Universidad de Sevilla
Departamento de Filosofía y
Lógica y Filosofía de la Ciencia

Modelos Formales de Explicación en Lógica e Inteligencia Artificial

Tesis presentada por
Fernando Soler Toscano
para optar al grado de Doctor
por la Universidad de Sevilla

Fernando Soler Toscano

V. B. Director

Ángel Nepomuceno Fernández

Sevilla, 12 de junio de 2005

La realización de esta tesis ha contado con la subvención de la Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía a través de una beca dentro del programa para la formación de doctores en universidades y centros de investigación andaluces.

Índice general

Capítulo 1

Introducción

—Usted pareció sorprenderse cuando le dije, en nuestra primera entrevista, que había venido de Afganistán —comentó Sherlock Holmes a Watson.

—Alguien se lo habría dicho, sin duda alguna.

—¡De ninguna manera! Yo descubrí que usted había venido de Afganistán. Por la fuerza de un largo hábito, el curso de mis pensamientos es tan rígido en mi cerebro, que llegué a esa conclusión sin tener siquiera conciencia de las etapas intermedias. Sin embargo, pasé por esas etapas. El curso de mi razonamiento fue el siguiente: "He aquí a un caballero que responde al tipo del hombre de medicina, pero que tiene un aire marcial. Es, por consiguiente, un médico militar con toda evidencia. Acaba de llegar de países tropicales, porque su cara es de un fuerte color oscuro, color que no es el natural de su cutis, porque sus muñecas son blancas. Ha pasado por sufrimientos y enfermedad, como lo pregona su cara macilenta. Ha sufrido una herida en el brazo izquierdo. Lo mantiene rígido y de una manera forzada... ¿En qué país tropical ha podido un médico del ejército inglés pasar por duros sufrimientos y resultar herido en un brazo? Evidentemente, en Afganistán". Toda esa trabazón de pensamientos no me llevó ni un segundo. Y entonces hice la observación de que usted había venido de Afganistán, lo cual lo dejó asombrado.

A.C. Doyle, *Estudio en Escarlata*

1.1. ¿Qué es la abducción?

Durante más de dos mil años, lo que se entendía por *lógica* era fundamentalmente la *silogística*, fundada por Aristóteles (384–322 a.C.) y continuada principalmente por la Escolástica en la Edad Media. En los *Analíticos primeros*, Aristóteles investiga las formas correctas de inferencia. Allí define el *silogismo* como un razonamiento en el que, establecidas algunas cosas, se sigue necesariamente otra distinta de ellas, por el mero hecho de estar ellas establecidas. En el cuadro 1.1 recogemos un ejemplo de silogismo. Resulta obvio que la verdad

Los animales sin bilis tienen larga vida
Pero el hombre, el caballo y la mula tienen no tienen bilis

Luego el hombre, el caballo y la mula tienen larga vida

Cuadro 1.1: Ejemplo de silogismo

de la conclusión —la tercera oración, debajo de la línea— se sigue de la verdad de las dos premisas, es decir, no es posible que las premisas sean verdaderas y la conclusión sea falsa. La silogística se encarga, pues, de determinar las formas gramaticales —a las que subyace cierta forma lógica— de los razonamientos correctos, con el objeto de poder aplicarse a las diferentes ciencias.

Así como Aristóteles es el fundador de la lógica antigua, la lógica moderna —que ya llamamos *clásica*— tiene su punto de arranque en Frege (1848–1925). En la *Conceptografía* (1879), Frege vuelve a enfrentarse a la tarea de sistematizar el razonamiento[1]. A diferencia de Aristóteles, Frege no estudia los razonamientos en su propia lengua, sino que crea un nuevo lenguaje, puramente formal, en el que es posible representar los razonamientos de modo abstracto. Frege es el punto de inflexión en el que la lógica se convierte en una ciencia formal. Pero también, es el fundador del *proyecto logicista*, que tenía por empeño el fundamentar toda la matemática en la lógica.

Si hemos comenzado esta sección acudiendo a la historia de la lógica es para subrayar cómo ésta ha tenido fundamentalmente un carácter *deductivo*, de es-

[1]Entre Aristóteles y Frege ya había habido varios intentos más, siendo los de Leibniz (1646–1716) los de mayor repercusión.

tablecer nuevas verdades a partir de verdades ya establecidas. Esto es lo que pretendió la silogística y también lo que, a partir de Frege, se consuma con la introducción de los sistemas formales.

Pero hay otras formas no deductivas de razonamiento, menos afortunadas en cuanto a la atención que históricamente han recibido, entre las que se encuentra la *abducción*. De ello se dio cuenta Charles Sanders Peirce (1839–1914), iniciador del movimiento pragmatista en Norteamérica. Para Peirce, el pragmatismo es sobre todo un método de pensamiento orientado a aclarar nuestras ideas. Peirce considera que son pocas las ideas *infalibles*, que pueden establecerse de una vez para siempre. Salvo en el terreno de la matemática y la lógica, donde las ideas son infalibles y pueden aplicarse métodos deductivos, el resto de ideas son *falibles*, es decir, se mueven en el terreno de la *hipótesis*. Al principio, Peirce llama *hipótesis* a la forma de razonamiento que posteriormente llamó *abducción* y *retroducción*. Como observa A. Aliseda [Ali98], a estos cambios de terminología acompañan ciertas variaciones en la concepción que Peirce tiene de la abducción. Peirce caracteriza la abducción en forma de silogismo, de la siguiente manera:

> El hecho sorprendente, C, es observado. Pero si A fuera verdad, C sería aceptado como algo evidente. Por lo tanto, hay razón para sospechar que A es verdad (*CP 5.189, 1903*).

Para G. Génova [Gen96], el descubrimiento[2] de este modo de inferencia debe datarse alrededor de 1865, cuando Peirce encuentra que ya Aristóteles, en sus *Analíticos primeros* recoge, junto al razonamiento deductivo, otros silogismos en que el orden de las proposiciones se altera, produciendo esquemas de inferencia que aunque no son deductivamente válidos sí que se asemejan a ciertos razonamientos de sentido común. Así encuentra Peirce la *apagogé*, una forma de razonamiento que Aristóteles ilustra con el ejemplo que recogemos en al cuadro 1.2. La conclusión de este razonamiento no es necesaria, pero sí probable, ya que está sugerida por las premisas.

[2] G. Debrock [Deb98] encuentra también precedentes en Nicolás de Cusa (1401–1464) e incluso en Friedrich Schiller (1759–1805).

Los animales sin bilis tienen larga vida
Pero el hombre, el caballo y la mula tienen larga vida

Luego el hombre, el caballo y la mula no tienen bilis.

Cuadro 1.2: Ejemplo de *apagogé*

Junto a la abducción, la deducción y la inducción son para Peirce los tres modos de inferencia; sin embargo, la abducción es la única que puede generar ideas nuevas:

> La abducción es el proceso de formar una hipótesis explicativa. Es la única operación lógica que introduce alguna idea nueva; pues la inducción no hace más que determinar un valor, y la deducción desarrolla meramente las consecuencias necesarias de una pura hipótesis.
>
> La deducción prueba que algo *debe* ser, la inducción muestra que algo es *realmente* operativo; la abducción meramente sugiere que algo *puede ser*.
>
> Su única justificación es que, a partir de su sugerencia, la deducción puede extraer una predicción que puede comprobarse por inducción; y que, si es que podemos llegar a aprender algo o a entender completamente los fenómenos, debe ser por abducción como esto se lleve a cabo (*EP 2:216, 1903*).

Junto a estas tres formas de razonamiento, que como indica Bell [Bel] son más bien términos imprecisos que cada autor llena de contenido de diferente forma, es frecuente considerar la *analogía* como otro modo de inferencia. Gilles Défourneaux y Nicolas Peltier [DP97] indican que para Peirce la analogía se puede ver como una inducción y una abducción seguidas por una deducción. Es decir, cuando tratamos de resolver un problema por analogía lo que hacemos es, en primer lugar, mediante abducción e inducción, buscar un dominio *análogo* al del problema que tratamos de resolver, pero mejor conocido para nosotros. Entonces razonamos deductivamente en este segundo dominio y trasladamos *analógicamente* las conclusiones al dominio del problema.

Antes de continuar, digamos que en ocasiones nos referiremos al razonamiento abductivo como *razonamiento explicativo*. Con ello entendemos algo diferente a Peirce, para quien el razonamiento explicativo es el que Kant llama *analítico*. En el razonamiento analítico, la conclusión explicita cierta información que se encuentra contenida en las premisas, tal como hace el argumento del cuadro 1.1. Por contraposición, en el razonamiento *sintético*, la conclusión contiene información que no estaba en las premisas. A esta segunda clase sólo pertenecen la inducción y la abducción.

Características	**Gripe**	**Resfriado**
Inicio	Súbito	Paulatino
Fiebre	38–41°	Rara
Mialgia	Sí	No
Cefalea	Muy intensa	Rara
Tos (productiva)	No	Sí
Dolor lumbar	Sí	No
Estornudos	Raro	Sí
Rinorrea	A veces	Sí
Odinofagia	A veces	Sí
Irritación ocular	A veces	Sí
Secreción nasal acuosa	A veces	Sí (primeros días)
Duración	3–5 días	8–10 días
Virus	*Ortomixovirus* (influenza A y B)	Primavera y verano *picornavirus*, y en otoño e invierno *paramixovirus*

Cuadro 1.3: Diferencias entre los síntomas del resfriado y de la gripe

Volviendo a la abducción, veamos algunos ejemplos para ilustrar cómo se trata de una forma insegura pero interesante de razonamiento. En la página web del Colegio Oficial de Farmacéuticos de Valencia, http://www.cofv.es/, aparece una tabla para poder distinguir entre los síntomas del resfriado y de la gripe, que recogemos en el cuadro 1.3. Cada fila de esta tabla indica, para el resfriado y para la gripe, si cierto síntoma aparece o no. La tabla ha sido compuesta a partir del trabajo de especialistas que, conociendo ambas enfermedades, dan cuenta de los

síntomas de cada una de ellas. Sin embargo, su mayor utilidad se revela cuando se usa en sentido contrario[3] a como se compuso, es decir, cuando se recorre de los síntomas a la enfermedad, con lo que se hace de ella un uso abductivo. En este caso, se trata de un diagnóstico, que es una de las tareas para las que actualmente se considera interesante la abducción.

Un buen diagnóstico observa los síntomas del paciente y determina la enfermedad que más probablemente tenga. Parafraseando a Peirce, diríamos: "el conjunto de síntomas C es observado en el paciente; pero si el paciente tuviera la enfermedad A, los síntomas C serían los esperables; por tanto, hay razones para sospechar que el paciente tiene la enfermedad A".

El razonamiento abductivo es siempre inseguro, pues se basa en una conjetura, una sospecha, como el mismo Peirce indica. La conclusión abductiva —la enfermedad A, por ejemplo— puede invalidarse si se descubren nuevos hechos —que, por ejemplo, pueden llevarnos a concluir que el paciente no tiene la enfermedad A, sino otra más rara, B, con el mismo conjunto de síntomas C—. Por ello, como indica A. Aliseda [Ali98], la abducción puede entenderse como un cambio epistémico.

La aplicación que hemos comentado a la diagnosis nos hace pensar en la importancia, para la correcta abducción, del *ojo clínico*, que el diccionario de la Real Academia Española define como la "capacidad para captar una circunstancia o preverla". Esta capacidad la tenemos todos, cada uno en el dominio de conocimiento en que tiene mayor experiencia. Peirce considera esta capacidad como un "*flash* de entendimiento". Es este *flash* lo que Sherlock Holmes explica al doctor Watson en la cita con la que abrimos este capítulo[4]. Se trata de una sucesión de pensamientos que le han servido a Holmes para concluir que Watson viene de Afganistán. Curiosamente, esta cita pertenece a un capítulo titulado "La ciencia de la deducción", pero precisamente aquello en lo que Holmes es todo un maestro, como queda claro en los párrafos seleccionados, es en la abducción, pues

[3]En ocasiones se llama a la abducción "deducción hacia atrás", o "retroducción", término del propio Peirce.

[4]Fragmento de A.C. Doyle, *Estudio en Escarlata*, citado por Sheila A. McIlraith [McI98], tomado de Umberto Eco y Thomas A. Sebeok [ES83].

como buen detective es capaz de analizar —en menos de un segundo, debido al ojo clínico que la da su gran experiencia— las pistas que observa —los rasgos de Watson— para concluir aquello que las explica.

Uno de los campos donde más se estudia la abducción es en Filosofía de la Ciencia, como más adelante veremos, pues se considera una de las formas —para algunos la única— en que se generan las hipótesis científicas. Así, se reconoce el valor de la abducción en la práctica científica. Pero no sólo en las ciencias empíricas, sino que también se ha subrayado su importancia en la práctica matemática, por ejemplo en la enunciación de lemas que sirvan para probar teoremas complejos. Sin embargo, cuando se completan las pruebas de estos teoremas, su presentación siempre es deductiva —como debe ser, realmente— y se olvida que hubo un razonamiento de tipo abductivo para encontrar tales demostraciones. Esto contribuye a que la abducción no haya recibido el mismo tratamiento formal que la deducción.

Recientemente, el tratamiento formal de la abducción se ha convertido en un tema de gran interés, sobre todo por las aplicaciones que encuentra en diversas disciplinas, fundamentalmente relacionadas con la Inteligencia Artificial. Por ello existen bastantes acercamientos que tratan la abducción con los mismos formalismos que originalmente se desarrollaron con fines deductivos. ¿Tiene sentido esta tarea? Probablemente el propio Peirce se habría mostrado en contra de tratar la abducción con herramientas creadas para la deducción, pues se corre el peligro de reducir la abducción a deducción camuflada, y entonces deja de tener interés. Aún así, pensamos que merece la pena correr este riesgo si los resultados son sugerentes en algún sentido.

1.2. Aplicaciones del razonamiento abductivo

Al ser la abducción un modo de inferencia habitual dentro del razonamiento de sentido común, encuentra aplicaciones en diferentes disciplinas. En primer lugar, precisamente por su vinculación al razonamiento natural, la abducción se

ha convertido en objeto de estudio por parte de la Ciencia Cognitiva, disciplina que A. Gomila comprende como

> un programa científico comprometido con la teoría representacional de la mente, surgido en parte como reacción al predominio del conductismo en psicología, para el que debía explicarse la conducta como función de los estímulos. En cambio, para el cognitivismo es preciso postular representaciones mentales (que según el enfoque concreto adoptan la forma de esquemas, de modelos mentales, de *scripts* o *frames*, de proposiciones, de imágenes, etc.) que median entre los estímulos y la conducta, para dar cuenta de la flexibilidad y adaptabilidad (o racionalidad e inteligencia), que la distingue[5].

Uno de los enfoques más importantes dentro de la Ciencia Cognitiva es el que encabezan M.D.S. Braine [Bra78] y L. Rips [Rip83, Rip94], quienes sostienen que existe una *lógica mental*[6] que hace que el razonamiento humano se guíe por reglas formales, de modo parecido a como lo hace la deducción natural. Por ejemplo, para explicar por qué los humanos solemos tener más dificultades al aplicar la regla de *modus tollens* que la de *modus ponens* en los tests de razonamiento, ellos sostienen que mientras que el *modus ponens* sería una regla —mental— primitiva, el *modus tollens* tendría que ser derivado cada vez que hiciera falta usarlo.

En contra del formalismo de la *Lógica Mental*, P.N. Johnson-Laird [JL83] funda la corriente conocida como *Modelos Mentales*, que postula que al razonar no usamos reglas formales independientes del contexto —como el *modus ponens*— sino que procedemos construyendo modelos de las premisas con que contamos, y extrayendo conclusiones de dichos modelos. Mediante los modelos mentales, los seguidores de esta corriente pretenden dar cuenta de la dependencia del razonamiento humano respecto del contexto en que se mueve.

[5] A. Gomila [Gom96], Sección 3, "De la Semiótica a la Ciencia Cognitiva".

[6] La *Lógica Mental*, como teoría cognitiva, se debe a M.D.S. Braine y colaboradores. L. Rips es el creador de PSYCOP, un sistema de inferencia deductiva implementado en Prolog que sigue una concepción muy similar a la de Braine. Es por ello que reunimos a ambos autores dentro de un mismo enfoque.

Si bien las dos corrientes anteriores conceden más o menos importancia a la lógica en el razonamiento humano, existen otras propuestas que defienden que la lógica no tiene ninguna relevancia. Así, L. Cosmides [Cos89] sostiene que si en ocasiones parecemos razonar lógicamente es porque seguimos estrategias que se han ido desarrollando durante la evolución de nuestra especie para resolver problemas concretos.

Pese a orientarse la mayor parte de los experimentos fundacionales de la Ciencia Cognitiva a evaluar —y posteriormente explicar— las capacidades deductivas del razonamiento humano, la abducción es, aunque poco estudiada, un objetivo primordial, al estar a la base de buena parte de las inferencias que realiza la mente humana, que usualmente no se mueve dentro de la seguridad de la deducción, sino que opera en el terreno incierto de la conjetura, la sospecha, la hipótesis. En los textos de Peirce encontramos, de hecho, múltiples elementos para realizar un análisis cognitivo de la abducción. Por ejemplo, su afirmación de que la abducción es la única operación lógica que introduce nuevas ideas, el valor de la creatividad, o la propia idea del *flash* de entendimiento, contienen múltiples sugerencias cognitivas.

J. Nubiola [Nub98] relaciona la abducción con la creatividad, y recoge la siguiente cita, donde Peirce habla del *musement*, un juego libre del pensamiento, meditación sin más regla que la libertad.

> Sube al bote del *musement*, empújalo en el lago del pensamiento y deja que la brisa del cielo empuje tu navegación. Con tus ojos abiertos, despierta a lo que está a tu alrededor o dentro de ti y entabla conversación contigo mismo; para eso es toda meditación (*CP 6.461, 1908*).

A. Aliseda [Ali98] llama la atención sobre la importancia que tiene la *sorpresa* en la inferencia abductiva. La sorpresa —que para A. Aliseda puede tener la forma de *novedad* o *anomalía* [Ali97]— es el *detonador abductivo*. El mismo Peirce, en la caracterización de la abducción, considera el carácter *sorprendente* de la observación que despierta en la mente la *duda*, y con ella el impulso a explicarla.

Otro de los rasgos de la abducción con mayor importancia cognitiva es su doble carácter *intuitivo* y *racional*. En ciertas ocasiones, Peirce habla de la abducción como un *método* para generar buenas hipótesis, pero en otras la define como un *hábito* o capacidad. Como subraya A. Aliseda [Ali98], esta dualidad se ha convertido en ocasiones en un conflicto para los estudiosos de Peirce, tanto que ha llegado a conocerse incluso como "el dilema de Peirce". La relevancia de esta contraposición reside en que si el componente metodológico es el más importante, entonces es posible caracterizar congnitivamente el razonamiento abductivo al modo de un *proceso*. Por el contrario, si la abducción se debe principalmente a un hábito, o algún tipo de intuición, entonces no es posible más que dar constancia de los *productos* abductivos en determinados contextos.

También en Lingüística resulta interesante el estudio de la abducción. Existen propuestas, como la de P. Krause [Kra02], que emplean la abducción, dentro de la teoría de representación del discurso, para el estudio de las presuposiciones lingüísticas. Según esta idea, cuando escuchamos la oración "todos los amigos de Pedro se encuentran satisfechos en sus trabajos" y suponemos que todos los amigos de Pedro trabajan, lo que realizamos es una inferencia abductiva.

Más allá de la presuposición, Nubiola [Nub98] generaliza la importancia de la abducción a toda la actividad lingüística ordinaria, lo que ilustra con la siguiente cita de Peirce:

> Al mirar por mi ventana esta hermosa mañana de primavera veo una azalea en plena floración. ¡No, no! No es eso lo que veo; aunque sea la única manera en que puedo describir lo que veo. *Eso* es una proposición, una frase, un hecho; pero lo que yo percibo no es una proposición, ni una frase, ni un hecho, sino sólo una imagen, que hago inteligible en parte mediante un enunciado de hecho. Este enunciado es abstracto, mientras que lo que veo es concreto. Realizo una abducción cada vez que expreso en una frase lo que veo. La verdad es que toda la fábrica de nuestro conocimiento es una tela entretejida de puras hipótesis confirmadas y refinadas por la inducción. No puede realizarse el menor avance en el conocimiento más allá de la mirada vacía si no media una abducción en cada paso (*MS 692, 1901*).

Pero la contribución mayor de Peirce a la Lingüística se encuentra en la semiótica, de la que es fundador. Al estudiar el significado, Peirce distingue —como explica A. Gomila [Gom96]— entre el *representamen*, que es el signo lingüístico en cuanto objeto material, el *objeto*, al que se refiere el *representamen*, y el *interpretante*, efecto mental del *representamen* en el intérprete. También aquí encuentra la abducción una importante función, al considerar Peirce que el modo en que el *representamen* determina el *interpretante* es siempre mediante una inferencia abductiva.

Donde más se ha discutido la relevancia de la abducción es en la Filosofía de la Ciencia, al cuestionar su importancia en los procesos de descubrimiento científico. Es habitual la distinción entre el *contexto de descubrimiento*, en que se elabora cierta teoría científica, y el *contexto de justificación*, en que dicha teoría se establece. Si bien resultan más o menos claros los modos en que se lleva a cabo la justificación científica —el experimento en ciencias empíricas, la demostración en ciencias formales, etc.— no ocurre lo mismo con los procesos de descubrimiento. Es en este contexto donde se sitúa la polémica sobre la importancia del razonamiento abductivo. Autores como Lipton [Lip] —quien defiende que el descubrimiento científico se realiza al modo de una *inferencia de la mejor explicación*— defienden la importancia del razonamiento explicativo en la ciencia. Sin embargo, existen otras propuestas más críticas con la relevancia que pueda tener la abducción.

Seguiremos a S. Paavola [Paa04] para comentar los argumentos clásicos que se han formulado en contra de que la abducción se pueda comprender como una lógica del descubrimiento. Una primera crítica consiste en que según el esquema de inferencia abductiva que proporciona Peirce —(*CP 5.189, 1903*), ver más arriba— el único requisito que se exige a la hipótesis A que debe explicar cierto hecho sorprendente C es que si A fuera verdadera entonces C sería algo común. Sin embargo, no se exige que la hipótesis A sea algo esperable, por lo que el criterio es demasiado permisivo. En esta línea, Achinstein proporciona algunos ejemplos de inferencias abductivas absurdas. Según uno de estos ejemplos, si encontramos a algún amigo que dice estar feliz por cierta noticia que ha recibido, podríamos abducir la hipótesis de que su felicidad se debe a que la noticia que ha recibido es que ha ganado el premio Nobel de literatura, porque en tal caso sería normal

que estuviese muy feliz. Pero posiblemente no tengamos ninguna razón para suponer que nuestro amigo haya sido siquiera propuesto por nadie para el Nobel de literatura.

Otro grupo de críticas que recoge Paavola son las que defienden que la abducción no puede constituir una lógica del descubrimiento porque la hipótesis ya está incluida, de uno u otro modo, en las premisas, pues antes de abducir A ya debemos saber que A explicaría C, según el esquema de Peirce. Sin embargo, el descubrimiento científico introduce nuevas ideas en la ciencia, por lo que no podría ser el resultado de una inferencia abductiva. En este sentido, algunos autores relegan la abducción a un lugar intermedio entre el descubrimiento y la justificación. Dicho lugar sería el de la evaluación preliminar de las hipótesis ya descubiertas —de alguna otra forma—, antes de su justificación. Según estas tesis, el contexto de descubrimiento permanece como algo inexplicable, que cae fuera del análisis conceptual.

Para defender la abducción como lógica del descubrimiento, Paavola acude a la distinción que hace Hintikka, partiendo de la teoría de juegos, entre *reglas definidoras* y *reglas estratégicas*. Son reglas definidoras las que establecen los movimientos legales de un juego. Por el contrario las reglas estratégicas determinan cuál, de entre los movimientos legales, es el más acertado. Hintikka piensa que la lógica ha estado durante mucho tiempo centrada exclusivamente en las reglas definidoras de los cálculos, pero se ha atendido poco a las condiciones necesarias para obtener una prueba estratégicamente —según algún criterio que vaya más allá de la corrección— buena.

Siguiendo estas ideas, Paavola indica que es necesario incluir las estrategias en la caracterización del razonamiento abductivo. Así, la hipótesis explicativa no sólo debe cumplir que si fuera verdadera entonces lo sería también la observación que pretende explicar. Ahora, la hipótesis debe ser además estratégicamente buena. Entonces, la hipótesis de que a nuestro amigo le han concedido el Nobel de literatura resulta bastante inapropiada, por ser estratégicamente pésima, ya que seguramente es falsa.

Respecto del segundo tipo de objeciones recogidas, Paavola recurre al mismo Peirce:

> Es verdad que los diferentes elementos de la hipótesis estaban antes en nuestra mente; pero es la idea de relacionar lo que nunca antes habíamos soñado relacionar lo que ilumina de repente la nueva sugerencia ante nuestra contemplación (*CP 5.181, 1903*).

En efecto, lo que la hipótesis sugiere puede ser algo ya conocido. Pero ello no implica que la hipótesis deje de ser creativa, pues probablemente nunca se había pensado en ella como explicativa del fenómeno para el que ahora se emplea. A propósito de esto, Paavola recuerda que muchas de las ideas de Darwin no eran novedosas en su tiempo si se las considera por separado —ni siquiera la misma idea de evolución—. Pero la originalidad de Darwin consistió en relacionar todas esas ideas y aplicarlas para explicar unos fenómenos que ninguna de tales ideas, en solitario, habría podido explicar.

Sin ninguna duda, el campo donde más aplicaciones encuentra hoy día el razonamiento abductivo es la Inteligencia Artificial. Es también ésta la disciplina a la que más nos hemos acercado en este trabajo, especialmente a lo que se conoce como Programación Lógica Abductiva, o simplemente ALP, por sus siglas en inglés.

En 1965, tras diez años de investigación en Demostración Automática, Robinson [Rob65] aporta una idea que resultó sumamente fructífera: la combinación de resolución y unificación. Con ello, se reducía considerablemente el número de términos del universo de Herbrand que hacían falta en las pruebas, pues gracias a la idea de Robinson sólo se genera una nuevo término cuando resulta necesario para buscar contradicciones. Partiendo de estas ideas, en los años 70 aparece Prolog, de la mano de Colmerauer y Kowalski, al sistematizar trabajos anteriores como los de Boyer y Moore, que habían desarrollado algoritmos de unificación según las ideas de Robinson.

Con Prolog aparece la Programación Lógica, lo que supone un fuerte impulso para muchas de las áreas de la Inteligencia Artificial. También el razonamiento ab-

ductivo recibe un amplio tratamiento dentro de la Programación Lógica. En 1993, A.C. Kakas, R.A. Kowalski y F. Toni publican su trabajo *Abductive Logic Programming*, donde dan cuenta de numerosas aplicaciones que la abducción había encontrado ya en el marco de la Programación Lógica. En 1998, bajo el título *The Role of Abduction in Logic Programming* [KKT98], revisan el trabajo de 1993. Las aplicaciones del razonamiento abductivo en Inteligencia Artificial de que dan cuenta en este trabajo son:

Diagnosis. Como ya hemos comentado, la abducción está a la base de los procesos de diagnosis. Por ello, cuando se trata de automatizar este tipo de procesos, también es frecuente que se emplee el razonamiento abductivo, especialmente en el campo conocido como *diagnosis basada en modelos*. En este caso, se cuenta con una teoría que describe el comportamiento *normal* del sistema que va a diagnosticarse, y la abducción consiste en crear hipótesis de tipo "el componente A del sistema funciona de forma anómala", que deben explicar por qué todo el sistema funciona de modo anormal.

Visión de alto nivel. La visión artificial es usada en numerosas aplicaciones, tales como los sistemas de vigilancia. En este caso, las observaciones que se explican abductivamente son las imágenes parciales de los objetos que puede proporcionar una cámara. Las hipótesis explicativas son los objetos que producen tales imágenes.

Comprensión del lenguaje natural. También se emplea la abducción en los sistemas de *Procesamiento del Lenguaje Natural*, en este caso para interpretar oraciones ambiguas. Las hipótesis abductivas, ahora, son las posibles interpretaciones, determinándose por el contexto cuál es la más plausible.

Planificación. La planificación es una de las áreas de la Inteligencia Artificial con más aplicaciones industriales. En este caso, la observación que debe explicarse es el *estado objetivo* al que debe llegarse, y la explicación es el *plan* que para ello es necesario llevar a cabo. La mejor explicación corresponderá al mejor —más corto, más económico, etc.— plan.

