UNIVERSIDAD NACIONAL DEL SUR

Tesis de Doctor en Ciencias de la Computación

Una semántica basada en juegos para la Programación Lógica Rebatible

Laura Andrea Cecchi

Bahía Blanca Argentina

2010

UNIVERSIDAD NACIONAL DEL SUR

Tesis de Doctor en Ciencias de la Computación

Una semántica basada en juegos para la Programación Lógica Rebatible

Laura Andrea Cecchi

Director: Dr. Guillermo R. Simari

Bahía Blanca Argentina

2010

Prefacio

Esta Tesis se presenta 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 otra. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el ámbito del Departamento de Ciencias e Ingeniería de la Computación, durante el período comprendido entre el 22 de octubre de 2001 y el 15 de diciembre de 2010, bajo la dirección del Dr. Guillermo R. Simari, Profesor Titular del Departamento de Ciencias e Ingeniería de la Computación.

Laura Andrea Cecchi
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Bahía Blanca, 15 de diciembre de 2010.

UNIVERSIDAD NACIONAL DEL SUR

Secretaría General de Posgrado y Educación Continua

La presente tesis ha sido aprobada el día 30/09/2011, mereciendo la calificación de 10 (sobresaliente).

A Pablo, mi esposo.
A mis hijos, Fabrizio y Luciano.

Agradecimientos

Quiero comenzar por agradecer a mi director Dr. Guillermo Simari, quien tuvo la paciencia para guiarme, quien me enseñó que la investigación es divertida y quien hizo que ame mi trabajo. A él debo agradecer el empuje, la motivación y mi formación en investigación.

Al Dr. Pablo Fillottrani, de la Universidad Nacional del Sur, profesor y amigo, con quien compartimos muchas charlas, café y coquitos de por medio, quiero agradecer el ánimo brindado y sus aportes permanentes a mis trabajos.

A la Profesora Nelly Jenkins, del Departamento de Matemática, Facultad de Economía y Administración, de la Universidad Nacional del Comahue, quiero agradecer su predisposición permanente para responder mis consultas y todas las charlas que mantuvimos, que me hicieron crecer como docente e investigadora.

Al Dr. Alejandro García y al Dr. Carlos Chesñevar, de la Universidad Nacional del Sur, deseo agradecer las interesantes discusiones que hemos tenido sobre el tema.

Estoy en deuda con todos mis compañeros docentes de nuestra flamante Facultad de Informática, de la Universidad Nacional del Comahue, por ser parte de mi formación de grado y por fomentar el sueño de la investigadora que soy. Quiero agradecer en particular a Jorgelina Giorgetti y Laura Sánchez por apoyarme incondicionalmente. Gracias a mis compañeros del Departamento de Teoría de la Computación, por el alegre entorno de trabajo y por las discusiones enriquecedoras que tenemos en forma permanente, particularmente con Claudio Vaucheret, Gerardo Parra y Sandra Roger.

Gracias a todos mis amigos, que se preocuparon por cómo los cambios en mi vida personal afectaban la finalización de esta etapa y que siempre me brindaron una dosis suficiente de optimismo para continuar.

A Pablo, Fabrizio y Luciano, quiero agradecer la paciencia, la fuerza y las alegrías que me dan en forma continua.

A mis padres deseo agradecerles la posibilidad de estudiar una ciencia que amo profundamente.

Finalmente, a mis doctoras, mis hermanas, Amalia y Griselda, que estuvieron presentes en todos los momentos importantes de mi vida.

Resumen

El objetivo principal de esta tesis es estudiar la teoría de prueba de la Programación en Lógica Rebatible (P.L.R.) y brindar una caracterización declarativa equivalente. La P.L.R. es una herramienta valiosa para la representación de conocimiento tentativo, incierto y potencialmente inconsistente, que provee un mecanismo de inferencia basado en los sistemas argumentativos. En los últimos años, los sistemas argumentativos han comenzado a ser utilizados en diversos campos de aplicación como la web y sistemas multiagentes.

En esta Tesis se presenta una caracterización declarativa basada en la teoría de modelos y en la noción de juegos, de la P.L.R.. La semántica declarativa trivaluada desarrollada, que se denomina $\mathcal{GS}$, es sensata y completa con respecto a la teoría de prueba de P.L.R.. Como punto intermedio, se brinda una formalización declarativa equivalente de la estructura de argumento y se circunscribe el conjunto de todos los posibles argumentos a favor y en contra que pueden construirse para una consulta dada bajo un programa lógico rebatible.

A partir de los resultados obtenidos, se realizó un estudio de la complejidad computacional de la P.L.R.. En este sentido, se definieron problemas de decisión relevantes con respecto a los juegos bajo el contexto de un programa lógico rebatible y se calculó la complejidad computacional de la existencia de argumentos y contraargumentos. Asimismo, considerando el nexo existente entre la Programación en Lógica y las bases de datos deductivas se definieron las complejidades de datos, expresión y combinada, y se estableció una cota superior para la complejidad de datos. Dichos resultados nos dan un indicio para determinar el poder expresivo de la P.L.R..

Abstract

The main goal of this thesis is to study Defeasible Logic Programming (DeLP) proof theory and to provide an equivalent declarative characterization. DeLP is a valued tool for representing tentative, uncertain, and potentially inconsistent knowledge, that provide an inference mechanism based on argumentative systems. In the last decade, argumentative systems have been used in different applications fields, such as the web and multiagent systems.

In this thesis we present a declarative caracterization based in model theory and in game concepts of Defeasible Logic Programming (DeLP). The trivalued declarative semantics developped, noted $\mathcal{GS}$, is sound and complete with respect to DeLP proof theory. As an intermediate point, we provide an equivalent declarative formalization of argument structure and we circunscribe the set of all possible arguments for and against that can be constructed for a query under a defeasible logic program.

From the results we have obtained, we carry out an study of DeLP computational complexity. We have defined relevants decision problems with respect to the construction of a game under a defeasible logic program and we calculate the computational complexity of arguments and conterargument existence. Furthermore, considering the link between Logic Programmning and deductive data bases we have defined data complexity, expression complexity and combined complexity and we have established a upper bound for data complexity. These results give us a trace for determining DeLP expressive power.

Índice general

Capítulo 1

Introducción

Contenido

Syntax without semantics is blind, semantics without syntax is empty, once said...

En este capítulo, se introducen el contexto y la motivación que guiaron el desarrollo de esta investigación y que dieron como resultado la semántica basada en juegos $\mathcal{GS}$ y el estudio de la complejidad computacional de la Programación en Lógica Rebatible a través $\mathcal{GS}$ y de la teoría de prueba. Asimismo, se sintetizan las principales contribuciones que se han obtenido y que han sido publicadas en congresos nacionales e internacionales. Finalmente, se resume la organización de esta Tesis.

1.1. Motivación

Los humanos razonamos de forma no monotónica, es decir, nos permitimos cambiar nuestras ideas a partir de nueva información. Somos capaces de concluir algo, actuar de acuerdo a ello y luego, frente a nueva evidencia, retractar nuestras conclusiones. Un sistema de razonamiento formal, o lógica, es monótono si y sólo si todo lo que es probable a partir de un conjunto de hipótesis H es probable a partir de cualquier superconjunto de H.

Si queremos modelar agentes que produzcan un *resultado similar al comportamiento humano*, el enfoque adecuado es facultarlos con la *flexibilidad* del razonamiento no monotónico.

Por lo tanto, no es sorprendente que varios sistemas formales no monótonos hayan sido desarrollados e implementados, con el fin de caracterizar el razonamiento humano. Las lógicas no monótonas son particularmente útiles para alcanzar conclusiones tentativas a partir de información incompleta, incierta y potencialmente inconsistente.

En este contexto, durante la década de los '80 surgieron diversos formalismos no monotónicos: *Lógica Default*[Rei80], *Circunscripción*[McC80] y *Lógica No Monótona*[MD80]. Su atractivo fundamento matemático, en algunos casos, influía en la complejidad de su implementación. Por ejemplo, la inferencia en circunscripción proposicional es Π_2^P-completo, mientras que el mismo problema en lógica proposicional clásica es co-NP-completo.

En este sentido, no solo deseamos que un agente inteligente se comporte de la forma esperada, sino que tenemos la expectativa, de que la performance del razonamiento automatizado a través de la inclusión de principios de razonamiento no monotónico, sea la adecuada.

La argumentación es una forma básica muy común de comunicación entre humanos que puede darse a través de una discusión cortés y constructiva o de una querella vil. Un agente puede argumentar a favor de un hecho pero frente a la presencia de nueva información debe ser capaz de invalidar ese argumento. Los sistemas argumentativos presentan una forma de razonamiento rebatible y surgen como una propuesta de formalismos no monotónicos. Actualmente, la teoría de diálogos y la argumentación es de gran interés para la comunidad de Ciencias de la Computación por su aplicación en el razonamiento de agentes y en la negociación en sistemas multiagentes [RS09]. Los sistemas argumentativos son aplicables en razonamiento legal, sistemas de mediación, sistemas de toma de decisión y son especialmente útiles para formalizar el razonamiento entre agentes. El desafío de entender la argumentación y su rol en el razonamiento humano ha sido enfocado por varios investigadores en diferentes campos incluyendo la filosofía, la lógica y la Inteligencia Artificial.

Una herramienta de representación de conocimiento cuyo sistema de razonamiento está basado en la argumentación es la Programación en Lógica Rebatible [GS04]. La Programación en Lógica Rebatible es una extensión de la Programación en Lógica que permite representar conocimiento contradictorio, a través de la noción de negación fuerte, y conocimiento tentativo, incorporando a la sintaxis una nueva clase de reglas: las reglas rebatibles.

La teoría de prueba de la Programación en Lógica Rebatible se basa en un análisis dialéctico donde argumentos a favor y en contra de un literal interactúan a fin de determinar si un agente que razona bajo este formalismo cree en ese literal. De este modo, una consulta q tendrá exito si existe un argumento $\mathcal{A}$ de q que lo justifique, i.e., no existen contraargumentos que derroten a $\mathcal{A}$.

La justificación de un literal puede ser vista como un juego en donde un jugador propone

un argumento para una meta q mientras que el otro jugador, el oponente, trata de buscar contraargumentos que lo derroten. De este modo, podemos modelar al sistema de dialéctica a través de la interacción de dos jugadores.

El vínculo entre debate y juegos se remonta a las obras de Aristóteles, *Tópicos* (reglas para la argumentación y el debate eficientes) y *De Sophisticis Elenchis* (falacias informales), en las que introduce a la dialéctica como un proceso para arribar a creencias racionales. Paul Lorenzen [Lor58] y Charles Hamblin [Ham70] revivieron, en forma independiente, esta relación entre diálogo y reglas del razonamiento.

En el área teórica de la computabilidad, el estilo operacional es crucial, precediendo al estilo declarativo tanto histórica como conceptualmente. En este sentido, la escuela de la argumentación rebatible no es la excepción, aunque en los últimos años, la semántica operacional ha sido estudiada desde un punto de vista declarativo [Dun95, KT99], con el objeto de determinar el significado preciso de los formalismos sin recurrir al control.

Esto motivó el desarrollo de una semántica declarativa, que denominamos $\mathcal{GS}$, para la Programación en Lógica Rebatible sin negación por falla, basada en la noción de juego. En [Abr97] y [AM97], se introduce a la semántica basada en juegos con el objeto de modelar a la computación como un juego entre dos participantes. La idea presentada es la de utilizar un juego para modelar la *interacción* entre el Sistema y el Ambiente.

De la similitud entre un juego y un debate surge $\mathcal{GS}$, una semántica declarativa trivaluada basada en juegos para la Programación en Lógica Rebatible, que combina la teoría de modelos y el concepto de juego.

Nuestro interés es un razonamiento práctico, es decir, no solamente deseamos que nuestro agente dé una respuesta, sino que lo haga en un tiempo razonable. Esto motivó el estudio de la complejidad computacional del sistema, el que se llevó a cabo a través de las definiciones declarativas en las que se fundamenta $\mathcal{GS}$ y de algunos mecanismos operacionales de la teoría de prueba. Por otra parte, las aplicaciones de la Programación en Lógica Rebatible como un lenguaje de consulta, incentivaron el análisis de la complejidad computacional del sistema basado en la teoría de bases de datos. Considerando que en el razonamiento argumentativo, la complejidad de la inferencia depende fuertemente del tamaño de la base de conocimiento a partir de la cual se construyen los argumentos, el punto inicial de estudio fue la Complejidad de los Datos. Hasta donde conocemos, este estudio es novedoso en los sistemas argumentativos.

A partir de los resultados obtenidos, hemos podido contrastar la eficiencia en la construcción de argumentos, con algunos de los sistemas argumentativos que ya han sido estudiados desde el punto de vista de su complejidad computacional.

1.2. Contribuciones

Del estudio de la Programación en Lógica Rebatible (P.L.R.) y de las semánticas basadas en juego confluye una de las principales contribuciones de esta Tesis: la definición formal de la semántica basada en juegos $\mathcal{GS}$ para la P.L.R..

En primer lugar se analizó la teoría de prueba de la P.L.R. y se introdujo una definición declarativa equivalente a la de estructura argumentativa, basada en la noción de consecuencias de un programa lógico definida en [Lif96]. Se demostró que dicho concepto está bien definido, se probaron diversas propiedades de este conjunto y finalmente se mostró que las definiciones declarativa y procedural son equivalentes.

A partir del desarrollo anterior se caracterizó en forma declarativa a la teoría de prueba de la P.L.R. circunscripta a programas lógicos rebatibles donde las reglas no contuvieran negación por falla ni disyunciones. En este sentido, se formalizó el juego a través de sus partes: jugadores, arena, movimientos permitidos(piezas), reglas, turnos y vencedores.

En lo que respecta a los movimientos permitidos, se delimitó el conjunto de los posibles argumentos que podrían ser jugados en un juego comenzado por un argumento en particular y se establecieron y demostraron propiedades asociadas al conjunto.

La semántica $\mathcal{GS}$ fue definida en forma independiente del criterio de preferencia entre los argumentos, aunque históricamente la P.L.R. ha sido implementada utilizando a la especificidad como relación de preferencia.

Se demostró la sensatez y la completitud entre la definición formal de $\mathcal{GS}$, basada en la teoría de modelos, y la teoría de prueba de la P.L.R..

En la comparación con otros sistemas, se hizo un fuerte análisis con la definición alternativa de la P.L.R. presentada en [VTS08], en el que se demostró que una de las respuestas posibles del sistema, cuando se la analiza desde el enfoque de la P.L.R.B., puede ser mejorada en su implementación, ya que no requiere del análisis de dos árboles de decisión.

La segunda contribución relevante de esta Tesis es el estudio de la P.L.R. desde el punto de vista de la complejidad computacional. Se introdujeron problemas de decisión de importancia con respecto a la P.L.R. basados en la semántica $\mathcal{GS}$ y se calcularon la complejidad computacional de la existencia de argumentos y contraargumentos.

Asimismo, se comenzó el estudio de la P.L.R. como un lenguaje de consulta, se introdujo las complejidades de datos, de expresión y combinada, un concepto nuevo en el área de los sistemas argumentativos, heredada de la teoría de bases de datos. En este sentido, se calculó una cota superior para la complejidad de datos que permitiría hacer un análisis sobre el poder expresivo

de la P.L.R..

1.3. Organización de la Tesis

La tesis incluye los contenidos necesarios para que su lectura sea autocontenida, asumiéndose conocimientos a nivel de grado de lógica clásica e Inteligencia Artificial y nociones avanzadas de complejidad computacional. A continuación se describe la estructura de esta Tesis, organizada en siete capítulos y un apéndice.

Capítulo 1: Introducción Este capítulo describe las motivaciones y las contribuciones de la Tesis.

Capítulo 2: Sistemas Basados en Juegos Se analiza la evolución hacia los sistemas interactivos y se describe la semántica de la interacción. En este marco, se presentan dos enfoques distintos de las semánticas basadas en juegos.

Capítulo 3: Programación en Lógica Rebatible Se presenta la definición formal de la P.L.R., analizada desde el punto de vista sintáctico: el lenguaje y la teoría de prueba. Se introduce una clasificación de la P.L.R., que es heredada de la P.L. clásica. Se describen diversas propiedades del sistema.

Capítulo 4: Semántica de la P.L.R.B. basada en juegos Se introduce una de las contribuciones principales de la Tesis: la definición formal de la semántica declarativa basada en juegos para la P.L.R.. Se analiza en primer lugar a la estructura de argumentos, núcleo de la argumentación rebatible y luego se definen cada uno de los elementos que conforman el juego. Se demuestra la equivalencia entre las definiciones declarativa y procedural de argumento y otras propiedades relevantes. Finalmente, la sensatez y la completitud entre la semántica definida y la teoría de prueba son probadas.

Capítulo 5: Comparación con otros formalismos Se analizan distintos enfoques de la formalización de los sistemas argumentativos, contrastándose fundamentalmente en la definición de la semántica. La intención es cotejar diferentes enfoques que tienen en común la noción de juego o bien una similitud en la teoría de prueba.

Capítulo 6: Complejidad de la P.L.R.B. Se analiza a la P.L.R. a fin de determinar los problemas de decisión que son relevantes en este formalismo. Se calcula la complejidad computacional de los problemas de decisión relacionados con la existencia de argumentos y contraargumentos. Se estudia a la P.L.R.B. como un lenguaje de consulta, introduciendo conceptos heredados de la teoría de bases de datos como la complejidad de datos, de expresión y combinada. Finalmente, se estudia la complejidad de datos presentando una cota superior a su valor.

Capítulo 7: Conclusiones Se detallan los resultados y conclusiones de esta Tesis y se delinean líneas de investigación aún pendientes de estudio.

Bibliografía: Recopilación de las referencias bibliográficas de todos los capítulos y del apéndice, ordenadas alfabéticamente por autor.

Apéndice: Conjuntos Ordenados Recopilación del tema, analizado desde el punto de vista de la Ciencia de la Computación. Descripción de los conceptos básicos y propiedades que son utilizados en el desarrollo de esta Tesis, particularmente en el capítulo 4.

Índices de Temas y de Autores: Índice ordenado alfabéticamente de los principales temas tratados en esta Tesis y de los autores referenciados.

Capítulo 2

Semánticas basadas en juegos

Contenido

En este capítulo se presenta un estudio de las semánticas basadas en juegos. En la primer sección, se analiza la evolución hacia los sistemas interactivos. Asimismo, se explican las adaptaciones realizadas por Wegner, a las Máquinas de Turing, resultando en las Máquinas de Interacción Secuenciales. Estas últimas máquinas caracterizan la computación de la nueva generación de sistemas: los sistemas interactivos.

En la siguiente sección, se presenta la *semántica de la interacción*, aplicación de las semánticas basadas en juegos, que motivó el desarrollo de esta tesis.

A continuación se describen dos enfoques de juegos, HO-juegos y AJM-juegos. En ambos casos, se estudian los movimientos legales del juego, las estrategias sobre los juegos y el modo de construir nuevos juegos a partir de otros.

Finalmente, se presentan las conclusiones del capítulo.

2.1. Sistemas Interactivos

En los últimos treinta años ha habido una evolución en la Ciencia de la Computación que se ha reflejado, en forma paralela, en diferentes campos. La tecnología de la arquitectura de computadoras ha progresado desde los mainframes hacia las redes y workstations. El campo de la Ingeniería de Software se ha mudado sucesivamente de los sistemas orientados a procedimientos a los sistemas orientados a objetos y a los basados en componentes. La evolución de la Inteligencia Artificial, que va de la programación basada puramente en lógica y en búsqueda a la programación orientada a agentes, se evidencia en la investigación realizada actualmente sobre agentes[HS98] y sobre planificación y control interactivo y, finalmente, en una reformulación sistemática de la IA en términos de agentes inteligentes[RN09]. Estas transformaciones se han reflejado en el cambio producido en los sistemas, los que han variado hacia aquellos basados en interacción[Weg96].

En 1900, David Hilbert planteó el *Entscheidungsproblem* que preguntaba si existía un método para establecer el valor de verdad de cualquier sentencia del Cálculo de Predicados. En la década del 30, el brillante matemático Alan Turing [Tur36] resolvió dicho problema, estableciendo que era insoluble. Para llegar a esta conclusión debió reflejar la idea intuitiva de *método* o *algoritmo* en un modelo formal de lo efectivamente computable: *las máquinas de Turing*. De este modo, se estableció una caracterización de lo que puede ser computado a través de algoritmos: *las funciones computables*. La idea de que todos los lenguajes que puedan computar cualquier función computable por una máquina de Turing son igualmente expresivos se conoce como *Turing tar-pit*[1].

El modelo propuesto por Turing es el de *la mente humana realizando un cálculo*. Sólo reconoce un conjunto finito de símbolos, lee de a una cantidad fija de símbolos por vez ("No podemos decir con un vistazo si 9999999999999999 y 999999999999999 son iguales"[Tur36, Hod99]), tiene un conjunto finito de estados posibles y la operación que realiza está regida por lo que Turing denominó *tabla de comportamiento* que depende del estado mental del calculista y de los símbolos observados. Informalmente, las máquinas de Turing constan de una unidad de control finita que posee una cabeza lecto-escritora que actúa sobre una cinta infinita. La

[1]La traducción de *tar-pit* es la de un pozo que contiene brea del que no se puede escapar. Por no encontrar un vocablo en castellano que exprese el significado pretendido, utilizaremos el término en inglés.

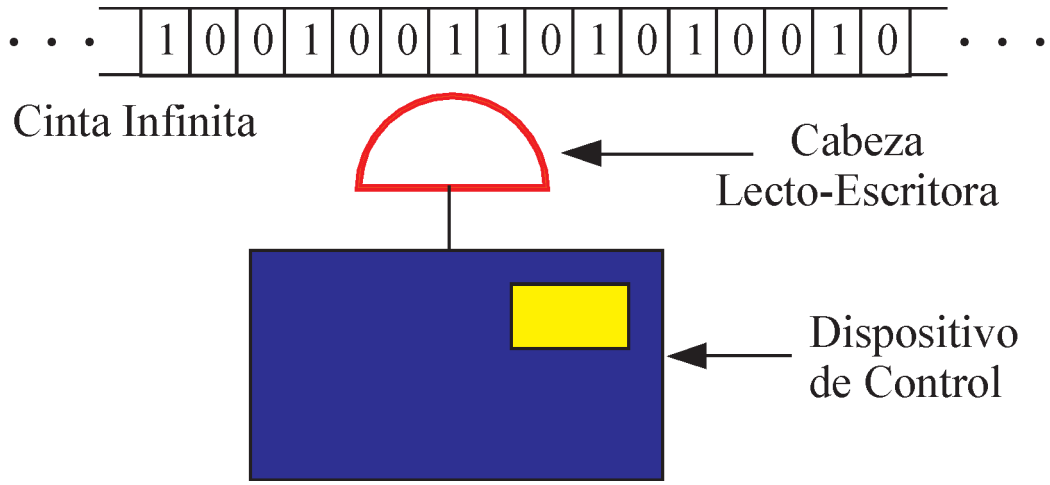


Figura 2.1: Máquina de Turing.

cabeza lecto-escritora puede realizar movimientos en ambas direcciones. La comunicación entre la cinta y el dispositivo de control la realiza la cabeza lecto-escritora a través de pasos discretos: leer un símbolo, escribir un símbolo, realizar un movimiento (ver figura 2.1).

Estas máquinas transforman cadenas de símbolos de entrada, que se encuentran en la cinta, en cadenas de símbolos de salida a través de secuencias de transiciones entre estados. Sin embargo, estas máquinas no pueden aceptar entradas externas mientras realizan un cómputo, i.e., están cerradas a lo que sucede en el mundo exterior y por lo tanto, no están capacitadas para modelar el pasaje del tiempo externo. Por esta razón, Wegner asegura que estas máquinas muestran un comportamiento no interactivo, lo que no les permite modelar sistemas como los de reserva de vuelos en aerolíneas, robots y el World-Wide Web, que tienen ciertas características claves: agentes interactuando entre sí (cooperan, negocian y compiten entre ellos o con el mundo) e información fluyendo a través del sistema.

La primera generación de modelos de computación, heredada de la matemática y de la lógica, afirma que los programas pueden ser vistos como una caja cerrada en la que se realiza el cómputo de funciones o relaciones de las entradas en las salidas, por lo que el mundo modelado es estático y debería ser completamente descripto por los argumentos de entrada. Según Wegner, los sistemas interactivos tienen un comportamiento más rico que los algoritmos y, por lo tanto, no puede ser capturado por estos modelos, ya que *la interacción no es expresable a través de una cadena de entrada inicial finita*. Así, Wegner y Goldin [Weg97, Weg98, WG99a, WG06]. sugieren que los cambios tecnológicos producidos se reflejan en el hecho de que *las Máquinas de Turing no son suficientemente poderosas como para modelar todo lo computable*.

En este sentido es interesante marcar el límite de lo que no puede ser expresado con un algoritmo. Así, los autores mencionados anteriormente, distinguen primero tres conceptos:

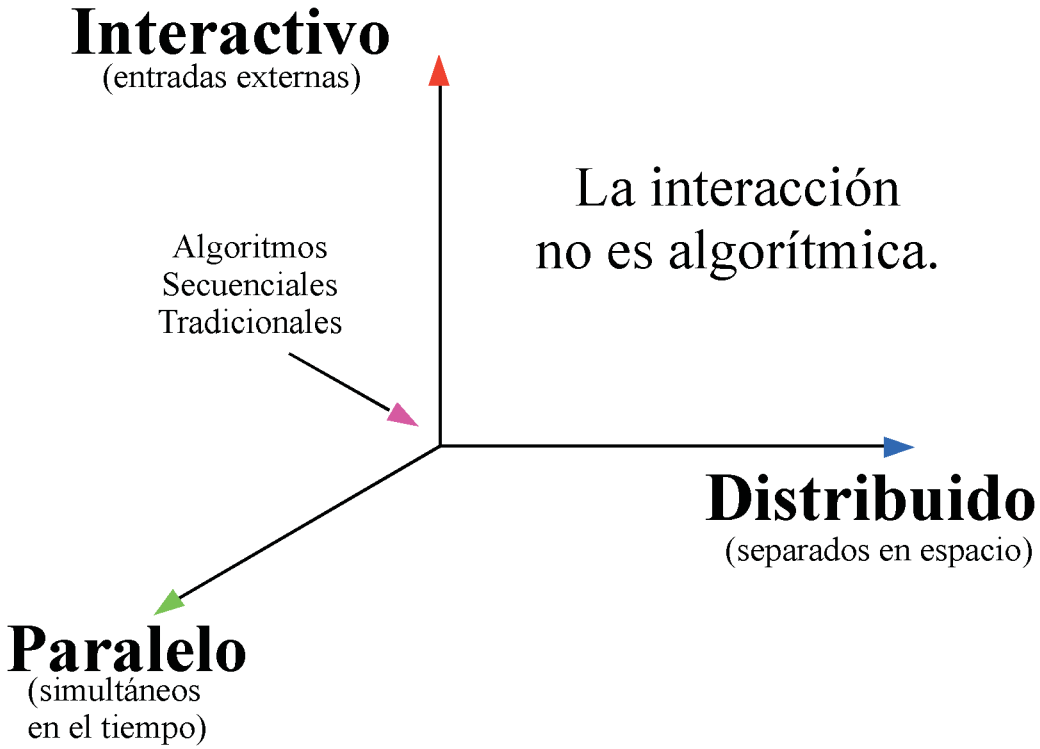


Figura 2.2: Espacio de los sistemas computacionales [Weg97].

1. Los *sistemas interactivos* interactúan con un ambiente externo dinámico que no pueden controlar.

2. El *paralelismo (concurrencia)* ocurre cuando la computación de un sistema se solapa en el tiempo.

3. La *distribución* ocurre cuando componentes de un sistema están separados geográficamente o lógicamente.

La computación distribuida y paralela en ausencia de interacción puede ser expresada con algoritmos. La figura 2.2, extraída de [Weg97], muestra el espacio de los sistemas. Aquellos que no están en el plano base son considerardos "no algorítmicos".

Wegner[Weg97] introduce una extensión de las máquinas de Turing, las *Máquinas Interactivas*, de ahora en más IM, que incorporan acciones dinámicas de entrada/salida, interactuando con su ambiente. Aunque las IM pueden variar en diferentes aspectos como tener flujos de entrada simples o múltiples o comunicación sincrónica o asincrónica, todas son sistemas abiertos que expresan el comportamiento externo dinámico que va más allá de lo computable por un algoritmo.

Las *Máquinas Interactivas Secuenciales (SIM)*[WG99b], a diferencia de las máquinas de Turing, son máquinas de transición de estados $M = (S, I, m)$ donde S es un conjunto *enumerable*

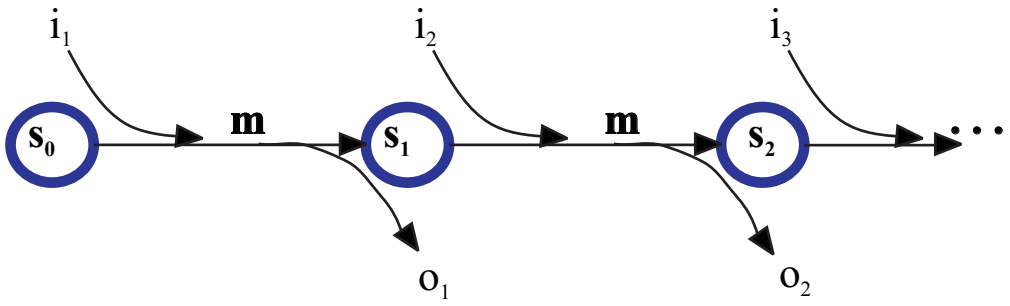


Figura 2.3: Máquinas de Interacción Secuenciales con entradas $i \in I$, salidas $o \in O$ y mapeo m.

de estados, I un conjunto enumerable de entradas (acciones) dinámicamente limitadas por *flujos* y el mapeo de transición $m : S \times I \rightarrow S \times O$ que mapea pares estado-acción en nuevos estados y salidas. El funcionamiento se muestra en la figura 2.3. Cada paso en el cómputo de una SIM $(s_j, i) \rightarrow (s_{j+1}, o)$ puede ser visto como el cómputo completo de una máquina de Turing donde i es una cadena de caracteres provisto dinámicamente, la salida o puede afectar subsecuentes entradas y s_{j+1} es el siguiente estado en la SIM.

Goldin et al. [GSAS04] presentan un modelo canónico de la interacción secuencial: la *Máquina de Turing Persistente (PTM)*. Una PTM es una Máquina de Turing que ejecuta una secuencia infinita de computaciones de una máquina de Turing "normal", donde cada computación comienza cuando la PTM lee una entrada de su cinta de entrada y termina cuando la PTM produce una salida sobre su cinta de salida.

La PTM contine una *cinta de trabajo persistente* W, cuyo contenido es preservado de un cómputo al siguiente y que representa un estado dependiente de la historia.Una máquina de Turing puede ser vista como una PTM *amnésica*, i.e., una PTM que ignora el contenido de su memoria.

Goldin et al. [GSAS04] en el mismo trabajo prueban que las PTM sin persistencia son menos expresivas que las PTM y haciendo una analogía con la Tesis de Church-Turing que relaciona las máquinas de Turing con la computación algorítmica, se realiza la hipótesis de que las PTM capturan la noción intuitiva de la computación interactiva secuencial.

Por otro lado, Wegner extiende la SIM a las *Máquinas Interactivas de Múltiples-Flujos (MIM)*[WG99b] con el objeto de formalizar la interacción distribuida. Estas máquinas modelan agentes finitos que interactúan con múltiples agentes autónomos, como por ejemplo, las bases de datos distribuidas que proveen servicios a varios clientes autónomos.

La noción intuitiva de cómputo de las SIM puede ser visto como un diálogo, un conductor que se adapta a las condiciones de la ruta o un *juego con dos participantes*. Por otro lado, las MIM modelan agentes finitos que interactúan con múltiples agentes autónomos, como por

ejemplo, las bases de datos distribuidas que proveen servicios a varios clientes autónomos, una negociación multi-agente o un *juego en equipo*.

En [Abr97], se presenta la semántica basada en juegos con la idea de modelar un cómputo interactivo, área cuyo estado del arte se presenta a continuación.

2.2. Semántica de la Interacción

La semántica basada en juegos ha sido utilizada para dar la primer construcción independiente de la sintaxis de modelos completamente abstractos de una variedad de lenguajes de programación. Existen actualmente dos enfoques: *HO-games* y *AJM-games*.

El primer modelo fue introducido por Hyland y Ong [HO94] y está basado en la idea de que un juego es jugado en una *arena*, que puede ser pensada como un "área o ambiente de juego" que establece ciertas reglas y convenciones básicas. El juego-diálogo en una arena está completamente determinado por el árbol de juego asociado. Este árbol de juego es especificado en dos etapas:

- Primero, la arena determina los movimientos del juego (que pueden ser *preguntas* o *respuestas*) y el orden de justificación o permiso entre los movimientos "preguntas".
- El árbol del juego es, luego, sistemáticamente generado a partir del conjunto de movimientos sujeto a ciertas reglas. Formalmente, el árbol del juego es representado por el conjunto de todas las *posiciones legales* indicadas con los caminos en el árbol.

Ya que no todos los movimientos *preguntas* tienen que estar disponibles al comienzo del juego, algunos de ellos pueden volverse disponibles a medida que el juego progresa. Excepto la pregunta inicial (que no necesita habilitación), un movimiento-pregunta sólo puede ser hecho si la jugada que lo habilita fue hecha antes. Esta noción de justificación o habilitación es formulada como un orden parcial entre preguntas.

El segundo enfoque se debe a Abramsky, Jagadeesan y Malacaria[AJM94] e incluye a los HO-games. En [Abr97], se introduce a las semánticas basadas en juegos con el objeto de modelar un cómputo interactivo. La característica clave de los juegos, en comparación con otros modelos de la computación, es que proveen una representación explícita del ambiente y, por lo tanto, se modela la interacción de un modo intrínseco. La noción de los juegos es formalizada para ser aplicada sobre dos participantes: el proponente y su oponente. En el juego, el proponente representará al *sistema* y el oponente al *ambiente*.

La razón de que la semántica analice sólo juegos de dos participantes está basada en la

cantidad de jugadores que se necesitan para interactuar. Sin ningún jugador sólo tendríamos valores de verdad instanciados y fijos. Con sólo un jugador, podríamos representar las acciones de tal participante. Con dos participantes, podremos modelar interacción. Por otra parte, la interacción entre varios participantes puede ser reducida al caso de dos jugadores.

Un cómputo simple involucra la interacción entre el proponente y el oponente y es representado por una secuencia de *movidas* que son hechas en forma alternada entre el sistema y el ambiente.

Ciertas secuencias de movidas en el juego se consideran *estrategias*. La noción intuitiva de las estrategias para jugar un juego se corresponde a la del comportamiento de un agente o proceso. Estas estrategias *interactúan* (composición de estrategias) jugando unas con otras, lo que puede ser visto como la interacción entre agentes.

La noción de correctitud lógica la obtenemos en términos de *estrategias ganadoras*, i.e., aquella que garantiza alcanzar un resultado "exitoso" sin importar cómo se comporte el ambiente. Esto, reemplaza o refina, en algún sentido, la noción lógica de "verdad": *las estrategias ganadoras son la versión dinámica de tautologías, o más precisamente, de pruebas.*

Los juegos pueden ser combinados para construir sistemas cuyo comportamiento sea más complejo. Así, el comportamiento de un sistema complejo, como aquel en el que interactúan agentes, puede ser entendido en términos del comportamiento de las partes. Las operaciones para construir nuevos juegos pueden ser vistas como formas dinámicas de *conectivos lógicos.*

Actualmente, se están estudiando semánticas probabilísticas basadas en juegos[DH00]. La filosofía de estas semánticas es que la estrategia se redefine como el libro de las reglas que indica cuáles movimientos pueden ser hechos por los jugadores y *con qué probabilidad.*

En este capítulo, se presenta la formalización de los juegos basados en el enfoque HO y en el enfoque AJM, y en particular, la adaptación del enfoque AJM utilizada en [Abr97] para modelar la interacción. Si bien los HO-juegos no serán utilizados como base para el desarrollo de la semántica que tiene como objetivo esta tesis, creemos que su incorporación en este capítulo ayudará a una mejor comprensión del enfoque alternativo y, por otra parte, dará una idea completa del estado del arte en este área. Con el objeto de precisar formalmente estos conceptos, primero introduciremos la notación a utilizar.

2.3. Notación

Antes de dar las definiciones formales de la semántica de juegos, introduciremos la notación a emplear. Sean Σ un alfabeto, s y t secuencias finitas de elementos de Σ, y a un elemento de

Σ. Entonces,

- Σ^* es el conjunto de todas las secuencias finitas sobre Σ. Cabe aclarar que el conjunto Σ^* es contable.
- ϵ denota la secuencia vacía.
- st denota la concatenación de las secuencias s y t.
- $[x_1, x_2, \ldots, x_n]$ denota una secuencia construida a partir de $\{x_1,\ x_2, \ldots,\ x_n\} \subseteq \Sigma$.
- $s[a]$ denota la secuencia obtenida agregando el elemento a a la secuencia s en la última posición. Si $s = \epsilon$ entonces $\epsilon[a] = [a]$.
- $|s|$ denota la longitud de una secuencia finita s, y s_i denota al i-ésimo elemento de s, para $1 \leq i \leq |s|$. En particular , $|\epsilon| = 0$.
- Dado $S \subseteq \Sigma^*$, S^{par} denota el conjunto de todas las secuencias s en S, tal que $|s|$ es par. Similarmente, S^{impar} denota el conjunto de todas las secuencias s en S, tal que $|s|$ es impar.

Presentamos a continuación dos conceptos que serán de gran utilidad para la definición de los enfoques AJM y HO de juegos.

Definición 2.1 (Prefijo de Secuencia)
Sean Σ un alfabeto, $S \subseteq \Sigma^*$ y $s,\ t,\ u\ \in\ \Sigma^*$. s es un *prefijo de* t, y lo notaremos $s \sqsubseteq t$, si $t = su$ para alguna secuencia u, posiblemente vacía. Pref(S) es el conjunto de todos los prefijos de los elementos de S. Diremos que S es *cerrado con respecto a prefijos* si $S = \text{Pref}(S)$. ■

Obsérvese que $\sqsubseteq$ forma un orden sobre las secuencias que es reflexivo y cuyo único primer elemento es la secuencia vacía ϵ.

Si $s_i \in \Sigma$ es un símbolo que ocurre en la secuencia s, con $1 \leq i \leq |s|$, denotaremos con $s_{\leq s_i}$ al prefijo de s hasta la posición i incluyendo a s_i, y con $s_{<s_i}$ al prefijo de s hasta la posición i, pero sin incluir a s_i.

Definición 2.2 (Secuencia cíclica)
Sean Σ un alfabeto y $s,\ t,\ u,\ v,\ w\ \in\ \Sigma^*$. Una secuencia s contiene un ciclo si $s\ =\ t\ u\ v\ u\ w$, para algunas secuencias t, u, v, w siendo que $u \neq \epsilon$. ■

En este punto estamos en condiciones de formalizar la idea intuitiva, ya presentada, de juegos.

2.4. HO-Juegos

En esta sección se presenta el enfoque HO de juegos introducido por Hyland y Ong en [HO94]. Este enfoque, contemporáneo del enfoque AJM, muestra cierta diferencia en el desarrollo de los juegos. Esto se hace visible, principalmente, en el hecho de que siempre que sea posible, las restricciones son desplazadas a las estrategias en vez de a los juegos. De este modo, restricciones que el enfoque AJM impone sobre las jugadas, bajo esta nueva forma de ver a los juegos, se transforman en propiedades de las jugadas que, luego, son utilizadas para restringir el comportamiento de las estrategias.

2.4.1. Arenas

La idea básica de la semántica de juegos al estilo HO es que un *juego* es jugado en una *arena*, la que puede ser pensado como un "área de juego" que impone ciertas reglas básicas y convenciones. Un juego diálogo que pueda ser jugado en una arena está completamente determinado por su árbol de juego asociado. La siguiente es la definición formal.

Definición 2.3 (Arena) [HO94, AM97]
Una *arena* A es especificada por la estructura $\langle \mathcal{M}_A, \lambda_A, \vdash_A \rangle$ donde

- $\mathcal{M}_A$ es el conjunto de *movimientos*. Un movimiento puede pensarse como una "burbuja" sin estructura interna, que se reconoce como un movimiento tan sólo por pertenecer a $\mathcal{M}_A$.
- $\lambda_A : \mathcal{M}_A \to \{O, P\} \times \{Q, A\}$ es la *función de etiquetamiento*, donde O, P, Q y A son símbolos distinguibles. El conjunto $\{O, P\} \times \{Q, A\}$ se corresponde con $\{OQ, OA, PQ, PA\}$. Utilizamos λ_A^{OP} para significar la proyección a izquierda de λ_A, de modo que $\lambda_A^{OP}(m) = O$ si $\lambda_A(m) = OQ$ o $\lambda_A(m) = OA$. Definimos λ_A^{QA} de modo similar. Así, $\lambda_A(m) = \lambda_A^{OP}(m)\lambda_A^{QA}(m)$.

  Sea $O_A = \{m \in \mathcal{M}_A | \lambda_A^{OP}(m) = O\}$ y $P_A = \{m \in \mathcal{M}_A | \lambda_A^{OP}(m) = P\}$. Ya que λ_A es una función total, $\{O_A, P_A\}$ es una partición de $\mathcal{M}_A$. Los conjuntos O_A y P_A definen los protagonistas en la arena: *Oponente* (O) y *Proponente* (P). Diremos que un movimiento m es un O-movimiento en la arena A si $m \in O_A$; de otro modo, m es un P-movimiento.

  Podemos utilizar λ_A^{QA} para definir otra partición de $\mathcal{M}_A$, que escribimos $\{Q_A, A_A\}$ estableciendo $Q_A = \{m \in \mathcal{M}_A | \lambda_A^{QA}(m) = Q\}$ y $A_A = \{m \in \mathcal{M}_A | \lambda_A^{QA}(m) = A\}$. Si $m \in Q_A$ decimos que es una *pregunta*; de otro modo es una *respuesta*.
- $\vdash_A$ es una relación de *habilitación o permiso* entre $\mathcal{M}_A + \{*\}$ (donde $*$ es sólo un símbolo comodín) y $\mathcal{M}_A$, que satisface:

(e1) Si $* \vdash_A m$ entonces $\lambda_A(m) = OQ$ y además se satisface que no existe ninguna otra secuencia que habilita a m, i.e., $n \vdash_A m$ si y sólo si $n = *$.

(e2) Si $m \vdash_A n$ y $\lambda_A^{QA}(n) = A$ entonces $\lambda_A^{QA}(m) = Q$.

(e3) Si $m \vdash_A n$ y $m \neq *$ entonces $\lambda_A^{OP}(m) \neq \lambda_A^{OP}(n)$.

■

La intuición de la función λ es indicar para cada movimiento quien es el participante que lo realiza: el Oponente (O) o el Proponente (P) y si es una pregunta (Q[2]) o una respueta (A[3]).

La relación de habilitación introduce *causalidad* en la arena. Cuando un movimiento es jugado, éste habilita a otros movimientos que pueden ser hechos subsecuentemente. Los movimientos no pueden ser jugados a menos que un movimiento de habilitación ya haya sido hecho. El habilitador $*$ es especial, ya que indica cuales movimientos son habilitados desde el principio. Un movimiento m tal que $* \vdash_A m$ es llamado *inicial*. Así, según el axioma (e1), los movimientos del Oponente pueden ser particionados en *iniciales* y *no iniciales*. Por lo tanto, el Oponente es quien comienza siempre el juego y lo hace con un movimiento *pregunta*.

El axioma (e2) indica que las *respuestas* siempre deben estar habilitadas por alguna *pregunta*. Sin embargo, no se restringe la habilitación de una pregunta por una respuesta.

Finalmente, el axioma (e3) expresa que los protagonistas del juego siempre se habilitan uno al otro, pero nunca se pueden habilitar ellos mismo. En otras palabras, si un movimiento habilita a otro movimiento entonces uno de esos movimientos es O y el otro es un movimiento P.

Ejemplo 2.1 La arena más simple es $\langle \varnothing, \varnothing, \varnothing \rangle$ que la denotamos con 1. La arena denotada $\perp$ es aquella que sólo tiene un O-movimiento q. ■

Definición 2.4 (Arena Plana)
Dado un conjunto contable X, construimos una *arena plana* sobre X con una pregunta simple del oponente q, que representa un pedido de un elemento de X y una respuesta del proponente x por cada $x \in X$. El O-movimiento q es inicial y habilita a todos los P-movimientos. ■

Ejemplo 2.2 Caracterizaremos tres arenas planas: aquellas generadas por un punto, dos puntos y un conjunto infinito contable. Estas arenas son denotadas por $\mathbb{C}$, $\mathbb{B}$ y $\mathbb{N}$ respectivamente. Tomaremos como convención los siguientes conjuntos generadores $\{a\}$, $\{tt, ff\}$ y el conjunto $\mathbb{N}_0 = \{0, 1, 2, ...\}$ de enteros no negativos. La arena $\perp$ puede ser pensada como la arena plana sobre $\varnothing$. ■

[2]Inicial de *query*.
[3]Inicial de *answer*.

2.4.2. Jugadas Legales

En una arena dada, permitiremos jugar ciertos juegos, i.e., estaremos interesados en cierta clase de movimientos. Así, una arena deberá cumplir ciertas reglas y principios.

Reglas Básicas **[HO94]:** una jugada que involucre a un Proponente y un Oponente deberá observar lo siguiente:

- Una jugada de un juego diálogo siempre la comienza el Oponente preguntando una pregunta inicial.
- Después de tal jugada inicial, la jugada alterna estrictamente movimientos O y P.
- Una jugada termina tan rápido como la pregunta inicial es respondida.

Principios de la Conversación Civil **:** cada jugada traza un diálogo de preguntas y respuestas que obedece los siguientes principios:

1. *Justificación.* Una pregunta q_1 es hecha sólo si en ese punto del diálogo, una instancia de la única pregunta q_0 que la justifica está pendiente, i.e., q_0 fue realizada pero no contestada aún. De igual modo, una respuesta es dada sólo si una instancia de una única pregunta con la que está asociada está aún pendiente.
2. *Prioridad.* Las preguntas pendientes en un diálogo son respondidas sobre la base "última pregunta-primero respondida": la pregunta que ha sido formulada última debe ser respondida primero.

Formalmente, estas características serán capturadas por las siguientes definiciones.

Definición 2.5 (Secuencia Justificada) [HO94]
Una *secuencia justificada* s en una arena A es una secuencia finita sobre el alfabeto $\mathcal{M}_A$, $s = [m_1, m_2, \ldots, m_n]$ tal que cada movimiento m_i tiene asociado un número natural μ_i llamado *puntero o índice de justificación* de m_i que satisface $\mu_i < i$ y las siguientes condiciones (w1)-(w3).Decimos que m_{μ_j} *justifica* a m_i o que m_i *está justificado por* m_{μ_j}.

(w1) *Pregunta inicial para comenzar.* El primer movimiento de cualquier secuencia justificada debe ser una pregunta inicial, ya que no es posible tener un puntero a un movimiento anterior a él (por convención $\mu_1 = 0$) y no puede existir ninguna otra ocurrencia de un movimiento inicial en el resto de la secuencia.

(w2) *Justificación explícita.* Existen dos casos:

- Cualquier pregunta no inicial puede ser formulada si una instancia de su única pregunta que la justifique ha sido hecha y todavía no ha sido respondida. Más precisamente, para cualquier pregunta no inicial m_j de s, el movimiento indexado con μ_j (que es m_{μ_j}) debe cumplir que $m_{\mu_j} \vdash_A m_j$, y el segmento $[m_{\mu_j}, m_{\mu_j+1}, \ldots, m_j]$ de la secuencia s no contiene ninguna respuesta de m_{μ_j}.
- Cualquier respuesta m_k puede ser ofrecida si una instancia de la única pregunta que la justifica ha sido formulada y no ha sido respondida todavía. Más precisamente, cualquier respuesta m_k en s responde explícitamente la pregunta m_{μ_k}, i.e., $m_{\mu_k} \vdash_A m_k$, y el segmento $[m_{\mu_k}, m_{\mu_k+1}, \ldots, m_{k-1}]$ de la secuencia s no contiene ninguna respuesta explícita de m_{μ_k}

(w3) *Ultimo Preguntado - Primero Contestado* o *Sin preguntas pendientes.* La secuencia s satisface esta condición si para cualquier movimiento de respuesta m_i en s, el movimiento m_{μ_i} que satisface $m_{\mu_i} \vdash_A m_i$ es la última pregunta no respondida en la subsecuencia $[m_1, \ldots, m_{i-1}]$ de s.

Notaremos J_A al conjunto de todas las secuencias justificadas de A. ■

Los índices pueden ser pensados como una forma de representar punteros de justificación: para cada ocurrencia de un movimiento m_i de s, existe un puntero a una ocurrencia de un movimiento anterior m_{μ_j} en $s_{<m_i}$, tal que $m_{\mu_j} \vdash_A m_i$.

De la definición anterior y (e1) se puede concluir que las secuencias justificadas siempre empiezan con una pregunta inicial del oponente.

Es importante remarcar que la justificación se refiere a una instancia particular de movimientos: dada una ocurrencia de un movimiento m que ocurra en una secuencia justificada s, podrían existir varias ocurrencias de movimientos en $s_{<m}$ que *habiliten* a m, pero un único movimiento de justificación es dado a través de la estructura del puntero. Esta observación es importante para la siguiente definición.

Definición 2.6 (Vista del Proponente) [HO94, Har99]
La *vista del proponente* o P-vista de una secuencia no vacía $s \in J_A$, que notaremos $\ulcorner s \urcorner$, se define inductivamente como sigue:

$$
\begin{array}{rl}
\ulcorner \epsilon \urcorner = \epsilon & \\
\ulcorner s[m] \urcorner = [m] & \text{si } m \text{ es inicial;} \\
\ulcorner s[m] \urcorner = \ulcorner s \urcorner [m] & \text{si } m \text{ es un movimiento } P \\
\ulcorner s[n]t[m] \urcorner = \ulcorner s \urcorner [n, m] & \text{si } m \text{ es un movimiento } O \text{ y } n \text{ justifica a } m;
\end{array}
$$

Sea $s[m] \in J_A$ donde m es un movimiento P. Decimos que $s[m]$ satisface *la visibilidad del Proponente* en m si y sólo si la justificación de m ocurre en $\ulcorner s \urcorner$. Si $s \in J_A$ luego s satisface la *visibilidad del Proponente* si y sólo si s satisface la visibilidad del Proponente en m para cada ocurrencia de un movimiento P m en s. ■

La intuición de la P-vista es "pasar por encima de" los movimientos P repetidamente y "perseguir hacia atrás" los punteros de los movimientos O hasta que se alcanza el movimiento inicial.

Si $s \in J_A$ satisface la visibilidad del Proponente luego su P-vista es una secuencia justificada. El recíproco no es verdad. Es posible para una P-vista de una secuencia justificada s ser justificada aún si s viola la visibilidad del proponente.

Ejemplo 2.3 Consideremos la siguiente secuencia de movimientos:

$$M_1^{OQ}\ M_2^{PQ}\ M_3^{PA}\ M_4^{OQ}\ M_5^{OQ}\ M_6^{PA}\ M_7^{OQ}\ M_8^{PQ}\ M_9^{OQ}\ M_{10}^{PA}$$

donde la secuencia de justificación es:

$M_1 \vdash M_2$
$M_1 \vdash M_4$
$M_1 \vdash M_5$
$M_1 \vdash M_7$
$M_2 \vdash M_3$
$M_4 \vdash M_8$
$M_5 \vdash M_6$
$M_8 \vdash M_9$
$M_9 \vdash M_{10}$

La P-vista es

$$M_1^{OQ}\ M_7^{OQ}\ M_8^{PQ}\ M_9^{OQ}\ M_{10}^{PA}$$

sin embargo, no se cumple la visibilidad del proponente, ya que la justificación de M_8^{PQ} no está en $\ulcorner M_1^{OQ}\ M_2^{PQ}\ M_3^{PA}\ M_4^{OQ}\ M_5^{OQ}\ M_6^{PA}\ M_7^{OQ} \urcorner$. ■

Definición 2.7 (Vista del Oponente) [HO94, Har99]
La *vista del oponente* u O-vista de una secuencia no vacía $s \in J_A$, que notaremos $\llcorner s \lrcorner$, se define inductivamente como sigue:

$$\begin{aligned} \llcorner \epsilon \lrcorner &= \epsilon \\ \llcorner s[m] \lrcorner &= \llcorner s \lrcorner [m] && \text{si } m \text{ es un movimiento } O \\ \llcorner s[n]t[m] \lrcorner &= \llcorner s \lrcorner [n, m] && \text{si } m \text{ es un movimiento } P \text{ y } n \text{ justifica a } m; \end{aligned}$$

Sea $s[m] \in J_A$ donde m es un movimiento O no inicial. Decimos que $s[m]$ satisface *la visibilidad del Oponente* en m si y sólo si la justificación de m ocurre en $\llcorner s \lrcorner$. Si $s \in J_A$ luego s satisface la *visibilidad del Oponente* si y sólo si s satisface la visibilidad del Oponente en m para cada ocurrencia de un movimiento P m en s. ■

Las definiciones anteriores permitirán restringir la clase de jugadas que se podrán hacer en el juego. Recordemos que bajo este enfoque se especifica el árbol del juego en términos del conjunto de caminos en el árbol. Estos caminos son las jugadas legales y se definen como sigue.

Definición 2.8 (Secuencia Legal)[HO94, Har99]
Una secuencia justificada s es *legal* o es una *posición legal*, si también satisface las siguentes condiciones:

Los Jugadores se Alternan: si $s = s_1[m, n]s_2$ luego $\lambda^{OP}(m) \neq \lambda^{OP}(n)$.

Visibilidad: Se cumple la visibilidad del Oponente y del Proponente.

Denotamos al conjunto de todas las secuencias legales de A como L_A. ■

Una secuencia legal de movimientos, o jugada legal, en un juego representará la *historia de tal jugada*.

Ejemplo 2.4 Existe una sola jugada legal en la arena 1, la secuencia ϵ. En una arena plana sobre un conjunto X, las jugadas legales (de longitud par) tienen la forma $[q, x_i]$ para cada $x_i \in X$. ■

Definición 2.9 (Movimiento Justificado en forma hereditaria)[HO94, Har99]
Sea s una secuencia legal de una arena A y sea m un movimiento en s. m está *justificado en forma hereditaria* por una ocurrencia de un movimiento inicial n en s si siguiendo la cadena de punteros de justificación que va hacia atrás desde m termina en n, i.e., m está justificado por algún movimiento m_k, el cual está justificado a su vez por m_{k-1} y así continuando hasta el movimiento inicial n. Denotaremos con $s \upharpoonright n$ la subsecuencia de s que contiene todos los movimientos justificados en forma hereditaria por n. Definimos en forma similar $s \upharpoonright I$ para el conjunto I de ocurrencias de movimientos iniciales en s como la subsecuencia de s que consiste de todos los movimientos justificados en forma hereditaria por un movimiento de I. ■

Una vez definido el ambiente y las reglas para jugar, estamos en condiciones de definir el juego.

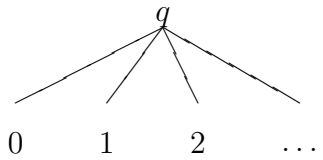
Definición 2.10 (Juego) [HO94, Har99]
Sea $A = \langle M_A, \lambda_A, \vdash_A \rangle$ una arena. Un *juego* G jugado en la arena A es una estructura $\langle M_A, \lambda_A, \vdash_A, P_G \rangle$ donde P_G es un subconjunto cerrado bajo prefijos, no vacío, de L_A, llamado posiciones válidas y que satisface

$$\text{si } s \in P_G \text{ e } I \text{ es el conjunto de movimientos iniciales de } s \text{ entonces } s \upharpoonright I \ \in \ P_G.$$

■

Los juegos modelados con la definición anterior tienen jugadas de longitud par, i.e., todas las preguntas son respondidas. Así, el juego lo empieza el oponente y lo termina el proponente. Cada movimiento en una jugada o secuencia está justificado en forma hereditaria por un estado inicial del conjunto I.

Ejemplo 2.5 La arena plana $\mathbb{N}$ modela a los valores de los números naturales. Cada número es modelado como una interacción simple: el ambiente comienza el cómputo con un movimiento inicial q (la pregunta: ¿Qué es un número?) y P puede responder jugando un número natural (una respuesta a la pregunta). Luego el juego $\mathbb{N}$ de los número naturales sería:



Este juego, que denominaremos $G_{\mathbb{N}}$ y que es jugado en el ambiente de la arena plana $\mathbb{N}$, define su árbol a través de $P_{G_{\mathbb{N}}} = \{\epsilon, [q], [q,0], [q,1], [q,2], \ldots\}$ ■

Ejemplo 2.6 Supongamos que queremos modelar la función SUCESOR. Modelar funciones requiere de una interacción un poco más complicada. En la semántica basada en juegos, el ambiente de una función consume la salida y provee la entrada, mientras que la función misma consume la entrada y produce la salida. El juego $\mathbb{N} \to \mathbb{N}$ es, por lo tanto, construido a partir de dos copias de $\mathbb{N}$, una para la entrada y una para la salida. La estrategia para modelar la función SUCESOR es:

> "Cuando O pregunta por una salida, P pregunta por una entrada; cuando O provee la entrada n, P dará la salida $n + 1$."

Una jugada en este juego tiene la siguiente forma:

$$\begin{array}{ccccc} \mathbb{N} & \Rightarrow & \mathbb{N} & & \\ & & q & O & \\ q & & & P & \\ 3 & & & O & \\ & & 4 & P & \end{array}$$

Este juego se define a partir de otro combinándolo de la forma correcta. En la siguiente sección se analiza la construcción de nuevas arenas a partir de otras. ■

2.4.3. Construcción sobre arenas

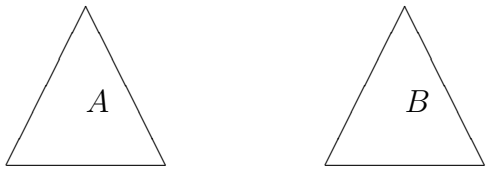
Dadas dos arenas A y B, podemos combinarlos para construir nuevas arenas de dos modos diferentes. Utilizaremos el símbolo + para representar la unión disjunta y notaremos con $s \upharpoonright A$ a todas las subsecuencias de s consistentes de movimientos de $\mathcal{M}_A$.

Producto: Denotaremos a la nueva arena $A \times B$.

- $\mathcal{M}_{A \times B} = \mathcal{M}_A + \mathcal{M}_B$
- $\lambda_{A \times B}(m) = \begin{cases} \lambda_A(m) & \text{si } m \in \mathcal{M}_A, \\ \lambda_B(m) & \text{si } m \in \mathcal{M}_B. \end{cases}$

  Notaremos a la definición de $\lambda_{A \times B}$ con $[\lambda_A,\ \lambda_B]$.
- $n \vdash_{A \times B} m$ si y sólo si $n \vdash_A m$ o $n \vdash_B m$

Intuitivamente, $A \times B$ es una arena consistente de A y de B ubicadas lado a lado, como se muestra en la figura:



No existe interacción entre las dos arenas: los movimientos iniciales de $A \times B$ son aquellos de A y aquellos de B y la habilitación es directamente heredada de igual modo. La arena vacía 1 es la unidad para este constructor.

El juego que se construye a partir de los juegos G_A y G_B, incluye la arena $A \times B$ y la siguiente definición de $P_{G_{A \times B}}$:

$$P_{G_{A \times B}} = \{s \in L_{A \times B} \mid s \upharpoonright A \in P_A \text{ y } s \upharpoonright B \in P_B\}$$

El juego 1 es $\langle \varnothing, \varnothing, \varnothing, \{\epsilon\} \rangle$.

Ya que no hay interacción entre las arenas, los juegos A y B son jugados en paralelo

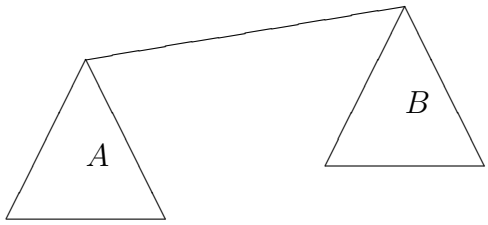
Espacio de Función o Implicación Lineal: Un modo alternativo de combinar dos arenas hace que una constituya un subordinado de la otra. Los movimientos iniciales de la arena combinada son aquellos del componente privilegiado; la nueva habilitación es derivada del modo obvio excepto que los movimientos iniciales de la arena subordinado son ahora habilitados por los movimientos iniciales de la arena privilegiada.

Notaremos con $\overline{\lambda_A}$ la etiqueta alternativa que cambia de Oponente a Proponente y viceversa, i.e., $\lambda_A^{OP} = O$ si y sólo si $\overline{\lambda_A^{OP}} = P$, pero que deja el estado pregunta/respuesta sin cambios.

- $\mathcal{M}_{A \Rightarrow B} = \mathcal{M}_A + \mathcal{M}_B$
- $\lambda_{A \Rightarrow\ B}(m) = \begin{cases} \overline{\lambda_A}(m) & \text{si } m \in \mathcal{M}_A, \\ \lambda_B(m) & \text{si } m \in \mathcal{M}_B. \end{cases}$

  Notaremos a la definición de $\lambda_{A \Rightarrow\ B}$ con $[\overline{\lambda_A},\ \lambda_B]$.
- $* \vdash_{A \Rightarrow B} m$ si y sólo si $* \vdash_B m$
- $n \vdash_{A \Rightarrow B} m$ si y sólo si $n \vdash_A m$ o $n \vdash_B m$ o $(* \vdash_B n$ y $* \vdash_A m)$, para $n \neq *$

Esto puede ser visualizado en la siguiente figura donde la línea que conecta a A y B pretende representar que los movimientos iniciales de B habilitan a los movimientos iniciales de A.



Nótese que, como los movimientos iniciales de A son habilitados por los movimientos iniciales de B en $A \Rightarrow B$, un movimiento O de A debe ser considerado como un P movimientos de $A \Rightarrow B$. En otras palabras, los movimientos del oponente de $A \Rightarrow B$ será el conjunto $P_A + O_B$. Esta es la razón de la diferencia entre la función de etiquetamiento de $A \times B$ y $A \Rightarrow B$.

La extensión del árbol cuyo ambiente es la arena $A \Rightarrow B$ contiene la siguiente definición de $P_{G_{A \Rightarrow B}}$:

$$P_{G_{A \Rightarrow B}} = \{s \in L_{A \Rightarrow B} \mid s \upharpoonright A \in P_A \text{ y } s \upharpoonright B \in P_B\}$$

Ejemplo 2.7 Consideremos nuevamente el ejemplo 2.6 en el que se calcula el sucesor de un número entero positivo. Identificaremos con $\mathbb{NA}$ al juego G_A y con $\mathbb{NB}$ al juego G_B. La nueva arena sobre la que se jugará el juego sucesor se define como sigue:

- $\mathcal{M}_{\mathbb{NA}\Rightarrow\mathbb{NB}} = \mathcal{M}_{\mathbb{NA}} + \mathcal{M}_{\mathbb{NB}}$
- $\lambda_{\mathbb{NA}\Rightarrow\ \mathbb{NB}}(m) = [\overline{\lambda_{\mathbb{NA}}}, \lambda_{\mathbb{NB}}]$
- El movimiento inicial es $q \in \mathcal{M}_{\mathbb{NB}}$. Note que $q \in \mathcal{M}_{\mathbb{NA}}$ es ahora jugado por el Proponente.
- $n \vdash_{\mathbb{NA}\Rightarrow\mathbb{NB}} m$ si y sólo si $n \vdash_{\mathbb{NA}} m$ o $n \vdash_{\mathbb{NB}} m$ o $(* \vdash_{\mathbb{NB}} n$ y $* \vdash_{\mathbb{NA}} m)$, para $n \neq *$

■

En la siguiente sección definiremos el comportamiento que debería mostrar un jugador de acuerdo al movimiento realizado en el juego por el otro participante.

2.4.4. Estrategias

Una *estrategia* es como un "libro de reglas" que indica qué movimientos pueden ser hechos por el Proponente, i.e., determina cómo el proponente debe responder en una posición donde se espera que haga un movimiento. Abstractamente, una estrategia para el proponente es una función parcial que mapea cierta clase de posiciones legales, en las que el turno es para P, en movimientos P. Una estrategia se representa como un subárbol del árbol del juego asociado a la arena.

Definición 2.11 **(Estrategia)** [HO94, Har99]
Una *estrategia* sobre una arena A, que notaremos σ_A, es un conjunto no vacío cerrado bajo prefijos de jugadas legales de longitud par que satisface:

- *Determinismo:* Para cualquier $s \in \sigma_A$ en la que el turno es del Proponente, si $s[a]$ y $s[b]$ están en σ_A entonces $a = b$.
- *Completitud Contingente:* Para cualquier $s \in \sigma_A$ en la que el turno es del Oponente y para cada movimiento del Oponente a, si $s[a]$ es una posición legal entonces está en σ_A.

■

Ejemplo 2.8 Consideremos a la estrategia sobre $\mathbb{B} \Rightarrow \mathbb{B}$. El conjunto estará formado solamente por dos jugadas legales:$[q, q, tt, ff]$ y $[q, q, ff, tt]$. ■

Las estrategias pueden ser compuestas para modelizar la interacción entre los diferentes comportamientos definidos para un proceso o agente. Dadas dos estrategias $\sigma_{A\Rightarrow B}$ y $\tau_{A\Rightarrow C}$

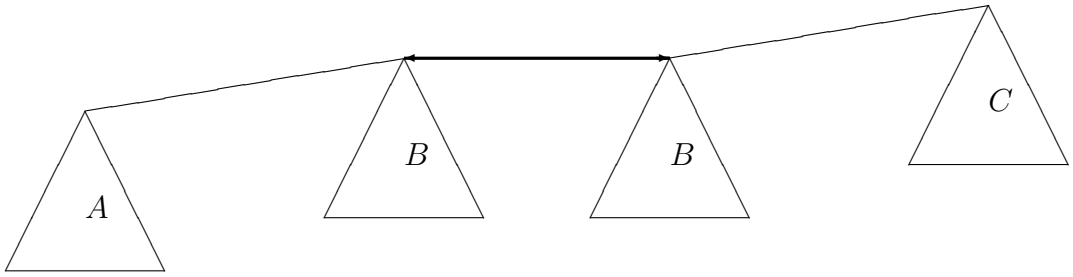


Figura 2.4: Composición de estrategias visto en forma gráfica.

querremos componerlas para formar la estrategia $\sigma; \tau_{B \Rightarrow C}$. Dicha composición de arenas puede ser visto gráficamente como se muestra en la figura 2.4.

Consideramos las dos ocurrencias de la arena B como si estuvieran conectadas por un buffer sincrónico. De este modo, cualquier movimiento del Proponente en B es inmediatamente copiado para volverse un movimiento Oponente en la otra ocurrencia. Esto lleva a una cierta cantidad de redundancia en la representación (varias repeticiones en B) que puede ser técnicamente *burlada* a través de las siguientes definiciones.

Definición 2.12 (Interacción) [HO94, Har99]
Sea u una secuencia de movimientos de las arenas A, B y C, junto con los punteros de justificación desde todos los movimientos excepto aquellos movimientos iniciales de C. Definimos $u \upharpoonright B, C$ como la subsecuencia de u que consiste de todos los movimientos desde B y C junto con sus punteros asociados, pero donde borramos cualquier puntero a un movimiento en A. Similarmente, se define $u \upharpoonright A, B$ borrando cualquier puntero a movimientos en C. Decimos que u es una *interacción* de A, B y C si y sólo si $u \upharpoonright A, B \in J_{A \Rightarrow B}$ y $u \upharpoonright B, C \in J_{B \Rightarrow C}$. ■

Si la secuencia u es una interacción entre las arenas A, B y C, luego *los punteros de justificación* desde cualquier movimiento de C debe ser necesariamente hacia otro movimiento de C. Sin embargo, si el movimiento es de A entonces su justificación puede ser un movimiento en A o uno inicial de B. En este último caso, se cumplirá que el movimiento inicial de B apunta a uno inicial de C. Consecuentemente, definimos $u \upharpoonright A, C$ como la subsecuencia de u que consiste de todos los movimientos de A y C, donde si tenemos un puntero que va de un movimiento de A a uno de B y desde el de B a uno de C, estos punteros son unidos para formar un nuevo puntero que vaya del movimiento de A al de C. La subsecuencia de u hereda todos los otro punteros directamente.

Definición 2.13 (Interacción Legal) [HO94, Har99]
Una *interacción legal* de arenas A, B y C es una interacción de A, B y C tal que $u \upharpoonright A, B \in L_{A \Rightarrow B}$, $u \upharpoonright B, C \in L_{B \Rightarrow C}$ y $u \upharpoonright A, C \in L_{A \Rightarrow C}$. Notaremos $int(A, B, C)$ al conjunto de todas las interacciones legales A, B y C. ■

Definición 2.14 (Componente) [HO94, Har99]
Si $u \in int(A, B, C)$ y $a \in u$ luego decimos que a es un *componente* $A \Rightarrow B$ si y sólo si es un movimiento de A o es un movimiento O de B; dualmente, a es un *componente* de $B \Rightarrow C$ si y sólo si es un movimiento C o es un movimiento P de B. ■

Analicemos las definiciones anteriores. Una secuencia no vacía $u \in int(A, B, C)$ debe comenzar con un movimiento inicial de C. Después de ésto, es el turno del proponente tanto en $A \Rightarrow C$ como en $B \Rightarrow C$, pero en $A \Rightarrow B$ no ha ocurrido nada, por lo tanto, sigue siendo el turno del oponente. Esto restringe las opciones del Proponente: puede mover en B o en C pero no en A, pues, de otro modo, si el proponente jugara en A violaría la legalidad de $u \upharpoonright A, B$. Si el Proponente juega otra vez en C, volvemos al punto de partida. Si por el contrario, mueve en B, ésto habilita el juego tanto en $A \Rightarrow B$ como en $B \Rightarrow C$. El juego lo continúa el Proponente en $A \Rightarrow B$ y el Oponente en $B \Rightarrow C$.

