Universidad Nacional del Sur

Tesis de Doctor
en Ciencias de la Computación

Argumentación rebatible en entornos dinámicos

Marcela Capobianco

Bahía Blanca – Argentina
2003

Universidad Nacional del Sur

Tesis de Doctor
en Ciencias de la Computación

Argumentación rebatible en entornos dinámicos

Marcela Capobianco

Bahía Blanca – Argentina
2003

Prefacio

Esta Tesis es presentada como parte de los requisitos para optar al grado académico de *Doctor 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 otras. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Departamento de Ciencias e Ingeniería de la Computación, durante el período comprendido entre noviembre de 1999 y diciembre de 2002, bajo la dirección del Dr. Guillermo R. Simari, Profesor Título del Departamento de Ciencias e Ingeniería de la Computación, y el Dr. Carlos I. Chesñevar, Profesor Adjunto del Departamento de Ciencias e Ingeniería de la Computación.

Marcela Capobianco
Departamento de Ciencias e
Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, 14 de diciembre de 2002.

Resumen

El desarrollo de agentes de software inteligentes es vital para la implementación de una vasta diversidad de aplicaciones. Existen numerosos problemas a resolver para poder diseñar esta clase de agentes, dentro de los cuales se destaca encontrar una forma adecuada de modelar el estado epistémico de los mismos. Esta tesis tiene por objeto definir un sistema de representación de conocimiento y razonamiento que posea el poder expresivo y la implementabilidad suficientes para cumplir con esta función. A tal fin se presenta un formalismo argumentativo con un lenguaje basado en la programación en lógica que combina las ventajas de ambas disciplinas.

Como resultado del trabajo realizado se obtuvo un sistema de representación de conocimiento y razonamiento capaz de manejar información incompleta y potencialmente contradictoria e interactuar con ambientes dinámicos. Este sistema incorpora además un mecanismo de inferencia basado en el uso de conocimiento precompilado que posee un conjunto de atractivas propiedades e inaugura el uso de este tipo de técnicas en la argumentación rebatible.

Palabras Clave: inteligencia artificial, representación de conocimiento, argumentación rebatible, sistemas de mantenimiento de verdad.

Abstract

Software agent development is a key issue for implementing a wide range of interesting applications. Many problems arise when designing this kind of agents, being one of the most significative modelling their epistemic state. The main goal of this thesis is to define a knowledge representation and reasoning system for this role. This system must posses enough expressive power and also be implementable. Therefore we define an argumentative formalism that uses a language based on logic programming. This approach combines the advantages of both fields.

As a result we have obtained a knowledge representation and reasoning system able to deal with incomplete potentially contradictory information. This system is also capable of interacting with dynamic environment and incorporates an inference mechanism based on the use of precompiled knowledge, pioneering the used of these techniques in defeasible argumentation. We have also analyzed the formal properties of the resulting system and formulated a set of design principles for argumentative systems.

KEYWORDS: artificial intelligence, knowledge representation, defeasible argumentation, argumentative systems, truth maintenance systems.

Índice general

Índice de figuras

Capítulo 1

Introducción

Modelar el razonamiento de un agente inteligente es una de las metas primordiales de la Inteligencia Artificial. No obstante, el estudio del razonamiento se remonta varios siglos antes del nacimiento de esta disciplina. Aristóteles propuso las leyes de los silogismos lógicos como un primer acercamiento al análisis del razonamiento. Varios siglos más tarde los filósofos Leibnitz y Kant realizaron nuevas contribuciones en este campo. En el siglo XX, Frege y Russell sentaron las bases de la lógica matemática, cuya primera formalización fue publicada en 1910 por Whitehead y Russell en el volumen denominado *Principia Mathematica*. A partir de esta publicación la comunidad científica adoptó a la lógica clásica como un formalismo adecuado para representar el conocimiento y modelar el razonamiento.

La lógica clásica posee una definición formal bien establecida, con una semántica clara y precisa. Sin embargo, se han identificado gran cantidad de problemas que surgen al utilizar este formalismo para modelar el razonamiento de sentido común. En primer lugar la lógica clásica sólo permite modelar el razonamiento de tipo *deductivo*. El razonamiento deductivo posee algunas características que no son deseables para modelar el sentido común. Tal como lo expresa Kant en su obra *Crítica de la razón pura*, las conclusiones que se obtienen mediante el razonamiento deductivo están en cierta forma contenidas en las premisas.

Por otra parte, el razonamiento deductivo no permite descartar inferencias a luz de nueva información. Agregar premisas nunca puede invalidar una conclusión obtenida anteriormente. Esto equivale a decir que cumple la propiedad de *monotonía*: si una conclusión q se obtiene a partir de un conjunto de premisas Γ entonces q también puede obtenerse a partir de cualquier superconjunto de Γ.

Hacia mediados de los años ‘70 tanto Minsky [Minsky, 1975] como McCarthy

[McCarthy, 1977] señalaron que el sentido común presente en el razonamiento humano es de naturaleza no monótona y por lo tanto la lógica clásica no es adecuada para modelar este razonamiento. Tal como lo expresa McCarthy ([McCarthy, 1977])

> ... Aunque el razonamiento humano puede ser modelado por la lógica clásica en muchas situaciones, existen importantes características del razonamiento de sentido común que parecen ser no monótonas. Los seres humanos obtenemos conclusiones de un conjunto de premisas que no seríamos capaces de alcanzar si se agregaran otras sentencias a este conjunto. ...

Para clarificar esta afirmación, consideremos un ejemplo clásico en la literatura sobre razonamiento no monótono. Supongamos que contamos con información de que Tweety es un pájaro. A partir de esta premisa concluiremos que Tweety vuela, sin requerir información adicional sobre si Tweety está enfermo, si es un pingüino o una gaviota, etc. Pero si posteriormente descubrimos que Tweety es un pingüino dejaremos de creer que Tweety puede volar.

La no monotonicidad del razonamiento humano se origina por el conocimiento incompleto que poseemos acerca del mundo. En consecuencia debemos revisar nuestras conclusiones al obtener nueva información. En este contexto surgieron numerosos formalismos para modelar el razonamiento no monótono. En esta etapa se destacan los trabajos de McCarthy [McCarthy, 1980], Moore [Moore, 1984], Reiter [Reiter, 1980] y Doyle [Doyle, 1979]. Sin embargo, a la hora de desarrollar agentes de software que utilicen estos formalismos para obtener inferencias, no es sencillo encontrar un sistema que combine poder expresivo con implementabilidad y eficiencia computacional. Existieron muchos sistemas para los cuales no era posible obtener una implementación o las implementaciones conocidas eran extremadamente ineficientes. Este problema resultó de gran relevancia, puesto que la comunidad de Inteligencia Artificial no sólo buscaba una herramienta para modelar el razonamiento, sino que además exigía que ésta fuera automatizable. En tal sentido merece ser destacado el trabajo realizado por J. Doyle mediante su sistema de mantenimiento de verdad [Doyle, 1979], que toma en cuenta la implementabilidad del sistema.

A partir del razonamiento no monótono nace un nuevo acercamiento denominado *razonamiento rebatible*[1] [Nute, 1988, Pollock, 1987]. Esta disciplina se caracteriza

[1]Este término proviene del vocablo en inglés *defeasible reasoning* utilizado por Nute. Sin embargo el origen de este terminología se remonta al ámbito del razonamiento legal *e.g.* [Toulmin, 1958]. Para más detalles es posible consultar [Chesñevar et al., 2000b].

por efectuar inferencias de carácter tentativo que se pueden invalidar frente a la presencia de nueva información. En consecuencia se obtienen conclusiones *rebatibles* en base a la información disponible hasta el momento, que deben desecharse ante el surgimiento de nueva evidencia que así lo justifique. Los sistemas argumentativos [Pollock, 1987, Loui, 1987, Simari y Loui, 1992, Prakken, 1997, Vreeswijk, 1993, Dung, 1995b] son una de las formalizaciones del razonamiento rebatible más fructíferas.

En lo que resta de este capítulo se resumen los componentes básicos de los sistemas de mantenimiento de verdad (sección 1.1) y se detallan los principales elementos que caracterizan a los sistemas argumentativos (sección 1.2). Es importante destacar que ambos formalismos tendrán especial relevancia para los aportes desarrollados en esta tesis. A continuación se introducen las motivaciones que guiaron el desarrollo de esta Tesis y se enuncian las contribuciones más relevantes que se han obtenido. Por último se detallan los trabajos de investigación realizados previamente, donde se presentaron resultados relacionados con los aportes de la presente disertación.

1.1. Sistemas de mantenimiento de verdad

Los sistemas de mantenimiento de verdad (TMS)[2] consisten en formalismos de razonamiento no monótono que proveen un conjunto de técnicas para manejar información potencialmente inconsistente. Inicialmente, los TMS fueron definidos por Jon Doyle [Doyle, 1979] para ser utilizados en el campo de los resolvedores generales de problemas, con el fin de almacenar y administrar las razones en las cuales se basan las creencias de un agente.

Los TMS almacenan argumentos[3] para las creencias potenciales de un agente utilizando dos estructuras de datos: *nodos* y *justificaciones*. Los nodos representan las creencias del agente y las justificaciones codifican razones para creer en los nodos. Se dice que el sistema cree en un nodo si posee un argumento *válido* para este nodo. Un argumento es válido si está basado en un conjunto de nodos que pertenecen a las creencias del TMS. Cabe destacar que aunque este procedimiento puede parecer circular, existen argumentos que no están basados en ningún nodo, denominados *suposiciones*, a partir de los cuales es posible clasificar los nodos del TMS.

[2]La sigla proviene del término en inglés *Truth Maintenance Systems*.

[3]Es importante destacar que el termino "argumento" es utilizado por Doyle con un significado diferente al empleado en el área de los sistemas argumentativos.

A fin de crear y mantener la estructura conformada por los nodos y las justificaciones se definen en un TMS las siguientes acciones fundamentales:

- Crear un nodo nuevo, al cual se le asocia una determina sentencia representando una creencia potencial.
- Agregar una justificación para un nodo determinado.
- Eliminar una justificación para un nodo determinado.
- Marcar un nodo como contradictorio, para señalar la inconsistencia de cualquier conjunto de creencias que respalden a un argumento para este nodo.

Doyle detalla cómo es posible implementar estas acciones y cuáles son las complicaciones que surgen en cada caso. Por ejemplo, el agregado de una nueva justificación para un determinado nodo n puede cambiar el estado del mismo. Es posible que mediante esta operación el TMS crea en un nodo n aún cuando previamente n no pertenecía al conjunto de creencias. Ante esta situación es factible que otros nodos además de n se agreguen a este conjunto por medio de argumentos ya existentes basados en el nodo n. Para actualizar el conjunto de creencias ante este evento el TMS invoca al *procedimiento de mantenimiento de verdad* que se encarga de actualizar las creencias. En [Doyle, 1979] se describe como funciona este procedimiento.

Es interesante observar que los TMS desarrollados en [Doyle, 1979] utilizan además un tipo particular de justificaciones, denominadas *justificaciones no monótonas*. Estos elementos tienen la particularidad de sustentar una determinada conclusión basándose en que el TMS *no crea* en un cierto conjunto de nodos. Este tipo de justificaciones permite realizar inferencias tentativas que pueden eliminarse al obtener nueva información. Cualquier nodo basado en una justificación no monótona se denomina *suposición*.

La estructura de un TMS esta compuesta de la siguiente manera. Cada nodo n en el sistema posee un conjunto de justificaciones asociado, donde cada justificación representa una razón diferente para creer en n. Se dice que el TMS cree en un nodo n si al menos una de las justificaciones asociadas a n es válida. Intuitivamente una justificación es válida si se cumplen las condiciones establecidas en la misma, esto es, si los nodos en los que debe creer el TMS están dentro del conjunto de creencias y los nodos en los cuales no debe creer el TMS están fuera del conjunto de creencias.

Los nodos que pertenecen al conjunto de creencias del TMS se denominan nodos *in* y el resto se marcan como nodos *out*. Es importante observar que la distinción

entre *in* y *out* no es equivalente a la categorización de las inferencias en verdaderas y falsas. La categorización de los nodos como *in* y *out* se realiza en base a las razones existentes para respaldar cada una de estas creencias, mientras que la asignación de valores *verdadero* o *falso* a las inferencias es independiente de cualquier razón y se basa en el valor de verdad que corresponde de acuerdo con la semántica del sistema.

Cada creencia potencial dentro del TMS debe asociarse a un nodo diferente. Al analizar una proposición en particular P, es posible crear nodos para P y su complemento con respecto a la negación clásica. Cada uno de estos nodos puede estar *in* o *out* del conjunto de creencias y esto conduce a cuatro situaciones epistémicas diferentes: ni P ni su complemento están en el conjunto de nodos *in*, sólo P es un nodo *in*, sólo el complemento de P es un nodo *in* o ambos son clasificados como nodos *in*. Doyle detalla los procedimientos necesarios para determinar el estado de cada nodo.

Dentro de las ventajas de los sistemas de mantenimiento de verdad el autor destaca que este formalismo provee un mecanismo para realizar inferencias no monótonas y permite además revisar las creencias del sistema ante el agregado de nuevas justificaciones. Por otra parte, la eficiencia del sistema se ve significativamente mejorada por medio del conocimiento precompilado almacenado por el mismo. Citando las palabras de Jon Doyle [Doyle, 1979]:

> ... La sobrecarga requerida a fin de mantener las justificaciones de las creencias del sistema puede parecer excesiva. Sin embargo, la cuestión fundamental no consiste en cual es el precio de mantener esta información, sino en el trabajo computacional que se desperdicia al no mantenerlos. Al descartar la información obtenida en las derivaciones estamos condenando al mecanismo de inferencia a realizar una y otra vez la misma tarea, cuando solamente han cambiado un conjunto de suposiciones irrelevantes ...

Desde la publicación del artículo de Jon Doyle [Doyle, 1979] que sentara las bases de los TMS, se han desarrollado gran cantidad de trabajos acerca de este tema [de Kleer, 1986, Elkan, 1990, McAllester, 1990, Forbus y de Kleer, 1993] . La idea principal que inspiró estos acercamientos no parece haber sido alguna técnica o mecanismo en particular, sino el concepto general de un módulo independiente para el almacenamiento de las creencias del sistema.

1.2. Sistemas Argumentativos

Los sistemas argumentativos [Pollock, 1987, Loui, 1987, Simari y Loui, 1992] [Prakken, 1997, Vreeswijk, 1993, Dung, 1995b] surgieron hacia fines de la década del '80 como herramientas para modelar el razonamiento capaces de manejar información incompleta y potencialmente contradictoria. Estos formalismos permiten representar el conocimiento que se posee acerca del mundo a través de una *base de conocimiento* y obtener conclusiones a partir de esta información por medio de un *mecanismo de inferencia* definido para tal fin.

En general la base de conocimiento de un sistema argumentativo permite representar información segura e irrefutable tanto como rebatible o sujeta a excepciones. Para distinguir entre las diferentes categorías se utilizan distintos tipos de reglas. Las reglas que representan conocimiento seguro suelen denominarse *reglas fuertes*, mientras que para representar conocimiento tentativo se utilizan *reglas rebatibles*.

El mecanismo de inferencia de un sistema argumentativo está basado en la construcción de argumentos. De la misma forma que la lógica clásica encadena una cierta cantidad de reglas para obtener una derivación, en los formalismos argumentativos se construyen argumentos para sustentar una determinada conclusión, mediante el uso de la información presente en la base de conocimiento. Dado que es posible utilizar reglas rebatibles para construir un argumento, estos elementos representan un esquema de razonamiento que puede estar sujeto a excepciones.

A partir de la base de conocimiento de un sistema argumentativo es posible construir argumentos cuya aceptación conjunta no sea posible, tal como se ilustra en el siguiente ejemplo:

1. Tweety es un pingüino.
2. Los pájaros generalmente vuelan.
3. Los pingüinos son pájaros.
4. En general los pingüinos no vuelan.

A partir de esta información es posible construir un argumento para sustentar que Tweety vuela, basado en que Tweety es un pingüino y por lo tanto Tweety es un pájaro (regla 3) y los pájaros generalmente vuelan (regla 2). Es también factible obtener un argumento para concluir que Tweety no vuela, dado que Tweety es un pingüino y por lo general los pingüinos no son capaces de volar (regla 4). Al observar

esta situación es fácil notar que ambos argumentos son contradictorios entre sí y no pueden ser aceptados en forma conjunta. Ante esta clase de conflicto el sistema debe adoptar algún criterio de comparación para decidir cuál de estos argumentos será aceptado, es decir cual de ellos resulta victorioso y puede por tanto sustentar su conclusión (que se convertirá en una de las inferencias del sistema).

Tanto el criterio de comparación entre argumentos como la definición de cuando un argumento resulta aceptable y puede justificar a su conclusión varían en cada sistema. En el ejemplo anterior, supongamos que el criterio escogido prefiere al argumento soportando que Tweety no vuela. De acuerdo con la terminología utilizada en los sistemas argumentativos podemos entonces afirmar que el argumento a favor de no vuela Tweety *derrota* a su contrincante

Existen varias definiciones de aceptabilidad de un argumento. Una posibilidad consiste en definir un argumento A como aceptable si es capaz de derrotar a cualquier argumento B tal que B está en conflicto con A. No obstante, A puede ser un argumento aceptable aún cuando sea derrotado por B. Es factible que el argumento B sea a su vez derrotado por un tercer argumento C y en este caso B no poseería la fuerza necesaria para impedir que el argumento A sea aceptable. Esto da lugar a una definición por niveles de la noción de aceptabilidad. Dado que el objetivo de esta sección es simplemente brindar un panorama sobre el funcionamiento de los sistemas argumentativos pospondremos el análisis de las diferentes definiciones de aceptabilidad hasta el capítulo 2 de esta tesis.

En los últimos años se desarrollaron numerosas contribuciones al campo de la argumentación [Pollock, 1987, Pollock, 1995, Simari y Loui, 1992, Vreeswijk, 1997, Prakken y Sartor, 1997, Dung, 1995b, García, 2000, Chesñevar, 2001] . A partir de estos trabajos se ha reconocido a la argumentación como una herramienta poderosa para representar el conocimiento y modelar el razonamiento de sentido común. También se han implementado prometedoras aplicaciones en el área de negociación entre agentes [Prakken y Sartor, 1997, Sierra et al., 1997, Parsons y Jennings, 1997, Gordon y Karacapadilis, 1997] y en la toma de decisiones basada en argumentación [Parsons y Fox, 1997, Fox y Parsons, 1998].

1.3. Objetivos

El desarrollo de agentes de software inteligentes, esto es, capaces de percibir su entorno, razonar, construir planes y actuar en un ambiente dinámico, es vital para

la implementación de gran cantidad de aplicaciones. Existen numerosos problemas a resolver para poder diseñar esta clase de agentes, entre los que se destaca encontrar una forma adecuada de modelar el estado epistémico de los mismos. Para esto es necesario utilizar un sistema de representación de conocimiento y razonamiento que posea el poder expresivo y la implementabilidad suficientes para cumplir con esta función. La investigación realizada en esta tesis tiene como objetivo definir un formalismo que cumpla con estos requisitos.

A partir del análisis llevado a cabo sobre los distintos sistemas de representación de conocimiento y razonamiento, se identificó a la Programación en Loǵica Rebatible (PLR) [García, 2000] como un candidato prometedor para modelar el estado epistémico de un agente. La PLR combina un lenguaje al estilo de la programación en lógica junto con un mecanismo de inferencia argumentativo, heredando las ventajas de ambas disciplinas. Sin embargo, se identificaron un conjunto de desventajas de la PLR, en lo que respecta a la capacidad del formalismo para desenvolverse en ambientes dinámicos. En consecuencia se plantearon los siguientes pasos para diseñar el nuevo sistema:

- Modificar el lenguaje de la programación en lógica rebatible a fin de que resulte adecuado para modelar el conocimiento de un agente inteligente. Existe concenso en que un formalismo apropiado para esta clase de agentes debería ser capaz de modelar al agente y a su ambiente. Al tratarse de agentes que habitan ambientes dinámicos, el sistema debe poseer la capacidad de integrar los cambios del ambiente dentro del estado mental del agente en forma simple y eficiente. Para incorporar estas habilidades en la PLR es necesario definir mecanismos de percepción y actualización del conocimiento.

- Optimizar el mecanismo de inferencia del sistema resultante. La mayoría de los agentes inteligentes debe mantener una interacción con su ambiente. Esta interacción está acotada por el tiempo de respuesta del agente a los estímulos y requerimientos del ambiente. En el caso de agentes inteligentes, cuyo proceso de inferencia es complejo y computacionalmente costoso, esta interacción es particularmente difícil de realizar dentro de los límites de tiempo requeridos por la aplicación.

- Estudiar las propiedades formales del lenguaje definido. Contar con un análisis formal del sistema permite una mejor compresión de sus capacidades y limitaciones.

- Diseñar una implementación eficiente del formalismo propuesto. Esto permitirá su aplicación directa a situaciones concretas.

1.4. Plan de tesis y principales contribuciones

A lo largo de esta tesis se abordan los objetivos identificados en la sección anterior. En este contexto se realizaron una serie de aportes entre los que se destacan las siguientes contribuciones de importancia:

- Se presenta un análisis de la argumentación en general desde una perspectiva histórica, comentando los trabajos fundacionales del área. Se identifican los elementos fundamentales presentes en un sistema basado en argumentación. Estos aspectos se describen en el capítulo 2.

- Se desarrolla una recopilación de las aplicaciones basadas en argumentación más destacadas. Se detallan las áreas de aplicación preponderantes (razonamiento legal, negociación entre agentes y sistemas para la toma de decisiones). Estos temas se analizan en el capítulo 2.

- Se analizan en detalle los sistemas argumentativos de mayor relevancia que han sido desarrollados hasta la actualidad, detallando las virtudes e inconvenientes de cada formalismo. Se presentan distintos ejemplos que clarifican el comportamiento de los distintos sistemas. Este análisis se desarrolla en el capítulo 3.

- Se define el lenguaje de la *programación en lógica rebatible con observaciones* (PLRO). Este formalismo incorpora mecanismos de percepción que posibilitan su uso en ambientes dinámicos. Para desarrollar estos mecanismos se analizan los problemas asociados a la percepción en agentes y se proponen soluciones simples a los mismos dentro del formalismo de la PLRO. Estos aspectos se tratan en el capítulo 4.

- Se propone un mecanismo de inferencia alternativo para la PLRO basado en el uso de conocimiento precompilado. Este nuevo procedimiento se define mediante una estructura asociada a cada programa de la PLRO denominada *base de dialéctica*. Se estudian las propiedades formales de las bases de dialéctica

y se muestran ejemplos de su uso. Finalmente se diseña una implementación eficiente del sistema propuesto. Estas contribuciones se encuentran en el capítulo 5.

- Se estudian formalmente las características de la PLRO como un sistema de representación de conocimiento y razonamiento. Por medio de este análisis se proponen además diversas propiedades que debieran satisfacer los sistemas argumentativos en general. La propuesta realizada constituye un área de trabajo de vanguardia dentro de la comunidad de argumentación. Este análisis se detalla en el capítulo 6.

1.5. Trabajos previos relacionados

La mayoría de los resultados contenidos en esta tesis fueron publicados como trabajos de investigación en diversos congresos y workshops. A continuación se detalla un resumen cronológico de las contribuciones desarrolladas en relación al tema de esta disertación.

En primer lugar, en la tesina presentada por la autora para acceder al título de Licenciada en Ciencias de la Computación [Capobianco, 1999] se investigó como optimizar el proceso de inferencia del sistema MTDR [Simari y Loui, 1992]. La propuesta desarrollada en esta oportunidad consistió en almacenar las inferencias obtenidas por el sistema, junto con el razonamiento realizado para obtenerlas. Esto permite ahorrar trabajo de cómputo al resolver la misma consulta más de una vez. Los resultados obtenidos se publicaron en el artículo *"Introducing Dialectical Bases in Defeasible Argumentation"* [Capobianco y Chesñevar, 1999]. Esta investigación preliminar resultó provechosa para comprender la naturaleza de los problemas existentes en el mecanismo de inferencia de los sistemas argumentativos.

Posteriormente se realizó un estudio sobre las propiedades y características de los agentes racionales y las alternativas existentes para modelar este tipo de entidades. Como consecuencia de esta investigación surgió la idea de utilizar formalismos argumentativos para codificar el estado epistémico de un agente racional. En particular se escogió a la programación en lógica rebatible (PLR) [García, 2000] para realizar esta tarea. En la publicación *"Using Logic Programs to Model an Agent's Epistemic State"* [Capobianco y Chesñevar, 2000] se analizan las ventajas y desventajas de este enfoque con respecto a otros acercamientos tradicionales como la programación en lógica extendida [Gelfond y Lifschitz, 1991] y las lógicas modales.

A partir del análisis de las alternativas existentes para modelar agentes racionales y el estudio de los distintos sistemas de representación de conocimiento, se reconoció a la programación en lógica rebatible como una opción prometedora para modelar el estado epistémico de un agente. Sin embargo, es importante notar que la mayoría de los agentes con utilidad práctica deben desenvolverse en ambientes dinámicos y en estos escenarios es necesario que el agente sea capaz de observar los cambios que suceden en el ambiente e incorporarlos a sus creencias. En consecuencia se definió una modificación del lenguaje de la PLR denominada Programación en Lógica Rebatible con Observaciones (PLRO). En este formalismo se incorporan mecanismos para actualizar el conocimiento del agente frente a los cambios del entorno. Por otra parte se diseñó un mecanismo de inferencia alternativo para la PLRO por medio del uso de conocimiento precompilado. Esta nueva propuesta inspirada en los sistemas de mantenimiento de verdad constituye una mejora al trabajo desarrollado en [Capobianco y Chesñevar, 1999], ya que permite agilizar la obtención de inferencias del sistema, sin importar si la consulta realizada ha sido resuelta previamente por el sistema. La definición de la PLRO junto con las optimizaciones propuestas para su mecanismo de inferencia fueron publicadas en el artículo *"Defeasible Reasoning in Dynamic Domains"* [Capobianco y Simari, 2000].

Finalmente en la publicación *"An argumentative formalism for implementing rational agents"* [Capobianco et al., 2001] se estudiaron las propiedades formales de la PLRO. Se diseñó además una implementación de este sistema argumentativo en la cual se utilizan algoritmos de reconocimiento de patrones para realizar en forma eficiente la construcción y el uso del conocimiento precompilado.

Capítulo 2

Fundamentos de la Argumentación Rebatible

Este capítulo contiene un análisis de los conceptos fundamentales subyacentes al campo de la argumentación rebatible. En primer lugar se presenta una síntesis sobre la evolución cronológica del área, comentando los trabajos fundacionales que sentaron las bases para el desarrollo de la argumentación. Posteriormente se realiza un estudio de los principales conceptos presentes en los sistemas basados en argumentos, destacando el rol preponderante de la dialéctica en estos sistemas. A continuación se desarrolla una recopilación de las aplicaciones basadas en argumentación más destacadas y se exponen las conclusiones de este capítulo.

2.1. Introducción

Los primeros pasos en la teoría de la argumentación fueron obra del mentado filósofo griego Aristóteles. Su objetivo, expresado en sus propias palabras, consistía en:

> *"...Encontrar un método que nos ponga en condiciones de sacar conclusiones sobre cualquier problema que se nos plantee, partiendo de un conjunto de opiniones admitidas..."*

Aristóteles se enorgullece de haber aportado en los *Tópicos* una importante contribución al arte de la obtención de inferencias mediante la definición de procedimientos generales de argumentación. El principal descubrimiento al que se refiere Aristóte-

les son los *silogismos*. Un silogismo puede abstraerse como *"un razonamiento que concluye en sí mismo, independientemente de toda concesión del contrincante"*.[1]

Muchos años pasarían antes de que se realizaran nuevos aportes en estos temas. En el siglo XVI el filósofo W. Leibnitz estudió las características del razonamiento de sentido común y defendió a la argumentación como un método capaz de ordenar y modelar el razonamiento humano:

> *"...nada más importante que el arte de argumentar en forma según la verdadera lógica, con claridad en cuanto al orden y la forma de las consecuencias, ya sean éstas evidentes por sí mismas o previamente demostradas..."*

Ya a mediados del siglo XX merece destacarse el trabajo del filosófo Stephen Toulmin, quien presentó en *"The uses of argument"* [Toulmin, 1958] una crítica al razonamiento puramente deductivo. Toulmin arguye que la lógica tradicional no resulta adecuada para modelar el razonamiento. Su análisis consiste en una serie de ensayos sobre la inferencia racional en los cuales se utiliza como guía la forma en que se construyen y presentan argumentos en el razonamiento legal.

En el más prominente de sus ensayos, Toulmin aborda la estructura interna los argumentos. De acuerdo a la usanza aristotélica un argumento está compuesto por tres clases de proposiciones: premisas menos significativas, premisas más significativas y conclusiones. El autor duda que este modelo simple sea suficiente o adecuado. Utilizando una analogía con la jurisprudencia, propone una nueva representación en la cual un argumento está constituido básicamente por cuatro elementos (que se esquematizan en la figura 2.1).

En esta propuesta la conclusión del argumento se denomina *afirmación*. La *normativa* representa la razón no demostrativa que permite postular la afirmación y está a su vez respaldada por los *fundamentos*. Los *datos* aportan la evidencia necesaria para poder aplicar la normativa. A manera de ejemplo se presenta el siguiente argumento:

(1) Harry nació en Bermudas.
(2) Un hombre nacido en Bermudas es generalmente un súbdito inglés.
(3) Harry es un súbdito inglés.

[1]Esta definición proviene de la comparación con el método dialéctico propugnado por sus antecesores que por ser inseparable del ejercicio del diálogo suponía la colaboración del contrincante.

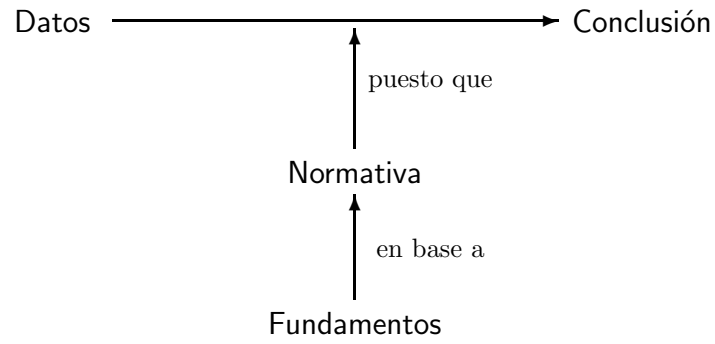


Figura 2.1: Estructura básica de un argumento en el formalismo de Toulmin

En este situación la proposición (1) representa los datos, (2) codifica la normativa y (3) modela la afirmación. La regla *Un hombre nacido en Bermudas es generalmente un súbdito inglés* está fundamentada en las leyes redactadas en las actas del parlamento y otros documentos oficiales concernientes a la nacionalidad de las personas nacidas en las colonias inglesas.

Toulmin señala que un argumento debe ser capaz de *defender* su afirmación. En el modelo de la figura 2.1 la afirmación está basada en los datos y la normativa utilizada. Se identifican entonces dos formas de desafiar a un argumento: desacreditar la afirmación o la validez de la normativa utilizada. La primera de estas debe responderse fortaleciendo los datos que respaldan la conclusión en jaque, mientras que para contestar a la segunda clase de desafío se utiliza la fundamentación. Toulmin también contempla la posibilidad de que las normativas estén sujetas a condiciones de refutación, esto es, escenarios que constituyen excepciones al comportamiento general en los cuales no pueden aplicarse estas normativas. Por ejemplo, Harry es un ciudadano inglés *a menos que* haya optado por otro nacionalidad.

Dado que su estudio se centra en la composición de los argumentos, Toulmin no define un procedimiento ni formal ni intuitivo para determinar su aceptabilidad. Sin embargo, sostiene que un argumento es válido si es capaz de sobrevivir a un proceso de disputa correctamente formulado y plantea como trabajo futuro la tarea de encontrar un criterio para determinar cúando una disputa es correcta.

Dentro de las consideraciones expuestas por Toulmin cabe destacarse la distinción entre argumentos *analíticos* y *formalmente válidos*. En un argumento analítico los *datos* junto con los *fundamentos* permiten derivar la conclusión. Los argumentos formalmente válidos tienen la forma *A partir de estos datos y de esta normativa se sigue esta conclusión* y es la normativa quien permite arribar a la afirmación. La diferencia primordial es que cualquier argumento puede plantearse en términos formalmente válidos, explicitando una normativa apropiada, y en cambio sólo los argumentos de carácter puramente deductivos son analíticos. Toulmin critica a lógicos de su época porque éstos utilizan un modelo matemático del razonamiento, obteniendo de esta forma solamente argumentos analíticos. En consecuencia no es posible aprovechar el poder expresivo de los argumentos formalmente válidos, elementos indispensables para codificar algunos aspectos del razonamiento.

Si bien la contribución de Stephen Toulmin es principalmente de interés histórico, existen numerosos tópicos abordados en su obra que vale la pena considerar. Por caso:

- Los argumentos utilizan inferencias de tipo deductivas, rebatibles y probabilísticas, resultando en un modelo completo e interesante desde el punto de vista filosófico.

- El sistema de Toulmin está guiado por las metas, comenzando con una afirmación no sustentada a la cual se añaden premisas en caso que ésta fuera cuestionada. Esta es una propiedad deseable de la que carecen algunos formalismos de argumentación actuales.

- Se discute la idea de que cada normativa confiere un grado de veracidad a las afirmaciones que justifica, lo que genera argumentos con diferente fuerza conclusiva.

La noción de *argumento* dentro del campo de Inteligencia Artificial fue introducida por Ronald Loui. Hasta la publicación de su trabajo *"Defeat among Arguments"* [Loui, 1987] la argumentación había constituido un tema de interés principalmente filosófico. La teoría de Loui utiliza como lógica subyacente un lenguaje de primer orden L. Se define además un conjunto de *reglas de inferencia rebatibles*, denotadas como $\Phi \succ \Psi$, donde Φ y Ψ son fórmulas de L. Estas reglas definen una relación metalingüística con la siguiente semántica: ante la falta de información que indique lo contrario, Φ es una razón para sustentar Ψ. La base de conocimiento del sistema

está constituida por dos elementos: un conjunto de hechos EK, denominado conocimiento evidencial, y un conjunto de reglas de inferencia rebatibles R. Cabe destacar que el conjunto EK está incluido en las fórmulas de L.

En la teoría de Loui, un argumento se define como un grafo de prueba en el cual *la información fluye desde las premisas (fuentes) hasta la conclusión (desagüe).* Formalmente, este grafo debe ser acíclico, dirigido y poseer un único desagüe. Los nodos del grafo están etiquetados por fórmulas del lenguaje L y no existen dos nodos que pueden tener la misma etiqueta. Para que un grafo de estas características sea un argumento A con respecto a una base de conocimiento (EK, R), deben cumplirse algunas restricciones adicionales. En primer lugar las fuentes del grafo deben ser elementos de EK. Además, cada arco desde un nodo P hacia un nodo Q debe ser tal que P se puede inferir clásicamente a partir de $EK \cup P$ o existe una regla rebatible en R de la forma $P \succ Q$.

Loui define un conjunto de relaciones entre argumentos, entre las que se destacan la de *contraargumentación* y *derrota*. Un *derrotador* B de un argumento A consiste en un argumento B tal que las conclusiones de B y A son contradictorias entre sí y B es preferido a A de acuerdo a un criterio de comparación entre argumentos definido oportunamente. Un argumento A es un *contraargumento* de B si existe alguna etiqueta n en el grafo de inferencia de B tal que la conclusión de A y n son contradictorias entre sí.

A partir de estas relaciones, nace el concepto de argumento *justificado* o *garantizado* para distinguir una clase particular de argumentos que prevalecen ante sus adversarios. Un argumento A *justifica* a su conclusión P si no existe un argumento B tal que B derrota a A y para cada contraargumento de A existe un argumento que lo derrota. En base a esto las inferencias del sistema son caracterizadas como las conclusiones de los argumentos *justificados.*

Es nuestra opinin que el sistema desarrollado en [Loui, 1987] presenta varias desventajas. Por ejemplo, no permite la *reinstanciación*[2] de argumentos ¿Es válido que un argumento A sea derrotado por un argumento B que a su vez se encuentra derrotado por un tercer argumento C?. Sin embargo, el mérito principal del trabajo de Loui consiste en definir las nociones primordiales de *interferencia*, *derrota* y *justificación* , que forman parte de la gran mayoría de los sistemas argumentativos actuales.

[2]La sección 2.2 contiene un análisis sobre el concepto de reinstación.

Ya en el año 1989 Fangzhen Lin y Yoav Shoham [Lin y Shoham, 1989] definieron un sistema argumentativo con el objeto de modelar el razonamiento de sentido común y en particular su naturaleza no monótona. Los autores manifiestan que mediante la noción de *argumento* es posible dilucidar algunas propiedades del razonamiento que son difíciles de capturar utilizando las lógicas no monótonas tradicionales.

El formalismo de Lin y Shoham está compuesto por un lenguaje $\mathcal{L}$ que debe contener un operador especial '$\neg$' y un conjunto de *reglas de inferencia*. No se asume ninguna propiedad o restricción adicional sobre $\mathcal{L}$; su función es solamente proveer al sistema con un conjunto de fórmulas. Las reglas de inferencia pueden clasificarse en tres categorías.

1. **Hechos:** toda fórmula bien formada A de $\mathcal{L}$ es una regla de inferencia denominada *hecho base*.

2. **Reglas monótonas:** toman la forma $A_1, \ldots, A_n \rightarrow B$, donde $n > 0$ y $A_1, \ldots, A_n, B$ son fórmulas bien formadas.

3. **Reglas no monótonas:** se denotan mediante $A_1, \ldots, A_n \Rightarrow B$, donde $n > 0$ y $A_1, \ldots, A_n, B$ son fórmulas bien formadas.

Un hecho base representa información explícita sobre un escenario en particular, *e.g.*, *ave(Tweety)*. Una regla monótona encierra conocimiento deductivo, es decir, siempre es posible concluir B a partir de $A_1, \ldots, A_n$. Por ejemplo, podemos expresar que todos los pingüinos son aves utilizando la regla *pingüino(a)* $\rightarrow$ *ave(a)*. Una regla no monótona modela nuestro conocimiento de sentido común sobre el dominio en consideración y por lo general nos permite concluir B a partir de $A_1, \ldots, A_n$. Por caso, la frase *si a es un ave entonces vuela* puede codificarse como *ave(a)* $\rightarrow$ *vuela(a)*. El uso de reglas de inferencia constituye una de las características originales de este sistema y lo distingue de las lógicas no monótonas tradicionales, basadas en sentencias.

En el formalismo de Lin y Shoham la noción de *argumento* se define como un árbol de reglas de inferencia de acuerdo con las siguientes condiciones:

1. Sea A es un hecho base, entonces el árbol formado por A como único nodo es un argumento que posee a A como raíz.

2. Sean $p_1, \ldots, p_n$ argumentos cuyas raíces son $A_1, \ldots, A_n$ respectivamente y sea $A_1, \ldots, A_n \rightarrow B$ una regla de inferencia tal que B no es un nodo en ninguno

de los árboles para $p_1, \ldots, p_n$; entonces el árbol p con raíz B y $p_1, \ldots, p_n$ como subárboles inmediatos constituye un argumento. La regla $A_1, \ldots, A_n \rightarrow B$ es la etiqueta de todos los arcos que conectan a B con sus hijos.

3. Sean $p_1, \ldots, p_n$ argumentos cuyas raíces son $A_1, \ldots, A_n$ respectivamente y sea $A_1, \ldots, A_n \Rightarrow B$ una regla de inferencia tal que B no es un nodo en ninguno de los árboles para $p_1, \ldots, p_n$; entonces el árbol p con raíz B y $p_1, \ldots, p_n$ como subárboles inmediatos constituye un argumento. La regla $A_1, \ldots, A_n \Rightarrow B$ es la etiqueta de todos los arcos que conectan a B con sus hijos.

Si φ es la raíz del árbol se dice que p es un argumento para φ o que φ está basada en p. Para evitar generar un número infinito de argumentos redundantes de la forma $A \rightarrow A, A \rightarrow A \rightarrow A, A \rightarrow A \rightarrow A \rightarrow A$, etc φ sólo puede aparecer como raíz del árbol.

La formalización del concepto de argumento como un árbol de reglas de inferencia fue utilizada posteriormente en algunos acercamientos (*e.g.*, [Vreeswijk, 1997, Prakken, 1993]). La estructura de los argumentos no juega un rol preponderante en el modelo de Lin y Shoham, dado que prácticamente no se definen relaciones entre los mismos. No se define un criterio de comparación ni se estudia la fuerza conclusiva de los argumentos.

Para definir el conjunto de inferencias de su formalismo Lin y Shoham apelan a la noción de *estructuras de argumentos.*[3] Estos elementos pueden interpretarse como conjuntos de argumentos lógicamente consistentes respaldados por un agente. Sea R un conjunto de reglas de inferencia, un conjunto de argumentos T es una *estructura de argumento* si cumple las siguientes condiciones:

1. Si p es un hecho base en R entonces $p \in T$.

2. T es cerrado, es decir, para cada $p \in T$ tal que p' es un subconjunto de p se cumple que $p' \in T$.

3. T es monótonamente cerrado, esto es, si p puede obtenerse a partir de $p_1, \ldots, p_n$ y mediante una regla monótona y $p_1, \ldots, p_n \in T$ entonces p también pertenece a T.

4. T es consistente, es decir, para cada fórmula ψ se cumple que T no contiene argumentos para ψ y $\neg\psi$ simultáneamente.

[3]Este término es utilizado con un significado diferente en la sección 3.2.

Este concepto no resulta suficiente para definir las creencias de un agente. En particular los autores observan que no se exige la propiedad de maximalidad en las estructuras de argumentos, propiedad indispensable para que el sistema obtenga tantas conclusiones como le sea posible a fin de poseer un conocimiento más completo y acabado del mundo. Para solucionar esta situación se define la noción de *completitud* con respecto a una fórmula. Una estructura de argumento es *completa* con respecto a una fórmula φ si contiene un argumento para φ o para su negación. Este concepto no se utiliza para definir una semántica particular del sistema, sino que los autores afirman que mediante la noción de estructuras de argumento completas con respecto a una clase de fórmulas es posible modelar un conjunto de diversas semánticas tan sólo variando las fórmulas en consideración.

Finalmente, para ilustrar el poder expresivo de su sistema, Lin y Shoham sostienen que diversos formalismos no monótonos (como la lógica default [Reiter, 1980], la lógica autoepistémica [Moore, 1984], los sistemas que usan el principio de la negación por falla [Clark, 1978] y la circunscripción [McCarthy, 1980]) pueden expresarse mediante su framework argumentativo. Para demostrar esta tesis detallan como es posible transformar a cada uno de estos formalismos en un framework argumentativo en particular con una semántica adecuada. Desafortunadamente la mayoría de estas transformaciones no constituye evidencia alguna, ya que no son realizables en la práctica. Por caso, el intento de codificar la lógica default resulta en un sistema argumentativo con una cantidad infinita de reglas de inferencia.

Una de las principales desventajas del formalismo de Lin y Shoham yace en que carece de un mecanismo de derrota y es por lo tanto incapaz de decidir entre argumentos contradictorios. Ante la existencia de un conflicto, esto genera distintas estructuras de argumentos sobre las que no existe ningún criterio de preferencia y la noción de completitud no aporta una solución razonable a este problema. Al parecer los autores intentaron diseñar un sistema lo suficientemente general para modelar distintas lógicas no monótonas, pero el proceso de inferencia adoptado provocó que el formalismo heredase las limitaciones de estas lógicas. En consecuencia la capacidad de sancionar conclusiones del sistema y su aplicabilidad a situaciones concretas se ven seriamente comprometidas.

2.2. Características generales

Aunque existe una gran variedad de sistemas basados en la argumentación rebatible, la mayoría de ellos comparten un conjunto de características comunes. A continuación analizaremos en detalle cada uno de estos elementos de acuerdo a lo expuesto en [Prakken y Vreeswijk, 1999]. En general estos formalismos están compuestos por una lógica subyacente, una definición de argumento, una relación de conflicto, una relación de derrota y una noción del estado de los argumentos a partir de la cual se definen las consecuencias del sistema.

La lógica subyacente depende del formalismo argumentativo en particular y se utiliza tanto para representar la base de conocimiento como para proveer una noción de consecuencia monótona en la que se basa la definición de argumento. Si bien este concepto varía de acuerdo al sistema, se destacan dos tipos de propuestas. Usualmente un argumento se representa como un conjunto de reglas, un árbol de inferencia o una secuencia de reglas encadenadas. También es posible definir un argumento como un par (`premisas`, `conclusión`) tal que en la lógica subyacente existe una prueba para la conclusión a partir de las premisas.

Al considerar los distintos argumentos que es posible construir a partir de una base de conocimiento puede suceder que la aceptación conjunta de algunos de ellos no sea posible. Esto da lugar a tres clases de conflictos. La primer categoría se manifiesta cuando dos argumentos tienen conclusiones contradictorias, como por ejemplo los argumentos *"Tweety vuela porque es un pájaro"* y *"Tweety no vuela porque es un pingüino"*. Esta situación se denomina *ataque por rebatimiento* y constituye una forma de ataque simétrico (si un argumento A rebate a B es trivial verificar que B rebate a A). La segunda categoría se denomina *ataque a las suposiciones* y ocurre cuando un argumento A se basa en la suposición de que la proposición r no puede ser probada, pero existe otro argumento B que constituye una prueba para r. Por ejemplo, *"Tweety vuela porque es un pájaro y no puede probarse que Tweety es un pingüino"* es atacado por *"Tweety es un pingüino porque mi percepción así lo indica"*. La última clase de conflicto, identificada por J.Pollock en [Pollock, 1987], se denomina *ataque por socavamiento*.[4] Estas situaciones no representan un ataque a una proposición en particular, sino que cuestionan la conexión entre las premisas y la conclusión, es decir, la regla de inferencia utilizada. Por caso, *"la rosa es roja porque yo la veo roja"* es atacado por *"la rosa está iluminada por luz roja y esta luz hace*

[4]Esta denominación en castellano corresponde al término en inglés *undercutting attack*.

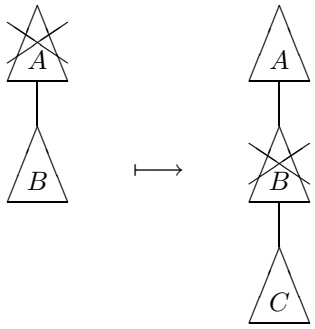


Figura 2.2: Argumento A reinstanciado al derrotarse B.

parecer a los objetos rojos aún cuando sean de otro color". Cabe acotar que tanto el ataque a las suposiciones como el ataque por socavamiento no son simétricos.

Para determinar que argumento prevalece ante un ataque es necesario introducir otro de los elementos claves de la argumentación: la relación de *derrota*. Este concepto permite comparar los argumentos en conflicto y decidir si el ataque tiene éxito. En general cuando un argumento A derrota a un argumento B es posible diferenciar dos situaciones: una versión estricta y asimétrica (A derrota a B y B no derrota a A) y una clase de derrota menos categórica de la forma A no es más débil que B. Los sistemas de argumentación existentes varían en el criterio utilizado para decidir que argumento prevalece en un ataque. Algunos autores optan por un criterio automatizable como la especificidad [Poole, 1985, Simari y Loui, 1992], mientras que otros formalismos [Vreeswijk, 1997, Prakken y Sartor, 1997] utilizan un criterio codificado por el usuario para cada dominio en particular.

Es importante destacar que la relación de derrota solamente provee información relativa sobre la fuerza conclusiva de los dos argumentos en conflicto y no es capaz de determinar el estado final de los mismos, que depende de la interacción entre todos los argumentos que pueden construirse a partir de la base de conocimiento. Puede ser el caso que A derrota a B pero B es a su vez derrotado por un tercer argumento C, tal como se muestra en la figura 2.2. En esta situación C reinstancia al argumento A.

Por lo tanto es necesario enunciar una definición del estadode los argumentos que permita obtener las conclusiones del sistema argumentativo. Generalmente se distinguen dos estados que establecen dos clases disjuntas de argumentos: los que permiten ganar una disputa en forma indiscutida (comúnmente denominados *justificados* o *garantizados*) y los argumentos que deberían perderla (llamados *denegados*

o *derrotados*). Algunos formalismos distinguen una tercer clase de argumentos sobre los cuales la disputa permanece indecisa. En general, el conjunto de inferencias del sistema está compuesto por las conclusiones de los argumentos justificados.

La noción de estado de los argumentos puede definirse en forma declarativa o procedimental. Las alternativas declarativas usualmente utilizan una ecuación de punto fijo y declaran como justificados un conjunto de argumentos que cumple las propiedades indicadas. En contraste, la definición procedimental especifica un procedimiento para determinar cuando un argumento en particular es miembro de este conjunto.

Tal como señalan Prakken y Vreeswijk [Prakken y Vreeswijk, 1999], no existe consenso sobre si las lógicas de razonamiento no monótono (y específicamente los sistemas argumentativos) deben poseer una semántica básada en modelos. Recientemente ha sido propuesta una alternativa que consiste en emplear una semántica basada en la argumentación teórica [Dung, 1995b, Bondarenko et al., 1997]. La idea básica de este acercamiento consiste en sancionar conjuntos de argumentos maximales que pueden defenderse adecuadamente de los ataques contra sus miembros. En consecuencia la definición declarativa del conjunto de argumentos justificados puede interpretarse como la semántica del sistema.

2.3. Dialéctica y argumentación

Algunos sistemas argumentativos proveen un marco teórico diferente para formalizar el razonamiento de sentido común, proveniente de un concepto presente en la filosofía occidental desde los tiempos de la Grecia clásica: la *dialéctica*. Los griegos entendían este concepto como *el arte de discurrir y disputar en forma dialogada*. Ya Sócrates concibe el razonamiento como una discusión o diálogo introspectivo y a la dialéctica como un instrumento para hallar la verdad. Posteriormente Platón acuña las siguientes ideas fundamentales: los pasos iniciales de la argumentación racional no pueden ser simplemente asumidos, sino que deben ser justificados y la dialéctica es el *modus operandi* propuesto para su justificación. En consecuencia Platón no coincide con la idea de basar el razonamiento en un conjunto de axiomas incuestionables (lo cual se corresponde con el modelo deductivo) y propone a la dialéctica como el mecanismo para independizarse de los mismos. Por desgracia en ninguno de sus diálogos se explicita con exactitud la naturaleza del proceso dialéctico.

Aristóteles distingue entre la *dialéctica* y la *analítica*. Para él la primera sólo

comprueba las opiniones de la segunda, que trabaja de forma deductiva a partir de principios que descansan sobre la experiencia y observaciones precisas. Esta concepción supone una ruptura con las ideas de Platón; para Aristóteles la dialéctica es sólo un elemento auxiliar. No obstante, es necesario reconocer que a él se deben las primeras reglas formales para la *técnica de la discusión*. Los principales tratados de Aristóteles comienzan con un examen dialéctico en el cual se expone el estado de la cuestión en consideración, se mencionan las teorías en curso, se detallan las posiciones encontradas de los sabios y se someten las distintas opiniones a una comprobación crítica.

Nicholas Rescher presenta en su obra *"Dialectics: a Controversy-Oriented Approach to the Theory of Knowledge"* [Rescher, 1977] un análisis de la dialéctica y la argumentación desde una perspectiva filosófica. Este trabajo se se centra en las nociones de disputación, debate y controversia racional. Por medio del estudio de estos conceptos el autor pretende dilucidar su utilidad en la teoría del conocimiento.

La disputación es la instancia más clara (e históricamente más prominente) del proceso dialéctico. Se define como un método para conducir discusiones controversiales en el cual un competidor defiende una determinada tesis de los contraargumentos realizados por un adversario. Por lo tanto una disputación formal comprende tres participantes: un *proponente*, su *oponente* y un *juez* o *árbitro*. Esencialmente el debate es gobernado por la siguiente regla: el proponente debe responder todos los postulados formulados por su adversario. Al finalizar el árbitro da su veredicto respecto al caso. Es importante señalar que este proceso no sólo permite modelar un debate entre dos participantes, sino también el análisis introspectivo del conocimiento de un agente.

Rescher sostiene que la disputación es fácil de sistematizar debido a las rígidas reglas que la definen. En su modelo se distinguen tres tipos de *jugadas fundamentales*:

Afirmaciones categóricas Se denotan como $!P$, donde P es una proposición;[5] su significado intuitivo es *es el caso que* P. Sólo pueden ser realizadas por el proponente.

Afirmaciones cautas Se identifican mediante $\dagger P$ y significan *por todo lo expuesto por mi adversario, es posible concluir que es el caso que* P.

[5]En la teoría de Rescher las letras P, Q, R, S, etc. se utilizan para representar proposiciones lógicas.

Afirmaciones provisorias Mantienen la forma P/Q, y su significado es *cuando es el caso que Q usualmente es el caso que P*. Algunos ejemplos de estos elementos son: *si Tweety es un ave entonces puede volar* o *un niño nacido en Estados Unidos aprende a hablar inglés*. Deben estar precedidas por alguna de las formas anteriores de afirmar Q.

Es interesante analizar que el concepto de regla por defecto[6] ya se encontraba presente en la teoría de Rescher por medio de la noción de afirmaciones provisorias.

Seguidamente se enuncia otra categoría de reglas denominadas *contrajugadas*, que se encargan de responder a las afirmaciones descriptas anteriormente.

Contrajugadas para afirmaciones categóricas Ante $!P$ el oponente puede formular un *desafío*, denotado como $\dagger{\sim}P$, el cual modela un petitorio de la forma *sírvase probar P.* Es también posible responder con una *negación provisoria* $({\sim}P/Q \;\&\; \dagger\, Q)$ que significa *P no es posible, dado que $\sim P/Q$, y por todo lo expuesto anteriormente es posible concluir Q.*

Contrajugadas para afirmaciones o negaciones cautas Dos jugadas diferentes pueden efectuarse a continuación de $\dagger P$: *contraafirmaciones categóricas* de la forma $!{\sim}P$ o *contraafirmaciones provisorias*, ${\sim}P/Q \;\&!Q$. Ambas pueden ser realizadas solamente por el proponente. Cabe observar que están prohibidas las secuencias circulares de la forma:

Proponente	Oponente
$!P$	$\dagger{\sim}P$
$!P$	

pues atentan contra la esencia *progresiva* de la disputación (más adelante discutiremos este tema con mayor detalle).

Contrajugadas para afirmaciones o negaciones provisorias Una afirmación provisoria P/Q sólo puede ser formulada en el contexto de una afirmación categórica o cauta de Q. Además de atacar a la afirmación para Q mediante las contrajugadas descriptas anteriormente, es posible replicar a P/Q usando una *excepción débil* de la forma ${\sim}P/(Q \;\&R) \;\&\, \dagger\,(Q \;\&R)$ o una *excepción fuerte*, ${\sim}P/(Q \;\&R) \;\&!(Q \;\&R)$. Es importante destacar que R debe ser distinto de Q, para evitar incurrir en un ciclo.

[6] Traducción del término en inglés *default rule*.

Por último Rescher analiza otro conjunto de jugadas denominadas *contrajugadas dialécticas*, diseñadas para responder a reglas complejas.

Contrajugadas para una negación provisoria Una negación provisoria de la forma $\sim P/Q$ puede ser atacada en cualquiera de sus componentes. Por lo tanto es posible responder con contraafirmación categórica $!\sim Q$ (atacando a $\dagger Q$), una contraafirmación provisoria $\sim Q/R \;\&!R$ o mediante un ataque a la negación provisoria $\sim P/Q$ que puede representarse mediante una excepción fuerte $\sim P/(Q \;\& S) \;\&!(Q \;\& S)$.

Contrajugadas para una contraarfirmación provisoria Una contraafirmación provisoria $\sim P/Q \;\&!Q$ puede ser respondida por el oponente utilizando un ataque a la afirmación categórica $!Q$ (para lo cual puede utilizarse una negación cauta o una negación provisoria) o un ataque a la afirmación provisoria $\sim P/Q$ (que puede tomar la forma de una excepción débil $\sim P/(Q \;\& S) \;\&\dagger\,(Q \;\& S)$).

Contrajugadas para una excepción débil Una excepción débil de la forma $\sim P/(Q \;\& R) \& \dagger\,(Q \;\& R)$ puede responderse con un ataque a $\dagger(Q \;\& R)$ (que puede realizarse usando una contraafirmación categórica o una contraafirmación provisoria) o por medio de una excepción fuerte contra $\sim P/(Q \;\& R)$ que toma la forma $\sim P/(Q \;\& R \;\& S) \;\&!(Q \;\& R \;\& S)$

Contrajugadas para una excepción fuerte Una excepción fuerte de la forma $\sim P/(Q \;\& R) \;\&\; !(Q \;\& R)$ puede ser contraargumentada por el oponente a través de un ataque a $!(Q \;\& R)$ (con un desafío o una negación provisoria $\sim(Q \;\& R)/S \;\&\dagger\, S$) o por medio de una excepción débil contra $\sim P/(Q \;\& R)$ de la forma $P/(Q \;\& R \;\& T) \;\&\dagger\,(Q \;\& R \;\& T)$.

El juego procede como un intercambio de argumentos entre el proponente y el oponente hasta que alguno de los participantes acepta los razones de su adversario. Las concesiones se realizan tácitamente, por caso, si el oponente no desafía una de las afirmaciones del proponente se asume que el oponente concede esta afirmación. Si los participantes no pueden realizar ninguna jugada adicional, o se excede algún límite de recursos, el árbitro entra en escena para determinar quien obtiene la victoria de acuerdo a un conjunto de estándares preestablecidos.

Ejemplo 2.1 El siguiente diálogo, extraído de [Rescher, 1977], constituye un debate formal que puede ser recreado mediante el formalismo de disputación descripto.

prop. Nosotros conocemos a los distintos objetos por medio de las sensaciones. [Yo se que *esto* es un mano humana]. En este caso $[] = P$ y entonces $!P$.

opon. No puedes saber que *eso* es una mano si no estas seguro de ello y por todo lo expuesto [no estas seguro de que eso es una mano humana]. En este caso $[] = Q$ y en consecuencia $\sim P/Q \,\&\dagger\, Q$.

prop. Yo estoy seguro de que esto es una mano, entonces $!\sim Q$.

opon. Pero no puedes estar seguro si [tus sentidos te desorientan]. $[] = R$ y entonces $R/S \,\&\dagger\, S$.

prop. Mis sentidos me han desorientado en algunas oportunidades pero en [este caso es diferente]. Luego mis sentidos no me están engañando. $[] = T$ y por lo tanto $\sim R/(S \,\&T) \,\&!(S \,\&T)$.

opon. [Seguramente es posible que tus sentidos te engañen en este caso]. []=U y entonces $Q/U \,\&\dagger\, U$.

prop. [La posibilidad que tu mencionas es puramente conjetural] y no puede bloquear el conocimiento. En consecuencia niego que sea real. En este caso $[] = V$, entonces $\sim Q/(U \,\&V) \,\&\, !(U \,\&V)$.

	Proponente	Oponente
(1)	$!P$	$\sim P/Q \,\&\dagger\, Q$
(2)	$!\sim Q$	$Q/R \,\&\dagger\, R$
(3)	$!\sim R$	$R/S \,\&\dagger\, S$
(4)	$\sim R/(S \,\&T) \,\&\, !(S \,\&T)$	$Q/U \,\&\dagger\, U$
(5)	$\sim Q/(U \,\&V) \,\&\, !(U \,\&V)$	

■

Ejemplo 2.2 Consideremos un escenario típico en el área de razonamiento no monótono:

Las aves usualmente vuelan.
Los pingüinos por lo general no vuelan.
Los pingüinos son una subclase de las aves.

Una disputa para determinar si un pingüino vuela puede ser representada como sigue:

	Proponente	Oponente
(1)	!~ vuela	vuela / ave & †ave
(2)	!~ vuela / (pingüino & ave) & !(pingüino & ave)	†~ pingüino

■

Rescher discute un conjunto de nociones vinculadas con el proceso de disputación. Uno de los principales conceptos es el de *peso de la prueba*[7] el cual intuitivamente significa que el adversario afirmando una determinada tesis (no el que intenta refutarla) tiene la responsabilidad de presentar la evidencia del caso. Se distinguen dos concepciones diferentes de esta noción: *el peso de la prueba de la afirmación inicial*, el cual recae en el proponente, y *el peso de la evidencia de la prueba*, esto es, la responsabilidad de asegurar que el proceso de argumentación sea *progresiva* (*i.e.*, que introduzca nueva evidencia en cada paso). Esta carga fluctúa entre ambos participantes mientras continúa la controversia. Rescher también describe el rol de las *suposiciones*. Una suposición modela a un hecho que se asume verdadero mientras no se conozca información que sustente lo contrario. Lo que distingue a una suposición de una proposición elegida aleatoriamente es el concepto de *plausibilidad*, esto es, cómo encaja la suposición en cuestión con el conocimiento aceptado hasta el momento.

Existen ciertas asimetrías entre los participantes de una disputación que merecen ser destacadas.

- El proponente debe inaugurar la disputa con una afirmación categórica de la tesis que se propone defender.
- Todas las contrajugadas que comprenden afirmaciones categóricas están restringidas al proponente. Además cada jugada del proponente encierra alguna afirmación categórica, dado que en él recae el peso de la prueba.
- Todas las jugadas que representan un desafío o una afirmación cauta están permitidas solamente para el oponente. Además cada maniobra del oponente debe comprender alguna de estas jugadas, ya que su función es cuestionar las afirmaciones de su adversario.

[7]El término utilizado originalmente en el idioma inglés es *burden of proof*, que a su vez proviene del vocablo latíno *onus probandi*.

Esta disparidad es producto de las diferencias entre los roles de ambos participantes. Cada paso del proponente contiene un compromiso que este protagonista está obligado a defender, mientras que el oponente no necesita efectuar ninguna afirmación. Su trabajo consiste en refutar las razones del proponente y tiene éxito en el mismo cuando descubre algún error en los argumentos de su adversario.

Tal como se señala en [Chesñevar, 1996], la presentación de Rescher es semiformal. Por ejemplo, no se define el concepto de argumento: los participantes intercambian proposiciones que juegan el rol de los mismos. Tampoco se presenta una noción rigurosa de la base de conocimiento y por consiguiente no queda claro como se obtiene la evidencia que será utilizada en la disputa. Sin embargo, al considerar la fecha en la que se desarrolló el trabajo, es necesario reconocer que Rescher plantea un conjunto de ideas adelantadas a su época. Mediante las afirmaciones provisorias se describe una versión de reglas por defecto cuando estas no habían sido todavía definidas en el área lógico-matemática. Asimismo se identifica la importancia de la disputación para modelar un proceso de razonamiento introspectivo y se desarrolla la teoría de la *dialéctica unilateral* para analizar este proceso. Esta interesante caracterización no sería usada en los sistemas argumentativos hasta la década del '90.

2.4. Aplicaciones

Esta sección contiene una recopilación de las aplicaciones más relevantes de la argumentación que han sido desarrolladas hasta la actualidad. En primer lugar se considera el campo de la inteligencia artificial y el razonamiento legal. A continuación se detallan las aplicaciones de la argumentación en la negociación entre agentes y luego se analizan los sistemas que utilizan argumentación para la toma de decisiones.

2.4.1. Inteligencia Artificial y Razonamiento legal

Gran parte de las aplicaciones de la argumentación han sido realizadas en el área de Inteligencia Artificial y Razonamiento Legal. Los pioneros en producir una contribución en este área fueron Thomas Gordon y Henry Prakken. Gordon desarrolló en [Gordon, 1987] a Oblog-2, un sistema híbrido de representación de conocimiento para realizar razonamiento rebatible. Oblog-2 combina un *razonador terminológico* junto con un mecanismo de inferencia al estilo Prolog. El componente terminológico

permite incorporar descripciones de taxonomía para tipos y atributos. Las entidades del sistema son un conjunto de tipos. En base a las descripciones existentes, se definen procedimientos para asignar valor al conjunto de atributos de una entidad. Estos procedimientos están codificados mediante claúsulas Horn indexadas de acuerdo al conjunto de tipos. Los tipos que se conocen de una entidad determinan el conjunto de reglas que son aplicables sobre los mismos, el cual cambia conforme el conocimiento sobre los tipos de una determinada entidad es refinado, permitiendo realizar un razonamiento rebatible. OBLOG-2 fue diseñado con el objetivo de modelar escenarios jurícos, en los cuales las leyes suelen ser representadas como reglas generales que pueden estar sujetas a excepciones.

Hage y Verheij [Hage y Verheij, 1995] desarrollaron una teoría para modelar el razonamiento legal denominada *Lógica Basada en Explicaciones* (RBL). En este formalismo la noción de *explicación* tiene un rol preponderante. La idea básica detras de RBL consiste en que la aplicación de una determinada regla r produce una explicación que argumenta a favor de la conclusión de r. Al discutir sobre una determinada conclusión, todas las explicaciones a favor y en contra de la misma son consideradas a fin de determinar cual de ellas prevalece. Cabe destacar que la lógica basada en explicaciones sirvió como base para el formalismo argumentativo de Verheij [Verheij, 1996].

Arno Lodder y Aimée Herczog [Lodder y Herczog, 1995] desarrollaron un sistema denominado `DiaLaw`, utilizando como herramienta a la lógica basada en explicaciones. `DiaLaw` representa el razonamiento legal en la forma de un diálogo. Esta elección es fundamentada por los autores con las con las siguientes razones. En primer lugar, la mayoría de las situaciones en la práctica legal pueden ser modeladas mediante un dialógo (como por ejemplo, juicios, debates e inclusive el razonamiento introspectivo de un abogado). Por otra parte, los diálogos hacen más fácil y natural considerar la noción de *peso de la prueba*.[8] `DiaLaw` formaliza las movidas permitidas en un discurso legal y permite realizar razonamiento por casos y razonamiento basado en reglas. Lodder y Herczog desarrollaron además un programa Prolog que implementa una versión reducida del modelo `DiaLaw`. Esta aplicación tiene como objetivo asistir en el análisis de decisiones legales y en la construcción de una justificación racional para solucionar un conflicto legal.

Ron Loui y Jeff Norman [Loui y Norman, 1995]Norman definieron un sistema para modelar el razonamiento legal basado en el uso de *razones*. Las razones con-

[8]La sección 2.3 contiene la descripción de este concepto.

sisten en estructuras asociadas a las reglas de la base de conocimiento del sistema, que contienen información relevante para la aplicación de la regla en cuestión. Loui y Norman estudiaron el problema de cómo representar razones en forma adecuada y cómo pueden las razones afectar los argumentos. El propósito de este trabajo es definir un modelo argumentativo con mayor expresividad en el cual las razones se integren en el mecanismo de inferencia.

2.4.2. Argumentación en la toma de decisiones

Las alternativas tradicionales que se emplean en el desarrollo de herramientas para la toma de decisiones ante información incompleta e incierta son los sistemas expertos o las teorías de decisión probabilísticas. Los sistemas expertos deben ser construidos de manera *ad hoc* y no son robustos ante los cambios del ambiente. Con respecto a las teorías de decisión probabilísticas, no siempre es posible obtener una descripción estadística completa y precisa del dominio en consideración. La argumentación ha sido utilizada en el desarrollo de herramientas para la toma de decisiones ante información incompleta e incierta con interesantes resultados. En general, estos sistemas obtienen las decisiones a partir del análisis de los argumentos a favor y en contra de las diferentes opciones disponibles.

La lógica argumentativa (LA) [Krause et al., 1995] fue uno de los primeros sistemas para la toma de decisiones basado en argumentación. La idea clave de este formalismo consiste en analizar la estructura de los argumentos que son relevantes a una proposición p, a fin de obtener un grado de confianza para p. Los grados de confianza constituyen una síntesis del proceso de razonamiento. En este formalismo, la estructura de los argumentos está basada en el modelo informal propuesto por Toulmin [Toulmin, 1958]. En esencia, los argumentos describen las justificaciones que sustentan a una cierta proposición. El mecanismo de inferencia obtiene todos los argumentos relevantes para un determinada sentencia y luego los grados de confianza de cada argumento se combinan para hallar un único grado de confianza para esta sentencia.