Asimilación de conocimientos. Muchas de las aplicaciones de la Inteligencia Artificial están relacionadas con la representación y asimilación de conocimientos. En este caso, puede emplearse la abducción para aumentar las

bases de conocimientos no tanto por acumulación sino más bien por asimilación de los nuevos datos. Así, al llegar un nuevo dato, se incorporan a la base de conocimientos las hipótesis capaces de explicar dicho dato.

Razonamiento por defecto. El razonamiento por defecto, que permite la aplicación de ciertas reglas generales en ausencia de contradicciones, se emplea frecuentemente para modelar sistemas en que no es posible razonar según los parámetros de la lógica clásica. Como indica P. Flach [Fla94], el razonamiento por defecto puede ser comprendido abductivamente.

En cuanto a las técnicas empleadas para implementar la abducción, debemos tener en cuenta que los programas lógicos son ejecutados mediante pruebas por resolución, de modo que la demostración de que el hecho A es una consecuencia del programa lógico P se hace mediante una prueba por resolución SLD[7] de que la negación de A es contradictoria con P. Sin embargo, si A no es consecuencia lógica de P, la prueba por resolución SLD llega a un estado que puede representarse como un árbol tal que ninguna de sus ramas alcanza el objetivo. Cada una de tales ramas termina en lo que se conoce como un *final muerto* —*dead end*, en inglés—, que contiene los hechos que sería necesario probar para que A quedase establecido. Pues bien, precisamente a partir de tales finales muertos se construyen las hipótesis abductivas, pues asumiendo su contenido se dispone de una explicación de A. Este es el proceder que siguen, por ejemplo, P. Flach [Fla94] y D. Poole, A. Mackworth, y R. Goebel [PMG98] en los programas abductivos que proponen. También está basado en este mismo sistema la propuesta de M. Denecker y D. De Schreye [DS98] de implementar lo que llaman resolución SLDNFA, un procedimiento abductivo que aprovecha la información contenida en los finales muertos de las pruebas fallidas en los programas lógicos.

El desarrollo de las numerosas aplicaciones que como hemos visto encuentra la Programación Lógica Abductiva, ha llevado también a la proliferación de numerosas técnicas para optimizar la búsqueda de explicaciones, de por sí bastante compleja. Una que nos parece interesante destacar aquí es el empleo de predicados *abducibles*. Dada una cierta teoría, se califica de *abducibles* sólo a ciertos

[7] En el apéndice A presentamos una introducción a Prolog, donde explicamos más detalladamente la resolución SLD.

predicados, de forma que son tales predicados los únicos que podrán aparecen en las explicaciones. Obsérvese que de esta forma se sale al paso del primero de los dos tipos de críticas a la abducción que recoge Paavola, y que antes comentamos: la crítica de que la explicación podría ser cualquier cosa, lo que llevaba a Achinstein a formular ejemplos de hipótesis absurdas. Ahora, por ejemplo, sólo calificaríamos de abducibles, para explicar la felicidad de nuestro amigo, aquellos predicados que de alguna forma se relacionen con su entorno. Por tanto, el uso de predicados abducibles lleva a la abducción a depender del contexto en que se realiza.

Otros elementos interesantes desarrollados en el marco de la Programación Lógica Abductiva son los criterios preferenciales que habitualmente se establecen para considerar mejores unas explicaciones que otras. Criterios basados en la longitud de una explicación —cantidad de hechos que supone—; en la preferencia por explicaciones últimas, no explicables a su vez dentro de la teoría, etc. Más adelante tendremos ocasión de volver sobre estas ideas.

1.3. Problemas abiertos

Terminamos este capítulo recogiendo los principales problemas que aparecen al tratar de desarrollar una lógica abductiva. J. Hintikka [Hin98], para quien la caracterización del razonamiento abductivo es el problema fundamental de la epistemología contemporánea, proporciona, siguiendo a T. Kapitan, cuatro tesis, a modo de acercamiento a los principales rasgos de la abducción:

Tesis inferencial. La abducción es, o incluye, un *proceso* inferencial.

Tesis de objetivo. El propósito de la abducción científica es doble: en primer lugar generar nuevas hipótesis y posteriormente seleccionar las mejores para su análisis.

Tesis de comprensión. La abducción científica incluye todas las operaciones por las que se engendran las teorías.

Tesis de autonomía. La abducción es un tipo de razonamiento irreductible tanto a la deducción como a la inducción.

Al hilo de estas cuatro tesis vamos a preguntarnos si, fiel a los anteriores rasgos, es posible una *lógica abductiva*. M. Hoffmann [Hof98], quien también se hace esta pregunta, considera que para hablar de una lógica de la abducción debemos considerar una noción de lógica más abierta que la imperante, de tipo deductivo o analítico, dedicada únicamente a "definir la validez de llegar a una verdad desde otra verdad".

En cuanto a la primera de las tesis de Kapitan, la inferencial, nos indica que debemos pensar en la abducción como un *proceso*. J. van Benthem [Ben03] considera que debido a la influencia de las Ciencias de la Computación en la Lógica, en las últimas décadas se ha producido lo que él llama un *giro dinámico*. Si tradicionalmente la lógica se ha ocupado de tareas *declarativas*, ya sea la representación del conocimiento o la definición de las relaciones de consecuencia, ahora la atención se ha vuelto sobre los *procesos* de inferencia. Entonces, "la representación de la información no puede separarse de los procesos que usan y transforman dicha información". J. van Benthem contrapone los *procesos* a sus correspondientes *productos*, constatando que el giro dinámico que ha experimentado la lógica se centra, en primer lugar, en los primeros.

También la *tesis inferencial* resalta el carácter de proceso del razonamiento abductivo. No sólo resultan interesantes las hipótesis que se generan, o las propiedades que verifican, sino que, en primer lugar, interesan los procesos que engendran tales hipótesis. En este sentido, Hintikka observa que el propio Peirce se refiere a la abducción como un método de generar buenas conjeturas. La idea de *método* alude a este carácter de proceso. Por tanto, la primera de las tesis nos impone que para que exista una lógica abductiva, debe seguir un método abductivo. Es decir, si quisiéramos calificar de abductivo cierto sistema formal, no sólo tendrían que ser buenas hipótesis los productos de sus inferencias, sino que el propio proceder de las inferencias debería ser, en algún sentido, abductivo. Como más tarde veremos, esta exigencia deja fuera ciertos acercamientos formales a la abducción en que los productos son aceptables, pero cuyos procesos difícilmente son reconocibles como abductivos.

La segunda tesis nos habla de un doble propósito de la abducción: la *generación* de hipótesis y la *selección* de las mejores. Esta distinción se relaciona con la que Hintikka hace, como hemos comentado, entre reglas definidoras y reglas estratégicas. Aunque la generación de las hipótesis pudiera realizarse exclusivamente mediante reglas definidoras, para la selección de las mejores necesitamos reglas estratégicas. Entran en juego los criterios preferenciales para elegir una hipótesis como mejor que las demás, aunque todas puedan explicar igualmente los hechos. Por tanto, el proceso abductivo debe incluir tanto la generación de hipótesis explicativas como la generación de las mejores, o preferibles, según algún criterio.

La tercera tesis, de comprensión, defiende la abducción como una lógica para la generación de nuevas teorías científicas. Como más arriba hemos comentado, en esta idea no coinciden todos los autores. En cualquier caso, esto nos dice que una lógica abductiva debe reunir suficientes herramientas como para engendrar teorías. En este sentido, consideramos fundamental que también fuera posible profundizar en los procesos mentales que acompañan a las inferencias abductivas humanas.

Por último, la tesis de autonomía, nos previene contra la reducción —frecuente en ciertos formalismos— de la abducción a deducción o inducción. Los procesos abductivos deben ser irreductibles a la deducción o a la inducción. En esta línea, Hoffmann [Hof98] considera que una lógica abductiva debe ser una "lógica contextualizada", ya que —según este autor— la articulación concreta de los diversos contextos en una situación determina de manera específica el campo de las hipótesis posibles. Esta contextualización distingue la abducción de cualquier otra forma de razonamiento.

En ocasiones Peirce se refiere a la abducción como "retroducción". Este término ha sugerido a ciertos autores la definición de la abducción como deducción hacia atrás, en el sentido de que se parte de una "conclusión" y debe encontrarse la "premisa" que falta. En ocasiones se ha querido ver una dualidad entre la deducción y la abducción, considerándolas dos formas diferentes y complementarias de recorrer —hacia adelante y hacia atrás— los mismos argumentos. La sugerencia

de esta dualidad nos parece de un interés suficiente como para considerar su elucidación otro de los objetivos a los que debiera enfrentarse una lógica abductiva.

1.4. Objetivos y metodología

El principal objetivo de este trabajo es el desarrollo de un cálculo que podamos calificar de *abductivo*, es decir, un cálculo que genere soluciones explicativas para los problemas abductivos que se planteen. Este cálculo deberá satisfacer los siguientes requisitos:

1. Integrar las concepciones de *abducción como proceso* y *abducción como producto*, lo que requiere que el cálculo no sólo devuelva soluciones abductivas correctas —producto—, sino que su propio proceder sea de alguna forma *abductivo* —proceso—, del mismo modo que podemos considerar deductiva cada aplicación de una regla en un cálculo deductivo natural. Esto es algo que no satisfacen los cálculos abductivos habituales, generalmente más centrados en los productos que en los procesos.

2. Integrar los procesos de *generación* y *selección* de explicaciones. Habitualmente se procede generando un conjunto de posibles explicaciones, y después seleccionando las que, de entre las posibles, son válidas. Lo que pretendemos es que *generación* y *selección* no se realicen mediante procesos diferentes, sino que se integren en el mismo cálculo.

3. Permitir incorporar *criterios preferenciales*. Los criterios preferenciales resultan muy útiles a la hora de seleccionar *la mejor explicación*. Pueden entenderse como algo similar a la idea de *reglas estratégicas* de Hintikka que —como se ha indicado— Paavola [Paa04] considera fundamentales para el razonamiento abductivo.

4. Mejorar la eficiencia y la expresividad con respecto a otros acercamientos. En concreto, deberá ser posible resolver —de forma eficiente— problemas abductivos en que la teoría no sea necesariamente representable mediante cláusulas de Horn y la observación no tenga por qué ser un literal.

5. Permitir explorar las relaciones que existen entre deducción y abducción, especialmente la *dualidad* que a veces se insinúa cuando se califica a la abducción de *deducción hacia atrás* o *retroducción*.

6. Permitir explorar los procesos que se llevan a cabo cuando la mente se enfrenta a situaciones reales donde entran en juego inferencias explicativas.

El objetivo último al que se encaminan los anteriores es acercarnos a la pregunta —que para autores como Aliseda continúa abierta— sobre si existe una *lógica abductiva*. Que nuestra respuesta sea o no afirmativa dependerá de si logramos o no los anteriores objetivos, es decir, de si el cálculo que definamos se puede considerar o no *abductivo*.

La metodología que seguimos a lo largo de este trabajo se puede dividir en tres tareas, diferentes pero complementarias, que se irán alternando. Se trata de:

1. Estudios formales. Consiste en presentar el cálculo de δ-resolución, y probar los teoremas fundamentales que permiten su uso para la resolución de problemas abductivos. Con este cálculo, dual al de resolución, podremos explorar las relaciones formales que existen entre deducción y abducción.

2. Implementaciones. Cada uno de los sistemas que se presenten se implementará en SWI-Prolog, versión[8] 5.4.3, desarrollado por la Universidad de Amsterdam.

3. Análisis comparativos de eficiencia. Se tratará de resolver los problemas abductivos incluidos en el apéndice C con cada uno de los sistemas implementados, midiendo en cada caso los gastos de espacio, tiempo e inferencias.

[8]Que era la última versión estable cuando se escribieron los programas.

Capítulo 2

Abducción como problema lógico

2.1. Preliminares lógicos

En esta sección introducimos los elementos básicos de lógica proposicional que serán empleados a lo largo de los capítulos 3–6. Seguimos la presentación de [Nep03], con algunas modificaciones. Comenzamos definiendo el lenguaje proposicional.

Definición 2.1 (Lenguaje proposicional) *El lenguaje formal $\mathcal{L}_p$ consta de:*

1. Operadores lógicos proposicionales: "$\neg$", "$\wedge$", "$\vee$", "$\rightarrow$", "$\leftrightarrow$", *llamados* negación, conjunción, disyunción, implicación *y* doble implicación, *respectivamente.*

2. $\mathcal{P}$, *conjunto de las* variables proposicionales, *que representaremos mediante letras minúsculas del alfabeto latino, con subíndices y superíndices cuando sea necesario.*

3. $FOR(\mathcal{L}_p)$, *conjunto de fórmulas de $\mathcal{L}_p$, que es el más pequeño que verifica*

   a) *Si $\alpha \in \mathcal{P}$, entonces $\alpha \in FOR(\mathcal{L}_p)$, y además a α se le llama fórmula atómica,*

b) *Si* $\alpha \in FOR(\mathcal{L}_p)$, *entonces* $\neg\alpha \in FOR(\mathcal{L}_p)$,

c) *Si* $\alpha, \beta \in FOR(\mathcal{L}_p)$, *entonces*

$$(\alpha) \vee (\beta), \quad (\alpha) \wedge (\beta), \quad (\alpha) \rightarrow (\beta), \quad (\alpha) \leftrightarrow (\beta) \in FOR(\mathcal{L}_p).$$

En adelante, salvo que se especifique lo contrario, emplearemos letras griegas minúsculas para referirnos a fórmulas de $\mathcal{L}_p$, tal como hemos hecho en la anterior definición. Igualmente, usaremos letras griegas mayúsculas para referirnos a conjuntos de fórmulas de $\mathcal{L}_p$. Así, mediante $\alpha \in \mathcal{L}_p$ indicamos que α es una fórmula proposicional, mientras que con $\Gamma \subset \mathcal{L}_p$ expresamos que Γ es un conjunto de fórmulas proposicionales.

A las fórmulas no atómicas las llamamos *complejas* o *moleculares*. Además, tanto a las fórmulas atómicas —variables proposicionales— como a sus negaciones, las llamamos *literales*. Cada literal γ será o bien *positivo*, si $\gamma \in \mathcal{P}$, o negativo, si γ es $\neg\lambda$ y $\lambda \in \mathcal{P}$. Por tanto, si $\eta \in \mathcal{P}$, decimos que η y $\neg\eta$ son literales complementarios. En ciertas ocasiones usaremos la notación $\overline{\lambda}$ para referirnos al literal complementario —positivo o negativo— del literal —negativo o positivo, respectivamente— λ.

Decimos que β es una subfórmula de α si y sólo si[1] β es una fórmula que o bien es α o forma parte de la fórmula α. El *alcance* de un operador lógico viene dado por las subfórmulas mayores a las que tal operador afecta. Así, el alcance del negador en $\neg(\alpha)$ es α, el alcance del disyuntor en $(\alpha) \vee (\beta)$ son las subfórmulas α y β, y así sucesivamente.

Cuando el alcance de un operador lógico pueda determinarse sin ambigüedad, podemos prescindir de algunos paréntesis. Así, podemos escribir $p \rightarrow q$ en vez de $(p) \rightarrow (q)$. Esto también es válido cuando se emplean letras griegas minúsculas para representar esquemas de fórmulas. Así, usaremos $\alpha \wedge \beta$ para representar $(\alpha) \wedge (\beta)$, por ejemplo. Aún pueden eliminarse más paréntesis teniendo en cuenta la *precendencia* de cada operador, que determina —de modo convencional— el modo en que debe desambiguarse la estructura de una fórmula en ausencia de

[1] En muchas ocasiones abreviaremos la expresión "si y sólo si" mediante "syss", especialmente en la enunciación y demostración de teoremas.

paréntesis, de forma que el alcance de cada operador sea mayor cuanto mayor es su precedencia. El orden de los operadores lógicos de menor a mayor precedencia es: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$.

Llamamos *grado lógico* de una fórmula al número de operadores lógicos que contiene. Así, el grado lógico de los literales positivos es 0, y el de los negativos 1. Igualmente, si n y m son, respectivamente, los grados lógicos de α y β, tenemos que el grado lógico de $\neg\alpha$ será $n+1$, y el grado lógico de $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$ y $\alpha \leftrightarrow \beta$ será $m+n+1$. La noción de grado lógico de una fórmula será importante para la demostración de ciertos teoremas.

Definición 2.2 (Valoración booleana) *Una* valoración booleana *—o simplemente* valoración*— es una función*

$$v : FOR(\mathcal{L}_p) \mapsto \{0,1\}$$

de forma que para cada variable proposicional $\gamma \in \mathcal{P}$, $v(\gamma) \in \{0,1\}$, y para cada $\alpha, \beta \in \mathcal{L}_p$,

1. $v(\neg\alpha) = 1$ *syss* $v(\alpha) = 0$.
2. $v(\alpha \vee \beta) = 1$ *syss* $v(\alpha) = 1$ *ó* $v(\beta) = 1$.
3. $v(\alpha \wedge \beta) = 1$ *syss* $v(\alpha) = 1$ *y* $v(\beta) = 1$.
4. $v(\alpha \rightarrow \beta) = 1$ *syss* $v(\alpha) = 0$ *ó* $v(\beta) = 1$.
5. $v(\alpha \leftrightarrow \beta) = 1$ *syss* $v(\alpha) = v(\beta)$.

A $v(\alpha)$ lo llamamos el *valor de verdad* de α para la valoración v, que siempre será 0 —falso— ó 1 —verdadero—. De los razonamientos en que, haciendo uso de la definición 2.2, se infiera el valor de verdad de una fórmula a partir del valor de verdad de sus subfórmulas, o viceversa, decimos que operan *por evaluación de* los operadores lógicos que entren en juego. Así por ejemplo, desde $v(\alpha) = 1$ y $v(\beta) = 0$ inferimos, *por evaluación de la implicación*, que $v(\alpha \rightarrow \beta) = 0$.

Definición 2.3 (Satisfactibilidad) *Una fórmula $\alpha \in \mathcal{L}_p$ es* satisfactible *syss existe una valoración booleana v tal que $v(\alpha) = 1$, en cuyo caso decimos que*

v satisface α. *Asimismo, un conjunto* $\Gamma \subset \mathcal{L}_p$ *es* simultáneamente satisfactible *—o simplemente* satisfactible*, si no conduce a ambigüedad— syss existe una valoración* v *tal que para cada* $\alpha \in \Gamma$, $v(\alpha) = 1$. *En tal caso decimos que* v satisface Γ.

En adelante, indicaremos que la valoración v satisface la fórmula α mediante $v \vDash \alpha$. También, mediante $v \vDash \Gamma$ expresamos que v satisface simultáneamente a Γ. Cuando una valoración v satisfaga cierta fórmula $\alpha \in \mathcal{L}_p$ (o cierto conjunto $\Gamma \subset \mathcal{L}_p$), diremos también que v es un *modelo* de α (o de Γ).

Definición 2.4 (Consecuencia lógica, independencia) *Una fórmula* $\alpha \in \mathcal{L}_p$ *es* consecuencia lógica *de un conjunto* $\Gamma \subset \mathcal{L}_p$*, en símbolos* $\Gamma \vDash \alpha$*, syss toda valoración que satisface* Γ *también satisface* α*. En otro caso, decimos que* α *es* semánticamente independiente *—o simplemente* independiente— de Γ*, lo que en símbolos expresamos como* $\Gamma \nvDash \alpha$.

Definición 2.5 (Validez universal) *Una fórmula* $\alpha \in \mathcal{L}_p$ *es* universalmente válida*, en símbolos*[2] $\vDash \alpha$*, syss para toda valoración* v, $v(\alpha) = 1$. *A las fórmulas universalmente válidas las llamamos* tautologías. *Igualmente, decimos que* $\Gamma \subset \mathcal{L}_p$ *es un conjunto de fórmulas* universalmente válido*, en símbolos* $\vDash \Gamma$*, syss para toda valoración* v *se cumple que* $v \vDash \Gamma$. *El conjunto de fórmulas vacío,* $\emptyset$*, es* universalmente válido.

Teorema 2.6 *Para todo conjunto* $\Gamma \subset \mathcal{L}_p$ *se verifica* $\vDash \Gamma$ *syss para cada* $\alpha \in \Gamma$ *se cumple* $\vDash \alpha$.

Demostración:

Sea $\Gamma \in \mathcal{L}_p$ un conjunto de fórmulas. Primero demostraremos que si $\vDash \Gamma$ —Γ es universalmente válido— entonces para cada $\alpha \in \Gamma$ se cumple $\vDash \alpha$. A continuación probaremos lo recíproco.

Supongamos que se cumple $\vDash \Gamma$. Entonces, por la definición 2.5, tenemos que para toda valoración v, $v \vDash \Gamma$. Por la definición 2.3 esto significa que para toda v

[2] Como se observa, el símbolo "$\vDash$" recibe varios usos. Lo usamos para expresar *satisfactibilidad*, relaciones de *consecuencia lógica* y *validez universal*.

se cumple que, para cada $\alpha \in \Gamma$, $v \vDash \alpha$. Pero esto, por la definición 2.5, significa que para cada α ocurre $\vDash \alpha$.

Ahora consideremos que para cada $\alpha \in \Gamma$ se cumple que $\vDash \alpha$. Entonces, por la definición 2.5, esto significa que para cada valoración v, y para toda $\alpha \in \Gamma$, $v \vDash \Gamma$. Pero esto, por la definición 2.3, significa que toda valoración v satisface Γ, lo que por la definición 2.5 implica que Γ es universalmente válido, es decir, $\vDash \Gamma$. ■

Definición 2.7 (No satisfactibilidad) *Una fórmula $\alpha \in \mathcal{L}_p$ es* no satisfactible, *en símbolos $\alpha \vDash \bot$, syss no existe ninguna valoración v tal que $v(\alpha) = 1$. Decimos de una fórmula no satisfactible que es* contradictoria. *Igualmente, un conjunto $\Gamma \subset \mathcal{L}_p$ es* no satisfactible, *en símbolos $\Gamma \vDash \bot$, syss no existe ninguna valoración v tal que $v \vDash \Gamma$.*

El símbolo "$\bot$" lo usaremos para indicar "lo falso", y puede tomarse como una abreviatura de cualquier fórmula no satisfactible —siempre falsa, por tanto—, como $p \wedge \neg p$. Por ello, mediante $\alpha \vDash \bot$ se expresa que toda valoración v que satisfaga α debe satisfacer $\bot$ —o si se quiere $p \wedge \neg p$—, pero como ninguna valoración satisface $\bot$, tenemos que ninguna valoración satisface α.

Definición 2.8 (Contingencia) *Decimos de una fórmula $\alpha \in \mathcal{L}_p$ o de un conjunto $\Gamma \subset \mathcal{L}_p$ que son* contingentes *si no son ni universalmente válidos ni no satisfactibles. Es decir, que existe al menos una valoración v que satisface α —o Γ— y otra valoración v' que no satisface α —respectivamente, Γ—.*

En adelante, para expresar que cierta fórmula β es consecuencia lógica del conjunto $\Gamma \cup \{\alpha_1, \ldots, \alpha_n\} \subset \mathcal{L}_p$, escribiremos indiferentemente $\Gamma \cup \{\alpha_1, \ldots, \alpha_n\} \vDash \beta$ o bien $\Gamma, \alpha_1, \ldots, \alpha_n \vDash \beta$.

A continuación presentamos algunos teoremas semánticos que se emplearán con frecuencia en los posteriores capítulos. Por esto, y por considerarlos elementales, en muchas ocasiones omitiremos su referencia.

Teorema 2.9 *Para cualesquiera* $\Gamma \subset \mathcal{L}_p$ *y* $\alpha, \beta \in \mathcal{L}_p$*, se cumple que*

$$\Gamma, \alpha \vDash \beta \quad syss \quad \Gamma \vDash \alpha \rightarrow \beta \tag{2.1}$$

$$\Gamma, \alpha \vDash \bot \quad syss \quad \Gamma \vDash \neg\alpha \tag{2.2}$$

$$\Gamma, \neg\alpha \vDash \bot \quad syss \quad \Gamma \vDash \alpha \tag{2.3}$$

Demostración:

Comenzamos probando (2.1). Supongamos que $\Gamma, \alpha \vDash \beta$. Entonces toda valoración que satisfaga $\Gamma \cup \{\alpha\}$ satisface β. Ahora bien, sea v una valoración que satisface Γ. Entonces, por la definición 2.2 sólo pueden ocurrir dos cosas:

- Que $v(\alpha) = 0$, en cuyo caso $v(\alpha \rightarrow \beta) = 1$.
- Que $v(\alpha) = 1$, pero en este caso v satisface $\Gamma \cup \{\alpha\}$, y por hipótesis v satisface β, es decir, $v(\beta) = 1$, pero por la definición 2.2 volvemos a tener $v(\alpha \rightarrow \beta) = 1$.

En ambos casos tenemos que $v(\alpha \rightarrow \beta) = 1$, luego $\Gamma \vDash \alpha \rightarrow \beta$.

Ahora probemos lo recíproco. Supongamos que $\Gamma \vDash \alpha \rightarrow \beta$. Entonces sea v una valoración que satisface $\Gamma \cup \{\alpha\}$. Esto, por la definición 2.3, implica que v satisface Γ, por lo que —definición 2.4— v satisface $\alpha \rightarrow \beta$, con lo que $v(\alpha) = 0$ ó $v(\beta) = 1$. Pero, por satisfacer v a $\Gamma \cup \{\alpha\}$, tenemos que $v(\alpha) = 1$, por lo que debe darse que $v(\beta) = 1$. Pero sólo hemos supuesto que v satisface $\Gamma \cup \{\alpha\}$, de lo que concluimos $\Gamma, \alpha \vDash \beta$.

Pasemos a (2.2). Supongamos que se da $\Gamma, \alpha \vDash \bot$. Esto, por la definición 2.7, significa que no existe ninguna valoración que satisfaga $\Gamma \cup \{\alpha\}$. Entonces, sea v una valoración tal que $v \vDash \Gamma$. Por hipótesis, sabemos que $v(\alpha) = 0$, de donde, por evaluación del negador, $v(\neg\alpha) = 1$. Entonces, como este razonamiento vale para toda v que satisfaga Γ tenemos $\Gamma \vDash \neg\alpha$.

Ahora, supongamos que se cumple $\Gamma \vDash \neg\alpha$. Sea v una valoración que satisface Γ. Entonces, $v(\neg\alpha) = 1$, de donde, por evaluación del negador, $v(\alpha) = 0$. Como

el razonamiento anterior vale para toda valoración que satisfaga Γ, tenemos que no es posible que ninguna valoración satisfaga simultáneamente Γ y α, de donde $\Gamma, \alpha \vDash \bot$.

Finalmente, para probar (2.3), comenzamos suponiendo $\Gamma, \neg\alpha \vDash \bot$. Sea v una valoración que satisface Γ. Entonces no puede ocurrir que $v(\neg\alpha) = 1$. Por tanto, $v(\neg\alpha) = 0$, y por evaluación del negador $v(\alpha) = 1$, de modo que $\Gamma \vDash \alpha$.

Igualmente, supongamos $\Gamma \vDash \alpha$. Sea v tal que $v \vDash \Gamma$. Entonces, tenemos que $v(\alpha) = 1$. De modo que toda valoración que satisfaga Γ satisface α. Por evaluación del negador, tenemos que entonces no existe ninguna valoración que satisfaga simultáneamente $\Gamma \cup \{\neg\alpha\}$, de donde $\Gamma, \neg\alpha \vDash \bot$. ■

En el teorema anterior, la relación expresada en (2.1) se conoce como *teorema de la deducción*, y cumple una función muy importante en los *cálculos lógicos*. Nosotros la hemos presentado junto a (2.2) y (2.3) porque, a efectos de los desarrollos posteriores, las tres relaciones tendrán un papel muy semejante.