Ejemplo 2.9 Consideremos a la arena $\mathbb{B}$ y a las arenas construidas a partir de ésta: $\mathbb{B} \Rightarrow \mathbb{B}$. Sean las estrategias σ y τ aquellas que marcan el comportamiento de la negación clásica: "frente al valor tt devuelve ff y frente al valor ff devuelve el valor tt". La composición de σ y τ tendrá como comportamiento final devolver el mismo valor de entrada (la estrategia σ cambia el valor y la estrategia τ lo vuelve a cambiar). A continuación se muestran dos secuencias de jugadas en cada una de las arenas construidas con el operador flecha.

A	$\Rightarrow$	B		B	$\Rightarrow$	C	
$\mathbb{B}$	$\Rightarrow$	$\mathbb{B}$		$\mathbb{B}$	$\Rightarrow$	$\mathbb{B}$	
						q	O
		q	O	q			P
q			P				
tt			O				
		ff	P	ff			O
						tt	P

Es interesante observar que cuando juega B en una de las arenas $A \Rightarrow B$ lo hace participando como oponente mientras que en la otra el turno corresponde al proponente. Analicemos la secuencia completa de movimientos:

$\overbrace{B \Rightarrow C}$		$\overbrace{A \Rightarrow B}$				$\overbrace{B \Rightarrow C}$	
O	P	O	P	O	P	O	P
C	B	B	A	A	B	B	C
q	q	q	q	tt	ff	ff	tt

La secuencia que pertenece a la composición de las estrategias, i.e., a $\sigma; \tau$, es la siguiente:

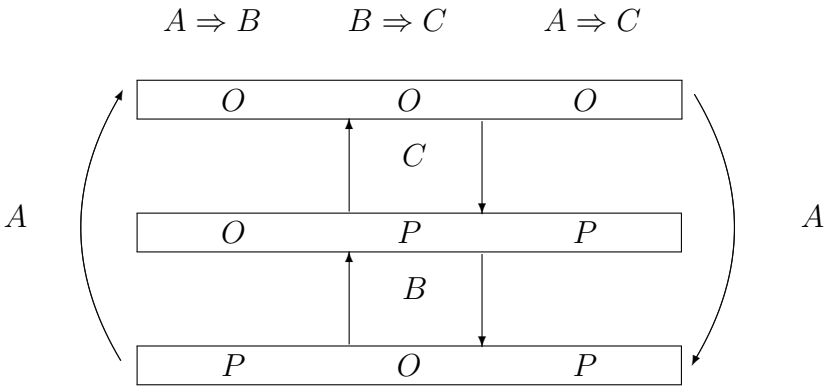


Figura 2.5: Diagrama de estados de las interacciones legales de la composición de estrategias $A \Rightarrow B$ y $B \Rightarrow C$

O	P	P	O	P	O	O	P
C	B	B	A	A	B	B	C
q	q	q	q	tt	ff	ff	tt

En la arena $A \Rightarrow B$, cada vez que juega B no cambia el participante. Esto es consecuencia de que el movimiento efectuado por B debe ser copiado a la otra arena para dar las consecuencias de esta jugada (ver figura 2.4). ■

Resumimos el análisis realizado sobre composición de estrategias en el diagrama de estados para las interacciones legales de la Figura 2.5, donde se muestran los turnos respectivos de $A \Rightarrow B$, $B \Rightarrow C$ y $A \Rightarrow C$ respectivamente.

En este punto estamos en condiciones de presentar la definición formal de composición de dos estrategias σ y τ.

Definición 2.15 (Composición Paralela de Estrategias) [HO94, Har99]
Sean $\sigma : A \Rightarrow B$ y $\tau : B \Rightarrow C$ dos estrategias. La *composición paralela* está definida como

$$\sigma||\tau \;=\; \{u \in int(A,B,C) |\; u \upharpoonright A,B \in \sigma \text{ y } \; u \upharpoonright B,C \in \tau\}.$$

■

La idea es que σ y τ sincronicen en su ambiente común, i.e., en la arena B. Si τ juega un movimiento m del Proponente (de $B \Rightarrow C$) en B, el efecto es que *el control es pasado* a σ quien ahora responde a m en A o en B.

Definición 2.16 (Composición de Estrategias) [HO94, Har99]
Sean $\sigma : A \Rightarrow B$ y $\tau : B \Rightarrow C$ dos estrategias. La *composición* de σ y τ, que denotaremos $\sigma;\tau$

se define como sigue:

$$\sigma;\tau \;=\; \{u \upharpoonright A, C | u \in \sigma || \tau\}$$

■

Se puede probar [Har99] que la composición de dos estrategias es una estrategia bien definida para $A \Rightarrow C$, i.e., que $\sigma;\tau$ es una estrategia en $A \Rightarrow C$, aunque esto queda fuera del alcance de este trabajo.

En esta sección hemos presentado y ejemplificado el enfoque de juegos basados en arenas definido por Hyland y Ong. En el resto del capítulo se estudia el enfoque alternativo de juegos y se lo compara con los HO-juegos.

2.5. AJM-Juegos

En [Abr97] y [AM97], se introduce la semántica de juegos con el objeto de modelar el cómputo como un juego entre dos participantes. Uno de los jugadores en el juego representa al Sistema y se denomina Proponente (P); el otro representa al Ambiente y se denomina Oponente (O). Una "corrida" o "cómputo" simple involucra la interacción entre el proponente y el oponente y es representado por una secuencia de movidas hechas en forma alternada por P y O.

2.5.1. Juegos

Comenzaremos dando la definición del enfoque AJM y luego presentaremos la definición dada por Abramsky en [Abr97], comparándola con la primera.

Definición 2.17 (Juego: Enfoque AJM) [Har99]
Un *juego* G es una estructura $(O_G, P_G, \approx_G)$, donde

- O_G y P_G son conjuntos contables de *movidas* del juego; denotamos su unión disjunta con M_G. Si $m \in O_G$, decimos que es una *O-movida* de G; similarmente, si $m \in P_G$, decimos que es una *P-movida* de G. El conjunto O_G define al *Oponente* en el juego G y el conjunto P_G define al *Proponente* en G.

  Sea L_G el conjunto de todas las secuencias s, $s \in M_G^*$, que satisfacen: $s_i \in O_G$ si y sólo si i es impar. De este modo, si $s \in L_G$ luego comienza con una O-movida y después de esto, las movidas alternan estrictamente entre O_G y P_G. Diremos que s es una *jugada legal* de G.

- $\approx_G$ es una relación de equivalencia parcial (i.e., simétrica y transitiva) sobre L_G que satisface:

  (e1) si $s \approx_G t$ entonces $|s| = |t|$

  (e2) si $s[a] \approx_G t[a']$ entonces $s \approx_G t$

  (e3) si $s \approx_G t$ y $s[a] \approx_G s[a]$ entonces existe $a' \in M_G$ tal que $s[a] \approx_G t[a']$.

■

La equivalencia $\approx_G$ es parcial pues podrían existir elementos en L_G que no estén en relación $\approx_G$. Como consecuencia, $\approx_G$ no genera una partición del conjunto L_G. Si necesitaramos construir una partición del conjunto podríamos hacerlo considerando dos subconjuntos. Por un lado, tendríamos en cuenta a las clases generadas por $\approx_G$ y por el otro, consideraríamos el conjunto R de todos los elementos de L_G que no están en relación. Así

$$L_G = R \bigcup \{C_i\}_{i \in I}$$

Analicemos el propósito de la relación de equivalencia parcial $\approx_G$. El axioma (e1) exige que aquellas secuencias que estén en relación tengan la misma longitud. En particular, si $s \approx_G \epsilon$ entonces $s = \epsilon$.

El axioma (e2) exige que se respeten los prefijos de las secuencias, i.e., $\mathcal{P}_G$ es cerrado con respecto a los prefijos. Si dos secuencias están en relación $\approx_G$, todos sus prefijos lo están también. De este modo, los juego que se modelan cumplen que si una jugada es equivalente a otra entonces todas las subjugadas deben ser equivalentes.

El axioma (e3), que es conocido como *de extensión* afirma que si tenemos dos jugadas equivalentes y una de ellas puede ser extendida con un movimiento a una nueva jugada entonces la otra jugada también puede ser extendida y las dos nuevas jugadas siguen siendo equivalentes.

2.5.2. Jugadas Válidas

A partir de la definición anterior podemos indicar cuales jugadas son legales en nuestro juego.

Definición 2.18 (Jugadas Válidas)
El conjunto de *jugadas válidas* o simplemente *jugadas* de G es el conjunto de todas las secuencias $s \in L_G$ tal que $s \approx_G s$ y las denotaremos $\mathcal{P}_G$. ■

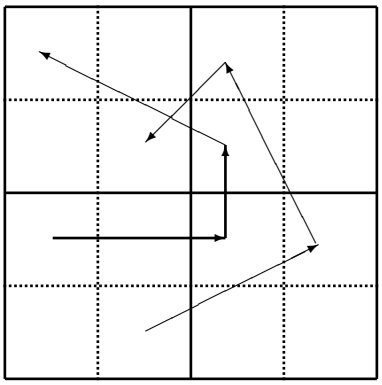


Figura 2.6: Diagrama que muestra una partición en clases de equivalencia parcial.

Si bien la relación no es necesariamente reflexiva, es interesante notar que aquellos elementos que están en relación $\approx_G$, por ser la relación simétrica y transitiva, cumplen la propiedad reflexiva. Por lo tanto, cualquier secuencia que esté en relación $\approx_G$ será una jugada válida.

Resumiendo, la relación de equivalencia parcial $\approx_G$ tiene dos propósitos. Por una parte, especifica las jugadas válidas como el conjunto L_G. Por otra parte, posibilita particionar las jugadas válidas en clases de equivalencia parcial. Esto nos permite ver al juego desde dos niveles: el nivel de las jugadas "concretas", i.e., sólo las jugadas válidas de G y el nivel de las jugadas "abstractas", i.e., las clases de equivalencia parcial de las jugadas válidas.

Ejemplo 2.10 [Har99] En el diagrama de la figura 2.6 se ilustran cuatro "estados", cada uno subdividido en cuatro "subestados". Se muestran dos secuencias de transiciones de estados como equivalentes si ellos comienzan en el mismo estado y pasan a través de los mismos estados en el mismo orden sin importar por cual subestado particular pasan. Así, tenemos dos secuencias diferentes que pueden ser vistas como equivalentes. ■

Ejemplo 2.11 Reformulemos las arenas preferidas a través de los juegos según el enfoque AJM. El juego más simple es el *juego vacío*, denotado por **1**, y está definido como $(\varnothing, \varnothing, \{(\epsilon, \epsilon)\})$. El siguiente juego simple es aquel con un movimiento por parte del Oponente,$(\{q\}, \varnothing, \{(\epsilon, \epsilon), (q, q)\})$, y será denotado como $\perp$.

Dado un conjunto contable X, el *juego plano sobre* X es definido como

$$(\{q\}, \{x | x \in X\}, \{(\epsilon, \epsilon), (q, q)\} \cup \{(qx, qx) | x \in X\})$$

Es interesante remarcar que con la relación de equivalencia especificada, estos juegos son "afines". Al menos cuatro juegos planos serán identificados, aquellos juegos generados por: un

conjunto vacío, un conjunto con sólo un elemento, un conjunto con dos elementos y un conjunto infinito contable. Estos juegos serán denotados como $\perp$, $\mathbb{C}$, $\mathbb{B}$ y $\mathbb{N}$ respectivamente, fijando los conjuntos que los generan como $\varnothing$, $\{a\}$,$\{tt, ff\}$ y el conjunto $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ de enteros no negativos. ■

Ejemplo 2.12 Consideremos un ejemplo un poco más interesante. Supongamos el siguiente juego $G = (\{a_1, a_2, c_1, c_2\}, \{b_1, b_2, b_3\}, \approx_G)$. La relación de equivalencia está generada a partir de $[a_1] \approx_G [a_3]$, $[a_1, b_1] \approx_G [a_3, b_2]$, $[a_2, b_1] \approx_G [a_2, b_2]$ y $[a_2, b_1, c_1] \approx_G [a_2, b_2, c_2]$. Ya que la relación es simétrica y transitiva y debe cumplir con (e1)-(e3) las clases que se pueden formar son:

C1 = $\{\epsilon\}$

C2 = $\{[a_1], [a_3]\}$

C3 = $\{[a_2]\}$

C4 = $\{[a_1, b_1], [a_3, b_2]\}$

C5 = $\{[a_2, b_1], [a_2, b_2]\}$

C6 = $\{[a_2, b_3]\}$

C7 = $\{[a_2, b_1, c_1], [a_2, b_2, c_2]\}$

De este modo, el juego tiene varias jugadas equivalentes como muestra en la figura 2.7. Nótese que el Oponente tiene dos posibles movimientos iniciales en las clases C2 o C3 y que el Proponente tiene dos posibles jugadas equivalentes después de que el Oponente jugó $[a_2]$ que se corresponden con la clase C5.

La secuencia legal $[c_1, b_1] \in L_G$ por no estar en relación con ninguna otra jugada legal no pertenece a ninguna clase y, por lo tanto, no es una jugada válida. ■

2.5.3. AJM-Juegos: una segunda aproximación

Habiendo presentado y analizado la definición del enfoque de los juegos AJM, presentamos a continuación una definición de juego que está contenida en la definición anterior.

Definición 2.19 (Juego) [Abr97]
Un *juego* G es una estructura (M_G, λ_G, P_G), donde

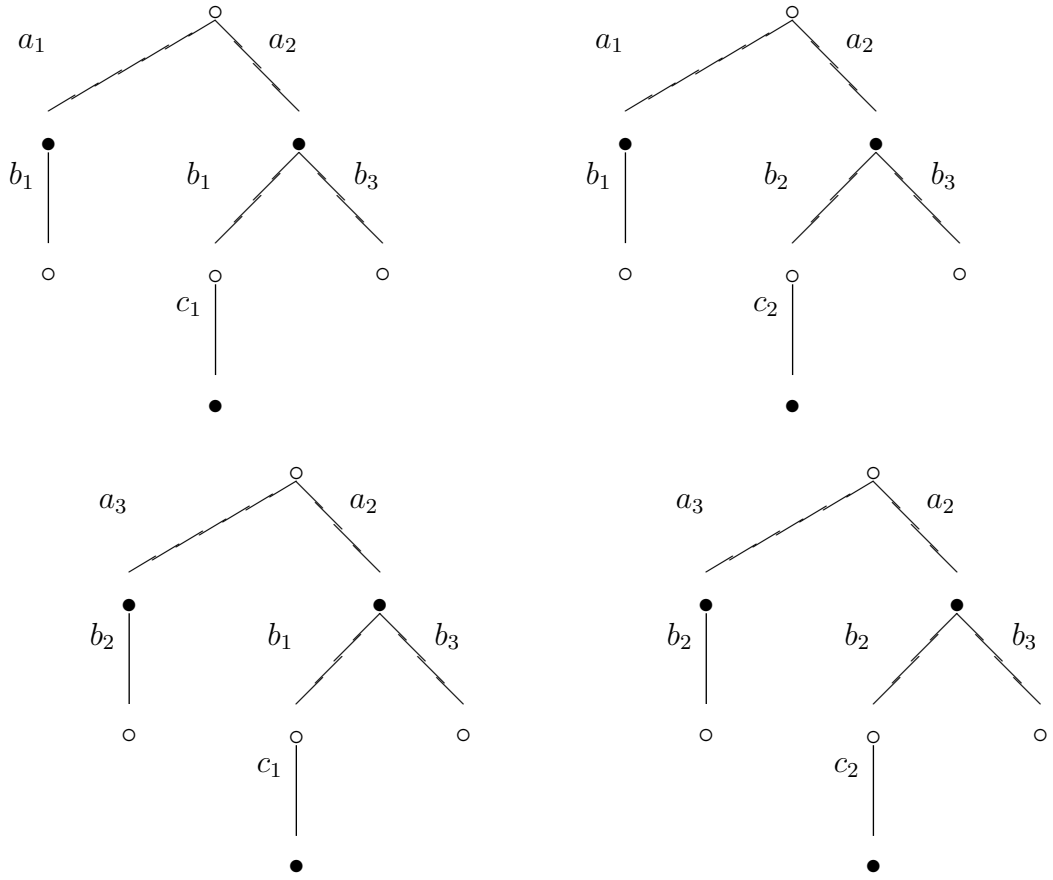


Figura 2.7: Árboles con jugadas equivalentes correspondientes al juego del ejemplo 2.12.

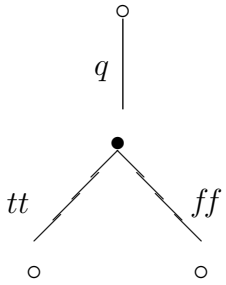


Figura 2.8: Árbol del juego del ejemplo 2.13.

- M_G es el conjunto de *movidas* del juego;
- $\lambda_G : \ M_G \to \{P, O\}$ es una función de etiquetamiento que designa a cada movimiento como del *Proponente* o del *Oponente*;
- $P_G \subseteq M_G^{alt}$, donde P_G es un subconjunto no vacío de M_G^{alt} cerrado con respecto a prefijos, y M_G^{alt} es el conjunto de todas las secuencias $s \in M_G^*$ tales que para todo $i : \ 1 \leq i \leq |s|$

$$\lambda_G(s_i) \ = \ \begin{cases} P & \text{si } i \text{ es par} \\ O & \text{si } i \text{ es impar} \end{cases}$$

■

Un juego especifica el conjunto de todas las posibles corridas y puede ser pensado como un árbol. P_G representa el árbol del juego.

Ejemplo 2.13 Consideremos nuevamente al juego $\mathbb{B}$, que puede ser visto como la representación del tipo de datos booleano. Utilizando la definición 2.20 dicho juego se formaliza como sigue:

$$\Big(\big\{q, tt, \mathit{ff}\big\}, \big\{\lambda(q) = O, \lambda(tt) = P, \lambda(\mathit{ff}) = P\big\}, \big\{\epsilon, [q], [q, tt], [q, \mathit{ff}]\big\}\Big)$$

El árbol que representa $P_{\mathbb{B}}$ se muestra en la figura 2.8. El movimiento de apertura q es un pedido de datos del Oponente, el que puede ser respondido con tt o ff por el Proponente. ■

Los juegos según la definición 2.20 sigue el enfoque AJM puro aunque no presenta el concepto de clases de equivalencia. El conjunto de movimientos del juego M_G y su clasificación en movimientos del Proponente o del Oponente (λ_G) tienen su equivalente en los conjuntos O_G y P_G del enfoque AJM puro.

Las jugadas legales L_G de la definición 2.17 se corresponde con M_G^{alt} en la segunda definición de juegos y las jugadas válidas de la primera con P_G en la segunda definición. Por último, podemos identificar jugadas de P_G en la definición 2.20 con alguna clase de equivalencia de las

jugadas en 2.17. En otras palabras, el enfoque AJM puro permite la representación de diversos juegos con jugadas equivalentes, mientras que la segunda definición sólo posibilita representar uno de tales juegos.

2.5.4. Construcción sobre juegos

Describiremos las construcciones sobre juegos que ya hemos definido para las arenas. Los nuevos juegos se basarán en la última de las definiciones dadas del enfoque AJM. Notaremos con $s \upharpoonright A$ o $s \upharpoonright \mathcal{M}_A$ a todas las subsecuencias de s consistentes de movimientos de $\mathcal{M}_A$. El símbolo $+$ y la función λ se definen del mismo modo que para el caso de las arenas.

Producto

Dados dos juegos A y B construimos el producto $A \otimes B$ del siguiente modo:

- $M_{A\otimes B} = M_A + M_B$
- $\lambda_{A\otimes B} = [\lambda_A, \lambda_B]$
- $P_{A\otimes B} = \{s \in M^{alt}_{A\otimes B} | s \upharpoonright M_A \in P_A \wedge s \upharpoonright M_B \in P_B\}$

Podemos pensar en $A \otimes B$ como un juego en el que se permite que las jugadas se realicen en ambos subjuegos A y B en una forma intercalada. En otras palabras, es una forma de composición paralela disjunta.

Proposición 2.1 Condición de Cambio*[Abr97]*
Si en una jugada $s \in P_{A\otimes B}$, sucesivos movimientos s_i y s_{i+1} están en diferentes subjuegos (i.e., uno está en A y el otro en B), luego $\lambda_{A\otimes B}(s_i) = P$ y $\lambda_{A\otimes B}(s_{i+1}) = O$.

El resultado anterior indica que sólo el Oponente puede hacer un cambio de un subjuego a otro; el Proponente debe responder siempre en el mismo subjuego en el que el Oponente ha jugado. Para mostrar la Condición de Cambio consideremos un par ordenado por cada secuencia s en el que el primer elemento del par indica el participante que tiene el turno en el subjuego A y el segundo elemento indica el participante que tiene el turno en el subjuego B. Inicialmente, el estado es $\epsilon = (O, O)$. La Figura 2.9 muestra el diagrama de transición de estados.

Nótese que O puede mover en cualquiera de los subjuegos en el estado inicial. Si O mueve en A, luego el estado cambia a (P, O). En este punto, P sólo puede mover el primer componente, volviendo al estado inicial. Note también que el estado (P, P) nunca será alcanzado.

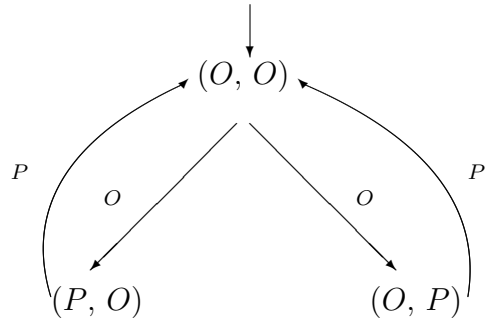


Figura 2.9: Diagrama de transición de estados para el Producto.

Implicación Lineal

Dados dos juegos A, B definimos al juego $A \multimap B$ como sigue:

- $M_{A \multimap B} = M_A + M_B$
- $\lambda_{A \multimap B} = [\overline{\lambda_A}, \lambda_B]$ donde $\overline{\lambda_A}(m) = \begin{cases} P & \text{si } \lambda_A(m) = O \\ O & \text{si } \lambda_A(m) = P \end{cases}$
- $P_{A \multimap B} = \{s \in M^{alt}_{A \multimap B} | s \upharpoonright M_A \in P_A \wedge s \upharpoonright M_B \in P_B\}$

La definición anterior es muy similar a la dada para construir el juego $A \otimes B$. La principal diferencia radica en la inversión de la función de etiquetamiento sobre los movimientos de A. Así los roles de Proponente y Oponente son intercambiados en el lado izquierdo de la flecha. La idea que se pretende reflejar es que el Sistema tiene su entrada y su salida, y esta última es la entrada del Ambiente, i.e., el Sistema produce y el Ambiente consume. Por lo tanto, los roles se invierten.

Otra diferencia entre la Implicación Lineal y el Producto reside en $M^{alt}_{A \multimap B}$ y $M^{alt}_{A \otimes B}$ y en consecuencia en $P_{A \multimap B}$ y en $P_{A \otimes B}$ respectivamente. El primer movimiento de $P_{A \multimap B}$ debe ser siempre de B, ya que este movimiento lo debe realizar el Oponente y todos los movimientos de apertura de A en $P_{A \multimap B}$ están etiquetados por $\overline{\lambda_A}$.

Proposición 2.2 Condición de Cambio*[Abr97]*
Si dos movimientos consecutivos lo realizan diferentes componentes, el primero lo realizó el Oponente y el segundo el Proponente. Luego, sólo el Proponente puede cambiar componentes.

Esto puede ser analizado a través del diagrama de estado-transición (adaptación del presentado en [Abr97]) que se presenta en la Figura 2.10. En la figura puede observarse que el

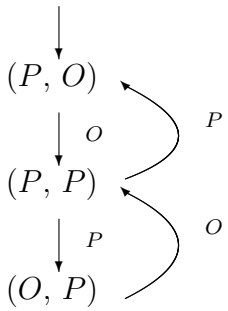


Figura 2.10: Diagrama de transición de estados para la Implicación Lineal.

movimiento inicial, sólo puede ser ejecutado dentro del juego B, ya que inicialmente juega el Oponente. Recordemos que la función de etiquetamiento invirtió el turno para el juego A y por lo tanto, todos los movimientos que eran iniciales en el juego A, ya no son iniciales ya que el turno le corresponde al Proponente. Así comenzamos el juego en B.

2.5.5. Estrategias

Los juegos clasifican comportamientos, de este modo los Programas serán modelados a través de estrategias, i.e., reglas que especifican cómo el Sistema debería jugar realmente.

Definición 2.20 (Estrategia) [AM97]
Una estrategia σ sobre un juego G [Abr97] es un subconjunto no vacío de posiciones de longitud par de P_G que satisface

- $\epsilon \in \sigma$ y si $s[a,b] \in \sigma$ entonces $s \in \sigma$. En otras palabras, debe ser cerrado con respecto al prefijo de las secuencias, $\sigma \subseteq P_G^{par}$.
- $s[a,b] \in \sigma$ y $s[a,c] \in \sigma$ entonces $b = c$.

■

Podemos considerar a una secuencia $s[a,b] \in \sigma$ del siguiente modo: *dado un estímulo a en un contexto s, responde con b*. Note que para cada estímulo una estrategia define una única respuesta, por lo tanto, una estrategia es siempre determinística.

Ejemplo 2.14 Consideremos nuevamente el ejemplo 2.13. Las estrategias sobre $\mathbb{B}$ son las siguientes:

$$\{\epsilon\}, \mathrm{Pref}([*, tt]), \mathrm{Pref}([*, ff])$$

La primera es la estrategia indefinida, $\perp$; la segunda y la tercera corresponden a los valores booleanos tt y ff. ■

Una propiedad que podemos pedir a las estrategias es *Libre de su Historia*[4]. La idea intuitiva de una estrategia libre de su historia es que la respuesta del Proponente a un movimiento dado del Oponente depende sólo de ese movimiento; es completamente independiente del contexto, i.e., de la historia precedente. Tales estrategias son algunas veces conocidas como *estrategias sin memoria*.

Definición 2.21 (Estrategia Libre de Historia) [Har99]
Una estrategia σ sobre un juego G es *libre de su historia* si y sólo si

$$(s[a,b] \in \sigma \text{ y } t \in \sigma \text{ y } t[a] \in P_G) \text{ entonces } t[ab] \in \sigma.$$

■

Una segunda restricción que podremos hacer sobre las estrategias es exigirle que sea *inyectiva*. En una estrategia inyectiva, una ocurrencia de un movimiento de P determina en forma precisa su *contexto local*, i.e., el movimiento O precedente.

Definición 2.22 (Estrategia Inyectiva) [Har99]
Una estrategia σ sobre un juego G es *inyectiva* si y sólo si

$$(s[a,b] \in \sigma \text{ y } t[a',b] \in \sigma) \text{ entonces } a = a'.$$

■

Analicemos la estrategia identidad que es determinística, libre de su historia e inyectiva.

Ejemplo 2.15 Estrategia Copy-Cat[5] [Abr97]
La idea de esta estrategia es vencer en ajedrez a Kasparov o a Short. Para llevar a cabo esto, jugamos dos juegos uno contra Kasparov, con las fichas negras, y uno contra Short, con fichas blancas. La situación se muestra en la Figura 2.11.

Comenzamos el juego contra Kasparov. Él realiza su jugada de apertura y nosotros jugamos su movida en nuestro juego contra Short. Luego Short responde, y nosotros jugamos su movimiento como respuesta a Kasparov. De este modo, *nosotros jugamos dos veces el mismo juego*, pero uno con las fichas blancas y uno con las negras. Así sin importar quien gane (si alguno gana), nosotros ganaremos un juego. Esto puede ser visto como un proceso en el que nosotros actuamos de buffer, ya que indirectamente están jugando Kasparov versus Short.

[4]En inglés *History-freedom*.
[5]Utilizaremos el término tal cual se lo presenta en la cita [Abr97], aunque el vocablo correcto es *copycat*, cuyo significado es *imitador*.

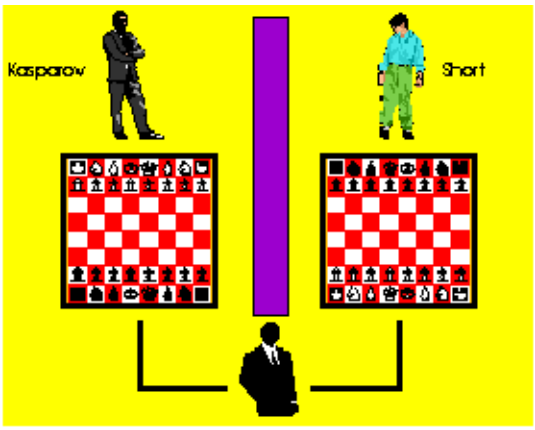


Figura 2.11: Estrategia Copy-Cat.

Esta estrategia, la podemos definir para implicación lineal $A \multimap A$, para cualquier juego A y su interpretación de axiomas lógicos es $A \vdash A$. Así el aspecto lógico de este proceso es la *conservación del flujo de información*, lo que asegura que nosotros ganemos un juego.

En general, una estrategia copy-cat sobre un juego A procede del siguiente modo[Abr97]:

	A	$\multimap$	A	
Tiempo				
1			a_1	O
2	a_1			P
3	a_2			O
4			a_2	P
$\vdots$		$\vdots$		$\vdots$

Así la identidad sobre un juego A se define como:

$$1_A = tt \text{ id}_A = \left\{ s \in P^{par}_{A_1 \multimap A_2} |\ s \upharpoonright A_1 = s \upharpoonright A_2 \right\}$$

donde A_1 y A_2 son las correspondientes dos ocurrencias de A en $A \multimap A$. ■

Composición de estrategias

La intuición informal de la composición no difiere de la vista para los HO-games: combinar juegos para producir comportamientos más complejos. Esto provee una base para el entendimiento de la composición de los sistemas de agentes interactuantes; entender el comportamiento de un sistema complejo en términos del comportamiento de las partes.

Definición 2.23 (Composición Paralela de Estrategias) [Har99]
Dadas dos estrategias σ y τ de los juegos $G \multimap H$ y $H \multimap J$ respectivamente, definimos la

composición en paralelo como

$$\sigma||\tau \;=\; \Big\{u \in (M_G + M_H + M_J)^* |\; u \upharpoonright G, H \in \sigma \text{ y } \; u \upharpoonright H, J \in \tau\Big\}$$

■

Recordemos que $u \upharpoonright G, H$ denota a la secuencia que se obtiene a partir de u borrando todos los elementos que no pertenecen a $(M_G + M_H)$ y que el signo + representa la unión disjunta de conjuntos.

Definición 2.24 (Composición de Estrategias) [Har99]
Dadas dos estrategias σ y τ de los juegos $G \multimap H$ y $H \multimap J$ respectivamente, definimos la *composición* como

$$\sigma;\tau \;=\; \Big\{u \in G, J |\; u \in \sigma||\tau\Big\}$$

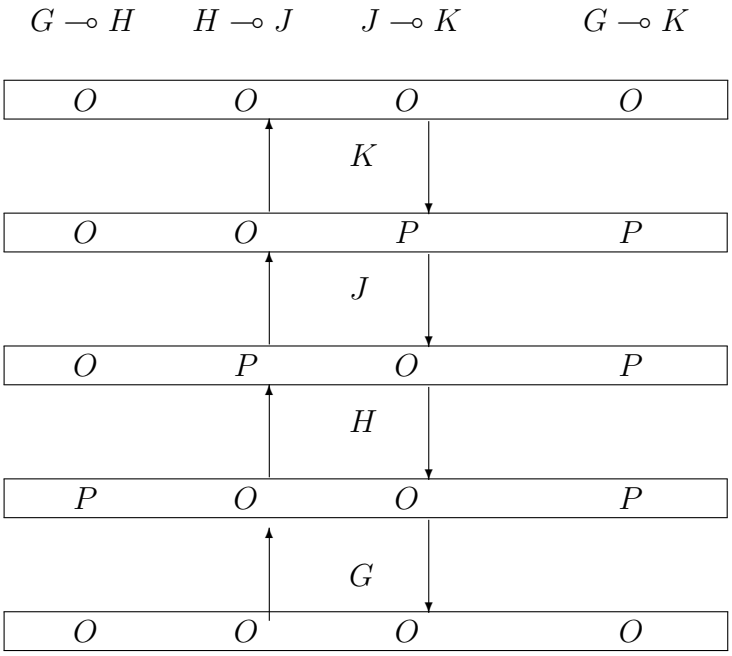
■

Dada la definición de composición deberemos exigir que esté bien definida, i.e., , que la composición sea una estrategia, y que además sea asociativa. Las siguientes proposiciones lo aseguran; sus demostraciones pueden encontrarse en [Har99].

Proposición 2.3 *[Har99] Si σ y τ son dos estrategias de $G \multimap H$ y $H \multimap J$ respectivamente, entonces $\sigma;\tau$ es una estrategia de $G \multimap J$, i.e., $\sigma;\tau$ está bien definida.*

Proposición 2.4 *[Har99] Si σ, τ y υ son estrategias de $G \multimap H$, $H \multimap J$ y $J \multimap K$ respectivamente, entonces $(\sigma;\tau);\upsilon \;=\; \sigma;(\tau;\upsilon)$.*

El siguiente diagrama muestra las interacciones en los cuatro juegos:



El juego $G \multimap K$ cambia de jugador sólo cuando el movimiento se realiza en G o en K. Los pasos intermedios están *escondidos* por la interacción.

Las siguientes proposiciones indican cómo se extienden las propiedades de las estrategias compuestas a la composición resultante.

Proposición 2.5 *[Har99] Si σ y τ son dos estrategias libres de su historia de $G \multimap H$ y $H \multimap J$ respectivamente, entonces $\sigma;\tau$ es una estrategia libre de su historia.*

Proposición 2.6 *[Har99] Si σ y τ son dos estrategias inyectivas de $G \multimap H$ y $H \multimap J$ respectivamente, entonces $\sigma;\tau$ es una estrategia inyectiva.*

2.5.6. Estrategias Ganadoras

Nosotros estamos interesados en cierta clase de estrategias: aquellas que Abramsky clasifica como *ganadoras*[Abr97]. Una estrategia es ganadora si cada posible movimiento del oponente tiene alguna respuesta. En otras palabras, el Sistema debe estar siempre preparado para responder a cualquier estímulo desde el Ambiente.

Definición 2.25 (Estrategia Total) [Abr97]
Una estrategia σ sobre un juego G es *total* si en cada etapa del juego tiene una respuesta para

cada movimiento posible del oponente, i.e.,

$$s \in \sigma,\ s[a] \in P_G \text{ entonces existe } b \text{ tal que } s[a,b] \in \sigma$$

■

Esta condición no es suficiente ya que no es cerrada bajo la composición, i.e., existen estrategias totales σ y τ, sobre los juegos $A \to B$ y $B \to C$ respectivamente, tales que $\sigma;\tau$ no lo es. La falla de esta restricción se da con jugadas infinitas sobre el juego B. Así si sólo consideramos jugadas finitas sobre nuestro árbol de juego, una estrategia ganadora es una estrategia total.

Aunque no está en el alcance de este trabajo y con el espíritu de complementar la idea, presentamos algunas definiciones para lograr que las estrategias ganadoras cumplan la condición, en el caso general. Para esto, formalizaremos una estructura de especificación adecuada.

Definición 2.26 **(P_G^∞)** [Abr97]
Dado un juego G, definimos P_G^∞, las jugadas infinitas sobre G como

$$P_G^\infty \ = \ \left\{ s \in M_G^\omega |\ \text{Pref}(s) \subseteq P_G \right\}$$

Recordemos que Pref(s) indica al conjunto de prefijos finitos de s. ■

De este modo, las jugadas infinitas se corresponden exactamente con las ramas infinitas del árbol de juego. Notaremos con PG al conjunto formado por las partes de P_G^∞, i.e.,

$$PG \ = \ \left\{ W |\ W \subseteq P_G^\infty \right\}$$

Tal conjunto puede ser interpretado como designado aquellas jugadas infinitas que son ganadoras para el Proponente.

Definición 2.27 **(Estrategia Ganadora)** [Abr97]
Una estrategia σ sobre un juego G es *ganadora* con respecto a W si

- σ es total.
- $\left\{ s \in P_G^\infty | \text{Pref}(s) \subseteq \sigma \right\} \subseteq W$

■

Así σ es ganadora si en cada etapa finita, cuando es el turno del Proponente, éste tiene una respuesta bien definida, y por otra parte, cada jugada infinita que sigue a σ es una ganada por el Proponente. Para los propósitos de esta tesis tomaremos la primer definición. Para aquellos lectores que deseen ampliar sus conocimientos sobre este tema, se recomienda la lectura de [Abr97].

2.6. Conclusiones

Los sistemas computacionales han evolucionado en los últimos treinta años de aquellos algorítmicos hacia los interactivos. Las Máquinas de Turing no son suficientemente poderosas como para reflejar estos cambios. Las Máquina de Interacción Secuenciales y de Múltiples Flujos, que son transformaciones de las Máquinas de Turing, permiten caracterizar el cómputo interactivo.

La interacción puede ser vista como un juego entre dos participantes: el *ambiente* y el *sistema*. Los enfoques de juegos, HO-juegos y AJM-juegos, modelan tal situación. En ambos casos, se plantea una interacción ordenada por turnos entre dos participantes, el Oponente y el Proponente. Las reglas del juego indicarán cuales son los movimientos legales. Sobre esos movimientos se definieron estrategias de juego, haciendo hincapié en aquellas que son ganadoras. Sobre los juegos definidos se construyeron nuevos juegos, mostrando la interacción entre ellos.

Este capítulo es la base matemática en la que se fundamenta el desarrollo de la semántica declarativa basada en juegos para la *plr*. El principal beneficio de utilizar la noción de juego reside en el sólido formalismos matemático, ya desarrollado [AJM94, AM97], que será adaptado y extendido en los próximos capítulos.

Capítulo 3

Programación en Lógica Rebatible

Contenido

Actualmente, la Programación en Lógica (de ahora en más P.L.) es reconocida como una herramienta valiosa para la representación del conocimiento y el razonamiento de sentido común. Con el objeto de incrementar los campos de aplicación de la P.L., se han propuesto varias extensiones a la clase de los programas definidos: los programas normales [Llo87] agregan la negación por falla, los programas básicos [GL90] permiten la negación fuerte y los programas disyuntivos [Min82] permiten disyunciones en la cabeza de las cláusulas de los programas.

El uso de la Programación en Lógica como sistema de representación del conocimiento y de razonamiento está basado en la idea de proveer a las máquinas con una especificación lógica del conocimiento, independiente de cualquier implementación, de tal modo que sea fácil de manipular y de razonar. Consecuentemente, un significado preciso debe ser asociado con cualquier programa lógico a fin de proveer una especificación declarativa. La performance de cualquier mecanismo computacional es evaluada comparando su comportamiento con la especificación provista por la semántica declarativa. Encontrar una semántica declarativa adecuada para los programas lógicos es una de las áreas de investigación de la P.L. más importantes y difíciles.

En la mayoría de los dominios del conocimiento humano, las creencias no son categóricas, por lo que las conclusiones que se obtienen a partir de ellas son potencialmente contradictorias. Precisamente, por esta razón, es que se requieren de técnicas de representación de conocimiento que resuelvan este problema. La Programación en Lógica Rebatible (de ahora en más P.L.R.) [Dun95, GS99, GSC98, GS04] es un sistema de representación de conocimiento y razonamiento, extensión de la P.L., que permite manipular tanto información certera como tentativa a través de dos clases diferentes de reglas: las reglas estrictas y las reglas rebatibles.

Las reglas estrictas son reglas en el sentido clásico de la programación en lógica: siempre que las premisas de una regla son satisfechas, tenemos permitido aplicarla y obtener la conclusión. Las reglas rebatibles son reglas que pueden ser derrotadas en presencia de evidencia contraria. Aunque la información tentativa representada puede ser eventualmente contradictoria, la P.L.R. provee un criterio para resolver los conflictos entre las reglas rebatibles con consecuentes contradictorios.

En este capítulo definiremos a la P.L.R. como una teoría formal. Analizaremos su lenguaje y el sistema deductivo. Si bien el capítulo incluye aquellos contenidos que necesitaremos para que el trabajo de esta tesis sea autocontenido, referiremos al lector interesado en obtener más información a [GS04].

Asimismo, introduciremos una clasificación de acuerdo a las características de los miembros de las reglas que está basada en la clasificación de la P.L. clásica. Finalmente, se presentan las conclusiones.

3.1. El Lenguaje

Comenzaremos presentando los ingredientes básicos de la P.L.R..

Definición 3.1 (Signatura)
Una *signatura* es una tupla $\Sigma = \langle \mathcal{V}, Func, Pred \rangle$, donde $\mathcal{V}$, $Func$ y $Pred$ son conjuntos de símbolos, disjuntos de a pares, tales que:

- $Pred$ es un conjunto enumerable no vacío de *símbolos predicativos* o *predicados*, donde cada predicado tiene asociado un número natural n. Diremos que n es su aridad. Un predicado de aridad cero es una *proposición*;
- $Func$ es un conjunto enumerable de *símbolos de función* o *funciones*, cada uno de los cuales tiene asociado un número natural n, su aridad. Una función de aridad cero se denomina *constante*;

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

Las variables serán denotadas por cualquier cadena de caracteres que comience con letras mayúsculas, mientras que los predicados y funciones serán notados por cualquier cadena que comience con letras minúsculas. El contexto nos ayudará a distinguir entre predicados y funciones siempre que encontremos una cadena de caracteres que comience con letras en minúscula.

Cuando sea requerido, notaremos a los símbolos predicativos y de funciones con un supraíndice que indicará la aridad de dichos predicados y funciones. Por ejemplo, si c es una constante, i.e., una función 0-aria, será notada c^0. ■

Definición 3.2 (Término)
Un *término* sobre una signatura $\Sigma = \langle \mathcal{V}, Func, Pred \rangle$ es definido inductivamente como el menor conjunto que cumpla las condiciones:

- toda variable $V \in \mathcal{V}$ es un término;
- toda constante $c \in Func$, de aridad 0, es un término;
- si $f \in Func$ es una función de aridad $n,\ n \geq 1$, y $t_1,\ \ldots, t_n$ son términos de Σ, entonces $f(t_1,\ \ldots, t_n)$ es un término;
- nada más es un término.

■

Los términos representan a los objetos del mundo que deseamos modelar y sobre los cuales definiremos relaciones.

Definición 3.3 (Alfabeto)
Un *alfabeto* $\mathcal{A}lp$ generado a partir de una signatura Σ es la unión del conjunto de los símbolos de la signatura Σ, el símbolo $\{\text{“}\sim\text{”}\}$ y el conjunto $\{\text{“(”,“)”, “,”}\}$ de *símbolos de puntuación*. ■

Nótese que el alfabeto introduce un nuevo símbolo: el conectivo de negación $\sim$. Este conectivo representa una noción de negación relevante en la representación del conocimiento: la *negación fuerte*. La negación fuerte de un átomo ocurrirá si el átomo es explícitamente falso, a diferencia de la *negación default*, que se nota con el símbolo “not”, y que es usada para representar información incompleta. Así, *not* L expresa que *no se conoce que* L *sea verdadera* y $\sim L$ expresar que L es falsa.

Definición 3.4 (Átomo - Literal Positivo - Literal)
Si $p \in \mathcal{A}lp$ es un predicado de aridad n y $t_1, \ldots, t_n$ son términos, entonces $p(t_1, \ldots, t_n)$ es una *fórmula atómica* o *átomo* o *literal positivo*. Un *literal* es un átomo A o un átomo negado $\sim A$. Diremos que un literal posiblemente precedido por el símbolo "*not*" se denomina un *literal extendido* [GL90]. ■

Los átomos representan las relaciones existentes entre los objetos del mundo a modelar y tendrán asignado un valor de verdad. El conectivo de negación fuerte pertenece al alfabeto y, por lo tanto, los átomos negados formarán parte del lenguaje. Sin embargo, ningún literal extendido formará parte del lenguaje, ya que símbolo *not* se considerará dentro del metalenguaje.

Definición 3.5 (Complemento de un Literal) [Fil01]
Sean X un literal extendido, L un literal y A un átomo. El *complemento de* L con respecto a la negación fuerte, que denotaremos $\overline{L}$, se define del siguiente modo:

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

El *complemento de* X con respecto a la negación default, que denotaremos $not(X)$, se define del siguiente modo:

$$not(X) = \begin{cases} not\ L & \text{si } X = L \\ L & \text{si } X = not\ L. \end{cases}$$

Si S es un conjunto de literales (respectivamente de literales extendidos) entonces se extiende la operación complemento $\overline{S}$ (respectivamente $not(S)$) como el conjunto $\{\overline{L} | L \in S\}$ (respectivamente $\{not(L) | L \in S\}$). ■

Definición 3.6 (Término, Átomo Literal y Literal Extendido Fijos)
Diremos que un término (respectivamente un átomo, un literal y un literal extendido) es *fijo*[1][Fil01] si no aparecen variables en su composición. ■

Asumiendo los axiomas UNA (*Unique Name Assumption*) y DCA (*Domain Clousure Assumption*), cada término fijo se corresponde con un único objeto en el mundo a modelar. Cada vez que una propiedad esté definida con variables, significará que la propiedad vale para cada objeto del mundo.

Con el fin de manejar literales fijos introduciremos las siguientes definiciones.

[1] *Ground* en inglés. En la mayoría de la literatura en idioma español, se traduce este vocablo como *básico*. Sin embargo, creemos que la traducción introducida por Fillottrani en [Fil01] es más adecuada.

Definición 3.7 (Sustitución)
Una *sustitución* σ es un conjunto finito de pares ordenados

$$\sigma = [(V_1/t_1), \ldots, (V_n/t_n)]$$

donde el primer elemento V_i de cada par es una variable y el segundo t_i es un término, tal que:

1. $V_i \neq V_j$ si $i \neq j$
2. V_i no ocurre en t_j para ningún i y j.

■

Definición 3.8 (Aplicación de la sustitución)
Sea $\sigma = [(V_1/t_1), \ldots, (V_n/t_n)]$ una sustitución. Una *aplicación de* σ a un término t, es otro término que notaremos $t\sigma$ y que se define como sigue:

- Si $t = c$, entonces $c\sigma = c$ para cualquier símbolo de constante c.
- Si $t = W$, siendo W una variable, tal que $W \notin \{V_1, \ldots, V_n\}$, entonces $W\sigma = W$.
- Si $t = V_i$, entonces $V_i\sigma = t_i$, para todo i, $1 \leq i \leq n$.
- Si $t = f(s_1, \ldots, s_m)$, entonces $f(s_1, \ldots, s_m)\sigma = f(s_1\sigma, \ldots, s_m\sigma)$ para cualquier símbolo de función f de aridad m.

■

Definición 3.9 (Composición de sustituciones)
Sean $\sigma = [(V_1/t_1), \ldots, (V_n/t_n)]$ una sustitución y (V, t) un par ordenado. La *composición* de σ y (V, t), que notaremos $\sigma[(V, t)]$ es la sustitución:

1. σ si $V \in \{V_1, \ldots, V_n\}$
2. $[(V_1/t_1[(V, t)]), \ldots, (V_n/t_n[(V, t)]), (V, t)]$ si $V \notin \{V_1, \ldots, V_n\}$

Sea $\tau = [(X_1, u_1), \ldots, (X_m, u_m)]$ otra sustitución. La *composición* de σ y τ, que notaremos $\sigma\tau$ es la sustitución $(((\sigma[(X_1, u_1)]) \ldots)[(X_m, u_m)])$. ■

Definición 3.10 (Instancia de un literal - Instancia Fija de un Literal)
Sean L un literal extendido y σ una sustitución. Una *instancia* de L por σ es otro literal extendido, que notaremos $L\sigma$ y que se obtiene del siguiente modo:

1. Si L es un átomo $p(t_1, \ldots, t_n)$, donde p es un símbolo de predicado de aridad n, entonces $p(t_1, \ldots, t_n)\sigma = p(t_1\sigma, \ldots, t_n\sigma)$.
2. Si $L = \sim A$, siendo A un átomo, entonces $(\sim A)\sigma = \sim (A\sigma)$.
3. Si $L = not\ X$, siendo X un literal, entonces $(not\ X)\sigma = not(X\sigma)$.

Si $L\sigma$ es un literal fijo, entonces diremos que $L\sigma$ es una *instancia fija* de L.
Sea S un conjunto de literales, entonces una instancia de S por σ es $S\sigma = \{s\sigma | s \in S\}$. Si todos los miembros de $S\sigma$ son fijos entonces diremos que $S\sigma$ es una instancia fija de S. ■

Definición 3.11 (Literal Fijo - Literal Extendido Fijo) [Lif96]
Sean $\mathcal{A}lp$ un alfabeto y L un literal asociado al alfabeto. Definimos al conjunto de todas las posibles instancias fijas de L, que notaremos **fijo**(L), como

$$\textbf{fijo}(L) \ = \ \{L' | L' \text{ es una instancia fija de } L\}$$

Análogamente se define para literales extendidos.
La definición se extiende para un conjunto de literales S como

$$\textbf{fijo}(S) = \{fijo(s) | s \in S\}$$

■

Definición 3.12 (Lenguaje)
Sea $\mathcal{A}lp$ un alfabeto bajo una signatura Σ. El *lenguaje de la P.L.R.* se define como el conjunto de todos los literales fijos bajo $\mathcal{A}lp$ y lo llamaremos Lit. ■

De forma análoga definiremos a $XLit$ con la sola intención de homogeneizar la notación, al conjunto de todos los literales extendidos fijos.

3.2. Representación del conocimiento con Programas Lógicos Rebatibles

La P.L.R. permite representar tanto el conocimiento tentativo como aquel que es certero. Para poder distinguirlos, se extiende la P.L. con un conjunto diferente de reglas:*las reglas rebatibles.*

Así, un programa rebatible consiste de dos clases de reglas de inferencia: [GS99] reglas *estrictas* y reglas *rebatibles*. Las reglas de inferencia son expresiones metalingüísticas. Aunque las reglas estén definidas en términos del lenguaje, ellas no son parte de Lit. Utilizaremos los símbolos $\leftarrow$ y $-\!\prec$ del meta-lenguaje para diferenciar las reglas estrictas de las reglas rebatibles.

Definición 3.13 (Regla Estricta) [Gar00]
Sean $\mathcal{A}lp$ un alfabeto, Lit su lenguaje asociado, L un miembro del lenguaje y $\{L_1, \ldots, L_n\}$ subconjunto de Lit. Una *Regla Estricta* es un par ordenado "$Cabeza \leftarrow Cuerpo$" donde $Cabeza$ es un miembro del lenguaje y $Cuerpo$ es $\{L_1, \ldots, L_i, not\ L_{i+1}, \ldots, not\ L_k\},\ 0 \leq k \leq n$. Si la cabeza es el literal L entonces se escribirá

$$L \leftarrow L_1, \ldots, L_i, not\ L_{i+1}, \ldots, not\ L_k,\ 0 \leq i \leq k \leq n$$

Como es usual, si el cuerpo es vacío, i.e., $n = 0$, entonces la regla estricta se transforma en $L \leftarrow true$ y es llamada *hecho*. En conformidad con la convención estándar, escribiremos a los hechos como "$L \leftarrow$" o simplemente como "L". ■

Definición 3.14 (Regla Rebatible) [Gar00]
Sean $\mathcal{A}lp$ un alfabeto, Lit su lenguaje asociado, L un miembro del lenguaje y $\{L_1, \ldots, L_n\}$ subconjunto de Lit. Una *Regla Rebatible* es un par ordenado "$Cabeza \prec Cuerpo$" donde $Cabeza$ es un miembro del lenguaje y $Cuerpo$ es $\{L_1, \ldots, L_i, not\ L_{i+1}, \ldots, not\ L_k\},\ 0 < k \leq n$. Si la cabeza es el literal L entonces se escribirá

$$L \prec L_1, \ldots, L_i, not\ L_{i+1}, \ldots, not\ L_k,\ 0 < i \leq k \leq n$$

■

En ambos casos, la parte izquierda del par, L, será llamada *consecuente de la regla* y la parte derecha será llamada *antecedente de la regla*.

Algunas representaciones permiten el cuerpo de la regla rebatible vacío, en cuyo caso se dice que es una *presuposición*. Por razones técnicas, los fundamentos de la P.L.R. no incluyen las presuposiciones, aunque se las contemplará en las extensiones.

Estamos interesados en conocer el poder expresivo de dichas reglas. Las reglas estrictas incluyen dos clases de conocimiento. Por un lado, permite expresar sentencias incondicionales, i.e., reglas que no contienen premisas, sobre las que nunca se pondrá en tela de juicio su valor de verdad. Los hechos estarán más allá del debate y, por lo tanto, son siempre verdaderos. Por ejemplo, representaremos con un hecho que *Tweety es un ave*

$$ave(tweety) \leftarrow true$$

Por otro lado, permite expresar conocimiento condicional certero. Si el cuerpo de la regla se basa sólo en conocimiento estricto, entonces estas reglas se comportarán en el sentido clásico: siempre que las premisas de la regla son satisfechas, tenemos permitido aplicarla y obtener la conclusión. En otras palabras, si reducimos un programa lógico rebatible a reglas estrictas,

recuperaremos a la P.L. tradicional. Una situación que podría representarse con estas reglas es que *si Fred es un avestruz entonces es un ave*

$$ave(fred) \leftarrow avestruz(fred)$$

Sin embargo, si alguno de los elementos del cuerpo depende de conocimiento rebatible, aunque la regla represente información certera, la incertidumbre del cuerpo de la regla se extenderá a la cabeza. Por lo tanto, la conclusión de la regla será puesta a consideración en un debate sobre su valor de verdad. La siguiente situación estaría en esta categoría:

$$vuela(tweety) \prec dos\_alas(tweety)$$

$$pájaro(twety) \leftarrow vuela(tweety), ave(tweety)$$

Las reglas rebatibles son reglas que pueden ser derrotadas en presencia de evidencia contraria. Estas reglas pueden interpretarse como *razones para creer en el antecedente Cuerpo proveen razones para creer en el consecuente Cabeza*[SL92] y siempre serán sujetas a debate. Por ejemplo, *si Tweety es un ave entonces vuela* sería representado como

$$vuela(tweety) \prec ave(tweety)$$

En ambos casos, se permite utilizar la negación fuerte tanto en la cabeza como en el cuerpo de las reglas. Así, podríamos representar información como:

$$\sim muerto \leftarrow vivo$$

$$falso \prec \sim verdadero$$

La negación default sólo puede ser usada en el cuerpo. Con esta clase de negación podríamos representar que si no es posible probar la culpabilidad de alguien, entonces es inocente:

$$inocente \prec not\ culpable$$

La representación de conocimiento presentada no permite anidamiento ni de la negación fuerte, ni de la negación default. Por ejemplo, no estará permitido "$\sim\sim A$" o "$not \sim not\ A$", siendo A un átomo. La única combinación que permite la sintáxis definida es de negación default seguida de negación fuerte, i.e., , "$not \sim A$", siendo A un átomo. Una lectura más explayada y profunda sobre el tema puede hallarse en [Fil01].

Algunos sistemas de razonamiento rebatible [Nut88, Bil93, Pol94] incorporan una tercer regla de inferencia, llamada *derrotador* o *undercutting defeater*. Si una regla estricta significa que "hay razones absolutas para creer la cabeza cuando se cree el cuerpo" el significado de una regla rebatible es que "el cuerpo es una buena razón para creer en la cabeza", entonces

un derrotador de la forma $L \leftarrow ?Cuerpo$ o $Cuerpo \rightsquigarrow L$, siguiendo la notacion de [Nut88], expresará que "el cuerpo es una buena razón para *no derivar* $\sim L$." "si se da el cuerpo, entonces es posible que se dé L". De este modo, mientras las dos primeras clases de reglas permiten derivar conclusiones, los derrotadores nunca soportan inferencias directamente. El rol de los derrotadores es derrotar la aplicación de una regla rebatible. En otras palabras, su propósito es prevenir derivar conclusiones que serían riesgosas.

No ha sido probado si dicha regla puede ser expresada como una combinación de las reglas del sistema definido.

3.3. Clasificación de los Programas Lógicos Rebatibles

La Programación en Lógica Rebatible puede ser clasificada de acuerdo a como las reglas, estrictas y rebatibles, son construidas. La clasificación que presentamos sigue las definiciones dadas por J. W. Lloyd en [Llo87], M. Gelfond y V. Lifschitz en [GL90] y J.J. Alferes y L. M. Pereira en [AP96].

Definición 3.15 (Regla Estricta Definida, Básica, Normal, Extendida)
Sean $\mathcal{A}lp$ un alfabeto, Lit su lenguaje asociado, $\{A, A_1, \ldots, A_n\} \subseteq Lit$ un conjunto finito de átomos y $\{L, L_1, \ldots, L_n\} \subseteq Lit$ un conjunto finito de literales.
Una *Regla Estricta* $Cabeza \leftarrow Cuerpo$, se clasifica del siguiente modo:

- Si $Cabeza = A$ es un átomo y $Cuerpo = \{A_1, \ldots, A_n\}$, entonces la regla estricta

  $$A \leftarrow A_1, \ \ldots, A_n$$

  será llamada *Regla Estricta Definida*.

- Si $Cabeza = A$ y $Cuerpo = \{A_1, \ \ldots, A_m\} \ \cup \ not\{A_{m+1}, \ \ldots, A_n\}$, con $0 \leq m \leq n$, entonces la regla estricta

  $$A \leftarrow A_1, \ \ldots, A_m, not\ A_{m+1}, \ \ldots, not\ A_n$$

  será llamada *Regla Estricta Normal*.

- si $Cabeza = L$ y $Cuerpo = \{L_1, \ \ldots, L_n\}$, entonces la regla estricta

  $$L \leftarrow L_1, \ \ldots, L_n$$

  será llamada *Regla Estricta Básica*.

- Si $Cabeza = L$ y $Cuerpo = \{L_1, \ldots, L_m\} \cup not\{L_{m+1}, \ldots, L_n\}$, con $0 \leq m \leq n$, entonces la regla estricta

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

será llamada *Regla Estricta Extendida*.

■

Una regla estricta no es equivalente a la implicación de la lógica clásica, i.e., no debe ser interpretada como una implicación material sino como una *regla de inferencia*. El metasímbolo $\leftarrow$ no tiene asignado una tabla de verdad y su relación con otros metasímbolos no es equivalente a la relación que puede existir con la implicación clásica. Por ejemplo, en la lógica clásica se cumple la relación contrapositiva entre la implicación y la negación $((A \rightarrow B) \Leftrightarrow (\neg B \rightarrow \neg A))$. Sin embargo, como se mostrará más adelante, las consecuencias de un programa lógico rebatible no cumplen esta relación entre los símbolos $\rightarrow$ y $\sim$.

Definición 3.16 (Regla Rebatible Definida, Básica, Normal, Extendida)
Sean $\mathcal{A}lp$ un alfabeto, Lit su lenguaje asociado, $\{A, A_1, \ldots, A_n\} \subseteq Lit$ un conjunto finito de átomos y $\{L, L_1, \ldots, L_n\} \subseteq Lit$ un conjunto finito de literales.
Una *Regla Rebatible* $Cabeza \prec Cuerpo$, se clasifica del siguiente modo:

- Si $Cabeza = A$ es un átomo y $Cuerpo = \{A_1, \ldots, A_n\}$, con $n > 0$, entonces la regla rebatible

$$A \prec A_1, \ldots, A_n$$

será llamada *Regla Rebatible Definida*.

- Si $Cabeza = A$ y $Cuerpo = \{A_1, \ldots, A_m\} \cup not\{A_{m+1}, \ldots, A_n\}$, con $0 < m \leq n$, entonces la regla rebatible

$$A \prec A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n$$

será llamada *Regla Rebatible Normal*.

- si $Cabeza = L$ y $Cuerpo = \{L_1, \ldots, L_n\}$, con $n > 0$,entonces la regla rebatible

$$L \prec L_1, \ldots, L_n$$

será llamada *Regla Rebatible Básica*.

- Si $Cabeza = L$ y $Cuerpo = \{L_1, \ldots, L_m\} \cup not\{L_{m+1}, \ldots, L_n\}$, con $0 < m \leq n$, entonces la regla rebatible

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

será llamada *Regla Rebatible Extendida*.

■

Definición 3.17 (Programa Lógico Rebatible)
Un *programa lógico rebatible*[GS99] es un conjunto contable de reglas estrictas y rebatibles. Si $\mathcal{P}$ es un programa lógico rebatible, distinguiremos el subconjunto Π de reglas estrictas en $\mathcal{P}$ y el subconjunto Δ de reglas rebatibles en $\mathcal{P}$. Cuando se requiera denotaremos $\mathcal{P}$ como $\langle \Pi, \Delta \rangle$. ■

De ahora en más, abreviaremos "programas lógicos rebatibles" con las iniciales correspondientes en letra minúscula, i.e., "*plr*". Esta abreviatura fue originalmente introducida por Fillottrani en [Fil01].

Algunos autores [Gar00] prefieren dividir al conjunto de reglas estrictas en Θ el conjunto de los hechos y Ω el resto de las reglas estrictas. Así un *plr* puede ser notado también como $\mathcal{P} = \langle \Theta, \Omega, \Delta \rangle$. Cada vez que querramos hacer explícito el conjunto de hechos utilizaremos esta notación.

Siempre que la signatura y, por lo tanto, el alfabeto de un programa no estén definidos, asumiremos que la signatura Σ está dada precisamente por todas los símbolos de función y símbolos predicativos que aparezcan explícitamente en el programa. En caso de que el programa no contenga ninguna constante, se eligirá una constante arbitraria y se la incorporará a la signatura.

Del mismo modo en que hemos clasificado a las reglas estrictas y rebatibles, lo haremos con los *plr*.

Definición 3.18 (Programa Lógico Rebatible Definido, Basico, Normal y Extendido)
Diremos que un *plr* es:

- *definido* si sus reglas estrictas y rebatibles son definidas.
- *básico* si sus reglas estrictas y rebatibles son básicas.
- *normal* si sus reglas estrictas y rebatibles son normales.
- *extendido* si sus reglas estrictas y rebatibles son extendidas.

■

Una vez representado el conocimiento en un *plr*, estaremos interesados en responder a consultas basados en tal conocimiento.

Definición 3.19 (Consulta Rebatible - Meta Rebatible)
Una *consulta rebatible* , denotada por $\{Q_1, \ldots, Q_n\}$, es una conjunto de literales fijos posiblemente precedidos por el meta-símbolo "not". La *meta rebatible* correspondiente a la consulta se define como $\multimap Q_1, \ldots, Q_n$.
Análogamente a una regla, una consulta rebatible y una meta rebatible pueden ser clasificadas en definida, básica, normal y extendida dependiendo de si cada Q_i $(1 \leq i \leq n)$ es un átomo o un literal o un literal extendido. ■

Entenderemos a una consulta rebatible como la conjunción de los literales que la componen, de modo que una consulta será consecuencia de un programa, si lo es cada uno de sus miembros. Esta idea quedará formalizada en la sección 3.4, al analizar la semántica operacional de la P.L.R..

El lenguaje de un *plr* no permite el uso de variables, y por lo tanto, un *plr* no permitirá en la definición de sus reglas términos no fijos. Es posible pensar a un programa lógico que contenga variables como un *esquema* del programa[Lif96]. Así una regla R que contenga variables es un esquema de regla y puede ser vista como una notación abreviada de todas sus posibles instanciaciones, i.e., como una abreviatura de **fijo**(R). Este concepto puede ser extendido para los *plr*. Formalizamos esta idea extendiendo las definiciones de **fijo** presentadas en [Lif96].

Definición 3.20 (Función Fijo)
Sea $\mathcal{A}lp$ un alfabeto basado en una signatura Σ. Si $Cabeza \leftarrow Cuerpo$ es un esquema de regla estricta, entonces

$$\textbf{fijo}(Cabeza \leftarrow Cuerpo) \;=\; \Big\{ \begin{array}{l} Cabeza' \leftarrow Cuerpo' | \text{ existe una sustitución } \sigma \text{ bajo } \Sigma \\ \text{tal que } Cabeza' = Cabeza\,\sigma \text{ y } Cuerpo' = Cuerpo\,\sigma \\ \text{siendo } Cabeza' \bigcup Cuerpo' \text{ literales fijos } \Big\} \end{array}$$

Análogamente se define para un esquema de regla rebatible, reemplazando en la definición anterior el meta-símbolo $\leftarrow$ por $\multimap$.
Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un esquema de *plr*. Definimos a fijo$(\mathcal{P})$ como

$$\textbf{fijo}(\mathcal{P}) \;= \bigcup_{R \in \Pi} \textbf{fijo}(R) \cup \bigcup_{R \in \Delta} \textbf{fijo}(R)$$

■

De este modo, mientras que un esquema de un *plr* puede contener un conjunto finito de esquemas de reglas rebatibles y estrictas, el *plr* fijo obtenido a partir de aquél, puede consistir de infinitas reglas fijas rebatibles y estrictas, con la sola presencia de un símbolo de función.

De ahora en más, siempre que hagamos referencia a un esquema de *plr* $\mathcal{P}$, nos estaremos refiriendo a su equivalente semántico **fijo**$(\mathcal{P})$.

Ejemplo 3.1 El siguiente es un esquema de *plr* básico $\mathcal{P}$ que representa información sobre la nacionalidad de una persona.

$$\Pi = \{ciudadanía(andrea, italiana), origen(andrea, argentina), optó(andrea, francesa)\}$$

$$\Delta = \{nacionalidad(X,Y) -\!\prec ciudadanía(X,Y)$$

$$nacionalidad(X,Y) -\!\prec origen(X,Y)$$

$$nacionalidad(X,Y) -\!\prec optó(X,Y)$$

$$\sim nacionalidad(X,Y) -\!\prec origen(X,Y), optó(X,Z), Z \neq Y$$

$$doble\_nacionalidad(X) -\!\prec nacionalidad(X,Y), nacionalidad(X,Z), Z \neq Y\}$$

Aunque los dos últimos esquemas de reglas contengan las condiciones $Z \neq Y$, éstas pueden ser eliminadas al reemplazarlas por las instancias fijas correspondientes, teniendo en cuenta la restricción.

En este caso la signatura subyacente es $\langle\{X,Y,Z\}, \{andrea^0, italiana^0, argentina^0, francesa^0\}, \{optó^2, origen^2, nacionalidad^2, ciudadanía^2, \neq^2, doble\_nacionalidad^1\}\rangle$ ■

Este trabajo estará circunscripto a programas lógicos rebatibles básicos,*plrb*[Fil01], i.e., *plr* con negación fuerte. Siempre que no sea ambiguo, nos referiremos a *plrb* simplemente como programas lógicos rebatibles o programas.

3.4. Semántica Operacional

La semántica operacional es el modo de describir el significado de un programa en forma procedural. Las fórmulas bien formadas y su sintáxis, los axiomas, reglas de inferencia y pruebas en el lenguaje del programa son tratados a través de esta semántica.

La semántica operacional de la P.L. tradicional está basada en el método clásico de resolución aplicado a cláusulas de Horn. A pesar de las ventajas teóricas y prácticas que esto conlleva, simultáneamente supone una limitación tanto sintáctica como semántica, como así también restringe en algunos casos la aplicabilidad de la programación lógica en resolución de problemas.

Con el objeto de superar estas limitaciones la P.L.R. ataca varios frentes. Por un lado, permite el uso de las dos clases de negación. Por el otro, la semántica operacional se basa en los *sistemas de argumentación*, que constituyen una formalización del razonamiento no monótono.

Una creencia será explicada después de que el agente realice un análisis dialéctico teniendo en cuenta las explicaciones tentativas que pueda construir para tal creencia. Las explicaciones que soportan a la creencia se denominan *argumentos* y son el fundamento del razonamiento rebatible.

La construcción de los argumentos es monotónica, i.e., un argumento permanece inalterable aún si agregamos nuevas premisas. La no monotonicidad no está fundamentada en términos de un argumento simple, sino en términos de interacción entre argumentos conflictivos. Una creencia q es aceptada por un agente si existe un argumento $\mathcal{A}$ de q que lo garantice, i.e., no existen argumentos en contra que derroten a $\mathcal{A}$. Ya que los derrotadores son también argumentos podrían existir derrotadores para estos últimos y así continuando. La pérdida de la monotonicidad se reflejará al descartar un argumento por ser derrotado por otro.

En las siguientes secciones formalizaremos la semántica operacional de la Programación en Lógica Rebatible Básica (de ahora en más P.L.R.B.).

3.4.1. Derivaciones Estricta y Rebatible

Al explicar las posibles situaciones que pueden ser representadas con la P.L.R. se hizo hincapié en diferenciar la información certera de aquella que siempre será puesta a consideración en un debate. Esto se refleja en las siguientes definiciones, que son adaptaciones de las dadas por [GS99, Gar00], y que formalizan la idea de las derivaciones estricta y rebatible.

Definición 3.21 (Derivable Rebatiblemente)
Sean $\prec Q_1, \ldots, Q_n$ una meta rebatible básica y R una regla estricta o rebatible con cabeza Q_i, $1 \leq i \leq n$ y cuerpo $L_1, \ldots, L_m$. Luego la nueva meta rebatible $\prec Q_1, \ldots, Q_{i-1}, L_1, \ldots, L_m, Q_{i+1}, \ldots, Q$
es *derivable rebatiblemente*. ■

Definición 3.22 (Derivación rebatible)
Sean $\mathcal{P}$ un programa rebatible y Q una consulta rebatible básica. Una *derivación rebatible* consiste de una secuencia finita de metas rebatibles $\prec Q = Q_0, Q_1, \ldots, Q_n$, donde cada Q_{i+1} es derivado rebatiblemente a partir de $Q_i, 0 \leq i < n$ y $Q_n = \square$ es la meta vacía. Diremos que Q se deriva rebatiblemente a partir de $\mathcal{P}$. ■

Es interesante notar que pueden existir diferentes derivaciones para un mismo literal. Informalmente, estamos representando que un agente tendrá diferentes formas de explicar una misma situación.