La lógica argumentativa ha sido utilizada como un componente en la arquitectura interna de agentes de Fox y Das [Fox y Das, 2000] para realizar razonamiento práctico. Se han desarrollado además herramientas para la toma de decisiones basadas en esta lógica, como es el caso del sistema de diagnóstico diseñado por Parsons y Fox [Parsons y Fox, 1997]. Posteriormente Fox y Parsons [Fox y Parsons, 1998] extendieron la lógica argumentativa a fin de que permita razonar sobre acciones. El

sistema obtenido incorpora los conceptos de *valor esperado* y *utilidades*, ingredientes tradicionales de las teorías de decisión.

Haenni [Haenni, 1998] desarrolló un acercamiento argumentativo a las teorías de decisión que integra ideas de los sistemas basados en suposiciones. Esta propuesta incorpora suposiciones dentro de la base de conocimiento, de manera análoga al framework abstracto presentado en [Bondarenko et al., 1997] [9] en el cual un argumento se interpreta como una deducción monótona a partir de un conjunto de premisas abductivas. De la misma forma que en la lógica argumentativa, todos los argumentos relevantes a una determinada hipótesis son considerados para obtener un grado de confianza de la misma. El formalismo de Haenni ha sido implementado en el sistema ABEL [10], un lenguaje para computar argumentos simbólicos y no simbólicos para una hipótesis determinada a partir de una base de conocimiento dependiente del dominio.

Gordon y Karapadilis diseñaron el sistema ZENO [Gordon y Karacapadilis, 1997] una aplicación para la toma de decisiones en contextos donde existen múltiples objetivos y múltiples participantes. Actualmente se ha implementado un prototipo de esta herramienta, que se encuentra en proceso de evaluación.

2.4.3. Negociación en sistemas multiagentes mediante argumentación

Recientemente se ha manifestado un creciente interés en aplicar sistemas argumentativos para modelar la negociación entre agentes, dado que es natural caracterizar al proceso para alcanzar un acuerdo entre dos contendientes como un intercambio de argumentos entre los mismos.

Gran cantidad de los formalismos de negociación basados en argumentación extienden a un sistema argumentativo desarrollado para modelar al conocimiento de un sólo agente por medio de un mecanismo de comunicación. De esta manera se generan argumentos que serán luego intercambiados con otros agentes a través de este protocolo. Por caso, Mora, Alferez y Schoeder [Mora et al., 1998] extienden la propuesta de Prakken y Sartor [Prakken y Sartor, 1997] [11] a un escenario multiagente. En esta trabajo los agentes cooperan durante el proceso de construcción de argumentos. Un agente A puede solicitar a un agente B que infiera un cierto conjunto

[9] Ver sección 3.5.

[10] La sigla surge del nombre en inglés: Assumption-Based Evidential Language.

[11] El sistema de Prakken y Sartor es analizado en la sección 3.3.

de conclusiones necesario para completar una prueba que conduce a la construcción de un argumento. El proceso de comunicación se implementa mediante un protocolo argumentativo que contiene las primitivas `ask`, `reply`, `propose`, `oppose` y `agree`.

El trabajo presentado en [Sierra et al., 1997] se concentra fundamentalmente en los aspectos sociales de la negociación . Este modelo está basado en un lenguaje de comunicación específico que incorpora elementos de *persuasión* [Sycara, 1990], como las primitivas `threat`, `reward` y `appeal`. Los agentes utilizan estos elementos para tratar de cambiar las preferencias, valores o creencias de otros agentes.

Parsons y Jennings [Parsons y Jennings, 1997] presentan un protocolo de negociación en particular y luego se concentran en desarrollar un modelo que provea los mecanismos necesarios para implementar este protocolo. El modelo propuesto usa la lógica argumentativa [Krause et al., 1995] para generar y evaluar las propuestas, contrapropuestas y explicaciones del sistema. Es importante destacar que las propuestas son representadas como argumentos y evaluadas en términos de su aceptabilidad. En este trabajo los agentes que participan de la negociación deben poseer una arquitectura uniforme.

Carbogim y Robertson [Carbogim y Robertson, 1999] desarrollaron un modelo de negociación un tanto diferente a las propuestas existentes en el área. En este formalismo, denominado negociación basada en contratos, el resultado de la negociación es un objeto o contrato sobre el cual se alcanza un acuerdo que es adecuado para todos los participantes. Al comienzo del proceso el contrato posee un conjunto de elementos o atributos sin especificar sobre los cuales los agentes realizan la negociación. Los agentes asignan valores a estos atributos a través de la generación y el intercambio de argumentos entre los participantes.

Stankevicius y Simari [Stankevicius y Simari, 2001] presentan un nuevo acercamiento a la negociación entre agentes basado en un modelo dialéctico. En este escenario dos o más agentes intercambian argumentos sobre una determinada tesis, que es el objeto de la negociación. Para gobernar este debate se define un modelo formal y un protocolo que establece las reglas a partir de las cuales se proponen y aceptan los argumentos.

2.5. Conclusiones

La argumentación ha estado presente en el mundo desde hace ya más de 2000 años. Este fenómeno puede atribuirse a que la argumentación constituye una forma

natural de resolver conflictos y obtener conclusiones. A partir del estudio cronológico realizado en este capítulo destacamos los trabajos de Aristóteles y Leibnitz como piedras basales de la argumentación. Ya en el siglo XX fue Toulmin quien revindicó la argumentación como una alternativa adecuada para modelar el razonamiento. Este trabajo posee el mérito de discutir a un nivel informal las ideas que medio siglo más tarde fueron implementadas en gran parte de los modelos argumentativos existentes.

La dialéctica permite modelar el proceso de argumentación como un debate en el cual se intercambian argumentos a favor y en contra de una determinada tesis. Esta conceptualización hace posible obtener un formalismo claro y sencillo sin sacrificar la expresividad del sistema. N. Rescher fue un pionero en analizar la integración de la dialéctica en la argumentación. Aunque Rescher no define un sistema formal, dado que su acercamiento es primordialmente filosófico, establece las bases del un enfoque dialéctico que no sería utilizado en los formalismos argumentativos hasta la década del '90.

En la sección 2.4 resumimos y analizamos las aplicaciones de la argumentación desarrolladas hasta la actualidad. Identificamos algunas áreas bien definidas, como son el razonamiento legal, la negociación entre agentes y la toma de decisiones. Sin embargo, creemos que la argumentación puede ser utilizada en gran cantidad de aplicaciones aunque este potencial no está siendo explotado. En particular los sistemas argumentativos constituyen una alternativa adecuada para representar el conocimiento y modelar el razonamiento de agentes inteligentes (esto es, entidades capaces de razonar, elaborar planes y actuar sobre su ambiente). Es claro que estas entidades son vitales para gran cantidad de herramientas, como por ejemplo, agentes de comercio electrónico, navegación asistida en Internet o búsqueda y procesamiento de información.

Capítulo 3

Principales Sistemas Argumentativos

En este capítulo se analizan en detalle algunos de los sistemas argumentativos más relevantes, sopesando sus virtudes e inconvenientes y destacando la evolución experimentada por cada formalismo. Entre los formalismos descriptos se encuentran los trabajos de John Pollock [Pollock, 1987], Guillermo R. Simari y Ronald P. Loui [Simari y Loui, 1992], Gerard Vreeswijk [Vreeswijk, 1997], Henry Prakken y G. Sartor [Prakken y Sartor, 1997], Phan M. Dung [Dung, 1995b] y Alejandro García [García, 2000]. Por último se discuten las conclusiones obtenidas mediante el estudio de estos sistemas.

3.1. La propuesta de John Pollock

En la década del '80, John Pollock comenzó el desarrollo del conocido proyecto OSCAR [Pollock, 1987, Pollock, 1992, Pollock, 1995] con el objetivo de definir e implementar una teoría general de racionalidad que sirviera de guía para la construcción de sistemas inteligentes. El formalismo argumentativo de Pollock fue presentado en el marco de este proyecto con el propósito de modelar el razonamiento rebatible y tiene el mérito de integrar el trabajo realizado por los filósofos en Epistemología con los estudios llevados a cabo en la comunidad de Inteligencia Artificial. A continuación se describe este sistema de acuerdo a lo expuesto en [Pollock, 1995].

El lenguaje utilizado en el formalismo de Pollock es la lógica de primer orden. Los argumentos se construyen a partir de *razones* que se codifican como un par ordenado $\langle \Gamma, p \rangle$, donde Γ es un conjunto de premisas y p una conclusión. Tanto los

elementos de Γ como p son elementos del lenguaje. Las *razones conclusivas* permiten derivar su conclusión en forma indiscutida; por el contrario las *razones prima facie* presentan evidencia rebatible a favor de la misma. En consecuencia las razones *prime facie* pueden poseer *derrotadores*, que se dividen en dos categorías: derrotadores por *rebatimiento* y derrotadores por *socavamiento*.

Los derrotadores por rebatimiento contradicen la conclusión de la razón *prima facie*. Por ejemplo, $\lceil$*Tweety vuela porque es un pájaro*$\rceil$ es derrotado por $\lceil$*Tweety no vuela porque es un pingüino*$\rceil$.[1] Formalmente estos derrotadores se definen como sigue:

Definición 3.1 (*Derrotador por rebatimiento*)
Sea $\langle \Gamma, p\rangle$ una razón *prima facie*. Una razón $\langle \Lambda, q\rangle$ es un *derrotador por rebatimiento* de $\langle \Gamma, p\rangle$ si y sólo si $q = \lceil \neg p \rceil$. ■

Los *derrotadores por socavamiento* no atacan la conclusión de la razón, sino la conexión entre las premisas y la conclusión. Por caso, $\lceil$*mis ojos ven la mesa de color rojo*$\rceil$ es una razón *prima facie* para $\lceil$*la mesa es roja*$\rceil$, pero si posteriormente se descubre que la mesa está iluminada por luz roja y los objetos iluminados por esta luz siempre se ven rojos no es razonable concluir que la mesa es roja. Sin embargo, aunque $\lceil$*la mesa está iluminada por luz roja*$\rceil$ es un derrotador de $\lceil$*la mesa es roja*$\rceil$, éste no constituye una razon para sancionar que la mesa no es de color rojo. El derrotador por socavamiento nos da una razón para cuestionar que la mesa no se vería roja a menos que fuera de color rojo. Si $\langle \Gamma, p\rangle$ es una razón *prima facie*, $\amalg\Gamma$ es la conjunción de las premisas en Γ y $\lceil P \gg Q\rceil$ se interpreta como $\lceil P$ no sería verdadero a menos que Q fuera verdadero$\rceil$ entonces cualquier razón para negar $\lceil \amalg\Gamma \gg p\rceil$ es un derrotador por socavamiento para $\langle \Gamma, p\rangle$. Formalmente:

Definición 3.2 (*Derrotador por socavamiento*)
Sea $\langle \Gamma, p\rangle$ una razón *prima facie*. Una razón $\langle \Lambda, q\rangle$ es un *derrotador por socavamiento* de $\langle \Gamma, p\rangle$ si y sólo si $q = \lceil \sim(\amalg\Gamma \gg p)\rceil$. ■

De ahora en más abreviaremos $\lceil \sim(P \gg Q)\rceil$ como $\lceil (P \otimes Q)\rceil$.

La base de datos consiste en un conjunto de premisas que constituyen la entrada del razonador. En los seres humanos estas premisas se obtienen por medio de la percepción.

[1] El operador $\lceil \cdot \rceil$ es utilizado por Pollock para denotar la transformación de una expresión al lenguaje objeto.

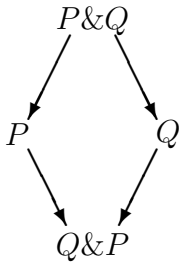


Figura 3.1: Grafo de inferencia en el sistema OSCAR

De acuerdo con Pollock, los argumentos resumen el razonamiento realizado por medio de las razones partiendo de las premisas y se clasifican en *lineales* y *suposicionales*. Los argumentos lineales son una secuencia de proposiciones, donde cada proposición es un miembro de las premisas de entrada o se puede inferir a partir de un elemento previo de la secuencia utilizando alguna razón conclusiva o *prima facie*.

Ejemplo 3.1 Consideremos el siguiente argumento para $P\&Q$ a partir de la premisa $Q\&P$.

1. $P\&Q$
2. Q
3. P
4. $Q\&P$

Este argumento puede representarse gŕaficamente como se muestra en la figura 3.1. ■

El gráfico de las relaciones de dependencia entre las razones utilizadas en la construcción del argumento se denomina *grafo de inferencia*. Pollock sostiene que el razonamiento puede entenderse como la producción de distintos argumentos que sustentan un conjunto de conclusiones y que todo este procedimiento puede representarse en un único grafo de inferencia. Esta estructura provee la herramienta principal para evaluar las creencias del razonador. No obstante, no debe minimizarse el rol de la noción de argumento. En nuestra opinion, los argumentos aportan claridad conceptual al proceso de inferencia, mientras que el uso de un único grafo lo complica innecesariamente.

El autor manifiesta que no todas las clases de razonamiento pueden modelarse mediante los argumentos lineales: el *razonamiento suposicional* también juega un rol

preponderante. Este tipo de razonamiento se caracteriza por suponer alguna proposición s que no se puede inferir de las premisas, obtener conclusiones a partir de s y finalmente descargar s para obtener la conclusión deseada sin que dependa de la suposición inicial. La condicionalización, la reducción al absurdo y el razonamiento por casos son ejemplos de razonamiento suposicional. Pollock no define formalmente los argumentos suposicionales, simplemente detalla como es posible codificar el razonamiento suposicional en el grafo de inferencia. Para esto cada nodo en el grafo se representa como un par ordenado $\langle X, p\rangle$ que consiste de una suposición X (formada por un conjunto de proposiciones) y una conclusión p.

Formalmente un *grafo de inferencia* es un conjunto de nodos, donde cada nodo consiste en un par ordenado $\langle X, p\rangle$. Este grafo puede construirse comenzando desde el conjunto de premisas y agregando nodos de acuerdo con las *reglas de inferencia*. A continuación se detallan algunos ejemplos de estas reglas.

Premisa: si p es una premisa y $\mathcal{G}$ un grafo de inferencia, entonces para cualquier suposición X es posible construir un nuevo grafo de inferencia agregando a $\mathcal{G}$ un nodo formado por $\langle X, p\rangle$ con ningún ancestro inmediato.

Suposición: si $\mathcal{G}$ es un grafo de inferencia y X es un conjunto finito de proposiciones tal que $p \in X$ entonces es posible construir un nuevo grafo de inferencia agregando a $\mathcal{G}$ un nodo formado por $\langle X, p\rangle$ con ningún ancestro inmediato.

Razón: si $\mathcal{G}$ es un grafo de inferencia formado por los nodos $\langle X, p_1\rangle, \ldots, \langle X, p_n\rangle$ y $\langle [p_1, \ldots, p_n], q\rangle$ es una razón (conclusiva o *prima facie*) entonces es posible construir un nuevo grafo de inferencia agregando a $\mathcal{G}$ un nodo formado por $\langle X, q\rangle$ con ancestros inmediatos $\langle X, p_1\rangle, \ldots, \langle X, p_n\rangle$.

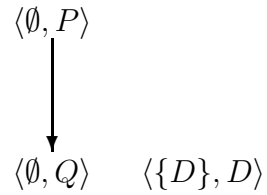
Condicionalización: si $\mathcal{G}$ es un grafo de inferencia que contiene al nodo $\langle X \cup \{p\}, q\rangle$ entonces es posible construir un nuevo grafo de inferencia agregando a $\mathcal{G}$ un nodo formado por $\langle X, (p \supset q)\rangle$ con $\langle X \cup \{p\}, q\rangle$ como ancestro inmediato.

Razonamiento por Casos: si $\mathcal{G}$ es un grafo de inferencia que contiene a los nodos $\langle X, p \vee q\rangle$, $\langle X \cup \{p\}, r\rangle$ y $\langle X \cup \{q\}, r\rangle$ entonces es posible construir un nuevo grafo de inferencia agregando a $\mathcal{G}$ un nodo $\langle X, r\rangle$ con $\langle X, p \vee q\rangle$, $\langle X \cup \{p\}, r\rangle$ y $\langle X \cup \{q\}, r\rangle$ como ancestros inmediatos.

En el sistema de Pollock la relación de derrota no se define entre pares de argumentos, sino entre pares de nodos en el grafo de inferencia. Un nodo α *derrota* a un

nodo β si contiene un derrotador (ya sea por rebatimiento o por socavamiento) de β. Cabe destacar que el razonamiento suposicional complica la noción de derrota. Si un nodo α se obtiene aplicando la razón *prima facie* $\langle \Gamma, p \rangle$ y un nodo β contiene un derrotador para $\langle \Gamma, p \rangle$ esto no garantiza que β derrota a α, pues α y β pueden basar su conclusión en suposiciones distintas.

Ejemplo 3.2 Consideremos el siguiente grafo de inferencia donde P es una razón *prima facie* para Q y D es un derrotador de esta razón.



Al analizar esta situación queda claro que la suposición de D no debería impedir la justificación de Q. ■

El ejemplo anterior evidencia que un derrotador basado en una suposición X sólo puede derrotar a un nodo basado en una suposición Y tal que $X \subseteq Y$.

A fin de comparar los nodos en conflicto, Pollock define la noción de fuerza conclusiva para cada nodo del grafo de inferencia. Para esto utiliza el concepto de *probabilidad epistémica* [Pollock, 1995] que asigna un valor denominado *grado de soporte* a cada premisa y cada razón *prima facie*. El grado de soporte de un nodo α se define entonces como el mínimo de los grados de soporte de las razones *prima facie* y las premisas utilizadas para inferir la conclusión de α, utilizando una formulación está basada en el principio del eslabón más débil. Este método posee una gran desventaja, dado que en numerosas oportunidades no es posible fijar en forma adecuada o realista los distintos grados de soporte para cada premisa y cada razón *prima facie*.

Finalmente enunciaremos la noción formal de derrota, la cual tiene en cuenta el grado de soporte de los nodos. En primer lugar se presenta la derrota por rebatimiento:

Definición 3.3 (*Rebatimiento*)
Sean α y β dos nodos en un grafo de inferencia; α *rebate* a β si y sólo si se cumplen las siguientes cláusulas:

1. β es un nodo que surge de una razón *prima facie* y sustenta una proposición q con fuerza ξ basado en una suposición Y;

2. α sustenta $\neg q$ con fuerza η basado en una suposición X, donde $X \subseteq Y$;

3. $\eta \geq \xi$.

■

En consecuencia un nodo α con una conclusión $\neg q$ derrota a un nodo β para q si β es al menos tan fuerte como α.

La derrota por socavamiento también utiliza la noción de grado de soporte.

Definición 3.4 (*Socavamiento*)
Sean α y β dos nodos en un grafo de inferencia. Se dice que α *socava* a β si y sólo si se verifican las siguientes condiciones:

1. β es un nodo que surge de una razón *prima facie* y sustenta una proposición q con fuerza ξ basado en una suposición Y y $p_1, \ldots, p_n$ son las proposiciones sustentadas por sus ancestros inmediatos;

2. α sustenta a $(p_1 \& \ldots \& p_n) \otimes q$ con fuerza η basado en una suposición X, donde $X \subseteq Y$;

3. $\eta \geq \xi$.

■

La relación de derrota puede denotarse en el grafo de inferencia mediante un nuevo conjunto de enlaces denominados *enlaces de derrota*. Si α y β son dos nodos en un grafo de inferencia entonces $\langle \alpha, \beta \rangle$ es un enlace de derrota si y sólo si α surge de una razón *prima facie* y β derrota a α.

Las proposiciones justificadas del sistema son las conclusiones de los nodos que no están derrotados en el grafo de inferencia. Formalmente:

Definición 3.5 (*Justificación*)
Una creencia está *justificada* en grado δ si y sólo si está basada en un nodo del grafo de inferencia que no está derrotado y posee una fuerza mayor o igual que δ. ■

Por lo tanto es mandatorio definir como se asignan el estado a los nodos de un grafo de inferencia de acuerdo a la relación de derrota existente entre los mismos. El autor distingue dos situaciones en las que un nodo α debe estar derrotado:

- α es derrotado por un nodo sin derrotar;

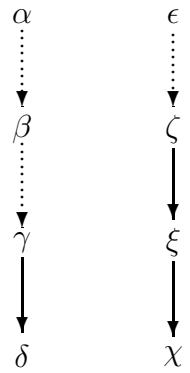


Figura 3.2: Derrota Colectiva en el sistema de Pollock

- α se infiere a partir de un nodo derrotado.

De acuerdo a esta política se enuncia la siguiente formalización:

1. Un nodo η tal que ni η ni ninguno de sus ancestros está derrotado por algún nodo en el grafo de inferencia no está derrotado. Estos nodos se denominan `D-iniciales`.

2. Si los ancestros inmediatos de un nodo η no están derrotados y todos los nodos que derrotan a η están derrotados entonces η no está derrotado.

3. Si η posee un ancestro inmediato derrotado o existe un nodo sin derrotar que lo derrota entonces η es un nodo derrotado.

Esta definición parece funcionar correctamente en casos sencillos, pero no es capaz de resolver correctamente algunos escenarios más complejos como la *derrota colectiva*. Esta situación se caracteriza por la existencia de un conjunto S de nodos tal que cada miembro de S está derrotado por algún otro nodo en S.

Ejemplo 3.3 Analicemos el grafo de inferencia de la figura 3.2. Las líneas punteadas representan inferencias por medio de razones prima facie y las líneas rectas inferencias conclusivas.[2] Supongamos que β derrota a ζ y viceversa. De acuerdo al principio anterior no es posible asignar ningún estado ni a β ni a ζ y por consiguiente γ,δ,ξ y χ tampoco reciben un estado correspondiente. ■

Otro problema del sistema de Pollock es originado por la carencia de una definición adecuada de la noción de argumento. En el grafo de inferencia cada nodo

[2]Esta convención se utilizará en el resto de la presente sección.

sustenta una conclusión q, pero no se exige consistencia en el conjunto de elementos utilizados para derivar q, lo que conduce a situaciones paradójicas. Por medio de derivaciones inconsistentes es posible inferir cualquier elemento del lenguaje. Los nodos que sufren de esta anomalía se denominan *auto derrotados*. Una característica común de estos nodos es que algunos de sus ancestros se atacan entre sí.

Para resolver los casos de derrota colectiva y los argumentos auto derrotados el autor presenta la siguiente extensión de la formulación anterior.

1. Los nodos `D-iniciales` no están derrotados.
2. Los nodos auto derrotados están `totalmente derrotados`.
3. Si η no es un nodo auto derrotado, sus ancestros inmediatos no están derrotados y todos los nodos que derrotan a η están `totalmente derrotados` entonces η no está derrotado.
4. Si η posee un ancestro inmediato totalmente derrotado o existe un nodo sin derrotar que lo derrota entonces η es un nodo `totalmente derrotado`.
5. En cualquier otro caso η está `provisionalmente derrotado`.

En esta clasificación se introducen dos tipos de derrota para solucionar los casos de derrota colectiva. Por caso, en el ejemplo 3.3 β, ζ, γ, δ,ξ y χ son catalogados como provisionalmente derrotados. Estos nodos no pertenecen a las conclusiones del sistema, pero tampoco están totalmente derrotados, esto es, no son rechazados en forma categórica. La diferencia entre los nodos totalmente derrotados y los nodos provisionalmente derrotados recae en que estos últimos conservan la capacidad de derrotar a sus adversarios, es decir, son capaces de propagar la derrota. Por lo tanto su situación está indecisa y esta indecisión se propaga a cualquier nodo que esté en conflicto o descienda de algún nodo provisionalmente derrotado. Los argumentos auto derrotados se excluyen explícitamente mediante la segunda claúsula, marcándolos como totalmente derrotados.

Este principio fue propugnado por Pollock en diversas publicaciones. Sin embargo, en [Pollock, 1995] el autor realiza una autocrítica de su trabajo. Pollock presenta varios ejemplos en donde utilizar esta formulación no conduce a una respuesta adecuada:

Ejemplo 3.4 Consideremos el grafo de inferencia diagramado en la figura 3.3. El nodo $P \otimes Q$ está auto derrotado, ya que derrota uno de sus ancestros y de acuerdo

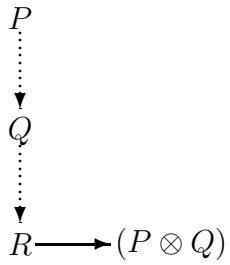


Figura 3.3: Un ejemplo problemático para OSCAR

con el último principio está totalmente derrotado. Por lo tanto ni Q ni R están derrotados. Este resultado es un tanto peculiar, pues si un nodo no está derrotado es deseable que sus consecuencias deductivas no estén derrotadas. ■

Pollock observa que los formalismos basados en modelos (como la lógica default y circunscripción) obtienen la respuesta correcta ante situaciones como la del ejemplo 3.4. La ventaja de estos sistemas radica en que son capaces de analizar las relaciones entre los argumentos provisoriamente derrotados. Por ejemplo, ante dos nodos provisoriamente derrotados $\sim R$ y R permiten concluir que $\sim R$ es aceptable cuando R no es aceptable y viceversa. Para incorporar esta capacidad a su teoría Pollock propone la siguiente definición:

Definición 3.6 (*Estado de asignación parcial*)
Una asignación σ de los estados `derrotado` y `no derrotado` a un subconjunto de nodos de un grafo de inferencia $\mathcal{G}$ es un *estado de asignación parcial* si y sólo si se verifican las siguientes condiciones:

1. σ asigna `no derrotado` a todos los nodos `D-iniciales`.

2. σ asigna `no derrotado` a un nodo α si y sólo si asigna `no derrotado` a todos sus ancestros inmediatos y `derrotado` a todos los nodos que derrotan a α.

3. σ asigna `derrotado` a un nodo α si y sólo si α posee un ancestro inmediato al que se catalogó como `derrotado` o existe un nodo β tal que β derrota a α y se le asignó el estado `no derrotado`.

■

La noción de estado de asignación parcial soluciona los problemas existentes con los ciclos de derrota impar, en donde no es posible asignar un estado a todos los

nodos del grafo. La definición 3.6 permite clasificar un subconjunto de los nodos problemáticos y las asignaciones maximales son consideradas para determinar el estado de los nodos.

Definición 3.7 (*Estado de asignación*)
Sea σ un estado de asignación parcial de un grafo $\mathcal{G}$; σ es un *estado de asignación* si y sólo si σ no esta propiamente contenido en ningún otro estado de asignación parcial de $\mathcal{G}$. ■

Ejemplo 3.5 Supongamos la existencia de un ciclo de derrota entre tres nodos Q, S y R, tal que Q derrota a S, S derrota a U y U derrota a R. Es claro que no existe ningún estado parcial que pueda clasificar a Q, S y R. En consecuencia en algunos estados maximales R está derrotado y en otros no. Lo mismo sucede para Q y S. Luego todos los nodos se clasifican como provisionalmente derrotados. ■

En base a las definiciones 3.1 y 3.6 se dice que un nodo α está sin derrotar si y sólo si todo estado de asignación posible cataloga a α como no derrotado; de lo contrario α está derrotado. Un nodo derrotado se denomina totalmente derrotado si ningún estado de asignación lo considera no derrotado, sino está provisionalmente derrotado.

Ejemplo 3.6 La nueva definición resuelve en forma correcta el ejemplo de la figura 3.3. Los nodos Q, R y $P \otimes R$ están provisionalmente derrotados y el nodo P permanece sin derrotar. ■

Pollock utiliza dos enfoques diferentes para caracterizar las inferencias sancionadas por el sistema. En primer lugar define la noción de conclusiones *idealmente garantizadas* en base al conjunto $\mathcal{G}$, compuesto por todos los nodos que el razonador puede construir a partir de las premisas.

Definición 3.8 (*Idealmente garantizado*)
Un par $S = \langle X, \sigma \rangle$ formado por una razón y una conclusión está *idealmente garantizado* en grado δ relativo a un conjunto de premisas P si y sólo si existe un nodo α con fuerza conclusiva mayor o igual a δ tal que:

1. $\alpha \in \mathcal{G}$,

2. α no está derrotado con respecto a $\mathcal{G}$ y

3. α sustenta a S.

■

Esta definición establece las creencias que poseería un razonador sin restricciones de recursos, es decir, si fuera capaz de producir y analizar todos los argumentos relevantes.

La segunda caracterización de las conclusiones del sistema posee un enfoque computacional. En este acercamiento el razonamiento comienza con el conjunto de premisas y a partir de este produce nuevos nodos que se agregan al grafo de inferencia. Sean $\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots$ los nodos que pueden construirse en caso de poseer recursos ilimitados y sea $\mathcal{G}_i$ el conjunto de los primeros i nodos $\alpha_1, \alpha_2, \ldots, \alpha_i$. Las conclusiones justificadas hasta el iésimo paso se definen como sigue:

Definición 3.9 (*Justificación epistémica*)
Un par $S = \langle X, \sigma \rangle$ formado por una razón y una conclusión está *justificado* en grado δ en el estadío i si y sólo si existe un nodo α con fuerza conclusiva mayor o igual a δ tal que:

1. $\alpha \in \mathcal{G}_i$,

2. α no está derrotado con respecto a $\mathcal{G}_i$ y

3. α sustenta a S.

■

La justificación epistémica permite obtener el estado actual de un proposición. Conforme el proceso continúa un nodo puede fluctuar entre los estados *justificado* y *no justificado*. De esta manera se modela el razonamiento acotado por los recursos. Para formalizar las inferencias del sistema en base a la definición 3.1 Pollock se vale del límite al que tiende la justificación epistémica a medida que avanza el razonamiento.

Definición 3.10 (*Garantización*)
Un par $S = \langle X, \sigma \rangle$ formado por una razón y una conclusión está *garantizado* en grado δ si y sólo si existe i tal que para todo $j \geq i$ se cumple que S está justificado en grado δ en el estadío j. ■

La definición 3.10 modela el comportamiento real del sistema, mientras que el concepto de idealmente garantizado representa el comportamiento esperado. Por lo tanto sería interesante que el conjunto de conclusiones obtenido mediante la definición 3.10 coincida con el conjunto de inferencias idealmente garantizadas. Pollock prueba que si bien estas nociones no son equivalentes es factible relacionarlas si se cumple un determinado conjunto de condiciones.

El trabajo de John Pollock posee sólidas bases epistemológicas y está respaldado por más de treinta años de investigación. La teoría desarrollada es sumamente abarcativa, abordando tanto argumentos lineales y suposicionales como deductivos y probabilísticos. Sin embargo, en algunos casos esta amplitud de la teoría desarrollada compromete su aplicabilidad a situaciones prácticas. Por ejemplo, la noción de fuerza conclusiva desarrollada por Pollock no resulta adecuada, pues en numerosas situaciones no es posible asignar valores probabilísticos a las premisas y razones *prima facie* que componen el conocimiento.

Otro aspecto interesante de la investigación de Pollock es el concepto de justificación epistémica, que permite modelar una tipo de razonamiento acotado por los recursos. Este tema ha sido ignorado en la mayoría de los sistemas argumentativos.

Por último es importante resaltar que uno de los aportes principales del trabajo de Pollock consiste en una metodología que combina el desarrollo teórico del sistema con su implementación. El autor afirma que la implementabilidad de la teoría es una condición necesaria para su correctitud. En consecuencia la confección de un agente basado en la arquitectura OSCAR constituye un campo de prueba para depurar la teoría de racionalidad subyacente. Este principio puede ser aplicado a cualquier sistema de Representación de Conocimiento y Razonamiento.

3.2. El sistema argumentativo de Simari y Loui

En el año 1989 Simari presentó su disertación doctoral denominada *"A mathematical treatment of defeasible reasoning and its implementation"* [Simari, 1989] en la cual se sientan las bases de un sistema argumentativo que fue uno de los pioneros en dar un tratamiento matemático formal a la argumentación rebatible. Las principales contribuciones de esta tesis fueron publicadas posteriormente en un artículo homónimo de la revista *Artificial Inteligence* [Simari y Loui, 1992]. A continuación se reseña el mencionado formalismo de acuerdo a las definiciones, conceptos y propiedades detalladas en este artículo.

La propuesta de Simari y Loui combina la noción de especificidad definida por D. Poole [Poole, 1985] con la teoría de justificación por niveles presentada por J. Pollock. Los autores postulan que por medio de esta fusión es posible obtener "lo mejor de ambos mundos" [Chesñevar, 1996], considerando que Poole define elegantemente el criterio de especificidad, pero no describe adecuadamente como aplicar este criterio a las distintas interacciones entre argumentos y Pollock especifica correctamente el proceso de justificación de argumentos, pero no define un criterio de comparación adecuado.

El sistema formal para la representación de conocimiento y razonamiento rebatible 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 [Davis, 1989] que resulte conveniente. Los autores definen además la noción de *reglas rebatibles*. Estos elementos son relaciones metalingüísticas sobre los elementos de $\mathcal{L}$, que se codifican mediante expresiones de la forma $\alpha \succ \beta$, donde α y β son fórmulas bien formadas de $\mathcal{L}$. Pragmaticamente, para que una regla rebatible tenga sentido α y β deben contener variables libres. La semántica de la relación denotada por el conectivo ' $\succ$ ' puede interpretarse 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 elemento y las instancias de r se obtienen reemplazando consistentemente todas las variables libres por constantes de $\mathcal{L}$.

El conjunto de sentencias de $\mathcal{L}$, denotado por $Sent(\mathcal{L})$, puede ser particionado en el subconjunto de información *necesaria*, denotado por $Sent_N(\mathcal{L})$ y compuesto por sentencias con variables libres, y el subconjunto de información *contingente*, denotado como $Sent_C(\mathcal{L})$ y formado por sentencias básicas. Es claro que $Sent_C(\mathcal{L}) \cap Sent_N(\mathcal{L}) = \emptyset$.

En el formalismo de Simari y Loui el conocimiento de un agente es representado mediante un par $(\mathcal{K}, \Delta)$, donde $\mathcal{K}$ es un subconjunto de $Sent(\mathcal{L})$ y Δ es un conjunto finito de reglas rebatibles. $\mathcal{K}$ modela la información fuerte o segura del agente mientras que Δ codifica información tentativa. El conjunto $\mathcal{K}$ también puede ser particionado en dos subconjuntos:

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

tal que $\mathcal{K} = \mathcal{K}_n \cup \mathcal{K}_c$. Es importante destacar que el conjunto $\mathcal{K}$ representa información irrefutable y debe por lo tanto ser consistente, esto es, $\mathcal{K} \nvdash \bot$.

$\mathcal{K}_c$= { ornitorrinco(will),
murciélago(mina) }

$\mathcal{K}_n$= { monotrema(X) $\rightarrow$ mamífero(X),
ornitorrinco(X) $\rightarrow$ monotrema(X),
equidna(X) $\rightarrow$ monotrema(X),
vivíparo(X) $\rightarrow$ $\neg$nace_huevo(X) }

Δ = { mamífero(X) $\succ$ vivíparo(X),
mamífero(X) $\succ$ $\neg$vuela(X),
monotrema(X) $\succ$ nace_huevo(X),
ornitorrinco(X), enfermo(X) $\succ$ $\neg$cria_hijos(X),
vivíparo(X) $\succ$ $\neg$construye_nido(X),
ornitorrinco(X) $\succ$ cria_hijos(X),
cria_hijos(X) $\succ$ construye_nido(X),
murciélago(X) $\succ$ vuela(X) }

Figura 3.4: Una base de conocimiento en el sistema de Simari y Loui

Ejemplo 3.7 En la figura 3.4 se detalla una base de conocimiento que codifica información acerca de distintos ordenes de animales. Los hechos contingentes expresan que `will` es un ornitorrinco y `mina` es un murciélago. El conjunto $\mathcal{K}_n$ modela las siguientes aserciones: todo monotrema[3] es un mamífero, tanto los ornitorrincos como los equidnas pertenecen a los monotremas y si un animal es vivíparo entonces no nace de huevos.
La información rebatible contiene las reglas: un mamífero es generalmente un animal vivíparo, en su mayoría los mamíferos no pueden volar, los murciélagos son animales voladores, los mamíferos monotremas suelen nacer de huevos, un ornitorrinco enfermo no puede criar a sus hijos, un animal vivíparo generalmente no construye nidos, los ornitorrincos normalmente crían a sus hijos y los animales que crían a sus hijos construyen nidos o madrigueras para procrear. ■

Dada una base de conocimiento $(\mathcal{K}, \Delta)$ el mecanismo de inferencia del sistema decide que hechos están *justificados* a partir de la información contenida en ella. Los

[3]El orden de los monotremas es una subcategoría de los mamíferos formada por especias primitivas exclusivas de la región australiana.

hechos justificados conforman las creencias del agente. Para definir correctamente este concepto se enuncian las principales nociones que componen al formalismo de Simari y Loui.

Definición 3.11 (*Consecuencia rebatible - Derivación rebatible*)
Sea A un miembro de $Sent(\mathcal{L})$ y $\Gamma = \{A_1, A_2, \ldots, A_n\}$ un conjunto formado por elementos de $\mathcal{K}$ 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 $B_1, \ldots, B_m$, 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 usaremos el símbolo $\Delta^{\downarrow}$ para representar a todas las instancias de los elementos de Δ que pueden obtenerse utilizando constantes individuales de $\mathcal{L}$. Considerando que no hay restricciones de consistencia sobre el conjunto Δ, en algunos casos es posible obtener derivaciones rebatibles para hechos complementarios. El sistema provee un mecanismo de inferencia con la capacidad de decidir que conclusiones son finalmente aceptadas. La noción de *argumento* juega un rol preponderante en este mecanismo.

Definición 3.12 (*Estructura de argumento - Argumento*)
Sea $h \in Sent_C(\mathcal{L})$, $\mathcal{K} = \mathcal{K}_n \cup \mathcal{K}_c$ y Δ un conjunto de reglas rebatibles. Se dice que un conjunto T de $\Delta^{\downarrow}$ es un *argumento* para h en el contexto $\mathcal{K}$, denotado por $\langle \mathcal{T}, h \rangle_{\mathcal{K}}$ o $\langle \mathcal{T}, h \rangle$ cuando el contexto es obvio, si y sólo si se cumple alguna de las siguientes condiciones:

1. $\mathcal{K} \cup T \mathrel{|\!\sim} h$.
2. $\mathcal{K} \cup T \not\vdash \bot$.
3. $\nexists\, T' \subset T, \mathcal{K} \cup T' \mathrel{|\!\sim} h$.

El par $\langle \mathcal{T}, h \rangle_{\mathcal{K}}$ se denomina *estructura de argumento*. ■

Definición 3.13 (*Subargumento*)
Sea $\langle \mathcal{T}, h \rangle$ una estructura de argumento para h y $\langle \mathcal{S}, j \rangle$ una estructura de argumento para j. Se dice que $\langle \mathcal{S}, j \rangle$ es un *subargumento* de $\langle \mathcal{T}, h \rangle$ si y sólo si $S \subseteq T$. ■

Los argumentos representan piezas de razonamiento tentativo que sustentan una determinada conclusión. La primer claúsula de la definición 3.12 exige la existencia de una derivación para la conclusión del argumento que este basada en $\mathcal{K}$ y el conjunto de reglas rebatibles que componen al mismo. La segunda condición impide la formación de argumentos inconsistentes que originan situaciones paradójicas. De esta forma el sistema Simari y Loui soluciona en forma elegante el problema de los argumentos auto derrotados, abordado posteriormente en distintos trabajos sobre argumentación rebatible [Prakken y Vreeswijk, 1999, Pollock, 1995] Finalmente el conjunto T debe ser minimal, es decir, no debe contener ninguna regla innecesaria para la obtención de la conclusión h. Esta propiedades hacen de la definición 3.12 una de las formalizaciones más precisas y adecuadas de la noción de argumento.

Aunque un argumento está basado en la existencia de una derivación, solamente está compuesto por el conjunto de reglas rebatibles utilizadas en la misma. Esta situación sorprende a Prakken y Vreeswijk [Prakken y Vreeswijk, 1999] quienes manifiestan:

> *"...La noción de argumento desarrollada en el sistema Simari y Loui es un tanto inusual, dado que no hace referencia a un árbol o a un encadenamiento de reglas de inferencia: un argumento se define como una colección no ordenada de reglas que juntas implican una determinada conclusión...."*

No obstante, al comprender el propósito de los autores, la definición 3.12 resulta lógica y elegante. Tal como menciona Chesñevar en [Chesñevar, 1996], los argumentos resumen la información rebatible que se utilizó para obtener su conclusión, permitiendo de esta forma concentrar en ella el proceso de inferencia. El resto de la derivación está compuesto por información segura e indiscutible que no es preciso analizar.

Ejemplo 3.8 Las estructuras de argumentos que se muestran a continuación pueden construirse a partir de la base de conocimiento del ejemplo 3.7.

- $\langle \mathcal{T}_1, \neg$`construye_nido(will)`$\rangle$, donde

$$\mathcal{T}_1 = \{\texttt{vivíparo(will)} \succ \neg\texttt{construye\_nido(will)},\ \texttt{mamífero(will)} \succ \texttt{vivíparo(will)}\}$$

- $\langle \mathcal{T}_2, \texttt{construye\_nido(will)} \rangle$, donde

  $$\mathcal{T}_2 = \{\texttt{ornitorrinco(will)} \succ \texttt{cria\_hijos(will)},\ \texttt{cria\_hijos(will)} \succ \texttt{construye\_nido(will)}\}$$

- $\langle \mathcal{T}_3, \texttt{vuela(mina)} \rangle$, donde

  $$\mathcal{T}_3 = \{\texttt{murcielago(mina)} \succ \texttt{vuela(mina)}\}$$

- $\langle \mathcal{T}_4, \texttt{vivíparo(will)} \rangle$, donde

  $$\mathcal{T}_4 = \{\texttt{mamífero(will)} \succ \texttt{vivíparo(will)}\}$$

- $\langle \mathcal{T}_5, \texttt{nace\_huevo(will)} \rangle$, donde

  $$\mathcal{T}_5 = \{\texttt{monotrema(will)} \succ \texttt{nace\_huevo(will)}\}$$

- $\langle \mathcal{T}_6, \neg\texttt{vuela(mina)} \rangle$, donde

  $$\mathcal{T}_6 = \{\texttt{mamífero(mina)} \succ \neg\texttt{vuela(mina)}\}$$

- $\langle \mathcal{T}_7, \neg\texttt{cria\_hijos(will)} \rangle$, donde

  $$\mathcal{T}_7 = \{\texttt{ornitorrinco(will),enfermo(will)} \succ \neg\texttt{cria\_hijos(will)}\}$$

Cabe acotar que $\langle \mathcal{T}_4, \texttt{vivíparo(will)} \rangle$ es un subargumento de $\langle \mathcal{T}_1, \neg\texttt{construye\_nido(will)} \rangle$. ■

En algunas situaciones dos argumentos pueden ser contradictorios (*i.e.*, su aceptación conjunta produce una inconsistencia). Este concepto se define formalmente a través de la relación de desacuerdo.

Definición 3.14 (*Desacuerdo*)
Sea T_1 una estructura de argumento para h_1 y T_2 una estructura de argumento para h_2. Se dice que T_1 y T_2 están en *desacuerdo* si, y sólo si, $\mathcal{K} \cup \{h_1, h_2\} \vdash \bot$. ■

Otra de las relaciones entre argumentos es la *contraargumentación*. Esta puede interpretarse como un refinamiento de la noción de desacuerdo que permite analizar la estructura interna de un argumento.

Definición 3.15 (*Contraargumento - Punto de contraargumentación*)
Se dice que una estructura de argumento $\mathcal{T}_1$ para h_1 *contraargumenta* a una estructura de argumento $\mathcal{T}_2$ para h_2 en un hecho h si y sólo si existe un subargumento $\langle \mathcal{T}, h \rangle$ de $\langle \mathcal{T}_2, h_2 \rangle$ tal que $\langle \mathcal{T}_1, h_1 \rangle$ está en desacuerdo con $\langle \mathcal{T}, h \rangle$. El hecho h se denomina *punto de contrargumentación.* ■

Dos argumentos en conflicto pueden compararse mediante el criterio de *especificidad*, que establece un orden de preferencia entre los mismos. La especificidad favorece argumentos que contengan mayor información o sustenten su conclusión en forma más directa. Seguidamente se enuncia su definición formal.

Definición 3.16 (*Estrictamente más específica*)
Dadas dos estructuras de argumentos $\langle \mathcal{T}_1, h_1 \rangle$ y $\langle \mathcal{T}_2, h_2 \rangle$, se dice que $\langle \mathcal{T}_1, h_1 \rangle$ es *estrictamente más específica* que $\langle \mathcal{T}_2, h_2 \rangle$, denotado como $\langle \mathcal{T}_1, h_1 \rangle \succ_{\text{spec}} \langle \mathcal{T}_2, h_2 \rangle$, si y sólo si se verifica que:

1. $\forall e \in \mathit{Sent}(\mathcal{L})$ tal que e activa no trivialmente a $\mathcal{T}_1$, *i.e.*, $\mathcal{K}_n \cup \{e\} \cup \mathcal{T}_1 \vdash h_1$ y $\mathcal{K}_n \cup \{e\} \nvdash h_1$, se verifica que e activa a $\mathcal{T}_2$ ($\mathcal{K}_n \cup \{e\} \cup \mathcal{T}_2 \vdash h_2$).

2. $\exists e \in \mathit{Sent}(\mathcal{L})$ tal que e no activa a $\mathcal{T}_1$ ($\mathcal{K}_n \cup \{e\} \cup \mathcal{T}_1 \nvdash h_1$) y e activa no trivialmente a $\mathcal{T}_2$, *i.e.*, $\mathcal{K}_n \cup \{e\} \cup \mathcal{T}_2 \vdash h_2$ y $\mathcal{K}_n \cup \{e\} \nvdash h_2$.

Una sentencia e activa a un argumento $\langle \mathcal{T}, h \rangle$ si es posible construir una derivación para h a partir de $\mathcal{K}_n \cup \mathcal{T} \cup \{e\}$. ■

Diversos autores [Prakken y Sartor, 1997, Vreeswijk, 1993, Pollock, 1995] han criticado la especificidad y cuestionado la aplicabilidad del sistema por utilizar este criterio. Cabe destacar que el carácter modular del formalismo de Simari y Loui permite redefinirlo en forma sencilla para reemplazar la 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 la especificidad tiene la ventaja de ser automatizable. Esta propiedad resulta sumamente atractiva, ya que facilita la codificación y la actualización de la base de conocimiento.

El uso de un criterio de comparación permite extender naturalmente la noción de contrargumentación incorporando la capacidad de decidir entre los argumentos en pugna.

Definición 3.17 (*Derrota*)
Se dice que un argumento $\mathcal{T}_1$ para h_1 *derrota* a un argumento $\mathcal{T}_2$ para h_2 si y sólo si existe un subargumento $\langle \mathcal{T}, h \rangle$ de $\langle \mathcal{T}_2, h_2 \rangle$ tal que:

1. $\langle \mathcal{T}_1, h_1 \rangle$ contraargumenta a $\langle \mathcal{T}_2, h_2 \rangle$ en h y
2. $\langle \mathcal{T}_1, h_1 \rangle$ es estrictamente más específico que $\langle \mathcal{T}, h \rangle$.

■

Ejemplo 3.9 Al considerar los argumentos del ejemplo 3.8, es claro que $\mathcal{T}_1$ está en desacuerdo con $\mathcal{T}_2$. Por otra parte $\mathcal{T}_5$ está en desacuerdo con $\mathcal{T}_4$ y en consecuencia, aplicando la definición 3.15, $\mathcal{T}_1$ contraargumenta a $\mathcal{T}_4$.
La relación de derrota se ilustra mediante los argumentos $\mathcal{T}_3$ y $\mathcal{T}_6$. $\mathcal{T}_3$ está basado es información más específica, dado que los murciélagos son una subcategoría de los mamíferos. Es fácil ver que existen escenarios (*e.g.*, `murciélago(mina)`) que activan a $\mathcal{T}_3$ y no activan a $\mathcal{T}_6$; por tanto $\mathcal{T}_3$ derrota a $\mathcal{T}_6$. ■

El proceso de inferencia se define a partir de los conceptos de contraargumentación y derrota. Para determinar si un hecho h pertenece al conjunto de creencias el sistema analiza las estructuras de argumentos basadas $(\mathcal{K}, \Delta)$ que sustentan al hecho h. Dado un argumento $\langle \mathcal{T}, h \rangle$ puede existir un conjunto I de estructuras de argumento que *interfieren* con $\langle \mathcal{T}, h \rangle$, *i.e.*, que contraargumentan a $\langle \mathcal{T}, h \rangle$. Algunos de estos argumentos pueden derrotar a $\langle \mathcal{T}, h \rangle$. Sin embargo, es posible que estos *derrotadores* sean a su vez vencidos por otros argumentos; si todos son derrotados la estructura de argumento original $\langle \mathcal{T}, h \rangle$ es reinstanciada. Esta situación motiva la siguiente caracterización inductiva del conjunto de argumentos justificados, basada en la definición por niveles de Pollock [Pollock, 1987].

1. Todos los argumentos que se pueden construir a partir del contexto $\mathcal{K}$ y de las reglas en $\Delta^{\downarrow}$ son *argumentos de soporte* (`argumento-S`) y *argumentos de interferencia* (`argumento-I`) en el nivel 0.
2. Una estructura de argumento $\langle \mathcal{T}_1, h_1 \rangle$ es un `argumento-S` en el nivel $n+1$ si, y sólo si, no existe un `argumento-I` en el nivel n que contraargumente a $\langle \mathcal{T}_1, h_1 \rangle$.
3. Una estructura de argumento $\langle \mathcal{T}_1, h_1 \rangle$ es un `argumento-I` en el nivel $n+1$ si y sólo si no existe un `argumento-I` en el nivel n que derrote a $\langle \mathcal{T}_1, h_1 \rangle$.

Definición 3.18 (*Justificación*)
Una estructura de argumento $\langle \mathcal{T}, h \rangle$ *justifica* a su conclusión h si y sólo si existe un m tal que para todo $n \geq m$ $\langle \mathcal{T}, h \rangle$ es un `argumento-S` en el nivel n. Un hecho h está *justificado* si existe un argumento $\langle \mathcal{T}, h \rangle$ tal que $\langle \mathcal{T}, h \rangle$ justifica a h. ■

A partir de la definición 3.18 es posible colegir que el conjunto `argumentos-I` debe estar formado por todos los argumentos que sobreviven el proceso de derrota por niveles. Estos argumentos tienen autoridad suficiente para impedir que otros justifiquen a sus conclusiones y por consiguiente determinan quienes pertenecen al conjunto de `argumentos-S`. Por lo tanto para que un argumento T permanezca en los `argumentos-S` es necesario que no exista ningún argumento que *contraataque* a T en el conjunto de `argumentos-I`.

Ejemplo 3.10 Al calcular los conjuntos de `argumentos-S` y `argumentos-I` sobre la base de conocimiento del ejemplo 3.7 se obtiene la siguiente secuencia:

	`Argumentos-S`	`Argumentos-I`
Nivel 0	$\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_7\}$	$\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_7\}$
Nivel 1	$\{\mathcal{T}_3, \mathcal{T}_5, \mathcal{T}_7\}$	$\{\mathcal{T}_3, \mathcal{T}_5, \mathcal{T}_7\}$
Nivel 2	$\{\mathcal{T}_3, \mathcal{T}_5, \mathcal{T}_7\}$	$\{\mathcal{T}_3, \mathcal{T}_5, \mathcal{T}_7\}$

A partir del segundo nivel todos los niveles son iguales. Cabe observar que aunque en este ejemplo los conjuntos de `argumentos-I` y `argumentos-S` coinciden en todos los niveles esta propiedad no se cumple en general, tal como se evidencia en el ejemplo 3.11. ■

Prakken y Vreeswijk analizan esta definición en [Prakken y Vreeswijk, 1999] donde manifiestan que la noción de justificación en el formalismo de Simari y Loui es equivalente a la siguiente formulación:

- Todos los argumentos que pueden construirse son `argumentos-S` en el nivel 0.
- Un argumento es un `argumento-S` en el nivel $n + 1$ si no está derrotado por un `argumento-S` presente en el nivel n.
- Un argumento T está justificado si y sólo si existe un m tal que para todo $n \geq m$ se cumple que T es un `argumento-S` en el nivel n.

No obstante, puede observarse que estas nociones no son equivalentes. La caracterización de Prakken cataloga como justificado a todo argumento que no esté derrotado por un argumento en el conjunto de `argumentos-I` de la definición 3.18, incluyendo erróneamente a los argumentos que sólo son contraatacados, tal como se observa en el siguiente ejemplo:

Ejemplo 3.11 Consideremos una base de conocimiento $(\mathcal{K}, \Delta)$ tal que sólo dos argumentos $\mathcal{A}, \mathcal{B}$ pueden construirse a partir de la misma. Supongamos que $\mathcal{A}$ contrataca a $\mathcal{B}$ y $\mathcal{B}$ contrataca a $\mathcal{A}$ y estos argumentos no están conectados por la relación de derrota. El sistema de Simari y Loui no sanciona ni $\mathcal{A}$ ni $\mathcal{B}$ como justificados, mientras que la definición de H. Prakken clasifica como justificados ambos argumentos. ■

3.2.1. El sistema de Simari, Chesñevar y García

En *"The role of dialectics in defeasible argumentation"* [Simari et al., 1994] se define una extensión del formalismo de Simari y Loui cuya principal contribución consiste en una caracterización dialéctica del proceso de justificación. La definición de justificación en términos de argumentos activos es elegante, pero poco intuitiva y difícil de manejar computacionalmente. Por otra parte, para decidir si un hecho h está o no justificado deben analizarse la totalidad de los argumentos. Sin embargo, en la implementación del sistema restringido a cláusulas Horn desarrollado en [Simari, 1989] ya se encontraba presente la idea de un método de obtención de inferencias guiado por las consultas, que solamente tuviera en cuenta los argumentos involucrados con el hecho h en consideración. Esta idea no fue formalizada hasta la introducción del concepto de *árboles dialécticos* en [Simari et al., 1994]. El uso de esta estructura para conceptualizar el proceso de inferencia posibilitó el análisis de diversos tipos de *argumentación falaz* y el desarrollo de soluciones para estos problemas dentro del nuevo formalismo.

Las principales definiciones existentes en [Simari y Loui, 1992] no experimentan cambios notables en el nuevo sistema, exceptuando la noción de contrargumentación (que detallaremos más adelante) y la relación de derrota, la cual se extiende mediante una distinción entre dos categorías de derrotadores: *propios* y *de bloqueo*. Se dice que $\langle \mathcal{T}_1, h_1 \rangle$ es un *derrotador propio* de $\langle \mathcal{T}_2, h_2 \rangle$ si existe un subargumento $\langle \mathcal{T}, h \rangle$ de $\langle \mathcal{T}_2, h_2 \rangle$ tal que $\langle \mathcal{T}_1, h_1 \rangle$ contraargumenta a $\langle \mathcal{T}_2, h_2 \rangle$ en h y $\langle \mathcal{T}_1, h_1 \rangle$ es estrictamente más específico que $\langle \mathcal{T}, h \rangle$. Si no existe relación entre $\langle \mathcal{T}, h \rangle$ y $\langle \mathcal{T}_1, h_1 \rangle$ usando la

noción especificidad $\langle \mathcal{T}_1, h_1 \rangle$ es un *derrotador de bloqueo*. Este concepto resume el protagonismo de la relación de contraargumentación en la definición de derrota. El mecanismo de inferencia se basa en la noción de árboles dialécticos.

Definición 3.19 (*Árbol dialéctico*)
Sea $\langle \mathcal{A}, h \rangle$ una estructura de argumento; un *árbol dialéctico* para $\langle \mathcal{A}, h \rangle$, denotado $\mathcal{T}_{\langle \mathcal{A},h \rangle}$, se define recursivamente como sigue:

1. un único nodo con un argumento $\langle \mathcal{A}, h \rangle$ sin derrotadores propios o de bloqueo es un árbol dialéctico para $\langle \mathcal{A}, h \rangle$;

2. sea $\langle \mathcal{A}, h \rangle$ un argumento con derrotadores $\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle$ (propios o de bloqueo). Entonces $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ se construye colocando $\langle \mathcal{A}, h \rangle$ como raíz del árbol y los árboles de dialéctica para $\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle$ como hijos de la raíz.

■

Definición 3.20 (*Línea de argumentación*)
Sea $\langle \mathcal{A}_0, h_0 \rangle$ un argumento y $\mathcal{T}_{\langle \mathcal{A}_0,h_0 \rangle}$ su árbol de dialéctica. Cada camino λ en $\mathcal{T}_{\langle \mathcal{A}_0,h_0 \rangle}$ desde la raíz hasta una hoja $\langle \mathcal{A}_n, h_n \rangle$, denotado como

$$\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$$

constituye una *línea de argumentación* para $\langle \mathcal{A}_0, h_0 \rangle$. ■

Definición 3.21 (*Argumentos de soporte e interferencia*)
Sea $\mathcal{T}_{\langle \mathcal{A}_0,h_0 \rangle}$ un árbol de dialéctica y $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ una línea de argumentación. Cada $\langle \mathcal{A}_i, h_i \rangle$ presente en λ puede rotularse como *argumento de soporte* o *argumento de interferencia* de acuerdo a la siguiente convención:

1. $\langle \mathcal{A}_0, h_0 \rangle$ es un argumento de soporte,

2. si $\langle \mathcal{A}_i, h_i \rangle$ es un argumento de soporte (interferencia) en λ entonces $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ es un argumento de interferencia (soporte) en λ.

Denotaremos como λ_S (respectivamente λ_I) al conjunto de todos los argumentos de soporte (respectivamente interferencia) en λ. ■

Tal como puede apreciarse en base a la definición anterior, una línea de argumentación es una secuencia ordenada de argumentos de soporte e interferencia intercalados.

Los árboles dialécticos se complementan con un *proceso de etiquetado* que permite determinar si el argumento presente en la raíz del árbol es o no aceptable asignando estados a los argumentos que lo componen.

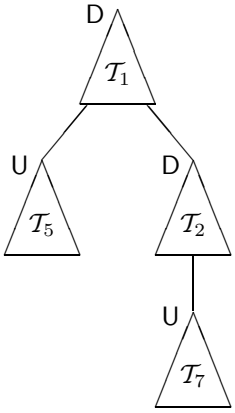


Figura 3.5: Árbol dialéctico correspondiente al ejemplo 3.12

Definición 3.22 (*Etiquetado de un árbol dialéctico*)
Sea $\langle \mathcal{A}, h \rangle$ un argumento y $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ su árbol de dialéctica asociado. Los nodos de $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ pueden etiquetarse recursivamente como nodos no derrotados (`Nodos U`) o nodos derrotados (`Nodos D`) de acuerdo al siguiente procedimiento:

1. las hojas de $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ son etiquetadas como `Nodos U`;

2. sea $\langle \mathcal{B}, q \rangle$ un nodo interno de $\mathcal{T}_{\langle \mathcal{A},h \rangle}$. $\langle \mathcal{B}, q \rangle$ es rotulado como `Nodo U` si, y sólo si, cada uno de sus hijos ha sido etiquetado como `Nodo D`. $\langle \mathcal{B}, q \rangle$ es rotulado como `Nodo D` si y sólo si al menos uno de sus hijos ha sido etiquetado como `Nodo U`.

■

Ejemplo 3.12 El árbol dialéctico para $\langle \mathcal{T}_1, \neg$`construye_nido(will)`$\rangle$ está compuesto por los siguientes argumentos (detallados en el ejemplo 3.8):

- $\langle \mathcal{T}_7, \neg$`nace_huevo(will)`$\rangle$
- $\langle \mathcal{T}_5,$ `nace_huevo(will)`$\rangle$
- $\langle \mathcal{T}_2,$ `construye_nido(will)`$\rangle$

En la figura 3.5 se muestra la estructura del árbol con los rótulos asignados a cada argumento. ■

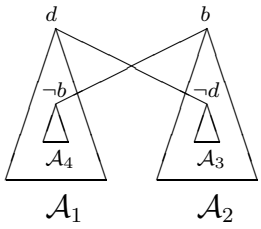


Figura 3.6: Argumentación recíproca

Para que el proceso de etiquetado sea factible es necesario garantizar su finalización, es decir, asegurar que los árboles dialécticos sean estructuras *finitas*. Dado que la base de conocimiento a partir de la cual se construyen los argumentos debe ser finita, la cantidad de argumentos posibles también lo es. Por lo tanto la existencia de caminos infinitos sólo puede ser causada por la presencia de ciclos en el árbol dialéctico. Desafortunadamente esta situación es posible, ya que la relación de derrota no es acíclica.

En primer lugar analizaremos la presencia de *derrotadores recíprocos*, esto es, pares de argumentos que se derrotan entre sí originando ciclos en la argumentación. El ejemplo que ilustra esta situación fue inspirado en el análisis realizado por H. Prakken [Prakken, 1993].

Ejemplo 3.13 Sea $\mathcal{K} = \{a, c\}$, y $\Delta = \{a \succ \neg b, c \succ \neg d, \neg b \wedge c \succ d, a \wedge \neg d \succ d\}$. Las siguientes estructuras de argumentos pueden construirse a partir de $(\mathcal{K}, \Delta)$:

1. $\langle \mathcal{A}_1, d \rangle$, donde $\mathcal{A}_1 = \{a \succ \neg b, \neg b \wedge c \succ d\}$.

2. $\langle \mathcal{A}_2, b \rangle$, donde $\mathcal{A}_2 = \{c \succ \neg d, a \wedge \neg d \succ d\}$.

Observemos que $\langle \mathcal{A}_1, d \rangle$ derrota a $\langle \mathcal{A}_2, b \rangle$, dado que es estrictamente más específico que el subargumento $\langle \mathcal{A}_3, \neg d \rangle$ contenido en $\langle \mathcal{A}_2, b \rangle$. Por otro parte $\langle \mathcal{A}_2, b \rangle$ derrota a $\langle \mathcal{A}_1, d \rangle$, pues es estrictamente más específico que el subargumento $\langle \mathcal{A}_4, \neg b \rangle$ de $\langle \mathcal{A}_1, d \rangle$. Esta situación falaz se representa gráficamente en la figura 3.6. ■

El ejemplo anterior muestra una clase de situación falaz. En general las falacias consisten en argumentos o procesos de argumentación aparentemente válidos en los cuales un análisis detallado revela un alguna falla en su construcción. Para solucionar esta clase de problemas, indeseables desde el punto de vista técnico e intuitivo, se refina la relación de contrargumentación:

Definición 3.23 (*Contraargumento no recíproco*)
Un argumento $\langle \mathcal{A}_1, h_1 \rangle$ *contraargumenta* a un argumento $\langle \mathcal{A}_2, h_2 \rangle$ si y sólo si se verifica que:

1. existe un subargumento $\langle \mathcal{A}, h \rangle$ de $\langle \mathcal{A}_2, h_2 \rangle$ tal que $\langle \mathcal{A}, h \rangle$ está en desacuerdo con $\langle \mathcal{A}_2, h_2 \rangle$, y

2. para todo subargumento propio $\langle \mathcal{S}, j \rangle$ de $\langle \mathcal{A}_1, h_1 \rangle$, no es el caso que $\langle \mathcal{A}_2, h_2 \rangle$ contrargumenta a $\langle \mathcal{S}, j \rangle$ (de acuerdo a la definición 3.15). ■

No obstante, este primer acercamiento no es suficiente para resolver el problema de la argumentación circular, ya que existe la posibilidad de construir ciclos de cualquier longitud dentro de una línea de argumentación. Consideremos el siguiente ejemplo que fuera presentado en [Simari et al., 1994].

Ejemplo 3.14 Sea $(\mathcal{K}, \Delta)$una base de conocimiento tal que $\mathcal{K} = \{e_1, e_2, e_3\}$ y

$$\Delta = \{e_1 \succ p, e_2 \succ q, e_3 \succ r, e_3 \wedge p \succ \neg r, e_1 \wedge q \succ \neg p, e_2 \wedge r \succ \neg q\}$$

a partir de la cual pueden formarse los argumentos:

1. $\langle \mathcal{A}, \neg r \rangle$, donde $\mathcal{A} = \{e_1 \succ p, e_3 \wedge p \succ \neg r\}$,

2. $\langle \mathcal{B}, \neg p \rangle$, donde $\mathcal{B} = \{e_2 \succ q, e_1 \wedge q \succ \neg p\}$,

3. $\langle \mathcal{C}, \neg q \rangle$, donde $\mathcal{C} = \{e_3 \succ r, e_2 \wedge r \succ \neg q\}$.

En este escenario $\langle \mathcal{B}, \neg p \rangle$ derrota a $\langle \mathcal{A}, \neg r \rangle$, $\langle \mathcal{C}, \neg q \rangle$ derrota a $\langle \mathcal{B}, \neg p \rangle$ y $\langle \mathcal{A}, \neg r \rangle$ derrota a $\langle \mathcal{C}, \neg q \rangle$, tal como se muestra en la figura 3.7 ■

Los conceptos de argumentos de soporte y argumentos de inferencia resultan de gran ayuda para analizar por qué este escenario constituye una falacia. Debemos tener en cuenta que el argumento $\langle \mathcal{C}, \neg q \rangle$ fue expuesto originalmente como un

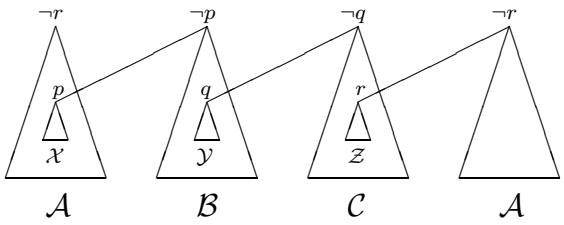


Figura 3.7: Línea de argumentación contradictoria

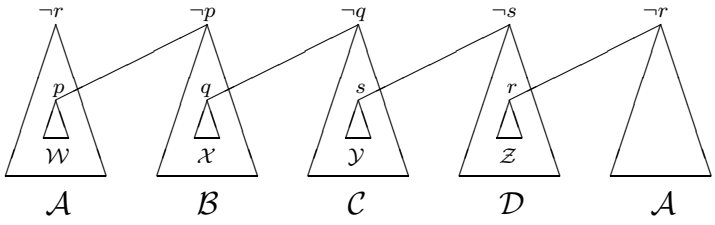


Figura 3.8: Línea de argumentación circular

argumento de soporte para $\langle \mathcal{A}, \neg r \rangle$ y no como un argumento de interferencia. Al reintroducir $\langle \mathcal{A}, \neg r \rangle$ en la línea de argumentación, este se convierte en un argumento de interferencia para si mismo. Este ejemplo evidencia que debería existir alguna noción de *acuerdo* o *coherencia* entre los elementos de soporte (respectivamente de interferencia) presentes en una línea de argumentación.

Es también posible la existencia de un ciclo formado por un número par de argumentos, tal como se analiza a continuación.

Ejemplo 3.15 Supongamos que $\mathcal{K} = \{e_1, e_2, e_3, e_4\}$ y $\Delta = \{e_1 \succ p, e_2 \succ q, e_3 \succ s, e_4 \succ r, e_4 \wedge p \succ \neg r, e_1 \wedge q \succ \neg p, e_2 \wedge s \succ \neg q, e_3 \wedge r \succ \neg s\}$ forman una base de conocimiento en la cual es posible obtener los argumentos:

1. $\langle \mathcal{A}, \neg r \rangle$, donde $\mathcal{A} = \{e_1 \succ p, e_4 \wedge p \succ \neg r\}$,
2. $\langle \mathcal{B}, \neg p \rangle$, donde $\mathcal{B} = \{e_2 \succ q, e_1 \wedge q \succ \neg p\}$,
3. $\langle \mathcal{C}, \neg q \rangle$, donde $\mathcal{C} = \{e_3 \succ s, e_2 \wedge s \succ \neg q\}$,
4. $\langle \mathcal{D}, \neg s \rangle$, donde $\mathcal{D} = \{e_4 \succ r, e_3 \wedge r \succ \neg s\}$.