Corolario 2.10 *Para cualesquiera $\alpha, \beta \in \mathcal{L}_p$ se verifica:*

$$\alpha \vDash \beta \quad syss \quad \vDash \alpha \to \beta \tag{2.4}$$

$$\alpha \vDash \bot \quad syss \quad \vDash \neg\alpha \tag{2.5}$$

$$\neg\alpha \vDash \bot \quad syss \quad \vDash \alpha \tag{2.6}$$

Demostración:

Las relaciones (2.4), (2.5) y (2.6) son, respectivamente, casos particulares de (2.1), (2.2) y (2.3), en que $\Gamma = \emptyset$. Las demostraciones son, pues, iguales, considerando que, por la definición 2.5, Γ es siempre universalmente válido. ■

Definición 2.11 (Equivalencia) *Dos fórmulas $\alpha, \gamma \in \mathcal{L}_p$ son equivalentes syss para toda valoración v,*

$$v(\alpha) = v(\gamma)$$

Corolario 2.12 *Dos fórmulas $\alpha, \gamma \in \mathcal{L}_p$ son equivalentes syss*

$$\vDash \alpha \leftrightarrow \gamma$$

Demostración:

Este corolario es una consecuencia inmediata de las definiciones 2.11 y 2.2. Si α y γ son equivalentes, entonces para toda valoración v, $v(\alpha) = v(\gamma)$, lo cual, por evaluación de la doble implicación, resulta $v(\alpha \leftrightarrow \gamma) = 1$. Pero si para toda v se verifica que $v(\alpha \leftrightarrow \gamma) = 1$, entonces $\vDash \alpha \leftrightarrow \gamma$. Del mismo modo, si se verifica esto último, es que para toda v se cumple que $v(\alpha \leftrightarrow \gamma) = 1$, y por evaluación de la doble implicación, $v(\alpha) = v(\gamma)$, luego α y γ son equivalentes. ■

Teorema 2.13 *Para cualesquiera $\alpha, \beta, \gamma \in \mathcal{L}_p$ se verifican las siguientes equivalencias:*

$$\vDash \neg\neg\alpha \leftrightarrow \alpha \tag{2.7}$$
$$\vDash (\alpha \vee \beta) \leftrightarrow (\beta \vee \alpha) \tag{2.8}$$
$$\vDash (\alpha \wedge \beta) \leftrightarrow (\beta \wedge \alpha) \tag{2.9}$$
$$\vDash \neg(\alpha \vee \beta) \leftrightarrow (\neg\alpha \wedge \neg\beta) \tag{2.10}$$
$$\vDash \neg(\alpha \wedge \beta) \leftrightarrow (\neg\alpha \vee \neg\beta) \tag{2.11}$$
$$\vDash (\alpha \rightarrow \beta) \leftrightarrow (\neg\alpha \vee \beta) \tag{2.12}$$
$$\vDash (\alpha \rightarrow \beta) \leftrightarrow \neg(\alpha \wedge \neg\beta) \tag{2.13}$$
$$\vDash \neg(\alpha \rightarrow \beta) \leftrightarrow (\alpha \wedge \neg\beta) \tag{2.14}$$
$$\vDash (\alpha \leftrightarrow \beta) \leftrightarrow (\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta) \tag{2.15}$$
$$\vDash \neg(\alpha \leftrightarrow \beta) \leftrightarrow (\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \beta) \tag{2.16}$$
$$\vDash (\alpha \wedge (\beta \vee \gamma)) \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \tag{2.17}$$
$$\vDash (\alpha \vee (\beta \wedge \gamma)) \leftrightarrow ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \tag{2.18}$$
$$\vDash (\alpha \vee (\beta \vee \gamma)) \leftrightarrow ((\alpha \vee \beta) \vee \gamma) \tag{2.19}$$
$$\vDash (\alpha \wedge (\beta \wedge \gamma)) \leftrightarrow ((\alpha \wedge \beta) \wedge \gamma) \tag{2.20}$$
$$\vDash ((\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)) \leftrightarrow \alpha \tag{2.21}$$
$$\vDash ((\alpha \vee \beta) \wedge (\alpha \vee \neg\beta)) \leftrightarrow \alpha \tag{2.22}$$

Demostración:

Sea v una valoración booleana cualquiera. Para probar que cada una de las fórmulas que aparecen en el teorema 2.13 es una equivalencia, basta con probar que v satisface una de las dos subfórmulas syss satisface la otra, pues esto implica que v no satisface a cada una de las subfórmulas syss no satisface a la otra, y a su vez que el valor de verdad de ambas subfórmulas es el mismo para toda valoración posible. Por ello, para cada equivalencia, supondremos que el valor de verdad de la subfórmula de la izquierda es 1, y tendremos que probar que esto sólo ocurre syss el valor de verdad de la subfórmula derecha es igualmente 1. Todos los razonamientos se realizan por evaluación de operadores lógicos. Por brevedad,se omiten las indicaciones.

Equivalencia (2.7): $v(\neg\neg\alpha) = 1$

$$\begin{aligned} &\text{syss} \quad v(\neg\alpha) = 0 \\ &\text{syss} \quad v(\alpha) = 1 \end{aligned}$$

Equivalencia (2.8): $v(\alpha \vee \beta) = 1$

$$\begin{aligned} &\text{syss} \quad v(\alpha) = 1 \ \text{ ó } \ v(\beta) = 1 \\ &\text{syss} \quad v(\beta) = 1 \ \text{ ó } \ v(\alpha) = 1 \\ &\text{syss} \quad v(\beta \vee \alpha) = 1 \end{aligned}$$

Equivalencia (2.9): $v(\alpha \wedge \beta) = 1$

$$\begin{aligned} &\text{syss} \quad v(\alpha) = 1 \ \text{ y } \ v(\beta) = 1 \\ &\text{syss} \quad v(\beta) = 1 \ \text{ y } \ v(\alpha) = 1 \\ &\text{syss} \quad v(\beta \wedge \alpha) = 1 \end{aligned}$$

Equivalencia (2.10): $v(\neg(\alpha \vee \beta)) = 1$

$$\begin{aligned} &\text{syss} \quad v(\alpha \vee \beta) = 0 \\ &\text{syss} \quad v(\alpha) = 0 \ \text{ y } \ v(\beta) = 0 \\ &\text{syss} \quad v(\neg\alpha) = 1 \ \text{ y } \ v(\neg\beta) = 1 \\ &\text{syss} \quad v(\neg\alpha \wedge \neg\beta) = 1 \end{aligned}$$

Equivalencia (2.11): $v(\neg(\alpha \wedge \beta)) = 1$

$$\begin{aligned}
&\text{syss} \quad v(\alpha \wedge \beta) = 0 \\
&\text{syss} \quad v(\alpha) = 0 \ \text{ ó } \ v(\beta) = 0 \\
&\text{syss} \quad v(\neg\alpha) = 1 \ \text{ ó } \ v(\neg\beta) = 1 \\
&\text{syss} \quad v(\neg\alpha \vee \neg\beta) = 1
\end{aligned}$$

Equivalencia (2.12): $v(\alpha \rightarrow \beta) = 1$

$$\begin{aligned}
&\text{syss} \quad v(\alpha) = 0 \ \text{ ó } \ v(\beta) = 1 \\
&\text{syss} \quad v(\neg\alpha) = 1 \ \text{ ó } \ v(\beta) = 1 \\
&\text{syss} \quad v(\neg\alpha \vee \beta) = 1
\end{aligned}$$

Equivalencia (2.13): $v(\alpha \rightarrow \beta) = 1$

$$\begin{aligned}
&\text{syss} \quad v(\alpha) = 0 \ \text{ ó } \ v(\beta) = 1 \\
&\text{syss} \quad v(\alpha) = 0 \ \text{ ó } \ v(\neg\beta) = 0 \\
&\text{syss} \quad v(\alpha \wedge \neg\beta) = 0 \\
&\text{syss} \quad v(\neg(\alpha \wedge \neg\beta)) = 1
\end{aligned}$$

Equivalencia (2.14): $v(\neg(\alpha \rightarrow \beta)) = 1$

$$\begin{aligned}
&\text{syss} \quad v(\alpha \rightarrow \beta) = 0 \\
&\text{syss} \quad v(\alpha) = 1 \ \text{ y } \ v(\beta) = 0 \\
&\text{syss} \quad v(\alpha) = 1 \ \text{ y } \ v(\neg\beta) = 1 \\
&\text{syss} \quad v(\alpha \wedge \neg\beta) = 1
\end{aligned}$$

Equivalencia (2.15): $v(\alpha \leftrightarrow \beta) = 1$

$$\begin{aligned}
&\text{syss} \quad v(\alpha) = v(\beta) \\
&\text{syss} \quad (v(\alpha) = 1 \text{ y } v(\beta) = 1) \ \text{ ó } \ (v(\alpha) = 0 \text{ y } v(\beta) = 0) \\
&\text{syss} \quad (v(\alpha) = 1 \text{ y } v(\beta) = 1) \ \text{ ó } \ (v(\neg\alpha) = 1 \text{ y } v(\neg\beta) = 1) \\
&\text{syss} \quad v(\alpha \wedge \beta) = 1 \ \text{ ó } \ v(\neg\alpha \wedge \neg\beta) = 1 \\
&\text{syss} \quad v((\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)) = 1
\end{aligned}$$

Equivalencia (2.16): $v(\neg(\alpha \leftrightarrow \beta)) = 1$

$$
\begin{array}{ll}
\text{syss} & v(\alpha \leftrightarrow \beta) = 0 \\
\text{syss} & v(\alpha) \neq v(\beta) \\
\text{syss} & (v(\alpha) = 1 \text{ y } v(\beta) = 0) \text{ ó } (v(\alpha) = 0 \text{ y } v(\beta) = 1) \\
\text{syss} & (v(\alpha) = 1 \text{ y } v(\neg\beta) = 1) \text{ ó } (v(\neg\alpha) = 1 \text{ y } v(\beta) = 1) \\
\text{syss} & v(\alpha \wedge \neg\beta) = 1 \text{ ó } v(\neg\alpha \wedge \beta) = 1 \\
\text{syss} & v((\alpha \wedge \neg\beta) \vee (\neg\alpha \wedge \beta)) = 1
\end{array}
$$

Equivalencia (2.17): $v(\alpha \wedge (\beta \vee \gamma)) = 1$

$$
\begin{array}{ll}
\text{syss} & v(\alpha) = 1 \text{ y } v(\beta \vee \gamma) = 1 \\
\text{syss} & v(\alpha) = 1 \text{ y } (v(\beta) = 1 \text{ ó } v(\gamma) = 1) \\
\text{syss} & (v(\alpha) = 1 \text{ y } v(\beta) = 1) \text{ ó } (v(\alpha) = 1 \text{ y } v(\gamma) = 1) \\
\text{syss} & v(\alpha \wedge \beta) = 1 \text{ ó } v(\alpha \wedge \gamma) = 1 \\
\text{syss} & v((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) = 1
\end{array}
$$

Equivalencia (2.18): $v(\alpha \vee (\beta \wedge \gamma)) = 1$

$$
\begin{array}{ll}
\text{syss} & v(\alpha) = 1 \text{ ó } v(\beta \wedge \gamma) = 1 \\
\text{syss} & v(\alpha) = 1 \text{ ó } (v(\beta) = 1 \text{ y } v(\gamma) = 1) \\
\text{syss} & (v(\alpha) = 1 \text{ ó } v(\beta) = 1) \text{ y } (v(\alpha) = 1 \text{ ó } v(\gamma) = 1) \\
\text{syss} & v(\alpha \vee \beta) = 1 \text{ y } v(\alpha \vee \gamma) = 1 \\
\text{syss} & v((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) = 1
\end{array}
$$

Equivalencia (2.19): $v(\alpha \vee (\beta \vee \gamma)) = 1$

$$
\begin{array}{ll}
\text{syss} & v(\alpha) = 1 \text{ ó } v(\beta \vee \gamma) = 1 \\
\text{syss} & v(\alpha) = 1 \text{ ó } (v(\beta) = 1 \text{ ó } v(\gamma) = 1) \\
\text{syss} & (v(\alpha) = 1 \text{ ó } v(\beta) = 1) \text{ ó } v(\gamma) = 1 \\
\text{syss} & v(\alpha \vee \beta) = 1 \text{ ó } v(\gamma) = 1 \\
\text{syss} & v((\alpha \vee \beta) \vee \gamma) = 1
\end{array}
$$

Equivalencia (2.20): $v(\alpha \wedge (\beta \wedge \gamma)) = 1$

$$\begin{aligned}
&\text{syss} \quad v(\alpha) = 1 \text{ y } v(\beta \wedge \gamma) = 1 \\
&\text{syss} \quad v(\alpha) = 1 \text{ y } (v(\beta) = 1 \text{ y } v(\gamma) = 1) \\
&\text{syss} \quad (v(\alpha) = 1 \text{ y } v(\beta) = 1) \text{ y } v(\gamma) = 1 \\
&\text{syss} \quad v(\alpha \wedge \beta) = 1 \text{ y } v(\gamma) = 1 \\
&\text{syss} \quad v((\alpha \wedge \beta) \wedge \gamma) = 1
\end{aligned}$$

Equivalencia (2.21): $v((\alpha \wedge \beta) \vee (\alpha \wedge \neg\beta)) = 1$

$$\begin{aligned}
&\text{syss} \quad (v(\alpha) = 1 \text{ y } v(\beta) = 1) \;\; \text{ó} \;\; (v(\alpha) = 1 \text{ y } v(\neg\beta) = 1) \\
&\text{syss} \quad (v(\alpha) = 1 \text{ y } v(\beta) = 1) \;\; \text{ó} \;\; (v(\alpha) = 1 \text{ y } v(\beta) = 0) \\
&\text{syss} \quad v(\alpha) = 1
\end{aligned}$$

Equivalencia (2.22): $v((\alpha \vee \beta) \wedge (\alpha \vee \neg\beta)) = 1$

$$\begin{aligned}
&\text{syss} \quad (v(\alpha) = 1 \text{ ó } v(\beta) = 1) \;\; \text{y} \;\; (v(\alpha) = 1 \text{ ó } v(\neg\beta) = 1) \\
&\text{syss} \quad (v(\alpha) = 1 \text{ ó } v(\beta) = 1) \;\; \text{y} \;\; (v(\alpha) = 1 \text{ ó } v(\beta) = 0) \\
&\text{syss} \quad v(\alpha) = 1
\end{aligned}$$

■

Teorema 2.14 (Teorema de intercambio) *Dadas las fórmulas $\gamma_\alpha \in \mathcal{L}_p$, de la que α es una subfórmula, y γ_β que difiere de γ_α sólo en que una ocurrencia de α se ha reemplazado por β, se cumple que*

$$\textit{Si} \quad \vDash \alpha \leftrightarrow \beta, \quad \textit{entonces} \quad \vDash \gamma_\alpha \leftrightarrow \gamma_\beta$$

Demostración:

Procedemos por inducción sobre el grado lógico de γ_α. En el caso base, γ_α tiene grado 0, por lo que se trata de una variable proposicional, de modo que γ_α es α, ya que no contiene otra subfórmula. Entonces, γ_β será β, igualmente, por tanto, si $\vDash \alpha \leftrightarrow \beta$, resulta trivial que $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

Como hipótesis de inducción, consideremos que el teorema se cumple para fórmulas de grado lógico menor o igual a n. Sea entonces γ_α de grado lógico $n+1$, α una subfórmula suya, y β otra subfórmula. Supongamos que $\vDash \alpha \leftrightarrow \beta$. Puede ocurrir una de las siguiente cosas:

1. Que $\alpha = \gamma_\alpha$. En este caso, resulta trivial que $\beta = \gamma_\beta$, así como que $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

2. Que $\alpha \neq \gamma_\alpha$, por lo que α está contenida en γ_α sin ser ella misma. Entonces, según la forma de γ_α puede ocurrir:

   a) γ_α es $\neg\eta_\alpha$, y η_β resulta de η_α tras intercambiar la ocurrencia de α por β. Por hipótesis de inducción, $\vDash \eta_\alpha \leftrightarrow \eta_\beta$. Entonces, para cualquier v se cumple $v(\eta_\alpha) = v(\eta_\beta)$, y por evaluación del negador $v(\neg\eta_\alpha) = v(\neg\eta_\beta)$, pero como γ_β es $\neg\eta_\beta$, $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

   b) γ_α es $\delta_\alpha \vee \eta_\alpha$, y δ_β y η_β se obtienen desde δ_α y η_α reemplazando en alguna de las dos una ocurrencia de α por β. Por hipótesis de inducción, $\vDash \delta_\alpha \leftrightarrow \delta_\beta$ y $\vDash \eta_\alpha \leftrightarrow \eta_\beta$. Para cualquier valoración v se cumple que

   $$v(\delta_\alpha) = v(\delta_\beta) \quad \text{y} \quad v(\eta_\alpha) = v(\eta_\beta)$$

   por lo que también

   $$v(\delta_\alpha \vee \eta_\alpha) \quad = \quad v(\delta_\beta \vee \eta_\beta)$$

   pero como γ_β es $\delta_\beta \vee \eta_\beta$, entonces $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

   c) γ_α es $\delta_\alpha \wedge \eta_\alpha$, y δ_β y η_β se obtienen igual que antes. Por hipótesis de inducción, $\vDash \delta_\alpha \leftrightarrow \delta_\beta$ y $\vDash \eta_\alpha \leftrightarrow \eta_\beta$. Para cualquier valoración v se cumple que

   $$v(\delta_\alpha) = v(\delta_\beta) \quad \text{y} \quad v(\eta_\alpha) = v(\eta_\beta)$$

   por lo que también

   $$v(\delta_\alpha \wedge \eta_\alpha) \quad = \quad v(\delta_\beta \wedge \eta_\beta)$$

   pero como γ_β es $\delta_\beta \wedge \eta_\beta$, entonces $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

d) γ_α es $\delta_\alpha \rightarrow \eta_\alpha$, y δ_β y η_β se obtienen igual que antes. Por hipótesis de inducción, $\vDash \delta_\alpha \leftrightarrow \delta_\beta$ y $\vDash \eta_\alpha \leftrightarrow \eta_\beta$. Para cualquier valoración v se cumple que

$$v(\delta_\alpha) = v(\delta_\beta) \quad \text{y} \quad v(\eta_\alpha) = v(\eta_\beta)$$

por lo que también

$$v(\delta_\alpha \rightarrow \eta_\alpha) \quad = \quad v(\delta_\beta \rightarrow \eta_\beta)$$

pero como γ_β es $\delta_\beta \rightarrow \eta_\beta$, entonces $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

e) γ_α es $\delta_\alpha \leftrightarrow \eta_\alpha$, y δ_β y η_β se obtienen igual que antes. Por hipótesis de inducción, $\vDash \delta_\alpha \leftrightarrow \delta_\beta$ y $\vDash \eta_\alpha \leftrightarrow \eta_\beta$. Para cualquier valoración v se cumple que

$$v(\delta_\alpha) = v(\delta_\beta) \quad \text{y} \quad v(\eta_\alpha) = v(\eta_\beta)$$

por lo que también

$$v(\delta_\alpha \leftrightarrow \eta_\alpha) \quad = \quad v(\delta_\beta \leftrightarrow \eta_\beta)$$

pero como γ_β es $\delta_\beta \leftrightarrow \eta_\beta$, entonces $\vDash \gamma_\alpha \leftrightarrow \gamma_\beta$.

■

Definición 2.15 (Conjunción elemental, disyunción elemental) *Una fórmula de $\mathcal{L}_p$ es una* conjunción elemental *syss es un literal o una conjunción de literales. Igualmente, decimos que es una* disyunción elemental *syss o bien es un literal o una disyunción de literales.*

Definición 2.16 (Formas normales) *Decimos que una fórmula de $\mathcal{L}_p$ está en* forma normal conjuntiva *syss es una conjunción de disyunciones elementales. Igualmente, decimos que está en* forma normal disyuntiva *syss es una disyunción de conjunciones elementales.*

Definición 2.17 (FNC y FND de una fórmula) *Dada una fórmula $\alpha \in \mathcal{L}_p$, representamos con $FNC(\alpha)$ una fórmula —no necesariamente única— que esté en forma normal conjuntiva y además se cumpla*

$$\vDash \alpha \leftrightarrow FNC(\alpha)$$

Del mismo modo, representamos con $FND(\alpha)$ una fórmula —no necesariamente única— que esté en forma normal disyuntiva y además se cumpla

$$\vDash \alpha \leftrightarrow FND(\alpha)$$

Teorema 2.18 (Construcción de formas normales) *Para cada fórmula $\alpha \in \mathcal{L}_p$ podemos encontrar $FNC(\alpha)$ y $FND(\alpha)$.*

Demostración:

Para la demostración, presentaremos los procedimientos que nos permiten obtener, a partir de cualquier $\alpha \in \mathcal{L}_p$, las fórmulas $FNC(\alpha)$ y $FND(\alpha)$, mediante tres pasos.

1. **Eliminación de implicaciones**. En ambos casos se reemplaza en α cada subfórmula de la forma $\beta \rightarrow \gamma$ por $\neg\beta \vee \gamma$, y cada subfórmula de la forma $\beta \leftrightarrow \gamma$ por $(\alpha \wedge \beta) \vee (\neg\alpha \wedge \neg\beta)$. El resultado, al que llamamos α_1 será una fórmula sin implicaciones ni dobles implicaciones. Por tanto, α_1 sólo contiene negaciones, conjunciones y disyunciones.

2. **Interiorización de negadores**. Para cada subfórmula de α_1 de la forma $\neg\beta$ se procede:

   a) Si β es una variable proposicional, se deja sin reemplazar,

   b) Si β es $\neg\gamma$, se reemplaza $\neg\beta$ por γ,

   c) Si β es $\gamma \wedge \eta$, se reemplaza $\neg\beta$ por $\neg\gamma \vee \neg\eta$,

   d) Si β es $\gamma \vee \eta$, se reemplaza $\neg\beta$ por $\neg\gamma \wedge \neg\eta$.

   Como en α_1 las negaciones sólo podían afectar a variables proposicionales, a otras negaciones, o bien a conjunciones o disyunciones, tras aplicar estas transformaciones todas las veces posibles —es decir, hasta que cada subfórmula de tipo $\neg\beta$ que quede sea un literal—, el resultado, al que llamamos α_2, será una fórmula proposicional compuesta sólo de variables proposicionales, conjunciones, disyunciones y negaciones que sólo afectan a variables proposicionales.

3. **Aplicación de la propiedad distributiva**. A continuación se hace, partiendo de α_2,

   a) Para construir $FNC(\alpha)$, mientras haya subfórmulas de la forma $\beta \vee (\gamma \wedge \delta)$ o bien $(\gamma \wedge \delta) \vee \beta$ —es decir, con un conjuntor bajo el alcance de un disyuntor—, se reemplazan por $(\beta \vee \gamma) \wedge (\beta \vee \delta)$,

   b) Para construir $FND(\alpha)$, mientras haya subfórmulas de la forma $\beta \wedge (\gamma \vee \delta)$ o bien $(\gamma \vee \delta) \wedge \beta$ —es decir, con un disyuntor bajo el alcance de un conjuntor—, se reemplazan por $(\beta \wedge \gamma) \vee (\beta \wedge \delta)$.

   Tras acabar este último paso, para $FNC(\alpha)$ no habrá ningún conjuntor bajo el alcance de un disyuntor, por lo que la fórmula resultante será una conjunción de disyunciones de literales, ya que las únicas negaciones que pueden ocurrir son de fórmulas atómicas. Por tanto, se trata de una conjunción de disyunciones elementales. Por la misma razón, para $FND(\alpha)$ se llega a una disyunción de conjunciones elementales. Por tanto, ya se cumple uno de los requisitos para poder considerar, a las fórmulas resultantes, las formas normales —conjuntiva y disyuntiva— de α, según la definición 2.17. Además, si repasamos los reemplazos que sucesivamente se han ido efectuando en α, todos se corresponden con equivalencias de las que aparecen en el teorema 2.13, de modo que por el teorema 2.14, tenemos que las formas normales resultantes son equivalentes a α, de forma que —por la definición 2.17— $FNC(\alpha)$ es la forma normal conjuntiva de α, así como $FND(\alpha)$ su forma normal disyuntiva.

■

Teorema 2.19 *Una fórmula $\alpha \in \mathcal{L}_p$ es universalmente válida syss en cada disyunción elemental de $FNC(\alpha)$ ocurren dos literales complementarios.*

Demostración:

Dada α tal que $\vDash \alpha$, por el teorema 2.18, $\vDash FNC(\alpha)$. Pero $FNC(\alpha)$ tendrá la forma

$$(\delta_{1_1} \vee \ldots \vee \delta_{1_{j_1}}) \wedge \ldots \wedge (\delta_{n_1} \vee \ldots \vee \delta_{n_{j_n}})$$

$n \leq 1$, siendo δ_{k_l} un literal, $1 \leq k \leq n$, $1 \leq l \leq j_k$. Entonces, para toda valoración posible v, tenemos que $v(\delta_{k_1} \vee \ldots \vee \delta_{k_{j_k}}) = 1$, para cada k. Pero esto sólo es posible si en cada disyunción elemental existen dos literales complementarios, ya que en otro caso es posible una valoración que no satisfaga alguna de las disyunciones elementales, pues basta sólo con que el valor de verdad de cada uno de los literales de tal disyunción sea 0 bajo v.

Recíprocamente, si en cada disyunción elemental de $FNC(\alpha)$ existen dos literales complementarios, entonces para toda valoración v, ésta satisface cada disyunción elemental y, por evaluación del conjuntor, también satisface $FNC(\alpha)$, es decir, $\vDash FNC(\alpha)$, pero como $\alpha \leftrightarrow FNC(\alpha)$, $\vDash \alpha$. ■

Teorema 2.20 *Una fórmula $\alpha \in \mathcal{L}_p$ es no satisfactible syss en cada conjunción elemental de $FND(\alpha)$ ocurren dos literales complementarios.*

Demostración:

Dada α tal que $\alpha \vDash \bot$, por el teorema 2.18, $FND(\alpha) \vDash \bot$. Pero $FND(\alpha)$ tendrá la forma

$$(\delta_{1_1} \wedge \ldots \wedge \delta_{1_{j_1}}) \vee \ldots \vee (\delta_{n_1} \wedge \ldots \wedge \delta_{n_{j_n}})$$

$n \leq 1$, siendo δ_{k_l} un literal, $1 \leq k \leq n$, $1 \leq l \leq j_k$. Entonces, para toda valoración posible v, tenemos que $v(\delta_{k_1} \wedge \ldots \wedge \delta_{k_{j_k}}) = 0$, para cada k. Pero esto sólo es posible si en cada conjunción elemental existen dos literales complementarios, ya que en otro caso es posible una valoración que satisfaga alguna de las conjunciones elementales, pues basta sólo con que el valor de verdad de cada uno de los literales de tal conjunción sea 1 bajo v.

Recíprocamente, si en cada conjunción elemental de $FND(\alpha)$ existen dos literales complementarios, entonces para toda valoración v, ésta no satisface ninguna conjunción elemental y, por evaluación del disyuntor, tampoco satisface $FND(\alpha)$, es decir, $FNC(\alpha) \vDash \bot$, pero como $\alpha \leftrightarrow FNC(\alpha)$, $\alpha \vDash \bot$. ■

2.2. Definiciones de *problema abductivo* y *solución abductiva*

En esta sección introducimos las definiciones formales de *problema abductivo* y de *solución abductiva*, tal como serán empleadas durante el resto de este trabajo. Seguimos, en parte, a M. Cialdea y F. Pirri [CP93] y Aliseda [Ali97].

Usamos la notación $\langle A, B\rangle$ para referirnos al par ordenado de elementos A y B, como es habitual. Por tanto, con $\langle \Theta, \phi\rangle$ representamos un par ordenado cuyo primer elemento es el conjunto $\Theta \subset \mathcal{L}_p$ y el segundo $\phi \in \mathcal{L}_p$.

Definición 2.21 (Problema abductivo) *Decimos que $\langle \Theta, \phi\rangle$ es un problema abductivo syss se verifican las dos siguientes condiciones*

$$\Theta \not\models \phi \tag{2.23}$$

$$\Theta \not\models \neg\phi \tag{2.24}$$

Teorema 2.22 *Para cualesquiera $\Theta \subset \mathcal{L}_p$ y $\phi \in \mathcal{L}_p$, si $\langle \Theta, \phi\rangle$ es un problema abductivo, entonces se cumple que*

1. *Θ es satisfactible.*
2. *ϕ es contingente.*

Demostración:

Si Θ no fuera satisfactible, tendríamos que $\Theta \models \bot$, y puesto que ninguna valoración satisface Θ, tampoco hay ninguna valoración que satisfaga $\Theta \cup \{\phi\}$, por lo que $\Theta, \phi \models \bot$. Pero esto, por el teorema 2.9 equivale a $\Theta \models \neg\phi$, y por la definición 2.21 tenemos que $\langle \Theta, \phi\rangle$ no puede ser un problema abductivo. Luego, si $\langle \Theta, \phi\rangle$ es un problema abductivo, Θ debe ser satisfactible.

