Universidad Nacional del Sur

Tesis de Doctorado en Ciencias de la Computación

Formalización y Generalización del Manejo de Preferencias en Servicios de Razonamiento Rebatible

Juan Carlos Lionel Teze

Bahía Blanca Argentina

2017

Prefacio

Esta Tesis se presenta como parte de los requisitos para optar al grado Académico de Doctorado en Ciencias de la Computación, de la Universidad Nacional del Sur y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el ámbito del Departamento de Ciencias e Ingeniería de la Computación durante el período comprendido entre el 1 de Abril de 2012 y el 30 de Marzo de 2017, bajo la dirección del Dr. Guillermo R. Simari, Profesor Titular del Departamento de Ciencias e Ingeniería de la Computación y del Dr. Alejandro J. García, Profesor Asociado del Departamento de Ciencias e Ingeniería de la Computación.

.....

Juan Carlos Lionel Teze
jct@cs.uns.edu.ar
Facultad de Ciencias de la Administración
Universidad Nacional de Entre Ríos
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, 30 de Marzo de 2017

UNIVERSIDAD NACIONAL DEL SUR
Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el .../.../..., mereciendo la calificación de .....(.....)

Agradecimientos

Esta tesis se pudo culminar gracias al aporte de varias personas e instituciones. En primer lugar, quiero agradecer a mis directores Guillermo Simari y Alejandro García que me enseñaron, apoyaron y acompañaron durante estos años de formación científica. Quiero agradecerles toda la paciencia y confianza que han tenido conmigo a través de los años. A pesar de venir de otra universidad y sin conocerme, además de mi poca experiencia en investigación, supieron guiarme y aconsejarme sabiamente para que logre las metas propuestas. Tengo solo palabras de agradecimientos por todo el tiempo que le dedicaron para que pueda finalizar este doctorado, y por sobre todas las cosas al hecho de que siempre dieron un esfuerzo extra, yendo más allá de sus roles como directores. Sin ellos no hubiese sido posible finalizar esta tesis.

Quiero hacer un agradecimiento especial a mi "mentor", el Gotti. Siempre lo digo y nunca me voy a cansar de decirlo, no hubiese terminado la tesis sin su ayuda. Gotty, además de todo lo que aprendí de investigación en estos años que trabajamos juntos, siempre te voy a estar agradecido por todas las charlas y consejos que me diste. No tengo dudas que sin tu ayuda no hubiese podido sobrellevar varios de los obstáculos que se fueron dando en este camino de formación. Mil gracias!

También quiero agradecer a todos los pibes de la "salita de becarios". Personas excepcionales con los que compartí muchísimo momentos felices. Gracias por todas las juntadas, asados, charlas, mates, fútbol y todos esos momentos de alegría vividos a lo largo de estos años. Asimismo, también quiero agradecer al personal administrativo del Departamento de Ciencias e Ingeniería de la Computación (DCIC) por toda la buena onda y predispocisión que siempre tuvieron conmigo. Quiero agradecer a la Universidad Nacional del Sur, por haberme brindado la oportunidad de desarrollar aquí mis tareas de investigación. Por otro lado, tampoco puedo olvidarme de agradecer a mi Universidad Nacional de Entre Rios por brindarme su apoyo institucional, y por todo lo que me brindaron para mi formación. En especial, quiero agradecer a un docente y amigo de esta universidad, Cristian

Pacífico. Gracias a él inicie este camino de doctorado, siempre le voy a estar eternamente agradecido por toda la ayuda que me dio durante estos años, siempre al pie del cañon en todo lo que necesité. Quiero agradecer también al Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) por haberme brindado el sustento necesario para el desarrollo y finalización de mi doctorado.

Quiero dedicar esta tesis a quienes son los pilares de mi vida, mi vieja y mi hermano, Gloria y Ariel. Madre, gracias por enseñarme con tanto amor que nunca hay que bajar los brazos en la búsqueda de aquello que uno anhela, y que en los momentos duros es donde hay que dar ese plus de esfuerzo y sacrificio para afrontarlos. En todos los momentos difíciles recordaba el esfuerzo, sacrificio y lo que la luchaste para que a mi y mi hermano no nos falte nada. Con este ejemplo de vida no había forma de que me rindiera. Hermano, gracias por siempre acompañarme y mostrarme que en las cosas simples y cotidianas radica la verdadera felicidad. Gracias por creer en mi! En especial quiero agradecerles los valores de vida que me enseñaron, que un logro académico no me define. Gracias por enseñarme lo valioso de la vida, a ser cada día mejor persona a partir de valores basados en la dignidad, el respeto, y la solidaridad. Esto es de ustedes!

Por último, quiero agradecer a dios por todas las personas que fue poniendo en mi camino que hicieron posible, de una manera u otra, la culminación de este doctorado. En especial agradecer a todos mis amigos, al grupo de Anímate y a la comunidad de la Parroquia de Lourdes (no me quiero olvidar de nadie). Gracias, gracias y más gracias!

Índice general

Capítulo 1

Introducción

Esta tesis aborda el estudio, diseño y formalización de herramientas computacionales concretas para seleccionar y cambiar el criterio de preferencia entre argumentos que es utilizado por el sistema de Programación Lógica Rebatible (DeLP) requerido para decidir derrotas al analizar ataques entre argumentos. Para lograr esto, se proponen varios servicios de razonamiento basados en DeLP que disponen de distintos criterios y permiten llevar a cabo esta tarea de diferentes maneras. Como parte de la contribución, se propone un servicio que utiliza expresiones condicionales para programar cómo seleccionar el criterio que mejor se ajusta a las preferencias del usuario o a una situación en particular. Por otra parte, en la tesis se aborda también la definición de un servicio con mecanismos que permiten no solo seleccionar sino también combinar criterios. Estos mecanismos permiten que sea posible comparar argumentos considerando de manera simultánea más de un criterio.

Como se detalla a continuación, DeLP ha demostrado ser de gran utilidad en diferentes dominios de aplicación [CCS05, RGS07, GCS08, GGS10]. Los formalismos propuestos incorporan herramientas concretas para tratar el manejo de múltiples criterios de preferencia entre argumentos, lo cual no ha sido considerado hasta el momento por otros trabajos. En consecuencia, los resultados obtenidos en esta tesis brindan una contribución importante a los desarrollos en la comunidad de argumentación, particularmente en el campo de los sistemas basados en Programación Lógica Rebatible, significando además un aporte dentro del área de Inteligencia Artificial en las Ciencias de la Computación.

En lo que resta de este capítulo se presenta brevemente el contexto que motiva esta investigación y se describen las principales contribuciones y resultados obtenidos, los que han sido publicados en revistas y congresos nacionales e internacionales. Finalmente, se

indica la estructura del resto de la tesis, detallando sintéticamente el contenido de los capítulos siguientes.

Argumentación Rebatible

La argumentación es un proceso de razonamiento donde se consideran argumentos, sus posibles conflictos, y las conclusiones que se pueden obtener a partir de los mismos. En esta forma de razonamiento, dada una consulta, se analizan todos los argumentos en conflicto relacionados a la consulta para luego determinar si la consulta será aceptada.

Desde hace tiempo, la argumentación ha evolucionado como una propuesta atractiva para modelar el razonamiento basado en sentido común [CML00, BCD07, BH08], que usualmente sucede en el contexto de información contradictoria, incompleta e incierta. Existen diferentes acercamientos para modelar este proceso, tales como los sistemas argumentativos abstractos [Dun95, Vre97] que se abstraen de la estructura interna de los argumentos, o los sistemas argumentativos estructurados [PS97, GS04], que sí tienen en cuenta la estructura interna de los mismos. Aquí nos limitaremos a estos últimos.

Los sistemas argumentativos estructurados son formalismos de argumentación basados en una lógica subyacente específica, la cual se emplea para representar el conocimiento acerca del dominio sobre el que se razona. Además, esta lógica cuenta con un conjunto de reglas de inferencia que permiten construir argumentos relacionados con una conclusión. Estos sistemas son de gran interés para la comunidad de Inteligencia Artificial dado que las reglas de inferencia permiten representar conocimiento de sentido común, posibilitando la construcción computacional de argumentos. Los sistemas argumentativos estructurados poseen características que los hacen especialmente aptos para su implementación; por otra parte, los sistemas argumentativos son particularmente atractivos para la toma de decisiones y la negociación [APM00, BH09], áreas de gran interés en diversas aplicaciones.

Programación Lógica Rebatible y la Comparación entre Argumentos

Como se ha mencionado, la importancia del uso de sistemas argumentativos como mecanismo de razonamiento en sistemas inteligentes es reconocida en diversos trabajos en la literatura [BCD07, CMS07, RS09]. Esta tesis se enfoca en un sistema argumentativo estructurado, denominado Programación Lógica Rebatible [GS04]. Este formalismo combina resultados de programación en lógica y argumentación rebatible, y ha sido aplicado exitosamente en diferentes dominios (ver *e.g.,* [CCS05, RGS07, GCS08, GGS10]).

La Programación Lógica Rebatible extiende a la Programación en Lógica permitiendo representar conocimiento potencialmente contradictorio, mediante el uso de la negación fuerte y conocimiento tentativo, incluyendo en la sintaxis un nuevo tipo de reglas: las reglas rebatibles.

El procedimiento de prueba de la Programación Lógica Rebatible se basa en un proceso de análisis dialéctico donde interactúan argumentos a favor y en contra de un literal a fin de determinar si ese literal está garantizado. Este proceso conducirá a la construcción de una estructura arbórea de derrotadores. De este modo, un literal se hallará garantizado si existe un argumento para ese literal que sobreviva a todas las derrotas que recibe en su árbol asociado. Para determinar si un argumento es un derrotador de otro con el que está en conflicto es necesario contar con un criterio de preferencia que decida preferencia entre estos argumentos.

Si bien al presentarse DeLP en [GS04] se asoció el criterio de Especificidad Generalizada como criterio por defecto, en DeLP el criterio es un elemento modular. Sin embargo, en la comunidad de argumentación no existe un consenso establecido acerca de qué criterio utilizar para evaluar argumentos. En este sentido, en trabajos como [Vre91, Pol95, PS96a] los autores declaran que el criterio de especificidad no corresponde a un criterio general del razonamiento de sentido común, sino que simplemente es uno de muchos métodos de comparación que podrían ser utilizados. Más aún, propuestas como [Kon88, Vre91] sugieren que la información acerca del dominio suele ser la herramienta principal para evaluar los argumentos del sistema. De allí la importancia de contar con un sistema como [GS04] que permita tratar la comparación de argumentos de forma modular, permitiendo que el usuario emplee el criterio que mejor se ajusta al dominio de aplicación reemplazando el provisto por defecto como parte del sistema.

Existen varios criterios concretos para comparar argumentos en DeLP (ver *e.g.,* [GS04, FEGS08]). No obstante, los sistemas que utilizan DeLP como mecanismo de razonamiento y consideran estos criterios en sus formalismos [CMS06, DDG$^+$12, TGG$^+$12, GMP12, BCM13, BBD$^+$14], por lo general adoptan un criterio o una combinación fija de criterios (establecida en la configuración del sistema) de acuerdo al dominio que se está representando. Si bien en DeLP la comparación de argumentos es modular, en la literatura existente no se han propuesto mecanismos programables concretos que le permitan al usuario seleccionar y cambiar dinámicamente el criterio de preferencia dependiendo de sus preferencias o necesidades. Si un usuario conoce las razones por las cuales se prioriza cierta información sobre otra, el mecanismo para comparar argumentos deja de ser una

caja negra, aumentando la confianza del usuario sobre las respuestas que puede recibir de estos sistemas en particular. Así el sistema se vuelve más confiable y transparente para el usuario ya que éste puede interactuar directamente con el mecanismo de razonamiento de dicho sistema.

Otra característica de los sistemas basados en DeLP, respecto al manejo de los mecanismos de comparación sobre argumentos, es que usualmente tampoco disponen de varios criterios para que el usuario pueda elegir el que más se adecua a un contexto determinado. En este sentido, en la vida real el proceso de razonamiento humano hace frente a una situación particular teniendo en cuenta diferentes criterios. Por ejemplo, supongamos una persona que se quiere hospedar en un hotel y consulta un sitio web que le ofrece información respecto a varios hoteles. Para elegir un hotel, esta persona podría considerar varios criterios que le permitan comparar la información que le brinda el sitio. Un posible criterio podría preferir aquella información que favorece al comfort de las habitaciones, mientras que otro criterio podría preferir la información que favorece a la seguridad del lugar en donde se encuentra el hotel. Finalmente, un tercer criterio podría ser uno que combine los dos anteriores, es decir un criterio que priorice la información relacionada al comfort y la seguridad. Por lo tanto, disponer de diferentes criterios de preferencias introduce un grado de flexibilidad adicional a los sistemas basados en DeLP. Sin embargo, y como se puntualizó anteriormente, en estos sistemas el criterio es un componente fijo, o si existe multiplicidad de criterios, no hay forma de cambiarlo una vez que uno es seleccionado e integrado al intérprete. De esta manera, una de las motivaciones de esta tesis es mejorar las capacidades del mecanismo de inferencia de un sistema basado en argumentación para que se pueda adaptar de una forma natural a varios criterios, a través de un mecanismo programable concreto.

A partir de lo mencionado anteriormente, las contribuciones de esta tesis conciernen a la formalización de varios servicios de razonamiento basados en DeLP que abordan los temas planteados en esta sección.

1.1. Contribuciones

En primer lugar se introduce formalmente la noción de *Servicio de Razonamiento Rebatible basados en Preferencias* (*SRPref*). Un *SRPref* corresponde a una entidad con la capacidad de responder consultas DeLP utilizando el criterio de preferencia entre argumentos que el usuario especifica junto a su consulta y el programa consultado. Un *SRPref*

cuenta con la implementación de varios criterios disponibles para el usuario. Por otra parte, como herramienta de interacción entre el usuario y un *SRPref* se introduce el concepto de *Consulta basada en Preferencias* (*CPref*). Una *CPref* se compone además de la consulta propiamente dicha, del programa DeLP que será procesado y una especificación declarativa del criterio de preferencia que el usuario desea utilizar. En la especificación también se podrá incluir información adicional que el criterio elegido podría llegar a necesitar para poder ejecutarse. Todos estos elementos serán los que un *SRPref* tendrá en cuenta al momento de computar la respuesta a una consulta.

Luego se presentan los conceptos de *Servicio de Razonamiento Rebatible basado en Preferencias Condicionales* (*SRPCond*) y *Consulta basada en Preferencias Condicionales* (*CPCond*). Una *CPCond* se caracteriza por incluir junto a la consulta y el programa a consultar, la especificación de una expresión condicional que servirá para programar cómo seleccionar el criterio que el *SRPCond* consultado utilizará a partir de los criterios especificados. La selección de un criterio en particular dependerá de la existencia de determinada información alojada en el programa consultado. Los *SRPConds* están especialmente diseñados para recibir y resolver *CPConds*.

Por último, se formaliza el concepto de *Servicio de Razonamiento Rebatible basado en Preferencias Combinadas* (*SRPComb*). Con estos servicios se introduce la noción de *Operador de Combinación de Preferencias*, el cual permite combinar criterios. La idea de contar con estos operadores es la de realizar una comparación de argumentos utilizando varios criterios al mismo tiempo. Para lograr esto, se estudiaron y analizaron diferentes formas de combinar criterios. Por otra parte, las consultas que responden estos servicios son las *Consultas basadas en Preferencias Combinadas* (*CPComb*). Estas consultas tienen la particularidad de permitir que el usuario pueda especificar una expresión que combine varios criterios, y que luego será utilizada por el *SRPComb* consultado.

1.2. Publicaciones

A continuación se indican los artículos publicados como resultado de los trabajos llevados a cabo durante la realización de esta tesis. Para cada uno de estos trabajos, se detalla brevemente su contribución y se indican los capítulos de esta tesis con los que se vinculan sus resultados.

- En el trabajo "*Improving argumentation-based recommender systems through*

context-adaptable selection criteria" [TGGS15] publicado en la revista *Expert Systems with Applications vol. 42*,

se introduce un Servidor de Razonamiento basado en DeLP, el cual permite una selección del criterio de preferencia adaptable al contexto. En particular se provee una forma de especificar cómo seleccionar el criterio más apropiado en base a las preferencias del usuario. Para lograr esto, se utiliza una expresión condicional que dependiendo del contexto le indicará al servidor qué criterio utilizar. Los resultados de este artículo son la base del trabajo desarrollado en el Capítulo 6.

- En los trabajos:
  - "*An Approach to Argumentative Reasoning Servers with Conditions based Preference Criteria*" [TGGS13a] publicado en *XIX Congreso Argentino de Ciencias de la Computación (CACIC 2013)*, y
  - "*Modelo de Servicio de Razonamiento con Preferencias*" [TGGS13b] publicado en el *XV Workshop de Investigadores en Ciencias de la Computación (WICC 2013)*,

    se introduce una versión preliminar del trabajo propuesto en [TGGS15]. En tales formalismos se introduce por primera vez un modelo de Servicio de Razonamiento basado en Argumentación Rebatible que cuenta con un conjunto de criterios de preferencias como un módulo independiente del mecanismo de inferencia. Los resultados de estos artículos se encuentran estrechamente relacionados con el Capítulo 5 y 6.

- En los trabajos:
  - "*An Approach to Argumentative Reasoning Servers with Multiple Preference Criteria*" [TGGS14a] publicado en la revista *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial vol. 53*, y
  - "*Servicios de Razonamiento con Múltiples Criterios de Preferencia*" [TGGS14b] publicado en *XVI Workshop de Investigadores en Ciencias de la Computación (WICC 2014)*,

    se presenta un modelo de Servicio de Razonamiento basado en DeLP en donde se pueden seleccionar y combinar diversos criterios de preferencias. Para esto, se

propuso un conjunto de operadores que permitan la combinación de múltiples criterios de preferencias. Por último, se introduce un tipo especial de consulta dirigida a estos servicios en particular. Los resultados de estos trabajos están vinculados con el desarrollo del formalismo del Capítulo 7.

1.3. Organización de la tesis

El resto de esta tesis se encuentra organizado de la siguiente manera:

- En el Capítulo 2 se presenta un estudio general sobre algunas nociones relacionadas al concepto de preferencias, las cuales serán esenciales para los desarrollos de esta tesis. Seguidamente, se aborda el concepto de combinación de preferencias presentando algunos trabajos que incluyen este concepto dentro de sus formalismos.

- En el Capítulo 3 se introduce una estructura conceptual que caracteriza los diferentes elementos presentes en los sistemas argumentativos. Luego, se describen algunos de estos sistemas, enfatizando en aquellos que incorporan la noción de preferencias entre argumentos en su formalismos. Finalmente, se realizará una discusión respecto de algunos aspectos relacionados a las posturas adoptadas por los investigadores al momento de definir el criterio de preferencia entre argumentos que debe utilizar un sistema argumentativo. El estudio de este tema resulta fundamental para esta tesis, ya que constituye una de las motivaciones de la misma.

- En el Capítulo 4 se presenta el formalismo base para los desarrollos de esta tesis; la Programación Lógica Rebatible (DeLP). En particular, se muestra un resumen del sistema de DeLP, describiendo el lenguaje y el procedimiento de prueba desde una perspectiva sintáctica. Por otro lado, también se analizan diferentes criterios que pueden ser utilizados por este formalismo. En este capítulo también se presentan los conceptos de Servidores de Razonamiento basados en Programación Lógica Rebatible y Consultas Contextuales. Estos conceptos representan la base para las formalizaciones que se desarrollarán en los capítulos siguientes.

- En el Capítulo 5 se definen los Servicios de Razonamiento Rebatible basados en Preferencias y las Consultas basadas en Preferencias, que representan el primer aporte principal de esta tesis doctoral. En primer lugar se definen los componentes de un Servicio de Razonamiento Rebatible basados en Preferencias. Luego se introduce

el concepto de Consulta basada en Preferencias. Estas consultas constituyen la herramienta de interacción que permite configurar el criterio de preferencia que debe utilizar un servicio cada vez que recibe una consulta.

- En el Capítulo 6 se introducen las Consultas basadas en Preferencias Condicionales. Estas consultas se caracterizan por incluir una expresión condicional que permite programar qué criterio seleccionar en base a las preferencias del usuario. Se definen también los Servicios de Razonamiento Rebatible basados en Preferencias Condicionales capaces de recibir este tipo de consultas. A su vez, se presenta una representación de árbol para el estudio de las expresiones condicionales.

- En el Capítulo 7 se presenta el Servicio de Razonamiento Rebatible basado en Preferencias Combinadas que se caracteriza por contar con los mecanismos necesarios para combinar criterios. Además, se introduce la Consulta basada en Preferencias Combinadas. Este tipo de consulta permite la construcción de expresiones que le indican el uso de criterios combinados al servicio, favoreciendo la comparación de argumentos a partir de la utilización de varios criterios simultáneamente.

- En el Capítulo 8 se presentan las conclusiones y el trabajo a futuro.

Capítulo 2

Preferencias en Inteligencia Artificial

Como fue indicado en la introducción, el objetivo de esta tesis es mejorar las capacidades de razonamiento de los sistemas basados en programación lógica rebatible. En este capítulo se incluirán algunas nociones vinculadas al concepto de *preferencia*; luego, se abordará la noción de combinación de preferencias mostrando su importancia como medio para expresar preferencias complejas. Para esto, se explorarán dos trabajos que incluyen operaciones que capturan esta noción, y que además resultan relevantes para los aportes de esta tesis. En particular, se describirán las propuestas de Chomicki [Cho03] y de Brewka *et al.* [BDRS15].

2.1. Preferencias

Dada su importancia, el concepto de preferencia ha sido ampliamente estudiado y sigue actualmente siendo un área de gran interés [DHKP11, Kac11, PTV14]. Para elegir una alternativa entre un conjunto de posibilidades implícitamente debemos ejecutar un proceso de comparación. Para describir estas comparaciones, las preferencias se modelan de diferentes maneras. Tradicionalmente el modelado de las preferencias se ha realizado de forma cuantitativa o cualitativa [Kac11, SKP11]. En los enfoques cuantitativos las preferencias se especifican usando una función de utilidad que asigna un valor numérico a cada uno de los elementos a comparar, representando a su vez un orden total sobre estos elementos. Por ejemplo, en el trabajo de [AW00] se utiliza una función de preferencia que asigna ponderaciones numéricas a tuplas de bases de datos. De esta manera, una tupla t_i es preferida a otra tupla t_j si y solo si la ponderación de la primera tupla es mayor que la ponderación de la segunda.

Por otro lado, en los enfoques cualitativos las preferencias se expresan utilizando relaciones (de preferencia) binarias, que a diferencia de los enfoques cuantitativos la relación no será necesariamente un orden total. Por lo general, cada trabajo que utiliza este tipo de preferencia establece en su formalismo qué propiedades debe cumplir la relación binaria que la define. Esta situación, por ejemplo, se puede observar en los trabajos que se presentarán en la siguiente sección [Cho03, BDRS15]. En [Cho03] no se asumen propiedades para la relación de preferencia definida sobre las tuplas de una base de datos, aunque los autores aclaran que en la mayoría de las aplicaciones se requiere que se satisfagan al menos las propiedades de un orden parcial estricto (irreflexividad, asimetría y transitividad). En cambio, en [BDRS15] la preferencia sobre modelos estables se describe directamente a partir de un orden parcial estricto.

A pesar que los enfoques cuantitativos permiten expresar explícitamente cuánto un elemento es preferido a otro, en Teoría de la Utilidad [Fis70, Fis99] es bien conocida la falta de expresividad por parte de estas perspectivas. Dado que todas las funciones de utilidad se pueden expresar mediante relaciones de preferencias, pero no todas las relaciones se pueden expresar mediante funciones de utilidad, es que en esta tesis se tomará el enfoque cualitativo.

Una *preferencia* es una relación binaria sobre un conjunto ϑ, *i.e.,* $\succeq \subseteq \vartheta \times \vartheta$. En general, la notación $x \succeq y$ se leerá como "x es al menos tan preferido como y". Así una preferencia establece una relación a la que nos referiremos como *relación de preferencia*. Para todo $x, y \in \vartheta$, a partir de $\succeq$ se pueden considerar tres casos:

- x es estrictamente preferido a y, denotado $x \succ y$, cuando se cumple que $x \succeq y$ pero $y \not\succeq x$; se llamará a la relación $\succ$ relación de preferencia estricta.
- x es indiferente a y, denotado $x \approx y$, si se cumple tanto $x \succeq y$ y $y \succeq x$. Se dirá que es indiferente preferir x o y.
- x es incomparable a y, denotado $x \bowtie y$, si se satisface que $x \not\succeq y$ y $y \not\succeq x$. Se dirá que x e y son incomparables.

Las relaciones binarias, en general, pueden satisfacer ciertas propiedades algunas de las cuales son útiles en la clasificación de las relaciones de preferencias. Se enumeran a continuación algunas a las que se hará referencia:

- $\succeq$ es reflexiva si y solo si $\forall x \in \vartheta$, $x \succeq x$.

- $\succeq$ es irreflexiva si y solo si $\forall x \in \vartheta$, se cumple que $x \not\succeq x$.
- $\succeq$ es completa si y solo si $\forall x, y \in \vartheta$, tenemos que $x \succeq y$ o $y \succeq x$.
- $\succeq$ es transitiva si y solo si $\forall x, y, z \in \vartheta$, si $x \succeq y$ y $y \succeq z$ entonces $x \succeq z$.
- $\succeq$ es simétrica si y solo si $\forall x, y \in \vartheta$, si $x \succeq y$ entonces $y \succeq x$.
- $\succeq$ es antisimétrica si y solo si $\forall x, y \in \vartheta$, si $x \succeq y$ y $y \succeq x$ entonces $x = y$.
- $\succeq$ es asimétrica si y solo si $\forall x, y \in \vartheta$, si $x \succeq y$ entonces se cumple que $y \not\succeq x$.

Dadas estas propiedades, una relación de preferencia $\succeq$ puede definirse como:

- Un *preorden total*, cuando $\succeq$ es reflexiva, completa y transitiva.
  - Si $\succeq$ es antisimétrica, entonces es un *orden total*;
  - si $\succeq$ es asimétrica, entonces $\succeq$ es un *orden total estricto*.
- Un *preorden parcial*, cuando corresponde a una relación reflexiva y transitiva.
  - Se dice que $\succeq$ es un *orden parcial* si es antisimétrica;
  - mientras que es un *orden parcial estricto* si es asimétrica.

En la siguiente sección se muestran algunas propuestas existentes para expresar preferencias complejas. Para esto se describirán dos trabajos cuyo formalismos ofrecen la posibilidad de establecer diferentes formas de combinar preferencias.

2.2. Combinación de Preferencias

Combinar preferencias es un proceso que los seres humanos realizan de forma cotidiana. Por ejemplo, una persona que quiere reservar una habitación de hotel podría preferir aquellos hoteles que ofrezcan habitaciones económicas, como así también aquellos que incluyan en sus servicios el uso de cochera. Dependiendo de las preferencias de la persona que realiza la reserva, estos dos criterios de selección de hoteles se podrían combinar de diferentes maneras. Una combinación podría preferir aquellos hoteles con habitaciones económicas que además proveen servicio de cochera, mientras que otra podría priorizar aquellos con habitaciones económicas únicamente si no existieran hoteles con servicio de

cochera. Otro ejemplo común de uso de preferencias se les plantea a los conductores al momento de elegir las rutas que lo llevarán a su destino; si el conductor desea gastar poco combustible y llegar lo antes posible a su destino, por lo general priorizará aquellas rutas más cortas y menos transitadas. Note que para que se satisfaga el requisito establecido por el conductor es necesario que se cumplan los dos criterios, si alguno no se cumple es probable que llegue tarde o gaste más combustible de lo previsto. Claramente, se podrían seguir enumerando situaciones en donde la combinación de preferencias juega un rol importante en las decisiones de las personas. Este aspecto ha llevado a investigadores en el campo de la Inteligencia Artificial y de las Bases de Datos a considerar múltiples relaciones de preferencia en sus investigaciones y desarrollar distintas formas para que éstas se puedan combinar [AW00, SP06, SKP11].

Es importante resaltar que si bien varios trabajos en la literatura han tratado de una manera u otra el problema de expresar y combinar preferencias, en lo que sigue se exploran los enfoques propuestos en [Cho03] y [BDRS15] los cuales se han desarrollado en el contexto de dos disciplinas ampliamente reconocidas dentro de las Ciencias de la Computación: *Base de Datos* y *Answer Set Programming* (*ASP*). En particular, la elección de estos trabajos se debe a los siguientes motivos.

Por un lado, en [Cho03] además de mostrar varias formas de combinar preferencias para expresar preferencias complejas, el autor realiza un estudio sobre las propiedades que preserva la relación que resulta de realizar este tipo de combinaciones. En particular, estos resultados serán de interés para el formalismo presentado en el Capítulo 7.

Por otra parte, en [BDRS15] se propone una especificación para expresar e implementar diferentes formas de combinación de preferencias utilizando codificación *ASP*. Es importante recordar que los desarrollos de esta tesis se centran en el formalismo argumentativo de programación lógica rebatible DeLP que se define como una extensión del lenguaje de la programación en lógica. Por lo tanto, si bien es claro que las contribuciones de esta tesis no están enfocadas al área de *ASP*, con este trabajo se busca mostrar una propuesta concreta en programación en lógica sobre combinación de preferencias.

Finalmente, un aspecto importante de estos dos formalismos es su generalidad y flexibilidad al momento de especificar preferencias complejas. Como se mostrará a continuación, estas características permiten capturar varios de los enfoques propuestos en la literatura que tratan con el manejo de preferencias, en consecuencia ésto también hace que sean formalismos de interés para la comunidad científica. En este aspecto, uno de los aportes de esta tesis concierne el estudio de estos trabajos para la definición de un formalismo que

trata la combinación de preferencias en un contexto de programación lógica rebatible.

2.2.1. Marco para combinar preferencias [Cho03]

Tratar con las preferencias del usuario es una cuestión importante al momento de realizar una consulta a una base de datos. Supongamos que un usuario está buscando un hotel en Bahía Blanca. Debido a la cantidad de hoteles en esta ciudad, es claro que proveer en la consulta información sobre las preferencias del usuario ayudará a eliminar hoteles irrelevantes dando mayor importancia a aquellos que sí son de interés, actuando como un filtro sobre la información procesada. A medida que aumenta la información procesada, incrementa el tiempo y esfuerzo de cómputo requerido. En este contexto el uso de preferencias en el procesamiento de las consultas ha sido un tema abordado por varios investigadores al momento de tratar eficientemente esta situación. En esta sección se presenta una propuesta que permite expresar preferencias complejas del usuario mediante el uso de diferentes formas de combinación de preferencias.

Desde hace tiempo la comunidad de investigadores que trabaja en el área de sistemas de bases de datos ha reconocido la importancia de integrar preferencias a las consultas. El trabajo de Lacroix y Lavency [LL87] fue el primero en tratar este tema al extender las consultas realizadas a bases de datos clásicas con preferencias. Siguiendo esta línea, se han desarrollado diversas propuestas al respecto; por ejemplo, en [BKS01] se introdujo el operador Skyline que compara pares de tuplas teniendo en cuenta diferentes dimensiones. Las mejores tuplas son aquellas que no son dominadas por ninguna otra tupla. Por otra parte, Kiessling [Kie02] formula las preferencias en las consultas a la base de datos mediante constructores de preferencias y operadores de composición. Chomicki [Cho02, Cho03] por su parte propone una variante que generaliza el enfoque de Kiessling [Kie02]. Finalmente, en la literatura se han propuesto otros trabajos que siguen esta misma línea, tales como el de Brafman and Domshlak [BD04] y el de Ciaccia [Cia07]. Note que los enfoques mencionados ofrecen una representación cualitativa de preferencias, no obstante en la literatura existen también propuestas que utilizan preferencias cuantitativas [FW97, AW00, Fag02, HKP11].

A continuación se describe el trabajo *Preference Formulas in Relational Queries* [Cho03] presentado en 2003 por Jan Chomiscki, en donde se introduce un marco lógico para expresar relaciones de preferencia sobre tuplas mediante fórmulas lógicas. Como se detalla a continuación, este marco permite definir diferentes formas de combinar preferencias, y para esto el autor distingue dos formas de composición: unidimensional y

multidimensional.

En [Cho03], las preferencias sobre tuplas se definen utilizando relaciones de preferencia. De esta manera, se denota $\succ$ a la relación de preferencia entre tuplas definida sobre un esquema relacional determinado. Se dice que una tupla t_1 domina a t_2 bajo $\succ$ si $t_1 \succ t_2$. El foco de atención del trabajo son las relaciones de preferencia definidas utilizando *fórmulas de preferencias*. Una fórmula de preferencia, denotada $\mathbb{C}(t_1, t_2)$, es una fórmula de primer orden que define una relación de preferencia $\succ_{\mathbb{C}}$ en el sentido que

$$t_1 \succ_{\mathbb{C}} t_2 \text{ si y solo si } \mathbb{C}(t_1, t_2).$$

La relación de indiferencia $\bowtie_{\mathbb{C}}$ generada a partir de $\succ$ se define como: t_1 y t_2 son indiferentes (denotado $t_1 \bowtie_{\mathbb{C}} t_2$) si ninguno es preferido al otro, es decir $t_1 \not\succ_{\mathbb{C}} t_2$ y $t_2 \not\succ_{\mathbb{C}} t_1$.

En el formalismo propuesto, las relaciones de preferencias pueden ser compuestas de diferentes maneras generando una nueva relación de preferencia. En general, se distinguen dos tipos de composición: *unidimensional* y *multidimensional*. En la composición unidimensional, se combina un número de relaciones de preferencias que son aplicadas a un mismo esquema de base de datos, produciendo otra relación de preferencia sobre el mismo esquema. Por otro lado, en la composición multidimensional se tiene un número de relaciones de preferencia definidas sobre varios esquemas relacionales, y el resultado es una relación de preferencia definida sobre el producto cartesiano de estos esquemas.

Dentro de la composición de relaciones se distinguen las composiciones booleanas. La relación resultante correspondiente a la unión, intersección, y diferencia entre dos relaciones de preferencia es capturada por las operaciones booleanas aplicadas sobre las fórmulas de preferencia correspondientes. Por lo tanto, dada las relaciones de preferencia $\succ_{\mathbb{C}_1}$ y $\succ_{\mathbb{C}_2}$, las relaciones de composición booleanas se definen de la siguiente manera:

- la *relación de preferencia unión* $\succ_{\mathbb{C}_1 \vee \mathbb{C}_2}$ corresponde a la relación $\succ_{\mathbb{C}_1} \cup \succ_{\mathbb{C}_2}$, tal que se cumple que $t_1 \succ_{\mathbb{C}_1 \vee \mathbb{C}_2} t_2$ si y solo si $(t_1 \succ_{\mathbb{C}_1} t_2) \vee (t_1 \succ_{\mathbb{C}_2} t_2)$.
- la *relación de preferencia intersección* $\succ_{\mathbb{C}_1 \wedge \mathbb{C}_2}$ corresponde a la relación $\succ_{\mathbb{C}_1} \cap \succ_{\mathbb{C}_2}$, tal que se cumple que $t_1 \succ_{\mathbb{C}_1 \wedge \mathbb{C}_2} t_2$ si y solo si $(t_1 \succ_{\mathbb{C}_1} t_2) \wedge (t_1 \succ_{\mathbb{C}_2} t_2)$.
- la *relación de preferencia diferencia* $\succ_{\mathbb{C}_1 - \mathbb{C}_2}$ corresponde al conjunto diferencia $\succ_{\mathbb{C}_1} - \succ_{\mathbb{C}_2}$, tal que se cumple que $t_1 \succ_{\mathbb{C}_1 - \mathbb{C}_2} t_2$ si y solo si $t_1 \succ_{\mathbb{C}_1} t_2 \wedge (t_1 \not\succ_{\mathbb{C}_2} t_2)$.

Existen situaciones en las cuales aparecen preferencias priorizadas. En una composición priorizada una de las relaciones de preferencia tiene prioridad sobre otra. La *relación de preferencia priorizada* $\succ_{\mathbb{C}_1 \rhd \mathbb{C}_2}$ se define como:

$$t_1 \succ_{\mathbb{C}_1 \rhd \mathbb{C}_2} t_2 \text{ si y solo si } t_1 \succ_{\mathbb{C}_1} t_2 \vee (t_1 \bowtie_{\mathbb{C}_1} t_2 \wedge t_1 \succ_{\mathbb{C}_2} t_2)$$

La composición priorizada $\succ_{\mathbb{C}_1} \rhd \succ_{\mathbb{C}_2}$ tiene la siguiente lectura intuitiva: *preferir de acuerdo a* $\succ_{\mathbb{C}_2}$ *si* $\succ_{\mathbb{C}_1}$ *no se puede aplicar.* La composición priorizada sobre relaciones de preferencia definidas sobre esquemas relacionales diferentes, se denomina composición lexicográfica.

Además de las composición unidimensional en [Cho03] también se han estudiado algunas formas de definir una relación de preferencia sobre el producto cartesiano de dos relaciones. En particular, el autor describe dos formas de composición multidimensional ampliamente reconocida en teoría de la decisión [Fis70]: pareto y lexicográfica. Dados dos esquemas de bases de datos R_1 y R_2 y las relaciones de preferencia $\succ_{\mathbb{C}_1}$ sobre R_1 y $\succ_{\mathbb{C}_2}$ sobre R_2. La *relación de preferencia pareto* $\succ_{P(\succ_{\mathbb{C}_1}, \succ_{\mathbb{C}_2})}$ definida sobre el producto cartesiano $R_1 \times R_2$ es tal que:

$$(t_1, t_2) \succ_{P(\succ_{\mathbb{C}_1}, \succ_{\mathbb{C}_2})} (t'_1, t'_2) \text{ si y solo si } t_1 \succeq_{\mathbb{C}_1} t'_1 \wedge t_2 \succeq_{\mathbb{C}_2} t'_2 \wedge (t_1 \succ_{\mathbb{C}_1} t'_1 \vee t_2 \succ_{\mathbb{C}_2} t'_2),$$
$$\text{donde } t \succeq t' \text{ si y solo si } t \succ t' \vee t \bowtie t'.$$

Por otro lado, la *relación de preferencia lexicográfica* $\succ_{L(\succ_{\mathbb{C}_1}, \succ_{\mathbb{C}_2})}$ se define como:

$$(t_1, t_2) \succ_{L(\succ_{\mathbb{C}_1}, \succ_{\mathbb{C}_2})} (t'_1, t'_2) \text{ si y solo si } t_1 \succ_{\mathbb{C}_1} t'_1 \vee (t_1 \bowtie_{\mathbb{C}_1} t'_1 \wedge t_2 \succ_{\mathbb{C}_2} t'_2).$$

Ejemplo 2.1. *Consideremos una base de datos que almacena información acerca de películas. En la Figura 2.1 se ilustra una instancia de dicha base de datos.*

Supongamos que Maria tiene las siguientes preferencias respecto a las películas que desea mirar. Prefiere las películas de romance sobre las de acción ($\mathbb{C}_1$) *y las de mayor duración* ($\mathbb{C}_2$)*. Por lo tanto,* $\mathbb{C}_1$ *se puede definir como:* $t_i \succ_{\mathbb{C}_1} t_j$*, si y solo si* $t_i[\mathsf{Genero}] = \mathsf{romance} \wedge t_j[\mathsf{Genero}] = \mathsf{accion}$*. Mientras que* $\mathbb{C}_2$ *se puede expresar como:* $t_i \succ_{\mathbb{C}_2} t_j$ *si y solo si* $t_i[\mathsf{Duracion}] > t_j[\mathsf{Duracion}]$.

De la misma manera, es posible también definir preferencias compuestas. La composición priorizada $\mathbb{C}_1 \rhd \mathbb{C}_2$ *se puede expresar como:* $t_i \succ_{\mathbb{C}_1 \rhd \mathbb{C}_2} t_j$*, si y solo si* $(t_i[\mathsf{Genero}] = \mathsf{romance} \wedge t_j[\mathsf{Genero}] = \mathsf{accion}) \vee (t_i[\mathsf{Genero}] \neq \mathsf{romance} \wedge t_i[\mathsf{Duracion}] > t_j[\mathsf{Duracion}]) \vee (t_j[\mathsf{Genero}] \neq \mathsf{accion} \wedge t_i[\mathsf{Duracion}] > t_j[\mathsf{Duracion}])$*. Teniendo en cuenta la relación* PELICULA *de la Figura 2.1, y bajo la composición priorizada* $\mathbb{C}_1 \rhd \mathbb{C}_2$*, tenemos que la tupla* t_3 *es preferida a* t_1 *y* t_2*, mientras que* t_2 *es preferida a* t_1.

PELICULA

	PeliculaID	Titulo	Genero	Director	Duracion
t_1	p1	X-Men: Apocalipsis	Ciencia Ficción	Bryan Singer	131
t_2	p2	Gladiador	Acción	Ridley Scott	155
t_3	p3	Titanic	Romance	James Cameron	195

Figura 2.1: Ejemplo de instancia de base de datos.

Como se detalla a lo largo de esta sección la combinación de preferencias se puede realizar de diferentes maneras. Un aspecto importante del marco presentado por Chomiski es que se pueden expresar preferencias complejas combinando cualquier número de relaciones aplicando gradualmente una secuencia de una misma operación de composición, como así también de una operación diferente. Sin embargo, la semántica de una composición n-aria, $n > 2$, dependerá de si los correspondientes operadores son asociativos. Por ejemplo, el autor muestra que mientras la composición priorizada es asociativa, pareto no lo es.

Cabe destacar que otro aspecto importante es el estudio realizado por Chomicki para determinar si los operadores de composición ya descriptos preservan las propiedades de las preferencias que combinan. El autor prueba que en la mayoría de los casos la composición de preferencias no define una relación de orden en particular (ver Figura 2.2). Es decir, no es posible asegurar que la relación que resulta de una operación de composición de preferencias mantenga las propiedades de las relaciones involucradas en dicha composición. En este sentido, en [Cho03] se exhiben varias situaciones en donde la relación de preferencia compuesta viola propiedades tales como la asimetría o la transitividad.

En la Figura 2.2 se ilustran las diferentes operaciones de composición introducidas en [Cho03], y además se muestran los resultados obtenidos por Chomicki respecto a los tipos de órdenes que son preservados por estas operaciones.

	Composición Priorizada	Composición Pareto	Composición Lexicográfica	Intersección	Unión	Diferencia
Orden Parcial Estricto	No	No	No	Si	No	No
Orden Débil	Si	No	Si	No	No	No
Orden Total	Si	No	Si	No	No	No

Figura 2.2: Órdenes preservados por las diferentes operaciones de composición.

Note que las conclusiones ilustradas en la Figura 2.2 serán de interés más adelante en

el Capítulo 7 donde se retomará sobre la combinación de preferencias. En dicho capítulo se introduce un formalismo para combinar preferencias diferente a los presentados en este capítulo. En particular, el formalismo a introducir se desarrollará dentro de un contexto argumentativo que es el área inherente a las contribuciones de esta tesis.

2.2.2. Marco para combinar preferencias [BDRS15]

Answer set programming (*ASP*) [MT98] es una forma de programación declarativa basada en la semántica de modelos estables (answer set) [GL91] de la programación lógica, que ha sido útil para resolver una gran variedad de problemas de *Inteligencia Artificial* [NBG$^+$01, BCVF06, HKLS15]. En el paradigma *ASP* las teorías lógicas sirven como especificación del problema y sus soluciones son capturadas mediante modelos estables. Básicamente la metodología predominante en *ASP* consiste en:

1. describir un dominio o problema utilizando reglas lógicas.
2. utilizar restricciones para eliminar soluciones (modelos estables).
3. computar los modelos estables que corresponden a soluciones del problema.
4. elegir una solución maximal entre las soluciones generadas.

Una cuestión importante para muchas aplicaciones es la representación y el razonamiento sobre preferencias. Siguiendo esta dirección, diversos trabajos [Bre04b, SP06, GKS11, BDRS15] han realizado varios avances en la representación y manejo de preferencias complejas entre soluciones de un problema (modelos estables). En esta sección se describirá en particular el trabajo realizado por Brewka, Delgrande, Romero y Schaud [BDRS15] en donde se propone un enfoque general para implementar combinaciones complejas de preferencias cualitativas y cuantitativas entre modelos estables de un programa lógico.

Desde un punto de vista abstracto, en [BDRS15] los autores capturan los modelos estables preferidos de un programa lógico P mediante una relación de preferencia definida como un orden parcial estricto $\succ$ sobre dichos modelos. En otras palabras, un modelo estable X de P es $\succ$-$preferido$, si y solo si no existe otro modelo estable Y tal que $Y \succ X$. En *asprin* se utiliza la noción de *tipo de preferencia* como una clase de relaciones de preferencia. En el formalismo las preferencias se declaran mediante una *declaración de preferencia* de la siguiente forma:

$$\#preference(s,t)\{e_1,e_2,\ldots,e_n\}$$

De esta manera, la sintaxis de una preferencia se define mediante s y t que son términos fijos dando el nombre y tipo de preferencia, y un conjunto de argumentos donde cada elemento e_j es un *elemento de preferencia*. Un elemento de preferencia tendrá la forma:

$$\Phi_1 > \ldots > \Phi_m \| \Phi$$

donde Φ_r (r = 1,...,m y $m \geq 1$) es un conjunto de fórmulas proposicionales con peso $w_1, w_2, \ldots, w_l : \phi$ tal que cada w_i es un término y ϕ es una fórmula booleana sobre el alfabeto predefinido $\mathcal{A}$ y Φ es una fórmula sin peso. Intuitivamente, cada elemento e_j es un conjunto de fórmulas proposicionales que pueden tener un peso (*e.g.,* 40), estar ordenadas (mediante >), condicionadas (a través de ‖), o vinculadas al predicado *name*. El predicado *name* permite hacer referencia a preferencias auxiliares, expresando de esta manera una preferencia compuesta. Es decir, un *átomo con denominación* $name(s)$, se refiere a las relaciones asociadas con una declaración de preferencia s. A continuación, se muestran algunos de los ejemplos de declaraciones de preferencia introducidos en [BDRS15].

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

$\#preference(fun, subset)\{sauna, dive, hike, \neg bunji\}$

$\#preference(all, pareto)\{name(costs), name(fun)\}$

$\#optimize(all)$

Como se puede observar, el conjunto de declaraciones de preferencias estará siempre acompañado de una simple directiva de optimización, $\#optimize(s)$, la cual le indica al solucionador de problemas que restrinja su modo de razonamiento a la preferencia declarada por s. De esta manera, se llamará *especificación de preferencia* a un conjunto de declaraciones de preferencias S junto con una directiva de optimización $\#optimize(s)$, donde s es la declaración de preferencia primaria de la especificación de preferencia, mientras que las demás serán declaraciones secundarias.

En una declaración de preferencia, el tipo de preferencia es utilizado para definir relaciones de preferencias específicas y determina el conjunto de elementos de preferencia admisible. Por ejemplo, en las declaraciones descriptas anteriormente el tipo de preferencia *subset* está sujeto a literales sin un peso asociado, mientras que, en el tipo *less(weight)* los literales tienen asociado un peso determinado. Análogamente, el tipo de preferencia podría

requerir o no del predicado *name*, dependiendo de su naturaleza; es decir, si éste tiene naturaleza primitiva o compuesta. Por ejemplo, el tipo de preferencia *subset* es primitivo, mientras que *pareto* es compuesto. Por otra parte, cada declaración de preferencia s está asociada con un orden parcial estricto $\succ_s$. Si la declaración de preferencia s no está sujeta a optimización, es decir no se da $\#optimize(s)$, entonces ésta podría servir como elemento auxiliar en una declaración de preferencia compuesta.

Una declaración de preferencia expresa una relación de preferencia obtenida instanciando el tipo de preferencia de la relación con una interpretación específica de los elementos de preferencia. Formalmente hablando, a partir de un alfabeto predefinido $\mathcal{A}$, un tipo de preferencia t se define como una función que mapea un conjunto de elementos de preferencia E a una relación de preferencia $t(E)$ sobre modelos estables. Por otra parte, con $dom(t)$ se específica el dominio de t estableciendo los elementos de preferencia admisibles para t. Consecuentemente, se dice que una declaración de preferencia $\#preference(s,t)E$ es admisible, si $E \in dom(t)$. A continuación, se ilustrará cómo diferentes tipos de preferencias son utilizados para definir relaciones de preferencia específicas.

- *subset* se define mediante
  - $dom(subset) = \mathcal{P}(\{a, \neg a \mid a \in \mathcal{A}\})$ donde $\mathcal{P}(S)$ es el conjunto potencia de S, y $(X,Y) \in subset(E)$, si y solo si $\{l \in E \mid X \models l\} \subset \{l \in E \mid Y \models l\}$
- *pareto* se define a través de
  - $dom(pareto) = \mathcal{P}(\{n \mid n \in N\})$, donde N es un conjunto de átomos $name(s)$, y $(X,Y) \in pareto(E)$ si y solo si $\bigwedge_{name(s)\in E}(X \ \succeq_s \ Y) \wedge \bigvee_{name(s)\in E}(X \succ_s Y)$

Como se puede observar, al aplicar un tipo de preferencia a un conjunto admisible de elementos de preferencia se obtiene una relación de preferencia $\succ_s$. Para una mayor ilustración considere las relaciones de preferencia inducidas a partir de las siguientes declaraciones de preferencias:

- $\#preference(1, less(weight))\{1:a, 2:\neg b, 3:c\}$ declara que

  $X \succ_1 Y$ si $\sum_{(w:l)\in\{1:a,2:\neg b,3:c\}, X\models l} w > \sum_{(w:l)\in\{1:a,2:\neg b,3:c\}, Y\models l} w$
- $\#preference(2, subset)\{a, \neg b, c\}$ declara que

  $X \succ_2 Y$ si $\{l \in \{a, \neg b, c\} \mid X \models l\} \subset \{l \in \{a, \neg b, c\} \mid Y \models l\}$

- $\#preference(3, pareto)\{name(1), name(2)\}$ declara que

  $X \succ_3 Y$ si $(X \succeq_1 Y) \wedge (X \succeq_2 Y)$ y $(X \succ_1 Y) \vee (X \succ_2 Y)$

Observe que las preferencias compuestas son formadas por agregación, y se obtienen mediante el predicado *name*, el cual se utiliza para hacer referencia a preferencias auxiliares. En este sentido, *subset* es un tipo de preferencia primitivo ya que no hace referencia a una preferencia auxiliar, mientras que *pareto* es un tipo de preferencia compuesta.

En [BDRS15], los programas lógicos sobre un conjunto $\mathcal{A}$ de átomos se denominan *programas base*. Para implementar preferencias entre modelos estables de programas base, los tipos de preferencia son implementados mediante programas lógicos denominados *programas de preferencias*. Por otra parte, los autores proponen utilizar un conjunto de hechos para representar internamente una declaración de preferencia; de esta manera, para implementar una declaración de preferencia de la forma:

$$\#preference(s, t)\{e_1, e_2, \ldots, e_n\}$$

se define la siguiente traducción:

$$F_s = \{preference(s, t_s)\} \cup \bigcup_{j=1}^{n} F_{s,j}$$

donde $F_{s,j}$ denota el conjunto de hechos obtenidos para todas las fórmulas con peso y átomos con denominación contenidos en s. Para cada fórmula de peso de la forma $w_1, w_2, \ldots, w_l : \phi$ que aparece en algún conjunto Φ_r de un elemento de preferencia e_j en la declaración de preferencia s, existe un hecho de la forma

$$preference(s, j, r, for(t_\phi), (w_1, w_2, \ldots, w_l)).$$

donde cada w_i representa un término en la fórmula proposicional de e_i; t_ϕ es un término representando la fórmula booleana ϕ utilizando los símbolos de función $neg/1$, $and/2$, y $or/2$; y los índices j y r identifican a los respectivos componentes estructurales e_j y Φ_r. Para representar la condición de e_j, r toma el valor 0. Por otra parte, un átomo con denominación $name(s)$ se representa de la misma forma, excepto que se reemplaza $for(t_\phi)$ por $name(s)$. De manera similar, para implementar la directiva de optimización $\#optimize(s)$ se agrega $optimize(s)$ a los hechos de F_s.

Teniendo en cuenta esta representación, las declaraciones de preferencia junto con la directiva de optimización mostradas previamente se pueden traducir de la siguiente manera:

- $\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

  $preference(costs, 1, 1, for(sauna), (40)).$

  $preference(costs, 2, 1, for(dive), (70)).$

  $preference(costs, less(weight)).$

- $\#preference(fun, subset)\{sauna, dive, hike, \neg bunji\}$

  $preference(fun, 1, 1, for(sauna), ()).$

  $preference(fun, 2, 1, for(dive), ()).$

  $preference(fun, 3, 1, for(hike), ()).$

  $preference(fun, 4, 1, for(\neg bunji), ()).$

  $preference(fun, subset).$

- $\#preference(all, pareto)\{name(costs), name(fun)\}$

  $preference(all, 1, 1, name(costos), ()).$

  $preference(all, 2, 1, name(fun), ()).$

  $preference(all, pareto).$

- $\#optimize(all)$

  $optimize(all).$

El propósito de un programa de preferencia es decidir si un modelo es estrictamente preferido a otro. A partir de esto surge la pregunta de cómo representar los modelos estables con el objetivo de poder compararlos. Para abordar esto, en *asprin* se utilizan las capacidades de meta interpretación de ASP y se los representa mediante los predicados $holds/1$ y $holds'/1$. Por lo tanto, para el conjunto de átomos $X \subseteq \mathcal{A}$ se tienen los siguientes conjuntos:

$H_X = \{holds(a) | a \in X\}$ *y* $H'_X = \{holds'(a). | a \in X\}$

$R_X = \{holds(a) \leftarrow a | a \in X\}$ *y* $R'_X = \{holds'(a) \rightarrow a | a \in X\}$

$G_X = \{holds(a) \leftarrow |a \in X\}$ *y* $G'_X = \{holds'(a) \rightarrow |a \in X\}$

Como se ha observado *asprin* provee la traducción de una declaración de preferencia s de tipo t y los predicados $holds/1$ y $holds'/1$ para cada par de modelos estables X e Y que son comparados. Por otra parte, para implementar un tipo de preferencia t se utiliza una codificación E_t basada en *ASP*. En E_t se define bajo qué condiciones un modelo es estrictamente preferido sobre otro. Para esto, se define el predicado $better/1$, donde $better(s)$ es verdadero si y solo si $X \succ_s Y$. A partir de esto, por ejemplo el tipo de preferencia *subset* puede ser implementado como:

$$better(P) \text{ :- } preference(P, subset),$$
$$1\ \#sum\{1, X : not\ holds(X) : holds'(X), preference(P, \_, \_, for(X), \_)\},$$
$$holds'(X) : preference(P, \_, \_, for(X), \_), holds(X).$$

Para implementar los tipos de preferencia compuestos también se define un predicado $better/1$, pero en este caso la implementación se basa en predicados que deben ser definidos por otros tipos de preferencias. Por ejemplo, el tipo de preferencia *pareto* se puede implementar mediante el siguiente programa de preferencia:

$$better(P) \text{ :- } preference(P, pareto),$$
$$better(Q), preference(P, \_, \_, name(Q), \_),$$
$$bettereq(R), preference(P, \_, \_, name(R), \_).$$

En particular, el programa para el tipo compuesto *pareto* usa los predicados $better/1$ y $bettereq/1$ representando las relaciones $\succ$ y $\succeq$, respectivamente. Estos predicados deben estar definidos por las implementaciones de los tipos de preferencias de las declaraciones identificadas con el predicado *name*.

Note que los programas de preferencias utilizarán un conjunto de reglas auxiliares, C, que describen conceptos internos independientes del problema como la satisfacción de las expresiones booleanas. Este conjunto también contiene la restricción de integridad:

$$\text{:- } not\ better(P), optimize(P).$$

asegurando que $better(P)$ se cumple mientras P esté sujeto a optimización.

A partir de todo lo explorado, se define $F_s \cup E_t \cup C$ como un programa de preferencia para s, si para todo $X, Y \subseteq \mathcal{A}$, tenemos que $X \succ_s Y$ si y solo si $F_s \cup E_t \cup C \cup H_X \cup H'_Y$ es

satisfacible. Note que los programas de preferencias se refieren a conjuntos de átomos y por lo tanto son independiente de los programas base, en principio. Esta situación cambia al momento de considerar un programa junto a reglas, como R_X o G_X (en lugar de H_X).

Ejemplo 2.2. *Considere el siguiente programa base:*

gr(1..5).

1 { m(1) ; m(2) ; m(3) } 1.

a(1) :- m(1).

b(1..5) :- m(1).

a(1..2) :- m(2).

b(1) :- m(2).

a(5) :- m(3).

b(2..5) :- m(3).

A partir de este programa se pueden obtener tres modelos estables:

$M1 = \{m(1), a(1), b(1), b(2), b(3), b(4), b(5)\}$

$M2 = \{m(2), a(1), b(2), b(1)\}$

$M3 = \{m(3), a(5), b(2), b(3), b(4), b(5)\}$

En lo que sigue se buscan los modelos estables del programa base que son subconjuntos minimales con respecto al predicado $a/1$*. Considere la siguiente declaración de preferencia.*

$\#preference(p1, subset)\{a(X) : gr(X)\}.$

$\#optimize(p1).$

La primer sentencia define una declaración de preferencia con denominación $p1$ *y de tipo subset. Intuitivamente, la declaración de preferencia define una preferencia de tipo subset sobre átomos de predicado* $a/1$*. La última sentencia es una directiva de optimización que le indica a asprin que debe computar los modelos estables óptimos según* $p1$*. A partir del programa base detallado arriba, la declaración de preferencia se podría escribir como:*

$\#preference(p1, subset)\{a(1), a(2), a(3), a(4), a(5)\}.$

$\#optimize(p1).$

Al momento de computar un modelo estable óptimo, M2 es el primer modelo estable encontrado por asprin. Luego, asprin busca un modelo estable que sea preferido a M2 y encuentra que $M1 \succ M2$ *dado que se cumple que* $\{a(1)\} \subset \{a(1), a(2)\}$*. Por último, asprin busca un modelo estable que sea preferido a M1. Al no encontrarlo, provee a M1 como modelo óptimo.*

Como hemos mencionado previamente, además de [BDRS15] existen otros trabajos en relación a *ASP* que abordan el manejo de preferencias en sus formalismos [DSTW04, CFP10, CGFZ13]. Sin embargo, el sistema *asprin* cuenta con algunos aspectos que lo hacen particularmente interesante respecto a otros encontrados en la literatura. Por un lado, en [BDRS15] los autores muestran que muchos de los enfoques propuestos en la literatura [SNS02, BNT03, SP06, RGM10] pueden ser implementados utilizando *asprin*. Esto se debe principalmente a la generalidad y flexibilidad con la cual este sistema ha sido desarrollado. Por otra parte, el formalismo presentado en *asprin* extiende las ideas tomadas de [BNT03, GKS11, GM12] ya que los métodos propuestos en estos artículos están definidos para tipos específicos de preferencia. *Asprin* puede manejar cualquier tipo de preferencia definida para *ASP*. Finalmente cabe mencionar que en [Bre04a] se presentó un enfoque similar al propuesto en [BDRS15] ya que se describe un lenguaje de preferencia específico con un conjunto predefinido de relaciones de preferencias y métodos de combinación. Sin embargo, el formalismo presentado es un lenguaje fijo y carece de la flexibilidad propuesta por *asprin*.

2.3. Conclusión

En este capítulo se presentó una vista general de la noción de preferencia, y además se mostraron las propiedades que ésta puede asumir. Luego se resumieron y analizaron en particular dos formalismos para representar y combinar preferencias vinculados a dos áreas específicas de investigación: Base de Datos y Anwer Set Programming.

El primer enfoque, presentado en [Cho03], introduce un marco para formular preferencias del usuario utilizando fórmulas de primer orden. El objetivo principal de este trabajo es proveer un lenguaje que permita incorporar preferencias en las consultas a las base de

datos. Note que dentro de las contribuciones de este trabajo se encuentra el estudio de diferentes operadores de composición. En particular, estos operadores representan varias formas de combinar preferencias. Dentro de las operaciones de composición de preferencias descriptas se distinguen: las composiciones booleanas (unión, intersección, diferencia), composición priorizada, lexicográfica, y pareto. Un aspecto interesante de este trabajo es el estudio realizado por el autor sobre las propiedades que preservan estas operaciones. Estos resultados serán de utilidad en el Capítulo 7.

El segundo enfoque, propuesto en [BDRS15], presenta un marco basado en *ASP* para expresar e implementar preferencias entre modelos estables de un programa lógico. En particular, el formalismo permite especificar un rango de relaciones de preferencia complejas mediante el uso de diferentes tipos de preferencia. Para implementar un tipo de preferencia determinado, los autores proponen una especificación basada en codificación *ASP*, el trabajo explora varias de estas implementaciones.

Cabe destacar que si bien estos dos enfoques fueron desarrollados en áreas de investigación diferentes y con objetivos distintos, ambos se caracterizan por su flexibilidad y generalidad al momento de expresar preferencias del usuario. Siguiendo esta misma línea, en el Capítulo 7 se introducirá un formalismo vinculado a la programación lógica rebatible que provee herramientas computacionales concretas para expresar y combinar preferencias.

Capítulo 3

Preferencias en Sistemas Argumentativos Estructurados

En este capítulo se introducirán los conceptos que caracterizan a los sistemas argumentativos. Luego se describirán algunos de los sistemas argumentativos relevantes, enfatizando en aquellos sistemas argumentativos que utilizan una lógica subyacente para la representación de conocimiento (*Sistemas Argumentativos Estructurados*). Como el manejo de preferencias es uno de los pilares de esta tesis, este capítulo se centra en aquellos sistemas argumentativos que incluyen en su formalismo algún mecanismo para comparar argumentos. Finalmente, se discutirán algunos aspectos relacionados al criterio para comparar argumentos empleado por estos sistemas.

3.1. Introducción

Una característica de los formalismos de argumentación es que permiten definir un tipo de razonamiento en el cuál puede haber información contradictoria, incompleta, e incierta. En las últimas décadas, la argumentación ha evolucionado como un atractivo paradigma para conceptualizar el razonamiento de sentido común. Esto produjo como resultado la formalización de diferentes marcos de argumentación abstracta como [Dun95, AC98, BGG05], entre otros; y de sistemas argumentativos estructurados como [SL92, PS97, GS04, DKT06, AK07, Wak10, Pra10, Wak11].

Para describir las etapas del proceso de argumentación en un sistema argumentativo, donde el uso de preferencias juega un rol importante, se presentan a continuación los principales elementos conceptuales que caracterizan a estos sistemas. En [PV02] se presenta

una estructura conceptual dentro de la cual se pueden caracterizar la mayoría de los sistemas argumentativos existentes. De acuerdo a este marco conceptual, generalmente, un sistema argumentativo presenta cinco elementos (en algunos casos de manera implícita).

- Un **lenguaje lógico de representación** subyacente, utilizado para representar la información del dominio en el que se basará la argumentación. Asociado a este lenguaje se establece la noción de consecuencia, central para la construcción de los argumentos. Algunos sistemas argumentativos adoptan una lógica particular [GS04, BH01], mientras que otros dejan la lógica subyacente parcial o completamente sin especificar. Estos últimos pueden ser instanciados con diferentes lógicas alternativas, por lo que son considerados *marcos argumentativos* [Pra10].

- La **construcción de argumentos**, los cuales constituyen "pruebas" a partir de la lógica subyacente y el operador de consecuencia mencionado en el punto anterior de la estructura conceptual. En general, en la literatura un argumento se construye a partir de un árbol de derivación [SL92, PS97, GS04]. A nivel de representación, un argumento se suele especificar como un par *premisas-conclusión*, dejando implícito el hecho de que existe una derivación para la conclusión a partir de las premisas [GS04]. Algunos formalismos se abstraen de cómo es la construcción y la estructura interna de los argumentos. El marco de Dung [Dun95] es un ejemplo de estos formalismos, ya que se abstrae tanto del formalismo subyacente como de la estructura interna de los argumentos.

- El **ataque entre argumentos** que aparece en algún sentido en la argumentación, el cual se captura mediante la noción de conflicto entre argumentos, también llamada contra-argumentación o ataque. Generalmente los conflictos que puedan surgir son inferidos a través de las características del lenguaje formal subyacente. En los marcos como [Dun95], donde hay abstracción del lenguaje, se asume que los conflictos entre los argumentos se encuentran preestablecidos de antemano.

- La **derrota entre argumentos**, necesaria para determinar qué argumento será aceptado entre un par de argumentos en conflicto. La noción de derrota se constituye en la evaluación comparativa de pares de argumentos en conflicto para determinar si un determinado ataque tiene éxito o no. Esta noción de *ataque exitoso* se formaliza mediante una relación binaria entre argumentos, comúnmente denominada derrota. En la comunidad de argumentación no hay un consenso establecido acerca de qué

criterio utilizar para comparar los argumentos. Aún así, algunas propuestas han tomado la postura de utilizar un criterio fijo en el mismo formalismo del sistema, sin proveer ningún mecanismo para cambiarlos [SL92, PS97, Wak10]. Sin embargo, otros formalismos optaron por una alternativa más flexible permitiendo ajustar el sistema al criterio que más se adecua al dominio de aplicación que está siendo representado [GS04].

- La **aceptabilidad de argumentos**, es decir deteminar aquellos argumentos que serán finalmente aceptados por el sistema de argumentación. Para esto será necesario analizar las derrotas que haya entre los argumentos. No obstante, no alcanza con solo considerar las derrotas directas al argumento bajo análisis, sino que también es necesario considerar las derrotas para sus derrotadores, las derrotas para los derrotadores de sus derrotadores, y así sucesivamente. Existen muchas variantes acerca de cómo calcular la aceptabilidad de los argumentos de un sistema de argumentación; en particular, éstas se pueden dividir en dos categorías: las aproximaciones declarativas y las aproximaciones procedurales. En las aproximaciones declarativas se proveen ciertas condiciones que un conjunto de argumentos aceptables debe cumplir, sin especificar cómo se computarán tales conjuntos; estos métodos son normalmente conocidos como semánticas de aceptabilidad. Por otra parte, las aproximaciones procedurales brindan un procedimiento para determinar si un argumento determinado es aceptable o no. En general, estos métodos suelen simular el juego argumentativo que ocurre en una discusión, y se modela a través de árboles de argumentos conocidos como árboles de dialéctica.

Dado que los aportes de esta tesis están asociados al manejo de preferencias, este capítulo se centra en aquellos sistemas argumentativos que incluyen en su formalismo algún mecanismo para comparar argumentos. En particular, se enfatizará sobre los sistemas argumentativos estructurados (*SAEs*) que usan preferencias. Por lo general, en estos sistemas la preferencia entre argumentos se puede computar de diferentes maneras, claramente esto dependerá de la estructura y origen que tengan los argumentos. Por lo tanto, como la comparación entre argumentos se puede definir de diversas formas, a partir de ahora se asumirá que existe una relación de preferencia genérica sobre argumentos, denotada $\succsim$. Así, si el argumento $\mathcal{A}$ es al menos tan preferido como $\mathcal{B}$, se lo denotará como $\mathcal{A} \succsim \mathcal{B}$, o bien como par ordenado $(\mathcal{A}, \mathcal{B})$.

Definición 3.1 (Relación de Preferencia entre Argumentos)**.** *Sea Args un conjunto de argumentos. Una relación de preferencia entre argumentos* $\succsim \subseteq Args \times Args$ *es una relación binaria sobre Args. Se denotará:*

(a) $\mathcal{A} \succ \mathcal{B}$ *si* $\mathcal{A} \succsim \mathcal{B}$ *y* $\mathcal{B} \not\succsim \mathcal{A}$*, y se lee como "*$\mathcal{A}$ *es estrictamente preferido a* $\mathcal{B}$ *bajo* $\succsim$*".*

(b) $\mathcal{A} \approx \mathcal{B}$ *si* $\mathcal{A} \succsim \mathcal{B}$ *y* $\mathcal{B} \succsim \mathcal{A}$*, y se lee como "es indiferente preferir* $\mathcal{A}$ *o* $\mathcal{B}$ *bajo* $\succsim$*".*

(c) $\mathcal{A} \bowtie \mathcal{B}$ *si* $\mathcal{A} \not\succsim \mathcal{B}$ *y* $\mathcal{B} \not\succsim \mathcal{A}$ *y se lee como "*$\mathcal{A}$ *y* $\mathcal{B}$ *son incomparables bajo* $\succsim$*".*

En las siguientes secciones, se mostrarán algunos *SAEs* que incorporan la noción de preferencia a sus formalizaciones. En particular, se describe cómo estos sistemas hacen para computar la relación de preferencia en el proceso de evaluación de argumentos.