En consecuencia, $\langle \mathcal{B}, \neg p \rangle$ derrota a $\langle \mathcal{A}, \neg r \rangle$, $\langle \mathcal{C}, \neg q \rangle$ derrota a $\langle \mathcal{B}, \neg p \rangle$, $\langle \mathcal{D}, \neg s \rangle$ derrota a $\langle \mathcal{C}, \neg q \rangle$ y $\langle \mathcal{A}, \neg r \rangle$ derrota a $\langle \mathcal{D}, \neg s \rangle$, tal como se muestra en la figura 3.8. ■

En este caso no son los argumentos de soporte (o interferencia) los que están en conflicto, pero el proceso de inferencia cae nuevamente en una situación circular. En el ejemplo 3.15 el cuarto argumento contiene un subargumento $\langle \mathcal{E}, r \rangle$ que contrargumenta a $\langle \mathcal{A}, \neg r \rangle$. Luego $\langle \mathcal{A}, \neg r \rangle$ está siendo interferido indirectamente por el argumento $\langle \mathcal{D}, \neg s \rangle$, pero $\langle \mathcal{D}, \neg s \rangle$ pudo ser construido suponiendo la falsedad de la conclusión de $\mathcal{A}$. Por lo tanto este tipo de razonamiento no es válido y debería rechazarse.

Las falacias se solucionan en este sistema mediante el concepto de *árbol dialéctico aceptable*, tendiente a distinguir una subcategoría de árboles dialécticos que no contienen *argumentación falaz*. Previamente es necesario introducir las siguientes definiciones:

Definición 3.24 (*Argumentos concordantes*)
Dos estructuras de argumento $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ son *concordantes* si, y sólo si, $\mathcal{K} \cup \mathcal{A}_1 \cup \mathcal{A}_2 \nvdash \bot$. En general una familia de argumentos $\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle$ es concordante si y sólo si $\mathcal{K} \cup \mathcal{F} \nvdash \bot$, donde $\mathcal{F} = \bigcup_n^{i=1} \mathcal{A}_i$. ■

La concordancia entre los argumentos de soporte (respectivamente interferencia) de una línea de argumentación y su no circularidad son las condiciones requeridas para su validez.

Definición 3.25 (*Línea de argumentación aceptable*)
Una línea de argumentación $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ se denomina *aceptable* si y sólo si:

1. los argumentos de soporte (interferencia) en λ son conjuntos de argumentos concordantes;

2. para todo argumento $\langle \mathcal{A}_i, h_i \rangle$ en λ_S (respectivamente λ_I), se cumple que no existe un argumento $\langle \mathcal{A}_j, h_j \rangle$ en λ_I (respectivamente λ_S) tal que $i < j$ y $\langle \mathcal{A}_i, h_i \rangle$ derrota a $\langle \mathcal{A}_j, h_j \rangle$.

■

Es importante destacar que una caracterización más elegante se obtiene agregando el control de reciprocidad en la noción de línea de argumentación aceptable, permitiendo resumir el control de falacias en un sólo concepto [Chesñevar, 1996].

Finalmente estamos en condiciones de enunciar la noción de *árbol dialéctico aceptable*, eje del proceso de justificación.

Definición 3.26 (*Árbol dialéctico aceptable*)
Sea $\langle \mathcal{A}, h \rangle$ una estructura de argumento. Un *árbol dialéctico aceptable* para $\langle \mathcal{A}, h \rangle$, denotado $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$, se define recursivamente como sigue:

1. un único nodo con un argumento $\langle \mathcal{A}, h \rangle$, sin derrotadores propios o de bloqueo es un árbol dialéctico para $\langle \mathcal{A}, h \rangle$;

2. Sea $\langle \mathcal{A}, h \rangle$ un argumento con derrotadores $\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle$, propios o de bloqueo, entonces $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ se construye colocando $\langle \mathcal{A}, h \rangle$ como raíz del árbol y los árboles de dialéctica para $\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle$ como hijos de la raíz. Si durante la construcción se forma una línea de argumentación que no es aceptable basta con extraer el subárbol que tiene como raíz el argumento que causa la no aceptabilidad.

■

Esta formalización es semánticamente correcta, pero operacional y poco elegante. Una caracterización equivalente puede obtenerse al definir un árbol dialéctico aceptable como un árbol dialéctico en el cual todas sus líneas de argumentación son aceptables.

Definición 3.27 (*Justificación*)
Se dice que un argumento $\langle \mathcal{A}, h \rangle$ es una *justificación para* h si y sólo si existe un árbol de dialéctica aceptable para $\langle \mathcal{A}, q \rangle$ cuya raíz está etiquetada como `Nodo U`. ■

De acuerdo a la noción de justificación, el sistema puede asociar cualquiera de los siguientes estados a un hecho h perteneciente a $Sent(\mathcal{L})$.

- **Justificado:** si existe una justificación para algún argumento $\langle \mathcal{A}, h \rangle$.
- **Rechazado:** si para cada posible argumento $\langle \mathcal{A}, h \rangle$ existe una justificación para al menos un derrotador propio de $\langle \mathcal{A}, h \rangle$.
- **Desconocido:** no es posible construir una estructura de argumento A que sustente a h.
- **Indeciso:** si para cada argumento $\langle \mathcal{A}, h \rangle$ no es posible hallar derrotadores propios, pero existe al menos un derrotador de bloqueo.

La clasificación propuesta otorga mayor expresividad al lenguaje, ya que permite distinguir diversos estados en las conclusiones analizadas. Los elementos del primer conjunto de respuestas constituyen las creencias del agente. Si un hecho está sustentado exclusivamente por argumentos que son a su vez derrotados por argumentos justificados se considera rechazado. En algunas situaciones el sistema puede declararse indeciso ante un determinado hecho h. En este caso el sistema distingue dos categorías: si no existe información en absoluto para h se dice que h es desconocido y si no es posible decidir el estado de h en función del conocimiento disponible entonces el agente está indeciso sobre h.

Ejemplo 3.16 Continuando con el ejemplo 3.7, la conclusión `vuela(mina)` está justificada, ¬`construye_nido(will)` está rechazada y la clasificación de `vuela(tom)` es desconocida. ■

Cabe acotar que la construcción de un árbol dialéctico puede recrearse como un debate entre un proponente y un oponente que intercambian argumentos a favor y en contra de la conclusión del argumento situado en la raíz del árbol. La caracterización dialéctica posibilitó el desarrollo de soluciones a diversas situaciones falaces en forma simple y elegante, que hubieran sido difíciles de formular en base a la definición inductiva.

El formalismo de [Simari et al., 1994] ha sido utilizado como base para el desarrollo de diversos trabajos en el campo de la argumentación rebatible. Carlos I. Chesñevar desarrolló en su Tesis de Magister [Chesñevar, 1996] un método de optimización para el proceso de inferencia por medio de podas en los árboles dialécticos, que permiten direccionar la búsqueda de derrotadores. Este trabajo analiza también una generalización de la noción de argumento mediante el uso de reglas no instanciadas denominado *forma argumental*, que permite realizar el proceso de inferencia en forma independientemente de la información contingente en la base de conocimiento. Alejandro J. García elaboró en diversos trabajos (*e.g.* [García, 1997, García, 2000]) un lenguaje de representación de conocimiento denominado *Programación en Lógica Rebatible* (PLR), que combina las propiedades de la programación en lógica con el mecanismo de inferencia del sistema de Simari y Loui. Los principales conceptos de la PLR son analizados en detalle en la sección 3.6.

3.3. El formalismo de Prakken y Sartor

La relación entre la argumentación y el razonamiento legal tiene sus comienzos en la disertación doctoral de Hendrik Prakken [Prakken, 1993], donde se realiza un análisis lógico-formal para clarificar algunos aspectos del razonamiento legal. En particular el autor se centra en el estudio del razonamiento rebatible, tópico que resulta de importancia para diversas disciplinas como la Inteligencia Artificial y la Filosofía. La contribución primordial de la tesis de Prakken consiste en una teoría argumentativa basada en la lógica default que pretende modelar el razonamiento legal. Desafortunadamente el sistema desarrollado posee numerosos incovenientes. Por caso, las reglas rebatibles que componen el lenguaje representan defaults normales y esto afecta notablemente el poder expresivo del lenguaje.

El acercamiento original fue superado en trabajos posteriores realizados en conjunción con Giovanni Sartor en los cuales se define un sistema argumentativo que combina el lenguaje de la programación en lógica extendida con la semántica básica definida por Dung en [Dung, 1995b]. Una característica novedosa de esta propuesta consiste en que las prioridades entre argumentos no son fijas, sino que son derivadas como conclusiones del sistema. A continuación se describe este formalismo de acuerdo a lo expuesto en [Prakken y Sartor, 1997].

El lenguaje del sistema sigue el estilo de la programación en lógica y sus reglas están formadas por literales *fuertes* y literales *débiles*. Un literal fuerte consiste en un átomo que puede estar precedido por el símbolo "¬" de la *negación fuerte* y un literal débil o *suposición* toma la forma $\sim L$, donde L es un literal fuerte y "$\sim$" es el símbolo de la *negación débil*. El símbolo "¬" representa la negación clásica mientras que "$\sim$" codifica una clase de negación por defecto.

El formalismo de Prakken y Sartor comparte varias características con el sistema de Simari y Loui, por ejemplo, la existencia de dos tipos de reglas: *estrictas* y *rebatibles*. Una regla rebatible es una expresión de la forma:

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

donde r es el nombre de la regla y cada L_i $(0 \leq i \leq n)$ es un literal fuerte. La conjunción a la izquierda del conectivo "$\Rightarrow$" se denomina *antecedente* y el literal L_n *consecuente*. El antecedente de una regla r puede ser vacío; en este caso r representa una afirmación incondicional. El sistema sólo considera programas básicos y por lo tanto una regla con variables se interpreta como un *esquema* que resume a todas sus instancias básicas.

Sintácticamente, las reglas estrictas difieren de las rebatibles en que no pueden contener suposiciones y están denotadas con el conectivo "$\rightarrow$" en lugar de "$\Rightarrow$". Pragmáticamente, las reglas estrictas representan conocimiento irrefutable mientras que las reglas rebatibles codifican información tentativa. Es importante observar que en estas últimas existen dos fuentes de rebatibilidad. Una regla rebatible r_1 puede ser atacada por una regla cuyo consecuente es complementario con el de r_1 o por una regla que invalide las suposiciones en las que se basa r_1.

Los autores mencionan que en el razonamiento legal los conflictos entre reglas incompatibles se solucionan indicando cuál de estas reglas tiene precedencia. Por consiguiente el sistema incluye un ordenamiento entre las reglas rebatibles que luego se extiende para decidir entre argumentos en conflicto. La base de conocimiento se

define como una tupla $\Gamma = (S, D, <)$ denominada *teoría ordenada*, donde S denota un conjunto de reglas estrictas, D es un conjunto de reglas rebatibles y $<$ es un orden parcial estricto sobre los elementos de D.

Un *argumento* se define como una secuencia de reglas encadenadas para obtener una determinada conclusión.

Definición 3.28 (*Argumento*)
Un *argumento* A basado en una teoría ordenada $\Gamma = (S, D, <)$ es una secuencia finita $A = [r_0, \ldots, r_n]$ de reglas rebatibles tal que:

1. para cada r_i $(0 \leq i \leq n)$ se cumple que $r_i \in S \cup D$,
2. para cada r_i $(0 \leq i \leq n)$ y para cada literal fuerte L_j presente en los antecedentes de r_i existe $k < i$ tal que L_j es el consecuente de r_k,
3. no existen en la secuencia A dos reglas rebatibles con el mismo consecuente.

■

La primer condición requiere que el argumento esté efectivamente basado en la teoría ordenada Γ. La segunda establece como formar los argumentos a partir de un encadenamiento de reglas. En este encadenamiento los literales débiles son ignorados, resultando en una regla de inferencia denominada *modus ponens rebatible* que puede formalizarse como sigue:

$$\frac{\begin{array}{ll} r: & L_0 \wedge \ldots \wedge L_j \wedge {\sim}L_k \wedge \ldots \wedge {\sim}L_m \Rightarrow L_n \\ & L_0 \wedge \ldots \wedge L_j \end{array}}{L_n}$$

La tercer condición previene la formación de argumentos que contienen reglas redundantes. Sin embargo, no se requiere que el conjunto de reglas del argumento sea minimal con respecto a la inclusión de conjuntos [Simari y Loui, 1992]. Por lo tanto nada impide el agregado de reglas irrelevantes que no toman parte en la derivación de la conclusión.

A continuación Prakken y Sartor definen la siguiente terminología:

- Para cualquier teoría ordenada Γ se denota como $Args_\Gamma$ al conjunto de todos los argumentos basados en Γ.
- Para cualquier conjunto de reglas R se denota como $Args_R$ al conjunto de todos los argumentos formados con reglas de R.

- Un argumento A' es un *subargumento* de A si y sólo si A' es una subsecuencia de A. En el caso que $A' \neq A$, A' es un *subargumento propio*.
- Un literal L es una *conclusión* de un argumento A si L es el consecuente de alguna regla en A.
- Un literal L es una *suposición* de un argumento A si $\sim\overline{L}$ pertenece a alguna regla en A, donde $\overline{L}$ denota el complemento de L con respecto a la negación fuerte.

Ejemplo 3.17 Consideremos el argumento[4]

$$A = [r_1 :\longrightarrow a,\ r_2 : a \wedge {\sim}\neg b \Rightarrow c,\ r_3 : c \wedge {\sim}d \Rightarrow e]$$

Las conclusiones de A son $\{a, c, e\}$, sus subargumentos son $\{[], [r_1], [r_1, r_2], [r_1, r_2, r_3]\}$ y sus suposiciones son $\{b, \neg d\}$. ■

La relación de *contrargumentación* o *ataque* determina cuándo un par de argumentos está en conflicto. Las dos clases de negación dan lugar a distintas formas de ataque. La primera consiste en contradecir a una conclusión del argumento utilizando la negación fuerte, es decir, si A es un argumento y L una de sus conclusiones entonces A ataca a cualquier argumento B con conclusión $\neg L$. La segunda forma de ataque utiliza la negación débil y se basa en contradecir una suposición: un argumento A para L ataca a cualquier argumento B con una suposición $\overline{L}$. Prakken y Sartor sostienen que esta última puede verse como un caso especial de la noción de socavamiento enunciada por Pollock [Pollock, 1987] (detallada en la sección 3.1). Los autores sustentan esta afirmación interpretando al ataque a las suposiciones como una objeción al modus ponens rebatible usado en el encadenamiento de las reglas. No obstante, entendemos que esta noción es diferente al ataque por socavamiento de Pollock, dado que este último ataca a reglas de inferencias de un dominio en particular y no a patrones de razonamiento en general.

Aunque la descripción informal del concepto de contraargumentación resulta sencilla, la inclusión de las reglas fuertes en los argumentos hace difícil otorgar un tratamiento diferencial a cada regla de acuerdo a su categoría.

Ejemplo 3.18 Observemos el siguiente conjunto de reglas:

[4]La mayoría de los ejemplos presentados en el trabajo de Prakken y Sartor re restringen a un lenguaje proposicional para simplificar el análisis. Cabe acotar que, como se mencionó en la sección 3.2, las reglas rebatibles proposicionales no tienen sentido desde un punto de vista semántico.

$$r_1 : A \Rightarrow x \texttt{ está casado}$$
$$r_2 : B \Rightarrow x \texttt{ es soltero}$$
$$s_1 : x \texttt{ está casado} \rightarrow \neg\, x \texttt{ es soltero}$$
$$s_2 : x \texttt{ es soltero} \rightarrow \neg\, x \texttt{ está casado}$$
$$f_1 : \rightarrow A$$
$$f_2 : \rightarrow B$$

Intuitivamente las reglas r_1 y r_2 están en conflicto aunque no poseen consecuentes complementarios. Las reglas s_1 y s_2 simplemente declaran que los predicados `soltero` y `casado` son incompatibles. ■

Si solamente se consideran en conflicto las reglas con consecuentes complementarios, los argumentos $[f_1, r_1]$ y $[f_2, r_2]$ del ejemplo anterior no son contradictorios: esto difiere del resultado esperado. Por lo tanto los autores proponen una caracterización distinta. Si un argumento A posee una conclusión o suposición L entonces A es atacado por cualquier argumento B tal que B pueden ser extendido, usando sólo reglas estrictas, a un argumento con conclusión $\overline{L}$. Esto se formaliza en base a los siguientes conceptos.

Definición 3.29 (*Concatenación*)
Sea A un argumento y T una secuencia de reglas, $A + T$ denota la concatenación de A con las reglas en T. ■

Definición 3.30 (*Ataque*)
Sean A_1 y A_2 dos argumentos, A_1 *ataca* a A_2 si y sólo si existen dos secuencias S_1, S_2 de reglas estrictas tal que $A_1 + S_1$ es un argumento con conclusión L y se cumple que:

1. $A_2 + S_2$ es un argumento con conclusión $\overline{L}$ o
2. A_2 es un argumento que contiene una suposición $\overline{L}$.

■

Habiendo definido qué argumentos están en conflicto, el próximo paso consiste en detallar un método para compararlos a fin de formular la noción de *derrota*. En congruencia con la definición de ataque, la relación de derrota se enuncia en términos de *rebatimiento* y *socavamiento*. Cuando un argumento A contradice una conclusión de otro argumento B y A es preferido a B estamos en presencia de una derrota

por rebatimiento. En este caso las prioridades entre las reglas de D deben utilizarse para comparar A con respecto a B. Aquí se evidencia el costo acarreado por utilizar un orden entre reglas, ya que debe definirse algún mecanismo para determinar la precedencia de un argumentos en base a la prioridad de sus reglas. En primer lugar es necesario elegir qué reglas se utilizarán en la comparación.

Definición 3.31 (*Reglas relevantes*)
Sea A un argumento y S un conjunto de reglas estrictas tal que $A + S$ es un argumento con conclusión L. El conjunto de *reglas relevantes a* L en el argumento $A + S$, denotado como $R_L(A + S)$, se define de la siguiente manera.

1. Si A incluye a una regla rebatible r_d con consecuente L entonces $R_L(A+S) = \{r_d\}$.

2. Si A es rebatible y S incluye una regla estricta $r_s : L_1 \wedge \ldots \wedge L_n \rightarrow L$ entonces $R_L(A + S) = R_{L_1}(A + S) \cup \ldots \cup R_{L_n}(A + S)$.

■

Formalmente, el ordenamiento entre reglas se extiende a conjuntos como sigue:

Definición 3.32 (*Criterio de preferencia*)
Sean R y R' dos conjuntos de reglas rebatibles, $R < R'$ si, y sólo si, existe $r \in R$ tal que para todo $r' \in R'$ se cumple que $r < r'$. ■

De acuerdo a la definición 3.31 la regla rebatible que permite obtener la conclusión del argumento es el único elemento relevante en la comparación. En casos complejos, donde la conclusión se deduce mediante reglas estrictas, se consideran todas las reglas rebatibles que permiten deducir los antecedentes de las reglas estrictas. Este principio no tiene en cuenta la totalidad del argumento y en consecuencia no funciona correctamente en gran variedad de situaciones.

Ejemplo 3.19 A partir de las siguientes reglas:

r_1 : se derrumbó $x \Rightarrow$ obstaculiza la calle x
r_2 : obstaculiza la calle $x \Rightarrow$ derrumbar x
r_3 : la casa x es histórica $\Rightarrow \neg$ derrumbar x
f_1 : $\rightarrow$ se derrumbó la casa x
f_2 : $\rightarrow$ la casa x es histórica

es posible construir los argumentos $A = [f_1, r_1, r_2]$ y $B = [f_2, r_3]$. Si r_3 tiene prioridad sobre r_2 entonces A derrota a B, pero intuitivamente es claro que B es preferible a A, pues si la casa está destruida y sus escombros obstaculizan el paso sería mejor derrumbarla. Por tanto el criterio de comparación propuesto en la definición 3.32 resulta desafortunado en esta situación. ■

Un requerimiento mínimo de las teorías argumentativas es que dos argumentos en conflicto no pueden estar justificados conjuntamente. Por otra parte los argumentos auto derrotados [Prakken y Vreeswijk, 1999, Pollock, 1995] son elementos paradójicos y no es correcto que pertenezcan al conjunto de justificaciones. Para solucionar este tipo de problemas son necesarios los siguientes conceptos:

Definición 3.33 (*Conjunto libre de conflictos*)
Un conjunto de argumentos *Args* se denomina *libre de conflictos* si y sólo si no existen dos argumentos A, B tal que $A, B \in Args$ y A ataca a B. ■

Definición 3.34 (*Argumento coherente*)
Un argumento es *coherente* si y sólo si no se ataca a sí mismo. ■

En este punto estamos en condiciones de formalizar la noción de derrota.

Definición 3.35 (*Rebatimiento - Socavamiento*)
Sean A_1 y A_2 dos argumentos, entonces:

1. A_1 *rebate* a A_2 si existen dos secuencias de reglas estrictas S_1, S_2 tal que $A_1 + S_1$ es un argumento con conclusión L y $A_2 + S_2$ es un argumento con conclusión $\overline{L}$ y se cumple que $R_L(A_1 + S_1) \not\preccurlyeq R_{\overline{L}}(A_2 + S_2)$,

2. A_1 *socava* a A_2 si existe una secuencias de reglas estrictas S_1 tal que $A_1 + S_1$ es un argumento con conclusión L y A_2 es un argumento con una suposición $\overline{L}$.

■

Definición 3.36 (*Derrota*)
Sean A_1 y A_2 dos argumentos, A_1 *derrota* a A_2 si y sólo si se cumple alguna de las siguientes condiciones:

1. A_1 es un argumento vacío y A_2 no es coherente,

2. A_1 socava a A_2, o

3. A_1 rebate a A_2.

Si A_1 derrota a A_2 y A_2 no derrota a A_1, entonces A_1 *derrota estrictamente* a A_2. ■

En la definición anterior los argumentos incoherentes son eliminados mediante un caso especial que los declara derrotados por cualquier argumento estricto. Observemos también que la noción de socavamiento no utiliza las prioridades entre reglas. Prakken y Sartor justifican esta elección en su experiencia en el dominio legal, donde las reglas de resolución de conflictos solamente son utilizadas en los ataques a las conclusiones. Es importante destacar que esta decisión de diseño no permite el reinstanciamiento de los argumentos derrotados por socavamiento.

A continuación Prakken y Sartor definen el conjunto de argumentos justificados por medio de dos caracterizaciones diferentes que denominan *semántica* y *teoría de prueba*. Los autores entienden que una semántica consiste en una especificación del conjunto de conclusiones que cumple con determinadas propiedades y una teoría de prueba es un procedimiento que permite decidir si un argumento dado es miembro de este conjunto.

La semántica del sistema se formaliza utilizando una ecuación de punto fijo fuertemente basada en la *semántica básica*[5] propuesta por Dung. Las definiciones de *aceptabilidad* y *función característica* presentadas en [Dung, 1995b] son adaptadas por Prakken y Sartor de la siguiente manera:

Definición 3.37 (*Argumento aceptable*)
Un argumento A es *aceptable* con respecto a un conjunto de argumentos *Args* si y sólo si cada argumento que derrota a A es derrotado estrictamente por un argumento perteneciente a *Args*. ■

Definición 3.38 (*Función caraterística*)
Sea Γ una teoría ordenada y T un subconjunto de $Args_\Gamma$. La *función característica* de Γ se define a continuación:

$$F_\Gamma : \wp(Args_\Gamma) \mapsto \wp(Args_\Gamma),\ F_\Gamma(T) = \{A \in Args_\Gamma \mid A \text{ es aceptable con respecto a T}\}$$

donde $\wp(Args_\Gamma)$ denota el conjunto de partes de $Args_\Gamma$. ■

[5]En la sección 3.5 se detalla este concepto.

Los autores demuestran que el operador F_Γ es monótono. La propiedad de monotonía asegura la existencia del menor punto fijo de F_Γ, que será utilizado para describir el conjunto de argumentos justificados.

Definición 3.39 (*Estado de los argumentos*)
Sea Γ una teoría ordenada y A un argumento basado en Γ, entonces:

1. A es un argumento *justificado* si y sólo si A pertenece al menor punto fijo de F_Γ, denotado como $JustArgs_\Gamma$.

2. A es un argumento *denegado* si y sólo si A es atacado por un argumento justificado.

3. A es un argumento *defendible* si y sólo si A no está justificado ni denegado.

■

Un literal L es una *conclusión justificada* en base a Γ si y sólo si L es la conclusión de un argumento justificado. L es una *conclusión defendible* si y sólo si no está justificado y es la conclusión de un argumento defendible. Finalmente L es una *conclusión denegada* si y sólo si no está justificado ni es defendible y es la conclusión de un argumento denegado.

La proposición que se enuncia a continuación establece como hallar el conjunto de argumentos justificados en forma constructiva, comenzando desde el conjunto vacío. Si la relación de ataque es finita es posible hallar el punto fijo. En caso contrario sólo es posible aproximarlo.

Proposición 3.1 Sea Γ una teoría ordenada y F^i una secuencia de subconjuntos de $Args_\Gamma$ tal que $F^0 = F_\Gamma(\emptyset)$ y $F^{i+1} = F^i \cup F_\Gamma(F^i)$, entonces $\bigcup_{i=0}^{\infty}(F^i) \subseteq JustArgs_\Gamma$. En particular, si cada argumento de $Args_\Gamma$ es atacado por un número finito de argumentos, se cumple que $\bigcup_{i=0}^{\infty}(F^i) = JustArgs_\Gamma$. ■

Ejemplo 3.20 A partir del siguiente conjunto de reglas ilustraremos como se lleva a cabo la clasificación de argumentos de acuerdo a la proposición 3.1.

$$r_0 : \Rightarrow a$$
$$r_1 : {\sim}b \Rightarrow c$$
$$r_2 : \Rightarrow \neg a$$
$$r_3 : a \Rightarrow \neg b$$

Si $r_0 < r_3$ el argumento $A_1 = [r_0, r_1]$ derrota por socavamiento a $A_2 = [r_2]$ y $A_3 = [r_3]$ derrota a A_1, rebatiendo a su subargumento propio $A_0 = [r_0]$. El conjunto de argumentos justificados se construye comenzando con $F^1 = \{[], A_3\}$, dado que A_3 no está derrotado por ningún argumento (su único contraargumento A_0 es más débil) y es por tanto aceptable con respecto a $\emptyset$. Analicemos que argumentos pueden agregarse en F^2. A_1 y A_0 no son aceptables con respecto a F_1, pero A_2 si lo es. Aunque está derrotado por A_1, A_2 es reinstanciado por A_3 (que derrota a A_1) y en consecuencia $F^2 = \{[], A_3, A_2\}$. Repetir este proceso sobre F^2 no agrega argumentos a F^3, luego $F^2 = F^3$ y hemos hallado el punto fijo del operador F_Γ. ■

El operador de la definición 3.38 da origen a una semántica escéptica. Los autores reconocen que esto puede resultar demasiado restrictivo en algunos casos.

Ejemplo 3.21 Sea Γ una teoría ordenada con el conjunto de reglas:

$$
\begin{aligned}
&s_1 : \rightarrow a \\
&r_1 : a \Rightarrow b \\
&s_2 : \rightarrow c \\
&r_2 : a \wedge c \Rightarrow \neg b \\
&s_3 : \rightarrow d \\
&r_3 : d \Rightarrow b \\
&r_4 : c \wedge d \Rightarrow \neg b
\end{aligned}
$$

tal que $r_1 < r_2$ y $r_3 < r_4$. En este escenario todos los argumentos para b y $\neg b$ son catalogados como defendibles. $A_2 = [s_1, s_2, r_2]$ derrota en forma estricta a $A_1 = [s_1, r_1]$ y $A_3 = [s_2, s_3, r_4]$ derrota en forma estricta a $A_4 = [s_3, r_3]$. Por otro lado A_2 es derrotado por A_4 y A_3 es derrotado por A_1. Por consiguiente $F^0 = F^1$ y ninguno de los argumentos está justificado. ■

Este ejemplo, diseñado por Prakken y Sartor, codifica el clásico ciclo de derrota de longitud par formado en este caso por cuatro argumentos. Los autores señalan la existencia de conjuntos alternativos que se podrían sancionar como conclusiones del sistema utilizando una semántica más permisiva, tal como es el caso de $\{A_2, A_3\}$ (este conjunto es preferido por derrotar en forma estricta a sus competidores). Es interesante analizar este ejemplo con respecto a las extensiones *completas* de Dung, que constituyen una versión más osada de las extensiones básicas. La semántica completa permite hallar distintos conjuntos de respuesta alternativos que son libres de

conflicto pero no necesariamente minimales, como por ejemplo $\{A_1, A_4\}$ y $\{A_2, A_3\}$. Esto concuerda con la respuesta esperada en esta situación.

Seguidamente se enuncian algunas propiedades formales del operador de justificación. Por definición el conjunto de respuestas es único y aceptable con respecto a sí mismo. También vale que *JustArgs* es mínimal y no hay dependencias circulares entre sus elementos. Además se prueban las siguientes proposiciones:

- El conjunto de argumentos justificados en libre de conflictos.
- Si un argumentos está justificado entonces todos sus subargumentos están justificados.
- Sea Γ una teoría ordenada y S su conjunto de reglas estrictas. Si los argumentos que pueden construirse en base S no constituyen un conjunto libre de conflictos entonces todos los argumentos basados en Γ son incoherentes.

A partir de la última proposición es posible observar que las teorías ordenadas con un conjunto S no libre de conflictos deben evitarse, ya que a partir de ellas no es posible obtener conclusiones sensatas.

La teoría de prueba del sistema es un procedimiento que permite decidir si un determinado argumento A pertenece al conjunto de argumentos justificados. El procedimiento definido por Prakken y Sartor sigue un estilo dialéctico.[6] Cada jugada del diálogo es un argumento basado en la teoría ordenada en consideración, que ataca la movida de su oponente. Basándose en la propuesta de Rescher [Rescher, 1977], los autores identifican a un proponente que desea justificar A y un oponente que quiere refutar las razones de su adversario. Formalmente el proceso se define como sigue:

Definición 3.40 (*Diálogo*)
Un *diálogo* es una secuencia finita y no vacía de jugadas $move_i = (Player_i, Arg_i)$, $i > 0$, tal que:

1. $Player_i = P$ si y sólo si i es impar;
2. $Player_i = O$ si y sólo si i es par;
3. Si $Player_i = Player_j = P$ y $i \neq j$ entonces $Arg_i \neq Arg_j$;
4. Si $Player_i = P$ entonces el conjunto de reglas de Arg_i es minimal (con respecto a la inclusión de conjuntos) y Arg_i derrota estrictamente a Arg_{i-1};

[6] En la sección 2.3 se realiza un análisis de este concepto.

5. Si $Player_i = O$ entonces Arg_i derrota a Arg_{i-1}.

■

Los dos primeros items establecen el intercambio de jugadas entre el oponente y el proponente. La segunda condición prohíbe la repetición de jugadas, pero sólo para el proponente. Los autores manifiestan que es necesario permitir que el oponente presente más de una vez el mismo argumento para que las conclusiones sancionadas por la semántica coincidan con la teoría de prueba. Desafortunadamente, esta afirmación está respaldada solamente por un ejemplo que no resulta demasiado convincente por incurrir en una situación falaz. La tercer y cuarta cláusula estipulan la diferencia entre los roles de los dos participantes. La condición de minimalidad evita la repetición de argumentos esencialmente idénticos, que sólo difieren en una o más reglas irrelevantes. Este problema surge de utilizar una formalización inadecuada de la noción de argumento. Una caracterización más elegante se obtiene solicitando la minimalidad del conjunto de reglas en la definición de argumento, tal como se realiza en el sistema de Simari y Loui [Simari y Loui, 1992].

La siguiente estructura resume todos los posibles diálogos sobre un argumento en particular.

Definición 3.41 (*Árbol de diálogos*)
Un *árbol de diálogos* es un árbol finito de jugadas tal que :

1. cada camino de la raíz a una de las hojas es un diálogo;

2. si $Player_i = P$ entonces los hijos de $move_i$ son todos los derrotadores de Arg_i.

Uno de los participantes gana un diálogo si su adversario no puede responder a su jugada y gana un árbol de diálogos T si gana en todos los diálogos de T. ■

Intuitivamente, si el último argumento de P no posee derrotadores reinstancia a todos los argumentos previos de P que aparecen en el mismo camino y en particular al argumento en la raíz del árbol.

Definición 3.42 (*Justificado por una prueba*)
Un argumento A está *justificado por un prueba* si y sólo si existe un árbol de diálogos con raíz A ganado por el proponente. Un literal fuerte L es una *conclusión justificada por una prueba* si y sólo si es una conclusión de un argumento justificado por una prueba. ■

Finalmente los autores prueban las siguientes proposiciones, que vinculan la semántica con la teoría de prueba del sistema:

Proposición 3.2 Todos los argumento justificados por una prueba son argumentos justificados. ■

Proposición 3.3 Para cualquier teoría ordenada con una relación de derrota finita se cumple que todo argumento justificado está justificado por una prueba. ■

Esto concluye la presentación del sistema con prioridades fijas. Posteriormente se introduce una extensión que incorpora prioridades rebatibles. Los autores manifiestan que en gran cantidad de dominios no existe un orden único e indiscutido entre las reglas. Por ejemplo, en el razonamiento legal los estándares para decidir ante un conflicto están a su vez sujetos a un debate. Basándose en este modelo, los autores rediseñan el formalismo para permitir la construcción de argumentos sobre las prioridades.

En primer lugar se incorpora al lenguaje el símbolo '$\prec$' para expresar información respecto a las prioridades en el lenguaje objeto. En consecuencia ya no es necesario incluir al orden $<$ como el tercer componente de una teoría ordenada y esta se redefine como un par (S, D). Para asegurarse que el orden codificado verifica las propiedades de transitividad y antisimetría deben agregarse las siguientes axiomas:

t_1: $x \prec y \wedge y \prec z \rightarrow x \prec z$

t_2: $x \prec y \wedge \neg x \prec z \rightarrow \neg y \prec z$

t_3: $y \prec z \wedge \neg x \prec z \rightarrow \neg x \prec y$

a: $x \prec y \rightarrow \neg y \prec x$

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 sólo si existe un argumento justificado para $r \prec r'$. En consecuencia el ordenamiento sobre las reglas se obtiene paso a paso junto con los argumentos justificados y varía conforme se obtienen nuevas conclusiones. Por lo tanto es necesario establecer una dependencia explícita de los distintos conceptos del sistema sobre el orden utilizado. La notación que se define a continuación captura la información acerca de las prioridades que se obtiene a partir de un conjunto de argumentos. Para cada conjunto de argumentos *Args* denotaremos mediante $<_{Args}$ al conjunto:

$$<_{Args} = \{r < r' | r \prec r' \text{ es la conclusión de algún } A \in Args\}$$

Ahora es posible enunciar la nueva noción de derrota, que explicita el orden utilizado.

Definición 3.43 (*Derrota*)
Para cualquier par de argumentos A, B y cualquier conjunto de argumentos *Args* se dice que A *Args-derrota* a B en base a una teoría ordenada $\Gamma = (S, D)$ si y sólo si A derrota a B en base a $(S, D, <_{Args})$ aplicando la definición 3.36. ■

La definición de aceptabilidad también se adapta al nuevo formalismo. La aceptabilidad de un argumento depende de los argumentos de *Args* que contienen información sobre las prioridades.

Definición 3.44 (*Aceptabilidad*)
Un argumento A es *aceptable* con respecto a un conjunto de argumentos *Args* si y sólo si todos los argumentos que *Args*-derrotan a A están estrictamente *Args*-derrotados por algún argumento en *Args*. ■

La extensión de la funcion característica al sistema con prioridades rebatibles debe restringirse a conjuntos de argumentos libres de conflictos para que verifique la propiedad de monotonía.

Definición 3.45 (*Función característica*)
Sea $\Gamma = (S, D)$ una teoría ordenada, T un subconjunto de $Args_\Gamma$ y $Cargs_\Gamma$ el conjunto compuesto por todos los conjuntos de argumentos libres de conflicto que están formados por elementos de $Args_\Gamma$. La *función característica* de Γ se define como sigue:

$$G_\Gamma : Cargs_\Gamma \mapsto \wp(Args_\Gamma), G_\Gamma(T) = \{A \in Args \mid A \text{ es aceptable}^7 \text{ con respecto a } T\}$$

■

A partir de este concepto se obtiene la siguiente clasificación del estado de los argumentos.

Definición 3.46 (*Estado de los argumentos*)
Sea $\Gamma = (S, D)$ una teoría ordenada y A un argumento basado en Γ entonces:

1. A es un argumento *justificado* si y sólo si A pertenece al menor punto fijo de F_Γ, denotado como *JustArgs*.

2. A es un argumento *denegado* si y sólo si A es atacado por un argumento justificado.

3. A es un argumento *defendible* si y sólo si A no está justificado ni denegado.

■

La proposición 3.1 también se verifica para el formalismo con prioridades rebatibles, lo que permite calcular el conjunto de argumentos justificados en forma constructiva, comenzando desde el conjunto vacío.

Proposición 3.4 Sea $\Gamma = (S, D)$ una teoría ordenada y G^i una secuencia de subconjuntos de $Args_\Gamma$ tal que $G^0 = G_\Gamma(\emptyset)$ y $G^{i+1} = G^i \cup G_\Gamma(G^i)$, entonces $\bigcup_{i=0}^{\infty}(G^i) \subseteq JustArgs_\Gamma$. En particular, si cada argumento de $Args_\Gamma$ es atacado por un número finito de argumentos, se cumple que $\bigcup_{i=0}^{\infty}(G^i) = JustArgs_\Gamma$. ■

Es importante observar que la relación de derrota es reformulada en cada paso en base al conjunto de argumentos justificados construido hasta el momento, dado que el operador G utiliza la noción de aceptabilidad que a su vez se basa en la definición de $Args$-derrota y el conjunto $Args$ (y en consecuencia $<_{Args}$) cambia en cada iteración.

Ejemplo 3.22 Para ilustrar como funcionan las reglas de ordenamiento consideremos una teoría ordenada (S, D) tal que S contiene los axiomas de prioridad (que aseguran las propiedades de transitividad y antisimetría) y D está compuesto por r_1, r_2, r_3 más las siguientes reglas rebatibles sobre prioridades:

$$r_4 : \Rightarrow r_1 \prec r_2$$
$$r_5 : \Rightarrow r_2 \prec r_3$$
$$r_6 : \Rightarrow r_3 \prec r_1$$
$$r_7 : \Rightarrow r_6 \prec r_4$$
$$r_8 : \Rightarrow r_6 \prec r_5$$

$G^1 = \{[r_7], [r_8], [r_7, r_8]\}$ y entonces $<_{G^1} = \{r_6 < r_4, r_6 < r_5\}$. En la próxima iteración $[r_4, r_5]$ G^1-derrota a todos los argumentos que lo atacan ($[r_6]$, $[r_4, r_6]$ y $[r_5, r_6]$) y en consecuencia debe pertenecer a G^2. ■

Por último los autores redefinen la teoría de prueba para el sistema con prioridades rebatibles e investigan el comportamiento del sistema al reemplazar su semántica escéptica por una versión osada.

En general, la definición del formalismo de Prakken y Sartor resulta interesante y posee el mérito de enunciar y demostrar numerosas propiedades esenciales en los sistemas argumentativos. Aunque el sistema con prioridades fijas no presenta

un acercamiento novedoso, pues es una instancia en particular de los *frameworks argumentativos abstractos*, la principal contribución de Prakken y Sartor yace en el desarrollo del sistema con prioridades rebatibles que se representan mediante fórmulas en el lenguaje objeto. De esta forma es posible debatir sobre el orden de preferencia entre los argumentos.

3.4. El sistemas abstracto de Gerard Vreeswijk

El acercamiento de G. Vreeswijk consiste en el desarrollo de un *sistema argumentativo abstracto* (SAA) con el objetivo de realizar un estudio generalizado sobre las propiedades de la argumentación rebatible, sin preocuparse por definir un sistema en particular. Los SAA están inspirados en el trabajo de Lin y Shoham [Lin y Shoham, 1989] e intentan solucionar las carencias de este formalismo[8] incorporando las nociones de conflicto y derrota entre argumentos. La disertación doctoral de Vreeswijk [Vreeswijk, 1993] marcó el punto de partida de los SAA. Esta versión original fue posteriormente expandida y refinada en el artículo *"Abstract Argumentation Systems"* [Vreeswijk, 1997], que resumiremos a continuación.

Vreeswijk comienza la presentación de su teoría a través del siguiente concepto:

Definición 3.47 (*Sistema argumentativo abstracto*)
Un *sistema argumentativo abstracto* es un 3-upla

$$\mathcal{A} = (\mathcal{L}, R, \leq)$$

tal que $\mathcal{L}$ es un lenguaje, R es un conjunto de reglas de inferencia y $\leq$ es un orden reflexivo y transitivo sobre los argumentos. Cualquier conjunto $\mathcal{L}$ que posea al elemento distinguido $\perp$ constituye un lenguaje válido. ■

Tal como en el sistema de Lin y Shoham, la noción de lenguaje permanece sin especificar, sólo se requiere la existencia del símbolo $\perp$ para representar una contradicción. De esta forma Vreeswijk pretende probar que es factible construir una teoría argumentativa de gran poder expresivo basada en un arbitrario.

Las reglas del conjunto R son expresiones meta-lingüísticas, (es decir, que no pertenecen al lenguaje $\mathcal{L}$) que se definen en términos de las sentencias en $\mathcal{L}$.

Definición 3.48 (*Reglas de inferencia*)
Sea $\mathcal{L}$ un lenguaje. Entonces:

[8] En el análisis de la sección 2.1 se detallan los problemas del sistema de Lin y Shoham.

- Una *regla de inferencia estricta* es una expresión $\phi_1, \ldots, \phi_n \rightarrow \phi$, tal que $\phi \in \mathcal{L}$ y $\phi_1, \ldots, \phi_n$ es una secuencia posiblemente vacía de elementos de $\mathcal{L}$.

- Una *regla de inferencia rebatible* es una expresión $\phi_1, \ldots, \phi_n \Rightarrow \phi$, tal que $\phi \in \mathcal{L}$ y $\phi_1, \ldots, \phi_n$ es una secuencia posiblemente vacía de elementos de $\mathcal{L}$.

Una *regla de inferencia* es una *regla de inferencia estricta* o una *regla de inferencia rebatible.* ■

No se detalla cuál es la semántica asociada a cada tipo de regla y por lo tanto no deja clara la motivación para utilizar esta representación ni el poder expresivo de la misma frente a situaciones concretas. Considerando que el lenguaje $\mathcal{L}$ permanece sin especificar, Vreeswijk expresa que las reglas de inferencias son independientes del dominio *"tal como las reglas de la lógica proposicional son independientes del dominio"*. En este punto no estamos de acuerdo con Vreeswijk, dado que la noción de una regla rebatible independiente del dominio carece de sentido. En este aspecto, el autor no considera la ontología de las reglas rebatibles, cuya función primordial consiste en representar información parcialmente especificada de un escenario en particular.

Los *argumentos* se definen como árboles de reglas estrictas y rebatibles, en concordancia con la caracterización de Lin y Shoham.

Definición 3.49 (*Argumento*)
Sea R un conjunto de reglas de inferencia. Un *argumento* σ posee un conjunto de *premisas* (*prem*), una *conclusión* (*conc*), un conjunto de *sentencias* (*sent*), un conjunto de *suposiciones* (*asum*), *subargumentos* (*sub*), *argumentos top* (*top*), una *longitud* (*length*) y un *tamaño* (*size*) y se define como:

1. Un elemento de $\mathcal{L}$. En este caso $prem(\sigma) = \{\sigma\}$, $conc(\sigma) = \{\sigma\}$, $sent(\sigma) = \{\sigma\}$, $asum(\sigma) = \emptyset$, $sub(\sigma) = \{\sigma\}$, $top(\sigma) = \{\sigma\}$, $length(\sigma) = 1$, $size(\sigma) = 1$.

2. Una expresión $\sigma_1, \ldots, \sigma_n \rightarrow \phi$, donde $\sigma_1, \ldots, \sigma_n$ es una secuencia finita de argumentos (posiblemente vacía), tal que $conc(\sigma_1) = \phi_1, \ldots, conc(\sigma_n) = \phi_n$ para alguna regla $\phi_1, \ldots, \phi_n \rightarrow \phi$ de R y ϕ no pertenece a $sent(\sigma_1) \cup \ldots \cup sent(\sigma_n)$. Entonces:

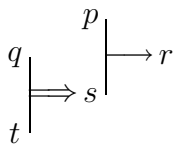


Figura 3.9: Un argumento en el sistema de Vreeswijk

$prem(\sigma) = prem(\sigma_1) \cup \ldots \cup prem(\sigma_n)$
$conc(\sigma) = \phi$
$sent(\sigma) = sent(\sigma_1) \cup \ldots \cup sent(\sigma_n) \cup \{\phi\}$
$asum(\sigma) = asum(\sigma_1) \cup \ldots \cup asum(\sigma_n)$
$sub(\sigma) = sub(\sigma_1) \cup \ldots \cup sub(\sigma_n) \cup \{\sigma\}$
$top(\sigma) = \{\tau_1, \ldots, \tau_n \rightarrow \phi \mid \tau_1 \in top(\sigma), \ldots, \tau_n \in top(\sigma)\} \cup \{\phi\}$
$length(\sigma) = \text{máx}\{length(\sigma_1), \ldots, length(\sigma_n)\} + 1$
$size(\sigma) = size(\sigma_1) + \ldots + size(\sigma_n) + 1$

3. Una expresión $\sigma_1, \ldots, \sigma_n \Rightarrow \phi$, donde $\sigma_1, \ldots, \sigma_n$ es una secuencia finita de argumentos (posiblemente vacía), tal que $conc(\sigma_1) = \phi_1, \ldots, conc(\sigma_n) = \phi_n$ para alguna regla $\phi_1, \ldots, \phi_n \Rightarrow \phi$ de R y ϕ no pertenece a $sent(\sigma_1) \cup \ldots \cup sent(\sigma_n)$. En este caso,

$$asum(\sigma) = asum(\sigma_1) \cup \ldots \cup asum(\sigma_n) \cup \phi$$

y el resto de los elementos se definen como en (2).

■

La introducción de un conjunto de elementos auxiliares en la definición de argumento es por demás inoportuna y afecta su claridad conceptual. Dado que es posible recuperar toda la información necesaria a partir de la estructura del argumento, una alternativa simple y correcta consiste en formalizar estos elementos como operadores unarios sobre un argumento. La condición que solicita la no pertenencia de ϕ a $sent(\sigma_1) \cup \ldots \cup sent(\sigma_n)$ ya estaba presente en los argumentos de Lin y Shoham para evitar la formación de argumentos con encadenamientos circulares de reglas redundantes. Sin embargo, no previene el agregado de reglas irrelevantes, que no contribuyen en la obtención de la conclusión y comparte los problemas de la definición 3.28.

En un abuso de notación, un argumento puede denotarse como $\sigma_1, \ldots, \sigma_n \rightarrow \sigma$ para abreviar que σ es un argumento que se construye a partir de $\sigma_1, \ldots, \sigma_n$ por medio de la regla $\phi_1, \ldots, \phi_n \rightarrow \phi$, tal que $conc(\sigma_1) = \phi_1, \ldots, conc(\sigma_n) = \phi_n$ y $conc(\sigma) = \phi$. Esta terminología también es válida para las reglas rebatibles. Cuando sea posible omitiremos los paréntesis en la notación; con esta convención $(p \rightarrow q) \rightarrow r$ se simplifica como $p \rightarrow q \rightarrow r$.

Ejemplo 3.23 En la figura 3.9 se muestra la representación gráfica de un argumento compuesto por las reglas $\{q, t \Rightarrow s\ ;\ p, s \rightarrow r\}$ utilizando como premisas al conjunto $\{q, t, p\}$. ■

Los argumentos se clasifican en *atómicos* (cuando están formados por un elemento de $\mathcal{L}$) y *compuestos* en caso contrario. Es también posible distinguir entre argumentos *estrictos* y *rebatibles*. Un argumento σ es *estricto* si $\sigma \in \mathcal{L}$ o $\sigma = \sigma_1, \ldots, \sigma_n \rightarrow \sigma$ y $\sigma_1, \ldots, \sigma_n$ son argumentos estrictos; en caso contrario el argumento se considera *rebatible*. Para caracterizar los argumentos que pueden construirse a partir de un conjunto de elementos de $\mathcal{L}$, se utilizan los siguientes conceptos:

Definición 3.50 (*Argumento basado en un conjunto*)
Sea $\mathcal{L}$ un lenguaje y P un conjunto tal que $P \subseteq \mathcal{L}$. Un argumento σ esta *basado* en P si $prem(\sigma) \subseteq P$; un conjunto de argumentos está basado en P si todos sus elementos lo están. ■

El conjunto de todos los argumentos basados en un conjunto P se denota mediante $Argumentos(P)$. Este conjunto se puede particionar en dos conjuntos disjuntos: los argumentos estrictos basados en P ($Estrictos(P)$) y los argumentos rebatibles basados en P ($Rebatibles(P)$). La clasificación existente para argumentos se extiende a los elementos del lenguaje. Un miembro ϕ de $\mathcal{L}$ esta basado en un conjunto P si es la conclusión de un argumento basado en P. Si ϕ es la conclusión de un argumento estricto (resp. rebatible) basado en P se dice que ϕ esta *estrictamente basado* (resp. *rebatiblemente basado*) en P.

Seguidamente nos concentraremos la relación de orden entre argumentos. Vreeswijk señala que una de las características que distingue al concepto tradicional de prueba de la noción de argumento es que la *fuerza conclusiva* de estos últimos es variable. En base a este concepto, la relación $\leq$ de un SAA define un ordenamiento entre los argumentos, que debe cumplir las siguientes propiedades:

1. La relación $\leq$ debe ser reflexiva y transitiva

2. No esta permita la existencia de cadenas infinitas: $\sigma_1 < \sigma_2 < \ldots < \sigma_n < \ldots$

3. Para todo par de argumentos σ y τ tal que σ es un subargumento de τ, debe valer que $\sigma \leq \tau$.

4. El orden inducido por $<$ debe propagarse a través de las reglas estrictas, es decir, para toda regla $\phi_1, \ldots, \phi_n \rightarrow \phi$ debe valer que $\phi_i \leq \phi$ (para algún i tal que $1 \leq i \leq n$).

Es sorprendente observar que la relación $\leq$ puede no ser un orden, dado que no se exige que posea la propiedad de antisimetría. La segunda condición procura evitar cadenas infinitas para garantizar la finitud del proceso de derrota; el resto aseguran una distribución razonable de la fuerza conclusiva: es claro que un argumento no puede fortalecerse al agregar reglas y tampoco es deseable el debilitamiento poco natural de un argumento al utilizar una regla estricta. Vreeswijk enfatiza que estos requerimientos son lo suficientemente generales para no excluir ningún orden interesante.

No existe consenso en la comunidad académica sobre que criterio de comparación entre argumentos es conveniente utilizar. Algunos investigadores respaldan a la noción de *especificidad* [Poole, 1985] como un acercamiento correcto, pero es innegable que no es suficiente. Vreeswijk se abstiene de definir un orden de comparación concreto y destaca que éste es un tópico difícil, en el cual la raíz del problema radica en la utilización de un métodos puramente sintácticos, como la especificidad. En su opinión, es necesario contar con información adicional sobre la semántica del dominio en consideración. El autor no repara en que la noción de especificidad definida por Poole no es un método puramente sintáctico, dado que toma en cuenta información del dominio en consideración. Además la utilización de un método automatizable de comparación entre argumentos es indispensable en la práctica. En caso contrario se debe dictaminar un orden *ad hoc* entre los argumentos en conflicto al realizar la codificación del conocimiento, tarea imposible de realizar en numerosas situaciones. Por otra parte, un criterio *ad hoc* complica la actualización de la información, dado que al agregar nuevos hechos o reglas debemos actualizar el conjunto de elementos que componen el orden entre argumentos.

Para definir la semántica del sistema, Vreeswijk presenta dos alternativas: una definición inductiva basada en el concepto de argumentos activos por niveles de J. Pollock [Pollock, 1987]; y una formulación en la cual se determina directamente el conjunto de argumentos activos. A continuación analizaremos cada una de ellas.

En primer lugar introduciremos un conjunto de terminología previa. Sea Σ un conjunto de argumentos. Un elemento σ se denomina *primer elemento* con respecto a $\leq$ de Σ si para todo τ perteneciente a Σ se cumple que $\sigma \leq \tau$. Un elemento σ es un *mínimo* con respecto a $\leq$ de Σ si para todo $\tau \in \Sigma$ tal que $\tau \leq \sigma$ se cumple que $\tau = \sigma$. En forma análoga se definen los conceptos de *último elemento* y *máximo* con respecto a $\leq$.

El concepto de *debilitamiento* es uno de los pilares en los que se basa la relación de derrota. Un argumento σ *debilita* a un conjunto de argumentos Σ si domina alguno de los elementos de Σ. En este caso, Σ no es capaz de respaldar a sus elementos frente a un conflicto. Formalmente:

Definición 3.51 (*Debilitamiento*)
Un argumento τ *debilita* a un conjunto de argumentos Σ si existe $\sigma \in \Sigma$ tal que $\sigma < \tau$. ■

Cabe mencionar que cuanto más grande es un conjunto de argumentos mayor es la cantidad de conclusiones que permite sancionar, pero más factible es la existencia de algún elemento que posibilite el debilitamiento.

Puesto que la estructura del lenguaje $\mathcal{L}$ permanece sin especificar, y no se asume la existencia de ningún conectivo que represente la negación, Vreeswijk presenta la siguiente generalización de la noción de consistencia:

Definición 3.52 (*Argumento contradictorio*)
Un argumento σ es *contradictorio* si $conc(\sigma) = \bot$. ■

Definición 3.53 (*Conjunto incompatible - Conjunto compatible*)
Un subconjunto P de $\mathcal{L}$ es *incompatible* si existe un argumento estricto σ basado en P tal que σ es contradictorio. En caso contrario, P es *compatible*. ■

Definición 3.54 (*Conjunto incompatible minimal*)
Se denomina conjunto *incompatible minimal* a todo conjunto incompatible P, tal que todo $S \subset P$ es un conjunto compatible. ■

Este último concepto se extiende a conjuntos de argumentos: un conjunto Σ es compatible si las conclusiones de los elementos de Σ forman un conjunto compatible. No resulta claro por qué el autor elige definir la noción de incompatibilidad basándose

únicamente en argumentos estrictos. De todas formas, al definir la noción de incompatibilidad de un conjunto de argumentos también se toman en cuenta los conflictos que surgen a partir de reglas rebatibles.

Ejemplo 3.24 Sea $\mathcal{L} = \{p, q, r, s\} \cup \{\bot\}$ y sea $R = \{p \Rightarrow r; p, q \rightarrow s; r, s \rightarrow \bot\}$. Entonces todos los subconjuntos de $\{p, q, s\}$ son compatibles, mientras que todos los superconjuntos de $\{p, q, r\}$ son incompatibles. En particular, $\{p, q, r\}$ y $\{r, s\}$ son conjuntos incompatibles minimales. El conjunto de argumentos $\Sigma = \{p \Rightarrow r, q \rightarrow s\}$, es incompatible; esto se evidencia al considerar que sus conclusiones forman el conjunto incompatible $\{r, s\}$. ■

Para formalizar la noción de justificación es necesario identificar un conjunto de información básica respecto al cual se realiza el análisis argumentativo, denominado *conjunto base*. A fin de evitar conflictos entre argumentos estrictos, el conjunto base se define como un subconjunto finito y compatible de los elementos de $\mathcal{L}$. Otra noción elemental para el proceso de inferencia es la *derrota* entre argumentos. En contraste con la mayoría de los formalismos argumentativos, en donde la relación de derrota vincula pares de argumentos, Vreeswijk define a los derrotadores como conjuntos de argumentos, lo cual resta elegancia y claridad conceptual al framework resultante.

Definición 3.55 (*Derrotador*)
Sea P un conjunto base y σ un argumento. Un conjunto de argumentos Σ es un *derrotador* de σ si Σ es incompatible con σ y σ no debilita a Σ. Se dice que Σ es un *derrotador minimal* de σ si ningún subconjunto propio de Σ derrota a σ. ■

Los derrotadores deben afrontar el siguiente compromiso: es conveniente que sean pequeños para que sean más difíciles de debilitar, pero deben sustentar una cantidad de conclusiones suficiente para crear una contradicción con otros argumentos. No queda claro por qué Vreeswijk se aleja de la formulación tradicional de derrota. En tal sentido en [Prakken y Vreeswijk, 1999], los autores señalan que los conjuntos de argumentos son necesarios, debido a que el lenguaje base no esta especificado y carece del poder expresivo necesario para reconocer las inconsistencias. Sin embargo, la derrota entre pares de argumentos podría expresarse de forma sencilla al definir que un argumento σ derrota a un argumento τ si es posible obtener un argumento para $\bot$ a partir de la unión de las conclusiones de σ y τ y se cumple que σ debilita al conjunto $\{\tau\}$.

Para distinguir los argumentos no derrotados el autor propone la noción de *habilitación*. Intuitivamente, un argumento σ está habilitado por un conjunto Σ si y sólo si todos los subargumentos de σ no están derrotados por algún subconjunto de Σ.

Definición 3.56 (*Argumento habilitado*)
Sea P un conjunto base y Σ un conjunto de argumentos. un argumento σ esta *habilitado por* Σ *en base a* P, denotado por $P_{|\Sigma}\sim\sigma$, si vale que

1. El conjunto P contiene a σ,
2. Existe una secuencia de argumentos $\sigma_1, \ldots, \sigma_n$ tal que $P_{|\Sigma}\sim\sigma_1$, $\ldots$, $P_{|\Sigma}\sim\sigma_n$ y $\sigma_1, \ldots, \sigma_n \rightarrow \sigma$, o bien
3. Existe una secuencia de argumentos $\sigma_1, \ldots, \sigma_n$ tal que $P_{|\Sigma}\sim\sigma_1$, $\ldots$, $P_{|\Sigma}\sim\sigma_n$ y $\sigma_1, \ldots, \sigma_n \Rightarrow \sigma$; y se cumple que Σ no contiene derrotadores de σ.

■

La primer condición de la definición 3.56 asegura que todos los elementos de P están habilitados. La segunda requiere que la propiedad de habilitación se propague a través de las reglas estrictas. Por último, la tercer condición implica que σ está habilitado por Σ si y sólo si todos los subconjuntos de Σ son compatibles con σ o pueden ser debilitados por este argumento.

Ejemplo 3.25 Consideremos el sistema argumentativo abstracto $\mathcal{A}$ compuesto por un lenguaje $\mathcal{L} = \{p, q, r\} \cup \{\bot\}$, un conjunto de reglas

$$R = \{p \Rightarrow q, q \Rightarrow r, p \Rightarrow s\} \cup \{q, r \rightarrow \bot\}$$

y un orden de fuerza conclusiva $\leq$. Si $P = \{p\}$ es posible ver que el argumento $\sigma = p \Rightarrow q \Rightarrow r$ está habilitado por $\Sigma = \{p, p \Rightarrow q\}$ en base a P. En efecto, dado que σ finaliza con una regla rebatible debemos ver que $p \Rightarrow q$ este habilitado por Σ y que Σ no contenga derrotadores para σ (como puede constatarse a través de la definición de derrota). Entonces el problema se reduce a determinar si $p \Rightarrow q$ está habilitado por Σ. Procediendo en forma análoga por medio de la tercer condición sólo nos resta ver si p está habilitado, lo cual se verifica trivialmente. ■

El concepto de habilitación se extiende a conjuntos de argumentos como sigue:

Definición 3.57 (*Operador de habilitación*)
Sea P un conjunto base y Σ un conjunto de argumentos. El operador $enable_P(\Sigma)$ denota el conjunto $\{\sigma | P_{|\Sigma} \sim \sigma\}$. ■

La teoría de habilitación sólo tiene en cuenta derrotadores directos. Para manejar otras formas de derrota, tal como reinstanciación y derrota en cascada, Vreeswijk plantea una caracterización inductiva del concepto de justificación, que se corresponde exactamente con la definición por niveles de J. Pollock.[9]

Definición 3.58 (*Niveles de activación*)
Sea P un conjunto base. Un argumento σ está *activo* en el nivel 1 en base a P si y sólo si σ está basado en P. Un argumento σ está *activo* en el nivel n ($n > 1$) en base a P, denotado como $P \vdash_n \sigma$, si vale alguna de las siguientes condiciones:

1. El conjunto P contiene a σ.

2. Existe una secuencia de argumentos $\sigma_1, \ldots, \sigma_n$ tal que $\sigma_1, \ldots, \sigma_n \rightarrow \sigma$ y $P \vdash_n \sigma_1, \ldots, P \vdash_n \sigma_n$.

3. Existe una secuencia de argumentos $\sigma_1, \ldots, \sigma_n$ tal que $\sigma_1, \ldots, \sigma_n \Rightarrow \sigma$ y $P \vdash_n \sigma_1, \ldots, P \vdash_n \sigma_n$; y se cumple que todo conjunto Σ de argumentos que está activo en el nivel $n-1$ no es un derrotador de σ.

La expresión $info_n(P)$ denota el conjunto de argumentos $\{\sigma | P \vdash_n \sigma\}$. ■

La definición 3.58 es equivalente a la aplicación repetida del operador de habilitación sobre un conjunto base. Vreeswijk formula y prueba la siguiente proposición:

Proposición 3.5 Sea P un conjunto base, entonces para cada $n \geq 1$ se cumple que $info_n(P) = enable^n_P(P)$, donde $enable^n_P(P)$ denota la enésima aplicación de operador de habilitación. ■

Basándose en la definición 3.58, se enuncia la siguiente clasificación de argumentos:

- *Argumentos definitivamente no derrotados.* Un argumento σ está definitivamente no derrotado si y sólo si para algún $n > 1$, se cumple que $P \vdash_{n+k} \sigma$ para todo $k > 1$.

[9] En la sección 3.1 se analiza la semántica de esta definición.

- *Argumentos provisoriamente derrotados.* Un argumento σ está provisoriamente derrotado si y sólo si para todo $n > 1$, se cumple que $P \vdash_{n+k} \sigma$, para algún $k > 1$, y no se cumple que $P \vdash_{n+l} \sigma$ para algún $l > 1$.

- *Argumentos definitivamente derrotados.* Un argumento σ está definitivamente derrotado si y sólo si para algún $n > 1$, no se cumple que $P \vdash_{n+k} \sigma$ para todo $k > 1$.

La semántica del sistema se obtiene a partir de esta clasificación de argumentos.

Definición 3.59 (*Justificación inductiva*)
Sea P un conjunto base y sea $n > 1$. Un elemento ϕ está justificado en el nivel n en base a P, denotado como $P \vdash_n \phi$, si existe un argumento σ tal que $P \vdash_n \sigma$ y $conc(\sigma) = \phi$. Se denomina $warrant_n(P)$ al conjunto $conc \circ info_n(P)$. ■

Por lo tanto, esta noción define un tipo de semántica cauta: sólo los argumentos definitivamente no derrotados constituyen las conclusiones del sistema.

Ejemplo 3.26 Sea $\mathcal{A}$ un sistema argumentativo abstracto que posee un lenguaje $\mathcal{L} = \{p, q, r, s, t\}$, un conjunto de reglas

$$R = \{p \Rightarrow q, q \Rightarrow r, p \Rightarrow s, p \Rightarrow t\}$$

y una relación de orden $\leq$ de acuerdo a la cual un argumento σ es mejor que τ si σ posee menor número de reglas que τ. Sea $P = \{p\}$ y supongamos que queremos determinar que argumentos están activos en cada nivel en base a P. Para ésto se desarrolla la secuencia:

$info_1(P) = \{p, p \Rightarrow q, p \Rightarrow q \Rightarrow r, p \Rightarrow s, p \Rightarrow t\}$
$info_2(P) = \{p, p \Rightarrow t\}$
$info_3(P) = \{p, p \Rightarrow q, p \Rightarrow s, p \Rightarrow t\}$
$info_4(P) = \{p, p \Rightarrow t\}$
$info_5(P) = \{p, p \Rightarrow q, p \Rightarrow s, p \Rightarrow t\}$
$\vdots$

Podemos observar que para $n > 1$, $info_{2n}(P) = \{p, p \Rightarrow t\}$ y $info_{2n-1}(P) = \{p, p \Rightarrow q, p \Rightarrow s, p \Rightarrow t\}$. En consecuencia los argumentos $\{p, p \Rightarrow t\}$ están definitivamente no derrotados, $\{p \Rightarrow q, p \Rightarrow s\}$ son argumentos provisoriamente derrotados y $\{p \Rightarrow q \Rightarrow r\}$ está definitivamente derrotado. ■

Vreeswijk también presenta una caracterización no inductiva del proceso de inferencia, basada en el uso de *extensiones*. La principal diferencia entre esta nueva definición y el operador por niveles es que este último determina una relación unívoca entre un conjunto base y un conjunto de argumentos. Por el contrario, la relación de consecuencia que se describe a continuación puede producir varias respuestas.

Definición 3.60 (*Consecuencia rebatible*)
Sea P un conjunto base. Una relación $\sim$ entre P y argumentos basados en P es una *relación de consecuencia rebatible* si para cada argumento σ basado en P se cumple que $P \sim \sigma$ (es decir, σ está activo en base a P) si y sólo si:

1. El conjunto P contiene a σ.

2. Existe una secuencia de argumentos $\sigma_1, \ldots, \sigma_n$ tal que $\sigma_1, \ldots, \sigma_n \rightarrow \sigma$ y $P \sim \sigma_1, \ldots, P \sim \sigma_n$.

3. Existe una secuencia de argumentos $\sigma_1, \ldots, \sigma_n$ tal que $\sigma_1, \ldots, \sigma_n \Rightarrow \sigma$ y $P \sim \sigma_1, \ldots, P \sim \sigma_n$; y vale que todo conjunto de argumentos Σ que está activo en base a P ($P \sim \Sigma$) no es un derrotador de σ.

■

De acuerdo con la tercer condición, para decidir si σ está activo es necesario analizar un conjunto de argumentos Σ y responder una cuestión similar para cada uno de sus elementos. En este punto se evidencia la recursividad de esta formulación. Al estudiar cuidadosamente la definición 3.60, se observa que para un mismo conjunto base puede existir un conjunto de relaciones de consecuencia rebatible válidas, tal como muestra el siguiente escenario.

Ejemplo 3.27 Sea $\mathcal{A}$ un sistema argumentativo abstracto con un lenguaje $\mathcal{L} = \{p, q, r\} \cup \{\bot\}$, un conjunto de reglas

$$R = \{p \Rightarrow q, p \Rightarrow r\} \cup \{q, r \rightarrow \bot\}$$

y una relación de orden $\leq$. Sea $P = \{p\}$, es claro que $P \sim p$ (usando la condición (1) de la definición 3.60), entonces p es un argumento activo. Para determinar si $p \Rightarrow q$ es un argumento activo resta verificar que no exista un conjunto de argumentos Σ, tal que Σ está activo en base P y Σ derrota a $p \Rightarrow q$. Para que se cumpla esta propiedad, cada conjunto de argumentos activo basado en P debe ser compatible

con $p \Rightarrow q$. Considerando que $p \Rightarrow r$ es el único argumento que puede provocar que no se cumpla la tercer claúsula, ésta puede simplificarse como $p \Rightarrow q$ está activo si no está activo $p \Rightarrow r$. Un razonamiento análogo surge al analizar si $p \Rightarrow r$ es un argumento activo: $p \Rightarrow q$ está activo si y sólo si $p \Rightarrow r$ no lo está y viceversa. ■

La situación del ejemplo anterior conduce al concepto de extensión:

Definición 3.61 (*Extensión*)
Un conjunto de argumento Σ es una *extensión* de un conjunto base P si existe alguna relación de inferencia rebatible $\mid\sim$ tal que $\Sigma = \{\sigma | P \mid\sim \sigma\}$. El conjunto $\{\sigma | P \mid\sim \sigma\}$ es la extensión generada por $\mid\sim$ y se denota como $info_{\mid\sim}(P)$. El número de extensiones de P se denomina *grado* de P. Si no hay duda sobre que relación de consecuencia rebatible genera la extensión esta puede denotarse como $info(P)$. ■

A partir de la nueva caracterización, es posible definir el estado de las fórmulas en $\mathcal{L}$.