Del mismo modo, si ϕ no fuera contingente, entonces sería al caso de que ϕ es universalmente válida o bien no satisfactible. Si ϕ es universalmente válida se cumple $\Theta \models \phi$, pues toda valoración satisface ϕ. Igualmente, si ϕ es no

satisfactible, como ninguna valoración la satisface, tenemos que toda valoración satisface $\neg\phi$, por evaluación del negador, y por tanto $\Theta \models \neg\phi$. Entonces, si ϕ no es contingente, se cumple $\Theta \models \phi$ o bien $\Theta \models \neg\phi$. Pero en ambos casos $\langle\Theta, \phi\rangle$ no sería un problema abductivo, por la definición 2.21. Por tanto, si $\langle\Theta, \phi\rangle$ es un problema abductivo, ϕ debe ser contingente. ■

Definición 2.23 (Solución abductiva plana) *Sea $\langle\Theta, \phi\rangle$ un problema abductivo. Decimos que α es una* solución (abductiva) plana *a dicho problema, syss*

$$\Theta, \alpha \models \phi \tag{2.25}$$

A la relación (2.25) *la llamamos* requisito (abductivo) fundamental.

Definición 2.24 (Solución consistente) *Dado el problema abductivo $\langle\Theta, \phi\rangle$, decimos que α es una* solución (abductiva) consistente *a tal problema, syss se cumple*

1. *α es una solución abductiva plana a $\langle\Theta, \phi\rangle$*
2. *Se verifica*

$$\Theta, \alpha \not\models \bot \tag{2.26}$$

A la relación (2.26) *la llamamos* requisito (abductivo) de consistencia.

Definición 2.25 (Solución explicativa) *Dado el problema abductivo $\langle\Theta, \phi\rangle$, decimos que α es una* solución (abductiva) explicativa *a tal problema, syss se cumple*

1. *α es una solución abductiva consistente a $\langle\Theta, \phi\rangle$*
2. *Se verifica*

$$\alpha \not\models \phi \tag{2.27}$$

A la relación (2.27) *la llamamos* requisito (abductivo) explicativo.

En un problema abductivo $\langle \Theta, \phi \rangle$, el lugar que ocupa Θ es el de la teoría[3] base, y ϕ el de la observación que ha de explicarse. Asimismo, cada solución abductiva α será una explicación de ϕ en Θ. En lo sucesivo, emplearemos frecuentemente estas mismas letras griegas para referirnos tanto a problemas abductivos como a sus soluciones. Igualmente, cuando enunciemos teoremas o definiciones que tengan aplicaciones abductivas, usaremos frecuentemente estas mismas letras para que resulte más fácil apreciar su uso abductivo. Esto ocurre, por ejemplo, en el siguiente teorema.

Teorema 2.26 *Para cualquier conjunto* $\Theta \subset \mathcal{L}_p$*, si* $\Theta = \{\theta_1, \ldots, \theta_n\}$*,* $n \geq 1$*, entonces las valoraciones que satisfacen* Θ *son las mismas que satisfacen la fórmula* $\theta_1 \wedge \ldots \wedge \theta_n$*.*

Demostración:

Probaremos primero que toda valoración que satisface Θ satisface $\theta_1 \wedge \ldots \wedge \theta_n$ y a continuación lo recíproco. Sea v tal que $v \vDash \Theta$. Entonces, por la definición 2.3, $v(\theta_i) = 1$ para todos los valores de i entre 1 y n. Por tanto, por evaluación del conjuntor, $v(\theta_1 \wedge \ldots \wedge \theta_n) = 1$.

Del mismo modo, sea una valoración v tal que $v(\theta_1 \wedge \ldots \wedge \theta_n) = 1$. Entonces, por evaluación del conjuntor, $v(\theta_i) = 1$ para todos los valores de i entre 1 y n. Por tanto, por la definición 2.3, v satisface Θ. ■

Por el teorema 2.26 tenemos que las valoraciones que satisfacen la teoría $\Theta = \{\theta_1, \ldots, \theta_n\}$ son exactamente las mismas que las que satisfacen $\theta_1 \wedge \ldots \wedge \theta_n$. Dicho de otra manera, Θ y $\theta_1 \wedge \ldots \wedge \theta_n$ pueden ser tratados como equivalentes. Por ello, en ocasiones, dado un problema abductivo $\langle \Theta, \phi \rangle$, trataremos —de forma

[3]En algunos contextos —como en el estudio de los sistemas axiomáticos—, es frecuente considerar que cierto conjunto de fórmulas Γ es una *teoría* cuando se verifica —habitualmente, entre otros requisitos— que, para toda fórmula α, si $\Gamma \vDash \alpha$, entonces $\alpha \in \Gamma$. En tal caso, se dice de Γ que es un conjunto de fórmulas *cerrado* bajo la relación de consecuencia lógica. Entonces, en $\mathcal{L}_p$, con la relación de consecuencia lógica que hemos definido, Γ sería siempre un conjunto infinito de fórmulas. Por tanto, en nuestro estudio —tal como es frecuente en otras aproximaciones al razonamiento abductivo—, consideramos que una teoría puede ser cualquier conjunto de fórmulas de $\mathcal{L}_p$.

explícita o implícita— a Θ como una fórmula. En tales casos, presupondremos que nos referimos a la fórmula formada por la conjunción de todas las θ_i, $1 \leq i \leq n$. En ciertos casos, incluso hablaremos del problema abductivo $\langle \Theta, \phi \rangle$ tal que $\Theta, \phi \in \mathcal{L}_p$, en este mismo sentido.

Teorema 2.27 *Para cualesquiera $\Theta \subset \mathcal{L}_p$ y $\phi \in \mathcal{L}_p$, si $\langle \Theta, \phi \rangle$ es un problema abductivo con al menos una solución explicativa, entonces se cumple que*

1. *Θ es contingente.*
2. *ϕ es contingente.*

Demostración:

Sea $\langle \Theta, \phi \rangle$ un problema abductivo, y α una solución explicativa suya. Por el teorema 2.22 tenemos ya probado que ϕ debe ser contingente, así como que Θ debe ser satisfactible. Por tanto, sólo nos queda probar que Θ no es universalmente válida, ya que entonces, puesto que es satisfactible, estará probada su contingencia. Dado que α es una solución explicativa, por la definición 2.25 tenemos que se cumplen

$$\alpha \not\models \phi \tag{2.28}$$

$$\Theta, \alpha \models \phi \tag{2.29}$$

Ahora bien, supongamos que Θ fuera universalmente válida. Por (2.28) tenemos que existe una valoración v tal que $v(\alpha) = 1$ y $v(\phi) = 0$. Pero puesto que si Θ es universalmente válida v satisface todas sus fórmulas y, siendo $v(\alpha) = 1$, tenemos que $v \models \Theta \cup \{\alpha\}$, por (2.29) llegamos a $v(\phi) = 1$. Entonces encontramos una contradicción, ya que el valor de verdad de ϕ bajo v debe ser único. Por tanto, negamos nuestra última hipótesis, y concluimos que Θ no es universalmente válida.

De forma que tanto Θ como ϕ deben ser contingentes. ■

Respecto a los problemas abductivos que no tienen soluciones explicativas, no puede probarse que Θ deba ser contingente. De hecho, puede verificarse que,

según la definición 2.21, $\langle\{p \vee \neg p\}, q\rangle$ es un problema abductivo, así como que q es una solución consistente a tal problema, según la definición 2.24, pero tanto el problema como la solución son sumamente triviales; el problema, por ser la teoría universalmente válida; y la solución, por ser igual a la observación.

Además, las soluciones no explicativas resultan poco interesantes, pues suponen asumir como explicación algo independiente de la teoría, cuando la principal motivación para el razonamiento abductivo es precisamente la contraria. Por ello, en este trabajo nos centraremos en los problemas abductivos que cuenten con soluciones explicativas. Serán también éstas las soluciones que busquemos.

Las definiciones 2.23–2.25 clasifican las soluciones abductivas de una forma que bien podemos llamar *semántica*, pues en cada caso se hace referencia a las relaciones de consecuencia lógica —en sentido clásico— que deben o no verificarse. Pero también podemos clasificar las soluciones abductivas por su forma *sintáctica*, es decir, según su estructura lógica. Siguiendo a Aliseda, distinguimos a continuación entre abducciones *atómicas*, *conjuntivas* y *complejas*[4].

Definición 2.28 *Dada una fórmula α que es solución abductiva a cierto problema abductivo,*

- *Si α es un literal, decimos que se trata de una* abducción atómica.
- *Si α es una conjunción de literales, decimos que se trata de una* abducción conjuntiva.
- *En otro caso, decimos que α es una* abducción compleja.

La primera observación que debemos hacer es que al decir que α es una abducción *atómica*, el sentido del término indica algo diferente a cuando decimos que α es una fórmula *atómica*. En este último caso, como hemos explicado, lo que indicamos con *atómica* es que α no contiene ningún operador, pues su grado lógico es 0. Sin embargo, esta propiedad no la comparten todas las abducciones

[4]Aliseda, al tercer tipo de abducciones las llama *disyuntivas*, y son todas las del tipo $\alpha \vee \beta$, siendo α y β conjunciones elementales. Fácilmente se observa que las *explicaciones disyuntivas* de Aliseda son un subconjunto de nuestras *abducciones complejas*.

atómicas, pues pueden ser también literales negativos, de grado 1. El sentido de *atómica* se refiere ahora a que consta de un solo literal. Dado que el adjetivo puede prestarse a ambigüedades, será siempre precedido del sustantivo correspondiente, distinguiendo entre *fórmula atómica* —grado lógico 0— y *abducción atómica* o *solución atómica* —literal— con lo que la ambigüedad desaparece.

Otra observación importante es que todas las abducciones atómicas pueden verse también como conjuntivas, si consideramos conjunciones de un solo literal, tal como hicimos en la definición 2.15. Entonces, las abducciones pueden clasificarse entre las que integran la clase formada por las atómicas y las conjuntivas, por un lado, y las complejas, por otro.

Así como al clasificar las abducciones en planas, consistentes y explicativas optamos por estas últimas, ahora elegimos las que son atómicas o conjuntivas. La razón para ello es que el conjunto de abducciones complejas resulta mucho menos abarcable, como posteriormente se verá. Sólo a finales del capítulo 6 abordaremos el problema de la generación de abducciones complejas.

Por tanto, buscaremos las soluciones conjuntivas —entre las que están las atómicas, como se ha observado— explicativas. Además, exigiremos que sean minimales, según el criterio que establece la siguiente definición.

Definición 2.29 (Solución minimal) *Dada una solución abductiva explicativa α para el problema abductivo $\langle \Theta, \phi \rangle$, si α es la conjunción de literales $\lambda_1 \wedge \ldots \wedge \lambda_n$, $n \geq 1$, decimos que α es* minimal *syss no existe ningún $\{\gamma_1, \ldots, \gamma_m\} \subset \{\gamma_1, \ldots, \gamma_n\}$, $1 \leq m < n$, tal que $\gamma_1 \wedge \ldots \wedge \gamma_m$ sea una solución abductiva explicativa al problema abductivo $\langle \Theta, \phi \rangle$.*

La definición 2.29 lo que establece es que si α es una solución abductiva a cierto problema abductivo $\langle \Theta, \phi \rangle$ y además α es una conjunción de literales, no exista ningún subconjunto propio de esos literales tal que su conjunción sea también una solución abductiva explicativa. Pero si unimos las tres relaciones que

deben verificarse para que α sea solución abductiva explicativa, que son

$$\begin{aligned} \Theta, \alpha &\models \phi \\ \Theta, \alpha &\not\models \bot \\ \alpha &\not\models \phi \end{aligned}$$

es muy fácil observar que la única relación que puede romperse al quitar literales de α es la primera. Por tanto, cuando decimos que α es una solución minimal, estamos diciendo que para cualquier α' que sea una conjunción de un subconjunto propio de los literales de α se cumple

$$\Theta, \alpha' \not\models \phi.$$

El criterio de minimalidad resulta adecuado exigirlo a las explicaciones, pues supone la ausencia de literales sobrantes. Es decir, lo que significa que una solución abductiva explicativa conjuntiva sea minimal es que se trata de una conjunción tal que todos los literales que contiene son necesarios para que se cumpla el requisito fundamental. En otro caso, como hemos mostrado poco más arriba, se trataría de una conjunción de la que bien podrían eliminarse literales.

Resumiendo, para cada problema abductivo $\langle \Theta, \phi \rangle$ nos interesarán, en principio, sólo las soluciones que sean a la vez *explicativas*, según la definición 2.25, *atómicas* o *conjuntivas*, según la definición 2.28, y *minimales*, según la definición 2.29.

Además, será deseable establecer *criterios preferenciales*, es decir, algún tipo de rasgo que haga a algunas de las abducciones obtenidas preferibles a las demás. Más adelante veremos algunos de los criterios preferenciales posibles.

2.3. Análisis estructural

Como hemos visto en la sección anterior, tanto para que $\langle \Theta, \phi \rangle$ pueda ser considerado un problema abductivo como para que α sea una solución explicativa, deben verificarse ciertas relaciones entre Θ, ϕ y α. En esta sección veremos cómo

tales relaciones son muy sensibles a lo que podemos llamar el *estado de la teoría*, que en términos formales viene determinado por las fórmulas que componen Θ. A la propia definición de problema abductivo le ocurre esto, como muestra el siguiente ejemplo.

Ejemplo 2.30 Mientras que $\langle\{p \rightarrow q\}, q\rangle$ es un problema abductivo, según la definición 2.21, si añadimos a la teoría el literal p, tenemos que $\langle\{p, p \rightarrow q\}, q\rangle$ ya no lo es, puesto que la observación q es consecuencia lógica de la teoría $\{p, p \rightarrow q\}$.

Como veremos más adelante, también para que cierta fórmula α sea una solución a un problema abductivo $\langle\Theta, \phi\rangle$ resulta determinante el estado de la teoría. Esto es sumamente interesante de cara al empleo de la abducción para formalizar ciertos razonamientos de sentido común, donde pequeños cambios en el estado de la teoría —lo que se piensa de las cosas— producen en ocasiones grandes cambios en las explicaciones que se consideran válidas. Para abordar formalmente esta cuestión, definimos la relación abductiva "$\Rightarrow_a$".

Definición 2.31 (Relación de consecuencia abductiva) *Dados $\Theta \subset \mathcal{L}_p$ y $\phi, \alpha \in \mathcal{L}_p$, decimos que se cumple la relación $\Theta \mid \phi \Rightarrow_a \alpha$ syss*

1. *$\langle\Theta, \phi\rangle$ es un problema abductivo, según la definición 2.21.*
2. *α es una solución explicativa al problema abductivo $\langle\Theta, \phi\rangle$, según la definición 2.25.*

Si no se verifica alguna de las dos condiciones anteriores escribimos $\Theta \mid \phi \not\Rightarrow_a \alpha$.

Nos centramos en la abducción explicativa porque, como ya hemos argumentado, es la que nos parece más interesante. Sin embargo, en la definición 2.31 no hemos exigido que α tenga ninguna forma lógica particular: podría ser atómica, conjuntiva o compleja.

Lo primero que observamos es que "$\Rightarrow_a$" no es una función, pues dado cierto problema abductivo puede haber más de una —al permitirse abducciones complejas habrá infinitas— solución explicativa. En este sentido, la relación "$\Rightarrow_a$"

nos recuerda a la relación "$\models$", que introdujimos en la definición 2.4, comúnmente conocida como *relación de consecuencia lógica clásica*. También para un mismo conjunto de fórmulas hay infinitas fórmulas que son consecuencia lógica, como puede comprobarse fácilmente. Sin embargo, lo que ahora más nos interesa es ver las diferencias entre ambas relaciones. La relación "$\models$" tiene una serie de propiedades llamadas *estructurales*, que se representan mediante reglas de la forma

$$\frac{A_1 \quad \ldots \quad A_n}{B}$$

indicando que siempre que se verifiquen todos los esquemas A_i, $1 \leq i \leq n$, se verificará el esquema B. Cada esquema tiene una forma del tipo $\Gamma \models \alpha$, siendo Γ cualquier conjunto de fórmulas y α cualquier fórmula. A continuación veremos las propiedades de "$\models$" que nos parecen más interesantes para comparar con "$\Rightarrow_a$". En parte, seguimos a G. Palau [Pal02], S.C. Kleene [Kle74] y A. Aliseda [Ali97][5].

Teorema 2.32 *La relación de consecuencia lógica clásica "$\models$" verifica —para cualesquiera $\Gamma, \Lambda \subset \mathcal{L}_p$ y $\alpha, \beta \in \mathcal{L}_p$— las siguientes propiedades estructurales*

- Reflexividad,

$$\frac{}{\Gamma \cup \{\alpha\} \models \alpha} \tag{2.30}$$

- Monotonía,

$$\frac{\Gamma \models \alpha}{\Gamma \cup \Lambda \models \alpha} \tag{2.31}$$

- Transitividad, *también llamada* regla de corte

$$\frac{\Gamma \models \alpha \quad \Lambda \cup \{\alpha\} \models \beta}{\Gamma \cup \Lambda \models \beta} \tag{2.32}$$

Demostración:

En cuanto a la reflexividad, resulta trivial que cada fórmula sea consecuencia lógica de cualquier conjunto que la contenga, puesto que por definición toda

[5]Las reglas estructurales para la abducción que a continuación presentamos no son exactamente las que proporciona Aliseda. De todos modos, en esta sección, más que un estudio sistemático pretendemos mostrar cómo el razonamiento abductivo no posee las mismas propiedades estructurales de la lógica clásica, ya que ésta se ocupa del estudio de las inferencias deductivas, y la abducción es un modo de razonamiento diferente.

valoración que satisfaga cualquier conjunto $\Gamma \cup \{\alpha\}$ satisface α. También es fácil demostrar la monotonía, puesto que si cada valoración que satisface Γ satisface α, también cada valoración que satisface $\Gamma \cup \Lambda$, puesto que por definición debe satisfacer Γ, satisface α.

Pasemos a la transitividad. Partimos de que se verifican

$$\Gamma \models \alpha \tag{2.33}$$

$$\Lambda \cup \{\alpha\} \models \beta \tag{2.34}$$

y tenemos que probar

$$\Gamma \cup \Lambda \models \beta \tag{2.35}$$

Ahora bien, sea v una valoración que satisface $\Gamma \cup \Lambda$. Entonces, por definición, v satisface Γ y v satisface Λ. Pero como satisface Γ, por (2.33) tenemos que v satisface α, y además como v satisface Λ se verifica que v satisface $\Lambda \cup \{\alpha\}$, de forma que por (2.34) probamos $v \models \beta$. Como este razonamiento vale para cada valoración que satisfaga $\Gamma \cup \Lambda$, llegamos a $\Gamma \cup \Lambda \models \beta$ (2.35). ■

A una lógica cuya relación de consecuencia lógica no verifique alguna de las propiedades estructurales de "$\models$", se le llama a veces *lógica subestructural*. Concretamente, a una lógica que no verifique la monotonía, que es la propiedad estructural más significativa de "$\models$", se le llama *lógica no monótona*.

En (2.31) se puede apreciar que la monotonía de "$\models$" consiste en que si se verifica $\Gamma \models \alpha$ siempre se va a verificar $\Gamma \cup \Lambda \models \alpha$ para cualquier Λ. Sin embargo, volviendo a la abducción, en el ejemplo 2.30 vimos que las relaciones que deben mantenerse entre Θ y ϕ para que $\langle \Theta, \phi \rangle$ sea un problema abductivo, según la definición 2.21, pueden perderse si se añaden fórmulas a la teoría. Por tanto, en cierto sentido, la propia noción de problema abductivo no es monótona.

A continuación comprobaremos que las propiedades estructurales de "$\models$" no se cumplen para "$\Rightarrow_a$", pues resulta evidente que las modificaciones en el estado de la teoría pueden anular la relación abductiva. Como es esperable, esta relación no es monótona. Dadas Θ, ϕ y α tales que $\Theta \mid \phi \Rightarrow_a \alpha$, es posible que exista una fórmula γ tal que $\Theta \cup \{\gamma\} \mid \phi \not\Rightarrow_a \alpha$. A continuación mostramos las cinco

condiciones que se deben dar para que se verifique $\Theta \mid \phi \Rightarrow_a \alpha$ y comentamos cuáles de ellas pueden romperse si se añade γ a Θ, según sea γ,

1. $\Theta \nvDash \phi$. En este caso, puede dejar de verificarse $\Theta \cup \{\gamma\} \nvDash \phi$ por ejemplo si γ es ϕ, ya que por definición toda valoración que satisfaga $\Theta \cup \{\phi\}$ satisface ϕ.

2. $\Theta \nvDash \neg\phi$. También se deja de verificar $\Theta \cup \{\gamma\} \nvDash \neg\phi$ si por ejemplo γ es $\neg\phi$, pues toda valoración que satisfaga $\Theta \cup \{\neg\phi\}$ satisface $\neg\phi$.

3. $\Theta, \alpha \vDash \phi$. En este caso, siempre ocurrirá que $\Theta \cup \{\gamma\}, \alpha \vDash \phi$ para cualquier γ, por la propia monotonía de la relación de consecuencia lógica clásica.

4. $\Theta, \alpha \nvDash \bot$. Esta relación también se puede romper, por ejemplo si γ es $\neg\alpha$, pues por definición ninguna valoración puede satisfacer $\Theta \cup \{\alpha, \neg\alpha\}$.

5. $\alpha \nvDash \phi$. Como la teoría no interviene en esta relación, siempre se mantendrá sea cual sea γ.

Por tanto, la relación $\Theta \mid \phi \Rightarrow_a \alpha$ puede romperse si se añade γ a Θ porque dejen de verificarse las condiciones 1, 2 y 4 anteriores. Una observación que no hemos encontrado en otro sitio es que precisamente estas tres condiciones que pueden fallar son aquellas en las que por un lado está envuelta la teoría y que además lo que se verifica es que no se establezca cierta relación de consecuencia lógica clásica. Pero a esta observación podemos añadir otra trivial, que si bien la relación "$\vDash$" es monótona, sin embargo no lo es "$\nvDash$", ya que por ejemplo $p \nvDash q$ pero sin embargo $p, q \vDash q$. Por esto, pensamos que la no monotonía de la relación abductiva "$\Rightarrow_a$" depende directamente de la no monotonía de "$\nvDash$".

Pasemos a la reflexividad, que referida a "$\vDash$" significa, como indica (2.30), que toda fórmula es consecuencia lógica de cada conjunto que la contenga. En cuanto a "$\Rightarrow_a$" no ocurre nada similar, es decir, nunca se verifica $\Theta \cup \{\alpha\} \mid \phi \Rightarrow_a \alpha$ ya que ninguna fórmula α puede ser una solución abductiva explicativa para un problema abductivo que la contenga, tipo $\langle \Theta \cup \{\alpha\}, \phi \rangle$, pues dado que por la definición 2.21 esto implica que $\Theta \cup \{\alpha\} \nvDash \phi$, es decir, que existe una valoración v tal que satisface $\Theta \cup \{\alpha\}$ pero no satisface ϕ, tendríamos, por definición, que v

satisface $\Theta \cup \{\alpha\} \cup \{\alpha\}$ —que, por cierto, es igual a $\Theta \cup \{\alpha\}$— pero no a ϕ, por lo que no se verifica $\Theta \cup \{\alpha\} \cup \{\alpha\} \models \phi$, de modo que no se cumple el requisito fundamental para que α pueda ser una solución abductiva de ningún tipo.

Aunque siempre estamos tratando las modificaciones en la teoría, a propósito de la reflexividad vemos que tampoco puede ocurrir nunca $\Theta \mid \alpha \Rightarrow_a \alpha$, es decir, que la explicación sea la propia observación, pues entonces no se trata de una abducción explicativa, que es uno de los requisitos que impone la definición 2.31.

A propósito de la transitividad, una versión interesante de la regla de corte para la relación de consecuencia abductiva podría ser

$$\frac{\Theta_1 \mid \alpha \Rightarrow_a \gamma \quad \Theta_2 \mid \phi \Rightarrow_a \alpha}{\Theta_1 \cup \Theta_2 \mid \phi \Rightarrow_a \gamma} \tag{2.36}$$

pues indica que si desde la teoría Θ_2 puede explicarse ϕ con α, pero a su vez α puede ser explicada por γ en cierta teoría Θ_1, entonces en una teoría que acumule todo el conocimiento de Θ_1 y Θ_2 juntas ϕ puede explicarse con γ. La regla (2.36), sin embargo, no se verifica en todos los casos, pues si por ejemplo

$$\begin{aligned} \Theta_1 &= \{s \rightarrow p, \neg q\} \\ \Theta_2 &= \{p \rightarrow q\} \\ \alpha &= p \\ \gamma &= s \\ \phi &= q \end{aligned}$$

tenemos que $\Theta_1 \mid \alpha \Rightarrow_a \gamma$ y $\Theta_2 \mid \phi \Rightarrow_a \alpha$, pero sin embargo $\Theta_1 \cup \Theta_2 \mid \phi \not\Rightarrow_a \gamma$, ya que $\Theta_1, \Theta_2 \models \neg\phi$, por lo que ni siquiera es un problema abductivo $\langle \Theta_1 \cup \Theta_2, \phi \rangle$.

Aliseda, tras constatar que la abducción no posee unas reglas estructurales equivalentes o asimilables a las de la relación de consecuencia lógica clásica, explora si al menos se cumple alguna de tales reglas, aunque sea en una versión *cautelosa*, con ciertas restricciones adicionales. La motivación para realizar este estudio es la duda que se podría presentar de llamar *lógica* a una relación de consecuencia que no tenga ninguna de las propiedades estructurales conocidas de la relación de consecuencia lógica clásica.

Volviendo a la monotonía, se verifican ciertas versiones de *monotonía cautelosa*, como por ejemplo

$$\frac{\Theta \mid \phi \Rightarrow_a \alpha}{\Theta, \gamma \mid \phi \Rightarrow_a \alpha}*$$

siendo la restricción indicada por $*$ que deben verificarse $\Theta, \gamma, \alpha \nvdash \bot$ y $\Theta, \gamma, \neg\phi \nvdash \bot$.

También la regla (2.36) se verifica si se añaden restricciones, aunque en este caso en mayor número, pues deberían cumplirse $\Theta_1, \Theta_2 \nvdash \phi$, $\Theta_1, \Theta_2 \nvdash \neg\phi$ —estas dos para que $\langle \Theta_1 \cup \Theta_2, \phi \rangle$ sea un problema abductivo—, $\Theta_1, \Theta_2, \gamma \nvdash \bot$ y $\gamma \nvdash \phi$.

Capítulo 3

Abducción y tablas semánticas

En este capítulo comenzamos nuestro estudio de la abducción proposicional, tomando como punto de partida el método que proponen Marta Cialdea y Fiora Pirri [CP93] y Atocha Aliseda [Ali97] para la generación de hipótesis explicativas a partir de tablas semánticas. En una primera sección presentamos las tablas semánticas de Beth [Bet69] en su versión proposicional. A continuación repasamos diversas formas de implementación y optamos por la más eficiente. En la tercera sección se expondrá el sistema de abducción de las autoras mencionadas y, por último, la cuarta sección está dedicada a mostrar una implementación del procedimiento que propone Aliseda para producir soluciones abductivas explicativas a problemas abductivos proposicionales. Tomando esta implementación, realizaremos un análisis de su eficiencia.

3.1. Tablas semánticas proposicionales

El método de las tablas semánticas, propuesto por Beth [Bet69], permite realizar una búsqueda sistemática de modelos para un conjunto de fórmulas dado. De esta forma, hace posible establecer relaciones de consecuencia lógica de un modo semántico. Así, para determinar mediante tablas semánticas si la fórmula β es consecuencia lógica del conjunto de fórmulas $\alpha_1, \ldots, \alpha_n$, se buscará, mediante

una tabla semántica, un modelo para el conjunto de fórmulas $\{\alpha_1, \ldots, \alpha_n, \neg\beta\}$. En caso de que sea posible encontrar tal modelo, tendremos un contraejemplo para la relación de consecuencia lógica entre las fórmulas propuestas —al satisfacer las premisas pero no la conclusión—. En otro caso, si el conjunto de fórmulas dado no tiene modelo sabemos que

$$\alpha_1, \ldots, \alpha_n, \neg\beta \vDash \bot,$$

de donde[1],

$$\alpha_1, \ldots, \alpha_n \vDash \beta.$$

De entre las ventajas que presenta el método de las tablas semánticas, hay dos que lo hacen muy útil para su aplicación a la resolución de problemas abductivos:

1. Se trata de un método de carácter general. Aunque su formulación fue originalmente propuesta para lógica clásica, es posible extender el procedimiento para lógicas no clásicas, como la lógica modal [BG96] o lógicas multivaluadas [BFZ93]. De esta forma, tomando las tablas semánticas como base para procesos explicativos, puede esperarse que sea posible su extensión a diferentes lógicas.

2. Su atractivo computacional. A pesar de que, desde su introducción, el método de resolución de Robinson [Rob65] supuso la casi completa preeminencia de este cálculo dentro de los sistemas de demostración automática, es posible realizar implementaciones muy eficientes —en ciertos casos superando la eficiencia de la resolución— del cálculo de tablas semánticas, tal como Beckert y Posegga han mostrado con la realización de leanTAP [BP95].