Ejemplo 3.2 Consideremos el ejemplo 3.1. Existen tres derivaciones rebatibles diferentes para la consulta $doble\_nacionalidad(andrea)$:

$-\prec doble\_nacionalidad(andrea)$,
$-\prec nacionalidad(andrea, argentina), nacionalidad(andrea, italiana)$,
$-\prec origen(andrea, argentina), nacionalidad(andrea, italiana)$,
$-\prec ciudadanía(andrea, italiana)$,
$\square$

$-\prec doble\_nacionalidad(andrea)$,
$-\prec nacionalidad(andrea, argentina), nacionalidad(andrea, francesa)$,
$-\prec origen(andrea, argentina), nacionalidad(andrea, francesa)$,
$-\prec optó(andrea, francesa)$,
$\square$

$-\prec doble\_nacionalidad(andrea)$,
$-\prec nacionalidad(andrea, italiana), nacionalidad(andrea, francesa)$,
$-\prec ciudadanía(andrea, italiana), nacionalidad(andrea, francesa)$,
$-\prec optó(andrea, francesa)$,
$\square$

Hemos obtenido diferentes modos de que el agente explique la conclusión rebatible de que Andrea tiene doble nacionalidad. ■

Un caso particular de la definición anterior son las derivaciones que no involucran reglas rebatibles,i.e., las derivaciones estrictas. Una derivación estricta no deberá ser puesta a consideración en un debate, ya que su explicación depende solamente de información que es certera.

Definición 3.23 (Derivable estrictamente)
Sean $-\prec Q_1, \ldots, Q_n$ una meta rebatible básica y R una regla estricta con cabeza Q_i, $1 \leq i \leq n$ y cuerpo $L_1, \ldots, L_m$. Luego la nueva meta rebatible $-\prec Q_1, \ldots, Q_{i-1}, L_1, \ldots, L_m, Q_{i+1}, \ldots, Q_n$ es *derivable estrictamente*. ■

Definición 3.24 (Derivación Estricta)
Sean $\mathcal{P}$ un *plr* y Q una consulta rebatible básica. Una *derivación estricta* consiste de una secuencia finita de metas rebatibles $-\prec Q = Q_0, Q_1, \ldots, Q_n$, donde cada Q_{i+1} es derivado estrictamente a partir de $Q_i, 0 \leq i < n$ y $Q_n = \square$ es la meta vacía. Diremos que Q se deriva estrictamente a partir de $\mathcal{P}$. ■

Ejemplo 3.3 Consideremos el ejemplo de las nacionalidades. El literal

$$ciudadanía(andrea, italiana)$$

tiene una derivación estricta

$$-\prec ciudadanía(andrea, italiana),\ \square$$

utilizando la regla estricta $ciudadanía(andrea, italiana) \leftarrow true$. ■

Observación 3.1 *[Gar00] La derivación rebatible es monotónica:"Sean $\mathcal{P}$ un plr y $\mathcal{R}$ un conjunto de reglas, si h tiene una derivación rebatible desde $\mathcal{P}$, entonces h tiene una derivación rebatible desde $\mathcal{P} \cup \mathcal{R}$".*

A pesar de que el proceso de construcción de un argumento para un literal es monotónico, se denomina rebatible a la derivación, ya que al intervenir en el proceso de interacción entre diferentes argumentos, podría existir información contradictoria que no permitiera aceptar como creencia de un agente a dicho literal.

Proposición 3.1 *[Gar00] Si un plr $\mathcal{P}$ no contiene hechos, entonces no se puede obtener ninguna derivación rebatible.*

Ya que un *plr* básico permite representar información contradictoria, podremos derivar rebatiblemente un literal L y su complemento $\overline{L}$.

Definición 3.25 (Reglas Contradictorias) [Gar00]
Un conjunto de reglas es contradictorio si y sólo si, a partir de ese conjunto es posible obtener derivaciones rebatibles para un literal L y su complemento $\overline{L}$. ■

Definición 3.26 (Programa Lógico Rebatible Contradictorio)[Gar00]
Un programa rebatible $\mathcal{P}$ es contradictorio si $\Pi \cup \Delta$ es contradictorio. ■

Siendo que está permitido el uso de la negación fuerte en la cabeza de las reglas, los *plr* podrán tener representada información contradictoria. Así $\Pi \cup \Delta$ será contradictorio en la mayoría de los casos. Sin embargo, la información certera deberá ser consistente y, por lo tanto, no permitiremos contradicciones en Π. La siguiente es una suposición que haremos para todos los programas lógicos:

> "En todo *plr* $\mathcal{P} = \langle \Pi, \Delta \rangle$, se asume que el conjunto Π es un conjunto no contradictorio."

Como consecuencia inmediata de esta suposición tenemos el siguiente lema.

Lema 3.1 *[Gar00] Sea* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un plr, dado que el conjunto* Π *se asume no contradictorio, no existen dos literales complementarios que tengan derivaciones estrictas a partir de* $\mathcal{P}$*.*

Ejemplo 3.4 Considere nuevamente el ejemplo de las ciudadanías. Note que aunque es posible derivar rebatiblemente información contradictoria como $nacionalidad(andrea, argentina)$ y $\sim nacionalidad(andrea, argentina)$, no es posible derivar en forma estricta información contradictoria. ■

3.4.2. Argumentación Rebatible

De un programa contradictorio podremos derivar algún literal L y su complementario $\overline{L}$. Un agente que razone de este modo deberá decidir si cree en L o en $\overline{L}$. Es de esperar que el agente no crea simultáneamente en ambos literales. Para esto deberemos proveer alguna técnica que ayude a decidir. Existen diferentes enfoques que proponen un mecanismo de decisión y que requieren de un análisis global o parcial de la información del programa. La propuesta desarrollada por Gelfond y Lifschitz [GL90] para la P.L. Extendida tradicional (i.e., no incluye reglas rebatibles) imita el comportamiento de la lógica clásica. Frente a información contradictoria, la lógica clásica deriva el lenguaje completo. Cuando un *pl* extendido permite la derivación de literales complementarios, la semántica correspondiente refleja esta situación derivando *Lit* (el lenguaje del programa lógico extendido). Un agente que incorpora alguna información contradictoria, no puede aislar la inconsistencia y, por lo tanto, creerá en absolutamene todo. Esta solución al problema no es la adecuada para representar las creencias de un agente.

Una propuesta alternativa la presenta Vaucheret [VS99] combinando Revisión de Creencias y la Programación en Lógica Extendida. Al incorporar un nuevo literal cuya negación se derive del *pl* se requiere de un proceso de revisión. Una vez que se encontraron todas las explicaciones para el literal, se selecciona un conjunto de hechos miembros de dichas explicaciones, de forma tal que al eliminar ese conjunto del *pl* todas las explicaciones sean *heridas*. En un *pl* sólo los hechos son considerados revisables, ya que el conocimiento general se prefiere al conocimiento general. Si bien este proceso maneja la inconsistencia de una manera elegante ante nueva información debe retractar algún conjunto de hechos del programa o bien su semántica se tranformará en el lenguaje

Las propuestas analizadas fueron desarrolladas e implementadas para la P.L.B.. Existen otros enfoques que intentan solucionar el problema de representar inconsistencias en la P.L.R.B..

El enfoque que se estudiará con el objeto de determinar cuales literales son consecuencia de nuestro programa es la *agumentación rebatible*[Sim89, SL92]. Un *análisis dialéctico* será construido para determinar si un literal es consecuencia de nuestro programa. Los argumentos son las unidades básicas de esta clase de razonamiento y se formalizan a continuación.

Definición 3.27 (Estructura de Argumento)[GSC98]
Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un *plr*. Una *estructura de argumento* para un literal fijo h es un par $\langle \mathcal{A}, h \rangle$, donde $\mathcal{A}$ es un conjunto de reglas rebatibles de Δ que cumple las siguientes condiciones:

1. existe una derivación rebatible de h a partir de $\Pi \cup \mathcal{A}$,
2. $\Pi \cup \mathcal{A}$ no es contradictorio, y
3. $\mathcal{A}$ es minimal con respecto a la inclusión de conjuntos, i.e., no existe $\mathcal{A}' \subset \mathcal{A}$ tal que $\mathcal{A}'$ satisface las dos primeras condiciones.

■

Ejemplo 3.5 Consideremos nuevamente el programa del ejemplo 3.1. Algunas posibles estructuras de argumentos para la consulta $doble\_nacionalidad(andrea)$ son

$$\langle \mathcal{A}_1, doble\_nacionalidad(andrea) \rangle$$

$$\langle \mathcal{A}_2, doble\_nacionalidad(andrea) \rangle$$

$$\langle \mathcal{A}_3, doble\_nacionalidad(andrea) \rangle$$

donde:

$$\mathcal{A}_1 = \{ \; doble\_nac(andrea) -\!\prec nac(andrea, argentina), nac(andrea, italiana);$$
$$nac(andrea, argentina) -\!\prec origen(andrea, argentina);$$
$$nac(andrea, italiana) -\!\prec ciudadanía(andrea, italiana) \; \}$$

$$\mathcal{A}_2 = \{ \; doble\_nac(andrea) -\!\prec nac(andrea, argentina), nac(andrea, francesa);$$
$$nac(andrea, argentina) -\!\prec origen(andrea, argentina);$$
$$nac(andrea, francesa) -\!\prec optó(andrea, francesa) \; \}$$

$$\mathcal{A}_3 = \{ \; doble\_nac(andrea) -\!\prec nac(andrea, italiana), nac(andrea, francesa);$$
$$nac(andrea, italiana) -\!\prec ciudadanía(andrea, italiana);$$
$$nac(andrea, francesa) -\!\prec optó(andrea, francesa) \; \}$$

Existen otras estructuras de argumentos que se generan al instanciar en forma simétrica las nacionalidades en la regla $doble\_nacionalidad(andrea)$. ■

Definición 3.28 (Igualdad de Estructuras de Argumento)[Gar00]
Dos estructuras de argumento $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ son *iguales* si y sólo si $\mathcal{A}_1 = \mathcal{A}_2$ y $h_1 = h_2$. ■

Definición 3.29 (Subargumento) [Gar00]
Una estructura de argumento $\langle \mathcal{B}, q \rangle$ es una *subestructura de argumento* de $\langle \mathcal{A}, h \rangle$ si y sólo si $\mathcal{B} \subseteq \mathcal{A}$. ■

Ejemplo 3.6 Consideremos las estructuras de argumentos del ejemplo 3.1. Diremos que $\langle \mathcal{A}, nac(andrea, argentina) \rangle$ es una subestructura de $\langle \mathcal{A}_1, doble\_nacionalidad(andrea) \rangle$, siendo

$$\mathcal{A} = \{ nac(andrea, argentina) -\!\prec origen(andrea, argentina) \}$$

ya que $\mathcal{A} \subseteq \mathcal{A}_1$. ■

Aunque cuando se referencia a un *argumento* para h se hace mención de $\mathcal{A}$, siempre que no sea ambiguo abusaremos del lenguaje denominando a la estructura de argumento $\langle \mathcal{A}, h \rangle$ simplemente argumento. Análogamente, llamaremos a una subestructura *subargumento*.

3.4.3. Relaciones entre argumentos

Con el objeto de exponer a un debate a los argumentos, deberemos definir algunas relaciones entre ellos, como por ejemplo, cuáles argumentos son contra-argumentos de otros y frente a esta situación cuál argumento gana el debate.

Definición 3.30 (Literales en Desacuerdo) [Gar00]
Sea $\mathcal{P} = (\Pi, \Delta)$ un *plr*. Dos literales $h,\ h'$ están en desacuerdo, si y sólo si el conjunto $\Pi \cup \{h, h'\}$ es contradictorio. ■

Ejemplo 3.7 Supongamos que representamos la siguiente información: los perros juguetones no son esbeltos; los perros que no son esbeltos no son lindos; Pichus es un perro.

$$\begin{array}{rcll} \Pi & = & \{ & \sim esbelto(X) \leftarrow perro(X), juguetón(X). \\ & & & \sim lindo(X) \leftarrow perro(X), \sim esbelto(X) \\ & & & perro(pichus) \ \} \end{array}$$

Los literales $lindo(pichus)$ y $juguetón(pichus)$ son literales en desacuerdo, ya que $\Pi \cup \{lindo(pichus), juguetón(pichus)\}$ es contradictorio, ya que es posible derivar a partir de $juguetón(pichus)$ el literal $\sim lindo(pichus)$. ■

Un caso trivial en que los literales están en desacuerdo es cuando los literales son complementarios. Así $\Pi \cup \{p, \sim p\}$ es contradictorio.

Definición 3.31 (Contraargumento) [Gar00]
Sean $\mathcal{P} = (\Pi, \Delta)$ un *plr* y $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ dos estructuras de argumento obtenidas a partir de $\mathcal{P}$. Decimos que $\langle \mathcal{A}_1, h_1 \rangle$ *contra-argumenta* a $\langle \mathcal{A}_2, h_2 \rangle$ en el literal h, si y sólo si, existe un subargumento $\langle \mathcal{A}, h \rangle$ de $\langle \mathcal{A}_2, h_2 \rangle$ tal que h y h_1 están en desacuerdo. El argumento $\langle \mathcal{A}, h \rangle$ se llama *subargumento de desacuerdo* y el literal h será el *punto de contra-argumentación*. Si $\langle \mathcal{A}_1, h_1 \rangle$ contra-argumenta a $\langle \mathcal{A}_2, h_2 \rangle$, entonces se dirá que $\langle \mathcal{A}_1, h_1 \rangle$ *ataca* a $\langle \mathcal{A}_2, h_2 \rangle$, o que $\langle \mathcal{A}_1, h_1 \rangle$ es un contra-argumento para $\langle \mathcal{A}_2, h_2 \rangle$. ■

Ejemplo 3.8 Consideremos el ejemplo 3.1. El siguiente argumento

$$\mathcal{A}_4 \;=\; \{\sim nac(andrea, argentina) \prec origen(andrea, argentina), optó(andrea, francesa)\}$$

está en desacuerdo con los argumentos $\mathcal{A}_1$ y $\mathcal{A}_2$. Los literales en desacuerdo son

$$nac(andrea, argentina) \text{ y } \sim nac(andrea, argentina).$$

Los subargumentos de $\mathcal{A}_1$ y $\mathcal{A}_2$ atacados por $\mathcal{A}_4$ son

$$\{nac(andrea, argentina) \prec origen(andrea, argentina)\}$$

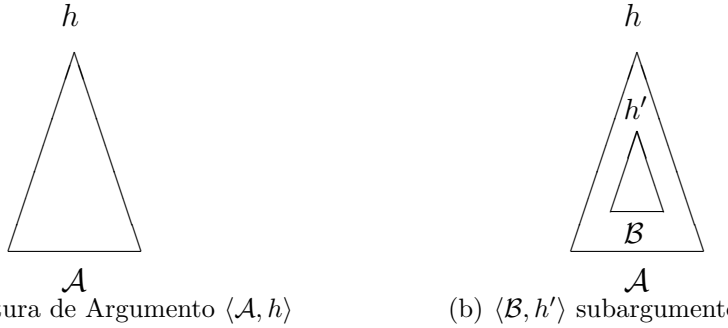
y

$$\{nac(andrea, argentina) \prec origen(andrea, argentina)\}$$

■

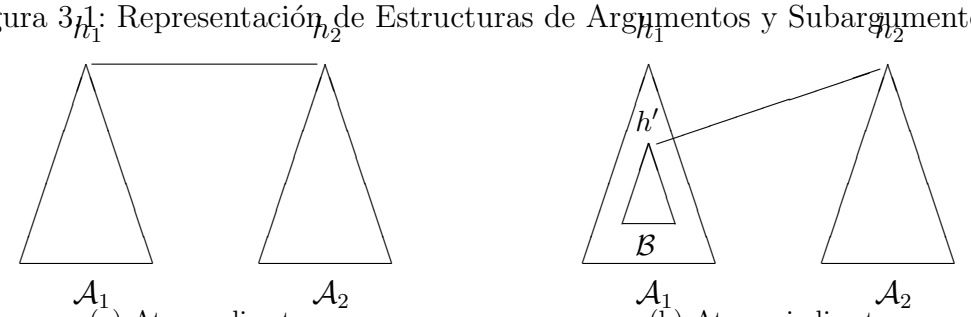
Un modo más sencillo de comprender la relación de ataque entre estructuras de argumentos es a través de la gráfica. Utilizaremos a partir de ahora una notación gráfica que fue introducida en [SCnG94]. Bajo dicha notación, las estructuras de argumentos estarán representadas por triángulos isóceles cuyo vértice superior será etiquetado con el literal que sustenta dicho argumento. La base del triángulo se etiqueta con el nombre que referencia al argumento. Un triángulo dentro de otro, indicará una subestructura de argumento de la estructura de argumento representada por el triángulo externo (Figura 3.1).

Para representar que una estructura de argumento ataca a otra estructura de argumento, se traza una línea recta desde los vértices de los triángulos que los representan. La Figura 3.2 presenta dos modos de ataque: directo e indirecto. En el primer caso se muestra un ataque directo, i.e., los literales h_1 y h_2 están en desacuerdo y, por lo tanto, las estructuras $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ son contra-argumentos. En el ataque indirecto, la estructura de argumento $\langle \mathcal{A}_2, h_2 \rangle$ contraargumenta a la subestructura de argumento $\langle \mathcal{B}, h' \rangle$ de $\langle \mathcal{A}_1, h_1 \rangle$,i.e., h_2 y h' están en desacuerdo.



(a) Estructura de Argumento $\langle \mathcal{A}, h \rangle$ (b) $\langle \mathcal{B}, h' \rangle$ subargumento de $\langle \mathcal{A}, h \rangle$

Figura 3.1: Representación de Estructuras de Argumentos y Subargumentos.



(a) Ataque directo. (b) Ataque indirecto

Figura 3.2: Representación de ataque entre estructuras de argumentos.

La semántica operacional está basada en la construcción de argumentos y contraargumentos para los literales, luego es interesante determinar cuando no es posible encontrar contraargumentos en un programa. Un caso particular que analizaremos es el de los literales que tienen una derivación estricta.

Lema 3.2 *Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb. Si $\langle \mathcal{A}_1, h_1 \rangle$ es un contra-argumento para $\langle \mathcal{A}_2, h_2 \rangle$, entonces el conjunto $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2$ es contradictorio.*

Demostración: Por hipótesis $\langle \mathcal{A}_1, h_1 \rangle$ es un contra-argumento para $\langle \mathcal{A}_2, h_2 \rangle$, luego por definición existe una subestructura $\langle \mathcal{A}, h \rangle$ de $\langle \mathcal{A}_2, h_2 \rangle$ tal que h y h_1 están en desacuerdo, i.e., $\Pi \cup \{h, h_1\}$ es contradictorio. Por definición de estructura de argumento, existe una derivación rebatible de h a partir de $\Pi \cup \mathcal{A}$ y una derivación rebatible de h_1 a partir de $\Pi \cup \mathcal{A}_1$. Así $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2$ es contradictorio. ■

Lema 3.3 *[Gar00] Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb. Si $\langle \mathcal{A}, h \rangle$ es una estructura de argumento donde $\mathcal{A} = \emptyset$, i.e., h tiene una derivación estricta, entonces no existe un contraargumento para $\langle \mathcal{A}, h \rangle$.*

Demostración: Supongamos por el contrario que existe un contrargumento $\langle \mathcal{B}, h' \rangle$ de $\langle \mathcal{A}, h \rangle$. Luego por proposición 3.2 $\Pi \cup \mathcal{A} \cup \mathcal{B}$ es contradictorio. Siendo $\mathcal{A} = \emptyset$, entonces $\Pi \cup \mathcal{B}$ es contradictorio, lo que contradice la condición (2) de la definición de estructura de argumento. Por lo tanto, no existe ningún contraargumento para $\langle \mathcal{A}, h \rangle$. ■

Lema 3.4 *[Gar00] Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb. Si $\langle \mathcal{A}, h \rangle$ es una estructura de argumento donde $\mathcal{A} = \emptyset$, i.e., h tiene una derivación estricta, entonces no existe argumento $\langle \mathcal{B}, q \rangle$ tal que $\langle \mathcal{A}, h \rangle$ es un contraargumento para $\langle \mathcal{B}, q \rangle$.*

Demostración: Supongamos que existe una estructura de argumento $\langle \mathcal{B}, q \rangle$, en las condiciones del lema. Por lema 3.2 $\mathcal{B} \cup \mathcal{A} \cup \Pi$ es contradictorio. Por hipótesis, $\mathcal{A} = \emptyset$, luego $\mathcal{B} \cup \Pi$ es contradictorio. Pero entonces $\mathcal{B}$ no es un argumento, ya que se estaría violando la segunda condición de la definición de estructura de argumento. ■

Proposición 3.2 *Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb. Si $\langle \mathcal{A}, h \rangle$ es una estructura de argumento donde $\mathcal{A} = \emptyset$, i.e., h tiene una derivación estricta, entonces no existe ninguna estructura de argumento $\langle \mathcal{B}, q \rangle$ que contraargumente o sea contraargumentada por $\langle \mathcal{A}, h \rangle$.*

Demostración: Por lema 3.3, no existe ninguna estructura de argumento $\langle \mathcal{B}, q \rangle$ que sea contraargumento de $\langle \mathcal{A}, h \rangle$.
Por lema 3.4, $\langle \mathcal{A}, h \rangle$ no es contraargumento de ninguna estructura de argumento $\langle \mathcal{B}, q \rangle$. ■

La proposición anterior establece que no es posible que un literal con una derivación estricta sea considerado en un debate para determinar su valor de verdad. Dichos literales son siempre verdaderos.

La siguiente proposición muestra que la definición de contraargumento cumple la propiedad simétrica.

Proposición 3.3 *[Gar00] Si una estructura de argumento $\langle \mathcal{A}_1, h_1 \rangle$ contraargumenta a otra $\langle \mathcal{A}_2, h_2 \rangle$ en el punto h, con el subargumento de desacuerdo $\langle \mathcal{A}, h \rangle$, entonces también se cumple que $\langle \mathcal{A}, h \rangle$ contraargumenta a $\langle \mathcal{A}_1, h_1 \rangle$ en el punto h_1.*

Demostración: Como $\langle \mathcal{A}_1, h_1 \rangle$ contraargumenta a $\langle \mathcal{A}_2, h_2 \rangle$ en el punto h, entonces h y h_1 están en desacuerdo, por lo tanto, $\langle \mathcal{A}, h \rangle$ contraargumenta a $\langle \mathcal{A}_1, h_1 \rangle$ en el punto h_1, ya que existe un subargumento de $\langle \mathcal{A}_1, h_1 \rangle$, que es el mismo $\langle \mathcal{A}_1, h_1 \rangle$, tal que h y h_1 están en desacuerdo. ■

Corolario 3.1 *Si una estructura de argumento $\langle \mathcal{A}_1, h_1 \rangle$ contraargumenta a otra $\langle \mathcal{A}_2, h_2 \rangle$ en el punto h_2, entonces $\langle \mathcal{A}_2, h_2 \rangle$ contraargumenta a $\langle \mathcal{A}_1, h_1 \rangle$.*

La proposición y el colorario anteriores muestran una evidente necesidad de un mecanismo que nos permita comparar los argumentos y determinar cuál de ellos es más fuerte y, por lo tanto,

gana en el debate. En la sección que sigue definiremos formalmente como se realiza el debate (análisis dialéctico) sin especificar el mecanismo de comparación. Más adelante estudiaremos los diferentes modos de comparar y decidir entre argumentos contradictorios.

3.4.4. Relación de Preferencia

Ya que no hemos especificado ningún modo de comparar argumentos, asumiremos una relación de preferencia genérica $\mathcal{R}$.

Definición 3.32 (Relación de Preferencia $\mathcal{R}$)[CS02]
Sean $\mathcal{P}$ un programa lógico rebatible y $\texttt{Args}(\mathcal{P})$ el conjunto de todos los argumentos que se pueden obtener a partir de $\mathcal{P}$. Una *relación de preferencia* $\mathcal{R} \subseteq \texttt{Args}(\mathcal{P}) \times \texttt{Args}(\mathcal{P})$ es cualquier relación binaria definida sobre $\texttt{Args}(\mathcal{P})$. ■

Así, si un argumento $\langle \mathcal{A}_2, h_2 \rangle$ es mejor o igual que otro $\langle \mathcal{A}_1, h_1 \rangle$, lo notaremos

$$\langle \mathcal{A}_1, h_1 \rangle \; \mathcal{R} \; \langle \mathcal{A}_2, h_2 \rangle$$

o bien como par ordenado

$$\Big(\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle \Big).$$

Para distinguir entre los diferentes mecanismos de comparación, notaremos cada vez que sea necesario $\mathcal{R}_{\texttt{P}}$, indicando con $\texttt{P}$ el mecanismo de preferencia que se esté utilizando.

Dados dos argumentos deberemos considerar dos casos especiales, alguno de ellos es mejor o bien son incomparables. Diferenciamos dos relaciones binarias, basadas en la relación de preferencia $\mathcal{R}$, que formalizan estas ideas y que nos permitirán caracterizar posteriormente la noción de derrotador.

Definición 3.33 (Relaciones Binarias $\succ$ y $\asymp$) [Cn01]
Sean $\texttt{Args}(\mathcal{P})$ el conjunto de todos los argumentos que se pueden obtener a partir de un programa lógico $\mathcal{P}$ y $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle \in \texttt{Args}(\mathcal{P})$. Entonces, escribiremos,

1. $\mathcal{A}_1 \succ \mathcal{A}_2$ si $\mathcal{A}_1 \mathcal{R} \mathcal{A}_2$ y $\mathcal{A}_2 \not\mathcal{R} \mathcal{A}_1$. En este caso diremos que $\mathcal{A}_1$ es estrictamente preferido por sobre $\mathcal{A}_2$.

2. $\mathcal{A}_1 \asymp \mathcal{A}_2$ si

   a) $\mathcal{A}_1 \mathcal{R} \mathcal{A}_2$ y $\mathcal{A}_2 \mathcal{R} \mathcal{A}_1$.

   b) $\mathcal{A}_1$ y $\mathcal{A}_2$ no son comparables por $\mathcal{R}$, i.e., los pares $(\mathcal{A}_1, \mathcal{A}_2)$ y $(\mathcal{A}_2, \mathcal{A}_1)$ no pertenecen a la relación $\mathcal{R}$.

■

Notaremos $\succ_{\mathtt{P}}$ y $\asymp_{\mathtt{P}}$ cuando querramos asociar las relaciones a un criterio de preferencia determinado.

3.4.5. Análisis Dialéctico

Cuando deseamos determinar si un literal es consecuencia de nuestro *plr* debemos considerar todos los argumentos a favor y en contra de tal literal. Con esta información se realiza un análisis dialéctico en el que los argumentos son comparados y ....

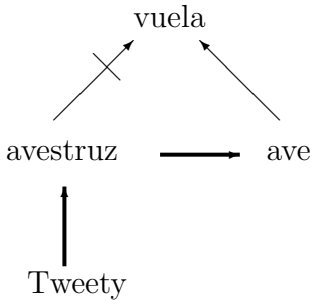
Dados dos contraargumentos deberemos considerar dos casos, alguno de ellos es mejor o bien son incomparables. Las definiciones que siguen formalizan estas ideas.

Definición 3.34 (Derrotador Propio) [Cn01, Gar00]
Sean $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ dos estructuras de argumento. La estructura $\langle \mathcal{A}_1, h_1 \rangle$ es un *derrotador propio* para $\langle \mathcal{A}_2, h_2 \rangle$ en el literal h, si y sólo si existe un subargumento $\langle \mathcal{A}, h \rangle$ de $\langle \mathcal{A}_2, h_2 \rangle$, tal que $\langle \mathcal{A}_1, h_1 \rangle$ contraargumenta a $\langle \mathcal{A}_2, h_2 \rangle$ en el literal h, y $\langle \mathcal{A}_1, h_1 \rangle$ es estrictamente mejor que $\langle \mathcal{A}, h \rangle$ con respecto al criterio de comparación que se utilice ($\langle \mathcal{A}_1, h_1 \rangle \succ_P \langle \mathcal{A}, h \rangle$). ■

Ejemplo 3.9 (¿Tweety vuela?)
Consideremos un ejemplo clásico de la Inteligencia Artificial:

$$\Pi = \left\{ \begin{array}{l} \texttt{avestruz(tweety).} \\ \texttt{ave(X)} \leftarrow \texttt{ avestruz(X).} \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} \sim \texttt{ vuela (X)} \dashrightarrow \texttt{avestruz(X).} \\ \texttt{vuela(X)} \dashrightarrow \texttt{ave(X).} \end{array} \right\}$$



En la figura hemos representado con flechas anchas a las reglas estrictas y con flechas finas las reglas rebatibles.

Los siguientes son argumentos contradictorios:

$$\mathcal{A}_1 = \{vuela(tweety) \prec ave(tweety)\}$$

$$\mathcal{A}_2 = \{\sim vuela(tweety) \prec avestruz(tweety)\}$$

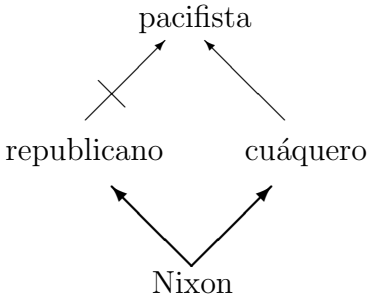
La información de que Tweety es un avestruz es más específica que el hecho de que Tweety es un ave y, por lo tanto, bajo este criterio de preferencia el argumento que sustenta $\sim$ `vuela(tweety)` es preferible a $\mathcal{A}_1$. Luego, $\mathcal{A}_2$ es un derrotador propio de $\mathcal{A}_1$. ■

Definición 3.35 (Derrotador de bloqueo) [Cn01, Gar00]
Sean $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ dos estructuras de argumento. La estructura $\langle \mathcal{A}_1, h_1 \rangle$ es un *derrotador de bloqueo* para $\langle \mathcal{A}_2, h_2 \rangle$ en el literal h, si y sólo si existe un subargumento $\langle \mathcal{A}, h \rangle$ de $\langle \mathcal{A}_2, h_2 \rangle$, tal que $\langle \mathcal{A}_1, h_1 \rangle$ contraargumenta a $\langle \mathcal{A}_2, h_2 \rangle$ en el literal h, y $\langle \mathcal{A}, h \rangle \asymp \langle \mathcal{A}_1, h_1 \rangle$. ■

Ejemplo 3.10 (Diamante de Nixon) [R. Reiter]
Consideremos otro ejemplo clásico de la Inteligencia Artificial:

$$\Pi = \left\{ \begin{array}{l} \texttt{cuáquero(nixon).} \\ \texttt{republicano(nixon).} \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} \sim \texttt{pacifista(X)} \dashrightarrow \texttt{republicano(X).} \\ \texttt{pacifista(X)} \dashrightarrow \texttt{cuáquero(nixon).} \end{array} \right\}$$



Al analizar si Nixon es pacifista debemos distinguir dos argumentos contradictorios:

$$\mathcal{A}_1 = \{\sim pacifista(X) \prec republicano(X)\}$$

$$\mathcal{A}_2 = \{pacifista(X) \prec cuáquero(X)\}$$

Utilizando como criterio de preferencia entre argumentos la especificidad, no podremos determinar cuál de los argumentos es preferible, i.e., $\mathcal{A}_1 \asymp \mathcal{A}_2$. Luego $\mathcal{A}_1$ es un derrotador por bloqueo de $\mathcal{A}_2$. ■

Definición 3.36 (Derrotador) [SCnG94]
Sean $\langle \mathcal{A}_1, h_1 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ dos estructuras de argumento. La estructura $\langle \mathcal{A}_1, h_1 \rangle$ es un *derrotador* para $\langle \mathcal{A}_2, h_2 \rangle$ en el literal h si,

(a) $\langle \mathcal{A}_1, h_1 \rangle$ es un derrotador propio de $\langle \mathcal{A}_2, h_2 \rangle$.

(b) $\langle \mathcal{A}_1, h_1 \rangle$ es un derrotador de bloqueo para $\langle \mathcal{A}_2, h_2 \rangle$.

■

La idea de derrota nos permite definir el análisis dialéctico, que formaliza la interacción entre argumentos contradictorios. Inicialmente, se presenta un argumento a favor de un literal y luego, se consideran todos los contraargumentos que lo derroten. Ya que los contraargumentos son argumentos, se deberá considerar los contraargumentos que derroten a los argumentos predecesores y así sucesivamente. En otras palabras, en forma alternada, se suceden argumentos a favor y en contra de ese literal. Cada línea en el análisis dialéctico es una línea de argumentación en la que, los argumentos a favor serán denominados *argumentos de soporte* y los argumentos en contra *argumentos de interferencia*.

Definición 3.37 (Línea de Argumentación) [Gar00]
Sean $\mathcal{P}$ un *plr* y $\langle \mathcal{A}_0, h_0 \rangle$ una estructura de argumento obtenida a partir de $\mathcal{P}$. Una *línea de argumentación* a partir de $\langle \mathcal{A}_0, h_0 \rangle$, es una secuencia de estructuras de argumentos obtenidas a partir de $\mathcal{P}$, denotada $\Lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \langle \mathcal{A}_3, h_3 \rangle, \ldots]$, donde para cada elemento de la secuencia $\langle \mathcal{A}_i, h_i \rangle$, su sucesor $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ es un derrotador para $\langle \mathcal{A}_i, h_i \rangle$. ■

Definición 3.38 (Argumentos de soporte y de interferencia) [Gar00]
Sean $\mathcal{P}$ un *plr* y $\Lambda = [\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \langle \mathcal{A}_3, h_3 \rangle, \ldots]$ una línea de argumentación. El conjunto de *argumentos de soporte* está formado por los elementos de posiciones pares de la secuencia $\Lambda_S = \{\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \langle \mathcal{A}_4, h_4 \rangle, \ldots]$, mientras que el conjunto de *argumentos de interferencia* está formado por los elementos de posiciones impares $\Lambda_I = [\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_3, h_3 \rangle, \langle \mathcal{A}_5, h_5 \rangle, \ldots]$. ■

Una línea de argumentación puede ser pensada como un posible debate sobre el literal inicial. Lamentablemente, en un debate es posible utilizar argumentos que, aunque parecen válidos, contienen fallas lógicas. Estos argumentos se denominan falacias. Existen diferentes tipos de de argumentos falaces. Aquellos relacionados a aspectos subjetivos del debate (*falacias de relevancia*) y aquellos relacionados con la estructura y el flujo de la discusión. En el sistema en estudio pueden encontrarse del segundo grupo de falacias. Entre las que sufre [MG99] se encuentran las falacia *petitio principii* que tiene que ver con la argumentación circular (se pretende justificar la conclución con la conclusión misma) y las *argumentum non sequitur* en su versión *stolen concept*: usar un concepto para atacar un concepto del cual depende lógicamente. Un análisis detallado de las diferentes falacias y el efecto de ellas en diferentes sistemas argumentativos se pueden encontrar en [MG99].

Por otra parte, existen otras situaciones indeseables como:

- los argumentos que se auto-derrotan: en el contexto de la P.L.R., esta situación no se puede presentar, ya que viola la definición de argumento.

- los derrotadores recíprocos: la noción de contra-argumentación permite la derrota mútua entre dos argumentos.

- los argumentos flotantes[MS91]: se produce cuando dados tres argumentos $\langle \mathcal{A}_1, h_1 \rangle$, $\langle \mathcal{A}_2, h_2 \rangle$ y $\langle \mathcal{A}_3, h_3 \rangle$, $\langle \mathcal{A}_1, h_1 \rangle$ es derrotado tanto por $\langle \mathcal{A}_2, h_2 \rangle$ como por $\langle \mathcal{A}_3, h_3 \rangle$ y $\langle \mathcal{A}_2, h_2 \rangle$ derrota a $\langle \mathcal{A}_3, h_3 \rangle$ y $\langle \mathcal{A}_3, h_3 \rangle$ derrota a $\langle \mathcal{A}_2, h_2 \rangle$. Luego existen dos dos líneas de argumentación posibles:

  $$\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle, \langle \mathcal{A}_3, h_3 \rangle$$

  y

  $$\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_3, h_3 \rangle, \langle \mathcal{A}_2, h_2 \rangle$$

  Luego tanto $\langle \mathcal{A}_2, h_2 \rangle$ como $\langle \mathcal{A}_3, h_3 \rangle$ atacan y defienden a $\langle \mathcal{A}_1, h_1 \rangle$.

Una discusión ampliada de estos problemas, junto con las consecuencias en el sistema de razonamiento pueden encontrarse en [Gar00].

Con el objeto de evitar estos inconvenientes es que se deberá restringir los debates a aquellos correctos en cuanto a su lógica. En otras palabras, reduciremos las posibles líneas de argumentación a aquellas que representen situaciones de debate deseadas.

Definición 3.39 (Concordancia) [Gar00]
Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$. Un conjunto de estructuras de argumento $\left\{ \langle \mathcal{A}_i, h_i \rangle \right\}_{i=1}^{n}$ se dice concordante, si el conjunto $\Pi \cup \bigcup_{i=1}^{n} \mathcal{A}_i$, no es contradictorio. ■

Definición 3.40 (Línea de Argumentación Aceptable) [Gar00]
Una línea de argumentación $\Lambda = [\langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_i, h_i \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ será *aceptable* si:

1. Λ es una secuencia finita.

2. El conjunto Λ_S, de estructuras de argumentos de soporte de Λ, es concordante, y el conjunto Λ_I, de estructuras de argumentos de interferencia de Λ, también es concordante.

3. Ningún argumento $\langle \mathcal{A}_k, h_k \rangle$, $1 \leq k \leq n$ de Λ es un sub-argumento de una estructura de argumento $\langle \mathcal{A}_j, h_j \rangle, 1 \leq j < n$ que aparece previamente en Λ $(j < k)$.

4. Para toda estructura $\langle \mathcal{A}_i, h_i \rangle$ de 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 $\langle \mathcal{A}_{i+1}, h_{i+1} \rangle$ existe, entonces es un derrotador propio de $\langle \mathcal{A}_i, h_i \rangle$.

■

Podemos hacer una analogía entre una línea de argumentación aceptable y un debate correcto entre dos personas que argumentan y contraargumentan sobre algún tema. Claro que, seguramente, no es el único debate que se puede generar sobre ese tema. Cada debate diferente que comience con el mismo argumento de soporte del tema en discusión, generará una línea de argumentación diferente. Para poder decidir si un agente cree en un literal, deberemos tener en cuenta todos los debates posibles. Las diferentes líneas de argumentación que los representan, se agrupan con cierta estructura formando un árbol de dialéctica.

Definición 3.41 (Árbol de dialéctica) [Gar00]
Sea $\langle \mathcal{A}_0, h_0 \rangle$ una estructura de argumento obtenida a partir de un programa lógico $\mathcal{P}$. Un *árbol de dialéctica* para $\langle \mathcal{A}_0, h_0 \rangle$, a partir de $\mathcal{P}$, se denota $\mathcal{T}_{\langle \mathcal{A}_0, h_0 \rangle}$, y se construye de la siguiente forma:

1. La raíz del árbol es etiquetada con $\langle \mathcal{A}_0, h_0 \rangle$.

2. Sea N un nodo interno del árbol etiquetado $\langle \mathcal{A}_n, h_n \rangle$, y $[\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle]$ la secuencia de etiquetas del camino que va desde la raíz hasta el nodo N.
   Sean $\langle \mathcal{B}_1, h_1 \rangle, \langle \mathcal{B}_2, h_2 \rangle, \ldots, \langle \mathcal{B}_k, h_k \rangle$ todos los derrotadores de $\langle \mathcal{A}_n, h_n \rangle$.
   Para cada derrotador $\langle \mathcal{B}_i, h_i \rangle,\ 1 \leq i \leq k$, tal que, la línea de argumentación

   $$[\langle \mathcal{A}_0, h_0 \rangle, \langle \mathcal{A}_1, h_1 \rangle, \ldots, \langle \mathcal{A}_n, h_n \rangle, \langle \mathcal{B}_i, h_i \rangle]$$

   sea aceptable, existe un nodo hijo N_i de N etiquetado con $\langle \mathcal{B}_i, h_i \rangle$. Llamamos *vecinos del nodo* $\langle \mathcal{A}_n, h_n \rangle$ a todos los nodos N_i.
   Si no existe ningún derrotador $\langle \mathcal{B}_i, h_i \rangle$, en tales condiciones, entonces el nodo N es una hoja.

■

Ejemplo 3.11 Consideremos el siguiente *plr*:

$$\Pi = \left\{ \begin{array}{l} b \leftarrow c \\ d. \\ f. \\ e. \\ h. \end{array} \right\} \qquad \Delta = \left\{ \begin{array}{l} a \prec b \\ c \prec d \\ \sim a \prec d \\ a \prec f \wedge d \\ \sim b \prec d \\ g \prec h \\ a \prec g \\ \sim a \prec h \\ \sim b \prec d \wedge e \\ b \prec d \wedge e \wedge f \end{array} \right\}$$

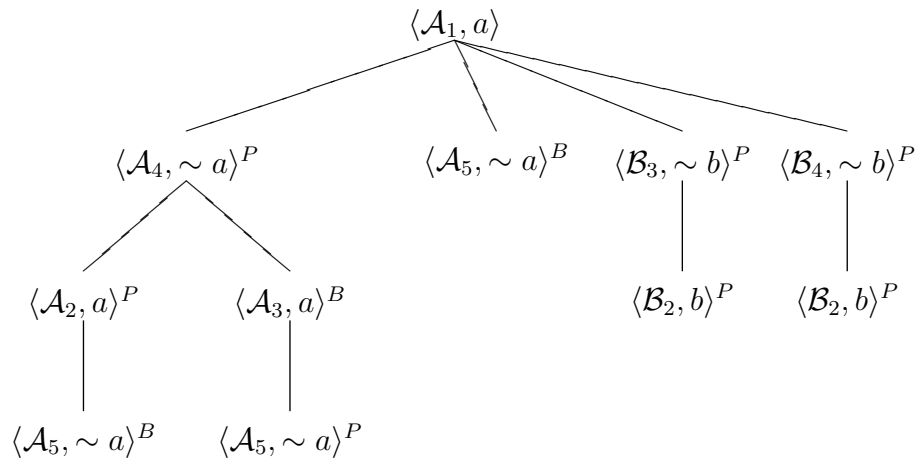


Figura 3.3: Árbol dialéctico para la estructura de argumento $\langle \mathcal{A}_1, a\rangle$ del ejemplo 3.11. Las etiquetas de las estructuras de argumento indican si es un derrotador propio o por bloqueo.

Las estructuras de argumento a favor y en contra de a son:

$\langle \mathcal{A}_1, a\rangle = \langle a -\!\prec b\ ;\ c -\!\prec d, a\rangle$ $\quad$ $\langle \mathcal{A}_4, \sim a\rangle = \langle \sim a -\!\prec d, \sim a\rangle$
$\langle \mathcal{A}_2, a\rangle = \langle a -\!\prec d \wedge f, a\rangle$ $\quad$ $\langle \mathcal{A}_5, \sim a\rangle = \langle \sim a -\!\prec h, \sim a\rangle$
$\langle \mathcal{A}_3, a\rangle = \langle a -\!\prec g\ ; g -\!\prec h, a\rangle$

Los argumentos a favor y en contra de b son:

$\langle \mathcal{B}_1, b\rangle = \langle c -\!\prec d, b\rangle$ $\quad$ $\langle \mathcal{B}_3, \sim b\rangle = \langle \sim b -\!\prec d, \sim b\rangle$
$\langle \mathcal{B}_2, b\rangle = \langle b -\!\prec d \wedge e \wedge f, b\rangle$ $\quad$ $\langle \mathcal{B}_4, \sim b\rangle = \langle \sim b -\!\prec d \wedge e, \sim b\rangle$

En la Figura 3.3 se muestra el árbol de dialéctica para $\langle \mathcal{A}_1, a\rangle$. En la Figura 3.4 se muestra el árbol de dialéctica para $\langle \mathcal{A}_3, a\rangle$. Si bien la definición de árbol dialéctico no lo exige, se han etiquetado los árboles con las etiquetas P y B, que indican si es un derrotador por bloqueo o propio. esto tiene como objeto, clarificar las definiciones dadas hasta ahora. En próximos ejemplos, se omitirá dicha etiqueta.

■

Las creencias de un agente dependen de los árboles de dialéctica que podemos construir a partir de las diferentes estructuras de argumentos. Así, dada una estructura de argumento deberíamos analizar cuantos árboles de dialéctica podremos construir. El siguiente lema indica que existe sólo un árbol de dialéctica por estructura de argumento.

Lema 3.5 *El árbol de dialéctica para una estructura de argumento $\langle \mathcal{A}, h\rangle$ es único, excepto por el orden en que se agregan los nodos vecinos.*

Demostración: Sea $\langle \mathcal{A}, h\rangle$ una estructura de argumento de un programa lógico $\mathcal{P}$. Supongamos que existen dos árboles dialécticos diferentes para tal estructura $\mathcal{T}_{\langle \mathcal{A},h\rangle}$ y $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$. Ya que

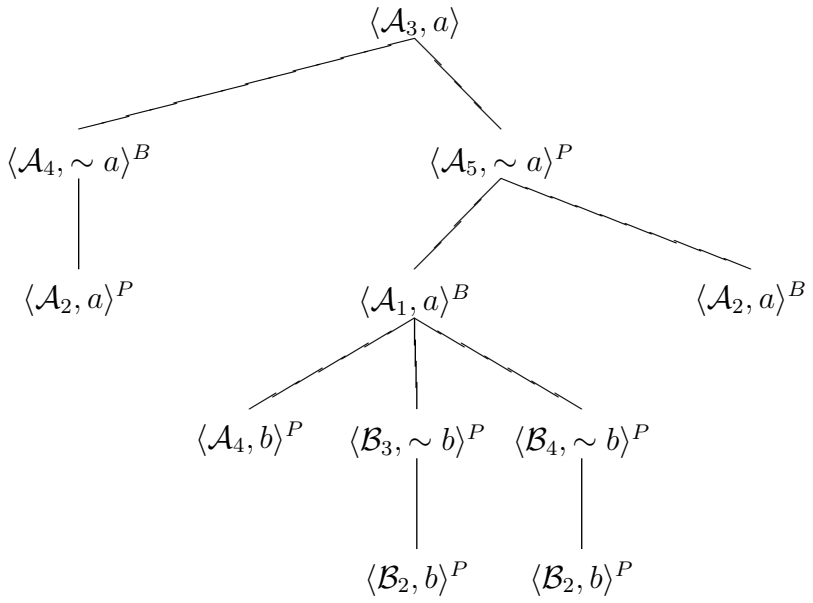


Figura 3.4: Árbol dialéctico para la estructura de argumento $\langle \mathcal{A}_3, a\rangle$ del ejemplo 3.11. Las etiquetas de las estructuras de argumento indican si es un derrotador propio o por bloqueo.

no consideramos árboles diferentes a aquellos que difieren sólo en el orden en que se agregan al árbol los nodos vecinos, debe ser el caso de que existe al menos una línea de argumentación en uno de los árboles, digamos en $\mathcal{T}_{\langle \mathcal{A},h\rangle}$, que no aparece en $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$.
Sea $[\langle \mathcal{A}_0, h_0\rangle, \langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_n, h_n\rangle]$ tal línea de argumentación en $\mathcal{T}_{\langle \mathcal{A},h\rangle}$, que no aparece en $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$. Sea $[\langle \mathcal{A}_0, h_0\rangle, \langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_i, h_i\rangle]$, con $i < n$, la línea de argumentación que pertenece a $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$ con i maximal, en el sentido que $[\langle \mathcal{A}_0, h_0\rangle, \langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_i, h_i\rangle, \langle \mathcal{A}_{i+1}, h_{i+1}\rangle]$ no es una línea de argumentación en $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$.
Ya que $\mathcal{T}_{\langle \mathcal{A},h\rangle}$ es un árbol dialéctico cuyas líneas de argumentación son aceptables, por definición, $\langle \mathcal{A}_{i+1}, h_{i+1}\rangle$ es uno de los derrotadores de $\langle \mathcal{A}_i, h_i\rangle$ que deben ser considerados al construir un árbol dialéctico que incluya entre ramas a $[\langle \mathcal{A}_0, h_0\rangle, \langle \mathcal{A}_1, h_1\rangle, \ldots, \langle \mathcal{A}_i, h_i\rangle]$. Recordemos que por definición se deben considerar todos los derrotadores de un nodo que permitan líneas de argumentación aceptables. Así $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$ no es un árbol dialéctico para $\langle \mathcal{A}, h\rangle$, ya que viola una de las condiciones de la definición. Contradicción.
Esta contradicción provino de suponer que $\mathcal{T}_{\langle \mathcal{A},h\rangle}$ y $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$ eran árboles dialécticos diferentes.
$\therefore$ $\mathcal{T}_{\langle \mathcal{A},h\rangle}$ y $\mathcal{T}'_{\langle \mathcal{A},h\rangle}$ son árboles idénticos o bien difieren sólo en el orden en que fueron insertados los vecinos de los nodos. ■

El lema anterior indica que existe sólo un árbol de dialéctica por estructura de argumento, pero existirá un árbol de dialéctica por cada estructura de argumento diferente de un mismo literal, como se vió en el ejemplo 311.

Una vez que se ha construido el árbol dialéctico para un literal, deberemos analizar si la estructura de argumento de la raíz fue derrotada, analizando el estado de las esructuras de

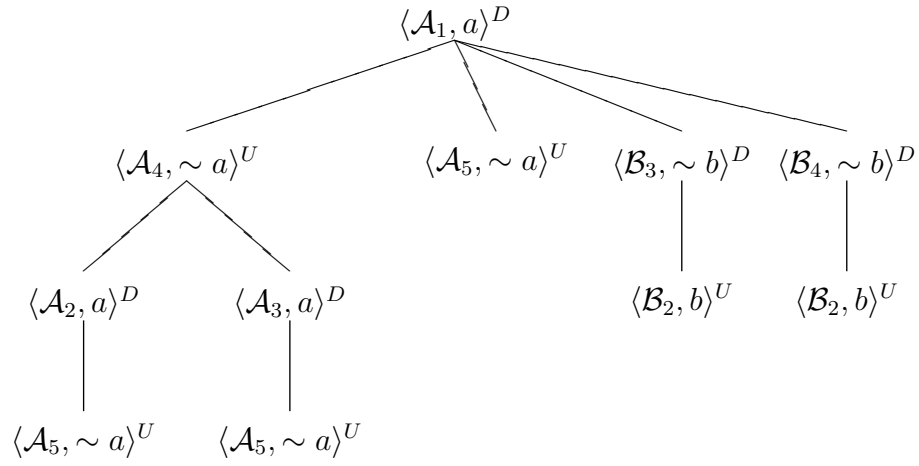


Figura 3.5: Árbol dialéctico marcado para la estructura de argumento $\langle \mathcal{A}_1, a \rangle$ del ejemplo 3.12.

argumentos internas del árbol. La siguiente definición especifica el procedimiento que marca a los nodos de un árbol dialéctico con las etiquetas "U", no derrotado, o "D", derrotado.

Definición 3.42 (Marcado de un árbol de dialéctica) [Gar00]
Sea $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ un árbol de dialéctica de $\langle \mathcal{A}, h \rangle$. Un árbol de dialéctica marcado, denotado $\mathcal{T}^*_{\langle \mathcal{A},h \rangle}$ puede obtenerse marcando cada nodo en $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ de la siguiente forma:

1. Todas las hojas de $\mathcal{T}_{\langle \mathcal{A},h \rangle}$ se marcan con "U" en $\mathcal{T}^*_{\langle \mathcal{A},h \rangle}$.
2. Sea N un nodo interno de $\mathcal{T}_{\langle \mathcal{A},h \rangle}$. El nodo N se marca con "U"si todo nodo hijo de N está marcado con "D", y N se marca con "D"si existe al menos un nodo hijo de N marcado con "U".

■

Ejemplo 3.12 Consideremos nuevamente el ejemplo 3.11. Para el literal a se pueden construir tres árboles dialécticos diferentes. Las Figuras 3.5 y 3.6 muestran dos de los árboles de dialéctica marcados. Nótese que uno de los árboles tiene la raíz etiquetada con "D", lo que indica que la estructura de argumento fue derrotad. Por otro lado, el otro árbol de dialéctica tiene su raíz marcada con "U", i.e., que no ha sido derrotada. ■

Como muestra el ejemplo anterior dado un literal h podemos construir un árbol dialéctico diferente para cada estructura de argumento que exista para h. No todos los árboles dialécticos tienen la misma etiqueta en la raíz. Ya que la semántica operacional está basada en la construcción de árboles no derrotados, deberemos encontrar un árbol dialéctico cuya raíz esté marcado

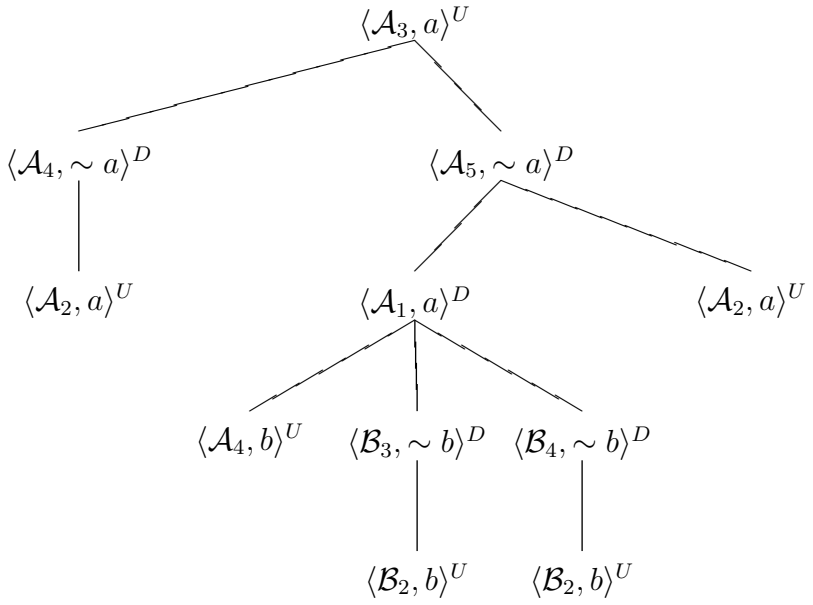


Figura 3.6: Árbol dialéctico marcado para la estructura de argumento $\langle \mathcal{A}_3, a\rangle$ del ejemplo 3.12.

con la etiqueta "U". Así, diremos que un argumento $\langle \mathcal{A}, h\rangle$ está *derrotado*, si la raíz del único árbol dialéctico marcado $\mathcal{T}*_{\langle \mathcal{A},h\rangle}$ está marcada con "D" y diremos que está *garantizado*[Gar00] si la raíz está marcada con "U". A partir de este concepto [Gar00] podremos determinar el conjunto de consecuencias de un *plr*.

Definición 3.43 (Literales garantizados)
Sean $\mathcal{P} = (\Pi, \Delta)$, un *plr*, y h un literal en *Lit*. El literal h está garantizado si existe una estructura de argumento para h, $\langle \mathcal{A}, h\rangle$, tal que la raíz del único árbol de dialéctica asociado a $\langle \mathcal{A}, h\rangle$, $\mathcal{T}^*_{\langle \mathcal{A},h\rangle}$, está marcada con "U". Si tal estructura de argumento no existe, diremos que h no está garantizado. ■

En este punto estamos en condiciones de definir cuáles son las creencias de un agente. Utilizaremos un operador modal de creencia "B" para representar que el agente *cree* en un literal h, y lo notaremos Bh. Diremos que se verifica Bh[Gar00] *si y sólo si* h *es un literal garantizado*. Así, un agente que implemente esta semántica operacional, responderá SI cuando se consulte por un literal h en el lenguaje del programa, tal que Bh. Diremos que no creemos en un literal en el lenguaje, i.e., $\sim Bh$, si y sólo si h no está garantizado. La siguiente definición indica cuáles serán las posibles respuestas de un agente a una consulta.

Definición 3.44 (Respuestas a una consulta) [Gar00]
Sean $\mathcal{P} = (\Pi, \Delta)$, un *plr*, y h un literal que representa una consulta para el programa P. La *respuesta a* h será:

SI: si Bh (cree en h), i.e., h está garantizado.

NO: si $B\overline{h}$ (cree en $\overline{h}$), i.e., $\overline{h}$ está garantizado.

INDECISO: si $\sim Bh$ y $\sim B\overline{h}$ (no cree en h ni en $\overline{h}$), i.e., no está garantizado ni h, ni $\overline{h}$.

DESCONOCIDO: si h no pertenece al lenguaje generado bajo la signatura del programa.

■

Una respuesta INDECISO para un literal $h \in Lit$, indica que no hemos podido decidir entre creer en h o creer en su complemento. Esta respuesta contempla dos casos algo diferentes. Por un lado, tiene en cuenta los casos en que no exista ningún argumento para el literal,i.e., el agente no tiene ninguna información sobre h. Por otro lado, contempla el caso de que existan estructuras de argumentos, pero que al generar sus respectivos árboles de dialéctica, la raíz sea marcada con "D". En otras palabras, el agente tiene información sobre h, pero entre su conocimiento hay información preferible que la derrota. Por último, se ha considerado la respuesta DESCONOCIDO para aquellas consultas no estén en el lenguaje.

Nótese que no se ha implementado la suposición de mundo cerrado. Ante información que no aparece en la base de conocimiento, nuestro agente es muy cauteloso y responde que está indeciso o bien que es desconocido para él.

Ejemplo 3.13 Consideremos nuevamente el ejemplo 3.11. El literal a está garantizado, el árbol de dialéctica marcado para la estructura $\langle \mathcal{A}_3, a \rangle$ tiene su raíz etiquetada con "U". Luego la respuesta para la consulta a es SI. ■

Ejemplo 3.14 Este ejemplo es una adaptación del presentado por Ron Loui. Consideremos el *plr* que contiene la siguiente información sobre cerdos:

> Los clones de cerdos son cerdos alterados genéticamente. Todos los cerdos alterados genéticamente son cerdos. Los cerdos generalmente son gordos y fofos. Los cerdos alterados genéticamente usualmente no son gordos. Los cerdos que no son gordos generalmente no son fofos. Los cerdos que son graciosos tienden a ser gordos. Cualquiera que no sea fofo, generalmente será ágil. Los cerdos usualmente no son ágiles. Bola de Maíz es un cerdo y Trici es un clon de Bola de Maíz.

La representación a través de un *plr* es:

$$\Pi = \left\{ \begin{array}{l} \texttt{cerdo\_alt\_gen(X)} \leftarrow \texttt{clon(X,Y)} \wedge \texttt{cerdo(Y)}. \\ \texttt{cerdo(X)} \leftarrow \texttt{cerdo\_alt\_gen(X)}. \\ \texttt{clon(trici,bola\_de\_maíz)}. \\ \texttt{cerdo(bola\_de\_maíz)}. \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} \texttt{gordo(X)} \dashrightarrow \texttt{gracioso(X)} \wedge \texttt{cerdo(X)} \\ \texttt{gordo(X)} \dashrightarrow \texttt{cerdo(X)} \\ \texttt{fofo(X)} \dashrightarrow \texttt{cerdo(X)} \\ \sim\texttt{gordo(X)} \dashrightarrow \texttt{cerdo\_alt\_gen(X)} \\ \sim\texttt{fofo(X)} \dashrightarrow \texttt{cerdo(X)} \wedge \sim\texttt{gordo(X)} \\ \texttt{ágil(X)} \dashrightarrow \sim\texttt{fofo(X)} \\ \sim\texttt{ágil(X)} \dashrightarrow \texttt{cerdo(X)} \end{array} \right\}$$

La siguiente es una tabla que muestra las respuestas a posibles consultas:

Consulta	Respuesta	Explicación
gordo(bola_de_maíz)	SI	Literal garantizado
fofo(bola_de_maíz)	SI	Literal garantizado
ágil(bola_de_maíz)	NO	∼fofo(trici) está garantizado
gordo(trici)	NO	∼gordo(trici) está garantizado
fofo(trici)	NO	∼fofo(trici) está garantizado
ágil(trici)	INDECISO	No se puede garantizar ni ágil(trici), ni ∼ágil(trici)
gracioso(trici)	INDECISO	No se puede garantizar ni gracioso(trici), ni ∼gracioso(trici)
pesado(trici)	DESCONOCIDO	El símbolo "pesado" no pertenece a la signatura y, por lo tanto, pesado(trici)$\notin$ *Lit*.
gracioso(tweety)	DESCONOCIDO	El símbolo "tweety" no pertenece a la signatura y, por lo tanto, gracioso(tweety)$\notin$ *Lit*.

■

3.5. Conclusiones

En este capítulo hemos presentado a la P.L.R. como un sistema de representación de conocimiento y razonamiento no monotónico. El lenguaje de la P.L.R. permite representar información certera, libre de excepciones y no contradictorio, a través de los hechos y de las reglas estrictas y conocimiento tentativo y potencialmente contradictorio a través de las reglas rebatibles.

Asimismo, hemos introducido una clasificación de la P.L.R. siguiendo la clasificación clásica

de la P.L.. Así de acuerdo a la forma de las reglas, dividimos a los *plr* en definidos, básicos, normales y extendidos. En particular, hemos estudiado a los *plr* básicos, i.e., , aquellos que permiten negación fuerte en el cuerpo y en la cabeza de las reglas.

La semántica operacional de la P.L.R.B. está basada en los sistemas argumentativos. Un agente en forma introspectiva trata de *convencerse* de que una inferencia es correcta bajo su base de conocimiento. Para esto construye argumentos a favor de la conclusión a la que quiere llegar y trata de derrotarlos con contraargumentos, a los que les aplica el mismo procedimiento. De este modo construye un árbol dialéctico, que luego se analiza para determinar si el argumento raíz fue derrotado. En la construcción del árbol se deben tener en cuenta varias restricciones como no contradecirse (i.e., que los argumentos a favor sean consistentes entre ellos y también que los argumentos en contra sean consistentes entre sí) ni incorporar en el debate argumentos que ya fueron dados.

Con el objeto de determinar si un argumento derrota a otro, se tuvo en cuenta que uno ataque alguna conclusión del otro y que además sea mejor según algún criterio de preferencia. En este capítulo y en los que siguen se trabaja sobre una relación de preferencia genérica, aunque algunos casos ejemplificaremos con el criterio de preferencia de especificidad [GS04].

La P.L.R.B. es una herramienta muy atractiva para el desarrollo e implementación de agentes deliberativos. Permite trabajar en forma dinámica con información incompleta y potencialmente contradictoria. Actualmente, varias aplicaciones han sido desarrolladas e implementadas usando sistemas argumentativos relacionadas con, por ejemplo, sistemas multiagentes y búsqueda en web [CM04a, CM04b, BAV04].

Capítulo 4

Semántica declarativa basada en juegos de la P.L.R.B.

Contenido

Todos mis trabajos son juegos. Juegos serios.

M. C. ESCHER.

En este capítulo se introduce una caracterización declarativa de la semántica de la P.L.R.B., una de las principales contribuciones de este trabajo. La semántica declarativa presentada está basada en la noción de juegos, descripta en el capítulo 2 y utiliza a la teoría de modelos

para determinar las consecuencias de un *plr*. En la primer sección, presentamos las motivaciones de esta semántica. Luego se introduce una definición declarativa de los argumentos, mostrando la equivalencia con la forma procedural definida en el capítulo 3. Finalmente, se brinda la caracterización semántica de las consecuencias de un *plrb* a través de modelos.

4.1. Motivación

Históricamente, los sistemas argumentativos han sido estudiados desde un punto de vista procedural, haciendo hincapié en el proceso de argumentación y en la construcción de los argumentos mismos. Estos formalismos son una herramienta poderosa como sistema de representación de conocimiento y razonamiento no monotónico. En particular, lo es la P.L.R., que combina a los sistemas argumentativos y a la P.L.. Por esta razón, algunos autores [Dun95, JV99, VTS08] comenzaron a estudiar a los sistemas argumentativos como teorías formales, analizando su semántica.

Existen dos enfoques a través de los que podemos estudiar una teoría formal[VERAK76]:

- Enfoque Sintáctico: trabaja con la parte formal del lenguaje, abstrayéndose del significado. No solamente trata con la definición de las fórmulas bien formadas, sino también con el estudio de los axiomas, reglas de inferencia y pruebas, lo que constituye la teoría de prueba.
- Enfoque Semántico: trata con las interpretaciones del lenguaje e incluye las nociones de significado, implicación lógica y verdad.

La P.L. como sistema formal ha sido analizada desde los dos enfoques. Por un lado, desde el punto de vista de su teoría de prueba, es decir donde enfatizamos el proceso de deducción. El conjunto de los literales que son consecuencia de este proceso, es la semántica operacional del formalismo. Y por el otro, el enfoque semántico, en donde se enfatiza qué debería ser consecuencia del formalismo y se trabaja a partir de la teoría de modelos.

Los sistemas argumentativos también han sido estudiados desde ambos enfoques, aunque los desarrollos son mayores en cuanto a la teoría de prueba. Desde el punto de vista sintáctico, los argumentos se construyen a partir de las fórmulas de la teoría y la argumentación se analiza como un proceso basado en la teoría de prueba del sistema. La semántica operacional es definida en función de los argumentos que son deducidos en ese sistema. Desde el punto de vista semántico, se ha caracterizado el conjunto de argumentos aceptados, utilizando una relación de ataque entre los argumentos *genérica*. Asimismo, pocas semánticas hacen referencia a la forma declarativa del argumento.

Si bien la semántica operacional puede considerarse suficiente para verificar si una consulta a un programa lógico rebatible está justificada o no, creemos que el beneficio de una semántica declarativa para la P.L.R. es doble. Por un lado, es necesaria para ayudar al programador a especificar el conocimiento y razonar a partir de él sin preocuparse por la parte de control del sistema. Por otro lado, la definición de una semántica declarativa ayuda en el estudio de la P.L.R. como sistema de razonamiento no monótono y a probar propiedades sobre el sistema.

Esto motivó la búsqueda de una semántica declarativa que satisfaga las propiedades de sensatez y completitud con respecto a la teoría de prueba de la P.L.R.B.. En este sentido se observó una similitud entre un juego y lo que un agente realiza con el árbol dialéctico para una consulta y su argumento a favor. Un agente realiza un juego en forma introspectiva, simulando dos jugadores que realizaran cada uno sus mejores jugadas para defender u oponerse a un argumento dado. Si ese juego es ganado por el argumento inicial, i.e., no fue derrotado por ningún contraargumento, el agente puede afirmar la conclusión sustentada por ese argumento inicial.

Esta similitud de los sistemas argumentativos con los juegos ha sido estudiada por otros autores (ver capítulo 5) pero desde el punto de vista sintáctico o bien para teorías de prueba que no son equivalentes a la de la P.L.R.B..

La semántica desarrollada en este trabajo, que se denomina $\mathcal{GS}$, es trivaluada y está basada en los componentes de los juegos. Los valores de verdad que permite son tres, VERDADERO,FALSO e INDECISO, a diferencia de la P.L.R.B. que permite cuatro. Esta diferencia está sustentada en que $\mathcal{GS}$ sólo trabaja sobre literales generados bajo la signatura del programa, en oposición a la teoría de prueba que en caso de no pertenecer al lenguaje del programa da como respuesta DESCONOCIDO.

El núcleo de esta semántica es la estructura de argumento, para el que hemos dado una definición semántica heredada de la P.L.. A partir de este concepto se desarrolló $\mathcal{GS}$, como una semántica en donde cada elemento de la teoría de prueba es relacionada con un elemento en el juego. Cada árbol dialéctico es equivalente a un juego. La semántica declarativa trivaluada basada en juegos está definida a partir de las interpretaciones que están justificadas por alguno de los juegos de la familia de juegos, que sea ganado por el participante que lo inicia.

El estudio de ciertas propiedades, como la complejidad temporal, es una de las principales motivaciones para obtener una semántica declarativa del sistema, con fundamento teórico de la P.L. clásica y de los juegos, que sea sensata y completa con respecto a su teoría de prueba.

4.2. Estructura de Argumento: un estudio declarativo

Dado un programa, estamos interesados en obtener el conjunto de sus consecuencias lógicas, i.e., su semántica. Para esto necesitamos dar una definición declarativa equivalente a la definición operacional de argumento, ya que es la unidad básica del razonamiento rebatible. La mayoría de los frameworks de sistemas argumentativos especifican la semántica declarativa dejando indefinida la noción de argumento. En otras palabras, se asume que el conjunto de argumentos para un literal es dado por algún oráculo. Esto motivó una caracterización declarativa de la definición procedural de argumento. El estudio presentado está basado en un concepto introducido, en primer instancia por Tarski para la lógica clásica y luego adaptado por Lifschitz para programas lógicos básicos.

Una operación de consecuencia Cn_T en el sentido estándar deriva de Tarski y es una operación sobre conjuntos de sentencias. Dado un conjunto de sentencias Γ se define $Cn_T(\Gamma)$ como el conjunto de sentencias que puede ser derivado clásicamente a partir de Γ y de los axiomas lógicos usando las reglas de inferencia. Esta operación satisface tres condiciones[Mak89]:

***Inclusión*:** $\Gamma \subseteq Cn_T(\Gamma)$

***Idempotencia*:** $Cn_T(\Gamma) = Cn_T(Cn_T(\Gamma))$

***Monotonía*:** si $\Gamma \subseteq \Upsilon$ entonces $Cn_T(\Gamma) \subseteq Cn_T(\Upsilon)$

Estas propiedades son heredadas de la lógica clásica. Bajo el contexto de la P.L.R., la clausura lógica se realizará sobre las reglas del programa y $\Gamma = \Theta$, i.e., Γ es el conjunto de hechos. Siendo que nuestro objetivo es capturar la noción de argumento, que como se analizó en el capítulo 3 es monótono, exigiremos que las consecuencias rigurosas cumplan las tres propiedades.