Definición 3.62 (*Justificación*)
Sea P un conjunto base, un elemento ϕ está *garantizado* en base a P, denotado como $P \mid\sim \phi$, si existe un argumento σ tal que $P \mid\sim \sigma$ y $conc(\sigma) = \phi$. Se denomina $warrant_{\mid\sim}(P)$ al conjunto $conc \circ info_{\mid\sim}(P)$. ■

Esta definición no deja en claro cuándo una fórmula está garantizada. Dado que pueden existir distintas relaciones de consecuencia rebatibles, ¿es suficiente la existencia de una relación en particular para que ϕ este garantizado? De ser así, estamos en presencia de una semántica osada. Vreeswijk declara que, a diferencia de la caracterización anterior, esta semántica no es escéptica. No obstante, no especifica correctamente, ni en la teoría ni en los ejemplos, cuál es el comportamiento de la misma.

Cabe observar que en su disertación doctoral Vreeswijk desarrolla un marco dialéctico para los SAA que modela la semántica de la definición 3.62. El autor observa que esta presentación está motivada por un conjunto de ventajas auspiciadas por la dialéctica, como la transparencia conceptual que provee, la posibilidad de explicar el proceso de argumentación en forma sencilla y la obtención de un razonador guiado por las metas, que permite iniciar un debate sobre una cierta tesis considerando solamente los argumentos relevantes a la misma.

Las distintas semánticas presentadas por Vreeswijk son relacionadas entre sí mediante un conjunto de teoremas. También se describen numerosos ejemplos para

evaluar la aplicabilidad del sistema ante diversos problemas tradicionales en la comunidad de razonamiento no monótono. Por último, el autor estudia algunos métodos para la construcción de extensiones, dado que la caracterización de punto fijo no resulta de utilidad a la hora de implementar el sistema.

En general, la definición de los SAA resulta poco clara y se introducen numerosos conceptos innecesarios que hacen el formalismo un tanto engorroso. El trabajo contiene un análisis de diversos tópicos presentes en los sistemas argumentativos, pero no posee la simplicidad conceptual necesario para formular una teoría abstracta que permita unificar y analizar los formalismos existentes. En nuestra opinion, es razonable afirmar que el sistema desarrollado por Vreesiwjk no es de utilidad práctica, ya que existen numerosos aspectos que no han sido claramente especificados. Por ejemplo, la noción de contradicción es incorporada en forma explícita al indicar cuando dos argumentos son mutuamente excluyentes. Esta caracterización resulta elegante a nivel teórico, pero impracticable al codificar un escenario concreto, ya que en numerosas oportunidades no es posible determinar cuándo una regla debe derivar inconsistencia.

3.5. Frameworks argumentativos abstractos

Phan Ming Dung desarrolló un acercamiento original e interesante a la argumentación rebatible mediante los *frameworks argumentativos abstractos* [Dung, 1995b]. Una característica original de estos sistemas consiste en que la estructura de los argumentos permanece sin especificar y el análisis del autor se centra en la interacción existente entre los mismos.

La noción principal de la teoría de Dung es la de *framework argumentativo*, que se define como un par ordenado compuesto por un conjunto de argumentos y una relación binaria de ataque[10] entre los elementos de este conjunto. Un argumento es una entidad abstracta cuyo único rol está determinado por su relación con otros argumentos. Cabe destacar que Dung no define un sistema en particular, sino que los elementos sin especificar se pueden instanciar de distintas formas y dar origen a un grupo de distintos formalismos.

Definición 3.63 (*Framework argumentativo*)

[10]Cabe aclarar que Dung utiliza el término *ataque* para denotar el concepto denominado como *derrota* en la mayoría de los sistemas argumentativos

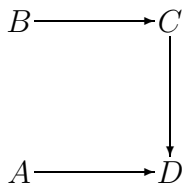


Figura 3.10: Framework argumentativo abstracto

Un *framework argumentativo* consiste en un par:

$$\text{AF} = \langle \text{AR}, \texttt{attacks} \rangle$$

donde AR es un conjunto de argumentos y `attacks` es una relación binaria sobre AR, *i.e.*, $\texttt{attacks} \subseteq \text{AR} \times \text{AR}$. ■

Si consideramos dos argumentos A y B la semántica de la relación `attacks`(A,B) puede expresarse como A *representa un ataque hacia* B.

Ejemplo 3.28 El framework argumentativo $\text{AF} = \langle \text{AR}, \texttt{attacks} \rangle$, en donde $\text{AR} = A, B, C, D$ y $\texttt{attacks} = \{(A, B), (A, C), (C, D), (D, C)\}$ se puede graficar como se muestra en la figura 3.10. ■

La motivación inicial del trabajo de Dung consistió en encontrar una caracterización argumentativa para las distintas semánticas de la programación en lógica. Por lo tanto el autor explora diversas formas de caracterizar a los argumentos, que capturan distintos tipos de semánticas argumentativas.

En primer lugar se presenta la noción de *aceptabilidad*. Dung postula que un argumento A es aceptable para un agente racional G si G puede defender a A de todos los ataques posibles. En consecuencia el conjunto de los argumentos aceptados por G debe ser capaz de defenderse a sí mismo. Esta visión intuitiva se formaliza a través las siguientes definiciones.

Definición 3.64 (*Conjunto libre de conflictos*)
Un conjunto S de argumentos se denomina *libre de conflictos* si no existen dos argumentos A,B en S tal que A ataca a B. ■

Definición 3.65 (*Argumento aceptable*)
Sea $\text{AF} = \langle \text{AR}, \texttt{attacks} \rangle$ un framework argumentativo. Un argumento $A \in \text{AR}$ es *aceptable* con respecto a un conjunto de argumentos S si y sólo si para cada $B \in \text{AR}$ se cumple que si B ataca a A entonces B es atacado por S. ■

Definición 3.66 (*Conjunto admisible*)
Un conjunto libre de conflictos S es *admisible* si y sólo si cada argumento en S es aceptable con respecto a S. ■

Ejemplo 3.29 Consideremos AF = $\langle \text{AR}, \texttt{attacks} \rangle$, donde AR = $\{A, B, C\}$ y la relación $\texttt{attacks} = \{(B, A), (C, B)\}$. Los conjuntos $\emptyset$, $\{C\}$ y $\{A, C\}$ son admisibles; por el contrario $\{A\}$ y $\{B\}$ no lo son. ■

Finalmente la primera de las semánticas para el framework argumentativo se define mediante la noción de *extensión preferida*.

Definición 3.67 (*Extensión preferida*)
Sea AF un framework argumentativo. Se denomina *extensión preferida* de AF a un conjunto de argumentos admisible y maximal con respecto a la inclusión de conjuntos. ■

Una extensión preferida modela un escenario coherente y maximal, ya que debe ser libre de conflictos y clasificar como aceptables a la mayor cantidad posible de argumentos.

Ejemplo 3.30 Continuando con el ejemplo 3.29, la única extensión preferida en este caso es el conjunto $\{A, C\}$. ■

En [Dung, 1995b] se demuestra que siempre existe al menos una extensión preferida, lo que constituye una interesante propiedad de esta semántica. No obstante, en algunos casos conflictivos se pueden sancionar varias extensiones alternativas.

Ejemplo 3.31 Consideremos el famoso diamante de Nixon, que puede ser representado mediante un framework argumentativo AF = $\langle \text{AR}, \texttt{attacks} \rangle$, en donde AR = $\{A, B\}$ y $\texttt{attacks} = \{(B, A), (C, B)\}$. Los argumentos A y B representan respectivamente las aserciones: *Nixon no es pacifista, dado que es republicano* y *Nixon es pacifista porque es un cuáquero*. Este framework posee dos extensiones preferidas: A y B. ■

Seguidamente Dung introduce las *extensiones estables*, que producen una división un tanto extremista en el conjunto de argumentos.

Definición 3.68 (*Extensión estable*)
Un conjunto libre de conflictos S es una *extensión estable* si y sólo si S ataca a todo argumento A tal que $A \notin S$. ■

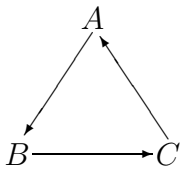


Figura 3.11: Framework argumentativo circular

La definición anterior es equivalente a la siguiente proposición:

Proposición 3.6 El conjunto S es una extensión estable si y sólo si

$$S = \{A \mid A \text{ no es atacado por } S\}$$

■

Esta semántica es más restrictiva que la anterior, pues todos los argumentos que no atacan al conjunto S deben incluirse en él. Dung demuestra que la semántica estable se corresponde con los modelos estables de la programación en lógica [Gelfond y Lifschitz, 1991], las extensiones de la lógica default [Reiter, 1980] y las expansiones estables de la lógica autoepistémica de Moore [Moore, 1984].

Para relacionar las semánticas definidas hasta el momento, Dung enuncia y prueba un lema de acuerdo al cual toda extensión estable es también una extensión preferida. Cabe mencionar que la recíproca no es válida.

Ejemplo 3.32 En los ejemplos 3.29 y 3.31, las extensiones estables coinciden con las preferidas. ■

Ejemplo 3.33 Sea AF = $\langle \text{AR}, \texttt{attacks} \rangle$ un framework argumentativo compuesto por tres argumentos A, B y C tal que A derrota a B, B derrota a C y C derrota a A, como se ilustra en la figura 3.11. AF posee al conjunto vacío como única extensión preferida y no es posible construir extensiones estables. ■

En la semántica estable la existencia de extensiones no está asegurada y esto representa un inconveniente para su uso. Esta situación dio lugar al siguiente planteo controversial *¿es posible construir un sistema argumentativo que represente un problema real y no posea semántica estable?* Dung provee evidencia a favor de una respuesta afirmativa a través de ejemplos obtenidos en las áreas de economía y teoría de juegos.

A continuación se presenta una tercer semántica de caráter escéptico. En lugar de definir esta semántica como la intersección de un conjunto de extensiones osadas, como en numerosos sistemas de razonamiento no monótono, Dung utiliza una teoría de punto fijo que resulta en una elegante formalización.

Definición 3.69 (*Función característica*)
Sea AF = $\langle \text{AR}, \texttt{attacks} \rangle$ un framework argumentativo. La *función característica* de AF, denotada por F_{AF}, se define como sigue:

$$F_{AF} : 2^{AF} \mapsto 2^{AF},\ F_{AF}(S) = \{A \mid A \text{ es aceptable con respecto a S}\}$$

■

Esta función permite determinar cuando un conjunto S de argumentos libre de conflictos es admisible. Dung demuestra que S es admisible si vale que $S \subseteq F_{AF}(S)$. Puede también probarse que F_{AF} es monótona: si un argumento A es aceptable con respecto a S debe ser aceptable con respecto a cualquier superconjunto de S.

En base a la definición 3.69 se enuncia la noción de extensión básica:

Definición 3.70 (*Extensión básica*)
Sea AF un framework argumentativo. La *extensión básica* de AF, denotado como GE_{AF}, se obtiene como el menor punto fijo de la función F_{AF}. ■

Una extensión básica puede construirse en base al siguiente proceso. En primer lugar se incorporan al conjunto los argumentos que no poseen derrotadores. Luego se van agregando los argumentos que son reinstanciados por alguno de los elementos del conjunto. Considerando que la función F_{AF} es monótona, es posible asegurar la existencia de un menor punto fijo que puede ser aproximado mediante la aplicación repetida de F_{AF}.

Ejemplo 3.34 El menor punto fijo correspondiente al framework argumentativo del ejemplo 3.29 se obtiene como $F^0_{AF}(\emptyset) = \{C\}$, $F^1_{AF}(\emptyset) = \{A, C\}$, $F^1_{AF} = F^2_{AF}$. ■

Ejemplo 3.35 En el framework del ejemplo 3.31 la extensión básica es el conjunto vacío, pues ante un situación conflictiva un razonador escéptico no obtiene ninguna de las conclusiónes alternativas. ■

Para establecer una conexión entre las distintas semánticas especificadas anteriormente se presenta la noción de *extensión completa*. Esta semántica modela a un razonador confiado que cree en todo lo que puede defender.

Definición 3.71 (*Extensión completa*)
Un conjunto de argumentos admisible S se denomina una *extensión completa* si y sólo si contiene cada argumento que es aceptable con respecto a S. ■

En base a este concepto Dung enuncia y demuestra las siguientes propiedades:

1. Un conjunto de argumentos libre de conflictos E es una extensión completa si y sólo si $E = F_{AF}(E)$.

2. Toda extensión preferida es una extensión completa.

3. Una extensión básica es la mínima extensión completa (con respecto a la inclusión de conjuntos).

Por último Dung define condiciones necesarias para la equivalencia de las distintas semánticas. Luego se abordan diversos ejemplos tradicionales en las áreas de economía y teoría de juegos para demostrar la aplicabilidad de la argumentación a problemas reales. Para ilustrar la expresividad de la teoría desarrollada, Dung manifiesta que varios sistemas de razonamiento no monótono [McDermott y Doyle, 1980, Pollock, 1987, Poole, 1988, Simari y Loui, 1992, Reiter, 1980] pueden ser codificados como instanciaciones de la misma. En particular se detallan las formulaciones de la lógica default [Reiter, 1980], para representar de los acercamientos basados en extensiones y la lógica inductiva de Pollock [Pollock, 1987], para ejemplificar un sistema basado en argumentos. Esta capacidad hace del trabajo de P. M. Dung una herramienta invaluable para estudiar las diferencias y similitudes existentes entre las diversas lógicas no monótonas [Prakken, 1997].

El trabajo de Dung tiene continuidad en un árticulo denominado *"An abstract argumentation-theoretic approach to default reasoning* [Bondarenko et al., 1997] en el cual un argumento se interpreta como una deducción monótona a partir de un conjunto de premisas abductivas. En este framework se fusionan el sistema de Dung descripto anteriormente con la noción de extensión presente en THEORIST [Poole, 1988]. En consecuencia puede interpretarse como una generalización del sistema THEORIST que permite extender mediante un conjunto de suposiciones rebatibles cualquier teoría formulada sobre una lógica monótona. El objetivo del formalismo de Bondarenko *etal*. consiste en modelar la semánticas de un conjunto de lógicas no monótonas con una semántica alternativa, general y unificadora, que utiliza a la argumentación para resolver conflictos entre distintos conjuntos de suposiciones. Los autores muestran como es posible codificar estas lógicas como instancias particulares del framework propuesto.

3.6. La programación en lógica rebatible

La *programación en lógica rebatible* (PLR) [García, 1997, García, 2000] surge como un formalismo unificador que combina las virtudes de la argumentación rebatible y la programación en lógica. El mecanismo de inferencia de la PLR está basado en el sistema argumentativo de Simari y Loui [11] y en consecuencia resulta en un lenguaje de gran expresividad que permite representar información incompleta y potencialmente contradictoria. A continuación reseñaremos los principales conceptos de la programación en lógica rebatible de acuerdo a lo expuesto en [García, 2000].

3.6.1. Lenguaje

Para definir el lenguaje de la PLR es necesario introducir un conjunto de conceptos que forman parte de la terminología estándar de la programación en lógica.

Definición 3.72 (*Signatura*)
Una signatura es una tupla $\sigma = \langle \mathcal{V}, \mathit{Func}, \mathit{Pred} \rangle$, donde $\mathcal{V}$, *Func* y *Pred* son conjuntos finitos de variables, funciones y predicados respectivamente. ■

Como es usual, un predicado sin argumentos es una *proposición* y una función 0-aria se denomina *constante*. Los términos del lenguaje se definen inductivamente a partir de las variables, constantes y funciones del programa. Todo término T que no contiene variables se denomina término fijo.

Definición 3.73 (*Átomo*)
Sea $\sigma = \langle \mathcal{V}, \mathit{Func}, \mathit{Pred} \rangle$ una signatura y $t_1, \ldots, t_n$ un conjunto de términos basados en σ. Si existe un predicado $p \in \mathit{Pred}$ con aridad n entonces $p(t_1, \ldots, t_n)$ es un *átomo* basado en σ. Si los términos $t_1, \ldots, t_n$ son fijos entonces $p(t_1, \ldots, t_n)$ es un *átomo fijo*. ■

Los literales son átomos que pueden estar precedidos por el símbolo de la negación fuerte '~'. El complemento de un literal L con respecto a la negación fuerte, $\overline{L}$, se define como $\sim A$, si $L = A$ o simplemente A si $L = \sim A$.

El lenguaje de la programación en lógica rebatible se define en base a las nociones anteriores. Cada programa lógico rebatible tiene asociada una signatura σ que está compuesta solamente por un conjunto de funciones *Func* y conjunto de

[11] Este formalismo fue detallado en la sección 3.2

predicados *Pred*, dado que la sintaxis de estos programas no incluye variables. Un programa lógico rebatible está formado por *hechos*, *reglas estrictas* y *reglas rebatibles*. Un hecho es un literal fijo. La sintaxis de las reglas estrictas y rebatibles se define a continuación:

Definición 3.74 (*Regla estricta*)
Una *regla estricta* es un par ordenado, denotado como '`cabeza`← `cuerpo`', donde '`cabeza`' es un literal fijo y '`cuerpo`' es un conjunto finito no vacío de literales fijos. Una regla estricta con cabeza L_0 y cuerpo $\{L_1, \ldots, L_n\}$ $(n > 0)$ puede también denotarse mediante $L_0 \leftarrow L_1, \ldots, L_n$. ■

Definición 3.75 (*Regla rebatible*)
Una *regla rebatible* es un par ordenado, denotado como '`cabeza` $\prec$ `cuerpo`', donde '`cabeza`' es un literal fijo y '`cuerpo`' es un conjunto finito no vacío de literales fijos. Una regla estricta con cabeza L_0 y cuerpo $\{L_1, \ldots, L_n\}$ $(n > 0)$ puede también denotarse mediante $L_0 \prec L_1, \ldots, L_n$. ■

Aunque la sintaxis de las ambas clases de reglas es similar, el autor destaca que su semántica es diferente. Las reglas estrictas codifican conocimiento seguro y libre de excepciones: siempre que se cree en los literales en el cuerpo de una regla r se cree en la cabeza de r. Por el contrario, las reglas rebatibles se utilizan para representar información tentativa y establecen una conexión débil entre el cuerpo y la cabeza . De la misma manera que en el sistema de Simari y Loui [Simari y Loui, 1992], la semántica de la relación denotada por el conectivo ' $\prec$ ' puede interpretarse como *"en la mayoría de los casos, creer en el cuerpo de una regla nos conduce a creer en su cabeza"*. En consecuencia las reglas rebatibles expresan una propiedad general del dominio en consideración y para que estas reglas tengan sentido tanto el cuerpo como la cabeza deben contener variables libres que se puedan instanciar con cada caso en particular. Una regla rebatible formada puramente por literales fijos es verdadera o falsa. En caso de ser verdadera debería modelarse como una regla estricta y si es falsa no debe existir.

Sin embargo, como puede observarse a partir de la definición 3.75, la PLR sólo utiliza literales fijos. Esta situación motivó la introducción de *esquemas de reglas rebatibles*, es decir elementos que comparten las características de las reglas rebatibles con la particularidad que pueden contener variables. Cabe destacar que las variables utilizadas en los esquemas no forman parte del lenguaje sino que denotan una meta-relación entre el cuerpo y la cabeza, mediante la cual es posible representar con un

esquema a todo un conjunto de reglas rebatibles. Formalmente un esquema se define como el conjunto de todas las reglas fijas que representa.

En base a las reglas estrictas y rebatibles se introduce el siguiente concepto:

Definición 3.76 (*Programa lógico rebatible*)
Un *programa lógico rebatible* $\mathcal{P} = (\Pi, \Delta)$ es un conjunto de reglas estrictas, hechos y reglas rebatibles, donde Π denota al conjunto de hechos y reglas estrictas y Δ al conjunto de reglas rebatibles. ■

Ejemplo 3.36 A continuación se ilustra un programa lógico rebatible que codifica información sobre un grupo de aves.

Π	Δ
`ave(X)` $\leftarrow$ `gallina(X)`	`vuela(X)` $\prec$ `ave(X)`
`ave(X)` $\leftarrow$ `pingüino(X)`	~`vuela(X)` $\prec$ `gallina(X)`
`ave(X)` $\leftarrow$ `paloma(X)`	`vuela(X)` $\prec$ `gallina(X), asustada(X)`
`paloma(loly)`	~`vuela(X)` $\prec$ `cansada(X)`
`gallina(tina)`	
`pingüino(tweety)`	
`cansada(tweety)`	

■

3.6.2. Proceso de inferencia

El mecanismo de inferencia del sistema decide qué hechos están *justificados* a partir de la información contenida en un determinado programa. En la PLR este proceso está basado en el formalismo de argumentación rebatible de G. R. Simari y R. P. Loui [Simari y Loui, 1992] y se define a partir de las nociones que se detallan a continuación.

Definición 3.77 (*Derivación rebatible*)
Sea $\mathcal{P} = (\Pi, \Delta)$ un programa lógico rebatible y L un literal. Una *derivación rebatible* para L a partir de $\mathcal{P}$ consiste en una secuencia finita de literales fijos $L_1, L_2, \ldots, L_n$ tal que $L_n = L$ y cada L_i $(1 \leq i \leq n)$ cumple las siguientes condiciones:

1. L_i es un hecho, o

2. existe una regla R_i en $\mathcal{P}$ tal que la cabeza de $R_i = L_i$ y el cuerpo de R_i está formado por un conjunto de literales $B_1, B_2, \ldots, B_k$, donde todo B_j $(1 \leq j \leq k)$ es un elemento de la secuencia que precede a L_i.

■

La noción de derivación rebatible se hereda del sistema de Simari y Loui (ver sección 3.2). Sin embargo consituye una restricción de la noción de consecuencia rebatible (definición 3.11) dado que las reglas fuertes que ya no se utilizan los axiomas para obtener inferencias y las reglas estrictas no forman parte del lenguaje sino que conforman simplemente reglas de inferencia. Además los elementos que pueden derivarse a partir de un programa $\mathcal{P}$ se restringen a literales fijos, abandonando la lógica de primer orden en la cual se basa el sistema de Simari y Loui.

En todo programa lógico rebatible $\mathcal{P} = (\Pi, \Delta)$ se asume que el conjunto Π no es contradictorio, esto es, que no es posible construir una derivación para un literal y su complemento utilizando solamente reglas estrictas y hechos en Π. Sin embargo, es factible obtener derivaciones de literales complementarios a partir de un mismo programa $\mathcal{P}$, dado que las reglas en Δ pueden representar información potencialmente contradictoria. El mecanismo de inferencia debe resolver este tipo de situaciones.

El concepto de inferencia rebatible permite definir la noción de *estructura de argumento*, que es la piedra angular del proceso de inferencia de la PLR.

Definición 3.78 (*Estructura de argumento*)
Sea h un literal y $P = (\Pi, \Delta)$ un programa lógico rebatible. El par $\langle \mathcal{A}, h \rangle$ es una *estructura de argumento* para h si y sólo si $\mathcal{A}$ es un conjunto de reglas rebatibles que cumple las siguiente condiciones:

1. Es posible construir una derivación rebatible para h utilizando las reglas y hechos presentes en $\Pi \cup \mathcal{A}$.

2. $\Pi \cup \mathcal{A}$ no es un conjunto contradictorio.

3. $\mathcal{A}$ es minimal con respecto a la inclusión de conjuntos, esto es, no existe $\mathcal{A}' \subset \mathcal{A}$ tal que $\mathcal{A}'$ satisface la primer claúsula.

Una estructura de argumento $\langle \mathcal{B}, q \rangle$ es una subestructura de argumento de $\langle \mathcal{A}, h \rangle$ si $\mathcal{B} \subset \mathcal{A}$. ■

Cabe destacar que la noción de argumento basada en estas restricciones se desarrolló por primera vez en [Simari y Loui, 1992], donde se requería además que el conjunto $\mathcal{A}$ estuviera compuesto por instancias fijas de las reglas rebatibles en Δ. En la PLR este requerimiento no es necesario (recordar que las reglas están formadas por literales fijos).

Ejemplo 3.37 Las estructuras de argumento que se describen a continuación se pueden obtener a partir del programa lógico rebatible presentado en el ejemplo 3.36.

- $\langle \mathcal{A}_1, \texttt{vuela(tina)} \rangle$, donde $\mathcal{A}_1 = \texttt{vuela(tina)} \prec \texttt{ave(tina)}$.
- $\langle \mathcal{A}_2, \sim\texttt{vuela(tina)} \rangle$, donde $\mathcal{A}_2 = \sim\texttt{vuela(tina)} \prec \texttt{gallina(tina)}$.
- $\langle \mathcal{A}_3, \texttt{vuela(tina)} \rangle$, donde
  $\mathcal{A}_3 = \texttt{vuela(tina)} \prec \texttt{gallina(tina), asustada(tina)}$.

■

En algunas situaciones dos argumentos pueden sustentar conclusiones contradictorias, lo que hace imposible su aceptación conjunta. Este concepto se define formalmente a través de la relación de *desacuerdo*. Una estructura de argumento $\langle \mathcal{A}, h \rangle$ está en desacuerdo con $\langle \mathcal{B}, q \rangle$ si el $\Pi \cup \{q, h\}$ es un conjunto contradictorio. Observemos que esta noción, presente en el sistema de Simari y Loui, ha sido modificada de acuerdo a la noción de derivación existente en la PLR, que no coincide con la inferencia clásica. En consecuencia se exige que el conjunto formado por $\Pi \cup \{q, h\}$ no sea contradictorio.

La relación de *ataque* o *contraargumentación* surge a partir de la noción de desacuerdo.

Definición 3.79 (*Ataque*)
Sean $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ dos estructuras de argumento construidas a partir de un programa $\mathcal{P}$. $\langle \mathcal{A}_1, h_1 \rangle$ *ataca* a $\langle \mathcal{A}_2, h_2 \rangle$ es un literal h si y sólo si existe una subestructura de argumento $\langle \mathcal{A}, h \rangle$ de $\langle \mathcal{A}_2, h_2 \rangle$ tal que $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}, h \rangle$ están en desacuerdo. El literal h se denomina *punto de contrargumentación*. ■

Para determinar que argumento prevalece ante un ataque es necesario introducir un *criterio de comparación*, que permite decidir qué argumento prevalece en el

conflicto. García presenta dos propuestas alternativas para comparar argumentos. La primer propuesta está basada en la noción de especificidad definida por Poole [Poole, 1985] y se denomina *especificidad generalizada*. Este mecanismo favorece a los argumentos que utilizan más información o sustentan su conclusión en forma más directa. La segunda propuesta utiliza un orden explícito entre las reglas rebatibles del programa y define un orden parcial entre argumentos en base al mismo. Este método posee algunos inconvenientes, dado que en numerosas situaciones no es posible escoger en forma adecuada o realista las prioridades entre las reglas rebatibles. Por otra parte, esta clase de criterio complica la actualización de la información, dado que al agregar nuevos hechos o reglas debemos actualizar el orden existente entre las reglas, mientras que la especificidad generalizada no se ve afectada por la actualización de la información.

A partir de la discusión previa queda claro que el criterio de comparación puede definirse de varias formas diferentes, siempre que induzca un orden parcial sobre el conjunto de argumentos. En la definición que se presenta a continuación la noción de *derrota* entre argumentos del sistema de Simari y Loui es modificada para que no dependa de un criterio de comparación en particular. El autor asume la existencia de un orden parcial denotado mediante el símbolo '$\succ$', que permite decidir cuando un argumento es preferible a otro. De esta manera se define un sistema flexible, que no se compromete con un criterio de preferencia particular.

Definición 3.80 (*Derrotador*)
Sean $\langle\langle A, , \rangle h\rangle$ y $\langle \mathcal{B}, q\rangle$ dos estructuras de argumentos. $\langle\langle A, , \rangle h\rangle$ es un *derrotador* para $\langle \mathcal{B}, q\rangle$ si, y sólo si:

1. $\langle\langle A, , \rangle h\rangle$ es un *derrotador propio* de $\langle \mathcal{B}, q\rangle$, esto es, existe una subsestructura de argumento $\langle \mathcal{B}_1, q_1\rangle$ de $\langle \mathcal{B}, q\rangle$ tal que $\langle\langle A, , \rangle h\rangle$ ataca a $\langle \mathcal{B}, q\rangle$ en el literal q_1 y $\langle\langle A, , \rangle h\rangle \succ \langle \mathcal{B}_1, q_1\rangle$.

2. $\langle\langle A, , \rangle h\rangle$ es un *derrotador de bloqueo* de $\langle \mathcal{B}, q\rangle$, esto es, existe una subsestructura de argumento $\langle \mathcal{B}_1, q_1\rangle$ de $\langle \mathcal{B}, q\rangle$ tal que $\langle\langle A, , \rangle h\rangle$ ataca a $\langle \mathcal{B}, q\rangle$ en el literal q_1 y $\langle\langle A, , \rangle h\rangle \not\succ \langle \mathcal{B}_1, q_1\rangle$ y $\langle \mathcal{B}_1, q_1\rangle \not\succ \langle\langle A, , \rangle h\rangle$.

■

Para decidir si un literal h se puede inferir a partir de un programa $\mathcal{P}$, es necesario analizar la totalidad de las estructuras de argumento, tanto a favor como en contra de h, que pueden construirse en base a $\mathcal{P}$. De la misma manera que en el sistema de

[Simari et al., 1994], este análisis se define formalmente a partir de los conceptos de *árbol dialéctico* y *línea de argumentación.*[12] Este último elemento se define en forma análoga a la definición 3.20 y en consecuencia no evita la ocurrencia de situaciones falaces en el proceso de inferencia, como las consideradas en los ejemplos 3.13 y 3.14. García presenta un nuevo mecanismo para eliminar estas situaciones, redefiniendo la noción de *línea argumental aceptable*:

Definición 3.81 (*Línea de argumentación aceptable*)
Una línea de argumentación $\lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ es *aceptable* si, y sólo si se cumplen las siguientes condiciones.

1. Las estructuras de argumento de soporte (interferencia) en λ son conjuntos concordantes. [13]

2. Ningúna estructura de argumento $\langle \mathcal{A}_k, h_k \rangle$ en λ es una subestructura de argumento de $\langle \mathcal{A}_i, h_i \rangle$ (donde $i < k$ y por lo tanto $\langle \mathcal{A}_i, h_i \rangle$ aparece previamente en λ).

3. Para toda estructura de argumento $\langle \mathcal{A}_i, h_i \rangle$ en λ tal que $\langle \mathcal{A}_i, h_i \rangle$ es un derrotador de bloqueo de $\langle \mathcal{A}_{i-1}, h_{i-1} \rangle$, si existe $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ entonces se cumple que $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ es un derrotador propio de $\langle \mathcal{A}_i, h_i \rangle$.

■

La tercer claúsula de la definición anterior evita la aparición de dos derrotas por bloqueo consecutivas. Esta es una decisión de diseño que tiene por objetivo eliminar la *acumulación de razones*,[14] es decir, el sistema no acepta que un literal h prevalezca sólo por estar sustentado por un mayor número de estructuras de argumento que su complemento $\overline{h}$.

Finalmente es posible enunciar las definiciones de árbol dialéctico y marcado de un árbol dialéctico que resumen el proceso de inferencia de la PLR.

Definición 3.82 (*Árbol dialéctico*)
Sea $\langle \mathcal{A}, h \rangle$ una estructura de argumento obtenida a partir de un programa $\mathcal{P}$. El árbol dialéctico para $\langle \mathcal{A}, h \rangle$ en base a $\mathcal{P}$, denotado $\mathcal{T}_{\langle \mathcal{A}, h \rangle}$, se define como sigue:

[12] En la sección 3.2.1 se discurre sobre estos conceptos en mayor detalle.

[13] La noción de concordancia de un conjunto de estructuras de argumento se define de la misma manera que en la sección 3.2.1.

[14] Traducción del término inglés *accrual of reasons.*

1. La raíz del árbol es etiquetada con $\langle \mathcal{A}, h \rangle$.

2. Sea $\langle \mathcal{A}_n, h_n \rangle$ un nodo de $\mathcal{T}_{\langle \mathcal{A},h \rangle}$, $[\langle \mathcal{A}, h \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ la secuencia de estructuras de argumento que se encuentran en el camino desde la raíz hasta el nodo $\langle \mathcal{A}_n, h_n \rangle$ y sea $\{\langle \mathcal{B}_0, q_0 \rangle, \langle \mathcal{B}_1, q_1 \rangle, \ldots, \langle \mathcal{B}_k, q_k \rangle\}$ el conjunto de derrotadores de $\langle \mathcal{A}_n, h_n \rangle$. Para cada $\langle \mathcal{B}_i, q_i \rangle$ $(1 \leq i \leq k)$ tal que

$$[\langle \mathcal{A}, h \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \langle \mathcal{B}_i, q_i \rangle]$$

es una línea de argumentación aceptable se cumple que $\langle \mathcal{B}_i, q_i \rangle$ es un nodo hijo de $\langle \mathcal{A}_n, h_n \rangle$ en $\mathcal{T}_{\langle \mathcal{A},h \rangle}$. Si no existe ningún derrotador $\langle \mathcal{B}_i, q_i \rangle$ en estas condiciones entonces $\langle \mathcal{A}_n, h_n \rangle$ es un nodo hoja.

■

Definición 3.83 (*Marcado de un árbol dialéctico*)
Sea $\mathcal{T}_{\langle A,h \rangle}$ un árbol dialéctico para $\langle \mathcal{A}, h \rangle$. El árbol dialéctico marcado de $\mathcal{T}_{\langle A,h \rangle}$, denotado $\mathcal{T}^*_{\langle A,h \rangle}$, se obtiene de la siguiente forma:

1. todas las hojas de $\mathcal{T}_{\langle A,h \rangle}$ se marcan como no derrotadas,

2. si n es un nodo interno de $\mathcal{T}_{\langle A,h \rangle}$ entonces n se marca como no derrotado si todos los hijos de n están marcados como derrotados y n se marca como derrotado si posee al menos un hijo marcado como no derrotado.

■

Ejemplo 3.38 A partir de los argumentos presentados en el ejemplo 3.36 es posible construir un árbol dialéctico para $\langle \mathcal{A}_1, \texttt{vuela(tina)} \rangle$ en donde $\langle \mathcal{A}_3, \texttt{vuela(tina)} \rangle$ aparece como hijo de $\langle \mathcal{A}_2, \sim\texttt{vuela(tina)} \rangle$, quien a su vez es hijo de la raíz. ■

A partir de estas nociones se define el conjunto de *literales garantizados*, esto es, los literales que conforman las inferencias del sistema.

Definición 3.84 (*Literales garantizados*)
Sea $\mathcal{P} = (\Pi, \Delta)$ un programa lógico rebatible y h un literal. Se dice que h está *garantizado* si existe al menos una estructura de argumento para h basada en $\mathcal{P}$, denominada $\langle \mathcal{A}, h \rangle$, tal que la raíz del árbol de dialéctica marcado asociado a $\langle \mathcal{A}, h \rangle$, $\mathcal{T}^*_{\langle A,h \rangle}$, (construido de acuerdo a la información en $\mathcal{P}$) está marcada como un nodo no derrotado. ■

Finalmente García desarrolla una optimización para el proceso de construcción de los árboles dialécticos basada en un mecanismo de poda al estilo del que fuera presentado en [Chesñevar, 1996], que permite agilizar la obtención de inferencias.

El análisis de la PLR como formalismo de representación de conocimiento y razonamiento nos permite concluir que constituye un sistema con atractivas propiedades prácticas y teóricas. Estas virtudes son el resultado de combinar elegantemente las cualidades de la programación en lógica y la argumentación rebatible. En consecuencia la PLR puede usarse como un formalismo de representación de conocimiento en agentes obteniendo una razonable eficiencia y una alta expresividad. Esta alternativa ha sido explorada en diversos artículos, *e.g.*, [Capobianco y Chesñevar, 2000, Capobianco y Simari, 2000].

3.7. Conclusiones

A lo largo de este capítulo hemos estudiado diversos sistemas de argumentación rebatible con diferentes características. Sin embargo, en todos estos formalismos es posible identificar una serie de elementos en común. Tal se expresa en [Prakken y Vreeswijk, 1999], las nociones de argumento, relación de conflicto, relación de derrota y estado de los argumentos (a partir de la cual se definen las consecuencias del sistema) están presentes en todos los formalismos considerados. A continuación analizaremos cada uno de estos elementos en particular, considerando cómo están definidos en los distintos sistemas.

Las nociones de argumento propuestas pueden clasificarse de acuerdo a su estructura. La categoría más simple consiste en argumentos con una estructura abstracta. Esta visión posee una gran generalidad y permite concentrarse en el estudio de otros componentes del sistema (como la relación de derrota o la clasificación del estado de los argumentos). Sin embargo, la estructura de los argumentos debe especificarse para poder aplicar el formalismo a un problema concreto. Pollock emplea una noción de argumento extremadamente simple y define un argumento como un par (`premisas, conclusion`) tal que alguna de las reglas del lenguaje conecta las premisas con su conclusión. Esto crea inconvenientes en el formalismo OSCAR, dado que la simpleza de los argumentos complica la definición del resto de los elementos. Tanto Vreeswijk como Prakken y Sartor definen un argumento como una derivación basada en las fórmulas del lenguaje. En particular, la definición de Vreeswijk resulta un tanto engorrosa, ya que introduce un conjunto de elementos auxiliares

que complican el entendimiento y análisis de esta noción. Otra de las alternativas es definir un argumento como un conjunto de elementos del lenguaje lógico subyacente, tal como sucede en el sistema de Simari-Loui y la programación en lógica rebatible de García. En estos formalismos se incorpora además un conjunto de condiciones que deben cumplir los argumentos válidos, para evitar que argumentos incorrectos participen del proceso de inferencia. Cabe destacar que la estructura elegida para los argumentos influye sobre la definición de las relaciones de conflicto y derrota. Por ejemplo, el sistema de Simari y Loui solamente incluye reglas rebatibles en la estructura de los argumentos. Esto permite focalizarse en los conflictos sobre el conocimiento tentativo, y dejar al conocimiento fuerte fuera del análisis.

En las relaciones de derrota utilizadas por los diversos autores pueden distinguirse varias clases: derrota por rebatimiento, derrota por socavamiento y derrota compuesta. La derrota por rebatimiento depende de las sentencias que componen a un determinado argumento. Por ejemplo, un argumento con A con q como conclusión que derrota a un argumento B con una conclusión $\neg q$. Los formalismos de Simari y Loui, Prakken y Sartor, Pollock y García poseen este tipo de conflictos. La derrota por socavamiento depende de un paso de inferencia utilizado para construir el argumento. En este caso no se invalida la conclusión sino que se cuestiona la validez del paso realizado al obtener la conclusión a partir de las premisas. Este tipo de derrota solamente se encuentra presente en el sistema de Pollock. La derrota compuesta posee la particularidad de relacionar un único argumento con un conjunto de argumentos. Considerando los sistemas analizados, esta categoría sólo existe en el sistema de Vreeswijk. Entendemos que la derrota compuesta no es adecuada, dado que confunde la computación de la relación de derrota con el análisis del estado de los argumentos. En consecuencia el formalismo resultante pierde elegancia y claridad conceptual.

Finalmente, resta considerar como se comportan los formalismos analizados en términos del mecanismo elegido para determinar el estado de los argumentos. En general, la relación de derrota provee información relativa sobre la fuerza conclusiva de los dos argumentos en conflicto y no es capaz de determinar el estado final de los mismos, que depende de la interacción entre todos los argumentos que puedan construirse a partir de la base de conocimiento. A partir de la definición del estado de los argumentos se obtienen las conclusiones del sistema argumentativo.

La semántica de los formalismos argumentativos estudiados se dividen fundamentalmente en declarativas o procedimentales. Las alternativas declarativas usual-

mente utilizan una ecuación de punto fijo y declaran como justificados a un conjunto de argumentos que cumple las propiedades indicadas en la misma. Los sistemas de Simari y Loui, Prakken y Sartor y Dung siguen esta idea. En contraste, la definición procedural especifica un procedimiento para determinar cuando un argumento en particular es miembro de este conjunto, como es el caso de [Simari et al., 1994] y la programación en lógica rebatible. Si bien las versiones declarativas son conceptualmente claras y elegantes, las versiones procedimentales parecen brindar implementaciones con mayor eficiencia.

Finalmente cabe destacar que a través del análisis de realizado en este capítulo identificamos un conjunto de características que deberían estar presentes en un sistema argumentativo. En primer lugar, y como en todo modelo, la declaratividad y simpleza conceptual del sistema juegan un rol preponderante. La división del sistema en distintas capas conceptuales (tal como fuera propuesta por Prakken y Vreeswijk en [Prakken y Vreeswijk, 1999]) favorece esta propiedad, al mismo tiempo que produce un sistema modular, en el cual se pueden introducir modificaciones con mayor facilidad. Es importante distribuir la complejidad uniformemente en estas capas. Por ejemplo, una noción débil de argumento conduce a una definición compleja del estado de los argumentos, como sucede en OSCAR. La eficiencia es también un elemento a tener en cuenta a la hora de producir aplicaciones basadas en argumentación. En general, los formalismos existentes son computacionalmente costosos y carecen de optimizaciones. La programación en lógica rebatible [García, 2000] constituye una excepción en este sentido, dado que logra alcanzar un interesante compromiso entre eficiencia y poder expresivo.

Capítulo 4

La Programación en Lógica Rebatible basada en Observaciones

La construcción de agentes de software capaces de razonar, planificar sus acciones y actuar en un ambiente dinámico es vital para la implementación de gran cantidad de aplicaciones. Una de las tareas claves al diseñar esta clase de agentes consiste en modelar su estado epistémico en forma adecuada. Tradicionalmente se han utilizado sistemas basados en lógicas modales para realizar esta tarea [Bratman et al., 1988, Rao y Georgeff, 1991]. Si bien estos formalismos permiten un interesante desarrollo teórico, los mismos son computacionalmente costosos y su implementación sigue siendo un problema abierto. En consecuencia, las potenciales ventajas de las lógicas modales se diluyen a la hora de realizar una implementación concreta, dado que existe poca relación entre los modelos teóricos y sus implementaciones [Rao y Georgeff, 1995].

No obstante, existen otros acercamientos para representar el estado epistémico de un agente que utiliza a la programación en lógica [Lloyd, 1987]. Por caso, en el trabajo de Pereira y Quaresma [Pereira y Quaresma, 1998], los agentes se definen como programas lógicos extendidos [Gelfond y Lifschitz, 1991] y sus creencias se obtienen mediante la semántica bien fundada[1] del programa lógico en cuestión. Esta propuesta posee una importante limitación: aunque la programación en lógica extendida puede representar una clase interesante de bases de conocimiento no monótonas en forma relativamente sencilla, no es capaz de trabajar con información incompleta y potencialmente contradictoria. Para desarrollar un agente con estas capacidades, esenciales en gran cantidad de aplicaciones, es necesario utilizar un

[1] En inglés: *well-founded semantics*.

sistema con mayor poder expresivo.

La programación en lógica rebatible (PLR) [García y Simari, 2003] reune en gran medida las propiedades necesarias para modelar el conocimiento de un agente inteligente. Este formalismo combina las ventajas de la programación en lógica y la argumentación rebatible, permitiendo obtener poder expresivo e implementabilidad. Sin embargo, a partir de un análisis detallado de la PLR, descubrimos que no resulta adecuada para modelar agentes que actúan en ambientes dinámicos, dado que no provee mecanismos para incorporar información a la base de conocimiento. Por otra parte, sería deseable optimizar el mecanismo de inferencia del sistema para poder interactuar con el ambiente en forma rápida y eficiente.

Estas razones motivaron la definición de un nuevo formalismo argumentativo, basado en la PLR, que contara con las capacidades anteriormente mencionadas. El lenguaje resultante, denominado *Programación en Lógica Rebatible basada en Observaciones* (PLRO), se presenta en detalle en este capítulo. A continuación definiremos formalmente el lenguaje y el mecanismo de inferencia de la PLRO.

4.1. Definición del lenguaje

A fin de describir el lenguaje de la programación en lógica rebatible basada observaciones, presentaremos en primer lugar un conjunto de terminología estándar de la programación en lógica.[2] Comenzaremos introduciendo los elementos a partir de los cuales se construyen los programas y las consultas. Para esto enunciaremos primeramente el concepto de signatura, que parametriza el lenguaje del sistema con respecto a una situación en particular, ya que contiene los elementos que varían de un programa a otro.

Definición 4.1 (*Signatura*)
Una *signatura* Σ es una tupla $\langle \mathcal{V}, \mathit{Pred}, \mathit{Func} \rangle$, donde $\mathcal{V}$ es un conjunto numerable de *variables*, *Pred* es un conjunto finito de predicados y *Func* es un conjunto finito de funciones, tales que $\mathcal{V} \cap (\mathit{Pred} \cup \mathit{Func}) = \emptyset$. ■

Siguiendo la notación estándar de PROLOG las variables se denotan con identificadores que comienzan con letras mayúsculas, mientras que las funciones y predicados comienzan con letras minúsculas. Es posible distinguir entre estos últimos

[2] Las definiciones de esta sección están basadas en el trabajo de J. W. Lloyd [Lloyd, 1987]. En particular la definición de signatura utilizada, sobre la cual se construyen los conceptos de términos y literales, fue presentada en [Fillotrani, 2001].

por su ubicación dentro del programa. Cada signatura tiene asociada una función de *aridad* que asigna un número natural a cada una de las funciones y predicados. Si una función f es tal que $aridad(f) = 0$ se denomina *constante* y si un predicado p es tal que $aridad(p) = 0$ se denomina *proposición*.

Combinando los miembros de la signatura con un conjunto de símbolos especiales presentes en todos los programas es posible definir el *alfabeto* de la PLRO.

Definición 4.2 (*Alfabeto*)
El *alfabeto* generado a partir de los miembros de una signatura Σ está formado por los miembros de Σ, el símbolo "~" para la negación fuerte [Gelfond y Lifschitz, 1991] y los símbolos de puntuación "(", ")", "." y ",". ■

Los *términos* representan los objetos del ambiente que están presentes en el modelo epistémico del agente. Las propiedades de estos objetos se describen mediante los *átomos* del lenguaje.

Definición 4.3 (*Término*)
Sea $\Sigma = \langle \mathcal{V}, Pred, Func \rangle$ una signatura. Un *término* de Σ se define inductivamente como sigue:

1. toda variable $V \in \mathcal{V}$ es un término,
2. toda constante $c \in Func$ es un término,
3. si $f \in Func$, $aridad(f) = n$ y $t_1, \ldots, t_n$ son términos entonces $f(t_1, \ldots, t_n)$ también es un término. ■

Definición 4.4 (*Átomo*)
Sea $\Sigma = \langle \mathcal{V}, Pred, Func \rangle$ una signatura, $t_1, \ldots, t_n$ términos de Σ y $p \in Pred$ tal que $aridad(p) = n$ entonces $p(t_1, \ldots, t_n)$ es un *átomo* de Σ. ■

Los átomos son las unidades básicas del lenguaje. A partir de la interacción entre estos elementos y la negación fuerte surgen los siguientes conceptos.

Definición 4.5 (*Literal*)
Sea Σ una signatura, entonces todo átomo A de Σ es un *literal positivo* mientras que todo átomo negado $\sim A$ es un *literal negativo*. Un *literal* de Σ es un literal negativo o un literal positivo. ■

Definición 4.6 (*Complemento de un literal*)
Sea L un literal y A un átomo. El *complemento* de L, notado como $\overline{L}$, se define de la siguiente manera:

$$\overline{L} = \begin{cases} A & \text{si } L = {\sim}A \\ {\sim}A & \text{si } L = A \end{cases}$$ ■

Es interesante introducir un conjunto de terminología estándar referente a los literales. Se dice que dos literales L_1 y L_2 son *contradictorios* si L_1 es el complemento de L_2. Un conjunto de literales se dice contradictorio si al menos dos de sus miembros son contradictorios. Un literal es *fijo* cuando no contiene variables.

Dado que la PLRO está basada en el sistema de la PLR [García y Simari, 2003] (analizado previamente en la sección 3.6), resulta oportuno comparar ambos formalismos teniendo en cuenta sus similitudes y diferencias. En consecuencia, en cada uno de los conceptos heredados de la PLR, detallaremos las diferencias con la versión original. Como el mecanismo de inferencia de la PLR está a su vez basado en el sistema de [Simari et al., 1994] (considerado en la sección 3.2) en algunos casos realizaremos un estudio de mayor profundidad, destacando la evolución de un determinado concepto desde su definición inicial hasta su inclusión en la PLRO.

Intgroduciremos ahora los programas lógicos rebatibles con observaciones. Estos programas están compuestos por dos conjuntos disjuntos de elementos: las *observaciones* y las *reglas rebatibles*. Las observaciones describen los hechos concretos que el agente conoce acerca de su ambiente. Estos hechos pueden haber sido obtenidos a través de un mecanismo de percepción o codificados explícitamente en la creación del agente. Las reglas rebatibles son reglas de inferencia que, tal como sucede en la programación en lógica rebatible [García y Simari, 2003], representan información tentativa o sujeta a excepciones. Estas reglas permiten al agente ampliar el conocimiento presente en las observaciones con deducciones de carácter tentativo.

Definición 4.7 (*Observación*)
Una *observación* es un literal fijo. ■

Definición 4.8 (*Regla rebatible*)
Una *regla rebatible* es un par ordenado, denotado como `cabeza` $\prec$ `cuerpo` , donde el primer componente, `cabeza`, es un literal y el segundo, `cuerpo`, es un conjunto no vacío de literales, tal que la intersección entre las variables presentes en `cabeza` y `cuerpo` es un conjunto no vacío. ■

Ejemplo 4.1 Algunos ejemplos de reglas rebatibles son:
`vuela(X)` $\prec$ `ave(X)`,
`ciudadano(X,Y)` $\prec$ `nacido(X,Y)`,
`llueve(X)` $\prec$ `nublado_cielo(X), presion_baja(X)`.

Por el contrario `vuela(tweety)` $\prec$ `ave(tweety)` y `vuela(Y)` $\prec$ `ave(Z)` no son reglas rebatibles. ■

Definición 4.9 (*Instancia fija de una regla rebatible*)
Sea δ una regla rebatible. Una *instancia fija* γ de δ se obtiene reemplazando cada una de las variable de δ por un término fijo, de forma tal que las variables de δ con el mismo nombre sean reemplazadas consistentemente. ■

Ejemplo 4.2 Algunas instancias de la regla rebatible

`ciudadano(X,Y)` $\prec$ `nacido(X,Y)`

son `ciudadano(italia,giuseppe)` $\prec$ `nacido(italia,giuseppe)` y
`ciudadano(argentina,carlos)` $\prec$ `nacido(argentina,carlos)`.
Por el contrario `ciudadano(italia,giuseppe)` $\prec$ `nacido(argentina,carlos)` no es una instancia de `ciudadano(X,Y)` $\prec$ `nacido(X,Y)`.
Para la regla rebatible `vuela(X)` $\prec$ `ave(X)` algunas de sus instancias posibles son `vuela(tweety)` $\prec$ `ave(tweety)` y `vuela(claude)` $\prec$ `ave(claude)`. Por el contrario, `vuela(tweety)` $\prec$ `ave(claude)` no es una instancia de `vuela(X)` $\prec$ `ave(X)`.
■

Es importante resaltar que dado su carácter de reglas de inferencia, las reglas rebatibles no son elementos del lenguaje. Se asume que las variables con el mismo nombre que están presentes tanto en el cuerpo como en la cabeza de una regla rebatible representan el mismo elemento. Cada regla rebatible es un *esquema* que representa al conjunto de sus *instancias fijas*.

Pragmáticamente, las reglas rebatibles deben contener variables, dado que modelan un esquema de comportamiento general y no un caso en particular. Esta consideración fue introducida en el sistema de Simari y Loui, [Simari y Loui, 1992] en donde se exige que las reglas rebatibles contengan variables para que su semántica tenga sentido.

En base a las reglas rebatibles y las observaciones se definen los *programas lógicos rebatibles basados en observaciones*, que codificarán el conocimiento del agente.

Definición 4.10 (*Programa lógico rebatible basado en observaciones*)
Un *programa lógico rebatible basado en observaciones* P está formado por un conjunto Ψ de observaciones y un conjunto Δ de reglas rebatibles tales que:

1. tanto Ψ como Δ son conjuntos finitos, y

2. Ψ es un conjunto no contradictorio de literales.

Cuando la situación así lo requiera denotaremos al programa P mediante el par (Ψ, Δ). ■

La segunda restricción presente en la definición 4.10 establece que el conjunto de premisas del agente posea un cierto grado de coherencia. Esta misma política se encuentra presente tanto en el sistema de Simari y Loui [Simari y Loui, 1992] como en la programación en lógica rebatible [García y Simari, 2003], donde el conjunto de conocimiento seguro debe ser consistente para que la obtención de inferencias a partir del mismo resulte de interés.

Observemos las diferencias entre un programa de la PLR y la definición 4.10. En la PLRO hemos eliminado el conjunto de reglas estrictas presentes en los programas lógicos rebatibles. Esta modificación permite obtener un acercamiento de mayor simplicidad y, como veremos en lo que resta de esta tesis, el sistema resultante posee la expresividad suficiente para ser utilizado en gran cantidad de aplicaciones.[3]

Ejemplo 4.3 Las reglas que se exponen a continuación componen un programa lógico con observaciones $\mathcal{P} = (\Psi, \Delta)$ que contiene información sobre un grupo de pequeños felinos.

Ψ	Δ
`gato(tom).`	`tiene-cola(X)` $\prec$ `gato(X).`
`gato(grace).`	`~tiene-cola(X)` $\prec$ `gato(X), manx(X).`
`manx(grace).`	`~sociable(X)` $\prec$ `solitario(X).`
`joven(tom).`	`sociable(X)` $\prec$ `gato(X), joven(X).`
`mascota(tom).`	`sociable(X)` $\prec$ `mascota(X).`
	`solitario(X)` $\prec$ `gato(X).`

[3]Los beneficios de esta alternativa se evidencian en etapas posteriores del trabajo, dado que por ejemplo, permiten obtener mecanismos de percepción para el sistema más simples y eficientes.

Las observaciones establecen que `tom` y `grace` son gatos. Además `tom` es un individuo joven y una mascota, mientras que `grace` pertenece a la raza manx. El conjunto Δ contiene reglas que codifican las siguientes afirmaciones: un gato generalmente posee una cola, los gatos que pertenecen a la raza manx no tienen cola, un individuo solitario no suele ser sociable, un gato de corta edad es sociable, las mascotas son generalmente sociables y los gatos suelen ser solitarios. ■

4.2. Procedimiento de inferencia

El procedimiento de inferencia de la PLRO permite obtener conclusiones a partir de la información almacenada en un programa. De acuerdo con la definición de las reglas rebatibles, un programa lógico rebatible con observaciones puede contener información en conflicto. Por otra parte, en gran cantidad de situaciones un agente debe razonar utilizando información incompleta acerca del mundo. En consecuencia es deseable que el mecanismo de inferencia de la PLRO sea capaz de obtener conclusiones a partir de información incompleta y potencialmente contradictoria. Para alcanzar esta capacidad la PLRO adopta la misma política que la programación en lógica rebatible [García y Simari, 2003], combinando un proceso de inferencia basado en la argumentación rebatible.

A fin de describir el método para obtener conclusiones de la PLRO, definiremos en primer lugar qué clase de inferencias es posible obtener a partir de un programa lógico con observaciones. En este sentido hemos elegido concentrarnos solamente en literales fijos dado que esto simplifica y agiliza el proceso de inferencia. Por lo tanto un programa lógico rebatible basado en observaciones responderá consultas sobre el conjunto de literales fijos presentes en su signatura.

El primer paso para responder una consulta q consiste en construir una *derivación* para q, si es esto posible. Para determinar que literales es posible derivar a partir de un programa dado introduciremos el concepto de *derivación rebatible*. La definición de este concepto en la PLRO se corresponde con la definición 3.77, pero debe modificarse en forma acorde con la nueva formulación de reglas rebatibles.

Definición 4.11 (*Derivación rebatible*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa y q un literal fijo. Una secuencia finita de literales fijos $s = q_1, q_2, \ldots, q_{n-1}, q$ es una *derivación rebatible* para q a partir de $\mathcal{P}$, denotado como $\mathcal{P} \mid\sim q$, si para cada q_i existe una regla rebatible $r \in \Delta$ y una instancia fija t de

r, $t = q_i \prec l_1, \ldots, l_m$, donde $l_1, \ldots, l_m$ son literales fijos que aparecen previamente en la secuencia s. ■

Cabe destacar que aunque el conjunto Ψ debe ser no contradictorio, un programa $\mathcal{P} = (\Psi, \Delta)$ puede permitir derivar dos literales complementarios, dado que no se exige ninguna restricción sobre las reglas rebatibles presentes en Δ. A partir de esta situación queda en evidencia que la existencia de una derivación para un literal q no es suficiente para respaldarlo como una conclusión válida. Esto motiva que se exploren nuevas alternativas para sustentar los literales que es posible derivar a partir del programa, introduciendo la noción de *argumento*, elemento clave para el procedimiento de inferencia de la PLRO.

Definición 4.12 (*Argumento – Subargumento*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa, un *argumento* $\mathcal{A}$ para un literal fijo q, denotado como $\langle \mathcal{A}, q \rangle$ o simplemente $\mathcal{A}$, es un subconjunto de instancias fijas de las reglas rebatibles en Δ tal que:

1. $\Psi \cup \mathcal{A} \vdash q$, *i.e.*, existe una derivación rebatible para q a partir de Ψ utilizando las reglas en $\mathcal{A}$,
2. el conjunto de literales que se infieren rebatiblemente a partir de $\Psi \cup \mathcal{A}$ es no contradictorio y
3. no existe $\mathcal{A}' \subset \mathcal{A}$ tal que $\Psi \cup \mathcal{A}' \vdash q$.

Un argumento $\langle \mathcal{A}_1, q_1 \rangle$ es un *subargumento* de $\langle \mathcal{A}_2, q_2 \rangle$ si $\mathcal{A}_1 \subseteq \mathcal{A}_2$. ■

La presente noción de argumento se desarrolló originalmente en el sistema de Simari y Loui. En la versión presentada dentro de la PLRO ésta se adapta para que concuerde con el resto de los elementos del sistema (como los conceptos de reglas rebatibles y programas lógicos rebatibles con observaciones).

Ejemplo 4.4 Los argumentos que se presentan a continuación se obtienen a partir del programa detallado en el ejemplo 4.3.

- $\langle \mathcal{A}_1, \texttt{tiene-cola(grace)} \rangle$, donde
  $\mathcal{A}_1 = \{\texttt{tiene-cola(grace)} \prec \texttt{gato(grace)}\}$.
- $\langle \mathcal{A}_2, \sim\texttt{tiene-cola(grace)} \rangle$, donde
  $\mathcal{A}_2 = \{\sim\texttt{tiene-cola(grace)} \prec \texttt{gato(grace),manx(grace)}\}$

- $\langle \mathcal{A}_3, \sim$ `sociable(tom)`$\rangle$, donde
  $\mathcal{A}_3 = \{\sim$`sociable(tom)` $\prec$ `solitario(tom)`, `solitario(tom)` $\prec$ `gato(tom)`$\}$
- $\langle \mathcal{A}_4,$ `sociable(tom)`$\rangle$, donde
  $\mathcal{A}_4 = \{$`sociable(tom)` $\prec$ `gato(tom), joven(tom)`$\}$
- $\langle \mathcal{A}_5,$ `sociable(tom)`$\rangle$, donde
  $\mathcal{A}_5 = \{$`sociable(tom)` $\prec$ `mascota(tom)`$\}$

El primer argumento establece que `Grace` posee una cola puesto que es un gato. El segundo argumento afirma que `Grace` no posee cola dado que pertenece a una raza particular de gatos que nacen sin ésta. $\mathcal{A}_3$ concluye que `Tom` no es individuo sociable porque es un gato y estos animales son en general solitarios. Sin embargo, $\mathcal{A}_4$ afirma que `Tom` es sociable porque es un animal joven y $\mathcal{A}_5$ también sustenta esta conclusión dado que `Tom` es una mascota. ■

Las siguiente definiciones presentan una serie de conceptos útiles para analizar la estructura de un argumento.

Definición 4.13 (*Cabezas - Cuerpos*)
Sea
ARGA un argumento para una conclusión h , entonces $cabezas(\mathcal{A})$ (respectivamente $cuerpos(\mathcal{A})$) denota los literales que aparecen en la cabeza (respectivamente cuerpo) de las instancias de reglas rebatibles que componen $\mathcal{A}$. ■

Definición 4.14 (*Literales de un argumento*)
Sea $\mathcal{A}$ un argumento para una conclusión h, entonces $literales(\mathcal{A}) = cabezas(\mathcal{A}) \cup cuerpos(\mathcal{A})$ denota al conjunto de los literales presentes en $\mathcal{A}$. ■

Definición 4.15 (*Soporte de un argumento*)
Sea $\mathcal{A}$ un argumento para una conclusión h, entonces el conjunto

$$\mathcal{S}(\mathcal{A}) = cuerpos(\mathcal{A}) - cabezas(\mathcal{A})$$

es llamado el *soporte de* $\mathcal{A}$. ■

Ejemplo 4.5 Consideremos el argumento $\mathcal{A}_3$ presentado en el ejemplo 4.4. En este caso:

- $cabezas(\mathcal{A}_3) = \{\sim$`sociable(tom)`, `solitario(tom)`$\}$,

- $cuerpos(\mathcal{A}_3) = \{\texttt{solitario(tom)}, \texttt{gato(tom)}\}$
- y $\mathcal{S}(\mathcal{A}_3) = \{\texttt{gato(tom)}\}$.

■

4.2.1. Contraargumentación y derrota

En la PLRO una respuesta a una consulta q debe estar respaldada por un argumento $\mathcal{A}$ que sustente a q. Sin embargo un argumento puede ser *derrotado* por otros argumentos. Informalmente diremos que una consulta tiene éxito si el argumento que la sustenta no está derrotado. Para hallar los derrotadores de un argumento $\mathcal{A}$ se analiza en primer lugar los argumentos que contradicen o están en conflicto con $\mathcal{A}$. Para identificar claramente estas situaciones se introduce la siguiente definición.

Definición 4.16 (*Contraargumentación*)
Un argumento $\langle \mathcal{A}_1, q_1 \rangle$ *contraargumenta* a un argumento $\langle \mathcal{A}_2, q_2 \rangle$ en un literal q si y sólo si existe un subargumento $\langle \mathcal{A}, q \rangle$ de $\langle \mathcal{A}_2, q_2 \rangle$ tal que el conjunto $\{q_1, q\}$ es contradictorio. ■

Los argumentos en relación de contraargumentación plantean un conflicto que se debe resolver determinando cual de ellos prevalece. Cabe observar que la definición 4.16 es una simplificación del concepto de *ataque* de la PLR (definición 3.79). Para caracterizar la relación de ataque entre argumentos, en la PLR se utiliza la noción de *desacuerdo*, que establece cuando dos argumentos están en conflicto examinando el conjunto de conocimiento estricto Π con respecto a los argumentos en consideración. De esta forma se toma en cuenta la interacción de las reglas estrictas. En el caso de la PLRO basta con verificar si el conjunto de literales de los argumentos en consideración es o no contradictorio, dado que en este formalismo no existen reglas estrictas. Esto hace más sencillo determinar cuando dos argumentos están en conflicto.

Informalmente, una consulta q en la PLRO tiene éxito si existe un argumento $\langle \mathcal{A}, q \rangle$ no *derrotado* (es decir, que prevalece en el análisis dialéctico). En este caso diremos que q está *garantizada* por el sistema. Para decidir si un argumento $\mathcal{A}$ permanece sin derrotar es necesario sopesarlo con respecto a cada uno de sus contraargumentos. Esta comparación permite averiguar si $\mathcal{A}$ es derrotado por alguno de sus contendientes, esto es, si alguno de los contraargumentos en cuestión es preferido al argumento $\mathcal{A}$ de acuerdo a un criterio de comparación preestablecido.

Para comparar argumentos en la PLRO es posible usar cualquier criterio que resulte adecuado para la aplicación en particular del sistema. Cabe destacar que la especificidad definida por Poole [Poole, 1985] es un criterio de comparación que posee gran generalidad y en consecuencia desarrollaremos el análisis del sistema sobre este criterio. No obstante no es mandatorio utilizar este criterio, sino que puede ser reemplazado en forma modular, obteniendo un sistema con un comportamiento diferente. A continuación definiremos una versión de especificidad adecuada para la PLRO.

Definición 4.17 (*Especificidad en la PLRO*)
Sea $\mathcal{P}$ un programa y sea $lit(P)$ el conjunto de literales fijos que es posible derivar a partir de $\mathcal{P}$. Un argumento $\langle \mathcal{A}_1, h_1 \rangle$ es estrictamente más específico que un argumento $\langle \mathcal{A}_2, h_2 \rangle$ (denotado como $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) si y sólo si

1. Para cada $H \subseteq lit(P)$ vale que $H \cup \mathcal{A}_1 \mathrel{|\!\sim} h_1$ y $H \not\mathrel{|\!\sim} h_1$ implican que $H \cup \mathcal{A}_2 \mathrel{|\!\sim} h_2$.
2. Existe $H' \subseteq lit(P)$ tal que $H' \cup \mathcal{A}_2 \mathrel{|\!\sim} h_2$, $h_2 \notin H'$ y $H' \cup \mathcal{A}_1 \not\mathrel{|\!\sim} h_1$.

■

Intuitivamente, la especificidad favorece aquellos argumentos que son más directos o contienen más información. Este criterio fue utilizado previamente para comparar argumentos en el sistema de Simari y Loui [Simari y Loui, 1992] y en la programación en lógica rebatible [García y Simari, 2003].

Para comprender cómo funciona la especificidad analizaremos en detalle la definición 4.17. La primer cláusula ($H \cup \mathcal{A}_1 \mathrel{|\!\sim} h_1$) normalmente se satisface con un conjunto H distinto de vacío, dado que los argumentos no contienen hechos. En este caso se dice que H *activa* a $\mathcal{A}_1$. La expresión $H \not\mathrel{|\!\sim} h_1$ evita los casos triviales, ya que fuerza el uso del conjunto H para derivar h_1. Por lo tanto la definición 4.17 puede interpretarse como $\langle \mathcal{A}_1, h_1 \rangle$ es más específico que $\langle \mathcal{A}_2, h_2 \rangle$ si y sólo si para cada conjunto H tal que H activa no trivialmente a $\langle \mathcal{A}_1, h_1 \rangle$ se cumple que H activa no trivialmente a $\langle \mathcal{A}_2, h_2 \rangle$.

Ejemplo 4.6 Consideremos los argumentos que se detallan en el ejemplo 4.4. En este caso el argumento $\langle \mathcal{A}_2, \sim$`tiene-cola(tom)`$\rangle$ es estrictamente más específico que el argumento $\langle \mathcal{A}_1,$ `tiene-cola(tom)`$\rangle$. En efecto, al aplicar la definición 4.17 cada subconjunto de $lit(P)$ que activa a $\mathcal{A}_1$ también activa a $\mathcal{A}_2$, pero existen subconjuntos de $lit(P)$ (como por ejemplo {`gato(tom)`}) que activan a $\mathcal{A}_2$ y no activan a $\mathcal{A}_1$. ■

Observación 4.1 Cabe destacar que es posible utilizar una formulación alternativa de la especifidad que resulta equivalente a la definición 4.17: Sea $\mathcal{P}$ un programa y sea *lit*(P) el conjunto de literales fijos que es posible derivar a partir de $\mathcal{P}$. Un argumento $\langle \mathcal{A}_1, h_1 \rangle$ es más específico que un argumento $\langle \mathcal{A}_2, h_2 \rangle$ (denotado como $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) si y sólo si para cada $H \subseteq$ *lit*(P) vale que $H \cup \mathcal{A}_1 \sim h_1$ y $H \not\sim h_1$ implican que $H \cup \mathcal{A}_2 \sim h_2$. $\langle \mathcal{A}_1, h_1 \rangle$ es estrictamente más específico que $\langle \mathcal{A}_2, h_2 \rangle$ (denotado como $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) si y sólo si $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle \not\succeq \langle \mathcal{A}_1, h_1 \rangle$. Para mas detalles sobre esta formulación es posible consultar [Stolzenburg et al., 2000]. ■

En la definición 4.17 debemos considerar todos los literales fijos presentes en la signatura del programa. Si estudiamos el problema con mayor detalle descubriremos que en realidad no es necesario tener en cuenta a los subconjuntos de *lit*(P). Para determinar si un argumento $\mathcal{A}_1$ es más específico que un argumento $\mathcal{A}_2$ basta con examinar a los subconjunto de $literales(\mathcal{A}_1)$. Esta nueva formulación permitiría optimizar eventuales implementaciones de la especificidad en la PLRO.