3.2. Sistema Argumentativo de Simari y Loui

En [SL92] se introduce uno de los primeros sistemas argumentativos que dieron un tratamiento matemático formal a la argumentación rebatible. De hecho, este trabajo también fue el primero en este tipo de sistemas en introducir la noción de preferencia entre argumentos. La propuesta de Simari y Loui combina la noción de especificidad definida por D. Poole [Poo85] con la teoría de justificación presentada por J. Pollock. Los autores postulan que Poole define un criterio basado en especificidad, pero no describe adecuadamente cómo aplicar este criterio a las distintas interacciones entre argumentos, mientras que Pollock especifica correctamente el proceso de justificación de argumentos, pero no define un criterio de preferencia.

El sistema formal está compuesto por un lenguaje de primer orden $\mathcal{L}$. Las reglas de inferencia de $\mathcal{L}$ son *modus ponens* y *generalización* y puede utilizarse cualquier axiomatización completa [Dav89] que resulte conveniente. Los autores definen además la noción de reglas rebatibles $\alpha \succ\!\!-\beta$, donde α y β son fórmulas bien formadas de $\mathcal{L}$. La expresión $\alpha \succ\!\!-\beta$ se puede interpretar como *típicamente las razones para creer en el antecedente* α *proveen razones para creer en el consecuente* β. Se asume que todo nombre de variable que aparece en ambos lados de una regla rebatible "r" corresponde al mismo término y las instancias de "r" se obtienen reemplazando consistentemente todas las variables libres por términos básicos de $\mathcal{L}$.

El conjunto de sentencias de $\mathcal{L}$, denotado con $Sent(\mathcal{L})$, puede ser particionado en el subconjunto de información *necesaria*, denotado por $Sent_N(\mathcal{L})$ y compuesto por senten-

cias con variables libres, y el subconjunto de información *contingente*, denotado como $Sent_C(\mathcal{L})$ y formado por sentencias básicas.

El conocimiento se representa con el par $(\mathcal{K}, \Delta)$, donde $\mathcal{K}$ es un subconjunto de $Sent(\mathcal{L})$ representando el conocimiento no rebatible y Δ es un conjunto de reglas rebatibles expresando conocimiento rebatible. El conjunto $\mathcal{K}$ puede ser particionado en dos subconjuntos:

$$\mathcal{K}_n = Sent_N(\mathcal{L}) \cap \mathcal{K}, \ y$$

$$\mathcal{K}_c = Sent_C(\mathcal{L}) \cap \mathcal{K},$$

tal que $\mathcal{K} = \mathcal{K}_n \cup \mathcal{K}_c$. Es importante destacar que el conjunto $\mathcal{K}$ representa información irrefutable, por lo tanto debe ser consistente, *i.e.,* $\mathcal{K} \nvdash \bot$. Sea A un elemento de $Sent(\mathcal{L})$ y $\Gamma = \{A_1, A_2, \ldots, A_n\}$ un conjunto formado por elementos que pertenecen a $\mathcal{K}$ o instancias básicas de las reglas en Δ. Se dice que una fórmula bien formada A es una *consecuencia rebatible* del conjunto Γ (denotado por $\Gamma \mathrel{|\!\sim} A$) si y sólo si existe una secuencia $B_1, B_2, \ldots, B_m$, denominada *derivación rebatible*, tal que $A = B_m$ y para cada uno de los B_i, $1 \leq i \leq m-1$, se cumple que B_i es un axioma de $\mathcal{L}$ o $B_i \in \Gamma$ o B_i es consecuencia directa de alguno de los elementos anteriores en la secuencia utilizando *modus ponens* o instanciación de una sentencia cuantificada universalmente. Las instancias básicas de las reglas rebatibles son consideradas como implicaciones materiales para la aplicación de *modus ponens*.

A continuación se utilizará el símbolo $\Delta^{\downarrow}$ para representar a todas las instancias de los elementos de Δ que se pueden obtener utilizando constantes individuales de $\mathcal{L}$. Una *estructura de argumento* es un par $\langle \mathcal{A}, L \rangle$, donde $\mathcal{A}$ es el soporte del argumento y L su conclusión. Sea $\mathcal{A} \subseteq \Delta^{\downarrow}$, $\mathcal{A}$ es un *argumento* para L si y solo si cumple con las siguientes propiedades:

- $\mathcal{K} \cup \mathcal{A} \mathrel{|\!\sim} L$.
- $\mathcal{K} \cup \mathcal{A}$ es consistente.
- $\nexists\, \mathcal{A}' \subset \mathcal{A}$ tal que $\mathcal{K} \cup \mathcal{A}' \mathrel{|\!\sim} L$.

Se dice que $\langle \mathcal{A}, L \rangle$ es un *subargumento* de $\langle \mathcal{A}_1, L_1 \rangle$ si y solo si $\mathcal{A}$ es un argumento y $\mathcal{A} \subseteq \mathcal{A}_1$.

A partir del soporte y la conclusión de un argumento, dos tipos de conflictos pueden aparecer. Sean las estructuras de argumento $\langle \mathcal{A}_1, L_1 \rangle$ y $\langle \mathcal{A}_2, L_2 \rangle$:

- $\langle \mathcal{A}_1, L_1 \rangle$ y $\langle \mathcal{A}_2, L_2 \rangle$ están en *desacuerdo* cuando sus conclusiones son contradictorias, *i.e.,* $\mathcal{K} \cup \{L_1, L_2\}$ es inconsistente.

- $\langle \mathcal{A}_1, L_1 \rangle$ es un *contra-argumento* de $\langle \mathcal{A}_2, L_2 \rangle$ cuando su conclusión contradice una fórmula utilizada en el soporte de $\langle \mathcal{A}_2, L_2 \rangle$. Formalmente existirá un subargumento $\langle \mathcal{A}_3, L_3 \rangle$ de $\langle \mathcal{A}_2, L_2 \rangle$ tal que $\mathcal{A}_3 \subseteq \mathcal{A}_2$ y $\mathcal{K} \cup \{L_1, L_3\}$ es inconsistente.

Para resolver el conflicto entre dos argumentos, el sistema establece una relación de derrota entre estos argumentos mediante un mecanismo de evaluación que determina cuándo un argumento es mejor que otro. Este mecanismo involucra el uso de un criterio de comparación de argumentos. De esta forma, el uso de un criterio de comparación permite extender naturalmente la noción de contra-argumentación incorporando la capacidad de decidir entre los argumentos en conflicto.

Se dice que un argumento puede ser activado si la premisa de cada regla en su soporte es verdadera. Esta noción permite definir la relación de especificidad [Poo85] entre dos argumentos. Intuitivamente, la especificidad favorece argumentos que contengan mayor información o sustenten su conclusión en forma más directa. Entonces, un argumento será más específico que otro si y solo si cada vez que el primero es activado el segundo también, pero no a la inversa. De esta manera, dos argumentos en conflicto pueden compararse mediante el criterio de especificidad.

Se dice que $\langle \mathcal{A}_1, L_1 \rangle$ *es estrictamente más específico* que $\langle \mathcal{A}_2, L_2 \rangle$, denotado como $\langle \mathcal{A}_1, L_1 \rangle \succ \langle \mathcal{A}_2, L_2 \rangle$, si se verifica que:

1. $\forall e \in Sent(\mathcal{L})$ tal que e activa no trivialmente a $\mathcal{A}_1$, *i.e.,* $\mathcal{K}_n \cup \{e\} \cup \mathcal{A}_1 \mid\sim L_1$ y $\mathcal{K}_n \cup \{e\} \nvdash L_1$, se verifica que e activa a $\mathcal{A}_2$, *i.e.,* $\mathcal{K}_n \cup \{e\} \cup \mathcal{A}_2 \mid\sim L_2$.

2. $\exists e \in Sent(\mathcal{L})$ tal que e no activa a $\mathcal{A}_1$, *i.e.,* $\mathcal{K}_n \cup \{e\} \cup \mathcal{A}_1 \not\mid\sim L_1$ y e activa no trivialmente a $\mathcal{A}_2$, *i.e.,* $\mathcal{K}_n \cup \{e\} \cup \mathcal{A}_2 \mid\sim L_2$ y $\mathcal{K}_n \cup \{e\} \nvdash L_2$.

Una sentencia e activa a un argumento $\langle \mathcal{A}_1, L_1 \rangle$ si es posible construir una derivación para L_1 a partir de $\mathcal{K}_n \cup \mathcal{A} \cup \{e\}$. A partir de las definiciones anteriores tenemos que $\langle \mathcal{A}_1, L_1 \rangle$ *derrota* a $\langle \mathcal{A}_2, L_2 \rangle$ si y solo si existe un subargumento $\langle \mathcal{A}, L \rangle$ de $\langle \mathcal{A}_2, L_2 \rangle$ tal que $\langle \mathcal{A}_1, L_1 \rangle$ contra-argumenta a $\langle \mathcal{A}, L \rangle$ en L y $\langle \mathcal{A}_1, L_1 \rangle$ es estrictamente más especifico que $\langle \mathcal{A}, L \rangle$.

Ejemplo 3.1. *El argumento* $\langle \{B(r) \wedge A(r) \succ\!\!\!- C(r)\}, C(r) \rangle$ *es más específico que* $\langle \{A(r) \succ\!\!\!- \neg C(r)\}, \neg C(r) \rangle$ *por que cada vez que el primer argumento se activa para so-*

portar $C(r)$ el segundo también lo hace para soportar la conclusión $\neg C(r)$. Por otra parte, $A(r)$ puede activar al segundo argumento pero no puede activar al primero.

Es importante destacar que el carácter modular de este formalismo permite de forma sencilla reemplazar el método de especificidad por algún otro método de comparación de argumentos que establezca un orden parcial entre los mismos, sin que el comportamiento global del sistema se vea afectado. Por otra parte, el criterio especificidad tiene la ventaja de ser sintáctico.

3.3. Sistema Argumentativo de Prakken y Sartor

Existen en la literatura de argumentación, sistemas que utilizan criterios para comparar argumentos que asumen un orden de prioridad explícito entre reglas de programa. Algunos de estos sistemas utilizan preferencias estáticas entre reglas, preferencias fijadas al momento de especificar la teoría de argumentación, o dinámicas, *i.e.,* preferencias derivadas como conclusiones dentro del sistema que argumenta [DSTW04]. Inspirados en el razonamiento legal, Prakken y Sartor [PS97] presentan un sistema argumentativo donde los conflictos entre argumentos son resueltos con una relación de prioridad entre reglas. En el formalismo desarrollado a continuación, los autores proponen dos enfoques: uno que utiliza prioridades fijas entre las reglas del programa, y otro con prioridades rebatibles.

3.3.1. Enfoque con prioridades fijas

En [PS97], el sistema desarrollado usa el lenguaje de la programación en lógica extendida. Por lo tanto, la información de entrada está constituida por una *teoría ordenada* $\Gamma = (S,D,>)$, donde S es un conjunto de reglas estrictas, D un conjunto de reglas rebatibles, y $>$ un orden parcial estricto sobre D. En el lenguaje, $\neg$ representa la negación clásica (o fuerte) mientras que $\sim$ codifica una clase de negación por falla (o débil). Las reglas son denotadas

$$r : L_0 \wedge \ \ldots \wedge L_j \wedge \sim L_k \wedge \ldots \wedge \sim L_m \ \Rightarrow L_n,$$

donde r es el nombre de la regla, y cada L_i $(0 \leq i \leq n)$ es un literal *fuerte* que puede estar precedido por el símbolo "$\neg$" de la *negación fuerte*, o un literal *débil* o *suposición* si toma la forma $\sim L_k$, donde L_k es un literal fuerte y"$\sim$" es el símbolo de la *negación débil*. La conjunción a la izquierda del conectivo "$\Rightarrow$" se denomina *antecedente* y el literal

L_n consecuente. Sintácticamente, las reglas estrictas difieren de las rebatibles en que no pueden contener suposiciones y están denotadas con el conectivo "$\rightarrow$" en lugar de "$\Rightarrow$".

Un *argumento* es una secuencia finita $\mathcal{A} = [r_1, r_2, \ldots, r_n]$ de instancias fijas de reglas que satisfacen las siguientes dos condiciones:

1. para cada i $(0 \leq i \leq n)$, y para cada L_j en el antecedente de r_i existe un $k < i$ tal que L_j es el consecuente de r_k, y

2. no existen dos reglas rebatibles en la secuencia $\mathcal{A}$ con el mismo consecuente.

Un argumento $\mathcal{A}'$ es un *subargumento* de $\mathcal{A}$ si y solo si $\mathcal{A}'$ es una subsecuencia de $\mathcal{A}$. En el caso que $\mathcal{A}' \neq \mathcal{A}$, $\mathcal{A}'$ es un *subargumento propio*. Cada consecuencia de una regla en $\mathcal{A}$ es una *conclusión* de $\mathcal{A}$, y cada literal L es una *suposición* de $\mathcal{A}$ si y solo si $\sim\overline{L}$ aparece en alguna regla en $\mathcal{A}$, donde $\overline{L}$ denota el complemento de L con respecto a la negación fuerte. Un argumento es estricto si y solo si no contiene reglas rebatibles, de lo contrario es rebatible.

La presencia de suposiciones en una regla da lugar a dos tipos de conflictos entre argumentos. Un argumento $\mathcal{A}_1$ *ataca* a otro argumento $\mathcal{A}_2$ si y solo si existen dos secuencias S_1 y S_2 de reglas estrictas tal que $\mathcal{A}_1 + S_1$[1] es un argumento con conclusión L, y se cumple que:

1. $\mathcal{A}_2 + S_2$ es un argumento con conclusión $\overline{L}$, o

2. $\mathcal{A}_2$ es un argumento con una suposición $\overline{L}$.

Habiendo definido cuándo un argumento está en conflicto con otro, lo siguiente consiste en describir un método para compararlos a fin de establecer la relación de derrota. A partir de la noción de ataque mencionada previamente, la relación de derrota se enuncia en términos de *socavamiento* (en inglés, *undercutting*) y *refutación* (en inglés, *rebutting*). Si dos argumentos en conflicto tienen conclusiones complementarias y se induce una preferencia de uno sobre el otro, entonces estamos en presencia de una derrota por *refutación*. En este caso las prioridades entre las reglas rebatibles D se deben considerar en la comparación de los argumentos.

La preferencia entre argumentos se basa en la comparación de los conjuntos de reglas rebatibles que contribuyen a la derivación de las conclusiones de los argumentos en conflicto, *i.e.,* aquellas reglas relevantes para el conflicto. Sea $\mathcal{A}$ un argumento y S un

[1]Si $\mathcal{A}$ es un argumento, y T una secuencia de reglas, entonces $\mathcal{A} + T$ denota concatenar $\mathcal{A}$ con T.

conjunto de reglas estrictas tal que $\mathcal{A} + S$ es un argumento con conclusión L. El conjunto de *reglas relevantes* a L en el argumento $\mathcal{A} + S$ se define de la siguiente manera:

- Si L es el consecuente de una regla rebatible en $\mathcal{A}$, entonces ésta es la única regla relevante.
- Si $\mathcal{A}$ es rebatible y L es el consecuente de una regla estricta, entonces el conjunto de reglas relevantes se obtiene a partir de la unión del conjunto de reglas relevantes a cada literal que se encuentra en el antecedente de dicha regla estricta.

Finalmente, el siguiente criterio de prioridad extiende el orden entre pares de reglas a pares de conjuntos de reglas. Dado dos conjuntos de reglas rebatibles R y R', $R > R'$ si y solo si, $\exists r' \in R'$ tal que $\forall r \in R$ se cumple que $r > r'$. Este criterio será utilizado para preferir un argumento sobre otro en la noción de refutación que se describe a continuación.

- Un argumento $\mathcal{A}_1$ *refuta* a un argumento $\mathcal{A}_2$ si y solo si los dos argumentos tienen conclusiones complementarias, y no se cumple que el conjunto de reglas relevantes R' a la conclusión del segundo argumento tenga prioridad sobre el conjunto de reglas relevantes R a la conclusión del primero, *i.e.,* $R' \not> R$.
- Un argumento $\mathcal{A}_1$ *socava* a un argumento $\mathcal{A}_2$ si el primero tiene una conclusión que es el complemento de una suposición del segundo.

Se dice que un argumento es *coherente* si no se ataca a sí mismo. Un conjunto de argumentos es *libre de conflicto* si no contiene argumentos que se ataquen mutuamente. En este punto se está en condiciones de formalizar la noción de derrota. Un argumento $\mathcal{A}_1$ *derrota* a $\mathcal{A}_2$ si y solo si:

1. $\mathcal{A}_1$ es un argumento vacío, y $\mathcal{A}_2$ no es coherente, o bien
2. $\mathcal{A}_1$ socava a $\mathcal{A}_2$, o bien
3. $\mathcal{A}_1$ refuta a $\mathcal{A}_2$ y $\mathcal{A}_2$ no socava a $\mathcal{A}_1$.

Se dirá que $\mathcal{A}_1$ *derrota estrictamente* a $\mathcal{A}_2$ si y solo si $\mathcal{A}_1$ derrota a $\mathcal{A}_2$ y $\mathcal{A}_2$ no derrota a $\mathcal{A}_1$.

Ejemplo 3.2. *A continuación se ilustra la relación de derrota entre los argumentos construidos a partir del siguiente conjunto de reglas:*

r_0: $\Rightarrow a$

r_1: $a \Rightarrow b$

r_2: $\sim b \Rightarrow c$

r_3: $\Rightarrow \neg a$

Si $r_3 > r_0$*, el argumento* $\mathcal{A}_1 = [r_0, r_1]$ *derrota por socavamiento a* $\mathcal{A}_2 = [r_2]$ *y* $\mathcal{A}_3 = [r_3]$ *derrota a* $\mathcal{A}_1$*, refutando a su subargumento propio* $\mathcal{A}_0 = [r_0]$.

A pesar de conseguir una solución apropiada al problema del conflicto entre argumentos, los autores manifiestan que asumir un orden fijo sobre las reglas es una alternativa que con frecuencia es poco realista en el razonamiento legal. Por ejemplo, en el razonamiento legal los estándares para decidir ante un conflicto están a su vez sujetos a un debate. Esto da lugar a un segundo enfoque, en el cual los autores redefinen el formalismo para permitir la construcción de argumentos sobre las prioridades. De esta forma, las prioridades son rebatibles y dejan de ser fijas.

3.3.2. Enfoque con prioridades rebatibles

En primer lugar, se extiende el lenguaje con un predicado especial $\rhd$, donde $r_2 \rhd r_1$ expresa que la regla r_2 tiene prioridad sobre r_1. En consecuencia ya no es necesario incluir al orden $>$ como el tercer componente de una teoría ordenada y ésta se redefine como un par $\Gamma = (S, D)$. Como $\rhd$ denota un orden parcial estricto, se agrega asimismo al conjunto de reglas estrictas S un conjunto de axiomas de manera tal que el orden sobre las reglas es un orden parcial estricto.

- $t_1 : (\mathrm{z} \rhd \mathrm{y}) \wedge (\mathrm{y} \rhd \mathrm{x}) \rightarrow (\mathrm{z} \rhd \mathrm{x})$
- $t_2 : (\mathrm{y} \rhd \mathrm{x}) \wedge \neg (\mathrm{z} \rhd \mathrm{x}) \rightarrow \neg (\mathrm{z} \rhd \mathrm{y})$
- $t_3 : (\mathrm{z} \rhd \mathrm{y}) \wedge \neg (\mathrm{z} \rhd \mathrm{x}) \rightarrow \neg (\mathrm{y} \rhd \mathrm{x})$
- $t_4 : (\mathrm{y} \rhd \mathrm{x}) \rightarrow \neg (\mathrm{x} \rhd \mathrm{y})$

Los autores mencionan que el orden de la teoría debe ser determinado por todos los argumentos justificados que contengan información sobre las prioridades, esto es, $(r, r') \in >$ si y solo si existe un argumento justificado para $r \rhd r'$. En consecuencia el ordenamiento

sobre las reglas se obtiene paso a paso junto con los argumentos justificados y varía a medida que se obtienen nuevas conclusiones.

Si *Args* es un conjunto de argumentos, entonces $>_{Args} = \{r > r' | r \rhd r'$ es una conclusión de algún $\mathcal{A}_1 \in Args\}$. Entonces se dirá que $\mathcal{A}_1$ (estrictamente) *Args-derrota* a $\mathcal{A}_2$ en base a una teoría ordenada $\Gamma = (S, D)$ si y sólo si $\mathcal{A}_1$ (estrictamente) derrota a $\mathcal{A}_2$ en base a $(S, D, >_{Args})$. En el segundo caso, la noción de derrota corresponde a la previamente definida para el enfoque con prioridades fijas.

Un argumento $\mathcal{A}$ será *aceptable* con respecto a un conjunto *Args* de argumentos si y solo si todos los argumentos *Args*-derrotantes para $\mathcal{A}$ son estrictamente *Args*-derrotados por algún argumento en *Args*. La definición de argumentos aceptables nos lleva a la noción de *argumentos justificados*, es decir aquellos argumentos que resultan ser aceptables luego de realizar todas las comparaciones posibles entre pares de argumentos.

En ambos enfoques presentados, el conjunto $JustArgs_\Gamma$ de todos los argumentos justificados para una teoría ordenada Γ se define en términos de un operador de punto fijo, denominado la *función característica* de la teoría ordenada. Si $\Gamma = (S, D)$ es una teoría ordenada, $T \subset Args_\Gamma$ y $Cargs_\Gamma$ es el conjunto de todos los subconjuntos libres de conflicto de $Args_\Gamma$, entonces la función característica de Γ, destinada al enfoque con prioridades rebatibles, se define como:

- $G_\Gamma : Cargs_\Gamma \longrightarrow 2^{Args_\Gamma}$
- $G_\Gamma(T) = \{\mathcal{A} \in Args_\Gamma |\ \mathcal{A}$ *es aceptable con respecto a* $T\}$

En el formalismo se demuestra que el operador G_Γ, al estar restringido a conjuntos de argumentos libre de conflicto, verifica la propiedad de monotonía. Esta propiedad asegura la existencia del menor punto fijo de G_Γ, que será utilizado para describir el conjunto de argumentos justificados. Por otra parte, los autores muestran como capturar la definición de $JustArgs_\Gamma$ de una manera constructiva, comenzando desde el conjunto vacío, definiendo la siguiente secuencia de subconjuntos de $Args_\Gamma$.

- $G^1 = G_\Gamma(\emptyset)$
- $G^{i+1} = G^i \cup G_\Gamma(G^i)$

Se puede demostrar que para un Γ finito, se cumple que $\bigcup_{i=0}^{\infty}(G^i) = JustArgs_\Gamma$.

Ejemplo 3.3. *Para ilustrar cómo razonar sobre las prioridades entre reglas se considera una teoría* (S, D) *tal que* S *contiene los axiomas de prioridad para asegurar las prioridades*

de transitividad y asimetría y D está compuesto por las reglas r_0, r_1, r_2, *y* r_3 *presentadas en el Ejemplo 3.2 más las siguientes reglas rebatibles:*

r_4: $\Rightarrow r_3 \rhd r_0$

r_5: $\Rightarrow r_0 \rhd r_3$

r_6: $\Rightarrow r_4 \rhd r_5$

$G^1 = \{[r_6]\}$ *y entonces* $>_{G^1} = \{r_4 > r_5\}$. *En la próxima iteración podemos resolver el conflicto entre* $[r_4]$ *y* $[r_5]$, *y* $[r_4]$ *estrictamente* G^1*-derrota a* $[r_5]$ *y en consecuencia* $G^2 = G^1 \cup \{[r_4]\}$. *Ahora,* $>_{G^2} = >_{G^1} \cup \{r_3 > r_0\}$.

Para concluir, es importante mencionar que la principal contribución de este trabajo yace en el uso de prioridades rebatibles. Esta característica posibilita debatir sobre el orden de preferencia entre los argumentos. Por otra parte, en cualquiera de los dos enfoques propuestos la fuerza de los argumentos se infiere desde la fuerza de las reglas con las cuales los argumentos están construidos. En particular, ésto hace que no se tenga en cuenta la totalidad del argumento, y en consecuencia el criterio de prioridad sobre reglas utilizado no funciona correctamente en gran variedad de situaciones.

3.4. Sistema Argumentativo de García y Simari

La Programación Lógica Rebatible (DeLP, por sus siglas en inglés) [Gar00, GS04] es un formalismo de representación de conocimiento y razonamiento que combina resultados de la programación en lógica y la argumentación rebatible, que ha sido aplicado exitosamente en diferentes dominios concretos (*e.g.,* [GGS08, RGS07, FEGS08, BBD$^+$14]). Por otro lado, el formalismo de DeLP es una evolución del sistema argumentativo de Simari y Loui [SL92] en donde se han mejorado varios aspectos, en particular la eficiencia computacional, adoptando una perspectiva basada en programación en lógica; en consecuencia, resulta un sistema de gran expresividad que permite representar información incompleta y potencialmente contradictoria en una forma efectiva. A diferencia de los sistemas argumentativos existentes en la literatura [Pra10, Wak14a], DeLP adopta como lenguaje lógico subyacente una extensión de la programación en lógica, para incluir negación fuerte y representar información rebatible (además de estricta). A partir de estos elementos se construyen argumentos, se identifican conflictos (considerando la negación fuerte) entre

los argumentos, se determinan las correspondientes derrotas, y se utilizan procedimientos de prueba dialéctica para determinar qué argumentos son aceptados.

El proceso de aceptación requiere tener definido un criterio que resuelva el conflicto entre dos argumentos, estableciendo cuándo un argumento es preferido a otro. A diferencia del trabajo presentado por Prakken y Sartor [PS97], una característica interesante de DeLP es la forma modular con la cual maneja los criterios de comparación de argumentos. Esta capacidad hace que aquellos sistemas que utilizan a DeLP como motor de razonamiento puedan configurar de antemano el criterio de preferencia entre argumentos que más se ajusta al dominio de aplicación.

Por lo mencionado anteriormente, el sistema DeLP resultará de particular interés en esta tesis. De hecho, se dedicará el Capítulo 4 a presentar la sintaxis de DeLP y algunas de sus principales características; en particular, este lenguaje será utilizado para el desarrollo de los modelos de servicios de razonamiento presentados en los capítulos siguientes.

3.5. Sistema Argumentativo de Amgoud y Kaci

La propuesta realizada por Amgoud y Kaci en [AK07] desarrolla un formalismo de argumentación para razonar con bases de conocimiento en conflicto. La propuesta trata con el problema de la inconsistencia que puede surgir al integrar múltiples fuentes de información. Como el uso de prioridades es una de las herramientas cruciales para resolver dichos conflictos, los autores asumen que las bases de conocimiento cuentan con prioridades que pueden estar implícitamente o explícitamente asociadas a la información. A partir de esto, en [AK07] se formaliza, por un lado un modelo de sistema argumentativo basado en lógica proposicional que maneja bases de conocimiento con prioridades implícitas; y, por el otro, para tratar con información con prioridades explícitas introducen un modelo basado en lógica posibilística que permite representar información priorizada mediante fórmulas proposicionales las cuales tienen asociado un peso.

El marco general de argumentación utilizado por los autores es el propuesto en [AC02], es decir, un marco de argumentación es una tupla $\langle Args, Def, \succsim \rangle$, donde $Args$ es un conjunto de argumentos, Def es una relación binaria que representa la relación de derrota entre los argumentos, y $\succsim$ es un preorden sobre $Args \times Args$. Note que las diferentes definiciones de $Args$ y Def da lugar a diferentes sistemas argumentativos; en particular, las nociones de argumento y derrota que se usan son las presentadas en [EH95]. Por otro lado, se introduce una relación de preferencia entre argumentos cuyo cómputo dependerá

del tipo de prioridad a tratar (implícita o explícita).

3.5.1. Prioridades implícitas

Como se mencionó antes, los autores proponen dos enfoques para tratar con la información contradictoria proveniente de múltiples fuentes de información: uno basado en prioridades implícitas y otro en prioridades explícitas. Para el primer caso donde se utilizan prioridades implícitas la información será modelada a partir de un lenguaje proposicional. Sea K una base proposicional, un *argumento* es un par $\langle A, h\rangle$, donde A es el soporte de la conclusión h, y se cumple que:

1. h es una fórmula del lenguaje,
2. $A \subseteq K$,
3. A es consistente,
4. $A \vdash h$
5. A es minimal (no existe un subconjunto de A que cumpla 1, 2, 3, 4).

La relación de derrota se define de la siguiente manera. Un argumento $\langle A_1, h_1\rangle$ *socava* a otro $\langle A_2, h_2\rangle$ si y solo si para algún $k \in A_2$, $h \equiv \neg k$. Un argumento está socavado si y solo si existe al menos un argumento en contra de un elemento de su soporte.

A continuación se describirá cómo obtener la relación de preferencia entre argumentos. Para este caso en donde se utilizan prioridades implícitas, el soporte de cada argumento tendrá una *fuerza* asociada. Esta fuerza corresponde a la distancia entre el soporte del argumento y las diferentes bases de conocimiento. Dicha distancia está basada en la distancia de Hamming [Dal88]. La distancia local entre una interpretación w y una base proposicional K_i es el número de átomos en los cuales esta interpretación difiere de algún modelo de la base proposicional, *i.e.,*

$$d(\omega, K_i) = min\{dist(\omega, \omega') \mid \omega' \in Mod(K_i)\},$$

donde $dist(\omega, \omega')$ es el número de átomos cuyas valuaciones difieren en las dos interpretaciones, y $Mod(K_i)$ es el conjunto de modelos de K_i. Para el caso donde existen varias bases proposicionales, la distancia completa se obtiene a partir de la agregación de las distancias locales utilizando un operador concreto de integración de bases. En esta

sección se mostrará un ejemplo en particular de un operador de integración $\circledast$ basado en distancia.

Entonces, siendo $E = \{K_1 \dots K_n\}$ el conjunto de bases a integrar, y $\langle A, h\rangle$ un argumento construido desde E, tal que el soporte de $\langle A, h\rangle$ se obtiene de $K_1 \cup \dots \cup K_n$, *i.e.,* $A \subseteq K_1 \cup \dots \cup K_n$. La *fuerza del soporte* de $\langle A, h\rangle$ se define como:

$$Force(A) = \circledast(\delta(A, K_1), \dots, \delta(A, K_n)),$$

siendo $\delta(A, K_i) = min\{dist(\omega, \omega') \mid \omega \vDash A \ y \ \omega' \ \vDash K_i \ (1 \leq i \leq n)\}$ la distancia minimal de A a la base K_i. La notación $\omega \vDash \phi$ expresa que la interpretación ω es un modelo de la fórmula ϕ.

De acuerdo a esta noción de fuerza es posible definir una relación de preferencia $\succ_{\circledast}$ para situaciones de información con prioridades implícitas. Dados dos argumentos $\langle A_1, h_1\rangle$ y $\langle A_2, h_2\rangle$ construidos desde el conjunto E, se dice que $\langle A_1, h_1\rangle$ es *preferido* a $\langle A_2, h_2\rangle$, denotado $\langle A_1, h_1\rangle \succ_{\circledast} \langle A_2, h_2\rangle$, si y solo si $Force(A_2) > Force(A_1)$. Dado un argumento $\langle A, h\rangle$, a continuación se muestra un ejemplo donde el operador de integración $\circledast$ basado en distancia se define como:

$$\circledast(\delta(A, K_1), \dots, \delta(A, K_n)) = \sum_{i=1}^{n} \delta(A, K_i)$$

Ejemplo 3.4. *Considere las bases* $K_1 = a, K_2 = a \to b$ *y* $K_3 = \neg b$*, y los argumentos* $\langle A_1, b\rangle$ *y* $\langle A_2, \neg b\rangle$*, donde* $A_1 = \{a, a \to b\}$ *y* $A_2 = \{\neg b\}$*.*

- $\delta(A_1, K_1) = \delta(A_1, K_2) = 0$*,* $\delta(A_1, K_3) = 1$*,*
- $\delta(A_2, K_1) = \delta(A_2, K_2) = \delta(A_2, K_3) = 0$*.*

Utilizando el operador $\circledast$*, se tiene que la fuerza de los soportes de los argumentos* A_1 *y* A_2*, es:*

$$Force(A_1) = \circledast(\delta(A_1, K_1), \dots, \delta(A_1, K_n)) = \sum_{i=1}^{3} \delta(A_1, K_i) = 1$$

$$Force(A_2) = \circledast(\delta(A_2, K_1), \dots, \delta(A_2, K_n)) = \sum_{i=1}^{3} \delta(A_2, K_i) = 0$$

Por lo tanto, el argumento A_2 *es preferido a* A_1 *ya que* $Force(A_1) > Force(A_2)$*.*

3.5.2. Prioridades explícitas

Para el caso de información con prioridades explícitas, la propuesta para integrar diferentes fuentes de información consiste en usar un marco basado en lógica posibilística el cual permite representar la información priorizada. A diferencia del modelo anterior en este enfoque se usan bases de conocimiento posibilísticas para construir los argumentos y se asume un operador $\oplus$ para integrar las bases posibilísticas. Se denotará con $B_\oplus$ a la base obtenida luego de aplicar $\oplus$ a un conjunto $B_1, B_2, \ldots, B_n$ de diferentes bases posibilísticas. La noción de fuerza también se aplica a este marco. Cada argumento tiene una fuerza intrínseca basada en el nivel de certidumbre de la información utilizada. De hecho, la fuerza de un argumento se establece a partir del grado de certeza de la información menos cierta que se encuentra en el argumento.

Una base posibilística de la forma $B = \{(\phi_i, a_i) : i = 1, \ldots, n\}$ es un conjunto de formulas lógicas proposicionales con peso donde $a_i \in [0, 1]$. Sea $B^* = \{\phi_i | (\phi_i, a_i) \in B\}$ y $\mathcal{A} = \langle A, h\rangle$ un argumento tal que $A \subseteq B^*$. La fuerza de $\mathcal{A}$ denotada por $Force(\mathcal{A})$, se define como:

$$Force(\mathcal{A}) = min\{a_i : \phi_i \in A \text{ } y \text{ } (\phi_i, a_i) \in B_\oplus\}$$

La fuerza de los argumentos provee una forma de comparar pares de argumentos, y elegir el mejor. El formalismo define una relación de preferencia $\succ_\oplus$ que encuentra preferencia en aquellos argumentos que utilizan creencias más ciertas. Sea *Args* un conjunto de argumentos obtenido a partir de una base $B_\oplus$, y $\mathcal{A}_1, \mathcal{A}_2 \in Args$, $\mathcal{A}_1$ será preferido a $\mathcal{A}_2$ con respecto a la relación de orden $\succ_\oplus$, denotado $\mathcal{A}_1 \succ_\oplus \mathcal{A}_2$, si y solo si $Force(\mathcal{A}_1) > Force(\mathcal{A}_2)$.

Ejemplo 3.5. *Considere la base posibilística* $B_\oplus = \{(\phi \vee \psi, .9), (\neg\phi, .7), (\xi \vee \psi, .6), (\neg\xi, .5)\}$. *Dos argumentos a favor de* ψ *pueden ser construidos:*

- $\mathcal{A}_1 = \langle\{\phi \vee \psi, \neg\phi\}, \psi\rangle$ *y*
- $\mathcal{A}_2 = \langle\{\xi \vee \psi, \neg\xi\}, \psi\rangle$

donde $\mathcal{A}_1$ *es preferido a* $\mathcal{A}_2$ *ya que* $Force(\mathcal{A}_1) = .7$ *mientras* $Force(\mathcal{A}_2) = .5$.

Para concluir, obsérvese que, a diferencia de aquellos enfoques que utilizan un criterio modular, en cada uno de los casos las autoras muestran una forma particular de comparar argumentos basada en la fuerza de los argumentos. En la siguiente sección se presenta un formalismo que utiliza un criterio diferente basado en comparación de literales.

3.6. Sistema Argumentativo de Wakaki

En [Wak10] se presenta un sistema de argumentación para manejar prioridades estáticas en un escenario de *ASP*. La propuesta presentada permite obtener un marco de argumentación no abstracto a partir de un programa lógico priorizado (PLP) [SI00]. Para asegurar la correctitud de su enfoque el autor muestra que las extensiones (conjuntos de argumentos aceptables) obtenidas del marco de argumentación asociado a un PLP dado, captura los conjuntos respuestas de este programa. Para esto último, el autor basa su trabajo en el estudio realizado por Dung [Dun93] que muestra que las extensiones estables de un marco de argumentación asociado a un programa lógico extendido (PLE) $\mathcal{P}$ dado captura los conjuntos respuestas de $\mathcal{P}$.

Un programa lógico priorizado (PLP) permite representar información de prioridad asociada. Mediante este tipo de programas es posible expresar una relación de preorden $\geq$ para establecer prioridades sobre un conjunto Lit de literales fijos. La prioridad $L_2 \geq L_1$ expresa que el literal L_2 tiene mayor o igual prioridad que L_1. Decimos que L_2 tiene prioridad estricta sobre L_1, denotado $L_2 > L_1$, si $L_2 \geq L_1$ y $L_1 \ngeq L_2$. Un PLP se define como un par ($\mathcal{P}$, Φ) donde $\mathcal{P}$ es un PLE [GL88, GL91, PS97] y Φ un conjunto de prioridades sobre $Lit_{\mathcal{P}}$.

Para introducir las formalizaciones de argumentación no abstracta los autores utilizan las nociones para $PLEs$ ya definidas en los trabajos [PS97, SS05]. De esta manera, en el formalismo se asume que una base de conocimiento se puede expresar como un PLE $\mathcal{P}$, cuyas reglas tienen la forma

$$L \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n,$$

donde L y L_i son literales o átomos precedidos por la negación clásica $\neg$. La negación por falla se denota con *not*. Cada regla tiene a L como la cabeza de la regla, denotado $head(r)$, y a $\{L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\}$ como el cuerpo de la regla, denotado $body(r)$. A partir de $\mathcal{P}$ varios argumentos podrían ser construidos. Un *argumento* [SS05] asociado a $\mathcal{P}$ es una secuencia finita $\mathcal{A}$= $[r_1, r_2, \ldots, r_n]$ de instancias fijas de reglas $r_i \in \mathcal{P}$, donde la cabeza de cada regla corresponde a una conclusión de $\mathcal{A}$, mientras que un literal *not* L en el cuerpo de una regla es una suposición de $\mathcal{A}$. Se escribe $assum(\mathcal{A})$ y $conc(\mathcal{A})$ para expresar el conjunto de suposiciones y conclusiones de $\mathcal{A}$, respectivamente. En especial se dice que la cabeza de la primera regla de un argumento $\mathcal{A}$ es la afirmación de dicho argumento, y se escribe $claim(\mathcal{A})$.

Un *subargumento* de $\mathcal{A}$ es una subsecuencia de $\mathcal{A}$ la cual es un argumento en si misma. Un argumento con una conclusión L es un argumento minimal para L si no existen subargumentos de $\mathcal{A}$ con conclusión L. Un argumento $\mathcal{A}$ es minimal si es minimal para su afirmación $claim(\mathcal{A})$.

Dados dos argumentos $\mathcal{A}_1$ y $\mathcal{A}_2$, las nociones de ataque tales como socava, refuta, ataque, y derrota se definen como:

- $\mathcal{A}_1$ *refuta* a $\mathcal{A}_2$ si existe un literal $L \in conc(\mathcal{A}_1)$ y $\neg L \in conc(\mathcal{A}_2)$.
- $\mathcal{A}_1$ *socava* a $\mathcal{A}_2$ si existe un literal $L \in conc(\mathcal{A}_1)$ y $not\ L \in assum(\mathcal{A}_2)$.
- $\mathcal{A}_1$ *ataca* a $\mathcal{A}_2$ si $\mathcal{A}_1$ socava o refuta a $\mathcal{A}_2$.
- $\mathcal{A}_1$ *derrota* a $\mathcal{A}_2$ si $\mathcal{A}_1$ socava a $\mathcal{A}_2$, o $\mathcal{A}_1$ refuta a $\mathcal{A}_2$ y $\mathcal{A}_2$ no socava a $\mathcal{A}_1$.

Dado un *PLE* $\mathcal{P}$, $Args_{\mathcal{P}}$ representa el conjunto de argumentos minimales asociados con $\mathcal{P}$, y $attacks_P$ una relación binaria de ataques sobre $Args_{\mathcal{P}}$ definida a partir de alguna de las nociones de ataque descriptas arriba. A partir de estas nociones Wakaki basa la formalización de su marco de argumentación no abstracto en el marco de argumentación basado en preferencias (MAP) de Amgoud y Cayrol [AC02].

Entonces, un MAP no abstracto que maneja preferencias estáticas [Wak10] construido desde un PLP $(\mathcal{P},\Phi)$ se define como $(Args_{\mathcal{P}}, attacks_{\mathcal{P}}, \succsim)$, donde $\succsim$ es una relación reflexiva y transitiva sobre $Args_{\mathcal{P}}$, que expresa prioridad entre argumentos. Para dos argumentos $\mathcal{A}_1, \mathcal{A}_2 \in Args_{\mathcal{P}}$,

$$\mathcal{A}_2 \succsim \mathcal{A}_1 \text{ si } L_2 \geq L_1 \in \Phi^* \text{ siendo } claim(\mathcal{A}_1) = L_1 \text{ y } claim(\mathcal{A}_2) = L_2,$$

donde Φ^* denota la clausura reflexiva y transitiva de Φ. Las preferencias obtenidas luego serán tenidas en cuenta al momento de seleccionar las extensiones mayormente preferidas del sistema. Se puede demostrar que estas extensiones coinciden con los conjuntos de respuestas preferidos del PLP utilizado para definir el marco de argumentación en cuestión.

Ejemplo 3.6. *Considere el siguiente PLP* $(\mathcal{P}, \Phi)$*:*

$\mathcal{P}:$ $p \leftarrow not\ q, not\ r$
$q \leftarrow not\ q, not\ r$
$r \leftarrow not\ p, not\ q$
$s \leftarrow p$

$\Phi:$ $p \geq q, r \geq s.$

podemos obtener un marco de argumentación basado en preferencia a partir de PLP, donde $Args_{\mathcal{P}} = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$ *tal que*

$$\mathcal{A} = [p \leftarrow not\ q, not\ r]$$

$$\mathcal{B} = [q \leftarrow not\ p, not\ r]$$

$$\mathcal{C} = [r \leftarrow not\ q, not\ q]$$

$$\mathcal{D} = [s \leftarrow p; p \leftarrow not\ q, not\ r]$$

y si se considera la relación de ataque por socavamiento, tenemos que $attacks_{\mathcal{P}} = \{(\mathcal{A}, \mathcal{B}), (\mathcal{B}, \mathcal{A}), (\mathcal{C}, \mathcal{A}), (\mathcal{A}, \mathcal{C}), (\mathcal{B}, \mathcal{C}), (\mathcal{C}, \mathcal{B}), (\mathcal{B}, \mathcal{D}), (\mathcal{D}, \mathcal{B}), (\mathcal{C}, \mathcal{D}), (\mathcal{D}, \mathcal{C})\}$ *y* $\succsim = \{(\mathcal{A}, \mathcal{B}), (\mathcal{C}, \mathcal{D})\}$ *ya que* $p \geq q \in \Phi$ *para* $claim(\mathcal{A}) = p$, $claim(\mathcal{B}) = q$ *y* $r \geq s \in \Phi$ *para* $claim(\mathcal{C}) = r$, $claim(\mathcal{D}) = s$.

Note que los autores también introducen un marco de argumentación no abstracto basado en *PLE* con restricciones de integridad. Las restricciones de integridad son reglas de la forma: $\leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$. Informalmente, la idea de esta propuesta es que, dado un *PLE* $\mathcal{P} \cup IC$ y una semántica de aceptabilidad específica, se tengan en cuenta las restricciones de integridad IC para obtener los argumentos aceptables. Básicamente, esta formalización es una extensión del primer marco propuesto que permite manejar preferencias y restricciones de integridad.

Por último, note que en [Wak10] se reconoce que en la literatura [Šef08, PS97] existen otros trabajos que construyen un sistema de argumentación a partir de un programa lógico con preferencias. No obstante en la mayoría de estos sistemas la preferencia entre argumentos se define a partir de un orden sobre reglas, de hecho, en la Sección 3.3 se muestra un sistema de argumentación de este tipo basado en programación lógica extendida. El hecho de utilizar *PLP*s permite que las preferencias entre argumentos se definan mediante un criterio basado en comparación entre literales en lugar de usar un orden sobre reglas, esto marca una de las diferencias más grandes entre el trabajo propuesto en [Wak10] y otros sistemas similares que se pueden encontrar en la literatura.

3.7. Sistema Argumentativo de Amgoud *et al.*

El manejo de inconsistencia es un problema central en las bases de conocimiento que ha sido abordado por varios trabajos en la literatura de argumentación [APP00, AC02,

AK07]. En particular, en [APP00] se introduce una propuesta para tratar este problema que permite razonar sobre múltiples relaciones de preferencia sobre argumentos. El marco general de argumentación considerado por los autores es el presentado en [AC98] donde un marco de argumentación basado en preferencias (MAP) es una tupla $\langle Args, Def, \succsim \rangle$, tal que $Args$ es un conjunto de argumentos, Def es una relación binaria que representa la relación de derrota entre los argumentos, y $\succsim$ es un preorden sobre $Args \times Args$.

Dado que el trabajo se centra en el manejo de inconsistencia en bases de conocimientos, en lo que sigue se ilustran los conceptos de argumento, relación de derrota y relación de preferencia siguiendo este contexto. Los argumentos son construidos desde una base de conocimiento K, donde las formulas de K están expresadas en un lenguaje proposicional $\mathcal{L}$. Un *argumento* de K es un par (A, h), donde A es el soporte de la conclusión h, y se cumple que:

1. h es una fórmula del lenguaje,
2. A es una subbase de K,
3. A es consistente,
4. $A \vdash h$
5. A es minimal (no existe un subconjunto estricto de A que cumpla 3 y 4).

La relación de derrota se define a partir de las nociones de refutación y socavamiento [EH95]. Sea $Args$ el conjunto de argumentos obtenidos de una base de conocimiento K. Sean (A, h) y (A', h') dos argumentos de $Args$.

- (A, h) *refuta* a (A', h') si y solo si $h \equiv \neg h'$.
- (A, h) *socava* a (A', h') si y solo si para algún $k \in A'$, $h \equiv \neg k$.

Si bien en [APP00] se reconoce que la relación de preferencia entre argumentos se puede definir de varias maneras [ACB96], los autores utilizan el criterio *ELI-preference* introducido en [CRS92] el cual compara el soporte de los argumentos involucrados. Este criterio asume que existe una relación de prioridad sobre la base de conocimiento. Sea $\geq$ un preorden total sobre la base K y $>$ la relación estricta asociada a $\geq$. En este caso, la base de conocimiento K se asume que está estratificada en $(K_1, K_2, \ldots, K_n)$ tal que K_1 es el conjunto de elementos maximales de $\geq$ en K y K_{i+1} el conjunto de elementos

maximales de $\geq$ en $K \setminus (K_1, K_2, \ldots, K_i)$. Entonces, un argumento (A, h) es *preferido* a (A', h'), denotado $(A, h) \succ (A', h')$, si y solo si $\forall\ k \in A \setminus A', \exists\ k' \in A' \setminus A$ tal que $k > k'$.

Ejemplo 3.7. *Sea* $K = K_1 \cup K_2 \cup K_3$ *tal que* $K_1 = \{a, \neg a\}, K_2 = \{a \rightarrow b\}$ *y* $K_3 = \{\neg b\}$. *Entonces,* $(\{a, a \rightarrow b\}, b) \succ (\{\neg b\}, \neg b)$.

Note que para obtener los argumentos aceptables del sistema el enfoque propuesto utiliza la noción de defensa que se define a partir de las nociones de derrota y preferencia entre argumentos. Sea $\langle Args, Def, \succsim \rangle$ un *MAP*. Dado $\mathcal{A}, \mathcal{B} \in Args$, se dice que $\mathcal{A}$ se *defiende* a sí mismo de $\mathcal{B}$ si y solo si $(\mathcal{B}, \mathcal{A}) \in Def$ y $\mathcal{A} \succ \mathcal{B}$. Se denotará con $C_{Def,\succsim}$ al conjunto de argumentos que se defienden a sí mismos de todos sus derrotadores. Un argumento $\mathcal{A}$ es defendido por un conjunto de argumentos $S \subseteq Args$ si y solo si $\forall\ \mathcal{B} \in Args$, si $(\mathcal{B}, \mathcal{A}) \in Def$ y no se cumple que $\mathcal{A} \succ \mathcal{B}$, entonces $\exists\ \mathcal{C} \in S$ tal que $(\mathcal{C}, \mathcal{B}) \in Def$ y no se cumple que $\mathcal{B} \succ \mathcal{C}$.

A partir de los conceptos anteriores se está en condiciones de definir la aceptabilidad de los argumentos. El *conjunto de argumentos aceptables* S_a se obtiene como el menor punto fijo de la siguiente función:

- $F : 2^{Args} \rightarrow 2^{Args}$
- $S \rightarrow F(S) = \{\mathcal{A} \in Args \mid \mathcal{A}\ es\ defendido\ por\ S\}$.

Entonces, el menor punto fijo de F se calcula de la siguiente manera:

$$S_a = C_{Def,\succsim} \cup [\cup F^{i \geq 1}(C_{Def,\succsim})].$$

Observe que las preferencias pueden estar en conflicto. Esto se da cuando por ejemplo un argumento $\mathcal{A}$ es preferido a otro argumento $\mathcal{B}$ en un contexto c_1 y el argumento $\mathcal{B}$ es preferido a $\mathcal{A}$ en un contexto c_2. Para resolver el conflicto entre preferencias que puedan surgir cuando estas preferencias se expresan en contextos diferentes, en [APP00] proponen extender el marco de argumentación basado en preferencias presentado en [AC98] para que tome en cuenta preferencias contextuales.

La principal contribución de este trabajo es contar con varios preordenes sobre las creencias de la misma base de conocimiento. Estos preordenes están dados a partir de la noción de preferencia contextual, es decir preferencias que dependen de un contexto en particular. De hecho, el marco está constituido por un conjunto de relaciones de preferencia sobre argumentos, donde cada relación de preferencia $\succsim_i$ se deriva de un preorden $\geq_i$

expresado en un contexto particular c_i. Para determinar los argumentos aceptables, el conjunto de relaciones de preferencia sobre argumentos está linealmente ordenado a partir una relación de preorden denotada $\rhd$ que se aplica sobre el conjunto de contextos C.

Un marco de argumentación basado en preferencias contextuales ($MAPC$) es una tupla $\langle Args, Def, \mathsf{C}, \rhd, \{\succsim_1, \succsim_2, \ldots, \succsim_n\}\rangle$ que consiste de cinco elementos:

1. un conjunto de argumentos $Args$,
2. una relación Def representando una relación de derrota sobre $Args$,
3. un conjunto de contextos $\mathsf{C} = \{c_1, c_2, \ldots, c_n\}$,
4. una relación de preorden completa $\rhd$ sobre C, y
5. un conjunto de relaciones de preferencia $\{\succsim_1, \succsim_2, \ldots, \succsim_n\}$ tal que $\succsim_i$ define un preorden (total o parcial) sobre $Args$ inducido a partir del contexto c_i.

Luego de construir los argumentos y contra-argumentos (derrotadores), el próximo paso en el proceso de argumentación es la selección de los argumentos aceptables. Para obtener los argumentos aceptables los autores proponen tres soluciones alternativas:

1. La primera consiste en generar una nueva relación de preferencia $\succsim$ a partir del conjunto de relaciones de preferencia $\{\succsim_1, \succsim_2, \ldots, \succsim_n\}$. La idea es mantener las preferencias expresadas en el "mejor" contexto, luego elegir el próximo mejor contexto entre los restantes, y de las preferencias de este contexto agregar únicamente las que no contradicen a las preferencias ya guardadas.

   Por ejemplo, sea K una base de conocimiento tal que $K = \{a, a \rightarrow b, \neg b, c, \neg c\}$. Considere los siguientes argumentos: $\mathcal{A} = (\{a, a \rightarrow b\}, b)$, $\mathcal{B} = \{(\{\neg b\}, \neg b)\}$, $\mathcal{C} = (\{c\}, c)$, y $\mathcal{D} = (\{\neg c\}, \neg c)$. Suponga ahora que $\mathsf{C}= \{c_1, c_2, c_3\}$ tal que $c_1 \rhd c_2 \rhd c_3$, y $\succsim_1= \{(\mathcal{A}, \mathcal{B})\}$, $\succsim_2= \{(\mathcal{B}, \mathcal{A}), (\mathcal{C}, \mathcal{D})\}$, $\succsim_3= \{(\mathcal{D}, \mathcal{C})\}$. Según esta estrategia tenemos que $\succsim= \{(\mathcal{A},\mathcal{B}), (\mathcal{C},\mathcal{D})\}$.

2. La segunda consiste en preferir aquellos argumentos derrotados sobre sus derrotadores si dicha preferencia se da en un contexto que tiene precedencia sobre todos los demás contextos donde la preferencia opuesta está disponible. Se dice que estos argumentos se defienden a si mismos de sus posibles derrotadores. Luego, el conjunto de argumentos aceptables se obtiene como el menor punto fijo de la función

F definida arriba reemplazando la definición de defensa por la descripta en esta estrategia.

Continuando con el ejemplo de la estrategia anterior, y dado el *MAPC* $\langle Args, Def, \mathsf{C}, \rhd, \{\succsim_1, \succsim_2, \ldots, \succsim_n\}\rangle$. Asumiendo que se utiliza derrota por refutación tenemos que $\mathcal{A}$ se defiende de $\mathcal{B}$ mientras $\mathcal{B}$ no se defiende de $\mathcal{A}$ ya que el contexto donde $\mathcal{B}$ es preferido a $\mathcal{A}$ (c_2) tiene menos prioridad que el contexto (c_1) en el cual $\mathcal{A}$ es preferido a $\mathcal{B}$.

3. La última es similar a la primer estrategia. Se obtiene el conjunto de argumentos aceptables S_i para cada $\langle Args, Def, \succsim_i\rangle$. Se mantienen todos los argumentos del conjunto S_i tal que c_i es el contexto más preferido. De los contextos restantes se selecciona el más preferido (con respecto a la relación $\rhd$), c_j, de los argumentos de S_j únicamente mantenemos aquellos que no son derrotados por los argumentos ya guardados. Se denotará S_{a3} al conjunto de argumentos aceptables que resulta de aplicar esta estrategia.

   Continuando con el *MAPC* $\langle Args, Def, \mathsf{C}, \rhd, \{\succsim_1, \succsim_2, \ldots, \succsim_n\}\rangle$ utilizado con anterioridad, y considerando la tercer estrategia, resulta que $\{\mathcal{A}\} \subseteq S_1$, $\{\mathcal{B}, \mathcal{C}\} \subseteq S_2$ y $\{\mathcal{D}\} \subseteq S_3$. Entonces $\{\mathcal{A}, \mathcal{C}\} \subseteq S_{a3}$.

Como se puede observar, este trabajo introduce un enfoque argumentativo que permite manejar múltiples relaciones de preferencias. Como parte de la contribución, los autores proponen tres estrategias para el cómputo de aceptabilidad de argumentos. Dentro de estas estrategias, la primera provee un proceso de combinación de preferencias, la cual resulta de particular interés en esta tesis. Combinar preferencias es un tema que será abordado en el Capítulo 7 donde se presenta un formalismo cuya máquina de inferencia la define el sistema argumentativo de programación lógica rebatible DeLP. De manera similar al trabajo presentado en esta sección, este formalismo considera varias relaciones de preferencias entre argumentos a partir del uso de diferentes criterios de preferencia, no obstante existen ciertas diferencias a tener en cuenta. En primer lugar, en [APP00] no se especifican mecanismos concretos para seleccionar y combinar relaciones de preferencias, de hecho se propone un método fijo de combinación de preferencias que utiliza todas las relaciones disponibles. En cambio, en la propuesta que se desarrolla en esta tesis es posible definir diferentes métodos que permiten el uso combinado de varios criterios de preferencia. Además, se introducen también herramientas para el usuario que permiten seleccionar y anidar varios de estos métodos. Finalmente, es importante remarcar que uno

de los aportes principales de esta tesis es la posibilidad de establecer preferencias entre argumentos a partir del uso combinado de varios criterios en un contexto de programación lógica rebatible.

3.8. Otros Sistemas Argumentativos

Los marcos de argumentación basada en suposiciones (AS) [DKT06, DKT09] se pueden definir a partir de cualquier lógica cuyo operador de consecuencia está especificado mediante reglas de inferencia, identificando sentencias en el lenguaje subyacente que puedan ser tratadas como suposiciones. Intuitivamente, los argumentos son deducciones de una conclusión sustentada por un conjunto de suposiciones. La argumentación basada en suposiciones (AS) al igual que la argumentación abstracta son marcos de argumentación de propósito general. Sin embargo, una de las diferencias esenciales entre éstos es que en la AS los argumentos tienen una estructura, mientras que en la argumentación abstracta la estructura no está definida. Por otra parte, estos marcos coinciden en que no requieren preferencias para resolver conflictos entre argumentos, ni tampoco establecen mecanismos para tratar con preferencias explícitas. En este contexto, en [Wak14b] proponen un marco de argumentación basada en suposiciones equipado con preferencias (p_AS), el cual incorpora prioridades explícitas sobre sentencias. Para lograr esto, los p_ASs son instanciados con Programas Lógicos Priorizados (PLP) [SI00].

Así como en [Wak10] el autor presenta un enfoque de argumentación basado en programación lógica priorizada que maneja preferencias estáticas, en [Wak11] Wakaki desarrolla un sistema argumentativo pero con el objetivo de tratar el manejo de preferencias dinámicas. En este trabajo se utiliza un PLP jerárquico, el cual permite a los formalismos de programación en lógica priorizada representar y razonar sobre preferencias dinámicas. Utilizando el PLP jerárquico como lenguaje de representación es que se construye un MAP no abstracto. Para lograr esto, se extiende el formalismo de un PLP para que se pueda razonar sobre prioridades entre reglas. El formalismo permite derivar conclusiones con preferencias sobre reglas dentro del mismo sistema. Intuitivamente, una vez que se obtiene uno o varios conjuntos de argumentos aceptables, el conjunto Φ contará con prioridades que son conclusiones de argumentos que forman parte de dichos conjuntos. Note que el manejo de preferencias dinámicas ha sido un tema abordado por varios autores en la literatura de argumentación; por ejemplo, en la Sección 3.3 se presentó el trabajo de Prakken y Sartor [PS97] donde se introduce un enfoque para razonar sobre prioridades

rebatibles.

3.9. Discusión

Como se ha mostrado a lo largo de este capítulo, contar con un mecanismo para expresar que un argumento es preferido a otro es un requisito importante para el proceso de argumentación. Vimos que distintos sistemas argumentativos adoptan diferentes criterios de preferencia entre argumentos. Algunos optan por un criterio que utiliza un orden explícito sobre reglas y comparan solo reglas [PS97, Šef08, GS04], mientras que otros por ejemplo consideran un criterio basado en un orden sobre literales [Wak10, Wak11, FEGS08]. Sin embargo, estos criterios poseen algunos inconvenientes, dado que en numerosas situaciones no es posible escoger en forma adecuada o realista estas prioridades. Por otra parte, también complica la actualización de la información, dado que al agregar nueva información a la base de conocimiento debemos actualizar el orden existente entre las reglas o literales según cual sea el criterio utilizado. Por otra parte, otros sistemas como el presentado en [SL92, GS04] han optado por definir el criterio siguiendo el principio de especificidad, que prefiere aquellos argumentos con información más específica. Finalmente, entre otros sistemas están los que utilizan criterios que consideran el tipo de prioridad asociada a la información representada, es decir, consideran si la prioridad de la información está explícitamente o implícitamente expresada en el formalismo [AK07].

A partir de la discusión previa, es claro que existen diferentes posturas por parte de los investigadores respecto al criterio de preferencia. En este sentido, algunos investigadores, por ejemplo [Vre91, Pol95, PS96a], consideran que especificidad no constituye un criterio general del razonamiento de sentido común, sino simplemente es un criterio más que podría adoptarse. Otros investigadores sostienen que no existen principios generales, independientes del dominio, dado que éstos conllevan a un estado de indecisión en la mayoría de los casos, y que la información acerca del dominio constituye la herramienta más importante para decidir entre argumentos en conflicto [Kon88, Vre91, SL92, FEGS08, Wak11]. Por esta razón varios sistemas de argumentación consideran la comparación de argumentos de una forma modular, esperando que se utilice el criterio que mejor se adecue al dominio que esta siendo representado.

Teniendo en cuenta lo anterior, en [TGGS15] se ha reconocido la importancia de cambiar el criterio de preferencia entre argumentos dentro del sistema. No obstante, en argumentación no hay una postura establecida sobre cómo debe encararse este desafío.

En la mayoría de los formalismos, una vez establecida la configuración del sistema, el criterio es fijo (*i.e.,* no puede ser cambiado) [PS97, Wak10, Wak11, AK07]. Por otra parte, en otras aproximaciones, que consideran que hay disponible varios criterios, se utiliza una combinación fija de criterios sin proveer al sistema de mecanismos formales para seleccionar y cambiar de criterio [DDG$^+$12, GMP12]. En esta tesis se abordará este problema introduciendo una formalización que permite ajustar dinámicamente el criterio que se vaya a utilizar dependiendo de las necesidades o preferencias del usuario. Para esto, se utilizará como base el sistema DeLP [Gar00, GS04]. Contar con herramientas computacionales que permitan seleccionar y cambiar de criterio representa un importante avance en el área de los sistemas de argumentación.

3.10. Conclusión

Este capítulo se dividió en tres partes. En la primera, se introdujo resumidamente la estructura conceptual identificada en [PV02] con el objetivo de presentar los principales conceptos que caracterizan a los sistemas argumentativos. Luego, como esta tesis involucra a un sistema argumentativo estructurado (DeLP) [Gar00, GS04], en la segunda parte se presentó una visión general de algunos sistemas argumentativos, mostrando en particular cómo éstos tratan la comparación de argumentos. Y finalmente, se discutieron algunos aspectos relacionados a las posturas adoptadas por los investigadores al momento de definir el criterio de preferencia entre argumentos que un sistema argumentativo debe utilizar. El estudio de este tema resulta fundamental para esta tesis, ya que constituye una de las motivaciones de la misma.

En el próximo capítulo se introducirán los conceptos básicos del sistema de DeLP. Este formalismo constituirá la herramienta base de representación de conocimiento y razonamiento de los formalismos presentados en los siguientes capítulos.

Capítulo 4

Programación lógica rebatible - DeLP

La Programación Lógica Rebatible (DeLP por sus siglas en inglés de *Defeasible Logic Programming*) [Gar00, GS04] permite modelar conocimiento que involucra información incompleta o potencialmente contradictoria. El mecanismo de inferencia sobre el cual está basado permite decidir entre conclusiones contradictorias y adaptarse fácilmente a entornos dinámicos como los tratados en esta tesis. El hecho de representar y razonar sobre información tentativa (además de estricta) hacen de DeLP un atractivo sistema de representación de conocimiento y razonamiento en un gran número de dominios de aplicación [RGS07, GGS08, SMCS08, DFDG+13].

Un programa lógico rebatible está formado por un conjunto de *hechos*, *reglas estrictas*, y *reglas rebatibles* permitiendo estas últimas la representación de información tentativa. Toda conclusión del programa estará sustentada por algún *argumento* que pueda construirse utilizando las reglas y los hechos del programa. Cuando se utilicen reglas rebatibles para decidir cuándo una conclusión L puede aceptarse a partir de un programa lógico rebatible se realizará un *análisis dialéctico*, construyendo argumentos a favor y en contra de la conclusión L.

Un argumento que sustenta una conclusión L podrá ser atacado por otros argumentos *derrotadores* que lo contradigan y que pueden construirse a partir del programa. Dichos derrotadores podrán a su vez ser atacados, y así sucesivamente, generando una secuencia de argumentos llamada *línea de argumentación*. Cada línea de argumentación deberá satisfacer ciertas propiedades, a fin de evitar que se generen argumentaciones falaces. El proceso completo considera para cada argumento todos sus posibles argumentos derrota-

dores, que en lugar de una única línea de argumentación, genera un conjunto de líneas que serán representadas de una manera compacta por medio de un *árbol de dialéctica*. Este análisis dialéctico determinará si la conclusión en cuestión está *garantizada* o no a partir del programa.

A continuación se introducirán los conceptos preliminares de la programación en lógica que son utilizados por DeLP. Luego, se introducirán los conceptos de programa lógico rebatible, y posteriormente se introducirán las nociones necesarias para realizar el análisis dialéctico. Este análisis dialéctico será la base para la definición de la respuesta a una consulta. En la Sección 4.4 se detallarán varias propuestas de criterios de preferencia entre argumentos que pueden ser utilizados por el lenguaje DeLP. En la Sección 4.5 se explicará paso a paso, con un ejemplo de aplicación en un dominio concreto, cómo DeLP resuelve sus consultas. Esta parte del capítulo será de especial interés por que ayudará a mostrar y entender la importancia que tiene el criterio de preferencia entre argumentos al momento que DeLP debe procesar una consulta. Con el ejemplo se mostrará cómo las estructuras de árboles obtenidas en el análisis dialéctico, y consecuentemente la respuesta a una consulta, podrían variar dependiendo del criterio elegido. Finalmente, en la Sección 4.6 se presentarán los conceptos de Servicios de Razonamiento basados en Programación Lógica Rebatible o usualmente llamados DeLP-servers y Consultas Contextuales. Estos conceptos representan la base para las formalizaciones que se presentarán en los capítulos siguientes.

4.1. Conceptos Preliminares

En esta sección se introducirán los conceptos y terminología estándar de lógica y programación en lógica, algunos de los cuales serán necesarios para presentar el lenguaje de la Programación Lógica Rebatible. En [Gar00] siguen los lineamientos presentados en [Fil01] para las definiciones de esta sección.

Como se muestra a continuación, lo primero que se asumirá es que entre los elementos básicos del alfabeto del lenguaje se incluye un conjunto de símbolos de variables, funciones y predicados. El concepto de *signatura* contiene los elementos del lenguaje que variarán de un programa a otro.

Definición 4.1 (Signatura)**.** *Una signatura es una tupla* $\sigma = \langle \mathcal{V}, Func, Pred \rangle$ *donde* $\mathcal{V}$*, Func, y Pred son conjuntos finitos de símbolos, disjuntos de a pares, tales que:*

- $\mathcal{V}$ *es un conjunto de variables,*

- *Func es un conjunto de símbolos de funciones,*
- *Pred es un conjunto no vacío de símbolos de predicados.*

Dada una signatura σ, la función "*aridad*" le asignará a cada elemento de *Func* y *Pred* un número entero positivo. Si $f \in Func$ y $aridad(f) = 0$, entonces f se denominará *constante*, por otra parte, si $p \in Pred$, y $aridad(p) = 0$, entonces p se denominará *proposición*.

Las variables serán denotadas por cualquier cadena de caracteres que tenga una letra mayúscula inicial, mientras que los predicados y funciones serán notados por cualquier cadena que comience con letras minúsculas. El contexto permitirá distinguir entre predicados y funciones siempre que se encuentre una cadena de caracteres que comience con letras en minúscula.

Definición 4.2 (Alfabeto)**.** *El alfabeto generado a partir de una signatura σ, consiste del conjunto de elementos miembros de la signatura, el símbolo de negación "$\sim$", y los símbolos de puntuación "(", ")", y ",".*

Definición 4.3 (Término)**.** *Sea $\sigma = \langle \mathcal{V}, Func, Pred \rangle$ una signatura. Un término t, y respectivamente sus componentes $comp(t)$, se definen inductivamente como sigue:*

- *toda variable $V \in \mathcal{V}$ es un término, y $comp(V) = \{V\}$;*
- *toda constante $c \in Func$, $aridad(c) = 0$, es un término, y $comp(c) = \{c\}$*
- *si $f \in Func$, $aridad(f) = n$ con $n \geq 1$, y $t_1, t_2, \ldots, t_n$ son términos, entonces $f(t_1, t_2, \ldots, t_n)$ también es un término.*
  Se define $comp(f(t_1, t_2, \ldots, t_n)) = \{f\} \cup (\bigcup_{1 \leq i \leq n} comp(t_i))$.

Definición 4.4 (Término fijo)**.** *Sea $\sigma = \langle \mathcal{V}, Func, Pred \rangle$ una signatura y t un término, t será un término fijo si no contiene variables. Esto es, $comp(t) \cap \mathcal{V} = \emptyset$.*

Los términos representan a los objetos del dominio que se desea representar y sobre los cuales se definirán relaciones.