A continuación pasamos a la presentación formal del método de las tablas semánticas para lógica proposicional. Partimos de la exposición de [Nep03], aunque para la representación de las tablas usaremos ciertos elementos de la teoría de grafos.

Definición 3.1 (Tipos de fórmulas) *Clasificamos las fórmulas de $\mathcal{L}_p$ en las cuatro clases siguientes:*

[1] Por el teorema 2.9. En adelante se omitirán las referencias a teoremas elementales.

1. Dobles negaciones. *Son las fórmulas de tipo* $\neg\neg\phi$*, siendo* $\phi \in \mathcal{L}_p$ *cualquier fórmula proposicional.*

2. Literales. *Son las fórmulas de tipo* $\neg\phi$ *o* ψ*, para* $\phi, \psi \in \mathcal{P}$ *variables proposicionales. Decimos que* ϕ *y* $\neg\phi$ *son literales complementarios.*

3. Fórmulas α***.*** *Son las que en el siguiente cuadro aparecen debajo de la primera columna —etiquetada con* α*—, siendo* $\phi, \psi \in \mathcal{L}_p$ *cualesquiera fórmulas proposicionales:*

α	α_1	α_2
$\phi \wedge \psi$	ϕ	ψ
$\neg(\phi \vee \psi)$	$\neg\phi$	$\neg\psi$
$\neg(\phi \rightarrow \psi)$	ϕ	$\neg\psi$

Para cada fórmula de tipo α*, decimos de cada una de las dos fórmulas que aparecen en su misma fila, pero en las columnas etiquetadas con* α_1 *y* α_2*, que son, respectivamente, sus componentes* α_1 *y* α_2*.*

4. Fórmulas β***.*** *Son las que en el siguiente cuadro aparecen debajo de la primera columna —etiquetada con* β*—, siendo* $\phi, \psi \in \mathcal{L}_p$ *cualesquiera fórmulas proposicionales:*

β	β_1	β_2
$\phi \vee \psi$	ϕ	ψ
$\neg(\phi \wedge \psi)$	$\neg\phi$	$\neg\psi$
$\phi \rightarrow \psi$	$\neg\phi$	ψ
$\phi \leftrightarrow \psi$	$\phi \wedge \psi$	$\neg\phi \wedge \neg\psi$
$\neg(\phi \leftrightarrow \psi)$	$\phi \wedge \neg\psi$	$\neg\phi \wedge \psi$

Para cada fórmula de tipo β*, decimos de cada una de las dos fórmulas que aparecen en su misma fila, pero en las columnas etiquetadas con* β_1 *y* β_2*, que son, respectivamente, sus componentes* β_1 *y* β_2*.*

Observación 3.2 Por evaluación de los operadores lógicos se puede comprobar que dada cualquier fórmula $\phi \in \mathcal{L}_p$ y cualquier valoración v,

- Si ϕ es de la clase α, entonces v satisface ϕ syss v satisface las dos componentes α_1 y α_2 de ϕ.
- Si ϕ es de la clase β, entonces v satisface ϕ syss v satisface al menos una de las dos componentes β_1 y β_2 de ϕ.

Observación 3.3 Es fácil comprobar que para cada fórmula proposicional, la definición 3.1 la clasifica en una y sólo una clase. Esta observación resultará útil en la demostración de los teoremas de corrección y completud del método de las tablas semánticas.

Como vamos a representar las tablas semánticas como árboles, introducimos algunas definiciones elementales de la teoría de grafos[2].

Definición 3.4 (Grafo) *Un* grafo, *$G = (N, L)$ se compone de un conjunto N de* nodos *y un conjunto L de* lados *formado por pares no ordenados de nodos diferentes.*

Definición 3.5 (Caminos) *Dado un grafo G se denomina* camino *en G una secuencia de* nodos *n_i y* lados *l_i de G*

$$n_1, l_1, n_2, l_2, n_3, \ldots, n_k, l_k, n_{k+1}$$

que tiene como origen el nodo n_1 y como extremo el nodo n_{k+1} y que contiene k lados $l_i = \{n_i, n_{i+1}\}$, siendo $1 \leq i \leq k$.

Llamamos longitud *de un camino al número de lados que contiene.*

Decimos de un camino que es simple *si no tiene* lados repetidos. *Además, decimos que un camino es* elemental *si es simple y no tiene* nodos repetidos*; es decir, son* caminos elementales *los que no pasan dos veces por el mismo lado ni por el mismo nodo. Llamamos* ciclo *a todo* camino elemental cerrado*, es decir, un camino sin nodos ni lados repetidos, donde coinciden los nodos inicial y final.*

[2]Para más detalles sobre grafos, árboles y sus propiedades, ver, por ejemplo, [Gri97].

De un grafo $G = (N, L)$ donde existe un camino elemental entre cualquier par de vértices decimos que es conexo.

Definición 3.6 (Árbol) *Sea $G = (N, L)$ un grafo. G es un* árbol *si es conexo y no contiene ciclos.*

Definición 3.7 (Árbol con raíz) *Dado un árbol A decimos que es un* árbol con raíz *si uno de sus vértices se nombra como* raíz.

Definición 3.8 (Nivel de un nodo) *Dado un árbol con raíz A, y un nodo n cualquiera, llamamos* nivel *de n a la longitud del camino que lo conecta con la raíz de A. Además, si existe un camino en A que pasa por los nodos n_1 y n_2, siendo el nivel de n_2 superior al de n_1, decimos que n_2 es un* descendiente *de n_1, y éste un* ancestro *de n_2. Si, además, el nivel de n_2 es sólo una unidad mayor que el de n_1, decimos que n_1 es* padre *de n_2, y éste* hijo *de n_1.*

Llamamos hojas *de un árbol con raíz a los nodos que no tienen hijos. A los demás nodos los llamamos* internos. *Llamamos* ramas *a todos los caminos de un árbol con raíz que van desde una hoja hasta la raíz.*

Definición 3.9 (Árbol n-ario) *Dado un árbol con raíz, decimos que es n-ario si todo vértice interno tiene, a lo sumo, n hijos.*

A continuación, explicamos la construcción de las tablas semánticas empleando la terminología presentada.

Definición 3.10 (Tabla semántica) *Una tabla semántica $\mathcal{T}(\Gamma)$ para un conjunto de fórmulas proposicionales $\Gamma \subset \mathcal{L}_p$ es un árbol binario que se construye partiendo de una rama que tiene $|\Gamma|$ nodos, que llevarán por nombre cada una de las fórmulas de Γ. Al principio, consideraremos tales nodos* no usados. *A continuación, se sigue el procedimiento no determinista*[3] *que establecen las siguientes reglas:*

[3]Un procedimiento o algoritmo es *determinista* cuando a partir de cada uno de sus pasos se puede determinar de forma unívoca el siguiente. Sin embargo, cuando el procedimiento o algoritmo es *no determinista*, para cada uno de sus pasos debe elegirse entre varias alternativas, y a menudo agotarlas todas —no será este el caso de las tablas semánticas proposicionales— antes de encontrar una solución.

1. Regla de cierre: *Si en cierta rama ocurren dos nodos* ϕ *y* $\neg\phi$*, siendo* ϕ *cualquier variable proposicional, entonces consideramos dicha rama* cerrada*, detenemos su construcción y le añadimos al final un nuevo nodo que llamamos* $\bigotimes$.

2. Regla de doble negación: *Si en una rama hay un nodo* $\neg\neg\phi$ no usado*, añadimos a cada rama que contenga este nodo un nuevo nodo que llamamos* ϕ*. El nodo* $\neg\neg\phi$ *se considerará en adelante* usado*, y las* ϕ *añadidas,* no usadas*.*

3. Regla-α *Si en una rama hay un nodo* no usado ϕ*, siendo* ϕ *una fórmula de la clase* α*, continuamos la construcción de cada rama de la tabla que comparta el nodo* ϕ *añadiéndole el grafo de la figura 3.1, siendo* fin *el nodo que hasta ahora ocupaba el último lugar de cada una de las ramas que comparten el nodo* ϕ*. Pasamos a considerar* ϕ usado *y todos los nodos* α_1 *y* α_2 *añadidos* no usados*.*

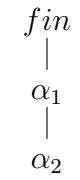

Figura 3.1: Grafo para fórmulas de tipo α

4. Regla-β *Si en una rama hay un nodo* no usado ϕ*, siendo* ϕ *una fórmula de la clase* β*, continuamos la construcción de cada rama de la tabla que comparta el nodo* ϕ *añadiéndole el grafo de la figura 3.2, siendo* fin *el nodo que hasta ahora ocupaba el último lugar de cada una de las ramas que comparten el nodo* ϕ*. Pasamos a considerar* ϕ usado *y todos los nodos* β_1 *y* β_2 *añadidos* no usados*.*

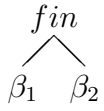

Figura 3.2: Grafo para fórmulas de tipo β

La construcción de la tabla termina cuando todas sus ramas son cerradas*, en cuyo caso decimos que la tabla es* cerrada*, o bien cuando queda alguna rama no cerrada cuya construcción no puede continuarse. Llamamos a tal rama* abierta*, y a la tabla que tiene al menos una* rama abierta *la llamamos* tabla abierta*.*

Tal como hemos redactado la definición anterior, podría ocurrir que al construir una tabla semántica aparecieran nodos diferentes con el mismo nombre. Si quiere evitarse esta duplicidad, una solución sencilla es añadir al nombre de cada nodo, junto a la fórmula que contiene, un número que lo identifica de forma unívoca. Es lo que hacemos en el siguiente ejemplo.

Ejemplo 3.11 Explicamos a continuación la construcción de la tabla semántica para el conjunto de fórmulas $\{\neg p \to q, \neg(a \vee q)\}$ que aparece en la figura 3.3. Al

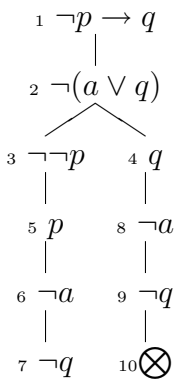

Figura 3.3: Tabla semántica de $\{\neg p \to q, \neg(a \vee q)\}$

haber añadido números a los nombres de los nodos, nos referiremos a ellos en lugar de las fórmulas correspondientes. La construcción comienza creando un grafo con los dos nodos 1 y 2. Hemos comenzado usando el nodo 1, aunque podríamos haber empezado por 2, debido al carácter no determinista del procedimiento[4]. Al tratarse una fórmula de la clase β añadimos los nodos 3 y 4 a la única rama que contiene el nodo 1. En adelante, el nodo 1 se considerará *usado* y los demás, de momento, *no usados*. A continuación, usamos el nodo 3, que al ser una doble negación nos hace añadir el nodo 5. El nodo 3 pasa a ser *usado*, y el 5 *no usado*. Por último, usamos 2, que es de la clase α, por lo que añadimos el correspondiente grafo a las dos ramas que contienen el nodo 2, produciendo en una de ellas los nodos 6–7 y en la otra los 8–9. Como en la segunda rama —contamos de izquierda a derecha— ocurren los literales complementarios 4 y 9, la consideramos cerrada, y añadimos el nodo 10 que lo indica. La rama primera ha sido completada, pues

[4]Será este no determinismo lo que dará sentido a la búsqueda de estrategias que hagan más eficiente la creación de tablas semánticas, tal como discutiremos en la sección dedicada a la implementación.

sus fórmulas no literales han sido ya usadas. Por tanto, dicha rama —y también la tabla— es abierta.

Teorema 3.12 (Completud) *Dado un conjunto* $\Gamma \subset \mathcal{L}_p$ *de fórmulas proposicionales, si hay una tabla semántica de* Γ *con al menos un rama abierta, entonces es definible una valoración* v *que satisface todas las fórmulas de* Γ.

Demostración:

Sea $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$, $1 \leq n$. Si existe una tabla semántica de Γ con una rama abierta, dicha rama contendrá k nodos, y $k \geq n$, pues se compondrá de todas las fórmulas de Γ más las que se hayan añadido —si se añadió alguna—. Por tanto, los nodos de tal rama serán $N = \{\gamma_1, \gamma_2, \ldots, \gamma_n, \eta_1, \eta_2, \ldots, \eta_{k-n}\}$.

Sean $\{\lambda_1, \lambda_2, \ldots, \lambda_l\}$ los literales que pertenecen a N. Definimos entonces los siguientes conjuntos:

1. $\Phi_0 = \{\lambda_1, \lambda_2, \ldots, \lambda_l\}$
2. Para cada $h \geq 0$, Φ_{h+1} es el más pequeño conjunto que verifica:
   - *a)* si $\phi \in \Phi_h$ y $\neg\neg\phi \in N$, entonces $\neg\neg\phi \in \Phi_{h+1}$,
   - *b)* si $\phi \in \Phi_h$ y ϕ es una componente de una fórmula π de la clase β de N, entonces $\pi \in \Phi_{h+1}$,
   - *c)* si $\phi \in \Phi_h$ y $\psi \in \Phi_j$, $j \leq h$, y ϕ, ψ son las dos componentes de una fórmula π de la clase α de N, entonces $\pi \in \Phi_{h+1}$.

Definimos la valoración v de forma que para cada variable proposicional x,

$$v(x) = \begin{cases} 1 & \text{si } x \in \Phi_0, \\ 0 & \text{en otro caso.} \end{cases}$$

Veamos que v satisface las fórmulas de todos los conjuntos Φ_j, $0 \leq j$. Procedemos por inducción sobre j:

1. Para $j = 0$, encontramos que Φ_0, por definición, sólo contiene literales, y que por partir de una rama abierta, no tiene literales complementarios. Además, por definición de v, sabemos que v satisface todos sus literales positivos, pues satisface todas las variables proposicionales que ocurren en Φ_0. Además, para cada literal negativo $\neg\epsilon \in \Phi_0$, tenemos que como $\epsilon \notin \Phi_0$ —al no tener Φ_0 literales complementarios—, entonces por definición de v se verifica $v(\epsilon) = 0$, por lo que $v(\neg\epsilon) = 1$, con lo que v satisface también todos los literales negativos de Φ_0, y por tanto todo Φ_0.

2. Supongamos probado que v satisface todos los Φ_j para $j \leq n$. Veamos si v satisface Φ_{n+1}. Para cada fórmula $\delta \in \Phi_{n+1}$ debe ocurrir, según la construcción de Φ_{n+1}, una de las tres siguientes posibilidades:

   a) δ es $\neg\neg\phi$. Entonces, como $\phi \in \Phi_n$, v satisface ϕ y por tanto v satisface δ.

   b) δ es una fórmula de la clase β, uno de cuyos constituyentes es $\phi \in \Phi_n$. Como, por hipótesis, v satisface ϕ, entonces v satisface δ (observación 3.2).

   c) δ es una fórmula de la clase α cuyas componentes ϕ y ψ pertenecen, respectivamente a Φ_n y Φ_h, $h \leq n$. Entonces, por hipótesis, v satisface ambas componentes y, por tanto, v satisface δ (observación 3.2).

Por tanto, v satisface todas las fórmulas de cada uno de los Φ_j. Probemos que entre tales fórmulas están todas las que han aparecido en la rama abierta N. Lo haremos por inducción sobre el grado lógico de tales fórmulas:

1. Para los literales, tenemos que todos pertenecen a Φ_0, por definición.

2. Si suponemos probado que todas las fórmulas de la rama, hasta el grado lógico n, pertenecen a algún Φ_j, veamos qué ocurre para las de grado lógico $n + 1$. Cada una de tales fórmulas será (observación 3.3):

   a) Una doble negación $\neg\neg\phi$. Entonces al usarse tal fórmula se añadió a la rama ϕ. Por hipótesis de inducción, $\phi \in \Phi_i$ para algún i, y por construcción de Φ_{i+1}, $\neg\neg\phi \in \Phi_{i+1}$.

b) Una fórmula de tipo α con las componentes ϕ y ψ. En tal caso, por la construcción de la tabla, ϕ y ψ se añadieron a la rama, y por hipótesis, ambas subfórmulas pertenecen a conjuntos Φ_i, Φ_j, $j \leq i$ y, por construcción de Φ_{i+1}, la fórmula pertenece a Φ_{i+1}.

c) Una fórmula de tipo β. En tal caso, por la construcción de la tabla, alguna de sus componentes, digamos ϕ, se añadió a la rama, y por hipótesis, $\phi \in \Phi_i$, y, por construcción de Φ_{i+1}, la fórmula pertenece a Φ_{i+1}.

De modo que todas las fórmulas de la rama abierta N pertenecen a algún Φ_j, pero como las fórmulas de Γ pertenecen a la rama, todas pertenecen por tanto a algún Φ_j. Y como v satisface todas las fórmulas de los Φ_j, v satisface todas las fórmula de Γ. ■

Teorema 3.13 (Corrección) *Si $\Gamma \subset \mathcal{L}_p$ es un conjunto finito y satisfactible de fórmulas, entonces todas las tablas semánticas de Γ tienen al menos una rama abierta.*

Demostración:

Sea v una valoración que satisface todas las fórmulas de Γ. Para la prueba, procedemos por inducción sobre el número de aplicaciones en toda la tabla de las reglas de formación de tablas semánticas, demostrando que debe haber siempre una rama tal que todas su fórmulas sean satisfechas por v. Como caso base, consideremos que hay 0 aplicaciones de reglas; es decir, la tabla consta de una única rama con las fórmulas de Γ. Entonces resulta trivial que v satisface todas las fórmulas de dicha rama.

Supongamos que, en toda la tabla, se han aplicado las reglas n veces, $n \geq 0$, y tenemos una rama, que representamos por el conjunto de fórmulas —nodos— Φ, tal que v satisface todas sus fórmulas. Al aplicar la $(n+1)$-ésima regla, puede ocurrir que se aplique sobre una fórmula de Φ o sobre una fórmula de otra rama. En este último caso, la rama Φ seguirá siendo satisfecha por v, así que conside-

remos ahora el caso en que la $(n+1)$-ésima regla se aplica sobre una fórmula de Φ. Entonces estamos ante una de las siguientes posibilidades:

1. La fórmula es $\neg\neg\phi$, de forma que se aplica doble negación, con lo que la nueva rama resulta ser $\Phi \cup \{\phi\}$, y dado que v satisface $\neg\neg\phi$, por evaluación del negador también satisface ϕ, por lo que sigue satisfaciendo la nueva rama.

2. Es una fórmula ϕ de la clase α, de componentes ϕ_1 y ϕ_2, por lo que al aplicar la regla α, la rama resultante es $\Phi \cup \{\phi_1, \phi_2\}$. Como por hipótesis v satisface ϕ, entonces (observación 3.2) v satisface las dos componentes ϕ_1 y ϕ_2, por lo que v satisface la nueva rama.

3. Es una fórmula ϕ de la clase β, de componentes ϕ_1 y ϕ_2, por lo que al aplicar la regla β, Φ se divide en las dos ramas $\Phi \cup \{\phi_1\}$ y $\Phi \cup \{\phi_2\}$. Como por hipótesis v satisface ϕ, entonces (observación 3.2) v satisface una de las dos componentes ϕ_1 ó ϕ_2, por lo que v satisface alguna de las dos ramas.

Por tanto, tras $n+1$ aplicaciones de las reglas, sigue quedando una rama cuyas fórmulas son todas satisfechas por v. De modo que al completarse la tabla semántica sigue quedando una rama completa Φ^* cuyas fórmulas son todas satisfechas por v, por lo que tal rama debe ser abierta, pues de ser cerrada habría dos literales complementarios, con lo que v no podría satisfacer a ambos. ■

Observación 3.14 Hasta ahora nos hemos referido en plural al conjunto de tablas semánticas posibles de un conjunto Γ de fórmulas pues, al ser no determinista el procedimiento de la definición 3.10, un conjunto de fórmulas Γ puede tener más de una tabla semántica. Sin embargo, al probar en el teorema precedente que si Γ es satisfactible todas sus tablas semánticas serán abiertas, emplearemos el singular para referirnos por ejemplo a *la tabla semántica de* Γ, ya que lo que más nos importa de las tablas semánticas, su carácter abierto o cerrado, es independiente del orden en que se apliquen las reglas y, por tanto, de la forma concreta del árbol resultante.

Corolario 3.15 (Teorema fundamental) *Un conjunto finito de fórmulas* $\Gamma \subset \mathcal{L}_p$ *es satisfactible syss su tabla semántica es abierta.*

Demostración:

Por el teorema 3.12 sabemos que si la tabla semántica Γ es abierta, entonces Γ es satisfactible. En el otro sentido, por el teorema 3.13 tenemos que si Γ es satisfactible entonces su tabla semántica es abierta. ■

Corolario 3.16 *Un conjunto finito de fórmulas $\Gamma \subset \mathcal{L}_p$ es no satisfactible syss su tabla semántica es cerrada.*

Demostración:

Del corolario 3.15, por contraposición. ■

Estos dos corolarios prueban la corrección y la completud del método de las tablas semánticas. El siguiente nos servirá para encontrar modelos a partir de las ramas abiertas.

Corolario 3.17 (Construcción de modelos) *Dada una tabla semántica del conjunto de fórmulas $\Gamma \subset \mathcal{L}_p$, si tiene una rama abierta con el conjunto de literales Φ, entonces una valoración que satisface todos los $\phi \in \Phi$ satisface igualmente Γ.*

Demostración:

Para la prueba de este corolario basta retomar la demostración del teorema 3.12. Entonces, a partir de una rama abierta, definimos una valoración v que satisfacía todas las fórmulas de Γ. La única característica de tal valoración es que hacía verdaderos todos los literales de la rama abierta. ■

Ya tenemos los elementos que nos permiten emplear el método de las tablas semánticas como un cálculo. Así, para cualquier conjunto de fórmulas Γ —incluido el vacío— y cualquier fórmula ϕ,

$$\Gamma \vDash \phi$$

syss $\Gamma \cup \{\neg\phi\}$ no es satisfactible, pero esto, por el corolario 3.16 equivale a que la tabla semántica de $\Gamma \cup \{\neg\phi\}$ sea cerrada. Si tal tabla fuese abierta, por el

corolario 3.17 podríamos construir una valoración v que satisfaría $\Gamma \cup \{\neg\phi\}$, con lo que v sería un contramodelo que refuta que $\Gamma \models \phi$.

3.2. Implementación de las tablas semánticas

Como al final de este capítulo construiremos un programa que resuelva problemas abductivos mediante tablas semánticas proposicionales, ahora vamos a detenernos para proponer una implementación eficiente del cálculo de tablas semánticas para lógica proposicional.

Para nuestra implementación —que forma parte del fichero `libreria.pl` que mostramos en la página 282— partimos de la que Atocha Aliseda aporta en [Ali97], aunque con ciertas optimizaciones para hacer el código más eficiente. Algunas de las ideas para optimizar el programa las hemos tomado de leanTAP, un demostrador completo y correcto para lógica de primer orden, escrito en Prolog por Bernhard Beckert y Joachim Posegga [BP94, BP95], así como de algunas modificaciones de dicho programa, ya sean las propuestas por los propios autores [Pos93, PS99], por Fitting [Fit98], o por nosotros mismos [Sol04b]. El resultado es un demostrador proposicional sumamente eficiente[5].

La idea de la que partimos es la distinción entre diferentes tipos de fórmulas, tal como se hace en la definición 3.1. Ello lo hacemos mediante los predicados `alfa/3` y `beta/3`, de forma que, por ejemplo, `alfa(+FormAlfa,?Alfa1,?Alfa2)` se verifica syss `FormAlfa` es una fórmula de la clase α y `Alfa1` y `Alfa2` son, respectivamente, sus dos componentes α_1 y α_2. Téngase en cuenta que los operadores que usamos en Prolog[6], de menor a mayor precedencia son: "`-`" para la negación, "`&`" para la conjunción, "`v`" para la disyunción, "`=>`" para la implicación y "`<=>`" para la doble implicación.

[5] De hecho, hemos realizado análisis comparativos de eficiencia tomando fórmulas de las que Roy Dyckhoff propone en [Dyc97], así como las fórmulas proposicionales de [Pel86] y resulta ser —para lógica proposicional— más eficiente que las versiones que conocemos de leanTAP.

[6] En el apéndice A —página 245— se proporciona una introducción a la programación en Prolog. En los apéndices posteriores aparecen las implementaciones de los sistemas abductivos que se comentan a lo largo de este trabajo.

```
alfa(A & B, A, B).                beta(-(A <=> B), A & -B, -A & B).
alfa(-(A v B), -A, -B).           beta(A v B, A, B).
alfa(-(A => B), A, -B).           beta(-(A & B), -A, -B).
beta(A <=> B, A & B, -A & -B).    beta(A => B, -A, B).
```

En cuanto a las tablas, las representamos como una lista de sus ramas abiertas, y la representación de cada rama es también una lista de los literales que contiene. Además, las variables proposicionales serán representadas por *átomos* de Prolog. Así `[[a,b,c],[d,-e],[-f]]` se corresponde con una tabla semántica que tiene tres ramas abiertas, una de ella con los literales a, b y c, otra con los literales d y $\neg e$ y una última con el literal $\neg f$. La tabla cerrada, al no tener ramas abiertas, será representada con la lista vacía `[]`.

El predicado principal es `tabla(+ListaFml,-Tab)` que recibe como entrada una lista de fórmulas proposicionales y devuelve su tabla semántica. Así,

```
?- tabla([a => (b v c), a, -c],Tabla).
   Tabla = [[a, b, -c]]
   Yes
?- tabla([a => (b v c), a, -c, -b],Tabla).
   Tabla = []
   Yes
```

La construcción de las tablas se realiza con el predicado

`construye_tabla(+ListaFml,+TablaProv,-Tabla)`

que opera partiendo de la tabla provisional `TablaProv`, a la que se van añadiendo poco a poco las fórmulas de `ListaFml`, aplicando las reglas oportunas, hasta llegar a la tabla final `Tabla`. Las fórmulas se añaden con el predicado `añadir_fórmula_tabla/3`, de comportamiento semejante. Para más detalles sobre estos dos predicados, remitimos de nuevo al código de la página 282.

El predicado más interesante, donde puede observarse cómo se aplican las reglas, es `añadir_fórmula_rama(+Rama,+Fml,-Res)` que dada la rama `Rama`, le añade, aplicando las reglas necesarias, la fórmula `Fml` —lo que conllevará aplicar

sucesivamente una serie de reglas— y obtiene el resultado recogido en `Res`. Si durante el proceso se ha aplicado la regla β, `Res` constará de varias ramas. Como en la definición 3.10, el proceso para añadir fórmulas a las tablas distingue cuatro casos, según sea `Fml`:

1. Si `Fml` es la doble negación `-(-F)`, lo que debe añadirse es `F`.

```
añadir_fórmula_rama(R,-(-F),Res) :- !,
        añadir_fórmula_rama(R,F,Res).
```

2. Si `Fml` es una fórmula de tipo α de componentes `A` y `B`, entonces se añade `A` y al resultado se añade `B`. Como el resultado de añadirse `A` puede tener más de una rama —pues en `A` puede haber subfórmulas de tipo β—, para añadirle `B` se trata como una tabla.

```
añadir_fórmula_rama(R,Alfa,Res):-
        alfa(Alfa,A,B),!,
        añadir_fórmula_rama(R,A,Res1),
        añadir_fórmula_tabla(Res1,B,Res).
```

3. Si `Fml` es una fórmula de tipo β de componentes A y B, entonces se crean dos ramas: `R1`, que es el resultado de añadir `A`, y `R2`, que resulta de añadir `B`. El resultado final, `Tab`, es la unión de `R1` y `R2`.

```
añadir_fórmula_rama(R,Beta,Tab) :-
        beta(Beta,A,B),!,
        añadir_fórmula_rama(R,A,R1),
        añadir_fórmula_rama(R,B,R2),
        union(R1,R2,Tab).
```

4. Cuando `Fml` es un literal `Lit`, debemos comprobar si su complementario está o no en la rama. En caso de que esté, la rama se cierra. Si su complementario no está, debemos añadirlo. Sin embargo, si el propio literal estuviese ya en la rama no hace falta añadirlo, pues sería algo redundante, que aumentaría además innecesariamente el espacio que consume la representación de la tabla, así como el tiempo que se va a tardar, si aparece un nuevo literal `Lit2`, en comprobar si el complementario de éste está o no en la rama, ya que habrá un elemento más donde buscar. Por tanto, debemos comprobar tanto si `Lit`

como su complementario están o no en la rama. Para ello hemos definido el predicado `busca(+Lit,+ListaLits,?Res)` que se verifica si —siendo `Lit` un literal y `ListaLits` una lista de literales— `ListaLits` contiene o bien `Lit` —en este caso unifica `Res` con `i` de *idéntico*— o su complementario —entonces unifica `Res` con `n` de *negación*—. En caso de que `ListaLits` no contenga ni `Lit` ni su complementario, falla. La definición es la siguiente:

```
busca(L,[L|_],i):-!.
busca(-L,[L|_],n):-!.
busca(L,[-L|_],n):-!.
busca(L,[_|R],X):-
        busca(L,R,X).
```

Por tanto, cuando se encuentra un literal `Lit` se busca en la `Rama`. Si se encuentra el complementario, la rama que resulta es la vacía. Si se encuentra el propio literal, resulta la misma rama, aunque se mete dentro de una lista a efectos de mantener la coherencia con la representación que estamos siguiendo. Si la búsqueda no tuvo éxito —no están ni el literal ni su complementario— se añade el literal a la rama.

```
añadir_fórmula_rama(Rama,Lit,NuevaRama):-
        ( busca(Lit,Rama,R) ->
            ( R = n -> NuevaRama = []
            ; % R = i
                NuevaRama = [Rama])
        ; % no están ni F ni su complementario
            NuevaRama = [[Lit|Rama]]).
```

La implementación que hemos presentado tiene algunos rasgos que la hacen muy interesante desde un punto de vista computacional. Destacamos especialmente los siguientes factores:

- Su carácter determinista, habitual entre los demostradores de teoremas proposicionales, que hace que al no contener puntos de elección, la ejecución sea mucho más rápida y las pilas donde Prolog almacena los objetivos necesiten guardar menos información, lo que permite ahorrar bastante memoria.