Como paso intermedio deberemos definir el conjunto de *literales fijos* que son consecuencias de los programas lógicos. Así la operación de consecuencia se define sobre un programa lógico rebatible en Lit. Es importante recordar que las consecuencias de un programa $\mathcal{P}$ (esquema o fijo) están siempre definidas a partir de $\mathbf{fijo}(\mathcal{P})$ y son siempre un subconjunto de Lit.

Dado un programa lógico rebatible exigiremos que el conjunto de las consecuencias de un programa sea "cerrado" bajo las reglas estrictas y rebatibles, i.e., si una regla puede ser aplicada, entonces la consecuencia de esa regla debe pertenecer al conjunto de las consecuencias del programa. Extenderemos las definiciones dadas en [Lif96] con el objeto de manejar *plrb*.

Definición 4.1 (Conjunto de literales cerrado rigurosamente)
Sea X un conjunto de literales fijos. X es *cerrado rigurosamente bajo un programa* $\mathcal{P}$, si para

cada regla estricta $Cabeza \leftarrow Cuerpo$ y rebatible $Cabeza' \prec Cuerpo'$ de $\mathcal{P}$, vale que $Cabeza \in X$ siempre que $Cuerpo \subseteq X$ y que $Cabeza' \in X$ siempre que $Cuerpo' \subseteq X$. ■

Así, cada literal que esté en el conjunto de consecuencias, debería estar en la cabeza de una regla. Esto tiene su motivación en el hecho de que no estamos interesados en que un agente crea literales que no puede explicar a partir de sus conocimientos, aún cuando ese literal no contradiga sus creencias.

La definición anterior no refleja el comportamiento de las reglas rebatibles, ya que es aplicada, sin considerar posibles elementos en el conocimiento que invaliden su uso mostrando, finalmente, una conducta idéntica a la regla estricta.

Deberemos distinguir dos casos de acuerdo a si el programa es consistente o inconsistente. En caso de que el programa sea inconsistente, la clausura de sus consecuencias debería reflejar esta situación. Heredado de la lógica clásica, la clausura de un conjunto inconsistente será el lenguaje completo. La siguiente definición formaliza esta idea.

Definición 4.2 (Conjunto de literales consistente. Conjunto de literales inconsistente)

Sea X un conjunto de literales fijos. X es *consistente* si no existe un literal L tal que $\{L, \overline{L}\} \subseteq X$. De otro modo, diremos que X es *inconsistente*. ■

Definición 4.3 (Conjunto de literales lógicamente cerrado)
X es *lógicamente cerrado* si es consistente o es igual a Lit. ■

A partir de los conceptos anteriores definiremos el conjunto de consecuencias rigurosas de un programa lógico rebatible.

Definición 4.4 (Consecuencias rigurosas de un conjunto de reglas)
Sea Ψ un conjunto de reglas estrictas y rebatibles. El *conjunto de las consecuencias rigurosas* bajo las reglas Ψ, que notaremos $Cn_R(\Psi)$, es el menor conjunto de literales, con respecto a la inclusión, que es lógicamente cerrado y cerrado rigurosamente bajo Ψ. ■

Dado un conjunto de reglas estrictas y rebatibles Ψ, demostraremos que $Cn_R(\Psi)$ está bien definida. Con este fin, definiremos un reticulado formado por los conjuntos lógicamente cerrado y cerrado rigurosamente bajo Ψ que son parte de Lit y demostraremos un conjunto de propiedades intermedias.

Definición 4.5 (Reticulado de los conjuntos lógicamente cerrado y cerrado rigurosamente bajo un conjunto de reglas)
Sea Ψ un conjunto de reglas estrictas y rebatibles y Lit el lenguaje asociado. El conjunto parcialmente ordenado bajo la relación de inclusión, de todos los conjuntos lógicamente cerrado y cerrado rigurosamente bajo Ψ, que notaremos $\mathcal{P}_\Psi(Lit)$, es un reticulado con operaciones ínfimo y supremo que se definen como sigue:

- el ínfimo de X e Y es la intersección de los elementos, i.e., $X \cap Y$;
- el supremo de X e Y es el conjunto generado por la unión de X e Y lógicamente cerrado y cerrado rigurosamente bajo Ψ, que notaremos $X \cup_G Y$. Formalmente,

$$X \cup_G Y = \bigcap_{Z \in \mathcal{P}_\Psi(Lit): X \cup Y \subseteq Z} Z$$

  Nótese que el conjunto generado por otro es el menor conjunto con respecto a la inclusión que contiene al que lo genera y que cumple la condición deseada.

■

Lema 4.1 *La intersección de dos miembros de $\mathcal{P}_\Psi(Lit)$ es un miembro de $\mathcal{P}_\Psi(Lit)$.*

Demostración: Sean X e Y dos miembros de $\mathcal{P}_\Psi(Lit)$. Supongamos que $X \cap Y \notin \mathcal{P}_\Psi(Lit)$. Luego existe una regla de Ψ cuyo cuerpo $\{l_1, \ldots, l_n\} \subseteq X \cap Y$ y cuya cabeza $l_0 \notin X \cap Y$. Siendo que l_0 no está en la intersección de X e Y, entonces l_0 no pertenece a alguno de los dos conjuntos o a ninguno. Supongamos que no pertenece a X. Luego $\{l_1, \ldots, l_n\} \subseteq X$ y $l_0 \notin X$. Contradicción, pues X es un miembro de $\mathcal{P}_\Psi(Lit)$. Esta contradicción provino de suponer que la intersección de X e Y no es miembro de $\mathcal{P}_\Psi(Lit)$.
Análogamente, se prueba si l_0 no pertenece a ninguno de los dos conjuntos.
$\therefore X \cap Y \in \mathcal{P}_\Psi(Lit)$ ■

Lema 4.2 *La intersección de un conjunto numerable de miembros de $\mathcal{P}_\Psi(Lit)$ es un miembro de $\mathcal{P}_\Psi(Lit)$.*

Demostración: Demostraremos por inducción sobre el número n de elementos a intersectar en el conjunto.

Caso Base (n=2): Lema 4.1.

Paso Inductivo : Supongamos que el lema vale para la intersección de k miembros de $\mathcal{P}_\Psi(Lit)$. Demostraremos que el lema vale para $k+1$ miembros de $\mathcal{P}_\Psi(Lit)$. Sea $C = \bigcap_{i=1}^{k} X_i$, siendo $X_i \in \mathcal{P}_\Psi(Lit)$. Por hipótesis inductiva $C \in \mathcal{P}_\Psi(Lit)$. Por lema 4.1, la intersección de C y X_{k+1} es un miembro de $\mathcal{P}_\Psi(Lit)$.

■

Lema 4.3 *Sean X e Y elementos de $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$. $X \cap Y$ es la mayor de las cotas inferiores.*

Demostración: $X \cap Y$ es una cota inferior de X y de Y bajo la relación de inclusión.
Supongamos que existe otra cota inferior de X e Y, $Z \in \mathcal{P}_\Psi(Lit)$, debemos probar que $Z \subseteq X \cap Y$. Como Z es cota inferior, entonces $Z \subseteq X$ y $Z \subseteq Y$. Así

$$Z \subseteq X \cap Y \tag{4.1}$$

Por lo tanto, $X \cap Y$ es la mayor de las cotas inferiores, luego es el ínfimo. ■

Lema 4.4 *Sean X e Y elementos del reticulado $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$. $X \cup_G Y$ es la menor de las cotas superiores.*

Demostración: Por definición de unión generada por X e Y, $X \cup Y \subseteq X \cup_G Y$. Así $X \cup_G Y$ es una cota superior de X y de Y bajo la relación de inclusión.
Supongamos que existe otra cota superior de X e Y, $Z \in \mathcal{P}_\Psi(Lit)$ debemos probar que $X \cup_G Y \subseteq Z$. Como Z es cota superior, entonces $X \subseteq Z$ e $Y \subseteq Z$. Luego $X \cup Y \subseteq Z$. Ya que la unión generada es la intersección de todos los elementos del conjunto que contengan a $X \cup Y$, entre ellos Z,

$$X \cup_G Y \subseteq Z \tag{4.2}$$

Por lo tanto, $X \cup_G Y$ es la menor de las cotas superiores, luego es el supremo. ■

Lema 4.5 *El supremo y el ínfimo de dos elementos de $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$ siempre existen y pertenecen al reticulado.*

Demostración: Sean X e Y dos miembros de $\mathcal{P}_\Psi(Lit)$.

- $X \cap Y \in \mathcal{P}_\Psi(Lit)$: Por lema 4.1 y por lema 4.3 la intersección es la mayor de las cotas superiores, ie el infimo de los elementos.
- $X \cup_G Y \in \mathcal{P}_\Psi(Lit)$: Por lema 4.2 la unión generada es un miembro de $\mathcal{P}_\Psi(Lit)$ y por lema 4.4 es el supremo de los elementos.

$\therefore$ dados dos elementos cualesquiera de $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$ su supremo y su ínfimo existen en $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$.
Por lo tanto, $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$ es un reticulado. ■

Lema 4.6 *El último elemento del reticulado* $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$ *es* Lit.

Demostración: Deberemos mostrar que:

1. $Lit \in \mathcal{P}_\Psi(Lit)$: Lit es lógicamente cerrado y por ser el lenguaje asociado a Ψ contiene a todos los literales que pueden ser generados por la signatura y por lo tanto, es cerrado bajo Ψ.
2. Para todo elemento $X \in \mathcal{P}_\Psi(Lit)$ vale que $X \subseteq Lit$. Trivial, ya que Lit es el lenguaje.

■

Lema 4.7 *El primer elemento del reticulado* $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$ *es* $\bigcap_{X \in \mathcal{P}_\Psi(Lit)} X$.

Demostración: Deberemos mostrar que:

1. $\bigcap_{X \in \mathcal{P}_\Psi(Lit)} X \in \mathcal{P}_\Psi(Lit)$: Demostrado por lema 4.2.
2. Para todo elemento $Y \in \mathcal{P}_\Psi(Lit)$ vale que

$$\bigcap_{X \in \mathcal{P}_\Psi(Lit)} X \subseteq Y$$

   Trivial.

■

Observación 4.1 *En el caso de que el conjunto de reglas no contenga hechos, el primer elemento del reticulado será el conjunto vacío.*

Lema 4.8 $Cn_R(\Psi)$ *es el primer elemento del reticulado* $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$.

Demostración: El primer elemento del reticulado es el conjunto minimal con respecto a la inclusión de conjuntos que es lógicamente cerrado y cerrado bajo Ψ, ya que es la intersección de todos ellos. Así cumple la definición de $Cn_R(\Psi)$ ■

Lema 4.9 *$Cn_R(\Psi)$ está bien definida.*

Demostración: Por lema 4.8, $Cn_R(\Psi)$ siempre existe y es el primer elemento de $\langle \mathcal{P}_\Psi(Lit), \cap, \cup_G \rangle$. ■

A partir de la noción de $Cn_R(\Psi)$ definiremos el concepto de consecuencias rigurosas de un programa.

Definición 4.6 (Consecuencias rigurosas de un programa)
Sea $\mathcal{P}$ un *plrb*. El *conjunto de las consecuencias rigurosas* de $\mathcal{P}$ lo definimos como $Cn_R(\mathcal{P})$. ■

Ejemplo 4.1 Consideremos el ejemplo 3.14. El conjunto de las consecuencias rigurosas de este programa son:

$$Cn_R(\mathcal{P}) = Lit$$

ya que para cumplir la condición de que sea cerrado rigurosamente deberemos incluir `gordo(trici)` y $\sim$`gordo(trici)`, y en consecuencia, por la definición de lógicamente cerrado, se genera el lenguaje completo. ■

Lema 4.10 *Si X es un conjunto consistente de hechos, entonces $Cn_R(X) = X$.*

Demostración: $X \subseteq Cn_R(X)$ Por definición de cerrado rigurosamente, el conjunto literales que son cabeza de hechos pertenecen a Cn_R. Así $X \subseteq Cn_R(X)$.

$Cn_R(X) \subseteq X$ Supongamos que existe $x \in Cn_R(X)$ tal que $x \notin X$, entonces

(a) x es la cabeza de una regla que está en X y no es un hecho. Contradición pues todos los elementos de X son hechos.

(b) x fue generado porque $Cn_R(X)$ es insatisfacible. Contradicción porque X es consistente.

La contradicción provino de suponer que $Cn_R(X) \not\subseteq X$.

Por lo tanto $Cn_R(X) = X$ ■

Lema 4.11 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb y $\Theta \subseteq \Pi$ el conjunto de hechos entonces:*

(I) $X \subseteq Cn_R(\mathcal{P})$, *para todo* $X \subseteq \Theta$

(II) $Cn_R(\mathcal{P}) = Cn_R(Cn_R(\mathcal{P}))$

(III) Si $\mathcal{P}' \subseteq \mathcal{P}$ entonces $Cn_R(\mathcal{P}') \subseteq Cn_R(\mathcal{P})$

Demostración: (I) $X \subseteq Cn_R(\mathcal{P})$ para todo $X \subseteq \Theta$

Sea $x \in X$. Por hipótesis, $x \in \Theta$. Luego x es la cabeza de una regla estricta con cuerpo vacío (hecho). Por definición de cerrado rigurosamente, tenemos que el cuerpo de dicha regla pertenece a X y, por lo tanto, $x \in Cn_R(\mathcal{P})$.

(II) $Cn_R(\mathcal{P}) = Cn_R(Cn_R(\mathcal{P}))$

$Cn_R(\mathcal{P})$ es un conjunto de literales fijos, que asumiremos hechos al calcular sus consecuencias. Distinguiremos dos casos:

- $Cn_R(\mathcal{P}) = Lit$, en cuyo caso, trivialmente, $Cn_R(\mathcal{P}) = Cn_R(Cn_R(\mathcal{P}))$
- $Cn_R(\mathcal{P})$ es un conjunto de literales fijos consistente. Por lema 4.10 $Cn_R(\mathcal{P}) = Cn_R(Cn_R(\mathcal{P}))$

(III) Si $\mathcal{P}' \subseteq \mathcal{P}$ entonces $Cn_R(\mathcal{P}') \subseteq Cn_R(\mathcal{P})$

Si $\mathcal{P}'$ es inconsistente, entonces también lo será $\mathcal{P}$ y por lo tanto, $Cn_R(\mathcal{P}') = Cn_R(\mathcal{P})$. Si $\mathcal{P}'$ es consistente y $\mathcal{P}$ es inconsistente, trivialmente se cumple el lema.

Supongamos que $\mathcal{P}'$ es consistente y que existe $x \in Cn_R(\mathcal{P}')$ y $x \notin Cn_R(\mathcal{P})$. Como por definición de consecuencia, el conjunto es el menor que cumple las condiciones, x debe haber sido generado a partir de una regla R cuya cabeza sea x y cuyo cuerpo pertenezca a $Cn_R(\mathcal{P}')$. Si tal regla es un hecho, entonces la misma regla pertenece a $\mathcal{P}$ y, por lo tanto, $x \in Cn_R(\mathcal{P})$.

De otro modo, se puede probar inductivamente que el cuerpo de la regla R debe pertenecer a $Cn_R(\mathcal{P}')$. Ya que todas las reglas que fueron aplicadas para obtener el cuerpo de R pertenece a $\mathcal{P}$, entonces el cuerpo de R también está en $Cn_R(\mathcal{P})$. Por lo tanto, $x \in Cn_R(\mathcal{P})$, ya que R también pertenece a $\mathcal{P}$.

Por lo tanto, se cumple (i), (ii) y (iii). ■

Si el *plrb* es satisfacible, entonces un literal pertenecerá al conjunto de sus consecuencias si es la consecuencia de una regla estricta o rebatible aplicable. Si el programa es insatisfacible entonces sus consecuencias coinciden con *Lit*, i.e., todo el lenguaje es consecuencia de un programa inconsistente. Siendo que está permitido en la P.L.R.B. expresar información contradictoria, será de poca ayuda dar el conjunto de todas las consecuencias de un programa, pues en la mayoría de los casos este conjunto será *Lit* (como se aprecia en el ejemplo anterior). Sin embargo, la definición de argumento exige que el conjunto resultante de unir el subconjunto de reglas rebatibles que lo componen a Π sea consistente. Por esta razón, utilizaremos este concepto para dar la definición declarativa equivalente de argumento para un literal.

Definición 4.7 (Estructura de Argumento Declarativa)
Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un *plrb*. Una estructura de argumento para un literal fijo q es un par ordenado $\langle \mathcal{A}, q \rangle$, donde $\mathcal{A}$ es un subconjunto de las reglas rebatibles de $\mathcal{P}$, $\mathcal{A} \subseteq \Delta$, tal que:

1. $q \in Cn_R(\Pi \cup \mathcal{A})$
2. $Cn_R(\Pi \cup \mathcal{A}) \neq Lit$
3. $\mathcal{A}$ es minimal con respecto a la inclusión, i.e., no existe $\mathcal{A}' \subseteq \mathcal{A}$ que cumpla (1) y (2).

■

Ejemplo 4.2 Consideremos el ejemplo de los cerdos clonados 3.14. Una estructura de argumento para `gordo(trici)` es $\langle \{gordo(trici) -\!\prec cerdo(trici)\}, gordo(trici) \rangle$, ya que

$$Cn_R(\Pi \cup \{gordo(trici) -\!\prec cerdo(trici)\}) = \left\{ \begin{array}{l} clon(trici, bola\_de\_maíz), cerdo(bola\_de\_maíz), \\ cerdo(trici), cerdo\_alt\_gen(trici), gordo(trici) \end{array} \right\}$$

Así

- $gordo(trici) \in Cn_R(\Pi \cup \{gordo(trici) -\!\prec cerdo(trici)\})$,
- $Cn_R(\Pi \cup \{gordo(trici) -\!\prec cerdo(trici)\}) \neq Lit$ y
- el argumento es minimal.

Un argumento a favor de $\sim$ `gordo(trici)` es

$$\mathcal{A}' = \{\sim gordo(trici) -\!\prec cerdo\_alt\_gen(trici)\}$$

siendo

$$Cn_R(\Pi \cup \mathcal{A}') = \left\{ \begin{array}{l} clon(trici, bola\_de\_maíz), cerdo(bola\_de\_maíz), cerdo(trici), \\ cerdo\_alt\_gen(trici), \sim gordo(trici) \end{array} \right\}$$

■

Definición 4.8 (Subargumento Declarativo)
Una estructura de argumento $\langle \mathcal{B}, h' \rangle$ es una *subestructura de argumento* de $\langle \mathcal{A}, h \rangle$ si $\mathcal{B} \subseteq \mathcal{A}$. ■

Ejemplo 4.3 Consideremos el argumento para $\sim$ `fofo(trici)` bajo el programa del ejemplo 3.14.

$$\mathcal{A} = \left\{ \sim fofo(trici) -\!\prec cerdo(trici) \wedge \sim gordo(trici), \sim gordo(trici) -\!\prec cerdo\_alt\_gen(trici) \right\}$$

El argumento $\mathcal{A}'$ del ejemplo 4.2, es un subargumento de $\mathcal{A}$. ■

En la siguiente sección demostraremos que las definiciones procedural y declarativa son equivalentes.

4.3. Equivalencia entre la definición declarativa y procedural de argumento

En esta sección demostraremos que la definición anterior es la definición declarativa equivalente a la dada proceduralmente. En primer lugar, mostraremos que un literal que tiene una derivación rebatible, entonces pertenece a las consecuencias rigurosas correspondientes. En la segunda subsección, se introduce una transformación de programas rebatibles a programas definidos, un proceso intermedio necesario para llegar a la demostración deseada.

4.3.1. De lo procedural hacia lo declarativo

En esta subsección, demostraremos que si existe una derivación rebatible para q entonces q pertence a las consecuencias rigurosas del argumento correspondiente.

Lema 4.12 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un plrb,* $\mathcal{A} \subseteq \Delta$ *y* q *un literal.* $\Pi \cup \mathcal{A} |\!\sim q$ *entonces* $q \in Cn_R(\Pi \cup \mathcal{A})$.

Demostración: Probaremos el lema por inducción sobre la longitud de la derivación rebatible de q.
Caso Base: $n = 1,\ q = Q_0, Q_1 = \square$
Si la derivación de q tiene sólo un paso, entonces existe un hecho en Π cuya cabeza es q. Como $Cn_R(\Pi \cup \mathcal{A})$ es cerrado bajo $\Pi \cup \mathcal{A}$ entonces $q \in Cn_R(\Pi \cup \mathcal{A})$.

Paso inductivo: Suponemos $\Pi \cup \mathcal{A} |\!\sim p$, entonces $p \in Cn_R(\Pi \cup \mathcal{A})$ para cualquier p cuya derivación rebatible tiene k pasos, $k \leq n$.
Deberemos probar que $\Pi \cup \mathcal{A} |\!\sim q$, entonces $q \in Cn_R(\Pi \cup \mathcal{A})$ siendo que la derivación rebatible de q tiene n pasos.
Sea $q = Q_0, Q_1, \ldots, Q_n = \square$ la derivación rebatible de q. Sea $q \leftarcurve B_1, \ldots, B_r \in \Pi \cup \mathcal{A}$ la regla, estricta o rebatible, aplicada para obtener Q_1. Luego $Q_1 = B_1, \ldots B_r$. Así, cada uno de los literales $B_j,\ 1 \leq j \leq r$ tiene una derivación rebatible con longitud menor a n. Por hipótesis inductiva, $B_j \in Cn_R(\Pi \cup \mathcal{A}),\ 1 \leq j \leq r$. Ya que $Cn_R(\Pi \cup \mathcal{A})$ es cerrado con respecto a $\Pi \cup \mathcal{A}$, entonces $q \in Cn_R(\Pi \cup \mathcal{A})$. ■

Corolario 4.1 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$*, un plrb y* q *un literal fijo en su lenguaje.* q *tiene una derivación estricta entonces* $q \in Cn_R(\Pi)$.

4.3.2. De lo declarativo a lo procedural

Introducimos una transformación de programas rebatibles básicos en programas definidos sin reglas rebatibles. Esta transformación es una extensión de la presentada en [Lif96].

Definición 4.9 **(*definido*($\mathcal{P}$))**
Sea $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa rebatible básico. Para cada átomo $A \in Lit$, seleccione un nuevo símbolo A', tal que $A' \notin Lit$. Definimos en forma recursiva $definido(\mathcal{P})$,cuyo lenguaje será $Lit_{d(P)} = \{A \in Lit | A$ es un átomo $\} \cup \{A' | A$ es un átomo en
Lit y A' es el átomo fijo seleccionado para $A\}$

(I) $definido(A) = A$, si A es un átomo.

(II) $definido(\sim A) = A'$, siendo A' el nuevo símbolo fijo asignado al átomo A.

(III) $definido(Cabeza \leftarrow Cuerpo) = definido(Cabeza) \leftarrow definido(Cuerpo)$.

(IV) $definido(Cabeza -\!\prec Cuerpo) = definido(Cabeza) \leftarrow definido(Cuerpo)$.

(V) $definido(\langle \Pi, \Delta \rangle) = \{definido(R) | R \in \Pi \cup \Delta\}$.

Sea X un conjunto de literales llamaremos $definido(X) = \{definido(x) | x \in X\}$ ■

Probaremos algunas propiedades sobre la relación entre un programa rebatible y su transformación a través de *definido*.

Lema 4.13 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa rebatible básico, $C \subseteq Lit$ y $C' = definido(C)$. Si C' es cerrado bajo $definido(\Pi \cup \Delta)$, entonces C es cerrado bajo $\Pi \cup \Delta$.*

Demostración: Sea $A \twoheadleftarrow B_1, \ldots, B_n \in \Pi \cup \Delta$ y $\{B_1, \ldots, B_n\} \subseteq C$, debemos probar que $A \in C$, siendo que $\twoheadleftarrow$ representa al metasímbolo de una regla estricta ($\leftarrow$) o al metasímbolo de una regla rebatible ($-\!\prec$).
Ya que $A \twoheadleftarrow B_1, \ldots, B_n \in \Pi \cup \Delta$ entonces $definido\big(A \twoheadleftarrow B_1, \ldots, B_n\big) \in definido\big(\Pi \cup \Delta\big)$. Además $\{B_1, \ldots, B_n\} \subseteq C$, luego $definido\big(\{B_1, \ldots, B_n\}\big) \subseteq C'$. Por hipótesis, C' es cerrado bajo $definido(\Pi \cup \Delta)$, así $definido(A) \in C'$.
Siendo que la transformación *definido* no genera nuevos elementos, entonces $A \in C$. ■

Lema 4.14 *Sean $\mathcal{P} = \langle \Pi, \Delta \rangle$ un programa y $\Lambda \subseteq \Delta$, tal que $\Pi \cup \Lambda$ es consistente. Entonces*

$$Cn_R(definido(\Pi \cup \Lambda)) = definido(Cn_R(\Pi \cup \Lambda))$$

Demostración: Probaremos por definición que el conjunto de las consecuencias rigurosas de $(definido(\Pi \cup \Lambda))$ es $definido(Cn_R(\Pi \cup \Lambda))$.

I $definido(Cn_R(\Pi \cup \Lambda))$ es lógicamente cerrado. Ya que en cualquier transformación a través de $definido$ no existen literales complementarios, $definido(Cn_R(\Pi \cup \Lambda))$ es consistente.

II $definido(Cn_R(\Pi \cup \Lambda))$ es cerrado bajo $definido(\Pi \cup \Lambda)$.
Sea $A \leftarrow B_1, \ldots, B_n \in definido(\Pi \cup \Lambda)$ y $\{B_1, \ldots, B_n\} \subseteq definido(Cn_R(\Pi \cup \Lambda))$. Debemos probar que $A \in definido(Cn_R(\Pi \cup \Lambda))$.
Supongamos que $A \notin definido(Cn_R(\Pi \cup \Lambda))$, con $definido(A') = A$. Sabiendo que

$$A \leftarrow B_1, \ldots, B_n \ \in \ definido(\Pi \cup \Lambda)$$

entonces

$$A' \leftarrow B'_1, \ldots, B'_n \ \in \ \Pi \cup \Lambda$$

siendo $definido(B'_i) = B_i, \ 1 \leq i \leq n$. Ya que $\{B_1, \ldots, B_n\} \subseteq definido(Cn_R(\Pi \cup \Lambda))$, entonces $\{B'_1, \ldots, B'_n\} \subseteq Cn_R(\Pi \cup \Lambda)$. Pero $A' \notin Cn_R(\Pi \cup \Lambda)$. Contradicción.

III $definido(Cn_R(\Pi \cup \Lambda))$ es el conjunto minimal que cumple (i)-(ii).
Supongamos que existe $C \subset definido(Cn_R(\Pi \cup \Lambda))$, que cumple (i)-(ii). Luego existe un átomo A tal que $A \notin C$ y $A \in definido(Cn_R(\Pi \cup \Lambda))$.
Sea C' el conjunto de literales, tal que $C = definido(C')$. Como C es cerrado bajo $definido(\Pi \cup \Lambda)$, entonces, por el lema 4.13, C' es cerrado bajo $\Pi \cup \Lambda$. Por otra parte, $\Pi \cup \Lambda$ es consistente, así C' es lógicamente cerrado.
Por lo tanto, C' es lógicamente cerrado y cerrado bajo $\Pi \cup \Lambda$ y $C' \subset Cn_R(\Pi \cup \Lambda)$. Contradicción.

$\therefore \ Cn_R(definido(\Pi \cup \Lambda)) = definido(Cn_R(\Pi \cup \Lambda))$. ■

Corolario 4.2 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un programa y* $\Lambda \subseteq \Delta$. *Entonces*

$$Cn_R(definido(\Pi \cup \Lambda)) \subseteq definido(Cn_R(\Pi \cup \Lambda))$$

Demostración: Contemplaremos dos casos:

- Si $\Pi \cup \Lambda$ es consistente, entonces el corolario es directo por el lema 4.14.
- Si $\Pi \cup \Lambda$ es inconsistente, entonces $Cn_R(\Pi \cup \Lambda) = Lit$ y $definido(Lit)$ es igual al lenguaje de $definido(\Pi \cup \Delta)$, i.e., $Lit_{d(P)}$. Así, por definición, $Cn_R(definido(\Pi \cup \Lambda)) \subseteq Lit_{d(P)}$.

■

Evidentemente $definido(\mathcal{P})$ es un programa lógico definido no rebatible por lo que podremos aprovechar todas las propiedades demostradas para ellos.

Lema 4.15 *[Lif96] Sea* $\Pi \cup \Delta$ *un programa y* $\Lambda \subseteq \Delta$*, tal que* $\Pi \cup \Lambda$ *es consistente.* $Cn_R(definido(\Pi \cup \Lambda))$ *es el menor modelo de* $definido(\Pi \cup \Lambda)$*.*

Teorema 4.1 *[Llo87] Sean* $\mathcal{P}$ *un programa lógico definido no rebatible y* q *un átomo.* q *pertenece al modelo mínimo de* $\mathcal{P}$ *si y sólo si existe una refutación SLD de* $\mathcal{P} \cup \{\leftarrow q\}$*.*

Nótese que la definición de refutación a partir de un programa lógico rebatible es equivalente a la refutación a partir de programas no rebatibles. El siguiente lema lo formaliza.

Lema 4.16 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un programa,* $\Lambda \subseteq \Delta$*,* q *un literal y* $q' = definido(q)$*. Entonces,*

$definido(\Pi \cup \Lambda) \vdash q'$ *si y sólo si* q *es derivable rebatiblemente a partir de* $\Pi \cup \Lambda$

Demostración: Por hipótesis, $definido(\Pi \cup \Lambda) \vdash q'$, luego existe una refutación SLD de $definido(\Pi \cup \Lambda) \cup \{\leftarrow q'\}$. Sea tal refutación dada por la secuencia $\leftarrow q' = Q'_0, Q'_1, \ldots, Q'_n = \Box$. Probaremos por inducción sobre la longitud de la refutación SLD, que existe una derivación rebatible de q desde $\Pi \cup \Lambda$.

Caso Base: $n = 1$. En este caso la refutación SLD tiene la forma

$$\leftarrow q' = Q'_0, Q'_1 = \Box$$

Así debe existir una regla $q' \leftarrow true \in definido(\Pi \cup \Lambda)$. Luego existe $q \leftarrow true \in \Pi$. Por lo tanto, $Q_1 = \Box$ se deriva rebatiblemente desde $\Pi \cup \Lambda$.

Paso inductivo: Supongamos que el lema vale para $k < n$. Debemos probar que vale para n. Q'_1 fue obtenido a partir de Q'_0 utilizando una regla de la forma

$$q' \leftarrow A'_1 \ldots A'_m$$

siendo $Q'_1 = \leftarrow A'_1 \ldots A'_m$. Así existe

$$q \leftarrow A_1 \ldots A_m \in \Pi \text{ o bien } q \prec A_1 \ldots A_m \in \Lambda \tag{4.3}$$

siendo $A'_i = definido(A_i),\ 1 \leq i \leq m$.
La refutación SLD $Q'_1 = \leftarrow A'_1 \ldots A'_m, Q'_2, \ldots, Q'_n = \Box$ tiene menos de n pasos. Por hipótesis inductiva, existe una derivación rebatible $Q_1 = \prec A_1 \ldots A_m, Q_2, \ldots, Q_r = \Box$.
$\prec A_1 \ldots A_m$ es derivable rebatiblemente desde $\prec q$ utilizando alguna de las reglas (4.3). Por lo tanto,

$$Q_0 = \prec q,\ Q_1 = \prec A_1 \ldots A_m, Q_2, \ldots, Q_r = \Box$$

es una derivación rebatible de q a partir de $\Pi \cup \Lambda$.

El recíproco se prueba en forma análoga. ■

4.3.3. Equivalencia entre las definiciones

En esta sección presentamos la demostración más importante: la equivalencia entre las dos definiciones.

Lema 4.17 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un programa y* $\Lambda \subseteq \Delta$*, tal que* $\Pi \cup \Lambda$ *es consistente.* $h \in Cn_R(\Pi \cup \Lambda)$ *si y sólo si* h *es derivable rebatiblemente a partir de* $\Pi \cup \Lambda$.

Demostración: $h \in Cn_R(\Pi \cup \Lambda)$ si y sólo si
$h' = definido(h)$ y $h' \in definido(Cn_R(\Pi \cup \Lambda))$ si y sólo si
por ser $\Pi \cup \Lambda$ consistente, entonces aplicando el lema 4.14 $h' \in Cn_R(definido(\Pi \cup \Lambda))$ si y sólo si
por lema 4.15 $Cn_R(definido(\Pi \cup \Lambda))$ es el menor modelo de $definido(\Pi \cup \Lambda)$ si y sólo si
por lema 4.1 existe una refutación SLD de $definido(\Pi \cup \Lambda) \cup \{\leftarrow h'\}$ si y sólo si
h es derivable rebatiblemente a partir de $\Pi \cup \Lambda$. ■

Lema 4.18 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un programa y* $\Lambda \subseteq \Delta$. $\Pi \cup \Lambda$ *es consistente si y sólo si* $\Pi \cup \Lambda$ *no es contradictorio.*

Demostración: Supongamos que $\Pi \cup \Lambda$ es contradictorio. Luego existe un literal L tal que $\Pi \cup \Lambda$ deriva rebatiblemente a L y a $\overline{L}$. Por lema 4.16 $definido(\Pi \cup \Lambda) \vdash L'$ y $definido(\Pi \cup \Lambda) \vdash \overline{L}'$ siendo $L' = definido(L)$ y $\overline{L}' = definido(\overline{L})$.
Por el lema 4.1 $\{L', \overline{L}'\}$ está incluido en el modelo minimal y aplicando el lema 4.15 tenemos que $\{L', \overline{L}'\} \subseteq Cn_R(definido(\Pi \cup \Lambda))$.
Por corolario 4.2 $\{L', \overline{L}'\} \subseteq definido(Cn_R(\Pi \cup \Lambda))$. Por lo tanto, $\{L, \overline{L}\} \subseteq Cn_R(\Pi \cup \Lambda)$, i.e., $\Pi \cup \Lambda$ es inconsistente. Esta contradicción provino de suponer que $\Pi \cup \Lambda$ es contradictorio.

Recíprocamente, si $\Pi \cup \Lambda$ es inconsistente, entonces existe al menos un literal L que genera tal inconsistencia y que hace que $Cn_R(\Pi \cup \Lambda) = Lit$. Para tal literal L debe ser que $\{L'\overline{L}'\} \subseteq Cn_R(definido(\Pi \cup \Lambda))$, siendo $L' = definido(L)$ y $\overline{L}' = definido(\overline{L})$. Por los lemas 4.15 y 4.1 $\{L'\overline{L}'\}$ está incluido en el modelo mínimo de $definido(\Pi \cup \Lambda)$ y, por lo tanto, $definido(\Pi \cup \Lambda) \vdash L'$ y $definido(\Pi \cup \Lambda) \vdash \overline{L}'$. Luego tanto L como $\overline{L}$ son derivables rebatiblemente a partir de $\Pi \cup \Lambda$ por el lema 4.16, i.e., $\Pi \cup \Lambda$ es contradictorio. ■

Teorema 4.2 *Sean* $\mathcal{P} = \langle \Pi, \Delta \rangle$ *un programa y* $\mathcal{A} \subseteq \Delta$. $\mathcal{A}$ *es un argumento según la definición 4.7 si y sólo si* $\mathcal{A}$ *es un argumento según la definición 3.27.*

Demostración: Probaremos que las tres condiciones de las definiciones son equivalentes.

1. $Cn_R(\Pi \cup \mathcal{A})$ es consistente si y sólo si $\Pi \cup \mathcal{A}$ no es contradictorio. Lema 4.18.

2. Siendo que $\Pi \cup \mathcal{A}$ es consistente, $h \in Cn_R(\Pi \cup \mathcal{A})$ si y sólo si h es derivable rebatiblemente a partir de $\Pi \cup \mathcal{A}$, por lema 4.17.

3. Sea $\mathcal{A}$ un argumento para h bajo la definición declarativa 4.7. Debemos probar que $\mathcal{A}$ es el conjunto minimal que cumple las condiciones (1) y (2) de la definición procedural 3.27. Supongamos que existe $\mathcal{A}' \subseteq \mathcal{A}$ que cumple las condiciones (1) y (2) de la definición procedural 3.27.
   Ya que $\Pi \cup \mathcal{A}'$ deriva rebatiblemente a h y que $\Pi \cup \mathcal{A}'$ no es contradictotio, por lema 4.17 $h \in Cn_R(\Pi \cup \mathcal{A}')$. Además $Cn_R(\Pi \cup \mathcal{A}')$ es consistente. Por lo tanto, $\mathcal{A}$ no es un argumento para h según la definición declarativa, pues no es minimal. Contradicción. Dicha contradicción provino de suponer que $\mathcal{A}$ no es el conjunto minimal que cumple (1) y (2) de la definición procedural de argumento.
   El recírproco se demuestra en forma análoga.

Luego ambas definiciones son equivalentes. ■

Hemos presentado una definición declarativa de la estructura más importante de los Sistemas Argumentativos. Esta definición será utilizada a continuación para la determinar la Semántica basada en Juegos.

4.4. Semántica declarativa basada en juegos

El análisis dialéctico, en el que se basa la argumentación, se asemeja a un juego en el que de manera alternada de dos jugadores compiten. Un jugador, el *Proponente*, propone un argumento y luego trata de defenderlo contra cualquier ataque que provenga del otro participante, el *Oponente*. En este juego los jugadores hacen sucesivos *movimientos* de acuerdo a un conjunto de *reglas*, introduciendo argumentos. Las reglas del juego pueden variar de acuerdo a las necesidades y al consenso común. Al definir el juego, consideraremos, como es natural, aquellas posiciones que han sido defendidas en forma exitosa y que determinaran el criterio vencedor. Cambiar las reglas que gobiernan el juego, redundará en cambios en las posiciones consideradas ganadoras.

Los movimientos permitidos, las reglas y las posiciones ganadoras y perdedoras constituyen la semántica del análisis dialéctico y serán reflejados a través de la noción de juego entre dos participantes[Abr97].

Comenzaremos especificando los componentes en un juego, los que analizaremos en las siguentes secciones:

Jugadores: podemos considerar no sólo aquellos que juegan el juego sino también los jueces. Diferentes juegos pueden ser modelados de acuerdo al número de jugadores.

Arena y Piezas: debemos definir el ambiente donde jugaremos el juego y las piezas que se utilizarán en el juego.

Reglas: no todos los movimientos de las piezas en una arena son legales. Por lo tanto, debemos definir las reglas que determinaran cuáles de esos movimientos son los legales.

Turnos: Para cada estado del juego debemos determinar qué jugador es el próximo a jugar. Por otra parte, algunos juegos establecen cuál jugador es el primero en mover.

Ganador: debemos determinar cuándo el juego termina y qué jugador ganó la partida.

Si bien Prakken[Pra00, Pra01] sugiere otros elementos, nosotros consideramos que estos son los principales y que los demás están incluidos en este listado.

4.4.1. Jugadores

El conjunto de jugadores es de cardinalidad dos y son identificados como el *proponente* y el *oponente*.

Definición 4.10 (Jugadores - Jugador_Inicial - Adversario) Definimos al conjunto de *Jugadores* como $\{P, O\}$ donde P representa al Proponente y O representa al Oponente. El Proponente es el jugador que realiza la primer movida y lo notaremos *Jugador_Inicial*.

El *adversario o complemento de un jugador* está definido como sigue:

$$\overline{Jugador} = \begin{cases} O & \text{si } Jugador = P \\ P & \text{si } Jugador = O \end{cases}$$

■

Trivialmente podemos extender la definición de Jugadores a un conjunto posiblemente infinito de jugadores. Si fuera el caso, entonces $P = \{P_1, P_2, ...\}$ y $O = \{O_1, O_2, ...\}$, donde $P_i \neq O_j$, para todo $1 \geq i, j$. En este caso, *Jugador_Inicial* es cualquier miembro de P y el adversario de un jugador en el conjunto P es cualquier jugador del conjunto O y vice versa.

En una disputa existe un conjunto estático bien establecido de jugadores que juegan a favor y en contra de una posición inicial. Podríamos considerar una disputa en donde los jugadores cambien dinámicamente de un grupo a otro, si un argumento convence al jugador de hacerlo.

4.4.2. Arena y Piezas

Las instrucciones para jugar un juego comienzan con la definición del ambiente en el que se desarrollará el juego y con las piezas que necesitaremos para participar. En nuestro caso, las piezas del juego serán los *argumentos*. El juego comienza con un argumento y, en forma sucesiva y alternada, se jugarán diferentes argumentos hasta llegar a una situación en la que el juego termine y se pueda determinar qué jugador ha vencido.

Nuestro primer objetivo será circunscribir el conjunto de argumentos potenciales que pueden participar en un juego para un literal h. Así definiremos en primer lugar el conjunto de todos los posibles argumentos que eventualmente pueden intervenir en el análisis dialéctico que comienza con una estructura de argumento. Antes de dar la definición formal, haremos un análisis que ejemplifique las decisiones de diseño.

Estamos interesados en encontrar el conjunto minimal de estructuras de argumentos que podrían ser utilizadas en la construcción del árbol dialéctico con raíz $\langle \mathcal{A}, h \rangle$ bajo el *plrb* $\mathcal{P} = (\Pi, \Delta)$. Consideremos los argumentos vacío y analicemos su inclusión en el conjunto minimal que deseamos encontrar. Si existe una estructura de argumento $\langle \varnothing, l \rangle$, luego $l \in Cn_R(\Pi)$ por definición de consecuencias rigurosas. Ya que $Cn_R(\Pi) \subseteq Cn_R(\Pi \cup \mathcal{A})$ para todo $\mathcal{A}$, luego $l \in Cn_R(\Pi \cup \mathcal{A})$. Consideremos los dos casos:

- Si l no es necesario para derivar h, luego no deseamos incluir $\langle \varnothing, l \rangle$ en el conjunto de argumentos potenciales. Ilustremos este caso: $\mathcal{P} = (\Pi, \Delta)$, $\Pi = \{a, b\}$ y $\Delta = \{h \prec a\}$. Existe una estructura de argumento para h, $\langle \{h \prec a\}, h \rangle$. Así, $Cn_R(\Pi) = \{a, b\}$ y $Cn_R(\Pi \cup \{h \prec a\}) = \{a, b, h\}$. En este caso b no es utilizado para derivar h, luego no estamos interesados en considerar $\langle \varnothing, b \rangle$ como un movimiento potencial.

- Por otro lado, supongamos que l es necesario para derivar h. Tal como lo indican en [GS04], l tiene como única estructura de argumento a $\langle \varnothing, l \rangle$ y $\langle \varnothing, l \rangle$ no tiene contraargumentos. Entonces no podemos utilizar $\langle \varnothing, l \rangle$ para contraargumentar a ningún argumento. Por lo tanto, aun si l es necesario para derivar h, no es útil considerar $\langle \varnothing, l \rangle$ como un argumento.

Por esta razón, en la generación de posibles puntos de ataque para un conjunto de argumentos, no se considerarán a los argumentos vacío.

Dada una estructura de argumento $\langle \mathcal{A}, h \rangle$ analicemos qué argumentos podrían ser parte de su árbol dialéctico:

- **Ataque a la conclusión**: El primer punto de ataque es h, así todos los argumentos para $\overline{h}$ podrían, aunque no necesariamente, formar parte del árbol con raíz $\langle \mathcal{A}, h \rangle$.

- **Defensa de la conclusión**: Cualquier otro argumento a favor de h puede ser utilizado para defender la raíz del árbol de los contraargumentos. Luego, todos los argumentos $\langle \mathcal{A}', h\rangle$ podrían formar parte del árbol dialéctico.
- **Ataque al camino hacia la conclusión**: También son propensos a ataques los subargumentos de $\langle \mathcal{A}, h\rangle$. Por lo tanto, dado un literal l en $Cn_R(\Pi \cup \mathcal{A})$, todo argumento $\langle \mathcal{A}', \bar{l}\rangle$, excepto aquellos que son vacíos, i.e., $\mathcal{A}' = \varnothing$, pueden estar en el árbol dialéctico.
- **Defensa de las minas en el camino**: Los argumentos a favor de subargumentos de $\langle \mathcal{A}, h\rangle$ podrían estar en el árbol dialéctico, en caso de existir ataques en el camino a la conclusión h. Luego, dado un literal l en $Cn_R(\Pi \cup \mathcal{A})$, todo argumento $\langle \mathcal{A}', l\rangle$, excepto aquellos que son vacíos, i.e., $\mathcal{A}' = \varnothing$, pueden estar en el árbol dialéctico.
- Finalmente, todos los argumentos y constraargumentos a los argumentos incluidos en los cuatro puntos anteriores, también son potenciales elementos del árbol dialéctico.

A continuación definimos una función que toma como entrada un conjunto de estructuras de argumentos EA y devuelve todos los argumentos a favor y en contra de los miembros de EA. Esta noción utiliza el concepto de consecuencia rigurosa dado en la definición 4.6.

Definición 4.11 **($Arg_{\mathcal{P}}$)** Sean $\mathcal{P}$ un *plrb*, *Argumentos* el conjunto de todos las estructuras de argumentos que pueden ser generadas a partir de $\mathcal{P}$ y $EA \subseteq Argumentos$. Definimos la función $Arg_{\mathcal{P}} : 2^{Argumentos} \to 2^{Argumentos}$ como sigue:

$$\begin{aligned} Arg_{\mathcal{P}}(EA) \quad = \quad & \{\langle \mathcal{A}, l\rangle | \langle \mathcal{A}, l\rangle \text{ es una estructura de argumento bajo } \mathcal{P} \text{ y} \\ & \mathcal{A} \neq \emptyset,\ l \in\ Cn_R(\Pi \cup \mathcal{A}')\ y\ \langle \mathcal{A}', h'\rangle\ \in\ EA\} \cup \\ & \{\langle \mathcal{A}, \bar{l}\rangle | \langle \mathcal{A}, \bar{l}\rangle \text{ es una estructura de argumento bajo } \mathcal{P} \text{ y} \\ & \mathcal{A} \neq \emptyset,\ l \in\ Cn_R(\Pi \cup \mathcal{A}')\ y\ \langle \mathcal{A}', h'\rangle\ \in\ EA\} \end{aligned}$$

■

$Arg_{\mathcal{P}}$ está definido sobre el conjunto de partes de todas las estructuras de argumento generadas bajo *plrb* arbitrario. La estructura algebraica $\langle 2^{Argumentos}, \subseteq, \cup, \cap\rangle$ es un reticulado completo. Nuestra meta final es encontrar el menor punto fijo de $Arg_{\mathcal{P}}$. Como paso intermedio probaremos que $Arg_{\mathcal{P}}$ es monotónico -preserva el orden- y continuo.

Lema 4.19 **($Arg_{\mathcal{P}}$ es monotónico)** *Sea $\mathcal{P}$ un plrb, Argumentos todas las estructuras de argumento sobre $\mathcal{P}$ y E, E' subconjunto de Argumentos. Si $E \subseteq E'$ luego $Arg_{\mathcal{P}}(E) \subseteq Arg_{\mathcal{P}}(E')$.*

Demostración: Sea $\langle \mathcal{A}, h\rangle \in Arg_{\mathcal{P}}(E)$, $\mathcal{A} \neq \varnothing$ por definición del mapeo $Arg_{\mathcal{P}}$. Luego existe una estructura de argumento $\langle B, l\rangle \in E$ y $h \in Cn_R(\Pi \cup B)$ o bien $\overline{h} \in Cn_R(\Pi \cup B)$. Por hipótesis $E \subseteq E'$, luego $\langle B, l\rangle \in E'$ y como $h \in Cn_R(\Pi \cup B)$ o bien $\overline{h} \in Cn_R(\Pi \cup B)$ luego $\langle \mathcal{A}, h\rangle \in Arg_{\mathcal{P}}(E')$. ■

Teorema 4.3 *$Arg_{\mathcal{P}}$ tiene un menor punto fijo.*

Demostración: *$Arg_{\mathcal{P}}$* es una función definida sobre el conjunto $2^{Argumentos}$, siendo *Argumentos* el conjunto de todos los argumentos que se pueden construir a partir de $\mathcal{P}$. Dicho conjunto de partes es un reticulado completo y, por lo tanto, es un conjunto parcialmente ordenado completo. Por otra parte, por el lema 4.19, *$Arg_{\mathcal{P}}$* preserva el orden. Estamos en las condiciones del teorema A.19 de Punto Fijo (ver Apéndice página 209), por lo que *$Arg_{\mathcal{P}}$* tiene un menor punto fijo. ■

Lema 4.20 *Sean $\mathcal{P}$ un plr, Argumentos todas las estructuras de argumentos sobre $\mathcal{P}$ y Ψ un conjunto dirigido de $2^{Argumentos}$. $\{A_1, \ldots, A_n\} \subseteq \bigvee \Psi$ si y solamente si $\{A_1, \ldots, A_n\} \subseteq \psi_i$, para algún $\psi_i \in \Psi$.*

Demostración: $\Leftarrow$) Dado que $\{A_1, \ldots, A_n\} \subseteq \psi_i$ para algún i, $\psi_i \in \Psi$ luego

$$\{A_1, \ldots, A_n\} \subseteq \bigcup_{i \in I_\Psi} \psi_i,$$

siendo I_Ψ un índice en Ψ. Por lo tanto, $\{A_1, \ldots, A_n\} \subseteq \bigvee_{i \in I_\Psi} \psi_i$.

$\Rightarrow$) Por hipótesis, $\{A_1, \ldots, A_n\} \subseteq \bigvee_{i \in I_\Psi} \psi_i$, así $\{A_1, \ldots, A_n\} \subseteq \bigcup_{i \in I_\Psi} \psi_i$. Entonces debe ser el caso de que

$$A_1 \in \psi_{l_1}, A_2 \in \psi_{l_2}, \ldots A_n \in \psi_{l_n}$$

con $l_j \in I_\Psi$. Ya que Ψ es un conjunto dirigido $\bigvee\{\psi_{l_1}, \ldots, \psi_{l_n}\} \in \Psi$. Por lo tanto, $\{A_1, \ldots, A_n\} \subseteq \psi_i$, para algún $\psi_i \in \Psi$. ■

Lema 4.21 (*$Arg_{\mathcal{P}}$* **es continuo)** *Sea $\mathcal{P}$ un plrb y Argumentos todas las estructuras de argumentos sobre $\mathcal{P}$. Luego*

$$Arg_{\mathcal{P}}(\bigcup_{i=1} \psi_i) = \bigcup_{i=1} Arg_{\mathcal{P}}(\psi_i)$$

para cada conjunto dirigido Ψ de $2^{Argumentos}$ y $\psi_i \in \Psi$.

Demostración: Sea Ψ un subconjunto dirigido de $2^{Argumentos}$.

	$\langle \mathcal{A}', l \rangle \in Arg_{\mathcal{P}}(\bigcup_{i \in I_\Psi} \psi_i)$,	
si y sólo si	$l \in Cn_R(\Pi \cup \mathcal{A})$ o $\overline{l} \in Cn_R(\Pi \cup \mathcal{A})$ y	
	$\langle \mathcal{A}, h \rangle \in \bigcup_{i \in I_\Psi} \psi_i$.	por definición de Arg
si y sólo si	$\langle \mathcal{A}, h \rangle \in \psi_j$ para algún $j \in I_\Psi$.	por el lema anterior
si y sólo si	$\langle \mathcal{A}', l \rangle \in Arg_{\mathcal{P}}(\psi_j)$	por definición de Arg
si y sólo si	$\langle \mathcal{A}', l \rangle \in \bigcup_{i \in I_\Psi} Arg_{\mathcal{P}}(\psi_i)$.	

■

Lema 4.22 (Punto Fijo de $Arg_{\mathcal{P}}$) *Sea $\mathcal{P}$ un plrb y Argumentos todas las estructuras de argumentos sobre $\mathcal{P}$. El menor punto fijo de $Arg_{\mathcal{P}}$ existe y se encuentra en ω.*

Demostración: Por el teorema del punto fijo de Knaster-Tarski[Tar55] existe el menor punto fijo para $Arg_{\mathcal{P}}$ (ver teorema 4.3) y ya que $Arg_{\mathcal{P}}$ es continuo, por la proposición A.2, su menor punto fijo se encuentra en ω. ∎

El objetivo de definir el mapeo $Arg_{\mathcal{P}}$ es circunscribir el conjunto de los argumentos que pueden ser jugados a favor y en contra para un literal h. Así, definiremos una secuencia de estructuras que se generan aplicando $Arg_{\mathcal{P}}$ que se ajustarán estrictamente al literal h sobre el que vamos a construir el juego.

Definición 4.12 Sean h un literal y $\mathcal{P} = \langle \Pi, \Delta \rangle$ un *plrb*. $Arg^i(h, \mathcal{P})$ se define como sigue:

$$\begin{array}{lcl} Arg^0(h,\mathcal{P}) & = & \{\langle \mathcal{A}, h\rangle | \langle \mathcal{A}, h\rangle \text{ es una estructura de argumento para } h \text{ bajo } \mathcal{P}\} \\ Arg^i(h,\mathcal{P}) & = & Arg_{\mathcal{P}}(Arg^{i-1}(h,\mathcal{P})) \end{array}$$ ∎

Ejemplo 4.4 Consideremos el siguiente *plrb*:

$$\Pi = \left\{ \begin{array}{l} d. \\ a \leftarrow d. \\ \sim e. \end{array} \right\} \qquad \Delta = \left\{ \begin{array}{ll} h \dashrightarrow b, c. & c \dashrightarrow d. \\ h \dashrightarrow a. & \sim c \dashrightarrow f. \\ h \dashrightarrow \ \sim e, d. & f \dashrightarrow a. \\ b \dashrightarrow d. & \sim f \dashrightarrow d. \\ b \dashrightarrow \ \sim e. & g \dashrightarrow a, \sim k. \\ \sim k \dashrightarrow d. & \sim g \dashrightarrow \ \sim e. \end{array} \right\}$$

Analicemos en primer lugar al literal h y todos sus argumentos:

$$\begin{array}{lcl} Arg^0(h,\mathcal{P}) & = & \Big\{ \ \langle \{h \prec b, c.; b \prec d.; c \prec d.\}, h\rangle, \ \langle \{h \prec b, c.; b \prec \ \sim e.; c \prec d.\}, h\rangle, \\ & & \quad \langle \{h \prec a.\}, h\rangle, \ \langle \{h \prec \ \sim e, d.\}, h\rangle \ \Big\} \\ Arg^1(h,\mathcal{P}) & = & Arg^0(h,\mathcal{P}) \ \cup \Big\{ \ \langle \{b \prec d.\}, b\rangle, \ \langle \{b \prec \ \sim e.\}, b\rangle, \\ & & \quad \langle \{c \prec d.\}, c\rangle, \ \langle \{\sim c \prec f.; f \prec a.\}, \sim c\rangle \ \Big\} \\ Arg^2(h,\mathcal{P}) & = & Arg^1(h,\mathcal{P}) \ \cup \Big\{ \ \langle \{f \prec a.\}, f\rangle, \ \langle \{\sim f \prec d\}, \sim f\rangle \Big\} \\ Arg^3(h,\mathcal{P}) & = & Arg^2(h,\mathcal{P}) \end{array}$$

Analicemos ahora al literal a y todos sus argumentos:

$$Arg^0(a,\mathcal{P}) = \Big\{\langle\varnothing, a\rangle\Big\}$$

$$Arg^1(a,\mathcal{P}) = \Big\{\ \Big\}$$

Nótese que los argumentos vacíos desaparecen al aplicar la función $Arg_{\mathcal{P}}$. Esto tiene su razón en el hecho de que dichos argumentos no tienen contraargumentos y que además no pueden ser contraargumentos de ningún otro argumento. Así en este caso el conjunto minimal de potenciales argumentos a favor y en contra de $\langle\varnothing, a\rangle$ es vacío. ■

Dicha secuencia cumple la condición de que cada elemento está incluido en el siguiente, excepto cuando una estructura de argumento de la forma $\langle\varnothing, h\rangle$ es miembro de $Arg^0(h,\mathcal{P})$, ya que no existen puntos de ataque para este argumento.

Proposición 4.1 *Sean h un literal y $\mathcal{P} = \langle\Pi,\Delta\rangle$ un plrb. Si $\langle\varnothing, h\rangle \notin Arg^0(h,\mathcal{P})$, entonces*

$$Arg^i(h,\mathcal{P}) \subseteq Arg^{i+1}(h,\mathcal{P}),\ 0 \leq i.$$

Demostración: Lo probaremos por inducción sobre i.

Caso Base: Sea $\langle\mathcal{A}, h\rangle \in Arg^0(h,\mathcal{P})$, siendo $\mathcal{A} \neq \varnothing$ por hipótesis. Dado que $h \in Cn_R(\Pi \cup \mathcal{A})$, luego por definición de $Arg_{\mathcal{P}}$, tenemos que $h \in Arg_{\mathcal{P}}(Arg^0(h,\mathcal{P}))$.

Paso Inductivo: Supongamos que vale la proposición para $k < n$ debo probar que vale $Arg^n(h,\mathcal{P}) \subseteq Arg^{n+1}(h,\mathcal{P})$.

Sea $\langle\mathcal{A}, l\rangle \in Arg^n(h,\mathcal{P})$. Luego por definición, existe $\langle\mathcal{B}, m\rangle \in Arg^{n-1}(h,\mathcal{P})$ tal que $l \in Cn_R(\Pi \cup \mathcal{B})$ o bien $\bar{l} \in Cn_R(\Pi \cup \mathcal{B})$. Por hipótesis inductiva,

$$Arg^{n-1}(h,\mathcal{P}) \subseteq Arg^n(h,\mathcal{P})$$

Así, $\langle\mathcal{B}, m\rangle \in Arg^n(h,\mathcal{P})$ y por lo tanto por definición $\langle\mathcal{A}, l\rangle \in Arg_{\mathcal{P}}(Arg^n(h,\mathcal{P}))$, es decir, $\langle\mathcal{A}, l\rangle \in Arg^{n+1}(h,\mathcal{P})$.

$$\therefore Arg^i(h,\mathcal{P}) \subseteq Arg^{i+1}(h,\mathcal{P}),\ 0 \leq i.$$

■

Proposición 4.2 *Sean h un literal y $\mathcal{P} = \langle\Pi,\Delta\rangle$ un plrb. Si $\langle\varnothing, h\rangle \in Arg^0(h,\mathcal{P})$, entonces $Arg^1(h,\mathcal{P}) = \varnothing$*

Demostración: Si $\langle \varnothing, h\rangle \in Arg^0(h, \mathcal{P})$, entonces todos los argumentos que existen para h son vacíos, de otro modo no serían minimales. Todos los literales de $Cn_R(\Pi)$ tienen estructuras con su argumento vacío. Así, por definición, ninguna puede pertenecer a $Arg_{\mathcal{P}}(Arg^0(h, \mathcal{P}))$, es decir que $Arg^1(h, \mathcal{P})$ es vacío. ■

La secuencia generada a través de $Arg^i(h, \mathcal{P})$ no es necesariamente finita. Veamos un ejemplo.

Ejemplo 4.5 Consideremos el siguiente esquema de programa $\mathcal{P}$:

$$\Pi \quad = \quad \left\{ \quad \texttt{p(a).} \quad \right\} \qquad\qquad \Delta \quad = \quad \left\{ \begin{array}{l} \texttt{q(X)} \dashrightarrow \texttt{q(f(X)).} \\ \texttt{q(f(X))} \dashrightarrow \texttt{p(a).} \end{array} \right\}$$

$$\begin{array}{lcl} Arg^0(q(a), \mathcal{P}) & = & \{\langle \{q(a) \prec q(f(a)); q(f(a)) \prec p(a)\}, q(a)\rangle\} \\ Arg^1(q(a), \mathcal{P}) & = & \{\langle \{q(a) \prec q(f(a)); q(f(a)) \prec p(a)\}, q(a)\rangle, \\ & & \langle \{q(f(a)) \prec q(f(f(a))); q(f(f(a))) \prec p(a)\}, q(f(a))\rangle\} \\ Arg^2(q(a), \mathcal{P}) & = & \{\langle \{q(a) \prec q(f(a)); q(f(a)) \prec p(a)\}, q(a)\rangle, \\ & & \langle \{q(f(a)) \prec q(f(f(a))); q(f(f(a))) \prec p(a)\}, q(f(a))\rangle, \\ & & \langle \{q(f(f(a))) \prec q(f(f(f(a)))); q(f(f(f(a)))) \prec p(a)\}, q(f(a))\rangle\} \\ \vdots & & \\ Arg^{\omega}(q(a), \mathcal{P}) & = & \{\langle \{q(a) \prec q(f(a)); q(f(a)) \prec p(a)\}, q(a)\rangle, \\ & & \langle \{q(f(a)) \prec q(f(f(a))); q(f(f(a))) \prec p(a)\}, q(f(a))\rangle, \\ & & \langle \{q(f(f(a))) \prec q(f(f(f(a)))); q(f(f(f(a)))) \prec p(a)\}, q(f(a))\rangle, \ldots\} \end{array}$$

■

Definición 4.13 Sean h un literal y $\mathcal{P} = \langle \Pi, \Delta\rangle$ un *plrb*. El conjunto de todos los argumentos y contraargumentos que pueden participar en un análisis dialéctico que comienza con una estructura de argumento para h, que notaremos $Arg^h_{\mathcal{P}}$, se define como sigue:

$$Arg^h_{\mathcal{P}} = \left\{ \begin{array}{ll} Arg^0(h, \mathcal{P}) & \text{si } \langle \varnothing, h\rangle \in Arg^0(h, \mathcal{P}) \\ Arg(h, \mathcal{P}) \downarrow & \text{i.e., el menor punto fijo de } Arg. \end{array} \right.$$

■

$Arg^h_{\mathcal{P}}$ contiene todos los puntos de ataque potenciales, tanto a favor como en contra, para un literal h bajo un programa lógico rebatible $\mathcal{P} = \langle \Pi, \Delta\rangle$.

Proposición 4.3 *Sean h un literal y $\mathcal{P} = \langle \Pi, \Delta\rangle$ un plrb. $\langle \varnothing, h\rangle \in Arg^0(h, \mathcal{P})$ si y solamente si $Arg^h_{\mathcal{P}} = Arg^0(h, \mathcal{P}) = \{\langle \varnothing, h\rangle\}$.*

Demostración: Si $\langle \varnothing, h\rangle$ es una estructura de argumento para h bajo $\mathcal{P}$, luego no pueden existir otros argumentos a favor de h, ya que violarían la condición de minimalidad. Tampoco

existen contraargumentos porque no cumplirían la condición de consistencia. Así $Arg^0(h, \mathcal{P})$ es un conjunto unitario que contiene solamente a $\langle \varnothing, h \rangle$. Por lo tanto, por definición de $Arg^h_{\mathcal{P}}$ tenemos que $Arg^h_{\mathcal{P}} = Arg^0(h, \mathcal{P})$.

El recíproco es trivial. ∎

Proposición 4.4 *Sean h un literal y $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb. h no tiene argumentos bajo $\mathcal{P}$ si y solamente si $Arg^h_{\mathcal{P}}$ es vacío.*

Demostración: h no tiene argumentos bajo $\mathcal{P}$ si y solamente si $Arg^0(h, \mathcal{P}) = \varnothing$ si y solamente si $Arg^1(h, \mathcal{P}) = Arg^0(h, \mathcal{P}) = \varnothing$ si y solamente si $Arg^0(h, \mathcal{P})$ es el menor punto fijo de Arg^i si y solamente $Arg^h_{\mathcal{P}} = Arg^0(h, \mathcal{P}) = \varnothing$. ∎

Corolario 4.3 *Sean h un literal y $\mathcal{P} = \langle \Pi, \Delta \rangle$ un plrb. Si $\langle \varnothing, h \rangle$ es una estructura de argumento para h bajo $\mathcal{P}$, entonces $Arg^{\overline{h}}_{\mathcal{P}}$ es vacío.*

Demostración: Si $\langle \varnothing, h \rangle$ es una estructura de argumento para h bajo $\mathcal{P}$, entonces h tiene un solo argumento, ya que cualquier otro violaría la condición de minimalidad y no puede tener contraargumentos ya que violaría la condición de consistencia. Así, al no tener contraargumentos, no existen argumentos a favor de $\overline{h}$. Por la proposición 4.4, $Arg^{\overline{h}}_{\mathcal{P}} = \varnothing$. ∎

Si ni $Arg^h_{\mathcal{P}}$ ni $Arg^{\overline{h}}_{\mathcal{P}}$ son vacíos, luego todos los argumentos en $Arg^h_{\mathcal{P}}$ son contraargumentos de $Arg^{\overline{h}}_{\mathcal{P}}$ y vice versa. Por lo tanto, estos argumentos estarán en ambos conjuntos y todos los argumentos a favor y en contra estarán en $Arg^h_{\mathcal{P}}$ y en $Arg^{\overline{h}}_{\mathcal{P}}$.

Lema 4.23 *Si $Arg(h, \mathcal{P}) \neq \varnothing$ y $Arg(\overline{h}, \mathcal{P}) \neq \varnothing$, luego $Arg^0(h, \mathcal{P}) \subset Arg^1(\overline{h}, \mathcal{P})$.*

Demostración: Ya que $Arg(\overline{h}, \mathcal{P}) \neq \varnothing$, luego existe al menos una estructura de argumento $\langle \mathcal{A}, \overline{h} \rangle$, ya que de otro modo $Arg^0(\overline{h}, \mathcal{P})$ sería vacío y por lo tanto el menor punto fijo, i.e., que $Arg(\overline{h}, \mathcal{P})$ sería vacío. Además $\mathcal{A} \neq \varnothing$, ya que de otro modo $Arg(h, \mathcal{P})$ sería vacío. Por definición $Arg^1(\overline{h}, \mathcal{P}) = Arg_{\mathcal{P}}(Arg^0(\overline{h}, \mathcal{P}))$. Ya que $\langle \mathcal{A}, \overline{h} \rangle \in Arg^0(\overline{h}, \mathcal{P})$, $\overline{h} \in Cn_R(\Pi \cup \mathcal{A})$ y $\mathcal{A} \neq \varnothing$, entonces todas las estructuras de argumentos para h pertenecerán a $Arg^1(\overline{h}, \mathcal{P})$. El conjunto de todas las estructuras de argumentos para h es $Arg^0(h, \mathcal{P})$. Luego $Arg^0(h, \mathcal{P}) \subseteq Arg^1(\overline{h}, \mathcal{P})$. Al menos un argumento de la forma $\langle \mathcal{A}, \overline{h} \rangle$ pertenece a $Arg^1(\overline{h}, \mathcal{P})$, pues $Arg^0(\overline{h}, \mathcal{P}) \subseteq Arg^1(\overline{h}, \mathcal{P})$ y $\langle \mathcal{A}, \overline{h} \rangle$ no es miembro de $Arg^0(h, \mathcal{P})$.
$\therefore$ $Arg^0(h, \mathcal{P}) \subset Arg^1(\overline{h}, \mathcal{P})$. ∎

Proposición 4.5 *Si $Arg^h_{\mathcal{P}} \neq \varnothing$ y $Arg^{\overline{h}}_{\mathcal{P}} \neq \varnothing$, entonces $Arg^h_{\mathcal{P}} = Arg^{\overline{h}}_{\mathcal{P}}$.*

Demostración: Si ninguno de los conjuntos es vacío, luego los argumentos de $Arg^0(h, \mathcal{P}) \subset Arg^1(\overline{h}, \mathcal{P})$ y $Arg^0(\overline{h}, \mathcal{P}) \subset Arg^1(h, \mathcal{P})$, por el lema 4.23. Así los núcleos a partir de los que se generan $Arg(h, \mathcal{P})$ y $Arg(\overline{h}, \mathcal{P})$ son miembros de ambos conjuntos. Por lo tanto, $Arg(h, \mathcal{P}) = Arg(\overline{h}, \mathcal{P})$. ■

Habiendo caracterizado el conjunto de argumentos a favor y en contra de un literal bajo un programa, podremos a continuación definir los movimientos potenciales en el juego y la secuencia en que se deberían presentar.

4.4.3. Piezas y Movimientos Permitidos

Dada una estructura de argumento $\langle \mathcal{A}, h \rangle$ para un literal h, un juego que comienza con tal estructura de argumento será denotado $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$. Informalmente, en el contexto de la P.L.R., si el jugador que comienza el juego y que llamaremos Proponente gana un juego $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ cuya primera movida es una estructura de argumento para un literal h en un programa rebatible $\mathcal{P}$, entonces h pertenecerá al conjunto de literales garantizados de $\mathcal{P}$. Con el objeto de capturar esta semántica a través de juegos, definimos al conjunto de piezas $M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ como un conjunto de estructuras de argumentos. A partir de $Arg^h_{\mathcal{P}}$ especificaremos cuáles pueden ser posibles movimientos en un juego.

Definición 4.14 (Piezas del Juego)
Sea $\langle \mathcal{A}, h \rangle$ una estructura de argumento bajo un *plrb* $\mathcal{P}$. El conjunto de las piezas con las que tendremos permitido jugar en un juego cuyo primer movimiento es $\langle \mathcal{A}, h \rangle$, lo denotaremos $M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ y es un subconjunto de $Arg^h_{\mathcal{P}}$. Notaremos $M^*_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ al conjunto de todas las secuencias que se puedan obtener a partir de $M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ y $|s|$ a la longitud de una secuencia finita $s \in M^*_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$. ■

Definición 4.15 (Proyección)
Sea $s = \langle \mathcal{B}, l \rangle$ un elemento en $M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$. La proyección del primer elemento de s, que notaremos $s^{\mathcal{A}}$, es $\mathcal{B}$ y la proyección del segundo elemento de s es $s^h = l$. ■

Las piezas para jugar este juego son los argumentos bajo el *plrb*. Sin embargo, dependiendo de con que argumento comience el juego no todos los argumentos podrán ser jugados. Así hemos circunscripto el conjunto de piezas a jugar: $Arg^h_{\mathcal{P}}$.

No toda secuencia de estos argumentos será legal como posible parte del juego dialéctico por lo que deberemos caracterizar secuencias legales. Recordemos que cada línea argumentativa debe respetar algunas pautas respecto a las derrotas entre argumentos.