Definición 4.5 (Átomo)**.** *Sea $\sigma = \langle \mathcal{V}, Func, Pred \rangle$ una signatura, y $t_1, t_2, \ldots, t_n$ términos. Si $p \in Pred$ con $aridad(p) = n$ entonces $p(t_1, t_2, \ldots, t_n)$ es un átomo. Un átomo fijo es un átomo $p(t_1, t_2, \ldots, t_n)$, donde todos sus términos $t_1, t_2, \ldots, t_n$ son fijos.*

Los átomos expresan las relaciones que existen entre los objetos del dominio a representar y tendrán asignado un valor de verdad.

Definición 4.6 (Literal)**.** *Un literal L es un átomo A o un átomo negado $\sim A$, donde "$\sim$" representa la negación ($fuerte$). Un literal L se dirá negativo si es un átomo negado, y positivo en caso contrario. Un literal fijo es un átomo fijo o un átomo fijo negado.*

Observación 4.1. *Los literales podrán ser átomos negados utilizando el símbolo "$\sim$" de la negación(fuerte). No debe confundirse este tipo de negación, con la negación por falla (o negación default) que es utilizada en Programación en Lógica, y que normalmente se denota con el símbolo "not".*

Definición 4.7 (Complemento de un literal)**.** *Sea L un literal y A un átomo, el complemento de L con respecto a la negación fuerte, denotado $\overline{L}$, se define de la siguiente manera:*

- *si $L = A$, entonces $\overline{L} = \sim A$;*
- *si $L = \sim A$, entonces $\overline{L} = A$.*

Los literales A y $\sim A$ serán llamados literales complementarios.

4.2. Lenguaje de Representación de Conocimiento

Definición 4.8 (Lenguaje DeLP)**.** *El lenguaje de DeLP se compone de:*

- *hechos, que son literales fijos,*
- *reglas estrictas, denotadas "$L_0 \leftarrow L_1, \ldots, L_k$",*
- *reglas rebatibles, denotadas "$L_0 \prec L_1, \ldots, L_k$", y*
- *presuposiciones, denotadas "$L_0 \prec$ "*

  donde $L_0, L_1, L_2, \ldots, L_k$, con $k > 0$, son literales fijos.

Los hechos y reglas estrictas se emplean para representar conocimiento seguro, libre de excepciones, mientras que las reglas rebatibles actúan como reglas de inferencia que se utilizan para obtener conclusiones tentativas. De esta manera, una regla rebatible, denotada $Cabeza \prec Cuerpo$, donde *Cabeza* es un literal fijo, y *Cuerpo* es un conjunto

finito no vacío de literales fijos, expresa que "razones para creer en *Cuerpo* proveen razones para creer en *Cabeza*". Por otro lado, una presuposición es un caso particular de regla rebatible cuyo cuerpo contiene un conjunto vacío de literales. Una presuposición "$L \prec$" expresa que "se puede suponer L", con lo cual representa una afirmación mucho más débil que un hecho.

Definición 4.9 (Programa DeLP)**.** *Un programa lógico rebatible es un conjunto $\mathcal{P}$, de hechos, reglas estrictas y reglas rebatibles. En un programa $\mathcal{P}$ se identificará con Θ al conjunto de hechos, con Ω al conjunto de reglas estrictas, y con Δ al conjunto de reglas rebatibles. Por conveniencia, también se denotará con Π al conjunto $\Theta \cup \Omega$. Por lo tanto, en algunas ocasiones denotaremos a un programa lógico rebatible $\mathcal{P} = (\Pi, \Delta)$, y en otros casos $\mathcal{P} = (\Theta, \Omega, \Delta)$.*

Cabe destacar que si bien DeLP solo admite literales fijos en las reglas, siguiendo [Lif96], en algunos ejemplos se utilizarán variables esquemáticas solamente como una forma de denotar *esquemas de reglas*. Para diferenciar las variables, éstas serán denotadas con una letra mayúscula inicial. A continuación se muestran varios ejemplos de programas DeLP, que serán utilizados a lo largo de la tesis.

Ejemplo 4.1. *El programa DeLP $\mathcal{P}_{4.1}$ presentado a continuación representa información sobre aves.*

$$\mathcal{P}_{4.1} = \left\{ \begin{array}{l} vuela(X) \prec ave(X) \\ \sim vuela(X) \prec gallina(X) \\ vuela(X) \prec gallina(X), asustada(X) \\ \\ ave(X) \leftarrow gallina(X) \\ gallina(tina) \end{array} \right\}$$

Como puede observarse, el programa $\mathcal{P}_{4.1}$ contiene: tres reglas rebatibles, una regla estricta y un hecho. Las reglas rebatibles indican razones a favor y en contra para creer que las aves vuelan. Intuitivamente, la primer regla expresa que "las aves normalmente vuelan", la segunda "las gallinas por lo general no vuelan", y la última "las gallinas asustadas usualmente vuelan". Por otro lado, la regla estricta y el hecho aseguran que "una gallina es un ave", y "tina es una gallina", respectivamente.

Ejemplo 4.2. *El programa DeLP $\mathcal{P}_{4.2}$ expresa información sobre el mercado de valores.*

$$\mathcal{P}_{4.2} = \left\{ \begin{array}{l} comprar\_acciones(T) \prec buen\_precio(T) \\ \sim comprar\_acciones(T) \prec buen\_precio(T), empresa\_riesgosa(T) \\ \sim comprar\_acciones(T) \prec empresa\_riesgosa(T) \\ empresa\_riesgosa(T) \prec huelga\_empleados(T) \\ huelga\_empleados(acme) \prec \\ buen\_precio(acme) \end{array} \right\}$$

Para representar la información tentativa en este caso se utilizaron cuatro reglas rebatibles y una presuposición. De la primer regla se puede concluir que se deben comprar acciones de una empresa si hay razones para creer que están a buen precio. La segunda y tercer regla presentan razones en contra de comprar acciones: si las acciones están a buen precio pero son de una empresa riesgosa entonces hay razones tentativas para no comprar dichas acciones, a la misma conclusión se puede llegar si hay razón para creer que la empresa es una empresa en riesgo. La cuarta regla tentativa indica que si hay huelga de empleados ésto es una razón para suponer que es una empresa en riesgo. Por último, se puede suponer que en la empresa "acme" hay huelga de empleados.
La parte estricta del programa cuenta solamente con un hecho el cual afirma que las acciones de la empresa "acme" están a buen precio.

Ejemplo 4.3. *El programa DeLP $\mathcal{P}_{4.3}$ representa información para el diagnóstico de una enfermedad:*

$$\mathcal{P}_{4.3} = \left\{ \begin{array}{l} enferm(E) \prec sintC(S, E) \\ \sim enferm(E) \prec sintC(S, E), result(pruebaD(P, E), V), V = negativo \\ \sim enferm(E) \prec trat(T, E) \\ sintC(s1, e1) \prec \\ trat(t1, e1) \\ result(pruebaD(p1, e1), negativo) \end{array} \right\}$$

Observe que el programa tiene dos hechos. Estos hechos representan indicadores que el médico utiliza para posiblemente diagnosticar a un paciente una determinada enfermedad: el paciente se encuentra bajo el tratamiento "t1" para la enfermedad "e1", y la prueba diagnóstica "p1" para "e1" dio un resultado negativo.
La información tentativa está representada por tres reglas rebatibles y una presuposición. Las reglas identifican razones a favor y en contra de diagnosticar una enfermedad. La

primer regla expresa que si el paciente presenta algún síntoma asociado a una enfermedad, ésto es razón para concluir que padece la enfermedad. La siguiente regla concluye en no diagnosticar una enfermedad si hay razón para creer que el paciente presenta algún síntoma clínico pero la prueba diagnóstica para la enfermedad dio un resultado negativo.

Por último, la tercer regla expresa que si el paciente está bajo tratamiento, ésto es una razón para no diagnosticar la enfermedad tratada. Observe que la presuposición representa determinados síntomas que el paciente manifiesta al médico. De esta manera, la presuposición del programa expresa que existe razón para suponer que el paciente presenta el síntoma "s1" correspondiente a la enfermedad "e1".

Definición 4.10 (Consulta Lógica Rebatible)**.** *Una consulta lógica rebatible corresponde a un literal. Si el literal es fijo, la consulta será una consulta fija lógica rebatible. En caso contrario, será una consulta esquemática lógica rebatible.*

El objetivo de una consulta fija lógica rebatible es saber si el literal fijo correspondiente está garantizado a partir de un programa lógico rebatible. En la siguiente sección se describirá cómo se calcula el estado de garantía para un literal fijo. Mientras que, en el caso de una consulta esquemática lógica rebatible, el objetivo es determinar si existe algún literal fijo que sea instancia de la consulta esquemática y que esté garantizado a partir del programa lógico rebatible correspondiente.

4.3. Argumentación Rebatible

A continuación se introducirá el concepto de *derivación rebatible*, con el cual se define qué literales pueden ser derivados utilizando las reglas de un programa lógico rebatible.

Definición 4.11 (Derivación rebatible)**.** *Sea $\mathcal{P}$ un programa DeLP y L un literal fijo. Una derivación rebatible de L a partir de $\mathcal{P}$, denotado $\mathcal{P} \mid\sim L$, consiste de una secuencia finita $L_1, L_2, \ldots, L_n = L$ de literales fijos tal que para cada i, $1 \leq i \leq n$, se cumple que:*

1. *L_i es un hecho en $\mathcal{P}$, o*
2. *L_i es una presuposición en $\mathcal{P}$, o*
3. *existe una regla R_i (estricta o rebatible) en $\mathcal{P}$ con cabeza L_i y cuerpo $B_1, B_2, \ldots, B_k$ y cada literal en el cuerpo B_j $(1 \leq j \leq k)$ de la regla es un literal que aparece antes que L_i en la secuencia.*

La derivación se dice rebatible, porque aunque un literal L pueda ser derivado, puede existir en el programa información que contradiga a L, y entonces L podría no ser aceptado como una creencia válida del programa. Por ejemplo, a partir del programa $\mathcal{P}_2$ del Ejemplo 4.2 es posible obtener una derivación rebatible para el literal "*compra_acciones*(*acme*)", ya que existe la secuencia de literales: *buen_precio*(*acme*), *compra_acciones*(*acme*); que se obtiene utilizando el hecho "*buen_precio*(*acme*)", y la regla rebatible "*compra_acciones*(*acme*) $-\!\prec$ *buen_precio*(*acme*)". Además, también es posible obtener una derivación para el literal "$\sim$*compra_acciones*(*acme*)", ya que existe la secuencia de literales: *huelga_empleados*(*acme*), *empresa_riesgosa*(*acme*), $\sim$*compra_acciones*(*acme*). Un caso especial de derivación, denominado *derivación estricta*, corresponde al caso donde todas las reglas utilizadas en la derivación son estrictas.

Definición 4.12 (Derivación estricta)**.** *Sea $\mathcal{P} = (\Pi, \Delta)$ un programa DeLP y L un literal para el cual existe una derivación rebatible $L_1, L_2, \ldots, L_n = L$. Se dirá que L tiene una derivación estricta a partir de $\mathcal{P}$, denotado $\mathcal{P} \vdash L$, si todas las reglas de programa utilizadas para obtener la secuencia $L_1, L_2, \ldots, L_n = L$ son reglas estrictas.*

Como las reglas de programa permiten utilizar literales negados en la cabeza, entonces es posible derivar literales complementarios; se dice que dos literales son contradictorios si son complementarios. Como las reglas rebatibles permiten derivar literales complementarios, el conjunto $\Pi \cup \Delta$ puede ser contradictorio. Un *conjunto de reglas es contradictorio* si y solo si es posible obtener derivaciones rebatibles para un literal L y su complemento $\overline{L}$, a partir de este conjunto.

Observación 4.2. *Dado un programa lógico rebatible $\mathcal{P} = (\Pi, \Delta)$, en DeLP se asume que el conjunto Π es un conjunto no contradictorio.*

DeLP utiliza como procedimiento de prueba un mecanismo de análisis global para decidir qué literales fijos serán aceptados a partir de un programa. Este procedimiento de prueba permitirá definir argumentos para un literal fijo. Luego, un análisis dialéctico permitirá decidir cuándo un literal fijo está aceptado considerando los posibles argumentos de manera recurrente. A continuación se muestra la noción de argumento para programas lógicos rebatibles.

Definición 4.13 (Argumento)**.** *Sea $\mathcal{P} = (\Pi, \Delta)$ un programa DeLP. Se dirá que el par $\langle \mathcal{A}, L \rangle$ es un argumento para el literal L a partir de $\mathcal{P}$, si $\mathcal{A}$ es un conjunto de reglas rebatibles de Δ, tal que:*

1. $\Pi \cup \mathcal{A} \vdash L$,

2. $\Pi \cup \mathcal{A}$ *es no contradictorio, i.e.,* $\Pi \cup \mathcal{A}$ *no deriva de forma estricta un literal y su complemento y*

3. $\mathcal{A}$ *es minimal: no existe* $\mathcal{A}' \subset \mathcal{A}$ *tal que satisface las condiciones (1) y (2).*

Por conveniencia a veces se llamará a un argumento $\langle \mathcal{A}, L \rangle$*, simplemente un argumento* $\mathcal{A}$ *para* L*.*

Observación 4.3. *Sea* L *un literal para el cual existe una derivación estricta a partir de* $\mathcal{P}$*, entonces existe un único argumento* $\langle \mathcal{A}, L \rangle$*, donde* $\mathcal{A} = \emptyset$*, y por la definición de argumento no puede existir un argumento para* $\overline{L}$*.*

Definición 4.14 (Sub-Argumento)**.** *Un argumento* $\langle \mathcal{B}, Q \rangle$ *es un subargumento de un argumento* $\langle \mathcal{A}, L \rangle$ *si* $\mathcal{B} \subseteq \mathcal{A}$*.*

Ejemplo 4.4. *Considere los argumentos presentados en el Ejemplo 4.2. El argumento* $\langle \mathcal{B}_4, empresa\_riesgosa(acme) \rangle$*, es un subargumento de* $\langle \mathcal{B}_1, \sim comprar\_acciones(acme) \rangle$ *y de* $\langle \mathcal{B}_3, \sim comprar\_acciones(acme) \rangle$*.*

Dos literales L y L_1 están en desacuerdo con respecto a un programa (Π, Δ), si $\Pi \cup \{L, L_1\}$ es contradictorio. De esta manera, se dirá que dos argumentos estarán en conflicto cuando sustenten conclusiones en desacuerdo.

Definición 4.15 (Contra-argumento o ataque)**.** *Sean* $\langle \mathcal{A}_1, L_1 \rangle$ *y* $\langle \mathcal{A}_2, L_2 \rangle$ *argumentos obtenidos a partir de un programa lógico rebatible* $\mathcal{P}$*. El argumento* $\langle \mathcal{A}_1, L_1 \rangle$ *contra-argumenta o ataca al argumento* $\langle \mathcal{A}_2, L_2 \rangle$ *en el literal* L*, si y solo si existe un subargumento* $\langle \mathcal{A}, L \rangle$ *de* $\langle \mathcal{A}_2, L_2 \rangle$ *tal que* L *y* L_1 *están en desacuerdo. El argumento* $\langle \mathcal{A}, L \rangle$ *se llama subargumento de desacuerdo, y el literal* L *será el punto de contra-argumentación.*

Observación 4.4. *Sea* $\langle \mathcal{A}, L \rangle$ *un argumento donde* $\mathcal{A} = \emptyset$*, esto es,* L *tiene una derivación estricta, entonces no existe un contra-argumento para* $\langle \mathcal{A}, L \rangle$*. Del mismo modo,* $\langle \mathcal{A}, L \rangle$ *tampoco puede ser contra-argumento de ningún otro argumento* $\langle \mathcal{A}_1, L_1 \rangle$*, ya que* $\langle \mathcal{A}_1, L_1 \rangle$ *no puede existir por que* $\langle \mathcal{A}_1, L_1 \rangle$ *violaría la condición* (2) *de la definición de argumento.*

La noción de contra-argumento determina cuándo un argumento ataca a otro, capturando la noción de conflicto entre argumentos a través de la negación fuerte. Observe, por ejemplo, que considerando los argumentos $\langle \mathcal{B}_1, \sim comprar\_acciones(acme) \rangle$ y

$\langle \mathcal{B}_2, comprar\_acciones(acme)\rangle$ del Ejemplo 4.2, $\langle \mathcal{B}_1, \sim comprar\_acciones(acme)\rangle$ es un contra-argumento de $\langle \mathcal{B}_2, comprar\_acciones(acme)\rangle$. Es posible identificar dos situaciones de ataque entre argumentos: el *ataque directo* cuando un argumento ataca a otro directamente a su conclusión, y el *ataque interno* que ocurre cuando un argumento ataca a un subargumento propio del argumento atacado.

Observación 4.5. *Para decidir si un ataque tiene éxito y constituye una derrota se necesita de un procedimiento de comparación de argumentos que establezca cuándo un argumento es preferido a otro. Como la comparación entre argumentos puede definirse de varias maneras, en lo que sigue se asume la existencia de una relación de preferencia genérica $\succsim$ entre argumentos que permite distinguir cuándo un argumento es preferido a otro.*

Por ejemplo, si $\langle \mathcal{A}_1, L_1\rangle$ es al menos tan preferido como $\langle \mathcal{A}_2, L_2\rangle$ se lo denotará como $\langle \mathcal{A}_1, L_1\rangle \succsim \langle \mathcal{A}_2, L_2\rangle$. Dados dos argumentos, para distinguir cuándo uno de los argumentos es estrictamente preferido o bien son incomparables, siguiendo la notación de la Definición 3.1, se definen dos relaciones binarias basadas en $\succsim$. Estas relaciones serán de utilidad más adelante en la definición de derrota.

Definición 4.16 (Relaciones $\succ$ y $\asymp$)**.** *Sea Args el conjunto de todos los argumentos que se pueden obtener de un programa DeLP $\mathcal{P}$, y $\langle \mathcal{A}_1, L_1\rangle, \langle \mathcal{A}, L\rangle \in Args$. Se denotará:*

- $\langle \mathcal{A}_1, L_1\rangle \succ \langle \mathcal{A}, L\rangle$*, expresando una preferencia estricta, es decir si* $\langle \mathcal{A}_1, L_1\rangle \succsim \langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{A}, L\rangle \not\succsim \langle \mathcal{A}_1, L_1\rangle$.
- $\langle \mathcal{A}_1, L_1\rangle \asymp \langle \mathcal{A}, L\rangle$*, expresando incomparabilidad si*
  1. $\langle \mathcal{A}_1, L_1\rangle \approx \langle \mathcal{A}, L\rangle$*, o*
  2. $\langle \mathcal{A}_1, L_1\rangle \bowtie \langle \mathcal{A}, L\rangle$

Observación 4.6. *Como siempre se compará un argumento con su contra-argumento, si un literal L tiene una derivación estricta, (i.e.,$\langle \mathcal{A}, L\rangle$ tiene $\mathcal{A} = \emptyset$), entonces $\langle \mathcal{A}, L\rangle$ no será comparado con otro argumento ya que por la Observación 4.4, no puede existir un contra-argumento para $\langle \mathcal{A}, L\rangle$.*

El lector ha podido observar en el Capítulo 3 varios sistemas argumentativos que utilizan preferencias entre argumentos dentro de sus procesos de razonamiento, y donde cada uno provee una forma particular para computarlas. En la mayoría de estos sistemas

el criterio para comparar argumentos es fijo (*i.e.,* no se puede cambiar). A pesar de que algunos sistemas permiten adoptar un criterio (establecido en la configuración del sistema) acorde al dominio que se este representando, no existen aún en la literatura de argumentación formalismos que provean a estos sistemas de mecanismos para que puedan seleccionar y cambiar dinámicamente el criterio de preferencia. En DeLP el criterio puede ser reemplazado de forma modular, por lo que se puede utilizar el criterio más adecuado al problema que se intenta modelar. En la Sección 4.4 se mostrarán ejemplos de diferentes criterios definidos para DeLP.

Si $\langle \mathcal{A}_1, L_1 \rangle$ es un contra-argumento para $\langle \mathcal{A}_2, L_2 \rangle$, es necesario analizar si el ataque de $\langle \mathcal{A}_1, L_1 \rangle$ sobre $\langle \mathcal{A}_2, L_2 \rangle$ es lo suficientemente fuerte como para derrotar a $\langle \mathcal{A}_2, L_2 \rangle$. En consecuencia se deben considerar dos casos, si alguno de ellos es mejor o bien son incomparables. La definición que sigue formaliza esta idea.

Definición 4.17 (Derrota)**.** *Sean $\langle \mathcal{A}_1, L_1 \rangle$ y $\langle \mathcal{A}_2, L_2 \rangle$ dos argumentos. El argumento $\langle \mathcal{A}_1, L_1 \rangle$ es un derrotador para el argumento $\langle \mathcal{A}_2, L_2 \rangle$ en el literal L, si y solo si existe un subargumento $\langle \mathcal{A}, L \rangle$ de $\langle \mathcal{A}_2, L_2 \rangle$ tal que $\langle \mathcal{A}_1, L_1 \rangle$ contra-argumenta a $\langle \mathcal{A}_2, L_2 \rangle$ en el literal L, y vale que:*

- *$\langle \mathcal{A}_1, L_1 \rangle \succ \langle \mathcal{A}, L \rangle$, denotando que el argumento $\langle \mathcal{A}_1, L_1 \rangle$ es un derrotador propio para el argumento $\langle \mathcal{A}_2, L_2 \rangle$; o*
- *$\langle \mathcal{A}_1, L_1 \rangle \asymp \langle \mathcal{A}, L \rangle$, denotando que el argumento $\langle \mathcal{A}_1, L_1 \rangle$ es un derrotador por bloqueo para el argumento $\langle \mathcal{A}_2, L_2 \rangle$.*

Dado el conjunto de argumentos que pueden obtenerse de un programa $\mathcal{P}$, la relación de derrota entre argumentos sólo puede decidir entre dos argumentos en conflicto. Sin embargo, el estado de un argumento $\langle \mathcal{A}_1, L_1 \rangle$ con respecto al programa $\mathcal{P}$, dependerá de la interacción de $\langle \mathcal{A}_1, L_1 \rangle$ con el resto de los argumentos que pueden obtenerse.

Por ejemplo, si a partir de un programa $\mathcal{P}$ se obtiene un argumento $\langle \mathcal{A}_0, L_0 \rangle$ que no tiene derrotadores, entonces un agente que disponga de $\mathcal{P}$ para razonar, podrá "creer" en el literal L_0. Sin embargo, puede ocurrir que $\langle \mathcal{A}_0, L_0 \rangle$ tenga un derrotador $\langle \mathcal{A}_1, L_1 \rangle$, con lo cual existirán razones para invalidar la creencia en L_0. Pero también puede ocurrir que el argumento $\langle \mathcal{A}_1, L_1 \rangle$ tenga a su vez un derrotador $\langle \mathcal{A}_2, L_2 \rangle$, reinstaurando la creencia en L_0. La secuencia de interacciones puede ir más lejos, y a su vez puede existir un derrotador $\langle \mathcal{A}_3, L_3 \rangle$ para $\langle \mathcal{A}_2, L_2 \rangle$, que vuelve a invalidar la creencia en L_0, y así siguiendo. Lo cual da origen una secuencia de argumentos [$\langle \mathcal{A}_0, L_0 \rangle$, $\langle \mathcal{A}_1, L_1 \rangle$, $\langle \mathcal{A}_2, L_2 \rangle$, $\langle \mathcal{A}_3, L_3 \rangle$,. . .] que se llamará línea de argumentación, donde cada elemento es un derrotador de su predecesor.

Definición 4.18 (Línea de argumentación). *Sea $\mathcal{P}$ un programa lógico rebatible y $\langle \mathcal{A}_1, L_1 \rangle$ un argumento obtenido a partir de $\mathcal{P}$. Una línea de argumentación para $\langle \mathcal{A}_1, L_1 \rangle$, es una secuencia de argumentos de $\mathcal{P}$ denotada Λ= [$\langle \mathcal{A}_1, L_1 \rangle$, $\langle \mathcal{A}_2, L_2 \rangle$, $\langle \mathcal{A}_3, L_3 \rangle$,...], donde para cada elemento de la secuencia $\langle \mathcal{A}_i, L_i \rangle$, el siguiente elemento $\langle \mathcal{A}_{i+1}, L_{i+1} \rangle$ es un derrotador para $\langle \mathcal{A}_i, L_i \rangle$.*

En una línea argumentativa para un argumento $\langle \mathcal{A}_1, L_1 \rangle$ los argumentos en posiciones impares juegan el rol de argumentos de soporte (*i.e.,* están a favor de $\langle \mathcal{A}_1, L_1 \rangle$), y los argumentos en posiciones pares actúan como argumentos de interferencia (*i.e.,* actúan en contra de $\langle \mathcal{A}_1, L_1 \rangle$). Para evitar situaciones falaces [Gar00, GS04], en DeLP se introduce la noción de *línea de argumentación aceptable*. Una línea de argumentación Λ es aceptable si cumple las siguientes condiciones: (1) Λ es finita, (2) el conjunto de argumentos de soporte en Λ es no contradictorio y el conjunto de argumentos de interferencia en Λ también es no contradictorio, (3) ningún argumento $\mathcal{A}_j$ en Λ es un subargumento de un argumento $\mathcal{A}_i$ en Λ, $i < j$, y (4) todo derrotador por bloqueo $\mathcal{A}_i$ en Λ solo puede ser seguido por un derrotador propio $\mathcal{A}_{i+1}$ en Λ.

Claramente un argumento podría tener varios derrotadores a partir de un programa $\mathcal{P}$. La presencia de múltiples derrotadores para un mismo argumento produce una ramificación de líneas de argumentación, dando origen a un estructura de árbol de derrotadores. En DeLP esta estructura se denomina *árbol de dialéctica*.

Definición 4.19 (Árbol de dialéctica). *Sea $\langle \mathcal{A}_0, L_0 \rangle$ un argumento obtenido a partir de un programa $\mathcal{P}$. Un árbol de dialéctica para $\langle \mathcal{A}_0, L_0 \rangle$, a partir de $\mathcal{P}$, se denota $\mathcal{T}_{\langle \mathcal{A}_0, L_0 \rangle}$, y se construye de la siguiente forma:*

1. *$\langle \mathcal{A}_0, L_0 \rangle$ es raíz de $\mathcal{T}_{\langle \mathcal{A}_0, L_0 \rangle}$;*
2. *Sea N un nodo del árbol etiquetado $\langle \mathcal{A}_n, L_n \rangle$, y $[\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_i, L_i \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle]$ la secuencia de etiquetas del camino que va desde la raíz hasta el nodo N.*
   Sean $\langle \mathcal{B}_1, Q_1 \rangle$, $\langle \mathcal{B}_2, Q_2 \rangle$, ..., $\langle \mathcal{B}_k, Q_k \rangle$ todos los derrotadores de $\langle \mathcal{A}_n, L_n \rangle$.
   Para cada derrotador $\langle \mathcal{B}_i, Q_i \rangle$ $(1 \leq i \leq k)$, tal que, la línea de argumentación $[\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle, \langle \mathcal{B}_i, Q_i \rangle]$ sea aceptable, existe un nodo N_i hijo de N etiquetado con $\langle \mathcal{B}_i, Q_i \rangle$.
   Si no existe ningún derrotador $\langle \mathcal{B}_i, Q_i \rangle$ en tales condiciones, entonces el nodo N es una hoja.

En un árbol de dialéctica, cada camino desde la raíz a una hoja corresponde a una línea de argumentación aceptable. Note también que todo nodo (excepto el raíz) es un derrotador de su padre, y las hojas son argumentos no derrotados.

El marcado de un árbol de dialéctica es un proceso que marca cada nodo como derrotado "D" o no derrotado "U", de la siguiente forma: los nodos hojas son marcados como "U"; y, un nodo interno es marcado como "D" si tiene al menos un hijo marcado como "U", o como "U" si todos sus hijos están marcados como "D".

Un árbol de dialéctica marcado $\mathcal{T}^*_{\langle \mathcal{A}, L\rangle}$ representa el análisis dialéctico en el cual todos los argumentos construibles a partir de un programa $\mathcal{P}$ que se relacionan con el análisis son considerados a fin de decidir el estado del argumento $\langle \mathcal{A}, L\rangle$. Por lo tanto, el estado del argumento $\langle \mathcal{A}, L\rangle$ será "garantizado" si la raíz tiene como marca "U".

Definición 4.20 (Argumento garantizado)**.** *Sea $\mathcal{P}$ un programa DeLP y $\langle \mathcal{A}, L\rangle$ un argumento construible a partir de $\mathcal{P}$. Se dirá que $\mathcal{P}$ garantiza $\langle \mathcal{A}, L\rangle$ si y solo si el argumento $\langle \mathcal{A}, L\rangle$ se encuentra marcado como "U" en $\mathcal{T}^*_{\langle \mathcal{A}, L\rangle}$.*

Definición 4.21 (Literal garantizado)**.** *Sea $\mathcal{P}$ un programa DeLP y L un literal. Un literal L está garantizado en $\mathcal{P}$ si y solo si existe un argumento $\langle \mathcal{A}, L\rangle$ para L y $\langle \mathcal{A}, L\rangle$ está garantizado en $\mathcal{P}$.*

Observación 4.7. *En DeLP no es posible garantizar al mismo tiempo dos literales complementarios L y $\overline{L}$, ya que ésto lleva directamente a un absurdo. Si L está garantizado, entonces es un derrotador para $\overline{L}$ y viceversa.*

Definición 4.22 (Respuesta a una Consulta Lógica Rebatible)**.** *Dado un programa lógico rebatible $\mathcal{P} = (\Pi, \Delta)$. Sea L un literal que representa una consulta lógica rebatible para un programa $\mathcal{P}$. La respuesta a L será:*

- SI: *si L está garantizado en $\mathcal{P}$.*
- NO: *si $\overline{L}$ está garantizado en $\mathcal{P}$.*
- INDECISO: *si L no está garantizado en $\mathcal{P}$y $\overline{L}$ no está garantizado en $\mathcal{P}$.*
- DESCONOCIDO: *si L no pertenece a la signatura del programa $\mathcal{P}$.*

Observación 4.8. *Por la Proposición 5.1 [GS04], si un literal L tiene una derivación estricta, entonces L está garantizado.*

A continuación se ejemplificarán los conceptos descriptos en esta sección. El siguiente ejemplo se abstrae del criterio para comparar argumentos, y como ya se ha mencionado previamente se asume que existe un criterio, el cual se denota "$\succsim$", que establece las preferencias entre los argumentos.

Ejemplo 4.5. *Considere el siguiente programa DeLP* $\mathcal{P}_{4.5}$*:*

$$\mathcal{P}_{4.5} = \left\{ \begin{array}{ll} a \prec f & \sim b \prec e \\ \sim a \prec b, c & \sim c \prec \\ c \prec d & e \\ b \prec & \\ f & \\ d & \end{array} \right\}$$

A partir del programa anterior es posible construir un argumento $\langle \mathcal{A}, a \rangle$*, con* $\mathcal{A} = \{(a \prec f)\}$*. De hecho, es posible construir un contra-argumento* $\langle \mathcal{B}, \sim a \rangle$*, con* $\mathcal{B} = \{(\sim a \prec b, c), (c \prec d), (b \prec)\}$*, para* $\langle \mathcal{A}, a \rangle$*. Asumiendo que* $\langle \mathcal{B}, \sim a \rangle \succ \langle \mathcal{A}, a \rangle$*,* $\langle \mathcal{B}, \sim a \rangle$ *es un derrotador para* $\langle \mathcal{A}, a \rangle$*.*
El argumento $\langle \mathcal{B}, \sim a \rangle$*, a su vez, tiene dos contra-argumentos* $\langle \mathcal{C}, \sim b \rangle$ *y* $\langle \mathcal{D}, \sim c \rangle$ *que lo atacan indirectamente en los literales* b *y* c *respectivamente, siendo* $\langle \mathcal{E}, b \rangle$ *y* $\langle \mathcal{F}, c \rangle$ *los subargumentos de desacuerdo, donde*

- $\mathcal{C} = \{(\sim b \prec e)\}$
- $\mathcal{D} = \{(\sim c \prec)\}$
- $\mathcal{E} = \{(b \prec)\}$
- $\mathcal{F} = \{(c \prec d)\}$

Asumiendo que $\langle \mathcal{C}, \sim b \rangle \succ \langle \mathcal{E}, b \rangle$ *y que* $\langle \mathcal{D}, \sim c \rangle$ *y* $\langle \mathcal{F}, c \rangle$ *son incomparables, i.e.,* $(\langle \mathcal{D}, \sim c \rangle \not\succ \langle \mathcal{F}, c \rangle$ *y* $\langle \mathcal{F}, c \rangle \not\succ \langle \mathcal{D}, \sim c \rangle)$ *de acuerdo al criterio* $\succsim$*, entonces* $\langle \mathcal{C}, \sim b \rangle$ *es un derrotador propio y* $\langle \mathcal{D}, \sim c \rangle$ *uno por bloqueo para* $\langle \mathcal{B}, \sim a \rangle$*. Por lo tanto, existen dos líneas de argumentación para* $\langle \mathcal{A}, a \rangle$*:*

- $\Lambda_1 = [\langle \mathcal{A}, a \rangle, \langle \mathcal{B}, \sim a \rangle, \langle \mathcal{C}, \sim b \rangle]$
- $\Lambda_2 = [\langle \mathcal{A}, a \rangle, \langle \mathcal{B}, \sim a \rangle, \langle \mathcal{D}, \sim c \rangle]$

Teniendo en cuenta estas líneas se puede construir el árbol de dialéctica $\mathcal{T}_{\langle \mathcal{A}, a\rangle}$ para $\langle \mathcal{A}, a\rangle$, el cual se ilustra en la Figura 4.1-(1). Observe que a partir del proceso de marcado el estado del argumento $\langle \mathcal{A}, a\rangle$ será "garantizado", y la respuesta a la consulta "a" es SI. *La siguiente tabla muestra algunas consultas al programa $\mathcal{P}_{4.5}$, tal que cubren las cuatros respuestas posibles de DeLP. La Figura 4.1 muestra los árboles construidos por el proceso de dialéctica para cada consulta en particular, donde los triángulos representan los argumentos mencionados en este ejemplo, las flechas punteadas son derrotas por bloqueo y las flechas enteras corresponden a derrotas propias.*

Consulta	**Respuesta**
a	SI
b	NO
c	INDECISO
f	DESCONOCIDO

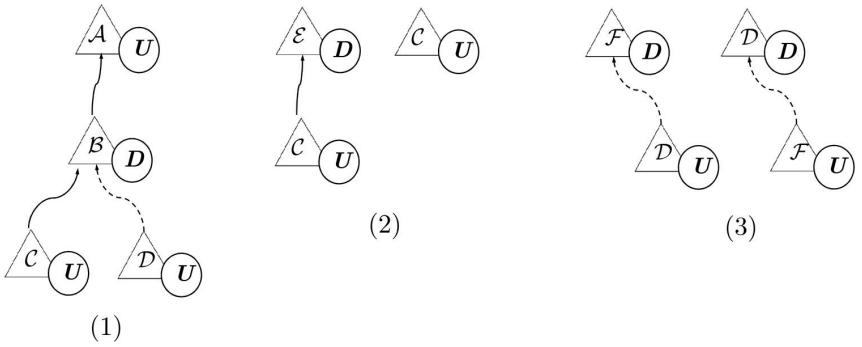

Figura 4.1: Análisis dialéctico ante las consultas por a (1), b (2), y c (3).

4.4. Comparación de Argumentos

Un elemento central en los sistemas argumentativos es la definición de un criterio que permita comparar argumentos [SL92, ACB96, Vre97, AMB00, GS04, GMP12]. Una ventaja de DeLP es la forma modular que tiene para reemplazar el criterio que compara los argumentos, de manera que dado un dominio de aplicación en particular, se puede

utilizar el criterio que se considere más apropiado. Por lo tanto, como la comparación de argumentos se puede definir de diversas formas, para representar un *criterio de preferencia entre argumentos* se utilizará de manera abstracta una relación de preferencia $\succsim_{\mathsf{lc}}$, donde lc es un identificador de criterio. A continuación se muestran algunos ejemplos de especificaciones para la implementación de diferentes criterios que podrían ser utilizados por DeLP.

4.4.1. Especificidad Generalizada

Si bien existen varios criterios concretos, DeLP provee *especificidad generalizada* [SGCS03] el cual utiliza prioridades implícitas como base para la comparación de argumentos. Informalmente, un argumento $\langle \mathcal{A}_1, L_1 \rangle$ es preferido a otro argumento $\langle \mathcal{A}_2, L_2 \rangle$, si $\langle \mathcal{A}_1, L_1 \rangle$ usa más información, o es más directo (usa menos reglas) que $\langle \mathcal{A}_2, L_2 \rangle$.

Definición 4.23 (Especificidad Generalizada)**.** *Sea $\mathcal{P} = (\Theta, \Omega, \Delta)$ un programa DeLP, donde Θ es el conjunto de hechos y Ω el conjunto de reglas estrictas. Sea $\mathcal{F}$ el conjunto de literales que tienen una derivación rebatible a partir de $\mathcal{P}$. Un argumento $\langle \mathcal{A}_1, L_1 \rangle$ es estrictamente más específico que otro argumento $\langle \mathcal{A}_2, L_2 \rangle$(denotado $\langle \mathcal{A}_1, L_1 \rangle \succ_{\mathsf{esp}} \langle \mathcal{A}_2, L_2 \rangle$) si se verifican las siguientes dos condiciones:*

1. *para todo conjunto $H \subseteq \mathcal{F}$,*
   si existe una derivación rebatible para L_1 a partir de $\Omega \cup H \cup \mathcal{A}_1$ (H activa a $\mathcal{A}_1$), y no existe una derivación estricta para L_1 a partir de $\Omega \cup H$, entonces existe una derivación rebatible para L_2 a partir de $\Omega \cup H \cup \mathcal{A}_2$, ($H$ activa a $\mathcal{A}_2$)

2. *existe al menos un conjunto $H' \subseteq \mathcal{F}$ tal que:*
   existe una derivación rebatible para L_2 a partir de $\Omega \cup H' \cup \mathcal{A}_2$ (H' activa a $\mathcal{A}_2$), no existe una derivación estricta de L_2 a partir de $\Omega \cup H'$, y no existe una derivación rebatible de L_1 a partir de $\Omega \cup H' \cup \mathcal{A}_1$ (H' no activa a $\mathcal{A}_1$).

Ejemplo 4.6. *Considere el programa $\mathcal{P}_{4.1}$ del Ejemplo 4.1, todo conjunto H que activa al argumento $\langle \mathcal{A}_1, vuela(tina) \rangle$ donde*

$$\mathcal{A}_1 = \left\{ \; vuela(tina) \prec gallina(tina), asustada(tina) \; \right\}$$

debe contener los literales "gallina(tina)" y "asustada(tina)", por lo tanto H también activa al argumento $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$ donde

$$\mathcal{A}_2 = \left\{ \sim vuela(tina) \prec gallina(tina) \right\}$$

Sin embargo, el conjunto $H' = \{gallina(tina)\}$ *activa a* $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$*, pero no activa a* $\langle \mathcal{A}_1, vuela(tina) \rangle$*. Por lo tanto,* $\langle \mathcal{A}_1, vuela(tina) \rangle$ *es más específico (más informado) que* $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$*.*

Por otra parte, en el caso del argumento $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$ *todo conjunto H contiene al literal "gallina(tina)" que activa al argumento* $\langle \mathcal{A}_3, vuela(tina) \rangle$ *donde* $\mathcal{A}_3 = \{vuela(tina) \prec ave(tina)\}$*. Pero sin embargo, el conjunto* $H' = \{ave(tina)\}$ *activa a* $\langle \mathcal{A}_3, vuela(tina) \rangle$*, pero no activa a* $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$*. Por lo tanto,* $\langle \mathcal{A}_2, \sim vuela(tina) \rangle$ *es más específico (más directo) que* $\langle \mathcal{A}_3, vuela(tina) \rangle$*.*

4.4.2. Especificidad Generalizada Extendida

Como se menciona en secciones anteriores, un programa DeLP $\mathcal{P}$ puede contener reglas rebatibles cuyo cuerpo es vacio; a este tipo de reglas se las denomina presuposiciones. Las presuposiciones representan una afirmación más débil que un hecho. Por lo tanto, cuando se comparan dos argumentos, se deberían preferir argumentos que se basan solo en hechos, por sobre los que contienen presuposiciones. Una alternativa para comparar argumentos es definir un orden entre reglas rebatibles del programa [PS97, GS04]. Sin embargo, un orden entre reglas rebatibles no soluciona el problema anterior, ya que no permite expresar que un hecho es mejor que una presuposición. En [Gar00] se introduce una extensión del criterio especificidad generalizada, la cual se comporta adecuadamente con argumentos que utilizan presuposiciones.

Definición 4.24 (Argumento basado en hechos, o basado en presuposiciones)**.** *Un argumento estará basado en hechos, si al menos un hecho fue utilizado en su derivación rebatible. De lo contrario, si se utilizaron solo presuposiciones, se dirá que está basado en presuposiciones.*

Ejemplo 4.7. *A partir del programa* $\mathcal{P}_{4.2}$ *es posible construir los argumentos basados en hechos* $\langle \mathcal{B}_1, \sim comprar\_acciones(acme) \rangle$ *y* $\langle \mathcal{B}_2, comprar\_acciones(acme) \rangle$ *y los argumentos basado en presuposiciones* $\langle \mathcal{B}_3, \sim comprar\_acciones(acme) \rangle$ *y* $\langle \mathcal{B}_4, empresa\_riesgosa(acme) \rangle$*, donde*

$$\mathcal{B}_1 = \left\{ \begin{array}{l} \sim comprar\_acciones(acme) \prec buen\_precio(acme), empresa\_riesgosa(acme) \\ empresa\_riesgosa(acme) \prec huelga\_empleados(acme) \\ huelga\_empleados(acme) \prec \end{array} \right\}$$

$$\mathcal{B}_2 = \left\{ \; comprar\_acciones(acme) \prec buen\_precio(acme) \; \right\}$$

$$\mathcal{B}_3 = \left\{ \begin{array}{l} \sim comprar\_acciones(acme) \prec empresa\_riesgosa(acme) \\ empresa\_riesgosa(acme) \prec huelga\_empleados(acme) \\ huelga\_empleados(acme) \prec \end{array} \right\}$$

$$\mathcal{B}_4 = \left\{ \begin{array}{l} empresa\_riesgosa(acme) \prec huelga\_empleados(acme) \\ huelga\_empleados(acme) \prec \end{array} \right\}$$

Dado un argumento $\langle \mathcal{A}, L \rangle$, se denotará con $RR(\mathcal{A})$, al conjunto de reglas rebatibles de $\mathcal{A}$. Por lo tanto, $RR(\mathcal{A})$ no contiene presuposiciones. La siguiente definición formaliza al criterio de *especificidad generalizada extendida*.

Definición 4.25 (Especificidad Generalizada Extendida)**.** *Sea $\mathcal{P} = (\Theta, \Omega, \Delta)$ un programa DeLP. Sea $\mathcal{F}$ el conjunto de literales que tienen una derivación rebatible a partir de $\mathcal{P}$. Un argumento $\langle \mathcal{A}_1, L_1 \rangle$ es estrictamente más específico que otro argumento $\langle \mathcal{A}_2, L_2 \rangle$(denotado $\langle \mathcal{A}_1, L_1 \rangle \succ_{\mathsf{espE}} \langle \mathcal{A}_2, L_2 \rangle$) si se verifica alguna de las siguientes tres condiciones:*

(a) $RR(\mathcal{A}_1) \neq \emptyset$, $RR(\mathcal{A}_2) \neq \emptyset$ y se cumplen las dos condiciones siguientes:

1. *para todo conjunto $H \subseteq \mathcal{F}$,*
   si existe una derivación rebatible para L_1 a partir de $\Omega \cup H \cup RR(\mathcal{A}_1)$, y no existe una derivación estricta para L_1 a partir de $\Omega \cup H$, entonces existe una derivación rebatible para L_2 a partir de $\Omega \cup H \cup RR(\mathcal{A}_2)$,
2. *existe al menos un conjunto $H' \subseteq \mathcal{F}$ tal que:*
   existe una derivación rebatible para L_2 a partir de $\Omega \cup H' \cup RR(\mathcal{A}_2)$, no existe una derivación estricta de L_2 a partir de $\Omega \cup H'$, y no existe una derivación rebatible de L_1 a partir de $\Omega \cup H' \cup RR(\mathcal{A}_1)$.

(b) $\langle \mathcal{A}_1, L_1 \rangle$ está basada en hechos, y $\langle \mathcal{A}_2, L_2 \rangle$ está basada en presuposiciones.

(c) Los argumentos $\langle \mathcal{A}_1, L_1 \rangle$ y $\langle \mathcal{A}_2, L_2 \rangle$ están ambas basadas en presuposiciones, y se cumple $RR(\mathcal{A}_1) = \emptyset$ y $RR(\mathcal{A}_2) \neq \emptyset$.

Ejemplo 4.8. *Considere el programa $\mathcal{P}_{4.2}$ del Ejemplo 4.2, por la condición (a) el argumento $\langle \mathcal{B}_1, \sim comprar\_acciones(acme) \rangle$ donde*

$$\mathcal{B}_1 = \left\{ \begin{array}{l} \sim comprar\_acciones(acme) \prec buen\_precio(acme), empresa\_riesgosa(acme) \\ empresa\_riesgosa(acme) \prec huelga\_empleados(acme) \\ huelga\_empleados(acme) \prec \end{array} \right\}$$

es más específico que $\langle \mathcal{B}_2, comprar\_acciones(acme) \rangle$ *donde*

$$\mathcal{B}_2 = \left\{ \; comprar\_acciones(acme) \prec buen\_precio(acme) \; \right\}$$

,

ya que $\langle \mathcal{B}_1, \sim comprar\_acciones(acme) \rangle$ *usa más información al basarse en "buen_precio(acme)" y "empresa_riesgosa(acme)", mientras que* $\langle \mathcal{B}_2, comprar\_acciones(acme) \rangle$ *solo se basa en "buen_precio(acme)". Sin embargo, observe que la condición* (a) *puede fallar. En el caso de* $\langle \mathcal{B}_2, comprar\_acciones(acme) \rangle$ *y el contra-argumento* $\langle \mathcal{B}_3, \sim comprar\_acciones \rangle$ *donde*

$$\mathcal{B}_3 = \left\{ \begin{array}{l} \sim comprar\_acciones(acme) \prec empresa\_riesgosa(acme) \\ empresa\_riesgosa(acme) \prec huelga\_empleados(acme) \\ huelga\_empleados(acme) \prec \end{array} \right\}$$

,

por la condición (a) *son incomparables, sin embargo por condición* (b), $\langle \mathcal{B}_2, comprar\_acciones(acme) \rangle$ *es más específico que* $\langle \mathcal{B}_3, \sim comprar\_acciones \rangle$ *por que el primero está basado en hechos, mientras que el segundo está basado en presuposiciones.*

Finalmente, observe que considerando el programa $\mathcal{P}_{4.5}$ *del Ejemplo* 4.5*, y para el argumento* $\langle \mathcal{E}, b \rangle$*, con* $\mathcal{E} = \{(b \prec)\}$*, y el contra-argumento* $\langle \mathcal{C}, \sim b \rangle$*, con* $\mathcal{C} = \{(\sim b \prec e)\}$*, ocurre que* $RR(\mathcal{E}) = \emptyset$ *y por lo tanto la condición* (a) *no puede utilizarse, sin embargo, por la condición (b),* $\langle \mathcal{C}, \sim b \rangle$ *es preferido a* $\langle \mathcal{E}, b \rangle$ *por que se cumple que* $\langle \mathcal{C}, \sim b \rangle$ *está basado en hechos mientras que* $\langle \mathcal{E}, b \rangle$ *está basado en presuposiciones.*

Note que en [MGS12] se introduce PreDeLP, un formalismo que extiende a DeLP dotándolo con la capacidad para tratar con presuposiciones. A diferencia del criterio para evaluar argumentos presentado en esta sección, los autores proponen un criterio el cual expresa que si un argumento usa un subconjunto de las presuposiciones que usa otro argumento, entonces el primero es preferido sobre el segundo. Esta evaluación de argumentos sigue el siguiente principio: *la acumulación de conocimiento presuntivo en una prueba debilita la prueba misma como así también a su conclusión.*

4.4.3. Comparación basada en literales

Otra alternativa para comparar argumentos es la propuesta en [FEGS08]. Los autores introducen un marco para toma de decisiones basado en argumentación, en consecuencia utilizan un criterio de preferencia apropiado a su modelo basado en prioridad entre

literales. El criterio de preferencia entre argumentos utilizado, se basa en definir prioridades entre los criterios de preferencia del agente (tomador de decisión). Para ello, a cada criterio de preferencia se le asocia un literal distinguido y al realizar cambios en las relaciones de preferencia (expresadas explícitamente) entre los mismos, se pueden cambiar de manera flexible las preferencias del agente y el criterio de preferencia entre argumentos. De esta manera, un conjunto $\mathcal{C}$ de *literales de comparación*, preestablecido de antemano, será distinguido de entre los literales que aparecen en el programa DeLP. Entonces, las preferencias del tomador de decisión serán establecidas explícitamente usando un orden de preferencia (prioridad) entre los elementos de $\mathcal{C}$. Sea $\mathcal{P}$ un programa DeLP y sea $\mathcal{C}$ un conjunto de literales de comparación de $\mathcal{P}$. Un L-orden " $>$" sobre $\mathcal{P}$ es un orden parcial estricto sobre los elementos de $\mathcal{C}$.

Asumiendo que existe un L-orden "$>$" sobre literales de un programa DeLP dado, en [FEGS08] introducen un criterio que utiliza dicho orden para decidir cuándo un argumento es mejor que otro. El proceso para decidir entre dos argumentos consiste en evaluar los literales de las reglas rebatibles que los componen, tal que los literales con mayor prioridad con respecto a "$>$" son preferidos.

Definición 4.26 (Comparación basada en Literales)**.** *Sea $\mathcal{P}$ un programa DeLP. Sea $\mathcal{C}$ el conjunto de literales de comparación de $\mathcal{P}$, y sea " $>$" un L-orden sobre $\mathcal{C}$. Dado un par de argumentos en conflicto $\langle \mathcal{A}_1, L_1 \rangle$ y $\langle \mathcal{A}_2, L_2 \rangle$, el argumento $\langle \mathcal{A}_1, L_1 \rangle$ será estrictamente preferido a $\langle \mathcal{A}_2, L_2 \rangle$(denotado $\langle \mathcal{A}_1, L_1 \rangle \succ_{\mathsf{prioL}} \langle \mathcal{A}_2, L_2 \rangle$), si y solo si:*

1. *hay dos literales L_1 y L_2 tal que $L_1 \in^* \mathcal{A}_1$, $L_2 \in^* \mathcal{A}_2$, $L_1 > L_2$, y*
2. *y no hay literales L'_1 y L'_2 tal que $L'_1 \in^* \mathcal{A}_1$, $L'_2 \in^* \mathcal{A}_2$, y $L'_2 > L'_1$.*

Notación: $L \in^* \mathcal{A}$ si y solo si existe una regla rebatible $(L_0 \prec L_1, L_2, \ldots, L_n)$ en $\mathcal{A}$ y $L = L_i$ para algún $i(1 \leq i \leq n)$.

Ejemplo 4.9. *Considere el dominio de aplicación descripto en [FEGS08] el cual consiste de un escenario donde un robot debe obtener una recomendación sobre cual será la próxima caja mas conveniente a seleccionar con el objetivo de llevarla a un lugar en particular denominado depósito. Considere a continuación parte del programa $\mathcal{P}_k$ presentado en [FEGS08].*

$$better(Box, Obox) \prec nearer\_robot(Box, Obox)$$
$$\sim better(Box, Obox) \prec nearer\_store(Box, Obox)$$
$$nearer\_robot(box1, box4)$$
$$nearer\_store(box4, box1)$$

A partir de $\mathcal{P}_k$ *pueden construirse los argumentos* $\langle \mathcal{E}_1, better(box4, box1)\rangle$ *y* $\langle \mathcal{E}_2, \sim better(box4, box1)\rangle$*, donde*

$$\mathcal{E}_1 = \{better(box4\,, box1\,) \prec nearer\_store(box4\,, box1\,)\}$$
$$\mathcal{E}_2 = \{\sim better(box4\,, box1\,) \prec nearer\_robot(box1\,, box4\,)\}$$

Supongamos un conjunto de literales de comparación $\mathcal{C}_k = \{nearer\_robot(.,.), nearer\_store(.,.)\}$*, y un L-orden sobre el conjunto* $\mathcal{C}_k$*, el cual incluirá (de manera esquemática con variables [Lif96]) la siguiente prioridad sobre las cajas,*

$$nearer\_robot(Z, W) > nearer\_store(W, Y).$$

Utilizando el criterio de preferencia basada en literales el argumento $\langle \mathcal{E}_2, \sim better(box4, box1)\rangle$ *es preferido a* $\langle \mathcal{E}_1, better(box4, box1)\rangle$*.*
Considere ahora el Ejemplo 4.3, y suponga que para diagnosticar una enfermedad un especialista en un contexto en particular podría priorizar conocer los síntomas del paciente a conocer bajo qué tratamiento médico se encuentra dicho paciente. Para reflejar esta preferencia, un L-orden $>$ *sobre el conjunto de literales* $\mathcal{C}_e = \{sintC(.,.), trat(.,.)\}$ *podría incluir la siguiente prioridad:*

$$sintC(S, E) > trat(T, E)$$

Desde el programa $\mathcal{P}_{4.3}$ *puede construirse el argumento* $\langle \mathcal{A}_2, enferm(e1)\rangle$*, con* $\mathcal{A}_2 = \{(enferm(e1\,) \prec sintC(s1\,, e1\,)), (sintC(s1\,, e1\,) \prec\,)\}$ *y el contra-argumento* $\langle \mathcal{A}_3, \sim enferm(e1)\rangle$ *con* $\mathcal{A}_3 = \{(\sim enferm(e1\,) \prec trat(t1\,, e1\,))\}$*, donde el primero es preferido ya que contiene un literal de comparación con mayor prioridad que uno del segundo argumento.*

4.4.4. Prioridad entre Reglas

Como se indicó en el capítulo anterior, para comparar argumentos en algunos sistemas como [PS97] se asume un orden explícito (o prioridades) entre las reglas del programa. En

dichos sistemas, para decidir entre dos literales contradictorios, las dos reglas que tienen a estos literales en sus cabezas son comparadas. El literal que prevalece es el que está en la regla de mayor prioridad. La formulación de DeLP impide que se puedan comparar reglas individuales, ya que se deben comparar argumentos entre sí.

En [Gar00, GS04] se presenta un criterio que asume un orden definido explícitamente entre reglas rebatibles. Resolver el conflicto entre argumentos involucra realizar un análisis sobre el conjunto de reglas rebatibles que los componen, tal que las reglas de mayor prioridad son preferidas. El orden se establece entre reglas rebatibles, ya que no debería haber reglas estrictas mejores que otras.

Definición 4.27 (Prioridad entre Reglas)**.** *Sea $\mathcal{P}$ un programa DeLP y " $>$" un orden parcial definido explícitamente sobre las reglas rebatibles de $\mathcal{P}$. Sean $\langle \mathcal{A}_1, L_1 \rangle$ y $\langle \mathcal{A}_2, L_2 \rangle$ dos argumentos obtenidas a partir de $\mathcal{P}$. El argumento $\langle \mathcal{A}_1, L_1 \rangle$ será estrictamente preferido sobre $\langle \mathcal{A}_2, L_2 \rangle$(denotado $\langle \mathcal{A}_1, L_1 \rangle \succ_{\mathsf{prioR}} \langle \mathcal{A}_2, L_2 \rangle$) cuando:*

1. *exista al menos una regla $r_a \in \mathcal{A}_1$, y una regla $r_b \in \mathcal{A}_2$, tal que $r_a > r_b$, y*
2. *no exista ninguna regla $r'_b \in \mathcal{A}_2$, y $r'_a \in \mathcal{A}_1$, tal que $r'_b > r'_a$.*

Es importante destacar que por una cuestión de simplicidad, en algunos ejemplos, las prioridades se indicarán referenciando a cada regla través de una etiqueta r_i. En estos casos, se denotará una regla como $r : L_0 \prec L_1, L_2, \ldots, L_n$.

Ejemplo 4.10. *Considere el programa $\mathcal{P}_{4.2}$ y suponga que se establece una prioridad entre las reglas expresando que:*

$$(comprar\_acciones(T) \prec buen\_precio(T)) > (\sim comprar\_acciones(T) \prec empresa\_riesgosa(T))$$

Utilizando el criterio de prioridad entre reglas, el argumento $\langle \mathcal{B}_2, comprar\_acciones(acme) \rangle$ donde

$$\mathcal{B}_2 = \{comprar\_acciones(acme) \prec buen\_precio(acme)\}$$

será preferido a $\langle \mathcal{B}_3, \sim comprar\_acciones \rangle$ donde

$$\mathcal{B}_3 = \{\ \sim comprar\_acciones(acme) \prec empresa\_riesgosa(acme)\ \}.$$

Considere nuevamente el programa $\mathcal{P}_{4.3}$ del Ejemplo 4.3, y el orden entre reglas " $>$" que solo incluirá la prioridad $r_1 > r_2$, donde

$r_1 : enferm(E) \prec sintC(S, E).$

$r_2 : \sim enferm(E) \prec sintC(S, E), result(pruebaD(P, E), V), V = negativo$

Utilizando el criterio de prioridad entre reglas, el argumento $\langle \mathcal{A}_1, enferm(e1)\rangle$ *con*

$$\mathcal{A}_1 = \left\{ \begin{array}{l} enferm(e1) \prec sintC(s1, e1) \\ sintC(s1, e1) \prec \end{array} \right\}$$

será preferido al argumento $\langle \mathcal{A}_2, \sim enferm(e1)\rangle$ *con*

$$\mathcal{A}_2 = \left\{ \begin{array}{l} \sim enferm(e1) \prec sintC(s1, e1), result(pruebaD(p1, e1), V), V = negativo \\ sintC(s1, e1) \prec \end{array} \right\}$$

ya que el primero tiene una regla con prioridad sobre otra del segundo, y no ocurre lo mismo a la inversa.

A continuación se presenta una variante del criterio de Prioridad entre Reglas donde se prefiere aquellos argumentos basados en hechos sobre los que se basan en presuposiciones.

Definición 4.28 (Prioridad entre Reglas Extendida)**.** *Sea* $\mathcal{P}$ *un programa DeLP. Sean* $\langle \mathcal{A}_1, L_1\rangle$ *y* $\langle \mathcal{A}_2, L_2\rangle$ *dos argumentos obtenidas a partir de* $\mathcal{P}$*. El argumento* $\langle \mathcal{A}_1, L_1\rangle$ *será estrictamente preferido sobre* $\langle \mathcal{A}_2, L_2\rangle$ *(denotado* $\langle \mathcal{A}_1, L_1\rangle \succ_{\mathsf{prioRE}} \langle \mathcal{A}_2, L_2\rangle$*) si:*

1. $\langle \mathcal{A}_1, L_1\rangle$ *es preferido a* $\langle \mathcal{A}_2, L_2\rangle$ *por el criterio de prioridad entre reglas (*$\langle \mathcal{A}_1, L_1\rangle \succ_{\mathsf{prioR}} \langle \mathcal{A}_2, L_2\rangle$*), y*

2. *no se cumple que* $\langle \mathcal{A}_1, L_1\rangle$ *es un argumento basado en presuposiciones y* $\langle \mathcal{A}_2, L_2\rangle$ *es un argumento basado en hechos.*

Ejemplo 4.11. *Siguiendo con el Ejemplo 4.10, utilizando el criterio de Prioridad entre Reglas el argumento* $\langle \mathcal{A}_1, enferm(e1)\rangle$ *es preferido a* $\langle \mathcal{A}_2, \sim enferm(e1)\rangle$*. Sin embargo, si ahora se utiliza el criterio Prioridad entre Reglas Extendido y como* $\langle \mathcal{A}_1, enferm(e1)\rangle$ *es un argumento basado en presuposiciones y* $\langle \mathcal{A}_2, \sim enferm(e1)\rangle$ *basado en hechos tenemos que* $\langle \mathcal{A}_1, enferm(e1)\rangle \not\succ_{prioRE} \langle \mathcal{A}_2, \sim enferm(e1)\rangle$*.*

4.5. Ejemplo de Aplicación

Para ejemplificar cómo se puede aplicar DeLP para resolver problemas del mundo real el dominio de aplicación que se utilizará en esta sección consiste en un computador a

bordo instalado en un vehículo y programado para ofrecer al conductor recomendaciones tales como sugerir un hotel cercano. Las sugerencias del sistema se basarán en información del contexto, por ejemplo información de las zonas donde se encuentran los hoteles cercanos, la cual es obtenida dinámicamente durante el viaje. Toda información que el sistema utilice en sus sugerencias estará representada por un programa DeLP. Por ejemplo, el programa DeLP $\mathcal{P}_h = (\Pi_h, \Delta_h)$ (ver Figura 4.2), presenta información para sugerir un hotel a partir de la consulta "*sugerir*" realizada por el conductor del vehículo.

$$\Pi_h = \left\{ \begin{array}{l} mParadas \\ mHManejando \\ deNoche \\ hCercano(h1) \\ estrellas(h1, 5) \\ zRobos(h1) \\ cVehicular \leftarrow tLento \end{array} \right\} \quad \Delta_h = \left\{ \begin{array}{l} sugerir(X) \prec bHotel(X), hCercano(X) \\ sugerir(X) \prec sParar, hCercano(X) \\ \sim sugerir(X) \prec zPeligrosa(X) \\ zPeligrosa(X) \prec zRobos(X) \\ \sim zPeligrosa(X) \prec zPolicias(X) \\ bHotel(X) \prec estrellas(X, S), S \geq 3 \\ \sim bHotel(X) \prec estrellas(X, S), S < 3 \\ \sim sParar \prec mParadas \\ sParar \prec mHManejando \\ sParar \prec deNoche \end{array} \right\}$$

Figura 4.2: Programa DeLP $\mathcal{P}_h$ con información para sugerir hoteles.

Observe que el conjunto Π_h tiene seis hechos y una regla estricta. Estos hechos representan información sobre la situación actual que puede ser automáticamente obtenida durante el viaje: se realizaron muchas paradas (mParadas), el conductor ha manejado muchas horas (mHManejando), se está manejando de noche (deNoche), "$h1$" es un hotel cercano (hCercano), "$h1$" es un hotel de cinco estrellas (estrellas($h1$,5)), y "$h1$" está en una zona de robos (zRobos($h1$)). La regla estricta representa que "hay congestión vehicular cuando el tránsito es muy lento".

El conjunto Δ_h contiene varias reglas rebatibles. Las primeras dos reglas representan razones para sugerir un hotel: "si es cercano y es un buen hotel" o "si es un hotel cercano y hay una razón que sugiere parar". Las últimas tres reglas representan razones a favor y en contra de sugerir dejar de conducir para descansar, siendo que "si el conductor lleva muchas horas manejando o está manejando de noche éstas son razones para sugerir una

parada", mientras que "si durante el viaje el conductor ha realizado varias paradas ésto es una razón en contra de sugerir que el conductor pare". Observe que la sexta y séptima reglas rebatibles son utilizadas para establecer si un hotel es un buen hotel. La tercer regla expresa que "si un hotel está ubicado en una zona peligrosa entonces hay una razón tentativa para no sugerirlo". Finalmente, la cuarta y quinta reglas pueden ser leídas de la siguiente manera: "un hotel está ubicado en una zona peligrosa si en la zona normalmente se reportan muchos casos de robos", y por lo contrario "si la zona generalmente cuenta con presencia policial, entonces el hotel no está ubicado en una zona peligrosa".

Además de esta información, el sistema del vehículo también requiere para decidir qué recomendar contar con ciertos criterios para comparar argumentos. Por ejemplo, se asume que el sistema dispone de dos criterios; uno basado en la seguridad, denotado $\succsim_{\mathsf{seg}}$, y otro basado en el confort, denotado $\succsim_{\mathsf{conf}}$. Intuitivamente,

- El criterio $\succsim_{\mathsf{seg}}$ significa que *el sistema del vehículo preferirá argumentos que contienen información que favorece la seguridad del conductor o el vehículo.*
- El criterio $\succsim_{\mathsf{conf}}$ significa que *el sistema del vehículo preferirá argumentos que contienen información que promueven el confort del conductor.*

Para implementar estos criterios se utilizará el criterio de prioridad entre reglas introducido en [GS04] (ver Definición 4.27), y presentado en la sección anterior, el cual asume un orden " $>$ " definido explícitamente entre reglas rebatibles. El significado informal de "$r_1 > r_2$" es que la regla r_1 es mejor que r_2. Para resolver el conflicto entre argumentos, las reglas de mayor prioridad son preferidas.

En la Figura 4.3 se presenta un orden de prioridad entre reglas asociado al criterio $\succsim_{\mathsf{seg}}$ y $\succsim_{\mathsf{conf}}$, respectivamente.

$(\sim sugerir(X) \prec zPeligrosa(X)) >_{\mathsf{seg}} (sugerir(X) \prec bHotel(X), hCercano(X))$

$(\sim sugerir(X) \prec zPeligrosa(X)) >_{\mathsf{seg}} (sugerir(X) \prec sParar, hCercano(X))$

$(\sim sParar \prec mParadas) >_{\mathsf{seg}} (sParar \prec mHManejando)$

$(sParar \prec deNoche) >_{\mathsf{seg}} (\sim sParar \prec mParadas)$

$(zPeligrosa(X) \prec zRobos(X)) >_{\mathsf{seg}} (\sim zPeligrosa(X) \prec zPolicias(X))$

(a) *orden de prioridad* $>_{\mathsf{seg}}$ *para el criterio* $\succeq_{\sim\mathsf{seg}}$.

$(sugerir(X) \prec bHotel(X), hCercano(X)) >_{\mathsf{conf}} (\sim sugerir(X) \prec zPeligrosa(X))$

$(sugerir(X) \prec sParar, hCercano(X)) >_{\mathsf{conf}} (\sim sugerir(X) \prec zPeligrosa(X))$

$(\sim sParar \prec mParadas) >_{\mathsf{conf}} (sParar \prec mHManejando)$

(b) *orden de prioridad* $>_{\mathsf{conf}}$ *para el criterio* $\succeq_{\sim\mathsf{conf}}$.

Figura 4.3: Prioridades sobre las reglas rebatibles del programa $\mathcal{P}_h$.

Considere el criterio $\succeq_{\sim\mathsf{seg}}$, y la consulta "$sugerir(h1)$" para el programa DeLP $\mathcal{P}_h$ introducido en esta sección. A partir de $\mathcal{P}_h$ es posible encontrar varios argumentos para "$sugerir(h1)$". A su vez, es posible construir varios derrotadores para estos argumentos, y así sucesivamente hasta construir los árboles de dialéctica necesarios para determinar si "$sugerir(h1)$" está garantizado. La Figura 4.4-(a) muestra una representación gráfica del análisis dialéctico resultante para "$sugerir(h1)$". En la Figura 4.4, cada argumento está representado con forma de triángulo (para la estructura detallada de estos argumentos ver Figura 4.5), las líneas punteadas denotan relaciones de derrota por bloqueo, y las líneas enteras relaciones de derrota propias. En particular, en la Figura 4.4-(a) puede verse que el argumento raíz del árbol de dialéctica que se encuentra más a la derecha, el cual soporta la conclusión "$\sim sugerir(h1)$", está marcado con "U"; por lo tanto la conclusión "$sugerir(h1)$" no está garantizada, y la respuesta para la consulta es NO.

Suponga ahora que se realiza la consulta "$sugerir(h1)$" para el programa $\mathcal{P}_h$, pero se utiliza el criterio $\succeq_{\sim\mathsf{conf}}$ que prioriza información que favorece el confort del conductor. La Figura 4.4-(b) muestra la representación gráfica de los árboles de dialéctica resultantes para "$sugerir(h1)$", tal que la conclusión "$sugerir(h1)$" está garantizada, y la respuesta para "$sugerir(h1)$" será SI.