- La clasificación de fórmulas —dobles negaciones, fórmulas α y fórmulas β—, que hemos encontrado en [AB02, Fit98] y que supone una mejora considerable de eficiencia respecto a la estrategia de leanT^AP, que consiste en realizar un preprocesamiento previo de las fórmulas para transformarlas en forma normal negada, donde sólo ocurren disyunciones, conjunciones y negadores —pero afectando sólo a variables proposicionales—. Aliseda [Ali97] propone algo similar a esto último, pues su implementación de las tablas sólo contempla conjunciones y disyunciones. Para fórmulas proposicionales, resulta más eficiente, por tanto, trabajar con las fórmulas directamente, sin hacer una transformación previa.

- Además, la representación de las tablas siempre contiene únicamente literales, pues las fórmulas que se van añadiendo se descomponen previamente. Esto también conlleva un ahorro de memoria. Se trata de un rasgo que se encuentra tanto en leanT^AP como en la implementación de Aliseda, y que no hemos modificado.

- Pero probablemente el rasgo más importante sea el uso de `busca/3`, que permite hacer más compacta la representación de las tablas, al eliminar literales repetidos, así como reducir el tiempo de búsqueda de literales complementarios. Se trata de una aportación propia que empleamos en primer lugar para mejorar la eficiencia de leanT^AP con fórmulas proposicionales, y que ahora hemos incorporado a esta implementación.

3.3. Abducción mediante tablas semánticas

Antes de presentar el método que Marta Cialdea, Fiora Pirri [CP93] y Atocha Aliseda [Ali97] proponen para encontrar soluciones abductivas mediante tablas semánticas, veamos cómo se traducen, al cálculo de tablas semánticas, las condiciones que hacen que $\langle \Theta, \phi \rangle$ sea un problema abductivo y que α sea una solución abductiva explicativa para dicho problema.

En primer lugar, para que $\langle \Theta, \phi \rangle$ sea un problema abductivo, deben darse las siguientes condiciones:

1. Que $\Theta \nvDash \phi$, es decir, que ϕ no sea consecuencia lógica de Θ. Esto equivale a que $\Theta \cup \{\neg\phi\}$ sea satisfactible, lo que por el corolario 3.15 supone que la tabla semántica de $\Theta \cup \{\neg\phi\}$ debe ser abierta.

2. Que $\Theta \nvDash \neg\phi$ que, por el mismo razonamiento que en el caso anterior, equivale a que la tabla semántica de $\Theta \cup \{\phi\}$ sea abierta.

Además, para que α sea una solución abductiva explicativa al problema abductivo $\langle \Theta, \phi \rangle$, se deben cumplir:

1. El *requisito fundamental*, $\Theta, \alpha \vDash \phi$, que en el cálculo de tablas semánticas equivale a que la tabla de $\Theta \cup \{\neg\phi, \alpha\}$ sea cerrada (corolario 3.16).

2. El *requisito de consistencia* de la explicación con la teoría, es decir, $\Theta, \alpha \nvDash \bot$, que exige que la tabla semántica de $\Theta \cup \{\alpha\}$ sea abierta.

3. El *requisito explicativo*, que exige $\alpha \nvDash \phi$, es decir, que la tabla de $\{\alpha, \neg\phi\}$ sea abierta.

Repasando las condiciones anteriores es fácil observar que la abducción mediante tablas semánticas incluirá los dos siguientes pasos:

1. **Constatación del problema abductivo**, que pasa por comprobar que la tabla de $\Theta \cup \{\neg\phi\}$ sea abierta, así como la de $\Theta \cup \{\phi\}$. En lo sucesivo, nos centraremos sobre todo en comprobar lo primero ya que, como veremos, la tabla resultante nos encaminará hacia la solución del problema. Además, si fuera el caso de que la tabla de $\Theta \cup \{\phi\}$ fuera cerrada, es decir, si es el caso de que $\Theta \vDash \neg\phi$, no es posible ninguna α que sea solución consistente, pues si α es una solución abductiva, entonces $\Theta, \alpha \vDash \phi$, pero como $\Theta \vDash \neg\phi$, por la monotonía de la relación de consecuencia lógica clásica tenemos que $\Theta, \alpha \vDash \neg\phi$, con lo que $\Theta \cup \{\alpha\}$ no puede ser consistente, al implicar lógicamente tanto ϕ como $\neg\phi$. De modo que no necesitamos comprobar si la tabla de $\Theta \cup \{\phi\}$ es abierta, pues de ser cerrada, nos daríamos cuenta al no hallar soluciones abductivas consistentes. Así se ahorra un paso prescindible en el proceso abductivo.

2. **Búsqueda de soluciones explicativas**, que consiste en encontrar una fórmula —conjunción de literales— α tal que haga cerrada la tabla semántica de $\Theta \cup \{\alpha, \neg\phi\}$ pero deje abiertas otras dos tablas —las tablas de α tanto con Θ, a efectos de probar la consistencia de la explicación, como con $\neg\phi$, para probar que α es explicativa—.

Tal como hemos descrito el procedimiento, puede dar la impresión de que se requiere la realización de muchas tablas semánticas. Sin embargo, es posible ahorrar alguna de ellas mediante el uso de las *extensiones de tablas* y de los *conjuntos de cierre*, tal como veremos un poco más adelante. Primero, comenzaremos definiendo estos conjuntos.

En lo sucesivo emplearemos notación conjuntista para referirnos a las tablas semánticas, tal como hicimos en la implementación que comentamos en la sección anterior. Así, la tabla $T = \{R_1, R_2, \ldots, R_n\}$ se representa como el conjunto de sus ramas abiertas R_i, $1 \leq i \leq n$. Cada rama $R_i = \{\lambda_{i_1}, \ldots, \lambda_{i_k}\}$ se representa como el conjunto de sus literales λ_{i_j}, $1 \leq j \leq k$. Además, con $\overline{\lambda_{i_j}}$ nos referimos al literal complementario de λ_{i_j}.

Definición 3.18 (Cierres totales de rama) *El conjunto de* cierres totales de una rama $R = \{\lambda_1, \ldots, \lambda_n\}$ *es el conjunto de los literales complementarios de* R*, es decir,*

$$CTR(R) = \{\overline{\lambda_1}, \ldots, \overline{\lambda_n}\}$$

Definición 3.19 (Cierres totales de tabla) *El conjunto de* cierres totales de una tabla $T = \{R_1, \ldots, R_n\}$ *es la intersección de los conjuntos de* cierres totales *de cada una de sus ramas, es decir,*

$$CTT(T) = \bigcap_{i=1}^{n} CTR(R_i)$$

Definición 3.20 (Cierres parciales de rama) *Dada la tabla semántica* $T = \{R_1, \ldots, R_n\}$*, el conjunto de* cierres parciales de una rama R_i, $1 \leq i \leq n$*, es el conjunto de literales que pertenecen a los cierres totales de* R_i *pero no a los de* T*, es decir,*

$$CPR(R_i) = CTR(R_1) - CTT(T)$$

Definición 3.21 (Cierres parciales de tabla) *Dada la tabla semántica* $T = \{R_1, \ldots, R_n\}$*, el conjunto de los* cierres parciales de la tabla T *es la unión de los conjuntos de cierres parciales de cada una de sus ramas, es decir,*

$$CPT(T) = \bigcup_{i=1}^{n} CPR(R_i)$$

Definición 3.22 (Extensión de una tabla) *Dada* $T = \{R_1, \ldots, R_n\}$*, una tabla semántica, y un conjunto de literales* $C = \{\lambda_1, \ldots, \lambda_m\}$*, definimos la* extensión de la tabla T *con los literales de* C *de la siguiente manera*

$$EXT(T, C) = \{R_i \cup C \mid R_i \cap \{\overline{\lambda_j} \mid \lambda_j \in C, 1 \leq j \leq m\} = \emptyset, 1 \leq i \leq n\}$$

Además, si $|EXT(T, C)| = |T|$*, decimos que* $EXT(T, C)$ *es una* extensión abierta *de* T*. Si* $|EXT(T, C)| = 0$*, decimos que* $EXT(T, C)$ *es una* extensión cerrada *de* T*. Y si* $0 < |EXT(T, C)| < |T|$*, decimos que* $EXT(T, C)$ *es una* extensión semicerrada *de* T*.*

Veamos cuál es el sentido de los conjuntos de cierre y de las extensiones de tablas, mientras comentamos, de modo informal, el procedimiento de abducción mediante tablas semánticas, basándonos en la explicación de Aliseda [Ali97]. Ilustraremos el proceso con un ejemplo, donde solucionaremos el problema abductivo $\langle \Theta, \phi \rangle$ para

$$\begin{aligned} \Theta &\equiv \{p \rightarrow q, t \wedge s \rightarrow p\} \\ \phi &\equiv q \end{aligned}$$

El primer paso es realizar la tabla semántica de la teoría, que debe ser abierta, pues de otro modo la teoría no sería satisfactible. En nuestro ejemplo, la tabla de la teoría se muestra en la figura 3.4. La tabla tiene cinco ramas abiertas, de forma que su representación conjuntista sería

$$\{\{\neg p, \neg t\}, \{\neg p, \neg s\}, \{q, \neg t\}, \{q, \neg s\}, \{q, p\}\}$$

El siguiente paso es extender la tabla de la teoría con el conjunto que contiene la negación del literal que quiere explicarse, en este caso $\{\neg q\}$. Intuitivamente,

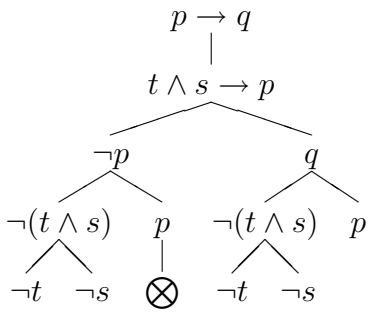

Figura 3.4: Tabla de la teoría

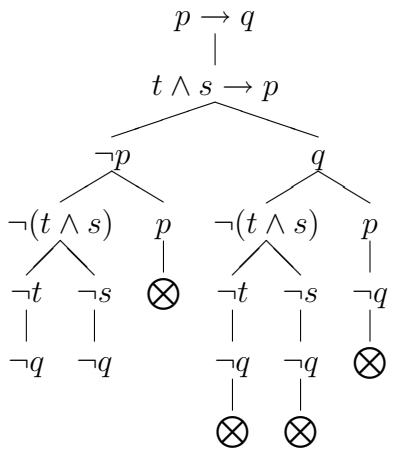

Figura 3.5: Tabla extendida

la extensión de una tabla no es más que añadir literales a sus ramas abiertas y cerrar las que proceda, como hemos hecho en la figura 3.5. De las cinco ramas abiertas de la tabla de la teoría se han cerrado tres.

De un modo más formal, siguiendo la definición 3.22, tenemos

$$EXT(\{\{\neg p, \neg t\}, \{\neg p, \neg s\}, \{q, \neg t\}, \{q, \neg s\}, \{q, p\}\}, \{\neg q\}) = \\ \{\{\neg p, \neg t, \neg q\}, \{\neg p, \neg s, \neg q\}\}$$

Recapitulando, hemos hecho la tabla de la teoría —que debe ser abierta— y la hemos extendido con la negación del literal que queremos explicar. Es fácil observar que el resultado de extender una tabla T con el conjunto de literales C es igual que hacer la tabla de $T \cup C$. Sin embargo, mediante las extensiones de tablas no sólo ahorramos el hacer dos tablas, sino que obtenemos más información según sea la extensión de un tipo u otro, tal como veremos a continuación.

Adelantamos que las soluciones abductivas serán conjuntos de literales —implícitamente formando una conjunción— tales que si la tabla de la teoría extendida con la negación de la observación vuelve a extenderse con tales literales, el resultado es una tabla cerrada. En nuestro ejemplo, se tratará de conjuntos de literales con los que extender la tabla de la figura 3.5 de forma que se cierre.

En la definición 3.22 hemos distinguido tres tipos de extensiones de tablas: *abiertas*, *cerradas* y *semicerradas*. Para que sea posible encontrar soluciones abductivas explicativas con forma *atómica* o *conjuntiva*, la extensión de la tabla de la teoría con la negación de la observación debe ser *semicerrada*. Veamos por qué. En primer lugar, si la extensión es cerrada, entonces, por el corolario 3.16 tenemos que —siendo Θ la teoría y ϕ la observación— $\Theta \cup \{\neg\phi\}$ no es satisfactible, por lo que $\Theta \models \phi$. Además, si la extensión es abierta, toda fórmula α que cierre la tabla extendida será una conjunción de los literales $\{\lambda_1, \ldots, \lambda_n\}$, $1 \leq n$ —si $n = 1$ la explicación será atómica—, que cierren las ramas de la tabla extendida $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$. Sin embargo, por ser esta extensión abierta, sus ramas son las mismas que las de $\mathcal{T}(\Theta)$ (definición 3.22) pero con el literal $\neg\phi$. De forma que para que los literales $\{\lambda_1, \ldots, \lambda_n\}$ cierren $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$ caben dos posibilidades:

- Que entre ellos se encuentre ϕ. En tal caso se cerrarían todas las ramas de $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$ con el par de literales complementarios ϕ, $\neg\phi$. Pero una conjunción de literales —la solución α— que contuviese ϕ no puede ser una abducción explicativa, así que no nos interesa.
- Que entre ellos no esté ϕ. Entonces los literales de $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$ con los que se producen los cierres al extenderse con $\{\lambda_1, \ldots, \lambda_n\}$ son literales que también estaban en $\mathcal{T}(\Theta)$, por lo que la extensión de la tabla de la teoría con los literales de α, $EXT(\mathcal{T}(\Theta), \{\lambda_1, \ldots, \lambda_n\})$ sería cerrada, lo que significa que en tal caso $\Theta \cup \{\alpha\}$ no es satisfactible, con lo que no serían posibles las abducciones consistentes.

Por tanto, la extensión de la tabla de la teoría con la negación de la observación debe ser *semicerrada*, como lo es en nuestro ejemplo la tabla extendida de la figura 3.5. Debemos aclarar que si aceptamos soluciones abductivas diferentes a las explicativas —o bien que no sean conjunciones de literales— puede interesar otro tipo de extensiones, tal como explica Aliseda [Ali97].

Una vez que tenemos la tabla de la teoría, que se ha hecho la extensión con la negación de la observación, y que se ha constatado que ésta es *semicerrada*, podemos buscar soluciones abductivas. Como ya hemos adelantado, nos centraremos en dos tipos: *atómicas* —un solo literal— y *conjuntivas* —una conjunción de literales—.

En cuanto a las explicaciones *atómicas*, se requiere que α sea un literal de modo que la tabla de la teoría Θ extendida con la negación de la observación ϕ, es decir $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$ se cierre al volver a extenderla con α, o sea, que debe ocurrir

$$EXT(EXT(\mathcal{T}(\Theta), \{\neg\phi\}), \{\alpha\}) = \emptyset$$

Pero si α debe cerrar todas las ramas de la tabla extendida, tenemos que, por las definiciones 3.18 y 3.19,

$$\alpha \in CTT(EXT(\mathcal{T}(\Theta), \{\neg\phi\}))$$

es decir, que α debe formar parte del conjunto de cierres totales de la tabla extendida, pues sólo así puede cerrar todas sus ramas. Además, para que α sea

una explicación consistente, no debe cerrar la tabla de la teoría Θ, por lo que no debe pertenecer al conjunto de cierres totales de la tabla de Θ. Por ello, conviene que pertenezca al conjunto de sus cierres parciales, de modo que

$$\alpha \in CPT(\mathcal{T}(\Theta))$$

Finalmente, para que α sea una abducción explicativa, no debe ser ϕ. Por tanto, el conjunto de explicaciones atómicas es el formado por todos los literales α que cumplan

$$\alpha \in (CTT(EXT(\mathcal{T}(\Theta), \{\neg\phi\})) \cap CPT(\mathcal{T}(\Theta))) - \{\phi\}$$

En nuestro ejemplo, tenemos que:

$$\begin{aligned} CPT(\mathcal{T}(\Theta)) &= \{p, t, s, \neg p, \neg q\} \\ CTT(EXT(\mathcal{T}(\Theta), \{\neg\phi\})) &= \{p\} \end{aligned}$$

por lo que la única explicación atómica a nuestro problema es el literal p.

En cuanto a las explicaciones *conjuntivas* de tipo $\lambda_1 \wedge \lambda_2 \wedge \ldots \wedge \lambda_n$, $n > 1$, cada rama de $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$ debe cerrarse con alguno de los λ_i, $1 \leq i \leq n$. Además, ninguno de tales λ_i debe cerrar por completo la tabla extendida. Por tanto, uno de los requisitos será que para todas las ramas

$$R \in EXT(\mathcal{T}(\Theta), \{\neg\phi\})$$

ocurra

$$CPR(R) \cap \{\lambda_1, \lambda_2, \ldots, \lambda_n\} \neq \emptyset$$

Una de las formas de conseguir esto —es la que vamos a proponer en la implementación— es formar todos los conjuntos $C_1, C_2, \ldots$ diferentes posibles que contienen un literal de los cierres parciales de cada una de las ramas de la tabla extendida y después eliminar los literales repetidos de cada C_i, así como los C_i no minimales, de decir, que exista otro C_j tal que

$$C_j \subset C_i$$

A efectos de la consistencia de las explicaciones conjuntivas, de los C_i resultantes sólo podemos elegir aquellos que no cierren la tabla de la teoría. Para ello basta comprobar que existe alguna rama R de $\mathcal{T}(\Theta)$ tal que

$$CTR(R) \cap C_i = \emptyset$$

Los C_i que no cumplan este requisito no son explicaciones consistentes. En cuanto al carácter explicativo de las abducciones conjuntivas, queda asegurado, ya que el literal ϕ nunca podrá pertenecer a los cierres parciales de ninguna rama de la tabla de la teoría extendida $EXT(\mathcal{T}(\Theta), \{\neg\phi\})$, pues al formar $\neg\phi$ parte de cada rama de dicha tabla, se encuentra entre sus cierres totales.

En nuestro ejemplo tenemos que la tabla extendida (figura 3.5) tiene dos ramas abiertas: $\{\{\neg p, \neg t, \neg q\}, \{\neg p, \neg s, \neg q\}\}$. El conjunto de cierres parciales de la primera rama es $\{t\}$, y el de la segunda, $\{s\}$. De modo que sólo hay un posible conjunto candidato a explicación conjuntiva: $\{t, s\}$. Además, para asegurar la consistencia de la explicación debemos comprobar que en la tabla de la teoría (figura 3.4) haya una rama R tal que $CTR(R) \cap \{t, s\} = \emptyset$. Tal rama es la última de la tabla —a contar desde izquierda a derecha—, $\{q, p\}$, cuyo conjunto de cierres totales, $CTR(\{q, p\} = \{\neg q, \neg p\})$.

Como puede observarse, este procedimiento depende demasiado de que la observación que quiera explicarse sea un literal. De otra forma, habría que definir una operación de extensión de tablas que contemplara la posibilidad de que las tablas extendidas —$EXT(\mathcal{T}(\Theta), \{\neg\phi\})$— tuviesen más ramas abiertas que las tablas que extienden —$\mathcal{T}(\Theta)$—, con lo que la definición de los tipos de extensiones (definición 3.22) habría que modificarla también y con ello todo el proceso abductivo.

Aliseda[Ali97] también explica cómo generar *explicaciones disyuntivas*, es decir, disyunciones de literales. No entraremos en más detalles que decir que considera explicaciones disyuntivas la disyunción de:

- Dos explicaciones atómicas.
- Dos explicaciones conjuntivas.

- Una explicación atómica y otra conjuntiva.
- Una explicación atómica y ϕ —la observación—.
- Una explicación conjuntiva y ϕ.

3.4. Implementación de la abducción con tablas semánticas

En la página 265 ofrecemos una implementación del procedimiento abductivo comentado en la sección anterior. Hemos partido del programa que ofrece Aliseda en el apéndice A de [Ali97], aunque lo hemos adaptado para que produzca abducciones explicativas, y sólo las atómicas y conjuntivas. Igualmente, le hemos incorporado ciertas optimizaciones, como la construcción de tablas semánticas según lo explicado en la sección 3.2, que hacen el código bastante más eficiente.

El predicado que empleamos para hacer las pruebas con los problemas del apéndice C es `abduce_estad(+Teo, +Obs)`, que dada la teoría `Teo` —conjunción de fórmulas proposicionales— y la observación `Obs` —un literal—, comienza un proceso que terminará ofreciendo la solución al problema abductivo propuesto. Además, devuelve información del tiempo y la memoria —pila global— que ha consumido, así como del número de inferencias que ha realizado. A continuación comentamos cada paso de este proceso, mientras vamos mostrando la salida en pantalla que produce la resolución del mismo problema abductivo que analizamos en la sección anterior.

```
?- abduce_estad((p => q) & (t & s => p), q).
   Teoría: (p=>q)& (t&s=>p)
   Observación: q
```

El primer paso es realizar la tabla de la teoría, que en la salida se llama `TabTeo`, y extenderla con el literal complementario de `Obs`, produciendo la tabla extendida `TabExt`.

```
Tabla de la teoría (TabTeo):
   [-t, -p], [-s, -p], [-t, q], [-s, q], [p, q]

Complementario del hecho: -q

Tabla extendida (TabExt):
   [-q, -t, -p], [-q, -s, -p]
```

Ocurre entonces uno de los tres casos siguientes:

Extensión cerrada. Entonces el programa avisa de que la extensión es cerrada, y que por tanto la observación está explicada por la teoría, con lo que termina el proceso abductivo. Obsérvese que esto también puede ocurrir si la teoría fuera inconsistente, pues, por una cuestión de eficiencia, al hacer `TabTeo` no se comprueba si es o no cerrada.

Extensión abierta. En este caso, el programa avisa de que no son posibles las explicaciones consistentes. Como se explicó anteriormente, serían posibles si buscamos abducciones disyuntivas, pero no es el caso de nuestro programa. Se comprueba si la extensión es abierta —en esto hacemos como Aliseda— comparando las cardinalidades de `TabTeo` y `TabExt`, que si son iguales nos indican que estamos ante una extensión abierta.

Extensión semicerrada. Este es el caso de nuestro ejemplo. Entonces se hacen todos los conjuntos de cierre —*parciales* y *totales*, de *ramas* y de *tablas*— necesarios para resolver el problema abductivo, y se muestran en pantalla.

```
Cierres totales de ramas de TabTeo:
   [p, t], [p, s], [t, -q], [s, -q], [-p, -q]

Cierres totales de tabla de TabTeo:

Cierres parciales de ramas de TabTeo:
   [p, t], [p, s], [t, -q], [s, -q], [-p, -q]

Cierres parciales de tabla de TabTeo:
   -p, -q, s, p, t
```

```
Cierres totales de ramas de TabExt:
   [p, q, t], [p, q, s]

Cierres totales de tabla de TabExt:
   p, q

Cierres parciales de ramas de TabExt:
   [t], [s]
```

Por último, sólo resta construir las explicaciones. Las explicaciones atómicas se obtienen con `explicaciones_atómicas(+CTTabExt, +CPTabTeo, -Explic)` que, dados los conjuntos `CTTabExt` de cierres totales de `TabExt` y `CPTabTeo` de cierres parciales de `TabTeo`, devuelve `Explic`, conjunto de explicaciones atómicas. La forma en que obtiene `Explic` es por la intersección entre ambos conjuntos, como se explicó en la sección anterior. En el procedimiento principal se eliminará `Obs` de `Explic` —en caso de que esté— para no devolverlo como abducción trivial no explicativa.

Las explicaciones conjuntivas se construyen con el predicado `explicaciones_conjuntivas(+CPRTabExt, +CTRTabTeo, -Explic)` que devuelve `Explic`, el conjunto de explicaciones conjuntivas, siendo `CPRTabExt` el conjunto que contiene, para cada rama de `TabExt`, el conjunto de sus cierres parciales, y `CTRTabTeo` el conjunto que contiene, para cada rama de `TabTeo`, el conjunto de sus cierres totales. Como se explicó en la sección anterior, `Explic` se obtiene:

1. Haciendo todas las combinaciones posibles que toman un literal de cada uno de los conjuntos de `CPRTabExt` y eliminando los repetidos y no minimales. Los predicados principales en este paso son `es_explicación_conjuntiva/2` y `quita_repet/2`.

2. Seleccionando, de las combinaciones obtenidas, sólo aquellas para las que exista al menos un elemento de `CTRTabTeo` —el conjunto de cierres totales de alguna rama de `TabTeo`— con el que no comparta ningún literal. El predicado usado en este paso es `exp_consistente/2`.

```
Las explicaciones atómicas son:
```

```
   p

Las explicaciones conjuntivas son:
   s & t
```

Finalmente, se muestran los recursos empleados para la resolución del problema abductivo.

```
RECURSOS EMPLEADOS:
   Tiempo (milisegundos):  0
   Espacio (bytes):        2532
   Inferencias:            763
```

	Instancia	*Milisegundos*	*Bytes*	*Inferencias*
implic	5	0	2640	751
	10	11	8900	2811
	15	10	18860	6496
conj	3	0	7748	1564
	4	—	—	—
	5	—	—	—
	6	—	—	—
	7	—	—	—
anidado	3	10	13768	2520
	4	—	—	—
	5	—	—	—
incon	1	0	22632	3402
	3	671	2523056	247275
	4	—	—	—
	5	—	—	—

Cuadro 3.1: Resultados para el sistema de Aliseda

En el cuadro 3.1 se muestran los recursos que con esta implementación han sido necesarios para solucionar cada uno de los problemas del apéndice C. En algunos problemas la pila global, de 25 megabytes, no fue suficiente, por lo que el programa no pudo solucionarlo. A ello se deben los huecos que aparecen en algunos problemas.

Los problemas más sencillos de resolver mediante tablas semánticas son los de la clase **implic**. Si acudimos al apéndice C y vemos las fórmulas que componen estos problemas, nos damos cuenta enseguida de que la tabla de la teoría extendida con la negación de la observación sólo tiene una rama abierta, con los literales $\{\neg t_1, \neg t_2, \ldots, \neg t_n\}$. Sin embargo, para las demás familias —para las que este sistema fracasa— las tablas extendidas tienen más ramas abiertas.

La eficiencia de este sistema depende tanto del número de ramas abiertas que tengan las tablas debido a la cantidad de operaciones conjuntistas que deben realizarse tanto para la creación de los conjuntos de cierre como para la generación de las explicaciones, así como para la comprobación de la consistencia de cada una de ellas.

Resulta paradigmático lo que ocurre con los problemas **incon**. La instancia primera se resuelve en 0 milisegundos —casi instantáneamente—, con sólo 3402 inferencias y empleando nada más que cerca de 23 kilobytes de memoria de la pila global. Sin embargo la tercera instancia, que sólo tiene dos literales más, requiere para resolverse más de medio segundo y casi un cuarto de millón de inferencias, así como algo más de 2 megabytes y medio de memoria de la pila global. La instancia siguiente —con sólo un literal más en la teoría— no puede ser completada por agotarse los 25 megabytes de la pila global.