Definición 4.18 (*Especificidad en la PLRO (2)*)
Un argumento $\langle \mathcal{A}_1, h_1 \rangle$ es estrictamente más específico que un argumento $\langle \mathcal{A}_2, h_2 \rangle$ (denotado como $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) si y sólo si

1. Para cada $H \subseteq literales(\mathcal{A}_1)$ vale que $H \cup \mathcal{A}_1 \sim h_1$ y $H \not\sim h_1$ implican que $H \cup \mathcal{A}_2 \sim h_2$.

2. Existe $H' \subseteq literales(\mathcal{A}_1)$ tal que $H' \cup \mathcal{A}_2 \sim h_2$, $h_2 \notin H'$ y $H' \cup \mathcal{A}_1 \not\sim h_1$.

■

Observación 4.2 Tal como en el caso de la definición 4.17 es posible utilizar una formulación alternativa y equivalente: un argumento $\langle \mathcal{A}_1, h_1 \rangle$ es más específico que un argumento $\langle \mathcal{A}_2, h_2 \rangle$ (denotado como $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) si y sólo si para cada $H \subseteq literales(\mathcal{A}_1)$ vale que $H \cup \mathcal{A}_1 \sim h_1$ y $H \not\sim h_1$ implican que $H \cup \mathcal{A}_2 \sim h_2$. $\langle \mathcal{A}_1, h_1 \rangle$ es estrictamente más específico que $\langle \mathcal{A}_2, h_2 \rangle$ (denotado como $\langle \mathcal{A}_1, h_1 \rangle \succ \langle \mathcal{A}_2, h_2 \rangle$) si y sólo si $\langle \mathcal{A}_1, h_1 \rangle \succeq \langle \mathcal{A}_2, h_2 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle \not\succeq \langle \mathcal{A}_1, h_1 \rangle$.

■

Es posible probar que el orden obtenido utilizando la definición 4.18 es idéntico al que provee la definición 4.17, aunque la definición 4.18 es claramente más eficiente.

Cabe destacar que por cuestiones de simplicidad en esta demostración utilizaremos las definiciones alternativas introducidas en las observaciones 4.1 y 4.2.

En primer lugar probaremos una serie de resultados auxiliares. La proposición 4.1 muestra que cada conjunto de activación H de un argumento $\mathcal{A}$ debe poseer literales en común con $literales(A)$.

Proposición 4.1 Para cada conjunto de activación H de un argumento $\mathcal{A}$ tal que $H \cup \mathcal{A} \sim h$ y $h \notin H$ vale que $H \cap literales(A) \neq \emptyset$ ■

Demostración: Supongamos que H es un conjunto de activación no trivial para un argumento $\langle \mathcal{A}, h \rangle$, tal que $H \cap literales(\mathcal{A}) = \emptyset$. Como H es un conjunto de activación no trivial de $\mathcal{A}$ debe existir una derivación rebatible para h a partir de $\mathcal{A} \cup H$. Sin embargo, no es posible construir una derivación a partir de $\mathcal{A} \cup H$ para un literal h_i tal que $h_i \in literales(\mathcal{A})$. Entonces $H \cup \mathcal{A} \not\sim h_i$ y en particular $H \cup \mathcal{A} \not\sim h$. Esto constituye un absurdo que proviene de suponer $H \cap literales(\mathcal{A}) = \emptyset$. □

Dado un determinado argumento $\mathcal{B}$ y un conjunto de literales H, sólo es necesario examinar los literales de H que pertenecen a las reglas en $\mathcal{B}$ para determinar si H activa a este argumento. En concordancia con esta afirmación, el siguiente lema demuestra que los elementos significativos del conjunto de activación de un determinado argumento $\mathcal{B}$ son los literales que pertenecen a las reglas de este argumento.

Lema 4.1 Sea $\langle \mathcal{A}, h \rangle$ un argumento construido a partir de un programa $\mathcal{P}$, y sean H_1 y H_2 dos conjuntos de literales presentes en la signatura de $\mathcal{P}$ tal que $H_1 \cap literales(\mathcal{A}) = H_2 \cap literales(\mathcal{A})$. Entonces H_1 activa a $\mathcal{A}$ si y sólo si H_2 activa a $\mathcal{A}$. ■

Demostración: (1) $\Rightarrow$ (2): Supongamos que H_1 activa a $\mathcal{A}$. En este caso, para cualquier literal l_i en $literales(\mathcal{A})$ se cumple que si $H_1 \cup \mathcal{A} \vdash l_i$ entonces $H_2 \cup \mathcal{A} \vdash l_i$. Es posible demostrar esta afirmación realizando inducción sobre la longitud de la derivación δ_1 para l_i a partir de $H_1 \cup \mathcal{A}$:

1. Si $\delta_1 = l_i$ entonces $l_i \in H_1$. En esta situación es posible construir una derivación δ_2 para l_i a partir de $H_2 \cup \mathcal{A}$ tal que $l_i \in literales(\mathcal{A})$ y $H_1 \cap literales(\mathcal{A}) = H_2 \cap literales(\mathcal{A})$. Luego l_i debe pertenecer a H_2.

2. Supongamos que para cada derivación δ_1 con una longitud menor o igual a $n-1$ vale que existe una derivación δ_2 a partir de $H_2 \cup \mathcal{A}$. Consideremos $\delta_1 = q_0, \ldots, q_{n-1}, q_n$, tal que $q_n = l_i$. Si $l_i \in H_1$ entonces $l_i \in H_2$ (como se

demostró en el caso base). Entonces existe una derivación δ_2 para l_i a partir de $H_2 \cup \mathcal{A}$. En caso contrario l_i debe ser el consecuente de una regla rebatible $r \in \mathcal{A}$. Utilizando la hipótesis inductiva, para cada literal q_i en δ_1 existe una derivación δ_i a partir de $H_2 \cup \mathcal{A}$. Podemos afirmar que cada literal l_r en el cuerpo de r debe pertenecer a δ_1. Entonces, utilizando la hipótesis inductiva, existe una derivación para l_r a partir de $H_2 \cup \mathcal{A}$. Por lo tanto, δ_2 puede ser obtenida combinando estas derivaciones y utilizando la regla r para concluir l_i.

En consecuencia, si $H_1 \cup \mathcal{A} \vdash h$ entonces $H_2 \cup \mathcal{A} \vdash h$, dado que $h \in literales(\mathcal{A})$. Podemos concluir que si H_1 activa a $\mathcal{A}$ entonces H_2 también activa a $\mathcal{A}$. La demostración en el otro sentido se realiza en forma análoga. □

Finalmente, utilizando el lema 4.1 y la proposición 4.1, probaremos la equivalencia entre ambas formulaciones de especificidad.

Teorema 4.1 Sean $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ dos argumentos con respecto a un programa $\mathcal{P}$ y $lit(P)$ el conjunto de literales fijos que es posible derivar a partir de $\mathcal{P}$. Las siguientes condiciones son equivalentes:

1. Para cada $H \subseteq lit(P)$ vale que $H \cup \mathcal{A}_1 \sim h_1$ y $H \not\sim h_1$ implican que $H \cup \mathcal{A}_2 \sim h_2$.
2. Para cada $H \subseteq literales(\mathcal{A}_1)$ vale que $H \cup \mathcal{A}_1 \sim h_1$ y $H \not\sim h_1$ implican que $H \cup \mathcal{A}_2 \sim h_2$.

■

Demostración:
(1) ⇒ (2): En este sentido la implicación es trivial, dado que la segunda cláusula es un caso particular de la primera.
(2) ⇒ (1): Supongamos por el absurdo que no se verifica esta implicación. Entonces debe existir un conjunto H_1 de literales fijos tal que $H_1 \not\subseteq literales(\mathcal{A}_1)$, H_1 activa no trivialmente a $\mathcal{A}_1$ y H_1 no activa a $\mathcal{A}_2$. Aplicando la proposición 4.1 es posible concluir que $H_1 \cap literales(\mathcal{A}_1) \neq \emptyset$ (dado que H_1 activa a $\mathcal{A}_1$). Entonces existe otro conjunto de activación H_2 tal que $H_2 = H_1 \cap literales(\mathcal{A}_1)$ y $H_2 \subset literales(\mathcal{A}_1)$. Como $H_1 \cap literales(\mathcal{A}_1) = H_2 \cap literales(\mathcal{A}_1)$ es posible aplicar el lema 4.1 y concluir que H_2 activa a $\mathcal{A}_1$. Dada nuestra hipótesis inicial, H_2 también activa a $\mathcal{A}_2$ y como $H_2 \subset H_1$, H_1 debe activar a $\mathcal{A}_2$. Esto contradice la suposición inicial bajo la cual

H_1 no activa a $\mathcal{A}_2$ y pone en evidencia una contradicción que proviene de asumir la existencia de H_1. En consecuencia podemos afirmar que (2) $\rightarrow$ (1). □
En base a este teorema hemos demostrado que nuestra versión optimizada de especifidad (definición 4.18) arroja resultados idénticos a los de la definición 4.17.

Como se mencionó anteriormente la especificidad es sólo un ejemplo de los criterios de comparación que es posible utilizar en la PLRO. También es factible codificar criterios especiales para un dominio en particular e incorporarlos al sistema. Esta propiedad favorece la modularidad y la flexibilidad de la PLRO, que le permite adaptarse a diferentes dominios.

La relación de derrota determina qué argumento resulta vencedor ante un conflicto por medio de un criterio de comparación. A continuación se expone la definición de este concepto:

Definición 4.19 (*Derrota*)
Un argumento $\langle \mathcal{A}_1, q_1 \rangle$ *derrota* a un argumento $\langle \mathcal{A}_2, q_2 \rangle$ en un literal q si y sólo si existe un subargumento $\langle \mathcal{A}, q \rangle$ de $\langle \mathcal{A}_2, q_2 \rangle$ tal que $\langle \mathcal{A}_1, q_1 \rangle$ contrargumenta $\langle \mathcal{A}_2, q_2 \rangle$ en q y se cumple alguna de las siguientes condiciones:

1. $\langle \mathcal{A}_1, q_1 \rangle$ prevalece sobre $\langle \mathcal{A}, q \rangle$ de acuerdo con el criterio de comparación (en este caso $\langle \mathcal{A}_1, q_1 \rangle$ es un *derrotador propio* de $\langle \mathcal{A}_2, q_2 \rangle$),

2. $\langle \mathcal{A}_1, q_1 \rangle$ no está relacionado con $\langle \mathcal{A}, q \rangle$ por el criterio de comparación (entonces $\langle \mathcal{A}_1, q_1 \rangle$ es un *derrotador de bloqueo* de $\langle \mathcal{A}_2, q_2 \rangle$).

■

Cabe destacar que el concepto de derrota que distingue dos categorías de derrotadores (propios y de bloqueo) fue definido por primera vez en [Simari et al., 1994]. Posteriormente esta misma definición fue adoptada por la PLR [García y Simari, 2003].

4.2.2. Árboles dialécticos

Dado que los derrotadores son argumentos, pueden a su vez ser derrotados, tal como sucede en la programación en lógica rebatible o en el sistema de Simari y Loui. De esta forma se crea una secuencia de argumentos donde cada elementos derrota a su predecesor. Esta estructura da lugar al concepto de *línea argumentativa*, que definiremos a continuación. Cabe destacar que por medio de las líneas argumentativas la PLRO provee además mecanismos para evitar que se produzcan

situaciones falaces como las consideradas en los ejemplos 3.13 y 3.14. Estos mecanismos están basados en las propuestas desarrolladas en el sistema [Simari et al., 1994] y la programación en lógica rebatible [Simari et al., 1994, García y Simari, 2003].[4] Hemos formulado una caracterización particular para la PLRO que se expone a continuación, basada en la definición formal de línea argumentativa desarrollada en el trabajo de [Chesñevar et al., 2000a], en donde se refinó este concepto brindando una definición versátil y elegante.

Definición 4.20 (*Línea argumentativa*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa y $\langle \mathcal{A}, q \rangle$ un argumento con respecto a $\mathcal{P}$. Una *línea argumentativa* a partir de $\langle \mathcal{A}, q \rangle$, denotada como $\lambda^{\langle \mathcal{A},q \rangle}$ (o simplemente λ), es una secuencia posiblemente infinita de argumentos basados en $\mathcal{P}$

$$\lambda^{\langle \mathcal{A},q \rangle} = [\langle \mathcal{A}_0, q_0 \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle, \ldots]$$

que satisface las siguientes condiciones:

1. Si $\langle \mathcal{A}, q \rangle$ no tiene derrotadores entonces $\lambda^{\langle \mathcal{A},q \rangle} = [\langle \mathcal{A}, q \rangle]$.
2. Si $\langle \mathcal{A}, q \rangle$ posee un derrotador $\langle \mathcal{B}, s \rangle$ en $\mathcal{P}$, entonces $\lambda^{\langle \mathcal{A},q \rangle} = \langle \mathcal{A}, q \rangle \circ \lambda^{\langle \mathcal{B},s \rangle}$.

El operator '$\circ$' denota el agregado de $\langle \mathcal{A}, q \rangle$ como el primer elemento de $\lambda^{\langle \mathcal{B},s \rangle}$. ■

En cada línea argumentativa $\lambda^{\langle \mathcal{A},q \rangle} = [\langle \mathcal{A}_0, q_0 \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle, \ldots]$ el argumento $\langle \mathcal{A}_0, q_0 \rangle$ soporta la consulta q_0 y cada argumento $\langle \mathcal{A}_i, q_i \rangle$ derrota a su predecesor $\langle \mathcal{A}_{i-1}, q_{i-1} \rangle$. Enconces, para $k \geq 0$, $\langle \mathcal{A}_{2k}, q_{2k} \rangle$ es un argumento de soporte para q_0 y $\langle \mathcal{A}_{2k+1}, q_{2k+1} \rangle$ es un argumento de interferencia. Luego, de la misma forma que en la PLR, cada línea se puede dividir en dos conjuntos disjuntos: λ_S de argumentos de soporte y λ_I de argumentos de interferencia.

La clasificación de las posibles falacias del proceso de inferencia coincide con la realizada en la sección 3.2.1. Por lo tanto podemos distinguir básicamente dos categorías de falacias: contradicción entre los argumentos de soporte (respectivamente interferencia) en una determinada línea argumentativa y circularidad en la argumentación. Para solucionar estos problemas se introducen los siguientes conceptos:

Definición 4.21 (*Conjunto contradictorio de argumentos*)
Un conjunto de argumentos $S = \bigcup_{i=1}^{n} \{\langle \mathcal{A}_i, q_i \rangle\}$ es *contradictorio* con respecto a un programa $\mathcal{P} = (\Psi, \Delta)$ si y sólo si el conjunto $\Psi \cup \bigcup_{i=1}^{n} \mathcal{A}_i$ permite derivar literales complementarios. ■

[4] En las sección 3.2.1 se presenta un análisis detallado sobre estos temas.

Definición 4.22 (*Línea de argumentación aceptable*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa y sea $\lambda = [\langle \mathcal{A}_0, q_0\rangle, \langle \mathcal{A}_1, q_1\rangle, \ldots, \langle \mathcal{A}_n, q_n\rangle, \ldots]$ una línea de argumentación con respecto a $\mathcal{P}$ tal que $\lambda' = [\langle \mathcal{A}_0, q_0\rangle, \langle \mathcal{A}_1, q_1\rangle, \ldots, \langle \mathcal{A}_k, q_k\rangle, \ldots]$ es un segmento inicial de λ. La secuencia λ' es una *línea de argumentación aceptable* en $\mathcal{P}$ si y sólo si es el segmento inicial más largo en λ que satisface las siguientes condiciones:

1. Los conjuntos ${\lambda'}_S$ y λ'_I no son contradictorios con respecto a $\mathcal{P}$.

2. No existe un argumento $\langle \mathcal{A}_j, q_j\rangle$ en λ' tal que $\langle \mathcal{A}_j, q_j\rangle$ es un subargumento $\langle \mathcal{A}_i, q_i\rangle$ y $\langle \mathcal{A}_i, q_i\rangle$ aparece previamente en λ' $(i < j)$.

3. No existe una secuencia de argumentos

$$[\langle \mathcal{A}_{i-1}, q_{i-1}\rangle, \langle \mathcal{A}_i, q_i\rangle, \langle \mathcal{A}_{i+1}, q_{i+1}\rangle]$$

en λ' tal que $\langle \mathcal{A}_i, q_i\rangle$, es un derrotador de bloqueo para $\langle \mathcal{A}_{i-1}, q_{i-1}\rangle$ y $\langle \mathcal{A}_{i+1}, q_{i+1}\rangle$ es un derrotador de bloqueo para $\langle \mathcal{A}_i, q_i\rangle$.

■

Las condiciones de esta definición se basan en la propuesta desarrollada por García para evitar falacias en la PLR[García, 2000, García y Simari, 2003]. Por lo tanto, para un análisis detallado de la misma se recomienda consultar la sección 3.6.

En la PLRO, para decidir que argumentos prevalecen ante una disputa es necesario analizar la totalidad de las estructuras de argumento relevantes al literal fijo por el cual se realizó la consulta. En consecuencia dado un argumento $\mathcal{A}$ sustentando la consulta inicial debemos considerar el conjunto de todos los derrotadores de $\mathcal{A}$. Por lo tanto pueden aparece más de una línea argumentativa a partir de $\mathcal{A}$, dando lugar a una estructura denominada *árbol dialéctico* [Simari et al., 1994, García y Simari, 2003].

Es importante mencionar que en las siguiente definiciones se asume que todos los argumentos involucrados se construyen en base a un programa $\mathcal{P}$ dado.

Definición 4.23 (*Árbol dialéctico*)
Sea $\mathcal{A}$ un argumento para un literal q. Un *árbol dialéctico* para $\langle \mathcal{A}, q\rangle$, denotado como $\mathcal{T}_{\langle \mathcal{A}, q\rangle}$, se define de la siguiente manera:

1. La raíz del árbol está etiquetada con $\langle \mathcal{A}, q\rangle$.

2. Sea N un nodo distinto de la raíz etiquetado como $\langle \mathcal{A}_n, q_n \rangle$, sea

$$\lambda = [\langle \mathcal{A}, q \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle]$$

la secuencia de etiquetas en el camino desde la raíz hasta N y sean $\langle \mathcal{B}_1, h_1 \rangle$, $\ldots$, $\langle \mathcal{B}_k, h_k \rangle$ todos los derrotadores para $\langle \mathcal{A}_n, q_n \rangle$. Para cada derrotador $\langle \mathcal{B}_i, h_i \rangle$ $(1 \leq i \leq k)$ tal que la línea de argumentación

$$\lambda' = [\langle \mathcal{A}, q \rangle, \langle \mathcal{A}_1, q_1 \rangle, \ldots, \langle \mathcal{A}_n, q_n \rangle, \langle \mathcal{B}_i, h_i \rangle]$$

es aceptable el nodo N tiene un hijo N_i etiquetado como $\langle \mathcal{B}_i, h_i \rangle$. En caso que no exista un derrotador para $\langle \mathcal{A}_n, q_n \rangle$ o que no exista un $\langle \mathcal{B}_i, h_i \rangle$ tal que λ' sea aceptable N es una hoja.

■

En un árbol dialéctico cada nodo (exceptuando la raíz) representa un derrotador de su padre. Las hojas corresponden a argumentos sin derrotar. Cada camino desde la raíz hasta una hoja representa una línea de argumentación aceptable diferente.

Para decidir si la raíz de un árbol dialéctico está derrotada definiremos el siguiente procedimiento de marcado:

Definición 4.24 (*Marcado de un árbol dialéctico*)
Sea $\langle \mathcal{A}_1, q_1 \rangle$ un argumento y $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$ su árbol dialéctico, entonces:

1. Todas las hojas en $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$ serán marcadas como `nodos U`.

2. Sea $\langle \mathcal{A}_2, q_2 \rangle$ un nodo interno de $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$. Entonces $\langle \mathcal{A}_2, q_2 \rangle$ será marcado como `nodo U` si y sólo si cada nodo hijo de $\langle \mathcal{A}_2, q_2 \rangle$ está marcado como `nodo D`. El nodo $\langle \mathcal{A}_2, q_2 \rangle$ será marcado como un `nodo D` si y sólo si tiene al menos un hijo marcado como `nodo U`.

■

Al finalizar el procedimiento de marcado del árbol dialéctico los argumentos sin derrotar quedan identificados como `nodos U` (del inglés, *undefeated*) y los derrotados como `nodos D` (del inglés, *defeated*).

Es interesante considerar que el espacio de búsqueda que se genera al construir un árbol dialéctico se puede reducir aplicando una estrategia de poda alfa-beta tal

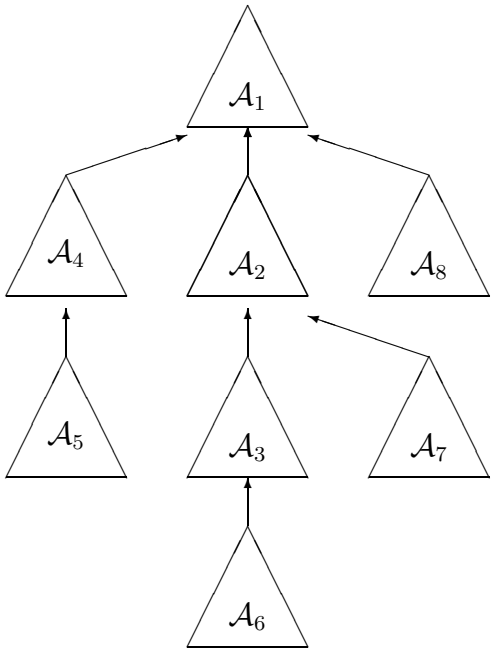


Figura 4.1: Esquema de un arbol de dialéctica

como se realiza en el sistema de [Chesñevar, 1996] y en la programación en lógica rebatible [García y Simari, 2003].

Finalmente, surge la noción de argumentos *garantizados*, que caracteriza las conclusiones del sistema. En la siguiente definición se asume que los argumentos involucrados se construyen en base a un programa $\mathcal{P}$ determinado.

Definición 4.25 (*Argumento garantizado*)
Sea $\mathcal{A}$ un argumento para un literal q y $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ su árbol dialéctico. $\mathcal{A}$ *garantiza* a q si y sólo si la raíz de $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ está marcada como un `nodo U`. ■

Observemos que los conceptos de argumento garantizado y árbol dialéctico se definen en forma idéntica a lo establecido en la PLR[García y Simari, 2003].

Se dice que un literal q está *garantizado* con respecto a un programa $\mathcal{P}$ si es posible obtener un argumento garantizado para q a partir de $\mathcal{P}$. Es importante destacar que los literales garantizados corresponden a las inferencias sancionadas por el sistema.

El siguiente ejemplo ilustra el proceso de inferencia de la PLRO.

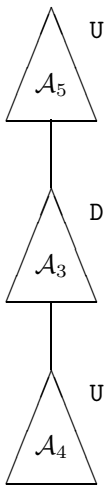


Figura 4.2: Árbol dialéctico correspondiente al ejemplo 4.7.

Ejemplo 4.7 Teniendo en cuenta los argumentos del ejemplo 4.4, se muestra en la figura 4.2 un árbol dialéctico para la consulta `sociable(tom)`. El nodo en la raíz $\langle \mathcal{A}_5, \texttt{sociable(tom)} \rangle$ tiene un único derrotador $\langle \mathcal{A}_3, \sim\texttt{sociable(tom)} \rangle$, el cual a su vez está derrotado solamente por $\langle \mathcal{A}_4, \texttt{sociable(tom)} \rangle$. En consecuencia el árbol dialéctico para $\langle \mathcal{A}_5, \texttt{sociable(tom)} \rangle$ tiene una única rama con tres nodos. De acuerdo a la política de marcado descripta en la definición 4.24 tanto $\langle \mathcal{A}_5, \texttt{sociable(tom)} \rangle$ como $\langle \mathcal{A}_4, \texttt{sociable(tom)} \rangle$ son `nodos U` y $\langle \mathcal{A}_3, \sim\texttt{sociable(tom)} \rangle$ es un `nodo D`. ■

4.3. Conclusiones

En este capítulo presentamos los principales conceptos del sistema de la PLRO, definiendo en forma detallada su lenguaje y mecanismo de inferencia. El trabajo realizado se complementa con las revisiones de los sistemas de [Simari et al., 1994] y la programación en lógica rebatible desarrollados en las secciones 3.2.1 y 3.6 respectivamente. Por esta razón, en la introducción de cada concepto en la PLRO se analiza la relación del nuevo elemento con respecto a las nociones existentes en [Simari et al., 1994] y la PLR. Esto permite situar los aportes de esta tesis en el contexto correspondiente, teniendo en cuenta los trabajos previos relacionados con la investigación llevada a cabo.

A tal fin resumiremos brevemente la evolución cronológica de la línea de investi-

gación fundada por el trabajo de Simari y Loui [Simari y Loui, 1992], en el marco de la cual se desarrollaron posteriormente el sistema de [Simari et al., 1994], la PLR y finalmente la PLRO. Detallaremos para cada caso los conceptos utilizados para definir el formalismo, destacando como evolucionaron las nociones fundacionales dentro de cada sistema.

En primer lugar el lenguaje utilizado por el sistema de Simari y Loui (lógica de primer orden) cambió en la PLR a un lenguaje basado en la programación en lógica, por ser más adecuado para aplicaciones prácticas. Las reglas estrictas también sufrieron cambios. En Simari y Loui se definen como implicaciones materiales que forman parte del lenguaje. En cambio en la PLR se las define como reglas de la programación en lógica. Pragmáticamente el significado de las reglas rebatibles continúa siendo el mismo que el sistema de Simari y Loui. La noción de argumento ha permanecido sin cambios en los diferentes acercamientos. Se mantienen las condiciones de consistencia y minimalidad como conceptos centrales de la misma.

En el sistema de Simari y Loui la relación de contrargumentación solamente resume la existencia de un conflicto entre un par de argumentos. Al incorporar el control de falacias en [Simari et al., 1994], se agregó a esta definición la misión de eliminar los derrotadores recíprocos. Esta versión fue refinada en la PLR donde el control de falacias se resume en la noción de línea argumentativa aceptable, lo que simplificó la relación de contraargumentación. En la PLRO esta definición se puede enunciar de forma aún más sencilla, dado que al eliminar las reglas estrictas, resulta más fácil verificar la existencia de un conflicto.

La relación de derrota decide qué argumento prevalece ante una situación de contraargumentación. En el sistema de Simari y Loui esta definición establecía el uso de especificidad como criterio de comparación entre argumentos. En el sistema desarrollado en [Simari et al., 1994] se modifica esta propuesta, definiendo a la derrota parametrizada con respecto a una criterio de comparación. Por otra parte se distinguen dos clases de derrota: propia y por bloqueo. Tanto en la PLR como en la PLRO esta noción permanece sin cambios.

El criterio de especificidad utilizado en el sistema de Simari y Loui es una adaptación del criterio de Poole presentado en [Poole, 1985]. En un principio estaba ligado al sistema mediante la definición de derrota, pero en [Simari et al., 1994] se convirtió en un elemento accesorio, que podría utilizarse como criterio de comparación, de la misma forma que cualquier otro criterio que resulta adecuado para algún dominio en particular. En la PLR la especificidad se adaptó para el nuevo lenguaje. Luego

en la PLRO se desarrolló una optimización de esta versión, dado que el sistema no contiene reglas estrictas, esto permite calcular especificidad en forma más directa.

El proceso de garantizar argumentos (justificación) se definió en el sistema de Simari y Loui mediante un operador de punto fijo. Esta formulación fue reemplazada en [Simari et al., 1994] por el concepto de árbol dialéctico, que presenta una caracterización del proceso de garantizar a los argumentos (la sección 3.2.1 contiene un análisis de las diferencias entre ambas alternativas). A diferencia de la definición de punto fijo de justificación, los árboles dialécticos presentan un método de obtención de inferencias guiado por las consultas, que solamente tiene en cuenta los argumentos involucrados con la consulta en consideración. Finalmente en el sistema de [Simari et al., 1994] se definen originalmente los conceptos de línea argumentativa y línea argumentativa aceptable para evitar la de falacias en el proceso de argumentación. Estos conceptos se refinan en la PLR donde se desarrolla un nuevo criterio para evitar falacias y en la PLRO se presenta una nueva caracterización de línea argumentativa.

En base a lo expuesto podemos afirmar que el sistema desarrollado posee una sólida base teórica, respaldada por una línea de investigación con más de diez años de evolución. En el próximo capítulo describiremos como es posible emplear a la PLRO para representar el conocimiento de un agente racional.

Capítulo 5

Extensiones a la PLRO

Gran cantidad de aplicaciones que utilizan formalismos de representación de conocimiento y razonamiento necesitan interactuar con ambientes dinámicos. Proveer esta habilidad de forma eficiente y natural es uno de los objetivos de diseño con que fue desarrollado el sistema de la PLRO. En consecuencia es posible introducir mecanismos de percepción para este sistema, tarea que emprenderemos en este capítulo.

En primer lugar presentaremos una reformulación del mecanismo de inferencia de la PLRO por medio del uso de conocimiento precompilado. Esta propuesta consiste en mantener un repositorio de información adicional sobre la estructura de un programa de la PLRO, que luego se utiliza en la obtención de inferencias del sistema. Ya en la década del '80, los sistemas de mantenimiento de verdad[1] resultaron provechosos para optimizar a los *general problem solvers*. De la misma manera, es interesante considerar como los formalismos basados en argumentos pueden ampliar sus capacidades a través del uso de conocimiento precompilado.

A continuación se introducen los mecanismos de percepción que serán utilizados en la PLRO. Con tal fin se detallan en primer lugar los problemas existentes con la percepción en general y se proponen soluciones para nuestro sistema en particular. Luego el mecanismo de inferencia de la PLRO se redefine para que permita actualizar la información existente en el sistema ante los cambios del ambiente.

Posteriormente se propone cómo incorporar un componente de conocimiento precompilado a la PLRO. Para esto se definen formalmente los basamentos teóricos necesarios y se proponen nuevos algoritmos para el proceso de inferencia. Finalmen-

[1] La sección 1.1 contiene información sobre los fundamentos generales de los sistemas de mantenimiento de verdad, así como sobre las aplicaciones desarrolladas en base a los mismos.

te se diseña una implementación eficiente de esta propuesta basada en el uso de algoritmos de reconocimiento de patrones.

5.1. Percepción

Un problema tradicional en la filosofía consiste en explicar cómo es posible para los seres humanos adquirir conocimiento sobre el mundo. Al diseñar un agente de software nos enfrentamos con esta misma pregunta ¿Cómo incorporar mecanismos de percepción en nuestro agente que le permitan acceder a su ambiente y obtener nuevas creencias a partir del mismo? John Pollock [Pollock, 1997] desarrolló un acercamiento a este problema a través de una perspectiva filosófica. Pollock sostiene que un agente que interactúa con un ambiente dinámico no puede poseer desde el momento de su creación toda la información necesaria para su funcionamiento, sino que debe ser capaz obtener nuevas creencias por sí mismo, esto es, *percibir* información de su ambiente.

Al reconocer que la capacidad de percibir es una característica indispensable para la mayoría de los agentes, se hace evidente la necesidad de definir un mecanismo para incorporar nueva información en la base de conocimiento de agente. Este mecanismo se combina con un módulo de observación del agente, que detecta los cambios en el mundo y reporta los literales representando estos cambios. Por medio de estos elementos el agente adquirirá la capacidad de actualizar el modelo del mundo frente a sus cambios.

El módulo de observación depende del dominio de aplicación particular de cada agente. Por lo tanto no es posible definir una estructura general del mismo. La única característica común de estos módulos es que deben ser capaces de representar los cambios en el ambiente mediante un conjunto de literales fijos. De esta forma se respetan las convenciones de la PLRO, formalismo en el cual la información percibida acerca del mundo se modela mediante un conjunto de literales fijos.

Para apreciar como interactúan el módulo de observación junto con el mecanismo de percepción del agente, consideremos la siguiente situación.

Ejemplo 5.1 Supongamos la existencia de un agente denominado `Diego` que modela un jugador de Robo-Cup (una versión del fútbol protagonizada por robots en lugar de los tradicionales jugadores humanos[2]). `Diego` utiliza a la PLRO para modelar su estado epistémico y, como debe ser capaz de percibir los cambios que se

[2]Más información sobre este tema puede encontrarse en `www.robocup.com`.

producen a lo largo del juego, posee además un módulo de observación dependiente de su dominio. A fin de brindar información concisa y directa sobre los cambios del ambiente, este módulo reporta observaciones (esto es, de acuerdo al lenguaje de la PLRO) representando los siguiente hechos:

- Posición de los jugadores del equipo de `Diego` que se encuentran a su alcance (es decir, a los cuales es posible realizar un pase o recibir un pase realizados por ellos).
- Posición de los jugadores del equipo contrario a `Diego` que se encuentran a su alcance (esto es, los jugadores que están en condiciones de interceptar un pase o a los cuales es posible interceptarles un pase).
- Posición de `Diego` dentro de la cancha.
- Posición de la pelota dentro de la cancha.

En base a la información recibida, el mecanismo de percepción se encarga de actualizar la base de conocimiento de `Diego`. ■

En el ejemplo 5.1 describe cómo realizar la primera etapa de la percepción del agente para un ejemplo en particular. Sin embargo, para definir completamente la percepción del agente resta detallar como incorporar las observaciones a la base de conocimiento del mismo (esto es el programa lógico rebatible), tarea que puede diseñarse independientemente del dominio de aplicación. A continuación presentaremos nuestra propuesta para dotar con esta capacidad a la PLRO.

El primer paso para diseñar un mecanismo de percepción en la PLRO consistió en analizar las dificultades inherentes a esta tarea. En este sentido J. Pollock [Pollock, 1987] identificó tres problemas trascendentales que surgen al implementar un mecanismo de percepción. En primer lugar, las observaciones de un individuo no necesariamente reflejan el estado del mundo. En muchas situaciones las apariencias del mundo engañan al observador y pueden producir creencias falsas. El segundo problema se hace evidente al reconocer que la percepción produce un muestreo del estado del mundo a través del tiempo: no es posible que un agente sense continuamente el estado de todos los componentes de su ambiente. En consecuencia el mecanismo de percepción solamente obtiene información de algunos componentes en un instante particular. El agente es responsable de realizar inferencias a partir

de esta información incompleta para lograr obtener un modelo coherente del mundo. Por último, es importante considerar que un agente interactuando en ambientes dinámicos debe actualizar su modelo del mundo al enfrentarse a nuevas percepciones, para lo cual debe ser capaz de razonar sobre la persistencia y el cambio de los hechos de su dominio.

Solucionar estos problemas debe ser un objetivo de diseño al elaborar un sistema con la capacidad de percibir. En nuestra propuesta tendremos en cuenta cada uno de estos inconvenientes. En primer lugar, para eliminar la posibilidad de incorporar percepciones falsas, nos concentraremos en dominios en donde el módulo de observación sólo reporte observaciones seguras. De esta forma las creencias del agente no pueden contaminarse por medio de percepciones erróneas. Asumiremos también que las percepciones son instancias de predicados presentes en la base de conocimiento del agente, para que su incorporación tenga sentido.

El segundo de los problemas mencionados anteriormente se soluciona mediante la interacción entre el módulo de observación del agente y el mecanismo de inferencia de la PLRO. El módulo de observación debe codificarse para que sense el ambiente en intervalos del tiempo adecuados para el dominio de aplicación. Luego el mecanismo de inferencia es responsable de obtener un conjunto de creencias que formen un modelo coherente del mundo a partir de la información incompleta que posee el agente.

Finalmente resta considerar como actualizar el modelo que posee el agente de acuerdo con los cambios del ambiente. Es importante notar que las observaciones no se pueden agregar en forma indiscriminada a la base de conocimiento, dado que esto generaría inconsistencias en el conjunto Ψ de observaciones:

Ejemplo 5.2 Supongamos que la observación `~joven(tom)` se agregara al programa del ejemplo 4.3, asumiendo que `Tom` se hubiera convertido ya en un gato adulto. En ese caso el nuevo conjunto de observaciones resultante, Ψ', no respeta la restricción de consistencia requerida en la definición 4.10. ■

Este tipo de inconvenientes se pueden evitar definiendo el mecanismo de percepción de la PLRO como una *función de actualización* [Katsuno y Mendelzon, 1992], capaz de decidir cómo incorporar las nuevas observaciones en la base de conocimiento y preservar la consistencia del conjunto Ψ.

La *actualización* de una base de conocimiento es la contrapartida dinámica de la revisión de conocimiento. Actualizar una base de conocimiento consiste en poner

al día el conocimiento del agente ante un cambio de su ambiente. En contraste, la revisión de una base de conocimiento se realiza al obtener nueva información sobre un mundo estático. Por lo tanto Katsuno y Mendelzon distinguen dos clases diferentes de modificaciones que pueden realizase en una base de datos.

La primer clase de modificaciones, denominada *actualización*, consiste en modificar la base de datos para incorporar un cambio que ha sucedido en el mundo que esta modela. La mayoría de los cambios en las bases de datos pertenecen a esta categoría. Por ejemplo, incrementamos el valor del salario de un individuo *Juan* en un 5 por ciento, ya que Juan obtuvo un aumento en su sueldo.

La segunda clase de cambios, denominada *revisión*, se utiliza cuando estamos averiguando nueva información sobre un mundo estático. Por ejemplo, al diagnosticar un circuito con fallas podemos desear incorporar en la base de conocimiento los resultados de los tests realizados y en este caso los nuevos resultados se pueden contradecir con los más antiguos.

Considerando esta clasificación se hace evidente que en nuestro caso necesitamos definir un operador de actualización para incorporar los cambios del ambiente a la base de conocimiento. La función de actualización que utilizaremos es simple y efectiva. Para evitar contradicciones, al agregar un nuevo elemento α al conjunto Ψ se elimina cualquier elemento de Ψ que sea contradictorio con α, si es que existe alguno. De acuerdo con este criterio, las percepciones nuevas son siempre preferidas a las más antiguas. La razón detrás de esta decisión, que en un primer momento podría parecer arbitraria, es en realidad muy simple. Dado que el módulo de observación sólo reporta hechos considerados seguros, ambas observaciones en conflicto eran correctas al momento de su percepción. En consecuencia, las única causa posible del conflicto es que haya ocurrido un cambio en el estado del mundo y en este caso la nueva observación debe ser preferida, dado que así obtendremos el modelo más actualizado. Se formalizará esto con la siguiente definición:

Definición 5.1 (*Actualización*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa lógico con observaciones y sea α una observación. La *actualización* de Ψ con respecto a α, denotada como $\Psi * \alpha$ se define como el conjunto $(\Psi - \{\overline{\alpha}\}) \cup \{\alpha\}$ ■

Es importante destacar que al actualizar el conjunto Ψ con respecto a una nueva observación se produce una modificación del conjunto de creencias del agente. De esta forma es posible acomodar el modelo del mundo a los cambios del ambiente. El ejemplo 5.3 muestra esta propiedad.

Ejemplo 5.3 Consideremos el programa del ejemplo 4.3 y supongamos que se actualiza al conjunto Ψ con respecto a ~`joven(tom)`. En este caso la observación `joven(tom)` será eliminada del conjunto Ψ.
Como consecuencia de este cambio varía también el conjunto de creencias obtenidas por medio del programa. Por caso, antes de realizar la actualización el literal `sociable(tom)` formaba parte de las inferencias, pero es descartada al eliminar el literal `joven(tom)`. Por otra parte, el argumento $\langle \mathcal{A}_3, \sim$`sociable(tom)`$\rangle$ (detallado en el ejemplo 4.4) se convierte en un argumento que garantiza a ~`sociable(tom)`.
■

5.2. Conocimiento precompilado

Es posible asociar a cada programa de la PLRO un repositorio de información adicional sobre la estructura del mismo, para luego utilizarlo en la obtención de inferencias del sistema. Este repositorio o módulo de conocimiento precompilado, permite reformular el proceso de inferencia de la PLRO bajo una nueva perspectiva. El objetivo de definir un módulo de conocimiento precompilado en la PLRO apunta a lograr un resultado similar al obtenido mediante el uso de los sistemas de mantenimiento de verdad en los *general problem solvers*, mejorando las capacidades del la PLRO tal como los TMS asisten a esos sistemas.[3] En esta sección se analiza cómo diseñar el componente de conocimiento precompilado asociado a cada programa y presenta los algoritmos necesarios para aprovechar esta estructura en el mecanismo de inferencia de la PLRO.

La idea fundamental para utilizar conocimiento precompilado en la PLRO consiste en asociar a cada programa $\mathcal{P}$ una estructura que contenga la estructura de los argumentos que se pueden construir a partir de las reglas de $\mathcal{P}$ y las relaciones entre los mismos. Esto permitiría agilizar la construcción de los árboles dialécticos.

Existen varias alternativas para realizar este acercamiento. Una posibilidad consiste en en almacenar todos los argumentos que se pueden obtener a partir de las reglas en $\mathcal{P}$ y la relación de derrota entre los mismos. Esta propuesta, aunque simple, posee varias desventajas. En primer lugar, para un programa $\mathcal{P}$ dado puede existir una gran cantidad de argumentos que pueden construirse a partir de $\mathcal{P}$ (aún cuando $\mathcal{P}$ no contenga demasiadas reglas). Almacenar todos estos elementos sería por tanto una tarea costosa. Por otra parte, algunos argumentos se obtienen mediante

[3] En la sección 1.1 se presenta un análisis de los sistemas de mantenimiento de verdad.

diferentes instancias fijas de las mismas reglas rebatibles, por lo cual podríamos simplemente almacenar el conjunto de reglas que se usaron para obtener el argumento.

Es también importante considerar que los argumentos dependen no sólo del conjunto de reglas rebatibles Δ, sino también del conjunto de observaciones Ψ. El conjunto Ψ se actualiza con respecto a los cambios del ambiente y por lo tanto el conjunto de argumentos que se pueden construir a partir de $\mathcal{P}$ se modifica ante estos cambios. Si el conocimiento precompilado del agente estuviera compuesto por los argumentos basados en $\mathcal{P}$ debería actualizarse constantemente conforme cambia el conjunto de observaciones y esto produciría un costo computacional excesivo. En consecuencia, sería deseable disponer de una estructura independiente del conjunto de observaciones.

Para alcanzar estos objetivos definiremos la noción de *argumento potencial*, piedra angular del conocimiento precompilado en la PLRO.

Definición 5.2 (*Instancia de un conjunto de reglas rebatibles*)
Sea A un conjunto de reglas rebatibles. Un conjunto $\mathcal{A}$ compuesto por instancias fijas de reglas rebatibles es una *instancia* de A si y sólo si cada elemento en $\mathcal{A}$ es instancia de alguna de las reglas rebatibles en A. ■

Ejemplo 5.4 El conjunto

$$\mathcal{A} = \{\texttt{anida\_arbol(tweety)} \prec \texttt{vuela(tweety)}, \texttt{vuela(tweety)} \prec \texttt{ave(tweety)}\}$$

es una posible instancia de

$$A = \{\texttt{anida\_arbol(X)} \prec \texttt{vuela(X)}; \texttt{vuela(Y)} \prec \texttt{ave(Y)}\}$$

El conjunto $B = \{\texttt{opera\_fabrica(X,Y)} \prec \texttt{ok(X),ok(Y)}; \texttt{ok(X)} \prec \texttt{paso\_test(X)}\}$ tiene como instancia a $\mathcal{B} = \{\texttt{opera\_fabrica(maqA,maqC)} \prec \texttt{ok(maqA),ok(maqC)}; \texttt{ok(maqA)} \prec \texttt{paso\_test(maqA)}; \texttt{ok(maqC)} \prec \texttt{paso\_test(maqC)}\}$. ■

Definición 5.3 (*Argumento potencial – versión declarativa*)
Sea Δ un conjunto de reglas rebatibles. Un subconjunto A de Δ es un *argumento potencial* para un literal q, denotado como $\langle\langle A, q \rangle\rangle$, si existe un conjunto consistente de literales Φ y una instancia $\mathcal{A}$ de las reglas en A tal que $\mathcal{A}$ es un argumento con respecto a $\mathcal{P} = (\Phi, \Delta)$. ■

Tal como sucede con los argumentos convencionales, diremos que un argumento potencial $\langle\langle A_1, q\rangle\rangle$ es un *subargumento* de $\langle\langle A_2, p\rangle\rangle$ si $A_1 \subseteq A_2$. Los conceptos de cabezas, cuerpos y conjunto soporte (definiciones 4.13 y 4.15 respectivamente) pueden extenderse en forma directa para argumentos potenciales.

En la definición 5.3 el conjunto Φ de observaciones codifica un estado particular del mundo en base al cual es posible construir un argumento utilizando las reglas presentes en A. Los argumentos potenciales resumen toda una clase de argumentos, dado que al utilizar reglas rebatibles no instanciadas, representan al conjunto de argumentos que se obtienen a partir de estas reglas.

Ejemplo 5.5 Si consideramos el programa del ejemplo 4.3 podemos obtener los siguientes argumentos potenciales con respecto a Δ.

- $\langle\langle A_1, \texttt{tiene-cola(X)}\rangle\rangle$,
  donde $\mathcal{A}_1 = \{\texttt{tiene-cola(X)} \prec \texttt{gato(X)}\}$.
- $\langle\langle A_2, \sim\texttt{sociable(X)}\rangle\rangle$,
  donde $\mathcal{A}_2 = \{\sim\texttt{sociable(X)} \prec \texttt{solitario(X)}, \texttt{solitario(X)} \prec \texttt{gato(X)}\}$.
- $\langle\langle A_3, \sim\texttt{tiene-cola(X)}\rangle\rangle$,
  donde $\mathcal{A}_3 = \{\sim\texttt{tiene-cola(X)} \prec \texttt{gato(X), manx(X)}\}$.

■

El primer paso para adquirir el conocimiento precompilado asociado a un programa $\mathcal{P} = (\Psi, \Delta)$ consiste en almacenar todos los argumentos potenciales que es posible obtener a partir de Δ. Desafortunadamente la definición 5.3 no describe un método para realizar esta tarea. Por otra parte, no cualquier subconjunto de las reglas en Δ constituye un argumento potencial: existen algunas restricciones adicionales que se deben cumplir, tal como se evidencia en el siguiente ejemplo:

Ejemplo 5.6 Continuando con el ejemplo 4.3, el conjunto de reglas rebatibles:
$A = \{\texttt{solitario(X)} \prec \texttt{gato(X)},$
$\texttt{tiene-cola(X)} \prec \texttt{gato(X)}\}$
no es un argumento potencial con respecto a Δ. Para cualquier instancia $\mathcal{A}$ de A se verifica que $\mathcal{A}$ no cumple con la restricción de minimalidad de los argumentos presente en la definición 4.12. En efecto, sea c un literal cualquiera. Si utilizando las reglas en A se quiere construir un argumento para `solitario(c)` entonces la regla `tiene-cola(X)` $\prec$ `gato(X)` hace que el argumento no sea minimal.

Por otra parte si se quiere un argumento para `tiene-cola(c)` entonces la regla `solitario(X)` $\prec$ `gato(X)` provocaría el mismo problema. ■

A fin de encontrar en forma eficiente el conjunto de argumentos potenciales de un determinado programa, hemos desarrollado una definición constructiva de este concepto.

Definición 5.4 (*Argumento potencial – versión constructiva*)
Sea Δ un conjunto de reglas rebatibles. Un subconjunto A de Δ es un *argumento potencial* para un literal q, denotado como $\langle\langle A, q\rangle\rangle$, si y sólo si existe una instancia $\mathcal{A}$ de A tal que

1. $literales(\mathcal{A})$ es un conjunto no contradictorio,
2. $\mathcal{S}(\mathcal{A}) \cup \mathcal{A} \vdash\!\!\sim\ q'$ (donde q' es una instancia fija de q) y
3. $\mathcal{A}$ es el mínimo conjunto (con respecto a la inclusión de conjuntos) que satisface las condiciones anteriores.

■

Es importante destacar que ambas formalizaciones son equivalentes, como lo demuestra la siguiente proposición:

Proposición 5.1 La definición 5.4 es equivalente a la definición 5.3. ■

Demostración: (1) $\Rightarrow$ (2): Sea A un conjunto de reglas rebatibles que cumple con las condiciones presentes en la definición 5.4 y sea $\mathcal{A}$ una instancia de A. Para satisfacer las condiciones de existencia de la definición 5.3 basta con considerar a $\mathcal{A}$ como el argumento solicitado y a $\Phi = \mathcal{S}(\mathcal{A})$. Resta probar que $\mathcal{A}$ es un argumento con respecto a (Φ, Δ). Como el conjunto de reglas en $\mathcal{A}$ es consistente, $\mathcal{S}(\mathcal{A})$ debe también ser consistente. En este caso es fácil probar que $\mathcal{A}$ es un argumento para p con respecto a (Φ, Δ). Analicemos las condiciones de la definición 4.12: es posible asegurar que $\mathcal{A} \cup \Phi \vdash\!\!\sim\ q$, dado que $\Phi = \mathcal{G}(\mathcal{A})$ y $\mathcal{G}(\mathcal{A}) \cup \mathcal{A} \vdash\!\!\sim\ q$. A continuación debemos verificar que $\mathcal{A}$ sea consistente con respecto a Φ. Esta restricción también se cumple, ya que $literales(A)$ es un conjunto consistente. Finalmente $\mathcal{A}$ debe cumplir con la condición de minimalidad, lo cual se requiere en forma explícita por medio de la última claúsula de la definición 5.4.
(2) $\Rightarrow$ (1): Se satisface trivialmente. □

A partir de este resultado los argumentos potenciales pueden hallarse examinando uno a uno los subconjuntos de Δ a fin de determinar si cumplen con la definición 5.4. Para cada subconjunto A es necesario hallar una instancia fija $\mathcal{A}$ de A que posea las condiciones adecuadas. Este procedimiento puede realizarse eficientemente de acuerdo al algoritmo 5.1, que utiliza un encadenamiento hacia atrás del conjunto de reglas rebatibles para obtener una instancia de las mismas. Antes de presentar el algoritmo para hallar una instancia definiremos la noción de sustitución [Lloyd, 1987], que será utilizada en este algoritmo. Cabe destacar que en este contexto los conceptos de términos, variables y literales se circunscriben al lenguaje de la PLRO.

Definición 5.5 (*Sustitución*)
Una *sustitución* θ es un conjunto finito de la forma $\{v_1/t_1, \ldots, v_n/t_n\}$ donde cada v_i es una variable, todas las variables son distintas entre sí y para todo i y todo j, v_i no está presente en t_j. Si todos los términos son fijos entonces θ es una *sustitución fija*. Cada elemento v_i/t_i se denomina una ligadura para v_i. ■

En un abuso de notación, denotaremos como $\{v_1, \ldots, v_n\}/\theta$ al resultado de aplicar una sustitución θ a cada uno de los miembros del conjunto $\{v_1, \ldots, v_n\}$. Si r es una regla (estricta o rebatible) denotaremos como r/θ al resultado de aplicar la sustitución θ en el cuerpo y la cabeza de r.

Definición 5.6 (*Instancia de un literal por una sustitución*)
Sea $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ una sustitución y sea l un literal. Entonces la *instancia* de l por θ se obtiene reemplazando simultáneamente cada aparación de la variable v_i en l por el término t_i. ■

Definición 5.7 (*Composición de sustituciones*)
Sean $\theta = \{u_1/s_1, \ldots, u_n/s_m\}$ y $\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$ dos sustituciones. Entonces la *composición* de $\theta\sigma$ de θ y σ es la sustitución que se obtiene del conjunto:

$$\{(u_1/s_1)\sigma, \ldots, (u_n/s_m)\sigma, v_1/t_1, \ldots, v_n/t_n\}$$

eliminando cualquier ligadura $(u_i/s_i)\sigma$ para la cual $u_i = s_i\sigma$ y cualquier ligadura v_j/t_j tal que $v_j \in \{u_1, \ldots, u_n\}$. ■

El algoritmo HallarInstancia construye una sustitución por medio de la cual es posible obtener el argumento $\mathcal{A}$ que es instancia del argumento potencial A que sustenta a la consulta q_1 en consideración. Para esto es necesario considerar el conjunto

Algoritmo 5.1 HallarInstancia

entrada: $\langle\langle A, q\rangle\rangle$, q_1 (donde q_1 es una instancia fija de q),
Ψ (un conjunto de observaciones)

salida: $\langle \mathcal{A}, q_1\rangle$ (tal que $\langle \mathcal{A}, q_1\rangle$ es una instancia de $\langle\langle A, q\rangle\rangle$) si es que existe)

Crear_pila(P)
apilar(q_1,P)
θ:= {}
Repetir mientras P no sea vacía
    meta:=desapilar(P)
    Si existe una regla r en A y una sustitución σ por medio de la cual
    `cabeza(`r`)`σ =meta
    entonces
        nuevo_cuerpo:=aplicar σ al cuerpo de r
        θ:=composición de θ con σ
        apilar(nuevo_cuerpo,P)
    sino r:=desapilar(P)
        Si existe una sustitución σ y una observación α en Ψ tales que $r\sigma = \alpha$
            entonces θ:=composición de θ con σ
            sino no es posible construir instancia
$\mathcal{A}$:=aplicar la sustitución θ a cada regla en A
Devolver($\langle \mathcal{A}, q_1\rangle$)

Figura 5.1: Algoritmo para hallar una instancia de un argumento potencial.

de observaciones Ψ y es por esto que este conjunto forma parte de la entrada del algoritmo. Es importante observar que para que este algoritmo funcione correctamente las reglas rebatibles que conforman al argumento potencial deben estandarizarse de forma tal que no contengan variables en común. Esto es, para cualquier par de reglas r_1, r_2 de A debe cumplirse que la intersección entre las variables de r_1 y r_2 sea el conjunto vacío.

Si bien hemos analizado cómo construir el conocimiento precompilado del agente, todavía no hemos descripto como este componente se puede utilizar en el proceso de inferencia. Consideremos un agente que usa un programa $\mathcal{P} = (\Psi, \Delta)$ para representar su estado epistémico. Para construir el conocimiento precompilado de este agente es necesario almacenar todos los argumentos potenciales que pueden hallarse a partir de las reglas de Δ. Al obtener inferencias a partir del programa $\mathcal{P}$ el agente utiliza las instancias de los argumentos potenciales que constituyen argumentos válidos con respecto a $\mathcal{P}$.

Por medio de este procedimiento es factible evitar el proceso de construcción de los argumentos y sus derrotadores que se realiza al resolver una consulta en la PLRO. Generar los argumentos potenciales del agente es una tarea costosa, pero es posible realizarlo sólo una vez. Como estas estructuras son independientes del conjunto de observaciones no se deben actualizar ante los cambios que se produzcan en el conjunto Ψ. Por lo tanto el uso del conocimiento precompilado no complica la interacción del agente con su ambiente.

Para que la información almacenada sea útil al momento de buscar los derrotadores de un determinado argumento es necesario almacenar la relación de derrota entre los argumentos potenciales. En consecuencia extenderemos las nociones de *contraargumentación* y *derrota*.

Definición 5.8 (*Contraargumentación potencial*)
Un argumento potencial $\langle\langle A_1, q_1 \rangle\rangle$ *contraargumenta* a otro $\langle\langle A_2, q_2 \rangle\rangle$ en un literal q si y sólo si existe un subargumento $\langle\langle A, q \rangle\rangle$ de $\langle\langle A_2, q_2 \rangle\rangle$ (A distinto de vacío) y una sustitución fija θ tal que el conjunto $\{q_1/\theta, q/\theta\}$ es contradictorio. ■

Ejemplo 5.7 En los argumentos potenciales:

- $\langle\langle A, \texttt{solitario(X)} \rangle\rangle$,
  donde $A = \{\texttt{solitario(X)} \prec \texttt{gato(X)}\}$
- $\langle\langle B, \sim\texttt{solitario(Y)} \rangle\rangle$,
  donde $B = \{\sim\texttt{solitario(Y)} \prec \texttt{sociable(Y)}, \texttt{sociable(Y)} \prec \texttt{mascota(Y)}\}$

A contrargumenta a B en $\sim$ `solitario(Y)`, dado que existe una substitución (por ejemplo, $\theta = \{Y/\texttt{tom}, X/\texttt{tom}\}$) tal que $\{\sim$ `solitario(Y)`$/\theta,$ `solitario(X)`$/\theta\}$ es contradictorio. ■

Observación 5.1 El subargumento potencial A que origina el conflicto debe ser distinto de vacío para que sea factible un escenario en el cual una instancia de $\langle\langle A_1, q_1\rangle\rangle$ ataque a una instancia de $\langle\langle A_2, q_2\rangle\rangle$. Si A fuera vacío esto significa que alguna instancia de q, la conclusión de A, es una observación perteneciente al conjunto Ψ y por lo tanto no puede existir un argumento que contradiga a q (un argumento que contradice a q sería contradictorio con Ψ). ■

Para determinar si un argumento potencial derrota a uno de sus contraargumentos es necesario extender el criterio de comparación. En esta presentación hemos optado por redefinir el criterio de especificidad (definición 4.17). En el capítulo anterior se enunció una versión de especificidad para comparar pares de argumentos independientemente del conjunto de observaciones. Esta versión será extendida para comparar argumentos potenciales de la siguiente manera: un argumento potencial $\langle\langle A_1, q_1\rangle\rangle$ es *más específico* que $\langle\langle A_2, q_2\rangle\rangle$ si y sólo si un argumento $\mathcal{A}_1$ que es además una instancia de $\langle\langle A_1, q_1\rangle\rangle$ es más específico que un argumento $\mathcal{A}_2$ instancia de $\langle\langle A_2, q_2\rangle\rangle$ de acuerdo a la definición 4.18. $\langle\langle A_1, q_1\rangle\rangle$ es *estrictamente más específico* que $\langle\langle A_2, q_2\rangle\rangle$ si y sólo si $\mathcal{A}_1$ es estrictamente más específico que $\mathcal{A}_2$ de acuerdo a la definición 4.18.

Ejemplo 5.8 Consideremos los argumentos potenciales:

- $\langle\langle A_1,$ `tiene-cola(X)`$\rangle\rangle$, donde
  $A_1 = \{$`tiene-cola(X)` $\prec$ `gato(X)`$\}$.
- $\langle\langle A_2, \sim$`tiene-cola(X)`$\rangle\rangle$, donde
  $A_2 = \{\sim$`tiene-cola(X)` $\prec$ `gato(X),manx(X)`$\}$

y sus instancias:

- $\langle\mathcal{A}_1,$ `tiene-cola(grace)`$\rangle$, donde
  $\mathcal{A}_1 = \{$`tiene-cola(grace)` $\prec$ `gato(grace)`$\}$.
- $\langle\mathcal{A}_2, \sim$`tiene-cola(grace)`$\rangle$, donde
  $\mathcal{A}_2 = \{\sim$`tiene-cola(grace)` $\prec$ `gato(grace),manx(grace)`$\}$

Aquí el argumento $\langle \mathcal{A}_2, \sim$`tiene-cola(tom)`$\rangle$ es estrictamente más específico que el argumento $\langle \mathcal{A}_1,$ `tiene-cola(tom)`$\rangle$. Luego $\langle\langle A_2,\sim$`tiene-cola(X)`$\rangle\rangle$ es más específico que $\langle\langle A_1,$ `tiene-cola(X)`$\rangle\rangle$. ■

Para que esta formulación tenga sentido, el criterio de preferencia debe depender únicamente de las reglas rebatibles utilizadas, sin importar las instancias particulares de las mismas. En la definición anterior podemos afirmar que si al argumento $\mathcal{A}_1$ es más específico que $\mathcal{A}_2$ esta mismo comportamiento se extrapola a todos los argumentos compuestos por las mismas reglas que $\mathcal{A}_1$ y $\mathcal{A}_2$. Esto se demuestra formalmente en la siguiente proposición:

Proposición 5.2 Sean $\mathcal{A}$, $\mathcal{B}$ dos argumentos tal que $\mathcal{A}$ es más específico que $\mathcal{B}$. Para todo par de argumentos $\mathcal{A}_1$, $\mathcal{B}_1$ tal que $\mathcal{A}_1$ está compuesto por las mismas reglas rebatibles que $\mathcal{A}$ y $\mathcal{B}_1$ está compuesto por las mismas reglas rebatibles que $\mathcal{B}$ se verifica que $\mathcal{A}_1$ es más específico que $\mathcal{B}_1$. ■

Demostración: Supongamos por el absurdo que existe un par de argumentos $\mathcal{A}_2$, $\mathcal{B}_2$ tal que $\mathcal{A}_2$ está compuesto por las mismas reglas que $\mathcal{A}$, $\mathcal{B}_2$ está compuesto por las mismas reglas que $\mathcal{B}$ y $\mathcal{A}_2$ no es más específico que $\mathcal{B}_2$. En este caso podemos distinguir dos escenarios:

1. $\mathcal{B}_2$ es estrictamente más específico que $\mathcal{A}_2$. En este caso existe un conjunto de literales H_1 tal que $H_1 \cup \mathcal{A}_2 \sim q$ y $H_1 \cup \mathcal{B}_2 \not\sim q$ y no existe un conjunto de literales H_2 tal que $H_2 \cup \mathcal{A}_2 \sim q$ y $H_2 \cup \mathcal{B}_2 \not\sim q$. Sin embargo, como $\mathcal{A}$ es más específico que $\mathcal{B}$, existe un conjunto H_3 tal que $H_3 \cup A \sim q$ y $H_3 \cup B \not\sim q$. Usando otras instancias fijas de los literales en H_3 es posible construir un conjunto H'_3 tal que $H'_3 \cup \mathcal{A}_2 \sim q$ y $H'_3 \cup \mathcal{B}_2 \not\sim q$. Por lo tanto $\mathcal{B}_2$ no es estrictamente más específico que $\mathcal{A}_2$.

2. $\mathcal{B}_2$ y $\mathcal{A}_2$ son incomparables usando especificidad. Este caso tampoco es posible; usando el mismo criterio que en el escenario anterior es posible demostrar que existe un conjunto H'_3 tal que $H'_3 \cup \mathcal{A}_2 \sim q$ y $H'_3 \cup \mathcal{B}_2 \not\sim q$. Luego $\mathcal{B}_2$ y $\mathcal{A}_2$ no son incomparables.

Esto constituye un absurdo que provino de suponer que $\mathcal{A}_2$ no es más específico que $\mathcal{B}_2$. □

La noción de derrota entre pares de argumentos potenciales puede enunciarse en forma análoga a la definición 4.19, es decir, parametrizandola con respecto al criterio

de comparación. La única restricción adicional que debe cumplir este criterio es la independencia del conjunto de observaciones Ψ. En caso contrario la relación de derrota se vería modificada al actualizar las observaciones del agente.

En base a los conceptos presentados hasta el momento es posible introducir la noción de *base de dialéctica*. Esta estructura resume el conocimiento precompilado asociado a un determinado programa.

Definición 5.9 (*Base de dialéctica*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa en la PLRO. La *base dialéctica* de $\mathcal{P}$, denotada como $\mathcal{DB}_\Delta$, consiste en una tupla $(ArgPot(\Delta), D_p, D_b)$ tal que:

1. $ArgPot(\Delta)$ es el conjunto compuesto por todos los argumentos potenciales que es posible construir a partir de Δ.

2. D_p y D_b son dos relaciones sobre los elementos de $ArgPot(\Delta)$, tal que un par de argumentos potenciales $(\langle\langle A_1, q_1\rangle\rangle, \langle\langle A_2, q_2\rangle\rangle)$ pertenece a D_p (o D_b respectivamente) si y sólo si $\langle\langle A_2, q_2\rangle\rangle$ es un derrotador propio (respectivamente un derrotador de bloqueo) de $\langle\langle A_1, q_1\rangle\rangle$.

■

Ejemplo 5.9 El siguiente programa codifica un conocido ejemplo de la literatura sobre razonamiento no monótono:

```
ave(tweety).
ave(pengo).
pingüino(tweety).
vuela(X) ≺ ave(X).
~vuela(X) ≺ ave(X), pingüino(X).
```

La base de dialéctica de este programa está compuesta por los argumentos potenciales que se detallan a continuación:

- $\langle\langle A_1, \texttt{vuela(X)}\rangle\rangle$,
  donde $A_1 = \{\texttt{vuela(X)} \prec \texttt{ave(X)}\}$.

- $\langle\langle A_2, \texttt{\~{}vuela(X)}\rangle\rangle$,
  donde $A_2 = \{\texttt{\~{}vuela(X)} \prec \texttt{ave(X),pingüino(X)}\}$.

y las relaciones $D_p = \{(A_2, A_1)\}$ and $D_b = \{\}$.

■

Ejemplo 5.10 Consideremos el siguiente programa:

```
desemp_insuficiente(juan).
enfermo(juan).
desemp_insuficiente(pedro).
altos_honorarios(antonia).
candidato(antonia).
buenas_ref(antonia).
contratar(X) ≺ bajos_honorarios(X), candidato(X).
~contratar(X) ≺ altos_honorarios(X), candidato(X).
contratar(X) ≺ altos_honorarios(X), candidato(X), buenas_ref(X).
suspender(X) ≺ ~responsable(X).
~suspender(X) ≺ responsable(X).
~responsable(X) ≺ desemp_insuficiente(X).
responsable(X) ≺ desemp_aceptable(X).
responsable(X) ≺ desemp_insuficiente(X), enfermo(X).
```

Las observaciones establecen que `juan` posee un desempeño insuficiente en su trabajo y que este empleado está enfermo. Además `pedro` también posee un desempeño insuficiente y `antonia` es una candidata para ocupar un puesto vacante en la empresa que posee honorarios altos y muy buenas referencias. Las reglas codifican las siguientes aserciones rebatibles: hay que contratar a un candidato con bajos honorarios, no hay que contratar a un candidato con honorarios elevados, conviene contratar a un candidato con honorarios elevados que posea buenas referencias, si un empleado no es responsable debemos suspenderlo, si un empleado es responsable no hay que suspenderlo, un empleado es responsable si se desempeña en forma aceptable en su trabajo, un empleado no es responsable si tiene un desempeño insuficiente (a menos que éste haya estado enfermo).
La base de dialéctica (que depende solamente de las reglas del programa) está compuesta por los siguientes argumentos potenciales:

- $\langle\langle A_1, \texttt{contratar(X)} \rangle\rangle$,
  donde $A_1 = \{\texttt{contratar(X)} \prec \texttt{bajos\_honorarios(X), candidato(X)}\}$.