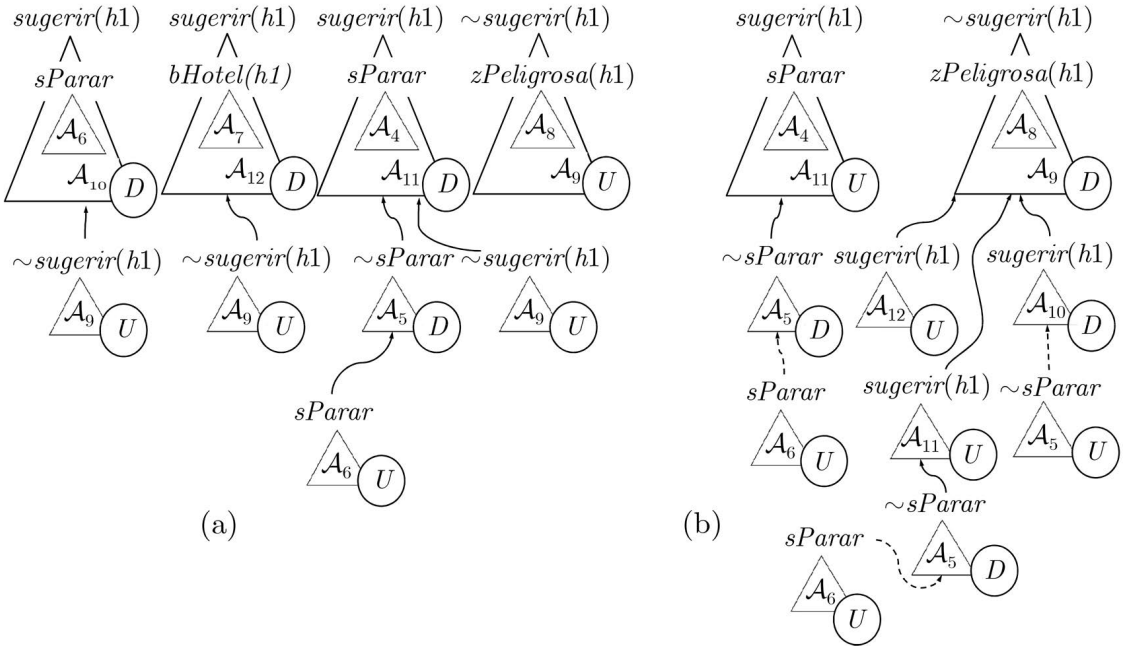

Figura 4.4: Análisis de dialéctica para la consulta $sugerir(h1)$.

$\langle \mathcal{A}_{12}, sugerir(h1)\rangle$, donde $\mathcal{A}_{12} = \left\{ \begin{array}{l} sugerir(h1) \prec bHotel(h1), hCercano(h1) \\ bHotel(h1) \prec estrellas(h1,5), 5 \geq 3 \end{array} \right\}$

$\langle \mathcal{A}_{11}, sugerir(h1)\rangle$, donde $\mathcal{A}_{11} = \left\{ \begin{array}{l} sugerir(h1) \prec sParar, hCercano(h1) \\ sParar \prec mHManejando \end{array} \right\}$

$\langle \mathcal{A}_{10}, sugerir(h1)\rangle$, donde $\mathcal{A}_{10} = \left\{ \begin{array}{l} sugerir(h1) \prec sParar, hCercano(h1) \\ sParar \prec deNoche \end{array} \right\}$

$\langle \mathcal{A}_{9}, \sim sugerir(h1)\rangle$, donde $\mathcal{A}_{9} = \left\{ \begin{array}{l} \sim sugerir(h1) \prec zPeligrosa(h1) \\ zPeligrosa(h1) \prec zRobos(h1) \end{array} \right\}$

$\langle \mathcal{A}_{8}, zPeligrosa(h1)\rangle$, donde $\mathcal{A}_{8} = \{zPeligrosa(h1) \prec zRobos(h1)\}$

$\langle \mathcal{A}_{7}, bHotel(h1)\rangle$, donde $\mathcal{A}_{7} = \{bHotel(h1) \prec estrellas(h1,5), 5 \geq 3\}$

$\langle \mathcal{A}_{6}, sParar\rangle$, donde $\mathcal{A}_{6} = \{sParar \prec deNoche\}$

$\langle \mathcal{A}_{5}, \sim sParar\rangle$, donde $\mathcal{A}_{5} = \{\sim sParar \prec mParadas\}$

$\langle \mathcal{A}_{4}, sParar\rangle$, donde $\mathcal{A}_{4} = \{sParar \prec mHManejando\}$

Figura 4.5: Argumentos considerados como parte del análisis dialéctico mostrado en Figura 4.4-(a) y 4.4-(b).

De esta manera, en esta sección se presentó un dominio de aplicación concreto sobre el cual ejemplificar el proceso de razonamiento que lleva a cabo el sistema de DeLP para responder sus consultas. Asimismo, el foco de la sección se ha centrado en resaltar cómo el comportamiento del mecanismo de razonamiento de DeLP varia según el criterio que se haya elegido. Observe que las relaciones de derrota que surgen de utilizar los criterios $\succeq_{\mathsf{seg}}$ y $\succeq_{\mathsf{conf}}$ difieren, por lo tanto diferentes árboles son construidos por el proceso de dialéctica, los cuales son mostrados en la Figura 4.4-(a) y la Figura 4.4-(b), respectivamente. Por ejemplo, para el caso donde se utiliza el criterio $\succeq_{\mathsf{seg}}$ el argumento $\langle \mathcal{A}_9, \sim sugerir(h1) \rangle$ prevalece en todos los conflictos en donde participa, de hecho no existe un argumento que lo derrote, sin embargo, para el caso donde $\succeq_{\mathsf{conf}}$ ha sido el criterio utilizado el mismo argumento tiene asociado un árbol de dialéctica con varios argumentos que lo derrotan. En este sentido, el ejemplo de aplicación también ha mostrado, mediante las diferentes respuestas para la consulta "$sugerir(h1)$", cómo el criterio de preferencia entre argumentos está directamente relacionado con las inferencias que el sistema obtiene.

En la siguiente sección se presenta el concepto de *Servicio de Razonamiento basado en Programación Lógica Rebatible* el cual fue introducido en [GRTS07] y luego formalizado en [Tuc12]. En particular, este tipo de servicio fue desarrollado con el objetivo de proveer la base para la implementación de servicios de razonamiento argumentativo como parte de la infraestructura basada en conocimiento de sistemas multi-agentes.

4.6. Servicios de Razonamiento basados en Programación Lógica Rebatible

Una característica esencial de los Sistemas Multi-Agente es la interacción entre sus integrantes. Los agentes pertenecientes a estos sistemas interactúan con el objetivo de llevar a cabo tareas, ya sean individuales o colectivas. Los agentes deliberativos generalmente razonan utilizando dos tipos de conocimiento: uno público que es el que comparten con otros agentes y otro privado que surge en parte de su percepción del mundo. En algunos casos el conocimiento que se maneja es incompleto o potencialmente contradictorio lo cual brinda un escenario ideal para sistemas argumentativos [GS04, ACGS08, CS09] capaces de representar el conocimiento y definir el razonamiento de agentes inteligentes.

En [GRTS07] se propone un formalismo que permite a los agentes compartir conocimiento con otros agentes y representar el conocimiento privado de cada uno. En este

trabajo se introduce la noción de servicio de razonamiento basado en programación lógica rebatible a partir del concepto formal de servidor de razonamiento en Programación Lógica Rebatible (DeLP-server). El foco en [GRTS07] es el diseño e implementación de un modelo cliente-servidor basado en programación lógica rebatible que provea un formalismo de representación de conocimiento como así también un servicio de razonamiento basado en argumentación rebatible.

Los DeLP-servers se caracterizan por permitir representar información o conocimiento público y responder consultas de clientes utilizando dicho conocimiento público junto con información privada provista por el cliente. Básicamente un DeLP-server está constituido por:

1. el conocimiento público que será utilizado como base para la resolución de las consultas de los clientes;

2. un mecanismo de inferencia que utilizará ese conocimiento para computar las respuestas a las consultas de los clientes; y

3. un conjunto de operadores de tratamiento de contexto mediante los cuales se podrá, por ejemplo, agregar en forma temporal información para crear el contexto en el cual se procesará la consulta del cliente.

Por lo tanto, para responder consultas, los DeLP-servers tienen almacenado conocimiento público que es utilizado junto al conocimiento privado del agente cliente para crear un contexto adecuado a la consulta. Una *Consulta Contextual*, permite a los clientes incluir en sus consultas conocimiento privado y especificar qué tratamiento debe darse a dicho conocimiento al momento de integrarse con el conocimiento público almacenado en el servidor. Esta información es lo que se denomina contexto de la consulta y, en general, representa conocimiento propio de los clientes.

Por ejemplo, un DeLP-server podría implementar un servicio de recomendación de hoteles, el cual se podría aplicar al dominio presentado en el ejemplo de la Sección 4.5. Los usuarios accederían a este servicio mediante un sistema instalado a bordo en el vehículo consultando por sugerencias sobre qué hotel cercano elegir para hospedarse. Está claro que una consulta como "¿Qué hotel me recomienda?" puede ser demasiado general para que el servicio genere una respuesta satisfactoria para el usuario. En cambio, si se utiliza información obtenida durante el viaje, esto provee, además de la consulta, un contexto, el cual puede ser utilizado por el servicio para realizar una sugerencia que sea satisfactoria.

Por otro parte, en otro dominio de aplicación como el propuesto en el Ejemplo 4.3 de la Sección 4.2, un médico podría enviar en el contexto de la consulta los síntomas que presenta un paciente que deberían tenerse en cuenta al momento de dar un diagnóstico médico. Los operadores para el procesamiento del conocimiento privado permiten un manejo adecuado de la información enviada. Por ejemplo, una consulta para establecer el diagnóstico de una enfermedad, podría incluir un contexto en el cual se indique un operador para excluir determinados síntomas, y dar prioridad a ciertos resultados de estudios clínicos realizados por el paciente.

Finalmente, para responder las consultas contextuales, los DeLP-servers utilizarán un análisis argumentativo rebatible considerando tanto el conocimiento público almacenado como el conocimiento privado contextual enviado por los clientes. En la siguiente sección se procederá a describir cómo los DeLP-servers trabajan.

4.6.1. DeLP-Servers y Consultas Contextuales

El motor de razonamiento de DeLP define el módulo de inferencia de los DeLP-servers. Así, un DeLP-server provee el soporte para implementar servicios de razonamiento argumentativos que trabajan sobre bases de conocimiento representadas como programas lógicos rebatibles. Además de responder consultas DeLP a partir de una base de conocimiento en particular, estos servidores ofrecen la capacidad adicional de complementar las consultas con un fragmento adicional de programa lógico rebatible. Este programa actuará como contexto privado para las consultas, permitiendo que un DeLP-server pueda responder estas consultas contextualizadas luego de modificar, a partir de este contexto privado, el conocimiento que tiene almacenado. Un aspecto importante con respecto a las consultas contextuales es que los cambios que éstas hacen sobre el programa DeLP público almacenado en el servidor no son permanentes. Es decir, las modificaciones realizadas por el contexto privado de una consulta desaparecen cuando finaliza el proceso llevado a cabo por el DeLP-server para responder esta consulta contextualizada.

Una consulta contextual provee al servidor una consulta DeLP, información contextual en la forma de programa DeLP y la información de cómo integrar esta información con el programa almacenado en el servidor. Para resolver estas consultas, un DeLP-server utiliza tanto el conocimiento que tiene almacenado como el contexto que recibe junto a la consulta. La Figura 4.6 muestra la estructura conceptual del modelo cliente-servidor propuesto a través de los DeLP-servers. En particular, en esta figura se observa la consulta

contextual enviada a un DeLP-server por un cliente. El DeLP-server recibe la consulta y un contexto para dicha consulta. Este contexto se integra al programa alojado en el servidor aplicando alguno de los operadores, que serán introducidos a la brevedad, para programas DeLP disponibles en dicho servidor. El programa que resulta de esta operación junto con la consulta recibida serán utilizados por el intérprete DeLP para retornar la respuesta que finalmente dará el DeLP-server ante la consulta contextual en cuestión.

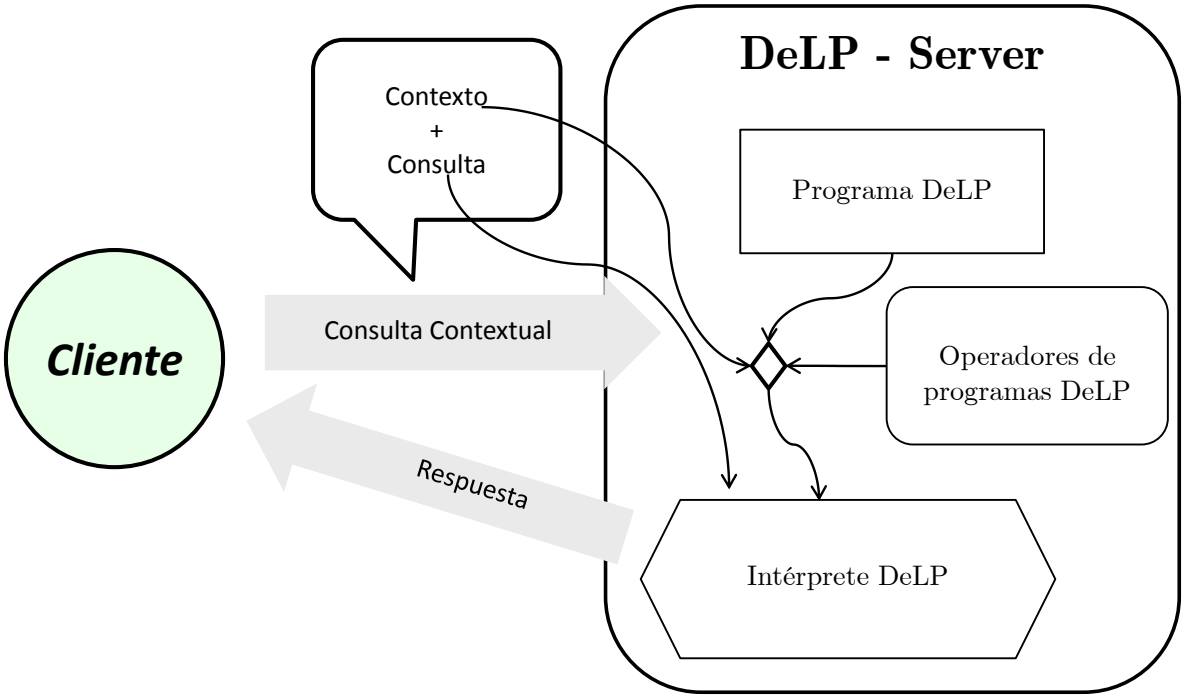

Figura 4.6: DeLP-server recibiendo una consulta contextual.

A partir de lo expuesto hasta el momento, un DeLP-server es una tripla $\langle \mathbb{I}, \mathbf{O}, \mathcal{P} \rangle$ donde $\mathbb{I}$ es un intérprete DeLP, $\mathbf{O}$ es un conjunto de operadores de programas DeLP y $\mathcal{P}$ es un programa DeLP. Por lo tanto, un DeLP-server está constituido por un intérprete DeLP, la posibilidad de representar el conocimiento en la forma de un programa DeLP, y el conjunto de operadores que van a manejar la integración de la información contextual con el programa almacenado. Como se ha venido explicando a lo largo de este capítulo un intérprete DeLP toma un programa DeLP y una consulta DeLP, y retorna una de las siguientes posibles respuestas DeLP: SI, NO, INDECISO y DESCONOCIDO. Entonces un intérprete DeLP se define como una función de la siguiente forma. Sea D_P el dominio de

todos los posibles programas DeLP, y $\mathsf{D_c}$ el dominio de las consultas DeLP. Un intérprete DeLP es una función

$$\mathbb{I} : \mathsf{D_p} \times \mathsf{D_c} \rightarrow \{\text{SI}, \text{NO}, \text{INDECISO}, \text{DESCONOCIDO}\}$$

Note que la formalización en la definición anterior del concepto de intérprete DeLP será utilizada en los capítulos que se presentarán más adelante. Concretamente, los servicios de razonamientos propuestos en esta tesis contarán con un intérprete DeLP como uno de los elementos que los constituye.

Habiendo definido la noción de intérprete DeLP, otros de los componentes en un DeLP-server son los operadores de programa DeLP. Un operador de programa $o \in \mathbf{O}$ es un operador binario que toma dos programas DeLP $\mathcal{P}_1$ y $\mathcal{P}_2$, y retorna un nuevo programa DeLP; este programa DeLP es el resultado de integrar $\mathcal{P}_1$ y $\mathcal{P}_2$ según la definición del operador o. Un operador de programa DeLP es una función

$$o : \mathsf{D_p} \times \mathsf{D_p} \rightarrow \mathsf{D_p}$$

Una consulta contextual enviadas a un DeLP-server involucra dos elementos: una consulta DeLP Q para ser respondida, e información contextual $\mathcal{C}o$ que será utilizada junto con el programa DeLP almacenado en el servidor para responder Q. Esta información contextual se compone de un programa DeLP que representa conocimiento privado y de operadores apropiados que declaran cómo integrar el conocimiento privado con el programa público almacenado en el servidor. Formalmente, una consulta contextual para un DeLP-server $\langle \mathbb{I}, \mathbf{O}, \mathcal{P} \rangle$ es un par $(\mathcal{C}o, Q)$ donde Q es una consulta DeLP, y el contexto $\mathcal{C}o$ es una secuencia $[(\mathcal{P}_1, o_1), \ldots, (\mathcal{P}_n, o_n)]$, tal que $\mathcal{P}_i$ es un programa DeLP y $o_i \in \mathbf{O}$, $1 \leq i \leq n$.

Los operadores de programas DeLP están definidos de forma modular de manera que cada servidor en particular pueda tener un conjunto de operadores diseñados específicamente para el dominio de aplicación en uso. En [GRTS07], [TGS09] y [GS14] se presentaron ejemplos concretos de operadores de programas. A continuación se mostrarán tres operadores denotados $\oplus, \otimes$ y $\ominus$ los cuales fueron introducidos en [GS14]. Sea L un literal y $\overline{L}$ el complemento de L con respecto a la negación fuerte, si X es un conjunto de literales representando hechos, el complemento de X se definirá como $\overline{X} = \{\overline{L} \mid L \in X\}$. Dado que un operador de programa DeLP puede verse como una función que recibe dos programas DeLP, los operadores $\oplus, \otimes$, y $\ominus$ se definen de la siguiente manera:

$$((\Pi, \Delta)) \oplus ((\Pi_a, \Delta_a)) = ((\Pi \backslash \overline{\Pi_a}) \cup \Pi_a, \Delta \cup \Delta_a)$$

$$((\Pi, \Delta)) \otimes ((\Pi_a, \Delta_a)) = (\Pi \cup (\Pi_a \backslash \overline{\Pi}), \Delta \cup \Delta_a)$$

$$((\Pi, \Delta)) \ominus ((\Pi_a, \Delta_a)) = (\Pi \backslash \Pi_a, \Delta \setminus \Delta_a)$$

El primer operador, denotado $\oplus$, integra el conocimiento privado de una consulta representado con el programa DeLP $((\Pi_a, \Delta_a))$ con el programa DeLP (Π, Δ) alojado en el servidor, dando en caso de conflicto prioridad a la información contenida en Π_a. Por ejemplo, si el conocimiento público alojado en el servidor es $((\Pi, \Delta)) = (\{a, b\}, \Delta)$ y la información privada recibida en una consulta contextual es $((\Pi_a, \Delta_a)) = (\{\sim a, c\}, \Delta_a)$, entonces

$$(\{a, b\}, \Delta) \oplus (\{\sim a, c\}, \Delta_a) = (\{b, \sim a, c\}, \Delta \cup \Delta_a)$$

Note que al integrar la información aparece una contradicción dado que Π contiene los literales $a, \sim a$; sin embargo, el operador $\oplus$ prioriza la información recibida, por lo tanto el literal a será ignorado por el servidor al momento de responder la consulta.

Por lo contrario al operador anterior, el operador $\otimes$ prioriza los hechos que son parte del programa DeLP $((\Pi, \Delta))$ alojado en el servidor. Entonces,

$$(\{a, b\}, \Delta) \otimes (\{\sim a, c\}, \Delta_a) = (\{a, b, c\}, \Delta \cup \Delta_a)$$

En este caso, como la información que el servidor tiene almacenada tiene prioridad sobre la recibida en la consulta, el literal a será considerado al momento de responder la consulta.

Finalmente, el operador denotado $\ominus$ fue desarrollado con el propósito de no considerar como parte del programa alojado en el servidor aquella información privada que se indica en la consulta. Por ejemplo, si el conocimiento público en el servidor es $((\Pi, \Delta)) = (\{a, b, p\}, \Delta)$ y la información recibida es $((\Pi_a, \Delta_a)) = (\{a, b, \sim p\}, \Delta_a)$, entonces tenemos que,

$$(\{a, b, p\}, \Delta) \ominus (\{a, b, \sim p\}, \Delta_a) = (\{p\}, \Delta \setminus \Delta_a)$$

Para responder una consulta contextual $(\mathcal{C}o, Q)$, un DeLP-server $\langle \mathbb{I}, \mathbf{O}, \mathcal{P} \rangle$ debe considerar primero cómo integrar el contexto $\mathcal{C}o$ al programa DeLP $\mathcal{P}$. Como el contexto puede ser una secuencia de pares $[(\mathcal{P}_1, o_1), \ldots, (\mathcal{P}_n, o_n)]$, entonces la aplicación de $[(\mathcal{P}_1, o_1), \ldots, (\mathcal{P}_n, o_n)]$ a $\mathcal{P}$ se define recursivamente como:

- $[\,](\mathcal{P}) = \mathcal{P}$,
- $[(\mathcal{P}_1, o_1), \ldots, (\mathcal{P}_n, o_n)](\mathcal{P}) = [(\mathcal{P}_2, o_2), \ldots, (\mathcal{P}_n, o_n)](\mathcal{P}\ o_1\ \mathcal{P}_1)$

La aplicación del contexto $\mathcal{C}o$ al programa DeLP $\mathcal{P}$ se denotará $(\mathcal{C}o \diamond \mathcal{P})$. Entonces la respuesta del DeLP-server $\langle \mathbb{I}, \mathbf{O}, \mathcal{P} \rangle$ para la consulta contextual $(\mathcal{C}o, Q)$ es el resultado de $\mathbb{I}((\mathcal{C}o \diamond \mathcal{P}), Q)$. Es importante mencionar que dado que un contexto es una secuencia, entonces será procesada por el DeLP-server siguiendo esa secuencia.

Ejemplo 4.12. *Considere el dominio de aplicación presentado en el ejemplo de aplicación de la Sección 4.5. A partir de la información presentada en el programa DeLP $\mathcal{P}_h$ de la Figura 4.2 es posible definir un DeLP-server $\langle \mathbb{I}, \{\oplus, \otimes, \ominus\}, \mathcal{P}_{h'} \rangle$ que ofrece un servicio de recomendación de hoteles el cual se puede acceder desde el computador a bordo instalado en un vehículo.*

El programa DeLP $\mathcal{P}_{h'} = (\Pi_{h'}, \Delta_{h'})$ que se muestra a continuación representa el conocimiento público del servidor y es una versión reducida del programa $\mathcal{P}_h$ ilustrado en la Figura 4.2 ya que no considera las reglas estrictas e incluye unicamente dos hechos.

$$\Pi_{h'} = \left\{ \begin{array}{l} mParadas \\ deNoche \end{array} \right\}\ \Delta_{h'} = \left\{ \begin{array}{l} sugerir(X) \prec bHotel(X), hCercano(X) \\ sugerir(X) \prec sParar, hCercano(X) \\ \sim sugerir(X) \prec zPeligrosa(X) \\ zPeligrosa(X) \prec zRobos(X) \\ \sim zPeligrosa(X) \prec zPolicias(X) \\ bHotel(X) \prec estrellas(X, S), S \geq 3 \\ \sim bHotel(X) \prec estrellas(X, S), S < 3 \\ \sim sParar \prec mParadas \\ sParar \prec mHManejando \\ sParar \prec deNoche \end{array} \right\}$$

En el ejemplo de la Sección 4.5 se presentaron los criterios $\succsim_{\mathsf{seg}}$ y $\succsim_{\mathsf{conf}}$ ambos basado en el mecanismo de comparación de prioridad entre reglas [GS04]. Como los DeLP-servers tienen el criterio de preferencia entre argumentos embebido dentro del intérprete DeLP, el siguiente ejemplo asume que el servidor dispone del criterio $\succsim_{\mathsf{seg}}$ que tiene asociado el orden de prioridad $>_{\mathsf{seg}}$ ilustrado en la Figura 4.3-(a), el cual se muestra a continuación.

$(\sim sugerir(X) \prec zPeligrosa(X)) >_{\mathsf{seg}} (sugerir(X) \prec bHotel(X), hCercano(X))$

$(\sim sugerir(X) \prec zPeligrosa(X)) >_{\mathsf{seg}} (sugerir(X) \prec sParar, hCercano(X))$

$(\sim sParar \prec mParadas) >_{\mathsf{seg}} (sParar \prec mHManejando)$

$(sParar \prec deNoche) >_{\mathsf{seg}} (\sim sParar \prec mParadas)$

$(zPeligrosa(X) \prec zRobos(X)) >_{\mathsf{seg}} (\sim zPeligrosa(X) \prec zPolicias(X))$

En lo que sigue se incluyen algunas consultas contextuales que puede recibir el servidor y se exponen las respuestas que serán retornadas. La estructura de cada uno de los argumentos tenidos en cuenta en el proceso de análisis de dialéctica para cada una de las consultas recibidas se muestra en la Figura 4.7. Entonces un conductor podría consultar al DeLP-server $\langle \mathbb{I}, \{\oplus, \otimes, \ominus\}, \mathcal{P}_{h'} \rangle$ *enviando una de las siguientes tres consultas:*

1. $([(\{zRobos(h1)\}, \oplus)], sugerir(h1))$

   respuesta: NO

2. $([(\{zRobos(h1), zPolicias(h1)\}, \oplus)], sugerir(h1))$

   respuesta: NO

3. $([(\{mHManejando, hCercano(h1)\}, \oplus), (\{deNoche\}, \ominus)], sugerir(h1))$

   respuesta: SI

Si el conductor ejecuta la primer consulta, luego de integrar a $\mathcal{P}_{h'}$ *la información contextual, el proceso de argumentación de DeLP obtiene el argumento garantizado* $\langle \mathcal{A}_2, \sim sugerir(h1) \rangle$*. Como se obtuvo un argumento no derrotado para el complemento del literal* $sugerir(h1)$ *la respuesta del servidor será,* NO*. Por otro lado, si se envía la segunda consulta contextual, entonces se obtendrá el argumento* $\langle \mathcal{A}_2, \sim sugerir(h1) \rangle$ *y el contra-argumento que lo ataca* $\langle \mathcal{A}_4, \sim zPeligrosa(h1) \rangle$*. En este caso,* $\langle \mathcal{A}_3, zPeligrosa(h1) \rangle$ *es el subargumento de desacuerdo. Dado que* $\langle \mathcal{A}_2, \sim sugerir(h1) \rangle$ *derrota a* $\langle \mathcal{A}_4, \sim zPeligrosa(h1) \rangle$ *ya que* $\langle \mathcal{A}_3, zPeligrosa(h1) \rangle$ *es preferido a* $\langle \mathcal{A}_4, \sim zPeligrosa(h1) \rangle$ *por tener una regla de mayor prioridad bajo el criterio* $\succ_{\mathsf{prioR}} ((zPeligrosa(X) \prec zRobos(X)) >_{\mathsf{seg}} (\sim zPeligrosa(X) \prec zPolicias(X)))$*, se tiene que la respuesta del servidor será,* NO*. Por último, el DeLP-server podría recibir la tercer consulta contextual. Si esto sucede se agregará al conjunto* Π *los hechos* $mHManejando$ *y* $hCercano(h1)$*, e ignorará en la recomendación el hecho* $deNoche$ *que forma parte de* Π*. A*

partir del programa modificado por la información contextual recibida, el proceso de argumentación obtiene el argumento $\langle \mathcal{A}_1, sugerir(h1)\rangle$ *y el contra-argumento* $\langle \mathcal{A}_6, \sim sParar\rangle$. *Note que el argumento* $\langle \mathcal{A}_1, sugerir(h1)\rangle$ *es un derrotador de* $\langle \mathcal{A}_6, \sim sParar\rangle$ *debido a que el subargumento de desacuerdo* $\langle \mathcal{A}_5, sParar\rangle$ *es preferido a* $\langle \mathcal{A}_6, \sim sParar\rangle$ *por tener el primero una regla rebatible de mayor prioridad bajo el criterio* $\succeq_{\mathsf{seg}}$. *Como no existe otro argumento que contradiga a* $\langle \mathcal{A}_1, sugerir(h1)\rangle$, *entonces la respuesta del DeLP-server para la consulta* $sugerir(h1)$ *será,* SI.

$\langle \mathcal{A}_1, sugerir(h1)\rangle$, donde $\mathcal{A}_1 = \left\{ \begin{array}{l} sugerir(h1) \prec sParar, hCercano(h1) \\ sParar \prec mHManejando \end{array} \right\}$

$\langle \mathcal{A}_2, \sim sugerir(h1)\rangle$, donde $\mathcal{A}_2 = \left\{ \begin{array}{l} \sim sugerir(h1) \prec zPeligrosa(h1) \\ zPeligrosa(h1) \prec zRobos(h1) \end{array} \right\}$

$\langle \mathcal{A}_3, zPeligrosa(h1)\rangle$, donde $\mathcal{A}_3 = \{zPeligrosa(h1) \prec zRobos(h1)\}$

$\langle \mathcal{A}_4, \sim zPeligrosa(h1)\rangle$, donde $\mathcal{A}_4 = \{\sim zPeligrosa(X) \prec zPolicias(X)\}$

$\langle \mathcal{A}_5, sParar\rangle$, donde $\mathcal{A}_5 = \{sParar \prec mHManejando\}$

$\langle \mathcal{A}_6, \sim sParar\rangle$, donde $\mathcal{A}_6 = \{\sim sParar \prec mParadas\}$

Figura 4.7: Argumentos considerados como parte del análisis de dialéctica para la consulta "$sugerir(h1)$".

4.6.2. Discusión

Como se mostró a lo largo de este capítulo, en [GRTS07] se introduce por primera vez la noción formal de servicio de razonamiento basado en DeLP mediante la formalización de los DeLP-servers. Estos servidores se caracterizan por proveer un servicio de razonamiento argumentativo sobre una base de conocimiento representada mediante un programa DeLP. Asimismo, el proceso de razonamiento de estos servidores resolverá los conflictos entre información contradictoria utilizando un criterio de preferencia para argumentos. A continuación se discutirán algunas cuestiones sobre cómo es el manejo del criterio de preferencia entre argumentos en los DeLP-servers las cuales serán interesantes abordar para la mejora en las capacidades de razonamiento en este tipo de servidores.

En un DeLP-server el criterio de preferencia se establece en la configuración inicial del servidor. Esto conlleva a una limitación en estos servidores dado que no es posible cambiar de criterio una vez que uno es elegido. Para ilustrar mejor esta observación considere el

DeLP-server $\langle \mathbb{I}, \{\oplus, \otimes, \ominus\}, \mathcal{P}_{h'} \rangle$ del Ejemplo 4.12 de la Sección 4.6.1. Note que $\succsim_{\mathsf{seg}}$ es el criterio de preferencia embebido dentro del intérprete DeLP $\mathbb{I}$, el cual asume el orden entre reglas $>_{\mathsf{seg}}$ para todas las consultas contextuales recibidas por el servidor, sin tener el usuario la posibilidad de cambiar dicho orden y mucho menos el criterio de preferencia.

Una posible solución al problema planteado arriba es disponer de varios DeLP-servers y cada uno con un criterio diferente y contar con un mecanismo que permita seleccionar de manera automática un servidor en base al criterio que el agente cliente desea. Sin embargo, este modelo de múltiples DeLP-servers no permite estudiar propiedades que relacionan un DeLP-server con múltiples criterios de preferencia. En el Capítulo 5 se introducirá un servicio de razonamiento basado en DeLP el cual cuenta con la capacidad de tener implementado varios criterios de preferencia.

Considerando nuevamente el Ejemplo 4.12, asuma ahora que el servidor además del criterio $\succsim_{\mathsf{seg}}$ contiene la implementación del criterio $\succsim_{\mathsf{conf}}$, presentado en la Sección 4.5. Entonces, la selección de uno de estos criterios podría depender de la existencia de determinada información alojada en el servidor. Por ejemplo, un usuario podría elegir para ser utilizado en su consulta el criterio basado en el confort $\succsim_{\mathsf{conf}}$ si sabe que existe un hotel cercano de cinco estrellas, en caso contrario preferiría que se aplique para su consulta el criterio basado en la seguridad $\succsim_{\mathsf{seg}}$. Si embargo, para lograr esto es necesario que los DeLP-servers se puedan adaptar a más de un criterio en base a la información que procesan, situación que no puede suceder a partir de como hoy en día estos servidores están definidos. En el Capítulo 6 se formaliza una herramienta concreta basada en condiciones para la selección de criterios, y se introduce un nuevo tipo de servicio de razonamiento basado en DeLP con la capacidad de seleccionar un criterio a partir del uso de esta herramienta.

Como se viene mencionando el criterio de preferencia es un elemento fijo una vez que se establece la configuración del DeLP-server. Por lo tanto, no es posible que estos servidores puedan actuar con varios criterios simultáneamente, salvo que el mecanismo de comparación utilice una combinación fija de criterios. Por ejemplo, siguiendo con el ejemplo del DeLP-server presentado en el Ejemplo 4.12, este servidor podría tener implementado un criterio que utilice una combinación fija de los criterios $\succsim_{\mathsf{conf}}$ y $\succsim_{\mathsf{seg}}$. Sin embargo con esta alternativa no es posible utilizar otros criterios que no sean los que fueron combinados. En este sentido, en el Capítulo 7 se proponen diferentes mecanismos para combinar criterios, y se presenta un servicio de razonamiento basado en DeLP capaz de tratar con la combinación de múltiples criterios.

4.7. Conclusión

En este capítulo se describió el lenguaje de *Programación Lógica Rebatible (DeLP)*, de forma similar a como se lo presenta en el Capítulo 2 de [Gar00]. En primer lugar, se definió la sintaxis de este lenguaje y luego se mostró el procedimiento de prueba para los programas lógicos rebatibles. Por último, además de mostrar un ejemplo de aplicación en un dominio concreto, se describieron varios criterios de preferencia entre argumentos que podrían ser utilizados por DeLP.

Como se mencionó al comienzo del capítulo, DeLP puede ser utilizado básicamente en cualquier aplicación que requiera representación de conocimiento, por lo que resulta una herramienta atractiva tanto para representar conocimiento como para definir el razonamiento de agentes deliberativos. De hecho, para dar una respuesta, el mecanismo de razonamiento sobre el cual está basado lleva a cabo una serie de etapas. Una muy importante es la comparación de argumentos en conflicto. Una característica ventajosa de DeLP con respecto a dicha evaluación es la posibilidad de reemplazar de forma modular el criterio de preferencia entre argumentos. Como se pudo ver en el ejemplo de la Sección 4.5, si se cambia de criterio, la respuesta del mecanismo de razonamiento para una misma consulta posiblemente varie. Resaltar el impacto que tiene el criterio que evaluá los argumentos sobre el proceso de inferencia llevado a cabo en un sistema argumentativo estructurado como lo es DeLP ha sido una de las principales motivaciones para el desarrollo de este capítulo.

En la Sección 4.4 se muestran algunos criterios de preferencia entre argumentos en los que DeLP se puede basar para computar sus respuestas. Si bien DeLP permite ajustar el criterio a un dominio de aplicación concreto, algunos sistemas que utilizan este lenguaje permiten usar dos criterios [DFDG+13, GMP12], pero por lo general, la combinación propuesta es fija para el formalismo. Existen otros sistemas que permiten trabajar con información dinámica, por ejemplo [TGS09] que puede responder a consultas a partir de información contextual. No obstante, estos sistemas no permiten ajustar a determinadas condiciones o requisitos el criterio que resuelva los conflictos entre argumentos. Contar con herramientas que le permitan a estos formalismos combinar y cambiar de criterio dinámicamente son algunos de los pilares dentro de los cuales se basan gran parte de los aportes de esta tesis. DeLP constituye el formalismo fundamental empleado para desarrollar el modelo de los servicios de razonamiento que se presentarán en los capítulos siguientes de esta tesis.

Finalmente, en la Sección 4.6 se presentan los conceptos de Servicios de Razonamiento basados en Programación Lógica Rebatible o usualmente llamados DeLP-servers y Consultas Contextuales. Estos servicios tienen la capacidad de representar conocimiento mediante programas DeLP y permiten realizar modificaciones temporales a dicho conocimiento para responder consultas de clientes. Para permitir modificaciones sobre su conocimiento, estos servicios proveen operadores específicos, mientras que para responder las consultas poseen un intérprete DeLP que computa las respuestas. La base conceptual del formalismo presentada en la Sección 4.6 será de gran importancia para las contribuciones de esta tesis. Dentro de estas contribuciones se introducen diferentes servicios de razonamiento que toman como base el modelo cliente-servidor de los DeLP-servers.

Capítulo 5

Servicio de Razonamiento Rebatible basado en Preferencias

En este capítulo se presenta un formalismo para el diseño de Servicios de Razonamiento basados en DeLP que cuentan con la implementación de múltiples criterios de preferencia, el cual representa el primer aporte de esta tesis. Estos servicios permitirán mejorar las capacidades de razonamiento de aquellos sistemas basados en DeLP. Para esto, dichos servicios proveerán un razonamiento lógico rebatible que cuenta con la capacidad de modificar las preferencias sobre la información procesada cambiando, a partir de cada consulta recibida, el criterio de preferencia utilizado.

En particular, se definen las nociones de *Servicio de Razonamiento Rebatible basado en Preferencias* (*SRPref*) y *Consulta basada en Preferencias* (*CPref*). El mecanismo que se utilizará para consultar un *SRPref* será el de las *CPrefs*. Este tipo de consulta no solo incluirá la consulta DeLP propiamente dicha, sino que agregará el programa DeLP a ser consultado, y una referencia al criterio de preferencia que será utilizado para decidir entre argumentos en conflicto. Por último, en este capítulo se mostrarán ejemplificados los conceptos desarrollados en dos dominios de aplicación distintos.

5.1. Introducción

En el formalismo de DeLP el criterio de preferencia entre argumentos es modular. Asimismo, independientemente de cómo el criterio realiza la comparación, para DeLP el criterio actúa como una caja negra. De esta manera, el mecanismo de razonamiento de este formalismo no está vinculado a un criterio en particular. Esta situación permite que sea

posible tener implementado varios criterios de preferencia y elegir el que mejor se ajusta a una situación o a la preferencia particular del usuario. No obstante, en la literatura no se ha estudiado como formalizar el uso de diferentes criterios que se adecúen dinámicamente al contexto. Por lo tanto, contar con mecanismos computacionales que permitan llevar a cabo esta tarea resultaría un avance importante para estos tipos de sistemas.

Teniendo en cuenta la consideración anterior, una característica deseable es proveer a DeLP la capacidad de recibir junto con la consulta el criterio que se desea utilizar para responder esa consulta. En este capítulo se propone un formalismo para resolver este aspecto. La formalización propuesta presenta un nuevo tipo de servicio de razonamiento basado en DeLP inspirado en el modelo cliente-servidor de los DeLP-servers [GRTS07].

Como se detalla en el Capítulo 4, se han propuesto diferentes criterios para ser utilizados por el mecanismo de inferencia de DeLP. En varios de estos criterios se puede observar el uso de información adicional a la estructura interna de los argumentos que serán comparados. Por ejemplo, el criterio de prioridad entre reglas $\succeq_{\mathsf{prioR}}$ [GS04] (explicado en la Sección 4.4.4) considera un orden sobre reglas rebatibles, mientras que el criterio basado en literales $\succeq_{\mathsf{prioL}}$ [FEGS08] (explicado en la Sección 4.4.3) utiliza un orden sobre literales. Por lo tanto, si se quiere adaptar el mecanismo de comparación de argumentos de un sistema basado en DeLP a un nuevo criterio de preferencia será necesario tener en cuenta la información que internamente la implementación de dicho criterio podría llegar a necesitar.

Como se muestra más adelante, las consultas recibidas por los servicios de razonamiento propuestos en este capítulo incluirán una referencia a un criterio de preferencia junto a información adicional requerida por su implementación. Por ejemplo, supongamos un servicio que tiene implementado el criterio de prioridad entre reglas $\succeq_{\mathsf{prioR}}$ y el criterio basado en literales $\succeq_{\mathsf{prioL}}$. Si el criterio considerado es el primero, entonces en la consulta se deberá especificar una referencia al mismo, junto a un orden de prioridad sobre reglas rebatibles. Asimismo, si el criterio es el segundo, la consulta deberá tener una referencia a este criterio, y un orden de prioridad entre literales que el criterio asumirá para esa consulta. Por otro lado, supongamos ahora que el servicio tiene implementado el criterio especificidad generalizada [GS04] (explicado en la Sección 4.4.1). Para utilizar este criterio, y a diferencia de los casos anteriores, en la consulta se debería especificar únicamente la referencia al mismo. Especificidad generalizada es un criterio sintáctico basado en la estructura de los argumentos que no requiere información adicional que no se pueda obtener a partir del programa consultado (ver [Gar97, Gar00] para más detalles sobre este

criterio).

De esta manera, las consultas que se presentan en este capítulo proveen un mecanismo computacional concreto para que un servicio de razonamiento pueda cambiar de criterio. En la siguiente sección se describe la notación que se utilizará para especificar la información adicional que la implementación de un criterio de preferencia podría llegar a necesitar al momento de comparar argumentos.

5.2. Cómputo de Preferencias

En varias ocasiones se ha mencionado que el cómputo de preferencias entre argumentos en DeLP se trata de forma modular. A partir de esto, en la Sección 4.4 se define el concepto de criterio de preferencia de manera general y abstracto como una relación $\succsim_{\mathsf{lc}}$ donde lc es un identificador de criterio. Dado el manejo flexible de criterios que tiene el mecanismo de inferencia sobre el cual se basa DeLP, los *SRPref* propuestos en esta tesis podrán contar con la implementación computacional de múltiples criterios, y utilizar el criterio que mejor se ajuste al dominio representado. En lo que sigue se denotará $\mathsf{impC}(\mathsf{lc})$ a la implementación computacional del criterio de preferencia $\succsim_{\mathsf{lc}}$ que un *SRPref* podría llegar a disponer.

Como se ha detallado en capítulos anteriores existen en la literatura muchos criterios para comparar argumentos. Sin embargo, dentro de los criterios más importantes que se han estudiado para el sistema DeLP, y que se consideran para las formalizaciones de esta tesis, se pueden distinguir;

1. los que se basan en la estructura sintáctica de los argumentos, como es el criterio de especificidad generalizada [GS04],

2. los que utilizan información adicional externa al programa sobre el cual se construyen los argumentos, como el criterio basado en prioridad entre literales [FEGS08], y

3. aquellos que combinan los enfoques anteriores, como el criterio desarrollado en [DDG$^+$12].

En lo que sigue se introduce una notación para que la información adicional externa al programa consultado que algunos criterios requieren pueda ser suministrada a los *SRPrefs* a partir de las consultas que se presentan más adelante. De aquí en adelante se asume que existe para cada implementación $\mathsf{impC}(\mathsf{lc})$ un conjunto de *átomos distinguidos* $\mathsf{atomsC}(\mathsf{lc})$

dedicados a representar la información adicional externa que dicha implementación utilizará para el cómputo de preferencias. Esta estructura representa meta información utilizada para resolver la comparación, *i.e.,* los átomos no corresponden al lenguaje de DeLP, sino que permiten dar meta-información de naturaleza variada. Obsérvese que si el criterio de preferencia para comparar argumentos se basa únicamente en la estructura de los argumentos, $\mathsf{atomsC}(\mathsf{lc})$ será un conjunto vacío.

Ejemplo 5.1. *Considere* $\mathsf{impC}(\mathsf{prioR})$ *como una implementación computacional del criterio de prioridad entre reglas* $\succeq_{\mathsf{prioR}}$ *presentado en la Definición 4.27. Supongamos que el conjunto de átomos distinguidos* $\mathsf{atomsC}(\mathsf{prioR})$ *para* $\mathsf{impC}(\mathsf{prioR})$ *se compone de átomos de la forma* $es\_mejorR(R_1, R_2)$ *que describen un orden de prioridad de la regla rebatible* R_1 *sobre la regla* R_2.

A partir de las prioridades entre reglas $>_{\mathsf{seg}}$ *y* $>_{\mathsf{conf}}$ *ilustradas en las Figuras 4.3-(a) y 4.3-(b), respectivamente, es posible especificar los siguientes dos conjuntos,* $\mathsf{D}_{\mathsf{seg}}, \mathsf{D}_{\mathsf{conf}} \subseteq \mathsf{atomsC}(\mathsf{prioR})$.

$$\mathsf{D}_{\mathsf{seg}} = \left\{ \begin{array}{l} (es\_mejorR((\sim sugerir(X) \prec zPeligrosa(X)), (sugerir(X) \prec bHotel(X), hCercano(X)))), \\ (es\_mejorR((\sim sugerir(X) \prec zPeligrosa(X)), (sugerir(X) \prec sParar, hCercano(X))), \\ (es\_mejorR((\sim sParar \prec mParadas), (sParar \prec mHManejando))), \\ (es\_mejorR((sParar \prec deNoche), (\sim sParar \prec mParadas))), \\ (es\_mejorR((zPeligrosa(X) \prec zRobos(X)), (\sim zPeligrosa(X) \prec zPolicias(X))) \end{array} \right\}$$

$$\mathsf{D}_{\mathsf{conf}} = \left\{ \begin{array}{l} (es\_mejorR((sugerir(X) \prec bHotel(X), hCercano(X)), (\sim sugerir(X) \prec zPeligrosa(X)))), \\ (es\_mejorR((sugerir(X) \prec sParar, hCercano(X)), (\sim sugerir(X) \prec zPeligrosa(X)))), \\ (es\_mejorR((\sim sParar \prec mParadas), (sParar \prec mHManejando))) \end{array} \right\}$$

El ejemplo anterior ilustra dos casos concretos de conjuntos con información para la implementación de criterio $\mathsf{impC}(\mathsf{prioR})$. En particular, esta implementación utiliza el predicado *es_mejorR* para expresar las prioridades entre reglas requeridas por la especificación de criterio descripta en la Definición 4.27. Más adelante en el capítulo se muestra que la información externa necesaria por las implementaciones de criterios será suministrada por el usuario como parte de la información incluida en su consulta junto con un indicador del criterio que el *SRPref* consultado debe utilizar.

5.3. Servicio de Razonamiento

Como se explicó previamente, en este capítulo se presentan los conceptos de *SRPref* y *CPref*. Un servicio de razonamiento rebatible basado en preferencias Σ estará conformado por:

- el *intérprete* DeLP $\mathbb{I}$ que se encarga de procesar las consultas hechas al servicio Σ; y
- un *módulo de comparación de argumentos* $\mathbb{M}$ que, a través de la *función de cómputo de preferencias*, obtiene las preferencias entre argumentos que el intérprete $\mathbb{I}$ necesita resolver utilizando para ello una de las *implementaciones de criterios de preferencia* almacenadas en Σ.

La definición formal de *SRPref* se presenta luego de definir cada uno de sus componentes. En la Figura 5.1 se muestra un esquema con los componentes que conforman la arquitectura de los *SRPrefs*.

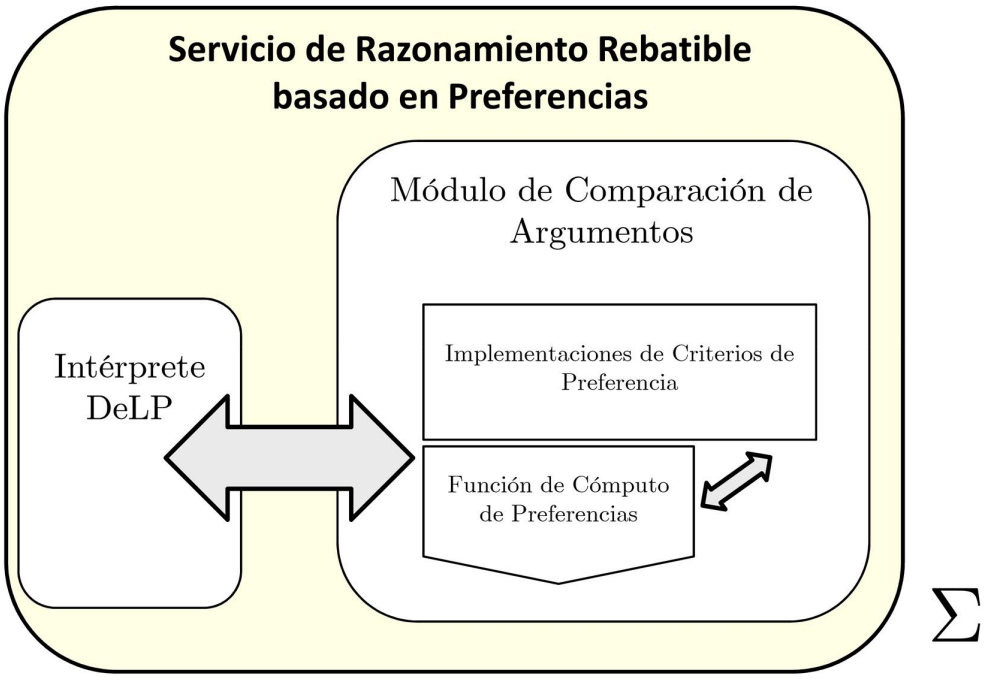

Figura 5.1: Esquema del Servicio de Razonamiento Rebatible basado en Preferencias.

En el caso del intérprete DeLP, éste será el responsable de procesar y responder las consultas DeLP que recibe un *SRPref*. En el Capítulo 4, un intérprete DeLP se definió

en forma general como una función $\mathbb{I}(Q,\mathcal{P})$ que recibe una consulta Q para un programa consultado $\mathcal{P}$ retornando una de las siguientes respuestas: SI, NO, INDECISO, o DESCONOCIDO. Obsérvese que esta definición de intérprete DeLP será la que se utilice a lo largo de esta tesis.

Para responder una consulta DeLP el proceso que lleva a cabo el intérprete tiene varias etapas. Una de estas es decidir qué argumento prevalece ante una situación de conflicto entre dos argumentos, para esto se necesita establecer si uno es preferido sobre el otro. El módulo de comparación de argumentos se encarga de resolver estas preferencias en los *SRPrefs*. Por lo tanto, para cada par de argumentos en conflicto el intérprete delegará esta tarea al módulo de comparación del *SRPref* en cuestión.

A partir de los argumentos recibidos el módulo de comparación debe poder calcular las preferencias que el intérprete necesita. Sin embargo, y como se ha venido observando, en determinados casos para realizar esta tarea no alcanza con considerar únicamente los argumentos que serán comparados. Teniendo en cuenta esto, el módulo de comparación de argumentos que se presenta más adelante recibirá:

- los argumentos a ser comparados, e
- información vinculada a la implementación del criterio de preferencia que se vaya a utilizar.

La comparación de argumentos se realiza a partir de la selección y ejecución de una de las implementaciones de criterio que el *SRPref* consultado dispone. Si bien existen implementaciones que no requieren información adicional externa, en esta tesis se asume que dado un criterio $\succsim_{\mathsf{Ic}}$ para toda implementación $\mathsf{impC}(\mathsf{Ic})$ de este criterio, que un *SRPref* dispone, existe un conjunto de átomos distinguidos que permiten representar información que no puede obtenerse a partir del programa consultado. Para aquellos casos en donde el criterio implementado se basa únicamente en la estructura de los argumentos se asumirá que $\mathsf{atomsC}(\mathsf{Ic}) = \emptyset$. Más adelante, se mostrará que el usuario podrá enviar junto a su consulta no solo un identificador Ic del criterio que se desea utilizar, sino que se podrá agregar información expresada a través de un conjunto de átomos $\mathsf{D} \subseteq \mathsf{atomsC}(\mathsf{Ic})$ que deberá hacer referencia a información necesaria para la ejecución de la implementación del criterio elegido.

Como se define más adelante, el módulo de comparación de argumentos de un *SRPref* dispone de un conjunto C de implementaciones de criterios de preferencia a través del cual será posible resolver las preferencias que el intérprete necesita. La función de cómputo de

preferencias compPref que se formaliza a continuación permite encapsular en el módulo de comparación el mecanismo encargado de ejecutar una de las implementaciones de C a fin de establecer si un argumento es preferido sobre otro.

Definición 5.1 (Función de Cómputo de Preferencias)**.** *Sea* Σ *un SRPref. Sea* D_{p} *el dominio de todos los posibles programas DeLP, y Args el conjunto de todos los argumentos que se pueden obtener de un programa* $\mathcal{P}$ *en el dominio* D_{p}*. Sea* C *un conjunto de implementaciones computacionales de criterios de preferencia almacenadas en el servicio* Σ*. Sea* $\mathsf{atomsC(Ic)}$ *un conjunto de átomos distinguidos para una implementación* $\mathsf{impC(Ic)} \in \mathsf{C}$*. La función de cómputo de preferencias para las implementaciones en* C*,*

$$\mathsf{compPref} : \mathsf{C} \times 2^{\mathsf{atomsC(Ic)}} \times Args \times Args \mapsto \{\top, \bot\},$$

es tal que dada una implementación de criterio $\mathsf{impC(Ic)} \in \mathsf{C}$*, un conjunto de átomos* $\mathsf{D} \subseteq \mathsf{atomsC(Ic)}$*, y dos argumentos* $\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle \in Args$ *se define como:*

$$\mathsf{compPref}(\mathsf{impC(Ic)}, \mathsf{D}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) = \left\{ \begin{array}{l} \top \textit{ si } \langle \mathcal{A}, L\rangle \succsim_{\mathsf{Ic}} \langle \mathcal{B}, M\rangle, \\ \bot \textit{ en caso contrario.} \end{array} \right\}$$

De esta manera, la Definición 5.1 muestra que para determinar si un argumento $\langle \mathcal{A}, L\rangle$ es preferido sobre otro $\langle \mathcal{B}, M\rangle$ según un criterio de preferencia $\succsim_{\mathsf{Ic}}$, la función compPref ejecutará la implementación computacional impC(Ic) de ese criterio que el módulo de comparación del *SRPref* consultado tiene almacenado. En particular, el proceso de ejecución de la implementación impC(Ic) consistirá en ejecutar el código asociado a impC(Ic) que en este caso implementa al criterio identificado con Ic, tal que junto a un conjunto de átomos $\mathsf{D} \subseteq \mathsf{atomsC(Ic)}$ determina si un argumento es preferido sobre otro. Informalmente, el módulo de comparación de argumentos será el encargado en un *SRPref* de mantener almacenado el código asociado a cada implementación impC(Ic) que el servicio puede utilizar.

Observación 5.1. *Cabe destacar que si el conjunto de implementaciones de criterios de preferencia almacenadas en un SRPref fuese vacío nunca se podrá decidir la preferencia entre dos argumentos; por este motivo, salvo que se indique lo contrario, cuando se habla del conjunto* C *se asume que éste es un conjunto no vacío.*

La función de cómputo de preferencias compPref resuelve todas las comparaciones de argumentos que un intérprete DeLP necesita durante el análisis dialéctico. Para ilustrar

mejor esta idea, considere el conjunto de átomos $\mathsf{D}_{\mathsf{seg}}$ para la implementación $\mathsf{impC}(\mathsf{prioR})$ introducido en el Ejemplo 5.1, y los argumentos ilustrados en la Figura 4.5. Nótese que utilizando la función $\mathsf{compPref}$ es posible obtener las preferencias que resultan del análisis dialéctico ilustrado en la Figura 4.4-(a).

- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle, \langle \mathcal{A}_{10}, sugerir(h1)\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_{10}, sugerir(h1)\rangle, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle, \langle \mathcal{A}_{12}, sugerir(h1)\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_{12}, sugerir(h1)\rangle, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_5, \sim sParar\rangle, \langle \mathcal{A}_4, sParar\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_4, sParar\rangle, \langle \mathcal{A}_5, \sim sParar\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle, \langle \mathcal{A}_{11}, sugerir(h1)\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{seg}}, \langle \mathcal{A}_{11}, sugerir(h1)\rangle, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_6, sParar\rangle, \langle \mathcal{A}_5, \sim sParar\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_5, \sim sParar\rangle, \langle \mathcal{A}_6, sParar\rangle) = \bot$.

Del mismo modo, si en lugar de considerar $\mathsf{D}_{\mathsf{seg}}$ se utiliza el conjunto $\mathsf{D}_{\mathsf{conf}}$ del Ejemplo 5.1, es posible conseguir las preferencias que se obtienen a partir del análisis que se muestra en la Figura 4.4-(b).

- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_5, \sim sParar\rangle, \langle \mathcal{A}_4, sParar\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_4, sParar\rangle, \langle \mathcal{A}_5, \sim sParar\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_{12}, sugerir(h1)\rangle, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle, \langle \mathcal{A}_{12}, sugerir(h1)\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_{11}, sugerir(h1)\rangle, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle) = \top$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle, \langle \mathcal{A}_{11}, sugerir(h1)\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D}_{\mathsf{conf}}, \langle \mathcal{A}_{10}, sugerir(h1)\rangle, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle) = \top$.

- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D_{conf}}, \langle \mathcal{A}_9, \sim sugerir(h1)\rangle, \langle \mathcal{A}_{10}, sugerir(h1)\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D_{conf}}, \langle \mathcal{A}_6, sParar\rangle, \langle \mathcal{A}_5, \sim sParar\rangle) = \bot$.
- $\mathsf{compPref}(\mathsf{impC}(\mathsf{prioR}), \mathsf{D_{conf}}, \langle \mathcal{A}_5, \sim sParar\rangle, \langle \mathcal{A}_6, sParar\rangle) = \bot$.

Continuando con la formalización, un módulo de comparación de argumentos se describe como un par que contiene el conjunto de implementaciones de criterios de preferencia disponibles en un *SRPref* específico y una función que es responsable de ejecutar la implementación de un criterio en particular. A continuación se introduce esta noción.

Definición 5.2 (Módulo de Comparación de Argumentos (*MCA*))**.** *Sea* Σ *un SRPref. Sea* C *un conjunto de implementaciones computacionales de criterios de preferencia almacenados en el servicio* Σ*, y* $\mathsf{compPref}$ *la función de cómputo de preferencias para las implementaciones en* C*. El par* $\langle \mathsf{C}, \mathsf{compPref}\rangle$ *efectiviza el módulo de comparación de argumentos* $\mathbb{M}$ *en* Σ*.*

El conjunto C de la definición anterior establece el conjunto de implementaciones de criterios que un *SRPref* puede utilizar. Para asegurar que existe un vínculo entre la información del dominio, modelada mediante programas DeLP, y los criterios implementados en un servicio de razonamiento, se asume que estos criterios estarán relacionados a un dominio de aplicación concreto. De esta manera, los *SRPrefs* serán específicamente definidos para un dominio en particular.

Habiendo introducido todos los componentes, la siguiente definición formaliza los Servicios de Razonamiento Rebatible basados en Preferencias de la siguiente manera:

Definición 5.3 (Servicio de Razonamiento Rebatible basado en Preferencias (*SRPref*))**.** *Un Servicio de Razonamiento Rebatible basado en Preferencias* Σ *es un par* $\langle \mathbb{I}, \mathbb{M}\rangle$*, donde* $\mathbb{I}$ *es un intérprete DeLP, y* $\mathbb{M}$ *un módulo de comparación de argumentos.*

Ejemplo 5.2. *Teniendo en cuenta las implementaciones computacionales* $\mathsf{impC}(\mathsf{prioR})$ *e* $\mathsf{impC}(\mathsf{esp})$ *para los criterios* $\succeq_{\mathsf{prioR}}$ *y* $\succeq_{\mathsf{esp}}$ *presentados en la Definición 4.27 y 4.23, respectivamente, es posible definir un servicio de razonamiento basado en preferencias* $\Sigma_h = \langle \mathbb{I}_h, \mathbb{M}_h\rangle$*, el cual se utilizará más adelante en este capítulo, donde* $\mathbb{I}_h$ *representa un intérprete DeLP y* $\mathbb{M}_h = \langle \{\mathsf{impC}(\mathsf{prioR}), \mathsf{impC}(\mathsf{esp})\}, \mathsf{compPref}_h\rangle$ *el MCA. Note que para la implementación* $\mathsf{impC}(\mathsf{esp})$ *no se utilizan átomos distinguidos, es decir el conjunto* $\mathsf{atomsC}(\mathsf{esp})$ *es vacío, esto se debe a que* $\mathsf{impC}(\mathsf{esp})$ *no requiere información adicional*

externa al programa consultado. Por otra parte, para impC(prioR) *consideraremos el átomo* $es\_mejorR(R_1, R_2)$ *introducido en el Ejemplo 5.1.*

Como se muestra en los siguientes ejemplos, este servicio concreto, a través del intérprete $\mathbb{I}_h$ *puede resolver las consultas recibidas, mientras que el módulo de comparación de argumentos* $\mathbb{M}_h$ *es quien se encarga, mediante la función de cómputo de preferencias* $\mathsf{compPref}_h$*, de resolver todas las preferencias entre argumentos que el intérprete necesita computar. Concretamente, esta función selecciona y ejecuta alguna de las implementaciones con las que cuenta el módulo de comparación.*

En la siguiente sección se introducen las consultas basadas en preferencias, las cuales permiten incluir junto con las consultas, el programa a ser consultado y una especificación del criterio sobre el cual se basará el *SRPref* para responder dicha consulta.

5.4. Consulta basada en Preferencias

En esta sección se define el concepto de *Consulta basada en Preferencias* (*CPref*). Este tipo de consulta tiene la particularidad de incluir la información necesaria para que una consulta en cuestión pueda ser resuelta por un *SRPref* particular. Esta información incluye el conocimiento sobre el cual se va a razonar, y una referencia al mecanismo de comparación encargado de decidir qué información prevalece sobre otra cuando estén en conflicto. Este mecanismo es definido por uno de los criterios de preferencia disponibles en el *SRPref* consultado.

A continuación se describe una especificación declarativa que indica el criterio de preferencia que un *SRPref* debe considerar para responder una consulta. Esta especificación está constituida por el identificador de uno de los criterios provistos por el propio *SRPref*, e información adicional que la implementación de dicho criterio puede requerir.

Definición 5.4 (Especificación de Criterio (*ECrit*))**.** *Dado un conjunto* C *de implementaciones computacionales de criterios de preferencia. Sea* impC(Ic) *un elemento en* C*, y* atomsC(Ic) *un conjunto de átomos distinguidos para* impC(Ic)*. Una especificación de criterio para* impC(Ic) *es una tupla* (Ic, D) *donde* Ic *es el identificador de criterio y* $\mathsf{D} \subseteq \mathsf{atomsC(Ic)}$*. Se dirá que* (Ic, D) *es adecuada para un módulo de comparación de argumentos* $\mathbb{M} = \langle \mathsf{C}, \mathsf{compPref} \rangle$ *si para el identificador de criterio* Ic *existe una implementación* $\mathsf{impC(Ic)} \in \mathsf{C}$*.*

Para ilustrar la idea de la definición anterior consideremos el siguiente ejemplo.

Ejemplo 5.3. *Dados los conjuntos de átomos* $\mathsf{D}_{\mathsf{seg}}$ *y* $\mathsf{D}_{\mathsf{conf}}$ *para la implementación* $\mathsf{impC}(\mathsf{prioR})$ *presentados en el Ejemplo 5.1. A continuación se especifican los criterios basados en la seguridad y el confort del ejemplo de la Sección 4.5, junto con sus correspondientes órdenes de prioridad, mediante las siguientes ECrits, respectivamente:*

- $\mathcal{EC}_{\mathsf{seg}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{seg}})$
- $\mathcal{EC}_{\mathsf{conf}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{conf}})$

A continuación se presenta la noción de *Consulta basada en Preferencias*(*CPref*) y el comportamiento que tiene un *SRPref* para computar la respuesta a una consulta. Una *CPref* está constituida por un programa, una consulta para dicho programa, y una declaración del criterio de preferencia utilizado por los argumentos obtenidos a partir del programa incluido en la consulta. En lo que sigue se introduce la definición de *CPref*.

Definición 5.5 (Consulta basada en Preferencias (*CPref*))**.** *Sea* D_{p} *el dominio de todos los posibles programas DeLP, y* D_{c} *el dominio de las consultas DeLP. Una consulta basada en preferencias es una tupla* $[\mathcal{EC}, \mathcal{P}, Q]$*, donde* $\mathcal{EC}$ *es una especificación de criterio,* $\mathcal{P}$ *un programa en el dominio* D_{p}*, y* Q *una consulta perteneciente al dominio* D_{c}*. Una consulta basada en preferencias* $[\mathcal{EC}, \mathcal{P}, Q]$ *será adecuada para un servicio de razonamiento rebatible basado en preferencias* $\Sigma = \langle \mathbb{I}, \mathbb{M} \rangle$*, si* $\mathcal{EC}$ *es adecuada para* $\mathbb{M}$*.*

Una vez que un *SRPref* recibe una *CPref* el *MCA* debe seleccionar la implementación de criterio que se debe utilizar. Para esto, dicho módulo contará con la información suministrada por la *ECrit* de la consulta recibida. En la siguiente definición se asumirá que existe una función de selección "σ" que dado un identificador de criterio Ic y un conjunto de implementaciones de criterios C retornará la implementación del criterio indicado por Ic, es decir $\mathsf{impC}(\mathsf{Ic}) \in \mathsf{C}$.

Definición 5.6 (Respuesta del Modulo de Comparación de Argumentos)**.** *Sea* $\Sigma = \langle \mathbb{I}, \mathbb{M} \rangle$ *un SRPref tal que* $\mathbb{M} = \langle \mathsf{C}, \mathsf{compPref} \rangle$*, y* $\mathcal{CP} = [\mathcal{EC}, \mathcal{P}, Q]$ *una consulta basada en preferencias adecuada para* Σ *donde* $\mathcal{EC} = (\mathsf{Ic}, \mathsf{D})$*. Sea Args el conjunto de todos los argumentos que se pueden obtener a partir del programa DeLP* $\mathcal{P}$*, y* $\langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle \in Args$*. La respuesta de* $\mathbb{M}$ *que resulta de comparar* $\langle \mathcal{A}, L \rangle$ *con* $\langle \mathcal{B}, M \rangle$ *utilizando la especificación de criterio* $\mathcal{EC}$*, denotada* $\mathsf{respMod}(\mathbb{M}, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle, \mathcal{EC})$*, corresponde a una de las siguientes:*

- $\langle \mathcal{A}, L \rangle$ *si*
  $\mathsf{compPref}(\sigma(\mathsf{Ic}, \mathsf{C}), \mathsf{D}, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle) = \top$ *y*
  $\mathsf{compPref}(\sigma(\mathsf{Ic}, \mathsf{C}), \mathsf{D}, \langle \mathcal{B}, M \rangle, \langle \mathcal{A}, L \rangle) = \bot$,

- $\langle \mathcal{B}, M\rangle$ *si*
  $\mathsf{compPref}(\sigma(\mathsf{Ic}, \mathsf{C}), \mathsf{D}, \langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) = \top$ *y*
  $\mathsf{compPref}(\sigma(\mathsf{Ic}, \mathsf{C}), \mathsf{D}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) = \bot$,

- $\emptyset$ *en caso contrario.*

Finalmente, el intérprete DeLP será el encargado de calcular las respuestas a las consultas DeLP que recibe un *SRPref*. Para esto, el intérprete interactuará con el *MCA* a fin de resolver las preferencias entre argumentos en conflicto. De acuerdo a lo establecido en la Definición 4.17, si un argumento es un contra-argumento de otro, es necesario analizar si el ataque es lo suficientemente fuerte como para derrotarlo. En consecuencia se deben considerar dos casos, si alguno de ellos es estrictamente preferido sobre el otro o bien son incomparables. A continuación se introduce la definición de respuesta para una *CPref* por un *SRPref*.

Definición 5.7 (Respuesta para una *CPref*)**.** *Sea* $\Sigma = \langle \mathbb{I}, \mathbb{M}\rangle$ *un SRPref y* $\mathcal{CP} = [\mathcal{EC}, \mathcal{P}, Q]$ *una consulta basada en preferencias adecuada para* Σ*. La respuesta para* $\mathcal{CP}$ *en* Σ*, denotada* $\mathsf{resp}(\Sigma, \mathcal{CP})$*, corresponde al resultado del intérprete DeLP* $\mathbb{I}(\mathcal{P}, Q)$*, siendo que se verifica la siguiente condición. Dados dos argumentos* $\langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{B}, M\rangle$ *obtenidos desde el programa DeLP* $\mathcal{P}$*,*

- $\langle \mathcal{A}, L\rangle \succ \langle \mathcal{B}, M\rangle$ *si* $\mathsf{respMod}(\mathbb{M}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle, \mathcal{EC}) = \langle \mathcal{A}, L\rangle$*, y*

- $\langle \mathcal{A}, L\rangle \asymp \langle \mathcal{B}, M\rangle$ *si* $\mathsf{respMod}(\mathbb{M}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle, \mathcal{EC}) = \emptyset$*.*

Con el fin de aclarar los conceptos descriptos previamente, en lo que sigue se muestra un ejemplo que describe cómo la situación planteada en el ejemplo de la Sección 4.5 se puede representar mediante el uso de *SRPrefs* y *CPrefs*.