Capítulo 4

El cálculo de δ-resolución

En este capítulo presentamos el cálculo de δ-resolución, que constituye el núcleo de este trabajo. Se trata de un cálculo dual a la resolución con el que podremos resolver problemas abductivos, como veremos, de forma *directa*, ahondando en la dualidad *deducción-abducción*. En la primera sección exponemos las motivaciones que nos han llevado a definir este cálculo, así como una introducción informal, de manera que pueda apreciarse la relación que existe entre los procesos de δ-resolución y ciertos razonamientos que hacemos, si no cuando nuestra mente trata de explicarse cierto fenómenos, al menos sí cuando tratamos de expresar y justificar las hipótesis que nuestro sentido común ha creado. La segunda sección se dedica al estudio formal del cálculo de δ-resolución proposicional, la presentación de los teoremas fundamentales y las relaciones —de dualidad— con la resolución clásica. La tercera sección aborda el problema de la generación de abducciones explicativas mediante δ-resolución, y aporta un primer método, que en el capítulo siguiente será refinado. Por último, la cuarta sección se dedica a presentar la primera implementación que hemos hecho del cálculo de δ-resolución, y a estudiar su eficiencia con los problemas del apéndice C, en comparación con el sistema de Aliseda.

4.1. Introducción

En el capítulo anterior, al comentar los resultados de las pruebas de eficiencia —cuadro 3.1— observamos que el sistema de Cialdea, Pirri [CP93] y Aliseda [Ali97] acusa un problema de complejidad excesiva, debido al gran número de operaciones conjuntistas que su ejecución requiere. Asimismo, también hemos comentado otra limitación de expresividad, pues al sólo poder explicarse literales, se dejan de lado cada vez más aplicaciones que la abducción encuentra en Inteligencia Artificial, como ciertos problemas de diagnosis o planificación donde lo que debe explicarse —el comportamiento anómalo de cierto dispositivo, en un problema de diagnosis; la descripción del estado *objetivo*, en un problema de planificación— no es un literal, sino alguna fórmula más compleja como una conjunción de literales.

Cuando, en 1993, Marta Cialdea y Fiora Pirri proponen el método de abducción a través de tablas semánticas, lo que pretenden es dotar a la abducción de un tratamiento formal que no acaba de establecerse de forma general —prácticamente aún hoy día en cada trabajo que aparece publicado sobre abducción se define qué se entiende por *problema abductivo*, así como por *solución abductiva*; definiciones que no suelen coincidir entre diferentes autores—. Ellas entendían que en Inteligencia Artificial se habían llevado a cabo numerosos trabajos prácticos, a nivel de implementación, especialmente en el marco resolutivo de la Programación Lógica, pero en cambio no se había realizado aún un tratamiento general fuera de este contexto.

Si bien es verdad que sigue siendo necesaria una caracterización formal del razonamiento abductivo que pueda ser aceptada mayoritariamente, no es menos cierto que las cuestiones de complejidad y eficiencia han llegado a ser tan importantes que merecen igualmente atención desde un punto de vista lógico. Así, si es preciso un tratamiento formal de la abducción, éste no será aceptado si no muestra ciertos mínimos de eficiencia, así como una expresividad que permita explicar observaciones que no necesariamente se restrinjan a literales.

Este es el planteamiento con el que iniciamos nuestra investigación, y pensamos que la aportación del cálculo de δ-resolución permite —aunque no sea de

forma definitiva— dotar a la abducción de un cálculo lógico que no olvida ni la eficiencia ni la expresividad. En cierto sentido nos situamos a medio camino entre el enfoque de las autoras citadas y el de la Programación Lógica; de hecho el cálculo de δ-resolución es dual al de resolución, con lo que por un lado mantiene la eficiencia del cálculo de resolución —que con unas u otras variantes es el más implementado en Demostración Automática— y por otro permite generar directamente las hipótesis explicativas —no como los clásicos enfoques basados en resolución, o incluso la abducción por tablas semánticas, en que las explicaciones se encuentran negando los resultados que se obtienen por uno u otro procedimiento—. Además, al emplear un cálculo basado en resolución pensamos que sería posible integrar la abducción con ciertas formas de razonamiento por analogía, otra importante forma de razonamiento explicativo, de modo parecido a como Christophe Bourely, Gilles Défourneaux y Nicolas Peltier hacen en [BDP96].

Comenzaremos mostrando el punto de partida que tomamos para diseñar el cálculo de δ-resolución. Dado el problema abductivo $\langle \Theta, \phi \rangle$, $\Theta = \{\theta_1, \ldots, \theta_n\}$, decimos que α es un solución abductiva[1] si se cumple

$$\Theta, \alpha \models \phi \tag{4.1}$$

Ahora bien, todos los tratamientos formales de la abducción que hemos encontrado —tanto los basados en resolución como en tablas semánticas— parten de considerar la equivalencia de lo anterior con

$$\Theta, \alpha, \neg\phi \models \bot \tag{4.2}$$

o bien

$$\Theta, \neg\phi \models \neg\alpha \tag{4.3}$$

con lo que la resolución del problema abductivo consiste en obtener consecuencias lógicas de $\Theta \cup \{\neg\phi\}$ y negarlas para obtener α.

[1] Se trata del *requisito fundamental* —definición 2.23—, suficiente para generar abducciones *planas*. En este capítulo daremos una primera forma —simple pero no demasiado eficiente— de elegir las abducciones explicativas de entre las que se generan por δ-resolución. En el siguiente capítulo mostramos cómo integrar la generación de abducciones *explicativas* dentro de proceso de δ-resolución.

Sin embargo, la condición (4.1) también equivale, por el teorema de la deducción, a

$$\alpha \vDash \theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi \tag{4.4}$$

con lo que la solución del problema abductivo puede verse como una búsqueda de modelos de $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$. Si queremos que α sea una conjunción *minimal* de literales, se tratará de encontrar un conjunto *minimal* de literales tales que todo modelo que los satisfaga, satisfaga igualmente a $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$.

Si comparamos las relaciones (4.2) y (4.4) observamos que en el planteamiento tradicional —mostrado en (4.2)— se trata de encontrar una fórmula α tal que el conjunto de fórmulas $\Theta \cup \{\alpha, \neg\phi\}$ no sea satisfactible. Entonces, el conjunto $\Theta \cup \{\alpha\}$ no será consistente con $\neg\phi$, con lo que se verificará $\Theta, \alpha \vDash \phi$. Se trata de un procedimiento que opera de forma *indirecta* o *inversa*, similar al razonamiento por reducción al absurdo, pues α hace imposible —dentro de la teoría— $\neg\phi$, con lo que ϕ será necesaria. Sin embargo, en el planteamiento de partida de la δ-resolución —que aparece en (4.4)—, se busca una fórmula α que haga que la teoría Θ implique ϕ, con lo que el proceder es más *directo*. Digamos que los planteamientos tradicionales explican "no $\neg\phi$", mientras que mediante δ-resolución se explica "ϕ". ¿Son ambos planteamientos idénticos? La cuestión, desde un punto de vista formal y en lógica clásica, está cerrada, pues las relaciones en que se basan son equivalentes. Sin embargo, si nos preguntamos qué recoge mejor el proceder del razonamiento de sentido común, la respuesta es menos cierta. Desde

1	Θ	Teoría
2	$\neg\phi$	Negación de la observación
$\vdots$	$\vdots$	$\vdots$
n	$\neg\alpha$	Consecuencia de 1 y 2
n+1	α	Conclusión abductiva

Cuadro 4.1: Esquema de razonamiento explicativo *inverso*

luego, parece muy poco natural el esquema de razonamiento que hemos recogido en el cuadro 4.1, que interpreta el proceder de los planteamientos tradicionales de la abducción. Por ello, nos resulta preferible el modo *directo* de generación de hipótesis explicativas.

Para ilustrar cómo se resuelven problemas abductivos mediante δ-resolución, veamos, de modo informal, la solución mediante δ-resolución a un problema abductivo típico, presentado por Kakas, Kowalski y Toni [KKT98]. El lenguaje consta de las siguientes variables proposicionales

$$\begin{aligned} a &= \text{los aspersores regaron anoche} \\ l &= \text{llovió anoche} \\ c &= \text{el césped está mojado} \\ z &= \text{los zapatos se mojan} \end{aligned}$$

Lo que quiere explicarse es z, a partir de la teoría

$$(a \rightarrow c) \wedge (l \rightarrow c) \wedge (c \rightarrow z)$$

Por tanto, la relación que presentamos en (4.4), para este problema, queda

$$\alpha \vDash (a \rightarrow c) \wedge (l \rightarrow c) \wedge (c \rightarrow z) \rightarrow z$$

con lo que habrá que buscar modelos para

$$(a \rightarrow c) \wedge (l \rightarrow c) \wedge (c \rightarrow z) \rightarrow z \tag{4.5}$$

Un procedimiento para encontrar modelos de (4.5) es construir su forma normal disyuntiva

$$(a \wedge \neg c) \vee (l \wedge \neg c) \vee (c \wedge \neg z) \vee z \tag{4.6}$$

Por tanto, todo conjunto de literales del que (4.6) sea consecuencia lógica, será una solución abductiva. Desde luego, cada una de las conjunciones elementales de (4.6) valdrían como explicaciones, según este criterio. Por ejemplo, valen $c \wedge \neg z$ y z. Pero si valen éstas, también servirá c, ya que $c \vDash z \vee (c \wedge \neg z)$, pues todo modelo que satisfaga c, o bien satisface $\neg z$, con lo que satisface la explicación válida $c \wedge \neg z$, o bien satisface z que también es una explicación válida. Por la misma razón, al ser (4.6) consecuencia lógica de c —según acabamos de probar—, como también lo es de $l \wedge \neg c$ —por ser una de sus conjunciones elementales—, lo será igualmente de l, ya que $l \vDash c \vee (l \wedge \neg c)$. Del mismo modo, puesto que $a \vDash c \vee (a \wedge \neg c)$, al ser c una explicación válida y $a \wedge \neg c$ una conjunción elemental de (4.6), tenemos que a es otra explicación válida. Recogemos este razonamiento en el cuadro 4.2. Si lo comparamos con el cuadro 4.1, observamos que el proceder

1.	$a \wedge \neg c$	Conjunción elemental de (4.6)
2.	$l \wedge \neg c$	Conjunción elemental de (4.6)
3.	$c \wedge \neg z$	Conjunción elemental de (4.6)
4.	z	Conjunción elemental de (4.6)
5.	c	Desde 4 y 3
6.	l	Desde 5 y 2
7.	a	Desde 5 y 1

Cuadro 4.2: Esquema de razonamiento explicativo *directo*

directo resulta mucho más natural. Razonando de manera inversa, se partiría de la teoría con $\neg z$ —negación de la observación— para obtener, como consecuencia, los literales complementarios a los que en el cuadro 4.2 alcanzamos de manera directa.

Cada uno de los literales 4–7 del cuadro 4.2 es una solución abductiva plana al problema abductivo planteado. Como puede apreciarse, la δ-resolución —que ahora sólo hemos mostrado de modo informal— es completamente dual a la resolución. Comenzamos transformando lo que quiere explicarse —la teoría implica la observación— a forma normal disyuntiva —mientras que en la resolución se emplea forma normal conjuntiva— y a partir de ahí procedemos por una regla dual a la de resolución —como puede observarse, por ejemplo, en la inferencia de 5 a partir de 3 y 4 en el cuadro 4.2, justificada, como vimos antes, porque $c \vDash z \vee (c \wedge \neg z)$—, de manera que lo que obtenemos son, no consecuencias lógicas de las premisas, sino fórmulas de las que las fórmulas de partida son consecuencia lógica. De modo que conseguimos realizar abducción de modo directo, por lo que podremos hablar de la δ-resolución como de un auténtico *cálculo abductivo*.

La dualidad con la resolución va a suponer ventajas en varios aspectos. Por un lado, las propiedades de la δ-resolución serán duales —así como los teoremas, demostraciones, etc— a las de la resolución. Además, dispondremos de la posibilidad de encontrar implementaciones eficientes —la complejidad será la misma que la de la resolución, así como las estrategias de búsqueda, eliminación de cláusulas subsumidas, etc—. Pero también la propia resolución nos ofrece ciertos elementos que pueden interpretarse en clave explicativa. Por ejemplo, la *historia* de las cláusulas se puede convertir en un muy buen criterio preferencial a la hora de

elegir la mejor explicación[2]. Así, de entre los literales 4–7 del cuadro 4.2 los que tienen mayor historia son 6 y 7, que coinciden con las abducciones más fuertes, que llevan más lejos el razonamiento explicativo. Son las explicaciones últimas que la teoría permite pues, a diferencia del literal 5, los literales 6 y 7 no pueden ser a su vez explicados dentro de la teoría por otros literales diferentes.

Volveremos con mayor detenimiento sobre estas mismas ideas a lo largo de las siguientes secciones, así como en los próximos capítulos, donde desarrollamos formalmente los elementos del cálculo de δ-resolución, así como sus estrategias para hacer más eficiente la búsqueda de abducciones explicativas.

4.2. Presentación del cálculo de δ-resolución

En esta sección presentamos el cálculo de δ-resolución, así como sus teoremas fundamentales. Comenzamos por las definiciones de *δ-cláusula* y *forma δ-clausal*. Como puede apreciare, se trata de conceptos duales a los empleados en el cálculo de resolución.

Definición 4.1 (δ-cláusula) *Una δ-cláusula Σ es un conjunto de literales*

$$\Sigma = \{\lambda_1, \ldots \lambda_n\}$$

y equivale a la conjunción de literales

$$\lambda_1 \wedge \ldots \wedge \lambda_n,$$

de modo que dada una valoración v, v satisface Σ syss v satisface todos los λ_i, $1 \leq i \leq n$. En adelante, nos referiremos a las δ-cláusulas mediante letras griegas mayúsculas. Con $\square$ nos referimos a la δ-cláusula vacía, universalmente válida.

Definición 4.2 (Forma δ-clausal) *Una forma δ-clausal A es un conjunto de δ-cláusulas*

$$A = \{\Sigma_1, \ldots, \Sigma_n\}$$

[2]De hecho, en Programación Lógica Abductiva, aunque proceda según lo que hemos llamado el modo *tradicional*, expresado en la ecuación (4.2), también se suele emplear un criterio preferencial equivalente por el que se prefieren las explicaciones construidas a partir de *dead ends* más largos de las pruebas por resolución.

y equivale a la disyunción de sus δ-cláusulas, de forma que dada una valoración v, v satisface A syss v satisface al menos una de las δ-cláusulas Σ_i, $1 \leq i \leq n$. En adelante, nos referiremos a las formas δ-clausales mediante letras mayúsculas latinas. La forma δ-clausal vacía es no satisfactible[3].

Dada una fórmula proposicional β, si existe cierta forma δ-clausal B tal que $\models \beta \leftrightarrow B$, decimos que B es la forma δ-clausal de β.

Las definiciones anteriores nos permiten extender la relación de consecuencia lógica clásica para incluir δ-cláusulas y formas δ-clausales tanto a la izquierda como a la derecha de "$\models$". Así, por ejemplo, mediante $\Sigma \models A$ expresamos que toda valoración que satisfaga la δ-cláusula Σ satisface la forma δ-clausal A. Del mismo modo, podemos extender otras nociones como satisfactibilidad, contingencia y equivalencia. La transformación de una fórmula proposicional cualquiera a forma δ-clausal se hace igualmente de forma dual a la transformación a forma clausal. En vez de construir la *forma normal conjuntiva*, ahora hacemos la *forma normal disyuntiva*.

Teorema 4.3 (Conversión a forma δ-clausal) *Para toda fórmula proposicional $\phi \in \mathcal{L}_p$ existe una forma δ-clausal A tal que*

$$\models \phi \leftrightarrow A$$

Demostración:

A partir de una fórmula proposicional dada ϕ puede obtenerse una *forma normal disyuntiva* $FND(\phi)$ que tendrá la forma

$$(\delta_{1_1} \wedge \ldots \wedge \delta_{i_{j_i}}) \vee \ldots \vee (\delta_{n_1} \wedge \ldots \wedge \delta_{n_{j_n}}),$$

[3]Nótese que mientras que la δ-cláusula vacía $\square$ es universalmente válida, lo que concuerda con la definición 2.5 —puesto que una δ-cláusula se evalúa, por la definición 4.1, igual que un conjunto de fórmulas—, la forma δ-clausal vacía no es satisfactible. Se trata, pues, de dos conjuntos vacíos de diferente nivel, pues los elementos que integran las δ-cláusulas y las formas δ-clausales son diferentes —en el primer caso son literales; en el segundo, δ-cláusulas—. Obsérvese que una forma δ-clausal Σ no se evalúa igual que un conjunto de fórmulas, ya que para que una valoración v satisfaga Σ debe cumplirse que haya al menos una δ-cláusula $\gamma \in \Sigma$ que sea satisfecha por v. Esto nunca puede verificarse para la forma δ-clausal vacía, como resulta obvio por no contener ninguna δ-cláusula.

es decir, se trata de la disyunción de n conjunciones elementales, $n \geq 1$ —si $n = 1$ sólo hay una conjunción elemental; en otro caso, hay más—, cada una con la forma $(\delta_{k_1} \wedge \ldots \wedge \delta_{k_{j_k}})$, $1 \leq k \leq n$, $j_K \geq 1$ —si $j_k = 1$ la conjunción elemental sólo tiene un literal; en otro caso, tiene más de uno—, tal que cada δ_{k_l}, $1 \leq l \leq k_{j_k}$, es un literal. Las dos fórmulas ϕ y $FND(\phi)$ son equivalentes, de forma que dada una valoración v, v satisface ϕ syss v satisface $FND(\phi)$. Pero a su vez, podemos transformar $FND(\phi)$ en la forma δ-clausal

$$A = \{\Sigma_1, \ldots, \Sigma_n\}$$

tal que cada una de las Σ_k, $1 \leq k \leq n$, es una δ-cláusula

$$\{\delta_{k_1}, \ldots, \delta_{k_{j_k}}\}$$

Veamos ahora que $\vDash A \leftrightarrow FND(\phi)$. Supongamos, en primer lugar, que v satisface A. Entonces, por la definición 4.2, v satisface alguna de las δ-cláusulas Σ_k, y por la definición 4.1 ésto significa que v satisface todos los literales $\delta_{k_1}, \ldots, \delta_{k_{j_k}}$. Pero por evaluación del conjuntor, v satisface entonces

$$\delta_{k_1} \wedge \ldots \wedge \delta_{k_{j_k}}$$

y ésta es una de las conjunciones elementales de $FND(\phi)$, por lo que, por evaluación del disyuntor, v satisface $FND(\phi)$.

Igualmente, si v satisface $FND(\phi)$, ello significa, por evaluación del disyuntor, que v satisface alguna de las conjunciones elementales de $FND(\phi)$, y por evaluación del conjuntor, sabemos que v satisface todos los literales de dicha conjunción elemental. Pero tales literales son exactamente los que integran una de las δ-cláusulas de A, por lo que, por la definición 4.1, v satisface dicha δ-cláusula, y de ahí, por la definición 4.2, v satisface A.

Por tanto, si ϕ es equivalente a $FND(\phi)$ y ésta es equivalente a A, entonces ϕ es equivalente a A. De modo que toda fórmula proposicional puede transformarse a una forma δ-clausal equivalente. ■

La demostración del teorema 4.3 muestra el modo en que las fórmulas proposicionales se transforman a forma δ-clausal. Se trata de convertir cada una de las

conjunciones elementales de su forma normal disyuntiva en una δ-cláusula con exactamente los mismos literales.

La única regla que introduciremos es la regla de δ-resolución, que tiene igual forma sintáctica que la regla de resolución binaria, aunque semánticamente es dual.

Definición 4.4 (Regla de δ-resolución) *Dadas las δ-cláusulas $\Sigma_1 \cup \{\lambda\}$ y $\Sigma_2 \cup \{\neg\lambda\}$, la regla de δ-resolución se expresa de la siguiente forma:*

$$\frac{\Sigma_1 \cup \{\lambda\} \qquad \Sigma_2 \cup \{\neg\lambda\}}{\Sigma_1 \cup \Sigma_2}$$

Decimos que la cláusula $\Sigma_1 \cup \Sigma_2$ es un δ-resolvente de las dos primeras.

Definición 4.5 (Demostración por δ-resolución) *Una secuencia de δ-cláusulas es una* demostración mediante δ-resolución *de la δ-cláusula Λ a partir de las δ-cláusulas $\Sigma_1, \ldots, \Sigma_n$, lo que expresamos con*

$$\Sigma_1, \ldots, \Sigma_n \vdash_\delta \Lambda$$

syss se cumplen las dos siguientes condiciones:

- *Cada una de las δ-cláusulas de la secuencia es o bien una de las Σ_i, $1 \leq i \leq n$ o un δ-resolvente de δ-cláusulas anteriores.*
- *La secuencia termina con la δ-cláusula Λ.*

Ya que acabamos de definir el cálculo de δ-resolución, a continuación mostramos sus dos teoremas fundamentales, la *corrección* y la *completud débil*. Comenzaremos por el primero de ellos. También ahora puede observarse que las demostraciones son duales a las de los correspondientes teoremas en el cálculo de resolución.

Teorema 4.6 (Corrección) *Para toda forma δ-clausal A y toda δ-cláusula Σ, si $A \vdash_\delta \Sigma$, entonces*

$$\Sigma \vDash A$$

Demostración:

Cada vez que aplicamos la regla de δ-resolución obtenemos una δ-cláusula tal que el conjunto de δ-cláusulas original es consecuencia lógica de la δ-cláusula obtenida. Demostraremos esta afirmación por inducción sobre el número de veces que se aplica la regla de δ-resolución.

Si no se aplica ninguna vez la regla de δ-resolución, entonces $\Sigma \in A$. Por la definición 4.2, toda valoración que satisfaga Σ debe satisfacer A, por lo que $\Sigma \vDash A$.

Supongamos que el teorema se cumple hasta para δ-cláusulas obtenidas mediante n usos de la regla de δ-resolución. Veamos si se cumple para $n+1$ usos. Consideremos que el $n+1$-ésimo uso de la regla se aplica sobre las cláusulas Σ_1 y Σ_2, para obtener Σ. Entonces

$$\Sigma_1 = \Sigma_1' \cup \{\lambda\}, \quad \Sigma_2 = \Sigma_2' \cup \{\neg\lambda\}, \quad \Sigma = \Sigma_1' \cup \Sigma_2'.$$

Sea v una valoración que satisface Σ. Entonces v satisface todos los literales de Σ_1' y todos los de Σ_2', por la definición 4.1. Además, o bien $v \vDash \lambda$ o bien $v \vDash \neg\lambda$. Por tanto, o bien $v \vDash \Sigma_1$ o bien $v \vDash \Sigma_2$. Es decir, toda valoración que satisface Σ satisface también a Σ_1 o a Σ_2. Pero tanto Σ_1 como Σ_2 son δ-cláusulas obtenidas en a lo sumo n pasos, por lo que toda valoración que las satisfaga, por hipótesis de inducción, satisface también A.

Por tanto, toda valoración que satisface Σ satisface A. ■

Este teorema, dual al de corrección del cálculo de resolución, nos permite ver una característica interesante de este cálculo. Si en el cálculo de resolución obteníamos cláusulas que eran consecuencia lógica del conjunto de cláusulas original, ahora obtenemos δ-cláusulas tales que el conjunto de δ-cláusulas original es consecuencia lógica de cada una de las δ-cláusulas obtenidas. Podemos ver las pruebas en este cálculo como una búsqueda abductiva, pues se obtienen fórmulas que si se probaran harían posible demostrar las fórmulas de las que partimos. Se trata de un razonar de las conclusiones hasta las hipótesis, de invertir el proceso deductivo. Estas propiedades son típicamente abductivas, aunque, como veremos,

el cálculo de δ-resolución también sirve para realizar inferencias deductivas. Ya que las δ-cláusulas de partida son consecuencia lógica de cada δ-cláusula que se obtenga, cuando se alcance la δ-cláusula vacía, al ser universalmente válida, sabremos que toda valoración —pues todas las valoraciones satisfacen la δ-cláusula vacía— satisface el conjunto de δ-cláusulas de partida. Es decir, el conjunto inicial de δ-cláusulas corresponde a una fórmula universalmente válida.

El siguiente teorema prueba la completud del cálculo de δ-resolución. Nos dice lo siguiente:

Teorema 4.7 (Completud débil) *Si un conjunto de δ-cláusulas A es universalmente válido, entonces*

$$A \vdash_\delta \square$$

Demostración:

Sea $A = \{\Sigma_1, \ldots, \Sigma_n\}$, $n \geq 1$ un conjunto de δ-cláusulas universalmente válido. Si $\square \in A$, entonces la prueba es inmediata. En otro caso, procedemos por inducción sobre el número $m \geq 0$ que resulta de restar el número de δ-cláusulas de A al número de literales que ocurren entre todas las Σ_i, $1 \leq i \leq n$. En el caso base, $m = 0$, el número de δ-cláusulas es igual al número total de literales, por lo que cada δ-cláusula Σ_i tiene un solo literal –ya que estamos en el caso de que $\square \notin A$–. Como todas las δ-cláusulas son unitarias y el conjunto A es universalmente válido, debe haber dos δ-cláusulas $\Sigma_j, \Sigma_k \in A$ tales que $\Sigma_j = \{\gamma\}$ y $\Sigma_k = \{\overline{\gamma}\}$, ya que en otro caso sería posible una valoración que no satisfaría ninguna δ-cláusula de A. Entonces podemos aplicar la regla de δ-resolución a Σ_j y Σ_k, obteniendo $\square$.

Supongamos probado el teorema para los conjuntos de δ-cláusulas con $m = k$. Veamos qué ocurre cuando $m = k + 1$. Ahora tampoco pertenece la δ-cláusula vacía a A, y al ser $m > 0$, debe haber al menos una δ-cláusula Σ_j, $j \leq n$ no unitaria. Sea $\eta \in \Sigma_j$ un literal, y $\Sigma_j^* = \Sigma_j - \{\eta\}$. Entonces, el conjunto de δ-cláusulas

$$A^* = (A - \{\Sigma_j\}) \cup \{\Sigma_j^*\}$$

también es universalmente válido, ya que toda valoración que satisfaga Σ_j debe satisfacer Σ_j^*, por la definición 4.2, y, del mismo modo, toda valoración que satisfaga A satisface también A^*. Al ser A un conjunto de δ-cláusulas universalmente válido, también lo será, pues, A^*. Además, para A^* el valor $m = k$. Así, por hipótesis de inducción, podemos calcular $\square$ mediante δ-resolución desde A^*.

Además, el conjunto

$$A^{**} = (A - \{\Sigma_j\}) \cup \{\{\eta\}\}$$

cumple igualmente que es universalmente válido —por las definiciones 4.1 y 4.2— y que $m < k$, por lo que de nuevo por hipótesis de inducción puede obtenerse la δ-cláusula $\square$ mediante δ-resolución a partir de A^{**}.

Hagamos una demostración mediante δ-resolución en A paralela a la que se hizo en A^* para obtener $\square$, de forma que cada vez que en la prueba que parte de A^* se emplean o bien la δ-cláusula Σ_j^* o una δ-cláusula derivada de Σ_j^*, en la prueba que parte de A se emplearán o bien Σ_j o la correspondiente δ-cláusula derivada de Σ_j, respectivamente. De esta forma, cuando en la prueba que parte de A se llega a $\square$, en la prueba que toma A^* como conjunto inicial de δ-cláusulas pueden ocurrir dos cosas:

1. Que también se alcance $\square$.

2. Que se llegue a la δ-cláusula $\{\eta\}$. En este caso, como tenemos todas las δ-cláusulas de A^{**}, repetimos el proceso, asegurado por hipótesis de inducción, para obtener $\square$.

■

Veamos un ejemplo del *uso deductivo* del cálculo de δ-resolución, para comprobar relaciones de validez y consecuencia lógica. Sea la fórmula

$$(p \wedge (p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow r \tag{4.7}$$

Para ver si es universalmente válida[4], primero la ponemos en forma normal disyuntiva

$$\neg p \vee (p \wedge \neg q) \vee (q \wedge -r) \vee r$$

y obtenemos el conjunto de δ-cláusulas

$$\{\{\neg p\}, \{p, \neg q\}, \{q, \neg r\}, \{r\}\} \tag{4.8}$$

La figura 4.1 muestra la prueba de la δ-cláusula vacía a partir del conjunto anterior. Cada línea representa un paso de la prueba. A la derecha de cada δ-cláusula aparece un comentario que indica si tal δ-cláusula pertenece al conjunto inicial o se obtiene de las anteriores mediante δ-resolución.

