3-642-10383-4_17.

[94] Petar Kochovski, Pavel D. Drobintsev y Vlado Stankovski. «Formal Quality of Service Assurances, Ranking and Verification of Cloud Deployment Options with a Probabilistic Model Checking Method». En: *Information and Software Technology* 109 (mayo de 2019), págs. 14-25. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2019.01.003.

[95] Jörg Kreiker y col. «Modeling, Analysis, and Verification - The Formal Methods Manifesto 2010 (Dagstuhl Perspectives Workshop 10482)». En: *Dagstuhl Manifestos* 1.1 (2011), págs. 21-40. DOI: 10.4230/DAGMAN.1.1.21.

[96] Marta Kwiatkowska. «Quantitative Verification: Models, Techniques and Tools». En: *The 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering: Companion Papers*. ESEC-FSE Companion '07. New York, NY, USA: Association for Computing Machinery, sep. de 2007, págs. 449-458. ISBN: 978-1-59593-812-1. DOI: 10.1145/1295014.1295018.

[97] Marta Kwiatkowska, Gethin Norman y David Parker. «PRISM 4.0: Verification of Probabilistic Real-Time Systems». En: *Computer Aided Verification*. Ed. por Ganesh Gopalakrishnan y Shaz Qadeer. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, págs. 585-591. ISBN: 978-3-642-22110-1. DOI: 10.1007/978-3-642-22110-1_47.

[98] Julien Lange, Emilio Tuosto y Nobuko Yoshida. «A Tool for Choreography-Based Analysis of Message-Passing Software». En: *Behavioural Types: From Theory to Tools English*. River Publishers, jul. de 2017, págs. 125-146.

[99] Kim G. Larsen, Paul Pettersson y Wang Yi. «Uppaal in a Nutshell». En: *International Journal on Software Tools for Technology Transfer* 1.1 (dic. de 1997), págs. 134-152. ISSN: 1433-2779. DOI: 10.1007/s100090050010.

[100] Daniel Le Berre y Anne Parrain. «The Sat4j Library, Release 2.2: System Description». En: *Journal on Satisfiability, Boolean Modeling and Computation* 7.2-3 (jul. de 2010), págs. 59-64. ISSN: 15740617. DOI: 10.3233/SAT190075.

[101] B. H. Liskov. «A Design Methodology for Reliable Software Systems». En: *Proceedings of the December 5-7, 1972, Fall Joint Computer Conference, Part I*. AFIPS '72 (Fall, Part I). New York, NY, USA: Association for Computing Machinery, dic. de 1972, págs. 191-199. ISBN: 978-1-4503-7912-0. DOI: 10.1145/1479992.1480018.

[102] Barbara Liskov y Stephen Zilles. «Programming with Abstract Data Types». En: *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*. New York, NY, USA: Association for Computing Machinery, mar. de 1974, págs. 50-59. ISBN: 978-1-4503-7884-0. DOI: 10.1145/800233.807045.

[103] Chung L. Liu. *Introduction to Combinatorial Mathematics*. Computer Science Series. New York, NY: McGraw-Hill, 1968. ISBN: 978-0-07-038124-7.

[104] Yutu Liu, Anne H. Ngu y Liang Z. Zeng. «QoS Computation and Policing in Dynamic Web Service Selection». En: *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters*. WWW Alt. '04. New York, NY, USA: Association for Computing Machinery, mayo de 2004, págs. 66-73. ISBN: 978-1-58113-912-9. DOI: 10.1145/1013367.1013379.

[105] Alberto Lluch-Lafuente y Ugo Montanari. «Quantitative μ-Calculus and CTL Based on Constraint Semirings». En: *Electronic Notes in Theoretical Computer Science*. Proceedings of the Second Workshop on Quantitative Aspects of Programming Languages (QAPL 2004) 112 (ene. de 2005), págs. 37-59. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2004.02.063.

[106] Carlos G. Lopez Pombo, Agustín E. Martinez Suñé y Emilio Tuosto. «A Dynamic Temporal Logic for Quality of Service in Choreographic Models». En: *Theoretical Aspects of Computing – ICTAC 2023*. Ed. por Erika Ábrahám, Clemens Dubslaff y Silvia Lizeth Tapia Tarifa. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2023, págs. 119-138. ISBN: 978-3-031-47963-2. DOI: 10.1007/978-3-031-47963-2_9.

[107] Carlos G. Lopez Pombo, Agustín E. Martinez Suñé y Emilio Tuosto. *MoCheQoS: Automated Analysis of Quality of Service Properties of Communicating Systems*. Nov. de 2023. DOI: 10.48550/arXiv.2311.01415. arXiv: 2311.01415 [cs].

[108] Chengying Mao y col. «Search-Based QoS Ranking Prediction for Web Services in Cloud Environments». En: *Future Generation Computer Systems*. Quality of Service in Grid and Cloud 2015 50 (sep. de 2015), págs. 111-126. ISSN: 0167-739X. DOI: 10.1016/j.future.2015.01.008.

[109] Octavio Martín-Díaz y col. «Automating the Procurement of Web Services». En: *Service-Oriented Computing - ICSOC 2003*. Ed. por Maria E. Orlowska y col. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003, págs. 91-103. ISBN: 978-3-540-24593-3. DOI: 10.1007/978-3-540-24593-3_7.

[110] Agustín E. Martinez Suñé y Carlos G Lopez Pombo. *Probabilistic Quality of Service Aware Service Selection*. Mayo de 2022. DOI: 10.48550/arXiv.2205.09877. arXiv: 2205.09877 [cs].

[111] Agustín E. Martinez Suñé y Carlos G. Lopez Pombo. «Automatic Quality-of-Service Evaluation in Service-Oriented Computing». En: *Coordination Models and Languages*. Ed. por Hanne Riis Nielson y Emilio Tuosto. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, págs. 221-236. ISBN: 978-3-030-22397-7. DOI: 10.1007/978-3-030-22397-7_13.

[112] Agustín Eloy Martinez Suñé y Carlos Gustavo Lopez Pombo. «Quality of Service Ranking by Quantifying Partial Compliance of Requirements». En: *Coordination Models and Languages*. Ed. por Simon Bliudze y Laura Bocchi. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, págs. 181-189. ISBN: 978-3-030-50029-0. DOI: 10.1007/978-3-030-50029-0_12.

[113] Eyhab Al-Masri y Qusay H. Mahmoud. «QoS-based Discovery and Ranking of Web Services». En: *2007 16th International Conference on Computer Communications and Networks*. Ago. de 2007, págs. 529-534. DOI: 10.1109/ICCCN.2007.4317873.

[114] Thomas J. McCabe. «A Complexity Measure». En: *Proceedings of the 2nd International Conference on Software Engineering*. ICSE '76. Washington, DC, USA: IEEE Computer Society Press, oct. de 1976, pág. 407.

[115] Pedro Meseguer, Francesca Rossi y Thomas Schiex. «Soft Constraints». En: *Foundations of Artificial Intelligence*. Ed. por Francesca Rossi, Peter van Beek y Toby Walsh. Vol. 2. Handbook of Constraint Programming. Elsevier, ene. de 2006, págs. 281-328. DOI: 10.1016/S1574-6526(06)80013-1.

[116] Michael Shamos. «Computational Geometry». Tesis doct. Yale University, 1978.