- $\langle\!\langle A_2, \sim$`contratar(X)`$\rangle\!\rangle$,
  donde $A_2 = \{\sim$`contratar(X)` $\prec$ `altos_honorarios(X), candidato(X)`$\}$.
- $\langle\!\langle A_3,$ `contratar(X)`$\rangle\!\rangle$,
  donde $A_3 = \{$`contratar(X)` $\prec$ `altos_honorarios(X), candidato(X), buenas_ref(X)`$\}$.
- $\langle\!\langle B_1,$ `suspender(X)`$\rangle\!\rangle$,
  donde $B_1 = \{$`suspender(X)` $\prec \sim$`responsable(X)`$\}$.
- $\langle\!\langle B_2,$ `suspender(X)`$\rangle\!\rangle$,
  donde $B_2 = \{$`suspender(X)` $\prec \sim$`responsable(X)`;
  $\sim$`responsable(X)` $\prec$ `desemp_insuficiente(X)`$\}$.
- $\langle\!\langle B_3, \sim$`suspender(X)`$\rangle\!\rangle$,
  donde $B_3 = \{\sim$`suspender(X)` $\prec$ `responsable(X)`$\}$.
- $\langle\!\langle B_4, \sim$`suspender(X)`$\rangle\!\rangle$,
  donde $B_4 = \{\sim$`suspender(X)` $\prec$ `responsable(X)`;
  `responsable(X)` $\prec$ `desemp_aceptable(X)`$\}$.
- $\langle\!\langle B_5, \sim$`suspender(X)`$\rangle\!\rangle$,
  donde $B_5 = \{\sim$`suspender(X)` $\prec$ `responsable(X)`;
  `responsable(X)` $\prec$ `desemp_insuficiente(X), enfermo(X)`$\}$.
- $\langle\!\langle C_1,$ `responsable(X)`$\rangle\!\rangle$,
  donde $C_1 = \{$`responsable(X)` $\prec$ `desemp_aceptable(X)`$\}$.
- $\langle\!\langle C_2, \sim$`responsable(X)`$\rangle\!\rangle$,
  donde $C_2 = \{\sim$`responsable(X)` $\prec$ `desemp_insuficiente(X)`$\}$.
- $\langle\!\langle C_3,$ `responsable(X)`$\rangle\!\rangle$,
  donde $C_3 = \{$`responsable(X)` $\prec$ `desemp_insuficiente(X), enfermo(X)`$\}$.

y las relaciones de derrota:

- $D_p = \{(A_3, A_2), (C_3, C_2), (C_3, B_2)\}$
- $D_b = \{(A_1, A_2), (A_2, A_1), (C_1, C_2), (C_2, C_1), (C_1, B_2), (C_2, B_4), (B_1, B_3), (B_1, B_4), (B_3, B_1), (B_4, B_1), (B_1, B_5), (B_5, B_1), (B_2, B_5), (B_5, B_2), (B_2, B_3), (B_3, B_2)\}$.

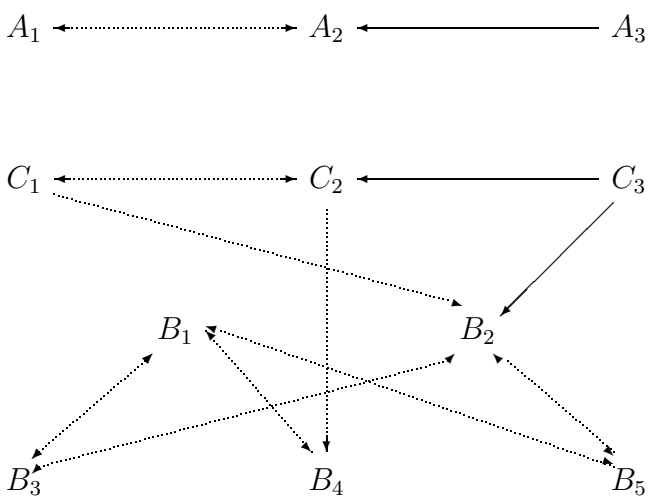


Figura 5.2: Base de dialéctica del ejemplo 5.10.

también se indican en la figura 5.2 donde A_1 derrota en forma propia a A_2 se indica con una flecha desde A_1 hasta A_2 y la derrota por bloqueo se denota con flechas punteadas.
Observemos que en caso de consultar al sistema con `contratar(antonia)` ésta se puede resolver por medio del argumento potencial A_3 (llegando a un árbol de dialéctica compuesto solamente por un argumento, que no posee derrotadores) o utilizando el argumento potencial A_1 que tiene como derrotador por bloqueo a una instancia de A_2 el cual a su vez es derrotado por una instancia de A_3. La base de dialéctica, que constituye un grafo, resume ambos árboles de dialéctica. ■

Para obtener la base dialéctica de un programa $\mathcal{P} = (\Psi, \Delta)$ se genera en primer lugar el conjunto $ArgPot(\Delta)$, incluyendo en el mismo cada subconjunto de las reglas Δ que satisface las restricciones de la definición 5.4. El algoritmo 5.1 permite realizar esta tarea en forma eficiente. A continuación se computa la relación de derrota entre los elementos de $ArgPot(\Delta)$ por medio del criterio de comparación. Cada par de argumentos potenciales $(\langle\langle A_1, q_1\rangle\rangle, \langle\langle A_2, q_2\rangle\rangle)$ se analiza a fin de determinar si $\langle\langle A_2, q_2\rangle\rangle$ es un derrotador (propio o de bloqueo) de $\langle\langle A_1, q_1\rangle\rangle$. En tal caso el par encontrado se almacena en la relación D_p o D_b dependiendo del tipo del derrotador.

Cuando el proceso de inferencia de la PLRO debe resolver una consulta q con respecto a un programa $\mathcal{P} = (\Psi, \Delta)$ el procedimiento original comienza por construir un argumento $\mathcal{A}_1$ que sustente a q a partir de las reglas en Δ. Luego se generan los

derrotadores de $\mathcal{A}_1$ y se analiza si éstos permanecen sin derrotar a partir del análisis dialéctico de la totalidad del conocimiento. Esto involucra realizar la misma tarea que se lleva a cabo en $\mathcal{A}_1$ sobre cada uno de sus derrotadores. Luego se decide el estado de $\mathcal{A}_1$ a partir de la condición de sus derrotadores (es decir, si está marcado como un `nodo D` o un `nodo U`, ver definición 4.24).

El uso de la base de dialéctica agiliza el proceso de inferencia, tal como se describe en el algoritmo 5.2. Los argumentos potenciales almacenados en $\mathcal{DB}_\Delta$ simplifican la construcción del argumento para respaldar la consulta q. La búsqueda de los derrotadores también se beneficia mediante las relaciones D_p y D_b definidas a tal fin.

A continuación describiremos detalladamente el algoritmo que define el nuevo proceso de inferencia. En primer lugar el sistema selecciona algún argumento potencial $\langle\langle A, p\rangle\rangle$ tal que la consulta q sea una instancia de p. A partir de las reglas en A intenta conformar todas las instancias de A que son argumentos válidos con respecto a (Ψ, Δ) (esto se puede implementar mediante la función HallarInstancia). Si no existe una instancia en esas condiciones continúa examinando el resto de los argumentos potenciales. En caso contrario se construye un argumento $\mathcal{A}$ (instancia de A) que posteriormente se analiza para determinar si garantiza a q. En primer lugar se verifica que cumpla con las condiciones de consistencia y minimalidad con respecto al programa $\mathcal{P}$ mediante la función Aceptable. En caso afirmativo se utilizan las relaciones D_p y D_b a fin de encontrar los derrotadores de $\mathcal{A}$. Cada derrotador potencial $\langle\langle B, r\rangle\rangle$ de $\langle\langle A, p\rangle\rangle$ es examinado para averiguar si existe una instancia $\mathcal{B}$ de B que sea además un derrotador para $\mathcal{A}$. Esto se lleva a cabo mediante la función Aceptable. Si $\mathcal{B}$ es un argumento, se invoca a la función Estado sobre $\mathcal{B}$ (algoritmo 5.3), que realiza el análisis dialéctico sobre este derrotador y determina si $\mathcal{B}$ es un `nodo U` o un `nodo D`. Finalmente se computa el estado de $\mathcal{A}$ en base al estado de sus derrotadores. Observemos que el algoritmo termina al encontrar el primer argumento que garantiza a q, ya que en principio no es necesario obtener todos. En caso que se desee averiguar todas las formas posibles de garantizar a q se puede modificar en forma sencilla este algoritmo (en vez de finalizar al encontrar la primer solución, guarda las soluciones en un conjunto y continúa su ejecución).

Resta analizar el algoritmo Estado. Éste recibe un argumento $\mathcal{A}$, un programa $\mathcal{P}$ tal que $\mathcal{A}$ está basado en $\mathcal{P}$ y los argumentos de interferencia,`IL`, y soporte, `SL`, presentes en la línea argumentativa desde la raíz hasta $\mathcal{A}$. En base a estos elementos el algoritmo Estado realiza una tarea similar a la del algoritmo 5.2, analizando

Algoritmo 5.2 Proceso de inferencia
entrada: $\mathcal{P} = (\Psi, \Delta)$, q
salida: $\langle \mathcal{A}, q \rangle$ (un argumento que garantiza a q, si existe alguno)
Para cada instancia $\mathcal{A}$ de algún $\langle\langle A, p \rangle\rangle$ en $ArgPot(\Delta)$ tal que Aceptable($\mathcal{A}$,$\mathcal{P}$)
    estado := `nodoU`
    Para cada $\langle\langle B, r \rangle\rangle$ en $ArgPot(\Delta)$ tal que $(\langle\langle B, r \rangle\rangle, \langle\langle A, p \rangle\rangle) \in D_p$ o $(\langle\langle B, r \rangle\rangle, \langle\langle A, p \rangle\rangle) \in D_b$
        Para cada instancia $\mathcal{B}$ de B tal que $\mathcal{B}$ derrota a $\mathcal{A}$ y Aceptable($\mathcal{B}$,$\mathcal{P}$)
            Si Estado($\mathcal{B}$, $\mathcal{P}$, $\emptyset$, $\{\mathcal{A}\}$) = `nodoU`
                entonces estado := `nodoD`
    Si estado = `nodoU`
        entonces
            Devolver($\langle \mathcal{A}, q \rangle$)
            finalizar

Figura 5.3: Algoritmo del nuevo proceso de inferencia en la PLRO.

los derrotadores de $\mathcal{A}$ para determinar si constituyen `nodos U` o `nodos D`. La única diferencia es que el algoritmo Estado verifica además que los derrotadores cumplan con las condiciones establecidas en la definición 4.22 a fin de que las líneas argumentativas generadas sean válidas. Esto se realiza a través de la función Válido.

Finalmente es importante destacar que para realizar la implementación de los algoritmos presentados en esta sección es preponderante escoger un conjunto apropiado de estructuras de datos que permitan crear y utilizar la base de dialéctica en forma eficiente. En la sección siguiente definiremos una posible implementación del sistema que considera estos objetivos.

5.3. Consideraciones de implementación

En la sección anterior se desarrollaron un conjunto de algoritmos que definen al proceso de inferencia de la PLRO. A continuación diseñaremos una implementación para este proceso basada en *algoritmos de reconocimiento de patrones* [Cohen, 1977, Forgy, 1982].

Los problemas de reconocimiento de patrones se caracterizan por comparar una

Algoritmo 5.3 Estado
entrada: $\mathcal{A}_1$, $\mathcal{P}$, IL, SL
salida: estado
estadoActual := nodoU
Para cada par $(\langle\langle A_1, q\rangle\rangle, \langle\langle A_2, r\rangle\rangle) \in D_p \cup D_b$ tal que $\mathcal{A}_1$ es una instancia de A_1
    Para cada instancia $\mathcal{A}_2$ de A_2 tal que Aceptable($\mathcal{A}_2$,$\mathcal{P}$) y Válido($\mathcal{A}_2$,IL,SL)
        Si $\mathcal{A}_1$ es un arg. de soporte y Estado($\mathcal{A}_2$,$\mathcal{P}$,IL,SL $\cup \{\mathcal{A}_1\}$)=nodoU
            entonces estadoActual:=nodoD
        Si $\mathcal{A}_1$ es un arg. de interferencia y Estado($\mathcal{A}_2$,$\mathcal{P}$,IL $\cup \{\mathcal{A}_1\}$,SL)=nodoU
            entonces estadoActual:=nodoD
Devolver(estadoActual)

Figura 5.4: Algoritmo para determinar el estado de un argumento.

colección de objetos con un conjunto de patrones a fin de determinar todas las correspondencias existentes entre los mismos. Esta clase de algoritmos ha sido utilizada en gran cantidad de programas de inteligencia artificial, como por ejemplo los intérpretes de los sistemas de producciones. En ese caso los algoritmos de reconocimiento de patrones se utilizan para decidir cuáles producciones pueden dispararse, dado que están satisfechas sus condiciones. A continuación describiremos cómo es posible adaptar estas técnicas para desarrollar una implementación eficiente de los algoritmos presentados en la sección anterior.

Un sistema de producciones consiste en una colección de sentencias denominadas *producciones*. Las producciones controlan una base de datos global denominada *memoria de trabajo* y están formadas por dos elementos: el componente izquierdo (LHS) y el componente derecho (RHS).[4] El componente izquierdo está compuesto por una secuencia de patrones, esto es, una secuencia de descripciones parciales de la memoria de trabajo. Cuando un patrón Pa describe a un elemento E se dice que Pa *coincide* con E. El componente derecho de una producción consiste en una secuencia de *acciones*, que pueden modificar el contenido de la memoria de trabajo. El intérprete evalúa las condiciones en el componente izquierdo de las producciones a fin de determinar cuál de estos está satisfecho y permite disparar la producción, esto es, ejecutar las acciones descriptas en el componente derecho.

[4]Del inglés *left hand side* y *right hand side* respectivamente.

El acercamiento tradicional para implementar los sistemas de producciones combina un proceso denominado *indexado* con la *interpretación directa* de los componentes izquierdos. La idea clave en este proceso de indexado consiste en extraer una o más características de los elementos de la memoria de trabajo y utilizarlas para implementar una tabla de acceso directo al conjunto de producciones. Por medio de esta técnica es posible acceder, dado un estado en particular de la memoria de trabajo, a un grupo de producciones cuyos componentes izquierdos pueden estar satisfechos. En el paso de interpretación directa el intérprete examina cada uno de los componentes izquierdos de las producciones candidatas para decidir efectivamente si están o no satisfechos.

La implementación de las bases de dialéctica pueden también beneficiarse de esta técnica si las estructuras de datos necesarias se representan en forma apropiada. En primer lugar definiremos el conjunto de estructuras de datos necesarias para esta tarea. A continuación detallaremos como adaptar los algoritmos de la sección 5.2 para aprovechar tal representación.

En primer lugar definiremos los objetivos de diseño de nuestra implementación, esto es, que operaciones deseamos optimizar. Considerando las restricciones del mecanismo de inferencia de la PLRO se identifican las siguientes metas:

- Acceder en forma eficiente al conjunto de argumentos potenciales que respaldan a una determinada conclusión. De esta forma es posible encontrar los argumentos que sustentan a una determinada consulta.
- Recuperar en forma directa a los derrotadores potenciales de un determinado argumento potencial $\langle\langle A, q\rangle\rangle$. Esto permite acelerar la construcción del árbol dialéctico.
- Agilizar los chequeos de consistencia y minimalidad de los argumentos con respecto al conjunto actual de observaciones.

Teniendo en cuenta estos objetivos se diseñó la estructura de datos para representar cada argumento potencial, que está formada por el siguiente conjunto de atributos:

- Conclusión: el literal q que consiste en la única conclusión del argumento.
- Reglas: las reglas de Δ que conforman al argumento potencial $\langle\langle A, q\rangle\rangle$, denotadas mediante $reglas(A)$. Este operador se puede aplicar también a argumentos propiamente dichos.

- Literales de consistencia: este conjunto, denotado como $\texttt{LC}(A)$, está compuesto por el complemento de cada literal en A. Este operador se puede aplicar también a argumentos tradicionales.
- Literales de minimalidad: este conjunto, denotado como $\texttt{LM}(A)$, se define como $cabezas(A) - cuerpos(A)$. Este operador se puede aplicar también a argumentos tradicionales.
- Derrotadores de bloqueo: enlaces a los derrotadores de bloqueo para $\langle\langle A, q\rangle\rangle$.
- Derrotadores propios: enlaces a los derrotadores propios de $\langle\langle A, q\rangle\rangle$.

Cada uno de estos elementos posee una función tendiente a agilizar el proceso de inferencia de la PLRO. La conclusión permite localizar en forma expeditiva los argumentos potenciales para una tesis determinada. A partir del algoritmo 5.1 detallado en la sección anterior se construye una instancia $\langle \mathcal{A}_1, q_1\rangle$ de $\langle\langle A, q\rangle\rangle$, que es un argumento con respecto al conjunto actual de observaciones Ψ. En base a la instanciación $\langle \mathcal{A}_1, q_1\rangle$ también se instancian los literales de consistencia y minimalidad del argumento potencial, obteniendo los conjuntos $\texttt{LC}(\mathcal{A}_1)$ y $\texttt{LM}(\mathcal{A}_1)$. Luego se realizar esta tarea se puede verificar la consistencia y minimalidad de $\langle \mathcal{A}_1, q_1\rangle$ mediante las siguientes proposiciones:

Proposición 5.3 Sea $\langle \mathcal{A}_1, q_1\rangle$ una instancia de un argumento potencial $\langle\langle A, q\rangle\rangle$ obtenida a partir del algoritmo 5.1. Sean $\texttt{LC}(\mathcal{A}_1)$ y $\texttt{LM}(\mathcal{A}_1)$ los literales de consistencia y minimalidad de $\langle\langle A, q\rangle\rangle$ instanciados de la misma forma que al hallar la instancia $\langle \mathcal{A}_1, q_1\rangle$. Entonces $\langle \mathcal{A}_1, q_1\rangle$ es consistente con respecto al conjunto de observaciones Ψ si y sólo si la intersección de Ψ con $\texttt{LC}(\mathcal{A}_1)$ resulta en el conjunto vacío. ■

Proposición 5.4 Sea $\langle \mathcal{A}_1, q_1\rangle$ una instancia de un argumento potencial $\langle\langle A, q\rangle\rangle$ obtenida a partir del algoritmo 5.1. Sean $\texttt{LC}(\mathcal{A}_1)$ y $\texttt{LM}(\mathcal{A}_1)$ los literales de consistencia y minimalidad de $\langle\langle A, q\rangle\rangle$ instanciados de la misma forma que al hallar la instancia $\langle \mathcal{A}_1, q_1\rangle$. Entonces $\langle \mathcal{A}_1, q_1\rangle$ es minimal con respecto a Ψ si y sólo si la intersección de Ψ con $\texttt{LM}(\mathcal{A}_1)$ da como resultado el conjunto vacío. ■

La demostración de ambas proposiciones surge trivialmente a partir de la definición de LM y LC.

Los argumentos potenciales son los bloques constructores de la base de dialéctica, que se modela como una colección de estas estructuras (de acuerdo con la definición

5.9). A continuación detallaremos como utilizar a la representación definida en esta sección por medio de un conjunto de algoritmos adecuados.

La base de dialéctica $\mathcal{DB}_\Delta$ asociada a un programa lógico rebatible se representa por medio de una colección de las estructuras de datos descriptas en la sección anterior. Cada una de estas estructuras representa un argumento potencial, y los enlaces entre las mismas resumen la relación de derrota. Esta colección de argumentos potenciales se inicializa al construir la base de dialéctica.

Utilizando las estructuras de datos propuestas, podemos definir que una instancia $\mathcal{A}$ de un argumento potencial A en una base de dialéctica $\mathcal{DB}_\Delta$ coincide con el conjunto de observaciones actual Ψ (usando la terminología de los algoritmos de reconocimiento de patrones), siempre que $\mathcal{S}(\mathcal{A}) \in \Psi$, $\Psi \cap \texttt{LC}(\mathcal{A}) = \emptyset$, y $\Psi \cap \texttt{LM}(\mathcal{A}) = \emptyset$.

Es importante destacar que para poder recuperar fácilmente el conjunto de argumentos potenciales que sustentan a un determinado literal la base de dialéctica debe ser indexada con respecto a las conclusiones de los argumentos potenciales presentes en la misma. Luego, en base a esta información, es posible determinar qué instancias de estos argumentos potenciales sustentan a la consulta en consideración. Las instancias pueden hallarse de acuerdo al algoritmo HallarInstancia. De esta forma la condición $\mathcal{S}(\mathcal{A}) \in \Psi$ siempre se cumple, dado que así lo asegura este algoritmo.

El algoritmo `Aceptable` se implementa en forma directa verificando las condiciones expuestas anteriormente ($\Psi \cap \texttt{LC}(\mathcal{A}) = \emptyset$, y $\Psi \cap \texttt{LM}(\mathcal{A}) = \emptyset$). De esta forma, las costosas verificaciones de la consistencia y minimalidad de un argumento se reemplazan por el cálculo de simples intersecciones entre conjuntos de literales.

Utilizando la estructura que hemos propuesto, la construcción del árbol dialéctico para una consulta q se reduce a combinar los mecanismos de indexado e interpretación directa tal como sucede en los algoritmos de reconocimiento de patrones, como se muestra en el algoritmo 5.4. Notemos que este algoritmo constituye un refinamiento del algoritmo 5.2 que describe en forma independiente de la implementación el proceso de inferencia. En primer lugar usando a q como índice se obtiene un conjunto de argumentos potenciales por medio del procedimiento HallarCandidatos. Estos candidatos podrían convertirse en argumentos sustentando a q, por lo cual son analizados uno a uno hallando para cada candidato A una instancia $\mathcal{A}$ de A que sustenta a q, si es esto posible. Esta tarea puede concretarse usando el procedimiento HallarInstancia, detallado en el algoritmo 5.1. Si $\mathcal{A}$ es válido con respecto al conjunto Ψ actual (este chequeo se realiza mediante la función Coincide) se agrega como

Algoritmo 5.4 Proceso de inferencia (refinado)
entrada: $\mathcal{P} = (\Psi, \Delta)$, q
salida: $\langle \mathcal{A}, q \rangle$ (un argumento que garantiza a q, si existe alguno)
HallarCandidatos(q,ArgsPotenciales)
Para cada instancia $\mathcal{A}$ de algún $\langle\langle A, p \rangle\rangle \in$ ArgsPotenciales tal que coincide($\mathcal{A}$,$\mathcal{P}$)
    ObtenerCjtos($\mathcal{A}$,LC($\mathcal{A}$),LM($\mathcal{A}$))
    Si LC($\mathcal{A}$) $\cap \Psi = \emptyset$ y LM($\mathcal{A}$) $\cap \Psi = \emptyset$
    entonces
        estado:=nodoU
        PosiblesDerrotadores($\langle\langle A, p \rangle\rangle$, CjtoDerrotadores)
        Para cada $\mathcal{B}$ en CjtoDerrotadores
            ObtenerCjtos($\mathcal{B}$,LC($\mathcal{B}$),LM($\mathcal{B}$))
            Si Válido(B,$\emptyset$,{LC($\mathcal{A}$)},{$\mathcal{A}$})
                entonces ObtenerEstado($\mathcal{B}$,$\mathcal{P}$,$\emptyset$,{LC($\mathcal{A}$)},{$\mathcal{A}$},estadoB)
            Si estadoB=nodoU
                entonces estado:=nodoD
    Si estado=nodoU
        entonces
            devolver($\langle \mathcal{A}, q \rangle$)
            finalizar

Figura 5.5: Refinamiento del algoritmo 5.2 utilizando técnicas de reconocimiento de patrones.

raíz del árbol dialéctico en construcción. Luego se verifica que $\mathcal{A}$ sea consistente y minimal con respecto a Ψ y se obtienen los literales de minimalidad y consistencia de $\mathcal{A}$ usando el procedimiento ObtenerCjtos.

A continuación los enlaces a los derrotadores de A se utilizan para elegir un conjunto de posibles derrotadores del argumento $\mathcal{A}$. Para cada uno estos argumentos $\mathcal{B}$ se comprueba si efectivamente existe una instancia del mismo que derrota a $\mathcal{A}$ y se puede obtener a partir del conjunto actual de observaciones Ψ. Si $\mathcal{B}$ cumple con estas condiciones, aún debe cumplir una propiedad adicional antes de ser incorporado al árbol dialéctico: no producir una situación falaz en la línea argumentativa. En este punto se utilizan los literales de minimalidad y consistencia de cada uno de los argumentos en la línea de argumentación previos a $\mathcal{B}$. En el caso particular del argumento $\mathcal{B}$ sabemos que constituye un argumento de interferencia para la raíz $\mathcal{B}$. Sin embargo, la misma secuencia de acciones se realiza para cada uno de los argumentos que se incorporan al árbol dialéctico. En general, antes de agregar un nuevo derrotador se verifican las condiciones de aceptabilidad de la línea de argumentación, si $\mathcal{B}$ es un derrotador de bloqueo debemos ser cuidadosos en respectar la tercer condición de la definición 4.22. Si $\mathcal{B}$ no respeta estas condiciones simplemente se lo elimina. Si el argumento $\mathcal{B}$ es un argumento de soporte (respectivamente de interferencia) entonces ninguno de los literales presentes en $\mathcal{B}$ debe estar presente en $\mathtt{LC}(\mathcal{C}_i)$ donde $\mathcal{C}_i$ simboliza cualquier argumento de soporte (respectivamente interferencia) en la línea de argumentación. En forma similar, para determinar si $\mathcal{B}$ es capaz de introducir un ciclo en la línea de argumentación las reglas de $\mathcal{B}$ deben compararse con los argumentos previos a él. Este procedimiento se lleva a cabo en la función Válido, detallada en el algoritmo 5.5. Esta función toma como entrada el conjunto de literales de consistencia de los argumentos de soporte (LS), el conjunto de literales de consistencia de los argumentos de interferencia (LI), la línea de argumentos hasta $\mathcal{A}$ (λ), el argumento $\mathcal{A}$ y el programa $\mathcal{P}$ en consideración y devuelve verdadero si $\mathcal{A}$ es válido con respecto a la línea argumentativa.

Una vez que todos estos controles han sido realizados sobre $\mathcal{B}$ en forma satisfactoria, este argumento se agrega al árbol dialéctico y los enlaces a los derrotadores se utilizan para determinar qué elementos analizar en el futuro. Este proceso continúa hasta que no existan más derrotadores que considerar mediante el procedimiento ObtenerEstado, cuya estructura es análoga al procedimiento Estado (ver algoritmo 5.3), y sólo difiere en que utiliza las estructuras de datos definidas en esta sección. Finalmente el marcado del árbol dialéctico se realiza de la forma usual.

Algoritmo 5.5 Válido
entrada: $\mathcal{A}$, $\mathcal{P}$, LS, LI, λ
salida: es_válido
es_válido:=verdadero
Si $\mathcal{A}$ es el tercer derrotador por bloqueo de la línea argumentativa
entonces es_válido:=falso
Si $\mathcal{A}$ es un argumento de interferencia y $literales(A) \cap$ LI$\neq \emptyset$
entonces es_válido:=falso
Si $\mathcal{A}$ es un argumento de soporte y $literales(A) \cap$ LS$\neq \emptyset$
entonces es_válido:=falso
Si $\mathcal{A}$ es un subargumento de algún argumento en el conjunto λ
entonces es_válido:=falso
devolver(es_válido)

Figura 5.6: Algoritmo para determinar si un argumento es válido con respecto a su línea argumentativa.

5.4. Trabajos relacionados

En esta sección detallaremos algunos de los trabajos previos a nuestra presentación que elaboran sobre el uso de conocimiento precompilado en los formalismos argumentativos. A continuación resumiremos el contenido de estos trabajos y destacaremos las similitudes y diferencias con el conocimiento precompilado de la PLRO.

5.4.1. Bases de argumentos

Los primeros trabajos sobre conocimiento precompilado en sistemas argumentativos datan del año 1993, al publicarse el artículo *"Bases de argumentos: su mantenimiento y revisión"* [García et al., 1993a]. En esta contribución se presenta una modificación del sistema de Simari y Loui [Simari y Loui, 1992] que incorpora una estructura de conocimiento precompilado denominada *sistema de mantenimiento de argumentos*. Este componente permite almacenar los argumentos hallados en la obtención de inferencias para luego utilizarlos cuando es necesario resolver alguna consulta por segunda vez. En este esquema, si una consulta h es sustentada por un argumento $\mathcal{A}$ tal que $\mathcal{A}$ está garantizado entonces $\mathcal{A}$ se almacena en la *base de*

argumentos (conocimiento precompilado) del sistema.

Este primer acercamiento muestra varias desventajas. Entre las más preponderantes podemos señalar las siguientes:

- *El conocimiento almacenado en la base de argumentos depende de la información contingente de la base de conocimiento.* Esto implica que, por ejemplo, al agregar un nuevo hecho contingente a la base de conocimiento es necesario revisar el estado de todos los argumentos almacenados, dado que es posible que algunos de ellos no sean consistentes o no sean minimales con respecto a la nueva base de conocimiento. Esta tarea resulta computacionalmente costosa. Si la actualización de la base de conocimiento es frecuente esto no es recomendable.

- *El conocimiento precompilado solamente resulta de utilidad al resolver consultas que ya han sido consideradas en el pasado.* A diferencia de las bases de dialéctica, que resumen la información de la base de conocimiento y son útiles ante cualquier consulta, la base de argumentos solamente ayuda a resolver las consultas que ya han sido respondidas por el sistema.

- *Solamente se almacenan los argumentos justificados (garantizados), pero no se toman en cuenta las relaciones de éstos con los demás argumentos.*

En acercamientos posteriores basados en este trabajo se intenta solucionar algunos de estos inconvenientes. En [Capobianco y Chesñevar, 1999] se presenta una propuesta que utiliza como marco teórico al sistema MTDR. En este escenario se enfrentan los problemas de utilizar sistemas argumentativos en entornos dinámicos y se postula al conocimiento precompilado como una herramienta que puede mejorar la interacción con el ambiente. En lugar de almacenar solamente los argumentos se memorizan los árboles de dialéctica que permitieron justificar (garantizar) a las consultas realizadas hasta ese momento. Esto resulta más costoso en cuanto al espacio de almacenamiento requerido pero facilita el análisis que se debe llevar a cabo al actualizar la base de conocimiento.

Posteriormente este mismo esquema de conocimiento precompilado se adaptó a la programación en lógica rebatible [Capobianco et al., 2000]. En este trabajo se detallan además los algoritmos para la creación y mantenimiento de un *repositorio de justificaciones* (estructura que resume al conocimiento precompilado), definiendo como actualizar el mismo ante los cambios que puede sufrir la base de conocimiento

en un ambiente dinámico. Sin embargo los principales problemas, ya señalados, siguen presentes en este sistema. Cabe destacar que el concepto de bases de dialéctica propuestas en esta tesis solucionan estos inconvenientes, dado que el conocimiento precompilado es independiente de la información contingente (conjunto de observaciones) y además puede usarse para asistir en la resolución de cualquier consulta.

Justamente los problemas que posee esta versión motivaron parte del trabajo presentado en esta disertación. A partir de esta situación se observó que era posible modificar el sistema de la programación en lógica rebatible, para obtener un nuevo formalismo que permitiera utilizar otro esquema de conocimiento precompilado que solucionara estos inconvenientes. Al definir posteriormente el sistema de la PLRO, efectivamente resultó posible hallar una propuesta superadora.

5.4.2. Las formas argumentales

La noción de bases de dialéctica desarrollada en este capítulo tiene relación con las *formas argumentales* propuestas por C. Chesñevar en [Chesñevar, 1996]. El trabajo de Chesñevar pretende optimizar al sistema presentado en [Simari et al., 1994].[5] Las motivaciones de esta optimización consisten en aprovechar el trabajo repetido que se realiza al computar árboles dialécticos semejantes.

En el sistema en consideración se construyen árboles dialécticos a partir de argumentos cuyas conclusiones constituyen literales fijos. Estos argumentos se obtienen en forma dinámica a partir de la base de conocimiento. En este escenario, supongamos que se resuelve en primer lugar una consulta para el literal $p(a, b)$ y a continuación una consulta para el literal $p(a, a)$. Es lícito afirmar que existen numerosas similitudes entre el árbol dialéctico de $p(a, b)$ y el de $p(a, a)$. Dado que las reglas rebatibles vinculan predicados no instanciados que comparten variables, los argumentos que pueden construirse a partir de las mismas reglas exhibirán una estructura similar. Sin embargo el sistema no toma en cuenta esta característica, resolviendo ambas consultas en forma independiente. Sería deseable factorizar las tareas en común existentes en la resolución de estas consultas, a fin de ahorrar trabajo computacional realizando estas tareas sólo una vez.

Para solucionar esta situación se define un nuevo formalismo argumentativo, en el cual no se computan argumentos para literales fijos. En su lugar se obtendrán *formas argumentales* para literales cualesquiera. Una forma argumental tiene una apariencia

[5] Consultar la sección 3.2.1.

similar a un argumento pero permitirá sustentar a literales no necesariamente fijos.

Las formas argumentales están interrelacionadas entre sí mediante la relación de *conflicto*, una generalización de la noción de contraargumentación. Utilizando la relación de conflicto se construye un *árbol de formas argumentales* y mediante la relación de derrota se elaboran los árboles dialécticos tradicionales. De esta manera el árbol de formas argumentales para $p(X, Y)$ contiene una forma argumental A que respalda a $p(X, Y)$ y a partir de ésta se obtienen las formas argumentales en conflicto con A, repitiendo este proceso para cada una de estas, tal como se realiza para construir los árboles dialécticos. Al resolver una consulta sobre un literal fijo los árboles de formas argumentales se instancian a fin de hallar árboles dialécticos convencionales.

Estos elementos dan lugar al concepto de *estructura dialéctica* para un literal h. Un determinado literal puede poseer varias formas argumentales que lo respaldan. Cada una de estas formas argumentales tiene un árbol de formas argumentales asociado. El conjunto de todos los árboles de formas argumentales para un literal h constituye una foresta denominada estructura dialéctica para h, esta estructura posibilita realizar una análisis sobre h en forma independiente de los hechos particulares de la base de conocimiento y contiene toda la información sobre las formas argumentales que sustentan o están en conflicto con h. Al resolver una consulta sobre un literal h el mecanismo de inferencia del sistema instancia los árboles presentes en la estructura dialéctica de h a fin de determinar si existe algún árbol dialéctico que constituya una justificación para h.

Es posible afirmar que el trabajo realizado por Chesñevar posee relación con el conocimiento precompilado desarrollado para la PLRO. Sin embargo existen numerosas diferencias. En primer lugar, en esa propuesta se define un formalismo argumentativo independiente. Por el contrario, las bases de dialéctica constituyen una reformulación del mecanismo de inferencia de la PLRO, pero no producen un cambio en la definición de los elementos básicos del sistema (*i.e*, no definen un nuevo sistema).

Cada estructura dialéctica resume una foresta de árboles para un determinado literal. De esta forma existen numerosas formas argumentales que serán almacenadas más de una vez por pertenecer a diferentes árboles. En cambio las bases de dialéctica almacenan sólo una vez cada argumento potencial. Dado que una base de dialéctica representa un grafo de argumentos la información necesaria para todas las consultas del sistema se resume en una sola estructura. Esto provoca una mayor eficiencia con

respecto a la memoria utilizada por las implementaciones basadas en esta técnica.

Por otra parte los árboles de formas argumentales solamente almacenan la relación de conflicto, sin identificar cuál de los argumentos es el derrotador. En consecuencia al instanciar cada uno de los árboles de formas argumentales es necesario cuál de los argumentos en conflicto prevalece sobre su contrincante. Las bases de dialéctica permiten registrar la relación de derrota entre los argumentos potenciales. Por lo tanto al utilizar esta estructura para resolver una consulta en particular no hace falta calcular la relación de derrota.

Finalmente cabe destacar que nuestro trabajo también abarca cuestiones de implementación del conocimiento precompilado, detallando cuándo y cómo se debe crear y acceder este componente. Así nuestra propuesta presenta una alternativa superadora al de las propuestas existentes para el uso de conocimiento precompilado en los sistemas argumentativos.

5.5. Conclusiones

En este capítulo hemos culminado la presentación del formalismo de la PLRO. Este sistema cumple con los objetivos fijados al comienzo de esta tesis, ya que que permite manejar información incompleta y potencialmente inconsistente.

Asimismo se ha provisto al sistema con mecanismos para manejar la percepción, en base a los cuales se actualiza el programa representando la información del agente. Esta característica proporciona al agente la capacidad de interactuar con ambiente dinámicos. Cabe destacar que la incorporación de mecanismos de actualización en formalismos basados en argumentos constituye una propuesta original que posibilita el uso de sistemas argumentativos en un nuevo conjunto de aplicaciones prácticas.

El mecanismo de inferencia de la programación en lógica rebatible con observaciones se define desde una nueva perspectiva en base al uso de conocimiento precompilado. Esto funda una línea de investigación en el área de los sistemas argumentativos: los formalismos basados en argumentos pueden beneficiarse de esta propuesta de la misma forma en que los sistemas de mantenimiento de verdad ampliaron las capacidades de los resolvedores generales de problemas.

Finalmente se diseño una implementación eficiente del mecanismo de inferencia definido oportunamente aprovechando conceptos presentes en los algoritmos de reconocimiento de patrones.

Capítulo 6

Propiedades de la PLRO

En este capítulo presentaremos algunas de las propiedades formales más interesantes que emergen del formalismo de la programación en lógica rebatible con observaciones. A partir de este análisis identificaremos un conjunto de propiedades que permitan caracterizar a los sistemas argumentativos en general.

En primer lugar consideraremos el procedimiento de prueba de la PLRO en función de construir argumentos y establecer las relaciones entre los mismos. A continuación se estudiarán las propiedades de los árboles dialécticos y se propondrá una nueva clasificación de los argumentos asociados a un programa de la PLRO. Esta clasificación permitirá identificar un conjunto de categorías sobre las inferencias de los sistemas argumentativos.

Posteriormente se presenta una definición del conjunto de creencias inducidas por un programa, destacando las propiedades del mecanismo de inferencia de la PLRO y proponiendo además un conjunto de pautas generales para estudiar la inferencia en los formalismos argumentativos.

6.1. Argumentos en la PLRO: Propiedades

Los sistemas argumentativos existentes, a pesar de su diversidad, comparten en general una división por niveles del proceso de inferencia [Prakken y Vreeswijk, 1999], donde cada nivel contribuye al refinamiento de la información a fin de obtener las conclusiones del sistema. En esta sección estudiaremos los basamentos del proceso de inferencia, como son la capa lógica y la construcción de argumentos. Cabe destacar que del estudio particular de la PLRO se desprende un conjunto de propiedades recomendables para los sistemas argumentativos en general. Esta misma metodología

caracterizará el trabajo realizado en el resto de este capítulo.

6.1.1. Nivel lógico

En los sistemas argumentativos, el primer proceso que se aplica a la información presente en la base de conocimiento consiste en una relación de consecuencia lógica. Esta relación puede estar basada en la lógica de primer orden o en un lenguaje al estilo de la programación en lógica. La PLRO opta por la segunda de estas alternativas.

En tal sentido, las derivaciones rebatibles de la PLRO (definición 4.11) corresponden al primer nivel del proceso de inferencia. Recordemos que una derivación rebatible constituye un simple encadenamiento de reglas en la base de conocimiento que permite obtener un literal determinado. Sin embargo, estas derivaciones no constituyen las inferencias del sistema propiamente dicho, sino que deben construirse argumentos y posteriormente someterse a un proceso dialéctico para determinar si serán o no aceptadas.

Comenzaremos analizando el rol de esta relación de inferencia básica en la PLRO. En primer lugar observemos que la derivación rebatible no preserva la consistencia lógica de las premisas, dado que a partir de un conjunto consistente de observaciones es factible derivar un par de literales complementarios.

Ejemplo 6.1 Consideremos el siguiente programa en la PLRO:

$$\begin{array}{l} \mathtt{p} \\ \mathtt{q} \prec \mathtt{p} \\ \sim\mathtt{p} \prec \mathtt{q} \end{array}$$

Es claro que p es una derivación rebatible (estos es, es derivable rebatiblemente) a partir de este programa, dado que pertenece al conjunto de observaciones. Sin embargo ~p es también una derivación rebatible. ■

Esto no constituye una situación indeseable, sino que por el contrario es producto de la flexibilidad que caracteriza a los primeros niveles del proceso de inferencia en los sistemas argumentativos. Posteriormente el formalismo realizará una selección los literales contradictorios a partir de un análisis dialéctico que involucra un criterio de preferencia. En tal sentido cada nivel del mecanismo de inferencia se puede interpretar como un filtro que selecciona aquellos literales candidatos tendientes a convertirse en las conclusiones definitivas del sistema.

6.1.2. Construcción de argumentos

El segundo nivel del mecanismo de inferencia en los sistemas argumentativos involucra la construcción de argumentos para sustentar a una determinada conclusión. Todo argumento require la existencia de una derivación (eventualmente vacía) para su conclusión a partir de la base de conocimiento. Sin embargo, no cualquier derivación constituye un argumento, dado que se deben satisfacer algunas restricciones adicionales.

En el caso de la PLRO todo argumento debe satisfacer las restricciones de minimalidad y consistencia. Por tanto, el conjunto de reglas del ejemplo 6.1 no constituye un argumento, dado que permite la derivación de literales contradictorios (p y $\sim p$) y por lo tanto no es consistente.

Se podría suponer que estos requerimientos dificultan la obtención de los argumentos, pero son características esenciales para alcanzar la integridad necesaria en este nivel de inferencia. La consistencia evita la formación de argumentos *auto-derrotados* [Pollock, 1995]. Se dice que un argumento $\mathcal{A}$ es auto-derrotado si se derrota a sí mismo por medio de la relación de derrota del sistema. Por caso, Pollock [Pollock, 1995] menciona que en su sistema este tipo de argumentos hace colapsar el proceso de inferencia argumentativo del formalismo OSCAR, por lo cual debe evitarselos mediante condicionamientos especiales en la construcción del grafo de derrota.[1] Problemas similares a este se manifestarían en la PLRO y en otros sistemas argumentativos si se permitiera la existencia de argumentos auto-derrotados.

Se puede demostrar que en la PLRO no se generan argumentos auto-derrotados, tal como lo evidencia la siguiente proposición.[2]

Proposición 6.1 No existen argumentos auto-derrotados en la PLRO. ■

Demostración: Supongamos que existe un argumento $\langle \mathcal{A}, p\rangle$ que se derrota a sí mismo. En consecuencia, de acuerdo con la definición de derrota de la PLRO, existe un subargumento $\langle \mathcal{B}, q\rangle$ de $\langle \mathcal{A}, p\rangle$ tal que $\langle \mathcal{B}, q\rangle$ contraargumenta a $\langle \mathcal{A}, p\rangle$ en p y por lo tanto el conjunto $\{q, p\}$ es contradictorio. Pero tanto q como p son cabezas de reglas rebatibles en $\mathcal{A}$ y entonces es posible deducir un conjunto de literales contradictorios a partir de $\Psi \cup \mathcal{A}$ (por lo cual $\mathcal{A}$ no respeta la definición

[1] Este tema se discute en detalle en la sección 3.1.

[2] En este capítulo, a menos que se indique lo contrario, todas las proposiciones se refieren al sistema de la PLRO.

de consistencia presente en la definición de argumento). Este absurdo proviene de suponer que $\mathcal{A}$ se derrota a sí mismo. □

La minimalidad tiene también como resultado limitar el conjunto de argumentos que se pueden construir a partir de un programa, evitando en este caso la construcción de argumentos redundantes.

Consideremos al conjunto de reglas $\{\mathtt{q} \prec \mathtt{r}, \mathtt{p} \prec \mathtt{q}, \mathtt{b} \prec \mathtt{a}\}$ que permite concluir el literal $\mathtt{p}$ (con $\mathtt{r}$ presente en el conjunto de observaciones). Al intentar construir un argumento en base a este conjunto, es importante notar que la tercer regla no interviene en la derivación de $\mathtt{p}$, y por tanto no tendría sentido alguno incluirla en el mismo. Además agregar reglas superfluas debilitaría los argumentos. Observemos que si un argumento $\mathcal{A}$ estuviese formado por este conjunto de reglas sería vulnerable a un ataque por parte de un argumento $\mathcal{B}$ formado por $\{\sim\mathtt{b} \prec \mathtt{c}\}$ aunque este no tiene ninguna relación con el literal p en cuestión. Notemos que aunque parezca obvio evitar este tipo de situaciones, los sistemas argumentativos que no exigen minimalidad en sus argumentos (como por ejemplo el formalismo de Vreeswijk [Vreeswijk, 1997]) permiten este tipo de elementos.

Los argumentos utilizan una noción de consistencia local. No obstante es posible construir pares de argumentos $\langle \mathcal{A}, p\rangle$, $\langle \mathcal{B}, q\rangle$ tal que p y q son complementarios. Por lo tanto la aceptación conjunta de ambos argumentos produciría una contradicción. Para que el sistema decida qué conclusiones sancionar en presencia de este tipo de conflictos se analizan las distintas relaciones existentes entre los argumentos.

6.2. Relaciones entre argumentos

En la actualidad, existe un gran diversidad de sistemas argumentativos y cada sistema posee su propio proceso de inferencia. Con el fin de analizar las propiedades de la inferencia en los sistemas argumentativos hemos propuesto una abstracción basada en identificar un conjunto de relaciones básicas entre argumentos.

El proceso de inferencia de un sistema argumentativo puede por lo general dividirse en dos etapas. La primer etapa se encarga de la construcción de los argumentos en base al conocimiento del sistema. La segunda etapa consiste en detectar un conjunto de relaciones entre estos argumentos que permitirán clasificarlos en argumentos *aceptados* y argumentos *rechazados*. Unicamente los argumentos aceptados serán aquellos que sustenten las inferencias del sistema.

En esta sección estudiaremos las relaciones entre argumentos definidas en la

programación en lógica rebatible con observaciones. Utilizando a la PLRO como caso de estudio es posible derivar un conjunto de propiedades aplicables a los formalismos argumentativos en general.

En la PLRO podemos distinguir cuatro relaciones preponderantes: subargumentación, concordancia, contrargumentación y derrota. A continuación estudiaremos las características de cada una de ellas.

6.2.1. Subargumentación

Esta relación se desprende directamente del concepto de argumento. En la PLRO este concepto se introduce en la definición 4.12, de acuerdo a la cual un argumento $\mathcal{B}$ es subargumento de $\mathcal{A}$ si $\mathcal{B} \subseteq \mathcal{A}$.

Definición 6.1 (*Subargumentación*)
Dado un par de argumentos $(\mathcal{B}, \mathcal{A})$ construidos en base a un programa $\mathcal{P} = (\Psi, \Delta)$ diremos que $(\mathcal{B}, \mathcal{A})$ pertenece a la relación de subargumentación asociada a $\mathcal{P}$ si y sólo si $\mathcal{B}$ es un subargumento de $\mathcal{A}$. En general, denotaremos la relación de subargumentación asociada a un programa $\mathcal{P} = (\Psi, \Delta)$ como $S_{(\Psi,\Delta)}$. ■

Cabe destacar que de acuerdo a esta definición, la subargumentación en la PLRO cumple con las propiedades de reflexividad, antisimétrica y transitividad, por lo que es posible afirmar que es una relación de orden parcial entre los argumentos.

Pragmáticamente, la subargumentación debe relacionar argumentos entre los cuales no existen conflictos. Notemos que ésta es una propiedad es deseable en los sistemas argumentativos. Si por el contrario existiera un par de argumentos $(\mathcal{B}, \mathcal{A})$ tal que $\mathcal{B}$ es un subargumento de $\mathcal{A}$ y además $\mathcal{A}$ y $\mathcal{B}$ son contradictorios entre sí, entonces podemos afirmar que el sistema argumentativo en cuestión permite generar argumentos auto-derrotados (en este ejemplo $\mathcal{B}$ sería un argumento de esta clase). En la PLRO es trivial demostrar que si $\mathcal{B}$ es un subargumento de $\mathcal{A}$ entonces $\mathcal{A}$ y $\mathcal{B}$ no contienen literales contradictorios. En caso contrario, $\mathcal{A}$ no sería un argumento, dado que no cumpliría con la segunda condición de la definición 4.12. En consecuencia, la subargumentación es una *relación de acuerdo* entre argumentos.

Definición 6.2 (*Relación de acuerdo*)
Una relación R sobre pares de argumentos es una relación de acuerdo si no existe un par de argumentos $(\mathcal{A}, \mathcal{B}) \in R$ tal que la aceptación conjunta de $\mathcal{A}$ y $\mathcal{B}$ conduce a una contradicción. ■

La subargumentación puede también ser usada para caracterizar la clase de aquellos programas de la PLRO que no poseen información contradictoria. Para mostrar esta propiedad definiremos en primer lugar algunas nociones auxiliares que serán de utilidad:

Definición 6.3 (*Conjunto de argumentos*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO. Denotaremos mediante args($\mathcal{P}$) al conjunto de todos los argumentos que se pueden obtener a partir de $\mathcal{P}$. ■

Definición 6.4 (*Programa libre de conflictos*)
Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO. Diremos que $\mathcal{P}$ es *libre de conflictos* si para todo argumento $\langle \mathcal{A}, q \rangle$ de $\mathcal{P}$ vale que q es un literal garantizado. ■

Por medio de la subargumentación es posible enunciar una condición suficiente, aunque no necesaria, para que un programa de la PLRO sea libre de conflictos:

Proposición 6.2 Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO y sea $S_{(\Psi,\Delta)}$ la relación de subargumentación asociada a $\mathcal{P}$. Si $S_{(\Psi,\Delta)}$ posee un elemento supremo, entonces $\mathcal{P}$ es un programa libre de conflictos. ■

Demostración: Supongamos por el absurdo que existe un programa $\mathcal{P} = (\Psi, \Delta)$ tal que $S_{(\Psi,\Delta)}$ posee un elemento supremo, pero sin embargo existen conflictos entre los argumentos generados a partir de $\mathcal{P}$. Entonces existe un par de argumentos $(\mathcal{A}, \mathcal{B})$ tal que $\mathcal{A}$ está en conflicto con $\mathcal{B}$. Pero como $S_{(\Psi,\Delta)}$ posee un elemento supremo existe un argumento $\mathcal{C}$ tal que tanto $(\mathcal{A}, \mathcal{C})$ como $(\mathcal{B}, \mathcal{C})$ pertenecen a $S_{(\Psi,\Delta)}$. Luego $\mathcal{C}$ es un argumento auto-derrotado, pero de acuerdo con la proposición 6.1 esto constituye un absurdo que provino de suponer la existencia de un conflicto entre $\mathcal{A}$ y $\mathcal{B}$. □

Si analizamos el sentido de esta proposición resulta obvio que si todos los argumentos están relacionados por la relación de subargumentación entonces no pueden existir conflictos entre los mismos. Esta propiedad puede también generalizarse a otros sistemas argumentativos que posean una noción de subargumento y en los cuales no existan argumentos auto-derrotados.

6.2.2. Concordancia

La segunda relación que estudiaremos se denomina *concordancia*[3] y es, tal como la subargumentación, una relación de acuerdo entre argumentos. Intuitivamente, dos

[3]Este concepto fue introducido en [Simari y Loui, 1992].

argumentos son concordantes entre sí cuando su aceptación conjunta no conduce a una contradicción.

En la PLRO la noción de concordancia entre argumentos se introduce en la definición 4.22, a fin de evitar la formación de falacias en el proceso de inferencia de la PLRO. En tal sentido, las líneas argumentativas aceptables (definición 4.22) deben contener argumentos de soporte (definición 4.20) no contradictorios (o lo que es equivalente, concordantes entre sí). Lo mismo debe suceder con los argumentos de interferencia (definición 4.20). Aunque a partir de esta definición podría minimizarse la importancia de la concordancia entre argumentos, es importante reconocer que esta relación merece ser analizada desde una perspectiva más general.

Definición 6.5 (*Argumentos concordantes - Concordancia*)

Dado un programa $\mathcal{P} = (\Psi, \Delta)$ de la PLRO, un argumento $\mathcal{A}$ es *concordante* con un argumento $\mathcal{B}$ si y sólo si el conjunto $\mathcal{A} \cup \mathcal{B} \cup \Psi$ no permite derivar literales complementarios.

Dado un par de argumentos $(\mathcal{A}, \mathcal{B})$ construidos en base a un programa $\mathcal{P} = (\Psi, \Delta)$ diremos que $(\mathcal{A}, \mathcal{B})$ pertenece a la relación de concordancia asociada a $\mathcal{P}$ si y sólo si $\mathcal{A}$ es concordante con $\mathcal{B}$. En general, denotaremos la relación de concordancia asociada a un programa $\mathcal{P} = (\Psi, \Delta)$ como $CC_{(\Psi,\Delta)}$. ■

Observemos en primer lugar que la concordancia es una relación de acuerdo más general que la relación de subargumentación. En el formalismo de la PLRO resulta sencillo demostrar que la subargumentación es un subconjunto de la relación de concordancia.

Proposición 6.3 Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO. Entonces $S_{(\Psi,\Delta)} \subseteq CC_{(\Psi,\Delta)}$. ■

Demostración: Sea $(\mathcal{B}, \mathcal{A}) \in S_{(\Psi,\Delta)}$. Como $\mathcal{B}$ es un subargumento de $\mathcal{A}$ podemos afirmar que el conjunto $\mathcal{A} \cup \mathcal{B} \cup \Psi = \mathcal{B} \cup \Psi$ no permite derivar literales contradictorios (en caso contrario, $\mathcal{B}$ no sería un argumento). En consecuencia el par $(\mathcal{A}, \mathcal{B})$ pertenece a $CC_{(\Psi,\Delta)}$. □

Otra propiedad coherente de la relación de concordancia de un sistema argumentativo es que satisfaga las propiedades de reflexividad y simetría. En la PLRO es posible demostrar estas propiedades.

Proposición 6.4 Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO. Entonces $CC_{(\Psi,\Delta)}$ es una relación reflexiva y simétrica. ■

Demostración: En primer lugar, $CC_{(\Psi,\Delta)}$ es reflexiva, dado que para todo argumento $\mathcal{A}$ se cumple por definición que $\mathcal{A} \cup \Psi$ es un conjunto no contradictorio de argumentos y por lo tanto es concordante. $CC_{(\Psi,\Delta)}$ es también simétrica, lo cual surge trivialmente de la definición de concordancia. □

Ejemplo 6.2 Observemos que la concordancia no constituye una relación de orden, puesto que no es transitiva. Para evidenciar esta caraterística consideremos los argumentos:

- $\mathcal{A} = \{\mathtt{b} \prec \mathtt{d}; \mathtt{d} \prec \mathtt{c}\}$
- $\mathcal{B} = \{\mathtt{a} \prec \mathtt{b}; \mathtt{b} \prec \mathtt{e}\}$
- $\mathcal{C} = \{\sim\mathtt{d} \prec \mathtt{f}\}$

En este caso $\mathcal{A}$ es concordante con $\mathcal{B}$ y $\mathcal{B}$ es concordante con $\mathcal{C}$, aunque $\mathcal{A}$ no es concordante conn $\mathcal{C}$. ■

Dado un argumento $\mathcal{A}$ es posible identificar a todos los argumentos concordantes con $\mathcal{A}$ como concordantes($\mathcal{A}$). En la PLRO, para todo árbol de dialéctica para $\mathcal{A}$ se cumple que sólo los argumentos que pertenecen a concordantes($\mathcal{A}$) pueden intervenir como argumentos de soporte para $\mathcal{A}$. Esta característica puede también extrapolarse a otros sistemas argumentativos. Es razonable que sólo los argumentos de acuerdo con $\mathcal{A}$ puedan intervenir para defenderlo. De lo contrario podría darse el caso que un argumento esté al mismo tiempo a favor y en contra de $\mathcal{A}$.

El análisis precedente también justifica la decisión de no permitir argumentos conflictivos en el conjunto de soporte o interferencia de una línea argumentativa. Sólo los argumentos que "están de acuerdo" pueden respaldarse unos a otros.

Cabe destacar que la concordancia también permite caracterizar a los programas libres de conflictos (definición 6.4), aunque en forma menos restrictiva que la proposición 6.2.

Proposición 6.5 Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO y sea $CC_{(\Psi,\Delta)}$ la relación de concordancia asociada a $\mathcal{P}$. Si para todo argumento $\mathcal{A}$ obtenido en base a $\mathcal{P}$ se satisface que concordantes($\mathcal{A}$) = args($\mathcal{P}$) entonces $\mathcal{P}$ es un programa libre de conflictos. ■

Demostración: Como para cada argumento $\mathcal{A}$ vale que concordantes($\mathcal{A}$) = args($\mathcal{P}$) podemos afirmar que no existe un par de argumentos $(\mathcal{B}, \mathcal{C})$ tal que $\mathcal{B}$ y $\mathcal{C}$ sustenten literales contradictorios. Por lo tanto no existen contraargumentos en base a $\mathcal{P}$ y en consecuencia $\mathcal{P}$ es un programa libre de conflictos. □

Esta proposición brinda una condición necesaria y suficiente para describir a un programa libre de conflictos. Es decir, si $\mathcal{P}$ es un programa libre de conflictos entonces para todo argumento $\mathcal{A}$ obtenido en base a $\mathcal{P}$ se cumple que concordantes($\mathcal{A}$) = args($\mathcal{P}$).

Notemos que la proposición 6.2 implica la proposición 6.5, pero no vale la recíproca, dado que dos argumentos no contradictorios no necesariamente están conectados por la relación de subargumentación y sin embargo dos subargumentos de un mismo argumento son siempre no contradictorios.

6.2.3. Contraargumentación

La primera de las relaciones de conflicto que analizaremos es la de *contraargumentación*. Esta relación distingue los pares de argumentos cuya aceptación conjunta conduciría a un razonamiento inconsistente.[4]

Definición 6.6 (*Contraargumentación*)
Dado un par de argumentos $(\mathcal{A}, \mathcal{B})$ construídos en base a un programa $\mathcal{P} = (\Psi, \Delta)$ diremos que $(\mathcal{A}, \mathcal{B})$ pertenece a la relación de contraargumentación asociada a $\mathcal{P}$ si y sólo si $\mathcal{A}$ es un contraargumento de $\mathcal{B}$. En general, denotaremos la relación de contraargumentación asociada a un programa $\mathcal{P} = (\Psi, \Delta)$ como $C_{(\Psi,\Delta)}$. ■

A partir de la definición formal de contraargumento (definición 4.16) es posible distinguir dos clases de contraargumentación:

- Recíproca: se produce cuando para dos argumentos $\mathcal{A}$ y $\mathcal{B}$ tanto ($\mathcal{A}$,$\mathcal{B}$) como ($\mathcal{B}$,$\mathcal{A}$) figuran en la relación.
- Unilateral: sólo uno de los pares ($\mathcal{B}$,$\mathcal{A}$) o ($\mathcal{A}$,$\mathcal{B}$) pertenece a la relación.

Ejemplo 6.3 Consideremos al par de argumentos $\mathcal{A} = \{\mathtt{q} \prec \mathtt{p}\}$ y $\mathcal{B} = \{\sim\mathtt{q} \prec \mathtt{p}\}$. En este caso la contraargumentación entre $\mathcal{A}$ y $\mathcal{B}$ es recíproca. Por el contrario entre $\mathcal{A}_1 = \{\mathtt{s} \prec \mathtt{q}, \mathtt{q} \prec \mathtt{p}\}$ y $\mathcal{B}$ la contrargumentación es unilateral. ■

[4]Cabe destacar que en algunos sistemas argumentativos se denomina esta relación como *ataque*.

En la PLRO es posible enunciar las siguientes propiedades de la relación de contraargumentación, cuya demostración es directa a partir de la definición:

Proposición 6.6 Dados dos argumentos $\mathcal{A}$ y $\mathcal{B}$, $\mathcal{A}$ es un contraargumento de $\mathcal{B}$ si y sólo si $literales(\mathcal{A}) \cup literales(\mathcal{B})$ es un conjunto contradictorio. ■

Proposición 6.7 La contrargumentación entre dos argumentos $\mathcal{A}$ y $\mathcal{B}$ es recíproca si y sólo si sus conclusiones son complementarias. ■

Es importante destacar que la contrargumentación unilateral entre un par de argumentos $\mathcal{A}$ y $\mathcal{B}$ se basa en la existencia de una contrargumentación recíproca entre $\mathcal{A}'$ y $\mathcal{B}$ (donde $\mathcal{A}'$ es un subargumento de $\mathcal{A}$) o $\mathcal{A}$ y $\mathcal{B}$' (donde $\mathcal{B}'$ es un subargumento de $\mathcal{B}$). Esto se deriva trivialmente de la definición de contrargumentación y muestra que la contrargumentación recíproca representa la base de los conflictos entre argumentos.

Habiendo ya definido las relaciones de acuerdo entre argumentos presentes en los sistemas argumentativos, cabe preguntarse cuál será el vínculo existente entre éstas y la relación de contraargumentación. Como la concordancia es la más general de las relaciones de acuerdo analizaremos que sucede entre ésta y la contraargumentación.

Observemos que la contraargumentación refleja los conflictos existentes entre argumentos. Por lo tanto es razonable suponer que no deben existir pares en común entre ésta y la concordancia entre argumentos. Un argumento $\mathcal{A}$ no puede estar en conflicto y de acuerdo con un argumento $\mathcal{B}$. Esta propiedad se satisface en la PLRO.

Proposición 6.8 Para todo programa $\mathcal{P} = (\Psi, \Delta)$ de la PLRO se cumple que $CC_{(\Psi,\Delta)} \cap C_{(\Psi,\Delta)} = \emptyset$. ■

Demostración: Supongamos por el absurdo que existe un par de argumentos $(\mathcal{A}, \mathcal{B})$ que pertenece a $CC_{(\Psi,\Delta)} \cap C_{(\Psi,\Delta)}$. Entonces debe verificarse que a partir de $\mathcal{A} \cup \mathcal{B} \cup \Psi$ no se pueden inferir literales complementarios (por ser concordantes $\mathcal{A}$ y $\mathcal{B}$). Pero por ser $\mathcal{A}$ un contraargumento de $\mathcal{B}$ debe existir un literal q en las reglas de $\mathcal{A}$ y un literal p en las reglas de $\mathcal{B}$ tal que el conjunto $\{p, q\}$ es contradictorio. Luego p y q pueden inferirse a partir de $\mathcal{A} \cup \mathcal{B} \cup \Psi$, dado que p es un literal en $\mathcal{A}$, q es un literal en $\mathcal{A}$ y tanto $\mathcal{A}$ como $\mathcal{B}$ son argumentos obtenidos a partir de (Ψ, Δ). Esto es un absurdo que provino de suponer al conjunto $CC_{(\Psi,\Delta)} \cap C_{(\Psi,\Delta)}$ distinto de vacío. □

Finalmente, y como era de esperar, la contrargumentación tampoco contiene pares en común con la relación de subargumentación. La siguiente propiedad es un corolario de la proposición 6.8.

Corolario 6.1 Sea $\mathcal{P} = (\Psi, \Delta)$ un programa de la PLRO. Entonces podemos afirmar que $C_{(\Psi,\Delta)} \cap S_{(\Psi,\Delta)} = \emptyset$. ■

Demostración: Por la proposición 6.3 sabemos que $S_{(\Psi,\Delta)} \subseteq CC_{(\Psi,\Delta)}$ y como $CC_{(\Psi,\Delta)} \cap C_{(\Psi,\Delta)}$ podemos concluir que $C_{(\Psi,\Delta)} \cap S_{(\Psi,\Delta)} = \emptyset$. □

6.2.4. Derrota

Los conflictos planteados mediante la contrargumentación se resuelven en la relación de derrota (definición 4.19) que introduce un criterio de preferencia entre argumentos.

Definición 6.7 (*Derrota*)
Dado un par de argumentos $(\mathcal{A}, \mathcal{B})$ construidos en base a un programa $\mathcal{P} = (\Psi, \Delta)$ diremos que $(\mathcal{A}, \mathcal{B})$ pertenece a la relación de derrota asociada a $\mathcal{P}$ si y sólo si $\mathcal{A}$ es un derrotador de $\mathcal{B}$. En general, denotaremos la relación de derrota asociada a un programa $\mathcal{P} = (\Psi, \Delta)$ como $D_{(\Psi,\Delta)}$. ■

Observemos que por definición, el conjunto de pares de la relación de derrota es un subconjunto de la relación de contrargumentación. Esto es, para todo programa $\mathcal{P} = (\Psi, \Delta)$ vale que $D_{(\Psi,\Delta)} \subset C_{(\Psi,\Delta)}$.

La derrota entre argumentos depende fuertemente del criterio de preferencia utilizado. Por lo tanto reduciremos el estudio de la relación de derrota a un análisis general de este criterio. En la PLRO el criterio de preferencia se define en forma parametrizada, para otorgar más flexibilidad al sistema. De esta forma es posible utilizar diversos criterios dependiendo del dominio de aplicación. A continuación analizaremos las características deseables en un criterio de preferencia entre argumentos. Para esto compararemos las restricciones propuestas en [Vreeswijk, 1997] y [Simari et al., 1994].

En algunas propuestas (como por ejemplo, [Simari et al., 1994]) suele exigirse que el criterio de preferencia induzca un orden parcial en el conjunto de argumentos. Por su parte Vreeswijk [Vreeswijk, 1997] estudió en forma abstracta las propiedades de los criterios de preferencia en los sistemas argumentativos y enunció un conjunto

de restricciones que asegurarían la coherencia de una relación de derrota. En este sentido, resulta interesante comparar la propuesta de Vresswijk con la exigencia de que el criterio induzca un orden parcial.

Para analizar en forma abstracta el criterio de preferencia $<$ de un sistema argumentativo Vreeswijk propuso las siguientes propiedades:

1. Reflexividad y transitividad.
2. No existen cadenas infinitas: $\sigma_1 < \sigma_2 < \ldots < \sigma_n < \ldots$.
3. Para todo par de argumentos σ y τ tal que σ es un subargumento de τ, debe valer que $\tau \leq \sigma$.

Observemos que la relación $\leq$ puede no ser un orden parcial, dado que no necesariamente posee la propiedad de antisimetría. La segunda condición procura evitar cadenas infinitas para garantizar la finitud del proceso de derrota; el resto aseguran una distribución razonable de la fuerza conclusiva: un argumento no se puede fortalecer al agregar reglas y tampoco es deseable el debilitamiento poco natural de un argumento al utilizar una regla estricta.

Al comparar ambos acercamientos podemos observar que reflexividad y transitividad son elementos presentes en cualquier criterio de preferencia que induzca un orden parcial.

Para asegurar que no existen cadenas infinitas, basta combinar la propiedad de transitividad con antisimetría, tal como se muestra en la siguiente proposición:

Proposición 6.9 Sea $\preceq$ un criterio de preferencia reflexivo, antisimétrico y transitivo sobre un conjunto finito de argumentos. Entonces no existen cadenas infinitas de la forma $A_1 < A_2 < \ldots < A_n < \ldots$. ■

Demostración: Supongamos que existe una cadena infinita. Como la cantidad de argumentos es finita, para lograr que la cadena sea infinita debe existir un argumento A tal que $\sigma = A_1 < A_2 < \ldots < A_n < A < \ldots$. Entonces la cadena σ es circular. Aplicando la propiedad de transitividad, podemos deducir a partir de σ que $A_1 < A_n$. Sin embargo, aplicando la propiedad de antisimetría y dado que $A_n < A$ obtenemos que $A \not< A_n$. Esta contradicción provino de suponer la existencia de una cadena infinita σ. □

Resta analizar que sucede con los subargumentos. Es razonable que un argumento no pueda fortalecerse al agregar reglas, pero no esta claro porque esta cualidad debe

reflejarse necesariamente en el criterio de preferencia. Por caso tanto en la PLRO como en el sistema de [Simari et al., 1994] todo argumento $\mathcal{A}$ es más vulnerable un subargumento propio $\mathcal{B}$ de $\mathcal{A}$, debido a la existencia de mayor cantidad de puntos de contrargumentación en $\mathcal{A}$ que en $\mathcal{B}$.

En el caso de la PLRO se optó por no limitar los criterios de preferencia y dejar a criterio del usuario la elección de una relación que resulte razonable para la aplicación en particular. Sin embargo podría resultar interesante analizar como las distintas clases de criterios de preferencia afectan el funcionamiento del sistema. De esta forma al agregar restricciones sobre el criterio de preferencia se podrían obtener distintos tipos de formalismos, con diferentes propiedades.

6.3. Árboles dialécticos en la PLRO

La noción de conflicto localizado planteada por la contrargumentación y refinada por la derrota entre argumentos se extiende a un análisis global de un argumento $\mathcal{A}$ dado. Este análisis se sintetiza en la construcción de un árbol dialéctico para el argumento $\mathcal{A}$ (definición 4.23). Como resultado de este análisis los argumentos del árbol dialéctico son clasificados como `nodosU` (no derrotados) o `nodosD` (derrotados) de acuerdo con la definición 4.24. En base a esta clasificación se decide finalmente si $\mathcal{A}$ es un argumento garantizado.

A continuación analizaremos algunas características de los árboles de dialéctica y otras nociones derivadas de los mismos, tales como la noción de línea argumentativa aceptables. Este estudio se llevará a cabo utilizando como herramienta a las relaciones entre argumentos definidas en la sección anterior.