Definición 4.16 (Relación entre argumentos)
Sea $\texttt{Args}(\Gamma)$ el conjunto de todos los argumentos que se pueden obtener a partir de un sistema argumentativo Γ. Una *relación entre argumentos* $\mathcal{R} \subseteq \texttt{Args}(\Gamma) \times \texttt{Args}(\Gamma)$ es cualquier relación binaria definida sobre $\texttt{Args}(\Gamma)$. ■

Dados un sistema argumentativo Γ, dos estructuras de argumento $\langle \mathcal{A}_1, h_1\rangle$ y $\langle \mathcal{A}_2, h_2\rangle$ de $\texttt{Args}(\Gamma)$ y una relación binaria $\mathcal{R}$ entre argumentos de $\texttt{Args}(\Gamma)$, pueden presentarse las siguientes situaciones:

(a) $\langle \mathcal{A}_1, h_1\rangle \mathcal{R} \langle \mathcal{A}_2, h_2\rangle$ y $\langle \mathcal{A}_2, h_2\rangle \not\mathcal{R} \langle \mathcal{A}_1, h_1\rangle$,

(b) $\langle \mathcal{A}_2, h_2\rangle \mathcal{R} \langle \mathcal{A}_1, h_1\rangle$ y $\langle \mathcal{A}_1, h_1\rangle \not\mathcal{R} \langle \mathcal{A}_2, h_2\rangle$,

(c) $\langle \mathcal{A}_1, h_1\rangle$ y $\langle \mathcal{A}_2, h_2\rangle$ son incomparables bajo $\mathcal{R}$,i.e., $\langle \mathcal{A}_1, h_1\rangle \not\mathcal{R} \langle \mathcal{A}_2, h_2\rangle$ y $\langle \mathcal{A}_2, h_2\rangle \not\mathcal{R} \langle \mathcal{A}_1, h_1\rangle$, o bien

(d) $\langle \mathcal{A}_1, h_1\rangle \mathcal{R} \langle \mathcal{A}_2, h_2\rangle$ y $\langle \mathcal{A}_2, h_2\rangle \mathcal{R} \langle \mathcal{A}_1, h_1\rangle$

Distinguiremos dos relaciones binarias especiales sobre un orden de preferencia entre los argumentos y que luego nos permitirá caracterizar los derrotadores y finalmente, las secuencias legales.

Definición 4.17 [Cn01]**(Orden de Preferencia $\preceq$ - Relaciones Binarias $\prec$, $\succ$ y $\asymp$)**
Sean Γ un sistema argumentativo, $\texttt{Args}(\ \Gamma)$ el conjunto de todas las estructuras de argumentos que se pueden obtener a partir de Γ y $\{\texttt{A},\ \texttt{B}\} \subseteq \texttt{Args}(\Gamma)$. Un *orden de preferencia* $\preceq \subseteq \texttt{Args}(\Gamma) \times \texttt{Args}(\Gamma)$ es cualquier orden parcial definido sobre $\texttt{Args}(\Gamma)$. Diremos que:

1. $\texttt{A} \prec \texttt{B}$ o bien $\texttt{B} \succ \texttt{A}$ si $\texttt{A} \preceq \texttt{B}$ y $\texttt{B} \not\preceq \texttt{A}$. En este caso diremos que $\texttt{B}$ es estrictamente preferido por sobre $\texttt{A}$.
2. $\texttt{A} \asymp \texttt{B}$ si
   a) $\texttt{A} \preceq \texttt{B}$ y $\texttt{B} \preceq \texttt{A}$.
   b) $\texttt{A}$ y $\texttt{B}$ no son comparables por $\preceq$ (i.e., $\texttt{A} \not\preceq \texttt{B}$ y $\texttt{B} \not\preceq \texttt{A}$).

Escribiremos $\prec_{CP}$, $\preceq_{CP}$ y $\asymp_{CP}$ para denotar las relaciones binarias asociadas a un criterio de preferencia CP determinado. ■

Deberemos determinar que secuencia de estructuras argumentativas está permitida. En este aspecto y al desear capturar la idea de una discusión requeriremos que se presente una serie alternada de argumentos y contraargumentos.

Definición 4.18 (Derrota entre elementos de $M_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$)
Sean $\mathcal{P} = (\Pi, \Delta)$ un *plrb*, h un literal, $\langle \mathcal{A}, h\rangle$ una estructura de argumento para h bajo $\mathcal{P}$ y $s_i,\ s_j \in M_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$. Diremos que s_j derrota a s_i o que s_j es un derrotador de s_i, si existe $s' \in M_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$ tal que ${s'}^{\mathcal{A}} \subseteq {s_i}^{\mathcal{A}}$ y $\{{s'}^{h}, {s_j}^{h}\} \cup \Pi$ es inconsistente, y

1. $s_j \succ s'$, en cuyo caso diremos que s_j es un *derrotador propio* para s' y de s_i ; o bien
2. $s_j \asymp s'$, en cuyo caso diremos que s_j es un *derrotador por bloqueo* para s' y por lo tanto, de s_i .

■

Existen algunas falacias que deberemos tener en cuenta por lo que necesitaremos distinguir entre todas las secuencias aquellas que son legales en nuestro juego.

No siempre las discusiones entre dos agentes son correctas. En el proceso dialéctico es posible encontrar fallas que invalidan la discusión. Diremos entonces que tal proceso contienen falacias. Petitio Principii y Argumentum non sequitur en sus formas "stolen concept" y "inconsistency" son las falacias que pueden presentarse en la P.L.R.B.[MG99].

Stolen concept usa un concepto para atacar un concepto del cual depende lógicamente [MG99]. Petitio Principii [MG99] se puede presentar de dos formas diferentes : ofreciendo como premisa una simple redefinición de la conclusión deseada o bien justificando la conclusión con la conclusión misma.

En el proceso dialéctico también puede presentarse la falacia Argumentum non sequitur en su forma inconsistency, que involucra el sustento de afirmaciones contrarias o contradictorias como verdaderas [MG99]. Por lo tanto, necesitaremos detectar tales falacias en nuestros juegos.

Definición 4.19 (Secuencia legal)
Sean $\langle \Pi, \Delta\rangle$ un programa lógico rebatible y $s \in M^{*}_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$ una secuencia finita, con $|s| = n$. El conjunto de todos los movimientos legales para esa secuencia está definido a través de la función $legal : M^{*}_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})} \rightarrow 2^{M_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}}$. Una estructura de argumento s_{n+1} bajo $\mathcal{P}$ pertenece a $legal(s)$ si y sólo si se cumple lo siguiente:

(a) La secuencia s es vacía, entonces s_{n+1} es el primer movimiento en el juego, i.e., $legal(\epsilon) = \{\langle \mathcal{A}, h\rangle\}$.

(b) La secuencia s no es vacía, luego

   a) s_{n+1} es un derrotador de s_n y si $s_{n-1} \approx s_n$, luego $s_{n+1} \succ s_n$.

b) $1 \leq i \leq |s|+1$

$$\Pi \cup \bigcup_{even(i)} s_i^{\mathcal{A}} \not\models \bot \text{ and } \Pi \cup \bigcup_{odd(i)} s_i^{\mathcal{A}} \not\models \bot$$

c) $s_{n+1}^{\mathcal{A}} \not\subseteq s_i^{\mathcal{A}},\ 1 \leq i \leq |s|$.

La secuencia s, no vacía, tal que $|s| = 1$, es una secuencia legal. La secuencia finita $s[s_{n+1}]$ es legal si s es legal y $s_{n+1} \in legal(s)$. ■

En una secuencia legal cada jugada deberá derrotar a la anterior, reflejando un debate en donde los argumentos introducidos tratarán de anular al argumento presentado inmediatamente antes.

El punto (b) de la definición anterior no permitirá que un jugador contradiga sus propios dichos en el debate. En otras palabras, no se aceptarán situaciones en las que un participante argumente por A y luego argumente por el complemento de A, evitando de este modo, las inconsistencia entre las jugadas del mismo jugador.

El punto (c) no permite reintroducir en el debate subargumentos de argumentos ya argüidos. Así si un argumento fue derrotado en la línea de debate no es posible que un subargumento suyo pudiera aparecer en esa misma línea.

Con toda esta artillería estamos en condiciones de definir un juego.

4.4.4. Juego: Reglas y Turnos

Serán las reglas que gobiernan el juego las encargadas de determinar cuáles movimientos estarán permitidos, teniendo en cuenta a las secuencias legales, y quién es el jugador que tiene permitido jugar.

Definición 4.20 (Juego)
Sea $\mathcal{P} = (\Pi, \Delta)$ un programa lógico rebatible, h un literal y $\langle \mathcal{A}, h \rangle$ una estructura de argumento para h bajo $\mathcal{P}$. Un *juego* para $\langle \mathcal{A}, h \rangle$ con respecto a $\mathcal{P}$, que notaremos $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$, es una estructura $(M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}, J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}, P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})})$, donde

- $M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})} \subseteq Arg(h, \mathcal{P})$.
- $J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})} :\ M_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})} \times I \rightarrow Jugadores$ donde I es un índice enumerable;
- $P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})} \subseteq M^*_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$, donde $P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ es no vacío y cerrado con respecto a los prefijos de sus secuencias.

  Cada secuencia s de $P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ satisface:

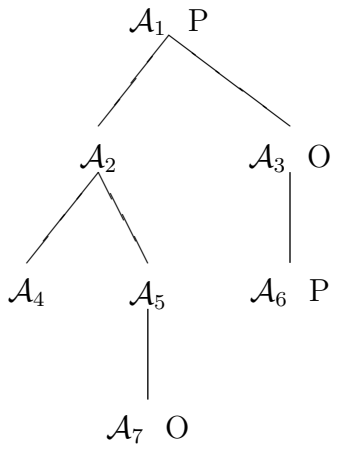


Figura 4.1: Árbol dialéctico del ejemplo 4.6, junto con el participante que juega.

1. $s = \epsilon$ o bien $s = [\langle \mathcal{A}, h \rangle]s'$, con s' posiblemente vacía.
2. Para todo $i,\ 1 \leq i \leq |s|$

$$\begin{array}{rcl} J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}(s_1, 1) & = & \mathit{Jugador\_Inicial} \\ J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}(s_i, i) & = & \overline{J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}(s_{i-1}, i-1)} \end{array}$$

3. Si $s = s'[\langle \mathcal{A}_1, h_1 \rangle] \in P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ donde s' es una secuencia posiblemente vacía, entonces para cada estructura de argumento $\langle \mathcal{A}_2, h_2 \rangle \in \mathit{legal}(s)$, existe una secuencia $t \in P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$ tal que $t = s[\langle \mathcal{A}_2, h_2 \rangle] = s'[\langle \mathcal{A}_1, h_1 \rangle, \langle \mathcal{A}_2, h_2 \rangle]$.
4. Ninguna otra secuencia pertenece a $P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$.

■

Ejemplo 4.6 Sea $\mathcal{P}$ un *plrb*. Consideremos el siguiente conjunto de argumentos $\{\mathcal{A}_1, \ldots, \mathcal{A}_{12}\}$ sobre $\mathcal{P}$. Los pares de argumentos que se derrotan están relacionados del siguiente modo:

$$\begin{array}{lllll} \mathcal{A}_2 \prec \mathcal{A}_1 & \mathcal{A}_3 \prec \mathcal{A}_6 & \mathcal{A}_{11} \prec \mathcal{A}_{12} & \mathcal{A}_5 \prec \mathcal{A}_7 & \mathcal{A}_{10} \prec \mathcal{A}_7 \\ \mathcal{A}_3 \asymp \mathcal{A}_1 & \mathcal{A}_2 \asymp \mathcal{A}_4 & \mathcal{A}_2 \asymp \mathcal{A}_5 & \mathcal{A}_4 \asymp \mathcal{A}_8 & \mathcal{A}_9 \asymp \mathcal{A}_5 \end{array}$$

El resto de los pares cumple $\mathcal{A}_i \asymp \mathcal{A}_j$ pues son incomparables entre sí y por otra parte no son derrotadores entre ellos. Gráficamente, la relación definida anteriormente se puede ver en el árbol dialéctico de la Figura 4.1.

Definamos el juego $G(\mathcal{A}_1, \mathcal{P})$.

$$M_{G(\mathcal{A}_1, \mathcal{P})} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_7\}$$

$J_{G(\mathcal{A}_1, \mathcal{P})}$ se define como sigue:

$J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_1,1) = P \quad J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_2,2) = O \quad J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_3,2) = O \quad J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_4,3) = P$
$J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_5,3) = P \quad J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_6,3) = P \quad J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_7,4) = O$

Definimos $P_{G(\mathcal{A}_1,\mathcal{P})}$ a partir de *legal*.

$legal(\mathcal{A}_1) = \{\mathcal{A}_2, \mathcal{A}_3\} \quad legal(\mathcal{A}_1\mathcal{A}_2) = \{\mathcal{A}_4, \mathcal{A}_5\} \quad legal(\mathcal{A}_1\mathcal{A}_2\mathcal{A}_4) = \{\ \}$
$legal(\mathcal{A}_1\mathcal{A}_2\mathcal{A}_5) = \{\mathcal{A}_7\} \quad legal(\mathcal{A}_1\mathcal{A}_2\mathcal{A}_5\mathcal{A}_7) = \{\ \}$
$legal(\mathcal{A}_1\mathcal{A}_3) = \{\mathcal{A}_6\} \quad legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_6) = \{\ \}$

$$P_{G(\mathcal{A}_1,\mathcal{P})} = \{\epsilon,\ \mathcal{A}_1,\ \mathcal{A}_1\mathcal{A}_2,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_4,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_5,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_5\mathcal{A}_7,\ \mathcal{A}_1\mathcal{A}_3,\ \mathcal{A}_1\mathcal{A}_3\mathcal{A}_6\}$$

■

A diferencia de la definición original de juego[AM97], la primer movida en el juego $G(\langle\mathcal{A}, h\rangle, \mathcal{P})$ es siempre hecha por el Jugador_Inicial que en nuestro caso es el proponente. En forma alternada juegan el Proponente y el Oponente. La función $J_{G(\langle\mathcal{A},\, h\rangle,\, \mathcal{P})}$ determina que jugador jugó un movimiento en una posición dada en la secuencia. Ya que la misma estructura de argumento puede ser jugada tanto por el proponente como por el oponente, siempre y cuando sean legales, este mapeo tiene dos argumentos a diferencia de la definición original. El segundo argumento de $J_{G(\langle\mathcal{A},\, h\rangle,\, \mathcal{P})}$ corresponde a la profundidad en el árbol que se genera. Veamos un caso que ejemplifica esta situación.

Ejemplo 4.7 Consideremos el siguiente programa:

$$\Pi = \left\{ \begin{array}{l} c. \\ g. \\ j. \\ p. \end{array} \right\} \qquad \Delta = \left\{ \begin{array}{rcl} a & \prec & b \\ b & \prec & c \\ \sim b & \prec & c, f \\ \sim b & \prec & c, r \\ f & \prec & g \\ \sim f & \prec & g, h, c, p \\ h & \prec & j \\ r & \prec & p \\ \sim r & \prec & p, f, c \end{array} \right\}$$

En la figura 4.2 se muestra el árbol de dialéctica para la consulta `a`. Los argumentos son:

$\mathcal{A}_1 = \langle(a \prec b), (b \prec c)\rangle$
$\mathcal{B}_1 = \langle(\sim\ b \prec c, f), (f \prec g)\rangle$
$\mathcal{B}_2 = \langle(\sim\ b \prec c, r), (r \prec p)\rangle$
$\mathcal{C}_1 = \langle(\sim\ f \prec g, h, c, p), (h \prec j)\rangle$
$\mathcal{C}_2 = \langle(\sim\ r \prec p, f, c), (f \prec g)\rangle$

El argumento $\mathcal{C}_1$ es jugado tanto por el proponente como por el oponente en diferentes líneas de argumentación. ■

Definición 4.21 (Turno)
Sea s una secuencia de movimientos en un juego $G(\langle\mathcal{A}, h\rangle, \mathcal{P})$. El jugador que tiene el *turno* en

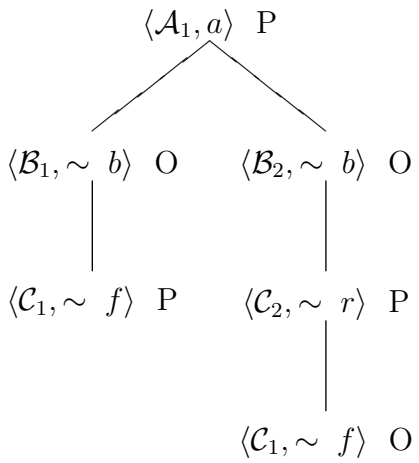


Figura 4.2: Árbol dialéctico para la estructura de argumento $\langle \mathcal{A}_1, a \rangle$ del ejemplo 4.7, junto con el participante que juega.

la posición s_i es aquel a quien le toca jugar cuando el movimiento de $J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}(s_i, i)$ ha sido completado y se define como $\overline{J_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}(s_i, i)}$. ■

Dependiendo del jugador, cada movida captura un argumento de soporte o de interferencia con respecto al argumento inicial. De todas las posibles secuencias, nosotros estamos interesados solamente en secuencias de movimientos de cierta clase: aquellas que caracterizan la construcción del árbol dialéctico. Por esta razón se han impuesto tres condiciones en la definición 4.20. Teniendo en cuenta solamente las secuencias en las que los jugadores deberán jugar en forma alternada, el punto (1) indica que el juego contiene a la secuencia vacía, esto se explica por el hecho de que se requiere que sea cerrado bajo los prefijos, o bien comienza con la estructura de argumento $\langle \mathcal{A}, h \rangle$. El siguiente inciso condiciona los movimientos a argumentos que se ataquen, i.e., cada movimiento es un contraargumento que derrota al argumento de la movida precedente. Asimismo, se exige que cada juego contemple todos los derrotadores de cada argumento jugado. En ambos casos, consideraremos que sean movimientos legales en el juego. Por último, se condiciona a ninguna otra secuencia que no cumpla (1) y (2) puede pertenecer a $P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$.

Ejemplo 4.8 Considere el ejemplo 6. El conjunto de las secuencias de este juego es:

$$P_{G(\mathcal{A}_1, \mathcal{P})} = \{\epsilon,\ \mathcal{A}_1,\ \mathcal{A}_1\mathcal{A}_2,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_4,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_5,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_5\mathcal{A}_7,\ \mathcal{A}_1\mathcal{A}_3,\ \mathcal{A}_1\mathcal{A}_3\mathcal{A}_6\}$$

ϵ pertenece ya que es cerrado bajo los prefijos. Analisemos la secuencia $\mathcal{A}_1\mathcal{A}_3\mathcal{A}_6$. Ésta cumple las condiciones dadas, ya que $\mathcal{A}_3$ pertenece a $legal(\mathcal{A}_1)$, es decir ataca al argumento $\mathcal{A}_1$, pero no es mejor según la relación de preferencia (derrotador por bloqueo). Así $\mathcal{A}_6$ que ataca a $\mathcal{A}_3$ debe ser preferido según la relación de preferencia.

Todas las secuencias que pertenecen consideran a la función legal y la forma en que se relacionan los argumentos según la preferencia entre ellos. ■

4.4.5. Vencedores

Una vez definido el juego con sus reglas, necesitaremos determinar quién es el vencedor. Con este objetivo en mente, daremos algunas definiciones preliminares, entre la que se encuentra la idea de estrategia para el juego.

Definición 4.22 (Secuencia Completa - Secuencia Preferida)
Una secuencia s es completa si $legal(s) = \varnothing$. Una secuencia s' se dice preferida si $|s'|$ es impar. ■

Una secuencia completa es una línea de argumentación, i.e. una camino desde el primer movimiento hasta un movimiento que nos permita alcanzar una hoja. Una secuencia preferida captura la idea de que dicha secuencia termina con un movimiento del proponente, es decir, en toda secuencia preferida cada movimiento del oponente tiene una respuesta del proponente.

En este punto estaremos interesados en conocer cuáles debates son ganados por el proponente. Considerar a las secuencias preferidas es una estrategia ganadora simple, ya que en estas secuencias el último movimiento es hecho por el proponente y deja sin posibilidades de movimiento al oponente. Sin embargo, aún cuando existan algunas disputas cuyo último movimiento es hecho por el oponente, el debate completo podría ser ganado por el proponente. La siguiente definición caracteriza estos casos.

Definición 4.23 (Estrategia)
Una estrategia σ sobre un juego G es un conjunto de secuencias S, tal que para toda secuencia $s \in S$:

- s es preferida; o
- existe otra secuencia $s' \in S$, s' es preferida, tal que si $s = [s_1, \ldots, s_n]$, entonces $s' = [s_1, \ldots, s_i, s'_{i+1}, s'_{i+2}, \ldots, s'_r]$ con $1 \leq i \leq n$ e i par, es decir s y s' difieren a partir de una posición impar.

■

Ejemplo 4.9 Considere la figura 4.3. Las ramas preferidas, es decir aquellas que son impares terminan con argumento del Proponente, así la discusión se da por ganada Las ramas encerradas en los cuadrados son irrelevantes para determinar si A_1 está derrotada. Esto tiene su razón en que si uno de los nodos hijos del nodo que vamos a etiquetar está etiquetado con U, el análisis de las otras ramas es irrelevante al resultado: el nodo padre estará etiquetado con

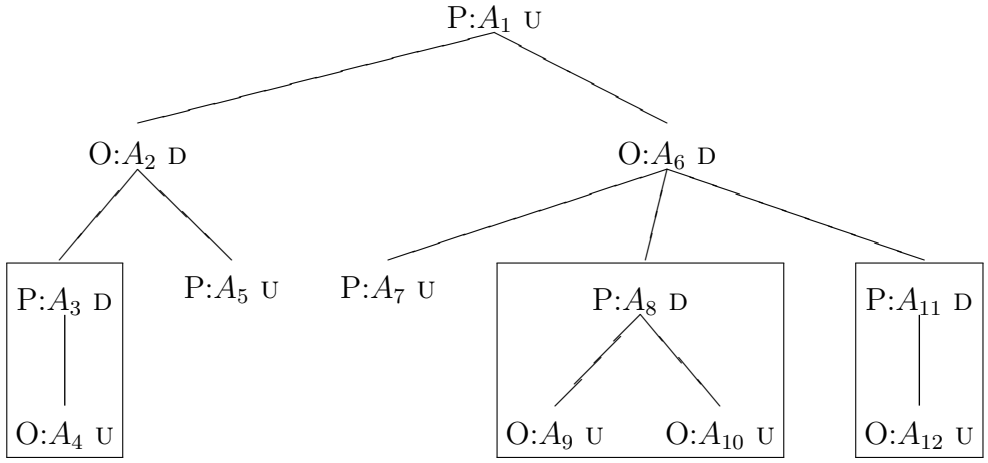


Figura 4.3: Árbol dialéctico con argumentos irrelevantes. Ejemplo 4.9.

D. Note que todas las secuencias irrelevantes difieren en una posición impar de la secuencia preferida.
Por ejemplo, la secuencia $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_8\mathcal{A}_9$ no es preferida, sin embargo existe $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_7$ que es preferida (y por lo tanto está etiquetada con U) y que difiere de la primer secuencia en la posición impar 3. ■

Deseamos caracterizar en forma declarativa al árbol dialéctico. Para esto tendremos en cuenta a la secuencias que sean completas, i.e., cada una de las ramas del árbol. La teoría de prueba realiza un etiquetado sobre el árbol para determinar si el argumento raíz es consecuencia del programa lógico. Un conjunto de secuencias que formen una estrategia caracterizará un árbol dialéctico cuya raíz no haya sido derrotada. Para esto, analizaremos todas las secuencias *legales completas* que conforman el juego. Si dichas secuencias forman una estrategia entonces diremos que el juego fue ganado por el proponente.

Definición 4.24 (Criterio de Triunfo)
Sea $\mathcal{P}$ un *plrb*, $h \in Lit$ y $G(\langle \mathcal{A}, h\rangle, \mathcal{P}) \in \mathcal{F}(h, \mathcal{P})$. Diremos que *P gana el juego* $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ o que $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ *es ganado por P*, si el conjunto de las secuencias completas de $P_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$ es una estrategia. De otro modo, diremos que *O gana el juego* o que $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ *es ganado por O*. ■

Con el objeto de mejorar la eficiencia de la teoría de prueba de P.L.R. se analiza parcialmente el árbol dialéctico cuando una respuesta es requerida. Tales árboles son construidos podando las ramas irrelevantes [GS04], basados en el algoritmo de poda α-β. Prakken [PV00] también determina que no todos los argumentos jugados en un juego son relevantes. Caracterizamos tales secuencias irrelevantes a través de la noción de estrategia. Las secuencias completas que

terminan con un movimiento del oponente y que difieren en una posición impar de otra secuencia completa cuyo último movimiento fue hecho por el proponente es irrelevante. Estas ideas son el reflejo de la noción de que si un argumento está no derrotado entonces su padre está derrotado.

Ejemplo 4.10 Consideremos el juego que se muestra gráficamente en la Figura 4.3. En el primer subárbol la jugada del proponente con $\mathcal{A}_5$, que no está derrotada, permite determinar que el padre está derrotado, sin importar como serán el resto de las jugadas. Está información es la que se aprovecha para, en forma operacional podar el árbol, y en forma declarativa determinar si el juego es ganado por el proponente.

El conjunto de las secuencias completas en este juego son:

(S1) $\mathcal{A}_1\mathcal{A}_2\mathcal{A}_3\mathcal{A}_4$

(S2) $\mathcal{A}_1\mathcal{A}_2\mathcal{A}_5$

(S3) $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_7$

(S4) $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_8\mathcal{A}_9$

(S5) $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_8\mathcal{A}_9$

(S6) $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_8\mathcal{A}_{10}$

(S7) $\mathcal{A}_1\mathcal{A}_6\mathcal{A}_{11}\mathcal{A}_{12}$

Las secuencias (S2) y (S3) son preferidas. La secuencia (S1) no es preferida pero existe la secuencia (S2), preferida, que difiere de (S1) en la posición impar 3.

Las secuencias (S4),(S5) y (S6) tampoco son preferidas, pero todas difieren en la posición impar 3 de la secuencia (S3), que es preferida.

Así el conjunto de las secuencias completas de este juego es una estrategia y por lo tanto, según el criterio de triunfo, el juego es ganado por el proponente. ■

Ejemplo 4.11 Consideremos el siguiente juego $G(\mathcal{A}_1, \mathcal{P})$, donde:

$$M_{G(\mathcal{A}_1,\mathcal{P})} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{15}\}$$

$J_{G(\mathcal{A}_1,\mathcal{P})}$ se define como sigue:

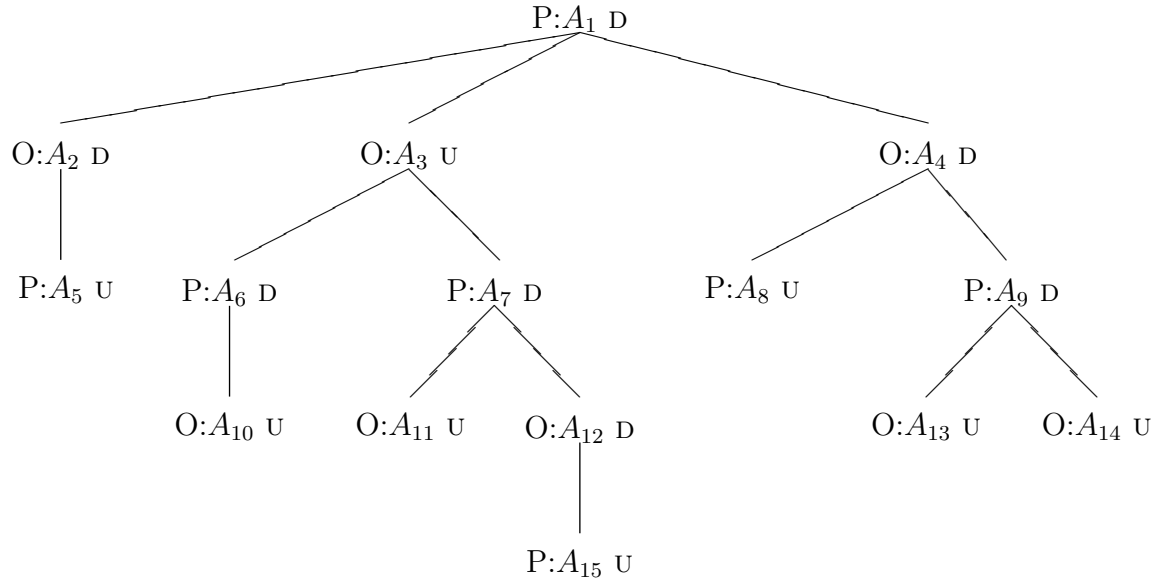


Figura 4.4: Árbol dialéctico ganado por Oponente.

$J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_1,1) = P$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_2,2) = O$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_3,2) = O$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_4,2) = O$
$J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_5,3) = P$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_6,3) = P$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_7,3) = P$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_8,3) = P$
$J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_9,3) = P$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_{10},4) = O$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_{11},4) = O$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_{12},4) = O$
$J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_{13},4) = O$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_{14},4) = O$ $J_{G(\mathcal{A}_1,\mathcal{P})}(\mathcal{A}_{15},5) = P$

Definimos $P_{G(\mathcal{A}_1,\mathcal{P})}$ a partir de $legal$.

$legal(\mathcal{A}_1) = \{\mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4\}$ $legal(\mathcal{A}_1\mathcal{A}_2) = \{\mathcal{A}_5\}$ $legal(\mathcal{A}_1\mathcal{A}_2\mathcal{A}_5) = \{\ \}$
$legal(\mathcal{A}_1\mathcal{A}_3) = \{\mathcal{A}_6, \mathcal{A}_7\}$ $legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_6) = \{\mathcal{A}_{10}\}$ $legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_6\mathcal{A}_{10}) = \{\ \}$
$legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7) = \{\mathcal{A}_{11}, \mathcal{A}_{12}\}$ $legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{11}) = \{\ \}$ $legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{12}) = \{\mathcal{A}_{15}\}$
$legal(\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{12}\mathcal{A}_{15}) = \{\ \}$ $legal(\mathcal{A}_1\mathcal{A}_4) = \{\mathcal{A}_8, \mathcal{A}_9\}$ $legal(\mathcal{A}_1\mathcal{A}_4\mathcal{A}_8) = \{\ \}$
$legal(\mathcal{A}_1\mathcal{A}_4\mathcal{A}_9) = \{\mathcal{A}_{13}, \mathcal{A}_{14}\}$ $legal(\mathcal{A}_1\mathcal{A}_4\mathcal{A}_9\mathcal{A}_{13}) = \{\ \}$ $legal(\mathcal{A}_1\mathcal{A}_4\mathcal{A}_9\mathcal{A}_{14}) = \{\ \}$

$P_{G(\mathcal{A}_1,\mathcal{P})} = \{\epsilon,\ \mathcal{A}_1,\ \mathcal{A}_1\mathcal{A}_2,\ \mathcal{A}_1\mathcal{A}_2\mathcal{A}_5,\ \mathcal{A}_1\mathcal{A}_3,\ \mathcal{A}_1\mathcal{A}_3\mathcal{A}_6,\ \mathcal{A}_1\mathcal{A}_3\mathcal{A}_6\mathcal{A}_{10},\ \mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{11},\ \mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{12},$
$\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{12}\mathcal{A}_{15},\ \mathcal{A}_1\mathcal{A}_4,\ \mathcal{A}_1\mathcal{A}_4\mathcal{A}_8,\ \mathcal{A}_1\mathcal{A}_4\mathcal{A}_9,\ \mathcal{A}_1\mathcal{A}_4\mathcal{A}_9\mathcal{A}_{13}\ \mathcal{A}_1\mathcal{A}_4\mathcal{A}_9\mathcal{A}_{14}\}$

Analicemos las secuencias completas de este juego(ver Figura 4.4):

(S1) $\mathcal{A}_1\mathcal{A}_2\mathcal{A}_5$

(S2) $\mathcal{A}_1\mathcal{A}_3\mathcal{A}_6\mathcal{A}_{10}$

(S3) $\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{11}$

(S4) $\mathcal{A}_1\mathcal{A}_3\mathcal{A}_7\mathcal{A}_{12}\mathcal{A}_{15}$

(S5) $\mathcal{A}_1\mathcal{A}_4\mathcal{A}_8$

(S6) $\mathcal{A}_1\mathcal{A}_4\mathcal{A}_9\mathcal{A}_{13}$

(S7) $\mathcal{A}_1\mathcal{A}_4\mathcal{A}_9\mathcal{A}_{14}$

Las secuencias (S1), (S4) y (S5) son preferidas. (S2) no es preferida, pero existe (S4) que es preferida y que difiere en posición impar 3. (S3) no es preferida y no existe ninguna secuencia preferida que difiera de ella en posición impar ((S1) difieren en posición 2, (S4) en posición 4 y (S5) en posición 2).

Luego las secuencias completas no son una estrategia y por lo tanto el juego no es ganado por el proponente. ■

4.4.6. Semántica GS

Dado un literal h, el proponente podría jugar como movimiento inicial, cualquiera de los argumentos que soporta a h. De este modo, tendríamos para cada literal un juego por cada argumento que lo soporte. Esto genera una familia de juegos por cada literal.

Definición 4.25 (Familia de Juegos)
Sean $\mathcal{P}$ un *plrb*, h un literal bajo la signatura de $\mathcal{P}$, $\langle\mathcal{A}_1, h\rangle, \ \ldots\ , \langle\mathcal{A}_n, h\rangle$ todos las estructuras de argumento de h bajo $\mathcal{P}$ y

$$G(\langle\mathcal{A}_1, h\rangle, \mathcal{P}), G(\langle\mathcal{A}_2, h\rangle, \mathcal{P}), \ \ldots\ , G(\langle\mathcal{A}_n, h\rangle, \mathcal{P})$$

los juegos correspondientes a todos los argumentos de h. Diremos que

$$\{G(\langle\mathcal{A}_1, h\rangle, \mathcal{P}), G(\langle\mathcal{A}_2, h\rangle, \mathcal{P}), \ \ldots\ , G(\langle\mathcal{A}_n, h\rangle, \mathcal{P})\}$$

es la familia de juegos de h y la notaremos $\mathcal{F}(h, \mathcal{P})$. ■

Definición 4.26 (G-Interpretación)
Sea $\mathcal{P}$ un *plrb*. Una *interpretación basada en juegos para* $\mathcal{P}$, que llamaremos G-Interpretación para $\mathcal{P}$, es una tupla $\langle V, F\rangle$, tal que V y F son subconjunto de átomos bajo la signatura de $\mathcal{P}$ y $V \cap F = \varnothing$. ■

Intuitivamente, el conjunto V contendrá a los átomos garantizados ($h \in V$, creemos en h, Bh) bajo esa interpretación y ($h \in F$, creemos en $\overline{h}$, $B\overline{h}$) contendrá a aquellos átomos que cuya negación está garantizada bajo esa interpretación. Asumimos que el conjunto de átomos INDECISOS está definido como $Lit^+ - \{V \cup F\}$, siendo Lit^+ el conjunto de todos átomos de Lit. Nótese que la definición anterior no contempla la respuesta DESCONOCIDO, ya que sólo consideramos literales bajo la signatura del *plrb*.

Observación 4.1 *La intersección de G-interpretaciones es una G-interpretación. La unión de G-interpretaciones no es una G-interpretacion.*

Antes de definir un G-modelo analicemos la siguiente situación:

Ejemplo 4.12 Consideremos el siguiente *plrb*:

$$\Pi = \left\{ \begin{array}{l} \texttt{b.} \\ \sim\texttt{c.} \\ \sim\texttt{d} \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} \texttt{a} \dashrightarrow \texttt{b} \\ \sim\ \texttt{a} \dashrightarrow \texttt{e,b} \end{array} \right\}$$

Una G-interpretación "adecuada" a este programa sería $\langle\{b, a\}\{c, d\}\rangle$. Otras G-interpretaciones que también podrían ser "adecuadas" serían:

- aquella que considera a e falso: $\langle\{b, a\}\{c, d, e\}\rangle$ y que no afecta a los otros resultados; y
- aquella que considera a e verdadero. Si asumimos e verdadero el programa tendría otra interpretación(debido al caracter no monotónico del sistema), ya que ahora a tiene un contraargumento que lo derrota y, por lo tanto, deja de ser verdadero. Así, asumiendo e verdadero, la G-interpretación más adecuada sería: $\langle\{b, e\}\{a, c, d\}\rangle$.

De las tres G-interpretaciones, la que se ajusta a la teoría de prueba de la P.L.R. es la primera. ■

Definición 4.27 (G-Modelo)
Sean $\mathcal{P}$ un *plrb*, h un átomo bajo la signatura de $\mathcal{P}$, $\mathcal{F}(h, \mathcal{P})$ la familia de juegos de h y $\mathcal{F}(\overline{h}, \mathcal{P})$ la familia de juegos del literal $\overline{h}$ para el *plrb* $\mathcal{P}$. Un *modelo basado en juegos para* $\mathcal{P}$, que llamaremos G-Modelo de $\mathcal{P}$, es una G-interpretación $\langle V, F\rangle$ tal que:

- h pertenece a V si y solamente si existe un juego $G(\langle\mathcal{A}, h\rangle, \mathcal{P})$ en la familia $\mathcal{F}(h, \mathcal{P})$ ganado por P.
- h pertenece a F si y solamente si existe un juego $G(\langle\mathcal{A}, \overline{h}\rangle, \mathcal{P})$ en la familia $\mathcal{F}(\overline{h}, \mathcal{P})$ ganado por P.
- Ningún otro átomo pertenece al G-modelo.

■

Teorema 4.4 *Sea $\mathcal{P}$ un plrb. El G-modelo de $\mathcal{P}$ es único.*

Demostración: Sea $\mathcal{P}$un *plrb*. Supongamos que existen dos G-modelos G1 y G2. Luego existe un literal h en la asignatura de $\mathcal{P}$ tal que h pertenece a uno de los G-modelos y no pertenece al otro. Supongamos que sea el caso de que $h \in G1$ y $h \notin G2$.
Por definición, si $h \in G1$ entonces existe un juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P}) \in \mathcal{F}(h, \mathcal{P})$ ganado por el proponente.
Por otro lado, por definición de G-modelo si $h \notin G2$ entonces no existe ningún juego ganado por el proponente en su familia.

Contradicción. Dicha contradicción provino de suponer que existe el literal h. Luego $G1 = G2$. ■

Definición 4.28 (Semántica GS)
Sea $\mathcal{P}$ un *plrb*. La *semántica basada en juegos para* $\mathcal{P}$, que notaremos $\mathcal{GS}$, es el único G-modelo de $\mathcal{P}$. Diremos que un literal h es consecuencia de $\mathcal{GS}$, y lo notaremos $\mathcal{P} \models_{\mathcal{GS}} h$, cuando $h \in T$ o $\overline{h} \in F$, siendo $\langle T, F\rangle$ el único G-modelo de $\mathcal{P}$. ■

Ejemplo 4.13 Consideremos la representación del problema en el que Nixon está políticamente motivado.

$$\Pi = \left\{ \begin{array}{l} \sim \texttt{paloma(n)} \leftarrow \texttt{halcón(n).} \\ \sim \texttt{halcón(n)} \leftarrow \texttt{paloma(n).} \\ \texttt{cuáquero(n).} \\ \texttt{republicano(n)} \end{array} \right\}$$

$$\Delta = \left\{ \begin{array}{l} \texttt{polit\_motivado(n)} \dashrightarrow \texttt{paloma(n)} \\ \texttt{polit\_motivado(n)} \dashrightarrow \texttt{halcón(n)} \\ \texttt{paloma(n)} \dashrightarrow \texttt{cuáquero(n)} \\ \texttt{halcón(n)} \dashrightarrow \texttt{republicano(n)} \end{array} \right\}$$

En lo que sigue utilizaremos las iniciales de los literales.

Políticamente Motivado

$$\mathcal{F}(pm(n), \mathcal{P}) = \{G1(\langle pm(n) \prec p(n); p(n) \prec c(n), pm(n)\rangle, \mathcal{P}),$$
$$G2(\langle pm(n) \prec h(n); h(n) \prec r(n), pm(n)\rangle, \mathcal{P})\}$$

Definimos al juego $G1(\langle pm(n) \prec p(n); p(n) \prec c(n), pm(n)\rangle, \mathcal{P})$:

$$M_{G1} = \{\langle \{pm(n) \prec p(n); p(n) \prec c(n)\}, pm\rangle, \langle \{h(n) \prec r(n)\}, \sim p(n)\rangle\}$$

J_{G1} se define como sigue:

$$J_{G1}(\langle\{pm(n) -\!\prec p(n); p(n) -\!\prec c(n)\}, pm\rangle, 1) = P \quad J_{G1}(\langle\{h(n) -\!\prec r(n)\}, \sim p(n)\rangle, 2) = O$$

Definimos P_{G1} a partir de *legal*.

$$legal(\langle\{pm(n) -\!\prec p(n); p(n) -\!\prec c(n)\}, pm\rangle) = \{\langle\{h(n) -\!\prec r(n)\}, \sim p(n)\rangle\}$$
$$legal(\langle\{pm(n) -\!\prec p(n); p(n) -\!\prec c(n)\}, pm\rangle\langle\{h(n) -\!\prec r(n)\}, \sim p(n)\rangle) = \{\ \}$$

$$P_{G1} = \{\epsilon,\ \langle\{pm(n) -\!\prec p(n); p(n) -\!\prec c(n)\}, pm\rangle,$$
$$\langle\{pm(n) -\!\prec p(n); p(n) -\!\prec c(n)\}, pm\rangle\langle\{h(n) -\!\prec r(n)\}, \sim p(n)\rangle\}$$

$G2$ es similar a $G1$. Ni $G1$ ni $G2$ son juegos ganados por el proponente. Así políticamente_motivado(n) no está garantizado y por lo tanto no pertenece a V del G-modelo. Dado que no hay argumentos para $\sim pm(n)$, luego tampoco creemos en $\sim$ políticamente_motivado(n).

Halcón

$\mathcal{F}(h(n), \mathcal{P}) = \{G3(\langle h(n) -\!\prec r(n), h(n)\rangle, \mathcal{P})\}$

Definimos al juego $G3(\langle h(n) -\!\prec r(n), h(n)\rangle, \mathcal{P})$:

$$M_{G3} = \{\langle\{h(n) -\!\prec r(n)\}, h(n)\rangle, \langle\{p(n) -\!\prec c(n)\}, \sim h(n)\rangle\}$$

J_{G3} se define como sigue:

$$J_{G3}(\langle\{h(n) -\!\prec r(n)\}, h(n)\rangle, 1) = P \quad J_{G3}(\langle\{p(n) -\!\prec c(n)\}, \sim h(n)\rangle, 2) = O$$

Definimos P_{G3} a partir de *legal*.

$$legal(\langle\{h(n) -\!\prec r(n)\}, h(n)\rangle) = \{\langle\{p(n) -\!\prec c(n)\}, \sim h(n)\rangle\}$$
$$legal(\langle\{h(n) -\!\prec r(n)\}, h(n)\rangle\langle\{p(n) -\!\prec c(n)\}, \sim h(n)\rangle\}) = \{\ \}$$

$$P_{G3} = \{\epsilon,\ \langle\{h(n) -\!\prec r(n)\}, h(n)\rangle,\ \langle\{h(n) -\!\prec r(n)\}, h(n)\rangle\langle\{p(n) -\!\prec c(n)\}, \sim h(n)\rangle\}$$

El único juego para $h(n)$ no es ganado por el proponente, luego $h(n)$ no pertenece a V. El único juego para $\sim (h(n))$ tampoco es ganado por el proponente. Luego $h(n)$ tampoco pertenece a F.

En forma similar podemos definir el juego para $paloma(n)$, notando que el único juego no es ganado por el proponente. Tampoco el juego para $\sim paloma(n)$ es ganado por el proponente.

Finalmente, los únicos juegos ganados por el proponente son $G4(\langle\varnothing, r(n)\rangle, \mathcal{P})$ y $G5(\langle\varnothing, c(n)\rangle, \mathcal{P})$.

Por lo tanto, el G-modelo de $\mathcal{P}$ es $\langle\{r, c\}, \varnothing\rangle$. ■

Dada la similitud entre un juego de dos participantes y un árbol dialéctico, $\mathcal{GS}$ permite vincular a los juegos con la teoría de modelos mostrando de un modo natural las relaciones

sintácticas entre los argumentos. A partir de $\mathcal{GS}$ no sólo caracterizamos declarativamente a la P.L.R.B., sino que podemos distinguir las diferentes formas de garantizar un literal a través de la familia de juegos.

4.4.7. Sensatez y Completitud entre $\mathcal{GS}$ y la Teoría de Prueba

Esta sección permite establecer el resultado más importante ya que demuestra que la semántica desarrollada es sensata y completa con respecto a la teoría de prueba.

Daremos algunas definiciones que caracterizan la equivalencia entre un árbol dialéctico y un juego y que luego utilizaremos para establecer las propiedades.

Definición 4.29 (Equivalencia entre rama y secuencia)
La secuencia vacía ϵ y una rama vacía son equivalentes. Una rama de un árbol $\theta = \kappa\sigma$ y una secuencia $s = [a]s'$ son equivalentes, si la raíz de la rama es igual al primer elemento de la secuencia, i.e., $\kappa = a$ y σ y s' son equivalentes. ■

Definición 4.30 (Equivalencia entre árbol dialéctico y juego)
Un árbol dialéctico y un juego son *equivalentes* si cada rama del árbol es equivalente a una secuencia completa del juego y cada secuencia completa del juego es equivalente a una rama del árbol dialéctico. ■

El siguiente lema establece que las secuencias legales y las líneas de argumentación aceptables son equivalentes. Más adelante mostramos que la forma constructiva con la que obtenemos un árbol dialéctico fue capturado con la manera declarativa de definir un juego.

Lema 4.24 *s es una secuencia legal si y sólo si s una línea de argumentación aceptable.*

Demostración: ($\Rightarrow$) Probaremos por inducción sobre la longitud de la secuencia. Sea s una secuencia legal completa tal que $|s| = n$.

CASO BASE: $n = 1$. Trivial

PASO INDUCTIVO: La secuencia es finita, condición que exige la línea de argumentación aceptable.

Por otra parte, s es legal si $[s_1, ..., s_{n-1}]$ es legal y $s_n \in legal([s_1, ..., s_{n-1}])$.
Por hipótesis inductiva como $[s_1, ..., s_{n-1}]$ es legal entonces es una línea de argumentación aceptable. Como $s_n \in legal([s_1, ..., s_{n-1}])$ entonces s_n es un derrotador propio de s_{n-1} o bien si es

un derrotador por bloqueo entonces no puede ser el caso de que s_{n-1} sea un derrotador por bloqueo de s_{n-2}.

Por otra parte es concordante ya que por definición de la función legal tienen que ser consistente los argumentos en posición par con Π. Lo mismo debe ocurrir con lo argumentos en posición impar (Condición 2 de la definición).

Finalmente la tercer condición de la definición de legal impide que un argumento sea subconjunto de otro argumento anterior en la secuencia.

Así la secuencia legal es una línea de argumentación aceptable.

($\Leftarrow$)Probaremos por inducción sobre la longitud n de la línea de argumentación aceptable.

CASO BASE: $n = 1$. Trivial

PASO INDUCTIVO: Sea $\Lambda = [\lambda_1, \ldots, \lambda_n]$ una línea de argumentación aceptable de longitud n. Por definición es finita.

Por hipótesis inductiva, la línea de argumentación $[\lambda_1 \ldots \lambda_{n-1}]$ es una secuencia legal. Luego debo probar que $\lambda_n \in legal([\lambda_1 \ldots \lambda_{n-1}])$. Ya que λ_n está en la línea de argumentación debe ser un derrotador de λ_{n-1}. Si λ_n es un derrotador por bloqueo, es decir según la relación de preferencia no es mejor que su antecesor en la secuencia, entonces por definición λ_{n-1} no puede ser un derrotador por bloqueo. Así se cumple la primer condición de la definición de legal.

Por otro lado, en la línea de argumentación aceptable, el conjunto de soporte es concordante, esto es el conjunto de los argumentos en posición impar no son inconsistentes con Π. Además el conjunto de interferencia también es concordante, es decir el conjunto de los argumentos en posición par no es inconsistente con Π.

Finalmente, ningún argumento es subargumento de otro en la línea. Por lo tanto, λ_n no es subconjunto de ningún argumento anterior.

Por lo tanto, $\lambda_n \in legal[\lambda_1 \ldots \lambda_{n-1}])$. Luego $[\lambda_1 \ldots \lambda_n]$ es una secuencia legal. ■

Lema 4.25 *Sean $\mathcal{P}$ un plrb, h un literal y $\langle \mathcal{A}, h \rangle$ una estructura de argumento. Existe un árbol dialéctico $T_{\langle \mathcal{A},h \rangle}$ si y solamente si existe un juego $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ tales que $T_{\langle \mathcal{A},h \rangle}$ y $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ son equivalentes.*

Demostración: ($\Rightarrow$) Por inducción sobre la profundidad del árbol.

CASO BASE: n=0

Si el árbol dialéctico tiene profundidad 0, entonces la raíz $\langle \mathcal{A}, h\rangle$ es nodo hoja, así por definición no existe ningún derrotador para $\langle \mathcal{A}, h\rangle$. Construimos el árbol $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$, siendo $M_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})} = \{\langle \mathcal{A}, h\rangle\}$, $J_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}(\langle \mathcal{A}, h\rangle, 1) = \mathrm{P}$ y $P_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$ es el conjunto $\{\epsilon, \langle \mathcal{A}, h\rangle\}$ ya que no existe ningún derrotador para $\langle \mathcal{A}, h\rangle$.

El árbol dialéctico tiene sólo un nodo y el juego sólo una secuencia completa y la etiqueta del nodo y la secuencia completa tienen al mismo elemento. Luego el árbol dialéctico y el juego son equivalentes.

PASO INDUCTIVO: Supongamos que el árbol dialéctico tiene profundidad n y que el lema vale para todo $k < n$. Debemos probar que el lema vale para n.

Sea $T_{\langle \mathcal{A},h\rangle}$ el árbol dialéctico con profundidad n. Sean $T_{\langle \mathcal{A}_1,h_1\rangle}, T_{\langle \mathcal{A}_2,h_2\rangle}, \ldots, T_{\langle \mathcal{A}_r,h_r\rangle}$ los subárboles hijos de la raíz de $T_{\langle \mathcal{A},h\rangle}$, $\langle \mathcal{A}, h\rangle$. $T_{\langle \mathcal{A}_1,h_1\rangle}, T_{\langle \mathcal{A}_2,h_2\rangle}, \ldots, T_{\langle \mathcal{A}_r,h_r\rangle}$ son árboles dialécticos con profundidad menor que n, luego estamos en las condiciones del lema. Por lo tanto, por hipótesis inductiva, existen $G(\langle \mathcal{A}_1, h_1\rangle, \mathcal{P}), G(\langle \mathcal{A}_2, h_2\rangle, \mathcal{P}), \ldots, G(\langle \mathcal{A}_r, h_r\rangle, \mathcal{P})$, juegos equivalentes a los árboles dialécticos antes mencionados.
Construyamos el siguiente juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$:

$M_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})} = \{\langle \mathcal{A}, h\rangle\} \cup M_{G(\langle \mathcal{A}_1,h_1\rangle,\mathcal{P})} \cup \ldots \cup M_{G(\langle \mathcal{A}_r,h_r\rangle,\mathcal{P})}$

$J_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}(\langle \mathcal{A}, h\rangle, 1) = \mathrm{P}$ y $J_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}(\langle \mathcal{A}_i, h_i\rangle, j+1) = \overline{J_{G(\langle \mathcal{A}_k,h_k\rangle)}(\langle \mathcal{A}_i, h_i\rangle, j)}$ para cada $\langle \mathcal{A}_i, h_i\rangle \in M_{G(\langle \mathcal{A}_k,h_k\rangle,\mathcal{P})}$ y para cada juego $G(\langle \mathcal{A}_k, h_k\rangle, \mathcal{P})$.

Ya que el árbol dialéctico tiene como hijo de $\langle \mathcal{A}, h\rangle$ a $\langle \mathcal{A}_1, h_1\rangle, \langle \mathcal{A}_2, h_2\rangle, \ldots, \langle \mathcal{A}_r, h_r\rangle$ entonces $legal(\langle \mathcal{A}, h\rangle) = \{\langle \mathcal{A}_1, h_1\rangle, \langle \mathcal{A}_2, h_2\rangle, \ldots, \langle \mathcal{A}_r, h_r\rangle\}$ y cada rama en el árbol es una línea de argumentación aceptable, luego

$$P_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})} = \{\epsilon, \langle \mathcal{A}, h\rangle\} \cup \{[\langle \mathcal{A}, h\rangle]s | s \in P_{G(\langle \mathcal{A}_1,h_1\rangle)}\} \cup \ldots \cup \{[\langle \mathcal{A}, h\rangle]s | s \in P_{G(\langle \mathcal{A}_r,h_r\rangle)}\}$$

Este juego y el árbol $T_{\langle \mathcal{A},h\rangle}$ son equivalentes ya que cada rama del árbol comienza con $\langle \mathcal{A}, h\rangle$, al igual que las secuencias completas y por hipótesis inductiva el resto de la secuencia es equivalente a cada una de las ramas del árbol dialéctico.

($\Leftarrow$) Sea $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ un juego, entonces existe un árbol dialéctico equivalente.
Construyamos el árbol a partir de las secuencias completas en $P_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$. Por cada secuencia completa generamos una rama en el árbol. Por el lema anterior cada secuencia en un juego es una línea de argumentación aceptable. El árbol dialéctico obtenido es equivalente a las secuencias completas del juego. ■

Finalemente probaremos la sensatez y completitud de la semántica $\mathcal{GS}$ con respecto a la teoría de prueba.

Lema 4.26 *Sean $\mathcal{P}$ un plrb, h un literal y $\langle \mathcal{A}, h\rangle$ una estructura de argumento. Existe un árbol dialéctico $T_{\langle \mathcal{A},h\rangle}$ cuya raíz está etiquetada con U si y solamente si existe un juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ ganado por el proponente.*

Demostración: ($\Rightarrow$) Sea $T_{\langle \mathcal{A},h\rangle}$ un árbol dialéctico cuya raíz está etiquetada con U. Por el lema 4.25, existe un juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ equivalente a $T_{\langle \mathcal{A},h\rangle}$. Debemos probar que $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ es ganado por el proponente.
Para esto consideraremos cada una de las ramas del árbol dialéctico. Si la rama es de longitud impar entonces se corresponde con una secuencia completa preferida. Si la rama r no es de longitud impar, entonces el subárbol que la contiene, tiene alguna otra rama, ya que de otro modo por ser una rama par la raíz estaría etiquetada con D. Así existe otra rama t en el subárbol que cambia la etiqueta U en posición par y D en posición impar a las etiquetas inversas. El único modo de lograr este cambio es que t sea de longitud impar, ya que entonces las etiquetas son inversas a la de r. Si r y t difieren en una posición par entonces sigue prevaleciendo las etiquetas de la rama par y por lo tanto la raíz debería estar etiquetada con D. Así las ramas difieren en posición impar. Luego, si bien la secuencia completa equivalente a r es de longitud par, existe otra secuencia completa preferida t que difiere de r en posición impar.

Por lo tanto, el conjunto de las secuencias completas es una estrategia y $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ es ganado por el proponente.

($\Leftarrow$) Sea $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ un juego para $\langle \mathcal{A}, h\rangle$ bajo $\mathcal{P}$ ganado por el proponente. Por el lema 4.25, existe un árbol dialéctico $T_{\langle \mathcal{A},h\rangle}$ equivalente a $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$. Debemos probar que la raíz de $T_{\langle \mathcal{A},h\rangle}$ está etiquetada con U.
Como $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ es un juego ganado por el proponente entonces todas sus secuencias completas s, equivalentes a las ramas en el árbol cumplen que:

- s es preferida: su longitud es impar, así el primer elemento en la rama equivalente, analizada en forma aislada, está etiquetado con U, ya que proceso de etiquetado sobre una rama alterna entre U y D comenzando desde la hoja con U.

- s no es preferida, pero existe otra secuencia completa preferida t que difiere de s en posición impar. En el árbol hay una rama equivalente a s de longitud par $s = s_1 s_2 \ldots s_n$ con n par y también existe una rama equivalente a $t = t_1 t_2 \ldots t_k$ con k impar y s y t difieren en la posición r impar. Reescribimos a $t = s_1 s_2 \ldots s_{r-1} t_r \ldots t_k$. Si analizamos a s en forma aislada s_1 estaría etiquetada con D, ya que su longitud es par. Sin embargo, el etiquetado de los nodos depende de todos sus hijos. El punto conflictivo es s_{r-1}, ya que a partir de sus hijos las dos ramas difieren. La rama s_{r-1} es una posición par, ya que s_r y t_r es una posición impar. Si analizamos a la rama s, s_r estará etiquetada con D, ya que s es par, y por lo tanto, las posiciones pares estarán etiquetadas con U y las

impares con D. En forma contraria, t_r estará etiquetado con U, ya que k es impar, luego las posiciones pares estarán etiquetadas con D y las impares con U. En el árbol dialéctico, el nodo $s_{r-1} = t_{r-1}$ estará etiquetado con D ya que uno de sus hijos no fue derrotado. Dado que esta es una posición par si continuamos hacia atrás en el etiquetado llegaremos a que la posición impar 1 está etiquetada con U.

Por lo tanto, el árbol equivalente al juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ tiene su raíz etiquetada con U. ■

Teorema 4.5 (Sensatez y Completitud) *Sean $\mathcal{P}$ un plrb, h un literal y G el único G-modelo de $\mathcal{P}$. h está garantizado bajo $\mathcal{P}$ si y solamente h pertenece a G.*

Demostración: h está garantizado bajo $\mathcal{P}$ si y solo si existe un árbol dialéctico cuya raíz es $\langle \mathcal{A}, h\rangle$ y está etiquetado con U si y solo si por lema anterior existe un juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ ganado por el proponente si y solo si $h \in V$ si h es un átomo o bien $h \in F$ si h es un átomo negado, siendo $G = \langle V, F\rangle$ el único G-modelo de $\mathcal{P}$. ■

4.5. Conclusiones

En este capítulo hemos presentado uno de los resultados más importantes de esta tesis. Hemos capturado en forma declarativa la teoría de prueba de la P.L.R.B. a través de un modelo sustentado por juegos y hemos mostrado que es sensata y completa.

Los juegos son análogos a una disputa y por lo tanto, esa analogía se extiende al razonamiento basado en argumentos. Una disputa puede ser visto como un juego donde de una manera alternada, el jugador P, el proponente, comienza con un argumento para un literal. El jugador O, el oponente, ataca el argumento previo con un contraargumento suficientemente fuerte como para derrotarlo. La disputa podría continuar con un contraargumento del proponente y así sucesivamente. Cada jugada en el juego es un movimiento permitido para el participante que tiene el turno, su equivalente, un argumento a favor o en contra dependiendo del jugador, para el argumento que inicia el debate.

Cuando un jugador se queda sin movimientos, i.e., el jugador no puede encontrar un contraargumentos de ninguno de los argumentos de su adversario, el juego se termina. Si el argumento del proponente no ha sido derrotado, luego ha ganado el juego

La semántica $\mathcal{GS}$ es una semántica declarativa trivaluada basada en juegos para P.L.R.B.que relaciona las semánticas de juegos [AM97] y la teoría de modelos.

Inicialmente se trabajó en los movimientos de los juegos, los argumentos. Se realizó un estudio declarativo de su versión procedural y se mostró que eran equivalentes. A partir de

este concepto se circunscribió el conjunto de potenciales argumentos y contraargumentos que pueden ser parte de un juego, dado un argumento inicial.

Luego se determinaron todos los elementos que contiene un juego, con el objeto de caracterizar un árbol dialéctico. Cabe aclarar que el criterio preferencia entre argumentos, no quedó fijo, sino que se trabajó en forma genérica. Dado que para un literal se podrían jugar más de un juego, se definió el concepto de familia de juegos para un literal dado.

Finalmente, a fin de mantener la equivalencia entre la teoría de prueba de la P.L.R.B. y los juegos, se introdujo la noción de estrategia ganadora para los juegos y se clasificó a los literales como VERDADEROS cuando existe un juego ganado por el proponente del argumento inicial del literal, FALSO cuando existe un juego ganado por el proponente del argumento inicial del complemento del literal e INDECISO cuando no existe ningún juego ganado por el proponente del argumento inicial para el literal ni para el complemento. A partir de esta clasificación se genera el G-modelo de un *plrb*.

Destacamos que la semántica trivaluada basada en juegos $\mathcal{GS}$ es sensata y completa con respecto a la teoría de prueba, i.e., existe equivalencia entre las nociones sintáctica y semántica.

Capítulo 5

Comparación con otros formalismos

Contenido

En este capítulo se presentan otros formalismos referidos a la argumentación rebatible junto con su semántica declarativa, analizándose las características más significativas. La intención es realizar una comparación de las semánticas de estos sistemas con la semántica basada en juegos desarrollada en este trabajo. No es el objetivo de este capítulo realizar un estudio exhaustivo de todos los formalismos relacionados a la argumentación, sino solamente individualizar aquellos para los que se ha desarrollado una semántica declarativa. En particular, hemos hecho hincapié en aquellos cuya semántica tiene componentes de los juegos, aunque no nos hemos restringido solo a estos.

En la sección 5.1 se introduce marco argumentativo de Dung junto con las nociones de extensión preferida, estable y ground y se discuten las diferencias existentes en cuanto a la teoría de prueba subyacente. En la sección 5.2, se describe el marco argumentativo basado en diálogos de Prakken. Este formalismo presenta muchas similitudes con $\mathcal{GS}$ en cuanto a la definición como juego de los debates, aunque existen diferencias en cuanto a cuando un literal es consecuencia del sistema. Luego se presenta el sistema argumentativo de Amgoud y Cayrol, que puede verse como una convergencia de los dos sistemas anteriores. En la sección 5.4, se considera la semántica dialéctica de Jakobovits y Vermeir y se analizan las principales razones por las que esta semántica no es adecuada para nuestro sistema. En la sección 5.5, se detalla una definición alternativa de P.L.R.B. basada en la teoría de juegos [VTS08] y se la compara con la presentada

en nuestro trabajo. Cabe aclarar que el trabajo fue presentado con posterioridad al nuestro. El capítulo concluye con una discusión de cómo se relacionan los formalismos analizados.

5.1. Teoría de argumentación de Dung y su semántica

P. Dung [Dun93, Dun95] propone un marco argumentativo abstracto $AF = \langle AR, attacks \rangle$ donde AR es un conjunto de argumentos y $attacks \subseteq AR \times AR$. Si bien, Dung no da una definición formal de la relación de ataque, se deduce de su trabajo que corresponde al análogo de ataque y derrota débil entre argumentos[1] [Pra98, Gar00]. El autor muestra que su teoría de argumentación provee un fundamento unificado para diferentes enfoques al razonamiento no monótono.

La semántica para este enfoque está basada en la aceptabilidad de los argumentos. El conjunto de argumentos aceptados por un agente racional es el conjunto de argumentos que puede defenderse de todos los ataques.

Definición 5.1 Sea S un conjunto de argumentos.

1. S se dice *libre de conflictos* si no existen dos argumentos A, B en S tal que A ataque a B o B ataque a A.
2. Un argumento A es *aceptable* con respecto a S si y sólo si para cada argumento B: si B ataca a A luego B es atacado por algún argumento en S.
3. Un conjunto de argumentos libre de conflictos S' es admisible si y sólo si cada argumento en S' es aceptable con respecto a S'.

■

La semántica crédula está basada en el concepto de *extensión preferida*: Una *extensión preferida* de un marco argumentativo AF es un conjunto de agumentos de AF admisible y maximal con respecto a la inclusión. Dung demuestra que la semántica de extensiones preferidas siempre está definida para marcos argumentativos.

Asimismo, presenta la idea de *extensión estable* con el objeto de comparar este marco con otros enfoques. Un conjunto de argumentos S es llamado una *extensión estable* si y sólo si S ataca a cada argumento que no pertenece a S. Un resultado inmediato de estas nociones [Dun95] es que cada extensión estable es una extensión preferida. El recíproco no vale.

[1] A derrota en forma débil a B si A y B están en conflicto y A no es peor que B.

La *extensión ground* es la semántica escéptica que está definida a través del menor punto fijo de la función característica $F_{AF} : 2^{AR} \to 2^{AR}$:

$$F_{AF}(S) = \{A | A \text{ es aceptable con respecto a S}\}$$

Ejemplo 5.1 [Dun95] Consideremos el diamante de Nixon. Este caso puede ser especificado a través de dos argumentos A y B, donde A establece que *"Nixon no es pacifista puesto que es republicano"* y B establece que *"Nixon es pacifista puesto que es cuáquero"*. El diamante de Nixon puede ser representado como $\Big\langle \big\{A,\ B\big\}, \big\{(A,B),(B,A)\big\} \Big\rangle$. Este marco argumentativo tiene dos extensiones preferidas, una en la cual Nixon es pacifista y otra en la que Nixon es anti-pacifista. ■

Ejemplo 5.2 [Dun95] Considere la siguiente disputación entre dos personas **I** y **A**:

I: Mi gobierno no puede negociar con su gobierno pues su gobierno no reconoce al mío.

A: Su gobierno tampoco reconoce a mi gobierno.

I: Pero su gobierno es un gobierno terrorista.

Asumimos que i_1 e i_2 denotan al primer y segundo argumento de **I**, respectivamente y a denota el argumento de **A**. La disputación anterior puede representarse con el marco argumentativo abstracto

$$AF = \Big\langle \big\{i_1,\ i_2,\ a\big\}, \big\{(i_1,a),(a,i_1),(i_2,a)\big\} \Big\rangle$$

AF tiene una extensión preferida $E = \{i_1, i_2\}$. Asimismo, puede verse que la extensión ground de AF coincide con su extensión preferida dado que $F_{AF}(\varnothing) = \{i_2\}$, $F^2_{AF}(\varnothing) = \{i_1, i_2\}$ y $F^3_{AF}(\varnothing) = F^2_{AF}(\varnothing)$. ■

En [DS96] y [DS01], se avanza sobre la idea del marco argumentativo razonando con especificidad. La relación de especificidad está dada entre reglas default, a diferencia del sistema P.L.R.B. en el que la especificidad está definida entre argumentos. Dada una teoría default $T = (E, B, D)$ donde E es el conjunto de literales fijos que representan la evidencia de la teoría, B es un conjunto de cláusulas fijas, siendo $E \cup B$ consistentes y D es un conjunto de defaults, existen dos clases de ataques bien diferenciados. Así un conjunto de defaults K ataca a un default d en una teoría T si:

Ataque por conflicto: $E \cup B$ derivan utilizando las reglas default de K al complemento de la consecuencia de d.

Ataque por Especificidad: existe un default $d' \in D$ tal que

- B en conjunto con las consecuencias de d' y d es inconsistente,
- el antecedente de d' unido a B derivan en conjunto con K el antecedente de d,
- el antecedente de d' unido a B y las reglas en K no es inconsistente y
- E unido a B derivan a través de las reglas K al antecedente de d'.

Ejemplo 5.3 [DS01] Consideremos el famoso ejemplo de las aves y los pingüinos. La teoría $T = (E, B, D)$ se define como $E = \{p\}$, $B = \{p \Rightarrow a\}$ que representa los pingüinos son aves y $D = \{d_1 = a \rightarrow v,\ d_2 = p \rightarrow \neg v\}$ que representan *normalmente, las aves vuelan* y *normalmente, los pingüinos no vuelan*. En este caso, d_2 ataca por especificidad a d_1. ■

A partir de la definición de ataque se define la semántica estable. K se dice que ataca a algún conjunto $H \subseteq D$ si K ataca algún default en H. Un conjunto de defaults S es llamado una *extensión* de T si S no se ataca a sí mismo y ataca a cada default que no pertenece a S. Dado un literal fijo l y una teoría T, decimos que l es consecuencia de T, si para cada extensión S de T, $E \cup B$ derivan en conjunto con las reglas de S a l.

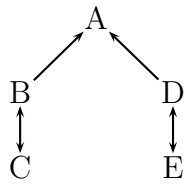
Ejemplo 5.4 [DS01] Consideremos nuevamente el ejemplo de las aves y los pingüinos. La consulta para esta teoría es si los pingüinos vuelan. Sean $K = \{d_1\}$ y $H = \{d_2\}$. Ya que $\{p\} \cup B$ derivan junto a H $\neg v$ y $\{p\} \cup B$ derivan junto con K a v, tenemos que K y H atacan d_2 y d_1 por conflicto, respectivamente. Así K y H atacan a cada default que no pertenece a ellos. Ahora, mientras H no se ataca a sí mismo, K se ataca a sí mismo por especificidad, ya que $d_1 \in K$, K ataca a d_1, pues existe d_2 en las condiciones expresadas anteriormente. Por lo tanto, H es la única extensión de T y la respuesta a la consulta es $\neg v$, i.e., un pingüino no vuela. ■

Análisis y Discusión

El trabajo de Dung brinda una contribución sumamente importante a la argumentación rebatible debido a que permite capturar diversos sistemas no monotónicos y compararlos entre ellos. Sin embargo, las semánticas definidas presentan algunas deferencias con respecto al sistema de la P.L.R.B. en estudio:

- Existen diferencias en cuanto a la semántica de los sistemas. El hecho de que existan derrotadores por bloqueo en la P.L.R.B. los que no pueden ser derrotados por otros derrotadores por bloqueo, hace que las consecuencias de un programa sean diferentes. Veamos un ejemplo que clarificará esta situación.

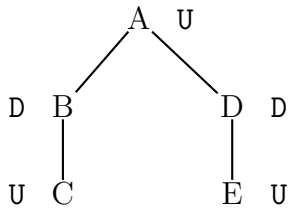
  Ejemplo 5.5 Supongamos que tenemos la siguiente situación de derrota:



La semántica escéptica de Dung, i.e., la extensión grounded es:

$$F_{AF}(\varnothing) = \varnothing$$

Por otro lado el árbol de dialéctica para A es



Por lo tanto, el literal cuya estructura de argumento se representa con A es consecuencia de nuestro *plr*, aunque no es consecuencia de la semántica escéptica.