[117] Chris Miller. *Chip War: The Fight for the World's Most Critical Technology*. London: Simon & Schuster, 2022. ISBN: 978-1-398-50410-3 978-1-398-50409-7.

[118] Mahboobeh Moghaddam y Joseph G. Davis. «Service Selection in Web Service Composition: A Comparative Review of Existing Approaches». En: *Web Services Foundations*. Ed. por Athman Bouguettaya, Quan Z. Sheng y Florian Daniel. New York, NY: Springer, 2014, págs. 321-346. ISBN: 978-1-4614-7518-7. DOI: 10.1007/978-1-4614-7518-7_13.

[119] Leonard Gaines Monk. *An elementary-recursive decision procedure for* $\mathsf{Th}(\mathbb{R}, +, .)$. Manuscript. 1974.

[120] Leonard Gaines Monk. «Elementary Recursive Decision Procedures». Tesis doct. University of California, Berkeley, 1975.

[121] Gordon E. Moore. «Lithography and the Future of Moore's Law». En: *IEEE Solid-State Circuits Society Newsletter* 11.3 (sep. de 2006), págs. 37-42. ISSN: 1098-4232. DOI: 10.1109/N-SSC.2006.4785861.

[122] P. Naur y B. Randell, eds. *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th October 1968*. Brussels: Scientific Affairs Division, NATO, 1969.

[123] Pierluigi Nuzzo y col. «CalCS: SMT Solving for Non-Linear Convex Constraints». En: *Formal Methods in Computer Aided Design*. Oct. de 2010, págs. 71-79.

[124] Justin O'Sullivan, David Edmond y Arthur ter Hofstede. «What's in a Service?» En: *Distributed and Parallel Databases* 12.2 (sep. de 2002), págs. 117-133. ISSN: 1573-7578. DOI: 10.1023/A:1016547000822.

[125] Maria E. Orlowska y col., eds. *Service-Oriented Computing - ICSOC 2003: First International Conference, Trento, Italy, December 15-18, 2003. Proceedings*. Vol. 2910. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2003. ISBN: 978-3-540-20681-1 978-3-540-24593-3. DOI: 10.1007/b94513.

[126] Amir Pnueli. «The Temporal Semantics of Concurrent Programs». En: *Theoretical Computer Science*. Special Issue Semantics of Concurrent Computation 13.1 (ene. de 1981), págs. 45-60. ISSN: 0304-3975. DOI: 10.1016/0304-3975(81)90110-9.

[127] Klaus Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*. 1st. Springer Publishing Company, Incorporated, 2010. ISBN: 978-3-642-12577-5.

[128] *Post Office Protocol: Version 2*. RFC 937. Feb. de 1985. DOI: 10.17487/RFC0937.

[129] Vaughan Pratt. «Modeling Concurrency with Partial Orders». En: *International Journal of Parallel Programming* 15.1 (feb. de 1986), págs. 33-71. ISSN: 1573-7640. DOI: 10.1007/BF01379149.

[130] Vaughan R. Pratt. «SEMANTICAL CONSIDERATIONS ON FLOYD-HOARE LOGIC». En: *17th Annual Symposium on Foundations of Computer Science (Sfcs 1976)*. Oct. de 1976, págs. 109-121. DOI: 10.1109/SFCS.1976.27.

[131] Mark Priestley. *A Science of Operations: Machines, Logic and the Invention of Programming*. History of Computing. New York ; London: Springer, 2010. ISBN: 978-1-84882-554-3 978-1-84882-555-0.

[132] Sidney Rosario, Albert Benveniste y Claude Jard. «Flexible Probabilistic QoS Management of Transaction Based Web Services Orchestrations». En: *2009 IEEE International Conference on Web Services*. Jul. de 2009, págs. 107-114. DOI: 10.1109/ICWS.2009.38.

[133] Murray Rosenblatt. «Remarks on Some Nonparametric Estimates of a Density Function». En: *The Annals of Mathematical Statistics* 27.3 (sep. de 1956), págs. 832-837. ISSN: 0003-4851, 2168-8990. DOI: 10.1214/aoms/1177728190.

[134] K. Rosenthal. *Quantales and Their Applications*. Vol. 234. Pitman Research Notes in Mathematics Series. Longman Scientific & Technical, 1990.

[135] Stephan R. Sain, Keith A. Baggerly y David W. Scott. «Cross-Validation of Multivariate Densities». En: *Journal of the American Statistical Association* 89.427 (1994), págs. 807-817. ISSN: 0162-1459. DOI: 10.2307/2290906. JSTOR: 2290906.

[136] Yasser Shoukry y col. *SatEX: A Satisfiability Modulo Convex programming (SMC) solver*. https://yshoukry.bitbucket.io/SatEX.

[137] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. 1.ª ed. Wiley Series in Probability and Statistics. Wiley, ago. de 1992. ISBN: 978-0-471-54770-9 978-0-470-31684-9. DOI: 10.1002/9780470316849.

[138] Yasser Shoukry y col. «SMC: Satisfiability Modulo Convex Optimization». En: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*. HSCC '17. New York, NY, USA: Association for Computing Machinery, abr. de 2017, págs. 19-28. ISBN: 978-1-4503-4590-3. DOI: 10.1145/3049797.3049819.

[139] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability 26. Boca Raton: Chapman & Hall/CRC, 1986. ISBN: 978-0-412-24620-3.

[140] Alfred Tarski. «A Decision Method for Elementary Algebra and Geometry». En: *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Ed. por Bob F. Caviness y Jeremy R. Johnson. Texts and Monographs in Symbolic Computation. Vienna: Springer, 1998, págs. 24-84. ISBN: 978-3-7091-9459-1. DOI: 10.1007/978-3-7091-9459-1_3.

[141] Javier Troya y col., eds. *Service-Oriented Computing: 20th International Conference, ICSOC 2022, Seville, Spain, November 29 – December 2, 2022, Proceedings*. Vol. 13740. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2022. ISBN: 978-3-031-20983-3 978-3-031-20984-0. DOI: 10.1007/978-3-031-20984-0.

[142] G. S. Tseitin. «On the Complexity of Derivation in Propositional Calculus». En: *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*. Ed. por Jörg H. Siekmann y Graham Wrightson. Symbolic Computation. Berlin, Heidelberg: Springer, 1983, págs. 466-483. ISBN: 978-3-642-81955-1. DOI: 10.1007/978-3-642-81955-1_28.

[143] Emilio Tuosto y Roberto Guanciale. «Semantics of Global View of Choreographies». En: *Journal of Logical and Algebraic Methods in Programming* 95 (feb. de 2018), págs. 17-40. ISSN: 2352-2208. DOI: 10.1016/j.jlamp.2017.11.002.

[144] L. G. Valiant. «The Complexity of Computing the Permanent». En: *Theoretical Computer Science* 8.2 (ene. de 1979), págs. 189-201. ISSN: 0304-3975. DOI: 10.1016/0304-3975(79)90044-6.

[145] Paul van Dooren y Luc de Ridder. «An Adaptive Algorithm for Numerical Integration over an N-Dimensional Cube». En: *Journal of Computational and Applied Mathematics* 2.3 (sep. de 1976), págs. 207-217. ISSN: 0377-0427. DOI: 10.1016/0771-050X(76)90005-X.