6.3.1. Eliminación de falacias

En primer lugar nos ocuparemos de la noción de línea argumentativa aceptable presentada en la definición 4.22. Los árboles dialécticos resumen el proceso de inferencia en la PLRO. Estos pueden representarse como un conjunto de líneas argumentativas donde cada línea representa un posible rumbo de la discusión introspectiva que se desarrolla para decidir si aceptar una determinada tesis (la conclusión en la raíz del árbol). En las líneas argumentativas se destaca el rol de las relaciones de concordancia, subargumentación y derrota.

Una línea argumentativa es *aceptable* si cumple con un conjunto de condiciones establecidas en su definición (ver definición 4.20). La primer condición establece que

los argumentos de soporte (o respectivamente interferencia) de la línea argumentativa deben pertenecer a la relación de concordancia inducida por el programa. Esto resulta coherente, porque los argumentos que sustentan una determinada posición no deben poseer conflictos entre sí.

La segunda condición puede analizarse a la luz de las relaciones de derrota y subargumentación. La misma podría reformularse de la siguiente manera: para cada par de argumentos $(\mathcal{B}, \mathcal{A}) \in S_{(\Psi,\Delta)}$ si tanto $\mathcal{B}$ como $\mathcal{A}$ aparecen en una línea argumentativa aceptable entonces $\mathcal{B}$ se introduce previamente a $\mathcal{A}$. La racionalidad de esta condición proviene de las características de la relación de derrota. Si $\mathcal{B}$ es un subargumento de $\mathcal{A}$ entonces todos los derrotadores de $\mathcal{B}$ están incluidos en los derrotadores de $\mathcal{A}$. Por lo tanto no tiene sentido introducir a $\mathcal{B}$ en el árbol dialéctico cuando ya ha sido utilizado $\mathcal{A}$. Esto conduciría a una argumentación circular.

En base a las líneas argumentativas aceptables se definen los árboles dialécticos aceptables. A continuación analizaremos las propiedades de estos elementos, comenzando por garantizar la finitud de los mismos.

Proposición 6.10 Los árboles de dialéctica aceptables de la PLRO son finitos. ■

Demostración: Supongamos la existencia de un programa $\mathcal{P}$ tal que existe un árbol dialéctico aceptable $\mathcal{T}_{\langle \mathcal{B},q \rangle}$ construído en base a $\mathcal{P}$ que posee un número infinito de nodos. Esto podría suceder por alguna de las siguientes causas:

1. Existe un argumento $\mathcal{A}$ que posee un número infinito de derrotadores.

2. Existe una línea argumentativa infinita.

Sin embargo, la primer alternativa no es posible en la PLRO porque a partir de un programa $\mathcal{P}$ sólo se pueden construir un número finito de argumentos y por lo tanto $\mathcal{A}$ no puede poseer un número infinito de derrotadores. La segunda propuesta tampoco es factible dado que un árbol dialéctico aceptable debe estar compuesto por líneas argumentativas aceptables, las cuales deben ser finitas por definición (no pueden contener ciclos). Por lo tanto $\mathcal{T}_{\langle \mathcal{B},q \rangle}$ debe ser finito. □

Es importante destacar que esta propiedad constituye una prueba de la terminación del proceso de inferencia en la PLRO. Esta es una característica indispensable en todo sistema argumentativo. Observemos que sin embargo los árboles de dialéctica sin control de falacias (definición 4.23) no son necesariamente finitos, dado que pueden existir en los mismos una o más líneas de argumentación circulares.

6.3.2. Etiquetado de argumentos

Los árboles de dialéctica aceptables nos permiten realizar una primera clasificación de argumentos al decidir cuando un argumento está o no garantizado. Internamente, un árbol de dialéctica también clasifica a los argumentos que intervienen en su análisis como derrotados y no derrotados (`NodosD` y `NodosU` respectivamente). Sin embargo, esta clasificación de los argumentos es *local* a la construcción de un determinado árbol, a diferencia de la distinción entre argumentos garantizados y no garantizados que es *global* a un determinado programa. Resulta interesante determinar si existe una relación entre la partición entre `NodosD` y `NodosU` realizada en un árbol en particular, y el hecho de que un argumento esté garantizado.

En primer lugar veremos qué significa que un determinado argumento $\mathcal{A}$ esté marcado como no derrotado en un árbol de dialéctica. De acuerdo a la definición 4.24 un argumento es un `NodoU` si no posee derrotadores marcados como no derrotados. Esto puedo conducirnos a concluir que entonces $\mathcal{A}$ no posee derrotadores aceptables y por lo tanto está garantizado. Sin embargo, esta propiedad no se cumple: que un argumento $\mathcal{A}$ sea etiquetado como no derrotado en un árbol dialéctico no implica que $\mathcal{A}$ sea un argumento garantizado.

Ejemplo 6.4 Consideremos a $\mathcal{P} = (\Psi, \Delta)$ donde $\Psi = \{\texttt{f, c, d, e}\}$ y

$$\Delta = \{\texttt{a} \prec \texttt{b}; \texttt{b} \prec \texttt{f}; \sim\texttt{a} \prec \texttt{b,c}; \texttt{a} \prec \texttt{b,c,d}; \sim\texttt{b} \prec \texttt{e}\}$$

en base a $\mathcal{P}$ se generan los argumentos $\mathcal{A}$,$\mathcal{B}$,$\mathcal{C}$,$\mathcal{D}$ donde

- $\mathcal{A} = \{\texttt{a} \prec \texttt{b}; \texttt{b} \prec \texttt{f}\}$
- $\mathcal{B} = \{\sim\texttt{a} \prec \texttt{b,c}; \texttt{b} \prec \texttt{f}\}$
- $\mathcal{C} = \{\texttt{a} \prec \texttt{b,c,d}; \texttt{b} \prec \texttt{f}\}$
- $\mathcal{D} = \{\sim\texttt{b} \prec \texttt{e}\}$

A partir de estos argumentos se construye el árbol de dialéctica $\mathcal{T}_{\langle \mathcal{A}, a \rangle}$ que consiste de una única línea argumentativa que comienza en $\mathcal{A}$ que luego es derrotado por $\mathcal{B}$ el cual a su vez es derrotado por $\mathcal{C}$ (ver figura 6.1). En $\mathcal{T}_{\langle \mathcal{A}, a \rangle}$ $\mathcal{C}$ está etiquetado como un argumento no derrotado, dado que $\mathcal{D}$ no puede ingresar como derrotador de $\mathcal{C}$ por ser contradictorio con $\mathcal{B}$.
Por otra parte existe sólo un árbol dialéctico para $\mathcal{C}$ (ver figura 6.1) formado por la raíz $\mathcal{C}$ que posee un único derrotador $\mathcal{D}$. Por lo tanto $\mathcal{C}$ no está garantizado. ■

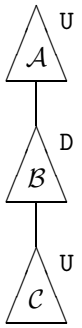


Figura 6.1: Primer árbol dialéctico del ejemplo 6.4.

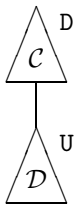


Figura 6.2: Segundo árbol dialéctico del ejemplo 6.4.

La razón detrás de este comportamiento se origina al evitar las situaciones falaces por medio de las líneas de dialéctica aceptables. En el primer árbol dialéctico el argumento $\mathcal{D}$ no puede introducirse como derrotador de $\mathcal{C}$, por no ser concordante con $\mathcal{B}$. No obstante esta característica del sistema no es arbitraria. Un árbol de dialéctica representa el *contexto* en el que se lleva a cabo el debate introspectivo entre el proponente y el oponente. En un determinado contexto puede suceder que un argumento no pueda ser introducido en función de compromisos tomados anteriormente. Esta situaciones son reguladas por el control de falacias. Resulta por lo tanto natural que el conjunto de derrotadores válidos para un argumento cambie dependiendo del contexto, esto es, del árbol dialéctico en consideración.

Siguiendo este mismo razonamiento, tampoco es posible afirmar que un argumento garantizado debe ser etiquetado como `NodoU` en todo árbol dialéctico. Esto se ejemplifica en el siguiente escenario.

Ejemplo 6.5 Consideremos a $\mathcal{P} = (\Psi, \Delta)$ donde $\Psi = \{\texttt{b, c, d, f}\}$ y

$$\Delta = \{\texttt{a} \prec \texttt{b,c}; \sim\texttt{a} \prec \texttt{b,c,d}; \sim\texttt{a} \prec \texttt{c}; \texttt{a} \prec \texttt{e}; \texttt{e} \prec \texttt{f}; \sim\texttt{e} \prec \texttt{a}; \texttt{a} \prec \texttt{b}\}$$

Figura 6.3: Primer árbol dialéctico del ejemplo 6.5.

en base a $\mathcal{P}$ se generan los argumentos $\mathcal{A}$,$\mathcal{B}$,$\mathcal{X}$,$\mathcal{Y}$,$\mathcal{Z}$ donde

- $\mathcal{A} = \{\sim\mathtt{e} \prec \mathtt{a}; \mathtt{a} \prec \mathtt{b}\}$
- $\mathcal{B} = \{\sim\mathtt{a} \prec \mathtt{c}\}$
- $\mathcal{X} = \{\mathtt{a} \prec \mathtt{b},\mathtt{c}\}$
- $\mathcal{Y} = \{\sim\mathtt{a} \prec \mathtt{b},\mathtt{c},\mathtt{d}\}$
- $\mathcal{Z} = \{\mathtt{a} \prec \mathtt{e}; \mathtt{e} \prec \mathtt{f}\}$

En consecuencia existe un árbol de dialéctica $\mathcal{T}_{\langle\mathcal{X},a\rangle}$ en el cual $\mathcal{X}$ es derrotado por $\mathcal{Y}$ el cual a su vez es derrotado por $\mathcal{Z}$ (ver figura 6.3). Entonces el argumento $\mathcal{X}$ está garantizado.
Además existe un árbol dialéctico para $\mathcal{A}$ en el cual $\mathcal{X}$ está etiquetado como derrotado. Este árbol está compuesto por $\mathcal{A}$ en la raíz, $\mathcal{B}$ como derrotador de $\mathcal{A}$, $\mathcal{X}$ como derrotador de $\mathcal{B}$ e $\mathcal{Y}$ como derrotador de $\mathcal{X}$ (ver figura 6.4). El argumento $\mathcal{Z}$ no puede utilizarse para sustentar $\mathcal{X}$ por no ser concordante con $\mathcal{A}$. ■

El análisis precedente ilustra el hecho de que si un nodo es marcado como no derrotado en un árbol dialéctico esto no necesariamente implica que posea esta misma etiqueta en todo árbol dialéctico en el cual aparezca.

6.4. Clasificación de argumentos

En la sección anterior establecimos que no es posible clasificar a los argumentos en derrotados y no derrotados de acuerdo a la etiqueta que poseen en un árbol de

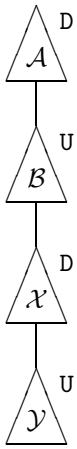


Figura 6.4: Segundo árbol dialéctico del ejemplo 6.5.

dialéctica. No obstante, el mecanismo de inferencia de la PLRO permite particionar los argumentos del sistema en dos clases bien diferenciadas: los que están garantizados y los que no lo están. Esto resulta en una primer clasificación global de los argumentos de un programa.

Definición 6.8 (*Argumentos garantizados – Argumentos denegados*)
Un argumento $\mathcal{A}$ está *garantizado* con respecto a un programa $\mathcal{P}$ si y sólo si es la raíz de un árbol dialéctico en el cual está etiquetado como `NodoU`. En caso contrario $\mathcal{A}$ es un argumento *denegado*. ■

Es interesante recordar que los argumentos garantizados son los que respaldan a las conclusiones del sistema. En este contexto resulta natural pensar que este conjunto de argumentos no debe poseer conflictos entre sí. Este conjunto de argumentos representa las posiciones para las que el sistema arribó a un acuerdo. Por lo tanto no sería tolerable afirmar que se aceptó un par de argumentos $\mathcal{A}$ y $\mathcal{B}$ cuando este par es un elemento de la relación de contraargumentación.

Para demostrar formalmente esta propiedad en la PLRO utilizaremos un par de resultados auxiliares que nos conducirán finalmente hacia el lema 6.1.

Proposición 6.11 Sea $\mathcal{P}$ un programa y $\langle \mathcal{A}, q \rangle$, $\langle \mathcal{B}, p \rangle$ dos argumentos construidos a partir de $\mathcal{P}$ tal que $\langle \mathcal{A}, q \rangle$ es un derrotador de $\langle \mathcal{B}, p \rangle$ y $\langle \mathcal{A}, q \rangle$ está garantizado con respecto a $\mathcal{P}$. Sea $\langle \mathcal{A}_s, q_s \rangle$ un argumento de soporte de $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ tal que $\langle \mathcal{A}_s, q_s \rangle$ es un `nodo U`. Si $\langle \mathcal{A}_s, q_s \rangle$ aparece en $\mathcal{T}_{\langle \mathcal{B},p \rangle}$ entonces también debe estar marcado como `nodo U` en $\mathcal{T}_{\langle \mathcal{B},p \rangle}$. ■

Demostración: Para demostrar esta propiedad usaremos inducción estructural sobre el subárbol con raíz en $\langle \mathcal{A}_s, q_s \rangle$. Para el caso base consideremos que $\langle \mathcal{A}_s, q_s \rangle$ no posee derrotadores en $\mathcal{T}_{\langle \mathcal{A}_1, q_1 \rangle}$. Si $\mathcal{A}_s$ pertenece a $\mathcal{T}_{\langle \mathcal{A}_2, q_2 \rangle}$ debe estar marcado como `nodo U`, dado que no es posible construir nuevos derrotadores.

En el paso inductivo asumamos que $\langle \mathcal{A}_s, q_s \rangle$ posee k derrotadores en $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$, denotados como $\mathcal{B}_1, \mathcal{B}_2 \ldots, \mathcal{B}_k$, y que $\langle \mathcal{A}_s, q_s \rangle$ está presente en $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$. Como $\mathcal{A}_s$ es un `nodo U` en $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$ cada uno de los derrotadores $\mathcal{B}_i$ debe ser un `nodo D` en $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$. A continuación demostraremos que si $\mathcal{B}_i$ está presente en $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$ debe estar marcado como un `nodo D` en este árbol. Dado que $\mathcal{B}_i$ es un `nodo D` en $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$ debe existir un argumento de soporte no derrotado $\mathcal{C}$ en el subárbol con raíz en $\mathcal{B}_i$. Si usamos la hipótesis inductiva en este subárbol es claro que si $\mathcal{C}$ está presente en $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$ debe también estar marcado como un *nodo U* y entonces $\mathcal{B}_i$ debe ser un `nodo D` en $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$. Resta verificar que sucede en el caso que $\mathcal{C}$ no pueda ser introducido como un derrotador de $\mathcal{B}_i$ en $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$. Esto puede suceder debido a alguna de las siguientes razones:

1. $\mathcal{C}$ es contradictorio con algún argumento de interferencia presente en la línea de argumentación en la cual se pretende agregar a $\mathcal{C}$.

2. $\mathcal{C}$ es un subargumento de un alguno de los argumentos de esta línea de argumentación.

La primer situación no es posible, dado que en este caso $\mathcal{C}$ no podría aparecer en $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$ por ser contradictorio con un argumento de soporte de la línea de argumentación.

En el segundo caso, $\mathcal{C}$ debe ser un subargumento de $\mathcal{B}$ (de lo contrario no es posible que pertenezca a $\mathcal{T}_{\langle \mathcal{A}, q \rangle}$). Sin embargo, como $\mathcal{C}$ es un derrotador de $\mathcal{B}_i$, el conjunto $\mathcal{C} \cup \mathcal{B}_i \cup \Psi$ permite la derivación de literales complementarios. Entonces $\mathcal{B} \cup \mathcal{B}_i \cup \Psi$ también permite obtener dos literales complementarios ($\mathcal{C} \subseteq \mathcal{B}$). En esta situación no es posible que $\mathcal{B}_i$ esté presente en $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$, dado que contradice a un argumento de soporte de su línea argumentativa.

Consecuentemente si $\mathcal{B}_i$ está presente $\mathcal{T}_{\langle \mathcal{B}, p \rangle}$ debe ser estar marcado como un `nodo U` en este árbol, como queríamos demostrar. □

La siguiente propiedad se desprende como un corolario de la proposición anterior.

Corolario 6.2 Sea $\mathcal{P}$ un programa y $\langle \mathcal{A}, q \rangle$, $\langle \mathcal{B}, p \rangle$ dos argumentos construidos a partir de $\mathcal{P}$. Si $\langle \mathcal{A}, q \rangle$ es un derrotador de $\langle \mathcal{B}, p \rangle$ y $\langle \mathcal{A}, q \rangle$ es un argumento garantizado

con respecto a $\mathcal{P}$ entonces $\langle \mathcal{A}, q \rangle$ debe estar marcado como un `nodo U` en el árbol dialéctico para $\langle \mathcal{B}, p \rangle$. ■

Demostración: $\langle \mathcal{A}, q \rangle$ es un argumento de soporte no derrotado en $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ que esta presente en $\mathcal{T}_{\langle \mathcal{A}_2,q_2 \rangle}$. Aplicando la proposición 6.11 podemos concluir que $\langle \mathcal{A}, q \rangle$ debe ser un `nodo U` en el árbol dialéctico para $\langle \mathcal{B}, p \rangle$. □

Lema 6.1 Sea Warr_P el conjunto de argumentos garantizados de un programa $\mathcal{P}=(\Psi, \Delta)$. Para cualquier par de argumentos $\mathcal{A}$, $\mathcal{B}$ tal que $\mathcal{A} \in \text{Warr}_P$ y $\mathcal{B} \in \text{Warr}_P$ se cumple que el par ordenado $(\mathcal{A}, \mathcal{B})$ no pertenece a $C_{(\Psi,\Delta)}$. ■

Demostración: Supongamos por el absurdo que existe un par de argumentos $(\mathcal{A}, \mathcal{B})$ en $C_{(\Psi,\Delta)}$ con estas condiciones. En este caso podemos asumir sin pérdida de generalidad que la contraargumentación entre $\mathcal{A}$ y $\mathcal{B}$ es recíproca (si solamente $\mathcal{A}$ contraargumenta a $\mathcal{B}$ existe un par $(\mathcal{A}, \mathcal{B}')$ cuya argumentación es recíproca) y que $\mathcal{A}$ derrota a $\mathcal{B}$. Como $\mathcal{A}$ y $\mathcal{B}$ están garantizados podemos asumir que existe un árbol dialéctico con $\mathcal{A}$ como raíz, denominado $\mathcal{T}_{\langle \mathcal{A},q \rangle}$ que garantiza a q. Lo mismo se cumple para el argumento $\mathcal{B}$.

Como $\mathcal{A}$ derrota a $\mathcal{B}$, $\mathcal{A}$ debe estar marcado como `nodo D` en el árbol diaéctico asociado a $\mathcal{B}$, $\mathcal{T}_{\langle \mathcal{B},p \rangle}$, ya que en caso contrario $\mathcal{B}$ no sería un argumento garantizado. Sin embargo, si aplicamos el corolario 6.2 por estar $\mathcal{A}$ garantizado y ser $\mathcal{B}$ un derrotador de $\mathcal{A}$ entonces $\mathcal{A}$ debe estar marcado como un `nodo U` en el árbol dialéctico para $\langle \mathcal{B}, p \rangle$. Esto constituye un absurdo que proviene de suponer que la existencia del par de argumentos $(\mathcal{A}, \mathcal{B})$. □

Esta primer partición, que distingue los argumentos aceptados por el sistema de los argumentos rechazados, puede realizarse en cualquier sistema argumentativo. Por lo tanto, la proposición 6.1 es una propiedad recomendable para los formalismos argumentativos en general que resume una forma de coherencia del mecanismo de inferencia.

Como corolario de esta propiedad se desprende la consistencia de las conclusiones del sistema. Recordemos que las inferencias de la PLRO corresponden a los literales sustentados por argumentos garantizados. Esto resulta de gran importancia si deseamos usar a la PLRO para representar el conocimiento de un agente en forma adecuada. Un individuo racional no puede creer al mismo tiempo en un literal q y su complemento $\overline{q}$. Para cualquier programa $\mathcal{P}$ se cumple que si un literal q está garantizado, entonces $\overline{q}$ no puede estarlo.

Lema 6.2 El conjunto de literales garantizados en base a un programa $\mathcal{P}$ de la PLRO es un conjunto no contradictorio. ■

Demostración: Supongamos que existe un par de literales complementarios q y $\overline{q}$ que están garantizados en base a $\mathcal{P}$. Entonces existe un par de argumentos garantizados $\langle \mathcal{A}, q \rangle$ y $\langle \mathcal{B}, \overline{q} \rangle$. Luego el par $(\mathcal{A}, \mathcal{B})$ pertenece a $C_{(\Psi,\Delta)}$. Esto constituye un absurdo que contradice al lema 6.1. □

En consecuencia un razonador que usa un programa $\mathcal{P}$ de la PLRO para representar su conocimiento puede sostener una de estas tres posturas epistémicas para un literal q en la signatura del programa $\mathcal{P}$:

- Creer que q es verdadero (si q es un literal garantizado con respecto a $\mathcal{P}$)
- Creer que q es falso (si $\overline{q}$ es un literal garantizado con respecto a $\mathcal{P}$)
- No creer en la veracidad ni en la falsedad de q (en cualquier otro caso).

Es por tanto coherente definir el conjunto de creencias del agente como los literales garantizados con respecto al programa $\mathcal{P}$ que codifica su conocimiento.

Otra de las propiedades del conjunto de argumentos garantizados se vincula con la relación de subargumentación. Aceptar un argumento como garantizado implica en forma implícita aceptar a todos sus subargumentos.

Proposición 6.12 Si un argumento $\mathcal{A}$ está garantizado entonces todos sus subargumentos lo están. ■

Demostración: Sea $\mathcal{B}$ un subargumento de $\mathcal{A}$ y supongamos por el absurdo que $\mathcal{B}$ no es un argumento garantizado. Para demostrar que esto no es posible analizaremos esta situación por casos, dependiendo de la cantidad de derrotadores que posee $\mathcal{A}$.

1. $\mathcal{A}$ no posee derrotadores: en este caso $\mathcal{B}$ tampoco posee derrotadores, dado que todo derrotador de $\mathcal{B}$ es derrotador de $\mathcal{A}$. En consecuencia $\mathcal{B}$ es un argumento garantizado.

2. $\mathcal{A}$ posee uno o más derrotadores: en este caso $\mathcal{B}$ puede o no tener uno o más derrotadores. Si $\mathcal{B}$ no posee derrotadores esta situación es equivalente a la analizada en el caso anterior. Si $\mathcal{B}$ posee uno o más derrotadores, sea $\mathcal{C}$ un derrotador arbitrario de $\mathcal{B}$. Podemos afirmar que $\mathcal{C}$ está derrotado en el árbol

dialéctico $\mathcal{T}_{\langle \mathcal{A},p \rangle}$ por ser $\mathcal{A}$ un argumento garantizado. Por lo tanto existe un derrotador $\mathcal{D}$ de $\mathcal{C}$ que es a su vez concordante con $\mathcal{A}$ y está marcado como `nodo U` en $\mathcal{T}_{\langle \mathcal{A},p \rangle}$. Analicemos entonces que sucede con $\mathcal{C}$ en el árbol dialéctico para $\mathcal{B}$, $\mathcal{T}_{\langle \mathcal{B},q \rangle}$.

Como $\mathcal{D}$ derrota a $\mathcal{C}$ y además está marcado como `nodo U` en $\mathcal{T}_{\langle \mathcal{A},p \rangle}$ existen sólo dos alternativas para que $\mathcal{C}$ sea un `nodo U` y pueda así derrotar a $\mathcal{B}$. Una alternativa consistiría en que algún argumento de soporte de $\mathcal{A}$ o el mismo $\mathcal{D}$ no puedan ser introducidos por no ser concordantes con la línea argumentativa. Pero esto no es posible porque sabemos que estos argumentos son concordantes con $\mathcal{A}$ (por estar presentes en $\mathcal{T}_{\langle A,q \rangle}$) y por lo tanto deben ser concordantes con $\mathcal{B}$. La otra posibilidad sería que algún argumento de soporte de $\mathcal{B}$ no pueda ser introducido por ser circular con algún argumento previo en la línea argumentativa. Pero esto tampoco es posible dado que si no es circular con $\mathcal{A}$ tampoco puede ser circular con $\mathcal{B}$, ya que $\mathcal{B} \subseteq \mathcal{A}$. En consecuencia $\mathcal{B}$ debe ser un argumento garantizado.

□

Dentro del conjunto de argumentos garantizados pueden a su vez distinguirse otras categorías, tendientes a cuantificar el grado de certeza con que el sistema acepta un determinado argumento. Con este objetivo propondremos la siguiente partición:

- Observaciones: si la conclusión del argumento $\mathcal{A}$ es una observación perteneciente al conjunto Ψ, entonces $\mathcal{A}$ es el argumento vacío. Por lo tanto $\mathcal{A}$ no posee puntos de ataque y su grado de fuerza conclusiva es máximo.[5]

- Sin conflictos: en este caso el argumento $\mathcal{A}$ no posee conflictos con ninguno de los argumentos que se pueden construir a partir del programa. El árbol de dialéctica asociado a este argumento posee un único nodo.

- Sin derrotadores: esta clase agrupa los argumentos que resultaron victoriosos en todos los ataques, por medio del criterio de derrota del sistema. Su fuerza conclusiva es menor que la de los argumentos sin conflictos porque dependen del criterio de derrota.

[5]El grado de fuerza conclusiva de un argumento depende de la cantidad de derrotadores que posee. A mayor cantidad de derrotadores, menor es el grado de fuerza conclusiva

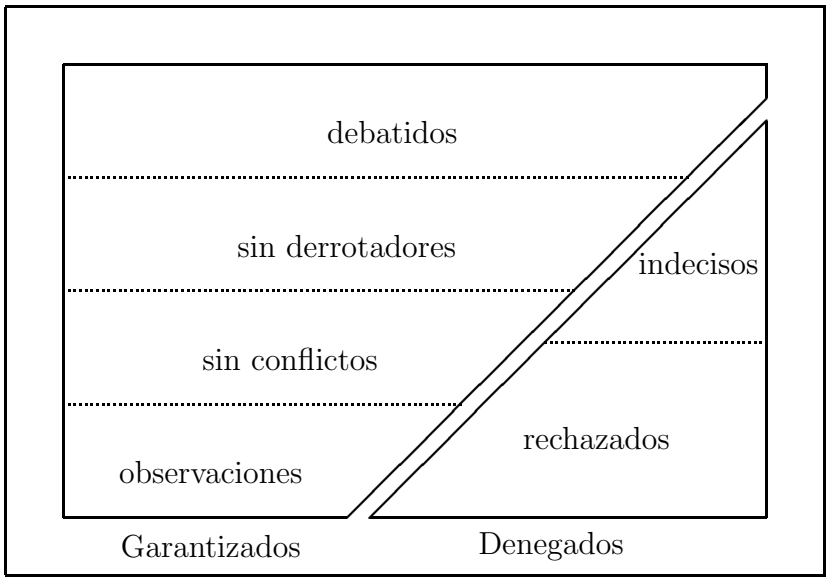


Figura 6.5: Partición del universo de argumentos

- Debatidos: son los argumentos que poseen derrotadores, pero que igualmente resultaron garantizados en virtud de otros argumentos de soporte que vencieron a sus derrotadores. Como su obtención involucró un debate en varios niveles son los que menor fuerza poseen.

Esta clasificación propuesta para la PLRO puede extenderse prácticamente a cualquier sistema argumentativo. Resulta además una forma sencilla de distinguir la fuerza conclusiva de los argumentos, con respecto a cuán debatida fue su inclusión en el conjunto de literales garantizados. Esta partición de argumentos será utilizada al estudiar las propiedades del proceso de inferencia en la PLRO.

Finalmente los argumentos denegados pueden también subdividirse en dos categorías dependiendo de la seguridad con que fueron rechazados por el sistema. Es posible distinguir a los argumentos *indecisos*, que sólo están derrotados por argumentos de bloqueo y a los argumentos *rechazados* que fueron derrotados en forma propia. En este caso los argumentos indecisos tuvieron más posibilidades de ser aceptados por el sistema. La figura 6.5 resume la categorización de argumentos presentada en esta sección.

6.5. Propiedades del mecanismo de inferencia

Las propiedades del mecanismo de inferencia en los formalismos no monótonos comenzaron a analizarse a partir del trabajo realizado por D. Gabbay [Gabbay, 1985]. En este artículo Gabbay propone concentrarse en la relación existente entre una determinada conclusión y las premisas que permitieron obtenerla. Por medio de este acercamiento, el trabajo realizado en [Kraus et al., 1990] y [Makinson, 1994] establece un conjunto de propiedades fundacionales para las teorías no monótonas.

En esta sección detallaremos las propiedades presentadas por Gabbay y Makinson, a fin de analizar si éstas pueden aplicarse al estudio de los sistemas argumentativos. A continuación usaremos como caso de estudio a la PLRO. Detallaremos la postura de este sistema con respecto a las propiedades de los sistemas no monótonos y propondremos nuevas alternativas para analizar el comportamiento de los sistemas argumentativos.

6.5.1. Propiedades de los formalismos no monótonos

El conjunto de propiedades propuestas para estudiar la inferencia en los sistemas no monótonos puede dividirse en las propiedades *puras*, que dependen exclusivamente del mecanismo de inferencia, y las propiedades que interactúan con los conectivos lógicos del lenguaje.

Como el lenguaje subyacente de la PLRO está basado en la programación en lógica, no posee muchos de los conectivos lógicos tradicionales (como la conjunción, la disyunción y la implicación). Por lo tanto la mayoría de las propiedades propuestas sobre los conectivos lógicos no son extensibles a la PLRO. Dado que nuestro objetivo al realizar este análisis es el estudio de las propiedades de este formalismo, nos concentraremos solamente en el conjunto de propiedades puras.

Sea $\vdash$ una relación de inferencia cualquiera. Entonces su operador de inferencia asociado, denotado C, se define como sigue:

$$C(\Phi) = \{\phi \mid \Phi \vdash \phi\}$$

En términos de este operador general y dos conjuntos de premisas Φ y Υ se definen las propiedades puras.

Inclusión: $\Phi \subseteq C(\Phi)$

Idempotencia: $C(\Phi) = C(C(\Phi))$

Cut: $\Phi \subseteq \Upsilon \subseteq C(\Phi)$ implica que $C(\Phi) \subseteq C(\Upsilon)$

Monotonía cauta: $\Phi \subseteq \Upsilon \subseteq C(\Phi)$ implica que $C(\Upsilon) \subseteq C(\Phi)$

Cumulatividad: $\Phi \subseteq \Upsilon \subseteq C(\Phi)$ implica que $C(\Upsilon) = C(\Phi)$

La propiedad de inclusión es claramente deseable en cualquier relación de inferencia, porque las premisas son la base del razonamiento y deben por tanto ser aceptadas en forma incuestionable. Idempotencia postula que una sola aplicación del operador de inferencia es suficiente para inferir todas las conclusiones posibles a partir de las premisas. No es posible hallar nuevas conclusiones a partir de la aplicación repetida del operador C.

Cut nos asegura que el agregado de nuevas consecuencias al conjunto de premisas no permite hallar nuevas conclusiones. En cambio la monotonía cauta propone que agregar consecuencias al conjunto de premisas no disminuye las conclusiones obtenidas. Finalmente la cumulatividad es la conjunción de cut y monotonía cauta. Una relación de inferencia que satisface estas propiedades se denomina *cumulativa*.

Algunos autores han manifestado que toda teoría no monótona debería estar basada en una relación de inferencia cumulativa [Kraus et al., 1990]. Sin embargo, esta postura resulta a veces demasiado restrictiva. Existen algunos sistemas no monótonos coherentes y correctamente definidos (como muchos de los sistemas argumentativos) para los cuales la cumulatividad no esta entre sus propiedades. A continuación estudiaremos que sucede en el caso de la PLRO. Cabe destacar que un acercamiento similar ya fue utilizado en [Chesñevar, 2001] para el estudio de las propiedades de la inferencia en su sistema deductivo etiquetado para argumentación rebatible.

6.5.2. Análisis de las propiedades en la PLRO

El propósito primordial de un programa de la PLRO es obtener un conjunto de inferencias. La noción de argumentos garantizados es la que realiza esta tarea, distinguiendo aquellos argumentos que pueden garantizar su conclusión. A continuación definiremos formalmente las inferencias de un programa dado.

Sea $\mathcal{P} = (\Psi, \Delta)$ un programa en la PLRO y sea Warr_P el conjunto de argumentos garantizados a partir de $\mathcal{P}$. Las inferencias obtenidas a partir del programa $\mathcal{P}$ se definen como las conclusiones de los argumentos en Warr_P.

Definición 6.9 (*Inferencia en la PLRO*)
Dado un programa $\mathcal{P}$=(Ψ, Δ) una conclusión ϕ es una inferencia de $\mathcal{P}$, denotado como $\Psi \ \sim_\Delta \phi$ si y sólo si ϕ es la conclusión de un argumento en Warr_P. ■

Observemos que las premisas del mecanismo de inferencia son solamente las observaciones del programa, dado que Δ juega el rol de un conjunto de reglas de inferencia con respecto al mecanismo de derivación. Finalmente el operador de consecuencia para $\sim_\Delta$ se define como $C_\Delta(\Psi) = \{\phi \mid \Psi \sim_\Delta \phi\}$. En base a este operador de consecuencia estudiaremos las propiedades de inclusión, idempotencia y cumulatividad.

Comenzaremos por analizar la propiedad de inclusión mediante la siguiente proposición.

Proposición 6.13 Para cualquier programa $\mathcal{P} = (\Psi, \Delta)$, el operador de consecuencia C_Δ satisface inclusión, *i.e.* $\Psi \subseteq C_\Delta(\Psi)$. ■

Demostración: Sea $\phi \in \Psi$. Entonces ϕ es la conclusión del argumento vacío (que pertenece a Warr_P por no tener derrotadores). Por lo tanto ϕ pertenece a $C_\Delta(\Psi)$. □

Con respecto a la idempotencia, podemos afirmar que el operador de consecuencia de la PLRO no cumple esta propiedad. Para fundamentar esta afirmación veamos el siguiente contraejemplo:

Ejemplo 6.6 Sea $\mathcal{P} = (\Psi, \Delta)$, donde $\Psi = \{\texttt{a}\}$ y $\Delta = \{\texttt{b} \prec \texttt{a}; \sim\texttt{b} \prec \texttt{a}\}$. Observemos que en esta situación $C_\Delta(\{\texttt{a}\}) = \{\texttt{b}; \sim\texttt{b}\} \neq \emptyset = C_\Delta(C_\Delta(\{\texttt{a}\}))$. ■

Finalmente, para mostrar que el operador de consecuencia de la PLRO no es cumulativo utilizaremos dos contraejemplos. El primero de ellos muestra por qué falla cut y el segundo por qué no se cumple la propiedad de monotonía cauta.

Ejemplo 6.7 Sea $\mathcal{P} = (\Psi, \Delta)$, $\Psi = \{\texttt{e}, \texttt{c}\}$ y $\Delta = \{\sim\texttt{a} \prec \texttt{d}; \texttt{d} \prec \texttt{e}; \texttt{a} \prec \texttt{b}; \texttt{b} \prec \texttt{c}\}$. Supongamos además que el criterio de derrota utilizado en este escenario se basa en el número de reglas del sistema (recordemos que el criterio de derrota utilizado en la PLRO se encuentra parametrizado). Un argumento $\mathcal{A}$ derrota a un argumento $\mathcal{B}$ si posee mayor o igual número de reglas.
En este ejemplo es posible construir los argumentos $\mathcal{A}_1 = \{\sim\texttt{a} \prec \texttt{d}; \texttt{d} \prec \texttt{e}\}$ para $\sim\texttt{a}$ y $\mathcal{A}_2 = \{\texttt{a} \prec \texttt{b}; \texttt{b} \prec \texttt{c}\}$ para a. En este caso $\mathcal{A}_1$ y $\mathcal{A}_2$ están relacionados mediante una derrota por bloqueo y por tanto $\mathcal{A}_1$ y $\mathcal{A}_2$ no forman parte de las conclusiones del sistema. Sin embargo la conclusión $\texttt{d}$ está sustentada por un argumento garantizado.

Supongamos que se agregara d al conjunto de premisas (observaciones). De acuerdo al nuevo conjunto de observaciones podríamos obtener el argumento $\mathcal{A}_1' = \{\sim\mathtt{a} \prec \mathtt{d}\}$ para ~a. De acuerdo al criterio de derrota $\mathcal{A}_1'$ derrota a $\mathcal{A}_2$ y por lo tanto ~a pasa a formar parte de las conclusiones del sistema. ■

Al demostrar que no se cumple la propiedad de monotonía cauta podemos afirmar que el sistema no es cumulativo. Para analizar si se cumple la propiedad de cut hemos propuesto el siguiente ejemplo:

Ejemplo 6.8 Sea $\mathcal{P} = (\Psi, \Delta)$ un programa en la PLRO donde $\Psi = \{\mathtt{a}\}$ y $\Delta = \{\mathtt{c} \prec \mathtt{b};\ \mathtt{b} \prec \mathtt{a}; \sim\mathtt{c} \prec \mathtt{a}\}$. Los argumentos $\mathcal{A}_1 = \{\mathtt{c} \prec \mathtt{b};\ \mathtt{b} \prec \mathtt{a}\}$, $\mathcal{A}_2 = \{\sim\mathtt{c} \prec \mathtt{a}\}$, y $\mathcal{A}_3 = \{\mathtt{b} \prec \mathtt{a}\}$, con conclusiones c, ~c, y b respectivamente, pueden construirse a partir de $\mathcal{P}$. Sin embargo, sólo ~c y b están garantizados en base a $\mathcal{P}$, dado que $\mathcal{A}_2$ derrota a $\mathcal{A}_1$, pero tanto $\mathcal{A}_2$ como $\mathcal{A}_3$ permanecen sin derrotar. Finalmente, si agregaramos b como una nueva observación en Ψ, el nuevo argumento $\mathcal{A}_4 = \{\mathtt{c} \prec \mathtt{b}\}$ para c derrota ahora a $\mathcal{A}_2$. En consecuencia, como $\mathcal{A}_2$ era el único argumento para ~c, esta conclusión desaparece del conjunto de inferencias. ■

En diversos análisis sobre los sistemas argumentativos y su relación con la cumulatividad (*e.g.* [Prakken y Sartor, 1997, Stankevicius et al., 2002]) se expresó que esta propiedad no se cumple debido a la existencia de distintos niveles de inferencia. Por lo tanto, al introducir una conclusión como premisa del mecanismo de inferencia estamos violando la jerarquía existente por la cual una inferencia no necesariamente posee el mismo nivel de certeza que una premisa. Esto provoca cambios en las conclusiones del sistema.

En efecto, en la PLRO no todas las inferencias poseen el mismo grado de certeza. Propondremos entonces una opción para categorizar las inferencias de acuerdo a su grado de certeza que consiste en clasificarlas de acuerdo al argumento que las sustenta. En la sección anterior los argumentos garantizados fueron divididos en argumentos debatidos, sin derrotadores, sin conflictos y observaciones. De esta misma forma clasificaremos las conclusiones del sistema en las siguientes categorías.

- Observaciones: si pertenecen al conjunto Ψ (lo que es equivalente a decir que está sustentada por el argumento vacío).
- No conflictivas: cuando están sustentadas por un argumento sin conflictos.
- No derrotadas: si están sustentadas por un argumento sin derrotadores.

- Debatidas: cuando están sustentadas por un argumento debatido.

Una vez realizada esta clasificación es posible plantear versiones más restrictivas de cumulatividad. Podemos preguntarnos si esta propiedad se cumple cuando el conjunto de conclusiones que se pretende agregar como premisas está conformado exclusivamente por miembros de alguna de estas categorías.

En primer lugar analizaremos que sucede para las observaciones. Recordemos entonces la definición de cumulatividad:

$$\Phi \subseteq \Upsilon \subseteq C(\Phi) \text{ implica que } C(\Upsilon) = C(\Phi)$$

Supongamos ahora que $\Upsilon \subseteq \Psi$ (el conjunto de observaciones del programa). En este caso es trivial verificar que se cumple la propiedad de cumulatividad.

Veamos ahora que ocurre si Υ está incluido en el conjunto de creencias no conflictivas. En este caso analizaremos que cambios ocurren en las inferencias del sistema al introducir estas conclusiones como premisas.

Previamente a este análisis categorizaremos los tipos de cambios que pueden ocurrir al agregar un literal cualquiera al conjunto de observaciones Ψ de un programa $\mathcal{P}$=(Ψ, Δ), dividiendo los mismos en tres niveles diferentes.

- *Cambia el universo de argumentos:* esto es, el conjunto de argumentos que se puede construir a partir del programa $\mathcal{P}$ en cuestión, args($\mathcal{P}$), pierde algunos de sus miembros y/o adquiere nuevos elementos. Por ejemplo, consideremos un programa $\mathcal{P} = (\Psi, \Delta)$ donde $\Psi = \{\mathtt{a}\}$ y $\Delta = \{\mathtt{\sim d} \prec \mathtt{a}\}$. En este caso los argumentos $\mathcal{A} = \{\mathtt{\sim d} \prec \mathtt{a}\}$ y $\emptyset$ forman el universo de argumentos de $\mathcal{P}$. Si agregaramos el literal $\mathtt{d}$ a Ψ entonces el argumento $\mathcal{A}$ se elimina de args($\mathcal{P}$).

- *Cambia la relación de contraargumentación:* Se introducen o eliminan pares de $C_{(\Psi,\Delta)}$. Por ejemplo, consideremos un programa $\mathcal{P} = (\Psi, \Delta)$ donde $\Psi = \{\mathtt{b,c}\}$ y $\Delta = \{\mathtt{\sim a} \prec \mathtt{b}; \mathtt{a} \prec \mathtt{c}\}$. En este caso los argumentos $\mathcal{A} = \{\mathtt{\sim a} \prec \mathtt{b}\}$, $\mathcal{B} = \{\mathtt{a} \prec \mathtt{c}\}$ y $\emptyset$ forman el universo de argumentos de $\mathcal{P}$. Si agregaramos el literal $\mathtt{a}$ en Ψ entonces el argumento $\mathcal{A}$ se elimina de args($\mathcal{P}$) y por consiguiente los pares $(\mathcal{A}, \mathcal{B})$ y $(\mathcal{B}, \mathcal{A})$ se eliminan de $C_{(\Psi,\Delta)}$.

- *Cambia la relación de derrota:* Se introducen o eliminan pares de $D_{(\Psi,\Delta)}$. Por ejemplo, consideremos un programa $\mathcal{P} = (\Psi, \Delta)$ donde $\Psi = \{\mathtt{c}\}$ y $\Delta = \{\mathtt{a} \prec \mathtt{b}; \mathtt{b} \prec \mathtt{c}; \mathtt{\sim a} \prec \mathtt{c}\}$. En este caso los argumentos $\mathcal{A} = \{\mathtt{a} \prec \mathtt{b}; \mathtt{b} \prec \mathtt{c}\}$, $\mathcal{B} = \{\mathtt{\sim a} \prec \mathtt{c}\}$ y $\emptyset$ forman el universo de argumentos de $\mathcal{P}$. Si agregaramos el

literal b a Ψ entonces el argumento $\mathcal{A}$ se elimina de args($\mathcal{P}$) y en su lugar se introduce $\mathcal{A}' = \{\mathtt{a} \prec \mathtt{b}\}$. Entonces los pares $(\mathcal{A}', \mathcal{B})$ y $(\mathcal{B}, \mathcal{A}')$ se introducen en $D_{(\Psi,\Delta)}$ y el par $(\mathcal{B}, \mathcal{A})$ se elimina de $D_{(\Psi,\Delta)}$.

¿Qué sucede en el caso que se introduzcan conclusiones no conflictivas? ¿Cuáles de estos tipos de cambios se producen? Podemos afirmar que ante el agregado una conclusión q con estas características los argumentos que se pueden construir a partir del programa $\mathcal{P}$ en cuestión cambian. Por ejemplo, el argumento para q no existe ya que ahora q está sustentado por el argumento vacío.

Sin embargo, como el argumento para q no poseía conflictos, para toda conclusión p que estaba sustentada por un argumento $\mathcal{B}$ seguirá existiendo un argumento que la respalde, ya sea el mismo $\mathcal{B}$ o un argumento $\mathcal{B}'$ tal que las reglas de $\mathcal{B}'$ están incluidas en las de $\mathcal{B}$. Esto se evidencia en el siguiente ejemplo.

Ejemplo 6.9 Consideremos el siguiente programa en la PLRO:

$$\mathtt{p}$$
$$\mathtt{q} \prec \mathtt{s}$$
$$\mathtt{s} \prec \mathtt{p}$$

Supongamos que agregaramos s al conjunto de observaciones. Entonces el argumento para q formado por las reglas $\{\mathtt{q} \prec \mathtt{s}, \mathtt{s} \prec \mathtt{p}\}$ no podría generarse a partir del programa en cuestión, pero surgiría en su lugar un nuevo argumento $\{\mathtt{q} \prec \mathtt{s}\}$ que sustenta a q. ■

Con respecto a la relación de contraargumentación es posible asegurar que no cambiará mediante el agregado de una conclusión sin conflictos q, ya que q no es un punto de ataque entre argumentos. No existe ningún argumento que incluya $\sim q$ entre sus literales. No obstante la relación de derrota puede variar ante el agregado de q si se ve afectado el criterio de preferencia entre argumentos.

Ejemplo 6.10 Consideremos el siguiente programa en la PLRO:

$$\mathtt{b}$$
$$\mathtt{q} \prec \mathtt{b}$$
$$\mathtt{p} \prec \mathtt{q}$$
$$\sim\mathtt{p} \prec \mathtt{s}$$

Donde el criterio de preferencia utilizado está basado en la cantidad de reglas rebatibles de cada argumento, favoreciendo a los que poseen un encadenamiento más corto de reglas.
Supongamos que agregaramos el literal q al conjunto de observaciones. Entonces el argumento para p formado por las reglas $\{\mathtt{p} \prec \mathtt{q}, \mathtt{q} \prec \mathtt{b}\}$ no podría generarse a partir de el programa en cuestión y surgiría en su lugar un nuevo argumento para q, $\{\mathtt{p} \prec \mathtt{q}\}$. En un principio el argumento $\{\sim\mathtt{p} \prec \mathtt{s}\}$ para $\sim p$ vencía al argumento para p, pero a partir de la introducción de q esto ya no se cumple. ■

Por lo tanto para asegurar que la PLRO es cumulativa debemos imponer una restricción adicional sobre el criterio de preferencia.

Definición 6.10 (*Tolerancia*)
Sean $\mathcal{A}$ y $\mathcal{B}$ dos argumentos tales que $\mathcal{A}$ es preferido a $\mathcal{B}$ de acuerdo al criterio de inferencia C y supongamos que $\mathcal{A}$ es eliminado del conjunto de argumentos al agregar una conclusión sin conflictos q, introduciendo en reemplazo de $\mathcal{A}$ un argumento $\mathcal{A}'$. Si $\mathcal{A}'$ es siempre preferido a $\mathcal{B}$ diremos que el criterio C es *tolerante*. ■

Proposición 6.14 Sea $\mathcal{P}$ un programa de la PLRO. Si Υ es un conjunto de conclusiones sin conflictos y el criterio de preferencia utilizado es *tolerante* entonces $\Phi \subseteq \Upsilon \subseteq C(\Phi)$ implica que $C(\Upsilon) = C(\Phi)$. ■

Demostración: Para demostrar esta proposición analicemos en primer lugar el caso en el cual se agrega solamente una conclusión q. En esta situación podrían producirse tres clases de cambios:

- *cambios en el conjunto de argumentos:* para todo argumento que contenía a q como literal se eliminan las reglas que se utilizaban para obtener a q del mismo (dado que ahora q es una observación). Por lo tanto para toda conclusión p que estaba sustentada por un argumento $\mathcal{B}$ seguirá existiendo un argumento que la respalde, ya sea el mismo $\mathcal{B}$ o un argumento $\mathcal{B}'$ tal que las reglas de $\mathcal{B}'$ están incluidas en las de $\mathcal{B}$.

- *cambios en la relación de contraargumentación:* es posible asegurar que no cambiará mediante el agregado de una conclusión sin conflictos q, ya que q no es un punto de ataque entre argumentos. Ninguno de los pares de esta relación involucra a un argumento con q como conclusión.

- *cambios en la relación de derrota:* como no hay cambios en la relación de contraargumentación y el criterio de preferencia es tolerante, los cambios que se producen no son significativos. Cada par $(\mathcal{A}, \mathcal{B})$ donde $\mathcal{A}$ fue reemplazado por $\mathcal{A}'$ se convierte en un par $(\mathcal{A}', \mathcal{B})$.

Podemos por tanto afirmar que todo argumento $\mathcal{A}$ (o en su defecto el nuevo argumento $\mathcal{A}'$ que surgió para reemplazarlo) conservará el mismo etiquetado en los árboles de dialéctica existentes y por lo tanto no se eliminan ni se agregan conclusiones.

Como además no existen dependencias entre las conclusiones que se agregan este mismo razonamiento se puede utilizar para concluir que lo mismo sucederá si se agrega cualquier conjunto de conclusiones sin conflictos. □

Cuando se introducen conclusiones que pertenecen a la categoría sin derrotadores la situación es similar a la anterior. En este caso pueden también desaparecer argumentos para algunas conclusiones sin generarse nuevos argumentos para sustentarlas.

Ejemplo 6.11 Consideremos el siguiente programa en la PLRO:

$$\begin{array}{l} \mathtt{p} \\ \mathtt{q} \prec \mathtt{s} \\ \mathtt{s} \prec \mathtt{p} \\ \sim\mathtt{q} \prec \mathtt{p} \end{array}$$

En este caso el argumento $\mathcal{A} = \{\sim\mathtt{q} \prec \mathtt{p}\}$ para $\sim q$ no posee derrotadores pero posee un contraargumento $\{\mathtt{q} \prec \mathtt{s}, \mathtt{s} \prec \mathtt{p}\}$. En este caso si $\sim q$ se agregara al conjunto de conclusiones entonces no existiría ya ningún argumento para sustentar a q. ■

Finalmente en el caso de las conclusiones debatidas (última categoría) pueden cambiar tanto los argumentos como las relaciones de contraargumentación y derrota. En los ejemplos 6.7 y 6.8 se evidenció como estos cambios eliminan y agregan inferencias al sistema. Este es el nivel en el que se producen cambios de mayor importancia por ser estas creencias de un menor nivel de certeza y por lo tanto su intromisión como premisa no resulta correcta de acuerdo con la semántica del sistema.

En base a lo expuesto podemos concluir que las creencias de la PLRO poseen efectivamente distintos grados de certeza y de acuerdo a los mismos producen diferentes niveles de cambios. La cumulatividad por tanto no es adecuada para analizar

las características de la PLRO en particular, ni de los sistemas argumentativos que comparten las características analizadas. Sin embargo, se pueden plantear versiones alternativas de la misma, por ejemplo mediante la clasificación de las conclusiones del sistema, tal como se realizó en esta sección.

6.6. Conclusiones

A lo largo de este capítulo hemos estudiado las propiedades formales de la PLRO. A partir de este análisis hemos al mismo tiempo propuesto algunos elementos claves que permiten analizar a los sistemas argumentativos en forma abstracta en base a un conjunto de relaciones entre argumentos.

Este acercamiento nos ha permitido identificar una serie de propiedades interesantes para los formalismos de representación de conocimiento y razonamiento basados en argumentación. Las propiedades analizadas constituyen un estudio de las características del análisis dialéctico presente en gran cantidad de estos sistemas.

Si S, CC, C y D son las relaciones de subargumentación, concordancia, contraargumentación y derrota de un sistema argumentativo dado, se destacan en particular las siguientes condiciones.

- No es conveniente permitir la formación de argumentos auto derrotados dado que complican la definición del mecanismo de inferencia.

- La concordancia es la más general de las relaciones de acuerdo definidas. Es natural que un argumento sea concordante con cualquiera de sus subargumentos. Además, si no se permiten argumentos auto derrotados, debe satisfacerse que la relación de subargumentación esté incluída en la relación de concordancia, *i.e.*, $S \subseteq CC$.

- La relación de derrota es un refinamiento de la relación de ataque o contraargumentación entre argumentos, por lo tanto $C \subseteq D$.

- La relación de derrota depende fuertemente de la elección de un criterio adecuado de preferencia entre argumentos. Para garantizar que este criterio se comporte en la forma deseada basta con requerir un conjunto de restricciones adicionales.

- Las relaciones de acuerdo entre argumentos (subargumentación y concordancia) no deben poseer pares en común con las relaciones de conflicto (contraargumentación y derrota). Esto es, $CC \cap C = \emptyset$.
- Es fundamental garantizar la finitud del proceso de inferencia.
- El conjunto de argumentos garantizados debe ser libre de conflictos. Esto es, para cada par de argumentos $\mathcal{A}, \mathcal{B}$ que están garantizados debe valer que tanto $(\mathcal{A}, \mathcal{B})$ como $(\mathcal{B}, \mathcal{A})$ no pertenecen a la relación de contraargumentación asociada al programa.
- Las conclusiones del sistema deben ser consistentes.
- Si un argumento está garantizado entonces resulta razonable suponer que todos sus subargumentos deben estar garantizados. Algunos sistemas requieren esta característica en forma explícita, pero resulta más interesante que se derive en forma implícita como una propiedad del sistema.

Se propuso además una clasificación de las creencias de la PLRO que resulta también aplicable a otros sistemas argumentativos. De esta forma se agrupan las creencias dependiendo de su grado de certeza. Por otra parte esta clasificación resulta de utilidad al analizar la propiedad de cumulatividad en la PLRO. A partir del análisis realizado se ilustró como el agregado de distintas categorías de inferencias producen diferentes niveles de cambios en el conjunto de conclusiones. Es también posible extrapolar este estudio a otros formalismos basados en argumentación.

Capítulo 7

Conclusiones y Resultados Obtenidos

La argumentación rebatible ha evolucionado rápidamente en la última década, consolidándose como una nueva disciplina dentro de las Ciencias de la Computación. En los últimos años se desarrollaron numerosas contribuciones al campo de la argumentación [Pollock, 1987, Pollock, 1995, Simari y Loui, 1992, Dung, 1995b, Prakken y Sartor, 1997, Chesñevar, 2001, García y Simari, 2003] y a partir de estos trabajos se la ha reconocido como una herramienta poderosa para representar el conocimiento y modelar el razonamiento de sentido común. Se han implementado prometedoras aplicaciones en distintas vertientes, como por ejemplo en el área de negociación entre agentes [Prakken y Sartor, 1997, Sierra et al., 1997, Parsons y Jennings, 1997, Gordon y Karacapadilis, 1997], en la toma de decisiones [Fox y Parsons, 1998] y para representar el conocimiento de agentes de comercio electrónico [García, 2000]. Por otra parte la demanda de aplicaciones que se pueden desarrollar mediante un sistema argumentativo continúa creciendo, dado el enorme potencial de estos formalismos.

En esta Disertación hemos estudiado las propiedades de la argumentación rebatible en general, y su aplicación en particular a un problema concreto de representación de conocimiento y razonamiento. Para resolver este problema hemos optado por combinar a la argumentación con la programación en lógica, a fin de aprovechar las ventajas de ambas vertientes. A continuación sintetizaremos las conclusiones de nuestra investigación recopilando los principales resultados obtenidos.

A partir del análisis de los sistemas argumentativos llevado a cabo identificamos un conjunto de propiedades deseables en estos formalismos. Se destacan entre las

mismas la declaratividad y simplicidad conceptual del sistema, el poder expresivo y la eficiencia computacional. En este sentido la dialéctica, concepto estudiado en el capítulo 2, juega un rol preponderante dado que su uso permite modelar el proceso de argumentación como un debate en el cual se intercambian argumentos a favor y en contra de una determinada tesis o proposición. Esta conceptualización hace posible obtener un formalismo claro y sencillo sin sacrificar la expresividad del sistema. A la luz de estos resultados se identificó a la programación en lógica rebatible [García y Simari, 2003] como un sistema argumentativo que reúne gran cantidad de propiedades interesantes y logra alcanzar un balance adecuado entre eficiencia y poder expresivo.

Por medio del estudio de las aplicaciones desarrolladas utilizando argumentación arribamos a la conclusión de que el potencial de esta disciplina aún no está siendo aprovechado. Existen gran cantidad de situaciones en las cuales las cualidades de la argumentación podrían ser de suma utilidad, pero aún no están siendo explotadas en forma adecuada. En consecuencia se identificaron distintas áreas de aplicación preponderantes entre las que se destaca la representación de conocimiento en agentes.

En el capítulo 4 se presentó un formalismo argumentativo, denominado programación en lógica rebatible con observaciones (PLRO), que refleja el análisis expuesto en los capítulos anteriores. Este sistema está basado en la programación en lógica rebatible, a fin de heredar las propiedades de este formalismo. Además el lenguaje de la PLRO fue diseñado para ser utilizado como sistema de representación de conocimiento en agentes de software, por lo cual se dedicó especial atención a obtener un conjunto de características auspiciosas para realizar esta tarea.

Cabe destacar que en la definición de la PLRO se considera en detalle cada uno de los elementos de la programación en lógica rebatible [García y Simari, 2003] detallando su evolución con respecto al sistema desarrollado en [Simari y Loui, 1992] (predecesor de la PLR). Por otra parte se redefinen algunos conceptos para que se integren en forma correcta con el nuevo sistema, como es el caso de la especificidad [Poole, 1985] de la cual se presenta una nueva versión, analizando formalmente las propiedades de la misma.

Con respecto a los objetivos de diseño del lenguaje de la PLRO, podemos afirmar que este sistema cumple satisfactoriamente las metas fijadas para el desarrollo de esta tesis. El uso de la PLRO como un sistema de representación de conocimiento y razonamiento arroja resultados alentadores; su lenguaje posee una alta expresividad y permite manipular información potencialmente contradictoria e incompleta. Es

importante destacar que estas afirmaciones están respaldadas por el análisis de las propiedades del sistema que hemos llevado a cabo (ver capítulo 4). En esta oportunidad se describe como representar las creencias de un agente inteligente[1] mediante la PLRO y se verifica que el mecanismo de representación de creencias posea las características necesarias para esta tarea.

El sistema de la PLRO completa su definición con dos características por demás interesantes. En primer lugar se introducen mecanismos de percepción, en base a los cuales se actualiza el programa representando la información del agente. La percepción proporciona al agente la capacidad de actuar en ambiente dinámicos. En este contexto se analizaron las dificultades que plantea esta tarea. Por medio del trabajo de J. Pollock [Pollock, 1987] se identificaron los principales problemas asociados a la implementación de un mecanismo de percepción. Cabe destacar que la incorporación de mecanismos de actualización de conocimiento en formalismos basados en argumentos constituye una propuesta original que posibilita el uso de estos sistemas en un nuevo conjunto de aplicaciones prácticas.

Por otra parte el mecanismo de inferencia de la programación en lógica rebatible con observaciones se define desde una nueva perspectiva, en base al uso de conocimiento precompilado. Esto funda una nueva línea de investigación en el área de los sistemas argumentativos, dado que permite que estos formalismos amplíen sus capacidades, jugando un rol similar al de los sistemas de mantenimiento de verdad en los resolvedores generales de problemas. Todos estos elementos se integran un forma adecuada en el framework propuesto, dado que por ejemplo los mecanismos para la creación y uso del conocimiento precompilado no dificultan la capacidad de percepción del sistema, sino que permiten realizar esta tarea en forma sencilla y eficiente. En este sentido podemos observar que el módulo de conocimiento precompilado se define en forma independiente de las observaciones de la base de conocimiento. De esta forma se evita actualizar este componente ante el agregado de nueva información.

También deseamos destacar que la implementación de la PLRO diseñada en este trabajo favorece la eficiencia del sistema por medio de técnicas de reconocida trayectoria en el área de inteligencia artificial, como son los algoritmos de reconocimiento de patrones. Los procedimientos propuestos en esta oportunidad son los suficientemente generales para ser usados en gran cantidad de aplicaciones para la

[1]Es importante recordar que en la introducción se definió a un agente de software inteligente como una entidad capaz de razonar, construir planes y actuar en un ambiente cambiante.

argumentación, inclusive en ambientes dinámicos.

El estudio de las propiedades de la PLRO en particular y los formalismos argumentativos en general llevado a cabo en el capítulo 6 constituye una contribución de vanguardia al área de la argumentación. La definición de un sistema argumentativo como un conjunto de relaciones entre argumentos permite analizar en forma abstracta la estructura de estos formalismos. Se identificaron además un conjunto de características deseables en los sistemas argumentativos, entre las que se destacan las siguientes.

1. No es conveniente permitir la formación de argumentos auto derrotados, que complican la definición del mecanismo de inferencia.

2. La relación de derrota debe ser un refinamiento de la relación de ataque.

3. Las relaciones de acuerdo entre argumentos (subargumentación y concordancia) no deben poseer pares en común con las relaciones de conflicto (contraargumentación y derrota).

4. Se debe asegurar la finitud del proceso de inferencia.

5. El conjunto de argumentos garantizados debe ser libre de conflictos.

6. Las conclusiones del sistema deben ser consistentes.

7. Si un argumento está garantizado entonces todos sus subargumentos deben estar garantizados.

Todas estas características se analizaron en el ámbito de la PLRO, obteniendo resultados alentadores.

A partir de la evidencia presentada a lo largo de esta Tesis podemos afirmar que se han alcanzado satisfactoriamente los objetivos propuestos para esta investigación. La PLRO constituye un sistema de representación de conocimiento y razonamiento capaz de manejar información incompleta y potencialmente contradictoria, cualidades debidas en gran parte a su mecanismo de inferencia basado en argumentación. Esta propiedad sumada a la capacidad de interactuar con ambientes dinámicos hace que la PLRO posea el poder expresivo y la implementabilidad suficientes para modelar el estado epistémico de un agente de software inteligente.

Trabajo futuro

Por medio de la investigación realizada durante el desarrollo de esta tesis se han presentado diversos resultados que señalan distintas líneas de investigación. En particular merece mención el estudio de las propiedades generales de los sistemas argumentativos. Resultaría interesante e innovador estudiar en detalle los sistemas argumentativos más destacados y analizar en cada caso que propiedades cumple cada sistema. A partir de este analisis sería posible extraer que relación existe entre las propiedades de los sistemas argumentativos presentados en el capítulo 6 y el comportamiento de cada formalismo.

Otro objetivo que resulta de interés consiste en diseñar e implementar una arquitectura de agentes de software basada en el uso de la PLRO como mecanismo de razonamiento y representación de conocimiento. En tal sentido es necesario integrar este sistema con el componente del agente que se encarga de realizar la planificación de acciones. Por otra parte es también posible agregar un protocolo de interacción entre agentes que permita la comunicación con sus pares. De esta forma sería posible obtener una arquitectura que permita modelar tanto la interacción entre agentes como la percepción de un ambiente cambiante, actualizando el conocimiento en forma dinámica.

Por otra parte esta arquitectura permitiría alcanzar un interesante compromiso entre expresividad y simpleza para realizar su implementación. Actualmente se ha comenzado a trabajar en este sentido, generando un reporte preliminar de esta propuesta en [Stankevicius et al., 2003]

Apéndice A

Glosario

A.1. Terminología

Término en inglés	**Traducción utilizada**
argument	argumento
argumentative system	sistema argumentativo
assumption	suposición
atom	átomo
backing	fundamentos
blocking defeater	derrotador de bloqueo
burden of proof	peso de la prueba
claim	afirmación
conclusive reasons	razones conclusivas
countermove	contrajugada
data	datos
default rules	reglas por defecto
defeasible reasoning	razonamiento rebatible
defeasible rule	regla rebatible
defeat	derrota
defeated outright	totalmente derrotado
defeater	derrotador
decision making	toma de desiciones
epistemic attitude	postura epistémica

general problem solving	resolvedores generales de problemas
ground instance	instancia fija
ground literal	literal fijo
ground term	término fijo
indexing	indexado
label	etiqueta
move	jugada
non-monotonic reasoning	razonamiento no monótono
n-person games	juegos de n participantes
opponent	oponente
overruled	denegado
pattern	patrón
pattern matching algorithms	algoritmos de reconocimiento de patrones
precompiled knowledge	conocimiento precompilado
preferred extension	extensión preferida
presupposition	presuposición
production systems	sistema de producción
proper defeater	derrotador propio
proponent	proponente
provisionally defeated	provisionalmente derrotado
qualification problem	problema de la calificación
rationales	razones
reason based logic	logica basada en explicaciones
reasoning under uncertainty	razonamiento bajo incertidumbre
rebut	rebatir
rebutting	rebatimiento
signature	signatura
specificity	especificidad
stable extension	extensión estable
stable marriage problem	problema del matrimonio estable
stable semantics	semantica estable
stage	estadío
truth maintenance systems	sistemas de mantenimiento de verdad
undercut	socavar

undermine	debilitar
updating function	función de actualización
warrant	normativa
warranted	garantizado
well-founded semantics	semántica bien fundada
working memory	memoria de trabajo

A.2. Simbología

Símbolo	**Página**	**Significado**
$\sim h$	103	negación fuerte del átomo h
$\langle \mathcal{V}, Pred, Func \rangle$	103	signatura de un programa en la PLRO
$\mid\sim$	107	inferencia rebatible
$\mathtt{H} \prec \mathtt{B}$	104	regla rebatible con cuerpo B y cabeza H
$cabezas(A)$	108	literales en las cabezas de las reglas en el conjunto A
$cuerpos(A)$	108	literales en los cuerpos de las reglas en el conjunto A
$literales(A)$	108	literales en las reglas presentes en el conjunto A
$\langle A, h \rangle$	108	$\mathcal{A}$ es un argumento para el literal h
$\mathcal{T}_{\langle A,h \rangle}$	114	árbol dialéctico para el argumento $\langle A, h \rangle$
$\mathcal{P}$	105	$\mathcal{P}$ es un programa lógico rebatible
$lit(\mathcal{P})$	109	literales que se pueden derivar a partir de $\mathcal{P}$.
Ψ	106	conjunto de observaciones
Δ	106	conjunto de reglas rebatibles
$\mathcal{P}{=}(\Psi, \Delta)$	108	$\mathcal{P}$ es un programa compuesto por un conjunto de observaciones Ψ y un conjunto de reglas rebatibles Δ
$\succ$	110	especificidad entre argumentos en la PLRO
$\mathcal{S}(A)$	108	conjunto soporte del argumento $\mathcal{A}$
$\langle\langle A, h \rangle\rangle$	130	A es un argumento potencial para h
$\mathcal{DB}_\Delta$	136	base de dialéctica

$ArgPot(A)$	136	conjunto de argumentos potenciales de la base de dialéctica
D_p	136	relación de derrota propia sobre la base de dialéctica
D_b	136	relación de derrota por bloqueo sobre la base de dialéctica
$\mathtt{LC}(A)$	143	literales de consistencia de A
$\mathtt{LM}(A)$	143	literales de minimalidad de A
$S_{(\Psi,\Delta)}$	165	relación de subargumentación sobre un programa de la PLRO
$CC_{(\Psi,\Delta)}$	167	relación de concordancia sobre un programa de la PLRO
$C_{(\Psi,\Delta)}$	169	relación de contraargumentación sobre un programa de la PLRO
$D_{(\Psi,\Delta)}$	171	relación de derrota sobre un programa de la PLRO
$\mid\!\sim_{\Delta}$	186	relación de inferencia de la PLRO
$C_{\Delta}(\Psi)$	186	relación de consecuencia en la PLRO

Bibliografía

[Alchourrón et al., 1985] Alchourrón, C., Gardenfors, P., y Makinson, D. (1985). On the logic of theory change: partial meet contraction and revision functions. *Journal of symbolic logic*, 50:510–530.

Este trabajo contiene la presentación formal de la conocida teoría AGM. En primer lugar se definen las operaciones de *partial meet contraction*, esto es, aquellas operaciones de contracción que se basan en la selección de los subconjuntos maximales del conjunto (lógicamente cerrado) original que fallan en implicar la sentencia a contraer. Luego se presenta una caracterización axiomática de las operaciones de *partial meet contraction* en términos de postulados. Se extienden los resultados para casos particulares de operaciones de *partial meet contraction* (como las *maxichoice contraction* y *full meet contraction*) y se obtienen caracterizaciones de operaciones de *partial meet revision* generadas a partir de operaciones de *partial meet contraction* mediante la identidad de Levi.