Si tenemos argumentos que se derrotan entre ellos, en la terminología de la P.L.R.B. derrotador por bloqueo, ninguno de ellos es mejor que el otro y por lo tanto es como si se anularan uno con el otro. Esto reinstala al argumento A. ■

- Por el modo en que la semántica operacional de la P.L.R.B. está definido, no existen argumentos que se autoderrotan. Sin embargo, el marco argumentativo de Dung los permite. En dichas situaciones la semántica estable no se comporta del modo esperado[PV00]. Supongamos un marco argumentativo $AF = \langle\{A, B\}, \{(A, A), (A, B)\}\rangle$. Para este marco no existe extensión estable, sin embargo existe extensión preferida pues Dung garantiza que siempre existe. El conjunto $\{B\}$ no es admisible, luego la única extensión preferida es vacía. Intuitivamente, pareciera que $\{B\}$ debería ser la única extensión preferida, ya que el único derrotador de B se auto derrota.

- Con respecto al trabajo [DS01] como apuntan los autores la especificidad está definida sólo para casos en conflictos. La especificidad generalizada y la equi-especificidad están definidas para cualquier conjunto de argumentos, aún si éstos no están en conflicto.

- El sistema de Dung cumple la propiedad de Cumulatividad, a diferencia del sistema P.L.R.B.[Cn01].

- Por último, como apuntan los autores en [DS01], la definición de la P.L.R.B. requiere de la condición de minimalidad para los argumentos. Esto hace que al agregar un nuevo hecho, algunos argumentos se eliminen. Creemos que este punto muestra dinamismo en el sistema con respecto a los cambios.

5.2. Marco argumentativo basado en diálogos de Henry Prakken

En [Pra01, PV00] el autor propone un marco argumentativo basado en diálogos en el que se modela la interacción entre dos agentes. Si bien la semántica $\mathcal{GS}$ y este marco comparten la mayoría de los componentes, los objetivos son diferentes. Para $\mathcal{GS}$ el objetivo es dar la semántica declarativa del sistema P.L.R.B., mientras que el marco modela el proceso dialéctico de dos agentes.

Según Prakken una abstracción de todo sistema argumentativo debería contar con tres elementos: un conjunto de argumentos, una relación de fuerza relativa entre los argumentos y una teoría de prueba dialéctica.

Definición 5.2 Un *marco dialéctico* es una tupla $(Arg, \preceq, T)$, donde

- $Args$ es un conjunto de argumentos
- $\preceq \subseteq Args \times Args$ es una relación de fuerza entre argumentos. $A \preceq B$ significa que B no es más débil que A. $A \prec B \ =_{def} \ A \preceq B$ y $B \not\preceq A$.
- T es una teoría de prueba dialéctica para $(Arg, \preceq)$

■

Prakken sostiene que una teoría de prueba dialéctica para una argumentación rebatible tiene los siguientes elementos:

- Jugadores: al menos se requieren de un proponente y un oponente de un argumento inicial. Otros posibles jugadores como referí, coordinadores, jueces o mediadores no pertenecen a la disputa sino a otras clases de protocolo.
- Un conjunto de Argumentos válidos, de acuerdo a una noción monotónica de consecuencia lógica.
- Una relación de fuerza relativa entre argumentos. A partir de esta noción se determinarán las movidas legales.
- Movidas bien formadas las que están compuestas por un jugador, un argumento y una referencia a la movida a la que se responde. El conjunto de movidas legales está completamente determinado por el conjunto $Args$.

- Una función JugadorAMover que determina en cada etapa del diálogo que jugador realiza la siguiente movida. El proponente es siempre el que tiene el primer turno para mover. En una teoría de prueba dialéctica, los jugadores se turnan para realizar sus movidas.

- Una función de movimientos Legales que especifica en cada estado del diálogo que movimientos son legales en cada etapa. La legalidad de un movimiento estará completamente determinada por el desarrollo del diálogo hasta el momento. En toda teoría de prueba dialéctica algunas consideraciones a tener en cuenta para la legalidad de un movimiento son: que el movimiento responda al movimiento precedente, regular los movimientos repetidos si fueran permitidos, y la fuerza relativa entre los movimientos.

- Noción de Diálogo, como una serie de movimientos legales, respetando los turnos de los jugadores.

- Un criterio que determine quien gana el diálogo. Un jugador gana si y sólo si el otro jugador no puede mover.

- Un criterio que permita determinar si un argumento es probable en esa teoría. Un argumento se considera probable si el proponente tiene una estrategia ganadora en un diálogo que comienza con este argumento.

A continuación se presenta la formalización de estas ideas.

Definición 5.3 Una *teoría de prueba dialéctica* para un par $(Args, \preceq)$ es una tupla

$$(Jugadores, Movimientos, JugadorAMover, Legal, Diálogos, Ganador),$$

donde:

- $Jugadores = \{P, O\}$, $\overline{Jugador} = O$ si $Jugador = P$ y $\overline{Jugador} = P$ si $Jugador = O$.

- Movimientos definido recursivamente como el menor conjunto que satisface:

  - Si $A \in Args$ y $J \in Jugadores$, entonces $(J, A) \in$ Movimientos. A estos movimientos se los denomina iniciales

  - Si $A \in Args$, $J \in Jugadores$ y $M_i \in$ Movimientos, entonces $(J, A, M_i) \in$ Movimientos. Estos movimientos se denominan de respuesta: el movimiento (J, A, M_i) responde al movimiento M_i.

  Utilizaremos la siguiente notación: Jugador$(J, A, M_i) = J$, Arg$(J, A, M_i) = A$ y Mov$(J, A, M_i) = M_i$.

- JugadorAMover es una función que determina qué jugador tiene que jugar a continuación en el diálogo. Sea Pow*(Movimientos) el conjunto de todas las secuencias finitas de subconjuntos de Movimientos. Entonces

$$JugadorAMover : Pow^*(Movimientos) \rightarrow Jugadores$$

tal que JugadorAMover(D)=P si y sólo si D es de longitud par y JugadorAMover(D)=O si y sólo si D es de longitud impar.

- Legal es una función que en cada punto del diálogo define los movimientos que pueden ser hechos en tal punto:

$$Legal : Pow^*(Movimientos) \rightarrow Pow(Movimientos)$$

tal que

1. $M_i \in Legal(\varnothing)$ si y sólo si M_i es un movimiento inicial;
2. Si $M_i \in Legal(M_1, \ldots, M_{i-1}), (i > 1)$, entonces M_i responde a M_{i-1}.

- Diálogos: es el conjunto de todas las secuencias de movimientos $M_1, \ldots, M_n$ de movimientos tal que para cada i:

1. Jugador(M_i)=JugadorAMover($M_1, \ldots, M_{i-1}$),
2. $M_i \in$ Legal($M_1, \ldots, M_{i-1}$)

■

Nótese que en esta definición solo se establece que para que un argumento sea legal debe "responder" al movimiento anterior, concepto que no queda completamente definido en términos de fuerza relativa entre los argumentos. A partir de la definición anterior, el autor presenta la noción de ganador.

Definición 5.4 Ganador es una función (parcial) que determina el ganador del diálogo, si lo hubiera.

$$Ganador : Diálogo \rightarrow Jugadores$$

tal que el jugador p gana un diálogo D si y sólo si JugadorAMover(D)=$\overline{p}$ y Legal(D)=$\varnothing$. ■

Bajo esta definición de marco argumentativo las nociones $Args$, $\preceq$ y $Legal$ quedan parametrizadas, de modo que las teorías de prueba dialécticas pueden diferir solamente en esos tres puntos según apunta el autor.

Finalmente, resta formalizar cuando un argumento se considera probado según la teoría de prueba dialéctica.

Definición 5.5 Una *estrategia* para un jugador p en un marco dialéctico F es un árbol de diálogos basado en F que ramifica después de las movidas de p y contiene todas las respuestas legales de $\overline{p}$. ■

Una estrategia para un jugador p permite a p hacer solamente una movida en respuesta a su oponente y a $\overline{p}$ le exige hacer todas las movidas posibles que respondan a p.

Definición 5.6 Un argumento A es *probable rebatiblemente* en un marco dialéctico F si y sólo si el proponente tiene una estrategia ganadora en F con el movimiento (P, A) como raíz. ■

Si existe una estrategia ganadora (P, A), entonces existe un árbol de diálogos en el que todas los contraargumentos dados por O, son respondidos por sólo un argumento de P. Esto es, en esa estrategia P juega su mejor argumento para responder a O y O ataca ese movimiento con toda su "artillería". Si cada rama en este árbol es ganada por el proponente, entonces el proponente respondió exitosamente a todos los contraargumentos de O dejándolo sin argumentos para seguir jugando y por lo tanto el argumento A es probable rebatiblemente.

A continuación se presenta una instanciación dada por el autor de la teoría de prueba.

Ejemplo 5.6 Dado un marco argumentativo, dejaremos sin especificar a la estructura de los argumentos $Args$ y la relación de fuerza entre argumentos $\preceq$ será una relación de ataque sobre $Args$ sin propiedades asumidas. La definición de teoría de prueba permanece equivalente excepto por la definición de la función Legal, a la que además de las condiciones dadas, deberá cumplir que:

- Si $Jugador(M_i) = P$, luego $Arg(M_i)$ ataca a $Arg(M_{i-1})$ mientras que $Arg(M_{i-1})$ no ataca a $Arg(M_i)$; y
- Si $Jugador(M_i) = O$, luego $Arg(M_i)$ ataca a $Arg(M_{i-1})$.

En este caso, el énfasis de la teoría de prueba está puesto sobre el proponente: sus argumentos deben ser más fuertes que los contraargumentos del oponente, mientras que los argumentos del oponente sólo pueden causar dudas. Esto puede compararse con P.L.R.B. en que el proponente solo puede jugar derrotadores propios mientras que el oponente puede jugar derrotadores propios o por bloqueo. El autor indica que podría agregarse una nueva condición a Legal sobre los movimientos de P: P no podría repetir alguno de sus argumentos anteriores[PS97] . Esto no cambiaría la condición de que un argumento sea probable en esta teoría, ya que el oponente O tendría al menos una respuesta la segunda vez que apareciera el argumento de P, la misma respuesta que dio la primera vez, pero evitaría los diálogos infinitos si $Args$ es finito. ■

El autor avanza un poco más y define un protocolo para disputas como caso particular de los diálogos. En particular se restringe a disputas en donde solamente hay intercambio de argumentos y contraargumentos, dejando posibles diálogos de persuasión fuera del alcance de este marco.

Existen dos diferencias importantes entre las teorías de prueba dialéctica y los protocolos para disputas:

- El foco sobre diálogos posibles vs. reales: en este sentido en la teoría de prueba se enfoca a la existencia de estrategias ganadoras, es decir, sobre la posibilidad de ganar un diálogo. En un protocolo de disputa, el enfoque está en evaluar la disputa a medida que va evolucionando, por lo que tenemos que analizar como es construido el árbol de diálogo durante la disputa.
- Base de información fija vs. dinámica: durante una disputa las partes pueden dar nueva información, que debería alterar los conocimientos de los participantes.

La formalización adoptada por el autor permite modelar disputas en las cuales los contendientes pueden realizar sólo una jugada por turno y en las que cada movimiento es respondiendo a lo sumo una vez.

Notación: el autor agrega el subíndice R a los conceptos correspondientes al protocolo de disputa y el subíndice T a aquellos pertenecientes a la teoría de prueba dialéctica. Por otra parte, da la siguiente notación auxiliar para los diálogos: para cualquier secuencia de movimientos $D = M_1, \ldots, M_n$, donde M_1 es un movimiento inicial, L_j es el diálogo $M_1, \ldots, M_j$ contenido en D.

Definición 5.7 Un *protocolo para disputas de movimiento único* R es una tupla

$$(F, ArgsIniciales, Jugadores_R, Movimientos_R, JugadorAMover_R, Legal_R, Disputas, Ganador_R),$$

donde:

- $E = (Args, \preceq, T)$ es un marco dialéctico;
- $ArgsIniciales \subseteq Args$, posiblemente vacío;
- $Movimientos_R = Movimientos_T$;
- $Jugadores_R = Jugadores_T$;
- $JugadorAMover_R = JugadorAMover_T$;

- $Legal_R : Pow^*(Movimientos_R) \rightarrow Pow(Movimientos_R)$ tal que para toda secuencia de movimientos $D = M_1, \ldots, M_i$:

  1. $M_i \in Legal(\varnothing)$ si y sólo si M_i es un movimiento inicial;
  2. Si $M_{i+1} \in Legal(D), (i > 1)$, entonces $Movimiento(M_{i+1}) \in D$ y

     $$Jugador(Movimiento(M_{i+1})) = \overline{Jugador(M_{i+1})}.$$

  3. a. Si $M_{i+1} \in Legal_T(L_i)$, entonces $M_{i+1} \in Legal_R(D)$;
     b. Si $M_{i+1} \in Legal_R(D)$ y $Movimiento(M_{i+1}) = M_k$, entonces $M_{i+1} \in Legal_T(L_k)$;
  4. Si M_{i+1} y M_j $(j < i)$ son ambos respuestas a M_k, $M_j \in D$ y $M_{i+1} \in Legal_R(M_1, \ldots, M_i)$, entonces $Arg(M_{i+1}) \neq Arg(M_j)$.

- Disputas es el conjunto de todas las secuencias finitas de movimientos $M_1, \ldots, M_n$ de movimientos tal que para cada i:

  1. $Jugador_R(M_i) = JugadorAMover_R(M_1, \ldots, M_{i-1})$,
  2. $M_i \in Legal_R(M_1, \ldots, M_{i-1})$.

- $Ganador_R : Disputas \rightarrow Jugadores_R$ es una función (posiblemente parcial) que determina el ganador de una disputa en cada etapa.

■

ArgsIniciales es el conjunto de información sobre la cual los jugadores se han puesto de acuerdo o a la cual se circunscriben al comenzar la discusión. *Args* ahora no será el conjunto de todos los argumentos que se generan a partir de *cierta información*, sino que será el conjunto de todos los argumentos que podrían ser posiblemente establecidos en *cierto lenguaje*.

Analicemos la función $Legal_R$. La primer condición es equivalente a la de $Legal_T$. La segunda indica que cada movimiento es respuesta a un movimiento anterior en la disputa, no necesariamente el precedente y establece que las jugadas son por turnos alternados entre los jugadores. La tercer condición establece una equivalencia entre aquello que es considerado legal bajo la teoría de prueba dialéctica y aquello que es considerado legal bajo una disputa. Finalmente, la última condición prohíbe que dos respuestas al mismo movimiento tengan el mismo contenido.

El criterio ganador es un punto en el que el autor se extiende un poco más y que resumiremos ya que esto queda fuera del alcance de la comparación. Dado que está asumiendo que *Args* contiene todos los argumentos sobre el lenguaje, el criterio ganador tiene que ser mucho más fuerte. Prakken da una función $Ganador_R$ alternativa en la que se considera que un participante ganará la disputa si no solo ha realizado la última jugada, sino que también logra

que su oponente no pueda dar una nueva respuesta a partir de la base de conocimientos del estado actual. Según el autor, existen situaciones en las que hay argumentos que podrían ser introducidos en el debate, ya que la información agregada los habilita, pero que el protocolo no lo permite. Por esta razón, Prakken define un nuevo tipo de disputa: la disputa liberal.

En este sentido, el autor identifica primero aquellos argumentos utilizados en la disputa hasta el momento, a través de una clausura sobre los argumentos y luego determina cuales argumentos pueden considerarse relevantes en la disputa. Para definir la relevancia de un argumento el autor utiliza el estado de la disputa, que se corresponde a la idea del marcado del árbol.

Definición 5.8 Un movimiento M de una disputa D se considera *in* en D si y sólo si todos los movimientos en D que lo responden están *out* en D. De otro modo, M se considera *out* en D. ■

Definición 5.9 Un movimiento M es *relevante* en una disputa D si y sólo si cambia el estado de la disputa del movimiento inicial de D. ■

La idea de un movimiento relevante es que al incorporarlo, sin importar quien lo juegue, el argumento inicial cambie entre los posibles valores de estados de la disputa (*in* y *out*).

A partir de esta idea construye una clase de disputa.

Definición 5.10 Un protocolo es una *disputa liberal* si y sólo si para cualquier disputa D y movimiento M, ocurre que $M \in Legal_R(D)$ si y sólo si

1. M satisface las condiciones de legalidad dadas en la definición 7: y
2. M es relevante en D

■

Aunque no entraremos en detalles sobre el tema, el autor finaliza su análisis probando que esta clase de protocolo cumple con dos propiedades: la sensatez y la justicia. Intuitivamente un protocolo es *sensato* si y solo si el proponente gana una disputa a partir de un argumento inicial, entonces ese argumento es inferible en la teoría de prueba dialéctica asociada al tomar como punto de partida la información base empleada en la disputa. Y es *justo* si y solo si dada una disputa tal que un argumento inicial es inferible *en forma finita*, luego el proponente debe disponer de alguna estrategia ganadora para la continuación de la disputa acerca de ese argumento.

Análisis y Discusión

A continuación se presentan diferencias y similitudes con respecto a la semántica declarativa $\mathcal{GS}$:

- Esta propuesta permite que la base de conocimiento a partir de la que cada agente razona pueda ser modificada en forma dinámica. Esta característica responde a la idea de que el autor desea modelar la disputa entre dos agentes que a medida que avanzan en su discusión pueden alterar sus conocimiento. Si bien $\mathcal{GS}$ es caracterizada como un juego entre dos jugadores, este juego modela el razonamiento de un solo agente y por lo tanto, la base de conocimiento permanece estática.

- El marco dialéctico permite persuadir al contrincante además de atacar con nuevos argumentos a diferencia del marco del protocolo de disputas en que solamente se permite contestar al contrincante con un argumento en contra. El sistema P.L.R.B. modelado por $\mathcal{GS}$ es un protocolo de disputa y no permite persuadir al contrincante. Cada jugada es un ataque (propio o por bloqueo) del argumento anterior.

- Si bien la definición de $Legal_R$ no permite jugar dos veces el mismo argumento, el marco propuesto por Prakken no tiene en cuenta a los subargumentos de las jugadas anteriores. P.L.R.B. considera no sólo la introducción de argumentos ya dados, sino que también impide la introducción de subargumentos de argumentos ya dados en el debate.

- En el sistema P.L.R.B. y por lo tanto, también en su semántica $\mathcal{GS}$, se desea conocer si un literal es consecuencia de la base de conocimiento del agente. Así, el foco está en el literal. La semántica define un juego por cada argumento que lo sustenta y luego una familia de juegos, en donde se tienen en cuenta cada uno de los argumentos que sustentan a ese literal. Si alguno de esos juegos es ganado por el proponente, el *literal* es consecuencia del sistema. Bajo la propuesta de Prakken, el foco está en el argumento. Es decir, permite determinar si un argumento es consecuencia del sistema.

- La función *Legal*, según como esté definida puede permitir más de un movimiento inicial. Así se podría determinar en simultáneo si diferentes argumentos son consecuencia de un sistema. Inclusive, se podrían analizar en simultáneo todos los argumentos de un lenguaje. En este sentido, la semántica $\mathcal{GS}$ permite solamente el análisis por argumento. Los diferentes argumentos que soportan a un literal son considerados a través del concepto de familia de juegos. De este modo, se estratifican los conceptos.

- La idea del estado de la disputa a través de *in* y *out*, es equivalente a la del marcado del árbol dialéctico del sistema P.L.R.B..

- Si bien el autor tiene en mente otro objetivo al introducir la relevancia de un argumento en una disputa, es interesante notar que este concepto permite una poda en la construcción del árbol dialéctico.

5.3. Sistema Argumentativo de Amgoud y Cayrol

En [AC97], se propone un marco argumentativo que extiende al marco de Dung [Dun93, Dun95] y el de Prakken[Pra01] presentados anteriormente, integrando el orden de preferencia. La teoría de prueba propuesta en [AC02] está basada en árboles dialécticos.

Definición 5.11 Un *marco argumentativo basado en preferencias* (PAF)[2] es una tupla $\langle A, R, \texttt{Pref}\rangle$, donde A es una relación binaria que representa a la relación de derrota entre argumentos, $R \subseteq A \times A$ y `Pref` es un preorden (parcial o total) sobre $A \times A$. Denotamos con $>>^{\texttt{Pref}}$ al orden estricto asociado con `Pref`. ■

Definición 5.12 Sean B_1, B_2 dos argumentos de A. B_2 *ataca* a B_1 si y solamente si B_2 R B_1 y no ocurre que $B_1 >>^{\texttt{Pref}} B_2$. ■

La definición de argumento no es parte del PAF, aunque según cómo se defina las preferencias entre argumentos puede ser necesaria. Las definiciones que siguen son propuestas en [AC02] con el objeto de ilustrar al PAF.

Definición 5.13 Sea Σ una base de conocimiento proposicional, que puede ser inconsistente. Un *argumento* de Σ es un par ordenado (H, h) donde $H \subseteq \Sigma$ y:

(I) H es consistente,

(II) $H \vdash h$ ($\vdash$ denota la derivación clásica) y

(III) es minimal (según la inclusión de conjuntos).

■

Nótese que la signatura subyacente no permite distinguir entre información estricta e información rebatible. Así toda la información representada en la base de conocimiento Σ parecería ser certera y posiblemente inconsistente.

[2]Corresponde a sus siglas en inglés Preference-based Argumentation Frame.

Si bien la definición de argumento parece ser muy similar a la propuesta para el sistema argumentativo P.L.R.B. la diferencia reside en la lógica subyacente. En el caso del sistema P.L.R.B. la lógica subyacente es la correspondiente a la Programación en Lógica, mientras que en la definición anterior es lógica proposicional.

Ya que la base de conocimiento es potencialmente inconsistente, luego los argumentos pueden estar en conflicto. En [RA06] se presenta una definición alternativa a 5.12, de modo de determinar si dos argumentos están en conflicto.

Definición 5.14 Sean $A_1 = (H_1, h_1)$ y $A_2 = (H_2, h_2)$ dos argumentos.

- A_1 undercuts[3] A_2 si $\exists h'_2 \in H_2$ tal que $h_1 \equiv \neg h'_2$.
- A_1 ataca a A_2 si y solamente si A_1 undercuts A_2 y $A_1 >>^{\texttt{Pref}} A_2$.

■

Con respecto a esta definición de ataque creemos que existen situaciones que no son manejadas en forma adecuada. Nótese que para que exista un ataque se requiere que la fórmula en conflicto sea parte de H. Esto hace que no se consideren fórmulas que pueden ser derivadas lógicamente pero que no están en H. Veamos un ejemplo.

Ejemplo 5.7 Sea B la base de conocimiento

$$\{a, d, (a \to c), (c \to \neg b), (d \to b), (b \to f)\}$$

Consideremos los siguientes dos argumentos:

$$A_1 = \left\langle \underbrace{\{a, (a \to c), (c \to \neg b)\}}_{H_1}, \underbrace{\neg b}_{h_1} \right\rangle$$

$$A_2 = \left\langle \underbrace{\{d, (d \to b), (b \to f)\}}_{H_2}, \underbrace{f}_{h_2} \right\rangle$$

Por definición, A_1 undercuts A_2 si $\exists h'_2 \in H_2$ tal que $h_1 \equiv \neg h'_2$. Así en este caso A_1 no undercuts A_2 ya que b no es una fórmula de H_2, aunque b es una consecuencia lógica de H_2. ■

[3]Se utiliza el término en inglés por no encontrar un vocablo adecuado en el idioma español.

Creemos que una definición que se adaptaría mejor sería:

A_1 undercuts A_2 si $\exists H_2' \subseteq H_2$ tal que $H_2' \vdash \neg h_1$, o alternativamente

A_1 undercuts A_2 si $\exists H_2' \subseteq H_2$ tal que $\bigwedge H_2' \equiv \neg h_1$.

Las definiciones de defensa y aceptabilidad están inspiradas en las propuestas por Dung.

Definición 5.15 Sean $S \subseteq A$ y $A_1 \in A$. S defiende a A_1 si y solamente si para cada argumento A_2 que ataca a A_1, existe algún argumento $A_3 \in S$ tal que A_3 ataca A_2. ■

En función de las definiciones dadas anteriormente se define la noción de cuando un argumento es aceptado. En este sentido, se definen tres categorías de argumentos:

- *Aceptados*
- *Rechazados*: aquellos atacados por argumentos aceptados.
- *En Suspenso*: aquellos argumentos que no son aceptados ni rechazados.

Definición 5.16 Un argumento $A_1 \in A$ es aceptado con respecto a un conjunto de argumentos $S \subseteq A$ si:

- $\nexists A' \in S$ tal que A' ataca a A_1; o bien
- $\forall A' \in S$ tal que A' ataca A_1 existe un argumento aceptado $A'' \in S$ tal que A'' ataca A'.

■

Finalmente, se presenta una teoría de prueba basada en diálogos, que fue inspirada en el trabajo de Prakken y Sartor [PS97] .

Definición 5.17 Un *diálogo* es una secuencia no vacía de movimientos: movimiento$_i$ = $(Jugador_i, Arg_i)$, $i \geq 0$ tal que:

1. Jugador$_i$=P sssi i es par, Jugador$_i$=C sssi i es impar.
2. Jugador$_0$=P and $Arg_0 = A$.
3. Si Jugador$_i$=Jugador$_j$=P e $i \neq j$, entonces $Arg_i \neq Arg_j$.
4. Si Jugador$_i$=P ($i > 1$) entonces Arg_i ataca a Arg_{i-1} y Arg_{i-1} no ataca a Arg_i.

5. Si Jugador$_i$=C entonces Arg_i ataca a Arg_{i-1}.

Un jugador gana un diálogo sssi termina el diálogo. ■

Al igual que la definición dada de secuencia legal en la semántica declarativa $\mathcal{GS}$, los jugadores juegan por turnos y P comienza el diálogo. Las diferencias se presentan en los argumentos permitidos en cada diálogo. Solo el jugador C tiene permitido en el diálogo introducir un argumento que bloquea al oponente. La definición de diálogo es similar a la de secuencia preferida completa, la diferencia reside en los movimientos legales que la conforman.

Definición 5.18 Un *árbol diálogo* es un árbol finito en el que cada rama es un diálogo. Un *subárbol candidato* es un subárbol de un árbol diálogo conteniendo todos los arcos de cada nodo AND (jugador P) y exactamente un arco de cada nodo OR (jugador O). Un *subárbol solución* es un subárbol candidato cuyas ramas son todas ganadas por P. El jugador P *gana un árbol diálogo* sssi el árbol diálogo tiene un subárbol solución. ■

Finalmente, se da la noción de cuándo un argumento está justificado.

Definición 5.19 Un argumento A' está *justificado* sssi existe un árbol de diálogo cuya raíz es A', ganado por el jugador P. ■

La noción de justificación a través del árbol de diálogo es análoga a la presentada en Prakken [Pra01, PV00].

Análisis y Discusión

El marco argumentativo propuesto en [AC97] es una extensión del propuesto por Dung[Dun93, Dun95], al que se le incorpora la noción de preferencia. La semántica declarativa es heredada de dicho sistema y la teoría de prueba está inspirada en el trabajo de Prakken[Pra01, PV00]. Existen algunas similitudes y diferencias con respecto a la teoría de prueba y a la semántica de P.L.R.B. a enumerar:

- A diferencia del marco de Dung, se define la relación de ataque en función de las relaciones de derrota y de preferencia. Introduce de este modo en su teoría de prueba, la idea de que un argumento no puede ser un derrotador por *bloqueo* (un argumento A' descalifica a otro A'' sssi A' ataca a A'' y A'' no ataca a A') si el que juega es el jugador P. El jugador C puede introducir esta clase de argumentos sin límite en el diálogo. Fundamenta esta

decisión en los papeles que juegan P y C en el diálogo. P debe justificar el argumento inicial, por lo que sus movimientos tienen que ser de derrota estricta; y C debe evitar que el argumento inicial sea justificado, por lo que sólo requiere movimientos de derrota.

- Esto también difiere del sistema P.L.R.B. ya que en una línea de argumentación aceptable los derrotadores son propios o si son por bloqueo no pueden derrotarse con otro por bloqueo. Veamos un ejemplo en donde el comportamiento de ambos sistemas es diferente,

  Ejemplo 5.8 Consideremos al siguiente conjunto de argumentos $A = \{H, I, J\}$ tal que I R H, H R J y H R I y H `Pref` J. Así en P.L.R.B. J está justificado pues si bien lo derrota H, I derrota por bloqueo a H. Este diálogo no es el legal en el sistema de Agmoud et al. ya que el jugador P no puede jugar un argumento por bloqueo. Así, el diálogo termina siendo ganado por el jugador C y por lo tanto, J no está justificado. ■

- Por otra parte, P.L.R.B. impone la restricción de que no puede reintroducirse un argumento en la línea de argumentación, sin embargo, en un diálogo solo existe tal restricción para el jugador P. Finalmente, no hay controles sobre si todos los argumentos introducidos en un diálogo por cada uno de los participantes es consistente.

- La semántica declarativa que presenta es una extensión de la de Dung, definida a través del punto fijo del concepto de defensa. La semántica declarativa definida en este trabajo para el sistema P.L.R.B. está basada en la teoría de juegos.

- El criterio ganador para un árbol diálogo consiste en que el jugador P gane cada diálogo en algún subárbol solución a diferencia de la P.L.R.B. que no requiere que todas las ramas sean ganadas por el P.

- La justificación se realiza sobre el argumento a diferencia de la P.L.R.B., en la que se garantiza el literal, sin importar cuál argumento lo sustenta.

5.4. Semántica Dialéctica de Jakobovits y Vermeir

En [JV99] se presenta una semántica basada en la dialéctica cuyo marco subyacente es el presentado por Dung y que fue analizado anteriormente. Dicha semántica propone una forma de diálogo en la que aparecen todos los elementos de un juego. Presentaremos algunas definiciones con el objeto de comparar dicha semántica con la introducida en esta tesis.

Definición 5.20 Una *marco de posición* correspondiente a un marco argumentativo[Dun95] $(\mathcal{A}, \rightsquigarrow)$ es un marco argumentativo $(\mathcal{P}, \rightsquigarrow^*)$ donde $\mathcal{P}$ consiste de un subconjunto consistente de $\mathcal{A}$. Los elementos de $\mathcal{P}$ son llamados *posiciones*. Un *jugador* es un miembro de $\{p, o\}$ donde p

es el proponente y o es el oponente. Para un jugador p, el adversario de p, denotado $\overline{\text{p}}$ se define como $\overline{p} = o$ y $\overline{o} = p$. Un movimiento en $\mathcal{P}$ es un par $[p, X]$ donde p es un jugador y $X \in \mathcal{P}$. Para un movimiento $m = [p, X]$, usaremos $pl(m)$ para notar p y $pos(m)$ para notar X. $\rightsquigarrow$ es la relación de ataque entre argumentos definida por Dung en [Dun95]. ■

La clase de disputa que se permite está definida en función de las siguientes definiciones.

Definición 5.21 Un *tipo de diálogo* es una tupla $(\mathcal{P}, \rightsquigarrow^*, \phi)$ donde $(\mathcal{P}, \rightsquigarrow^*)$ es un marco de posición y $\phi : \mathcal{P}^* \rightarrow 2^{\mathcal{P}}$ es una función de movimientos legales. Un *diálogo* d en $(\mathcal{P}, \rightsquigarrow^*, \phi)$ es cualquier secuencia contable $d_0, d_1, \ldots$ de movimientos en $\mathcal{P}$ que satisface, para todo $i \in \mathbb{N}$:

1. pl($di + 1$)= $\overline{\text{pl}(di + 1)}$
2. pos($di + 1$)$\in \phi$(pos(d_0),..., pos(d_i))
3. pos($di + 1$)$\rightsquigarrow^*$pos(d_i)) and
4. pos($d0$)=p.

■

El tipo de diálogo que se permite lo debe comenzar el proponente (punto 4 de la definición 5.21) y en forma alternada los jugadores deben presentar sus argumentos(punto 1 de la definición 5.21), los que deberán ser legales y atacar al último movimiento del adversario(puntos 2 y 3 de la definición 5.21).

Los autores presentan varios criterios de triunfo. El primero exige que cada diálogo termine con un movimiento del proponente.

Definición 5.22 Sea $D = (\mathcal{P}, \rightsquigarrow^*, \phi)$ un tipo de diálogo. Un diálogo d es *ganado* por p si d es finito y termina con un movimiento $[p, X]$, tal que $(\rightsquigarrow^* X) \cap \phi(d) = \varnothing$, i.e., el diálogo no puede ser continuado. Un jugado p que no gana d se dice que *pierde* d. ■

Así, el jugador que gana un diálogo deja sin movidas legales al adversario, i.e., el adversario no tiene nada para decir. Nótese que al igual que en P.L.R.B. y en el sistema de Prakken analizado anteriormente, el diálogo debe ser finito. A continuación se presenta un segundo criterio de éxito.

Definición 5.23 Sea $D = (\mathcal{P}, \rightsquigarrow^*, \phi)$ un tipo de diálogo. Una *estrategia ganadora* S de D para una posición $X \in \mathcal{P}$ es un conjunto finito de diálogos finitos sobre X en D cerrado con respecto a los prefijos, que satisface:

1. $\forall d \in S,\ \exists d',\ \ d' > d$, tal que d' es ganado por el proponente.

2. $\forall d[p, X] \in S, \forall Y \in \rightsquigarrow X \cap \phi(d[p, X]), \exists d',\ d' > d[p, X][o, Y]$ tal que $d' \in S$.

$d_1 > d_2$ denota que d_2 es un prefijo de d_1. ■

Una estrategia ganadora garantiza que todas las secuencias son ganadas por el proponente o que el proponente contraargumentará el movimiento del oponente en cada etapa del juego. El oponente debe jugar todos los movimientos para contraatacar al proponente.

Un tercer criterio es especificado por los autores.

Definición 5.24 Sea $d_1, \ldots d_n$ un diálogo en un tipo de diálogo $D = (\mathcal{P}, \rightsquigarrow^*, \phi)$. d_n es ganador en d si $\forall d_{n+1}$ tal que dd_{n+1} está en el diálogo D,

1. d_{n+1} no es ganador en dd_{n+1}; y

2. $\exists d_{n+2}$ tal que d_{n+2} es ganador en $dd_{n+1}d_{n+2}$.

Una posición $X \in \mathcal{P}$ es ganadora en D si y sólo si el movimiento $[p, X]$ es ganador en el diálogo $[p, X]$. ■

La definición anterior define el criterio de triunfo para las posiciones en un diálogo, permitiendo de este modo, trabajar con diálogos infinitos. Los autores muestran un ejemplo en el que según el criterio utilizado, la semántica varía.

Ejemplo 5.9 [JV99] Sean los argumentos A y B tal que ambos se derrotan mutuamente. Dado que el diálogo es infinito, el proponente no gana el diálogo según el primer criterio de triunfo. Tampoco lo gana según el segundo criterio de triunfo, por la misma razón. Sin embargo, la posición del proponente, supongamos A, es ganadora según el tercer criterio. Claro que la posición B también es ganadora según el tercer criterio, dando en este caso una semántica crédula. ■

Así, la semántica está determinada por la combinación del tipo de diálogo y del criterio de triunfo. Por esta razón los autores introducen dos teorías de prueba a través de juegos. En la primer teoría de prueba evitan los argumentos que se atacan a sí mismos y aquellos argumentos inútiles, i.e., argumentos que son atacados por argumentos mencionados anteriormente. Estas ideas son reflejadas en la función movimientos-legales.

Definición 5.25 Sea $(\mathcal{P}, \rightsquigarrow^*)$ un marco de posición. La función *movimiento-legal* $\psi_{(\mathcal{P},\rightsquigarrow^*)}$: $\mathcal{P} \to 2^{\mathcal{P}}$ se define como sigue: $\forall Y_0 \ldots Y_i \in \mathcal{P}^*$,

$$\psi_{(\mathcal{P},\rightsquigarrow^*)}(Y_0 \ldots Y_i) = \mathcal{P} \backslash (\{X | X \rightsquigarrow^* X\} \cup \bigcup_{j=0}^{i} Y_j \rightsquigarrow^*).$$

■

La definición formal de la primer teoría de prueba es la siguiente.

Definición 5.26 Sea $AF = (\mathcal{A}, \rightsquigarrow)$ un marco argumentativo. Un *diálogo de único argumento útil* en AF es un diálogo en el tipo de diálogo $(\mathcal{A}', \rightsquigarrow, \psi_{(\mathcal{A},\rightsquigarrow^*)})$, donde $\mathcal{A}' = \{\{a\} | a \in \mathcal{A}\}$ y $\psi_{(\mathcal{A},\rightsquigarrow^*)}$ designa los movimientos que no son inútiles o que se atacan a sí mismos. ■

Nótese que la función de los movimientos legales no controla si hay inconsistencia entre los argumentos a favor del argumento inicial o en contra del argumento inicial. Esto se debe a que el control se realiza hacia adelante, i.e., evitamos argumentos que sean atacados por argumentos que ya están en el diálogo, y no hacia atrás, i.e., no se controla si el argumento que incorporamos contradice alguno de los argumentos anteriores.

Los autores avanzan con una restricción extra sobre la definición anterior evitando el reuso de los argumentos a través del concepto de *defendible*.

Definición 5.27 Sea $AF = (\mathcal{A}, \rightsquigarrow)$ un marco argumentativo. $\forall T \subseteq \mathcal{A}$, $\forall a \in \mathcal{A}$,

- $def_{AF}(a, T)$ si $a \in T$
- $def_{AF}(a, T)$ si $\forall b \rightsquigarrow a$, $\neg def_{AF}(b, \{a\} \cup T)$.

Un argumento a es *defendible* en AF si y solamente si $def_{AF}(a, \varnothing)$. ■

T mantiene el historial de los argumentos que se fueron analizando. Así cada argumento es asociado a la historia que lleva a su análisis. Un argumento A será defendible con respecto al conjunto historial de argumentos, si ninguno de los atacantes de A es defendible con respecto al historial de A unido A.

Teorema 5.1 *Sea $AF = (\mathcal{A}, \rightsquigarrow)$ un marco argumentativo. $\forall a \in \mathcal{A}$, a es un argumento defendible en AF si y solamente si existe una estrategia ganadora para $\{a\}$ en un tipo de diálogo de argumento único útil $(\mathcal{A}', \rightsquigarrow, \psi_{(\mathcal{A}',\rightsquigarrow)})$.*

La segunda teoría de prueba que introducen, tiene como motivación que un jugador pueda en un movimiento jugar varios argumentos juntos que el oponente deberá contraatacar.

Definición 5.28 Sean $AF = (\mathcal{A}, \rightsquigarrow)$ un marco argumentativo y $\mathcal{P}$ el conjunto de subconjuntos consistentes de $\mathcal{A}$. El tipo de diálogo $(\mathcal{P}, \rightsquigarrow, \psi_{(\mathcal{P},\rightsquigarrow)})$ es llamado *tipo de diálogo de múltiples argumentos útiles.* ■

La diferencia está en que los elementos de $\mathcal{P}$ son conjuntos consistentes de argumentos que pueden ser jugados en forma simultánea.

Ejemplo 5.10 [JV99] Supongamos que tenemos el siguiente marco argumentativo $AF = (\mathcal{A}, \rightsquigarrow)$:

- $\mathcal{A} = \{a, b, c, d, e, f\}$
- $\{b \rightsquigarrow a,\ d \rightsquigarrow a,\ c \rightsquigarrow b,\ f \rightsquigarrow c,\ f \rightsquigarrow e,\ e \rightsquigarrow f,\ e \rightsquigarrow d\}$

El argumento a no es defendible en AF pues, para que $def_{AF}(a, \varnothing)$, debe ser que $\neg def_{AF}(b, \{a\})$, lo que requiere $def_{AF}(c, \{a, b\})$, que es verdadero solo si $\neg def_{AF}(f, \{a, b, c\})$, que a su vez requiere $def_{AF}(e, \{a, b, c, f\})$ que es falso ya que $def_{AF}(f, \{a, b, c, f, e\})$. Por el teorema anterior no hay una estrategia ganadora. Sin embargo, los autores muestran que utilizando más de un argumento por jugada obtenemos otro resultado. Supongamos que el proponente propone $[p, \{a, c, e\}]$. El oponente no tiene más movimientos. La posición es defendible en el tipo de diálogo de múltiples argumentos útiles $(\mathcal{P}, \rightsquigarrow, \psi_{(\mathcal{P},\rightsquigarrow)})$. Así el proponente, puede ganar si juega todos sus argumentos en conjunto. ■

Análisis y Discusión

- En la definición 5.22, sólo se considera al diálogo, el equivalente a una secuencia completa o a una línea de argumentación, mientras que en P.L.R.B. se analiza al juego o al árbol dialéctico completo. Bajo el criterio presentado por los autores, se considera la secuencia completa donde el jugador que participa último deja sin argumentos a su adversario. Asimismo exige que cada secuencia sea ganada por el proponente.
- Con respecto a la definición de estrategia ganadora 5.23, esta noción considera a todos los diálogos, cerrados con respecto a los prefijos, símil a $P_{G(\langle \mathcal{A}, h \rangle, \mathcal{P})}$. En estos diálogos exige que el oponente juegue todos los posibles argumentos que contraataquen al último argumento del proponente y que el proponente contraataque a cada uno de estos argumentos del oponente. Sin embargo, continúa con la obligación de que el proponente debe ganar cada diálogo. Veamos un ejemplo que diferencia los sistemas.

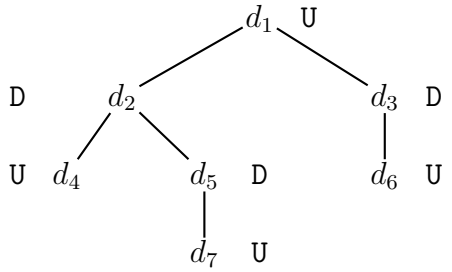


Figura 5.1: Ejemplo de un árbol dialéctico para mostrar diferencias entre la semántica $\mathcal{GS}$ y el sistema propuesto por [JV99].

Ejemplo 5.11 Supongamos el árbol dialéctico que se muestra en la Figura 5.1. El conjunto de las secuencias cerrado bajo prefijo es

$$\{\epsilon,\ d_1,\ d_2,\ d_3,\ d_1d_2,\ d_1d_3,\ d_1d_2d_4,\ d_1d_2d_5,\ d_1d_2d_5d_7,\ d_1d_3d_6\}$$

Bajo el criterio de [JV99] este conjunto no es una estrategia ganadora porque todas las secuencias excepto $d_1d_2d_5d_7$ son ganadas por el proponente o bien son prefijos de secuencias ganadas por el proponente. Sin embargo, $d_1d_2d_5d_7$ no es prefijo de ninguna secuencia ganada por el proponente.

No obstante bajo el criterio definido para la P.L.R.B. el conjunto de las secuencias completas

$$\{d_1d_2d_4,\ d_1d_2d_5d_7,\ d_1d_3d_6\}$$

es una estrategia ganadora, ya que o bien la secuencia es ganada por el proponente o bien difiere de otra secuencia que es ganada por el proponente en posición impar ($d_1d_2d_5d_7$ difiere de $d_1d_2d_4$ en la posición 3). ■

Así este criterio no se ajusta tampoco a nuestro sistema en estudio.

- El último criterio ganador tiene en cuenta las posiciones ganadoras, permitiendo que los diálogos sean infinitos a diferencia de la P.L.R.B. y de los dos criterios anteriores analizados. Sin embargo, no restringe a los argumentos jugados.

- Cuando nos referimos a un diálogo en que solo analizamos un única respuesta a cada argumento, el criterio de triunfo en el que el diálogo debe terminar con un movimiento del proponente y que el oponente no tenga contraargumentos que lo invaliden, es el que mejor se ajusta. Sin embargo, si consideramos que un agente puede, antes de argumentar realiza un árbol dialéctico introspectivo con el objeto de realizar su mejor jugada, el agente modelará una disputa en la que se consideren todas las respuestas para cada una de las jugadas y por lo tanto, este criterio de triunfo en el que solo se exige que gane el proponente cada diálogo no es el adecuado. Las estrategias ganadoras del trabajo [JV99] no se ajusta a la P.L.R.B., ya que analizan diálogos en forma aislada, nuestro equivalente a secuencias completas, y no el juego o árbol dialéctico en su totalidad.

5.5. Definición alternativa de P.L.R.B. basada en la Teoría de Juegos

En este trabajo [VTS08] los autores proponen una visión alternativa del sistema P.L.R.B. basada en la Teoría de Juegos, que acerca al sistema a los desarrollos de sistemas de diálogos. Asimismo, reformulan el modo en que las respuestas dadas a las consultas son calculadas.

Presentaremos en forma resumida las definiciones más importantes sobre las que realizaremos la comparación. Los autores asumen en este trabajo que un *plr* es un conjunto finito de reglas.

Definición 5.29 Un *juego extensivo con información perfecta* $G = \langle N, H, P, (U_i)_{i \in N} \rangle$ consiste de:

- Un conjunto N de jugadores.
- Un conjunto H de secuencias (finita o infinita) que satisface las siguientes propiedades:
  - La secuencia vacía $\varnothing$ está en H.
  - Si $(a_k)_{k+1,\ldots,K} \in H$ (donde K puede ser infinito) y $L < K$ entonces $(a_k)_{k=1,\ldots,L} \in H$.
  - Si una secuencia infinita $(a_k)_{k+1}^{\infty}$ satisface $(a_k)_{k=1,\ldots,L} \in H$ para cada entero positivo L, entonces $(a_k)_{k+1}^{\infty} \in H$.

  Los miembros de H se denominan *historias*. Cada componente a_k de una historia es una acción tomada por el jugador. Una historia $(a_k)_{k=1,\ldots,K} \in H$ es *terminal* si es infinita o no existe a_{K+1} tal que $(a_k)_{k=1,\ldots,K+1} \in H$. El conjunto de historias terminales lo denotaremos con Z.
- Una función $P : H \backslash Z \to N$, que indica para cada historia en H cuál jugador toma una acción después de la historia.
- Funciones $U_i : Z \to \Re$ para $i \in N$ que da para cada historia terminal y cada jugador, la utilidad después de la historia.

■

El conjunto H puede ser visto como el árbol con raíz vacía, sus nodos etiquetados por la función P y las hojas etiquetadas por las funciones U_n. Un juego es finito si H es finito. Para cada historia no terminal h el jugador $P(h)$ elije una acción del conjunto $A(h) = \{a : (h, a) \in H\}$.

Definición 5.30 Un *juego garantizado* para un literal l es un juego extensivo con información perfecta con dos jugadores: Proponente y Oponente. El juego se define como sigue:

- $P(\varnothing) = Proponente$
- Las acciones que el proponente puede tomar en la raíz del árbol son todos los argumentos de la forma $\langle \mathcal{A}, l \rangle$
- Las acciones después de una historia no terminal h son los argumentos $\langle \mathcal{A}', q \rangle$ tal que $\langle \mathcal{A}'', p \rangle$ es derrotado por $\langle \mathcal{A}', q \rangle$, que notaremos $\langle \mathcal{A}'', p \rangle \leq \langle \mathcal{A}', q \rangle$ siendo $\langle \mathcal{A}'', p \rangle$ el último componente en h.
- La utilidad para el proponente asume el valor ganador 1 en la historia $h \in Z$ si la longitud de h es impar y -1 de otro modo. La utilidad para el oponente es -1 por la utilidad del proponente.

■

Una estrategia para un jugador provee un plan de acciones en el que el jugador tiene una posible respuesta a cada historia no terminal en el juego. En otras palabras, el jugador puede contestar cada discusión no terminada a su contrincante.

Definición 5.31 Una *estrategia* para un jugador $i \in N$ en un juego extensivo con información perfecta $\langle N, H, P, (U_i)_{i \in N} \rangle$ es una función que asigna una acción en $A(h)$ a cada historia no terminal $h \in H \backslash Z$ para la cual $P(h) = i$. ■

Definición 5.32 Una *estrategia ganadora* para un jugador $i \in N$ en un juego garantizado es una estrategia que lleva a una historia terminal $z \in Z$ a tener utilidad 1 para i, sin importar cuales son las acciones del otro jugador. ■

Una estrategia es ganadora para un jugador si la estrategia conduce a alguna de las historias terminales a ser ganada por el jugador, es decir que la función de utilidad de esa historia terminal sea 1. Bajo esta definición podrían existir árboles garantizados en los que no hubiese ganadores.

Dados estos conceptos estamos en condiciones de determinar cuando un literal es consecuencia de un programa. Sea l la consulta a un *plr* P. Las posibles respuestas a l bajo P son YES, UNDECIDED y NO, pero difiere del sistema ya estudiado en el modo en que se llega a ellas.

Para responder a la consulta l se analizan dos juegos asociados. En primer lugar, se analiza el juego garantizado para el literal l y luego el juego garantizado para el complementos del

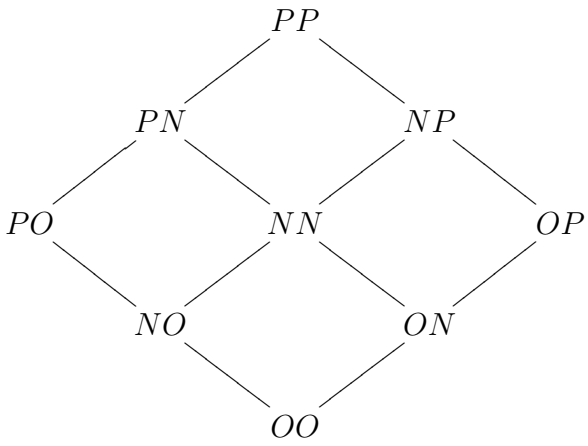


Figura 5.2: Gráfico correspondiente al conjunto parcialmente ordenado de los posibles valores de verdad en respuesta a una consulta l.

literal l, es decir para $\bar{l}$, en los que el Proponente y el Oponente intercambian sus roles. El Oponente empieza el juego eligiendo un argumento para $\bar{l}$. A partir de este análisis presentamos las posibles respuestas a una consulta, según quienes tengan estrategias ganadoras en los diferentes juegos:

Juego Garantizado para l Comienza Proponente	Juego Garantizado para $\bar{l}$ Comienza Oponente	Respuesta a la consulta l
Proponente	Proponente	YES
Proponente	Oponente	UNDECIDED
Proponente	Ninguno	YES
Oponente	Proponente	UNDECIDED
Oponente	Oponente	NO
Oponente	Ninguno	NO
Ninguno	Proponente	YES
Ninguno	Oponente	NO
Ninguno	Ninguno	UNDECIDED

Estas respuestas forman un conjunto parcialmente ordenado con primer elemento OO y último elemento PP (ver Figura 5.2), donde cada par de letras indican quién gano el juego para l y quién ganó el juego para $\bar{l}$. P corresponde al Proponente, O al oponente y N a ninguno de ellos, indicando quién tiene una estrategia ganadora.

Si el Proponente tiene una estrategia ganadora tanto cuando comienza el juego con un argumento para l y como cuando comienza el juego su Oponente con un argumento para $\bar{l}$ entonces en este caso la respuesta a la consulta debería ser YES, ya que el Proponente puede ganar al menos una discusión al Oponente sobre l. En modo inverso ocurre cuando es el Oponente el

que tiene estrategias ganadoras tanto para l como para $\bar{l}$.

Los autores avanzan en la definición restringiendo las historias a secuencias sin reintroducción de argumentos o subargumentos (falacias en la dialéctica). Esto tiene dos consecuencias fundamentales. La primera es que los árboles garantizados son finitos y la segunda es que los posibles valores de verdad que se pueden dar como respuesta se circunscriben a cuatro: PP,PO,OP y OO.

Análisis y Discusión

En lo que sigue haremos un análisis comparativo del los fundamentos de las definiciones presentadas en [VTS08] y P.L.R.B..

- La noción de historias H presentado por los autores es equivalente a la definición de conjunto de secuencias cerrada bajo prefijo dada para la semántica $\mathcal{GS}$. El concepto de historia terminal es equivalente al de secuencia completa. Estas equivalencias tiene su razón en la representación del árbol dialéctico.

- Los autores presentan la noción de estrategia ganadora para un jugador, en la que una historia final termina con una utilidad de 1 para ese jugador. En el caso de $\mathcal{GS}$ no existe la idea de utilidad del juego. Simplemente, se determina en forma implícita que todo nodo hoja de árbol de derivación, equivalente al último elemento en una secuencia completa es no derrotada.

- Asimismo difieren en la noción de estrategia ganadora en la estructura sobre la que se define. En el primer caso, se analiza la forma de una historia final, en cambio en el segundo caso, se requiere el análisis de todas las secuencias completas. Haciendo una analogía con el árbol dialéctico, en un caso se analiza solamente un camino de raíz a hoja y en el otro el árbol aunque no necesariamente en forma completa. En el trabajo[VTS08], si al menos *una* de las historias terminales es ganada, entonces el juego es ganado. Bajo el criterio del trabajo presentado en esta tesis, debemos analizar todas las secuencias completas para mantener las propiedades de sensatez y completitud con respecto a la teoría de prueba de la P.L.R.B..

- Si nos restringimos a la situación en la que un árbol dialéctico es ganado bajo el criterio de triunfo dado para $\mathcal{GS}$, los valores de verdad PP y OO, que se plantean en el trabajo [VTS08] se reducen a las respuestas YES y NO como era esperable.

- Los resultados presentados a continuación muestran que para la P.L.R.B. no es necesario analizar los dos juegos PP o OO. Si el primer juego es ganado por el jugador J, el segundo

también es ganado por el jugador J. Esto es, no es necesario bajo la teoría de prueba de la P.L.R.B. realizar ambos árboles.

Lema 5.1 *Sea $s = \langle \mathcal{A}, h\rangle s_2 \ldots s_n$ una secuencia legal completa y preferida de $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$, entonces la secuencia legal completa generada a partir de s anteponiendo $\langle \mathcal{A}', \overline{h}\rangle$ es par.*

Demostración: Haremos la demostración por casos:

- Si $\langle \mathcal{A}', \overline{h}\rangle$ es mejor que $\langle \mathcal{A}, h\rangle$ según la relación de preferencia entonces no existe ninguna secuencia legal de la forma $\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle\ s'$.
- Si $\langle \mathcal{A}', \overline{h}\rangle$ no mejor que $\langle \mathcal{A}, h\rangle$ según la relación de preferencia entonces podemos generar una secuencia de la forma $\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle\ s'$. Si dicha secuencia es legal, entonces es par, ya que $\langle \mathcal{A}, h\rangle\ s'$ es por hipótesis preferida y completa. Si no es legal entonces:
  1. $\Pi \cup \mathcal{A}' \cup s_2^{\mathcal{A}} \cup s_4^{\mathcal{A}} \cup s_{n-1}^{\mathcal{A}} \models \bot$, n impar por hipótesis. Sea i el menor índice en la secuencia tal que $\Pi \cup \mathcal{A}' \cup s_2^{\mathcal{A}} \cup s_4^{\mathcal{A}} \cup \ldots \cup s_i^{\mathcal{A}} \models \bot$. Dado que $\Pi \cup s_2^{\mathcal{A}} \cup s_4^{\mathcal{A}} \cup \ldots \cup s_{n-1}^{\mathcal{A}} \not\models \bot$ el argumento en conflicto es $\langle \mathcal{A}', \overline{h}\rangle$. Luego truncamos la nueva secuencia en $i-1$ con el objeto de hacerla legal, obteniendo $\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_{i-1}$. El índice $i-1$ es impar (es decir la secuencia $\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_{i-1}$ es impar), por lo tanto
  $$\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_{i-1}$$
  es par.
  2. Existe algún s_i, en la secuencia $\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_n$ que es subargumento de $\langle \mathcal{A}', \overline{h}\rangle$.
     i **es par:** La secuencia se trunca en $i-1$. Así, dado que la secuencia $\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_{i-1}$ es impar, la secuencia $\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_{i-1}$ es par.
     i **es impar:** Por definición de secuencia legal debe ser el caso que
     $$\Pi \cup \mathcal{A}' \cup s_2^{\mathcal{A}} \cup s_4^{\mathcal{A}} \cup \ldots \cup s_{n-1}^{\mathcal{A}} \not\models \bot$$
     y
     $$\Pi \cup \mathcal{A} \cup s_3^{\mathcal{A}} \cup s_5^{\mathcal{A}} \cup \ldots \cup s_n^{\mathcal{A}} \not\models \bot$$
     Por ser $s_i^{\mathcal{A}}$ y $s_{i-1}^{\mathcal{A}}$ contraargumentos entonces $\Pi \cup s_{i-1}^{\mathcal{A}} \cup s_i^{\mathcal{A}} \models \bot$ y por hipótesis $s_i^{\mathcal{A}} \subseteq \mathcal{A}'$. Luego $\Pi \cup \mathcal{A}' \cup s_{i-1}^{\mathcal{A}} \models \bot$. Así para que generar una secuencia legal, la secuencia original debe ser truncada en $i-2$:
     $$\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle s_2 s_3 s_4 \ldots s_{i-2}$$
     que es par.
  3. Los argumentos $\langle \mathcal{A}', \overline{h}\rangle$, $\langle \mathcal{A}, h\rangle$ y s_2 son derrotadores por bloqueo. Luego la secuencia resultante $\langle \mathcal{A}', \overline{h}\rangle\langle \mathcal{A}, h\rangle$ es par.

■

Lema 5.2 *Sean $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ un juego que inicia y gana el jugador J, y $\langle \mathcal{A}', \overline{h}\rangle$ un argumento tal que $\langle \mathcal{A}, h\rangle$ es mejor por la relación de preferencia que $\langle \mathcal{A}', \overline{h}\rangle$. Entonces el juego $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ comenzado por $\overline{J}$ con $\langle \mathcal{A}', \overline{h}\rangle$ es ganado también por el jugador J.*

Demostración: Sea $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ un juego que inicia y gana el jugador J. Por definición las secuencias son vacías o comienzan con $\langle \mathcal{A}, h\rangle$. Estas últimas tienen una ocurrencia por cada contraargumento que derrote (propio o por bloqueo) su movimiento anterior. Sea $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ el juego comenzado por $\overline{J}$ con el argumento $\langle \mathcal{A}', \overline{h}\rangle$. Como por hipótesis $\langle \mathcal{A}, h\rangle$ es mejor que $\langle \mathcal{A}', \overline{h}\rangle$ por la relación de preferencia, el juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ no contiene a $\langle \mathcal{A}', \overline{h}\rangle$ como segundo elemento de ninguna de las secuencias. Ahora, el juego $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ contiene al menos una secuencia con $\langle \mathcal{A}, h\rangle$.

1. Si $\langle \mathcal{A}', \overline{h}\rangle$ no aparece en ninguna secuencia de $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ entonces el juego $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ es ganado por J ya que las secuencias completas cuyo segundo elemento es $\langle \mathcal{A}, h\rangle$ tiene longitud par, o bien difieren en posición par con otras secuencias completas, teniendo en cuenta que las secuencias del juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ difieren en posición impar.
2. Si $\langle \mathcal{A}', \overline{h}\rangle$ aparece en alguna secuencia de $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ entonces debe aparecer en posición par (jugada del oponente a J) ya que las secuencias respetan la consistencia de los movimientos por jugador. Sea tal secuencia

   $$s = \langle \mathcal{A}, h\rangle s_2 \dots s_{2n-1} \langle \mathcal{A}', \overline{h}\rangle s_{2n+1} \dots s_m.$$

   Por lo tanto, el juego $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ contiene la secuencia $\langle \mathcal{A}', \overline{h}\rangle \langle \mathcal{A}, h\rangle s_2 \dots s_{2n-1}$ ya que la secuencia completa original no es legal por reintroducir un argumento. Por lo tanto, esta secuencia es par y si en $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ existía una secuencia completa que difería en posición impar, en $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ no existe tal secuencia porque o bien difiere en posición par o bien no difieren ya que fue truncada.

Por lo tanto, en ningún caso se verifica una estrategia ganadora en $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ para $\overline{J}$, así J gana el juego. ■

Lema 5.3 *Sean $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ un juego que inicia y gana el jugador J, y $\langle \mathcal{A}', \overline{h}\rangle$ un argumento tal que $\langle \mathcal{A}', \overline{h}\rangle$ es mejor por la relación de preferencia que $\langle \mathcal{A}, h\rangle$. Entonces el juego $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ comenzado por $\overline{J}$ con $\langle \mathcal{A}', \overline{h}\rangle$ es ganado también por el jugador J.*

Demostración: Sea $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ un juego que inicia y gana el jugador J. Dado que $\langle \mathcal{A}', \overline{h}\rangle$ es mejor que $\langle \mathcal{A}, h\rangle$ por la relación de preferencia, el juego $G(\langle \mathcal{A}, h\rangle, \mathcal{P})$ contiene a las secuencias que corresponden al juego $G'(\langle \mathcal{A}', \overline{h}\rangle, P)$ pero con $\langle \mathcal{A}, h\rangle$ como primer

elemento. El juego $G'(\langle \mathcal{A}', \overline{h} \rangle, P)$ es comenzado por $overline J$ y como $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ es ganado por J, el juego $G'(\langle \mathcal{A}', \overline{h} \rangle, P)$ no puede ser ganado por $\overline{J}$, ya que de otro modo su árbol dialéctico equivalente tendría la raíz etiquetada con U. Así el juego es ganado por J. ■

Lema 5.4 *Sean $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ un juego que inicia y gana el jugador J, y $\langle \mathcal{A}', \overline{h} \rangle$ un derrotador por bloqueo de $\langle \mathcal{A}, h \rangle$. Entonces el juego $G'(\langle \mathcal{A}', \overline{h} \rangle, P)$ comenzado por $\overline{J}$ con $\langle \mathcal{A}', \overline{h} \rangle$ es ganado también por el jugador J.*

Demostración: Sea $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ un juego que inicia y gana el jugador J. Por definición las secuencias son vacías o comienzan con $\langle \mathcal{A}, h \rangle$. Estas últimas tienen una ocurrencia por cada contraargumento que derrote (propio o por bloqueo) su movimiento anterior. Sea $G'(\langle \mathcal{A}', \overline{h} \rangle, P)$ el juego comenzado por $\overline{J}$ con el argumento $\langle \mathcal{A}', \overline{h} \rangle$. Como por hipótesis $\langle \mathcal{A}, h \rangle$ es un derrotador por bloqueo de $\langle \mathcal{A}', \overline{h} \rangle$, por lo tanto existen secuencias de la forma $\langle \mathcal{A}, h \rangle$ $\langle \mathcal{A}', \overline{h} \rangle$ s en $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$.

Si en $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ existen secuencias de la forma $\langle \mathcal{A}, h \rangle t$ siendo que $\langle \mathcal{A}', \overline{h} \rangle$ no es el primer elemento de t, se pueden considerar los siguientes casos:

- el primer elemento de t es un derrotador por bloqueo de su antecesor, en cuyo caso solo pertenecería a G' la secuencia par $\langle \mathcal{A}', \overline{h} \rangle \langle \mathcal{A}, h \rangle$, y que podría diferir solamente con el resto de las secuencias en posición impar si se cambiara la raíz.
- $\langle \mathcal{A}', \overline{h} \rangle$ está en t y por lo tanto t se trunca en ese punto. Caso similar al punto (2) del lema 5.2.

Ya que $\langle \mathcal{A}, h \rangle$ t pertenece a $G(\langle \mathcal{A}, h \rangle, \mathcal{P})$ que es ganado por el jugador J, entonces es una secuencia completa impar y por lo tanto,la secuencia $\langle \mathcal{A}', \overline{h} \rangle \langle \mathcal{A}, h \rangle$ t es par o bien $\langle \mathcal{A}, h \rangle$ t difiere de otra secuencia en posición impar, en cuyo caso $\langle \mathcal{A}', \overline{h} \rangle \langle \mathcal{A}, h \rangle$ t ahora difiere de esa secuencia (si existiera) en posición par.

Luego el juego comenzado por $\overline{J}$ con $\langle A', \overline{h} \rangle$ es ganado por J. ■

5.6. Conclusiones

En este capítulo se han descripto diferentes formalismos basados en argumentación rebatible, analizándose fundamentalmente la semántica de estos sistemas. Ninguna de las semánticas desarrolladas para los sistemas presentados se adapta en forma adecuada a la P.L.R.B..

El marco argumentativo de Dung presenta diferencias en cuanto al tratamiento de los derrotadores por bloqueo, los argumentos que se autoderrotan y en el manejo de los subargumentos entre otras.

El marco argumentativo basado en diálogos de Prakken tiene su motivación en la argumentación entre dos o más agentes en la que inclusive, uno de ellos puede persuadir al otro, a diferencia de la P.L.R.B. en donde la dialéctica modela el razonamiento de un solo agente. El formalismo trabaja sobre el árbol dialéctico en el que reconoce argumentos relevantes, realizando podas, y sobre el que hace un marcado con *in* y *out* equivalente a U y D.

El sistema argumentativo de Agmound y Cayrol es la convergencia de los dos sistemas anteriores. Bajo este sistema, el proponente no tiene permitido jugar un argumento que sea derrotador por bloqueo y además debe jugar todos sus posibles movimientos, a diferencia del oponente que solamente debe derrotar al último movimiento del proponente con exactamente un movimiento. El criterio ganador también difiere ya que exige que el proponente gane cada rama del subárbol solución. Finalmente, y al igual que el sistema de Prakken, la justificación se realiza sobre el argumento, a diferencia de la P.L.R.B. que garantiza al literal.

La semántica dialéctica de Jakobovits y Vermeir, está basada en el marco de Dung, y define diferentes estrategias ganadoras, en las que principalmente exige que cada diálogo sea ganado por el proponente. Por otra parte, permite bajo algunas especificaciones tratar con diálogos infinitos, considerando las posiciones ganadoras en un debate. Este sistema permite, por otra parte, la introducción de varios argumentos en forma simultánea en un diálogo.

Finalmente, la definición alternativa a P.L.R.B. basada en la teoría de juegos en [VTS08], presenta varias similitudes en lo referente al juego mismo. Sin embargo, difieren en la noción de estrategia ganadora, ya que solo analiza las historias terminales (equivalente a una secuencia completa). Las respuestas para garantizar a un literal ahora dependen de la solución de dos árboles. Las respuestas PP y OO se corresponden a *yes* y *no*. Por último, bajo la noción clásica de P.L.R.B. no es necesario la construcción de los dos árboles, ya que si el primero lo gana el jugador J, el segundo también lo gana el jugador J.

De lo expuesto anteriormente, ninguna de las semánticas definidas para otros sistemas son adecuadas para el nuestro.

Capítulo 6

Complejidad de la P.L.R.B.

Contenido

El razonamiento práctico es la meta de la argumentación sobre el conocimientos, luego es importante tener en cuenta la complejidad temporal del cómputo de la teoría de prueba de la P.L.R.B.. Asimismo, es importante en un formalismo analizar la clase de consultas que podemos realizar, i.e., su poder expresivo como lenguaje de consulta.

En este capítulo se realiza un estudio de estos temas y se presentan los resultados alcanzados. En la primer sección presentamos la motivación de este capítulo. En la sección 6.2, analizamos la P.L.R.B. a través de la semántica $\mathcal{GS}$ describiendo los problemas de decisión que son relevantes. Luego introducimos en el contexto de la P.L.R.B. la complejidad de los datos, la complejidad de expresión o de programa y la complejidad combinada.

En la sección 6.3, se presentan los resultados en complejidad relacionados con la existencia de argumentos y contraargumentos para un literal bajo un *plrb* y con la verificación de que un conjunto de reglas rebatibles sea un argumento. En la sección , analizamos la complejidad de datos para la P.L.R.B. y presentamos los resultados obtenidos para dos de los problemas de decisión definidos. Finalmente, resumimos las contribuciones de este capítulo y presentamos las conclusiones.

6.1. Motivación

La teoría de la complejidad es una herramienta importante para comparar diferentes formalismos y para ayudar a mejorar las implementaciones siempre que sea posible. Por esta razón, es importante analizar la complejidad computacional y el poder expresivo de la P.L.R.B.. El primero nos dice que tan difícil es responder a una consulta, mientras que el segundo da una caracterización precisa de los conceptos que pueden ser definibles como consultas.

Aunque la complejidad para los sistemas de razonamiento no monotónicos han sido estudiados en profundidad para diferentes formalismos tales como la lógica default, logica autoepistémica, circunscripción, abducción y P.L. [CS93, DEGV01] hasta el momento, existen pocos resultados publicados en cuanto a la complejidad de los sistemas argumentativos.

La situación puede ser explicada en parte, por el hecho de que, históricamente, las implementaciones de los sistemas argumentativos han sido limitados a áreas sin restricciones de tiempo real(ver [Ver98, GK97]). Recientemente, sin embargo, varias aplicaciones relacionadas con, por ejemplo, sistemas multiagentes y búsquedas en la web [ABCMB04, CM04a, CM04b, BAV04] , han sido desarrolladas e implementadas utilizando sistemas argumentativos.

La escalabilidad y robustez de tales enfoques dependen fuertemente de las propiedades computacionales de los algoritmos que lo computan. Por lo tanto, es esencial estudiar estas propiedades, con el objeto de expandir los campos de aplicación de los sistemas argumentativos.

Algunos resultados en complejidad computacional [DNT02, BC03, AC02, DBC02] han sido presentados sobre los marcos de argumentación abstractos [BDKT97, Dun95], basados en las semánticas de admisibilidad y preferencia. Sin embargo, estos resultados no se ajustan a la P.L.R.B., ya que su semántica es algo diferente.

Otro estudio importante de la complejidad computacional de los sistemas rebatibles ha sido el hecho en [Mah01]. Sin embargo, la teoría rebatible analizada en este trabajo difiere considerablemente de la P.L.R.B. en varios puntos, como la representación del conocimiento (hechos, reglas estrictas y reglas rebatibles y defendibles) y en su teoría de prueba.

Cuando se determina la complejidad de evaluar consultas en un lenguaje específico, distinguimos entre varias formas de complejidad de acuerdo a [Var82, PY97, DEGV01].