[146] Moshe Y. Vardi. «The Rise and Fall of LTL: Invited Presentation at the Second International Symposium on Games, Automata, Logics and Formal Verification». En: *Electronic Proceedings in Theoretical Computer Science* 54 (jun. de 2011). ISSN: 2075-2180. DOI: 10.4204/EPTCS.54.0.2.

[147] Pauli Virtanen y col. «SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python». En: *Nature Methods* 17.3 (mar. de 2020), págs. 261-272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2.

[148] Ignacio Vissani, Carlos Gustavo Lopez Pombo y Emilio Tuosto. «Communicating Machines as a Dynamic Binding Mechanism of Services». En: *Electronic Proceedings in Theoretical Computer Science* 203 (feb. de 2016), págs. 85-98. ISSN: 2075-2180. DOI: 10.4204/EPTCS.203.7. arXiv: 1602.03597.

[149] S. Wang y col. «Cloud Model for Service Selection». En: *2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011*. 2011, págs. 666-671. ISBN: 978-1-4577-0248-8. DOI: 10.1109/INFCOMW.2011.5928896.

[150] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. 2.ª ed. Berlin Heidelberg: Springer-Verlag, 2012. ISBN: 978-3-642-28615-5. DOI: 10.1007/978-3-642-28616-2.

[151] Martin Wirsing y Matthias Hölzl, eds. *Rigorous Software Engineering for Service-Oriented Systems: Results of the SENSORIA Project on Software Engineering for Service-Oriented Computing*. Vol. 6582. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011. ISBN: 978-3-642-20400-5 978-3-642-20401-2. DOI: 10.1007/978-3-642-20401-2.

[152] Stephen S. Yau y Yin Yin. «QoS-Based Service Ranking and Selection for Service-Based Systems». En: *2011 IEEE International Conference on Services Computing*. Jul. de 2011, págs. 56-63. DOI: 10.1109/SCC.2011.114.

[153] Liangzhao Zeng y col. «QoS-aware Middleware for Web Services Composition». En: *IEEE Transactions on Software Engineering* 30.5 (mayo de 2004), págs. 311-327. ISSN: 1939-3520. DOI: 10.1109/TSE.2004.11.

[154] Huiyuan Zheng, Jian Yang y Weiliang Zhao. «Probabilistic QoS Aggregations for Service Composition». En: *ACM Transactions on the Web* 10.2 (mayo de 2016), 12:1-12:36. ISSN: 1559-1131. DOI: 10.1145/2876513.

[155] Huiyuan Zheng, Jian Yang y Weiliang Zhao. «QoS Probability Distribution Estimation for Web Services and Service Compositions». En: *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Dic. de 2010, págs. 1-8. DOI: 10.1109/SOCA.2010.5707144.

[156] Zibin Zheng y col. «QoS Ranking Prediction for Cloud Services». En: *IEEE Transactions on Parallel and Distributed Systems* 24.6 (jun. de 2013), págs. 1213-1222. ISSN: 1558-2183. DOI: 10.1109/TPDS.2012.285.

[157] Günter M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. New York: Springer-Verlag, 1995. ISBN: 978-0-387-94329-9. DOI: 10.1007/978-1-4613-8431-1.

[158] Farhana H. Zulkernine y Patrick Martin. «An Adaptive and Intelligent SLA Negotiation System for Web Services». En: *IEEE Transactions on Services Computing* 4.1 (ene. de 2011), págs. 31-43. ISSN: 1939-1374. DOI: 10.1109/TSC.2010.44.

Apéndices

A. Cuerpos real-cerrados

Sea Σ_{field} la signatura $\langle\{0,1\},\{+,\cdot\},\{=\}\rangle$. Un cuerpo ordenado es una presentación de teoría de primer orden $\langle\Sigma_{\text{field}},\Gamma_{\text{field}}\rangle$ donde Γ_{field} son los axiomas del cuerpo.

Definición 24 (Cuerpo ordenado – [14](Sec. 2.1))**.** *Sea* $\Sigma_{\text{field}} = \langle\{0,1\},\{+,\cdot\},\{=\}\rangle$ *y* Γ_{field} *el conjunto que contiene las siguientes fórmulas:*

(asociatividad)	$(\forall x,y,z)(x\cdot(y\cdot z)=(x\cdot y)\cdot z \wedge x+(y+z)=(x+y)+z)$
(conmutatividad)	$(\forall x,y)(x\cdot y=y\cdot x \wedge x+y=y+x)$
(identidad)	$(\forall x)(x\cdot 1=x \wedge x+0=x)$
(distributividad)	$(\forall x,y)(\forall z)x\cdot(y+z)=(x\cdot y)+(x\cdot z)$
(inversa de $\cdot$)	$(\forall x)(x=0 \vee ((\exists y)x\cdot y=1))$
(inversa de $+$)	$(\forall x)((\exists y)x+y=0)$

Entonces, cualquier modelo de la presentación de teoría (de primer orden) $\langle\Sigma_{\text{field}},\Gamma_{\text{field}}\rangle$ *se dice que es un* cuerpo*. Un* cuerpo ordenado *es un cuerpo con un orden total estricto* [1]*.*

Los cuerpos real-cerrados son cuerpos ordenados cuyos subconjuntos no vacíos tienen un supremo.

Definición 25 (Real-closed field – [14](Sec. 4.3))**.** *Dado un cuerpo ordenado con conjunto portador S, su orden $\leq\subseteq S\times S$ es* Dedekind-completo *si todo subconjunto no vacío $S'\subseteq S$ con una cota superior en S tiene un supremo (también llamado límite superior) en S.*

Cuando $\leq$ es Dedekind-completo, el cuerpo ordenado se dice que es el cuerpo real-cerrado *(o un* cuerpo ordenado Dedekind-completo*).*

Intuitivamente, en palabras de Dedekind, la completitud implica que no hay ningún "hueco" en la recta real, lo que obliga al conjunto a ser continuo.

Teorema 3 (Isomorfismo de cuerpos real-cerrados [54](Thm. 2.1))**.** *Bajo la hipótesis del continuo, todos los cuerpos hiperreales $\aleph_1$ son isomorfos.*

Corolario 1. *Dado cualquier par de cuerpos ordenados Dedekind-completos con conjuntos portadores S y S' respectivamente, existe un único isomorfismo de cuerpos, que preserva el orden, de S a S'.*

[1] Sea S un conjunto, entonces $\leq\subseteq S\times S$ es un *orden total* si satisface las siguientes condiciones: *a*) Reflexividad: para todo $a\in S$, $a=a$, *b*) Antisimetría: para todo $a,b,c\in S$, si $a\leq b$ y $b\leq a$, entonces $a=b$, *c*) Transitividad: para todo $a,b,c\in S$, si $a\leq b$ y $b\leq c$, entonces $a\leq c$, y *d*) Totalidad: para todo $a,b\in S$, $a\leq b$ o $b\leq a$. Además, dado $\leq\subseteq S\times S$ un orden total, su *orden estricto asociado* $<\subseteq S\times S$ se define mediante una de las siguientes condiciones equivalentes: *a*) Reducción reflexiva: para todo $a,b\in S$, si $a\leq b$ y $\neg a=b$, entonces $a<b$, o *b*) $<$ es el complemento de la conversa de $\leq$: para todo $a,b\in S$, si $\neg a\leq b$, entonces $a<b$.