Ejemplo 5.4. *En el ejemplo de aplicación de la Sección 4.5 se describe un dominio donde un computador instalado a bordo en un vehículo está programado de manera tal que a partir del conocimiento que tiene almacenado puede dar diversas recomendaciones tales como sugerir hoteles. Para representar el conocimiento del sistema se presentó el siguiente programa DeLP* $\mathcal{P}_h = (\Pi_h, \Delta_h)$*, donde.*

$$\Pi_h = \left\{ \begin{array}{l} mParadas \\ mHManejando \\ deNoche \\ hCercano(h1) \\ estrellas(h1, 5) \\ zRobos(h1) \\ cVehicular \leftarrow tLento \end{array} \right\} \quad \Delta_h = \left\{ \begin{array}{l} sugerir(X) \prec bHotel(X), hCercano(X) \\ sugerir(X) \prec sParar, hCercano(X) \\ \sim sugerir(X) \prec zPeligrosa(X) \\ zPeligrosa(X) \prec zRobos(X) \\ \sim zPeligrosa(X) \prec zPolicias(X) \\ bHotel(X) \prec estrellas(X, S), S \geq 3 \\ \sim bHotel(X) \prec estrellas(X, S), S < 3 \\ \sim sParar \prec mParadas \\ sParar \prec mHManejando \\ sParar \prec deNoche \end{array} \right\}$$

Considere ahora el servicio $\Sigma_h = \langle \mathbb{I}_h, \langle \{\mathsf{impC(prioR)}, \mathsf{impC(esp)}\}, \mathsf{compPref}_h \rangle \rangle$ *introducido en el Ejemplo 5.2, el cual responde consultas al programa* $\mathcal{P}_h$ *adaptando el criterio utilizado según la CPref recibida. A continuación se muestran tres CPrefs para* Σ_h *con diferentes ECrits.*

Con el objetivo de especificar los criterios $\succsim_{\mathsf{conf}}$ *y* $\succsim_{\mathsf{seg}}$ *introducidos en la Sección 4.5 junto con sus respectivos ordenes de prioridad, en el Ejemplo 5.3 se presentaron las ECrits para la implementación* $\mathsf{impC(prioR)}$, $\mathcal{EC}_{\mathsf{seg}}$ *y* $\mathcal{EC}_{\mathsf{conf}}$. *Considerando estas especificaciones,* Σ_h *puede recibir una CPref incluyendo la especificación* $\mathcal{EC}_{\mathsf{seg}}$, *y el programa* $\mathcal{P}_h$ *junto con la consulta "*$sugerir(h1)$*"; es decir, la CPref sería* $\mathcal{CP}_{\mathsf{seg}} = [\mathcal{EC}_{\mathsf{seg}}, \mathcal{P}_h, sugerir(h1)]$. *El servicio* Σ_h, *al recibir* $\mathcal{CP}_{\mathsf{seg}}$, *por un lado, le indica al intérprete que utilice el programa* $\mathcal{P}_h$ *para calcular la respuesta a la consulta "*$sugerir(h1)$*", y por el otro, al MCA que considere la especificación de criterio* $\mathcal{EC}_{seg}$ *para la comparación de argumentos. Asimismo,* Σ_h *podría recibir otra CPref que, a diferencia de la anterior (*$\mathcal{CP}_{\mathsf{seg}}$*), incluya la especificación* $\mathcal{EC}_{\mathsf{conf}}$. *En este caso, la consulta basaba en preferencias sería* $\mathcal{CP}_{\mathsf{conf}} = [\mathcal{EC}_{\mathsf{conf}}, \mathcal{P}_h, sugerir(h1)]$.

Finalmente, en este escenario, un usuario podría querer no utilizar prioridades para sus consultas. En este caso, el servicio Σ_h *provee al usuario de una implementación* $\mathsf{impC(esp)}$ *para el criterio especificidad generalizada* $\succsim_{\mathsf{esp}}$ *(ver Definición 4.23). Resumidamente, este criterio prefiere aquellos argumentos con información más precisa o argumentos más directos. Entonces, considerando* $\succsim_{\mathsf{esp}}$ *es posible definir la siguiente ECrit:*

$\mathcal{EC}_{\mathsf{infoEsp}} = (\mathsf{esp}, \mathsf{D}_{\mathsf{infoEsp}})$, *donde* $\mathsf{D}_{\mathsf{infoEsp}} = \{\ \}$.

A partir de esta especificación, Σ_h *podría recibir la consulta basada en preferencias* $\mathcal{CP}_{\mathsf{infoEsp}} = [\mathcal{EC}_{\mathsf{infoEsp}}, \mathcal{P}_h, sugerir(h1)]$. *Para esta consulta en particular, y a diferencia*

de las consultas anteriores, el servicio devuelve una respuesta a la consulta sugerir($h1$) *considerando la implementación* impC(esp) *como el mecanismo utilizado para decidir entre información contradictoria.*

En la Figura 5.2, las flechas muestran cómo son utilizados los datos por el *SRPref*. Entonces, a partir de un programa DeLP $\mathcal{P}$ y una consulta DeLP Q, el intérprete calcula la respuesta para Q, y el *MCA* computa las preferencias entre argumentos en conflicto. Para lograr esto, la función compPref será la encargada de ejecutar la implementación del criterio indicado en $\mathcal{EC}$.

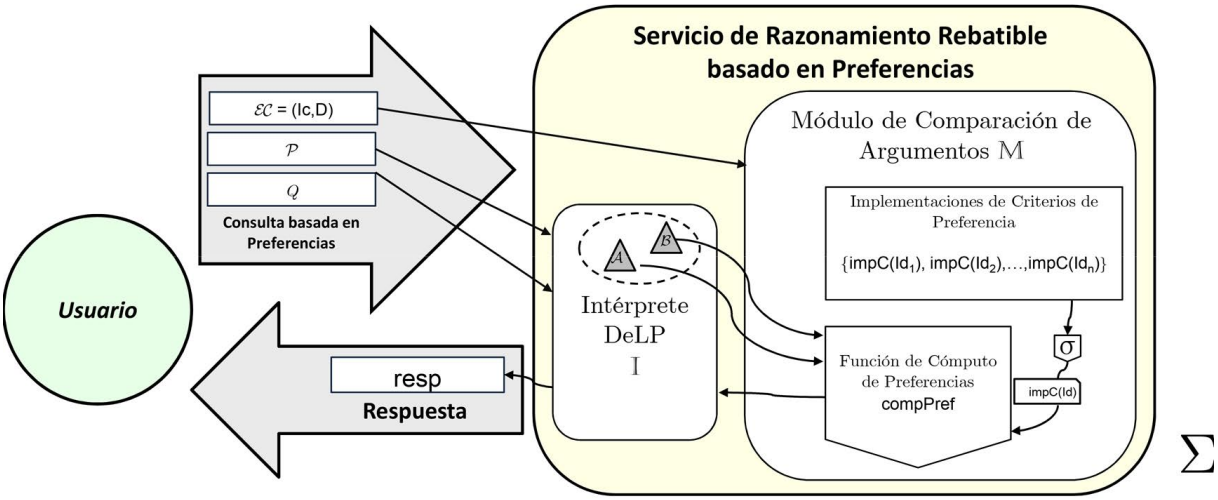

Figura 5.2: Respuesta a una Consulta basada en Preferencias.

En esta sección se definió la noción de *SRPref* y cómo estos servicios responden consultas. Para esto, un *SRPref* recibe una *CPref*, la cual incluye la consulta correspondiente junto con la información necesaria para resolverla, es decir el programa consultado y una especificación del criterio de preferencia entre argumentos que se utilizará. Una de las características más interesantes de los *SRPrefs* es la de permitir que el criterio utilizado varíe para cada consulta en particular. En la siguiente sección se muestran ejemplos de aplicación en dominios concretos donde se pueden aplicar los conceptos de *SRPref* y *CPref* introducidos previamente.

5.5. Ejemplo de aplicación para diagnóstico de enfermedad

A continuación se muestra una descripción detallada sobre cómo las respuestas a diferentes consultas basadas en preferencias pueden ser obtenidas considerando para esto el programa DeLP del Ejemplo 4.3 introducido en el Capítulo 4. En particular, se describe cómo las respuestas que un *SRPref* devuelve pueden cambiar dependiendo del criterio utilizado.

Como aplicación particular de un *SRPref*, en la Figura 5.3 se muestra un *Servicio de Razonamiento para Recomendación de Diagnósticos* (*SRD*) identificado con la tupla $\Sigma_d = \langle \mathbb{I}_d, \langle \{\mathsf{impC(prioR)}, \mathsf{impC(prioRE)}, \mathsf{impC(espE)}\}, \mathsf{compPref}_d \rangle \rangle$ donde el conjunto $\{\mathsf{impC(prioR)}, \mathsf{impC(prioRE)}, \mathsf{impC(espE)}\}$ corresponde a las implementaciones computacionales del criterio basadas en prioridad entre reglas $\succsim_{\mathsf{prioR}}$ (ver Definición 4.27), prioridad entre reglas extendida $\succsim_{\mathsf{prioRE}}$ (ver Definición 4.28), y especificidad generalizada extendida $\succsim_{\mathsf{espE}}$ (ver Definición 4.25), respectivamente.

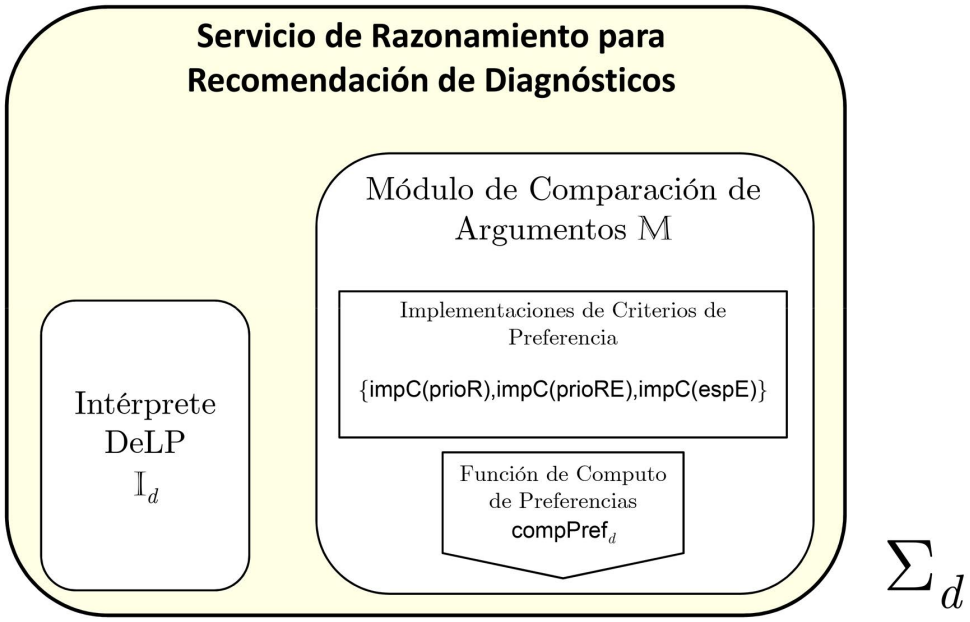

Figura 5.3: Servicio de Razonamiento para Recomendación de Diagnósticos.

Dentro de los criterios que el *SRD* tiene implementados se encuentran los criterios $\succsim_{\mathsf{prioR}}$ y $\succsim_{\mathsf{prioRE}}$, los cuales le permiten al usuario priorizar explícitamente la información representada. Los átomos distinguidos para las implementaciones de estos dos criterios

se representarán mediante el predicado *es_mejorR* introducido en el Ejemplo 5.1, el cual expresa una prioridad entre dos reglas. Por cuestiones de simplicidad, las prioridades entre reglas se indican referenciando a cada regla rebatible a través de una etiqueta r. De esta manera, como se especifica a continuación, una regla rebatible se denotará como $r : L_0 \prec L_1, L_2, \ldots, L_n$. Por otra parte, $\succsim_{\mathsf{espE}}$ es otro de los criterios implementado en el *SRD*. A diferencia de la implementación del criterio $\succsim_{\mathsf{prioR}}$, la implementación de $\succsim_{\mathsf{espE}}$ no utiliza información adicional externa al programa consultado. Por lo tanto, $\mathsf{atomsC}(\mathsf{espE}) = \emptyset$.

Suponga que el *SRD* es consultado para determinar si una determinada enfermedad puede ser diagnosticada. Para esto, un médico podría contar con un programa que representa la información que él considera necesaria para hacer un diagnóstico. Asumiremos que dicha información es la que se expresa en el programa DeLP $\mathcal{P}_{4.3}$ introducido en la Sección 4.2 del Capítulo 4.

$$\mathcal{P}_{4.3} = \left\{ \begin{array}{l} enferm(E) \prec sintC(S, E) \\ \sim enferm(E) \prec sintC(S, E), result(pruebaD(P, E), V), V = negativo \\ \sim enferm(E) \prec trat(T, E) \\ sintC(s1, e1) \prec \\ trat(t1, e1) \\ result(pruebaD(p1, e1), negativo) \end{array} \right\}$$

En lo que sigue se muestran cuatro *CPrefs* diferentes que el *SRD* deberá resolver, buscando en todos los casos un argumento a favor de la consulta DeLP $enferm(e1)$. La Figura 5.4 ilustra los árboles de dialéctica construidos durante el proceso de argumentación llevado a cabo para cada una de estas consultas. A continuación se muestran los argumentos tenidos en cuenta para la construcción de los árboles de la Figura 5.4.

- $\langle \mathcal{A}_1, enferm(e1) \rangle$, donde

$$\mathcal{A}_1 = \left\{ \begin{array}{l} enferm(e1) \prec sintC(s1, e1) \\ sintC(s1, e1) \prec \end{array} \right\}$$

- $\langle \mathcal{A}_2, \sim enferm(e1) \rangle$, donde

$$\mathcal{A}_2 = \left\{ \begin{array}{l} \sim enferm(e1) \prec sintC(s1, e1), result(pruebaD(p1, e1), negativo), V = negativo \\ sintC(s1, e1) \prec \end{array} \right\}$$

- $\langle \mathcal{A}_3, \sim enferm(e1) \rangle$, donde

$\mathcal{A}_3 = \left\{ \sim enferm(e1) \prec trat(t1, e1) \right\}$

- $\langle \mathcal{A}_4, sintC(s1, e1) \rangle$, donde

  $\mathcal{A}_4 = \left\{ sintC(s1, e1) \prec \right\}$

Considere la primer *CPref* enviada al *SRD*:

$\mathcal{CP}_{\mathsf{paciente_1}} = [\mathcal{EC}_{\mathsf{paciente_1}}, \mathcal{P}_{4.3}, enferm(e1)]$, donde

$\mathcal{EC}_{\mathsf{paciente_1}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{paciente_1}})$, y

$\mathsf{D}_{\mathsf{paciente_1}} = \{(es\_mejorR(r_1, r_2))\}$, tal que

$r_1 : enferm(E) \prec sintC(S, E)$

$r_2 : \sim enferm(E) \prec sintC(S, E), result(pruebaD(P, E), V), V = negativo$

Al momento de responder la consulta $\mathcal{CP}_{\mathsf{paciente_1}}$, el *SRD* utilizará el intérprete de programación lógica rebatible. Siguiendo con el proceso de argumentación para la consulta $\mathcal{CP}_{\mathsf{paciente_1}}$, el intérprete se encargará de construir los árboles de dialéctica con el objetivo de determinar si $enferm(e1)$ está garantizado. Como puede observarse en la Figura 5.4-(1), para la consulta $\mathcal{CP}_{\mathsf{paciente_1}}$ el intérprete encuentra que el único argumento $\langle \mathcal{A}_1, enferm(e1) \rangle$ que soporta a $enferm(e1)$ ha sido derrotado por bloqueo por el argumento $\langle \mathcal{A}_3, \sim enferm(e1) \rangle$; de la misma manera, el argumento $\langle \mathcal{A}_3, \sim enferm(e1) \rangle$ que soporta a $\sim enferm(e1)$ está derrotado también por bloqueo por $\langle \mathcal{A}_1, enferm(e1) \rangle$. Por lo tanto, el *SRD*, luego de procesar la consulta entregará la respuesta, INDECISO.

Suponga ahora la consulta $\mathcal{CP}_{\mathsf{paciente_2}}$, tal que el conjunto de átomos distinguidos de la *ECrit* incluida en esta consulta difiere con respecto a los átomos de la especificación de la consulta anterior:

$\mathcal{CP}_{\mathsf{paciente_2}} = [\mathcal{EC}_{\mathsf{paciente_2}}, \mathcal{P}_{4.3}, enferm(e1)]$, donde

$\mathcal{EC}_{\mathsf{paciente_2}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{paciente_2}})$, y

$\mathsf{D}_{\mathsf{paciente_2}} = \left\{ \begin{array}{l} (es\_mejorR(r_1, r_2)), \\ (es\_mejorR(r_1, r_3)) \end{array} \right\}$, tal que

$r_1 : enferm(E) \prec sintC(S, E)$

$r_2 : \sim enferm(E) \prec sintC(S, E), result(pruebaD(P, E), V), V = negativo$

$r_3 : {\sim}enferm(E) \prec trat(T, E)$

La situación mostrada en la Figura 5.4-(2) ilustra que para esta consulta, el argumento $\langle \mathcal{A}_1, enferm(e1)\rangle$ no tiene un argumento que lo derrota, por lo tanto la respuesta obtenida por el *SRD* será, SI. Ahora suponga que el *SRD* debe responder la consulta $\mathcal{CP}_{\mathsf{paciente_3}}$, la cual tiene en su *ECrit* el mismo conjunto de átomos distinguidos que la consulta anterior, pero el criterio referenciado es otro. .

$\mathcal{CP}_{\mathsf{paciente_3}} = [\mathcal{EC}_{\mathsf{paciente_3}}, \mathcal{P}_{4.3}, enferm(e1)]$, donde

$\mathcal{EC}_{\mathsf{paciente_3}} = (\mathsf{prioRE}, \mathsf{D}_{\mathsf{paciente_3}})$, y

$$\mathsf{D}_{\mathsf{paciente_3}} = \left\{ \begin{array}{l} (es\_mejorR(r_1, r_2)), \\ (es\_mejorR(r_1, r_3)) \end{array} \right\}$$

Esta consulta utiliza el criterio basado en prioridad entre reglas extendida $\succsim_{\mathsf{prioRE}}$, el cual prioriza los argumentos basados en hechos sobre los basados en presuposiciones. Este aspecto de utilizar un criterio diferente a las consultas anteriores tiene efecto directo sobre la relación de derrota entre los argumentos. En la Figura 5.4-(3) se puede observar que utilizando este criterio, y a diferencia del árbol de dialéctica generado para la consulta $\mathcal{CP}_{\mathsf{paciente_2}}$, se da que $\langle \mathcal{A}_1, enferm(e1)\rangle$ es derrotado por dos derrotadores por bloqueo, $\langle \mathcal{A}_2, {\sim}enferm(e1)\rangle$ y $\langle \mathcal{A}_3, {\sim}enferm(e1)\rangle$. Entonces, la respuesta del *SRD* será INDECISO. Por último, considere la consulta $\mathcal{CP}_{\mathsf{paciente_4}}$.

$\mathcal{CP}_{\mathsf{paciente_4}} = [\mathcal{EC}_{\mathsf{paciente_4}}, \mathcal{P}_{4.3}, enferm(e1)]$, donde

$\mathcal{EC}_{\mathsf{paciente_4}} = (\mathsf{espE}, \mathsf{D}_{\mathsf{paciente_4}})$, y $\mathsf{D}_{\mathsf{paciente_4}} = \{\ \}$

Obsérvese que a diferencia de las demás consultas, el conjunto de átomos distinguidos en $\mathcal{EC}_{\mathsf{paciente_4}}$ está vacío, esto se debe a que la implementación $\mathsf{impC}(\mathsf{espE})$ del criterio $\succsim_{\mathsf{espE}}$ no utiliza información priorizada expresada explícitamente. Entonces, como muestra la Figura 5.4-(4) para la consulta $\mathcal{CP}_{\mathsf{paciente_4}}$ tenemos que el argumento $\langle \mathcal{A}_2, {\sim}enferm(e1)\rangle$ es preferido a $\langle \mathcal{A}_1, enferm(e1)\rangle$, en consecuencia es un derrotador para este argumento. Por otro lado, $\langle \mathcal{A}_2, {\sim}enferm(e1)\rangle$ es un argumento sin derrotadores. A partir de esto, el *SRD* resuelve la consulta $\mathcal{CP}_{paciente_4}$ devolviendo como respuesta NO.

En esta subsección se mostraron varios aspectos importantes del modelo de servicio de razonamiento presentado en este capítulo. En especial se detalló un servicio de razonamiento para recomendación de diagnósticos (*SRD*) el cual incluye la implementación de

varios criterios de preferencia, mostrados en la Figura 5.3, siendo éstos la base para que el servicio pueda recibir diferentes *ECrits* al momento de ser consultado. Por otro lado, se describieron varias *CPrefs* ($\mathcal{CP}_{\mathsf{paciente}_1}, \mathcal{CP}_{\mathsf{paciente}_2}, \mathcal{CP}_{\mathsf{paciente}_3}$ y $\mathcal{CP}_{\mathsf{paciente}_4}$) las cuales coincidían en que incluían el mismo programa consultado y la misma consulta; sin embargo, para cada *CPref* en particular variaba la especificación del criterio utilizado.

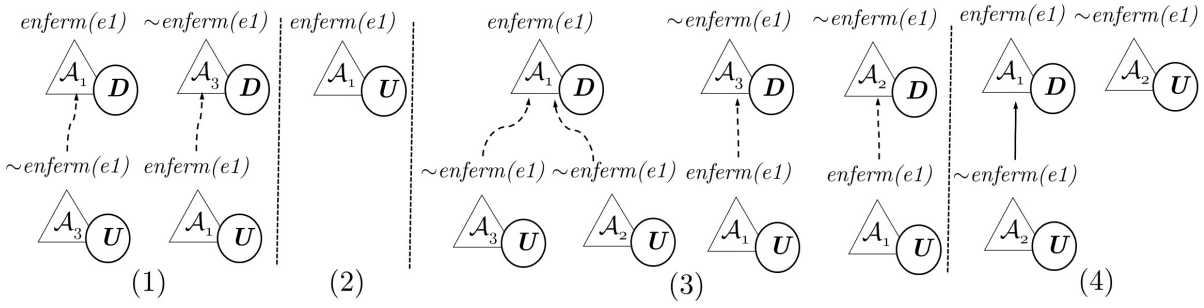

Figura 5.4: Arboles de dialéctica generados a partir de las consultas $\mathcal{CP}_{\mathsf{paciente}_1}$, $\mathcal{CP}_{\mathsf{paciente}_2}$, $\mathcal{CP}_{\mathsf{paciente}_3}$, y $\mathcal{CP}_{\mathsf{paciente}_4}$.

Para las primeras dos consultas ($\mathcal{CP}_{\mathsf{paciente}_1}$ y $\mathcal{CP}_{\mathsf{paciente}_2}$), el criterio utilizado fue el mismo, sin embargo diferían en los átomos distinguidos utilizados. Luego, para el caso de la consulta $\mathcal{CP}_{\mathsf{paciente}_3}$, a pesar que los átomos distinguidos eran los mismos que se definieron en la especificación de la consulta $\mathcal{CP}_{\mathsf{paciente}_2}$, era otro el criterio de preferencia utilizado. Por último, en la consulta $\mathcal{CP}_{\mathsf{paciente}_4}$ la *ECrit* hacía referencia a un criterio diferente al que se utilizó en consultas anteriores. A pesar de que el programa consultado y la consulta coincidían para cada *CPref*, en general las respuestas del *SRD* variaron para cada ejemplo de consulta en particular. Un sistema que pueda cambiar de criterio posiblemente obtenga respuestas diferentes. Esta consideración es una de las motivaciones para los desarrollos en esta tesis.

5.6. Ejemplo de aplicación en un entorno de recomendación de reportes de noticias

Mostraremos ahora como es posible mejorar la capacidad de razonamiento en ciertas aplicaciones que utilizan DeLP. En particular, se describirá como la implementación de un formalismo concreto de recomendación basado en DeLP [BCM13] puede beneficiarse al integrar a su arquitectura el modelo de los *SRPrefs* propuesto en este capítulo.

En [BCM13] se propone un marco para modelar la noción de confianza mediante programación lógica rebatible permitiendo que, a través de hechos, se puedan expresar declaraciones explícitas de confianza sobre reportes de noticias, fuentes de dichos reportes y sobre otros usuarios. Por otra parte, para modelar la propagación de la confianza los autores detallan un conjunto de postulados base representados mediante reglas rebatibles. Cada postulado expresa una situación diferente respecto a la credibilidad que el lector pueda asignar a los reportes de noticias; por ejemplo, un postulado podría decir que "*un reporte por lo general es poco confiable si proviene de una fuente poco confiable*". De esta manera, los postulados permitirán modelar la noción de confianza entre usuarios, reportes, y las fuentes de dichos reportes.

La Figura 5.5 muestra la arquitectura del sistema de manejo de confianza que los autores proponen utilizando el lenguaje de representación de conocimiento y razonamiento del sistema DeLP. Esta arquitectura se compone de una Base de Datos que contiene los datos asociados a las declaraciones de confianza del usuario; un Manejador de Noticias que se encarga de traducir a un lenguaje lógico apropiado, en este caso DeLP, los datos que le manda el sistema recomendador de noticias, y luego generar una lista de reportes respaldada por las declaraciones de confianza del usuario; y finalmente, el Sistema Recomendador de Noticias que es el componente que interactúa con el usuario final.

La idea subyacente en esta sección es integrar a la arquitectura del manejador de noticias un *SRPref* que permita tratar con preferencias definidas por el usuario. En otras palabras, se busca que el manejador pueda recibir y procesar preferencias definidas sobre las propias declaraciones de confianza de los usuario, las cuales se encuentran alojadas en la base de datos, como así también de los postulados de confianza que el sistema maneja. Como se muestra más adelante, el *SRPref* contará con la implementación de determinados criterios de preferencia cuyos átomos distinguidos facilitarán la tarea del usuario al momento de explicitar sus preferencias.

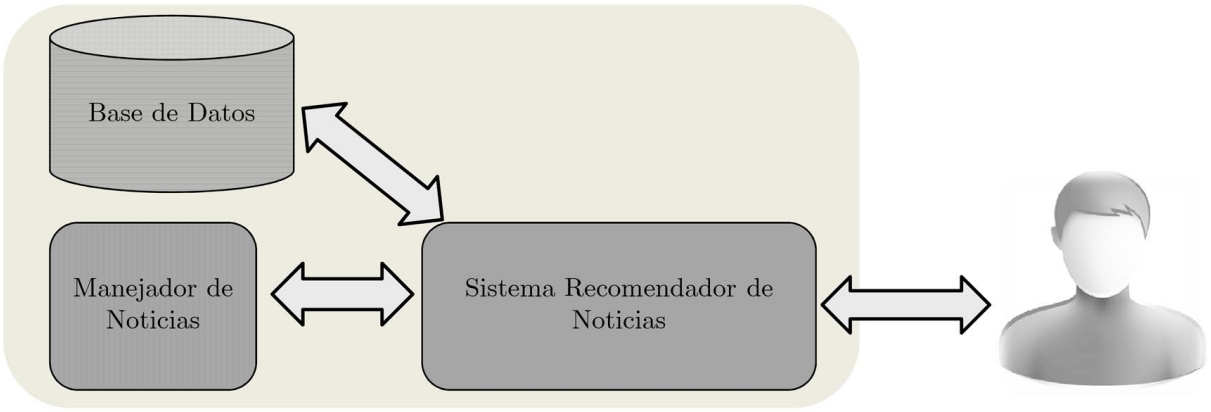

Figura 5.5: Arquitectura del Sistema de Manejo de Confianza.

El sistema recomendador le enviará al manejador de noticias toda la información necesaria para que construya las *CPrefs* que el *SRPref* deberá responder. Dentro de esta información se encuentran aquellos datos de usuarios almacenados en la base de datos que son relevantes al proceso de razonamiento. Esta información se traduce a hechos DeLP y luego se agrega a los postulados propuestos para completar el modelo de confianza. Como se muestra a continuación, a partir de este modelo, el *SRPref* computará las respuestas de las *CPrefs* recibidas y a partir de esta información el sistema generará las recomendaciones que luego serán entregadas al usuario.

Por simplicidad, a lo largo de este ejemplo de aplicación se utiliza parte del programa DeLP presentado en [BCM13]. El siguiente programa DeLP $\mathcal{P}_t$ representará la información considerada por el sistema de manejo de confianza.

$$
\begin{array}{rcl}
\sim trust\_report(V, R) & -\!\prec & report\_source(R, S), \sim trust\_source(V, S) \\
trust\_report(V, R) & -\!\prec & trust\_viewer(V, V_1), trust\_report(V_1, R) \\
\sim trust\_source(V, S) & -\!\prec & trust\_viewer(V, V_1), \sim trust\_source(V_1, S) \\
\sim trust\_report(V, R) & -\!\prec & report\_source(R, S), \sim trust\_source(V, S), \\
 & & trust\_viewer(V, V_1), trust\_report(V_1, R)
\end{array}
$$

$$
\begin{array}{l}
report\_source(facebook\_hits\_one\_billion, etc\_news) \\
\sim trust\_source(cristian, etc\_news) \\
trust\_report(marcela, facebook\_hits\_one\_billion) \\
trust\_viewer(ana, emanuel) \\
trust\_viewer(ana, cristian) \\
trust\_viewer(emanuel, marcela)
\end{array}
$$

Por un lado, los hechos en $\mathcal{P}_t$ expresan información relacionada a las fuentes de los reportes, *e.g.,* $report\_source(facebook\_hits\_one\_billion, etc\_news)$, como así también el juicio de confianza realizado por usuarios hacia: determinados reportes, *e.g.,* $trust\_report(marcela, facebook\_hits\_one\_billion)$, fuentes de donde se obtienen estos reportes, *e.g.,* $\sim trust\_source(cristian, etc\_news)$, y también hacia otros usuarios, *e.g.,* $trust\_viewer(ana, emanuel), trust\_viewer(ana, cristian), trust\_viewer(emanuel, marcela)$. Sea "*facebook_hits_one_billion*" un reporte informando que facebook recibió un billón de accesos durante el mes de octubre de 2016, las consultas que se detallan más adelante en esta sección se harán sobre dicho reporte.

Por otro lado, cada una de las reglas rebatibles en $\mathcal{P}_t$ modelan respectivamente los siguientes postulados:

Postulado 1. *Un reporte será por lo general poco confiable si proviene de una fuente poco confiable.*

Postulado 2. *Un reporte confiable para un usuario creíble, en general se dirá que es un reporte confiable.*

Postulado 3. *Una fuente poco confiable para un usuario creíble, en general se dirá que es una fuente poco confiable.*

Postulado 4. *Por más que exista un usuario creíble y el reporte sea confiable para este usuario, si proviene de una fuente poco confiable en general se dirá que el reporte es poco confiable.*

Como aplicación particular de un *SRPref*, en esta sección se utiliza un *Servicio de Razonamiento para Recomendación de Reportes* (*SRR*) $\langle \mathbb{I}_t, \langle \{\mathsf{impC}(\mathsf{prioR}), \mathsf{impC}(\mathsf{prioL})\}, \mathsf{compPref}_t \rangle \rangle$. Por lo tanto, para declarar las *ECrits* que el *SRR* recibe junto a las consultas se puede usar el criterio basado en prioridad entre reglas $\succsim_{\mathsf{prioR}}$ como así también el criterio basado en literales $\succsim_{\mathsf{prioL}}$, ambos presentados en la Sección 4.4. Los átomos del conjunto $\mathsf{atomsC}(\mathsf{prioR})$ para la implementación $\mathsf{impC}(\mathsf{prioR})$ se expresarán a través del predicado *es_mejorR*, el cual fue introducido en el Ejemplo 5.1. Por otra parte, los átomos distinguidos para la implementación $\mathsf{impC}(\mathsf{prioL})$ se especifican mediante el átomo $es\_mejorL(L_1, L_2)$ que expresa una preferencia del literal L_1 sobre L_2.

Para recomendar un reporte a un usuario el sistema debe determinar si es de confianza para dicho usuario. Para esto último, el *SRR* busca un argumento garantizado que soporte una declaración de confianza para dicho reporte, considerando para esto preferencias

específicas enviadas por el usuario. Como se describirá a continuación las preferencias se pueden definir sobre la información alojada en la base de datos o sobre los postulados de confianza del sistema. En lo que sigue se presentan algunas consultas al *SRR* que incluyen este tipo de preferencias.

Sea *ana* un usuario del sistema, suponiendo que *ana* da mayor importancia al juicio de confianza que ella hace de otros usuarios, la prioridad entre los postulados se expresa como:

$$\mathsf{D}_{\mathsf{confUsuario}} = \left\{ \begin{array}{l} (es\_mejorR(r_2, r_1)), \\ (es\_mejorR(r_2, r_3)) \end{array} \right\} \text{ tal que}$$

$$\begin{array}{lrcl} r_1: & \sim trust\_report(V, R) & \prec & report\_source(R, S), \sim trust\_source(V, S) \\ r_2: & trust\_report(V, R) & \prec & trust\_viewer(V, V_1), trust\_report(V_1, R) \\ r_3: & \sim trust\_report(V, R) & \prec & report\_source(R, S), \sim trust\_source(V, S), \\ & & & trust\_viewer(V, V_1), trust\_report(V_1, R) \end{array}$$

Por lo tanto, es posible construir una *ECrit* basada en la confiabilidad hacia los usuarios:

$$\mathcal{EC}_{\mathsf{confUsuario}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{confUsuario}})$$

A partir de esta especificación, es posible ejecutar la siguiente consulta para determinar si $facebook\_hits\_one\_billion$ es un reporte confiable para *ana*:

$$[\mathcal{EC}_{\mathsf{confUsuario}}, \mathcal{P}_t, trust\_report(ana, facebook\_hits\_one\_billion)]$$

Para responder la consulta, el intérprete de programación lógica rebatible del *SRR* buscará, a partir del programa $\mathcal{P}_t$, un argumento no derrotado para $trust\_report(ana, facebook\_hits\_one\_billion)$. A continuación, en la Figura 5.6 se introduce una notación gráfica que muestra que el reporte $facebook\_hits\_one\_billion$ es de confianza por que existe un argumento garantizado para $trust\_report(ana, facebook\_hits\_one\_billion)$, es decir la respuesta obtenida del servicio será SI. Note que la notación gráfica utilizada en la Figura 5.6 caracteriza a los argumentos mediante triángulos, y que los triángulos dentro de los triángulos mayores denotan subargumentos. La información en el vértice superior de los triángulos corresponde a la conclusión del argumento (subargumento), y la información dentro del triángulo corresponde a los literales utilizados para construir el argumento.

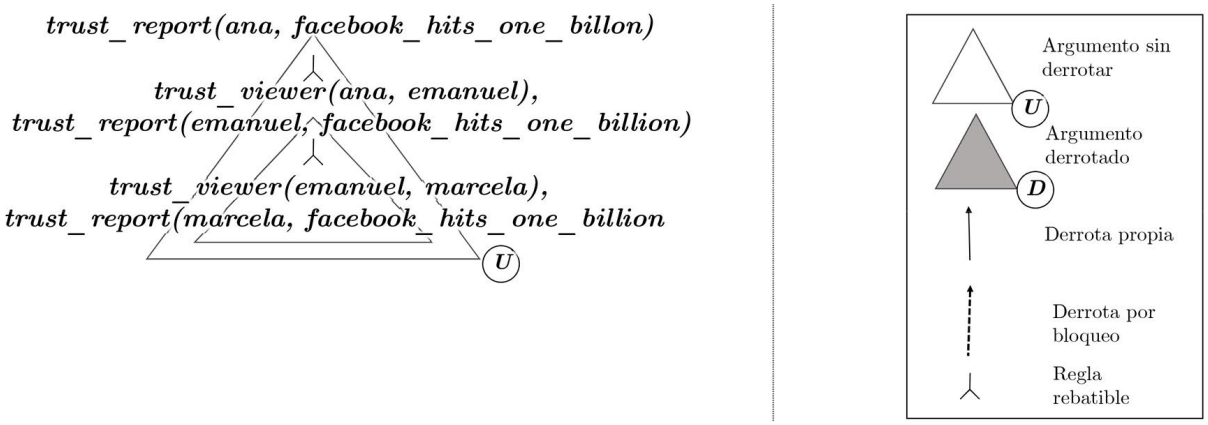

Figura 5.6: Árbol de dialéctica DeLP mostrando razones para confiar en $facebook\_hits\_one\_billion$.

Ahora supongamos que *ana* modifica sus preferencias sobre los postulados priorizando el juicio de confianza que tiene la fuente de donde proviene el reporte consultado. Entonces, las prioridades podrían expresar que:

$$\mathsf{D}_{\mathsf{confFuente}} = \left\{ \begin{array}{l} (es\_mejorR(r_1, r_2)), \\ (es\_mejorR(r_3, r_2)) \end{array} \right\}$$

Teniendo en consideración estas prioridades, se puede declarar una *ECrit* basada en el juicio de confianza realizado sobre las fuentes de los reportes.

$$\mathcal{EC}_{\mathsf{confFuente}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{confFuente}})$$

En este escenario, el *SRR* puede recibir la siguiente *CPref*:

$$[\mathcal{EC}_{\mathsf{confFuente}}, \mathcal{P}_t, trust\_report(ana, facebook\_hits\_one\_billion)]$$

Al momento de responder esta *CPref*, se calcula la respuesta para $trust\_report(ana, facebook\_hits\_one\_billion)$ dando mayor relevancia a la información de prioridad incluida en $\mathsf{D}_{\mathsf{confFuente}}$. En la Figura 5.7 se puede observar que existe un argumento garantizado soportando la declaración $\sim trust\_report(ana, facebook\_hits\_one\_billion)$. En este caso el *SRR* devuelve NO, y $facebook\_hits\_one\_billion$ no se recomienda como un reporte confiable para *ana*.

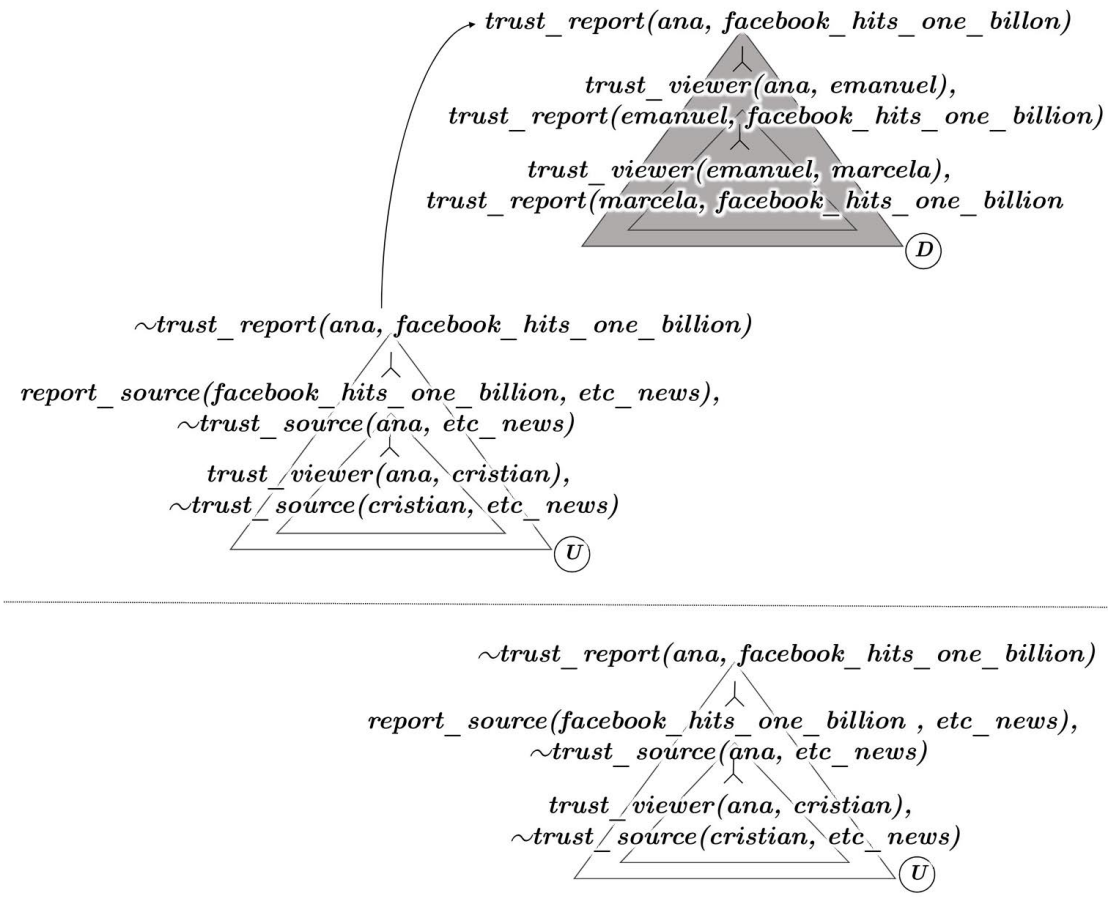

Figura 5.7: Árboles de dialéctica DeLP mostrando razones para no confiar en $facebook\_hits\_one\_billion$.

Hasta el momento se presentaron dos alternativas diferentes donde un usuario accede al sistema y decide priorizar ciertos postulados sobre otros. Consideremos ahora un escenario donde el usuario puede expresar prioridades sobre la información almacenada en la base de datos del sistema. Es importante recordar que esta información en [BCM13] se representa mediante hechos que se agregan al modelo de confianza (programa DeLP) del sistema en cuestión. Por lo tanto, suponga que *ana* prioriza la credibilidades que tiene sobre otros usuarios. Esta prioridad se puede expresar a través del conjunto:

$\mathsf{D}_{\mathsf{credibil}} = \left\{ \; es\_mejorL(trust\_viewer(ana, emanuel), trust\_viewer(ana, cristian)) \; \right\}$

Considerando este conjunto, es posible declarar la siguiente *ECrit*.

$$\mathcal{EC}_{\mathsf{credibil}} = (\mathsf{prioL}, \mathsf{D}_{\mathsf{credibil}})$$

Con esta especificación, el *SRR* puede recibir la siguiente *CPref*:

$$[\mathcal{EC}_{\mathsf{credibil}}, \mathcal{P}_t, trust\_report(ana, facebook\_hits\_one\_billion)]$$

La Figura 5.8 muestra que existe un argumento garantizado para $trust\_report(ana, facebook\_hits\_one\_billion)$. Por lo tanto, el *SRR* devuelve SI, y el sistema recomienda a *ana* el reporte $facebook\_hits\_one\_billion$.

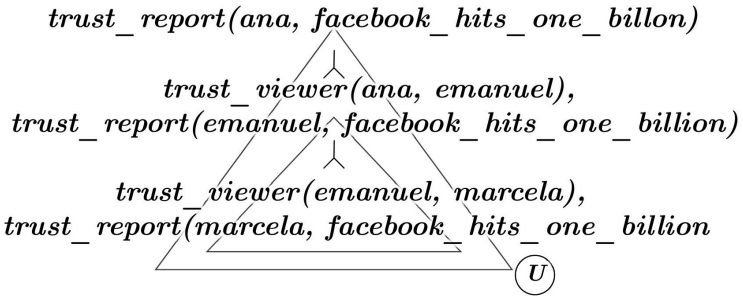

Figura 5.8: Árbol de dialéctica DeLP mostrando razones para confiar en $facebook\_hits\_one\_billion$.

En el ejemplo introducido en esta sección se mostró un sistema de recomendación que utiliza un *SRPref* para el soporte de sus recomendaciones. Además, se presentaron tres *CPrefs* cuyas *ECrits* se construyeron en base a las preferencias del usuario respecto a la información representada por el sistema. Finalmente, a partir del resultado obtenido para cada *CPref* se mostraron las diferentes respuestas generadas por el sistema.

5.7. Conclusión

En este capítulo se introdujeron los servicios de razonamiento rebatible basados en preferencias (*SRPref*) y las consultas basadas en preferencias (*CPref*). Dichas formalizaciones representan uno de los principales aportes de esta tesis y tienen como objetivo permitir el desarrollo de servicios que respondan consultas de manera personalizada, y se puedan configurar automáticamente para que resuelvan el conflicto entre información contradictoria mediante el criterio que mejor se ajusta a un escenario particular.

Para consultar un *SRPref* se utilizan las *CPrefs*, una *CPref* contiene toda la información que el servicio necesita para responder una consulta, siendo a la vez una herramienta para expresar las preferencias del usuario. Este tipo de consulta incluye el conocimiento

que un *SRPref* utiliza para obtener una respuesta, así como también una especificación del criterio que mejor se ajusta a la consulta correspondiente. Estas especificaciones indican declarativamente el criterio a utilizar para realizar la comparación entre argumentos, como también la información adicional al programa consultado que la implementación de dicho criterio puede llegar a necesitar. Cada servicio está constituido por un intérprete DeLP para resolver las consultas y de un módulo encargado de la comparación de argumentos. Este módulo a su vez contiene las implementaciones de los criterios de preferencia que el servicio puede utilizar.

Entonces, un *SRPref* permite configurar para cada consulta particular el criterio que se utiliza para resolver los conflictos entre información contradictoria que puedan surgir. Esta decisión, se basa, fundamentalmente en que en muchos escenarios del mundo real el criterio para comparar información varía dependiendo del contexto en el que se hace la comparación. En ciertos contextos se requiere la utilización de un criterio general para todos los escenarios planteados, es decir el sistema de razonamiento se configura de manera tal que se utiliza un criterio fijo; sin embargo, existen muchos otros casos, como el que se ilustra en el ejemplo de aplicación de la Sección 5.5, en el que el criterio puede cambiar dependiendo del escenario. Para estas situaciones el sistema debe contar con mecanismos computacionales que le permita un cambio de criterio haciéndolo adaptable. Este capítulo mostró un formalismo que contribuye a resolver esta cuestión.

Finalmente, en la Sección 5.5 y 5.6 se mostraron dos ejemplos concretos de posibles dominios de aplicación; uno para el diagnóstico de enfermedades y el otro para la recomendación de reportes de noticias basados en la confiabilidad de los mismos. En el primer ejemplo se detallaron diferentes posibles consultas que podría recibir el recomendador, en cada una de éstas se utilizó una *ECrit* diferente. El segundo ejemplo fue tomado de [BCM13], con el objetivo principal de mostrar cómo nuestra propuesta se puede integrar fácilmente a un formalismo ya existente.

Capítulo 6

Servicio de Razonamiento Rebatible basado en Preferencias Condicionales

Las consultas basadas en preferencias (*CPref*) definidas en el capítulo anterior permiten que el criterio de preferencia que vaya a utilizar un servicio de razonamiento rebatible basado en preferencias (*SRPref*) pueda variar para cada consulta recibida. En este capítulo se presentará el concepto de *Consulta basada en Preferencias Condicionales* (*CPCond*) que ofrece una alternativa distinta respecto a la forma en la cuál un usuario puede expresar sus preferencias. Estas consultas contarán con una expresión formal que permitirá decidir cuál es el criterio de preferencia que debe ser utilizado en cada situación específica.

Para resolver las *CPConds* se introducirá un nuevo tipo de servicio de razonamiento que extiende las capacidades de los *SRPrefs* recientemente introducidos, que denominaremos *Servicio de Razonamiento Rebatible basado en Preferencias Condicionales* (*SRPCond*). Estos servicios tienen la capacidad de responder consultas seleccionando un criterio de preferencia que será indicado utilizando una expresión condicional incluida en la propia consulta.

El capítulo estará estructurado como sigue. En la Sección 6.1 se formalizarán las nociones de *CPCond* y *SRPCond*, y cómo estas consultas interactúan con los componentes de dichos servicios. Luego, en la Sección 6.2 se mostrarán algunas propiedades de cómo un *SRPCond* maneja la selección de una especificación de criterio (*ECrit*). Finalmente, en la Sección 6.3 se muestra un ejemplo de aplicación que integra todo lo desarrollado en este capítulo.

6.1. Consulta basada en Preferencias Condicionales

Introduciremos ahora un nuevo tipo de consulta que permite que en la misma se puedan declarar varios *ECrits*. Asimismo, la elección del criterio de preferencia, por parte de los *SRPConds* encargados de responder este tipo de consultas, dependerá de una expresión que será utilizada para seleccionar una de las *ECrits* incluidas en la consulta correspondiente. Un *SRPCond* Φ estará conformado por:

- el *intérprete* DeLP que resuelve las consultas DeLP,
- una *función de evaluación* que, a partir de una *expresión de preferencia condicional*, obtiene la *ECrit* que declara la implementación de criterio que se utilizará en la comparación de argumentos, y
- un *módulo de comparación de argumentos* que computa las preferencias entre argumentos considerando para ello la *ECrit* que se obtiene desde la función de evaluación.

La Figura 6.1 muestra la arquitectura del modelo de servicio de razonamiento que se formalizará en este capítulo.

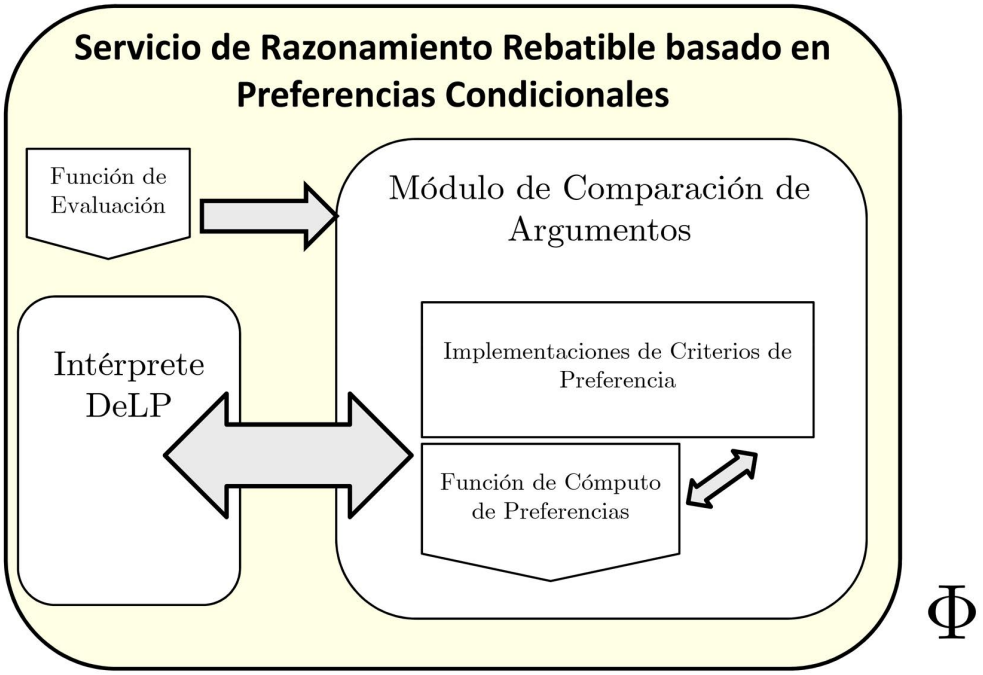

Figura 6.1: Arquitectura de un *SRPCond*.

Para la selección dinámica de *ECrits*, en esta tesis se propone la utilización de condiciones; en este sentido, las guardas ofrecen una forma de establecer condiciones. Una guarda podría ser vista como una forma de guiar la elección de una especificación dependiendo de cierta información almacenada en un programa DeLP. Formalmente:

Definición 6.1. *Un conjunto de literales $\mathcal{G}$ denominado guarda, se satisface para un programa DeLP $\mathcal{P}$ si y solo si para cada literal $L \in \mathcal{G}$ existe una derivación estricta a partir de $\mathcal{P}$.*

Observación 6.1. *La guarda $\emptyset$ se satisface para cualquier programa DeLP.*

Es importante destacar que la motivación para utilizar derivaciones estrictas para determinar si una guarda se satisface para un determinado programa DeLP $\mathcal{P}$, y a diferencia de otras alternativas como lo son la utilización de derivación rebatible o garantía, se debe a que no hay necesidad de contar con un criterio de preferencia auxiliar para decidir entre información contradictoria ya que el conjunto Π de $\mathcal{P}$ se asume no contradictorio.

Ejemplo 6.1. *Considere las guardas $\mathcal{G}_1 = \{e, f, \sim p\}$, $\mathcal{G}_2 = \{a, \sim p\}$, $\mathcal{G}_3 = \{b\}$ y $\mathcal{G}_4 = \emptyset$; y el programa DeLP $\mathcal{P}_{6.1} = (\Pi_{6.1}, \Delta_{6.1})$ con los siguientes conjuntos de reglas: $\Pi_{6.1} = \{(a \leftarrow b), (f \leftarrow h), (\sim p \leftarrow h, e), d, e, h\}$, y $\Delta_{6.1} = \{(b \prec d), (\sim b \prec d, e)\}$. Como todos los literales en $\mathcal{G}_1$ tienen derivaciones estrictas a partir de $\mathcal{P}_{6.1}$, la guarda $\mathcal{G}_1$ se satisface para $\mathcal{P}_{6.1}$. Como $\emptyset$ siempre se satisface para cualquier programa DeLP; entonces, $\mathcal{G}_4$ se satisface para $\mathcal{P}_{6.1}$. Sin embargo, $\mathcal{G}_2$ y $\mathcal{G}_3$ no se satisfacen para $\mathcal{P}_{6.1}$.*

Como se verá más adelante, un *SRPCond* responderá una consulta considerando la implementación del criterio de preferencia referenciado en la *ECrit* que resulta de evaluar una expresión de preferencia condicional (*exp-cond*). De hecho, una *CPCond* incluirá una *exp-cond* y, de esta manera, el servicio de razonamiento será capaz de determinar el criterio utilizado para responder la correspondiente consulta. Como se formaliza a continuación, una *exp-cond* será una *ECrit* o una guarda $\mathcal{G}$ seguida de dos expresiones de preferencias condicionales.

Definición 6.2 (Expresión de Preferencia Condicional)**.** *Dado un conjunto* C *de implementaciones computacionales de criterios de preferencia. Sea* $\mathsf{impC(Ic)}$ *un elemento en* C*, y* $(\mathsf{Ic}, \mathsf{D})$ *una especificación de criterio para* $\mathsf{impC(Ic)}$*. Sea $\mathcal{G}$ un conjunto de literales. Una exp-cond $\mathcal{E}$ es una secuencia finita de símbolos definida recursivamente como sigue:*

$$\mathcal{E} = \begin{cases} (\mathsf{Ic}, \mathsf{D}), \ o \\ [\mathcal{G} : \mathcal{E}_1; \mathcal{E}_2] \text{ donde } \mathcal{E}_1 \text{ y } \mathcal{E}_2 \text{ son exps-conds.} \end{cases}$$

Dado un conjunto de implementaciones de criterio de preferencia C, el conjunto de todas las posibles *exps-conds* construidas a partir de C se denotará D_{e}. Una *exp-cond* $\mathcal{E}$ se interpreta de la siguiente manera: si $\mathcal{E}$ es una especificación de criterio $(\mathsf{Ic}, \mathsf{D})$, entonces se aplica dicha especificación en la comparación entre argumentos, mientras que si $\mathcal{E}$ es $[\mathcal{G} : \mathcal{E}_1; \mathcal{E}_2]$ y $\mathcal{G}$ se satisface para un programa DeLP $\mathcal{P}$, entonces se evaluá $\mathcal{E}_1$, caso contrario, si $\mathcal{G}$ no es satisfecho, $\mathcal{E}_2$ será evaluada. Esta idea intuitiva de cómo una *exp-cond* se evalúa será capturada por la función evalE, la cual se define a continuación:

Definición 6.3 (Función de Evaluación)**.** *Sea Φ un SRPCond. Sea D_{p} el dominio de todos los posibles programas DeLP. Sea C un conjunto de implementaciones computacionales de criterios de preferencia almacenadas en el servicio Φ, D_{e} el conjunto de todas las posibles exps-conds construidas a partir de C, y S el conjunto de todas las especificaciones de criterios involucradas en las exps-conds de D_{e}. La función de evaluación para exps-conds $\mathsf{evalE} : D_{\mathsf{e}} \times D_{\mathsf{p}} \longrightarrow S$, es tal que dada una exp-cond $\mathcal{E}$ en D_{e} y un programa DeLP $\mathcal{P}$ en D_{p}, se define como:*

$$\mathsf{evalE}(\mathcal{E}, \mathcal{P}) = \begin{cases} (\mathsf{Ic}, \mathsf{D}) \text{ si } \mathcal{E} = (\mathsf{Ic}, \mathsf{D}), \text{ o} \\ \mathsf{evalE}(\mathcal{E}_1, \mathcal{P}) \text{ si } \mathcal{E} = [\mathcal{G} : \mathcal{E}_1; \mathcal{E}_2] \text{ y } \mathcal{G} \text{ se satisface para } \mathcal{P}, \text{ o} \\ \mathsf{evalE}(\mathcal{E}_2, \mathcal{P}) \text{ si } \mathcal{E} = [\mathcal{G} : \mathcal{E}_1; \mathcal{E}_2] \text{ y } \mathcal{G} \text{ no se satisface para } \mathcal{P} \end{cases}$$

Ejemplo 6.2. *Considere las especificaciones de criterios $\mathcal{EC}_{seg}$ y $\mathcal{EC}_{conf}$ ambas introducidas en el Ejemplo 5.3 del Capítulo 5. A partir de estas especificaciones es posible construir las siguientes exps-conds:*

$$\mathcal{E}_1 = [\{zPolicias(h1), estrellas(h1, 5)\} : \mathcal{EC}_{\mathsf{conf}}; \mathcal{EC}_{\mathsf{seg}}]$$
$$\mathcal{E}_2 = [\{zRobos(h1), cVehicular\} : \mathcal{EC}_{\mathsf{seg}}; [\{estrellas(h1, 5)\} : \mathcal{EC}_{\mathsf{conf}}; \mathcal{EC}_{\mathsf{seg}}]]$$

La expresión $\mathcal{E}_1$ se leerá como: "si existe una derivación estricta para $zPolicias(h1)$ y $estrellas(h1, 5)$ entonces se usa la especificación del criterio que favorece el confort del conductor, de lo contrario se usa la especificación del criterio que favorece la seguridad del conductor". Mientras que, la expresión $\mathcal{E}_2$ se interpreta de la siguiente manera: "si existe una derivación estricta para $zRobos(h1)$ y $cVehicular$, entonces se usa la especificación del criterio basado en la seguridad, de lo contrario la expresión $[\{estrellas(h1, 5)\} : \mathcal{EC}_{\mathsf{conf}}; \mathcal{EC}_{\mathsf{seg}}]$ debería ser evaluada". Considere el programa $\mathcal{P}_h$ del Ejemplo de Aplicación 4.5 del Capítulo 4. Para $\mathcal{E}_1$, la guarda $\{zPolicias(h1), estrellas(h1, 5)\}$ no se satisface para $\mathcal{P}_h$, ya que $zPolicias(h1)$ no tiene una derivación estricta; por lo tanto $\mathsf{evalE}(\mathcal{E}_1, \mathcal{P}_h) =$

$\mathcal{EC}_{\mathsf{seg}}$. *Para* $\mathcal{E}_2$*, la guarda* $\{zRobos(h1), cVehicular\}$ *no se satisface para el programa, por que cVehicular no tiene una derivación estricta; entonces se debe evaluar la subexpresión* $[\{estrellas(h1, 5)\} : \mathcal{EC}_{\mathsf{conf}}; \mathcal{EC}_{\mathsf{seg}}]$*. Para esta subexpresión,* $estrellas(h1, 5)$ *tiene una derivación estricta (es un hecho del programa* $\mathcal{P}_h$*), entonces la guarda* $\{estrellas(h1, 5)\}$ *se satisface para* $\mathcal{P}_h$*; por lo tanto se tiene que* $\mathsf{evalE}(\mathcal{E}_2, , \mathcal{P}_h) = \mathcal{EC}_{\mathsf{conf}}$.

A partir de la utilización de una *exp-cond* es posible programar cual es el criterio de preferencia que un *SRPCond* seleccionará en cada situación en particular. Como la forma en la que se ha modelado la función para la evaluación de las *exps-conds* corresponde a un mecanismo de selección que depende de las guardas que se satisfacen para un programa DeLP dado, al considerar programas diferentes es posible que la *ECrit* seleccionada varíe; por consiguiente, es posible que los criterios utilizados en cada situación en particular sean diferentes.

Como se ha mencionado anteriormente, un *SRPCond* puede responder consultas que contienen *exps-conds*. Además de una *exp-cond*, estas consultas se caracterizan por tener un programa DeLP, y un literal con la consulta DeLP correspondiente. Esta noción se define como sigue.

Definición 6.4 (Consulta basada en Preferencias Condicionales (*CPCond*))**.** *Sea* D_{p} *el dominio de todos los posibles programas DeLP, y* D_{c} *el dominio de las consultas DeLP. Una consulta basada en preferencias condicionales es una tupla* $\langle \mathcal{E}, \mathcal{P}, Q \rangle$*, donde* $\mathcal{E}$ *es una exp-cond,* $\mathcal{P}$ *un programa en el dominio* D_{p}*, y* Q *una consulta perteneciente al dominio* D_{c}.

Note que un aspecto importante en este tipo de consultas es la posibilidad de manejar, a partir de las *ECrits* que se encuentran en una *exp-cond*, múltiples criterios de una forma automática y en una misma consulta. Esta característica proporciona nuevos avances en la mejora de las capacidades de razonamiento de los sistemas basados en DeLP. En particular, el mecanismo de razonamiento de estos sistemas podrá contar con varias *ECrits* y ante una situación en particular se puede utilizar el criterio que mejor se le adapte. Esto hace que dichos sistemas sean muchos más versátiles a los cambios de preferencias de los usuarios o situaciones que vayan surgiendo. En la Sección 6.3 se presenta un ejemplo en donde se ilustra la importancia de tener disponible varias *ECrits* y usar la que mejor se ajusta a la información de dominio considerada en un momento dado.

Entonces, a partir de un programa DeLP y una consulta, el servicio de razonamiento que se presenta a continuación invocará al intérprete DeLP para que resuelva la consulta,

y enviará a su módulo de comparación de argumentos la *ECrit* obtenida luego de la evaluación de una *exp-cond*. Este tipo de servicio de razonamiento capacitado para resolver consultas con expresiones condicionales se denomina, Servicio de Razonamiento Rebatible basado en Preferencias Condicionales (*SRPCond*).

Definición 6.5 (Servicio de Razonamiento Rebatible basado en Preferencias Condicionales (*SRPCond*))**.** *Un servicio de razonamiento rebatible basado en preferencias condicionales* Φ *es una tupla* $\langle \mathsf{evalE}, \mathbb{I}, \mathbb{M} \rangle$*, donde* evalE *es una función de evaluación para exps-conds,* $\mathbb{I}$ *un intérprete DeLP, y* $\mathbb{M}$ *un módulo de comparación de argumentos.*

Definición 6.6 (*CPCond* adecuada para un *SRPCond*)**.** *Sea* $\Phi = \langle \mathsf{evalE}, \mathbb{I}, \mathbb{M} \rangle$ *un SRPCond. Una consulta basada en preferencias condicionales* $\langle \mathcal{E}, \mathcal{P}, Q \rangle$ *será adecuada para* Φ*, si cada especificación de criterio* $\mathcal{EC}$ *involucrada en* $\mathcal{E}$ *es adecuada para* $\mathbb{M}$ *en* Φ*.*

De manera similar a como se definió en la Definición 5.7 del Capítulo 5, la respuesta a una *CPCond* de un *SRPCond* corresponde a la respuesta que el intérprete pueda dar a la consulta DeLP recibida.

Definición 6.7 (Respuesta para una *CPCond*)**.** *Sea* $\Phi = \langle \mathsf{evalE}, \mathbb{I}, \mathbb{M} \rangle$ *un SRPCond y* $\mathcal{CPC} = \langle \mathcal{E}, \mathcal{P}, Q \rangle$ *una consulta basada en preferencias condicionales adecuada para* Φ*. Sea* $\mathcal{EC}$ *la especificación de criterio obtenida desde* $\mathsf{evalE}(\mathcal{E}, \mathcal{P})$*. La respuesta para* $\mathcal{CPC}$ *en* Φ*, denotada* $\mathsf{resp}(\Phi, \mathcal{CPC})$*, corresponde al resultado del intérprete DeLP* $\mathbb{I}(\mathcal{P}, Q)$*, siendo que se verifica la siguiente condición. Dados dos argumentos* $\langle \mathcal{A}, L \rangle$ *y* $\langle \mathcal{B}, M \rangle$ *obtenidos desde el programa DeLP* $\mathcal{P}$*,*

- $\langle \mathcal{A}, L \rangle \succ \langle \mathcal{B}, M \rangle$ *si* $\mathsf{respMod}(\mathbb{M}, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle, \mathcal{EC}) = \langle \mathcal{A}, L \rangle$*, y*
- $\langle \mathcal{A}, L \rangle \asymp \langle \mathcal{B}, M \rangle$ *si* $\mathsf{respMod}(\mathbb{M}, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle, \mathcal{EC}) = \emptyset$*.*

Proposición 6.1. *Las CPrefs para un SRPref* $\Sigma = \langle \mathbb{I}, \mathbb{M} \rangle$ *son un caso particular de las CPConds recibidas por un SRPCond* $\Phi = \langle \mathsf{evalE}, \mathbb{I}, \mathbb{M} \rangle$*.*

Demostración: Una consulta basada en preferencias $[\mathcal{EC}, \mathcal{P}, Q]$ se puede escribir como una consulta basada en preferencias condicionales $\langle \mathcal{EC}, \mathcal{P}, Q \rangle$. Para cada par de argumentos que el intérprete DeLP necesite comparar, la respuesta del módulo de comparación de argumentos $\mathbb{M}$ será la misma para ambos servicios ya que en ambos casos se utiliza la misma especificación de criterio $\mathcal{EC}$. Por Definición 5.7 y 6.7 la respuesta para ambos servicios corresponde al resultado del intérprete DeLP $\mathbb{I}(\mathcal{P}, Q)$, por lo tanto, la respuesta para $[\mathcal{EC}, \mathcal{P}, Q]$ en Σ es la misma respuesta que daría Φ para la consulta $\langle \mathcal{EC}, \mathcal{P}, Q \rangle$. □

Ejemplo 6.3. *Considere el servicio* $\Phi_h = \langle \mathsf{evalE}_h, \mathbb{I}_h, \langle \{\mathsf{impC}(\mathsf{prioR}), \mathsf{impC}(\mathsf{esp})\}, \mathsf{compPref}_h \rangle \rangle$, *cuyo principal objetivo será sugerir hoteles a partir del conocimiento representado por el programa* $\mathcal{P}_h$ *presentado en el ejemplo de la Sección 4.5. A partir de las exps-conds* $\mathcal{E}_1$ *y* $\mathcal{E}_2$ *presentadas en el Ejemplo 6.2, suponga la siguiente CPCond recibida por* Φ_h,

$$\mathcal{CPC}_1 = \langle \mathcal{E}_1, \mathcal{P}_h, sugerir(h1) \rangle$$

.

En primer lugar, para computar una respuesta para $\mathcal{CPC}_1$, *el servicio* Φ_h *evaluará la exp-cond* $\mathcal{E}_1$. *Como se muestra en el Ejemplo 6.2, el resultado de* evalE_h *será* $\mathcal{EC}_{\mathsf{seg}}$. *A partir de este momento,* Φ_h *resuelve la consulta como si hubiese recibido una CPref utilizando la especificación de criterio* $\mathcal{EC}_{\mathsf{seg}}$. *Por lo tanto, para determinar si se sugiere el hotel "*$h1$*", el intérprete DeLP utiliza el programa* $\mathcal{P}_h$, *y la consulta sugerir*$(h1)$*; mientras que las preferencias entre argumentos las resuelve el MCA del SRPCond considerando la especificación de criterio* $\mathcal{EC}_{\mathsf{seg}}$. *En el Ejemplo 5.4 se muestra que la respuesta del intérprete utilizando esta información es,* NO. *Ahora considere la consulta basada en preferencias condicionales,*

$$\mathcal{CPC}_2 = \langle \mathcal{E}_2, \mathcal{P}_h, sugerir(h1) \rangle$$

Según el Ejemplo 6.2, la evaluación de $\mathcal{E}_2$ *da como resultado la especificación de criterio* $\mathcal{EC}_{\mathsf{conf}}$. *A partir de esta ECrit, en el Ejemplo 5.4 se muestra que el intérprete DeLP determina que la respuesta de* $\mathcal{P}_h$ *para la consulta sugerir*$(h1)$, *es* SI.