- La *Complejidad de los Datos* es la complejidad de evaluar una consulta específica en el lenguaje, cuando la consulta está fija y estudiamos la complejidad de aplicar esta consulta a base de datos arbitrarias; la complejidad es dada en función del tamaño de la base de datos.
- La *Complejidad de los Programas o de Expresión* aparece cuando una base de datos

específica se fija, y estudiamos la complejidad de aplicar las consultas representadas por expresiones arbitrarias en el lenguaje; la complejidad es dada en función de la longitud de la expresión.

- La *Complejidad Combinada* considera tanto a la consulta como a la instancia de la base de datos como variables de entrada.

Estas clases de complejidad permiten estudiar a los formalismos como lenguajes de consulta.

Lo expresado anteriormente, motivó el trabajo presentado este capítulo. Inicialmente, hemos analizado a la P.L.R.B. a través del estudio de la complejidad de algunos problemas de decisión importantes.

Ya que la P.L.R.B. construye los argumentos a partir del *plrb*, hemos considerado y evaluado dos cuestiones relevantes: "¿es un conjunto de reglas rebatibles un argumento para un literal bajo un *plrb*?", problema que se probó que es **P**-completo, y "¿existe un argumento para un literal bajo un *plrb*?", que se probó que está en **NP**. Asimismo, se trabajó sobre la existencia de contraargumentos.

Por otra parte, analizamos a la P.L.R.B. como lenguaje de consulta, definiendo las complejidades de los datos, expresión y combinada en el contexto de la P.L.R.B.. En particular, estudiamos la complejidad de datos de la respuesta a consultas, a fin de valorar las aplicaciones de la P.L.R.B. sobre las tecnologías de base de datos. Hasta donde conocemos la complejidad de datos no ha sido introducida en el contexto de los sistemas de argumentación.

6.2. Análisis sobre la Complejidad de $\mathcal{GS}$

P.L.R.B. es un sistema de razonamiento rebatible donde cada consecuencia de un *plr* es analizado considerando todos sus argumentos a favor y en contra. La semántica $\mathcal{GS}$, es una semántica basada en juegos que caracteriza tal razonamiento a través de dos conjuntos, V y F, siendo $V \cup \overline{F}$ el conjunto de todos los literales garantizados. Los literales restantes son los indecisos y corresponden a aquellos para los que no hemos podido garantizar ni al literal ni a su complemento.

Cuando consideramos a P.L.R.B. en relación a la semántica definida, existen dos problemas de decisión computacionales relevantes a analizar en el contexto de un *plrb* $\mathcal{P}$:

- GAMESAT: Decidir si existe un juego para un literal h ganado por el proponente P.
- NOWINGAME: Decidir si no existe ningún juego para el literal h ni para su complemento que sea ganado por el proponente P.

El primer problema involucra encontrar solo un juego que sea ganado por el proponente, pero para poder decidir NOWINGAME es necesario encontrar todos los juegos para el literal y su complemento y establecer que ninguno fue ganado por el proponente.

Una respuesta positiva a GAMESAT para un *plrb* $\mathcal{P}$ y un literal h implica que $h \in V$, siendo $\langle V, F\rangle$ el único G-modelo, i.e., $\mathcal{P} \models_{\mathcal{GS}} h$. Una respuesta positiva a GAMESAT para $\overline{h}$ significa que $h \in F$ en el G-modelo, i.e., $\mathcal{P} \models_{\mathcal{GS}} \overline{h}$.

El problema de decisión NOWINGAME para un *plrb* $\mathcal{P}$ y un literal h es equivalente a determinar si dado el G-modelo de $\mathcal{P}$, $\langle V, F\rangle$, $h \in Lit^{+} - \{V \cup F\}$, i.e., $\mathcal{P} \not\models_{\mathcal{GS}} h$ y $\mathcal{P} \not\models_{\mathcal{GS}} \overline{h}$. En este caso, tres situaciones interesantes pueden ser consideradas y establecen los siguientes problemas de decisión:

- *No existe un juego para un literal h, ni existe un juego para su complemento* $\overline{h}$. La familia de juegos para el literal h y para su complemento $\overline{h}$, $\mathcal{F}(h, \mathcal{P})$ y $\mathcal{F}(\overline{h}, \mathcal{P})$ respectivamente, son vacíos. h no tiene argumentos ni a favor ni en contra. Por lo tanto, el agente no tiene información sobre esa consulta.

- *No existe un juego para el literal h, y el conjunto no vacío de todos los juegos en la familia del complemento* $\overline{h}$ *son ganados por el oponente.* La familia de juegos del literal h, $\mathcal{F}(h, \mathcal{P})$, es vacía y la familia, no vacía, $\mathcal{F}(\overline{h}, \mathcal{P})$ contiene solamente juegos ganados por el oponente. h no tiene argumentos a favor y todos los argumentos de su complemento son rebatidos. Por lo tanto, el agente no tiene información sobre h y no puede defender a $\overline{h}$.

  De un modo similar, podemos definir el caso donde el agente no puede defender al literal h y no tiene información sobre su complemento.

- *Todos los juegos en las familias no vacías de h y su complemento* $\overline{h}$ *son ganados por el oponente.* $\mathcal{F}(h, \mathcal{P})$ y $\mathcal{F}(\overline{h}, \mathcal{P})$ no son conjuntos vacíos y todos los argumentos son rebatidos. El agente no puede defender ningún argumento ni a favor de h, ni a favor de $\overline{h}$.

Con el objeto de determinar la complejidad computacional de los problemas de decisión introducidos anteriormente, estudiaremos a la P.L.R.B. desde dos enfoques: la complejidad de los datos y la complejidad combinada [DEGV01, Var82] .

La complejidad combinada de un fragmento de la programación en lógica ha sido definida y usada en [DEGV01] :

> *Complejidad de (un fragmento de) la Programación en Lógica:* es la complejidad de chequear si dado un programa $\mathcal{P}$ variable y un átomo fijo cualquiera A, $\mathcal{P} \models A$.

Por otra parte, la noción de complejidad de los datos es heredada de la teoría de bases de datos relacional [Var82]. Actualmente, las bases de datos son la principal herramienta de

almacenamiento y recuperación de grandes conjuntos de datos. La complejidad de los datos nos permite estudiar P.L.R.B. como un lenguaje de consulta midiendo su complejidad enfocada en el tamaño de la base de datos y usando reglas estrictas y rebatibles para la inferencia.

La complejidad de datos es la medida clave para determinar la eficiencia de la implementación de los sistemas argumentativos basados en la tecnología de las bases de datos.

A los propósitos metodológicos y de temas de complejidad, es importante distinguir en *plrb* los datos de entrada de las reglas de inferencia.

Así, de aquí en más, denotaremos a los programas lógicos rebatibles $\mathcal{P} = \langle \Pi_F, \Pi_R \cup \Delta \rangle$, donde Π_F es un conjunto finito de hechos fijos, y $\Pi_R \cup \Delta$ es un conjunto finito de reglas fijas estrictas y rebatibles.

Haciendo una analogía con los conceptos de base de datos, Π_F representa la entrada a la base de datos, también llamada la parte definida por *extensión*, y $\Pi_R \cup \Delta$ son la reglas de inferencia, llamadas la parte definida por *comprensión* de la base de datos. Definimos una *consulta booleana* como un conjunto finito de reglas estrictas y rebatibles junto con un literal fijo L. El significado intuitivo pretendido de la definición de tal consulta es el siguiente: queremos saber si un literal L es consecuencia de $\Pi_R \cup \Delta$ junto con la base de datos Π_F a través de $\mathcal{GS}$.

Siguiendo los principios y nociones anteriores, en el contexto de la P.L.R.B. definimos la complejidad de datos, la complejidad de programa y la complejidad combinada como sigue.

Definición 6.1 (Complejidad de Datos, de Programa y Combinada)
Sean Ω alguno de los problemas de decisión introducidos anteriormente, $\mathcal{P} = \langle \Pi_F, \Pi_R \cup \Delta \rangle$ y $(\Pi_R \cup \Delta, L)$ una consulta:

- La *complejidad de datos* de Ω es la complejidad de Ω cuando la consulta está fija y la base de datos varía, i.e., los parámetros $\Pi_R \cup \Delta$ y L están fijos.
- La *complejidad del programa o expresión* de Ω es la complejidad de Ω cuando la instancia de la base de datos está fija y la consulta varía, i.e., el parámetro Π_F está fijo.
- La *complejidad combinada* de Ω es la complejidad donde cada parámetro Π_F, $\Pi_R \cup \Delta$ y L varían.

■

La complejidad combinada y de expresión son muy similares y rara vez se las diferencia. Por esta razón solamente analizaremos la complejidad de datos y la combinada.

Con el objeto de llevar a cabo este análisis de la complejidad primero nos enfocaremos en la complejidad de determinar si existe un argumento $\mathcal{A}$ para un literal L. Luego estudiaremos si el juego que comienza con el movimiento inicial $\mathcal{A}$ es ganado por el proponente.

6.3. La complejidad de computar argumentos

Los argumentos y contraargumentos son los movimientos en un juego y así el núcleo de la P.L.R.B.. El formalismo de Dung [Dun95] y algunas extensiones desarrolladas por [BC02, BC03, AC02], ofrecen una herramienta muy poderosa para el análisis abstracto del razonamiento rebatible. Sin embargo, estos enfoques operan con argumentos y su relación de ataque y derrota en un nivel abstracto, evitando, de este modo, tratar con el lenguaje lógico subyacente usado para construir los argumentos.

Por otro lado, P.L.R.B. construye los argumentos y analiza la relación de derrota. Así, estudiar el problema de decisión: "¿es un subconjunto arbitrario de reglas rebatibles un argumento para un literal bajo *plrb*?" es de gran importancia.

Este problema de decisión tiene tres partes considerando la definición de argumento:

- ¿Es L una consecuencia de $\Pi \cup \mathcal{A}$?
- ¿Es $\Pi \cup \mathcal{A}$ consistente?; y
- ¿Existe un subconjunto $\mathcal{A}'$ de $\mathcal{A}$ tal que es consistente con Π y que junto con Π derive L?

Antes de comenzar el análisis de la complejidad haremos dos restricciones. Por un lado, trabajaremos sobre programas lógicos fijos(sin variables), en vez de sobre *esquemas de programas*. Por otro lado, limitaremos nuestro estudio a *plrb finitos*. Así, de ahora en más, cada vez que se mencione a un *plrb* nos referiremos a un *plrb* fijo y finito.

Sean $\mathcal{P} = \langle \Pi_F, \Pi_R \cup \Delta \rangle$ un *plrb*, L un literal y $\mathcal{A} \subseteq \Delta$. La primer condición de la definición 4.7, que involucra el concepto de consecuencia rigurosa es $L \in Cn_R(\Pi \cup \mathcal{A})$.

En el capítulo 4 hemos introducido la transformación *definido* (definición 4.9 en página 9) de un *plrb* en un programa lógico definido proposicional, i.e., un programa lógico proposicional con solo cláusulas de Horn. Dicha transformación, y el siguiente lema serán utilizados para reducir las consecuencias rigurosas de un *plrb* en consecuencias de cláusulas de Horn proposicionales.

Lema 6.1 *Sean DP un programa lógico definido y $\mathcal{M}$ el modelo minimal de DP, luego $\mathcal{M} = Cn_R(DP)$.*

Nuestro objetivo es determinar la complejidad temporal de la siguiente verificación: $L \in Cn_R(\Pi \cup \mathcal{A})$. Con este propósito construiremos un programa lógico con solo cláusulas de Horn, que denotaremos $\mathcal{HP}(\Pi, \mathcal{A}, L)$ tal que $L \in Cn_R(\Pi \cup \mathcal{A})$ si y solo si $\mathcal{HP}(\Pi, \mathcal{A}, L) \models$ SI.

Supongamos que $A_1, \ldots, A_n$ son todos átomos en $\Pi \cup \mathcal{A}$. Definimos $\mathcal{HP}(\Pi, \mathcal{A}, L)$ como sigue:

$$\begin{aligned} \mathcal{HP}(\Pi, \mathcal{A}, L) \quad = \quad & definido(\Pi) \cup definido(\mathcal{A}) \cup \{\text{SI} \leftarrow definido(L)\} \cup \\ & \{\text{SI} \leftarrow definido(A_i), definido(\overline{A_i}) : 1 \leq i \leq n\} \end{aligned}$$

Aunque el problema de decisión SAT es **NP**-completo, chequear si un programa lógico definido $\mathcal{DP}$ satisface un átomo fijo A, i.e., $\mathcal{DP} \models A$, como el problema de decisión HORNSAT, "¿existe una asignación de verdad que satisfaga un conjunto de cláusulas de Horn?", son **P**-completos [DEGV01, PY97].

Lema 6.2 *$\mathcal{HP}(\Pi, \mathcal{A}, L)$ es una transformación de un plrb P en cláusulas de Horn proposicional tal que verificar si un literal L pertenece a $Cn_R(\mathcal{P})$ es equivalente a verificar si* SI *es derivado desde programa Horn proposicional transformado. Así, $L \in Cn_R(\mathcal{P})$ reduce a $\mathcal{DP} \models$ SI, siendo $\mathcal{DP}$ un programa Horn proposicional.*

Demostración: Con el objeto de probar nuestra afirmación, debemos establecer que:

1. $L \in Cn_R(\Pi \cup \mathcal{A})$ si y solo si $\mathcal{HP}(\Pi, \mathcal{A}, L) \models$ SI.

   Consideraremos dos casos:

   - $\Pi \cup \mathcal{A}$ es consistente.
     $L \in Cn_R(\Pi \cup \mathcal{A})$ si y solo si $definido(L) \in definido(Cn_R(\Pi \cup \mathcal{A}))$ si y solo si $definido(L) \in Cn_R(definido(\Pi \cup \mathcal{A}))$si y solo si, por lema 6.1, $definido(L)$ está en el modelo minimal de $definido(\Pi \cup \mathcal{A})$ si y solo si $\mathcal{HP}(\Pi, \mathcal{A}, L) \models$ SI por la definición de modelo minimal, la propiedad de monotonicidad y el uso de la regla $\text{SI} \leftarrow definido(L)$.
   - $\Pi \cup \mathcal{A}$ es inconsistente.
     $L \in Cn_R(\Pi \cup \mathcal{A}) = Lit$ si y solo si existe $i, 1 \leq i \leq n$, tal que $definido(L_i)$ y $definido(\overline{L_i})$ están en $definido(Cn_R(\Pi \cup \mathcal{A}))$ si y sólo si $definido(L_i)$ y $definido(\overline{L_i})$ están en el modelo minimal de $\mathcal{HP}(\Pi, \mathcal{A}, L)$ si y solo si $\mathcal{HP}(\Pi, \mathcal{A}, L) \models$ SI por definición de modelo minimal, propiedad de monotonicidad y el uso de la regla $\text{SI} \leftarrow definido(L_i), definido(\overline{L_i})$.

2. $\mathcal{HP}$ es calculado en espacio logarítmico: la transformación es muy simple y es posible de realizar en espacio logarítmico, ya que las reglas pueden ser generadas en forma inde-

pendiente una de otras excepto aquellas de la forma si$\leftarrow$ $definido(L_i), definido(\overline{L_i})$ que dependen del literal de entrada.

Por lo tanto, $\mathcal{HP}(\Pi, \mathcal{A}, L)$ es una reducción desde $L \in Cn_R(\Pi \cup \mathcal{A})$ a cláusulas de Horn proposicionales. ■

Teorema 6.1 *Sean $\mathcal{P} = (\Pi_F, \Pi_R \cup \Delta)$ un plrb, $\mathcal{A} \subseteq \Delta$, y L un literal. Determinar si $L \in Cn_R(\Pi \cup A)$ es **P**-completo.*

Demostración:

- *Membresía:* dado un programa lógico definido $\mathcal{P}$ el menor punto fijo T_P^∞ del operador T_P puede ser computado en tiempo polinomial [Pap94, DEGV01]: el número de iteraciones está limitado al número de reglas más uno. Cada paso de la iteración es factible en tiempo polinomial. Así encontrar el modelo minimal de un programa lógico con sólo cláusulas de Horn está en **P** [DEGV01].

  Por lema 6.2, $L \in Cn_R(\Pi \cup A)$ ha sido reducido a la programación en lógica proposicional. Por lo tanto, $L \in Cn_R(\Pi \cup A)$ está en **P**.

- *Hardness:* Las reglas de Horn son reglas estrictas en un *plrb*, y el modelo mínimo de un programa lógico definido $\mathcal{DP}$ es igual a $Cn_R(\mathcal{DP})$. Por lo tanto, aplicando la reducción por generalización, tenemos que $\mathcal{DP} \models L$ reduce a $L \in Cn_R(\mathcal{DP})$. La programación en lógica proposicional es **P**-completa [DEGV01]. Esto completa la prueba.

■

Hasta ahora hemos probado que la primer condición de la definición de argumento es **P**-completo. Ahora tendremos que analizar el resto de las condiciones que necesitamos para computar un argumento. Denotaremos la cardinalidad del lenguaje con $|Lit|$ y la cardinalidad del conjunto de las reglas rebatibles con $|\Delta|$.

En la figura 6.1, presentamos un algoritmo para verificar si un conjunto de reglas rebatibles que derivan al literal L es minimal con respecto a la inclusión de conjuntos.

Consideremos el peor caso para la condición de minimalidad, es decir, sabiendo que el argumento tiene a lo sumo $|\Delta|$ reglas rebatibles, i.e., Δ es un argumento para algún literal.

El cómputo de la condición de minimalidad involucra $|\Delta|$ bucles para verificar que $L \in Cn_R(\Pi \cup A')$, que está en **P**. Así este problema es resoluble en tiempo polinomial y, por lo tanto, está en **P**.

Finalmente, para chequear si el conjunto de reglas rebatibles es consistente bajo un *plrb*, debemos verificar que no existe ningún átomo tal que ese átomo y su complemento pertenecen a

Algoritmo: Minimal
Input: $\mathcal{A}$ un argumento para el literal L, y Π un conjunto de reglas estrictas.
Output: true si A es un argumento minimal para L, false de otro modo.

minimal=true
Aux= $\mathcal{A}$
While minimal and not $Aux = \varnothing$ do
seleccione $H \prec B \in$ Aux
$A' = \mathcal{A} - \{H \prec B\}$
if $L \in Cn_R(\Pi \cup A')$
then minimal=false
else Aux= Aux - $\{H \prec B\}$

Figura 6.1: Algoritmo que verifica si un conjunto de reglas rebatibles es minimal con respecto a la inclusión de conjuntos, para derivar un literal L.

$Cn_R(\Pi \cup A)$. En el peor caso, cuando $Cn_R(\Pi \cup A)$ es consistente, este algoritmo debe controlar cada átomo en la signatura de *plrb*. Así, chequear si es consistente es proporcional al número de átomos $|Lit|/2$ y por lo tanto está en **P**.

Teorema 6.2 *El problema de decisión "¿es un subconjunto de reglas rebatibles un argumento para un literal bajo un plrb?" es* ***P****-completo.*

Demostración:

- *Membresía*: De los desarrollos anteriores se deduce la membresía a **P**.
- *Hardness:* Empleamos una reducción desde $\mathcal{DP} \models L$, siendo $\mathcal{DP}$ un programa Horn proposicional. Considere la siguiente transformanción $r(\mathcal{DP}) = \mathcal{DP}' = \langle \Pi, \Delta \rangle$, donde $\Pi = \mathcal{DP}$ y Δ es vacío. r es una transformación que puede ser computada en espacio logarítmico tal que siempre que un literal L es derivado de un programa Horn proposicional $\mathcal{DP}$, el problema de decisión "¿es un subconjunto de $\Delta = \varnothing$ un argumento para L bajo $\mathcal{DP}'$?" termina en un estado de aceptación.

  L está en el modelo minimal de un programa Horn proposicional si y solo si $L \in Cn_R(\mathcal{DP})$ si y solo si $\varnothing$ es un argumento para L, ya que es minimal y consistente con Π, si y solo si "¿es un subconjunto de $\Delta = \varnothing$ un argumento para L bajo $\mathcal{DP}'$?" termina en un estado aceptador. De este modo, queda establecido que el problema de decisión "¿es un subconjunto de reglas rebatibles un argumento para un literal bajo un *plrb*?" es **P**-completo.

■

Nuestro objetivo final es determinar la complejidad del cálculo del conjunto de todos los argumentos bajo un *plrb*. Esto está motivado en que GAMESAT y NOWINGAME requieren para

jugar un juego, determinar todos los argumentos que derroten al argumento introducido en el movimiento anterior. Un subconjunto $A \subseteq \Delta$ podría ser un argumento potencial de diferentes literales en el lenguaje.

Analicemos algunos casos. Note que cuando Δ es vacío, i.e., no existe ninguna regla rebatible, el número máximo de argumentos es $\frac{|Lit|}{2}$ porque por definición Π es consistente y por lo tanto, existe como máximo un argumento vacío para un literal L o para su complemento $\overline{L}$ pero no para ambos. Cuando $|\Delta| = 1$, el número máximo de argumentos es $|Lit|$, ya que podría existir un argumento vacío para un literal L y un argumento unitario para su complemento. Finalmente, consideremos $|\Delta| = 2$. En este caso, un literal L podría tener un argumento vacío o bien un argumento con las dos reglas o bien como máximo dos argumentos unitarios (estas opciones son excluyentes debido a la condición de minimalidad). Así el número máximo de argumentos para un literal, cuando Δ tiene dos reglas rebatibles es 2. Por lo tanto, la cota superior para cardinalidad de los argumentos potenciales es $2 * |Lit|$.

Así, el número máximo de chequeos para argumentos potenciales que dependen del tamaño del conjunto de las reglas rebatibles y del tamaño de Lit, es $|Lit| * 2^{|\Delta|}$.

Lema 6.3 *Sea AP el tiempo polinomial necesitado para resolver el problema de decisión "¿es un subconjunto de reglas rebatibles un argumento para un literal L bajo un plrb?". Luego, la cota superior del tiempo de cómputo de todos los argumentos es $|Lit| * 2^{|\Delta|} * AP$.*

El resultado anterior establece una cota superior exponencial para el cálculo del conjunto de todos los argumentos en el formalismo de Dung [Dun95].

Aunque debemos verificar si cada subconjunto de Δ es un argumento para cada literal en el lenguaje de un *plrb*, dada la condición de consistencia de la definición de argumento $A \subseteq \Delta$ no puede ser un argumento para un literal y para su complemento. Luego consideraremos solamente $\frac{|Lit|}{2} * 2^{|\Delta|} = |Lit| * 2^{|\Delta|-1}$ argumentos potenciales con el objeto de jugar un juego o equivalentemente para construir el árbol dialéctico.

Esta cota superior puede ser mejorada considerando la minimalidad sobre los argumentos, i.e., ningún subconjunto de Δ, $A_1 \subseteq \Delta$ será argumento de un literal L si existe A_2 argumento de L y $A_2 \subseteq A_1$.

Finalmente, consideramos el problema de decisión de la existencia de un argumento.

Corolario 6.1 (Existencia de un Argumento) *El problema de decisión "¿existe un argumento para un literal L bajo un plrb?" es* ***NP****.*

Demostración: Podemos atinar a elegir un subconjunto arbitrario de Δ y verificar si este subconjunto es un argumento para un literal L bajo *plrb* en tiempo polinomial. Esto prueba la membresía a **NP**. ■

Estos resultados contrastan con aquellos de [PWA03], donde determinar si existe un argumento para una fórmula h es Σ_2^P-completo. Aún cuando existen varias similitudes entre las definiciones de argumento, ellas difieren en la lógica subyacente. Mientras que en el enfoque P.L.R.B. un argumento es un subconjunto de reglas rebatibles y el mecanismo de inferencia para obtenerlo está basado en la programación en lógica, un argumento en el formalismo descripto en [PWA03] es un subconjunto de formulas de un lenguaje proposicional y $\vdash$ corresponde a la inferencia clásica.

Problema de decisión de la inexistencia de contraargumentos

En la definición de secuencia legal, equivalente a la línea de argumentación aceptable, es de vital importancia determinar si existen contraargumentos. La no existencia de contraargumentos se refleja en la definición de juegos con la función *legal* que devuelve el conjunto vacío. Así, planteamos el siguiente problema de decisión sobre la *inexistencia de un contraargumento*: "¿no existe ningún contraargumento para un argumento $\langle A, h\rangle$ bajo un *plrb*?".

Con el objeto de determinar la complejidad de este problema, consideramos el complemento del problema de decisión anterior. *Existencia de un contraargumento*: "¿existe un contraargumento para un argumento $\langle A, h\rangle$ bajo un *plrb*?". Un algoritmo simple que computa este problema de decisión se muestra en la Figura 6.2.

$Cn(A \cup \Pi)$ es un conjunto finito ya que hemos asumido que $\mathcal{P}$ era finito. Asimismo, $|CN| \leq |Lit|/2$, siendo CN la variable en el algoritmo de la Figura 6.2. Hemos probado que este cómputo es P-completo.

Nótese que verficamos solamente contraargumentos para $Cn(A \cup \Pi) - Cn(\Pi)$, ya que no existen contraargumentos para los argumentos vacíos. Así, no existen contraargumentos para los literales en $Cn(\Pi)$ y por lo tanto, $|CN| \leq (|Lit| - 2 * |Cn(\Pi)|)/2$.

El problema de decisión de la existencia de un argumento es NP. Así, el algoritmo está en NP. Como este algoritmo resuelve el problema complementario, el problema de decisión de la inexistencia de contraargumentos está en co-NP.

La lógica subyacente de P.L.R.B. es programación en lógica extendida con negación fuerte, lo que lleva a que a inferencia sea P-completo. La lógica subyacente de Theorist, circunscrición, AEL, Lógica Default and el sistema argumentativo en [RA06] es lógica clásica proposicional, de ahí que la inferencia sea co-NP-completo.

Algoritmo: Contraargumentos
Input: $\mathcal{A} = \langle A, h\rangle$ un argumento para un literal h bajo un *plrb* $\mathcal{P}$ y Π el conjunto de las reglas estrictas $\mathcal{P}$.
Output: *true* si $\mathcal{A}$ tiene un contraargumento bajo el *plrb* $\mathcal{P}$, *false* de otro modo.

CN = $Cn(A \cup \Pi) - Cn(\Pi)$
exists=false
while $CN \neq \varnothing$ and not exists do
    select $L \in CN$
    if existe un argumento para $\neg L$
        then exists=true
        else CN = CN -{L}
endwhile
return exists

Figura 6.2: Un algoritmo para determinar si existe un contraargumento para un argumento.

Presentamos una tabla que compara P.L.R.B. con el framework de argumentación sobre conocimientos propuesto en [RA06] considerando la complejidad computacional. Este análisis muestra que el sistema propuesto por los autores está basado en un sistema argumentativo que es más complejo que P.L.R.B..

	DeLP	Sistema Argumentativo de [RA06]
Existencia de Argumentos	NP	Σ_2^P-completo [PWA03] NP^{NP}
Inexistencia de Contraargumentos	co-Np	Π_2^P-complete[PWA03] co-NP^{NP}

6.4. Complejidad de los Datos de P.L.R.B.

Con el objeto de determinar la cota superior de la complejidad de los datos de los problemas de decisión GAMESAT y NOWINGAME, analizaremos primero la estructura del árbol dialéctico teniendo en consideración la cardinalidad del conjunto de hechos y de las reglas estrictas y rebatibles.

El árbol dialéctico es explorado de modo depth first, al igual que lo hace el algoritmo minimax. Si la profundidad máxima del árbol es m y existen b movimientos legales para cada nodo en el árbol, entonces la complejidad temporal será $O(b^m)$[RN09]. Si implementamos el algoritmo de poda alpha-beta y consideramos que los sucesores son examinados de un modo random, luego la complejidad temporal será $O(b^{\frac{3m}{4}})$[RN09]. La profundidad máxima del árbol

dialéctico para un argumento bajo un *plrb* con $|\Delta|$ reglas rebatibles es $2^{|\Delta|}$ (conjunto de partes de Δ),i.e., podemos considerar cada argumento potencial en una rama del árbol. Cada argumento puede aparecer más de una vez en el árbol pero como máximo una vez en una rama, debido a que la definición de línea argumentativa aceptable impide que las líneas de soporte y de interferencia sean inconsistentes y que contengan argumentos repetidos.

En cuanto al factor de ramificación, existen $\frac{|Lit|}{2}$ literales que pueden estar en conflicto con el último argumento. Estos literales pueden tener como máximo $2^{|\Delta|}$ argumentos potenciales. Luego el factor de ramificación es en el peor caso $\frac{|Lit|}{2} * 2^{|\Delta|}$.

Así, la complejidad temporal de explorar el árbol dialéctico del modo en que minimax lo hace tiene una cota superior de $O((|Lit| * 2^{|\Delta|-1}|)^{2^{|\Delta|}})$.

Cada vez que debemos insertar un nodo vecino B de un nodo A en la estructura del árbol o equivalentemente, cuando un jugador hace un movimiento, debemos chequear si es un movimiento legal en el juego, i.e., si B ataca y derrota a A, y si B no introduce inconsistencia. Con el objeto de determinar si B es un derrotador de A, debemos tener en consideración el criterio de preferencia entre argumentos. Cualquier criterio de preferencia definido entre argumentos puede ser utilizado en P.L.R.B.. Por esta razón, la clase de complejidad del siguiente problema de decisión "¿puede un argumento ser considerado en la estructura del $P_{G(\langle \mathcal{A}, h\rangle, \mathcal{P})}$ de un juego bajo un *plrb*?" o "¿puede un argumento ser considerado en la estructura de un árbol dialéctico bajo un *plrb*?" será dejado parametrizado en la clase C.

Teorema 6.3 *Sea C la clase de complejidad para el problema de decisión :"¿puede un argumento ser considerado en la estructura de un árbol dialéctico bajo un plrb?". La cota superior para la complejidad de los datos de* GAMESAT *es* $\boldsymbol{NP}^{C}$.

Demostración: Para un $\Pi_R \cup \Delta$ fijo, el tamaño del árbol dialéctico para un argumento $\langle \mathcal{A}, L\rangle$ es polinómico en el tamaño de los literales Π_F. Por otra parte, computar cada argumento está en $\mathbf{P}$, y considerar cada argumento en la estructura del árbol está en C.

Con el objeto de decidir si un literal L pertenece al conjunto V del G-modelo, intentamos adivinar un argumento para L tal que el juego jugado a partir de este argumento sea ganado por el Proponente. El número de argumentos es polinomial cuando $\Pi_R \cup \Delta$ está fijo, y determinar si el árbol dialéctico tiene su raíz etiquetada con U es polinómico en los literales de Π_F. Esto prueba la membresía a $\mathbf{NP}^{C}$. ∎

Ya que NOWINGAME es la conjunción de complementos de GAMESAT, un corolario inmediato del resultado anterior es el siguiente.

Corolario 6.2 *Sea C la clase de complejidad para el problema de decisión "¿puede un argumento ser considerado en la estructura de un árbol dialéctico bajo un plrb?". La cota superior para la complejidad de datos de* NOWINGAME *es co-*$\boldsymbol{NP}^{C}$.

Aunque no hemos analizado en profundidad la complejidad de calcular los criterios de preferencia, ilustramos este concepto con dos casos diferentes.

En [CM04a, CM04b], los autores utilizan la especificidad [SL92] como criterio de preferencia entre argumentos conflictivos, para valorar el uso del lenguaje natural basado en el corpus de la web y evaluar y categorizar los resultados de búsqueda, respectivamente. Este criterio, basado en sintaxis, prefiere aquellos argumentos que están más informados o que son más directos. Calcular la especificidad depende del conjunto 2^{Lit}.

Otra implementación de P.L.R.B. utiliza un relación estática de preferencia [CSAG04]. En este caso, el criterio de preferencia es computado comparando valores de los argumentos. Tales valores son obtenidos a través de diferentes fórmulas matemáticas aplicadas a la certeza de una fórmula en el lenguaje. Computar tal criterio de preferencia involucra solo una comparación entre los valores de certeza. Sin embargo, un costo extra debe ser considerado en la construcción de los argumentos, ya que el valor de certeza es calculado teniendo en cuenta toda la información incierta usada para derivar una meta.

6.5. Conclusiones

Hemos analizado la complejidad de la P.L.R.B. a través de la semántica $\mathcal{GS}$ y de la teoría de prueba, estableciendo algunos problemas de decisión relevantes. En particular, hemos analizado en profundidad GAMESAT y NOWINGAME. Con el objeto de lograr nuestro objetivo, distinguimos en un *plrb* la base de datos de la consulta y hemos definido la complejidad de datos, programas y combinada en el contexto de la P.L.R.B.. De lo que hemos investigado, vemos que los sistemas argumentativos no han sido estudiados hasta ahora como un lenguaje de consulta y, por lo tanto, no existe ningún análisis de complejidad de datos previo para el razonamiento rebatible. La Tabla 6.1 resume los problemas estudiados y los principales resultados obtenidos en cuanto a complejidad.

Ya que la P.L.R.B. no asume como entrada al conjunto de argumentos, los primeros resultados que han sido establecidos están relacionados con los argumentos, los movimientos del juego. Nos hemos enfocado en la existencia de argumentos y la inexistencia de contraargumentos con el objeto de jugar el juego y en verificar si el un conjunto es un argumento. Establecimos una cota superior exponencial para el conjunto de todos los argumentos. Debido a la lógica subyacente de la P.L.R.B. nuestros resultados en cuanto a la complejidad son un poco mejor que

Decision Problem	Complexity
¿Está $L \in Cn_R(Rules)$?	**P**-completo
¿Es $\langle \mathcal{A}, L \rangle$ un argumento?	**P**-completo
Existencia de un Argumento	**NP**
Existencia de un Contraargumento	**NP**
Inexistencia de un Contraargumento	co-**NP**
GAMESAT	Data Complexity $\mathbf{NP}^C$
NOWINGAME	Data Complexity $co - \mathbf{NP}^C$

Cuadro 6.1: Problemas estudiados y principales resultados obtenidos en cuanto a complejidad.

aquellos sistemas basados en lógica clásica.

Los resultados sobre la complejidad de datos de GAMESAT y NOWINGAME nos dan un indicio para determinar el poder expresivo de la P.L.R.B.. Ya que nuestros resultados están parametrizados, podemos establecer un límite inferior en **NP**, también notado Σ_1^1, que coincide con la clase de las propiedades de las estructuras finitas expresables en lógica de segundo orden existencial [Fag74].

Capítulo 7

Conclusiones

La Programación en Lógica Rebatible (P.L.R.) es una extensión de la Programación en Lógica que permite la representación de conocimiento tentativo, incierto y potencialmente inconsistente, cuyo sistema de inferencia no monotónico está basado en los sistemas argumentativos.

En los últimos años, la argumentación rebatible ha realizado un significativo aporte a la Ciencias de la Computación, hecho que se refleja en el creciente número de aplicaciones que lo incluyen como formalización del razonamiento del sentido común. Esto motivó el estudio de la teoría de prueba de la P.L.R. a fin de determinar su semántica sin recurrir al control.

P.L.R. determina su significado operacional a partir de la construcción de un árbol dialéctico que comienza con un argumento a favor del literal consultado y cuyos hijos son todos los posibles contraargumentos para el nodo padre que se está analizando.

En esta disertación hemos desarrollado una semántica declarativa $\mathcal{GS}$ para la teoría de prueba de la P.L.R.B.. $\mathcal{GS}$ vincula la teoría de juegos y la teoría de modelos. En [Abr97], se propone una semántica de interacción con el fin de modelar la interacción entre el Sistema y el Ambiente. Esta propuesta está basada en la noción de juego, concepto que se vincula naturalmente con la construcción de un árbol dialéctico en el que dos jugadores, Proponente y Oponente, hacen sus mejores jugadas a fin de derrotar el argumento de su oponente.

Como paso intermedio a $\mathcal{GS}$, se propuso una definición declarativa equivalente a la procedural de estructura de argumento y se circunscribió el conjunto de los argumentos que pueden ser utilizados en un juego a favor y en contra del argumento inicial para un literal. Asimismo se probaron ciertas propiedades de relevancia. A partir de estos conceptos se introdujo la idea de estrategia y los criterios vencedores para un juego. La definición de juego es independiente del criterio de preferencia utilizado entre argumentos, i.e., se trabajó sobre una relación genérica.

$\mathcal{GS}$ es definida a través del único G-modelo para un *plrb* y captura tres clases de respuestas posibles para un literal de la signatura del *plrb* que se está analizando:

- SI: cuando existe un juego que comienza el Proponente con un argumento a favor del literal y dicho juego es ganado por el Proponente;
- NO: cuando existe un juego que comienza el Proponente con un argumento a favor de la negación del literal y dicho juego es ganado por el Proponente; e
- INDECISO: en otro caso.

Dado que solo se tiene en cuenta los literales de la signatura $\mathcal{GS}$ no considera la respuesta DESCONOCIDO.

Finalmente, se probó la sensatez y la completitud de la semántica declarativa $\mathcal{GS}$ con respecto a la teoría de prueba basada en el análisis del árbol dialéctico que se construye a partir de una estructura de argumento.

La semántica $\mathcal{GS}$ fue comparada con diversas semánticas declarativas ya desarrolladas y con sistemas formales cuya teoría de prueba incluye la noción de juego. En todos los casos se explicaron las diferencias existentes, mostrando que ninguna de ellas es adecuada para caracterizar a la teoría de prueba de la P.L.R..

Motivados por la necesidad de lograr un razonamiento práctico, es decir con una respuesta en tiempo razonable, se avanzó en el estudio de la complejidad computacional. En esta tesis se introdujeron problemas de decisión de interés con respecto a los juegos. Particularmente, se definieron GAMESAT y NOWINGAME, relacionados a la verificación de la existencia e inexistencia de juegos.

Asimismo se definieron los problemas de decisión asociados a la existencia de argumentos e inexistencia de contraargumentos para un literal y se probó que el primero está en **NP** y el segundo en co-**NP**. Se compararon estos resultados contra uno de los formalismos argumentativos, con similitudes en la construcción de los argumentos, mostrando que la P.L.R.B. es menos compleja.

Considerando el nexo existente entre la Programación en Lógica y las bases de datos, se definieron las complejidades de datos, de expresión y combinada en el contexto de la P.L.R.B.. Estas definiciones no han sido introducidas hasta el momento en el contexto de los sistemas argumentativos.

Se calcularon cotas superiores de la complejidad de datos para los problemas de decisión GAMESAT y NOWINGAME, las que resultaron en $\mathbf{NP}^C$ y $co-\mathbf{NP}^C$, respectivamente, siendo C la clase de complejidad para el problema de decisión "¿puede un argumento ser considerado en la estructura de un árbol dialéctico bajo un *plrb*?".

A partir de la complejidad de datos de GAMESAT y NOWINGAME podemos conjeturar el

poder expresivo de la P.L.R.B.. Dado que estos resultados están parametrizados con C, podemos establecer un límite inferior en $\mathbf{NP}(\Sigma_1^1$ en la jerarquía aritmética), que coincide con la clase de las propiedades de las estructuras finitas expresables en lógica de segundo orden existencial [Fag74].

7.1. Trabajo Futuro

A partir de la investigación presentada en esta tesis se abren dos nuevas líneas de investigación diferenciadas.

Por un lado, el desafío de enriquecer a la semántica $\mathcal{GS}$. En este sentido deseamos modelar a la Programación en Lógica Normal, es decir caracterizar la negación por falla dentro de los sistemas argumentativos.

Por el otro, avanzar en cuanto a la complejidad computacional de la P.L.R.B.. Cuando analizamos la complejidad de datos hemos fijado la consulta y hemos parametrizado el criterio de preferencia entre argumentos. Así un tópico interesante para futuras investigaciones es estudiar si estos resultados pueden ser aplicados a otros sistemas de argumentación basados en reglas cuya teoría de prueba sea similar.

En el mismo contexto, y como trabajo futuro se plantea analizar la complejidad combinada de los problemas de decisión GAMESAT y NOWINGAME. Asimismo, se plantea el estudio del poder expresivo de la P.L.R.B., a fin de poder compararlo con otros formalismos no monotónicos.

Bibliografía

[ABCMB04] Atkinson, K., Bench-Capon, T., and Mc Burney, P. A dialogue game protocol for multi-agent argument over proposals for action. Tech. Rep. ULCS-04-007, Department of Computer Science, University of Liverpool, Liverpool, U.K., 2004.

[Abr97] Abramsky, S. Semantics of Interaction. In *Semantics and Logic Computation*, A.Pitts and P. Dibyer, Eds. Cambridge, 1997.

[AC97] Amgoud, L., and Cayrol, C. Integrating preference orderings into argument-based reasoning. In *ECSQARU-FAPR* (1997), D. M. Gabbay, R. Kruse, A. Nonnengart, and H. J. Ohlbach, Eds., vol. 1244 of *Lecture Notes in Computer Science*, Springer, pp. 159–170.

[AC02] Amgoud, L., and Cayrol, C. A reasoning model based on the production of acceptable arguments. *Annals of Math and Artificial Intelligence 34* (2002), 197–215.

[AJ94] Abramsky, S., and Jung, A. Domain Theory. In *Handbook of Logic in Computer Science*, S. Abramsky, G. D.M., and T. Maibaum, Eds., vol. 3. Oxford University Press, 1994.

[AJM94] Abramsky, S., Jagadeesan, R., and Malacaria, P. Full abstraction for PCF. Tech. rep., Draft disponible a través de ftp en `theory.doc.ic.ac.uk`, 1994.

[AM97] Abramsky, S., and McCusker, G. Game Semantics. In *Logic and Computation: Proceedings of the 1997 Marktoberdorf Summer School* (1997), H. Schwichtenberg and U. Berger, Eds., Springer-Verlag.

[AP96] Alferes, J. J., and Pereira, L. M. *Reasoning with Logic Programming.* Springer-Verlag, Berlin, 1996.

[BAV04] Bassiliades, N., Antoniou, G., and Vlahavas, I. A defeasible logic reasoner for the semantic web. In *Proc. of the Workshop on Rules and Rule Markup Languages for the Semantic Web* (2004), pp. 49–64.

[BC02] Bench-Capon, T. J. M. Value-based argumentation frameworks. In *NMR 2002* (2002), pp. 443–454.

[BC03] Bench-Capon, T. J. M. Persuasion in Practical Argument Using Value Based Argumentation Frameworks. *Journal of Logic and Computation 13*, 3 (2003), 429–448.

[BD74] Balbes, R., and Dwinger, P. *Distributive Lattices*. University of Missouri Press, Columbia, Missouri, 1974.

[BDKT97] Bondarenko, A., Dung, P., Kowalski, R., and Toni, F. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artificial Intelligence 93*, 1-2 (1997), 63–101.

[Bil93] Billington, D. Defeasible Logic is Stable. *Journal of Logic Computation 3*, 4 (1993), 379–400.

[Bir95] Birkhoff, G. *Lattice Theory*, third ed., vol. 25 of *Colloquium Publications*. American Mathematical Society, Providence, Rhode Island, 1995.

[BS81] Burris, S., and Sankappanavar, H. P. *A course in universal algebra*. Springer-Verlag, New York- USA, 1981.

[CDK05] Coulouris, G., Dollimore, J., and Kindberg, T. *Distributed Systems. Concepts and Design*, fourth ed. Pearson Education Limited, England, 2005.

[CM04a] Chesñevar, C., and Maguitman, A. An Argumentative Approach to Assessing Natural Language Usage based on the Web Corpus. In *Proc. of the European Conference on Artificial Intelligence (ECAI) 2004* (Valencia, Spain, August 2004), pp. 581–585.

[CM04b] Chesñevar, C., and Maguitman, A. ARGUENET: An Argument-Based Recommender System for Solving Web Search Queries. In *Proc. of the 2nd IEEE Intl. IS-2004 Conference* (Varna, Bulgaria, June 2004), pp. 282–287.

[Cn01] Chesñevar, C. I. *Formalización de los Procesos de Argumentación Rebatible como Sistemas Deductivos Etiquetados*. PhD thesis, Universidad Nacional del Sur, 2001.

[CS93] Cadoli, M., and Schaerf, M. A survey of complexity results for nonmonotonic logics. *Journal of Logic Programming 17* (1993), 127–160.

[CS02] Cecchi, L. A., and Simari, G. R. Un enfoque declarativo basado en juegos del razonamiento rebatible. In *Jornadas Chilenas en Computación 2002. III*

Workshop on Advances and Trends in Artificial Intelligence for Problem Solving (ATAI) (Copiapó - Chile, 2002).

[CSAG04] Chesñevar, C., Simari, G., Alsinet, T., and Godo, L. A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge. In *Proc. of the UAI-2004* (2004), pp. 76–84.

[DBC02] Dunne, P. E., and Bench-Capon, T. Coherence in finite argument systems. *Artificial Intelligence 141* (2002), 187–203.

[DEGV01] Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. Complexity and expressive power of logic programming. *ACM Computing Surveys (CSUR) 33*, 3 (September 2001), 374 – 425.

[DH00] Danos, V., and Harmer, R. Probabilistic game semantics. Tech. rep., Université Paris VII, 2000.

[DNT02] Dimopoulos, Y., Nebel, B., and Toni, F. On the Computational Complexity of Assumption-based Argumentation for Default Reasoning. *Artificial Intelligence 141*, 1 (2002), 57–78.

[DP02] Davey, B. A., and Priestley, H. A. *Introduction to Lattices and Order*, second ed. Cambridge University Press, 2002.

[DS96] Dung, P. M., and Son, T. C. An Argumentation-theoretic Approach to Default Reasoning with Specificity. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning (KRR'96)* (Cambridge - Massachusetts, 1996), Morgan Kaufmann Publischers,Inc., pp. 407–418.

[DS01] Dung, P. M., and Son, T. C. An Argument-Based Approach to Reasoning with Specificity. *Artificial Intelligence 133* (2001), 33–85.

[DSW94] Davis, M. D., Sigal, R., and Weyuker, E. J. *Computability, Complexity and Languages. Fundamentals of Theoretical Computer Science*, second ed. Academic Press. Elsevier, 1994.

[Dun93] Dung, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning and logic programming . In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)* (Chambéry, Francia, 1993), pp. 852–857.

[Dun95] Dung, P. M. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning annd logic programming and n-person games. *Artificial Intelligence 77* (1995), 321–357.

[Fag74] Fagin, R. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation. SIAM-AMS Proceedings* (1974), R. Karp, Ed., vol. 7, pp. 43–73.

[FD04] Ferreirós Domínguez, J. Un episodio de la crisis de fundamentos: 1904. *La Gaceta de la RSME 7*, 2 (2004), 449–467.

[FdTMdJ86] Fernández de Tassara, A., and Meske de Jenkins, N. Introducción al Álgebra Universal. Tech. rep., Cuadernos de Economía y Administración. Universidad Nacional del Comahue, Neuquén, Abril 1986.

[Fil01] Fillottrani, P. R. *La negación en la Programación en Lógica*. PhD thesis, Universidad Nacional del Sur, 2001.

[Gar00] García, A. J. *Programación en Lógica Rebatible: Lenguaje, Semántica Operacional y Paralelismo*. PhD thesis, Universidad Nacional del Sur, 2000.

[GK97] Gordon, T., and Karacapilidis, N. The Zeno Argumentation Framework. In *The Sixth International Conference on Artificial Intelligence and Law* (New York, 1997), ACM, Ed., pp. 10–18.

[GL90] Gelfond, M., and Lifschitz, V. Logic programs with classical negation. In *Logic Programming: Proceedings of the Seventh International Conference* (1990), D. Warren and P. Szeredi, Eds., pp. 579–597.

[GS99] García, A., and Simari, G. R. Strong and Default Negation in Defeasible Logic Programming. In *4th Dutch/German Workshop on Nonmonotonic Reasoning Techniques and their applications* (Amsterdam, 25 - 27, Marzo 1999).

[GS04] García, A., and Simari, G. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming 4*, 1 (2004), 95–138.

[GSAS04] Goldin, D. Q., Smolka, S. A., Attie, P. C., and Sonderegger, E. L. Turing machines, transition systems, and interaction. *Inf. Comput. 194* (November 2004), 101–128.

[GSC98] García, A., Simari, G. R., and Chesñevar, C. An Argumentative Framework for Reasoning with Inconsistent and Incomplete Information. In *ECAI 98, Proceedings de Workshop on Practical Reasoning and Rationality, 13th. European Conference on Artificial Intelligence* (Brighton, England, 1998).

[Ham70] Hamblin, C. L. *Fallacies*. Methuen, London, 1970.

[Har99] Harmer, R. S. *Games and Full Abstraction for Nondeterministic Language*. PhD thesis, University of London, Imperial College of Science, Technology and Medicine, Department of Computing, 1999.

[HO94] Hyland, M., and Ong, L. On full abstraction for PCF:I,II,III. A publicarse en Information and Computation. Tech. rep., Draft disponible a través de ftp en `theory.doc.ic.ac.uk`, 1994.

[Hod99] Hodges, A. *Turing* . Routledge, New York, 1999.

[Hol03] Holzmann, G. J. *The SPIN Model Checker: Primer and Reference Manual* . Addison-Wesley Professional, 2003.

[Hro04] Hromkovic, J. *Theoretical Computer Science* . Springer, 2004.

[HS98] Huhns, M., and Singh, M., Eds. *Readings in Agents*. Morgan Kaufmann Publishers, INC., San Francisco, California, 1998.

[JV99] Jakobovits, H., and Vermeir, D. Dialectic semantics for argumentation frameworks. In *Proceedings of the Conference on Artificial Intelligence and Law* (Oslo, Norway, 1999), ACM, Ed., pp. 53–62.

[KT99] Kakas, A., and Toni, F. Computing argumentation in logic programming. *Journal of Logic and Computation*, 4 (1999), 515–562.

[Lal93] Lalement, R. *Computation as Logic*. Masson - Prentice Hall, Paris, 1993.

[Lam78] Lamport, L. Time, Clocks and the Ordering of Events in a Distributed System. *Communications of the ACM 21*, 7 (July 1978), 558–565.

[Lif96] Lifschitz, V. Foundations of logic programming. In *Principles of Knowledge Representation*, G. Brewka, Ed. CSLI Publications, 1996, pp. 1–57.

[Llo87] Lloyd, J. W. *Foundations of Logic Programming*, second ed. Springer-Verlag, New York, 1987.

[Lor58] Lorenzen, P. Logic und agon. In *Atti del XII Congreso Internazionale di Filosofia* (Venezia, Septiembre 1958), pp. 187–194.

[Mah01] Maher, M. J. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming 1*, 6 (2001), 691–711.

[Mak89] Makinson, D. General theory of cumulative inference. In *Non Monotonic Reasoning*, M. Reinfrank, J. de Kleer, M.l.Ginsberg, and E. Sandewall, Eds., vol. 346 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 1989.

[McC80] McCarthy, J. Circumscription - A Form of Non-monotonic Reasoning. *Artificial Intelligence 13*, 1,2 (1980), 27–39.

[MD80] McDermott, D., and Doyle, J. Nonmonotonic logic 1. *Artificial Intelligence 13*, 1,2 (1980), 41–72.

[MdJFdT89] Meske de Jenkins, N., and Fernández de Tassara, A. Sobre Reticulados, Filtros e Ideales. Tech. Rep. 5, Cuadernos de Economía y Administración. Serie Matemática y Estadística. Universidad Nacional del Comahue, Neuquén, Abril 1989.

[MG99] Martínez, D. C., and García, A. Significancia de las falacias en los sistemas argumentativos. In *Proceedings of V CACiC* (1999), pp. 1–18.

[Mil08] Mileti, Joe. The Axiom of Choice and Zorn's Lemma. `http://www.math.uchicago.edu/~mileti/museum/choice.html`, 2008.

[Min82] Minker, J. On Indefinite Data Base and the Closed World Assumption. In *Proc. 6th Conf. on Automated Deduction (CADE'82)* (LNCS 138, New York, 1982), D. Loveland, Ed., Springer, pp. 292–308.

[MS91] Makinson, D., and Schlechta, K. Floating conclusions and zombie paths: two deep difficulties in the directly skeptical approach to defeasible inference nets. *Artificial Intelligence 48* (1991), 199–209.

[MV72] Manna, Z., and Vuillemin, J. Fixpoint Approach to the Theory of Computation. *Communications of the ACM 15*, 7 (July 1972), 528–536.

[Nut88] Nute, D. Defeasible reasoning and decision support systems. *Decision Support Systems 4*, 1 (1988), 97–110.

[Pap94] Papadimitriou, C. *Computational Complexity*. Addison-Wesley Publishing Company, 1994.

[Pol94] Pollock, J. L. Justification and Defeat. *Artificial Intelligence 67* (1994), 377–408.

[Pra98] Prakken, H. Dialectical proof theory for defeasible argumentation with defeasible priorities. Tech. rep., Computer/Law Institute, Free University, Amsterdam, 1998.

[Pra00] Prakken, H. On Dialogue Systems with Speech Acts, Arguments and Counterarguments. In *The 7th European Workshop on Logic for Artificial Intelligence - JELIA 2000* (Berlin, 2000), vol. 1919 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 224–238.

[Pra01] Prakken, H. Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese. Special issue on New Perspectives in Dialogical Logics 127* (2001), 187–219.

[PS97] Prakken, H., and Sartor, G. Argument-based extended logic programming with defeasible priorties. *Journal of Applied Non-Classical Logics 7* (1997), 25–75.

[PV00] Prakken, H., and Vreeswijk, G. Logical systems for defeasible argumentation. In *Handbook of Philosophical Logic*, D. Gabbay, Ed., 2nd ed. Kluwer Academic Pub., 2000.

[PWA03] Parsons, S., Wooldridge, M., and Amgoud, L. Properties and complexity of some formal inter-agent dialogue. *Journal of Logic and Computation 13*, 3 (2003), 347–376.

[PY97] Papadimitriou, C. H., and Yannakakis, M. On the complexity of database queries (extended abstract). In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems* (New York, NY, USA, 1997), ACM Press, pp. 12–19.

[RA06] Rahwan, I., and Amgoud, L. An argumentation based approach for practical reasoning. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2006), ACM, pp. 347–354.

[Rei80] Reiter, R. A Logic for Default Reasoning. *Artificial Intelligence 13*, 1,2 (1980), 81–132.

[RN09] Russell, S., and Norvig, P. *Artificial Intelligence: A modern approach*, third ed. Prentice Hall, New Jersey, 2009.

[RS09] Rahwan, I., and Simari, G. R. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 2009.

[SCnG94] Simari, G. R., Chesñevar, C. I., and García, A. J. The role of dialectics in defeasible argumentation. In *XIV International Conference of the Chilenean Computer Science* (1994), p. November.

[Sim89] Simari, G. R. *A Mathematical Treatment of Defeasible reasoning and its Implementation*. PhD thesis, Washington University, Department of Computer Science, December 1989.

[SL92] Simari, G. R., and Loui, R. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence 53* (1992), 125–157.

[Tar55] Tarski, A. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics 5* (1955), 285–309.

[Tor98] Torretti, R. *El paraíso de Cantor : La tradición conjuntista en la filosofía matemática*. Biblioteca Nacional de Chile, Chile, 1998.

[Tur36] Turing, A. M. On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings of the London Mathematical Society ser. 2* (1936), D. Loveland, Ed., no. 42, pp. 230–265; correction ibid. number 43 (1937), 544–546.

[TVS02] Tanenbaum, A., and Van Steen, M. *Distributed Systems. Principles and Paradigms*, fourth ed. Prentice Hall, New Jersey, 2002.

[Var82] Vardi, M. Y. The complexity of relational query languages. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, STOC82* (New York, NY, USA, May 1982), ACM Press, pp. 137–146.

[Ver98] Verheij, B. Argumed - a template-based argument mediation system for laweyers. In *Legal Knowledge Based Systems. JURIX: The Eleventh Conference* (1998), J. Hage, T. J. Bench-Capon, A. Koers, C. de Vey Mestdagh, and C. Grütters, Eds., pp. 113–130.

[VERAK76] Van Emden, M. H., and R. A. Kowalski, R. The Semantics of Predicate Logic as a Programming Language. *Journal of the Association for Computing Machinery 23*, 4 (1976), 733–742.

[VS99] Vaucheret, C. A., and Simari, G. R. Debugging declarativo basado en revisión de creencias.

[VTS08] Viglizzo, I. D., Tohmé, F. A., and Simari, G. R. An alternative foundation for delp: Defeating relations and truth values. In *FoIKS* (2008), S. Hartmann and G. Kern-Isberner, Eds., vol. 4932 of *Lecture Notes in Computer Science*, Springer, pp. 42–57.

[Weg96] Wegner, P. Interactive Software Technology. In *Handbook of Computer Science and Engineering*, A. Tucker, Ed. CRC Press, Cambridge, Mass., 1996, pp. 1–24.

[Weg97] Wegner, P. Why interaction is more powerful than algorithms. *Communications of the ACM 40*, 5 (1997), 80–91.

[Weg98] Wegner, P. Interactive foundations of computing. *Theorethical Computer Science* (Feb 1998).

[WG99a] Wegner, P., and Goldin, D. Interaction, computability and Church's thesis. Tech. rep., Brown University, Feb 1999.

[WG99b] Wegner, P., and Goldin, D. Mathematical Models of Interactive Computing. Tech. rep., Brown University, Jan 1999.

[WG06] Wegner, P., and Goldin, D. Principles of Interactive Computation. In *Interactive Computation: A New Paradigm*, D. Goldin, S. Smolka, and P. Wegner, Eds. Springer-Verlag, Berlin, 2006, pp. 25–37.

[Zer04] Zermelo, E. Beweis, dass jede Menge wohlgeordnet werden kann. *Mathematische Annalen*, 59 (1904).

Apéndice A

Conjuntos Ordenados

"Good order is the foundation of all things."

Edmund Burke(1729-1797)

"The Axiom of Choice is obviously true, the well-ordering principle obviously false and who can tell about Zorn's lemma?"

Jerry Bona

Diferentes áreas de la Ciencia de la Computación utilizan como fundamento matemático a las estructuras ordenadas. Por ejemplo, los modelos minimales definen la semántica de los programas lógicos, el menor punto fijo da la semántica de las definiciones recursivas de los programas y la sincronización de relojes en sistemas distribuidos se puede realizar bajo un quasi-orden. La importancia de estos tópicos motivan el desarrollo de este apéndice en donde se presentan las principales nociones de conjuntos ordenados que son utilizadas en esta tesis. Se asume que el lector tiene ciertos conocimientos básicos sobre estructuras algebraicas. Se recomienda ampliar las definiciones con la siguiente bibliografía [DP02], [Bir95], [BS81], [BD74], [DSW94], [AJ94], [FdTMdJ86] y [MdJFdT89].

A.1. Conjuntos Parcialmente Ordenados

Vivimos en un mundo en el que el orden es una de sus principales características. Los elementos son ordenados de acuerdo a su naturaleza. En esta sección, se formalizarán las ideas

imprecisas sobre orden, ejemplificando cada concepto tanto en la Ciencia de la Matemática como de la Computación.

Definición A.1 (Orden - Orden Parcial)
Sea P un conjunto. Un *orden parcial (u orden)* sobre P es una relación binaria $\leq$ sobre P tal que, para todo $x, y, z \in P$,

(I) $x \leq x$ (Reflexiva)

(II) $x \leq y$ e $y \leq x$ implica que $x = y$ (Antisimétrica)

(III) $x \leq y$ e $y \leq z$ implica que $x \leq z$ (Transitiva)

Notaremos a este conjunto ordenado $\langle P; \leq \rangle$, y diremos que es un orden. ■

Usaremos $x \leq y$ e $y \leq x$ de modo intercambiable y notaremos $x \not\leq y$ cuando "$x \leq y$ sea falso". El símbolo $||$ será utilizado para denotar que dos elementos no son comparables, i.e., son *incomparables*: $x||y$ si y solamente si $x \not\leq y$ e $y \not\leq x$.

Definición A.2 (Quasi-Orden - Orden Estricto)
Sea P un conjunto. Un *quasi-orden* u *orden estricto* sobre P es una relación binaria $<$ sobre P tal que, para todo $x, y, z \in P$,

(I) Ningún x cumple $x < x$ (Antireflexiva)

(II) $x < y$ e $y < z$ implica que $x < z$ (Transitiva)

Notaremos a un conjunto P con el quasi-orden $<$, $\langle P; < \rangle$ y diremos que $\langle P; < \rangle$ es un quasi-orden. ■

Un quasi-order cumple necesariamente la propiedad antisimétrica. Analizaremos, dos casos: si $x = y$, entonces se cumple la propiedad antisimétrica, ya que no se cumple que $x < x$, por antireflexiva. Si por el contrario $x \neq y$, entonces si $x < y$, no puede ser el caso de que $y < x$, ya que de otro modo por transitividad $x < x$, lo que resulta imposible por el punto I.

Definición A.3 (Cadena - Conjunto totalmente o linealmente ordenado)
Sea $\langle P; \leq \rangle$ un conjunto ordenado . Luego P es una *cadena* o un *conjunto linealmente ordenado* o un *conjunto totalmente ordenado* si, para todo $x, y \in P$, o bien $x \leq y$ o $y \leq x$, i.e., si cualesquiera dos elementos de P son comparables.

El concepto opuesto es una *anticadena*. Un conjunto ordenado P es una anticadena si $x \leq y$ en P solamente si $x = y$. ■

Ejemplo A.1 Veamos algunos ejemplos en campos de las Ciencias Matemáticas y de la Computación:

1. Sistemas de Números: Los conjuntos $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$ y $\mathbb{N}$ son conjuntos ordenados por la relación de orden usual $\leq$ y forman una cadena.

2. Conjunto de Partes: Sea X cualquier conjunto. El conjunto de partes $\mathcal{P}(X)$, consistente en todos los subconjuntos de X, es ordenado por la inclusión de conjuntos: sean $A, B \in \mathcal{P}(X)$, definimos $A \leq B$ si y sólo si $A \subseteq B$. $\mathcal{P}(X)$ no es una cadena. Este conjunto ordenado es fundamental al analizar los modelos de los programas lógicos.

3. Lenguajes Formales: Algunas veces es importante generar un orden sobre las cadenas de un lenguaje. Así, dado un alfabeto Σ, siendo Σ un conjunto totalmente ordenado, podremos definir un orden total sobre Σ^* llamado orden lexicográfico. Por ejemplo, sea $\Sigma = \{0, 1\}$ y $0 < 1$, entonces el orden lexicográfico sobre Σ^* es $\epsilon < 0 < 00 < 000 < \ldots < 01 < 010 < 011 < 0110 < \ldots < 01111 < \ldots < 1 < 10 < 100 < \ldots < 101 < \ldots < 111 \ldots$. Este orden es una cadena. Consideremos otro orden sobre $\Sigma = \{0, 1\}$. Sea Σ^{**} el conjunto de todas las secuencias finitas e infinitas. Ordenamos a Σ^{**} del siguiente modo: $u \leq v$ si y sólo si u es una subcadena inicial (el término técnico sería *prefijo*) de v. Entonces, por ejemplo, $0100 \leq 010011$, $010||100$ y $10101 \leq 101010 \ldots$ (la cadena infinita de unos y ceros alternados). Otro caso de orden conocido es el orden lexicográfico de las palabras en el diccionario.

4. Grafo Acíclico Dirigido o Digrafo Acíclico: En diferentes aplicaciones de la Ciencia de la Computación se requiere encontrar un orden de precedencia entre los vértices de un grafo acíclico dirigido, por ejemplo en la evaluación de una expresión aritmética al construir un compilador. Así podemos dar un orden sobre los vértices de un digrafo acíclico, de modo que dados dos vértices v_1 y v_2, v_1 aparece antes en el orden que v_2 si existe un arco (v_1, v_2). Este orden se conoce como *orden topológico*.

5. Revisión de Creencias: los sistemas no monotónicos deben reconsiderar sus creencias con el fin de manejar la inconsistencia. En el caso de los sistemas de revisión de creencias o sistemas de mantenimiento de la verdad basados en suposiciones (ATMS) se requiere considerar el nuevo dato y determinar que suposiciones serán retiradas del sistema manteniendo de este modo la consistencia de sus creencias. Sin embargo, no toda creencia será aceptada. Algunos sistemas implementan un orden parcial entre los informantes con el objeto de determinar si el nuevo dato será incorporado o será descartado. Si consideramos el modelo de la universidad, un informante profesor será preferido a un no docente,

cuando la creencia en cuestión es académica. Sin embargo, la preferencia será al revés cuando la creencia sea sobre la reglamentación de las inscripciones a las carreras.

6. Leslie Lamport[Lam78, TVS02, CDK05] definió la relación "ocurrió antes" que se corresponde con un quasi-orden y que se utiliza para la sincronización de relojes. Bajo esta relación aquellos eventos distintos incomparables se los denomina concurrentes.

■

Teorema A.1 *Sea S un subconjunto de un conjunto ordenado P. S es un conjunto ordenado bajo la misma relación de P, restringida a S. Cualquier subconjunto de una cadena es una cadena.*

La noción de "inmediatamente superior" en una jerarquía puede ser definida sobre un conjunto ordenado como sigue:

Definición A.4 (Cubrimiento - y cubre a x)
Sean P un conjunto ordenado y $x, y \in P$. Diremos que x *es cubierto por* y (o y *cubre a* x) en P, y lo notaremos $x \leftharpoonup y$ o $y \rightharpoonup x$, si $x < y$ y si $x \leq z < y$ entonces $z = x$. Esta última condición indica que no existe ningún elemento $z \in P$ tal que $x < z < y$. ■

Observación A.1 *Si P es finito, $x < y$ si y sólo si existe una secuencia finita de relación de cubrimiento $x = x_0 \leftharpoonup x_1 \leftharpoonup x_2 \leftharpoonup \ldots \leftharpoonup x_n = y$. Así, en el caso finito, la relación de orden determina y es determinada por la relación de cubrimiento.*

Ejemplo A.2 Consideremos algunos ejemplos:

1. En la cadena $\mathbb{N}$, tenemos que $m \leftharpoonup n$ si y sólo si $n = m + 1$.
2. En $\mathbb{R}$, no existe un par x, y, tal que $x \leftharpoonup y$.
3. En el conjunto de partes $\mathcal{P}(X)$ de un conjunto X, tenemos que $A \leftharpoonup B$ si y sólo si $B = A \cup \{b\}$, para algún $b \in X \backslash A$.

■

A.1.1. Mapeos entre conjuntos ordenados

Dados dos conjuntos ordenados siempre es deseable compararlos determinando si son esencialmente iguales (preservan la estructura), o bien es deseable generar nuevos conjuntos a partir

de un conjunto inicial que cumpla ciertas características. Así, debemos definir mapeos entre los conjuntos y analizar sus propiedades.

Definición A.5 (Función entre órdenes)
Sean $\langle P; \leq \rangle$ y $\langle Q; \leq' \rangle$ órdenes. Una función o aplicación $\varphi : P \to Q$ se dice:

1. *Isótona, Monótona o que Preserva el orden*: si $x \leq y$ en P entonces $\varphi(x) \leq' \varphi(y)$ en Q.
2. *Antítona o Antimonótona*: si $x \leq y$ en P entonces $\varphi(y) \leq' \varphi(x)$ en Q.
3. *Order-Embedding* : si $x \leq y$ en P si y solamente si $\varphi(x) \leq' \varphi(y)$ en Q.
4. *Isomorfismo de Orden*: si φ es una función order-embedding biyectiva. Los conjuntos ordenados P y Q se dicen *isomórficos*, en símbolos $P \cong Q$, si y sólo si existe un isomorfismo de orden entre ellos.

■

Ejemplo A.3 Utilizaremos los Diagramas de Hasse[1] para ejemplificar diferentes aplicaciones. En la figura A.1 se presentan diferentes situaciones. La función φ_1 no es monótona, φ_2 y φ_3 son monótonas pero no son order-embedding y finalmente, φ_4 es order-embedding pero no es un isomorfismo de orden.

■

Lema A.1 *Sean $\langle P; \leq \rangle$ y $\langle Q; \leq' \rangle$ dos conjuntos finitos ordenados y $\varphi : P \to Q$ una función biyectiva. Los siguientes enunciados son equivalentes:*

I *φ es un isomorfismo de orden;*

II *$x < y$ en P si y solamente si $\varphi(x) <' \varphi(y)$ en Q;*

III *$x \leftharpoonup y$ en P si y solamente si $\varphi(x) \leftharpoonup \varphi(y)$ en Q.*

Teorema A.2 *Sean $\langle P; \leq \rangle$ y $\langle Q; \leq' \rangle$ dos órdenes parciales u órdenes y $\varphi : P \to Q$ una función monótona. Si C es una cadena en P luego $\{\varphi(c) | c \in C\}$ es una cadena en Q.*

[1]Dado un conjunto ordenado P se representan con círculos a los elementos de P y con líneas no dirigidas a la relación de cubrimiento.

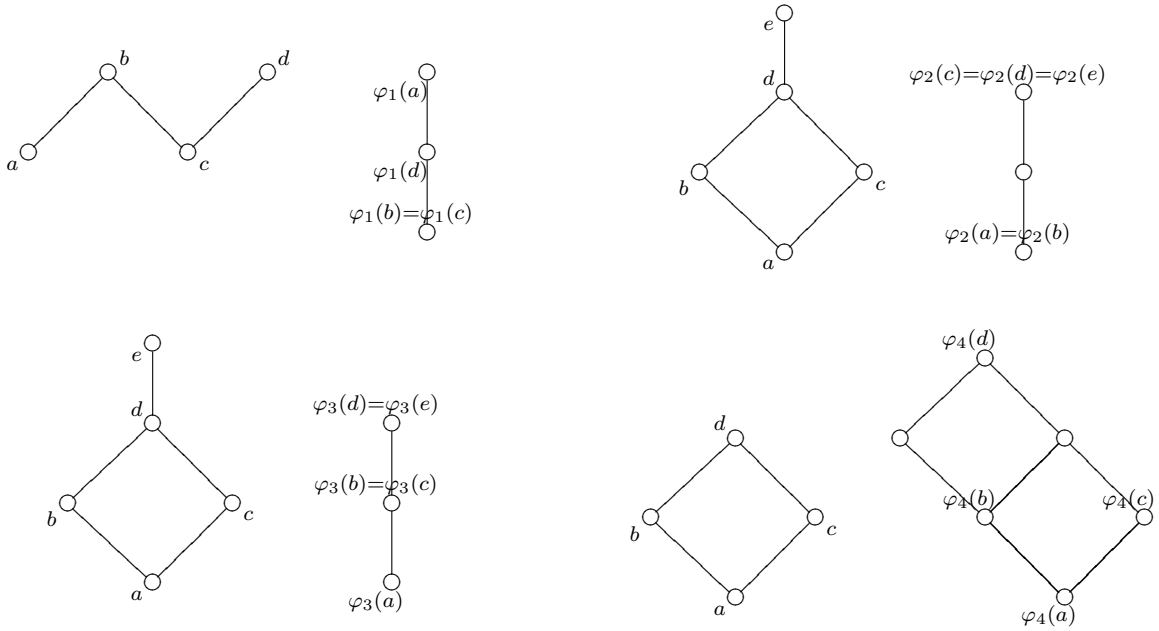


Figura A.1: Mapeos del ejemplo A.3. La función φ_1 no es monótona, φ_2 y φ_3 son monótonas pero no son order-embedding y finalmente, φ_4 es order-embedding pero no es un isomorfismo de orden.