En [140], Tarski introdujo lo que él llamó *el sistema de álgebra elemental*, su formalización de los números reales, como un cuerpo ordenado (ver [140, Nota 9]) que también satisface:

a) un axioma que establece que cada número positivo tiene una raíz cuadrada, y

b) un esquema de axioma que establece que cada polinomio de grado impar tiene, al menos, una raíz real,

lo que resulta en una teoría de primer orden que puede demostrarse que es un cuerpo ordenado Dedekind-completo [140, Nota 15].

Según Corolario 1, el sistema de álgebra elemental de Tarski es isomorfo al cuerpo real-cerrado (ver Definición 25). Además, Tarski demuestra que su sistema admite eliminación de cuantificadores [140, Coro. 32] y, en consecuencia, la existencia de un procedimiento de decisión [140, Thm. 37]. El procedimiento de decisión de Tarski tiene una complejidad no elemental, lo que significa que dada un problema de tamaño n, su complejidad no está acotada por ninguna función de la forma $2^{2^{\cdots n}}$. En [119], Monk proporcionó un algoritmo de complejidad elemental para la eliminación de cuantificadores. En [37, Corolario del Teorema 1], Davenport y Heintz demostraron que la eliminación de cuantificadores para cuerpos real-cerrados tiene una complejidad doblemente exponencial en el tamaño del problema. Algoritmos con tal complejidad fueron proporcionados por Collins en [33] y por Monk en [120], de forma independiente. Posteriormente, en [11], Ben-Or et. al propusieron un algoritmo diferente que se ejecuta en espacio exponencial o en tiempo exponencial paralelo en el tamaño del problema.

B. Semántica de G-coreografías basada en pomsets

Introducción intuitiva

La semántica de las g-coreografías ha sido definida en [143] en términos de *pomsets* [129] etiquetados sobre el conjunto de *acciones de comunicación* $\mathcal{L} = \mathcal{L}^! \cup \mathcal{L}^?$. Básicamente, la semántica $[\![\mathsf{G}]\!]$ de una g-coreografía G es un conjunto de pomsets que representan las relaciones causales entre los eventos de comunicación en G. Por ejemplo, la semántica de una interacción es simplemente un orden parcial en el que el evento correspondiente a la salida precede al evento correspondiente a la entrada, mientras que la semántica de una composición paralela $\mathsf{G} \mid \mathsf{G}'$ se obtiene tomando la unión disjunta de cada pomset en $[\![\mathsf{G}]\!]$ y cada uno en $[\![\mathsf{G}']\!]$. Es una observación directa que la semántica de las g-coreografías libres de $*$ es una unión disjunta finita de pomsets finitos. Ilustramos esta intuición en el siguiente ejemplo.

Ejemplo 12 (Un fragmento de la semántica de POP)**.** *La semántica de* $\mathsf{G}_{\mathrm{quit}}$ *en el Ejemplo 4 y de* $\mathsf{G} = \mathsf{C}{\to}\mathsf{S}\colon \mathsf{helo}; \mathsf{S}{\to}\mathsf{C}\colon \mathsf{int}$ *(que representa una posible ejecución inicial de POP) son los conjuntos unitarios*

$$[\![\mathsf{G}_{\mathrm{quit}}]\!] = \left\{ \left[\begin{array}{ccc} \mathsf{C}\,\mathsf{S}!\mathsf{quit} & \longrightarrow & \mathsf{C}\,\mathsf{S}?\mathsf{quit} \\ \downarrow & & \downarrow \\ \mathsf{S}\,\mathsf{C}?\mathsf{bye} & \longleftarrow & \mathsf{S}\,\mathsf{C}!\mathsf{bye} \end{array} \right] \right\} \qquad y \qquad [\![\mathsf{G}]\!] = \left\{ \left[\begin{array}{ccc} \mathsf{C}\,\mathsf{S}!\mathsf{helo} & \longrightarrow & \mathsf{C}\,\mathsf{S}?\mathsf{helo} \\ \downarrow & & \downarrow \\ \mathsf{S}\,\mathsf{C}?\mathsf{int} & \longleftarrow & \mathsf{S}\,\mathsf{C}!\mathsf{int} \end{array} \right] \right\}$$

donde representamos los pomsets como (una variante de los) diagramas de Hasse de la relación de predecesor inmediato. De acuerdo con esto, tenemos que $[\![\mathsf{G}_{\mathrm{quit}} + \mathsf{G}]\!] = [\![\mathsf{G}_{\mathrm{quit}}]\!] \cup [\![\mathsf{G}]\!]$ *mientras que*

$$[\![\mathsf{G}_{\mathrm{quit}} \mid \mathsf{G}]\!] = \left\{ \left[\begin{array}{ccc} \mathsf{C}\,\mathsf{S}!\mathsf{helo} & \longrightarrow & \mathsf{C}\,\mathsf{S}?\mathsf{helo} \\ \downarrow & & \downarrow \\ \mathsf{S}\,\mathsf{C}?\mathsf{int} & \longleftarrow & \mathsf{S}\,\mathsf{C}!\mathsf{int} \end{array} \quad \begin{array}{ccc} \mathsf{C}\,\mathsf{S}!\mathsf{quit} & \longrightarrow & \mathsf{C}\,\mathsf{S}?\mathsf{quit} \\ \downarrow & & \downarrow \\ \mathsf{S}\,\mathsf{C}?\mathsf{bye} & \longleftarrow & \mathsf{S}\,\mathsf{C}!\mathsf{bye} \end{array} \right] \right\}$$

formalizando que las comunicaciones en $\mathsf{G}_{\mathrm{quit}}$ *y las de* G *pueden ocurrir en cualquier orden ya que no están relacionadas por ninguna dependencia causal.* ◇

Las *linearizaciones* de un pomset r son las secuencias de acciones de comunicación obtenidas mediante una permutación de los eventos de r que preserva la relación de causalidad de r. Por ejemplo, la única linearización posible del pomset en $[\![\mathsf{G}_{\mathrm{quit}}]\!]$ (cf. Ejemplo 12) es la secuencia $\mathsf{C}\,\mathsf{S}!\mathsf{quit}\ \mathsf{C}\,\mathsf{S}?\mathsf{quit}\ \mathsf{S}\,\mathsf{C}!\mathsf{bye}\ \mathsf{S}\,\mathsf{C}?\mathsf{bye}$, mientras que $\mathsf{C}\,\mathsf{S}!\mathsf{quit}\ \mathsf{S}\,\mathsf{C}!\mathsf{bye}\ \mathsf{C}\,\mathsf{S}?\mathsf{quit}\ \mathsf{S}\,\mathsf{C}?\mathsf{bye}$ no es una linearización porque viola la dependencia causal entre $\mathsf{S}\,\mathsf{C}!\mathsf{bye}$ y $\mathsf{C}\,\mathsf{S}?\mathsf{quit}$. El *lenguaje* de G es el conjunto $\mathcal{L}[\mathsf{G}]$ de secuencias de linearizaciones de pomsets en $[\![\mathsf{G}]\!]$.