Note que una vez seleccionado la *ECrit* el proceso para resolver la respuesta para una *CPCond* corresponde al mismo que daría un *SRPref* para una *CPref*. En la Figura 6.2 se muestra de forma esquemática los elementos involucrados en el calculo de la respuesta para una *CPCond*. En particular, en esta figura, las flechas muestran como son utilizados los datos por el *SRPCond*. La interpretación de las flechas en el gráfico es la misma que se utilizó en la Figura 5.2. Por ejemplo, la *exp-cond* $\mathcal{E}$ y el programa DeLP $\mathcal{P}$ son utilizados por la función evalE para seleccionar una *ECrit*. Luego, a partir de esta especificación y los argumentos enviados por el intérprete DeLP, el módulo de comparación de argumentos devolverá una preferencia entre dichos argumentos. Finalmente, a partir del programa DeLP $\mathcal{P}$ y la consulta DeLP Q, el intérprete obtiene una respuesta para Q.

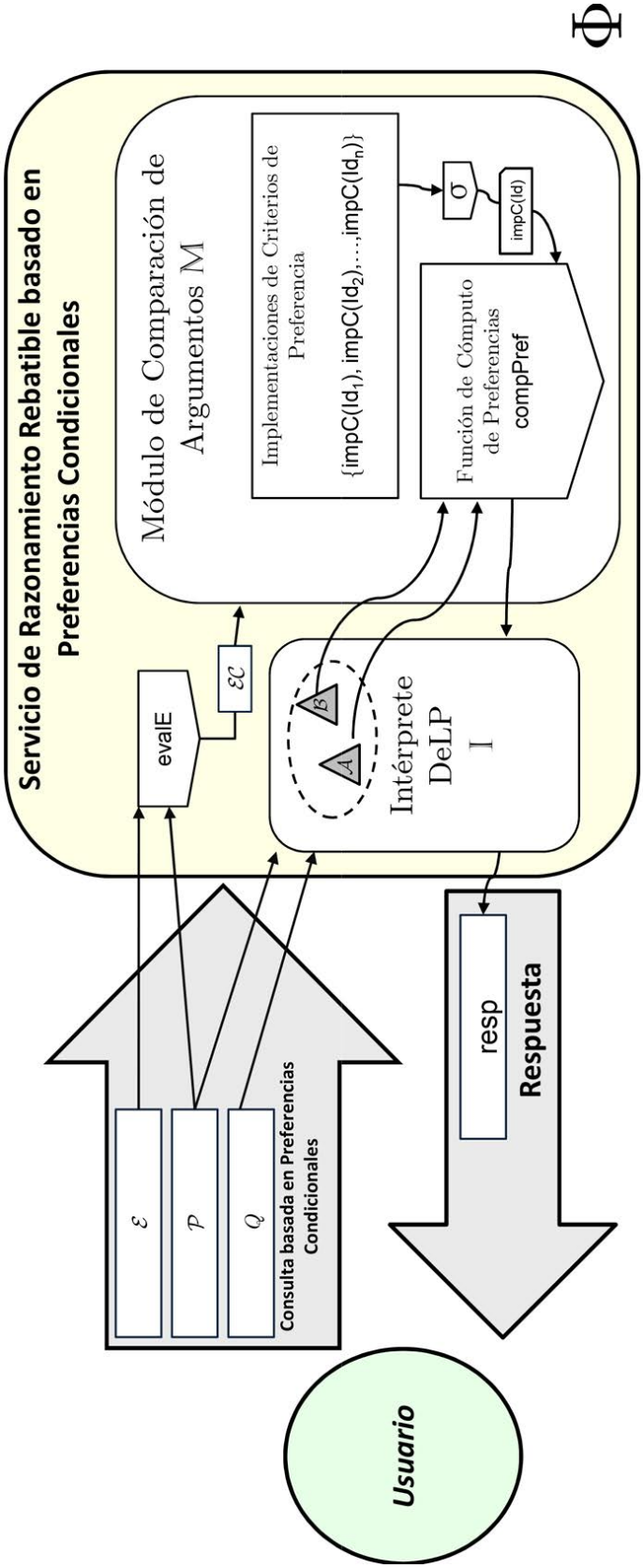

Figura 6.2: Principales componentes de una *CPCond* y un *SRPCond*.

En la próxima sección se analizarán y estudiarán algunas propiedades de las *exps-conds*. Luego, se muestra un ejemplo en un entorno de posibles robos donde se aplican los conceptos desarrollados en este capítulo.

6.2. Representación de Árbol para Expresiones Condicionales

A continuación se realiza un análisis detallado de las propiedades tanto de la *exp-cond* como de su semántica de evaluación, considerando para tal fin características relevantes que darán lugar a la optimización de estas *exps-conds*.

En esta sección, las expresiones de preferencias condicionales se representan a través de un árbol binario completo donde cada nodo interno está etiquetado con una guarda, mientras que cada nodo hoja con una *ECrit*. Un caso especial de este tipo de árbol binario será cuando la raíz es también un nodo hoja el cual está etiquetado con una especificación de criterio $(\mathsf{Ic}, \mathsf{D})$ y corresponde a la representación de la *exp-cond* más simple que se puede construir ($\mathcal{E} = (\mathsf{Ic}, \mathsf{D})$).

Definición 6.8 (Representación de Árbol)**.** *Dada una expresión de preferencia condicional $\mathcal{E}$ y el conjunto de especificaciones de criterios $\mathbb{N}$ involucradas en $\mathcal{E}$, una representación de árbol para $\mathcal{E}$, denotada $\mathsf{T}_{\mathcal{E}}$, es un árbol binario completo definido recursivamente como sigue:*

1. *si $\mathcal{E} = (\mathsf{Ic}, \mathsf{D})$, $(\mathsf{Ic}, \mathsf{D}) \in \mathbb{N}$, entonces $\mathsf{T}_{\mathcal{E}}$ contiene únicamente un nodo etiquetado con $(\mathsf{Ic}, \mathsf{D})$.*
2. *Si $\mathcal{E} = [\mathcal{G} : \mathcal{E}_i; \mathcal{E}_j]$, entonces la representación de árbol $\mathsf{T}_{\mathcal{E}}$ es un árbol binario completo donde:*
   - *la raíz está etiquetada con la guarda $\mathcal{G}$,*
   - *el hijo izquierdo de la raíz es la representación de árbol para $\mathcal{E}_i$, y*
   - *el hijo derecho de la raíz es la representación de árbol para $\mathcal{E}_j$.*

Como se mostrará a continuación la representación de árbol de una *exp-cond* será útil para describir algunos resultados formales. La siguiente proposición muestra que para cualquier *exp-cond* existe una única representación de árbol asociada a ésta. Esta proposición será utilizada también para la prueba de un resultado importante al final de esta sección.

Proposición 6.2. *Sea $\mathcal{E}$ una exp-cond. Existe una única representación de árbol $\mathsf{T}_{\mathcal{E}}$ para $\mathcal{E}$.*

Demostración: Directa por Definición 6.8. □

La Figura 6.3 ilustra la representación de árbol para las *exps-conds* $\mathcal{E}_1$ y $\mathcal{E}_2$ introducidas en el Ejemplo 6.2. En particular, estas *exps-conds* muestran varios niveles de condiciones anidadas, por lo que serán de gran utilidad en lo que queda de la sección.

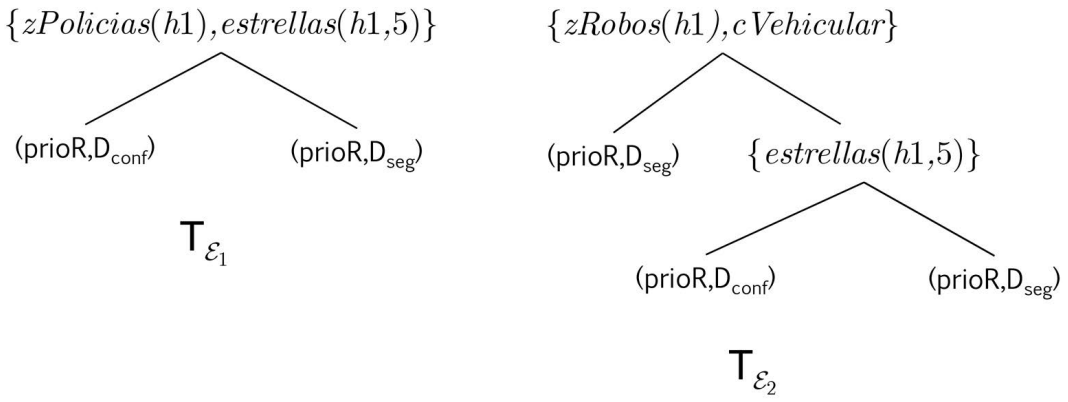

Figura 6.3: Las representaciones para las *exps-conds* $\mathcal{E}_1$ y $\mathcal{E}_2$ del Ejemplo 6.2.

Ejemplo 6.4. *Considere las siguientes expresiones de preferencias condicionales:*

$\mathcal{E}_a = (\mathsf{lc}_1, \mathsf{D}_1)$

$\mathcal{E}_b = [\{f, e, q\} : [\{\sim p, \sim a\} : [\{m, a\} : (\mathsf{lc}_1, \mathsf{D}_1); (\mathsf{lc}_2, \mathsf{D}_2)]; (\mathsf{lc}_3, \mathsf{D}_3)]; [\{\sim p, h\} : (\mathsf{lc}_1, \mathsf{D}_1); (\mathsf{lc}_3, \mathsf{D}_3)]]$

$\mathcal{E}_c = [\{f, e\} : [\{h, \sim p\} : [\{d\} : (\mathsf{lc}_1, \mathsf{D}_1); (\mathsf{lc}_2, \mathsf{D}_2)]; [\{\sim p, \sim e\} : (\mathsf{lc}_3, \mathsf{D}_3); (\mathsf{lc}_2, \mathsf{D}_2)]]; [\{\sim p, h\} : [\{e, \sim p, \sim f\} : (\mathsf{lc}_3, \mathsf{D}_3); (\mathsf{lc}_2, \mathsf{D}_2)]; [\{d\} : (\mathsf{lc}_1, \mathsf{D}_1); (\mathsf{lc}_2, \mathsf{D}_2)]]]$

La Figura 6.4 muestra las representaciones de árbol $\mathsf{T}_{\mathcal{E}_a}$, $\mathsf{T}_{\mathcal{E}_b}$, y $\mathsf{T}_{\mathcal{E}_c}$ asociadas a las exps-conds $\mathcal{E}_a$, $\mathcal{E}_b$, y $\mathcal{E}_c$. Nótese, que $\mathsf{T}_{\mathcal{E}_a}$ es un árbol con solamente un nodo, y que en un mismo árbol, diferentes hojas pueden estar etiquetadas con la misma especificación.

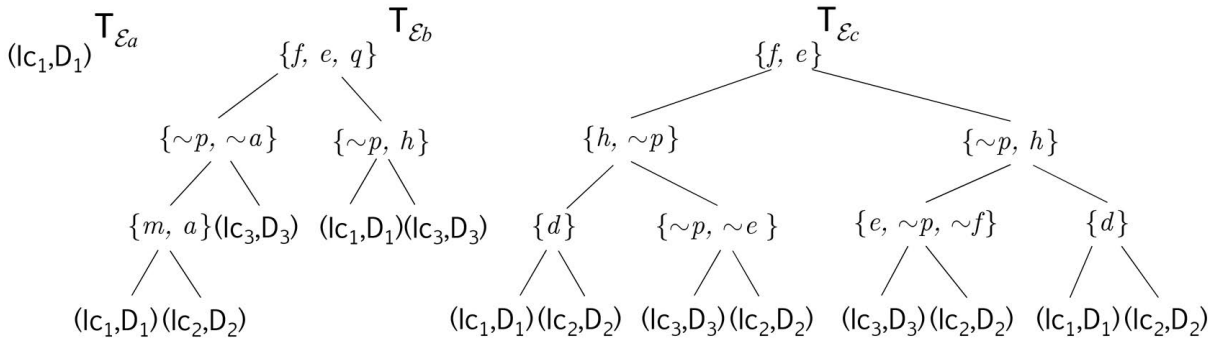

Figura 6.4: Representaciones de árbol $\mathsf{T}_{\mathcal{E}_a}$, $\mathsf{T}_{\mathcal{E}_b}$, y $\mathsf{T}_{\mathcal{E}_c}$.

Es interesante observar que todo camino desde la raíz a una hoja en un árbol $\mathsf{T}_{\mathcal{E}}$ representa una secuencia finita de guardas soportando la decisión que eligiera una *ECrit*. La siguiente definición caracteriza esta noción y será utilizada por las proposiciones que aparecen en el resto de la sección.

Definición 6.9 (Camino anotado / estructura de selección de criterio)**.** *Sea* $\mathsf{T}_{\mathcal{E}}$ *la representación de árbol para una exp-cond* $\mathcal{E}$*. Sea* (lc,D) *la etiqueta de una hoja* L *en* $\mathsf{T}_{\mathcal{E}}$ *y* $[\mathcal{G}_1,\mathcal{G}_2,\ldots,\mathcal{G}_n]$ $(n \geq 1)$*, la secuencia de etiquetas (guardas) de nodos internos que va desde la raíz de* $\mathsf{T}_{\mathcal{E}}$ *etiquetada con* $\mathcal{G}_1$ *a la hoja* L*. Un camino anotado para* L *en* $\mathsf{T}_{\mathcal{E}}$ *es una tupla denotada* $\langle \Gamma, (\mathsf{lc},\mathsf{D})\rangle_{\mathsf{T}_{\mathcal{E}}}$ *tal que* $\Gamma = [\mathcal{G}_1^{x_1}, \mathcal{G}_2^{x_2}, \ldots\ \mathcal{G}_n^{x_n}]$ $(n \geq 1)$*, y donde:*

1. *para cada* $\mathcal{G}_i$ $(1 \leq i \leq n-1)$ *la anotación* x_i *es "+" si* $\mathcal{G}_{i+1}$ *es un hijo izquierdo de* $\mathcal{G}_i$ *en* $\mathsf{T}_{\mathcal{E}}$*, o "−" en caso contrario, y*

2. *la anotación* x_n *es "+" si* (lc,D) *es el hijo izquierdo de* $\mathcal{G}_n$ *en* $\mathsf{T}_{\mathcal{E}}$*, o "−" en caso contrario.*

Se denominará estructura de selección de criterio (St) *para la especificación de criterio* (lc,D) *en* $\mathsf{T}_{\mathcal{E}}$*, extraída del camino anotado* $\langle \Gamma, (\mathsf{lc},\mathsf{D})\rangle_{\mathsf{T}_{\mathcal{E}}}$*, a la tupla* $(\Gamma^+, \Gamma^-, (\mathsf{lc},\mathsf{D}))_{\mathsf{T}_{\mathcal{E}}}$ *tal que:* $\Gamma^+ = \{\mathcal{G}_i |\ \mathcal{G}_i^+ \in \Gamma,\ 1 \leq i \leq n\}$*, y* $\Gamma^- = \{\mathcal{G}_j |\ \mathcal{G}_j^- \in \Gamma,\ 1 \leq j \leq n\}$*.*

Ejemplo 6.5. *Dada la representación de árbol* $\mathsf{T}_{\mathcal{E}_b}$ *en la Figura 6.4, la siguiente tabla muestra cada uno de los caminos anotados que se obtienen de* $\mathsf{T}_{\mathcal{E}_b}$*, junto con sus respectivas estructuras de selección de criterio:*

Camino anotado	**Estructura de selección de criterio**
$\langle[\{f,e,q\}^+,\{\sim p,\sim a\}^+,\{m,a\}^+],(\mathsf{Ic}_1,\mathsf{D}_1)\rangle_{\mathsf{T}_{\mathcal{E}_b}}$	$\mathsf{St}_1 = (\{\{f,e,q\},\{\sim p,\sim a\},\{m,a\}\},\{\},(\mathsf{Ic}_1,\mathsf{D}_1))_{\mathsf{T}_{\mathcal{E}_b}}$
$\langle[\{f,e,q\}^+,\{\sim p,\sim a\}^+,\{m,a\}^-],(\mathsf{Ic}_2,\mathsf{D}_2)\rangle_{\mathsf{T}_{\mathcal{E}_b}}$	$\mathsf{St}_2 = (\{\{f,e,q\},\{\sim p,\sim a\}\},\{\{m,a\}\},(\mathsf{Ic}_2,\mathsf{D}_2))_{\mathsf{T}_{\mathcal{E}_b}}$
$\langle[\{f,e,q\}^+,\{\sim p,\sim a\}^-],(\mathsf{Ic}_3,\mathsf{D}_3)\rangle_{\mathsf{T}_{\mathcal{E}_b}}$	$\mathsf{St}_3 = (\{\{f,e,q\}\},\{\{\sim p,\sim a\}\},(\mathsf{Ic}_3,\mathsf{D}_3))_{\mathsf{T}_{\mathcal{E}_b}}$
$\langle[\{f,e,q\}^-,\{\sim p,h\}^+],(\mathsf{Ic}_1,\mathsf{D}_1)\rangle_{\mathsf{T}_{\mathcal{E}_b}}$	$\mathsf{St}_4 = (\{\{\sim p,h\}\},\{\{f,e,q\}\},(\mathsf{Ic}_1,\mathsf{D}_1))_{\mathsf{T}_{\mathcal{E}_b}}$
$\langle[\{f,e,q\}^-,\{\sim p,h\}^-],(\mathsf{Ic}_3,\mathsf{D}_3)\rangle_{\mathsf{T}_{\mathcal{E}_b}}$	$\mathsf{St}_5 = (\{\},\{\{f,e,q\},\{\sim p,h\}\},(\mathsf{Ic}_3,\mathsf{D}_3))_{\mathsf{T}_{\mathcal{E}_b}}$

Dado un programa DeLP $\mathcal{P}$ y un camino anotado $\langle\Gamma,(\mathsf{Ic},\mathsf{D})\rangle_{\mathsf{T}_\mathcal{E}}$, una guarda $\mathcal{G}_i^+$ en la secuencia Γ se interpreta como *una guarda $\mathcal{G}_i$ que debería satisfacerse por $\mathcal{P}$ para poder aplicar la especificación de criterio* (Ic,D), mientras que $\mathcal{G}_j^-$, se interpreta como *una guarda $\mathcal{G}_j$ que no debería satisfacerse por $\mathcal{P}$*. Por lo tanto, si se considera la estructura de selección de criterio $(\Gamma^+,\Gamma^-,(\mathsf{Ic},\mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ asociada al camino $\langle\Gamma,(\mathsf{Ic},\mathsf{D})\rangle_{\mathsf{T}_\mathcal{E}}$, los elementos de Γ^+ se deberían satisfacer por $\mathcal{P}$ y los elementos de Γ^- no se deberían satisfacer por $\mathcal{P}$. Luego, cuando un programa y una estructura de selección de criterio cumplen con estas restricciones, se dice que la estructura de selección de criterio está en conformidad con respecto al programa.

Definición 6.10 (Conformidad)**.** *Dado un programa DeLP $\mathcal{P}$. Una estructura de selección de criterio $(\Gamma^+,\Gamma^-,(\mathsf{Ic},\mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ está en conformidad con respecto a $\mathcal{P}$ si se mantiene que:*

- *Para toda $\mathcal{G}\in\Gamma^+$, $\mathcal{G}$ se satisface para $\mathcal{P}$, y*
- *para toda $\mathcal{G}\in\Gamma^-$, $\mathcal{G}$ no se satisface para $\mathcal{P}$.*

Por ejemplo, la estructura St_4 del Ejemplo 6.5 está en conformidad con respeto al programa $\mathcal{P}_{6.1}$ del Ejemplo 6.1.

Observación 6.2. *Un árbol $\mathsf{T}_\mathcal{E}$ con un único nodo contiene una sola especificación (Ic,D); por lo tanto, este árbol tiene un único camino anotado $\langle\emptyset,(\mathsf{Ic},\mathsf{D})\rangle_{\mathsf{T}_\mathcal{E}}$ el cual tiene como su única estructura de selección de criterio a $(\emptyset,\emptyset,(\mathsf{Ic},\mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ y está en conformidad con respecto a cualquier programa DeLP $\mathcal{P}$. Por esta razón y para los propósitos de esta sección, de ahora en más el foco de atención estará en el estudio de árboles que contienen más de un nodo.*

La siguiente proposición muestra que para dos estructuras de selección de criterio arbitrarias que están en la misma representación de árbol, existe al menos una guarda $\mathcal{G}$

compartida por ambas estructuras. Por ejemplo, la estructuras St_1 y St_2 en el Ejemplo 6.5 comparten todas sus guardas, mientras que St_3 y St_4 comparten únicamente la guarda $\{f, e, q\}$.

Proposición 6.3. *Sea $\mathsf{T}_\mathcal{E}$ una representación de árbol. Dados dos caminos anotados $\langle [F_1^{x_1}, F_2^{x_2}, \ldots F_m^{x_m}], (\mathsf{lc}_i, \mathsf{D}_i)\rangle_{\mathsf{T}_\mathcal{E}}$ y $\langle [H_1^{x_1}, H_2^{x_2}, \ldots H_n^{x_n}], (\mathsf{lc}_j, \mathsf{D}_j)\rangle_{\mathsf{T}_\mathcal{E}}$ y sus respectivas estructuras de selección de criterios $({\Gamma_i}^+, {\Gamma_i}^-, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ y $({\Gamma_j}^+, {\Gamma_j}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$. Existe al menos una guarda $\mathcal{G}$ tanto en $({\Gamma_i}^+, {\Gamma_i}^-, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ como en $({\Gamma_j}^+, {\Gamma_j}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$ tal que $\mathcal{G} \in {\Gamma_i}^+ \cap {\Gamma_j}^-$ o $\mathcal{G} \in {\Gamma_j}^+ \cap {\Gamma_i}^-$.*

Demostración: Sea $\langle [F_1^{x_1}, F_2^{x_2}, \ldots F_m^{x_m}], (\mathsf{lc}_i, \mathsf{D}_i)\rangle_{\mathsf{T}_\mathcal{E}}$ y $\langle [H_1^{x_1}, H_2^{x_2}, \ldots H_n^{x_n}], (\mathsf{lc}_j, \mathsf{D}_j)\rangle_{\mathsf{T}_\mathcal{E}}$ dos caminos anotados diferentes. Es claro que ellos comparten un prefijo $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k)$, $k \leq m$ *and* $k \leq n$, tal que $F_i = H_i = \mathcal{G}_i (i \leq k)$. De hecho, si existe una única guarda $\mathcal{G}_i$ en el prefijo, esto es $i = k$ y $k = 1$, entonces el prefijo es la raíz etiquetada $\mathcal{G}_i$ del árbol $\mathsf{T}_\mathcal{E}$ y $\mathcal{G}_i \in {\Gamma_i}^+ \cap {\Gamma_j}^-$ o $\mathcal{G}_i \in {\Gamma_j}^+ \cap {\Gamma_i}^-$. Por lo contrario, si existe más de una guarda en el prefijo, entonces la última guarda $\mathcal{G}_k$ en $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_k)$ es tal que $\mathcal{G}_k \in {\Gamma_i}^+ \cap {\Gamma_j}^-$ o $\mathcal{G}_k \in {\Gamma_j}^+ \cap {\Gamma_i}^-$. □

Observación 6.3. *Para cualquier representación de árbol existen dos estructuras de selección de criterios $({\Gamma_i}^+, {\Gamma_i}^-, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ y $({\Gamma_j}^+, {\Gamma_j}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$ tal que ${\Gamma_i}^+ = \emptyset$ y ${\Gamma_j}^- = \emptyset$. Dado que $\mathsf{T}_\mathcal{E}$ es un árbol binario completo, entonces el camino más a la izquierda de la raíz de $\mathsf{T}_\mathcal{E}$ estará representado por la estructura $({\Gamma_i}^+, \emptyset, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ y el camino más a la derecha por $(\emptyset, {\Gamma_j}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$. En el Ejemplo 6.5, St_1 y St_5 son las estructuras de $\mathsf{T}_{\mathcal{E}_b}$(Figura 6.4) donde ${\Gamma_1}^- = \emptyset$ y ${\Gamma_5}^+ = \emptyset$. Note también que, dada el árbol $\mathsf{T}_{\mathcal{E}_a}$ de la Figura 6.4, la única estructura que se puede obtener es $(\emptyset, \emptyset, (\mathsf{lc}_1, \mathsf{D}_1))_{\mathsf{T}_{\mathcal{E}_a}}$, donde tanto $\Gamma^- = \emptyset$ y $\Gamma^+ = \emptyset$.*

La siguiente proposición establece que dada una representación de árbol cualquiera $\mathsf{T}_\mathcal{E}$ y un programa DeLP $\mathcal{P}$, no es posible encontrar dos estructuras de selección de criterio en $\mathsf{T}_\mathcal{E}$ tal que ambas estén en conformidad con respecto a $\mathcal{P}$. Como se mostrará más adelante, este resultado será utilizado para establecer que la *ECrit* seleccionada es la que se obtiene desde la estructura que está en conformidad con $\mathcal{P}$.

Proposición 6.4. *Dado un programa DeLP $\mathcal{P}$ y una exp-cond $\mathcal{E}$ y su representación de árbol $\mathsf{T}_\mathcal{E}$. Existe una única estructura de selección de criterio $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ en $\mathsf{T}_\mathcal{E}$ tal que $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ está en conformidad con respecto a $\mathcal{P}$.*

Demostración: Asumiendo que $(\Gamma_i{}^+, \Gamma_i{}^-, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ y $(\Gamma_j{}^+, \Gamma_j{}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$ son dos estructuras diferentes en el mismo árbol $\mathsf{T}_\mathcal{E}$. Por Proposición 6.3, existe al menos una guarda $\mathcal{G}$ en $(\Gamma_i{}^+, \Gamma_i{}^-, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ y $(\Gamma_j{}^+, \Gamma_j{}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$ tal que $\mathcal{G} \in \Gamma_i{}^+ \cap \Gamma_j{}^-$ o $\mathcal{G} \in \Gamma_j{}^+ \cap \Gamma_i{}^-$. Por lo tanto, una de las estructuras debería estar en conformidad con respecto a $\mathcal{P}$ con $\mathcal{G}$ que se satisface para $\mathcal{P}$, y la otra estructura con $\mathcal{G}$ que no se satisface para $\mathcal{P}$. Si embargo, por Definición 6.1 una guarda $\mathcal{G}$ se satisface o bien no se satisface para un programa DeLP $\mathcal{P}$ dado, entonces $(\Gamma_i{}^+, \Gamma_i{}^-, (\mathsf{lc}_i, \mathsf{D}_i))_{\mathsf{T}_\mathcal{E}}$ y $(\Gamma_j{}^+, \Gamma_j{}^-, (\mathsf{lc}_j, \mathsf{D}_j))_{\mathsf{T}_\mathcal{E}}$ nunca pueden estar en conformidad con respecto al mismo programa DeLP $\mathcal{P}$. □

Considere de nuevo el programa DeLP $\mathcal{P}_{6.1}$ del Ejemplo 6.1 y los árboles $\mathsf{T}_{\mathcal{E}_b}$ y $\mathsf{T}_{\mathcal{E}_c}$ de la Figura 6.4. La estructura $(\{\{\sim p, h\}\}, \{\{f, e, q\}\}, (\mathsf{lc}_1, \mathsf{D}_1))_{\mathsf{T}_{\mathcal{E}_b}}$ de $\mathsf{T}_{\mathcal{E}_b}$ es la única estructura en conformidad con respecto a $\mathcal{P}_{6.1}$ y la estructura $(\{\{f, e\}, \{h, \sim p\}, \{d\}\}, \{\}, (\mathsf{lc}_1, \mathsf{D}_1))_{\mathsf{T}_{\mathcal{E}_c}}$ de $\mathsf{T}_{\mathcal{E}_c}$ es la única que está en conformidad con respecto a $\mathcal{P}_{6.1}$.

Cuando se analiza si una estructura $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ está en conformidad con respecto a un programa DeLP, en las guardas del conjunto Γ^+ podrían aparecer literales repetidos. Por ejemplo, analizando la estructura $(\{\{\sim p, h\}, \{e, \sim p, \sim f\}\}, \{\{f, e\}\}, (\mathsf{lc}_3, \mathsf{D}_3))_{\mathsf{T}_{\mathcal{E}_c}}$ de la representación de árbol $\mathsf{T}_{\mathcal{E}_c}$ en la Figura 6.4, el literal $\sim p$ está repetido. En este caso, esta estructura está en conformidad con respecto a un programa DeLP si $\{\sim p, h\}$ y $\{e, \sim p, \sim f\}$ se satisfacen por un programa DeLP dado, esto es, existe una derivación estricta para cada literal contenido en estás guardas. Por lo tanto, en este caso en particular, se tendrá que analizar dos veces si existe una derivación estricta para $\sim p$, llevando a un cómputo redundante. En la siguiente definición se caracteriza qué caminos sufren de este tipo de redundancia.

Definición 6.11 (Redundancia)**.** *Sea* $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ *una estructura de selección de criterio. Se dice que* $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ *es redundante si y solo si* $\bigcap_{\mathcal{G}_i \in \Gamma^+} \mathcal{G}_i \neq \emptyset$.

Nótese que, este problema de redundancia puede ser resuelto utilizando la siguiente transformación. Sea $\langle \Gamma, (\mathsf{lc}, \mathsf{D}) \rangle_{\mathsf{T}_\mathcal{E}}$ el camino anotado asociado a la estructura $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ donde $\Gamma = [\mathcal{G}_1^{x_1}, \ldots, \mathcal{G}_i^{x_i}, \ldots, \mathcal{G}_n^{x_n}] (n \geq 0 \text{ y } i \leq n)$. Se puede construir un nuevo camino $\langle \Gamma', (\mathsf{lc}, \mathsf{D}) \rangle_{\mathsf{T}_\mathcal{E}}$ desde $\langle \Gamma, (\mathsf{lc}, \mathsf{D}) \rangle_{\mathsf{T}_\mathcal{E}}$ tal que no existen literales repetidos en las guardas $\mathcal{G}_i'^+ \in \Gamma'$, esto es

$$\mathcal{G}_i'^+ = \mathcal{G}_i'^+ \setminus \bigcap_{\mathcal{G}_j'^+ \in \Gamma', j \leq i} \mathcal{G}_j'^+.$$

Entonces, la estructura de selección de criterio $(\Gamma'^+, \Gamma'^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$, obtenida a partir de $\langle \Gamma', (\mathsf{lc}, \mathsf{D}) \rangle_{\mathsf{T}_\mathcal{E}}$, no es redundante. Por ejemplo, como se mostró anteriormente,

la estructura $\langle\{\{\sim p, h\}, \{e, \sim p, \sim f\}\}, \{\{f, e\}\}, (\mathsf{Ic}_3, \mathsf{D}_3)\rangle_{\mathsf{T}_{\mathcal{E}_c}}$ es redundante. Por lo tanto, después de la transformación recién descripta, se obtiene una nueva estructura $\langle\{\{\sim p, h\}, \{e, \sim f\}\}, \{\{f, e\}\}, (\mathsf{Ic}_3, \mathsf{D}_3)\rangle_{\mathsf{T}_{\mathcal{E}_c}}$ sin literales repetidos. Para toda estructura $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ en un árbol $\mathsf{T}_\mathcal{E}$ se asume que $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ es no redundante.

Continuando con el análisis de situaciones particulares que puedan surgir en una *exp-cond*, las Proposiciones 6.5 y 6.6 identificarán características que hacen que una estructura $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ no esté en conformidad con respecto a cualquier programa DeLP dado.

Proposición 6.5. *Dada una exp-cond $\mathcal{E}$, su representación de árbol $\mathsf{T}_\mathcal{E}$, y un programa DeLP $\mathcal{P}$, sea $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ una estructura de selección de criterio. Si $\Gamma^+ \cap \Gamma^- \neq \emptyset$, entonces $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ no está en conformidad con respecto a $\mathcal{P}$.*

Demostración: Suponiendo que $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ está en conformidad con respecto a $\mathcal{P}$ y $\Gamma^+ \cap \Gamma^- \neq \emptyset$. Por Definición 6.10 las guardas del conjunto Γ^+ se deberían satisfacer por $\mathcal{P}$, y las guardas de Γ^- no se deberían satisfacer por $\mathcal{P}$. Sin embargo, si $\Gamma^+ \cap \Gamma^- \neq \emptyset$, existe una guarda $\mathcal{G}$ tal que $\mathcal{G} \in \Gamma^+ \cap \Gamma^-$. Por lo tanto, $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ no estaría en conformidad con respecto a $\mathcal{P}$. □

Nótese que una guarda $\mathcal{G}$ puede contener literales contradictorios. Un caso interesante para analizar es cuando en el conjunto Γ^+ de una estructura $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ aparecen literales contradictorios. La siguiente proposición muestra que si en Γ^+ existen literales contradictorios, entonces al menos una guarda en Γ^+ no será satisfacible para algún programa DeLP $\mathcal{P}$ dado. Por lo tanto, se dirá que $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ no está en conformidad con respecto a $\mathcal{P}$.

Proposición 6.6. *Dada una estructura de selección de criterio $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ y un programa DeLP $\mathcal{P} = (\Pi, \Delta)$. Si el conjunto resultante $S = \bigcup_{\mathcal{G} \in \Gamma^+} \mathcal{G}$ es contradictorio, entonces $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ no está en conformidad con respecto a $\mathcal{P}$.*

Demostración: Asumiendo que S es un conjunto contradictorio, entonces, a partir de $\mathcal{P}$ existen derivaciones para dos literales complementarios $L, \sim L \in S$. Luego Π es contradictorio y $\mathcal{P}$ no sería un programa DeLP válido. □

Por ejemplo, sea el camino anotado $\langle[\{f, e\}^+, \{h, \sim p\}^-, \{\sim p, \sim e\}^+], (\mathsf{Ic}_3, \mathsf{D}_3)\rangle_{\mathsf{T}_{\mathcal{E}_c}}$ de $\mathsf{T}_{\mathcal{E}_c}$ y su estructura de selección de criterio $(\{\{f, e\}, \{\sim p, \sim e\}\}, \{\{h, \sim p\}\}, (\mathsf{Ic}_3, \mathsf{D}_3))_{\mathsf{T}_{\mathcal{E}_c}}$. Nótese que, no existe ningún programa DeLP que sea capaz de satisfacer tanto $\{f, e\}$ y $\{\sim p, \sim e\}$,

ya que ningún programa válido puede tener derivaciones estrictas para e y $\sim e$ al mismo tiempo.

Aquellas estructuras que no sufren de los problemas caracterizados en las Proposiciones 6.5 y 6.6 se consideran como estructuras de selección de criterio seguras. Esta noción se formaliza en la siguiente definición.

Definición 6.12 (Estructura de Selección de Criterio/Representación de Árbol Segura)**.** *Sea* $\mathsf{T}_{\mathcal{E}}$ *una representación de árbol y* $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_{\mathcal{E}}}$ *una estructura de selección de criterio. La estructura* $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_{\mathcal{E}}}$ *es una estructura de selección de criterio segura si y solo si:*

i) Γ^+ *y* Γ^- *son conjuntos disjuntos.*

ii) Γ^+ *es un conjunto no contradictorio.*

Una representación de árbol segura $\mathsf{T}_{\mathcal{E}}$ *contiene únicamente estructuras de selección de criterio seguras.*

Definición 6.13. *Sea* $\mathcal{E}$ *una exp-cond y* $\mathsf{T}_{\mathcal{E}}$ *su representación de árbol correspondiente. Se dice que la expresión* $\mathcal{E}$ *es válida si y solo si su representación de árbol* $\mathsf{T}_{\mathcal{E}}$ *es segura.*

Observe que en la Figura 6.4 no todas las representaciones de árbol son seguras. Por ejemplo, $\mathsf{T}_{\mathcal{E}_c}$ no es una representación segura ya que, como se discutió previamente, en la estructura $(\{\{f, e\}, \{\sim p, \sim e\}\}, \{\{\sim p, h\}\}, (\mathsf{lc}_3, \mathsf{D}_3))_{\mathsf{T}_{\mathcal{E}_c}}$, el conjunto $\{f, e\} \cup \{\sim p, \sim e\}$ es contradictorio. Sin embargo, tanto $\mathsf{T}_{\mathcal{E}_1}$ como $\mathsf{T}_{\mathcal{E}_2}$ en la Figura 6.3 son representaciones de árbol seguras.

Proposición 6.7. *Dada una exp-cond válida* $\mathcal{E}$ *y su representación de árbol* $\mathsf{T}_{\mathcal{E}}$*. Para toda estructura de selección de criterio* $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_{\mathcal{E}}}$*, existe un programa DeLP* $\mathcal{P}$ *tal que* $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_{\mathcal{E}}}$ *está en conformidad con respecto a* $\mathcal{P}$*.*

Demostración: Dado que la representación de árbol $\mathsf{T}_{\mathcal{E}}$ es segura y $(\Gamma^+, \Gamma^-, (\mathsf{lc}, \mathsf{D}))_{\mathsf{T}_{\mathcal{E}}}$ es una estructura arbitraria en $\mathsf{T}_{\mathcal{E}}$, entonces $\bigcup_{\mathcal{G}_i \in \Gamma^+} \mathcal{G}_i$ es un programa DeLP que satisface la condición del enunciado. □

Dada la representación de árbol $\mathsf{T}_{\mathcal{E}}$ asociada a la *exp-cond* $\mathcal{E}$, diferentes secuencias de guardas pueden posiblemente tender a diferentes *ECrits*. No obstante, a partir de un programa DeLP $\mathcal{P}$ el resultado de la evaluación obtenida desde $\mathsf{evalE}(\mathcal{E}, \mathcal{P})$ refleja el hecho que existe un solo camino desde la raíz del árbol $\mathsf{T}_{\mathcal{E}}$ a la *ECrit* que resulta de tal proceso de

evaluación. Por ejemplo, considerando la expresión $\mathcal{E}_2$ del Ejemplo 6.2 y el programa $\mathcal{P}_h$ del Ejemplo de Aplicación 4.5 del Capítulo 4, la respuesta de la función $\mathsf{evalE}(\mathcal{E}_2, \mathcal{P}_h)$ es la especificación de criterio $\mathcal{EC}_{\mathsf{seg}}$ debido al hecho que $zRobos(h1)$, $cVehicular$, y $estrellas(h1, 5)$ no tienen derivaciones estrictas. El siguiente lema muestra que siempre es posible obtener una *ECrit* desde una representación de árbol segura.

Lema 6.1. *Dado un programa DeLP $\mathcal{P}$, una exp-cond $\mathcal{E}$ con su representación de árbol $\mathsf{T}_\mathcal{E}$, y $\mathbb{N}$ el conjunto de especificaciones de criterios involucrados en $\mathcal{E}$, tal que $(\mathsf{Ic}, \mathsf{D}) \in \mathbb{N}$. Se cumple que $\mathsf{evalE}(\mathcal{E}, \mathcal{P}) = (\mathsf{Ic}, \mathsf{D})$ si y solo si $(\mathsf{Ic}, \mathsf{D})$ aparece en una estructura de selección de criterio $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ tal que está en conformidad con respecto a $\mathcal{P}$.*

Demostración: Si $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ está en conformidad con respecto a $\mathcal{P}$, entonces por Proposición 6.4, la estructura obtenida a partir de la representación de árbol $\mathsf{T}_\mathcal{E}$ utilizando el programa $\mathcal{P}$ será $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$. Por Proposición 6.2, $\mathsf{T}_\mathcal{E}$ es el árbol asociado a la expresión $\mathcal{E}$, entonces $(\mathsf{Ic}, \mathsf{D})$ debería ser la especificación de criterio que resulta de la evaluación obtenida desde $\mathsf{evalE}(\mathcal{E}, \mathcal{P})$.

Si $\mathsf{evalE}(\mathcal{E}, \mathcal{P}) = (\mathsf{Ic}, \mathsf{D})$, entonces por Definición 6.3, $\mathcal{E} = (\mathsf{Ic}, \mathsf{D})$ o $\mathcal{E} = [\mathcal{G} : \mathcal{E}_i; \mathcal{E}_j]$. Si $\mathcal{E} = (\mathsf{Ic}, \mathsf{D})$, por Definición 6.8 existe una representación de árbol para $\mathcal{E}$ con un único nodo etiquetado con $(\mathsf{Ic}, \mathsf{D})$. Luego, por Observación 6.2, este árbol tiene un único camino anotado $\langle \emptyset, (\mathsf{Ic}, \mathsf{D}) \rangle_{\mathsf{T}_\mathcal{E}}$ con la única estructura de selección de criterio respectivamente asociada $(\emptyset, \emptyset, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$. Si $\mathcal{E} = [\mathcal{G} : \mathcal{E}_i; \mathcal{E}_j]$, entonces por Definición 6.8, $\mathcal{E}$ tiene asociada una representación de árbol $\mathsf{T}_\mathcal{E}$ tal que por la Proposición 6.4 existe una estructura de selección de criterio $(\Gamma^+, \Gamma^-, (\mathsf{Ic}, \mathsf{D}))_{\mathsf{T}_\mathcal{E}}$ en $\mathsf{T}_\mathcal{E}$ en conformidad con respecto a $\mathcal{P}$. □

Varias conclusiones se pueden extraer de los resultados obtenidos en esta sección. En términos generales, se puede enfatizar el hecho que una representación de árbol no solo ayuda a entender la semántica de evaluación de las expresiones de preferencias condicionales, sino que también permite que se discutan varios aspectos relacionados a la selección de un criterio. Por ejemplo, es fácil ver que no sería una decisión coherente si un usuario especifica las guardas de manera tal que para seleccionar una especificación se requiere inferir literales contradictorios a partir de la parte estricta de un programa. De hecho, un punto importante a mencionar es que la flexibilidad del enfoque propuesto en este capítulo respecto a cómo las *exps-conds* son construidas por el usuario, permite la posibilidad de construir expresiones que pueden ser incoherentes o contradictorias. Situaciones de este tipo fueron detalladas en las Proposiciones 6.5 y 6.6. En este sentido, para tratar con estos posibles problemas y forzar la construcción de expresiones que satisfacen coherencia

interna se requiere que las *exps-conds* incluidas en las consultas basadas en preferencias condicionales sean válidas.

6.3. Ejemplo de aplicación en un entorno con robots

A continuación, se muestra cómo el modelo de consulta propuesto en este capítulo puede ser aplicado en un escenario con robots que fue descripto en [FEGS08]. En particular, el dominio de aplicación consiste de un micro escenario del mundo real que utiliza robots para realizar tareas de limpieza. El entorno consiste de cajas dispersas en un escenario donde un robot deberá obtener una recomendación sobre cual será la próxima caja más conveniente a seleccionar con el objetivo de llevarla a un lugar en particular denominado *depósito*. Para seleccionar cajas, un robot podrá optar por alguna de las siguientes estrategias: seleccionar la caja más chica, o la más cercana a sí mismo, o la caja que está más cerca al depósito. En las Figuras 6.5-(a) y 6.5-(b) se presentan dos escenarios, los cuales serán utilizados a lo largo de este ejemplo.

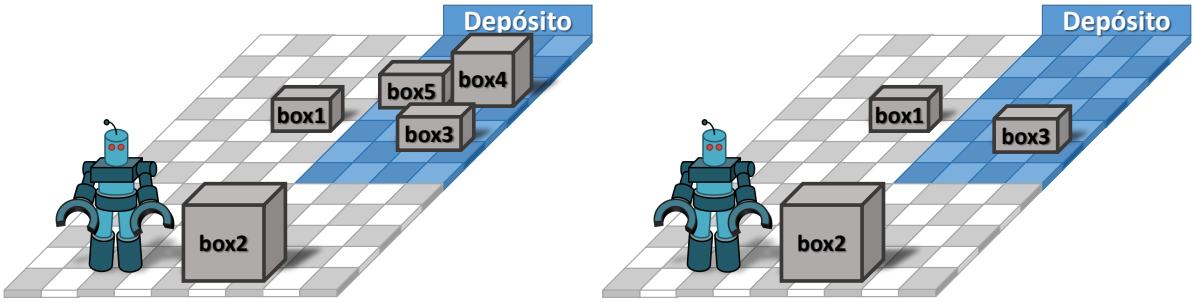

Figura 6.5: Escenarios del Robot.

En la Sección 4.4.3 se presenta el criterio de preferencia $\succsim_{\mathsf{prioL}}$ basada en prioridad entre literales (Definición 4.26); este criterio utiliza un orden parcial estricto (denotado $>$) sobre algunos literales distinguidos que son utilizados en los argumentos: $L > L'$ significa que el literal L es preferido al literal L'. Utilizando $\succsim_{\mathsf{prioL}}$ un argumento $\langle \mathcal{A}_1, L_1 \rangle$ será preferido a un argumento $\langle \mathcal{A}, L \rangle$ con respecto a un orden $>$, si y solo si existen dos literales $L_1 \in^* \mathcal{A}_1$ y $L_2 \in^* \mathcal{A}$ tal que, $L_1 > L_2$, y no existen los literales L_3 y L_4 tal que $L_3 \in^* \mathcal{A}_1$, $L_4 \in^* \mathcal{A}$, y $L_4 > L_3$. Note que $L \in^* \mathcal{A}'$ significa que existe una regla rebatible $(q_0 \prec q_1, q_2, \ldots, q_n)$ en $\mathcal{A}'$ y $L = q_i$ $(1 \leq i \leq n)$. Asimismo, en el ejemplo de aplicación

de la Sección 5.6 se introdujo la implementación de criterio $\mathsf{impC}(\mathsf{prioL})$ cuyos átomos distinguidos se expresan mediante el predicado *es_mejorL* el cual denota una prioridad entre dos literales. En el ejemplo presentado en esta sección se utilizará un *SRPCond* definido como $\langle \mathsf{evalE}_r, \mathbb{I}_r, \langle \{\mathsf{impC}(\mathsf{prioL})\}, \mathsf{compPref}_r \rangle \rangle$.

Considere el conjunto de literales $\{mas\_chica(.;.), cerca\_robot(.;.), cerca\_deposito(.;.)\}$ para expresar las estrategias de selección del robot para elegir la caja más chica, o la caja más cercana a éste, o la más cercana al depósito respectivamente. Observe que si se consideran diferentes órdenes de preferencias (prioridades) sobre este conjunto de literales se podrán definir diferentes *ECrits* para la implementación $\mathsf{impC}(\mathsf{prioL})$ disponible en el *SRPCond* definido anteriormente. A continuación se establecerán tres órdenes diferentes utilizando la notación de átomos distinguidos para la implementación $\mathsf{impC}(\mathsf{prioL})$ detallada arriba;

$$\mathsf{D}_{\mathsf{masChica}} = \left\{ \begin{array}{l} (es\_mejorL(mas\_chica(Z,W), cerca\_robot(W,Y))), \\ (es\_mejorL(mas\_chica(Z,Y), cerca\_robot(W,Z))), \\ (es\_mejorL(mas\_chica(Z,W), cerca\_deposito(W,Y))), \\ (es\_mejorL(mas\_chica(Z,Y), cerca\_deposito(W,Z))), \\ (es\_mejorL(cerca\_robot(Z,W), cerca\_deposito(W,Y))), \\ (es\_mejorL(cerca\_robot(Z,Y), cerca\_deposito(Y,Z))) \end{array} \right\}$$

$$\mathsf{D}_{\mathsf{cercaRobot}} = \left\{ \begin{array}{l} (es\_mejorL(cerca\_robot(Z,W), cerca\_deposito(W,Y))), \\ (es\_mejorL(cerca\_robot(Z,Y), cerca\_deposito(W,Z))), \\ (es\_mejorL(cerca\_robot(Z,W), mas\_chica(W,Y))), \\ (es\_mejorL(cerca\_robot(Z,Y), mas\_chica(W,Z))), \\ (es\_mejorL(cerca\_deposito(Z,W), mas\_chica(W,Y))), \\ (es\_mejorL(cerca\_deposito(Z,Y), mas\_chica(W,Z))) \end{array} \right\}$$

$$\mathsf{D}_{\mathsf{cercaDeposito}} = \left\{ \begin{array}{l} (es\_mejorL(cerca\_depostio(Z,W), cerca\_deposito(W,Y))), \\ (es\_mejorL(cerca\_deposito(Z,Y), cerca\_deposito(W,Z))), \\ (es\_mejorL(cerca\_deposito(Z,W), mas\_chica(W,Y))), \\ (es\_mejorL(cerca\_deposito(Z,Y), mas\_chica(W,Z))), \\ (es\_mejorL(cerca\_robot(Z,W), mas\_chica(W,Y))), \\ (es\_mejorL(cerca\_robot(Z,Y), mas\_chica(W,Z))) \end{array} \right\}$$

En este ejemplo de aplicación se consideran, a partir de estas preferencias, las siguientes tres *ECrits*.

- $\mathcal{EC}_{\mathsf{masChica}} = (\mathsf{impC}(\mathsf{prioL}), \mathsf{D}_{\mathsf{masChica}})$, expresa que *"el robot preferirá primero las cajas más chicas, luego las más cercanas a él, y por último las cajas mas cercanas al deposito"*.
- $\mathcal{EC}_{\mathsf{cercaRobot}} = (\mathsf{impC}(\mathsf{prioL}), \mathsf{D}_{\mathsf{cercaRobot}})$, declara que *"el robot preferirá primero las cajas cercanas a él, luego las cajas cercanas al depósito, y en último lugar las cajas más chicas"*.
- $\mathcal{EC}_{\mathsf{cercaDeposito}} = (\mathsf{impC}(\mathsf{prioL}), \mathsf{D}_{\mathsf{cercaDeposito}})$, representa que *"el robot preferirá primero las cajas cercanas al depósito, luego las cercanas a él, y por último las cajas más chicas"*.

Para recomendar la próxima caja que el robot debería mover, se utilizará el siguiente programa DeLP $\mathcal{P}_r = (\Pi_r, \Delta_r)$, donde.

$$\Pi_r = \left\{ \begin{array}{l} deposito\_lleno \leftarrow cajas\_depositadas(Num), Num \geq 3 \\ cajas\_dispersas \leftarrow cajas\_disponibles(Num), Num \geq 5 \end{array} \right\}$$

$$\Delta_r = \left\{ \begin{array}{l} recomend(Box) \prec mejor(Box, Obox) \\ mejor(Box, Obox) \prec cerca\_robot(Box, Obox) \\ mejor(Box, Obox) \prec cerca\_deposito(Box, Obox) \\ mejor(Box, Obox) \prec mas\_chica(Box, Obox) \\ \sim mejor(Box, Obox) \prec cerca\_robot(Obox, Box) \\ \sim mejor(Box, Obox) \prec cerca\_deposito(Obox, Box) \\ \sim mejor(Box, Obox) \prec mas\_chica(Obox, Box) \end{array} \right\}$$

En [FEGS08], se considera un orden sobre literales fijo para un entorno en particular; no obstante, acá se propone utilizar una *exp-cond* para programar cómo seleccionar dinámicamente la *ECrit* más adecuada dependiendo de las cajas que se encuentran en el entorno en un momento determinado. Para esto se presenta una *exp-cond* que implementa las siguientes intuiciones: "*si el depósito está completo con cajas entonces uso la especificación que prioriza cajas chicas, sino si hay varias cajas disponibles uso la especificación que prioriza cajas cercas del depósito, en caso contrario uso la especificación que prioriza cajas cercanas al robot*". Esta intuición puede ser capturada con la *exp-cond* $\mathcal{E}$ que se incluye a continuación. Observe que los literales *deposito_lleno* y *cajas_dispersas* se pueden derivar utilizando reglas estrictas a partir de Π_r y ambos se basan en información que depende del escenario particular donde el robot está involucrado.

$$\mathcal{E} = [\{deposito\_lleno\} : \mathcal{EC}_{\mathsf{masChica}}, [\{cajas\_dispersas\} : \mathcal{EC}_{\mathsf{cercaDeposito}}; \mathcal{EC}_{\mathsf{cercaRobot}}]]$$

Considerando el escenario representado en Figure 6.5(a) donde existen tres cajas en el depósito y dos cajas disponibles para seleccionar: $box1$ y $box2$. La siguiente *CPCond* puede

ser utilizada para preguntar por una recomendación que considere el escenario actual y la *exp-cond* $\mathcal{E}$ definida arriba:

$$\mathcal{CPC}_{(a)} = \langle \mathcal{E}, \mathcal{P}_{r(a)}, recomend(X) \rangle$$

Para responder a $\mathcal{CPC}_{(a)}$ el *SRPCond* considerará el programa $\mathcal{P}_{r(a)} = \mathcal{P}_r \cup \{cajas\_disponibles(2),\ cajas\_depositadas(3),\ cerca\_robot(box2, box1),\ cerca\_deposito(box1, box2),\ mas\_chica(box1, box2)\}$ que resulta de agregar a $\mathcal{P}_r$ la información sobre el entorno percibida por el robot: existen tres cajas en el depósito, dos cajas disponibles ($box1$ y $box2$), $box2$ está más cerca del robot que $box1$, $box1$ está más cerca del depósito que $box2$, y $box1$ es más chica que $box2$. Observe que *deposito_lleno* se puede derivar estrictamente desde $\mathcal{P}_{r(a)}$ utilizando una regla estricta de Π_r y el literal $cajas\_depositadas(3)$. Por lo tanto la evaluación de la *exp-cond* $\mathcal{E}$, a partir de $\mathcal{P}_{r(a)}$, resulta en la selección de la especificación de criterio $\mathcal{EC}_{\mathsf{masChica}}$, la cual prefiere primero levantar cajas pequeñas.

A continuación, en la Figura 6.6 se presentan los árboles de dialéctica construidos desde el programa $\mathcal{P}_{r(a)}$ para la consulta $\mathcal{CPC}_{(a)}$ considerando la especificación de criterio $\mathcal{EC}_{\mathsf{masChica}}$. Observe que existen tres árboles, el primero y el segundo árbol corresponden a argumentos para recomendar $box1$ (es decir $X = box1$), y el tercer árbol a un argumento para recomendar $box2$ (es decir $X = box2$). Como existe al menos un árbol de dialéctica para $recomend(box1)$ con el nodo raíz etiquetado con U (sin derrotar), entonces la respuesta a $\mathcal{CPC}_{(a)}$ será SI, con $X = box1$.

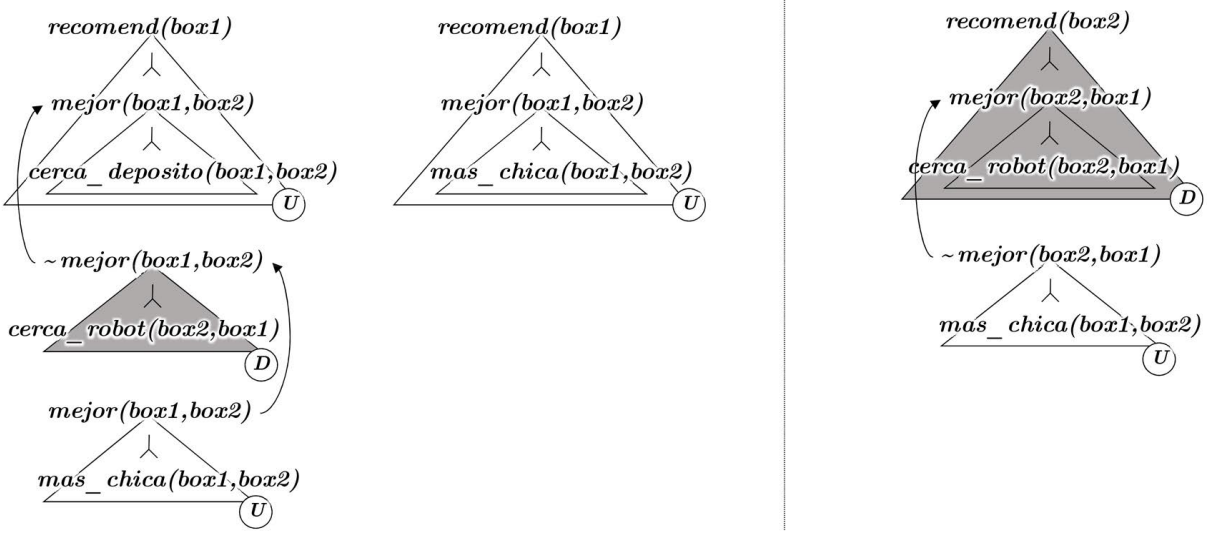

Figura 6.6: Árboles de dialéctica construidos para responder $\mathcal{CPC}_{(a)}$.

Por lo tanto, en este escenario, debido a la *ECrit* seleccionada $\mathcal{EC}_{\mathsf{masChica}}$, el servicio garantiza la recomendación para seleccionar primero $box1$. En lo que sigue, se muestra que en un escenario diferente, como el descripto en Figura 6.5-(b), la evaluación de la *exp-cond* $\mathcal{E}$ retorna una *ECrit* diferente.

Considerando ahora el escenario representado en la Figura 6.5-(b) donde existe una caja en el depósito y dos cajas disponibles para elegir: $box1$ y $box2$. La consulta basada en preferencias condicionales $\mathcal{CPC}_{(b)}$ que se muestra a continuación puede ser utilizada para preguntar al *SRPCond* por una recomendación que considere este escenario y la *exp-cond* $\mathcal{E}$ definida arriba.

$$\mathcal{CPC}_{(b)} = \langle \mathcal{E}, \mathcal{P}_{r(b)}, recomend(X) \rangle$$

Note que la única diferencia entre esta consulta y la anterior, es el programa utilizado. En $\mathcal{CPC}_{(b)}$, el programa será $\mathcal{P}_{r(b)} = \mathcal{P}_r \cup \{cajas\_disponibles(2), cajas\_depositadas(1), cerca\_robot(box2, box1), cerca\_deposito(box1, box2), mas\_chica(box1, box2)\}$. En este caso, luego de la evaluación de $\mathcal{E}$, se da que ni *deposito_lleno*, ni *cajas_dispersas* tienen derivaciones estrictas, por lo tanto, la *ECrit* seleccionada es $\mathcal{EC}_{\mathsf{cercaRobot}}$. Para responder la consulta, a partir de $\mathcal{P}_{r(b)}$ se construirán los árboles de dialéctica expresados en la Figura 6.7.

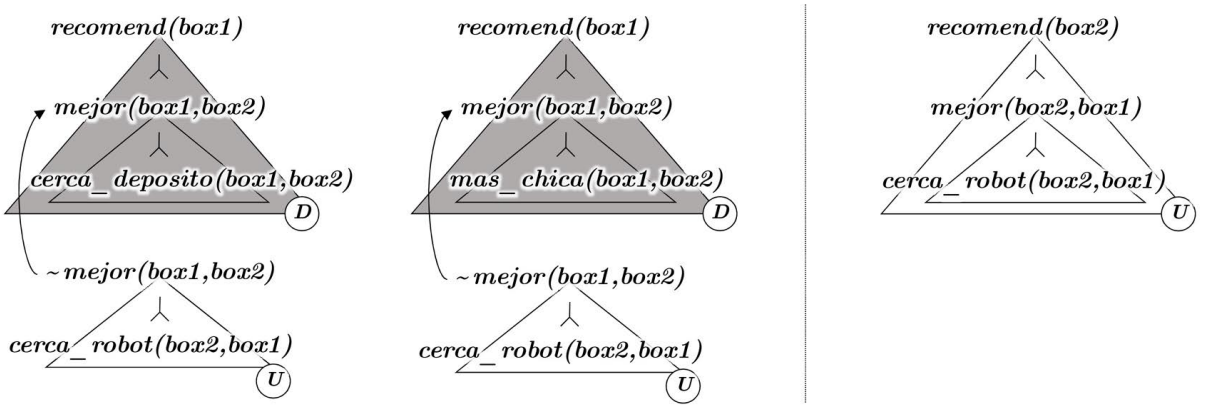

Figura 6.7: Árboles de dialéctica construidos para responder $\mathcal{CPC}_{(b)}$.

En este caso en particular, como existe un árbol de dialéctica para $recomend(box2)$ que tiene el nodo raíz etiquetado con U (sin derrotar), entonces la respuesta a $\mathcal{CPC}_{(b)}$ será SI, con $X = box2$. Entonces, en este escenario, debido a la *ECrit* seleccionada $\mathcal{EC}_{\mathsf{cercaRobot}}$, el *SRPCond* garantiza la recomendación para seleccionar primero $box2$.

6.4. Conclusión

En este capítulo se introdujo un mecanismo programable para seleccionar criterios de preferencias, y la formalización de la semántica para la interacción de tal mecanismo con los elementos que conforman un servicio de razonamiento rebatible. En particular, la consulta basada en preferencias condicionales (*CPCond*) y el servicio de razonamiento rebatible basado en preferencias condicionales (*SRPCond*), presentados en la Sección 6.1, y las formalizaciones introducidas en la Sección 6.2 son los principales aportes de este capítulo.

El formalismo presentado permite a través de una expresión de preferencia condicional seleccionar una *ECrit*. Esta expresión estará incluida en las *CPConds* que puede recibir un *SRPCond*, y la selección de una especificación dependerá de la información que el servicio utiliza para responder tal consulta. De esta manera, mediante una *CPCond* el usuario puede guiar el proceso de razonamiento acorde a sus preferencias o necesidades. En este sentido, las *exps-conds* proveen información acerca del conocimiento que será priorizado en el proceso de razonamiento, clarificando de este modo el mecanismo de cómputo que resuelve una consulta. De hecho, las respuestas que da un *SRPCond* serán más confiables ya que se puede entender *por qué* una razón (argumento) es preferida sobre otra. En consecuencia, el mecanismo propuesto en la Sección 6.1 contribuye a la confianza de los usuarios sobre las respuestas que reciben de sistemas que utilizan los servicios de razonamiento propuestos en este capítulo.

En la Sección 6.2 se introdujo una representación de árbol para las expresiones de preferencias condicionales que proporciona una forma clara para analizar varias propiedades de tales expresiones. Estas propiedades son útiles para identificar cuándo una expresión puede ser optimizada para evitar la computación de inferencias redundantes y caracterizar cuándo ciertos caminos en la expresión no son transitables. También, al utilizar estos resultados se ha caracterizado si una expresión es segura, es decir, expresiones donde todos los caminos que llevan a una especificación pueden ser transitados. Estas propiedades son de especial interés en esta tesis por que permiten construir expresiones válidas, es decir, expresiones que mantienen relaciones coherentes entre las guardas que justifican la elección de un criterio en particular.

Por último, en este capítulo se continuó con los ejemplos de aplicación introducidos en el capítulo anterior, mostrando la utilidad de la *CPCond* en los dominios de aplicación elegidos. Además, se presentó un nuevo dominio de aplicación en donde un robot

debe elegir que caja levantar para llevar a un depósito. Para esto, se mostró que con las *exps-conds* es posible cambiar de *ECrit* de una manera automática dependiendo de la información que el robot percibe del entorno donde se encuentra.

Capítulo 7

Servicio de Razonamiento Rebatible basado en Preferencias Combinadas

En el modelo desarrollado en el Capítulo 5 el criterio a ser utilizado por un servicio de razonamiento rebatible basado en preferencias (*SRPref*) se especifica directamente en las consultas basadas en preferencias (*CPrefs*). Por otra parte, en la propuesta del Capítulo 6 el criterio utilizado por un servicio de razonamiento rebatible basado en preferencias condicionales (*SRPCond*) se obtiene como resultado de la evaluación de una expresión condicional que es parte de las consultas basadas en preferencias condicionales (*CPConds*) que recibe. En este capítulo se introducirán las *Consultas basadas en Preferencias Combinadas* (*CPComb*), caracterizadas por incluir una expresión que permite combinar la especificaciones de varios criterios.

Para resolver las *CPCombs* se introducirá un tipo de servicio de razonamiento que extiende las capacidades de un *SRPCond*, denominado *Servicio de Razonamiento Rebatible basado en Preferencias Combinadas*(*SRPComb*). El objetivo de este tipo de servicios es proveer un razonamiento basado en más de un criterio.

7.1. Preferencias Combinadas

Como se pudo observar en capítulos anteriores un criterio de preferencia define una relación de preferencia sobre los argumentos. Existen escenarios donde las preferencias del usuario podrían estar ligadas a varios criterios; de hecho, un usuario de un sistema basado en DeLP podría requerir que sus consultas sean resueltas considerando diferentes criterios. En este sentido, en el Capítulo 2 se presentaron varios métodos que proponen

diferentes formas de combinar preferencias. En este capítulo se mostrará cómo aplicar varios de estos métodos de combinación en un servicio de razonamiento.

Una característica distintiva de los servicios que se definen en este capítulo corresponde a la capacidad de poder usar varios criterios a la vez para evaluar la preferencia entre argumentos. Es decir, estos servicios proveen operadores específicos que permitirán a las consultas considerar un uso combinado de varios criterios de preferencia.

Definición 7.1 (Operador de Combinación de Preferencias (OCP))**.** *Sean $\succsim_1$ y $\succsim_2$ dos relaciones de preferencia sobre argumentos, denotaremos $\theta(\succsim_1, \succsim_2)$ a un Operador de Combinación de Preferencias θ tal que dadas las relaciones $\succsim_1$ y $\succsim_2$ retorna una nueva relación de preferencia, $\succsim^*$.*

Siguiendo la notación de la Definición 4.16, $\succ^*$ y $\asymp^*$ corresponden respectivamente a la relaciones estricta y de incomparabilidad asociadas a $\succsim^*$.

En la Sección 7.3.1 se estudiarán algunas propiedades que servirán para establecer cuándo la relación que resulta de una operación de combinación de preferencias es apropiada a una consulta. Estas propiedades serán las mínimas requeridas para un operador de combinación; sin embargo, implementaciones particulares podrían agregar otras propiedades deseables.

En la literatura existen diversas propuestas para llevar a cabo la tarea de combinar preferencias [Cho03, SKP11, BDRS15]. A continuación se introducen algunos operadores de combinación de preferencias que serán utilizados a lo largo de este capítulo. En particular, se presentan dos operadores basados en operaciones de teoría de conjuntos [1] y un operador especial basado en prioridades de evaluación.

Definición 7.2 (Operación Intersección $\oplus$)**.** *Sea $\mathcal{P}$ un programa DeLP, y $\langle \mathcal{A}, L\rangle$ y $\langle \mathcal{B}, M\rangle$ dos argumentos construidos a partir de $\mathcal{P}$. Dados los criterios de preferencia $\succsim_{\mathsf{lc}_1}$ y $\succsim_{\mathsf{lc}_2}$, la operación intersección $\oplus(\succsim_{\mathsf{lc}_1}, \succsim_{\mathsf{lc}_2})$ es tal que el argumento $\langle \mathcal{A}, L\rangle$ es al menos tan preferido como $\langle \mathcal{B}, M\rangle$ (denotado $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$) si y solo si ($\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}_1} \langle \mathcal{B}, M\rangle$) y ($\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}_2} \langle \mathcal{B}, M\rangle$).*

La intuición detrás de la operación $\oplus(\succsim_{\mathsf{lc}_1}, \succsim_{\mathsf{lc}_2})$ es que un argumento es preferido si es preferido por los dos criterios considerados.