[Baral y Gelfond, 1999] Baral, C. y Gelfond, M. (1999). Reasoning Agents in Dynamic Domains. En Minker, J., editor, *Proceedings of the Workshop on Logic-Based Artificial Intelligence*, págs. 257–279. Kluwer Academic Publishers.

Este artículo define una arquitectura para agentes inteligentes basada en un lenguaje de programación en lógica denominado *A-prolog*. Este lenguaje se utiliza tanto para representar el conocimiento del agente como para modelar el razonamiento del mismo. Los autores muestran como estas tareas pueden realizarse mediante el análisis de programas *A-prolog* especificamente codificados para estas funciones y presentan una metodología para la construcción de estos programas.

[Bondarenko et al., 1997] Bondarenko, A., Dung, P. M., Kowalski, R., y Toni, F. (1997). An abstract argumentation–theoretic approach to default reasoning. *Artificial Intelligence*, 93(1–2):63–101.

Se presenta un formalismo argumentativo abstracto que subsume a varios sistemas de reconocida importancia como por ejemplo la lógica default, la programación en lógica y las lógicas autoepistémica. Esta formalismo permite extender cualquier lógica nomonótona con un conjunto rebatible de suposiciones, una suposicion es atacada si su 'contrario' puede derivarse, posiblemente con la ayuda de otras suposiciones en conflicto. Los autores muestran que en este framework la mayoria de las semánticas para las lógicas de razonamiento default pueden modelarse sancionando un conjunto de suposiciones como extensión de una determinada teoría sí y sólo sí esta conjunto es 'conflict free' (es decir que no se ataca a si mismo) y ataca cualquier suposición que no este presente en el. Posteriormente se proponen algunas semánticas más liberales, basadas en argumentación, y se identifican condiciones para la existencia de extensiones y la equivalencia de las distintas semánticas.

[Bratman et al., 1988] Bratman, M. E., Israel, D., y Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4:349–355.

En este artículo los autores postulan que un agente inteligente debe ser capaz de realizar razonamiento práctico para construir planes en forma adecuada y decidir entre distintos cursos de acción en forma racional. Estas tareas deben llevarse a cabo teniendo en cuenta que los recursos existentes son limitados. Se presenta entonces una especificación en alto nivel de un componente de razonamiento práctico para la arquitectura de un agente racional con recursos acotados.

[Capobianco, 1999] Capobianco, M. (1999). El Rol de las Bases de Dialéctica en la Argumentación Rebatible. tesis de licenciatura.

En esta tesis de licenciatura se estudiaron los sistemas argumentativos en general, para luego concentrarse en el formalismo MTDR [Simari y Loui, 1992]. En el marco de este sistema se investigó como optimizar el proceso de inferencia de este sistema argumentativo. La propuesta desarrollada consistió en almacenar las inferencias obtenidas por el sistema, junto con el razonamiento realizado para obtenerlas, lo que permite ahorrar trabajo de cómputo al resolver la misma consulta más de una vez.

[Capobianco, 2001] Capobianco, M. (2001). On the effect of dynamic environments on defeasible reasoning. En *Proceedings del 8vo Workshop en Aspectos Teóricos de la Inteligencia Artificial (ATIA), 3er Workshop de Investigadores en Ciencias de la Computación (WICC)*, págs. 89–91, San Luis. Universidad Nacional de San Luis.

Esta publicación discute brevemente los problemas asociados a razonar en un ambiente dinámico, tomando en cuenta en cambio y la persistencia de las características del mundo. Se presenta una posible alternativa solucionar este problema en un razonador argumentativo, incorporando mecanismos de percepción y optimizando el mecanismo de inferencia para responder en el tiempo adecuado a los requirimientos del ambiente.

[Capobianco y Chesñevar, 1999] Capobianco, M. y Chesñevar, C. I. (1999). Introducing Dialectical Bases in Defeasible Argumentation. En *Proceedings del 6to workshop sobre Aspectos Teóricos de la Inteligencia Artificial (ATIA), 1er Workshop de Investigadores en Ciencias de la Computación (WICC)*, págs. 1–10, San Juan. Universidad Nacional de San Juan.

Este artículo presenta un primer acercamiento para acelerar el proceso de inferencia de un sistema argumentativo utilizando conocimiento precompilado. El sistema MTDR es el formalismo elegido para desarrollar este trabajo y la idea novel de esta propuesta consiste en almacenar las justificaciones que ya han sido construidas con el fin de ahorrar esfuerzo computacional al resolver una misma consulta más de una vez. A continuación los autores analizan el problema de utilizar sistemas argumentativos para modelar entornos dinámicos y proponen una solución viable para el formalismo MTDR.

[Capobianco y Chesñevar, 2000] Capobianco, M. y Chesñevar, C. I. (2000). Using Logics Programs to Model an Agent's Epistemic State. En *Proceedings del 7mo Workshop en Aspectos Teóricos de la Inteligencia Artificial (ATIA), 2nd Workshop de Investigadores en Ciencias de la Computación (WICC)*, págs. 26–28, La Plata. Universidad Nacional de La Plata.

En este trabajo se estudia la posibilidad de utilizar programas lógicos rebatibles para codificar el estado epistémico de un agente racional y se analizan las ventajas de este acercamiento con respecto a otros sistemas, como la programación en lógica extendida y las lógicas modales.

[Capobianco et al., 2000] Capobianco, M., Chesñevar, C. I., y Simari, G. R. (2000). A Formalization of Dialectical Bases for Defeasible Logic Programming. En *Proceedings of the 6th Internacional Congress of Informatics Engineering*, Capital Federal. Universidad Buenos Aires.

Este artículo presenta una versión corregida y aumentada del trabajo desarrollado por los autores en [Capobianco y Chesñevar, 1999], adaptando el trabajo realizado al sistema de la programación en lógica rebatible. Se detallan los algoritmos para

la creación y mantenimiento de un repositorio de justificaciones, analizando como actualizar el mismo ante los cambios que puede sufrir la base de conocimiento en un ambiente dinámico.

[Capobianco et al., 2001] Capobianco, M., Chesñevar, C. I., y Simari, G. R. (2001). An argumentative formalism for implementing rational agents. En *Proceedings del 2do Workshop en Agentes y Sistemas Inteligentes (WASI), 7mo Congreso Argentino de Ciencias de la Computación (CACIC)*, págs. 1051–1062, El Calafate, Santa Cruz. Universidad Nacional de la Patagonia Austral.

Este artículo refina el lenguaje de la Programación en Lógica Rebatible con Observaciones (PLRO) presentado en [Capobianco y Simari, 2000]. Se diseña además una implementación de la PLRO basada en algoritmos de reconocimiento de patrones, que permite utilizar técnicas de reconocida trayectoria para realizar en forma eficiente la creación y uso de las bases de dialéctica.

[Capobianco y Simari, 2000] Capobianco, M. y Simari, G. R. (2000). Defeasible Reasoning in Dynamic Domains. En *Proceedings del 1er Workshop en Agentes y Sistemas Inteligentes (WASI), 6to Congreso Argentino de Ciencias de la Computacin (CACIC)*, págs. 1481–1492, Ushuaia. Universidad Nacional de la Patagonia San Juan Bosco.

Este artículo presenta el lenguaje de la Programación en Lógica Rebatible con Observaciones (PLRO) como una alternativa prometedora para modelar el conocimiento de agentes racionales. Como la mayoría de los agentes con utilidad práctica deben desenvolverse en ambientes dinámicos, es necesario que estos sean capaces de observar los cambios que suceden en el ambiente e incorporarlos a sus creencias. En consecuencia en este trabajo se elaboran mecanismos para actualizar el conocimiento del agente frente a los cambios del entorno. Finalmente, los autores definen una optimización del proceso de inferencia de la PLRO denominada base de dialéctica. Esta optimización permite agilizar el razonamiento del agente que utiliza el lenguaje de la PLRO permitiendo que éste mantenga una interacción adecuada con su ambiente.

[Carbogim y Robertson, 1999] Carbogim, D. y Robertson, D. (1999). Contract-based negociation via argumentation. En *Proceedings of the International Conference on Logic Programming.*

En este artículo los autores presentan un modelo general de negociación basada en contratos. Un contrato representa una colección de variables sobre las cuales se debe

acordar un valor por medio de la negociación. Se propone además una formalización de este modelo por medio de un framework argumentativo basado en la programación en lógica.

[Chesñevar, 1996] Chesñevar, C. I. (1996). El Problema de la Inferencia en Sistemas Argumentativos: Alternativas para su solución. Tesis de Magister, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina.

La mayoría de los formalismos existentes en el área de argumentación realizan una búsqueda exhaustiva para resolver una consulta a partir de la información presente en su base de conocimiento. En esta tesis se desarrollan acercamientos que permiten obtener modelos más acabados del proceso de razonamiento basado en argumentos, introduciendo elementos que guían la formación de los mismos en el proceso de inferencia. Las principales contribuciones consisten en el concepto de orden de evaluación en árboles dialécticos, que permite direccionar la búsqueda de derrotadores durante la construcción del árbol dialéctico, y la noción de forma argumental, que consiste en una generalización de la noción de argumento mediante el uso de reglas no instanciadas. Esta noción independiza la inferencia argumentativa de la información contingente presente en la base de conocimiento.

[Chesñevar, 2001] Chesñevar, C. I. (2001). *Formalización de los Procesos de Argumentación Rebatible como Sistemas Deductivos Etiquetados.* Tesis Doctoral, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina.

Esta disertación presenta un marco lógico formal unificado para modelar la argumentación rebatible, basado en *sistemas deductivos etiquetados* (SDE). El autor manifiesta que los SDEs constituyen una metodología flexible que resulta adecuada para formalizar sistemas lógicos complejos. Se utiliza como lenguaje objeto a la programación en lógica complementada con etiquetas que identifican elementos distinguidos para la representación de conocimiento y la obtención de inferencias. La definición del sistema es acompañada por un profundo estudio de sus propiedades.

[Chesñevar et al., 2000a] Chesñevar, C. I., Dix, J., Stolzenburg, F., y Simari, G. R. (2000a). Relating Defeasible and Normal Logic Programming through Transformation Properties. En *Proceedings del VI Congreso Argentino de Ciencias de*

la Computación, págs. 371–382, Ushuaia, Argentina. Universidad Nacional de la Patagonia.

En esta publicación se intenta relacionar a los programas lógicos rebatibles [García, 2000] con los programas lógicos tradicionales mediante un conjunto transformaciones que no modifiquen la semántica del programa. Posterioemente se analizan distintas propiedades de las transformaciones propuestas y se identifican aspectos en los cuales difieren ambos acercamientos a la programación en lógica.

[Chesñevar et al., 2000b] Chesñevar, C. I., Maguitman, A., y Loui, R. P. (2000b). Logical Models of Argument. *ACM Computing Surveys*, 32(4):337–383.

Este artículo contiene un profundo estudio del campo de la argumentación. En primer lugar se discuten las ideas principales que motivaron la definición de sistemas argumentativos para modelar el razonamiento de sentido común. Luego los autores analizan el uso de la argumentación en aplicaciones concretas. Se discurre además sobre el rol que cumple la dialéctica en la argumentación. Finalmente se resume la evolución de los formalismos argumentativos desde 1980 hasta la actualidad, detallando los sistemas más prominentes que han sido desarrollados en esta disciplina.

[Clark, 1978] Clark, K. L. (1978). Negation as failure. En Gallaire, H. y Minker, J., editores, *Logic and Databases*, págs. 293–322. Plenum Press.

Se define formalmente a la negación por falla de los programas lógicos y se define su semántica por medio del método de completamiento. Este método consiste en una forma de hacer explícitas las inferencias que pueden realizarse a partir del programa utilizando la regla de la negación por falla. Se añade información negativa al programa, de forma tal que un átomo falla si y sólo si su negación es derivable del completamiento del programa.

[Cohen, 1977] Cohen, B. L. (1977). A powerful and efficient structural pattern recognition system. *Artificial Intelligence*, 9(3):223–255.

Este artículo describe un estructura general para los sistemas de reconocimiento de patrones. Cohen presenta además a CODE, un lenguaje de descripción de patrones basado en la lógica booleana y la teoría de conjuntos. Finalmente se considera la representación de los elementos del sistema que pueden influir sobre la eficiencia del mismo.

[Davis, 1989] Davis, R. E. (1989). *Truth, Deduction, and Computation*. Computer Science Press.

Este libro contiene una introducción a la lógica para ciencias de la computación. En primer lugar se presentan los conceptos básicos sobre los sistemas formales, integrando las nociones de verdad, deducción y prueba. Tanto la lógica proposicional como el cálculo de predicado se estudian a la luz de estos tres aspectos. A continuación se estudia la teoría elemental de los números, definiendo en forma estricta el conceptos de algoritmo para luego probar que el sistema formal en cuestión en indecidible. También se ilustran los resultados de incompletitud de Gödel. Finalmente se analiza el λ-calculus como un nuevo ejemplo de teoría formal.

[de Kleer, 1986] de Kleer, J. (1986). An assumption based TMS. *Artificial Intelligence*, 28(2):127–162.

Este artículo presenta una nueva categoría de sistemas de mantemiento de verdad [Doyle, 1979] (TMS), denominada TMS basados en suposiciones. A diferencia de los TMS tradicionales que solamente manejan conjuntos de justificaciones, esta modificación se caracteriza por considerar además conjuntos de suposiciones, que se definen como datos básicos a partir de los cuales se puede derivar el resto de la información. Cada dato del resolvedor de problemas es asociado con un conjunto de suposiciones, que representa los contextos en los cuales este dato es válido. Esta nueva alternativa presenta varias ventajas con respecto al acercamiento tradicional de los TMS, en el cual cada dato es etiquetado como "in" o "out" dependiendo de si forma o no parte del conjunto de creencias y el contexto actual esta definido en forma implícita por el conjunto de datos etiquetados como "in". En consecuencia la base de datos debe ser siempre consistente y no es posible hacer referencia al contexto en forma explícita. Por el contario, los TMS basados en suposiciones permiten manejar información inconsistente utilizando diferentes contextos. Además mejoran la eficiencia de los TMS dado es posible evitar la mayor parte del backtraking necesario para manipular las justificaciones de los TMS.

[de Kleer, 1989] de Kleer, J. (1989). A comparison of ATMS and CSP techniques. En Sridharan, N. S., editor, *Proceedings of the 11th International Joint Conference on Artificial Intelligence, Workshop on Practical Reasoning and Rationality*, págs. 290–296, Detroit, USA. Morgan Kaufmann.

Este artículo elabora sobre como utilizar a los sistemas de mantenimiento de verdad basados en suposiciones para resolver problemas de satisfacción de restricciones. Se presentan en detalle algunas técnicas para realizar esta tarea y se muestran ejemplos de su uso.

[Doyle, 1979] Doyle, J. (1979). A Truth Maintenance System. *Artificial Intelligence*, 12(3):231–272.

Este artículo describe un módulo independiente llamado "sistema de mantenimiento de verdad", el cual maneja las creencias de un sistema general de resolución de problemas. Para esto almacena las justificaciones que sostienen cada creencia y las utiliza para construir explicaciones a las respuestas brindadas por este programa frente a las distintas consultas del usuario. Se describe además la representación de las estructuras usadas en el sistema de mantenimiento de verdad y los mecanismos para actualizar el conjunto de creencias.

[Dung, 1995a] Dung, P. M. (1995a). An argumentation-theoretic foundation of logic programming. *Journal of Logic Programming*, 77:321–357.

Este artículo presenta una extensión de la programación en lógica en la cual se utilizan reglas con literales negativos que representan hipótesis inductivas. En base a esta clase de programas Dung introduce una semántica simple para la programación en lógica basada en la noción de hipótesis aceptables. Posteriormente el autor demuestra que mediante esta semántica es posible unificar distintos acercamientos a la semántica de la programación en lógica ($e.g.$ modelos bien fundados, modelos estables, etc). Se estudian también condiciones de suficiencia para que coincidan las diferentes semánticas de un programa lógico.

[Dung, 1995b] Dung, P. M. (1995b). On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning and Logic Programming and n-person Games. *Artificial Intelligence*, 77:321–357.

Este trabajo presenta un estudio de los mecanismos de argumentación utilizados en el razonamiento de sentido común y explora distintas formas de automatizar estos mecanismos. Para esto el autor desarrolla un sistema argumentativo abstracto en el cual la noción de aceptabilidad juega un papel preponderante. Las propiedades de este sistema se analizan independientemente de la estructura de los argumentos. Para demostrar la expresividad de esta teoría, varios sistemas de razonamiento no-monótono son codificados como instanciaciones particulares de la misma.

[Elkan, 1990] Elkan, C. (1990). A Rational Reconstruction of Nonmonotonic Truth Maintenance Systems. *Artificial Intelligence*, 43:219–234.

Este artículo tiene como objetivo elucidar las capacidades y limitaciones de los sistemas de mantenimiento de verdad. Para esto se presenta una caracterización precisa de

la semántica de un sistema de mantenimiento de verdad utilizando en primer lugar la semántica de modelo estable para programas lógicos [Gelfond y Lifschitz, 1991] y luego la lógica autopistémica definida en [Moore, 1984]. El autor remarca que los resultados obtenidos permiten explicar el comportamiento de los TMS. Otro de las contribuciones principales de este trabajo consiste en una prueba de que cualquier implementación de un sistema de mantemiento de verdad debe resolver un problema NP-completo, junto con una análisis de las implicancias de esta prueba.

[Fillotrani, 2001] Fillotrani, P. R. (2001). *Semántica para la Negación en Programas Lógicos Extendidos*. Tesis Doctoral, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina.

Esta disertación propone un formalismo de programación en lógica cuyo lenguaje separa en forma explícita la asociación entre la negación y la inferencia no monótona. La sintáxis del lenguaje está basada en algunos teorías de *circumscripción* definidas por Lifschitz. La semántica se introduce de dos formas diferentes y se compara además con las formalizaciones existentes para *programas lógicos extendidos*. Luego se detallan varios ejemplos de aplicación del lenguaje obtenido y se presentan extensiones al formalismo básico.

[Forbus y de Kleer, 1993] Forbus, K. y de Kleer, J. (1993). *Building Problem Solvers*. MIT Press, Cambridge, Massachusetts.

Este libro presenta los principios fundamentales detras de la creación de una amplia clase de resolvedores de problemas, como son los sistemas de inferencia dirigidos por patrones, los lenguajes de restricciones y los sistemas de mantenimiento de verdad. Los autores también detallan como construir software robusto y confiable en el área de inteligencia artificial. Los ejemplos varían desde simples programas de búsqueda hasta modelos más complejos como razonadores cualitativos.

[Forgy, 1982] Forgy, C. (1982). Rete: A Fast Algorithm for the Many Patterns/Many Objects Match Problem. *Artificial Intelligence*, 19(1):17–37.

Este artículo presenta un algoritmo de reconocimiento de patrones que constituye un método eficiente para comparar gran cantidad de patrones con gran de cantidad de objetos. Rete fue desarrollado para ser utilizado en intérpretes de sistemas de producción y ha sido probado extensivamente con diferente cantidad de reglas. Para describir en detalle este algoritmo, el autor explica los conceptos básicos, como la

representación de los objetos y los patrones apropiada para el mismo y las operaciones llevadas a cabo por el reconocedor de patrones.

[Fox y Das, 2000] Fox, J. y Das, S. (2000). *Safe and Sound: Articial Intelligence in Hazardous Applications*. MIT Press, Cambridge, Massachusetts.

Este libro describe desde una perspectiva práctica y teórica los avances que la inteligencia artificial ha realizado para asistir a la toma de decisiones en la medicina tradicional. La técnología que se presenta en el libro se basa en un método sistemático para la construcción de agentes inteligentes. Aunque está orientado hacia la medicina, las ideas desarrolladas en el libro pueden ser utilizadas en aplicaciones de la inteligencia artificial en otras disciplinas. También se analizan un conjunto de problemas generales de la inteligencia artificial, como representación de conocimiento, planning y toma de decisiones bajo incertidumbre.

[Fox y Parsons, 1998] Fox, J. y Parsons, S. (1998). Arguing about beliefs and actions. En Hunter, A. y Parsons, S., editores, *Applications of uncertainty formalisms*, volumen 1455 de *Lecture Notes on Artificial Intelligence*, págs. 266–302, Berlin, Germany. Springer-Verlag.

Este artículo contiene un acercamiento para la toma de decisiones bajo incertidumbre mediante la construcción de argumentos a favor y en contra de una determinada opción. Los autores destacan que la teoría de desición clásica, basada en la asignación de probabilidades condicionales a los eventos y utilidades a las consecuencias de las acciones, en muchos casos no resulta adecuada, dado que no es fácil asignar probabilidades en forma confiable. En cambio, el acercamiento desarrollado soluciona estas limitaciones mediante el uso de argumentos. Este sistema se extiende posteriormente para razonar sobre el valor esperado de las distintas acciones.

[Gabbay, 1985] Gabbay, D. (1985). Theoretical foundations for nonmonotonic reasoning in expert systems. En Apt, K., editor, *Logics and Models of Concurrent Systems*. Springer-Verlag, Berlin, Germany.

En este artículo se analizan por primera vez las características de los sistemas de inferencia no monótonos. Gabbay plantea que las lógicas no monótonas solamente se describen por medio de la propiedad que no poseen (monotonicidad). Propone entonces un conjunto de características que debieran respetar los sistemas no monótonos para "razonar" en forma adecuada.

[García y Simari, 2003] García, A. y Simari, G. (2003). Defesible logic programming: An argumentative approach. To appear in Theory and Practice of Logic Programming.

Este artículo contiene la definición formal del lenguaje de la programación en lógica rebatible (PLR), refinando y actualizando el trabajo realizado en [García, 2000].

[García, 1997] García, A. J. (1997). La Programación en Lógica Rebatible: su definición teórica y computacional. Tesis de Magister, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina.

En esta tesis se define un lenguaje para la representación de conocimiento denominado *programación en lógica rebatible*, que combina las propiedades de la programación en lógica y la argumentación rebatible. Por sus capacidades argumentativas heredadas del sistema MTDR, este lenguaje captura aspectos del razonamiento de sentido común difíciles de expresar en otros formalismos (como es el caso de la programación en lógica convencional). El autor define una máquina abstracta para facilitar la implementación del lenguaje propuesto, que juega un rol similar al de la máquina abstracta de Warren (WAM) en la construcción de compiladores para el lenguaje PROLOG.

[García, 2000] García, A. J. (2000). *La Programación en Lógica Rebatible: Lenguaje, Semántica Operacional, y Paralelismo*. Tesis Doctoral, Departamento de Ciencias de la Computación, Universidad Nacional del Sur, Bahía Blanca, Argentina.

La contribución principal de esta tesis consiste en la definición formal del lenguaje de la programación en lógica rebatible (PLR). En tal sentido puede considerarse un refinamiento del trabajo realizado por el autor en [García, 1997]. En esta nueva versión se definen condiciones adicionales sobre los árboles dialécticos (el mecanimo de inferencia de la PLR), se detalla un nuevo criterio de comparación de argumentos y se demuestra un conjunto de propiedades. García desarrolla además una implementación en Prolog del lenguaje de la PLR, a fin de definir la semántica operacional del sistema. Otro aporte de este trabajo consiste en un estudio sobre las formas de paralelismo presentes en la PLR que culmina con la presentación de un mecanismo de computo en paralelo para el mecanismo de inferencia de este sistema.

[García et al., 1993a] García, A. J., Chesñevar, C. I., y Simari, G. R. (1993a). Bases de argumentos: su mantenimiento y revisión. En *Anales de las 22as Jornadas Argentinas de Informática e Investigación Operativa*, págs. 43–62. XIX Conferencia Latinoamericana de Informática.

Este trabajo presenta una modificación del formalismo argumentativo de Simari y Loui [Simari y Loui, 1992], denominada ARGUS, que incorpora un sistema de mantenimiento de argumentos (SMA). El SMA permite almacenar los argumentos hallados en la obtención de inferencias y luego utilizarlos cuando sea necesario recalcular alguna consulta. De esta forma ARGUS evita reconstruir justificaciones ya obtenidas en el pasado y así se reduce el tiempo de respuesta del mecanismo de inferencia. ARGUS también permite actualizar la base de conocimiento del sistema mediante la incorporación de hechos a la misma, siempre que la base de conocimiento resultante sea consiste. Ante esta operación todos los argumentos almacenados en el SMA deben ser revisados para determinar cuales siguen siendo argumentos válidos y cuales deben ser eliminados. A continuación en sistema obtiene aquellos argumentos que se deriven por medio del nuevo hecho, verificando si alguno de ellos resulta ser un contraargumento de un algún argumento A almacenado en el SMA. En este caso se actualiza la información sobre A en el SMA.

[García et al., 1993b] García, A. J., Chesñevar, C. I., y Simari, G. R. (1993b). Making Argument Systems Computationally Attractive. En *Anales de la XIII Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación*, págs. 335–344. Universidad de La Serena, La Serena (Chile).

Este artículo describe la implementación del sistema argumentativo ARGUS, una modificación del formalismo de Simari y Loui [Simari y Loui, 1992]. ARGUS incorpora un conjunto de nuevas características como un proceso optmizado de construcción de argumentos, un test de consistencia embebido en el mecanismo de inferencia, y una estratégia de poda para árboles dialécticos [Simari et al., 1994]. Los autores detallan también como agregar un sistema de mantenimiento de argumentos [García et al., 1993a], que permita almacenar los argumentos hallados en la obtención de inferencias y luego utilizarlos cuando sea necesario recalcular alguna consulta.

[Gelfond y Lifschitz, 1991] Gelfond, M. y Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New Generation Computing*, págs. 365–385.

Este artículo constituye una de las piedras angulares de la programación en lógica. Los autores extienden los programas lógicos tradicionales incluyendo a la negación clásica junto a la negación por falla. Gelfond y Lifschitz manifiestan que esta extensión aumenta el poder expresivo de la programación en lógica, permitiéndole manejar

información incompleta en forma más directa. Se define además la semántica de estos programas por medio del concepto de modelos estables.

[Gordon, 1987] Gordon, T. F. (1987). Oblog-2: a hybrid knowledge representation system for defeasible reasoning. En *Proceedings of the First International Conference on Artificial Intelligence and Law*, págs. 231–239. ACM Press.

En esta publicación Gordon desarrolla un sistema híbrido de representación de conocimiento para realizar razonamiento rebatible. OBLOG-2 combina un *razonador terminológico* junto con un mecanismo de inferencia al estilo Prolog. El componente terminológico permite incorporar descripciones de taxonomía para tipos y atributos. Las entidades del sistema son un conjunto de tipos. En base a las descripciones existentes, se definen procedimientos para asignar valor al conjunto de atributos de una entidad. Estos procedimientos están codificados mediante claúsulas Horn indexadas de acuerdo al conjunto de tipos. Los tipos que se conocen de una entidad determinan el conjunto de reglas que son aplicables sobre los mismos, el cual cambia conforme el conocimiento sobre los tipos de una determinada entidad es refinado, permitiendo realizar un razonamiento rebatible. El autor destaca que OBLOG-2 fue diseñado con el objetivo de modelar escenarios jurícos, en los cuales las leyes suelen ser representadas como reglas generales que pueden estar sujetas a excepciones.

[Gordon y Karacapadilis, 1997] Gordon, T. F. y Karacapadilis, C. (1997). The zeno argumentation framework. En *Proceedings of the 10th International Conference on Artificial Intelligence and Law*, págs. 10–18, Melbourne, Australia.

En este articulo se presenta el formalismo argumentativo Zeno, que consiste en un modelo formal de argumentación basado en el modelo informales de Toulmin [Toulmin, 1958]. La idea primordial de este sistema reside en una función de etiquetado que utiliza argumentación para computar información heurística sobre la calidad de las distintas opciones disponibles. Zeno fue diseñado para ser usado en los sistemas de mediación, un foro de discusión electrónico con soporte para argumentación y negociación.

[Haenni, 1998] Haenni, R. (1998). Modeling uncertainty with propositional assumption-based systems. En Hunter, A. y Parsons, S., editores, *Applications of uncertainty formalisms*, págs. 446–470. Springer-Verlag.

Este artículo propone a los sistemas basados en suposiciones como una forma eficiente y conveniente de codificar información incierta. Los sistemas basados en suposiciones

se obtienen a partir de la lógica proposicional, agregando una clase especial de proposiciones denominadas suposiciones, que expresan la incerteza de cierta información. Estos sistemas permiten juzgar hipótesis en forma cualitativa o cuantitativa. En este trabajo se describe como computar en forma eficiente los argumentos simbólicos para sustentar las distintas hipótesis y se presenta a ABEL, un lenguaje para el modelado de sistemas basados en suposiciones y una herramienta interactiva para razonamiento probabilístico basado en suposiciones.

[Hage y Verheij, 1995] Hage, J. C. y Verheij, B. (1995). Reason-based logic: a logic for reasoning with rules and reasons. *Law, Computers and Artificial Intelligence*, 3(2-3):171–209.

Este artículo presenta el formalismo que da su nombre a la publicación. Los autores destacan que la principal motivación de este trabajo consiste en distinguir el razonamiento con reglas (especialmente de tipo legal) del razonamiento con sentencias verdaderas o falsas al estilo de la lógica clásica. Para considerar esta cuestión, la idea básica de la lógica basada en explicaciones es que la aplicación de una regla conduce a una *razón* que clama por la conclusión de esta regla. En este marco la derivación de una sentencia esta basada en sopesar las razones que claman en contra o favor de esta sentencia. El artículo concluye comparando la lógica basada en explicaciones con otras lógicas para modelar el razonamiento rebatible.

[Haralick y Elliot, 1980] Haralick, R. M. y Elliot, G. L. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3):263–313.

En esta artículo se estudia el número de operaciones de búsqueda sobre un árbol requeridas para resolver problemas de satisfacción de restricciones binarios. Los autores demuestran en forma analítica y experimental que los principios de intentar primero en los lugares con más probabilidad de falla y recordar los errores cometidos para evitar repetirlos mejoran las técnicas estandar de backtracking.

[Kakas y Toni, 1999] Kakas, A. y Toni, F. (1999). Computing Argumentation in Logic Programming. *Journal of Logic and Computation*, 9:512–562.

Se presenta un sistema abstracto a partir del cual distintas clases de semánticas argumentativas pueden calcularse mediante diferentes instanciaciones de la teoría de prueba básica, ampliando el trabajo realizado en [Bondarenko et al., 1997]. La misma esta definida en términos de árboles de prueba en los cuales cada nodo codifica un

ataque a su nodo padre. Esto resulta en una caracterización simple y elegante del proceso de inferencia.

[Katsuno y Mendelzon, 1992] Katsuno, H. y Mendelzon, A. (1992). On the difference between updating a knowledge base and revising it. En P.Gardenfors, editor, *Belief Revision*, págs. 183–203. Cambridge University Press.

En este trabajo los autores destacan una diferencia fundamental entre dos clases de modificaciones que pueden realizarse sobre una base de datos. La primer clase se denomina actualización y consiste en modificar la base de datos para reflejar los cambios ocurridos en el mundo que ésta describe. Esta situación se produce en escenarios dinámicos. La segunda categoría, denominada revisión, se utiliza al obtener nueva información sobre un mundo estático. Los autores enfatizan que los conocidos postulados de racionalidad AGM propuestos por Alchourron, Gärdenfords y Makinson [Alchourrón et al., 1985] son adecuados solamente para modelar operadores de revisión. En congruencia, Katsuno y Mendelzon muestran como estos postulados deben ser drasticamente modificados para describir adecuadamente operadores de actualización.

[Kraus et al., 1990] Kraus, S., Lehmann, D., y Magidor, M. (1990). Non monotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207.

En este artículo los autores estudian las propiedades de las relaciones de inferencia no monótona a fin de encontrar un conjunto de principios que permitan caracterizarlas. Se analizan cinco familias diferentes de relaciones de inferencia no monótonas con diferentes características pero todas ellas basadas en un lenguaje proposicional.

[Krause et al., 1995] Krause, P., Ambler, S., Elvang-Goransson, M., y Fox, J. (1995). A logic of argumentation for reasoning under uncertainty. *Computational Inteligence*, 11(1):113–131.

Este artículo presenta el lenguaje y la teoría de prueba de una lógica para argumentación conocida como LA, que es el núcleo de un modelo teórico para realizar razonamiento bajo incertidumbre. En esta lógica las proposiciones están etiquetadas con una representación de los argumentos que sustentan su validez. Esto resulta en un sistema que combina atributos simbólicos y numéricos para asignar un valor de confianza a las proposiciones basado en el estado de sus argumentos.

[Lifschitz, 1994] Lifschitz, V. (1994). Foundations of Logic Programming. En Brewka, G., editor, *Principles of Knowledge Representation*, págs. 69–127. CSLI Publications.

Este trabajo contiene un survey de la teoría de la programación en lógica con negación clásica y negación con falla. En primer lugar Lifschitz analiza la clase de los programas lógicos proposicionales que no contienen negación por falla. A continuación se consideran los programas proposicionales con negación por falla y luego los programas que incluyen variables. Finalmente este trabajo presenta dos extensiones de la programación en lógica convencional: los programas disyuntivos y las teorías default. Una característica de este artículo que merece ser destacada consiste en que el autor se concentra principalmente en la semántica de los programas lógicos. Esto diferencia este survey de los acercamientos tradiciones focalizados en el mecanismo de evaluación de consultas.

[Lin y Shoham, 1989] Lin, F. y Shoham, Y. (1989). Argument systems: A uniform basis for nonmonotonic reasoning. En Brachman, H. J. L. R. J. y Reiter, R., editores, *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, págs. 245–255, Toronto, Canada. Morgan Kaufmann.

Este artículo contiene la definición de un sistema argumentativo que tiene como objetivo de modelar el razonamiento de sentido común y en particular su naturaleza no monótonona. Una característica original de este sistema consiste en que está basado enteramente en reglas de inferencia. Esto lo diferencia de los formalismos no monótonos existentes hasta ese momento, basados en sentencias. Los autores intentan modelar diversos formalismos no monótonos (en particular la lógica default, la lógica autoepistémica, el principio de la negación por falla y la circunscripción) mediante su framework argumentativo. Para esto detallan como es posible transformar a cada uno de estos formalismos en un framework argumentativo en particular con una semántica adecuada. Desafortunadamente la mayoría de estas transformaciones no constituye evidencia alguna, ya que no son realizables en la práctica.

[Lloyd, 1987] Lloyd, J. W. (1987). *Foundations of Logic Programming*. Springer-Verlag.

Este libro contiene un resumen de los conceptos fundacionales de la programación en lógica. El autor detalla una clasificación de los programas lógicos de acuerdo a la forma de sus reglas. Estas categorías se denominan programas definidos, normales

y finalmente se encuentran los programas completos que no poseen un nombre en especial. En cada capítulo Lloyd estudia en detalle una de estas categorías. Finalmente se presenta una introducción a las bases de datos deductivas.

[Lodder y Herczog, 1995] Lodder, A. R. y Herczog, A. (1995). Dialaw a dialogical framework for modelling legal reasoning. En *Proceedings of the fifth International Conference on Artificial Intelligence and Law*, págs. 146–155. ACM Press.

En este artículo los autores definen un framework para razonamiento legal denominado *DiawLaw*. En este formalismo el razonamiento legal se modela en forma procedimental mediante un diálogo entre dos participantes. Los autores definen los elementos básicos de este diálogo, tales como las jugadas que pueden realizar los participantes, el lenguaje utilizado. También formulan las reglas que gobiernan el desarrollo de los diálogos. Posteriormente se presentan algunos ejemplos de uso del sistema propuesto.

[Loui, 1987] Loui, R. P. (1987). Defeat Among Arguments: A System of Defeasible Inference. *Computational Intelligence*, 3(2):100–106.

En este artículo Loui introduce la noción de argumento en el campo de la Inteligencia Artificial. Se define una teoría que utiliza como lógica subyacente un lenguaje de primer orden y aumenta este lenguaje por medio de un conjunto de reglas de inferencia que permiten representar conocimiento tentativo. La base de conocimiento del sistema se representa por medio de este lenguaje y los argumentos se construyen como grafos de inferencias en base a la información presente en la misma. Se definen además las relaciones de contraargumentación y derrota entre los argumentos del sistema.

[Loui y Norman, 1995] Loui, R. P. y Norman, J. (1995). Rationales and argument moves. *Artificial Intelligence and Law*, 3:159–189.

En este artículo se analizan cinco formas de representar los *rationales* y se presenta una discución formal sobre como éstos pueden alterar a la disputa. El modelo formal de disputación utilizado esta basado en el uso de argumentos.

[Makinson, 1994] Makinson, D. (1994). General patterns in nonmonotonic reasoning. En Gabbay, H. y Robinson, editores, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volumen 3, págs. 35–110. Oxford University Press.

Este artículo fue uno de los pioneros junto con [Gabbay, 1985] en proponer un conjunto de propiedades para clasificar a las lógicas no monótonas. Se identifican un conjunto de

condiciones puras (es decir independientes del lenguaje lógico utilizado) y un conjunto de propiedades que dependen de los conectivos en particular. Se presentan ejemplos y se analiza el comportamiento de un conjunto de sistemas no monótonos destacados con respecto a las propiedades propuestas.

[Martínez y García, 1999] Martínez, D. C. y García, A. J. (1999). Significancia de las falacias en los sistemas argumentativos. En *Proceedings del 6to Congreso Argentino de Ciencias de la Computacin (CACIC)*.

En este artículo los autores exploran en profundidad el problema de las falacias en la argumentación. Martínez y Garía reconocen que en la mayoría de los modelos formales existentes para modelar la argumentación rebatible es posible contruir argumentos falaces, esto es, argumentos que, aunque parezcan válidos, encierran fallas en su razonamiento. A continuación se desarrolla una interesante clasificación de las falacias en la argumentación, distinguiendo las falacias relacionadas con apectos subjetivos del debate de aquellas intrisecas al mecanismo de argumentación. Se destaca que estas últimas deberían ser controladas por el sistema. Finalmente se analiza la posición que adoptan los diversos formalismos argumentativos existentes para el control de falacias.

[McAllester, 1990] McAllester, D. (1990). Truth Maintenance. En Smith, R. y Mitchell, T., editores, *Proceedings of the 8th National Conference on Artificial Intelligence*, volumen 2, págs. 1109–1116. American Association for Artificial Intelligence, AAAI Press.

Este artículo contiene un survey sobre los sistemas de mantenimiento de verdad (TMS), focalizado principalmente en los TMS monótonos. El autor compara varios de los algoritmos existentes para implementar estos sistemas y analiza las aplicaciones de los mismos que han sido desarrolladas para los resolvedores generales de problemas. Finalmente se discute sobre las líneas de investigación existentes para la construcción de algoritmos más poderosos y eficientes.

[McCarthy, 1977] McCarthy, J. (1977). Epistemological Problems in Artificial Intelligence. En *Proceedings of the International Joint Conference on Artificial Intelligence*, págs. 1038–1044.

En este trabajo el autor discute algunos de los problemas sufridos por la representación de conocimiento, en el campo de la inteligencia artificial. McCarthy expresa que no existe una forma adecuada para representar una muchas formas de conocimiento. En consecuencia, mejorar la eficiencia y velocidad de los sistemas computacionales

no servirán de ayuda mientras no exista un sistema de representación con mayor poder expresivo. Para solucionar esta situación se presenta una versión preliminar del formalismo denominado circunscripción [McCarthy, 1980], como una nueva forma de razonamiento.

[McCarthy, 1980] McCarthy, J. (1980). Circumscription — A Form of Non-monotonic Reasoning. *Artificial Intelligence*, 13(1,2):27–39,171–172.

Este artículo contiene la definición del conocido formalismo desarrollado por John McCarthy. Tal como el título lo expresa, la circunscripción surge como una forma de razonamiento no monótono que pretende modelar varios aspectos del razonamiento de sentido común. En particular, la circunscripción fue motivada por el problema de la calificación de acciones, enunciado por John McCarthy y Patrick Hayes en [McCarthy y Hayes, 1969]. El autor muestra un ejemplo canónico en el cual el uso de circunscripción soluciona el problema de la calificación.

[McCarthy y Hayes, 1969] McCarthy, J. y Hayes, P. J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502.

En esta trabajo se analizan desde un punto de vista filosófico los problemas que surgen a partir de la idea de diseñar un agente inteligente. Los autores definen posteriormente un lenguaje denominado 'situation calculus' en un intento de formalizar los conceptos considerados anteriormente. En este artículo se identifica por vez primera el problema de la cualificación como limitante en toda formalización de razonamiento de sentido común

[McDermott y Doyle, 1980] McDermott, D. y Doyle, J. (1980). Non-Monotonic Logic I. *Artificial Intelligence*, 13(1,2):41–72.

Este artículo contiene un estudio de las lógicas no monótonas, su historia y las motivaciones que condujeron al desarrollo estos sistemas. Los autores consideran en su análisis cada uno de los formalismos no monótonos existentes, como la circunscripción, los sistemas de mantenimento de verdad y la lógica default. Con el afán de disponer de un formalismo no monótono con una sólida base teórica McDermot y Doyle definen a continuación la *lógica no monótona*, cuya estructura sigue el estilo de las lógicas clásicas. Se detallan la sintáxis y semántica de este sistema, asi como también su teoría de prueba.

[Minsky, 1975] Minsky, M. (1975). A framework for representing knowledge. En Winston, P., editor, *The Psychology of Computer Vision*, págs. 211–280. Springer-Verlag.

En este artículo Minsky introduce el concepto de *frames* como estructuras que permiten organizar una base de conocimiento de forma tal que la información relevante para una determinada situación pueda ser accedida facilmente.

[Moore, 1984] Moore, R. (1984). Possible-world semantics for autoepistemic logic. En *Proceedings of the Non-Monotonic Reasoning Workshop*, págs. 344–354, New Paltz, USA. American Association for Artificial Intelligence.

Se define la formalmente al conocido sistema no monótono de la lógica autoepistémica utilizando para definir su semántica un mecanismo de mundos posibles.

[Moore, 1985] Moore, R. C. (1985). Semantical considerations on nonmonotonic logic. *Artificial Intelligence*, 25(1):75–94.

Este artículo analiza en profundidad las lógicas no monótonas propuestas por McDermott y Doyle [McDermott y Doyle, 1980]. Moore afirma que ese formalismo posee algunas peculiaridades que ponen en evidencia la falla del mismo para cumplir con sus objetivos de diseño. En consecuencia Moore plantea una reconstrucción de la lógica no monótona que denomina lógica autoepistémica. El autor define además una semántica para el sistema resultante en base a la cual demuestra que la lógica autoepistémica es sensata y completa. Finalmente se contrastan ambos formalismos, destacando las ventajas del nuevo acercamiento.

[Mora et al., 1998] Mora, I. A., Alferez, J., y Schoeder, M. (1998). Argumentation and cooperation for distributed extended logic programming. En *Proceedings of the 7th Workshop on Non-monotonic Reasoning*, Trento, Italy.

En este artículo se presenta un sistema para la cooperación en un sistema multiagente cuya semántica está basada en argumentación. Se presenta un algoritmo para realizar el proceso de inferencia y ejemplos del mecanismo de cooperación.

[Nute, 1987] Nute, D. (1987). Defeasible Reasoning. En *Proceedings of the XX Annual Hawaii International Conference on System Sciences*, págs. 470–477.

Este trabajo, junto con el acercamiento de John Pollock [Pollock, 1987], fue uno de los pioneros en formalizar el concepto de razonamiento rebatible. Nute identifica que en numerosas oportunidades conocer nuevos hechos puede conducirnos a descartar

algunas de nuestras creencias y denomina esta clase de inferencia como razonamiento rebatible. A continuación el autor define la lógica LDR para representar conocimiento y obtener conclusiones utilizando este tipo de razonamiento y desarrolla una implementación computacional de la misma como una extensión del lenguaje PROLOG. Por lo tanto el sistema de Nute posee el mérito de ser uno de los primeros formalismos en el campo de la Inteligencia Artifical en los cuales el trabajo teórico se complementa con una implementación concreta.

[Nute, 1988] Nute, D. (1988). Defeasible Reasoning: a Philosophical Analysis in PROLOG. En Fetzer, J. H., editor, *Aspects of Artificial Intelligence*, págs. 251–288. Kluwer Academic Publishers.

En este artículo Nute desarrolla un análisis descriptivo del razonamiento rebatible. Este análisis no es desarrollado de acuerdo con los métodos usuales de la lógica filosófica, dado que es formulado por medio de un programa Prolog. A través del programa obtenido como producto de este trabajo, el autor define e implementa de un sistema de razonamiento rebatible que sigue los lineamientos de la lógica LDR presentada en [Nute, 1987].

[Parsons y Fox, 1997] Parsons, S. y Fox, J. (1997). Argumentation and decision making. En *Proceedings of the IEEE Coloquium on Decision Making and Problem Solving*.

Este artículo resume la posición de sus autores respecto del uso de métodos simbólicos para razonar bajo incerteza. En particular se discute el uso de argumentación, enfatizando que ésta ofrece un complemento a los métodos numéricos. Se ofrece una presentación histórica, enfatizando las razones que motivaron el desarrollo del framework argumentativo y la investigación realizada en el "Imperial Cancer Research Fund" en los últimos 15 años.

[Parsons y Jennings, 1997] Parsons, S. y Jennings, N. (1997). Negotiation through argumentation: a preliminary report. En *Proceedings of the 2nd International Conference on on Multi-Agents Systems*, págs. 267–274, Kyoto, Japón.

En este artículo los autores definen un framework que permite negociar a un conjunto de agentes basándose en un sistema argumentativo. El formalismo argumentativo es usado para realizar el razonamiento introspectivo del agente y para describir el proceso de negociación entre los distintos agentes que participan del mismo.

[Pereira y Quaresma, 1998] Pereira, L. M. y Quaresma, P. (1998). Modelling agent interaction in logic programming. En *Proceedings of the 11th International Conference on Applications of Prolog*, págs. 150–156, Tokyo, Japón.

En este artículo los autores presentan un framework basado en la programación en lógica e implementado en PROLOG que prentede modelar el estado epistémico de un agente. Cada agente se modela como un conjunto de reglas de la programación en lógica extendida que representan las creencias, intenciones, metas del agente y el conocimiento acerca del mundo. El estado mental del agente se define entonces como el modelo estable de este programa lógico, agregando un conjunto de restricciones específicas sobre el mismo

[Pollock, 1987] Pollock, J. L. (1987). Defeasible Reasoning. *Cognitive Science*, 11(4):481–518.

Este artículo contiene uno de los primeros acercamientos al razonamiento rebatible por medio de la argumentación. Pollock integra el trabajo realizado en epistemologia con los estudios llevados a cabo en la comunidad inteligencia artificial para desarrollar una teoría del razonamiento rebatible. El autor destaca que este formalismo debe definirse en forma precisa, a fin de poder ser implementado en un programa concreto que provea un mecanismo de depuración y prueba de la teoría. A continuación presenta un sistema argumentativo que utiliza una definición inductiva para el conjunto de creencias justificados y permite modelar la reinstaciación de argumentos. Por ultimo Pollock enuncia las ideas fundacionales para la implementación de una versión restringida de la teoría desarrollada.

[Pollock, 1992] Pollock, J. L. (1992). How to Reason Defeasibly. *Artificial Intelligence*, 57(1):1–42.

Este artículo presenta una versión expandida y corregida del sistema desarrollado en [Pollock, 1987]. La nueva versión soluciona el problema de los argumentos auto derrotados y enuncia el principio de derrota colectiva. Pollock explora con mayor detalle como implementar la teoría propuesta y desarrolla un conjunto de algoritmos para el razonador rebatible. Es importante destacar que para hacer más eficiente la implementación el autor presenta un procedimiento guiado por las metas que permite determinar si un argumento está justificado.

[Pollock, 1995] Pollock, J. L. (1995). *Cognitive Carpentry: a Blueprint for How to Build a Person*. Bradford/MIT Press.

En este libro Pollock manifiesta que la mayoría de los sistemas para modelar el razonamiento carecen de fundamentos filosóficos adecuados y considera que este problema es una desventaja significativa de los formalismos existentes. Para solucionar esta situación presenta una propuesta que integra una teoría filosófica del razonamiento racional junto con conceptos de Inteligencia Artificial. Esta propuesta, denominada sistema OSCAR, consiste en una arquitectura para agentes racionales y autónomos que integra un formalismo argumentativo para modelar el razonamiento rebatible desarrollado en publicaciones previas ([Pollock, 1987, Pollock, 1992] entre otras) y un planificador de propósito general.

[Pollock, 1997] Pollock, J. L. (1997). Taking Perception Seriously. En *Proceedings of the 1st International Conference on Autonomous Agents*, págs. 526–527.

En esta publicación se estudia en detalle como incorporar mecanismos de percepción en agentes racionales. Para comenzar, el autor postula que un agente interactuando con un ambiente complejo y dinámico necesita recolectar información por medio de la percepción. Para realizar esta tarea Pollock identifica dos problemas preponderantes. En primer lugar, las observaciones adquiridas por medio de la percepción pueden ser falsas, dado que por ejemplo, el mundo puede ser diferente de los que nuestros sentidos indican. Por otra parte, la percepción produce en realidad alguna forma de muestreo, dado que el agente no puede sensar el mundo constantemente en todo momento. Las capacidades cognitivas del agente deben transformar estas muestras en una imagen coherente del mundo. A continuación Pollock propone soluciones para estos problemas dentro del marco del sistema OSCAR [Pollock, 1995].

[Pollock, 1998] Pollock, J. L. (1998). Perceiving and Reasoning about a Changing World. *Computational Intelligence*, 14:498–562.

En este artículo se aborda la problemática de la percepción en los agentes racionales. Pollock sostiene que un agente situado en un ambiente complejo debe poseer mecanismos para obtener información a partir del mismo. En primer lugar el autor identifica los problemas asociados a la percepción en agentes y propone soluciones para estos problemas basándose en el trabajo desarrollado en la epistemología. A continuación incorpora mecanismos de percepción en la arquitectura del sistema OSCAR [Pollock, 1995] y agrega nuevos esquemas de razonamiento en el razonador rebatible de OSCAR que se encarguen de manejar en forma correcta la adquisición de información.

[Poole, 1988] Poole, D. (1988). A Logical Framework for Default Reasoning. *Artificial Intelligence*, 36(1):27–47.

En este artículo se define un sistema lógico para razonamiento 'default usando como sustrato teórico la lógica de Reiter. El autor plantea una solución a la no monotonicidad de la lógica clásica por medio de una forma sencilla de razonamiento hipotético. El usuario debe proveer un conjunto de hipótesis que se utilizan para ampliar el razonamiento producido por la lógica. Poole propone una semántica para el sistema lógico resultante y diseña además una implementación del mismo.

[Poole, 1985] Poole, D. L. (1985). On the Comparison of Theories: Preferring the Most Specific Explanation. En *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, págs. 144–147. IJCAI.

Este artículo presenta una caracterización del razonamiento no monótono en el estilo de la lógica default [Reiter, 1980], en la cual las reglas default son consideradas como hipótesis o explicaciones de los resultados del sistema. En este marco Poole plantea una solución al problema de decidir entre dos posibles explicaciones o extensiones por medio de una caracterización semántica de cuando una teoría es *más específica*. El autor postula que este criterio de especificidad puede ser aplicado en las redes de herencia, donde se desea elegir los resultados sustentados por el conocimiento más específico. Posteriormente la especificidad fue utilizada en diversos formalismos argumentativos [Simari y Loui, 1992, García, 2000] para comparar teorías conflictivas.

[Prakken, 1993] Prakken, H. (1993). *Logical Tools for Modelling Legal Arguments*. Tesis Doctoral, Free University, Amsterdan, Holanda.

En esta tesis el autor realiza un análisis lógico-formal del razonamiento legal y destaca que algunas de sus características no pueden expresarse por medio de los sistemas lógicos tradicionales. A continuación se estudian diversos formalismos de razonamiento no monótono y se evaluan sus propiedades para modelar la rebatibilidad del razonamiento legal. Por último se desarrolla una teoría argumentativa basada en la lógica default, que constituye la contribución primordial de este trabajo.

[Prakken, 1997] Prakken, H. (1997). *Logical Tools for Modelling Legal Argument*. Kluwer Academic Publishers.

Este libro contiene una versión expandida y revisada de la tesis doctoral de su autor [Prakken, 1993]. Se analizan además los aspectos lógicos del razonamiento legal y se estudian los procesos de argumentación enfatizando su relación con el mismo. El

principal aporte de esta obra consiste en sistema de argumentación rebatible basado en la lógica default que intenta modelar el razonamiento legal.

[Prakken y Sartor, 1997] Prakken, H. y Sartor, G. (1997). Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–752.

En este artículo se presenta un sistema de argumentación rebatible inspirado en el razonamiento legal, que combina un lenguaje al estilo de la programación en lógica con la semántica básica de Dung [Dung, 1995b]. El criterio de preferencia entre argumentos se basa en un conjunto de prioridades entre reglas codificado en forma explícita. Una característica original de este formalismo consiste en que las prioridades no son fijas, sino que están sujetas al debate. Cabe destacar que los autores también desarrollan una teoría de prueba dialéctica, donde una prueba toma la forma de un diálogo entre un proponente y un oponente, siguiendo el modelo de Rescher [Rescher, 1977].

[Prakken y Vreeswijk, 1999] Prakken, H. y Vreeswijk, G. (1999). Logical systems for defeasible argumentation. En Gabbay, D., editor, *Handbook of Philosophical Logic*. Kluwer Academic Publisher.

Este trabajo contiene un análisis del trabajo realizado hasta la actualidad en la comunidad de razonamiento no monótono y en particular en el campo de la argumentación rebatible. En primer lugar los autores discurren sobre los principales acercamientos al razonamiento no monótono y su relevancia para la inteligencia artificial y la filosofía. A continuación se desarrolla un estudio abstracto sobre las características generales de las formalismos de argumentación rebatible. Se realiza también un análisis concreto de los principales sistemas de argumentación rebatible y finalmente se mencionan los principales problemas existentes en el área que, según Prakken y Vreeswijk, deberían ser abordados en trabajos futuros.

[Rao y Wooldridge, 1999] Rao, A. y Wooldridge, M. (1999). Foundations of Rational Agency. En *Foundations of Rational Agency*, número 14 en Applied Logic Series, págs. 1–10. Kluwer Academic Publishers.

En este artículo se distinguen algunos principios generales de los agentes racionales, identificando la problemática actual del área. En particular se destaca al modelo BDI como uno de los desarrollos teóricos predominantes en el área, pero también se reconocen sus limitaciones a la hora de elaborar implementaciones basadas en el mismo, debido a su importante complejidad computacional.

[Rao y Georgeff, 1991] Rao, A. S. y Georgeff, M. P. (1991). Modeling Agents Within a BDI-Architecture. En *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, págs. 473–484.

En este artículo se presenta una formalización diferente del modelo BDI en la cual las intenciones se modelan mediante un esquema de mundos posibles con tiempo ramificado. Es de destacar que en este escenario la noción de intención tiene el mismo estado que las nociones de creencias y deseos y no puede ser reducida a estas.

[Rao y Georgeff, 1995] Rao, A. S. y Georgeff, M. P. (1995). BDI Agents: from theory to practice. En Lesser, V., editor, *Proceedings of the First International Conference on Multi-Agent Systems*, págs. 312–319, San Francisco, CA. MIT Press.

Este artículo explora un tipo particular de agente racional denominado agente BDI (del inglés, *belief-desire-intention*). En primer lugar se investigan las características de la arquitectura BDI desde un punto de vista teórico. A continuación se estudian las implementaciones de los agentes BDI desde una perspectiva práctica y la posibilidad de construir aplicaciones a gran escala usando este tipo de arquitectura de agentes. Para emprender esta tarea los autores destacan que es necesario realizar algunas suposiciones que simplifiquen el formalismo teórico, sacrificando poder expresivo. Las teorías desarrolladas se integran en una aplicación para control del trafico aéreo, denominada OASIS.

[Reiter, 1980] Reiter, R. (1980). A Logic for Default Reasoning. *Artificial Intelligence*, 13(1–2):81–132.

Este artículo contiene la definición de la lógica default, el conocido formalismo no monótono de Raymond Reiter. En primer lugar el autor identifica una categoría particular de razonamiento plausible que denomina razonamiento no monótono y analiza diversas situaciones de importancia para la inteligencia artificial en las cuales se manifiesta este tipo de razonamiento. Luego presenta una serie de argumentos justificando porque la lógica clásica es inadecuada para formalizar los patrones de raonzamiento mencionados anteriormente. Finalmente presenta un estudio de los sistemas no monótonos existentes que culmina con la definición de la lógica default. Reiter detalla además diversos ejemplos que ilustran su comportamiento y analiza sus propiedades.

[Rescher, 1977] Rescher, N. (1977). *Dialectics, a Controversy-Oriented Approach to the Theory of Knowledge*. State University of New York Press.

Se presenta un análisis de la dialéctica y la argumentación desde una perspectiva filosófica. El objetivo de este trabajo consiste en explorar una versión particular de la dialéctica que se centra en las nociones de disputación, debate y controversia racional. Por medio del estudio de estos conceptos el autor pretende dilucidar la utilidad de la dialéctica en la teoría del conocimiento. A lo largo de la obra se abordan diferentes tópicos: la definición de un mecanismo formal de disputación, el análisis de la dialéctica unilateral y los conceptos de "peso de la prueba" y "plausibilidad.

[Russell, 1995] Russell, S. J. (1995). Rationality and Intelligence. *Artificial Intelligence*, 94(1–2):57–77.

En este artículo Russel afirma que el campo de investigación en inteligencia artificial se vería beneficiado ante la creación de una definición de inteligencia que sea al mismo tiempo precisa y general, para caracterizar la gran diversidad de entidades que poseen esta propiedad. Para alcanzar este objetivo el autor elabora la noción de *racionalidad*. A lo largo de este artículo se definen y analizan cuatro formas diferentes de racionalidad: *perfecta*, *calculativa*, *de metanivel* y *acotada*. Una característica común a estas definiciones consiste en que todas ellas la inteligencia de una entidad esta asociada con su capacidad de alcanzar un comportamiento exitoso.

[Sierra et al., 1997] Sierra, C., Jennings, N., Noriega, P., y Parsons, S. (1997). A framework for argumentation based negociation. En *In Proceedings of the 4th International Workshop on Agent Theories, Achitectures ang Languages*, págs. 167–182, Rhode Island, EEUU.

Este artículo presenta un framework general para la negociación entre agentes, en el cual los agentes intercambias propuestas y contra-propuestas respaldadas por argumentos que resumen las razones por las cuales una determinada propuesta debe ser aceptada. En este escenario las propuestas que se intercambian entre distintos agentes son capaces de modificar el estado mental de los mismo, por lo cual los autores denominan este tipo de argumentación como *persuasiva*. Definen además un lenguaje expresivo de comunicación entre agentes. El sistema resultante se ilustra mediante ejemplos en el área de los procesos de negocios.

[Simari, 1989] Simari, G. R. (1989). *A Mathematical Treatment of Defeasible Reasoning and its Implementation*. Tesis Doctoral, Washington University, Department of Computer Science, Saint Louis, Missouri, EE.UU.

Esta tesis presenta como contribución principal un sistema basado en argumentos para modelar el razonamiento de un agente inteligente. La definición de este sistema fue posteriormente refinada en [Simari y Loui, 1992], en donde se lo bautizó como sistema MTDR. En su tesis doctoral Simari realiza además una implementación computacional del sistema MTDR restringido a claúsulas de Horn, denominada **justification finder**.

[Simari et al., 1994] Simari, G. R., Chesñevar, C. I., y García, A. J. (1994). The Role of Dialectics in Defeasible Argumentation. En *Anales de la XIV Conferencia Internacional de la Sociedad Chilena para Ciencias de la Computación*, págs. 111–121, Concepción (Chile). Universidad de Concepción.

Este trabajo define un sistema que constituye una evolución del formalismo Simari-Loui en donde la definición inductiva del proceso de justificación es parafraseada en términos del concepto de árboles dialécticos. Esta nueva presentación posibilitó el análisis de diversos casos de argumentación falaz y el desarrollo de soluciones para las mismas dentro del sistema considerado.

[Simari y Loui, 1992] Simari, G. R. y Loui, R. P. (1992). A Mathematical Treatment of Defeasible Reasoning and its Implementation. *Artificial Intelligence*, 53(2–3):125–157.

Este artículo sintetiza las principales características del sistema argumentativo Simari-Loui, definido oportunamente por G.R.Simari en su tesis doctoral. Este formalismo combina la teoría general de justificación desarrollada por J. Pollock [Pollock, 1987] con la definición de especificidad de D. Poole [Poole, 1985]. Se incluyen distintos ejemplos que ilustran el comportamiento del sistema y se analizan sus propiedades. Al finalizar el árticulo los autores proponen distintas lineas de investigación futura que han dado origen a destacados formalismos, como es el caso de la programación en lógica rebatible [García, 1997].

[Stankevicius et al., 2002] Stankevicius, A. G., Capobianco, M., y Chesñevar, C. I. (2002). Logical properties in defeasible logic programming – a preliminary report –. En *Proceedings del 9no Workshop en Aspectos Teóricos de la Inteligencia Artificial (ATIA), 4to Workshop de Investigadores en Ciencias de la Computación (WICC)*, págs. 61–65, Bahía Blanca. Universidad Nacional del Sur.

Este artículo analiza la postura de la programación en lógica rebatible [García, 2000] con respecto a las propiedades de inclusión, idempotencia, cumulatividad y preservación de consistencia. Se presentan ejemplos de cada situación en particular.

[Stankevicius et al., 2003] Stankevicius, A. G., Capobianco, M., y Chesñevar, C. I. (2003). An architecture for rational agents interacting with complex environments. A ser publicado.

Este arículo analiza como diseñar e implementar una arquitectura de agentes de software basada en el uso de la PLRO como mecanismo de razonamiento y representación de conocimiento que además utilice el mecanismo de interacción propuesto en [Stankevicius y Simari, 2001]. De esta forma es posible obtener una arquitectura que permita modelar tanto la interacción entre agentes como la percepción de un ambiente cambiante, actualizando el conocimiento en forma dinámica.

[Stankevicius et al., 1999] Stankevicius, A. G., García, A. J., y Simari, G. R. (1999). Una arquitectura para la ejecución de Programas Lógicos Rebatibles. En *Procedings of the 5th International Congress on Informatics Engineering*, págs. 450–461, Capital Federal. Universidad de Buenos Aires.

Las arquitecturas definidas para la ejecución de programas lógicos (como por ejemplo Prolog) utilizan para su implementación una máquina abstracta. La programación en lógica rebatible [García, 1997, García, 2000] cuenta con una máquina abstracta definida a tal efecto que ha sido denominada Justification Abstract Machine (JAM). Este artículo presenta una arquitectura apropiada para la ejecución de programas lógicos rebatibles [García, 1997], basada en la JAM. Los autores desarrollan además una implementación particular de la arquitectura propuesta utilizando el lenguaje Java, que posee funcionalidades por demás interesantes.

[Stankevicius y Simari, 2001] Stankevicius, A. G. y Simari, G. R. (2001). An Abstract Model for the Process if Deliberation within Multiagent Systems. En *Procedings del 7mo Congreso Argentino de Ciencias de la Computación*, págs. 1017–1026, El Calafate. Universidad Nacional de la Patagonia Austral.

En este artículo se presenta un modelo abstracto para la deliberación en sistemas multiagentes basado en el uso de argumentación rebatible. Se detallan las definiciones formales y se muestran ejemplos de uso del sistema en escenarios concretos.

[Stolzenburg et al., 2000] Stolzenburg, F., García, A., Chesñevar, C. I., y Simari, G. R. (2000). Introducing Generalized Specificity in Logic Programming. En *Proceedings del VI Congreso Argentino en Cs. de la Computación*, págs. 359–370. CACIC, Ushuaia, Argentina.

Este artículo presenta un análisis detallado del concepto de especificidad y su definición en diferentes trabajos. Se desarrolla además una nueva versión de especificidad para la programación en lógica rebatible.

[Sycara, 1990] Sycara, K. (1990). Persuasive argumentation in negotiation. *Theory and Decision*, 28(3):203–242.

En este artículo se desarrolla un framework para resolución de conflictos mediante negociación. El modelo utilizado integra técnicas de inteligencia artificial y toma de desiciones en particular. Este formalismo ha sido implementado en PERSUADER, un programa que actua como mediador en disputas sobre una base común preestablecida.

[Toulmin, 1958] Toulmin, S. (1958). *The uses of argument*. Cambridge University Press.

Este libro consiste en una serie de ensayos que desarrollan una crítica a los filósofos que utilizan un razonamiento puramente deductivo. El autor manifiesta que la lógica tradicional no resulta adecuada para modelar el razonamiento. En el más prominente de estos ensayos se aborda la estructura interna los argumentos. Utilizando una analogía con la jurisprudencia, Toulmin propone una representación en la cual un argumento esta constituido básicamente por cuatro elementos: una *afirmación*, una *normativa*, un conjunto de *datos* y un *fundamento* de la normativa. En general esta obra aborda varios tópicos interesantes. Por caso, los argumentos utilizan inferencias de tipo deductiva, rebatible y probabilística, resultando en un modelo completo e interesante desde el punto de vista filosoófico.

[Verheij, 1996] Verheij, B. (1996). *Rules, Reasons and Arguments: formal studies of argumentation and defeat*. Tesis Doctoral, Maastricht University, Department of Computer Science, Maastricht, Holanda.

Esta disertación contiene un análisis de la argumentación rebatible. Se realiza una clasificación de los principales elementos presentes los sistemas argumentativos, abordando las nociones de reglas, argumentos, conflictos y fuerza conclusiva entre otros. A continuación se describe un acercamiento para la formalización de reglas denominado 'Reason-based logic'. Este se complementa con la definición de un formalismo de argumentación por etapas, CumulA, que modela el proceso de argumentación. Finalmente se presenta una comparación entre CumulA y diversos sistemas existentes.

[Vreeswijk, 1997] Vreeswijk, G. (1997). Abstract Argumentation Systems. *Artificial Intelligence*, 90(1–2):225–279.

Este artículo es un refinamiento de la teoría de los sistemas argumentativos abstractos desarrollada en [Vreeswijk, 1993]. Estos formalismos se distinguen por no especificar tanto el lenguaje subyacente como el criterio de preferencia que se utiliza para decidir entre los argumentos en conflicto. El autor se encarga principalmente de definir el proceso de justificación y estudiar sus características. Además se enuncian distintas versiones del proceso de inferencia que son relacionadas entre sí mediante un conjunto de teoremas y se describen numerosos ejemplos para evaluar la aplicabilidad del sistema ante diversos problemas tradicionales en la comunidad de razonamiento no monótono.

[Vreeswijk, 1993] Vreeswijk, G. A. W. (1993). *Studies in Defeasible Argumentation*. Tesis Doctoral, Vrije University, Department of Computer Science, Amsterdam, Holanda.

En esta Tesis Vreeswijk realiza un estudio de la argumentación en general y presenta una versión preliminar del sistema posteriormente publicado en [Vreeswijk, 1997]

[Wooldridge y Jennings, 1995] Wooldridge, M. y Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152.

En este artículo los autores detallan un conjunto de cuestiones teóricas y prácticas de gran preponderancia con respecto a la construcción y diseño de agentes inteligentes. Estas cuestiones se dividen principalmente en tres áreas: teoría de agentes, arquitectura de agentes y lenguajes orientados a agentes. La teoría de agentes concierne una definición formal de qué es un agente y el uso de formalismos matemáticos para representar las propiedades de un agente y razonar sobre ellas. Las arquitecturas de agentes pueden interpretarse como modelos de ingeniería de software para representar agentes. Por ultimo los lenguajes orientados a agentes son sistemas de software para programar agentes y experimentar con ellos. Wooldridge y Jennings concluyen este trabajo con una resumen de las principales aplicaciones para agentes existentes y un análisis de aquellas que podrían desarrollarse en un futuro cercano.

Índice alfabético