Definición formal

Definición 26. *Un conjunto parcialmente ordenado etiquetado (lposet) es una terna* $(\mathcal{E}, \leq, \lambda)$*, donde* $\mathcal{E}$ *es un conjunto de eventos,* $\leq \subseteq \mathcal{E} \times \mathcal{E}$ *es una relación reflexiva, antisimétrica y transitiva en* $\mathcal{E}$*, y* $\lambda : \mathcal{E} \to \mathcal{L}$ *es una función de etiquetado.*

Para $e \neq e'$, $\lambda(e) = \lambda(e')$ significa que e y e' representan diferentes ocurrencias de la misma acción. Intuitivamente, $\leq$ representa causalidad; para $e \neq e'$, si $e \leq e'$ y ambos eventos ocurren, entonces e' es causado por e. En lo siguiente, usamos ϵ para denotar el lposet vacío y $e \to e'$ to denote that $e \leq e'$.

Definición 27. *Dos lposets* $(\mathcal{E}, \leq, \lambda)$ *y* $(\mathcal{E}', \leq', \lambda')$ *son isomorfos si y solo si existe una biyección* $\phi : \mathcal{E} \to \mathcal{E}'$ *tal que* $e \leq e'$ *si y solo si* $\phi(e) \leq' \phi(e')$ *y* $\lambda = \lambda' \circ \phi$.

Definición 28. *Un multi-conjunto parcialmente ordenado (de acciones), llamado pomset, es una clase de isomorfismos de lposets.*

En lo siguiente, $[\mathcal{E}, \leq, \lambda]$ denota la clase de isomorfismo de $(\mathcal{E}, \leq, \lambda)$, dejamos que $r, r', \ldots$ (resp. $R, R', \ldots$) varíen entre pomsets (resp. conjuntos de pomsets), y asumimos que cualquier r contiene al menos un lposet que se referira como $(\mathcal{E}_r, \leq_r, \lambda_r)$.

En lo siguiente, dado un número natural n, $\mathbf{n}$ representa el conjunto unitario $\{n\}$. Además, usamos $X \uplus Y$ para representar la unión disjunta de dos conjuntos X e Y: $X \uplus Y = (X \times \mathbf{1}) \cup (Y \times \mathbf{2})$. Finalmente, dado una función f en X, definimos $f \otimes \mathbf{n} = \{(x, n) \mapsto f(x) \mid x \in X\}$ como la función que extiende f a $X \times \mathbf{n}$; de manera análoga, para una relación $R \subseteq X \times Y$, definimos $R \otimes \mathbf{n} = \{((x, n), (y, n)) \mid (x, y) \in R\}$ como la relación que extiende R a $(X \times \mathbf{n}) \times (Y \times \mathbf{n})$.

Ahora definimos dos construcciones importantes para componer pomsets en paralelo y secuencialmente.

Definición 29. *Sea* $r = [\mathcal{E}, \leq, \lambda]$ *y* $r' = [\mathcal{E}', \leq', \lambda']$ *dos pomsets. La* composición en paralelo de r y r' *es:*

$$\mathsf{par}(r, r') \quad = \quad [\mathcal{E} \uplus \mathcal{E}',\ (\leq \otimes \mathbf{1}) \cup (\leq' \otimes \mathbf{2}),\ (\lambda \otimes \mathbf{1}) \cup (\lambda' \otimes \mathbf{2})]$$

Para un pomset r *y un participante* $\mathsf{A} \in \mathcal{P}$, *sea* $\mathcal{E}_{r,\mathsf{A}} = \{e \in \mathcal{E}_r \mid \mathit{sbj}(\lambda_r(e)) = \mathsf{A}\}$ *sea el conjunto de eventos de* A *en* $\mathcal{E}_r$. *La* composición secuencial de r y r' *es:*

$$\mathsf{seq}(r, r') \quad = \quad [\mathcal{E} \uplus \mathcal{E}',\ \leq_{seq},\ (\lambda \otimes \mathbf{1}) \cup (\lambda' \otimes \mathbf{2})]$$

donde

$$\begin{aligned}
\mathcal{E}_{r,\mathsf{A}} &= \{e \in \mathcal{E}_r \mid \mathit{sbj}(\lambda_r(e)) = \mathsf{A}\} \\
\leq_{seq} &= \bigcup_{\mathsf{A} \in \mathcal{P}} \left((\mathcal{E}_{r,\mathsf{A}} \times \mathbf{1}) \times \left\{(e, 2) \in \mathcal{E}_{r',\mathsf{A}} \times \mathbf{2} \mid \lambda'(e) \in \mathcal{L}^!\right\}\right)^\star \cup (\leq \otimes \mathbf{1}) \cup (\leq' \otimes \mathbf{2})
\end{aligned}$$

y $\star$ *es la clausura reflexiva-transitiva.*

Tanto la composición en paralelo como la composición secuencial preservan las dependencias causales de sus componentes $\leq$ y $\leq'$. Sin embargo, mientras que la composición en paralelo no introduce nuevas dependencias, la composición secuencial de dos pomsets agrega a $\leq$ y $\leq'$ las dependencias entre eventos en r y r' con el mismo sujeto. Básicamente, se induce una relación causal cuando un participante que realiza una comunicación en r también realiza una comunicación en r'.

Para este trabajo, nos restringimos a g-coreografías bien formadas [143] para las cuales la semántica es:

Definición 30 (Semántica de g-coreografía)**.** *La semántica de las g-coreografías es*

$$\begin{aligned}
[\![\mathbf{0}]\!] &= \{\epsilon\} \\
[\![\mathsf{A} \to \mathsf{B} : \mathsf{m}]\!] &= \{[(\{e_1, e_2\}, \{(e_1, e_1), (e_2, e_2), (e_1, e_2)\}, \lambda)]\} \text{ donde } \lambda : \begin{cases} e_1 \mapsto \mathsf{A}\,\mathsf{B}!\mathsf{m} \\ e_2 \mapsto \mathsf{A}\,\mathsf{B}?\mathsf{m} \end{cases} \\
[\![\mathsf{G} \mid \mathsf{G}']\!] &= \{\mathsf{par}(r, r') \mid (r, r') \in [\![\mathsf{G}]\!] \times [\![\mathsf{G}']\!]\} \\
[\![\mathsf{G}; \mathsf{G}']\!] &= \{\mathsf{seq}(r, r') \mid (r, r') \in [\![\mathsf{G}]\!] \times [\![\mathsf{G}']\!]\} \\
[\![\mathsf{G} + \mathsf{G}']\!] &= [\![\mathsf{G}]\!] \cup [\![\mathsf{G}']\!]
\end{aligned}$$

C. Evaluación experimental de MoCheQoS

C.1. Unfoldings de ciclos

En la Figura .1 mostramos el archivo `.qosfsa` para el protocolo POP de tres participantes utilizado en el caso de estudio de unfoldings de ciclos; la sección `fsa` del archivo se omite por legibilidad. En las Figuras .2 y .3 mostramos los CFSMs y el sistema de transición, respectivamente.

```
fsa{
    ...
}

qos_attributes{
    att1:+,
    att2:+,
    att3:+,
    att4:+,
    att5:+
}

qos_specifications{
    C@0 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@1 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@2 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@3 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@4 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@5 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@6 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@7 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@8 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@9 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@10 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@11 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@12 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@13 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    C@14 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),

    S@0 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@1 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@2 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@3 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@4 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@5 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@6 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@7 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@8 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@9 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@10 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    S@11 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),

    A@0 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    A@1 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5)),
    A@2 : (and (= att1 1) (= att2 2) (= att3 3) (= att4 4) (= att5 5))
}

final_states{
    C: [11, 12, 9],
    S: [8, 9],
    A: [2, 3]
}
```

Fig. .1: `.qosfsa` file for the loop unfoldings case study.