A.1.2. Elementos minimales y maximales

En esta sección presentamos algunos elementos de especial importancia en un conjunto ordenado.

Definición A.6 (Elementos Maximal y Minimal - Primero y Último)
Sean P conjunto ordenado y $Q \subseteq P$. Entonces

- $a \in Q$ es un elemento *maximal* de Q si para cada $x \in Q$, tal que $a \leq x$ implica $a = x$.
  $b \in Q$ es un elemento *minimal* de Q si para cada $x \in Q$, tal que $x \leq b$ implica $b = x$.
- $a \in Q$ es el *último elemento* o *elemento top* de Q si $a \geq x$ para cada $x \in Q$, y en tal caso notaremos $a = maxQ$ o $\top$.
  $b \in Q$ es el *primer elemento* o *elemento bottom* de Q si $b \leq x$ para cada $x \in Q$, y en tal caso notaremos $a = minQ$ o $\bot$.

■

Observación A.2 *Q tiene último elemento sólo si tiene exactamente un elemento maximal y si el último elemento de Q existe, entonces es único(por la propiedad antisimétrica de la relación).*

Ejemplo A.4 Consideremos algunos casos:

Figura A.2: Flat orden.

(a) Sea X un conjunto, $\langle \mathcal{P}(X), \subseteq \rangle$ tiene a $\varnothing$ y X como primer y último elemento respectivamente. En el subconjunto ordenado $\langle \mathcal{P}(X) - \varnothing, \subseteq \rangle$ los elementos minimales son los conjuntos unitarios y no tiene primer elemento. En $\langle \mathcal{P}(X) - X, \subseteq \rangle$ los elementos maximales son los conjuntos de la forma $X - \{e\},\ e \in X$ y no contiene último elemento.

(b) Una cadena finita siempre tiene primer y último elemento, pero una cadena infinita puede no tenerlos. Por ejemplo, la cadena $\mathbb{N}$ tiene primer elemento 1, pero no tiene último elemento; mientras que $\mathbb{Z}$ no posee ni primer ni último elemento.

(c) Un subconjunto de la cadena $\mathbb{N}$ tiene elemento maximal si y solamente si es finito.

■

La falta del primer elemento $\perp$ puede ser remediado agregando uno arbitrario. Dado un conjunto ordenado P (con o sin $\perp$), formamos $P_\perp$ como sigue. Sea un elemento $\perp \notin P$ y definamos $\leq$ sobre $P_\perp = P \cup \{\perp\}$ como

$$x \leq y \text{ si y solamente si } x = \perp \text{ o } x \leq y \text{ en } P.$$

A partir de cualquier conjunto S se puede construir un conjunto ordenado con $\perp$ como sigue: ordene a S haciéndolo una anticadena $\overline{S}$ y luego forme $\overline{S}_\perp$. Los conjuntos ordenados de este modo se denominan *flat*[2]. Generalmente, en las aplicaciones tendremos que $S \subseteq \mathbb{R}$. En este contexto, abusaremos de la notación y escribiremos por simplicidad $S_\perp$ en vez de la expresión correcta $\overline{S}_\perp$. En la Figura A.2 se muestra a $\mathbb{N}_\perp$ y al conjunto flat de los booleanos.

La importancia de los conjuntos ordenados flat, en el contexto de la Ciencia de la Computación, radica en su utilización en la redefinición de las funciones parciales. Si f es una función parcial y n pertenece a su dominio, entonces $f(n)$ existe. Pero si n no pertenece a su dominio, entonces $f(n)$ no existe. Un tratamiento más justo, consiste en introducir el valor $\perp$ para definir $f(n) = \perp$ siempre que n no esté en el dominio. El símbolo $\perp$, el primer elemento de $X_\perp$, es la información minimal sobre un cómputo, expresando un resultado indefinido, un resultado cuyo cómputo no termina o bien un resultado cuyo cómputo *aún no ha terminado.*

A continuación se presentan algunos resultados interesantes sobre los elementos maximales y minimales.

[2]Mantendremos el vocablo en inglés por no encontrar uno en español que se ajuste al concepto.

Teorema A.3 *Cualquier subconjunto finito no vacío X de un conjunto ordenado tiene miembros minimales y maximales.*

Teorema A.4 *En cadenas, la noción de minimal y primer elemento (maximal y último elemento) de un subconjunto son equivalentes. Así cualquier cadena finita tiene primer y último elemento.*

Teorema A.5 *Cada cadena finita de n elementos es isomórfica con el ordinal n (la cadena de naturales $1, \ldots, n$).*

Ejemplo A.5 En el contexto de la Ciencia de la Computación el primer elemento representa mínima información y $\top$ corresponde a un conjunto maximal de información que puede ser inconsistente o contradictoria. Así, bajo la semántica declarativa de la Programación en Lógica Definida (sólo cláusulas de Horn) el modelo minimal corresponde con la menor cantidad de átomos que el programa deriva, mientras que el modelo maximal, la base de Herbrand, posee todos los potenciales átomos que el programa lógico podría derivar. ■

A.2. Reticulados

Existen dos formas estándar de definir reticulados. Una de ellas está basada en la noción de orden. Bajo este enfoque se estudia a los reticulados como una clase especial de los conjuntos ordenados, considerando las propiedades que cumplen los elementos del reticulado. La otra es como estructura algebraica, de la misma forma en que definimos los grupos o los anillos. En esta sección estudiaremos a los reticulados desde los dos enfoques.

Reticulados como conjuntos ordenados

Estableceremos algunas conceptos preliminares sobre las propiedades de ciertos elementos en un conjunto ordenado.

Definición A.7 (Cotas Superior e Inferior)
Sean $\langle P; \leq \rangle$ un orden y $S \subseteq P$. Un elemento $x \in P$ es una *cota superior* de S si $s \leq x$ para todo $s \in S$. Se define dualmente la *cota inferior*. El conjunto de todas las cotas superiores de S lo denotamos S^s y el conjunto de todas las cotas inferiores de S^i:

$$S^s = \{x \in P | (\forall s \in S) s \leq x\} \text{ y } S^i = \{x \in P | (\forall s \in S) x \leq s\}$$

■

P_1 P_2

Figura A.3: Conjuntos ordenados del ejemplo A.6.

Definición A.8 (Supremo e Ínfimo)
Sean $\langle P; \leq \rangle$ un orden y $S \subseteq P$. Diremos que x es la *menor cota superior* o *supremo* de S, y lo escribimos sup S, si:

(I) x es una cota superior de S, i.e., $x \in S^s$, y

(II) $x \leq y$ para toda cota superior y de S, i.e., $x \leq y$ para todo $y \in S^s$.

Dualmente, definimos a la *mayor cota inferior* o *ínfimo* y lo notamos inf S. ■

Nótese que si el supremo y el ínfimo existen, entonces son únicos por la propiedad antisimétrica de los conjuntos ordenados.

Observación A.3 *Consideremos al conjunto ordenado $\langle P; \leq \rangle$ y $S \subseteq P$. Analizaremos dos casos extremos: cuando $S = P$ y cuando S es vacío.*

Caso $S = P$: Si P tiene último elemento, luego $P^s = \{\top\}$, en cuyo caso el sup $P = \top$. Cuando P no tiene último elemento, entonces $P^s = \varnothing$ y así el sup P no existe. En forma dual, inf $P = \bot$, siempre que P tenga primer elemento.

Ahora, sea S el conjunto vacío. Luego, cada elemento $x \in P$ satisface $s \leq x$ para cada $s \in S$. Luego $\varnothing^s = P$ y por lo tanto, sup $\varnothing$ existe si y solamente si P tiene primer elemento y en este caso sup $\varnothing = \bot$. Dualmente, inf $\varnothing = \top$ siempre que P tenga último elemento.

Ejemplo A.6 En un conjunto ordenado la menor cota superior de dos elementos $\{x, y\}$ puede no existir por diferentes razones:

1. x e y no tienen cota superior común, como ocurre en el conjunto P_1 en la figura A.3; o
2. no tienen supremo. En el caso del conjunto P_2 en la figura A.3, tenemos que $\{a, b\}^s = \{c, d\}$, sin embargo no existe una menor cota superior.

■

Nosotros estamos interesados en los conjuntos ordenados donde existe el supremo y el ínfimo entre cada par de elementos del conjunto.

Definición A.9 (Reticulado - Reticulado Completo)
Sea P un conjunto no vacío ordenado bajo $\leq$. Si el $\sup\{x, y\}$ y el $\inf\{x, y\}$ existen en P para todo $x, y \in P$, entonces diremos que $\langle P; \leq \rangle$ es un *reticulado*. Si sup S y el inf S existen para todo $S \subseteq P$, entonces $\langle P; \leq \rangle$ es un *reticulado completo*. ■

Notaremos al primer y último elemento de un reticulado $\langle P, \leq \rangle$, $\bot$ y $\top$ respectivamente.

Ejemplo A.7

- Toda cadena es un reticulado en donde el supremo de dos elementos x, y es el máximo entre los elementos y el ínfimo es el mínimo entre x e y. Así, los conjuntos $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ y $\mathbb{R}$ son reticulados bajo el orden usual donde $\sup\{x, y\} = max\{x, y\}$ y el $\inf\{x, y\} = min\{x, y\}$. Sin embargo, ninguno de estos conjuntos es un reticulado completo; a cada uno le falta el último elemento. En particular, cuando nos restringimos a un intervalo cerrado $[x, y]$ en $\mathbb{R}$, $[x, y]$ es un reticulado completo. Esto difiere con $\mathbb{Q}$, en el que no solamente la falta de los elementos $\top$ y $\bot$ causan problemas; por ejemplo, el conjunto $\{s \in \mathbb{Q} | s^2 < 2\}$ tiene cotas superiores pero no tiene una menor cota superior en $\mathbb{Q}$.

- Sea X un conjunto arbitrario, el conjunto ordenado $\mathcal{P}(X)$ bajo la relación $\subseteq$ es un reticulado completo en el cual

$$sup\{A_i | i \in I\} = \bigcup_{i \in I} A_i$$

$$inf\{A_i | i \in I\} = \bigcap_{i \in I} A_i$$

■

Reticulados como estructura algebraica

Hemos definido a los reticulados como una clase especial de conjuntos ordenados. A continuación los estudiaremos con el enfoque de estructura algebraica y presentaremos la conexión correspondiente entre ambas definiciones.

Definición A.10 (Reticulado)
Sean L un conjunto no vacío, $\vee$ y $\wedge$ dos operaciones binarias sobre L, entonces $\langle L; \vee, \wedge \rangle$ es un reticulado si las operaciones satisfacen para todo $a, b, c \in L$:

L1 $(a \vee b) \vee c = a \vee (b \vee c)$
$(a \wedge b) \wedge c = a \wedge (b \wedge c)$ (Leyes Asociativa)

L2 $a \vee b = b \vee a$
$a \wedge b = b \wedge a$ (Leyes Conmutativa)

L3 $a \vee a = a$
$a \wedge a = a$ (Leyes de Idempotencia)

L4 $a \vee (a \wedge b) = a$
$a \wedge (a \vee b) = a$ (Leyes de Absorción)

■

Las operaciones *join* y *meet* sobre L se definen como:

$$a \vee b = sup\{a, b\} \quad a \wedge b = inf\{a, b\} \quad (a, b \in L).$$

Así escribiremos en forma intercambiable $a \vee b$ y $sup\{a, b\}$ cuando exista y de manera similar lo haremos con la operación meet. Asimismo, escribiremos $\inf S = \bigwedge S$ y $\sup S = \bigvee S$. Cuando sea necesario indicar que las operaciones join y meet están fundamentadas en un conjunto ordenado P, escribiremos $\bigwedge_P S$ y $\bigvee_P S$.

Ejemplo A.8 Analizamos algunos reticulados $\langle L; \vee, \wedge \rangle$, con el enfoque de las estructuras algebraicas:

- Sean L un conjunto no vacío de proposiciones, $\vee$ el conectivo de disyunción y $\wedge$ el conectivo de conjunción. Ambas operaciones cumplen de $L1$ a $L4$.
- Sean L el conjunto de números naturales, $\vee$ el mínimo común múltiplo y $\wedge$ el máximo común divisor. Ambas operaciones cumplen de $L1$ a $L4$.
- Sean L el conjunto de partes de un conjunto arbitrario X, $\vee$ la unión de conjuntos y $\wedge$ la intersección de conjuntos. Ambas operaciones cumplen de $L1$ a $L4$.

■

Lema A.2 *Sea P un conjunto ordenado con primer elemento $\bot$, luego*

$$\bot \wedge x = \bot \quad y \quad \bot \vee x = x, \text{ para todo } x \in P$$

Dualmente, si P tiene último elemento $\top$, luego

$$x \wedge \top = x \quad y \quad x \vee \top = \top, \text{ para todo } x \in P$$

Al referirnos al reticulado como estructura algebraica diremos que L tiene un elemento *unidad* o *identidad*, si existe $1 \in L$ tal que $a = a \wedge 1$ para todo $a \in L$. Dualmente, L tiene un elemento *cero* si existe $0 \in L$ tal que $a = a \vee 0$ para todo $a \in L$. Así, el reticulado $\langle L; \vee, \wedge \rangle$ tiene elemento unidad 1 si y solamente si $\langle L, \leq \rangle$ tiene último elemento $\top$ y en este caso $1 = \top$. Similarmente, ocurre con 0 y $\perp$. Un reticulado $\langle L; \vee, \wedge \rangle$ que posee 0 y 1 se denomina *acotado*. Un reticulado finito está acotado con $1 = \bigvee L$ y $0 = \bigwedge L$.

Conexión entre el enfoque algebraico y el basado en conjuntos ordenados

Las definiciones dadas de reticulados según el enfoque algebraico y el basado en conjuntos ordenados son equivalentes. A continuación presentamos una serie de lemas y teoremas que muestran este resultado. Las demostraciones se han omitido, se recomienda al lector interesado en ellas revisar la bibliografía específica enumerada al principio del apéndice.

Lema A.3 *Sea $\langle L, \wedge, \vee \rangle$ un reticulado y sean $y, z \in L$. Las operaciones join y meet son monótonas:*

$$\text{Si } y \leq z, \text{ entonces } (x \wedge y \leq x \wedge z) \quad y \quad (x \vee y \leq x \vee z), \forall x \in L.$$

Lema A.4 (El lema de conexión) *Sea $\langle L, \leq \rangle$ un reticulado y sean $a, b \in L$. Entonces las siguientes afirmaciones son equivalentes:*

1. $a \leq b$
2. $a \vee b = b$
3. $a \wedge b = a$

Teorema A.6 *Sea $\langle L, \leq \rangle$ un reticulado. Luego $\vee$ y $\wedge$ satisfacen L1-L4.*

De lo expresado anteriormente, concluimos el siguiente teorema:

Teorema A.7 *Sea $\langle L; \vee, \wedge \rangle$ un reticulado.*

1. *Para todo $a, b \in L$, $a \vee b = b$ si y solamente si $a \wedge b = a$.*
2. *Definimos $\leq$ como $a \leq b$ si $a \vee b = b$. Entonces $\leq$ es una relación de orden.*

3. *Siendo $\leq$ definida como en el punto anterior, $\langle L, \leq \rangle$ es un reticulado en el cual para todo $a, b \in L$*

$$a \vee b = sup\{a, b\} \quad y \quad a \wedge b = inf\{a, b\}$$

El siguiente lema contiene información que permite calcular las operaciones join y meet, como así también muestra su comportamiento con respecto a la unión de conjuntos.

Lema A.5 *Sean P un conjunto ordenado bajo relación $\leq$ y $S, T \subseteq P$. Supongamos que $\bigvee S$, $\bigvee T$, $\bigwedge S$ y $\bigwedge T$ existen en P.*

- *Para todo $s \in S$, $s \leq \bigvee S$ y $\bigwedge S \leq s$.*
- *Sea $x \in P$, luego $x \leq \bigwedge S$ si y solamente si $x \leq s$ para todo $s \in S$*
- *Sea $x \in P$, luego $\bigvee S \leq x$ si y solamente si $s \leq x$ para todo $s \in S$*
- *$\bigvee S \leq \bigwedge T$ si y solamente si $s \leq t$ para todo $s \in S$ y para todo $t \in T$.*
- *Si $S \subseteq T$, luego $\bigvee S \leq \bigvee T$ y $\bigwedge T \leq \bigwedge S$.*

Finalmente, daremos una definición de reticulados isomórficos utilizando el enfoque basado en estructuras algebraicas, que es equivalente, para la noción de reticulados, a la definición dada sobre conjuntos ordenados.

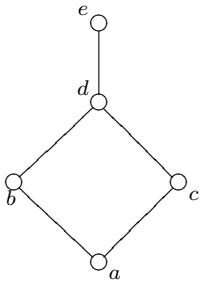
Definición A.11 (Homomorfismo sobre reticulados -Reticulados Isomórficos)
Sean L_1 y L_2 dos reticulados. Un mapeo $\alpha : L_1 \to L_2$ es un *homomorfismo de reticulados* si para cada $a, b \in L_1$, las siguientes dos ecuaciones ocurren:

- Preserva la operación Join: $\alpha(a \vee b) = \alpha(a) \vee \alpha(b)$
- Preserva la operación Meet: $\alpha(a \wedge b) = \alpha(a) \wedge \alpha(b)$.

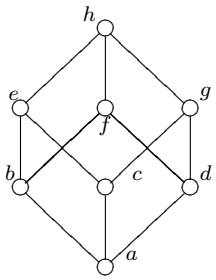
Un homomorfismo de reticulados biyectivo es un *isomorfismo de reticulados*. En tal caso, diremos que L_1 y L_2 son *isomórficos*. ■

El siguiente teorema asegura que las nociones de isomorfismo dadas son equivalentes.

Teorema A.8 *Dos reticulados L_1 y L_2 son isomórficos según la definición A.11 si y solamente si existe una biyección $\alpha : L_1 \to L_2$ tal que α es order-embedding.*



L_1



L_2

Figura A.4: Reticulados.

Hemos caracterizado a los reticulados en términos de las operaciones join y meet. Así podremos utilizar en forma equivalente $\langle L, \leq \rangle$ y $\langle L; \vee, \wedge \rangle$ según queramos enfatizar que L es una clase especial de conjuntos ordenados o que es una estructura algebraica. Utilizar el enfoque algebraico nos permite encontrar paralelismos entre los reticulados y otras estructuras algebraicas, tomando los elementos comunes. Este nivel de abstracción logrado a través de las álgebras nos permite, además, aplicar los resultados a situaciones enteramente nuevas.

A.2.1. Subconjuntos de Reticulados

En muchos casos, estamos interesados en analizar ciertas partes de un reticulado, por sus características particulares. Las siguientes nociones permiten determinar algunos subconjuntos de un reticulado.

Definición A.12 (Subreticulado)
Sean L un reticulado y $L' \neq \varnothing$ un subconjunto de L. Si para cada par de elementos $a, b \in L'$, se cumple que $a \vee b$ y $a \wedge b$ están en L', siendo $\vee$ y $\wedge$ las operaciones del reticulado L, entonces diremos que L' con las mismas operaciones de L restringidas a L' es un *subreticulado* de L . ■

Ejemplo A.9 Consideremos los reticulados de la figura A.4. El conjunto ordenado $P = \{a, b, c, e\}$ subconjunto de L_1 no es un subreticulado de L_1. El subconjunto $P' = \{d, e\}$ es un subreticulado de L_1. Los subconjuntos $P = \{a, c, d, g\}$, $P' = \{a, b, f, h\}$ y $P'' = \{a, d, e, h\}$ son subreticulados de L_2. Mientras que $\{a, b, c, d, e, g\}$ y $\{b, e, d, g\}$ no son subreticulados de L_2. ■

Definición A.13 (Ideal)
Sea L un reticulado. Un *ideal* I de L es un subconjunto no vacío de L tal que

1. $x, y \in I$ entonces $x \vee y \in I$
2. $x \in I,\ y \in L$, siendo $y \leq x$, luego $y \in I$.

I es un *ideal propio* si $I \neq L$. ■

Nótese que un ideal es cerrado con respecto al supremo (primer condición) y además es cerrado "hacia abajo" con respecto a la relación de orden. Un subconjunto I de un conjunto ordenado arbitrario que cumple la condición 2 de la definición A.13, se dice *conjunto inferior* o *conjunto decreciente*. Estos conjuntos contienen todo elemento que esté por debajo de ellos en el orden.

Definición A.14 (Filtro)
Sea L un reticulado. Un *filtro* F de L es un subconjunto no vacío de L tal que

1. si $x, y \in F$ entonces $x \wedge y \in F$
2. si $x \in F$, $y \in L$ y $x \leq y$, entonces $y \in F$.

F es un *filtro propio* si $F \neq L$. ■

Nótese que un filtro es cerrado con respecto al ínfimo (primer condición) y además es cerrado "hacia arriba" con respecto a la relación de orden. Un subconjunto F de un conjunto ordenado arbitrario que cumple la condición 2 de la definición A.14, se dice *conjunto superior* o *conjunto creciente*. Se puede probar que Q es un conjunto decreciente de un conjunto ordenado P si y solamente si $P - Q$ es un conjunto creciente.

Ejemplo A.10 Consideremos nuevamente los reticulados de la figura A.4. Los conjuntos $I = \{a, b, c, d\}$ y $F = \{d, e\}$ son ideal y filtro respectivamente del reticulado L_1. Los conjuntos $I = \{a, b, c, e\}$ y $F = \{b, e, f, h\}$ son ideal y filtro respectivamente del reticulado L_2. ■

Al conjunto de todos los ideales (filtros) de un reticulado L los denotamos $\mathcal{I}(L)$ ($\mathcal{F}(L)$).

Lema A.6 *Sea L un reticulado. $\mathcal{I}(L)$ y $\mathcal{F}(L)$ son cerrados bajo intersección finita y bajo intersección arbitraria siempre que la intersección no sea vacía.*

Teorema A.9 *Sea L un reticulado. $\mathcal{I}(L)$ ordenado por inclusión, forma un reticulado $\dot{L}$.*

Hemos caracterizado subconjuntos especiales de los reticulados. En la siguiente sección analizaremos reticulados de particular importancia por sus propiedades.

A.2.2. Reticulados Distributivos

Una de las clases de reticulados más estudiados es aquella que cumple la propiedad distributiva.

Definición A.15 (Reticulados Distributivos)
Un reticulado L se dice *distributivo* si verifica alguna de las siguientes leyes:

D1 $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$

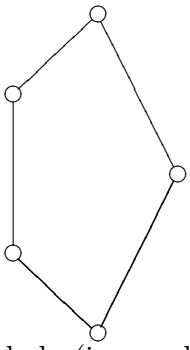
D2 $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

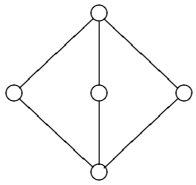
■

Teorema A.10 *Un reticulado $\langle L, \wedge\vee\rangle$ satisface D1 si y solamente si satisface D2.*

Ejemplo A.11 Dos clases importantes de reticulados distributivos son los conjuntos totalmente ordenados (cadenas) y las álgebras booleanas. ■

Birkhoff caracterizó los reticulados no distributivos mostrando que si un reticulado L contiene a alguno de los siguientes dos reticulados no distributivos:





como subreticulado (i.e., embebidos), entonces L no es distributivo.

A.2.3. Reticulados Completos

Recordemos que un reticulado es completo si el supremo y el ínfimo existen para cada subconjunto del reticulado (ver definición A.9). Los siguientes resultados permiten caracterizar a los reticulados completos.

Lema A.7 *Sea $\langle L, \wedge, \vee\rangle$ un reticulado. Luego $\bigvee F$ y $\bigwedge F$ existen para cada subconjunto finito no vacío F de L.*

Corolario A.1 *Cada reticulado finito es completo.*

Para mostrar que un reticulado es completo se requiere sólo realizar la mitad del trabajo que indica su definición. Los siguientes resultados lo prueban.

Lema A.8 *Sea P un conjunto ordenado tal que $\bigwedge S$ existe en P para cada subconjunto no vacío S de P. Luego $\bigvee S$ existe en P para cada subconjunto S de P que tiene una cota superior en P; en efecto, $\bigvee S = \bigwedge S^s$.*

Teorema A.11 *Sea P un conjunto ordenado no vacío. Entonces las siguientes afirmaciones son equivalentes:*

1. *P es un reticulado completo*
2. *Existe $\bigwedge S$ en P para cada subconjunto S de P*
3. *P tiene último elemento, $\top$ y $\bigwedge S$ existe para cada subconjunto no vacío S de P.*

Analicemos un poco más en profundidad la razón del tercer item. Nótese que a diferencia del segundo item, en el tercer item del teorema anterior S no puede ser vacío. Supongamos que S fuese vacío. Entonces $S^i = P$ trivialmente, ya que para todo $x \in P$, $x \leq s$, para todo $s \in S$. Luego $\bigwedge S = \bigwedge \varnothing = \top$. Así las condiciones de que P tenga último elemento y de que exista $\bigwedge \varnothing$ en P son equivalentes.

A.2.4. Sistema de Clausura

A partir del teorema A.11 se deriva un corolario que permitirá introducir una nueva estructura: los sistemas de clausura.

Un corolario interesante que deriva en un nuevo concepto de estructura es el que sigue:

Corolario A.2 *Sea X un conjunto y sea $\mathcal{L}$ una familia de subconjuntos de X, ordenado por inclusión, tal que*

(a) $\bigcap_{i\in I} A_i \in \mathcal{L}$ para cada familia no vacía $\{A_i\}_{i\in I} \subseteq \mathcal{L}$ y

(b) $X \in \mathcal{L}$

Entonces $\mathcal{L}$ es un reticulado completo en el cual

$$\bigwedge\{A_i | i \in I\} = \bigcap_{i\in I} A_i,$$

$$\bigvee\{A_i | i \in I\} = \bigcap\{B \in \mathcal{L} | \bigcup_{i\in I} A_i \subseteq B\}.$$

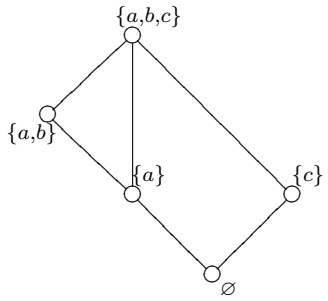


Figura A.5: Reticulado formado sobre el conjunto $\mathcal{L}$ del ejemplo A.12.

Ejemplo A.12 Consideremos $X = \{a, b, c\}$. $\mathcal{L}$ formado por

$$\{\varnothing, \{a\}, \{a, b\}, \{c\}, \{a, b, c\}\}$$

Entonces $\langle \mathcal{L}, \subseteq \rangle$ es el reticulado completo que se muestra en la figura A.5 ■

Definición A.16 ($\bigcap$-estructura. Sistemas de Clausura)
Sea $\mathcal{L}$ una familia no vacía de subconjuntos de X. Si $\mathcal{L}$ satisface la condición (a) del corolario A.2, entonces $\mathcal{L}$ se llama *estructura de intersección* o $\bigcap$*-estructura* sobre X. Si $\mathcal{L}$ también satisface la condición (b) del corolario A.2, entonces diremos que es una *estructura de intersección con tope* o *sistemas de clausura* sobre X. ■

Nótese que un sistema de clausura es cerrado bajo intersección por la condición (a) del corolario A.2 y además contiene al último elemento de la familia por la condición (b) del corolario A.2. Así los sistemas de clausura forman un reticulado completo (corolario A.2).

Generalizando la definición A.16, podemos decir que una propiedad de subconjuntos de X es llamada *propiedad de clausura* cuando (i) X tiene la propiedad y (ii) toda intersección de subconjuntos que tengan esta propiedad, tiene también esta misma propiedad.

Los subconjuntos cerrados bajo una propiedad de clausura dada en la definición A.16 forman una *familia de Moore* de X.

Los sistemas de clausura son estructuras que están íntimamente relacionadas a los sistemas computacionales. Esta relación queda fuera del alcance de este apéndice. Al lector interesado se le recomienda la lectura de [DP02, AJ94, DSW94], particularmente lo referente a sistemas de información y teoría de dominios.

A.2.5. Operador de Clausura

Existe una conexión entre los sistemas de clausura y el operador de clausura.

Definición A.17 (Operador de Clausura)
Sea X un conjunto. Un mapeo $C : \mathcal{P}(X) \to \mathcal{P}(X)$ es un *operador de clausura* sobre X si, para todo $A, B \subseteq X$,

C1 $A \subseteq C(A)$ (Extensivo)

C2 Si $A \subseteq B$, entonces $C(A) \subseteq C(B)$ (Monótona)

C3 $C(C(A)) = C(A)$ (Idempotente)

Un subconjunto A de X es llamado *cerrado* (con respecto a C) si $C(A) = A$. ■

La relación entre una familia de Moore o sistema de clausura y el operador de clausura es biyectiva.

Teorema A.12 *Sea C un operador de clausura sobre un conjunto X. Luego la familia*

$$\mathcal{L} = \{A \subseteq X | C(A) = A\}$$

de los subconjuntos cerrados de X es un sistema de clausura y por lo tanto, forma un reticulado completo, ordenado por inclusión y en el cual

$$\bigwedge\{A_i | i \in I\} = \bigcap_{i \in I} A_i,$$

$$\bigvee\{A_i | i \in I\} = C(\bigcup_{i \in I} A_i).$$

Recíprocamente, dado un sistema de clausura $\mathcal{L}$ sobre X la fórmula

$$C_{\mathcal{L}}(A) = \bigcap\{B \in \mathcal{L} | A \subseteq B\}$$

define el operador de clausura $C_{\mathcal{L}}$ sobre X.

La biyección presentada en el teorema anterior nos permite trabajar con el sistema de clausura o con su correspondiente operador de clausura según nuestra conveniencia.

Ejemplo A.13 Sea $\langle P, \leq \rangle$ un conjunto ordenado y $Q \subseteq P$. Definimos el mapeo $\downarrow$: $\mathcal{P}(P) \to \mathcal{P}(P)$ como:

$$\downarrow Q = \{y \in P | (\exists x \in Q) y \leq x\}$$

Dicho operador es de clausura, ya que:

- $Q \subseteq \downarrow Q$ Trivial, ya que la relación $\leq$ cumple la propiedad reflexiva.

- Si $A \subseteq B$, entonces $\downarrow A \subseteq \downarrow B$. Ya que cada elemento en A está también en B. Luego si se cumple $(\exists x \in A) y \leq x$ también se cumple que $(\exists x \in B) y \leq x$.
- $\downarrow (\downarrow Q) = \downarrow Q$
  $\downarrow (\downarrow Q) \subseteq \downarrow Q$: Sea x un elemento tal que $x \in \downarrow (\downarrow Q)$. Luego por definición $x \in P$ y $(\exists z \in \downarrow Q) x \leq z$. Asimismo, se cumple que $z \in P$ y $(\exists y \in Q) z \leq y$. Ahora, dado que $x \leq z$ y $z \leq y$ entonces $x \leq y$. Por lo tanto se cumple que $x \in P$ y $(\exists y \in Q) x \leq y$, es decir, $x \in \downarrow Q$.
  $\downarrow Q \subseteq \downarrow (\downarrow Q)$:por el primer item.

El sistema de clausura correspondiente es la familia $\mathcal{O}(P)$ de todos los conjuntos decrecientes de P. ■

Los conceptos de operador de clausura son utilizados en la definición de la semántica desarrollada en este trabajo.

A.3. Conjuntos Ordenados Completos

La idea de finitud y el concepto de punto fijo son de suma importancia en las Ciencias de la Computación. Uno de los problemas operacionales esenciales de la computación es el fin del cómputo. El hecho de que un cómputo no termine es interpretado a través de las funciones parciales. La idea de definición recursiva de las funciones y la de ecuación de punto fijo están íntimamente relacionadas. A continuación, daremos una serie de definiciones que derivarán en esta correspondencia.

Definición A.18 **(Conjunto Dirigido)** Sea S un subconjunto no vacío de un conjunto ordenado P. S se dice *dirigido*, si para cada subconjunto finito F de S, existe $z \in S$ tal que $z \in F^s$. ■

Ejemplo A.14 El conjunto $\mathbb{N}$ con la relación de orden $\leq$ es un conjunto dirigido. Cualquier cadena no vacía es un conjunto dirigido y cualquier subconjunto de un conjunto ordenado con último elemento es dirigido. ■

Observación A.4 *S es un conjunto dirigido si y solamente si, para cada par de elementos $x, y \in S$, existe $z \in S$ tal que $z \in \{x, y\}^s$.*

Observación A.5 *Sea X un conjunto. Si $\mathcal{D} = \{A_i\}_{i \in I}$ es un conjunto dirigido de $\mathcal{P}(X)$ e $Y = \{y_1, \ldots, y_n\}$ es un subconjunto finito de $\bigcup_{i \in I} A_i$ luego existe $A_k \in \mathcal{D}$ tal que $Y \subseteq A_k$. En*

efecto, ya que $\mathcal{D}$ es dirigido, existe $k \in I$ tal que $y_j \in A_{i_j} \subseteq A_k$ para todo j, A_k es el supremo de $A_{i_j},\ j : 1, \ldots n$ y consecuentemente $Y \subseteq A_k$.

Ejemplo A.15 Sean $X = \{a, b, c\}$, $\mathcal{D} = \{\{a\}, \{a, c\}, \{a, b\}, \{a, b, c\}\}$ e $Y = \{c, b\}$. Existe $\{a, b, c\}$, tal que $Y \subseteq \{a, b, c\}$. Además, $b \in \{a, b\}$ y $c \in \{a, c\}$ y $\{a, b\} \subseteq \{a, b, c\}$ y $\{a, c\} \subseteq \{a, b, c\}$, siendo $\{a, b, c\}$ el supremo de $\{a, b\}$ y $\{a, c\}$. ■

Observación A.6 *Los conjuntos inferiores dirigidos son ideales.*

Definición A.19 (Conjunto Ordenado Completo Dirigido - Conjunto Ordenado Completo)
Un conjunto (parcialmente) ordenado P es un *conjunto parcialmente ordenado completo dirigido*, que abreviaremos dcpo[3], o *pre-cpo*, si $\bigvee D$ existe, para cada subconjunto dirigido D de P. Un dcpo P es un *conjunto parcialmente ordenado completo*, que abreviaremos cpo, también llamado *pointed cpo*, si P tiene primer elemento $\bot$. ■

De ahora en más utilizaremos la notación especial $\bigsqcup D$ en lugar de $\bigvee D$ cuando D sea un conjunto dirigido.

Ejemplo A.16

- Los reticulados completos son cpo. En particular, lo son el conjunto de partes de un conjunto $\mathcal{P}(X)$ y los sistemas de clausura.
- Cada conjunto ordenado finito es un cpo.
- El orden flat $X_\bot = X \cup \{\bot\}$, donde $\bot \notin X$, bajo el orden $x \leq y$ si $x = y$ o $x = \bot$ es un cpo. Los conjuntos dirigidos de $X_\bot$ son $\{x, \bot\}$ para cada $x \in X$ y los conjuntos unitarios.
- No toda secuencia creciente tiene una menor cota superior. Por ejemplo, el conjunto de los números naturales con el orden usual no forma un dcpo, ya que tendríamos que agregar un elemento $\top$. Sin embargo, $\mathbb{N}_\bot$ es un cpo.

■

Ejemplo A.17 Un ejemplo más interesante en la Ciencia de la Computación es el de las funciones parciales. Una función parcial la denotaremos $f : A \rightharpoonup B$. Sea $D(f) \subseteq A$ el dominio de f. La notación $f(a) \downarrow$ indica que la función f está definida en a, i.e., $a \in D(f)$. De otro modo, escribimos $f(a) \uparrow$. Sea $\mathcal{F}(\mathbb{N}^k, \mathbb{N})$, simplemente $\mathcal{F}$ el conjunto de las funciones parciales de $\mathbb{N}^k$ en $\mathbb{N}$.

[3]Por sus siglas en inglés: Directed-Complete Partial Ordered Set.

Consideremos el siguiente conjunto:

$$G(f) = \{(\vec{m}, n) \in \mathbb{N}^{k+1} : f(\vec{m}) \downarrow \text{ y } f(\vec{m}) = n\}.$$

La proyección en $\mathbb{N}^k$ es el dominio de f, $D(f)$. Decimos que $f \subseteq g$ si $G(f) \subseteq G(g)$: esta relación indica que g es una extensión de f, i.e., que si $f(\vec{m}) \downarrow$ entonces $g(\vec{m}) \downarrow$ y $f(\vec{m}) = g(\vec{m})$. Si lo pensamos desde el punto de vista de información, podemos decir que ganamos información pasando de f a g.

Consideremos el conjunto ordenado $\langle \mathcal{F}, \subseteq \rangle$. Existe un primer elemento $\varnothing$, la función parcial con dominio vacío. Las funciones totales son los elementos maximales. Dos funciones arbitrarias tienen menor cota superior. Así $\langle \mathcal{F}, \subseteq \rangle$ es un cpo.

A través de la teoría de orden de aproximación podremos formalizar la noción de ecuación recursiva y dar la semántica denotacional a las funciones computables parciales (equivalentes a los programas). El cómputo procede a través de un orden parcial que representa una jerarquía de información o conocimiento. Cuanto más alto esté un elemento en el orden, más específico es y más información contiene. Los elementos más bajos representan conocimiento incompleto o resultados intermedios. La intuición con la *aproximación* es que un paso intermedio, representa un resultado parcial del cómputo de la función parcial. El cómputo de una función parcial será alcanzado a través de sus aproximaciones. ■

De lo anterior es lógico pensar que en la Ciencia de la Computación se trabaje sobre espacios completos. Ver a los conjuntos dirigidos como secuencias de aproximaciones generalizadas y los supremos como el límite de los cómputos aproximados respectivos es la principal motivación para requerir la existencia de las cotas superiores.

Proposición A.1 *Un conjunto parcialmente ordenado D es un dcpo si y solamente si cada cadena no vacía de D tiene supremo.*

Teorema A.13 *Un conjunto parcialmente ordenado D con $\perp$ es un cpo si y solamente si cada cadena no vacía de D tiene supremo.*

Los resultados anteriores, cuya demostración utiliza el Axioma de Elección (ver más adelante), no son triviales y permiten que trabajemos con cadenas ascendentes en vez de trabajar con conjuntos dirigidos.

A.3.1. Funciones Continuas y Punto Fijo

Los conjuntos ordenados, en donde todas sus cadenas son finitas, son estructuras relativamente sencillas de utilizar. Sin embargo, cuando consideramos estructuras más ricas, necesita-

remos propiedades más fuertes que la monotonicidad. Por ejemplo, sean D y E cpos y C una cadena en D. C podría ser un conjunto infinito, entonces podríamos encontrar $\sqcup C$ por una aproximación. Para una función $f : D \to E$ quisiéramos alcanzar $f(\sqcup C)$ por aproximaciones $\{f(c)|c \in C\}$.

Definición A.20 (Función continua)
Sean $\langle P, \subseteq_P \rangle$ y $\langle Q, \subseteq_Q \rangle$ dos órdenes . Una función $\varphi : P \to Q$ es *continua* si para cada conjunto dirigido D en P,

$$\varphi\left(\bigsqcup D\right) = \bigsqcup \varphi\left(D\right) \qquad \left(\bigsqcup\{\varphi(x)|x \in D\}\right)$$

■

Existe una estrecha relación entre las funciones monótonas y las funciones continuas: cada mapeo continuo es monotónico, pero el recíproco no es verdadero.

Teorema A.14 *Sean P y Q dos órdenes parciales y sea $\varphi : P \to Q$.*

1. *Si φ es continua entonces es monótona.*
2. *Si φ es monótona y C es una cadena finita en P, luego $\bigsqcup C$ y $\bigsqcup \varphi(C)$ existen y*
$$\varphi(\bigsqcup C) = \bigsqcup \varphi(C).$$
3. *Si todas las cadenas en P son finitas, φ es continua si y solamente si es monotónica.*

El siguiente teorema sugiere una técnica de prueba de la continuidad mucho más simple cuando los conjuntos P y Q son cpo.

Teorema A.15 *Sean $\langle P, \subseteq_P \rangle$ y $\langle Q, \subseteq_Q \rangle$ cpo y sea $\varphi : P \to Q$. Luego φ es continua si y solamente si:*

1. *φ es monótona; y*
2. *$\varphi(\bigsqcup C) \subseteq_Q \bigsqcup \varphi(C)$ para toda cadena C en P.*

Dado un conjunto ordenado P y un mapeo $\Phi : P \to P$ que preserva el orden, i.e., monotónico, existe un punto fijo para Φ? Una solución $x \in X$ a una ecuación de punto fijo $\Phi(x) = x$, $\Phi : X \to X$, si existe, generalmente es obtenida por un proceso de sucesivas aproximaciones. La teoría de orden tiene un rol que jugar cuando X es un orden y cuando la solución requerida puede ser obtenida como el join de elementos parciales que la aproximan. Antes de ejemplificar el uso del punto fijo en la Ciencia de la Computación, daremos su definición formal.

Definición A.21 (Punto Fijo - Menor Punto Fijo)
Sea $\langle P, \subseteq_P \rangle$ un orden parcial y sea $\Phi : P \to P$. Un elemento $p \in P$ es un *punto fijo* de Φ si $\Phi(p) = p$. El conjunto de todos los puntos fijos de Φ serán denotados $\texttt{fix}(\Phi)$. El primer elemento de $\texttt{fix}(\Phi)$, cuando existe, es el *menor punto fijo* de Φ en P y lo notamos $\mu(\Phi)$. ■

La composición n-fold, Φ^n, de un mapeo $\Phi : P \to P$ es definida como sigue: Φ^n es el mapeo identidad si $n = 0$ y $\Phi^n = \Phi \circ \Phi^{n-1}$ si $n \geq 1$.

Ejemplificaremos la importancia del concepto de punto fijo.

Ejemplo A.18 Función factorial: La función factorial **fact** es el mapeo sobre $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ donde $\mathbf{fact}(x) = x!$.

$$\mathbf{fact}(x) \begin{cases} 1 & \text{si } x = 0, \\ x * \mathbf{fact}(x-1) & \text{si } x \geq 1. \end{cases}$$

La función recursiva anterior puede ser interpretada como una ecuación de punto fijo sobre el operador $\Phi : \mathcal{F}(\mathbb{N}_0, \mathbb{N}_\perp) \to \mathcal{F}(\mathbb{N}_0, \mathbb{N}_\perp)$:

$$f \quad \longmapsto \quad \lambda x \in \mathbb{N}_0.\text{si } x = 0 \text{ entonces } 1 \text{ de otro modo } x * f(x-1).$$

Las aproximaciones sucesivas $f_n = \Phi^n(\perp_{\mathcal{F}(\mathbb{N}_0, \mathbb{N}_\perp)})$ serían:

$$f_0(x) = \perp \text{ para cada } x \in \mathbb{N}_0,$$

$$f_1(x) = \begin{cases} 1 & \text{si } x = 0, \\ \perp & \text{de otro modo} \end{cases}$$

$$f_n(x) = \begin{cases} x! & \text{si } x < n, \\ \perp & \text{de otro modo} \end{cases}$$

Así podremos ver a la función factorial como la solución de la función recursiva $\Phi(f) = f$.

■

Ejemplo A.19 While-Loop: Queremos asignar significado a comandos de la forma *while B do C*. Intuitivamente, la interpretación sería: "mientras que B sea verdadero, haga C repetidamente; una vez que B sea falso, pare en el estado corriente". Así, podemos pensar en:
while B do C = cond(B,(C; while B do C),skip)
siendo

$$cond(P, f, g) = \begin{cases} f & \text{si } P = true, \\ g & \text{de otro modo.} \end{cases}$$

y skip es un comando que no realiza ningún cambio en el estado actual, i.e., "no hace nada". Notamos la composición $A \circ B$ como "B;A". Así, el ciclo while está definido en función de

sí mismo y por lo tanto, podremos reducir el problema a encontrar una solución a una ecuación de punto fijo. ■

El primer teorema de punto fijo presenta un candidato para el menor punto fijo de un mapeo monotónico sobre un cpo y confirma su validez cuando el mapeo es continuo.

Teorema A.16 Teorema I del Punto Fijo para cpos *Sean P un cpo, $\perp$ su primer elemento, $\Phi : P \to P$ un mapeo monótono y $\alpha = \bigsqcup_{n\geq 0} \Phi^n(\perp)$.*

1. *Si $\alpha \in \mathtt{fix}(\Phi)$, entonces $\alpha = \mu(\Phi)$.*
2. *Si Φ es continuo entonces el menor punto fijo $\mu(\Phi)$ existe y es α*

El teorema del punto fijo tiene una variedad de aplicaciones. Una de ellas es la que hemos venido desarrollando y tiene que ver con la justificación de las definiciones recursivas de las funciones. Otra de las aplicaciones es la de justificar las definiciones inductivas de conjuntos. A continuación, ejemplificaremos ambas situaciones.

Ejemplo A.20 Consideremos nuevamente el ejemplo A.18 de la función factorial. Hemos reconocido a la función **fact** como la solución a la ecuación de punto fijo $\Phi(f) = f$ para $f \in (\mathbb{N}_0 \to \mathbb{N}_\perp)$, donde

$$(\Phi(f))(k) \;=\; \begin{cases} 1 & \text{si } k = 0, \\ k * f(k-1) & \text{si } k \geq 1. \end{cases}$$

El resultado de Φ lo podremos visualizar como pares ordenados $(x, f(x))$, lo que denominaremos el grafo de la función. Así, $\Phi(\perp) = \{(0,1)\}$, $\Phi^2(\perp) = \{(0,1),(1,1)\}$, $\Phi^3(\perp) = \{(0,1),(1,1),(2,2)\}$, etc. Claramente, Φ es una función monótona. Una inducción confirma que $f_n = \Phi^n(\perp)$ para todo n, donde $\{f_n\}$ es la secuencia de mapeos parciales que aproximan a **fact**. Formando el join, unión de los grafos, tenemos que $\bigsqcup_{n\geq 0} \Phi^n(\perp)$ es $\mathbf{fact}(k) = k!$ sobre $\mathbb{N}_0$. Este es el menor punto fijo de Φ tanto para funciones parciales sobre $\mathbb{N}_0$ como sobre $\mathbb{Z}$. Bajo el dominio $\mathbb{Z}$ existen otras soluciones a $\Phi(f) = f$, pero contienen información extraña. Naturalmente, deseamos que la información que se nos provea sea la correcta: en este caso deseamos que los elementos del grafo para $\mathbb{N}_0$ definan a **fact**, pero también deseamos no tener información superflua o que no aporte al conocimiento o bien que no tenga un sustento formal. Por esta razón, es que la solución computacional más natural a una ecuación recursiva para un mapeo o un procedimiento es provista por el *menor* punto fijo. ■

Ejemplo A.21 Consideremos nuevamente el ejemplo A.19 del ciclo while. Sucesivas aproximaciones a `while B do C` son dadas para $n \geq 1$ a través de $\{W_n(B,C)\}_{n\geq 1}$, donde $W_n(B,C)$

coincide con "while B do C" para computaciones que involucren menos de n iteraciones del ciclo y es indefinido de otro modo; $W_0(B, C)$ se define como "no hacer nada". La transición desde $W_n(B, C)$ a $W_{n+1}(B, C)$ ejecuta una pasada más por el ciclo. Nótese que una vez alcanzado el menor punto fijo, podremos seguir iterando tantas veces como queramos, aunque dado que la condición B del while será falsa no tendrá sentido computacional. Así, el *menor* punto fijo será la solución computacional adecuada a este problema. ■

Ejemplifiquemos la aplicación del punto fijo como justificación inductiva de las definiciones de conjuntos, que es utilizada a lo largo de este trabajo al introducir nuevos conceptos.

Ejemplo A.22 Consideremos el lenguaje proposicional compuesto de los siguientes átomos $\mathcal{A} = \{p, q\}$ y de las fórmulas con los conectivos $\neg$ y $\rightarrow$. Así el alfabeto es $B = \{p, q, \neg, \rightarrow, (,)\}$. Una definición de las fórmulas proposicionales sobre el alfabeto B es el menor subconjunto (con respecto a $\subseteq$) de B^* que:

1. contiene a $\mathcal{A}$,
2. es cerrado bajo la operación que transforma α en $(\neg\alpha)$, y
3. es cerrado bajo la operación que transforma α y β en $(\alpha \rightarrow \beta)$.

En otras palabras, el conjunto de fórmulas proposicionales sobre $\mathcal{A}$ es el menor conjunto $X \subseteq B^*$ que satisface

$$\mathcal{A} \cup \{(\neg\alpha) | \alpha \in X\} \cup \{(\alpha \rightarrow \beta) | \alpha, \beta \in X\} \subseteq X \tag{A.1}$$

Dado que la ecuación (A.1) puede ser satisfecha aun si X contiene elementos que no cumplen las tres condiciones de la definición y ya que estamos buscando el conjunto minimal que cumpla (A.1), podremos reescribir (A.1) como una igualdad:

$$X \;=\; \mathcal{A} \cup \{(\neg\alpha) | \alpha \in X\} \cup \{(\alpha \rightarrow \beta) | \alpha, \beta \in X\} \tag{A.2}$$

La ecuación (A.2) puede ser analizada considerando el lado derecho como una función $\Phi : \mathcal{P}(B^*) \rightarrow \mathcal{P}(B^*)$ que toma un subconjunto $Z \subseteq B^*$ y lo transforma en

$$\Phi(Z) \;=\; \mathcal{A} \cup \{(\neg\alpha) | \alpha \in Z\} \cup \{(\alpha \rightarrow \beta) | \alpha, \beta \in Z\}.$$

Una solución a la ecuación (A.2) es encontrar algún X tal que cuando aplicamos Φ a X, el resultado sigue siendo X; esto es, X es un punto fijo de Φ.
Dado que la definición requiere del menor conjunto de fórmulas, i.e., $\mu(\Phi)$. Luego para que esta definición tenga sentido necesitamos saber si $\mu(\Phi)$ existe. En este punto es donde el teorema del punto fijo es útil. Hemos mostrado que el orden parcial $\langle \mathcal{P}(D), \subseteq) \rangle$ es un cpo para cualquier

conjunto D, luego $\langle \mathcal{P}(B^*), \subseteq) \rangle$ es un cpo. Si Φ es continuo, luego $\mu(\Phi)$ existe por el teorema del punto fijo. Sea $\mathcal{C}$ una cadena de $\mathcal{P}(B^*)$. Luego $\bigsqcup \mathcal{C} = \cup \mathcal{C}$ en $\langle \mathcal{P}(B^*) \subseteq \rangle$, y

$$\begin{array}{rcl} \Phi(\cup\, \mathcal{C}) & = & \mathcal{A} \ \cup \ \{(\neg\alpha) | \alpha \in \cup\, \mathcal{C}\} \ \cup \ \{(\alpha \to \beta) | \alpha, \beta \in \cup\, \mathcal{C}\} \\ & = & \mathcal{A} \ \cup \ (\cup \{\{(\neg\alpha) | \alpha \in C\} | C \in \mathcal{C}\}) \ \cup \ (\cup \{\{(\alpha \to \beta) | \alpha, \beta \in C\} | C \in \mathcal{C}\}) \\ & = & \cup \{\mathcal{A} \cup \{(\neg\alpha) | \alpha \in C\} \cup \{(\alpha \to \beta) | \alpha, \beta \in C\} | C \in \mathcal{C}\} \\ & = & \cup \{\Phi(C) | C \in \mathcal{C}\} \\ & = & \cup \Phi(\mathcal{C}) \end{array}$$

Por lo tanto, Φ es continuo y $\mu(\Phi)$ existe.

Nótese que esta definición de punto fijo no es operacional ya que no menciona el proceso de construcción de las fórmulas a partir de $\mathcal{A}$, aunque se da la versión declarativa en la que se indica que el conjunto es la solución a una cierta ecuación. Sin embargo, podríamos definir el conjunto de fórmulas diciendo que es la aplicación reiterada de las operaciones $\neg$ y $\to$. El teorema de punto fijo, que no solamente nos dice que $\mu(\Phi)$ existe sino que también caracteriza al menor punto fijo $\mu(\Phi) = \cup\{\Phi^i(\varnothing) | i \in \mathbb{N}\}$ realiza la conexión entre la parte operacional y la declarativa. Así, $\Phi(\varnothing), \Phi^2(\varnothing), \ldots$ son subconjuntos de $\mu(\Phi)$ que se obtienen por aplicación de las operaciones de construcción de las fórmulas. ■

Otras aplicaciones en la Ciencia de la Computación, como el uso de esta herramienta en la definición de lenguajes y al problema de verificación de programas, pueden hallarse entre otros en [MV72, DP02, Lal93, AJ94, DSW94, Hro04, Hol03].

No siempre es posible pedir a las funciones entre cpos una propiedad tan fuerte como la continuidad. El siguiente teorema clásico del punto fijo que se lo debemos a Knaster y a Tarski, debilita las características deseadas para el mapeo, pero exige que el conjunto ordenado sea un reticulado completo.

Teorema A.17 Teorema del Punto Fijo Knaster-Tarski
Sea $\langle L, \leq \rangle$ un reticulado completo y $\Phi : L \to L$ un mapeo que preserva el orden (monótono). Luego

$$\bigvee \{x \in L | x \leq \Phi(x)\} \in \mathtt{fix}(\Phi).$$

El teorema de Knaster-Tarski muestra que cualquier mapeo monótono sobre el conjunto de partes de un conjunto arbitrario tiene un punto fijo.

Lema A.9 *Sean P un conjunto ordenado y $\Phi : P \to P$ un mapeo. Sea $H = \{x \in P | x \leq \Phi(x)\}$.*

(1) Supongamos que $A \subseteq H$ y $\Phi(A) \subseteq A$. Luego, cualquier elemento maximal de A pertenece a $\mathtt{fix}(\Phi)$.

(II) *Si* Φ *es monótono, entonces* $\Phi(H) \subseteq H$ *y*

(a) $\bigvee_P H \in \mathtt{fix}(\Phi)$ *si éste existe,*

(b) *cualquier elemento maximal de* H *pertenece a* $\mathtt{fix}(\Phi)$.

Los siguientes teoremas de punto fijo sobre cpos nos permitirá trabajar con mapeos monótono sobre cpo, asegurándonos que el mapeo tiene un punto fijo.

Teorema A.18 Teorema II del Punto Fijo para cpos *Sean* P *un cpo,* $\Phi : P \to P$ *un mapeo tal que* $x \leq \Phi(x)$ *para todo* $x \in P$. *Luego* Φ *tiene un punto fijo.*

Teorema A.19 Teorema III del Punto Fijo para cpos *Sean* P *un cpo,* $\Phi : P \to P$ *un mapeo monótono. Luego* Φ *tiene un menor punto fijo.*

Si el operador Φ es monótono, luego $\varnothing \subseteq \Phi(\varnothing) \subseteq \Phi(\Phi(\varnothing)) \subseteq \ldots$. Luego deseamos iterar hacia el punto fijo. Una secuencia incremental de subconjuntos de X es computada del siguiente modo

$$\Phi^{\uparrow 0} = \varnothing,$$

$$\Phi^{\uparrow n+1} = \Phi(\Phi^{\uparrow n}),$$

$$\Phi^{\omega} = \bigcup_{n \geq 0} \Phi^{\uparrow n}$$

La suposición de que Φ es monótono no implica que Φ^{ω} sea punto fijo. El proceso podría continuar con $\Phi^{\omega+1} = \Phi(\Phi^{\omega})$. Sin embargo, existe una propiedad que nos permite determinar si un proceso termina y es cuando un conjunto es dirigido, ya que cada subconjunto finito de ese conjunto tiene cota superior. Recordemos que todo reticulado completo es un conjunto dirigido.

Proposición A.2 Proposición del Punto Fijo para operadores contínuos *Sean* P *un reticulado completo y* $\Phi : P \to P$ *un mapeo continuo. Luego* $\mu(\Phi) = \Phi^{\uparrow \omega}$

En resumen, hemos dado cuatro teoremas que garantizan la existencia del menor punto fijo:

- El Teorema I del Punto Fijo para cpos, que se aplica a mapeos continuos sobre cpos.
- El Teorema del Punto Fijo de Knaster-Tarski, que se aplica a mapeos monótonos sobre reticulados completos.
- Teorema III del Punto Fijo para cpos, que se aplica a mapeos monótonos sobre cpos.
- La Proposición del Punto Fijo para operadores contínuos.

Los dos primeros teoremas cuyas hipótesis son más fuertes permiten una caracterización de la fórmula del menor punto fijo. La proposición nos da una definición denotacional del operador.

Una consecuencia de estos teoremas es el Principio de Inducción de Punto Fijo.

Principio de Inducción de Punto Fijo: Sea Φ un mapeo continuo sobre un cpo P y sea $S \subseteq P$ que satisface:

(IPF)$_1$ $\bot \in S$;

(IPF)$_2$ $x \in S$ implica $\Phi(x) \in S$;

(IPF)$_3$ para cada cadena $x_0 \leq x_1 \leq \ldots \leq x_n \leq \ldots$ en S, tenemos $\bigsqcup_{n\geq 0} x_n \in S$.

Entonces, por el Teorema I del Punto Fijo para cpo, $\mu(\Phi) \in S$.

Uno de los tópicos más importantes en la Ciencia de la Computación es la verificación de programas: probar matemáticamente que dado un programa, este termina y que su salida es lo que realmente se desea. En estos casos es posible utilizar el Principio de Inducción de Punto Fijo. Esta aplicación queda fuera del alcance de este trabajo, aunque se recomienda la lectura de [DSW94, Hro04, Hol03].

A.3.2. Existencia de elementos maximales

Finalmente, haremos una relación entre la convergencia computacional y la existencia de los elementos maximales, sin entrar en un análisis detallado. La existencia de los elementos maximales está sustentada en el Axioma de Elección y el Lema de Zorn, que son equivalentes.

Supongamos que deseamos elegir un gobernador por cada provincia en un país, asumiendo que al menos una persona vive en cada provincia. ¿Puede Ud. imaginar un país donde esto no sea posible? Sería un país donde no se podría elegir un representante por provincia. Esto parece absurdo, ya que eventualmente podría ir provincia por provincia y seleccionar alguien al azar y designarlo gobernador (solo estamos pidiendo un representante y no, un *buen* representante). (Ejemplo extraído y adaptado de [Mil08])

En el argumento anterior hemos utilizado lo que se conoce como el Axioma de Elección, formulado en 1904 por Ernst Zermelo [Zer04].

> **Axioma de Elección (AC)**[Zer04]: Dada una familia no vacía $\mathcal{A} = \{A_i\}_{i\in I}$ de conjuntos no vacíos, existe una **función de elección** para $\mathcal{A}$, es decir un mapeo
>
> $$f : I \to \bigcup_{i\in I} A_i \text{ tal que } (\forall i \in I) f(i) \in A_i.$$

Intuitivamente, el Axioma de Elección (AC) dice que si tenemos una colección de conjuntos no vacíos, cada uno con al menos un objeto, luego podemos tomar exactamente un objeto de cada conjunto, aun si existen infinitos conjuntos y si no existe ninguna regla predeterminada para tomar un objeto de cada uno de los conjuntos.

Ejemplo A.23 Un conjunto ordenado no vacío puede, pero no necesariamente tiene que, poseer elementos maximales. Consideremos el siguiente argumento que soporta el hecho de que un cpo P tiene un elemento maximal. Supongamos que ningún elemento de P es maximal. Esto significa que, para cada $x \in P$, el conjunto $\{y \in P | y > x\}$ es no vacío. Para cada x seleccione un punto $y > x$. Ya que y depende de x, lo etiquetaremos $\Phi(x)$. Luego $\Phi(x)$ define un mapeo sobre P y por el Teorema II del Punto Fijo sobre cpo, Φ tiene punto fijo. Luego, existe $z \in P$ tal que $z = \Phi(z)$, pero dado que $x < \Phi(x)$ para cada $x \in P$, tenemos una contradicción. Por lo tanto, P tiene un elemento maximal.

Para aplicar el AC sobre este caso tenemos que $I = P$ y $A_x = \{y \in P | y > x\}$ para cada $x \in P$. ■

No todas las situaciones requieren de la aplicación del AC. Por ejemplo, no es necesario si el número de conjuntos es finito, ya que el AC se deriva de otros axiomas de la teoría de conjunto. En este caso es equivalente a, dado un conjunto finito de cajas, cada una con al menos un objeto, tomar exactamente un ítem de cada una de ellas. Claramente, podemos hacer esto: comenzamos con la primer caja y elegimos un objeto, vamos a la segunda y elegimos un objeto y así sucesivamente. Como hay un número finito de cajas, nuestro procedimiento de elección terminará.

En el caso de las elecciones provinciales de gobernador, seguramente no ha pensado Ud. que el país tiene infinitas provincias. Ahora imaginemos tal país, tratemos de listar las provincias en algún orden y elijamos un representante por cada una de ellas. El proceso no terminaría después de un número finito de pasos. El poder del Axioma de Elección radica en que nos permite hacer infinitas elecciones arbitrarias de una vez. Si la elección hay que hacerla sobre una cantidad finita de conjuntos no vacíos, simplemente sacamos uno de cada uno de ellos, y aquí no necesitamos aplicar el Axioma de Elección.

Para ciertos conjuntos infinitos, es posible evitar la aplicación del AC. Esto se da si existe una manera clara y no ambigua de elegir a un elemento específico de cada uno de los conjuntos, i.e., si existe una regla de selección. Un ejemplo de Bertrand Russell nos ayudará a clarificar este caso. Supongamos que tenemos un conjunto infinito de pares de zapatos. Luego, sin aplicar el AC, Ud. puede elegir un zapato de cada par, por ejemplo, el izquierdo. Esta es una elección no ambigua y bien definida. Ahora suponga que dispone de una colección infinita de pares de medias indistinguibles. No existe una regla clara que pueda emplear para seleccionar una media de cada

par. En este caso necesitamos aplicar el AC. Ejemplificaremos esta situación con los números naturales. Consideremos un conjunto infinito de conjuntos no vacíos de números naturales. Cada conjunto de números naturales tiene un menor elemento, por ser bien ordenado, luego para especificar la función de elección simplemente decimos que tomamos de cada conjunto el menor de él. Esto da una elección definida de un elemento por cada conjunto. Así, cada vez que es posible especificar tal elección, el AC no es necesario. La dificultad aparece cuando no existe una elección natural de elementos por cada conjunto. Si no podemos hacer una elección explícita, ¿cómo sabremos que nuestro conjunto resultante existe? Por ejemplo, consideremos el conjunto X de todos los subconjuntos no vacíos de números reales. Como X es infinito, no podremos ir subconjunto por subconjunto como lo haríamos con un conjunto X de cardinalidad finita, ya que el proceso no terminaría. Entonces tendríamos que determinar una función de selección no ambigua. Claramente, la elección del menor elemento no funciona ya que existen subconjuntos, como el intervalo $(0, 1)$, que no tienen un menor elemento. Sin embargo, la aceptación del AC nos permite concluir que existe al menos una función de elección. El problema es que el AC no es un método constructivista y no indica cómo obtener tal función. Por esta razón, muchos matemáticos no aceptan su formulación.

Existen dos afirmaciones que tienen una estrecha relación con el AC: el lema de Zorn y el Teorema del Buen Orden. Cada una de ellas es equivalente a las otras dos.

> **Lema de Zorn** (LZ): Sea P un conjunto ordenado no vacío en el cual cada cadena no vacía tiene cota superior. Entonces P tiene un elemento maximal.

Una formulación equivalente al LZ, restringida a cpo es la que sigue:

> (LZ′): Sea P un cpo. Entonces P tiene un elemento maximal.

Finalmente, presentamos algunas definiciones que nos ayudarán a enunciar el Teorema del Buen Orden.

Definición A.22 (Relación bien fundada)
Una relación binaria R sobre un conjunto P es *bien fundada* si cada subconjunto S de P contiene un elemento minimal s_0 en S.

Equivalentemente, una relación es *bien fundada* si y solo si no existe ninguna cadena descendiente infinita, esto es, no existe una secuencia infinita de elementos $s_0,\ s_1,\ s_2,\ \ldots$ de P tal que $s_{n+1}Rs_n$ para todo número natural n. ■

Definición A.23 (Orden bien fundado)
Un orden parcial es llamado *bien fundado* si su correspondiente orden estricto es una relación bien fundada. ■

A continuación, daremos la definición de conjunto bien ordenado, que nos permitirá formular el Teorema del Buen Orden.

Definición A.24 (Buen Orden) (Conjunto bien ordenado)
Un *buen orden* es un orden total bien fundado. El conjunto P junto con la relación bien ordenada es llamado *conjunto bien ordenado*. ■

Intuitivamente, un conjunto bien ordenado está ordenado de tal modo que sus elementos pueden ser considerados uno por vez, en orden y cada elemento tiene un único sucesor.

Ejemplo A.24
1. El orden estricto usual es bien fundado sobre $\mathbb{N}$, pero no sobre $\mathbb{Z}$ o $\mathbb{Q}$. Así $\mathbb{N}$ con el orden usual es un conjunto bien ordenado.

   Este orden no es el único, existen otros buen orden sobre el mismo conjunto $\mathbb{N}$. Por ejemplo, $0 \prec 2 \prec 4 \prec 6 \prec \ldots \prec 1 \prec 3 \prec 5 \prec \ldots$ es un buen orden para $\mathbb{N}$.

2. En lógica, el razonamiento debería ser bien fundado. Esto es, si escribimos $j > j'$, j es la conclusión de la regla cuya premisa es j', y la secuencia debería tener un elemento minimal.

■

Como hemos visto en el ejemplo, no existe un único buen orden sobre un conjunto, así el concepto de ordinal clasifica a los buenos órdenes. Un *ordinal* es un conjunto que es transitivo y bien ordenado por la relación $\in$. Un conjunto T es *transitivo* si cada elemento de T es un subconjunto de T. Equivalentemente, un conjunto T es transitivo si $A \in T$ siempre que $A \in B$ y $B \in T$.

Ejemplo A.25 Podemos construir los siguientes ordinales partiendo del conjunto vacío:

$$\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}$$

Escribiendo $0, 1, 2$ y 3 obtenemos $1 = \{0\}$, $2 = \{0, 1\}$ y $3 = \{0, 1, 2\}$, así $0 \in 1 \in 2 \in 3$. ■

Todos los ordinales finitos pueden ser obtenidos desde 0 o $\varnothing$ a través de la función sucesor. Si α es un ordinal, entonces $\alpha \cup \{\alpha\}$ también lo es. Este es el sucesor ordinal de α y se denota $\alpha + 1$. El primer ordinal infinito es $\omega = \{0, 1, 2, 3, \ldots\}$.

Un ordinal α se dice *ordinal límite* si no es sucesor de ningún ordinal. El ordinal límite más pequeño es 0 y el siguiente es ω. Por supuesto, $\omega + 1 = \omega \cup \{\omega\}$, $\omega + 2 = (\omega + 1) + 1$, $\omega + 3, \ldots$ son ordinales. El siguiente ordinal límite es $\omega 2$ y el siguiente $\omega 3$ y así continúa.

En la práctica, la importancia del buen orden está justificada por la posibilidad de aplicar la inducción transfinita. Esencialmente, la inducción transfinita dice que cualquier propiedad que pasa de un conjunto de ordinales más chico que el ordinal dado al ordinal mismo, es verdadera para todos los ordinales. Si los estados de un cómputo pueden ser bien ordenados de tal modo que cada paso es seguido por otro paso "menor", podremos garantizar que el cómputo termina.

El siguiente teorema, formulado por Zermelo, es el resultado de una crisis de los fundamentos durante 1904 [FD04, Tor98].

Teorema A.20 Teorema del Buen Orden *[Zer04]*
Cada conjunto X puede ser bien ordenado.

Retomemos la discusión sobre la selección de la función correspondiente al Axioma de Elección para $\mathbb{R}$. La razón por la que elegir el menor elemento de cada subconjunto de los $\mathbb{N}$ es una función de selección correcta, es que $\mathbb{N}$ tiene un orden con la propiedad de ser un buen orden. El teorema anterior nos asegura que los reales están bien ordenados, aunque no nos indica cuál es el orden que debemos utilizar para que esto sea verdadero. Así, nuestro pensamiento debería ser que *aunque el orden usual sobre los números reales no funciona como lo esperábamos, es posible encontrar un orden diferente que sea un buen orden y entonces formular sobre ese orden una función de elección no ambigua y bien definida*, quizás el menor elemento de cada subconjunto bajo ese orden.

Nuestra discusión sobre el tema finaliza con un teorema fundamental, que relaciona las tres sentencias.

Teorema A.21 *Son equivalentes:*

(a) El Axioma de Elección

(b) El Teorema del Buen Orden.

(c) El Lema de Zorn

Esta equivalencia, particularmente la del AC y el teorema del Buen Orden, causó un gran desconcierto entre los especialistas. Inclusive aquellos matemáticos que utilizaban el AC, lo declararon inadmisible al ver que se lo invocaba para probar que el continuo se puede ordenar.

Índice de Temas

Índice de Autores