Ejemplo 7.1. *Considere los argumentos $\langle \mathcal{A}_{12}, sugerir(h1)\rangle$, $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle$, $\langle \mathcal{A}_5, \sim sParar\rangle$ y $\langle \mathcal{A}_4, sParar\rangle$ presentados en el ejemplo de la Sección 4.5 (la Figura 4.5*

[1] La semántica de los operadores se asemeja a la semántica de los operadores de teoría de conjuntos.

ilustra la estructura completa de estos argumentos). Para comparar argumentos, en este ejemplo, se introducen dos criterios de preferencia, $\succsim_{\mathsf{seg}}$ *y* $\succsim_{\mathsf{conf}}$*. Teniendo en cuenta* $\succsim_{\mathsf{seg}}$ *en la Figura 4.4-(a) se puede observar que* $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle \succ_{\text{sec}} \langle \mathcal{A}_{12}, sugerir(h1)\rangle$*. Asimismo, en la Figura 4.4-(b) se muestra que si el criterio considerado es* $\succsim_{\mathsf{conf}}$*, entonces* $\langle \mathcal{A}_{12}, sugerir(h1)\rangle \succ_{\mathsf{conf}} \langle \mathcal{A}_9, \sim sugerir(h1)\rangle$*. Por otra parte, si consideramos los argumento* $\langle \mathcal{A}_5, \sim sParar\rangle$ *y* $\langle \mathcal{A}_4, sParar\rangle$ *tenemos que para ambos criterios el primero es preferido sobre el segundo.*

La preferencia entre $\langle \mathcal{A}_5, \sim sParar\rangle$ *y* $\langle \mathcal{A}_4, sParar\rangle$ *también se puede establecer combinando los criterios* $\succsim_{\mathsf{seg}}$ *y* $\succsim_{\mathsf{conf}}$ *mediante el operador intersección,* $\oplus(\succsim_{\mathsf{seg}}, \succsim_{\mathsf{conf}})$*. En este caso en particular tenemos también que* $\langle \mathcal{A}_5, \sim sParar\rangle \succ^* \langle \mathcal{A}_4, sParar\rangle$*. No sucede lo mismo para los argumentos* $\langle \mathcal{A}_{12}, sugerir(h1)\rangle$ *y* $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle$ *donde* $\langle \mathcal{A}_{12}, sugerir(h1)\rangle \not\succ^* \langle \mathcal{A}_9, \sim sugerir(h1)\rangle$*.*

Definición 7.3 (Operación Diferencia $\ominus$)**.** *Sea* $\mathcal{P}$ *un programa DeLP, y* $\langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{B}, M\rangle$ *dos argumentos construidos a partir de* $\mathcal{P}$*. Dados los criterios de preferencia* $\succsim_{\mathsf{lc}_1}$ *y* $\succsim_{\mathsf{lc}_2}$*, la operación diferencia* $\ominus(\succsim_{\mathsf{lc}_1}, \succsim_{\mathsf{lc}_2})$ *es tal que el argumento* $\langle \mathcal{A}, L\rangle$ *es al menos tan preferido como* $\langle \mathcal{B}, M\rangle$ *(denotado* $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$*) si y solo si* $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}_1} \langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{A}, L\rangle \not\succsim_{\mathsf{lc}_2} \langle \mathcal{B}, M\rangle$

A diferencia de la operación anterior, la expresión $\ominus(\succsim_{\mathsf{lc}_1}, \succsim_{\mathsf{lc}_2})$ establece que $\langle \mathcal{A}, L\rangle$ es preferido a $\langle \mathcal{B}, M\rangle$ si $\langle \mathcal{A}, L\rangle$ es preferido por el criterio $\succsim_{\mathsf{lc}_1}$, y no se da que sea preferido por el criterio $\succsim_{\mathsf{lc}_2}$. En este caso, y considerando nuevamente los argumentos $\langle \mathcal{A}_{12}, sugerir(h1)\rangle$ y $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle$, a partir de $\ominus(\succsim_{\mathsf{seg}}, \succsim_{\mathsf{conf}})$ se da que $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle \succ^* \langle \mathcal{A}_{12}, sugerir(h1)\rangle$.

Definición 7.4 (Operación Priorizada $\circledast$)**.** *Sea* $\mathcal{P}$ *un programa DeLP, y* $\langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{B}, M\rangle$ *dos argumentos construidos a partir de* $\mathcal{P}$*. Dados los criterios de preferencia* $\succsim_{\mathsf{lc}_1}$ *y* $\succsim_{\mathsf{lc}_2}$*, la operación priorizada* $\circledast(\succsim_{\mathsf{lc}_1}, \succsim_{\mathsf{lc}_2})$ *es tal que el argumento* $\langle \mathcal{A}, L\rangle$ *es al menos tan preferido como* $\langle \mathcal{B}, M\rangle$ *(denotado* $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$*) si y solo si*

- $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}_1} \langle \mathcal{B}, M\rangle$*, o*
- $\langle \mathcal{A}, L\rangle \bowtie_{\mathsf{lc}_1} \langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}_2} \langle \mathcal{B}, M\rangle$.

Es decir, $\circledast(\succsim_{\mathsf{lc}_1}, \succsim_{\mathsf{lc}_2})$ expresa que el criterio $\succsim_{\mathsf{lc}_2}$ será tenido en cuenta únicamente si el criterio $\succsim_{\mathsf{lc}_1}$ no decide. A partir del Ejemplo 7.1, y teniendo en cuenta la combinación de criterios a través de la operación $\circledast(\succsim_{\mathsf{seg}}, \succsim_{\mathsf{conf}})$ se obtiene que $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle \succ^* \langle \mathcal{A}_{12}, sugerir(h1)\rangle$.

Observación 7.1. *Es importante notar que puede ocurrir que el orden de los operandos en un operador no sea indistinto, ya que puede darse el caso que las operaciones* $\theta(\succsim_1, \succsim_2)$ *y* $\theta(\succsim_2, \succsim_1)$ *den distintos resultados. Por ejemplo, en los OCPs introducidos en la Definiciones 7.2 y 7.3 se puede ver que si bien no importa el orden de los operandos en el operador* $\oplus$*, esto sí ocurre en el caso del operador* $\ominus$*. En particular, y considerando el escenario del Ejemplo 7.1 no tiene el mismo resultado la operación* $\ominus(\succsim_{\mathsf{seg}}, \succsim_{\mathsf{conf}})$ *que la operación* $\ominus(\succsim_{\mathsf{conf}}, \succsim_{\mathsf{seg}})$*. En el primer caso* $\langle \mathcal{A}_9, \sim sugerir(h1)\rangle \succ^* \langle \mathcal{A}_{12}, sugerir(h1)\rangle$*, mientras que en el segundo caso* $\langle \mathcal{A}_{12}, sugerir(h1)\rangle \succ^* \langle \mathcal{A}_9, \sim sugerir(h1)\rangle$*.*

El principal objetivo de los *OCPs* será proporcionar la posibilidad de utilizar varias de las implementaciones de criterios que se encuentran disponibles en el *SRPComb* consultado, manteniendo el control de cómo combinarlas en el propio servicio. Es decir, el servicio, al especificar qué operadores provee, está habilitando la forma en la cuál los usuarios pueden combinar los criterios almacenados utilizando las *CPComb*, que serán definidas en la siguiente sección.

7.2. Consulta basada en Preferencias Combinadas

En los Capítulos 5 y 6 se presentaron diferentes consultas distinguiendo diversas formas de expresar las preferencias del usuario. En esta sección se presentan las Consultas basadas en Preferencias Combinadas (*CPCombs*) las que permitirán al usuario combinar las especificaciones de varios criterios en una misma consulta favoreciendo de esta manera la representación de preferencias complejas. Dicha combinación se logra a partir de una expresión construida con *OCPs* que se encuentran definidos en el *SRPComb* consultado. La Figura 7.1 muestra los componentes que constituyen la arquitectura de los *SRPCombs*. Un *SRPComb* Υ estará conformado por:

- el *intérprete* DeLP que resuelve las consultas DeLP,
- una *función de evaluación* que a partir de una *expresión de preferencias combinadas* obtiene una especificación que declara las implementaciones de criterios que se utilizarán en la comparación de argumentos, y
- un *módulo extendido de comparación de argumentos* que a través de los *operadores de combinación de preferencias* y *las implementaciones de criterios de preferencia* almacenados en Υ se ocupa del cómputo de las preferencias entre argumentos que el intérprete necesita.

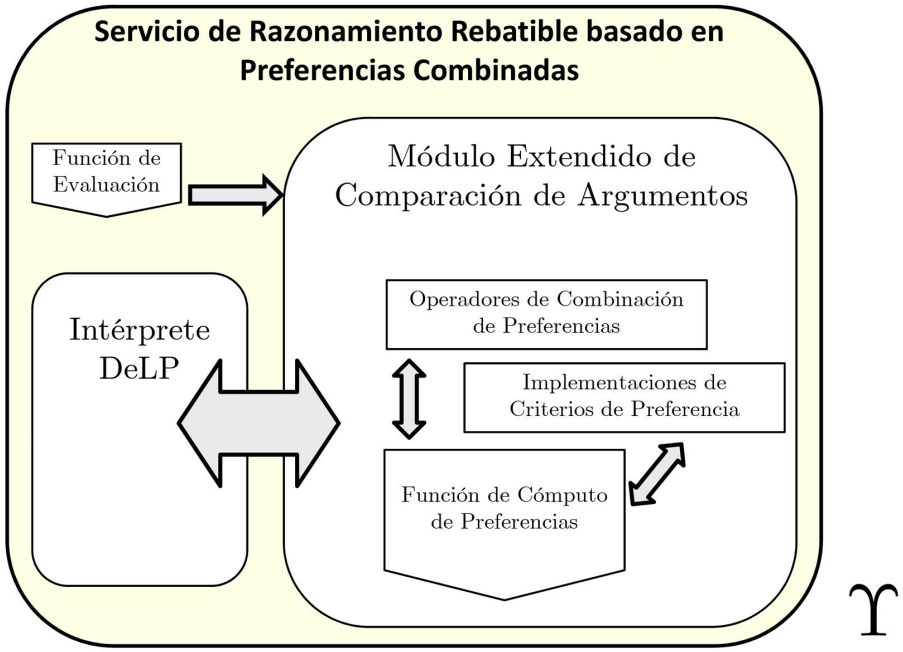

Figura 7.1: Arquitectura de un *SRPComb*.

Un *SRPComb* responderá una consulta siempre y cuando las especificaciones de criterios y los operadores de combinación de preferencias sean parte de una *especificación de preferencia* (*EPref*).

Definición 7.5 (Especificación de Preferencia (*EPref*))**.** *Dado un conjunto* C *de implementaciones computacionales de criterios de preferencia, sea* $\mathsf{impC}(\mathsf{Ic})$ *un elemento en* C*, y* $(\mathsf{Ic}, \mathsf{D})$ *una especificación de criterio para* $\mathsf{impC}(\mathsf{Ic})$*. Dado un conjunto de operadores de combinación de preferencias* $\mathbb{O}$*, una especificación de preferencia* $\mathcal{EP}$ *es una secuencia finita de símbolos definida como:*

$$\mathcal{EP} = \begin{cases} (\mathsf{Ic}, \mathsf{D}),\ o \\ \theta(\mathcal{EP}_1, \mathcal{EP}_2) \text{ donde } \mathcal{EP}_1 \text{ y } \mathcal{EP}_2 \text{ son especificaciones de preferencias y } \theta \in \mathbb{O}. \end{cases}$$

Observación 7.2. *Toda EPref tiene asociada una relación de preferencia* $\succsim$ *sobre argumentos. La relación para una EPref de la forma* $(\mathsf{Ic}, \mathsf{D})$ *corresponde a la definida por el criterio que en dicha especificación se indica,* $\succsim_{\mathsf{Ic}}$*. En el caso de una EPref de la forma* $\theta(\mathcal{EP}_1, \mathcal{EP}_2)$*, y como se detalla en la sección anterior, la relación asociada es la relación* $\succsim^*$ *que resulta de la operación de combinación de preferencias en cuestión que se vaya a aplicar.*

A continuación se ilustra un ejemplo de *EPrefs* con *OCPs* que luego será utilizado en ejemplos que se mostrarán más adelante en este capítulo.

Ejemplo 7.2. *Considerando las siguientes ECrits introducidas en el Ejemplo 5.3,*

- $\mathcal{EC}_{\mathsf{seg}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{seg}})$
- $\mathcal{EC}_{\mathsf{conf}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{conf}})$

A partir de los OCPs $\oplus$ y $\ominus$ presentados en la Definición 7.2 y 7.3 respectivamente, es posible construir las siguientes especificaciones de preferencias.

$$\mathcal{EP}_{\mathsf{segYconf}} = \oplus(\mathcal{EC}_{\mathsf{seg}}, \mathcal{EC}_{\mathsf{conf}})$$
$$\mathcal{EP}_{\mathsf{prioSeg}} = \ominus(\mathcal{EC}_{\mathsf{seg}}, \mathcal{EC}_{\mathsf{conf}})$$

Para determinar si un argumento es preferido a otro, el módulo de comparación de argumentos de los servicios de razonamiento presentados en los capítulos anteriores empleaban las *ECrits* que eran enviadas por el usuario junto a sus consultas. En particular, los servicios que se presentan en este capítulo se caracterizan por utilizar para tal fin las *EPrefs* que se formalizaron arriba. De esta manera, a partir de la *EPref* recibida, el módulo dedicado a la comparación de argumentos será quien se encargará de evaluar estas especificaciones utilizando para ello una función $\mathsf{evalEspP}$ que será introducida a continuación.

Dados dos argumentos $\langle \mathcal{A}, L\rangle$ y $\langle \mathcal{B}, M\rangle$, la idea intuitiva de cómo una especificación de preferencia $\mathcal{EP}$ se evalúa será capturada mediante la función $\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$ cuyo rango es $\{\top, \bot\}$, retornando $\top$ si $\langle \mathcal{A}, L\rangle$ es al menos tan preferido como $\langle \mathcal{B}, M\rangle$ bajo $\mathcal{EP}$ y $\bot$ en caso contrario.

Definición 7.6 (Función de Evaluación para *EPrefs*)**.** *Sea Υ un SRPComb. Sea D_{p} el dominio de todos los posibles programas DeLP, y Args el conjunto de todos los argumentos que se pueden obtener de un programa $\mathcal{P}$ en el dominio D_{p}. Sea D_{ep} el conjunto de todas las posibles especificaciones de preferencias construidas a partir de $(\mathsf{C}, \mathbb{O})$ tal que C es un conjunto de implementaciones computacionales de criterios de preferencia y $\mathbb{O}$ un conjunto de operadores de combinación de preferencias, ambos almacenados en el servicio Υ. La función de evaluación para especificaciones de preferencias,*

$$\mathsf{evalEspP} : D_{\mathsf{ep}} \times \mathit{Args} \times \mathit{Args} \mapsto \{\top, \bot\},$$

es tal que dada una especificación de preferencia $\mathcal{EP} \in D_{\mathsf{ep}}$*, y dos argumentos* $\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle \in Args$ *se define como:*

$$\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) = \left\{ \begin{array}{l} \top \textit{ si } \langle \mathcal{A}, L\rangle \succsim \langle \mathcal{B}, M\rangle \\ \bot \textit{ en caso contrario.} \end{array} \right\}$$

A continuación se detalla un ejemplo de la definición de evalEspP considerando para ello las operaciones de combinación de preferencias definidas en la sección anterior (intersección, diferencia, y priorizada).

Ejemplo 7.3. *Considere el conjunto de operadores de combinación de preferencias* $\mathbb{O} = \{\oplus, \ominus, \circledast\}$ *introducidos en la Sección 7.1, y las especificaciones de preferencias* $\mathcal{EP}_1$*,* $\mathcal{EP}_2$ *y* $\mathcal{EP}$*. Entonces, la evaluación de cada operador se podría definir por casos como sigue:*

- $\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) = \mathsf{compPref}(\mathsf{impC}(\mathsf{lc}), \mathsf{D}, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$ *si* $\mathcal{EP}$=(lc, D)*,*
- $\mathsf{evalEspP}(\oplus(\mathcal{EP}_1, \mathcal{EP}_2), \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$ *si* $\mathsf{evalEspP}(\mathcal{EP}_1, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$ *y* $\mathsf{evalEspP}(\mathcal{EP}_2, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$*,*
- $\mathsf{evalEspP}(\ominus(\mathcal{EP}_1, \mathcal{EP}_2), \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$ *si* $\mathsf{evalEspP}(\mathcal{EP}_1, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$ *y* $\mathsf{evalEspP}(\mathcal{EP}_2, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\bot$*,*
- $\mathsf{evalEspP}(\circledast(\mathcal{EP}_1, \mathcal{EP}_2), \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$ *si* $\mathsf{evalEspP}(\mathcal{EP}_1, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$ *o* $\mathsf{evalEspP}(\mathcal{EP}_1, \langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle)$=$\bot$ *y* $\mathsf{evalEspP}(\mathcal{EP}_2, \langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle)$=$\top$*,*
- $\bot$ *en caso contrario.*

Entonces, una especificación de preferencia $\mathcal{EP}$ se interpreta como sigue: si $\mathcal{EP}$ es de la forma (lc, D), entonces se utiliza para la comparación de argumentos la implementación del criterio identificado como lc. Por otra parte, si $\mathcal{EP} = \theta(\mathcal{EP}_1,\mathcal{EP}_2)$ entonces se aplica la operación de combinación θ en cuestión; cada operación particular define una manera diferente de combinar el uso de las *EPrefs* que recibe.

Ahora que se definieron las especificaciones que el módulo de un *SRPComb* utilizará para computar las preferencias entre argumentos, es posible definir la expresión desde la cual se podrán obtener dichas especificaciones. La expresión que se introducirá a continuación permite programar la selección de una *EPref* en particular a partir del uso de una

estructura condicional basada en guardas. Como se verá en la siguiente definición, la formalización del concepto de *expresión de preferencias combinadas* (*exp-comb*) está basada en las nociones de expresión de preferencia condicional (ver Definición 6.2) presentada en el capítulo anterior y especificación de preferencia definida arriba.

Definición 7.7 (Expresión de Preferencias Combinadas)**.** *Dado un conjunto* C *de implementaciones computacionales de criterios de preferencia, sea* $\mathsf{impC}(\mathsf{Ic})$ *un elemento en* C*, y* $(\mathsf{Ic}, \mathsf{D})$ *una especificación de criterio para* $\mathsf{impC}(\mathsf{Ic})$*. Dado un conjunto de operadores de combinación de preferencias* $\mathbb{O}$*, una exp-comb* $\mathcal{E}^*$ *es una secuencia finita de símbolos definida recursivamente como:*

$$\mathcal{E}^* = \begin{cases} (\mathsf{Ic}, \mathsf{D}),\ o \\ \theta(\mathcal{E}_1^*, \mathcal{E}_2^*) \text{ donde } \mathcal{E}_1^* \text{ y } \mathcal{E}_2^* \text{ son exps-combs y } \theta \in \mathbb{O},\ o \\ [\mathcal{G} : \mathcal{E}_1^*; \mathcal{E}_2^*] \text{ donde } \mathcal{E}_1^* \text{ y } \mathcal{E}_2^* \text{ son exps-combs.} \end{cases}$$

Dado un conjunto C de implementaciones computacionales de criterios de preferencia y un conjunto $\mathbb{O}$ de operadores de combinación de preferencias, el conjunto de todas las posibles *exps-combs* construidas a partir del par (C,$\mathbb{O}$) se denotará $\mathrm{D}_{\mathsf{e}^*}$. Claramente una *exp-comb* corresponde a la herramienta que poseen los usuarios para expresar sus preferencias al momento de realizar una consulta. Como se puede observar a través de una *exp-comb* es posible construir *exps-conds*. Sin embargo, note que lo interesante de las *exps-combs* es la posibilidad de construir otros tipos de expresiones. Por ejemplo expresiones donde las *EPrefs* que son operandos en una expresión que contiene un operador de combinación de preferencias pueden ser evaluadas luego de evaluar una expresión condicional. Un ejemplo de este tipo de expresiones podría ser $\theta([\mathcal{G} : \mathcal{EC}_1; \mathcal{EC}_2], \mathcal{EC}_1)$; como se puede observar, esta *exp-comb* se caracteriza por tener una expresión condicional $[\mathcal{G} : \mathcal{EC}_1; \mathcal{EC}_2]$ de la cual se obtendrá, luego de ser evaluada, una de las *EPrefs* que el operador θ podrá utilizar para sus cálculos.

Una *exp-comb* $\mathcal{E}^*$ se interpreta de la siguiente manera: si $\mathcal{E}^*$ es una especificación de preferencia, entonces se usa dicha especificación para la comparación de argumentos, mientras que si $\mathcal{E}^*$ es $[\mathcal{G} : \mathcal{E}_1^*; \mathcal{E}_2^*]$ y $\mathcal{G}$ se satisface para un programa DeLP $\mathcal{P}$, entonces se evaluá $\mathcal{E}_1^*$, caso contrario será evaluada $\mathcal{E}_2^*$. Cabe recordar que la guarda $\mathcal{G}$ se satisface si existe una derivación estricta desde $\mathcal{P}$ para cada uno de los literales que la constituyen. Para representar cómo una *exp-comb* se evalúa se empleará la función que se define a continuación.

Definición 7.8 (Función de Evaluación)**.** *Sea* Υ *un SRPComb. Sea* D_{p} *el dominio de todos los posibles programas DeLP. Sea* D_{e^*} *el conjunto de todas las posibles exps-combs construidas a partir de* $(\mathsf{C}, \mathbb{O})$ *tal que* C *es un conjunto de implementaciones computacionales de criterios de preferencia y* $\mathbb{O}$ *un conjunto de operadores de combinación de preferencias, ambos almacenados en el servicio* Υ*. Sea* S^* *el conjunto de todas las especificaciones de preferencias involucradas en las exps-combs de* D_{e^*}*. La función de evaluación para exps-combs* $\mathsf{evalE}^* : D_{\mathsf{e}^*} \times D_{\mathsf{p}} \longrightarrow S^*$*, es tal que dada una exp-comb* $\mathcal{E}^*$ *en* D_{e^*} *y un programa DeLP* $\mathcal{P}$ *en* D_{p}*, se define como:*

$$\mathsf{evalE}^*(\mathcal{E}^*, \mathcal{P}) = \begin{cases} (\mathsf{Ic}, \mathsf{D}) \text{ si } \mathcal{E}^* = (\mathsf{Ic}, \mathsf{D}), \text{ o} \\ \theta(\mathsf{evalE}^*(\mathcal{E}_1^*, \mathcal{P}), \mathsf{evalE}^*(\mathcal{E}_2^*, \mathcal{P})) \text{ si } \mathcal{E}^* = \theta(\mathcal{E}_1^*, \mathcal{E}_2^*), \text{ o} \\ \mathsf{evalE}^*(\mathcal{E}_1^*, \mathcal{P}) \text{ si } \mathcal{E}^* = [\mathcal{G} : \mathcal{E}_1^*; \mathcal{E}_2^*] \text{ y } \mathcal{G} \text{ se satisface para } \mathcal{P}, \text{ o} \\ \mathsf{evalE}^*(\mathcal{E}_2^*, \mathcal{P}) \text{ si } \mathcal{E}^* = [\mathcal{G} : \mathcal{E}_1^*; \mathcal{E}_2^*] \text{ y } \mathcal{G} \text{ no se satisface para } \mathcal{P} \end{cases}$$

Ejemplo 7.4. *Considere el OCP* $\circledast$ *definido en Definición 7.4. A partir de las especificaciones de preferencias* $\mathcal{EP}_{\mathsf{segYconf}}$ *y* $\mathcal{EP}_{\mathsf{prioSeg}}$ *presentadas en el Ejemplo 7.2, y la especificación de criterio* $\mathcal{EC}_{\mathsf{conf}}$ *utilizada en tales especificaciones es posible construir las siguientes dos exps-combs.*

- $\mathcal{E}_1^* = [\{zPolicias(h1), estrellas(h1, 5)\} : \mathcal{EP}_{\mathsf{segYconf}}; \mathcal{EP}_{\mathsf{prioSeg}}]$
- $\mathcal{E}_2^* = \circledast(\mathcal{E}_1^*, \mathcal{EC}_{\mathsf{conf}})$

Intuitivamente, la expresión $\mathcal{E}_1^*$ *se interpretará como sigue: "si existe una derivación estricta para* $zPolicias(h1)$ *y* $estrellas(h1, 5)$ *entonces se usa la especificación de preferencia* $\mathcal{EP}_{\mathsf{segYconf}}$*, de lo contrario se usa la especificación* $\mathcal{EP}_{\mathsf{prioSeg}}$*". Mientras que, la expresión* $\mathcal{E}_2^*$ *se interpreta de la siguiente manera: "usar la especificación que combina, a partir del OCP* $\circledast$*, la especificación de preferencia* $\mathcal{EC}_{\mathsf{conf}}$ *con la especificación de preferencia que resulta de evaluar* $\mathcal{E}_1^*$*". Considere el programa* $\mathcal{P}_h$ *del Ejemplo de Aplicación de la Sección 4.5 del Capítulo 4. Para* $\mathcal{E}_1^*$*, la guarda* $\{zPolicias(h1), estrellas(h1, 5)\}$ *no se satisface para* $\mathcal{P}_h$ *ya que* $zPolicias(h1)$ *no tiene una derivación estricta; por lo tanto* $\mathsf{evalE}^*(\mathcal{E}_1^*, \mathcal{P}_h) = \mathcal{EP}_{\mathsf{prioSeg}}$*. Para* $\mathcal{E}_2^*$*, se tiene en cuenta el resultado de la evaluación de la subexpresión* $\mathcal{E}_1^*$*. Como el resultado de evaluar* $\mathcal{E}_1^*$ *es* $\mathcal{EP}_{\mathsf{prioSeg}}$ *se tiene que* $\mathsf{evalE}^*(\mathcal{E}_2^*, \mathcal{P}_h) = \circledast(\mathcal{EP}_{\mathsf{prioSeg}}, \mathcal{EC}_{\mathsf{conf}})$ *donde* $\mathcal{EP}_{\mathsf{prioSeg}} = \ominus(\mathcal{EC}_{\mathsf{seg}}, \mathcal{EC}_{\mathsf{conf}})$*.*

El Capítulo 6 describe las consultas basadas en preferencias con expresiones condicionales. Estas consultas, a diferencia de las consultas presentadas en el Capítulo 5, incluyen

una expresión para seleccionar criterios. En este capítulo la estructura de las consultas será similar a las presentadas en el Capítulo 6; sin embargo, ahora en lugar de incluir una *exp-cond* las consultas podrán incluir una *exp-comb*. A continuación se formaliza la noción de *Consulta basada en Preferencias Combinadas*.

Definición 7.9 (Consulta basada en Preferencias Combinadas (*CPComb*))**.** *Sea* D_{p} *el dominio de todos los posibles programas DeLP, y* D_{c} *el dominio de las consultas DeLP. Una consulta basada en preferencias combinadas es una tupla* $(\mathcal{E}^*, \mathcal{P}, Q)$*, donde* $\mathcal{E}^*$ *es una exp-comb,* $\mathcal{P}$ *un programa en el dominio* D_{p}*, y* Q *una consulta perteneciente al dominio* D_{c}*.*

Para resolver las *CPCombs* más adelante en esta sección se introduce el concepto de Servicio de Razonamiento Rebatible basado en Preferencias Combinadas (*SRPComb*). De manera similar a un *SRPCond*, un *SRPComb* estará integrado por un intérprete y un módulo para decidir entre argumentos en conflicto. Como se verá a continuación dicho módulo será adaptado de manera tal que un *SRPComb* pueda resolver las preferencias entre argumentos considerando para ello la *EPref* que resulta de la función evalE^*.

En lo que sigue se definirá el concepto de Módulo Extendido de Comparación de Argumentos (*MEC*), que estará compuesto por un conjunto de operadores de combinación de preferencias, un conjunto de implementaciones de criterios de preferencia, la función de cómputo de preferencias para dicho conjunto de implementaciones, y la función de evaluación de especificaciones de preferencias. A continuación se introduce la definición de *MEC*.

Definición 7.10 (Módulo Extendido de Comparación de Argumentos (*MEC*))**.** *Sea* Υ *un SRPComb. Sea* $\mathbb{O}$ *un conjunto de operadores de combinación de preferencias,* C *un conjunto de implementaciones computacionales de criterios de preferencia, ambos almacenados en el servicio* Υ*,* $\mathsf{evalEspP}$ *una función de evaluación para especificaciones de preferencias, y* $\mathsf{compPref}$ *la función de cómputo de preferencias para las implementaciones en* C*. La tupla* $\langle \mathbb{O}, \mathsf{C}, \mathsf{evalEspP}, \mathsf{compPref} \rangle$ *efectiviza el módulo extendido de comparación de argumentos* $\mathbb{M}^*$ *en* Υ*.*

Habiendo definido todos los componentes, en lo que sigue se introduce el concepto de Servicio de Razonamiento Rebatible basado en Preferencias Combinadas.

Definición 7.11 (Servicio de Razonamiento Rebatible basado en Preferencias Combinadas (*SRPComb*))**.** *Un servicio de razonamiento rebatible basado en preferencias combinadas es una tupla* $\langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$*, donde* evalE^* *es una función de evaluación para*

exps-combs, $\mathbb{I}$ *un intérprete DeLP, y* $\mathbb{M}^*$ *un módulo extendido de comparación de argumentos.*

Definición 7.12 (Expresión de Preferencias Combinadas adecuada)**.** *Sea* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ *un SRPComb. Se dirá que una expresión de preferencias combinadas* $\mathcal{E}^*$ *es adecuada para un módulo extendido de comparación de argumentos* $\mathbb{M}^* = \langle \mathbb{O}, \mathsf{C}, \mathsf{evalEspP}, \mathsf{compPref} \rangle$ *si se cumple que:*

- *cada especificación de preferencia de la forma* $(\mathsf{Ic}, \mathsf{D})$ *involucrada en* $\mathcal{E}^*$ *es tal que para el identificador de criterio* Ic *existe una implementación* $\mathsf{impC}(\mathsf{Ic}) \in \mathsf{C}$*, y*
- *cada operador de combinación de preferencias* θ *involucrado en* $\mathcal{E}^*$ *es tal que* $\theta \in \mathbb{O}$*.*

Definición 7.13 (*CPComb* adecuada para un *SRPComb*)**.** *Sea* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ *un SRPComb. Una consulta basada en preferencias combinadas* $(\mathcal{E}^*, \mathcal{P}, Q)$ *será adecuada para* Υ*, si* $\mathcal{E}^*$ *es adecuada para* $\mathbb{M}^*$ *en* Υ*.*

Al momento de calcular la respuesta a una *CPComb*, el *MEC* del servicio consultado se encargará de procesar las preferencias entre argumentos utilizando la *EPref* obtenida de tal consulta. La respuestas que dará un *MEC* dependerá del valor que retorne la función evalEspP. A continuación se introduce la definición correspondiente.

Definición 7.14 (Respuesta del Módulo Extendido de Comparación de Argumentos)**.** *Sea* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ *un SRPComb tal que* $\mathbb{M}^* = \langle \mathbb{O}, \mathsf{C}, \mathsf{evalEspP}, \mathsf{compPref} \rangle$*, y* $\mathcal{CC} = (\mathcal{E}^*, \mathcal{P}, Q)$ *una consulta basada en preferencias combinadas adecuada para* Υ*. Sea Args el conjunto de todos los argumentos que se pueden obtener a partir del programa* $\mathcal{P}$*, y* $\langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle \in Args$*. La respuesta de* $\mathbb{M}^*$ *que resulta de comparar* $\langle \mathcal{A}, L \rangle$ *con* $\langle \mathcal{B}, M \rangle$ *utilizando la especificación de preferencia* $\mathcal{EP}$ *obtenida desde* $\mathsf{evalE}^*(\mathcal{E}^*, \mathcal{P})$*, denotada* $\mathsf{respMod}(\mathbb{M}^*, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle, \mathcal{EP})$*, corresponde a una de las siguientes:*

- $\langle \mathcal{A}, L \rangle$ *si*
  $\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle) = \top$ *y*
  $\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{B}, M \rangle, \langle \mathcal{A}, L \rangle) = \bot$,
- $\langle \mathcal{B}, M \rangle$ *si*
  $\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{B}, M \rangle, \langle \mathcal{A}, L \rangle) = \top$ *y*
  $\mathsf{evalEspP}(\mathcal{EP}, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle) = \bot$,
- $\emptyset$ *en caso contrario.*

A continuación se definirá cómo actúan los *SRPComb* para resolver las consultas. En lo que sigue se introduce la definición de respuesta para una *CPComb*.

Definición 7.15 (Respuesta para una *CPComb*)**.** *Sea* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ *un SRPComb y* $\mathcal{CC} = (\mathcal{E}^*, \mathcal{P}, Q)$ *una consulta basada en preferencias combinadas adecuada para* Υ*. Sea* $\mathcal{EP}$ *la especificación de preferencia obtenida desde* $\mathsf{evalE}^*(\mathcal{E}^*, \mathcal{P})$*. La respuesta para* $\mathcal{CC}$ *en* Υ*, denotada* $\mathsf{resp}(\Upsilon, \mathcal{CC})$*, corresponde al resultado del intérprete DeLP* $\mathbb{I}(\mathcal{P}, Q)$*, siendo que se verifica la siguiente condición. Dados dos argumentos* $\langle \mathcal{A}, L \rangle$ *y* $\langle \mathcal{B}, M \rangle$ *obtenidos desde el programa DeLP* $\mathcal{P}$*,*

- $\langle \mathcal{A}, L \rangle \succ \langle \mathcal{B}, M \rangle$ *si* $\mathsf{respMod}(\mathbb{M}^*, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle, \mathcal{EP}) = \langle \mathcal{A}, L \rangle$*, y*
- $\langle \mathcal{A}, L \rangle \asymp \langle \mathcal{B}, M \rangle$ *si* $\mathsf{respMod}(\mathbb{M}^*, \langle \mathcal{A}, L \rangle, \langle \mathcal{B}, M \rangle, \mathcal{EP}) = \emptyset$*.*

Proposición 7.1. *Las CPConds para un SRPCond* $\Phi = \langle \mathsf{evalE}, \mathbb{I}, \mathbb{M} \rangle$ *son un caso particular de las CPCombs recibidas por un SRPComb* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$*.*

Demostración: Una consulta basada en preferencias condicionales $\langle \mathcal{E}, \mathcal{P}, Q \rangle$ se puede escribir como una consulta basada en preferencias combinadas $(\mathcal{E}, \mathcal{P}, Q)$. Por Definición 6.3 y 7.8 la respuesta de $\mathsf{evalE}(\mathcal{E}, \mathcal{P})$ corresponde a una especificación de criterio que coincide con la respuesta de $\mathsf{evalE}^*(\mathcal{E}, \mathcal{P})$, en consecuencia también coinciden las respuestas del módulo de comparación de argumentos $\mathbb{M}$ en Φ con las del módulo extendido de comparación de argumentos $\mathbb{M}^*$ en Υ para cada par de argumentos que el intérprete DeLP necesite comparar. Por Definición 6.7 y 7.15 la respuesta para ambos servicios corresponde al resultado del intérprete DeLP $\mathbb{I}(\mathcal{P}, Q)$, por lo tanto la respuesta para $\langle \mathcal{E}, \mathcal{P}, Q \rangle$ en Φ es la misma respuesta que daría Υ para la consulta $(\mathcal{E}, \mathcal{P}, Q)$. □

Observación 7.3. *Es importante resaltar que si el conjunto de OCPs* $\mathbb{O}$ *es vacío en un SRPComb* Υ*, entonces el comportamiento de* Υ *corresponde al de un SRPCond. Esta situación se debe al hecho de que las exps-combs en las CPCombs no podrán contar con OCPs, en consecuencia tendrán una estructura igual a la de una exp-cond. Por lo tanto, la respuesta que dará* Υ *es equivalente a la que daría un SRPCond. Teniendo en cuenta esto, de aquí en adelante se asumirá que* $\mathbb{O}$ *no es un conjunto vacío.*

Dada la Observación 7.3, se puede ver que no sería una decisión coherente implementar un *SRPComb* si no se piensa en el uso de los *OCPs*. De hecho, los beneficios de este tipo de servicio radican justamente en proveer al usuario de herramientas concretas que le

permitan expresar y combinar sus preferencias. Es claro que la implementaciones de los servicios de razonamiento propuestos en esta tesis estarán relacionados a un dominio de aplicación o a un problema específico que se pueda abordar por uno de estos tipos de servicios en particular.

Ejemplo 7.5. *Considere el siguiente SRPComb*

$$\Upsilon_h = \langle \mathsf{evalE}^*{}_h, \mathbb{I}_h, \langle \{\oplus, \ominus, \circledast\}, \{\mathsf{impC}(\mathsf{prioR})\}, \mathsf{evalEspP}_h, \mathsf{compPref}_h \rangle \rangle$$

donde el conjunto $\{\oplus, \ominus, \circledast\}$ *contiene los OCPs introducidos en las Definiciones 7.2, 7.3, y 7.4 respectivamente, mientras que* $\mathsf{impC}(\mathsf{prioR})$ *corresponde a una implementación computacional del criterio de prioridad entre reglas* $\succeq_{\mathsf{prioR}}$ *presentado en la Definición 4.27. En particular este servicio responderá consultas realizadas al programa DeLP* $\mathcal{P}_h$ *introducido en el ejemplo de la Sección 4.5.*

Considere ahora las exps-combs $\mathcal{E}_1^*$ *y* $\mathcal{E}_2^*$ *introducidas en el Ejemplo 7.4. Note que a partir de la expresión* $\mathcal{E}_1^*$ *el SRPComb podría recibir la siguiente CPComb,*

$$\mathcal{CC}_1 = (\mathcal{E}_1^*, \mathcal{P}_h, sugerir(h1))$$

.

Como se muestra en el Ejemplo 7.4, el resultado de evaluar $\mathcal{E}_1^*$ *corresponde a la especificación de preferencia* $\mathcal{EP}_{\mathsf{prioSeg}} = \ominus(\mathcal{EC}_{\mathsf{seg}}, \mathcal{EC}_{\mathsf{conf}})$*, siendo ésta la especificación que se tendrá en cuenta al momento de resolver las preferencias en el proceso de cómputo de la respuesta a la consulta* $sugerir(h1)$*. Por otra parte, considerando la exp-comb* $\mathcal{E}_2^*$ *es posible enviar otra CPComb al SRPComb,*

$$\mathcal{CC}_2 = (\mathcal{E}_2^*, \mathcal{P}_h, sugerir(h1))$$

.

En este caso en particular, el servicio de razonamiento utilizará la especificación de preferencia $\circledast(\mathcal{EP}_{\mathsf{prioSeg}}, \mathcal{EC}_{\mathsf{conf}})$*. Esto último se debe a que* $\mathsf{evalE}^*(\mathcal{E}_2^*, \mathcal{P}_h) = \circledast(\mathcal{EP}_{\mathsf{prioSeg}}, \mathcal{EC}_{\mathsf{conf}})$*. Finalmente, obsérvese que tanto* $\mathcal{CC}_1$ *como* $\mathcal{CC}_2$ *son CPCombs adecuadas a* Υ_h*.*

Note que el ejemplo anterior muestra dos consultas con diferentes *exps-combs*. Por un lado se introduce la consulta basada en preferencias combinadas $\mathcal{CC}_1$ que utiliza la expresión $\mathcal{E}_1^*$, mientras que $\mathcal{CC}_2$ considera la expresión $\mathcal{E}_1^*$. Cabe destacar que de la evaluación

de estas expresiones se obtienen distintas *EPrefs*, en consecuencia como se viene detallando desde los capítulos precedentes es posible que la respuesta varíe para cada consulta en particular. Es importante resaltar que un usuario podría enviar al *SRPComb* consultas con preferencias más complejas, es decir, anidando *exps-combs*. En el Sección 7.4 se explorará un ejemplo de aplicación concreto en donde se aplica el formalismo propuesto.

Como se muestra de manera esquemática en la Figura 7.2 los pasos para responder una *CPComb* coinciden en ciertos aspectos con los descriptos para los servicios de razonamiento presentados en capítulos anteriores. Es decir, un *SRPComb* contará con un intérprete que responde consultas DeLP, y además se comunica con un módulo encargado de resolver las preferencias que éste necesita. Sin embargo, observe que la arquitectura de un *SRPComb* está diseñada de manera tal que puede recibir expresiones que contienen *OCPs*. Una vez que la función evalE* obtiene la *EPref* que se utilizará, el *MEC* es capaz de llevar a cabo, a través de la función evalEspP, todas las comparaciones entre argumentos que el intérprete le solicita.

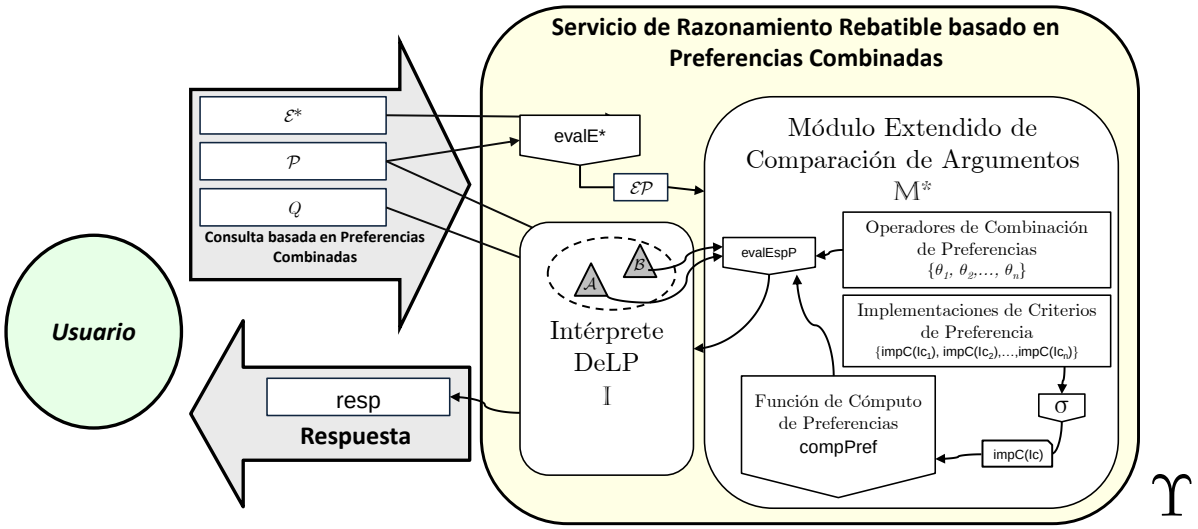

Figura 7.2: Principales componentes de una *CPComb* y un *SRPComb*.

En la siguiente sección se estudiarán algunas propiedades para que los criterios implementados en un servicio de razonamiento y los *OCPs* que un *SRPComb* pueda tener definido sean capaces, en lo posible, de resolver todas las comparaciones entre argumentos que se requieran. Finalmente, en la Sección 7.4 se desarrolla un ejemplo que se utilizará

para mostrar cómo se puede aprovechar el modelo propuesto en este capítulo en búsqueda de recomendaciones de películas de acuerdo a las preferencias del usuario.

7.3. Análisis y Propiedades

7.3.1. Indecisión por Incomparabilidad entre Argumentos

El Capítulo 2 muestra varias propiedades que las relaciones de preferencias pueden satisfacer. En lo que sigue se analiza este conjunto de propiedades en un contexto de argumentación con el objetivo de determinar qué propiedades deberían de cumplir los criterios de preferencia y *OCPs* de manera que las respuestas de los servicios de razonamiento que se proponen en esta tesis sean confiables. El objetivo de esto es evitar usar criterios que no puedan determinar preferencias entre argumentos, es decir evitar criterios que para cualquier par de argumentos la comparación establece que son incomparables.

Para definir la relación de derrota entre dos argumentos en DeLP se asume que existe un mecanismo que permite compararlos y decidir qué argumento es preferido. En el Capítulo 4 se definió un criterio de preferencia de forma general como una relación binaria $\succsim_{\mathsf{lc}}$ que permite establecer si un argumento es preferido a otro. Note que esta relación se podría caracterizar formalmente a través de diferentes propiedades. Dados los argumentos $\langle \mathcal{A}, L\rangle$, $\langle \mathcal{B}, M\rangle$ y $\langle \mathcal{C}, N\rangle$ construidos a partir de un programa DeLP $\mathcal{P}$, se dirá que:

- $\succsim_{\mathsf{lc}}$ es reflexivo si y solo si $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{A}, L\rangle$.
- $\succsim_{\mathsf{lc}}$ es irreflexivo si y solo si $\langle \mathcal{A}, L\rangle \not\succsim_{\mathsf{lc}} \langle \mathcal{A}, L\rangle$.
- $\succsim_{\mathsf{lc}}$ es completo si y solo si $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{B}, M\rangle$ o $\langle \mathcal{B}, M\rangle \succsim_{\mathsf{lc}} \langle \mathcal{A}, L\rangle$.
- $\succsim_{\mathsf{lc}}$ es transitivo si cuando $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{B}, M\rangle$ y $\langle \mathcal{B}, M\rangle \succsim_{\mathsf{lc}} \langle \mathcal{C}, N\rangle$, entonces $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{C}, N\rangle$.
- $\succsim_{\mathsf{lc}}$ es simétrico si cuando $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{B}, M\rangle$ entonces $\langle \mathcal{B}, M\rangle \succsim_{\mathsf{lc}} \langle \mathcal{A}, L\rangle$.
- $\succsim_{\mathsf{lc}}$ es antisimétrico si cuando $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{B}, M\rangle$ y $\langle \mathcal{B}, M\rangle \succsim_{\mathsf{lc}} \langle \mathcal{A}, L\rangle$, entonces $\mathcal{A} = \mathcal{B}$ y $L = M$.
- $\succsim_{\mathsf{lc}}$ es asimétrico si cuando $\langle \mathcal{A}, L\rangle \succsim_{\mathsf{lc}} \langle \mathcal{B}, M\rangle$ entonces $\langle \mathcal{B}, M\rangle \not\succsim_{\mathsf{lc}} \langle \mathcal{A}, L\rangle$.

En un contexto de argumentación ningún argumento debería ser mejor que sí mismo, de hecho $\succsim_{\mathsf{lc}}$ debería de ser irreflexiva. Por otro lado, la completitud tampoco es una característica deseable ya que llevaría a que un criterio de preferencia siempre deba determinar una preferencia y en situaciones del mundo real esto en general no sucede; es razonable pensar que en algunos casos los argumentos son incomparables. Asimismo, la transitividad si bien parece deseable, muchas veces los argumentos $\langle \mathcal{A}, L\rangle$ y $\langle \mathcal{C}, N\rangle$ no tienen una vinculación directa aunque si tengan una conexión con $\langle \mathcal{B}, M\rangle$. Del mismo modo, la simetría es otra propiedad poco deseable en una relación entre argumentos ya que para cualquier par de argumentos $\langle \mathcal{A}, L\rangle$ y $\langle \mathcal{B}, M\rangle$ obtenido desde un programa DeLP dado daría que son incomparables, $\langle \mathcal{A}, L\rangle \asymp_{\mathsf{lc}} \langle \mathcal{B}, M\rangle$. Si en todos las situaciones de comparación entre argumentos, los mismos son incomparables, el sistema fallaría en sus respuestas cayendo en la mayoría de los casos en una situación de indecisión; ésto lo haría poco confiable.

Note que el único caso en donde DeLP no dará una respuesta de indecisión a partir de una consulta DeLP, independientemente del criterio utilizado, es cuando existe un argumento que soporta dicha consulta o su complemento y a su vez no se ha encontrado un contra-argumento para dicho argumento. Como en esta sección se estudiará el caso de indecisión por incomparabilidad, se asumirá que toda consulta DeLP pertenece a la signatura del programa consultado, y aquellos argumentos que soportan una consulta o el complemento de ésta siempre tendrán un contra-argumento que los ataque a fin de poder ser comparados.

Dado un programa DeLP $\mathcal{P}$ y una consulta Q a $\mathcal{P}$, para las formalizaciones que siguen se denotará $Args_Q$ al conjunto de argumentos construidos en el proceso de garantizado de Q. Se dirá que $Args_Q$ es el conjunto de argumentos construido desde $\mathcal{P}$ a partir de la consulta Q.

Proposición 7.2. *Sea $\mathcal{P}$ un programa DeLP, y Q una consulta a $\mathcal{P}$. Dado el conjunto de argumentos $Args_Q$ construido desde $\mathcal{P}$ a partir de Q, y $\succsim$ la relación de preferencia sobre $Args_Q$ utilizada en el cálculo de la respuesta a la consulta Q. Si para cada par de argumentos comparados $\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle \in Args_Q$ se da que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \asymp$ de acuerdo a $\succsim$, entonces la respuesta para Q será* INDECISO.

Demostración: Evidente a partir de las definiciones relevantes. □

Ejemplo 7.6. *Considere el programa DeLP $\mathcal{P}_{7.6}$:*

$$\mathcal{P}_{7.6} = \left\{ \begin{array}{ll} a \prec b, c & f \leftarrow e \\ b \prec f & c \\ b \prec d & d \\ \sim b \prec \sim a, d & e \\ \sim a \prec c & \\ \sim b \prec e & \\ g \prec f & \\ \sim g \prec d & \end{array} \right\}$$

a partir de la consulta "a" es posible obtener el siguiente conjunto de argumentos:

$\langle \mathcal{A}_1, a \rangle$ *donde* $\mathcal{A}_1 = \{a \prec b, c, b \prec d\}$

$\langle \mathcal{A}_2, a \rangle$ *donde* $\mathcal{A}_2 = \{a \prec b, c, b \prec f\}$

$\langle \mathcal{A}_3, b \rangle$ *donde* $\mathcal{A}_3 = \{b \prec d\}$

$\langle \mathcal{A}_4, b \rangle$ *donde* $\mathcal{A}_4 = \{b \prec f\}$

$\langle \mathcal{A}_5, \sim b \rangle$ *donde* $\mathcal{A}_5 = \{\sim b \prec \sim a, d\}$

$\langle \mathcal{A}_6, \sim b \rangle$ *donde* $\mathcal{A}_6 = \{\sim b \prec e\}$

$\langle \mathcal{A}_7, \sim a \rangle$ *donde* $\mathcal{A}_7 = \{\sim a \prec c\}$

Suponga que existe un criterio de preferencia $\succeq_{\mathsf{lc}}$ *el cual establece las siguientes preferencias entre argumentos* $\succeq_{\mathsf{lc}} = \{(\langle \mathcal{A}_3, b \rangle, \langle \mathcal{A}_5, \sim b \rangle), (\langle \mathcal{A}_5, \sim b \rangle, \langle \mathcal{A}_3, b \rangle), (\langle \mathcal{A}_4, b \rangle, \langle \mathcal{A}_6, \sim b \rangle), (\langle \mathcal{A}_6, \sim b \rangle, \langle \mathcal{A}_4, b \rangle)\}$. *A partir de la consulta "a", tenemos que todos los pares de argumentos en conflicto que se pueden encontrar no son comparables por el criterio* $\succeq_{\mathsf{lc}}$. *Por lo tanto, la respuesta DeLP será* INDECISO.

Como se puede observar en el ejemplo anterior, al momento de elegir un criterio el peor caso sería contar con un criterio que no pueda decidir para cada par de argumentos a ser comparados, llevando que la respuesta del intérprete DeLP entre en un estado de indecisión por incomparabilidad entre argumentos. A continuación se introduce una noción que permite distinguir aquellos mecanismos de comparación que ante una consulta establecen al menos una preferencia en el proceso de comparación de argumentos, independientemente de si existe indecisión o no en la respuesta a dicha consulta.

Definición 7.16 (Confiabilidad)**.** *Sea $\mathcal{P}$ un programa DeLP, y Q una consulta para $\mathcal{P}$. Dado el conjunto de argumentos $Args_Q$ construido desde $\mathcal{P}$ a partir de Q, y $\succsim$ la relación de preferencia sobre $Args_Q$ utilizada en el cálculo de la respuesta a la consulta Q. Se dice que $\succsim$ es confiable para la consulta Q si $\succsim$ es no vacía, irreflexiva y asimétrica.*

Observación 7.4. *Sea $\mathcal{P}$ un programa DeLP, y Q una consulta para $\mathcal{P}$. Dado el conjunto de argumentos $Args_Q$ construido desde $\mathcal{P}$ a partir de Q, toda relación de preferencia $\succsim$ sobre $Args_Q$ se asume que es confiable para Q.*

En la Sección 7.3.2 y 7.3.2 se analizan los criterios de preferencias y operaciones de combinación de preferencias respectivamente introducidos en esta tesis con el objetivo de mostrar que cumplen la condición de irreflexividad y asimetría.

Ejemplo 7.7. *A partir del Ejemplo 7.6, se puede observar que $\succsim_{\mathsf{lc}}$ no es confiable para "a". Por lo contrario, si para este mismo ejemplo se asume que el criterio utilizado es especificidad generalizada $\succsim_{\mathsf{esp}}$ (ver Definición 4.23), entonces tenemos que $\succsim_{\mathsf{esp}}= \{(\langle\mathcal{A}_1, a\rangle, \langle\mathcal{A}_7, \sim a\rangle), (\langle\mathcal{A}_5, \sim b\rangle, \langle\mathcal{A}_3, b\rangle), (\langle\mathcal{A}_6, \sim b\rangle, \langle\mathcal{A}_4, b\rangle), (\langle\mathcal{A}_2, a\rangle, \langle\mathcal{A}_7, \sim a\rangle)\}$. En la Sección 7.3.2 se prueba que $\succsim_{\mathsf{esp}}$ cumple la propiedad de irreflexividad y asimetría.*

El tipo de relación caracterizada más arriba está estrechamente relacionada con las comparaciones que se generan a partir de una consulta en particular. Por lo tanto, dos consultas DeLP diferentes realizadas a un mismo programa $\mathcal{P}$, y utilizando un mismo criterio para ambas consultas podría dar como resultado que la relación de preferencia definida por este criterio para una consulta sea confiable pero para la otra no.

Ejemplo 7.8. *Considere de nuevo el Ejemplo 7.6, asumiendo que se utiliza el criterio $\succsim_{\mathsf{esp}}$, a partir de la consulta "a" se puede observar que $\succsim_{\mathsf{esp}}$ es confiable para "a". Sin embargo, para la consulta "g" y usando el mismo criterio, se construye el argumento $\langle\mathcal{A}_8, g\rangle$ donde $\mathcal{A}_8 = \{g \prec f\}$ y el contra-argumento $\langle\mathcal{A}_9, \sim g\rangle$ donde $\mathcal{A}_9 = \{\sim g \prec d\}$, tal que éstos son incomparables con respecto a $\succsim_{\mathsf{esp}}$. Por lo tanto, en este último caso $\succsim_{\mathsf{esp}}$ no es confiable para "g".*

Finalmente, la siguiente definición vincula el concepto de *EPref* con las nociones desarrolladas en esta sección. Se dirá que una especificación de preferencia es *efectiva* para una consulta Q si se da que la relación de preferencia entre argumentos que dicha especificación tiene asociada cumple con la característica de ser confiable para la consulta en cuestión.

Definición 7.17 (Especificación de Preferencia Efectiva)**.** *Sea $\mathcal{P}$ un programa DeLP, y Q una consulta para $\mathcal{P}$. La especificación de preferencia $\mathcal{EP}$ es efectiva para Q si*

1. *$\mathcal{EP} = (\mathsf{lc}, \mathsf{D})$ y $\succsim_{\mathsf{lc}}$ es confiable para Q, o*
2. *$\mathcal{EP} = \theta(\mathcal{EP}_1, \mathcal{EP}_2)$ y $\succsim^*$ es confiable para Q.*

Observación 7.5. *Una EPref efectiva para una consulta dada no evita que pueda existir indecisión en la respuesta a dicha consulta. Sea $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ un SRPComb, $\mathcal{CC} = [\mathcal{E}^*, \mathcal{P}, Q]$ una consulta basada en preferencias combinadas para Υ, y $\mathcal{R} = \mathsf{resp}(\Upsilon, \mathcal{CC})$ la respuesta para $\mathcal{CC}$ de Υ. Dada la especificación de preferencia $\mathcal{EP}$ que resulta de $\mathsf{evalE}^*(\mathcal{E}^*, \mathcal{P})$, tal que es efectiva para Q, se cumple que $\mathcal{R}$ puede ser* INDECISO *como muestra el siguiente ejemplo.*

Ejemplo 7.9. *Consideremos el programa DeLP $\mathcal{P}_{7.6}$ del Ejemplo 7.6. Suponga que se realiza la consulta "a", y $\succsim$ es la relación de preferencia asociada a la especificación de preferencia $\mathcal{EP}$ utilizada en el cálculo de la respuesta a dicha consulta. Asumiendo que $\mathcal{EP}$ es efectiva para "a" y $\succsim = \{(\langle \mathcal{A}_5, \sim b\rangle, \langle \mathcal{A}_4, b\rangle), (\langle \mathcal{A}_3, b\rangle, \langle \mathcal{A}_6, \sim b\rangle)\}$, tenemos que la respuesta para "a", será* INDECISO*. La Figura 7.3 muestra los árboles generados a partir del análisis dialéctico llevado a cabo para la consulta en cuestión. La estructura completa de los argumentos involucrados en tal análisis se pueden ver en el Ejemplo 7.6.*

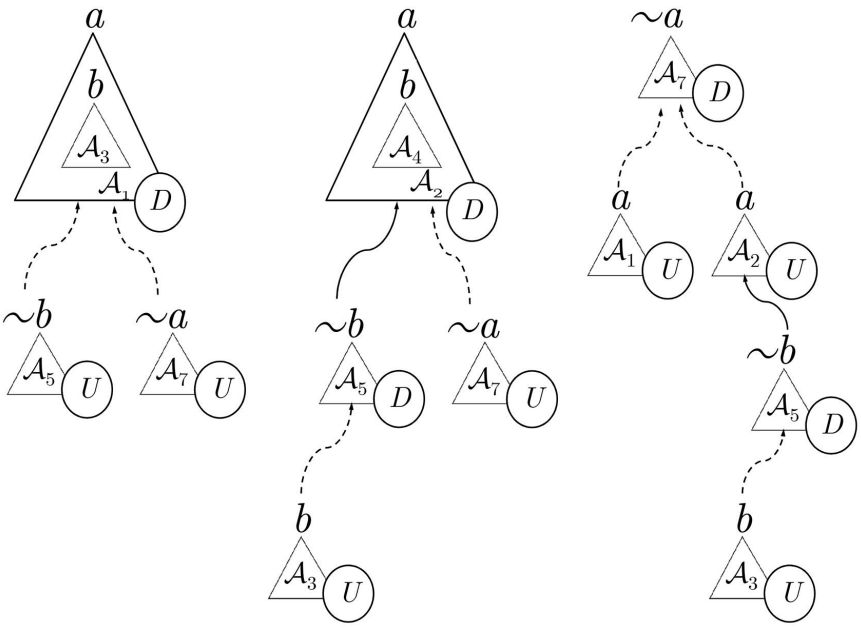

Figura 7.3: Árboles de dialéctica construidos para la consulta "a".

Como se muestra en el ejemplo anterior, una *EPref* que es efectiva para una consulta dada no garantiza que no pueda existir indecisión en la respuesta a dicha consulta, pero si éste es el caso, es claro que no se dará por incomparabilidad para cada par de argumentos comparados. Esto se debe al hecho de que una relación confiable establece como mínimo una preferencia estricta entre argumentos.

Definición 7.18 (Respuesta basada en Preferencias Estrictas)**.** *Sea* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ *un SRPComb,* $\mathcal{CC} = [\mathcal{E}^*, \mathcal{P}, Q]$ *una consulta basada en preferencias combinadas para* Υ*, y* $\mathcal{R} = \mathsf{resp}(\Upsilon, \mathcal{CC})$ *la respuesta para* $\mathcal{CC}$ *de* Υ*. Se dice que* $\mathcal{R}$ *está basada en preferencias estrictas si y solo si la* $\mathcal{EP}$ *que resulta de* $\mathsf{evalE}^*(\mathcal{E}^*, \mathcal{P})$ *es efectiva para* Q*.*

Proposición 7.3. *Sea* $\Upsilon = \langle \mathsf{evalE}^*, \mathbb{I}, \mathbb{M}^* \rangle$ *un SRPComb,* $\mathcal{CC} = [\mathcal{E}^*, \mathcal{P}, Q]$ *una consulta basada en preferencias combinadas para* Υ*. Si se cumple que la respuesta* $\mathcal{R}$ *para* $\mathcal{CC}$ *de* Υ *es* $\mathsf{resp}(\Upsilon, \mathcal{CC}) \in \{\text{SI}, \text{NO}\}$*, entonces* $\mathcal{R}$ *está basada en preferencias estrictas.*

Demostración: Por Observación 7.4 la relación de preferencia asociada a los criterios de preferencias implementados como así también la relación que resulta de aplicar los operadores de combinación de preferencias alojados en Υ se asume que son confiables para Q. Asumiendo que existe al menos un par de argumentos comparados, para que la respuesta $\mathcal{R}$ sea SI o NO es necesario que exista al menos una preferencia estricta en la relación de preferencia asociada a la *EPref* obtenida desde $\mathsf{evalE}^*(\mathcal{E}^*, \mathcal{P})$. Dado que dicha relación se asume no vacía, entonces se cumple que $\mathcal{R}$ está basada en preferencias estrictas. □

En esta sección se estudio el caso en donde las respuestas de un servicio de razonamiento caen en un estado de indecisión por incomparabilidad entre argumentos. Para intentar evitar esta situación, se propuso que tanto la relación de preferencia vinculada a los criterios de preferencias como la relación que resulta de una operación de combinación de preferencias deberían de cumplir las propiedades de irreflexividad y asimetría. En la Sección 7.3.3 se estudia qué criterios y qué operaciones de las descriptas en las secciones 4.4 y 7.1 respectivamente satisfacen dichas propiedades.

7.3.2. Propiedades de los Criterios de Preferencia

Un criterio de preferencia establece una relación de preferencia entre los argumentos que se construyen al momento de computarse la respuesta a una consulta DeLP. A con-

tinuación se analizarán las propiedades de irreflexividad y asimetría para los siguientes criterios: $\succsim_{\mathsf{esp}}$, $\succsim_{\mathsf{espE}}$, $\succsim_{\mathsf{prioL}}$, $\succsim_{\mathsf{prioR}}$, y $\succsim_{\mathsf{prioRE}}$.

Especificidad Generalizada (Definición 4.23)

Proposición 7.4. *El criterio de especificidad generalizada es irreflexivo, es decir para todo argumento $\langle \mathcal{A}, L\rangle$ obtenido de un programa DeLP $\mathcal{P}$, $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \notin \succsim_{\mathsf{esp}}$.*

Demostración: Sea $\langle \mathcal{A}, L\rangle$ un argumento obtenido de $\mathcal{P}$, y $\mathcal{F}$ el conjunto de literales que tienen una derivación rebatible a partir de $\mathcal{P}$. Asumiendo que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{esp}}$, por la condición (2) de la Definición 4.23 tenemos que existe al menos un conjunto $H' \subseteq \mathcal{F}$, tal que existe una derivación rebatible para L a partir de $\Omega \cup H' \cup \mathcal{A}$, no existe una derivación estricta de L a partir de $\Omega \cup H'$, y no existe una derivación rebatible de L a partir de $\Omega \cup H' \cup \mathcal{A}$. Entonces no puede ser que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{esp}}$. □

Proposición 7.5. *El criterio de especificidad generalizada es asimétrico, es decir para todo argumento $\langle \mathcal{A}, L\rangle$ y $\langle \mathcal{B}, M\rangle$ obtenido de un programa DeLP $\mathcal{P}$, si $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{esp}}$ entonces $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \notin \succsim_{\mathsf{esp}}$.*

Demostración: Sean $\langle \mathcal{B}, M\rangle$ y $\langle \mathcal{A}, L\rangle$ dos argumentos obtenidos de $\mathcal{P}$, y $\mathcal{F}$ el conjunto de literales que tienen una derivación rebatible a partir de $\mathcal{P}$. Asumiendo que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{esp}}$, por la condición (2) de la Definición 4.23 tenemos que existe al menos un conjunto $H' \subseteq \mathcal{F}$, tal que existe una derivación rebatible para M a partir de $\Omega \cup H' \cup \mathcal{B}$, no existe una derivación estricta de M a partir de $\Omega \cup H'$, y no existe una derivación rebatible de L a partir de $\Omega \cup H' \cup \mathcal{A}$. Por lo tanto, no es posible que se cumpla que para todo conjunto $H \subseteq \mathcal{F}$, existe una derivación rebatible para M a partir de $\Omega \cup H \cup \mathcal{B}$, y no existe una derivación estricta para M a partir de $\Omega \cup H$, entonces existe una derivación rebatible para L a partir de $\Omega \cup H \cup \mathcal{A}$. Entonces, no es posible que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{esp}}$. □

Especificidad Generalizada Extendida (Definición 4.25)

Proposición 7.6. *El criterio de especificidad generalizada extendida es irreflexivo, es decir para todo argumento $\langle \mathcal{A}, L\rangle$ de un programa DeLP $\mathcal{P}$, $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \notin \succsim_{\mathsf{espE}}$.*

Demostración. Probaremos que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{esp}}$ no se satisface para ninguna de las tres condiciones definidas en la Definición 4.25:

- *Sea* $\langle \mathcal{A}, L\rangle$ *un argumento obtenido de* $\mathcal{P}$*, y* $\mathcal{F}$ *el conjunto de literales que tienen una derivación rebatible a partir de* $\mathcal{P}$*. Asumiendo que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$*, por la condición (a).2 de la Definición 4.25 tenemos que existe al menos un conjunto* $H' \subseteq \mathcal{F}$*, tal que existe una derivación rebatible para* L *a partir de* $\Omega \cup H' \cup \mathcal{A}$*, no existe una derivación estricta de* L *a partir de* $\Omega \cup H'$*, y no existe una derivación rebatible de* L *a partir de* $\Omega \cup H' \cup \mathcal{A}$*. Entonces no puede ser que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$*.*

- *Sea* $\langle \mathcal{A}, L\rangle$ *un argumento obtenido de* $\mathcal{P}$*. Asumiendo que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$ *por la condición (b), se tiene que* $\langle \mathcal{A}, L\rangle$ *se basa en hechos, y* $\langle \mathcal{A}, L\rangle$ *se basa en presuposiciones. Entonces, no se cumple que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$*.*

- *Sea* $\langle \mathcal{A}, L\rangle$ *un argumento obtenido de* $\mathcal{P}$*. Asumiendo que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$*, luego por la condición (c), se tiene que ambos argumentos están basadas en presuposiciones y* $RR(\mathcal{A}) = \emptyset$ *y* $RR(\mathcal{A}) \neq \emptyset$*. Esto último no es posible, en consecuencia, no se cumple que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$*.*

□

Proposición 7.7. *El criterio de especificidad generalizada extendida es asimétrico, es decir para todo argumento* $\langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{B}, M\rangle$ *de un programa DeLP* $\mathcal{P}$*, si* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{espE}}$ *entonces* $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \notin \succsim_{\mathsf{espE}}$*.*

Demostración: Supongamos que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{espE}}$*. Probaremos que* $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$ *no se satisface para ninguna de las tres condiciones definidas en la Definición 4.25:*

- *Sean* $\langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{A}, L\rangle$ *dos argumentos obtenidos de* $\mathcal{P}$*, y* $\mathcal{F}$ *el conjunto de literales que tienen una derivación rebatible a partir de* $\mathcal{P}$*. Dado que asumimos que* $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{espE}}$*, luego por la condición (a).2 tenemos que existe al menos un conjunto* $H' \subseteq \mathcal{F}$*, tal que existe una derivación rebatible para* M *a partir de* $\Omega \cup H' \cup \mathcal{B}$*, no existe una derivación estricta de* M *a partir de* $\Omega \cup H'$*, y no existe una derivación rebatible de* L *a partir de* $\Omega \cup H' \cup \mathcal{A}$*. Por lo tanto, no es posible que se cumpla que para todo conjunto* $H \subseteq \mathcal{F}$*, existe una derivación rebatible para* M *a partir de* $\Omega \cup H \cup \mathcal{B}$*, y no existe una derivación estricta para* M *a partir de* $\Omega \cup H$*, entonces existe una derivación rebatible para* L *a partir de* $\Omega \cup H \cup \mathcal{A}$*. Entonces, no es posible que* $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$ *por la condición (a).*

- *Sean $\langle \mathcal{B}, M\rangle$ y $\langle \mathcal{A}, L\rangle$ dos argumentos obtenidos de $\mathcal{P}$. Dado que asumimos que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$ por la condición (b), se tiene que $\langle \mathcal{B}, M\rangle$ se basa en hechos, y $\langle \mathcal{A}, L\rangle$ se basa en presuposiciones, lo cual contradice la hipótesis.*

- *Sean $\langle \mathcal{B}, M\rangle$ y $\langle \mathcal{A}, L\rangle$ dos argumentos obtenidos de $\mathcal{P}$. Dado que asumimos que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{espE}}$, luego por la condición (c), se tiene que ambos argumentos están basados en presuposiciones y $RR(\mathcal{B}) = \emptyset$ y $RR(\mathcal{A}) \neq \emptyset$, lo cual contradice la hipótesis.*

□

Prioridad entre Literales (Definición 4.26)

Proposición 7.8. *El criterio de prioridad entre literales es irreflexivo.*

Demostración: Sea $\langle \mathcal{A}, L\rangle$ un argumento obtenido de $\mathcal{P}$. Asumamos que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioL}}$, por Definición 4.26 existe al menos un literal $M \in^* \mathcal{A}$, tal que $M > M$, arribando así a un absurdo. Entonces $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \notin \succsim_{\mathsf{prioL}}$. □

Proposición 7.9. *El criterio de prioridad entre literales es asimétrico.*

Demostración: Sean $\langle \mathcal{B}, M\rangle$ y $\langle \mathcal{A}, L\rangle$ dos argumentos obtenidos de $\mathcal{P}$. Suponiendo por el absurdo que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioL}}$ y asumiendo que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{prioL}}$, por la segunda condición de la Definición 4.26 no hay literales M_1 y M_2 tal que $M_1 \in^* \mathcal{A}$, $M_2 \in^* \mathcal{B}$, y $M_2 > M_1$. Por lo tanto no se cumple que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioL}}$. □

Prioridad entre Reglas (Definición 4.27)

Proposición 7.10. *El criterio de prioridad entre reglas es irreflexivo.*

Demostración: Sea $\langle \mathcal{A}, L\rangle$ un argumento obtenido de $\mathcal{P}$. Asumamos que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioR}}$, luego por Definición 4.27 existe al menos una regla $r \in \mathcal{A}$, tal que $r > r$. Por lo tanto, no se da que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioR}}$. □

Proposición 7.11. *El criterio de prioridad entre reglas es asimétrico.*

Demostración: Sean $\langle \mathcal{B}, M\rangle$ y $\langle \mathcal{A}, L\rangle$ dos argumentos obtenidos de $\mathcal{P}$. Suponiendo por el absurdo que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioR}}$ y asumiendo que $(\langle \mathcal{A}, L\rangle, \langle \mathcal{B}, M\rangle) \in \succsim_{\mathsf{prioR}}$, por la segunda condición de la Definición 4.27 no existe ninguna $r'_1 \in \mathcal{A}$ y $r' \in \mathcal{B}$, tal que $r' > r'_1$. Por lo tanto no se cumple que $(\langle \mathcal{B}, M\rangle, \langle \mathcal{A}, L\rangle) \in \succsim_{\mathsf{prioR}}$. □

Prioridad entre Reglas Extendido (Definición 4.28)

Proposición 7.12. *El criterio de prioridad entre reglas extendido es irreflexivo.*

Demostración: Directa por Proposición 7.10 y Definición 4.28. □

Proposición 7.13. *El criterio de prioridad entre reglas extendido es asimétrico.*

Demostración: Directa por Proposición 7.11 y Definición 4.28. □

Dados los resultados mostrados en esta sección, se puede concluir que todos los criterios de preferencia descriptos satisfacen las propiedades de irreflexividad y asimetría. Asimismo, esto no quita que un servicio de razonamiento considere para la comparación de argumentos criterios diferentes que cumplan otras propiedades. Sin embargo, como se ha discutido previamente, en lo posible sería preferible contar con criterios que en la mayoría de los casos puedan comparar argumentos. Es por esto que al momento de definir los criterios con los que va a contar un servicio de razonamiento es recomendable tener en cuenta el estudio realizado en la Sección 7.3.1.