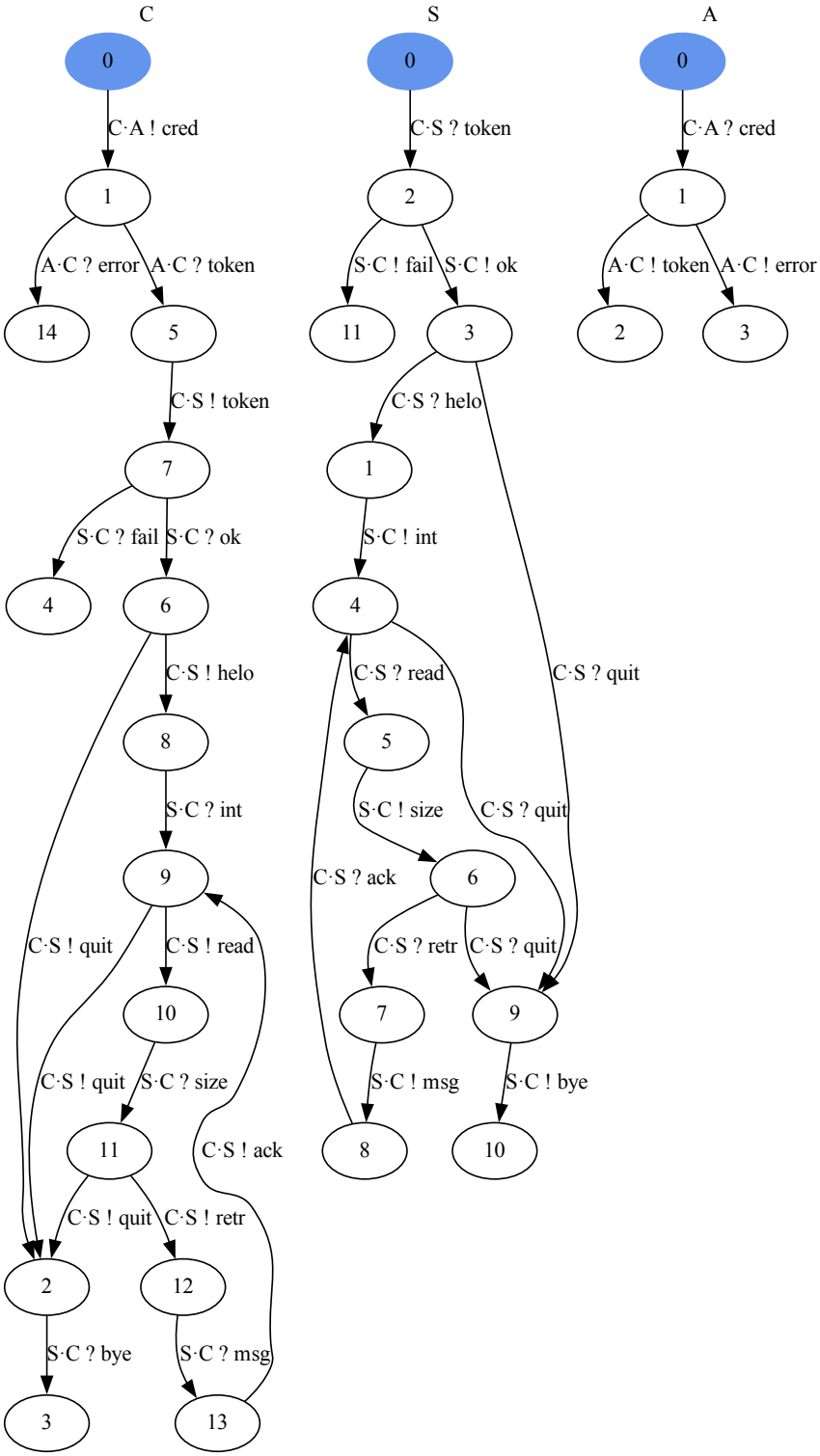


Fig. .2: CFSMs del caso de estuio de unfoldings de ciclos.

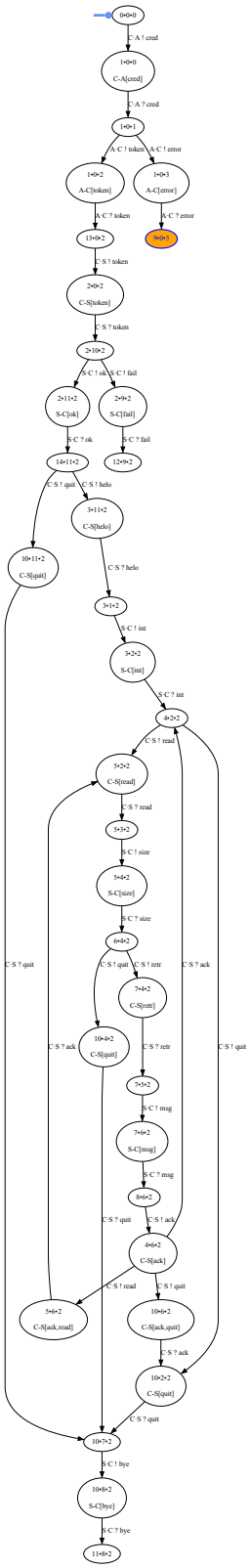


Fig. .3: Sistema de transición del caso de estuio de unfoldings de ciclos.

C.2. Elecciones anidadas

En la Figura .4 mostramos el archivo completo `.qosgc` para el caso de estudio con número de elecciones anidadas = 2. Este formato de archivo es una extensión del formato de archivo `.gc`, que es una versión simplificada de las expresiones regulares de g-coreografías donde $\mathsf{G}_1 + \cdots + \mathsf{G}_n$ y G^* se escriben respectivamente como $\texttt{sel}\ \mathsf{A}\ \{\mathsf{G}_1 + \ldots + \mathsf{G}_n\}$ y $\texttt{repeat}\ \mathsf{A}\ \{\mathsf{G}\}$, donde A es el participante habilitado para seleccionar una de las ramas o detener la iteración. La extensión `.qosgc`, como se discute en la Sección 9.2, contiene especificaciones de calidad adjuntas directamente a uno de los cuatro estados locales involucrados en una interacción: `sqos` denota el estado del remitente antes de la acción de salida, `rqos` el estado del receptor antes de la acción de entrada, `sqos'` el estado del remitente después de la acción de salida, y `rqos'` el estado del receptor después de la acción de entrada. El entorno `qos` que aparece en la línea 25 se utiliza para declarar los atributos de calidad y sus operadores de agregación. En la Figura .5 mostramos el archivo `.ql` para una de las fórmulas utilizadas en el caso de estudio con número de elecciones anidadas = 2; el valor utilizado en la restricción de calidad de la línea 22 fue elegido al azar y garantizado de ser satisfecho por solo una ejecución del sistema (ver línea 21 en la Figura .4). En la Figuras .6 y .7 mostramos los CFSMs y el sistema de transición, respectivamente, para el caso de estudio en la Figura .4.

```
1  sel Bob {
2      Bob -> Alice: m0 ;
3      sel Alice {
4          Alice -> Bob: m0 ;
5          Bob -> Alice: leaf1
6              { sqos: (= x1 1) (= x2 1) (= x3 1) (= x4 1) (= x5 1) }
7          +
8          Alice -> Bob: m1 ;
9          Bob -> Alice: leaf2
10             { sqos: (= x1 2) (= x2 2) (= x3 2) (= x4 2) (= x5 2) }
11     }
12     +
13     Bob -> Alice: m1 ;
14     sel Alice {
15         Alice -> Bob: m0 ;
16         Bob -> Alice: leaf3
17             { sqos: (= x1 3) (= x2 3) (= x3 3) (= x4 3) (= x5 3) }
18         +
19         Alice -> Bob: m1 ;
20         Bob -> Alice: leaf4
21             { sqos: (= x1 4) (= x2 4) (= x3 4) (= x4 4) (= x5 4) }
22     }
23 }
24
25  qos {x1:+,x2:+,x3:+,x4:+,x5:+}
```

Fig. .4: Archivo `.qosgc` para el caso de estudio con número de elecciones anidadas = 2.

```
True U [
sel Bob {
    Bob -> Alice: m0 ;
    sel Alice {
        Alice -> Bob: m0 ;
        Bob -> Alice: leaf1
        +
        Alice -> Bob: m1 ;
        Bob -> Alice: leaf2
    }
    +
    Bob -> Alice: m1 ;
    sel Alice {
        Alice -> Bob: m0 ;
        Bob -> Alice: leaf3
        +
        Alice -> Bob: m1 ;
        Bob -> Alice: leaf4
    }
}
]
qos{(= x1 4) (= x2 4) (= x3 4) (= x4 4) (= x5 4)}
```

Fig. .5: Archivo `.ql` para una de las fórmulas en el caso de estudio con número de elecciones anidadas = 2.

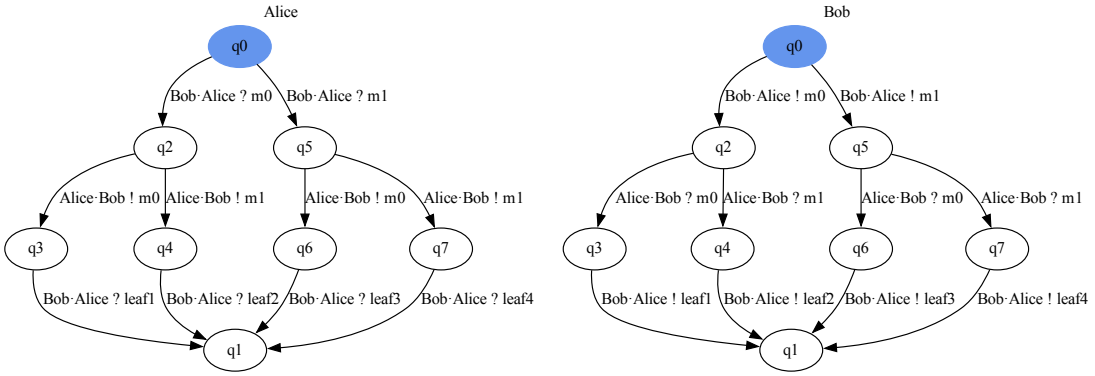


Fig. .6: CFSMs del caso de estudio con elecciones anidadas = 2.

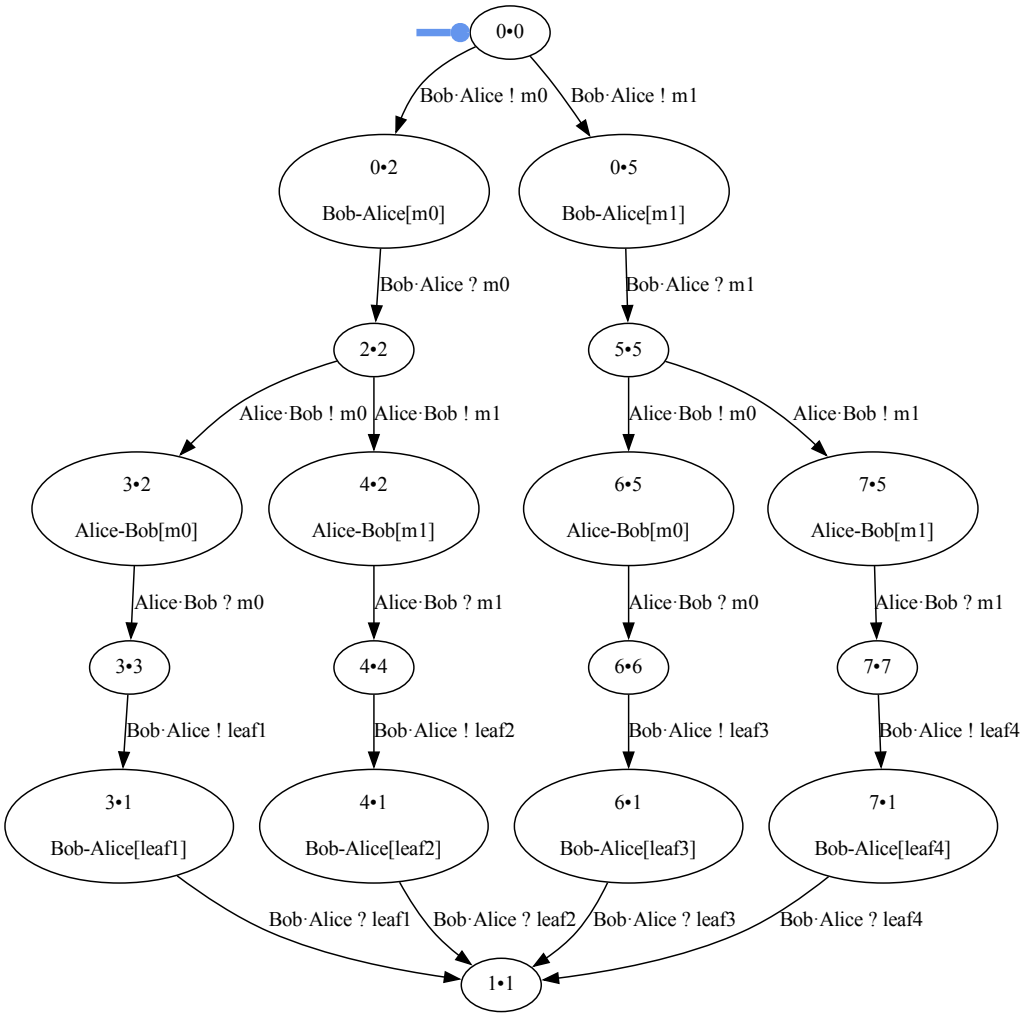


Fig. .7: Sistema de transición del caso de estudio con elecciones anidadas = 2.