7.3.3. Propiedades de las Operadores de Combinación de Preferencias

En la Sección 7.1 se presentaron varios operadores para combinar preferencias. A continuación se analiza si dichos operadores producen al aplicarlos relaciones que satisfacen las propiedades de irreflexividad y asimetría.

Operación Intersección (Definición 7.2)

Proposición 7.14. *Dadas $\succsim_1$ y $\succsim_2$, irreflexivas y asimétricas, la operación de intersección $\oplus$ aplicada a estas relaciones produce una relación que es irreflexiva y asimétrica.*

Demostración: Asumiendo que $\succsim_1$ y $\succsim_2$ son irreflexivos y asimétricos. Para mostrar que la intersección $\succsim^* = \oplus(\succsim_1, \succsim_2)$ es irreflexiva y asimetrica debemos probar que:

1. *No se cumple que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{A}, L\rangle$ (Irreflexividad)*

   Suponiendo por el absurdo que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{A}, L\rangle$, tendríamos que $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{A}, L\rangle$ y $\langle \mathcal{A}, L\rangle \succsim_2 \langle \mathcal{A}, L\rangle$, lo cual es un absurdo dado que $\succsim_1$ y $\succsim_2$ son irreflexivas.

2. *Si $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$ entonces $\langle \mathcal{B}, M\rangle \not\succsim^* \langle \mathcal{A}, L\rangle$ (Asimetría)*

Asumiendo que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$ *tenemos que* $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{A}, L\rangle \succsim_2 \langle \mathcal{B}, M\rangle$. *Suponiendo por el absurdo que* $\langle \mathcal{B}, M\rangle \succsim^* \langle \mathcal{A}, L\rangle$, *tendríamos que* $\langle \mathcal{B}, M\rangle \succsim_1 \langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{B}, M\rangle \succsim_2 \langle \mathcal{A}, L\rangle$, *arribando a un absurdo.*

3. *No se cumple que* $\succsim^*$ *es no vacía.*

   Para mostrar que no se cumple $\succsim^*$ *es no vacía considere el siguiente contraejemplo:* $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{B}, M\rangle \succsim_2 \langle \mathcal{A}, L\rangle$, *entonces* $\succsim^*$ *es vacío.*

□

Operación Diferencia (Definición 7.3)

Proposición 7.15. *Dadas* $\succsim_1$ *y* $\succsim_2$, *irreflexivas y asimétricas, la operación diferencia* $\ominus$ *aplicada a estas relaciones produce una relación que es irreflexiva y asimétrica.*

Demostración: Asumiendo que $\succsim_1$ *y* $\succsim_2$ *son irreflexivos y asimétricos. Para mostrar que la diferencia* $\succsim^* = \ominus(\succsim_1, \succsim_2)$ *es irreflexiva y asimetrica debemos probar que:*

1. *No se cumple que* $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{A}, L\rangle$ *(Irreflexividad)*

   Suponiendo por el absurdo que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{A}, L\rangle$, *tendríamos que* $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{A}, L\rangle \not\succsim_2 \langle \mathcal{A}, L\rangle$, *lo cual es un absurdo dado que* $\succsim_1$ *es irreflexivo.*

2. *Si* $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$ *entonces* $\langle \mathcal{B}, M\rangle \not\succsim^* \langle \mathcal{A}, L\rangle$ *(Asimetría)*

   Asumiendo $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$ *tenemos que* $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{A}, L\rangle \not\succsim_2 \langle \mathcal{B}, M\rangle$. *Suponiendo por el absurdo que* $\langle \mathcal{B}, M\rangle \succsim^* \langle \mathcal{A}, L\rangle$, *tendríamos que* $\langle \mathcal{B}, M\rangle \succsim_1 \langle \mathcal{A}, L\rangle$ *y* $\langle \mathcal{B}, M\rangle \not\succsim_2 \langle \mathcal{A}, L\rangle$, *arribando a un absurdo.*

3. *No se cumple que* $\succsim^*$ *es no vacía.*

   Para mostrar que no se cumple $\succsim^*$ *es no vacía considere el siguiente contraejemplo:* $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{B}, M\rangle$ *y* $\langle \mathcal{A}, L\rangle \succsim_2 \langle \mathcal{B}, M\rangle$, *entonces* $\succsim^*$ *es vacío.*

□

Operación Priorizada (Definición 7.4)

Proposición 7.16. *Dadas $\succsim_1$ y $\succsim_2$, irreflexivas y asimétricas, la operación priorizada $\circledast$ aplicada a estas relaciones produce una relación que es irreflexiva y asimétrica.*

Demostración: Asumamos que $\succsim_1$ y $\succsim_2$ son irreflexivos y asimétricos. Para mostrar que la operación priorizada $\succsim^* = \circledast(\succsim_1, \succsim_2)$ es irreflexiva y asimétrica debemos probar que:

1. *No se cumple que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{A}, L\rangle$ (Irreflexividad)*

   Suponiendo por el absurdo que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{A}, L\rangle$, tendríamos que $\langle \mathcal{A}, L\rangle \succ_1 \langle \mathcal{A}, L\rangle$ o $(\langle \mathcal{A}, L\rangle \bowtie_1 \langle \mathcal{A}, L\rangle$ y $\langle \mathcal{A}, L\rangle \succ_2 \langle \mathcal{A}, L\rangle)$, lo cual es un absurdo dado que $\succsim_1$ y $\succsim_2$ son irreflexivos.

2. *Si $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$ entonces $\langle \mathcal{B}, M\rangle \not\succsim^* \langle \mathcal{A}, L\rangle$ (Asimetría)*

   Dado que asumimos que $\langle \mathcal{A}, L\rangle \succsim^* \langle \mathcal{B}, M\rangle$ tenemos que $\langle \mathcal{A}, L\rangle \succsim_1 \langle \mathcal{B}, M\rangle$ o $(\langle \mathcal{A}, L\rangle \bowtie_1 \langle \mathcal{B}, M\rangle$ y $\langle \mathcal{A}, L\rangle \succsim_2 \langle \mathcal{B}, M\rangle)$. Suponiendo por el absurdo que $\langle \mathcal{B}, M\rangle \succsim^* \langle \mathcal{A}, L\rangle$, tendríamos que $\langle \mathcal{B}, M\rangle \succsim_1 \langle \mathcal{A}, L\rangle$ o $(\langle \mathcal{B}, M\rangle \bowtie_1 \langle \mathcal{A}, L\rangle$ y $\langle \mathcal{B}, M\rangle \succsim_2 \langle \mathcal{A}, L\rangle)$, arribando a un absurdo.

3. *No se cumple que $\succsim^*$ es no vacía.*

   Para mostrar que no se cumple $\succsim^*$ es no vacía considere el siguiente contraejemplo: $\langle \mathcal{B}, M\rangle \succsim_1 \langle \mathcal{A}, L\rangle$ y $\langle \mathcal{A}, L\rangle \succsim_2 \langle \mathcal{B}, M\rangle$, entonces $\succsim^*$ es vacío.

□

7.4. Ejemplo de aplicación en un entorno de recomendación de películas

Presentaremos ahora un ejemplo que muestra los conceptos desarrollados en este capítulo aplicados a un dominio específico de recomendación de películas el cual fue introducido en [BBD+14]. En particular, en [BBD+14] se presenta una extensión del marco para lograr recomendaciones basados en razonamiento rebatible presentado en [BCM13]. Una característica importante de este formalismo es la utilización de DeLP como lenguaje para representar el dominio de recomendación y computar el proceso de argumentación. De hecho, los autores utilizan reglas DeLP para representar el conjunto de postulados

que describirá las condiciones bajo las cuales una película debería ser recomendada a un usuario dado. Un aspecto importante de este trabajo al utilizar reglas es la posibilidad de soportar las recomendaciones ya sea con información del dominio de carácter cualitativo como también con información cuantitativa; de esta manera es posible implementar sistemas recomendadores que combinen ambos aspectos. A partir de esto último, en esta sección se describe como mejorar las capacidades de estos sistemas para que el mecanismo de inferencia se pueda ajustar con mayor facilidad a las preferencias del usuario. Para lograr esto, se muestra que utilizando el modelo de servicio de razonamiento presentado en este capítulo el usuario podrá elegir entre recibir recomendaciones basadas en preferencias cualitativas, cuantitativas, o una combinación de ambas si así lo desea. A continuación se ilustran algunos de los postulados definidos en [BBD$^+$14] con sus respectivas reglas que los modelan.

Postulado	**Descripción**
1	A un usuario le parece buena una película si la ponderación promedio de la película está por arriba de la ponderación promedio general.
2	A un usuario no le parece buena una película si la ponderación promedio de la película está por abajo de la ponderación promedio general.
3	A un usuario le gusta una película si el género de la película es de los géneros favoritos del usuario.
4	A un usuario no le gusta una película si el género de la película es uno de los que más le desagrada al usuario.
5	A un usuario le gusta una película si tiene uno de sus actores favoritos.
6	A un usuario no le gusta una película si tiene uno de los actores que más le desagrada al usuario.

Cuadro 7.1: Postulados

Postulado	Descripción
1	r_1: $good\_movie(M) \prec avg\_rating(R, M), R > 3.8$
2	r_2: $\sim good\_movie(M) \prec avg\_rating(R, M), R < 3.8$
3	r_3: $likes\_by\_genre(M, U) \prec top\_genre(U, G), genre(M, G)$
4	r_4: $\sim likes\_by\_genre(M, U) \prec bottom\_genre(U, G), genre(M, G)$
5	r_5: $likes\_by\_actor(M, U) \prec top\_actor(U, A), leads\_in(M, A)$
6	r_6: $\sim likes\_by\_actor(M, U) \prec bottom\_actor(U, A), leads\_in(M, A)$

Cuadro 7.2: Reglas Rebatibles

Obsérvese que los postulados utilizan información auxiliar para encontrar los actores y géneros que a los usuarios más les gusta o, en caso contrario, más le desagrada.

- *top_actor*: Existen dos formas para encontrar los actores favoritos. La primera es obtener los tres actores que aparecen con mayor frecuencia en las películas que el usuario mira. La otra alternativa es considerar únicamente aquellas películas que el usuario ha calificado con un cuatro o con un valor mayor.

- *bottom_actor*: Se obtiene a partir de los tres actores que aparecen con mayor frecuencia en aquellas películas que el usuario ha calificado con un dos o con un valor menor.

- *top_genre*: Se establece a partir de los tres géneros que aparecen con mayor frecuencia en aquellas películas que el usuario ha calificado con un cuatro o con un valor mayor.

- *bottom_genre*: Se determina a partir de los tres géneros que aparecen con mayor frecuencia en aquellas películas que el usuario ha calificado con un dos o con un valor menor.

El Cuadro 7.2 muestra un conjunto de reglas que en algunos casos para soportar la recomendación de una película dada utiliza información cuantitativa (postulado 1 y 2) y en otros información cualitativa (postulado 3, 4, 5, 6). Teniendo en cuenta esta información, en el ejemplo que se presentará a continuación un usuario podrá decidir que tipo de información prefiere que sea considerada en el proceso de recomendación.

Suponga que quien recibe y resuelve todas las consultas DeLP es el *SRPComb* Υ_p = $\langle \mathsf{evalE}^*_p, \mathbb{I}_p, \langle \{\circledast\}, \{\mathsf{impC(prioR)}, \mathsf{impC(esp)}\}, \mathsf{evalEspP}_p, \mathsf{compPref}_p \rangle \rangle$. Como se mostrará

en este ejemplo, el servicio Υ_p podrá, a través del intérprete $\mathbb{I}_p$, realizar recomendaciones sobre películas de interés que un usuario le consulte. Por su parte, un usuario, considerando el *OCP* priorizado $\circledast$ (explicado en Definición 7.4) y las implementaciones $\mathsf{impC}(\mathsf{prioR})$ y $\mathsf{impC}(\mathsf{esp})$ de los criterios $\succsim_{\mathsf{prioR}}$ (introducido en Definición 4.27) y $\succsim_{\mathsf{esp}}$ (explicado en Definición 4.27), respectivamente, podrá expresar sus preferencias con respecto a la información del dominio representado. En particular este servicio responderá consultas realizadas al programa DeLP $\mathcal{P}_m$= (Π_m, Δ_m) que se detalla a continuación.

$$\Pi_m = \left\{ \begin{array}{l} prefers\_by\_genreAndActor(U, M) \leftarrow popular\_movies(M) \\ prefers\_by\_rating(U, M) \leftarrow \sim popular\_movies(M) \end{array} \right\}$$

$$\Delta_m = \left\{ \begin{array}{l} recommend(M, U) \prec good\_movie(M) \\ \sim recommend(M, U) \prec \sim good\_movie(M) \\ recommend(M, U) \prec likes\_by\_genre(M, U) \\ recommend(M, U) \prec likes\_by\_actor(M, U) \\ \sim recommend(M, U) \prec \sim likes\_by\_genre(M, U) \\ \sim recommend(M, U) \prec \sim likes\_by\_actor(M, U) \\ good\_movie(M) \prec avg\_rating(R, M), R > 3.8 \\ \sim good\_movie(M) \prec avg\_rating(R, M), R < 3.8 \\ likes\_by\_genre(M, U) \prec top\_genre(U, G), genre(M, G) \\ \sim likes\_by\_genre(M, U) \prec bottom\_genre(U, G), genre(M, G) \\ likes\_by\_actor(M, U) \prec top\_actor(U, A), leads\_in(M, A) \\ \sim likes\_by\_actor(M, U) \prec bottom\_actor(U, A), leads\_in(M, A) \end{array} \right\}$$

Cada una de las reglas estrictas en Π_m se interpretan de la siguiente forma; la primer regla dice que un usuario prefiere evaluar una película teniendo en cuenta el género y el actor de la misma si es el caso que es una película popular. Mientras que la segunda regla dice que si no es una película popular, entonces el usuario preferirá evaluar la calificación de la misma. Cabe destacar que el conjunto de hechos es información que supuestamente se obtiene dinámicamente a partir de diferentes repositorios de información, para cada consulta en particular. Por otra parte, el conjunto Δ_m contiene varias reglas rebatibles. En particular, las últimas 6 reglas corresponden a las las reglas r_1, r_2, r_3, r_4, r_5 y r_6 ilustradas en el cuadro de la Figura 7.2.

Note que considerando los criterios implementados en Υ_p un usuario puede declarar las preferencias que el sistema podría considerar al momento de dar una recomendación. Para

esto, el usuario podría construir *exps-combs* tales como las que se muestran a continuación.

- $\mathcal{E}^*_{\mathsf{iCuant}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{iCuant}})$ donde

  $$\mathsf{D}_{\mathsf{iCuant}} = \left\{ \begin{array}{l} (es\_mejorR(r_1, r_6)), \\ (es\_mejorR(r_1, r_4)), \\ (es\_mejorR(r_2, r_3)), \\ \end{array} \right\}$$

- $\mathcal{E}^*_{\mathsf{iCual}} = (\mathsf{prioR}, \mathsf{D}_{\mathsf{iCual}})$ tal que

  $$\mathsf{D}_{\mathsf{iCual}} = \left\{ \begin{array}{l} (es\_mejorR(r_4, r_1)) \\ \end{array} \right\}$$

- $\mathcal{E}^*_{\mathsf{esp}} = (\mathsf{esp}, \mathsf{D}_{\mathsf{esp}})$ tal que $\mathsf{D}_{\mathsf{esp}} = \{\}$

- $\mathcal{E}^*_1 = [\{prefers\_by\_rating(U, M)\} : \circledast(\mathcal{E}^*_{\mathsf{iCuant}}, \mathcal{E}^*_{\mathsf{iCual}}); \mathcal{E}^*_{\mathsf{esp}}]$

- $\mathcal{E}^*_2 = [\{prefers\_by\_genreAndActor(U, M)\} : \circledast(\mathcal{E}^*_{\mathsf{iCual}}, \mathcal{E}^*_{\mathsf{iCuant}}); \mathcal{E}^*_{\mathsf{esp}}]$

Intuitivamente, $\mathcal{E}^*_{\mathsf{iCuant}}$ es una expresión que prioriza información cuantitativa (información acerca de las calificaciones de las películas), mientras que $\mathcal{E}^*_{\mathsf{iCual}}$, por lo contrario, prioriza información de tipo cualitativa (información acerca del género y los actores de las películas). Por otro lado, $\mathcal{E}^*_{\mathsf{esp}}$ expresa preferencias basadas en información más específica independientemente si dicha información es de carácter cuantitativa o cualitativa.

A diferencia de las expresiones anteriores, $\mathcal{E}^*_1$ y $\mathcal{E}^*_2$ describen una combinación anidada de preferencias. En particular, $\mathcal{E}^*_1$ se lee como sigue: "si existe una derivación estricta para $prefers\_by\_rating(U, M)$, entonces se realiza la operación de combinación de preferencias que utilizará la especificación de preferencia $\mathcal{E}^*_{\mathsf{iCual}}$ únicamente si el criterio indicado en $\mathcal{E}^*_{\mathsf{iCuant}}$ no se puede aplicar, en caso contrario se utiliza la especificación de preferencia $\mathcal{E}^*_{\mathsf{esp}}$". Por otra parte, $\mathcal{E}^*_2$ se interpreta de la siguiente manera: "si $prefers\_by\_genreAndActor(U, M)$ se deriva estrictamente, entonces se aplica la operación de combinación que utiliza la especificación de preferencia $\mathcal{E}^*_{\mathsf{iCuant}}$ únicamente si el criterio indicado en $\mathcal{E}^*_{\mathsf{iCual}}$ no se puede aplicar, en caso contrario se utiliza la especificación de preferencia $\mathcal{E}^*_{\mathsf{esp}}$". Note que a partir de las expresiones anteriores es posible seleccionar qué criterio de preferencia utilizar dependiendo de la información disponible en el programa $\mathcal{P}_m$.

Suponga ahora que un usuario ($juan$) realiza consultas al sistema para saber si dicho sistema le recomienda ver determinadas películas que él especifica. Considerando la información detallada hasta el momento, el *SRPComb* Υ_p podría recibir la siguiente *CPComb*.

$$\mathcal{CC}_{juan_1} = [\mathcal{E}_1^*, \mathcal{P}_m{}', recommend(the\_avengers, juan)]$$

$$\text{donde } \mathcal{P}_m{}' = \mathcal{P} \cup \left\{ \begin{array}{l} popular\_movies(the\_avengers), \\ avg\_rating(8.1, the\_avengers), \\ bottom\_genre(juan, action), \\ genre(the\_avengers, action), \\ bottom\_actor(juan, mark\_ruffalo), \\ leads\_in(the\_avengers, mark\_ruffalo) \end{array} \right\}$$

El proceso para responder $\mathcal{CC}_{juan_1}$ primero evaluará $\mathcal{E}_1^*$ y luego realizará el análisis de dialéctica correspondiente. Dado que $prefers\_by\_rating(juan, the\_avengers)$ se deriva estrictamente desde $\mathcal{P}_m{}'$, se tiene que la *EPref* seleccionada es $\circledast(\mathcal{E}^*_{\mathsf{iCuant}}, \mathcal{E}^*_{\mathsf{iCual}})$. Considerando esta *EPref*, en la Figura 7.6-(a) se muestra que existe un único argumento no derrotado a favor de recomendar la película $the\_avengers$. La Figura 7.4 ilustra gráficamente los argumentos tenidos en cuenta en el proceso de argumentación llevado a cabo para la consulta $\mathcal{CC}_{juan_1}$. Obsérvese que los argumentos basados en los postulados 4 (b) y 6 (c) atacan al argumento basado en el postulado 1 (a), no obstante, éstos no son preferidos con respecto a $\circledast(\mathcal{E}^*_{\mathsf{iCuant}}, \mathcal{E}^*_{\mathsf{iCual}})$. Por lo tanto, el sistema recomendará a $juan$ la película $the\_avengers$.

Considere otra consulta del usuario $juan$, ahora por la película $jurassic\_shark$. Teniendo en cuenta la *exp-comb* $\mathcal{E}_2^*$, Υ_p podría recibir la consulta que sigue.

$$\mathcal{CC}_{juan_2} = [\mathcal{E}_2^*, \mathcal{P}_m{}', recommend(jurassic\_shark, juan)]$$

$$\text{donde } \mathcal{P}_m{}' = \mathcal{P}_m \cup \left\{ \begin{array}{l} \sim popular\_movies(jurassic\_shark), \\ avg\_rating(2.1, jurassic\_shark), \\ top\_genre(juan, action), \\ genre(jurassic\_shark, action), \\ bottom\_actor(juan, mark\_ruffalo), \\ leads\_in(jurassic\_shark, mark\_ruffalo) \end{array} \right\}$$

Teniendo en cuenta lo anterior, $prefers\_by\_genreAndActor(juan, jurassic\_shark)$ tiene una derivación estricta desde $\mathcal{P}_m{}'$. Por lo tanto, $\circledast(\mathcal{E}^*_{\mathsf{iCual}}, \mathcal{E}^*_{\mathsf{iCuant}})$ será la *EPref* elegida a partir de la evaluación de la *exp-comb* $\mathcal{E}_2^*$. Considere los argumentos que se obtienen desde el programa $\mathcal{P}_m{}'$, a partir de la la consulta $recommend(jurassic\_shark, juan)$, los cuales se ilustran en la Figura 7.5. El único argumento que soporta la recomendación de

la película *jurassic_shark* es el argumento basado en el postulado 3 (b), el cual a su vez es atacado por dos argumentos, uno basado en el postulado 2 (a) y el otro basado en el postula 6 (c). Como el argumento (b) es derrotado por estos dos argumentos, y siendo que no existe otro argumento que soporte la conclusión $recommend(jurassic\_shark, juan)$, entonces se busca un argumento para su complemento. Note que el argumento basado en el postulado 2 (a) tiene un único argumento que lo ataca que es el argumento (b), no obstante el primero es preferido sobre el segundo siendo en consecuencia un derrotador propio de dicho argumento. Como existe un argumento no derrotado soportando $\sim recommend(jurassic\_shark, juan)$, la respuesta del sistema será no recomendar la película *jurassic_shark* a *juan*. La Figura 7.6-(b) muestra los árboles de dialéctica generados en el proceso de argumentación llevado a cabo para la consulta $\mathcal{CC}_{juan_2}$.

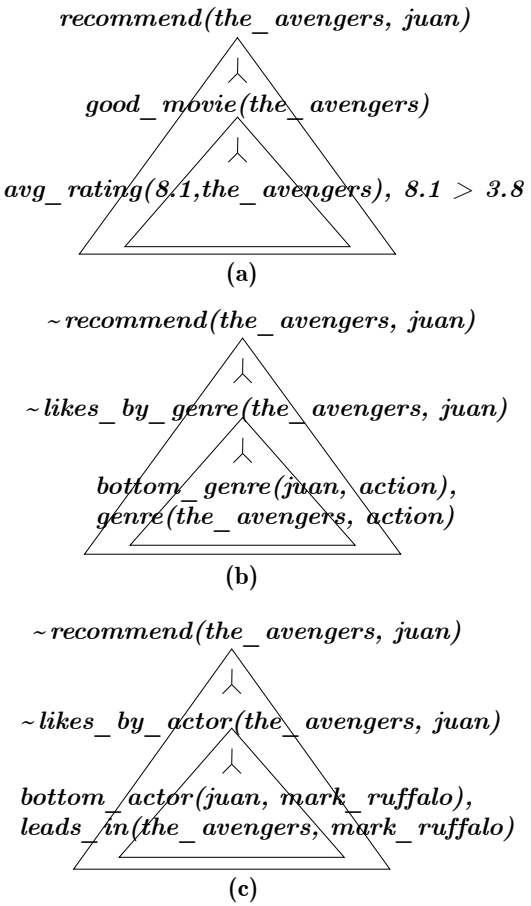

Figura 7.4: Argumentos a favor (a) y en contra ((b) y (c)) de recomendar la película *the_avengers*.

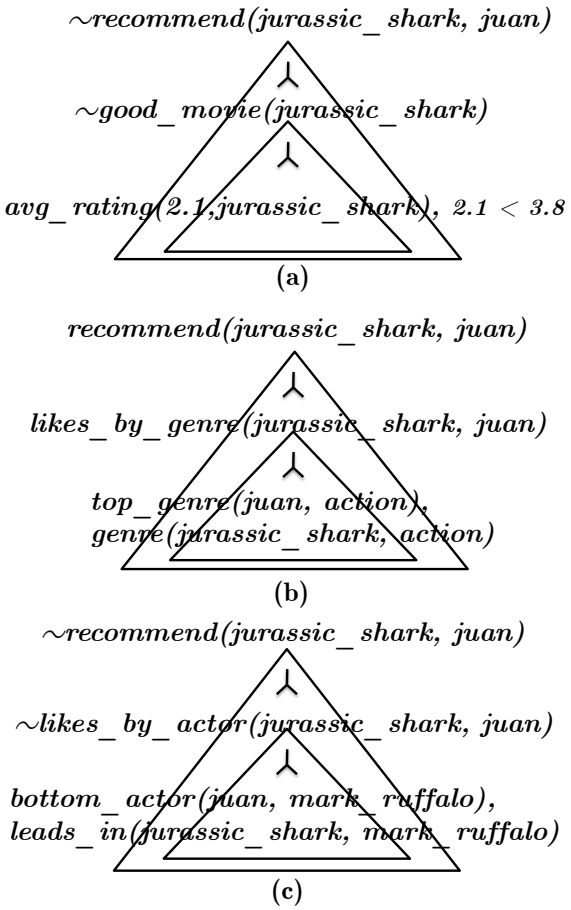

Figura 7.5: Argumentos a favor (b) y en contra ((a) y (c)) de recomendar la película *jurassic_shark*.

Como conclusión se puede observar que el proceso de razonamiento que dio respuesta a las consultas $\mathcal{CC}_{juan_1}$ y $\mathcal{CC}_{juan_2}$ priorizó diferentes tipos de información para cada consulta en particular. Esto se debe básicamente al hecho de que se utilizaron distintas *EPrefs* para comparar la información contradictoria que aparecía en dicho proceso. A partir de la consulta $\mathcal{CC}_{juan_1}$, la evaluación de $\mathcal{E}_1^*$ dio como resultado la especificación de preferencia $\circledast(\mathcal{E}^*_{\mathsf{iCuant}}, \mathcal{E}^*_{\mathsf{iCual}})$ que prioriza la especificación $\mathcal{E}^*_{\mathsf{iCuant}}$ la cual a su vez da relevancia a la información cuantitativa sobre la cualitativa. Por otra parte, considerando la consulta $\mathcal{CC}_{juan_2}$, el resultado de evaluar $\mathcal{E}_2^*$ ha sido $\circledast(\mathcal{E}^*_{\mathsf{iCual}}, \mathcal{E}^*_{\mathsf{iCuant}})$. Esta *EPref* en particular prioriza la especificación $\mathcal{E}^*_{\mathsf{iCual}}$, por lo tanto prefiere información basada en aspectos cualitativos. Teniendo en cuenta lo mencionado, en la primer consulta se utilizó un enfoque de comparación de argumentos que prioriza información cuantitativa, mientras que en la

segunda consulta se utilizó un enfoque que prioriza información cualitativa.

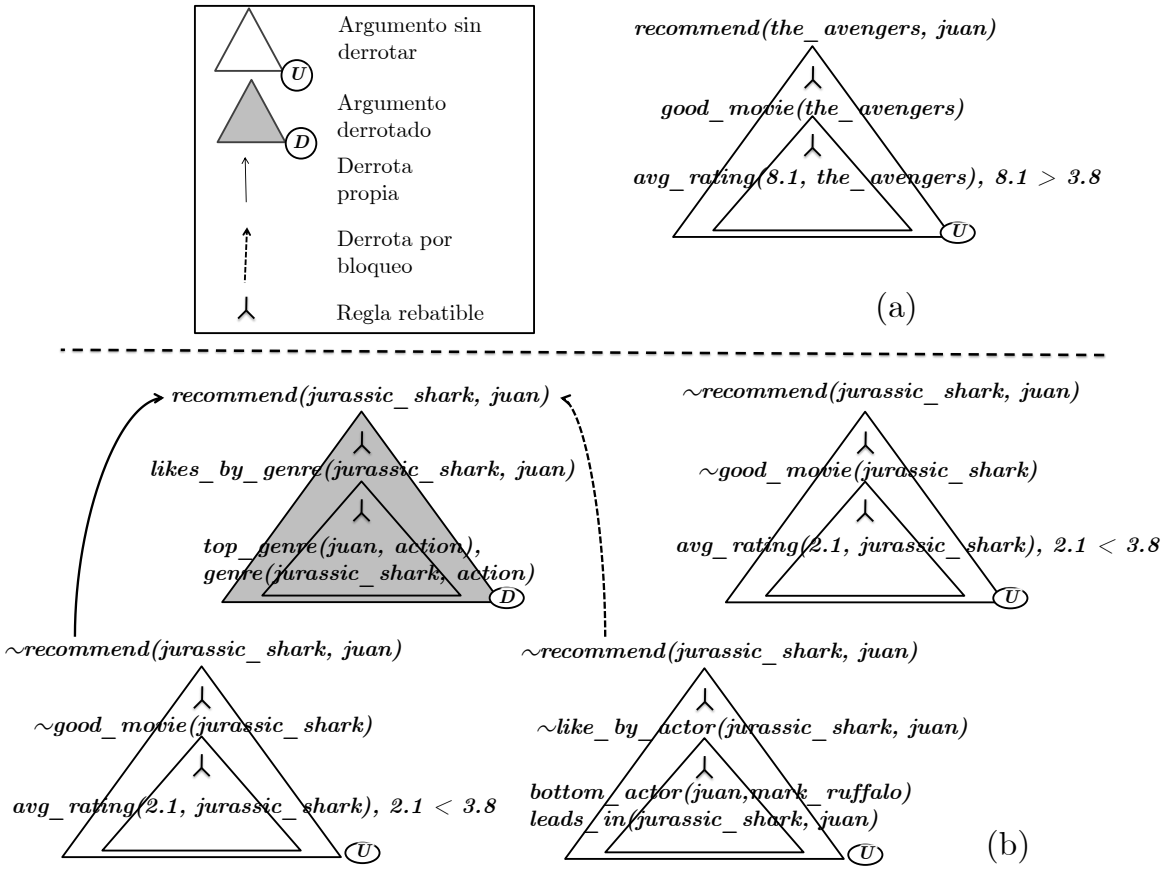

Figura 7.6: Árboles de dialéctica para responder $\mathcal{CC}_{juan_1}$ (a) y $\mathcal{CC}_{juan_2}$ (b).

Finalmente, note que en este ejemplo se presentaron dos *CPCombs* totalmente diferentes para un mismo *SRPComb*. La Figura 7.6 ilustra los distintos árboles generados a partir del análisis de dialéctica llevado a cabo para cada consulta en cuestión. En general, se puede observar que tanto los argumentos involucrados (ver Figura 7.4 y 7.5), en consecuencia las relaciones de derrota variaron para cada análisis en particular.

7.5. Conclusión

En este capítulo se mostró un nuevo tipo de consulta denominada *Consulta basada en Preferencias Combinadas* (*CPComb*). Estas consultas se caracterizan por estar constituidas por una *expresión de preferencias combinadas* (*exp-comb*) la cual además de usar expresiones condicionales permite utilizar *operadores de combinación de preferencias*. Es-

tos operadores se encuentran definidos en el servicio de razonamiento que será consultado y proveen al usuario de diferentes mecanismos para combinar los criterios implementados en dicho servicio.

Aquellos servicios de razonamiento particulares que están programados de manera tal que pueden responder estas nuevas consultas se los denomina *Servicios de Razonamiento Rebatible basado en Preferencias Combinadas* (*SRPCombs*). Al igual que en el Capítulo 6, un *SRPComb* que recibe una *CPComb* deberá evaluar la expresión (*exp-comb*) que se encuentra en dicha consulta. Para esto, los *SRPCombs* emplean la función de evaluación evalE^* la cual devuelve una *especificación de preferencia* que finalmente será utilizada por el *módulo de comparación de argumentos extendido* para computar las preferencias entre argumentos que se requieran.

En la Sección 7.3 se presentaron varios resultados. Dicha sección se dividió en dos subsecciones. En la primer subsección se estudio un aspecto poco deseable que está relacionado a los criterios de preferencias y a los operadores de combinación de preferencias que es la indecisión por incomparabilidad. Dentro de los resultados mostrados en esta sección se estableció que tanto la relación de preferencia asociada a los criterios de preferencias como la relación que resulta de una operación de combinación de preferencias deberían de ser no vacías y cumplir la propiedad de irreflexividad y asimetría. Finalmente, en la Sección 7.3.2 se mostró qué criterios y operadores de los definidos en las secciones 4.4 y 7.1 respectivamente mantienen dichas propiedades.

Con el objetivo de mostrar la flexibilidad del enfoque propuesto en este capítulo, en la Sección 7.4 se presenta un ejemplo donde integramos nuestra propuesta al trabajo realizado en [BBD$^+$14]. En particular, se muestra la aplicación de los conceptos descriptos en este capítulo a un dominio donde el escenario consiste en la recomendación de películas a partir de la información obtenida de diferentes bases de datos.

Capítulo 8

Conclusiones y Trabajos Futuros

En los últimos años, la argumentación rebatible ha realizado un importante aporte a la Inteligencia Artificial, hecho que se refleja en el creciente número de aplicaciones del mundo real que la incluyen como formalización del razonamiento del sentido común. En este sentido, los sistemas argumentativos proponen una formalización de este tipo de razonamiento justamente utilizando como mecanismo de inferencia la argumentación rebatible. Intuitivamente, la argumentación rebatible provee formas de confrontar declaraciones contradictorias para determinar si alguna afirmación puede ser aceptada o rechazada. Para obtener una respuesta, el proceso de razonamiento argumentativo lleva a cabo una serie de etapas [RS09]. Una etapa muy importante es la comparación de argumentos en conflicto para decidir qué argumento prevalece; ésto requiere introducir un criterio de preferencia entre argumentos que haga frente a esta situación.

Distintos sistemas argumentativos adoptan diferentes criterios de preferencia entre argumentos. Uno de los criterios más populares en Inteligencia Artificial es el criterio de especificidad, que prefiere argumentos con información más específica. Sin embargo, existen diferentes posturas respecto a este criterio de preferencia. Algunos investigadores [Vre91, Pol95, PS96a], por ejemplo, consideran que especificidad no constituiría un criterio general del razonamiento de sentido común, sino simplemente un criterio más que podría adoptarse. Otros sostienen que no existen criterios generales, independientes del dominio, dado que éstos conllevan a un estado de indecisión en la mayoría de los casos, y que la información acerca del dominio representa la herramienta más importante para la evaluación de argumentos en conflicto [Kon88, Vre91, SL92, FEGS08, Wak11]. Teniendo en cuenta esto, varios sistemas de argumentación se encuentran parametrizados respecto al criterio de preferencia, esperando se utilice el criterio que mejor se adecue al dominio

representado. Por otra parte, otros investigadores [PS96b] sostienen que los criterios son parte de la teoría de prueba, y por lo tanto sujetos a discusión, por lo que los sistemas argumentativos deberían permitir la construcción de argumentos acerca del criterio.

La Programación Lógica Rebatible (DeLP) es un sistema argumentativo estructurado que extiende de la Programación en Lógica y que permite la representación de conocimiento tentativo, incierto y potencialmente inconsistente. En DeLP el criterio de preferencia es un parámetro del sistema, por lo que la comparación de argumentos se maneja de forma modular. En este sentido, en la literatura existen formalismos basados en DeLP que utilizan un único criterio embebido en el sistema [FEGS08, CGS10, TGG$^+$12, BCM13], mientras que otros trabajos utilizan varios criterios [GMP12, BBD$^+$14], pero proponen una combinación fija de esos criterios. A pesar de la importancia que tiene la comparación de argumentos en el proceso de argumentación, los formalismos desarrollados hasta el momento no han considerado mecanismos computacionales que permitan cambiar dinámicamente el criterio de preferencia. Esto motivó el estudio de diferentes herramientas concretas que permitan tratar el manejo de múltiples criterios en aquellos formalismos cuyo mecanismo de inferencia sea DeLP, a fin de poder seleccionar y cambiar de criterio de una manera natural y acorde a las preferencias o necesidades del usuario. A continuación se sintetizan las conclusiones de nuestra investigación describiendo los principales resultados obtenidos.

En el Capítulo 5 de esta tesis se presentó un servicio de razonamiento basado en la programación lógica rebatible, denominado Servicio de Razonamiento Rebatible basado en Preferencias (*SRPref*), que puede cambiar de criterio de preferencia para cada consulta DeLP recibida. Estos servicios están constituidos por un intérprete DeLP el cual establece el mecanismo de inferencia utilizado para responder consultas, como así también de un módulo encargado de resolver las preferencias entre argumentos que dicho intérprete requiere. Como herramienta de interacción para consultar estos servicios se introdujo la Consulta basada en Preferencias (*CPref*). Esta herramienta, junto con el programa DeLP y la consulta DeLP, permite especificar de forma declarativa el criterio que será utilizado por el *SRPref* que recibe la consulta. Cabe destacar que para el diseño de los *SRPrefs* se tuvo en cuenta que pueden existir varios criterios disponibles, es decir que al momento de definir el servicio se podrá tener en cuenta el uso de diferentes criterios apropiados para un dominio de aplicación en particular.

En el Capítulo 6 se presentó el concepto de Servicio de Razonamiento Rebatible basado en Preferencias Condicionales (*SRPCond*) y se introdujo un tipo especial de consulta pa-

ra estos servicios, denominada Consulta basada en Preferencias Condicionales (*CPCond*), que le permite a un *SRPCond* seleccionar el criterio que deberá utilizar dependiendo de ciertas condiciones. Para esto la *CPCond* incorpora una expresión de preferencia condicional (*exp-cond*). Para la selección de un criterio se define la noción de *guarda*; las guardas forman parte de una *exp-cond*. Formalmente una guarda se definió como un conjunto de literales, y se dice que un programa DeLP dado la satisface, si todos esos literales son derivados estrictamente del programa. Por lo tanto, un criterio será seleccionado dependiendo de qué guardas de las incluidas en una *exp-cond* son satisfechas por el programa DeLP consultado. El uso de las guardas permite orientar y justificar la elección de un criterio en particular. Dada la importancia de las *exps-conds* como herramienta computacional concreta para guiar la selección de un criterio, en la Sección 6.2 se presenta una representación de árbol para *exps-conds* con el fin de proporcionar una forma de analizar varias propiedades de estas expresiones. Estas propiedades son útiles para identificar cuando una expresión puede ser optimizada y de esta manera evitar la computación de literales redundantes, y además caracterizar cuando ciertos caminos en la expresión no serán transitables. Estas propiedades son de especial interés en esta tesis por que permiten construir expresiones válidas; es decir, expresiones que mantienen relaciones coherentes entre las guardas que justifican la elección de un criterio en particular.

Una de las motivaciones importantes en esta tesis ha sido introducir la posibilidad de comparar argumentos considerando varios criterios al mismo tiempo. Siguiendo esta idea en el Capítulo 7 se presenta la Consulta basada en Preferencias Combinadas (*CPComb*). Una *CPComb* permite que se pueda especificar el uso de más de un criterio a partir de una expresión especial construida mediante los operadores de combinación de preferencias (*OCPs*) que el servicio consultado dispone. Entonces, los servicios capaces de resolver estas consultas, denominados Servicios de Razonamiento Rebatible basado en Preferencias Combinadas, podrán contar con operadores para combinar criterios. Por otra parte, en la Sección 7.3 se presentaron algunos resultados obtenidos. En primer lugar se analizó una situación habitual en el proceso de argumentación que es la indecisión por incomparabilidad. Como resultado se estableció que la relación de preferencia definida por los criterios de preferencias como así también la relación que resulta de las operaciones de combinación de preferencias deberían de ser no vacías y cumplir con las propiedades de irreflexividad y asimetría. La sección 7.3.2 muestra qué los criterios y operadores que fueron introducidos como ejemplo a lo largo de la tesis cumplen las propiedades descriptas previamente.

Si bien gran parte de los trabajos existentes en la literatura se concentran en el estudio

del concepto de preferencias en el contexto de marcos argumentativos abstractos como el propuesto en [ACB96, AC98], el estudio de los criterios de comparación de argumentos que se utilizan en los sistemas argumentativos estructurados constituye un prometedor tópico de investigación para la comunidad de argumentación. En particular, contar con un conjunto de herramientas computacionales que permitan el manejo de multiplicidad de criterios incrementa las capacidades de razonamiento efectivo en este tipo de sistemas. En consecuencia, esto contribuye al uso de argumentación en dominios de aplicación en donde resulta imprescindible la consideración de mecanismos que permitan ajustar a una determinada situación o contexto particular el criterio de preferencia que se vaya a utilizar.

Trabajo a Futuro

Como trabajo futuro se implementarán los desarrollos de esta tesis para poder ejercitar los formalismos y desarrollar nuevas alternativas a partir de las aplicaciones. A partir de los trabajos desarrollados en esta tesis se abren varias líneas de investigación sobre las cuales se planea seguir trabajando.

- Se buscará tratar con la formalización de otras herramientas de interacción. Ésto permitirá definir nuevas consultas basadas en preferencias que cumplan con nuevos requisitos del usuario con respecto a sus necesidades o preferencias. Por ejemplo, un usuario podría querer realizar diferentes consultas DeLP simultáneamente utilizando el mismo criterio y el mismo programa DeLP. En este sentido tener un tipo de consulta que cumpla con estos requisitos mejoraría los tiempos de respuesta de los servicios de razonamiento propuestos en esta tesis.

- A partir del Capítulo 3, se observó que la forma en la que es tratada la comparación de argumentos por los formalismos no siempre es la misma. Algunos sistemas utilizan un enfoque modular [SL92, GS04], otros argumentan sobre el criterio de preferencia [PS97], mientras que otros utilizan un criterio fijo [Wak10]. En este sentido, un tópico interesante es estudiar si es posible construir un marco general para sistemas argumentativos estructurados que permita el manejo de multiplicidad de criterios. Un primer paso para lograr esto, es analizar la posibilidad de definir servicios de razonamiento en donde el mecanismo de razonamiento sea el de otro sistema argumentativo estructurado, y no el de DeLP. Para esto será necesario diseñar un

marco lo suficientemente general como para ser instanciado por diferentes sistemas de este tipo en particular.

- En la literatura existen numerosos trabajos sobre marcos argumentativos abstractos basados en preferencias [KvdTW06, AV10, AV11], sin embargo pocos abordan la multiplicidad de criterios [APP00]. Otro trabajo a futuro es definir un marco argumentativo abstracto que incluya las expresiones de preferencias combinadas en su formalismo y permita el manejo de varias relaciones de preferencias. Contar con un marco de este tipo permitirá caracterizar varias propiedades interesantes para los servicios de razonamiento que fueron desarrollados a lo largo de esta tesis.

- Un tema interesante para analizar consiste en el diseño e implementación de una arquitectura de agentes basada en el esquema de alguno de los servicios de razonamiento propuestos. En tal sentido es necesario integrar a un servicio de razonamiento los componentes que se encarguen de la comunicación y el comportamiento. De esta manera es posible construir una arquitectura que permita modelar tanto la interacción entre agentes como la percepción del ambiente.

Apéndice A

Tabla de Acrónimos

La siguiente tabla incluye los acrónimos utilizados en esta tesis, listados en orden alfabético.

Acrónimo	**Expansión en Español**	**Ubicación**
AS	Argumentación basada en Suposiciones	Sección 3.8
ASP	Programación con Conjuntos Respuesta (*Answer Set Programming*)	Sección 2.2
asprin	Manejo de Preferencias en Programación con Conjuntos Respuesta (*ASP for Preference Handling*)	Sección 2.2.2
CPComb	Consulta basada en Preferencias Combinadas	Definición 7.9
CPCond	Consulta basada en Preferencias Condicionales	Definición 6.4
CPref	Consulta basada en Preferencias	Definición 5.5
DeLP	Programación Lógica Rebatible (*Defeasible Logic Programming*)	Capítulo 4
DeLP-server	Servidor de Razonamiento en Programación Lógica Rebatible	Capítulo 4
ECrit	Especificación de Criterio	Definición 5.4
EPref	Especificación de Preferencia	Definición 7.5
exp-comb	Expresión de Preferencias Combinadas	Definición 7.7
exp-cond	Expresión de Preferencia Condicional	Definición 6.2
MAP	Marco de Argumentación basado en Preferencias	Sección 3.6

MAPC	Marco de Argumentación basado en Preferencias Contextuales	Sección 3.7
MCA	Módulo de Comparación de Argumentos	Definición 5.2
MEC	Módulo Extendido de Comparación de Argumentos	Definición 7.10
PLE	Programa Lógico Extendido	Sección 3.6
PLP	Programa Lógico Priorizado	Sección 3.6
p_*AS*	Marco de Argumentación basada en Suposiciones equipado con Preferencias	Sección 3.8
OCP	Operador de Combinación de Preferencias	Sección 7.1
SAE	Sistema Argumentativo Estructurado	Sección 3.1
SRD	Servicio de Razonamiento para Recomendación de Diagnósticos	Sección 5.5
SRPComb	Servicio de Razonamiento Rebatible basado en Preferencias Combinadas	Definición 7.11
SRPCond	Servicio de Razonamiento Rebatible basado en Preferencias Condicionales	Definición 6.5
SRPref	Servicio de Razonamiento Rebatible basado en Preferencias	Definición 5.3
SRR	Servicio de Razonamiento para Recomendación de Reportes	Sección 5.6

Bibliografía

[AC98] Amgoud, L., and Cayrol, C. On the acceptability of arguments in preference-based argumentation. In *UAI* (1998), pp. 1–7.

[AC02] Amgoud, L., and Cayrol, C. A reasoning model based on the production of acceptable arguments. *Ann. Math. Artif. Intell. 34*, 1-3 (2002), 197–215.

[ACB96] Amgoud, L., Cayrol, C., and Berre, D. L. Comparing arguments using preference ordering for argument-based reasoning. In *Eigth International Conference on Tools with Artificial Intelligence, ICTAI '96, Toulouse, France, November 16-19, 1996* (1996), pp. 400–403.

[ACGS08] Alsinet, T., Chesñevar, C. I., Godo, L., and Simari, G. R. A logic programming framework for possibilistic argumentation: Formalization and logical properties. *Fuzzy Sets and Systems 159*, 10 (2008), 1208–1228.

[AK07] Amgoud, L., and Kaci, S. An argumentation framework for merging conflicting knowledge bases. *Int. J. Approx. Reasoning 45*, 2 (2007), 321–340.

[AMB00] Antoniou, G., Maher, M. J., and Billington, D. Defeasible logic versus logic programming without negation as failure. *Journal of Logic Programming 42*, 1 (2000), 47–57.

[APM00] Amgoud, L., Parsons, S., and Maudet, N. Arguments, dialogue, and negotiation. In *ECAI 2000, Proceedings of the 14th European Conference on Artificial Intelligence, Berlin, Germany, August 20-25, 2000* (2000), pp. 338–342.

[APP00] Amgoud, L., Parsons, S., and Perrussel, L. An argumentation framework based on contextual preferences. In *Proceedings of the 3rd Interna-*

tional Conference on Formal and Applied Practical Reasoning, FAPR '00, 2000 (2000), pp. 59–67.

[AV10] Amgoud, L., and Vesic, S. On the role of preferences in argumentation frameworks. In *22nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2010, Arras, France, 27-29 October 2010 - Volume 1* (2010), pp. 219–222.

[AV11] Amgoud, L., and Vesic, S. Two roles of preferences in argumentation frameworks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 11th European Conference, ECSQARU 2011, Belfast, UK, June 29-July 1, 2011. Proceedings* (2011), pp. 86–97.

[AW00] Agrawal, R., and Wimmers, E. L. A framework for expressing and combining preferences. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA.* (2000), pp. 297–306.

[BBD+14] Briguez, C. E., Budán, M. C., Deagustini, C. A. D., Maguitman, A. G., Capobianco, M., and Simari, G. R. Argument-based mixed recommenders and their application to movie suggestion. *Expert Systems with Applications 41*, 14 (2014), 6467–6482.

[BCD07] Bench-Capon, T. J. M., and Dunne, P. E. Argumentation in artificial intelligence. *Artif. Intell. 171*, 10-15 (2007), 619–641.

[BCM13] Briguez, C. E., Capobianco, M., and Maguitman, A. G. A theoretical framework for trust-based news recommender systems and its implementation using defeasible argumentation. *International Journal on Artificial Intelligence Tools 22*, 4 (2013).

[BCVF06] Brain, M., Crick, T., Vos, M. D., and Fitch, J. P. TOAST: applying answer set programming to superoptimisation. In *Logic Programming, 22nd International Conference, ICLP 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings* (2006), pp. 270–284.

[BD04] Brafman, R., and Domshlak, C. Database preference queries revisited. Tech. rep., Cornell University, 2004.

[BDRS15] Brewka, G., Delgrande, J. P., Romero, J., and Schaub, T. asprin: Customizing answer set preferences without a headache. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.* (2015), pp. 1467–1474.

[BGG05] Baroni, P., Giacomin, M., and Guida, G. Scc-recursiveness: a general schema for argumentation semantics. *Artif. Intell. 168*, 1-2 (2005), 162–210.

[BH01] Besnard, P., and Hunter, A. A logic-based theory of deductive arguments. *Artif. Intell. 128*, 1-2 (2001), 203–235.

[BH08] Besnard, P., and Hunter, A. *Elements of argumentation*, vol. 47. MIT press Cambridge, 2008.

[BH09] Black, E., and Hunter, A. An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems 19*, 2 (2009), 173–209.

[BKS01] Börzsönyi, S., Kossmann, D., and Stocker, K. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany* (2001), pp. 421–430.

[BNT03] Brewka, G., Niemelä, I., and Truszczynski, M. Answer set optimization. In *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003* (2003), pp. 867–872.

[Bre04a] Brewka, G. Complex preferences for answer set optimization. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004* (2004), pp. 213–223.

[Bre04b] Brewka, G. A rank based description language for qualitative preferences. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004* (2004), pp. 303–307.

[CCS05] Capobianco, M., Chesñevar, C. I., and Simari, G. R. Argumentation and the dynamics of warranted beliefs in changing environments. *Autonomous Agents and Multi-Agent Systems 11*, 2 (September 2005), 127–151.

[CFP10] Costantini, S., Formisano, A., and Petturiti, D. Extending and implementing RASP. *Fundam. Inform. 105*, 1-2 (2010), 1–33.

[CGFZ13] Costantini, S., Gasperis, G. D., Florio, N., and Zuppella, C. An asp-based system for preference handling and planning. In *Proceedings of the 28th Italian Conference on Computational Logic, Catania, Italy, September 25-27, 2013.* (2013), pp. 253–257.

[CGS10] Cohen, A., García, A. J., and Simari, G. R. Extending DeLP with attack and support for defeasible rules. In *IBERAMIA* (2010), pp. 90–99.

[Cho02] Chomicki, J. Querying with intrinsic preferences. In *Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27, Proceedings* (2002), pp. 34–51.

[Cho03] Chomicki, J. Preference formulas in relational queries. *ACM Trans. Database Syst. 28*, 4 (2003), 427–466.

[Cia07] Ciaccia, P. Querying databases with incomplete cp-nets. In *Multidisciplinary Workshop on Advances in Preference Handling (M-PREF 2007)* (2007), Citeseer.

[CML00] Chesñevar, C. I., Maguitman, A. G., and Loui, R. P. Logical models of argument. *ACM Computing Surveys 32*, 4 (2000), 337–383.

[CMS06] Chesñevar, C. I., Maguitman, A. G., and Simari, G. R. Argument-based critics and recommenders: A qualitative perspective on user support systems. *Data Knowl. Eng. 59*, 2 (2006), 293–319.

[CMS07] Chesñevar, C. I., Maguitman, A. G., and Simari, G. R. Recommender system technologies based on argumentation 1. In *Emerging Artificial Intelligence Applications in Computer Engineering.* 2007, pp. 50–73.

[CRS92] Cayrol, C., Royer, V., and Saurel, C. Management of preferences in assumption-based reasoning. In *IPMU* (1992), pp. 13–22.

[CS09] Capobianco, M., and Simari, G. R. A proposal for making argumentation computationally capable of handling large repositories of uncertain data. In *SUM* (2009), pp. 95–110.

[Dal88] Dalal, M. Investigations into a theory of knowledge base revision: preliminary report. In *Proceedings of the Seventh National Conference on Artificial Intelligence* (1988), vol. 2, Citeseer, pp. 475–479.

[Dav89] Davis, R. *Truth, Deduction, and Computation: Logic and semantics for computer science*. WH Freeman & Co., 1989.

[DDG+12] Deagustini, C. A. D., Dalibón, S. E. F., Gottifredi, S., Falappa, M. A., and Simari, G. R. Consistent query answering using relational databases through argumentation. In *Database and Expert Systems Applications - 23rd International Conference, DEXA 2012, Vienna, Austria, September 3-6, 2012. Proceedings, Part II* (2012), pp. 1–15.

[DFDG+13] Deagustini, C. A. D., Fulladoza Dalibón, S. E., Gottifredi, S., Falappa, M. A., Chesñevar, C. I., and Simari, G. R. Relational databases as a massive information source for defeasible argumentation. *Knowledge-Based Systems 51* (2013), 91–109.

[DHKP11] Domshlak, C., Hüllermeier, E., Kaci, S., and Prade, H. Preferences in AI: An overview. *Artif. Intell. 175*, 7-8 (2011), 1037–1052.

[DKT06] Dung, P. M., Kowalski, R. A., and Toni, F. Dialectic proof procedures for assumption-based, admissible argumentation. *Artif. Intell. 170*, 2 (2006), 114–159.

[DKT09] Dung, P. M., Kowalski, R. A., and Toni, F. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*. Springer, 2009, pp. 199–218.

[DSTW04] Delgrande, J. P., Schaub, T., Tompits, H., and Wang, K. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence 20*, 2 (2004), 308–334.

[Dun93] Dung, P. M. An argumentation semantics for logic programming with explicit negation. In *Logic Programming, Proceedings of the Tenth International Conference on Logic Programming, Budapest, Hungary, June 21-25, 1993* (1993), pp. 616–630.

[Dun95] Dung, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell. 77*, 2 (1995), 321–358.

[EH95] Elvang-Gøransson, M., and Hunter, A. Argumentative logics: Reasoning with classically inconsistent information. *Data Knowl. Eng. 16*, 2 (1995), 125–145.

[Fag02] Fagin, R. Combining fuzzy information: an overview. *SIGMOD Record 31*, 2 (2002), 109–118.

[FEGS08] Ferretti, E., Errecalde, M., García, A. J., and Simari, G. R. Decision rules and arguments in defeasible decision making. In *Computational Models of Argument: Proceedings of COMMA 2008, Toulouse, France, May 28-30, 2008.* (2008), pp. 171–182.

[Fil01] Fillottrani, P. R. *Semántica para la Negación en Programas Lógicos Extendidos*. PhD thesis, Universidad Nacional del Sur, Bahía Blanca, Argentina, 2001.

[Fis70] Fishburn, P. C. Utility theory for decision making. Tech. rep., DTIC Document, 1970.

[Fis99] Fishburn, P. C. Preference structures and their numerical representations. *Theor. Comput. Sci. 217*, 2 (1999), 359–383.

[FW97] Fagin, R., and Wimmers, E. L. Incorporating user preferences in multimedia queries. In *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings* (1997), pp. 247–261.

[Gar97] García, A. J. Defeasible logic programming: Definition and implementation. Master's thesis, Computer Science Department, Universidad Nacional del Sur, Bahía Blanca, Argentina, July 1997.

[Gar00] García, A. J. *Defeasible Logic Programming: Definition, Operational Semantics and Parallelism*. PhD thesis, Universidad Nacional del Sur, Bahía Blanca, Argentina, 2000.

[GCS08] Gómez, S. A., Chesñevar, C. I., and Simari, G. R. Defeasible reasoning in web-based forms through argumentation. *International Journal of Information Technology and Decision Making 7*, 1 (2008), 71–101.

[GGS08] Gottifredi, S., García, A. J., and Simari, G. R. Defeasible knowledge and argumentative reasoning for 3APL agent programming. In *Twelfth International Workshop on Non-Monotonic Reasoning (NMR 2008)* (2008).

[GGS10] Gottifredi, S., García, A. J., and Simari, G. R. Query-based argumentation in agent programming. In *Advances in Artificial Intelligence - IBERAMIA 2010, 12th Ibero-American Conference on AI, Bahía Blanca, Argentina, November 1-5, 2010. Proceedings* (2010), pp. 284–295.

[GKS11] Gebser, M., Kaminski, R., and Schaub, T. Complex optimization in answer set programming. *TPLP 11*, 4-5 (2011), 821–839.

[GL88] Gelfond, M., and Lifschitz, V. The stable model semantics for logic programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium, Seattle, Washington, August 15-19, 1988 (2 Volumes)* (1988), pp. 1070–1080.

[GL91] Gelfond, M., and Lifschitz, V. Classical negation in logic programs and disjunctive databases. *New Generation Comput. 9*, 3/4 (1991), 365–386.

[GM12] Giunchiglia, E., and Maratea, M. Algorithms for solving satisfiability problems with qualitative preferences. In *Correct Reasoning - Essays on Logic-Based AI in Honour of Vladimir Lifschitz* (2012), pp. 327–344.

[GMP12] Godo, L., Marchioni, E., and Pardo, P. Extending a temporal defeasible argumentation framework with possibilistic weights. In *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Toulouse, France, September 26-28, 2012. Proceedings* (2012), pp. 242–254.

[GRTS07] García, A. J., Rotstein, N. D., Tucat, M., and Simari, G. R. An argumentative reasoning service for deliberative agents. In *Knowledge Science, Engineering and Management, Second International Conference, KSEM 2007, Melbourne, Australia, November 28-30, 2007, Proceedings* (2007), pp. 128–139.

[GS04] García, A. J., and Simari, G. R. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming (TPLP) 4*, 1-2 (2004), 95–138.

[GS14] García, A. J., and Simari, G. R. Defeasible logic programming: Delp-servers, contextual queries, and explanations for answers. *Argument & Computation 5*, 1 (2014), 63–88.

[HKLS15] Hoos, H. H., Kaminski, R., Lindauer, M. T., and Schaub, T. aspeed: Solver scheduling via answer set programming. *TPLP 15*, 1 (2015), 117–142.

[HKP11] HadjAli, A., Kaci, S., and Prade, H. Database preference queries - a possibilistic logic approach with symbolic priorities. *Ann. Math. Artif. Intell. 63*, 3-4 (2011), 357–383.

[Kac11] Kaci, S. *Working with Preferences: Less Is More*. Cognitive Technologies. Springer, 2011.

[Kie02] Kiessling, W. Foundations of preferences in database systems. In *VLDB* (2002), pp. 311–322.

[Kon88] Konolige, K. Defeasible argumentation in reasoning about events. In *ISMIS* (1988), pp. 380–390.

[KvdTW06] Kaci, S., van der Torre, L. W. N., and Weydert, E. Acyclic argumentation: Attack = conflict + preference. In *ECAI* (2006), pp. 725–726.

[Lif96] Lifschitz, V. Foundations of logic programs. In *Principles of Knowledge Representation*, G. Brewka, Ed. CSLI Pub., 1996, pp. 69–128.

[LL87] Lacroix, M., and Lavency, P. Preferences; putting more knowledge into queries. In *VLDB'87, Proceedings of 13th International Conference on Very Large Data Bases, September 1-4, 1987, Brighton, England* (1987), pp. 217–225.

[MGS12] Martinez, M. V., García, A. J., and Simari, G. R. On the use of presumptions in structured defeasible reasoning. In *Computational Models of Argument - Proceedings of COMMA 2012, Vienna, Austria, September 10-12, 2012* (2012), pp. 185–196.

[MT98] Marek, V. W., and Truszczynski, M. Stable models and an alternative logic programming paradigm. *CoRR cs.LO/9809032* (1998).

[NBG+01] Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., and Barry, M. An A prolog decision support system for the space shuttle. In *Answer Set Programming, Towards Efficient and Scalable Knowledge Representation and Reasoning, Proceedings of the 1st Intl. ASP'01 Workshop, Stanford, March 26-28, 2001* (2001).

[Pol95] Pollock, J. L. *Cognitive Carpentry: A Blueprint for How to Build a Person*. MIT Press, 1995.

[Poo85] Poole, D. On the comparison of theories: Preferring the most specific explanation. In *IJCAI* (1985), pp. 144–147.

[Pra10] Prakken, H. An abstract framework for argumentation with structured arguments. *Argument & Computation 1*, 2 (2010), 93–124.

[PS96a] Prakken, H., and Sartor, G. A dialectical model of assessing conflicting arguments in legal reasoning. In *Logical Models of Legal Argumentation*. Springer, 1996, pp. 175–211.

[PS96b] Prakken, H., and Sartor, G. A system for defeasible argumentation, with defeasible priorities. In *Practical Reasoning*. Springer, 1996, pp. 510–524.

[PS97] Prakken, H., and Sartor, G. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics 7* (1997), 25–75.

[PTV14] Pigozzi, G., Tsoukiàs, A., and Viappiani, P. Preferences in artificial intelligence. *Annals of Mathematics and Artificial Intelligence* (2014), 1–41.

[PV02] Prakken, H., and Vreeswijk, G. Logics for defeasible argumentation. In *Handbook of Philosophical Logic*, D. Gabbay and F. Guenthner, Eds., vol. 4. Kluwer Academic Pub., 2002, pp. 218–319.

[RGM10] Rosa, E. D., Giunchiglia, E., and Maratea, M. Solving satisfiability problems with preferences. *Constraints 15*, 4 (2010), 485–515.

[RGS07] Rotstein, N. D., García, A. J., and Simari, G. R. Reasoning from desires to intentions: A dialectical framework. In *AAAI* (2007), pp. 136–141.

[RS09] Rahwan, I., and Simari, G. R. *Argumentation in Artificial Intelligence*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[Šef08] Šefránek, J. Preferred answer sets supported by arguments. In *Proceedings of 12th International Workshop on Non-Monotonic Reasoning (NMR 2008)* (2008), pp. 232–240.

[SGCS03] Stolzenburg, F., García, A. J., Chesñevar, C. I., and Simari, G. R. Computing generalized specificity. *Journal of Applied Non-Classical Logics 13*, 1 (2003), 87–113.

[SI00] Sakama, C., and Inoue, K. Prioritized logic programming and its application to commonsense reasoning. *Artif. Intell. 123*, 1-2 (2000), 185–222.

[SKP11] Stefanidis, K., Koutrika, G., and Pitoura, E. A survey on representation, composition and application of preferences in database systems. *ACM Trans. Database Syst. 36*, 3 (2011), 19.

[SL92] Simari, G. R., and Loui, R. P. A mathematical treatment of defeasible reasoning and its implementation. *Artif. Intell. 53*, 2-3 (1992), 125–157.

[SMCS08] Sagui, F. M., Maguitman, A. G., Chesñevar, C. I., and Simari, G. R. Modeling news trust: A defeasible logic programming approach. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 12*, 40 (2008), 63–72.

[SNS02] Simons, P., Niemelä, I., and Soininen, T. Extending and implementing the stable model semantics. *Artif. Intell. 138*, 1-2 (2002), 181–234.

[SP06] Son, T. C., and Pontelli, E. Planning with preferences using logic programming. *TPLP 6*, 5 (2006), 559–607.

[SS05] Schweimeier, R., and Schroeder, M. A parameterised hierarchy of argumentation semantics for extended logic programming and its application to the well-founded semantics. *TPLP 5*, 1-2 (2005), 207–242.

[TGG+12] Tamargo, L. H., Gottifredi, S., García, A. J., Falappa, M. A., and Simari, G. R. Deliberative delp agents with multiple informants. *Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 15*, 49 (2012), 13–30.

[TGGS13a] Teze, J. C., Gottifredi, S., García, A. J., and Simari, G. R. An approach to argumentative reasoning servers with conditions based preference criteria. In *XVIII Congreso Argentino de Ciencias de la Computación* (2013).

[TGGS13b] Teze, J. C., Gottifredi, S., García, A. J., and Simari, G. R. Modelo de servicio de razonamiento con preferencias. In *XV Workshop de Investigadores en Ciencias de la Computación* (2013).

[TGGS14a] Teze, J. C., Gottifredi, S., García, A. J., and Simari, G. R. An approach to argumentative reasoning servers with multiple preference criteria. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial 17*, 53 (2014), 68–78.

[TGGS14b] Teze, J. C., Gottifredi, S., García, A. J., and Simari, G. R. Servicios de razonamiento con múltiples criterios de preferencia. In *XVI Workshop de Investigadores en Ciencias de la Computación* (2014).

[TGGS15] Teze, J. C., Gottifredi, S., García, A. J., and Simari, G. R. Improving argumentation-based recommender systems through context-adaptable selection criteria. *Expert Syst. Appl. 42*, 21 (2015), 8243–8258.

[TGS09] Tucat, M., García, A. J., and Simari, G. R. Using defeasible logic programming with contextual queries for developing recommender servers. In *AAAI Fall Symposium: The Uses of Computational Argumentation* (2009).

[Tuc12] Tucat, M. *Grupos de Servicios de Razonamiento para el Procesamiento de Consultas Contextuales en Paralelo*. PhD thesis, Universidad Nacional del Sur, Bahía Blanca, Argentina, 2012.

[Vre91] Vreeswijk, G. The feasibility of defeat in defeasible reasoning. In *Proceedings of the 2nd International Conference on Principles of Knowledge*

Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991. (1991), pp. 526–534.

[Vre97] Vreeswijk, G. Abstract argumentation systems. *Artif. Intell. 90*, 1-2 (1997), 225–279.

[Wak10] Wakaki, T. Preference-based argumentation capturing prioritized logic programming. In *Argumentation in Multi-Agent Systems - 7th International Workshop, ArgMAS 2010, Toronto, ON, Canada, May 10, 2010 Revised, Selected and Invited Papers* (2010), pp. 306–325.

[Wak11] Wakaki, T. Preference-based argumentation handling dynamic preferences built on prioritized logic programming. In *Agents in Principle, Agents in Practice - 14th International Conference, PRIMA 2011, Wollongong, Australia, November 16-18, 2011. Proceedings* (2011), pp. 336–348.

[Wak14a] Wakaki, T. Assumption-based argumentation equipped with preferences. In *PRIMA 2014: Principles and Practice of Multi-Agent Systems*. Springer, 2014, pp. 116–132.

[Wak14b] Wakaki, T. Assumption-based argumentation equipped with preferences. In *PRIMA 2014: Principles and Practice of Multi-Agent Systems - 17th International Conference, Gold Coast, QLD, Australia, December 1-5, 2014. Proceedings* (2014), pp. 116–132.