C.3. Extracción de modelo

En la Figura .8 mostramos el archivo completo `.qosfsa` para el estudio de extracción de modelos; la sección `fsa` del archivo se omite por brevedad. En las Figura .9 y Figura .10 mostramos los CFSMs y el sistema de transición, respectivamente. En la Figura .11 mostramos el archivo `.ql` de la propiedad Φ_4 utilizado en el caso de estudio.

```
fsa{
    ...
}

qos_attributes{
    p:+,
    t:+,
    pmem:max
}

qos_specifications{
    U@0: (and (< 0 pmem) (< pmem 5) (= t 0) (= p 0)),
    U@2: (and (= pmem 1) (= t 0) (= p 0)),

    M@3: (and (< 0 pmem) (< pmem 10) (= t 0) (= p 0)),
    M@6: (and (= pmem 1) (= t 0) (= p 0)),
    M@8: (and (= pmem 1) (= p 10) (= t 0)),

    W@2: (and (< 0 pmem) (< pmem (+ (/ 5 2) 1)) (= t 1) (=> (<= (- pmem 1) 1) (= p
    1)) (=> (> (- pmem 1) 1) (= p (* 3 (- pmem 1)))))
}

final_states{
    U: [3],
    M: [2],
    W: [1]
}
```

Fig. .8: Archivo `.qosfsa` para el estudio de extracción de modelos.

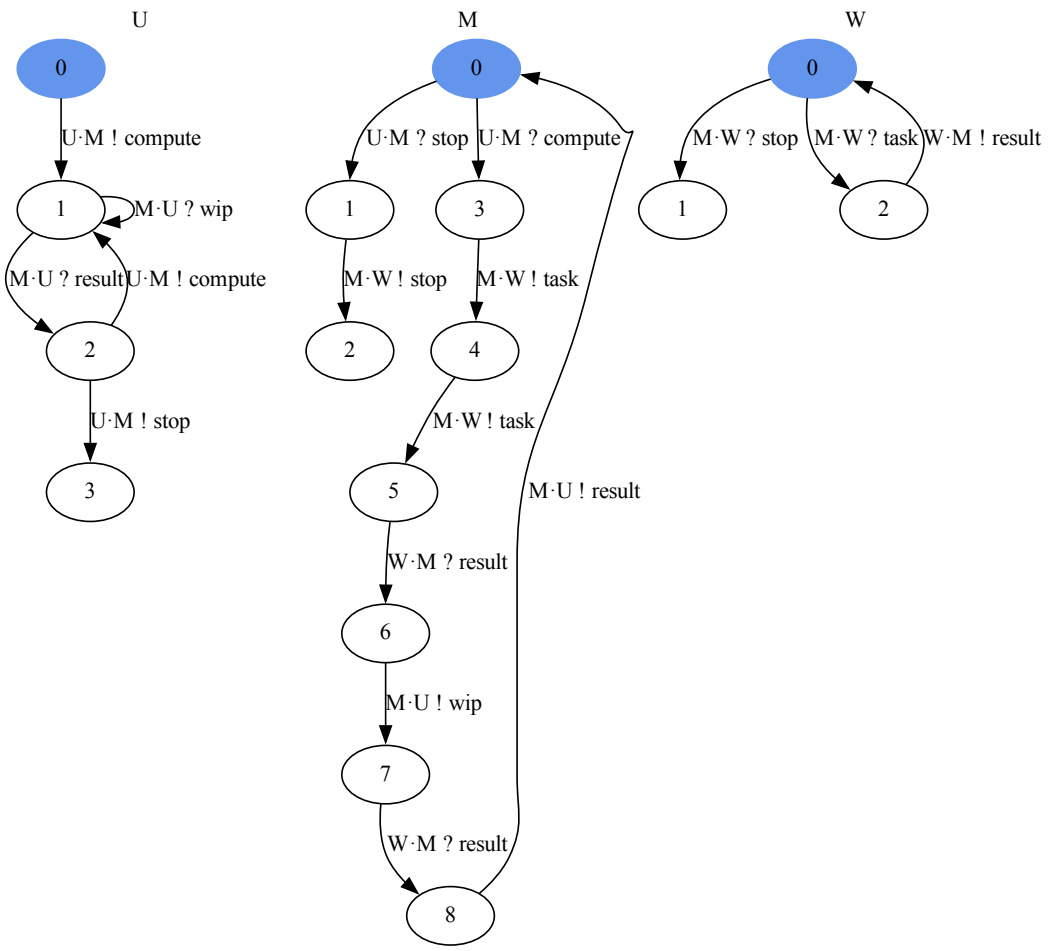


Fig. .9: CFSMs para el estudio de extracción de modelos.

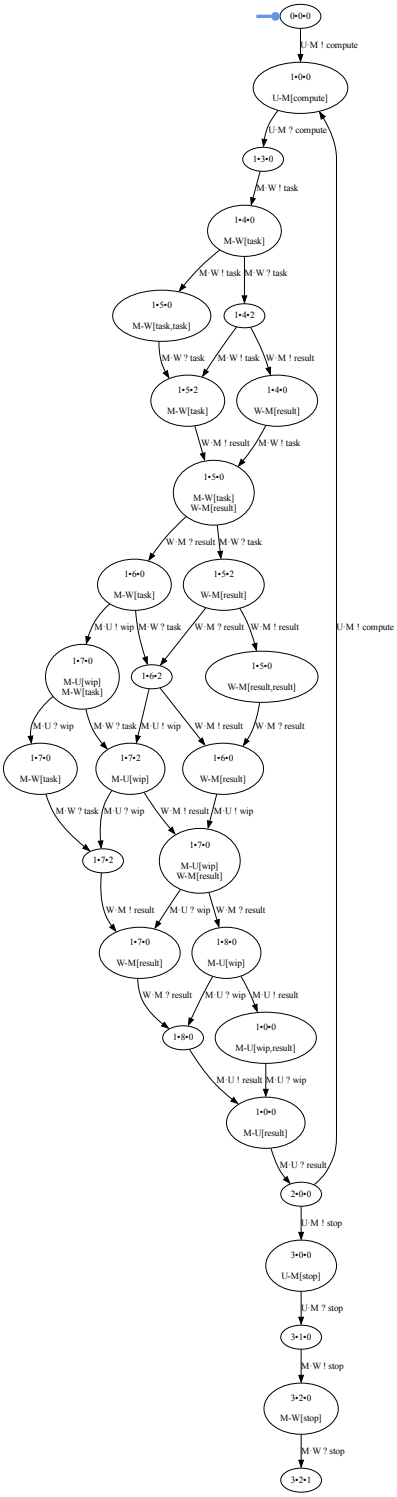


Fig. .10: Sistema de transición del estudio de extracción de modelos.

```
qos{(<= p (* 12.5 t))} U
    [
        U -> M : compute;
        M -> W : task;
        {
            M -> W : task
            |
            W -> M : result
        };
        {
            M -> U : wip
            |
            W -> M : result
        };
        M -> U : result
    ]
    (Not (True U
        [
            repeat {
                U -> M : compute;
                M -> W : task;
                {
                    M -> W : task
                    |
                    W -> M : result
                };
                {
                    M -> U : wip
                    |
                    W -> M : result
                };
                M -> U : result
            }
        ]
        (Not qos{(<= p (* 12.5 2))})
    ))
```

Fig. .11: Archivo `.ql` de la propiedad Φ_4 utilizado en el caso de estudio de extracción de modelos.