1	$\{\neg p\}$	δ-cláusula inicial
2	$\{p, \neg q\}$	δ-cláusula inicial
3	$\{q, \neg r\}$	δ-cláusula inicial
4	$\{r\}$	δ-cláusula inicial
5	$\{\neg q\}$	δ-resolvente de 1 y 2
6	$\{\neg r\}$	δ-resolvente de 3 y 5
7	$\square$	δ-resolvente de 4 y 6

Figura 4.1: Prueba mediante δ-resolución

Como se ha obtenido $\square$, por el teorema 4.6 tenemos que toda valoración que satisface la δ-cláusula vacía satisface el conjunto de δ-cláusulas (4.8). Pero como $\square$ es universalmente válida, entonces (4.8) también lo es. Además, por ser (4.8) equivalente a la fórmula (4.7), tenemos que ésta es también universalmente válida.

A continuación presentamos los elementos que nos permitirán hacer un uso abductivo del cálculo de δ-resolución, que sin duda es su aplicación más interesante.

Definición 4.8 (δ-resolvente minimal) *Decimos que Σ es un* δ-resolvente minimal *del conjunto de δ-cláusulas A syss se cumplen las tres siguientes condiciones:*

[4]Cuando queremos comprobar, por δ-resolución, si cierta fórmula α es consecuencia lógica de $\gamma_1, \ldots, \gamma_n$, lo que haremos es comprobar si $\gamma_1 \wedge \ldots \wedge \gamma_n \rightarrow \alpha$ es universalmente válida, lo cual es equivalente por el teorema de la deducción.

1. Σ *es satisfactible, es decir, no contiene literales complementarios.*
2. $A \vdash_\delta \Sigma$
3. *No existe ninguna* $\Sigma' \subset \Sigma$ *tal que* $A \vdash_\delta \Sigma'$.

Definición 4.9 (Abducción minimal) *Dada una fórmula proposicional no válida* $\alpha \in \mathcal{L}_p$*, decimos que la* δ*-cláusula* Σ *es una* abducción minimal *de* α *syss se cumplen las tres siguientes condiciones:*

1. Σ *es satisfactible, es decir, no contiene literales complementarios.*
2. $\Sigma \models \alpha$.
3. *No existe ninguna* $\Sigma' \subset \Sigma$ *tal que* $\Sigma' \models \alpha$.

Antes de continuar, aclaremos que la definición anterior de *abducción minimal* no debe confundirse con la definición 2.29 de *solución minimal* a cierto problema abductivo. La definición 4.9 se refiere siempre a una δ-cláusula, que en principio no tiene por qué ser solución a ningún problema abductivo. Posteriormente se observará la relación que existe entre ambos conceptos.

Hemos definido dos clases de δ-cláusulas, los *δ-resolventes minimales* y las *abducciones minimales*. La primera reúne las δ-cláusulas satisfactibles y minimales que pueden obtenerse a partir de un conjunto de δ-cláusulas A tras aplicar la regla de δ-resolución hasta la saturación —más adelante definiremos formalmente esta idea—. La segunda integra los conjuntos de literales, minimales y satisfactibles, tal que toda valoración que los satisfaga satisface necesariamente cierta fórmula α. Demostraremos que cuando A es la forma δ-clausal de α, los *δ-resolventes minimales* de A coinciden con las *abducciones minimales* de α. Para ello hace falta probar los dos siguientes teoremas:

- Si Σ es una *abducción minimal* de α, será también un *δ-resolvente minimal* de A. Llamamos a este teorema **completud abductiva** del cálculo de δ-resolución, pues asegura que todas las abducciones minimales se pueden encontrar mediante δ-resolución.

- Si Σ es un *δ-resolvente minimal* de A, será también una *abducción minimal* de α. A este segundo teorema lo llamamos **corrección abductiva** del cálculo de δ-resolución, pues asegura que las δ-cláusulas minimales satisfactibles que se encuentren por δ-resolución son abducciones minimales.

Comenzaremos demostrando el **teorema de completud abductiva**:

Teorema 4.10 (Completud abductiva) *Dada una fórmula proposicional no universalmente válida $\alpha \in \mathcal{L}_p$, cuya forma δ-clausal es A, cada δ-cláusula que sea una* abducción minimal *de α es también un* δ-resolvente minimal *de A.*

Demostración:

Sea α una fórmula proposicional no válida, y A su forma δ-clausal. Además, sea $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ una δ-cláusula que cumple los requisitos de la definición 4.9 para ser una abducción minimal de α. Demostraremos que Σ es igualmente un δ-resolvente minimal de A, según los requisitos de la definición 4.8. El primero de tales requisitos lo cumple por ser también una de las condiciones que establece la definición 4.9. En cuanto al segundo, debemos probar $A \vdash_\delta \Sigma$. Por la segunda condición de la definición 4.9 tenemos que

$$\Sigma \vDash \alpha \tag{4.9}$$

que por la definición 4.1 equivale a

$$\sigma_1 \wedge \ldots \wedge \sigma_n \vDash \alpha \tag{4.10}$$

lo cual es equivalente, por el teorema de la deducción, a

$$\vDash \sigma_1 \wedge \ldots \wedge \sigma_n \rightarrow \alpha \tag{4.11}$$

que se transforma, por evaluación de la implicación, en

$$\vDash \neg(\sigma_1 \wedge \ldots \wedge \sigma_n) \vee \alpha \tag{4.12}$$

lo que introduciendo negadores y eliminando negaciones resulta lo siguiente

$$\vDash \overline{\sigma_1} \vee \ldots \vee \overline{\sigma_n} \vee \alpha \tag{4.13}$$

que, por la definición 4.2, es equivalente a

$$\vDash \{\{\overline{\sigma_1}\}, \ldots, \{\overline{\sigma_n}\}\} \cup A \tag{4.14}$$

Como (4.14) establece la validez universal del conjunto de δ-cláusulas

$$\{\{\overline{\sigma_1}\}, \ldots, \{\overline{\sigma_n}\}\} \cup A \tag{4.15}$$

por el teorema 4.7 sabemos que existe una demostración por δ-resolución de la δ-cláusula vacía $\square$ a partir del conjunto de δ-cláusulas que aparece en (4.15). Llamaremos a esta primera demostración $\mathcal{D}em$. Pero podemos hacer una demostración $\mathcal{D}em'$ paralela a $\mathcal{D}em$, simplemente con δ-cláusulas de A, de forma que cuando en $\mathcal{D}em$ se emplea una de las δ-cláusulas de (4.15) que no está en A, no se hace nada en $\mathcal{D}em'$. Como las δ-cláusulas de (4.15) que no están en A son

$$\{\overline{\sigma_1}\}, \ldots, \{\overline{\sigma_n}\}$$

que, por ser unitarias, en la demostración $\mathcal{D}em$ lo único que pueden hacer es eliminar literales σ_i, $1 \leq i \leq n$, al final de $\mathcal{D}em'$ nos quedará una δ-cláusula Σ' tal que $\Sigma' \subseteq \Sigma$. Pero no puede ser $\Sigma' \subset \Sigma$, pues tendríamos, por el teorema 4.6, que $\Sigma' \vDash \alpha$. Sin embargo, esto contradice el hecho de que Σ sea una *abducción minimal* de α, pues se viola la tercera condición de la definición 4.9. Por tanto, tenemos que $\Sigma' = \Sigma$, con lo que se cumple la segunda condición de la definición 4.8 para que Σ sea un δ-resolvente minimal de A. Pero también se cumple la tercera, pues, como acabamos de ver, si hubiese otro $\Sigma' \subset \Sigma$, tal que $A \vdash_\delta \Sigma'$ fuera δ-resolvente de A, entonces Σ no podría ser una abducción minimal.

De forma que cada abducción minimal de α es también un δ-resolvente minimal de A. ■

Teorema 4.11 (Corrección abductiva) *Dada una fórmula proposicional no universalmente válida $\alpha \in \mathcal{L}_p$, cuya forma δ-clausal es A, cada δ-cláusula que sea un* δ-resolvente minimal *de A es también una* abducción minimal *de α.*

Demostración:

Sea α una fórmula proposicional no válida, y A su forma δ-clausal. Además, sea $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ una δ-cláusula que cumple los requisitos de la definición 4.8 para ser un δ-resolvente minimal de A. Demostraremos que Σ es igualmente una abducción minimal de α, según los requisitos de la definición 4.9. El primero de tales requisitos lo cumple por ser también una de las condiciones que establece la definición 4.8. En cuando al segundo, también se cumple pues, dado que $A \vdash_\delta \Sigma$ —segundo requisito de la definición 4.8—, por el teorema 4.6 tenemos que $\Sigma \vDash \alpha$. El tercer requisito también se cumple, pues si no, habría otra δ-cláusula $\Sigma' \subset \Sigma$ tal que $\Sigma' \vDash \alpha$, y por el teorema 4.10 tendríamos que $A \vdash_\delta \Sigma'$, lo cual contradice el tercer requisito que Σ cumple por ser un δ-resolvente minimal. ■

Corolario 4.12 *Dada $\alpha \in \mathcal{L}_p$, una fórmula proposicional cualquiera, cuya forma δ-clausal es A, el conjunto de* δ-resolventes minimales *de A es igual al conjunto de* abducciones minimales *de α.*

Demostración:

Según sea α, pueden darse tres casos:

- Si α es universalmente válida, entonces como $\vDash \alpha$, la única δ-cláusula que cumple los requisitos de la definición 4.9 es $\square$, por lo que el conjunto de abducciones minimales de α sólo contiene la δ-cláusula vacía. Además, por el teorema 4.7, tenemos $A \vdash_\delta \square$, por lo que también $\square$ es el único δ-resolvente minimal de A.

- Si α no es satisfactible, entonces no puede existir ninguna δ-cláusula satisfactible Σ tal que $\Sigma \vDash \alpha$, con lo que el conjunto de abducciones minimales de α es vacío. Tampoco A puede tener ningún δ-resolvente minimal Σ, pues al ser el segundo requisito de la definición 4.8 que $A \vdash_\delta \Sigma$, por el teorema 4.6 tendríamos que $\Sigma \vDash \alpha$, y como por el primer requisito de la definición 4.8 la δ-cláusula Σ sería satisfactible, esto contradice el que α no lo sea, que es el caso en que estamos. Por tanto, A no tiene ningún δ-resolvente minimal.

- Cuando α es meramente satisfactible, tenemos, por el teorema 4.10, que todas sus abducciones minimales son δ-resolventes minimales y, por el teorema 4.11, que todos sus δ-resolventes minimales son igualmente abducciones minimales.

■

Observación 4.13 Resulta interesante, en la demostración anterior, el caso en que α no es satisfactible. Entonces, no existe siquiera ninguna δ-cláusula satisfactible Σ tal que $A \vdash_\delta \Sigma$, pues, de existir, por el teorema 4.6 tendríamos que $\Sigma \vDash \alpha$, cosa que no puede ocurrir, pues α no es satisfactible. Por tanto, todas las δ-cláusulas de A serán contradictorias, es decir, tendrán literales complementarios, así como todas las δ-cláusulas que puedan obtenerse por δ-resolución desde A.

Teorema 4.14 *Dados dos conjuntos de δ-cláusulas A y B, si A y B son equivalentes, sus conjuntos de δ-resolventes minimales son iguales.*

Demostración:

Antes de comenzar la prueba, veamos que dado un conjunto de δ-cláusulas

$$C = \{\{\lambda_{1_1}, \ldots, \lambda_{1_{j_1}}\}, \ldots, \{\lambda_{i_1}, \ldots, \lambda_{i_{j_i}}\}\}$$

resulta trivial que la fórmula proposicional

$$(\lambda_{1_1} \wedge \ldots \wedge \lambda_{1_{j_1}}) \vee \ldots \vee (\lambda_{i_1} \wedge \ldots \wedge \lambda_{i_{j_i}})$$

tendrá C como forma δ-clausal, puesto que está ya en forma normal disyuntiva. Por tanto, dado un conjunto de δ-cláusulas cualquiera podemos encontrar una fórmula proposicional cuya forma δ-clausal es dicho conjunto. Emplearemos este resultado a continuación.

Sean α y β dos fórmulas proposicionales cuyas formas δ-clausales son, respectivamente, A y B —podemos obtener α y β como acabamos de explicar—. Por tanto, α y β son equivalentes, respectivamente, a A y B y como A y B son equivalentes entre sí, también lo serán α y β.

Para demostrar que los δ-resolventes minimales de A y B son los mismos, veamos primero que todos los δ-resolventes minimales de A son δ-resolventes minimales de B. Sea Σ un δ-resolvente minimal de A. Entonces, por el corolario 4.12, Σ es una abducción minimal de α. Ello significa, por la definición 4.9, que se cumple:

1. Σ es satisfactible.
2. $\Sigma \models \alpha$
3. No existe ninguna δ-cláusula Σ' tal que $\Sigma' \subset \Sigma$ y $\Sigma' \models \alpha$

Pero como α y β son equivalentes, toda valoración que satisface α satisface igualmente β por lo que desde el punto 2 precedente tenemos

$$\Sigma \models \beta \tag{4.16}$$

Además, si existiera alguna δ-cláusula Σ' tal que $\Sigma' \subset \Sigma$ y $\Sigma' \models \beta$, por ser α y β equivalentes, tendríamos que $\Sigma' \models \alpha$, lo cual va en contra del punto 3 anterior, con lo que debemos negar la hipótesis de que exista tal Σ'. Pero esta es la tercera condición de la definición 4.9 para afirmar que Σ sea una abducción minimal de β. Además, la segunda condición queda asegurada por (4.16), y la primera es la misma que se expresó en el punto 1 de la enumeración anterior.

Pero si Σ es una abducción minimal de β, por el corolario 4.12 también será un δ-resolvente minimal de B, con lo que hemos demostrado que todos los δ-resolventes minimales de A son también δ-resolventes minimales de B.

La prueba de que todos los δ-resolventes minimales de B lo son también de A es idéntica. ■

Corolario 4.15 (Eliminación de δ-cláusulas contradictorias) *El conjunto de δ-resolventes minimales del conjunto de δ-cláusulas*

$$C = B \cup \{\{\lambda, \neg\lambda, \gamma_1, \ldots, \gamma_n\}\}$$

es igual al conjunto de δ-resolventes minimales del conjunto de δ-cláusulas B.

Demostración:

Probemos que el conjunto de δ-cláusulas C es equivalente a B. Primero probamos que toda valoración que satisfaga C satisface también B. Así, dada una valoración v que satisfaga C, por la definición 4.2, v satisface una δ-cláusula de C. Pero siendo $C = B \cup \{\{\lambda, \neg\lambda, \gamma_1, \ldots, \gamma_n\}\}$, como la δ-cláusula $\{\lambda, \neg\lambda, \gamma_1, \ldots, \gamma_n\}$ no puede ser satisfecha, al contener dos literales complementarios, v debe satisfacer alguna de las δ-cláusulas de B y, por la definición 4.2, v satisface B.

Del mismo modo, sea v una valoración que satisface B. De nuevo por la definición 4.2, v satisface una de las δ-cláusulas de B, pero como $B \subset C$, v satisface también una de las δ-cláusulas de C, y por la definición 4.2 v satisface C.

Al ser B y C equivalentes, por el teorema 4.14, sus δ-resolventes minimales lo son también. ■

Este corolario nos permite ignorar las δ-cláusulas no satisfactibles —a las que en ocasiones también llamaremos *δ-cláusulas contradictorias*— del proceso de búsqueda, con lo que sólo tenemos que centrarnos en las que no contengan literales complementarios. Además, podremos desechar también algunas de las δ-cláusulas satisfactibles, las subsumidas. La noción de δ-cláusula subsumida es la misma que en el cálculo de resolución clásico.

Definición 4.16 (Subsunción de δ-cláusulas) *Decimos de una δ-cláusula Σ_1 que está* subsumida por *otra δ-cláusula Σ_2 syss $\Sigma_2 \subseteq \Sigma_1$. De Σ_2 decimos que* subsume a Σ_1.

Teorema 4.17 (Eliminación de δ-cláusulas subsumidas) *Para toda forma δ-clausal A tal que $\Sigma_1, \Sigma_2 \in A$, $\Sigma_1 \neq \Sigma_2$ y Σ_1 está subsumida por Σ_2, se cumple que A y $A - \{\Sigma_1\}$ comparten el mismo conjunto δ-resolventes minimales.*

Demostración:

Dado que Σ_2 subsume a Σ_1, y por la definición 4.16 ello significa $\Sigma_2 \subseteq \Sigma_1$, toda valoración v que satisfaga Σ_1, como por la definición 4.1 satisface todos los

literales de Σ_1, y los literales de Σ_2 son un subconjunto de los mismos, tenemos que v satisface todos los literales de Σ_2 y, de nuevo por la definición 4.1, v satisface Σ_2. Por tanto, toda valoración que satisfaga Σ_1 satisface igualmente Σ_2

Probemos ahora que A es equivalente a $A - \{\Sigma_1\}$. Sea v una valoración que satisface A. Entonces, por la definición 4.2, v satisface alguna δ-cláusula de A. Si se trata de Σ_1, entonces, como se vio en el anterior párrafo, v satisface Σ_2, y como $\Sigma_2 \in A - \{\Sigma_1\}$, tenemos, por la definición 4.2, que v satisface $A - \{\Sigma_1\}$. Si la δ-cláusula de A que satisface v es diferente a Σ_1, entonces es una de las que pertenecen a $A - \{\Sigma_1\}$, y de nuevo por la definición 4.2 ocurre que v satisface $A - \{\Sigma_1\}$. Por tanto, toda valoración que satisfaga A satisface $A - \{\Sigma_1\}$. Lo recíproco resulta trivial, pues al ser $A - \{\Sigma_1\} \subseteq A$ si v satisface $A - \{\Sigma_1\}$, como satisface una de sus δ-cláusulas, satisface por tanto una δ-cláusula de A, con lo que también satisface A.

Como A y $A - \{\Sigma_1\}$ son equivalentes, por el teorema 4.14 sabemos que comparten el mismo conjunto de δ-resolventes minimales. ■

Definición 4.18 (Saturación por δ-resolución) *Dado un conjunto de δ-cláusulas A, definimos su conjunto* saturación por δ-resolución*, A^δ, como el conjunto de δ-cláusulas que resulta de aplicar sucesivamente entre fórmulas de A, así como entre las que aparezcan durante el proceso, la regla de δ-resolución y eliminar cada la δ-cláusula que resulte subsumida por otra diferente o que sea no satisfactible.*

Por ser la δ-resolución dual a la resolución, se cumple que en lógica proposicional su saturación es finita, con lo que se da el siguiente teorema.

Teorema 4.19 *Dado un conjunto finito de δ-cláusulas A, el proceso para obtener A^δ termina en un número finito de pasos.*

Demostración:

Dado que la cardinalidad $|A|$ es finita, el número de variables proposicionales que aparecen en A también lo es, así como el número de δ-cláusulas diferentes que pueden formarse con dichas variables. Por tanto, las sucesivas aplicaciones

de la regla de δ-resolución no pueden producir indefinidamente nuevos δ-resolventes, con lo que debe llegar un momento en que todas las aplicaciones de la regla de δ-resolución que resulten posibles den lugar a δ-cláusulas contradictorias o subsumidas por otras que ya han aparecido. ■

Corolario 4.20 *Dado un conjunto finito de δ-cláusulas A, la cardinalidad $|A^\delta|$ es finita.*

Demostración:

Se trata de una consecuencia del teorema 4.19. Si el proceso de saturación por δ-resolución termina en un número finito de pasos, como de cada aplicación de la regla de δ-resolución se genera una sola δ-cláusula, y la cardinalidad de A es finita, también lo será la de A^δ. ■

Teorema 4.21 *Dado un conjunto de δ-cláusulas A, A^δ será el conjunto de sus δ-resolventes minimales.*

Demostración:

Sea A un conjunto de δ-cláusulas y A^δ su saturación por δ-resolución. Probemos primero que cada δ-cláusula $\Sigma \in A^\delta$ es un δ-resolvente minimal de A. Para ello Σ debe cumplir las tres condiciones que le impone la definición 4.8, que son:

1. Σ debe ser satisfactible. Lo es, pues en caso contrario, por la definición 4.18, no aparecería en A^δ.

2. $A \vdash_\delta \Sigma$. También se cumple, por la definición 4.18.

3. No existe ninguna Σ' tal que $A \vdash_\delta \Sigma'$ y

$$\Sigma' \subset \Sigma$$

   También se verifica, pues por la definición 4.18 si se cumpliera $A \vdash_\delta \Sigma'$ entonces Σ no pertenecería a A^δ.

Ahora debemos probar que todas las δ-cláusulas que sean δ-resolventes minimales de A estarán en A^δ. Como los δ-resolventes minimales de A son todas las δ-cláusulas Σ que cumplan las condiciones 1–3 anteriores, serán por tanto las δ-cláusulas demostrables por δ-resolución desde A —como indica la condición segunda— que sean satisfactibles —primera condición— y no sean subsumidas por otros δ-resolventes —tercera condición—. Pero A^δ está formado, por la definición 4.18, precisamente por tales δ-cláusulas, pues las que son eliminadas son simplemente las contradictorias y las subsumidas[5]. ■

Corolario 4.22 *Una fórmula proposicional es universalmente válida syss, dada su forma δ-clausal A, se cumple*

$$A^\delta = \{\square\}$$

Demostración:

Si α es valida, también lo será A, y por el teorema 4.7, $A \vdash_\delta \square$, por lo que, como $\square$ no puede ser subsumida, al no tener subconjuntos, tenemos que $\square \in A^\delta$. Además, $\square$ subsume cualquier otra δ-cláusula, pues $\square$ es subconjunto de cualquier otra, por lo que no puede haber en A^δ ninguna otra δ-cláusula.

En sentido inverso, si $A^\delta = \{\square\}$ entonces por el teorema 4.21 se cumple que $\square$ es un δ-resolvente minimal de A y, por ser uno de los requisitos de la definición 4.8, también $A \vdash_\delta \square$, y por el teorema 4.6 toda valoración que satisfaga $\square$ satisface α, pero por ser $\square$ universalmente válida, también lo será α. ■

Corolario 4.23 *Dada una fórmula proposicional $\alpha \in \mathcal{L}_p$ cuya forma δ-clausal sea A, el conjunto de δ-cláusulas que son abducciones minimales de α coincide con A^δ.*

[5]En realidad, todas las δ-cláusulas están subsumidas por sí mismas, según la definición 4.16. Sin embargo, las que son eliminadas durante el proceso de saturación, como indica la definición 4.18, son las subsumidas por δ-cláusulas distintas de sí mismas, es decir, por otras que sean subconjuntos propios suyos. Por ello, cuando se indique que en un proceso de saturación se eliminan las δ-cláusulas subsumidas se debe sobrentender que son las subsumidas por otras diferentes. En otro caso, la saturación por δ-resolución convertiría cualquier conjunto de δ-cláusulas en la forma δ-clausal vacía.

Demostración:

Por el corolario 4.12, el conjunto de δ-cláusulas que son abducciones minimales de α coincide con el conjunto de δ-resolventes minimales de A. Pero por el teorema 4.21, el conjunto de δ-resolventes minimales de A es A^{δ}. ■

El corolario 4.23 nos dice que, dada una fórmula proposicional cualquiera α, podemos obtener todas las δ-cláusulas que sean abducciones minimales de α obteniendo su forma δ-clausal A y saturando dicho conjunto por δ-resolución, eliminando las δ-cláusulas contradictorias y subsumidas —se sobrentiende que por otras diferentes— que aparezcan.

En la siguiente sección veremos cómo emplear estos resultados para encontrar *soluciones abductivas explicativas* a problemas abductivos proposicionales. Ahora, diremos algo sobre la dualidad entre el cálculo de δ-resolución y el cálculo de resolución. Dada cualquier fórmula proposicional α, $FND(\alpha)$ tendrá la forma

$$(\gamma_{1_1} \wedge \gamma_{1_2} \wedge \ldots) \vee (\gamma_{2_1} \wedge \gamma_{2_2} \wedge \ldots) \vee \ldots \vee (\gamma_{k_1} \wedge \gamma_{k_2} \wedge \ldots) \tag{4.17}$$

siendo γ_{i_j} un literal para cada $i, j \geq 1$. Al ser equivalentes α y $FND(\alpha)$, si negamos ésta obtenemos una fórmula equivalente a $\neg\alpha$. Pero tal fórmula es

$$\neg((\gamma_{1_1} \wedge \gamma_{1_2} \wedge \ldots) \vee (\gamma_{2_1} \wedge \gamma_{2_2} \wedge \ldots) \vee \ldots \vee (\gamma_{k_1} \wedge \gamma_{k_2} \wedge \ldots))$$

que se convierte, introduciendo negadores, en

$$\neg(\gamma_{1_1} \wedge \gamma_{1_2} \wedge \ldots) \wedge \neg(\gamma_{2_1} \wedge \gamma_{2_2} \wedge \ldots) \wedge \ldots \wedge \neg(\gamma_{k_1} \wedge \gamma_{k_2} \wedge \ldots)$$

y a su vez en

$$(\neg\gamma_{1_1} \vee \neg\gamma_{1_2} \vee \ldots) \wedge (\neg\gamma_{2_1} \vee \neg\gamma_{2_2} \vee \ldots) \wedge \ldots \wedge (\neg\gamma_{k_1} \vee \neg\gamma_{k_2} \vee \ldots) \tag{4.18}$$

Si cada ocurrencia de $\neg\neg\lambda$ la sustituimos por λ, llegamos a una fórmula en forma normal conjuntiva que es equivalente a $\neg\alpha$. Se trata de $FNC(\neg\alpha)$. Comparando (4.17) y (4.18) constatamos la dualidad entre $FND(\alpha)$ y $FNC(\neg\alpha)$.

Esta dualidad hace que si la forma δ-clausal de $FND(\alpha)$ es

$$\{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$$

la forma clausal —en el sentido común en que se usa en el cálculo de resolución— de $FNC(\neg\alpha)$ será

$$\{\Sigma_1^*, \Sigma_2^*, \ldots, \Sigma_n^*\}$$

siendo $\Sigma_i^* = \{\overline{\gamma} \mid \gamma \in \Sigma_i\}$, $i \leq n$, y γ, $\overline{\gamma}$ literales complementarios.

La dualidad que hemos indicado asegura que para toda δ-cláusula Σ que pueda obtenerse mediante el cálculo de δ-resolución a partir de $\{\Sigma_1, \Sigma_2, \ldots, \Sigma_n\}$, la cláusula Σ^* puede obtenerse mediante resolución desde $\{\Sigma_1^*, \Sigma_2^*, \ldots, \Sigma_n^*\}$, y viceversa. Por tanto, no hay ningún problema que pueda resolverse mediante δ-resolución que no pueda resolverse —en su versión dual— por resolución. Entonces, ¿por qué hemos definido el cálculo de δ-resolución? Hay varias razones que nos han movido a ello:

- En primer lugar, usando el cálculo habitual de resolución, las soluciones abductivas no vendrían dadas directamente, sino que serían las negaciones de las cláusulas que obtuviéramos mediante resolución. Por tanto, no estaríamos ante un cálculo abductivo —como sí lo es el de δ-resolución— sino ante un uso abductivo de un cálculo genuinamente deductivo.
- Además, en esta sección se han probado ciertos teoremas cuyas formas duales no suelen ser demostradas para el cálculo de resolución, pues algunos de ellos no tienen sentido de cara a la búsqueda de refutaciones que hace la resolución.
- Otra de las ventajas que encontramos en el cálculo de δ-resolución es que permite una integración muy eficiente con las tablas semánticas para resolver problemas abductivos, como veremos más adelante.

4.3. Abducción explicativa mediante δ-resolución

En esta sección presentamos un primer procedimiento —que más adelante sufrirá varias optimizaciones— para realizar abducción explicativa mediante δ-

resolución. En primer lugar, dado un problema abductivo $\langle \Theta, \phi \rangle$, cuya teoría Θ es el conjunto de fórmulas proposicionales $\{\theta_1, \ldots, \theta_n\}$, como explicamos en la página 84 —relación de consecuencia lógica (4.4)— se debe cumplir, para toda conjunción de literales $\lambda_1 \wedge \ldots \wedge \lambda_m$ que sea una solución abductiva plana:

$$\lambda_1 \wedge \ldots \wedge \lambda_m \vDash \theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$$

Además, si queremos que las soluciones sean minimales, es decir, que no haya ningún subconjunto de $\{\lambda_1, \ldots, \lambda_m\}$ que sea a su vez una explicación, y que entre los λ_i, $1 \leq i \leq m$ no haya literales complementarios, para que las soluciones sean consistentes en primer lugar consigo mismas —pues una fórmula inconsistente lo explica todo, en sentido *plano*—, lo que estamos buscando son las δ-cláusulas $\{\lambda_1, \ldots, \lambda_m\}$ que sean *abducciones minimales* de $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$, pues los requisitos que acabamos de imponer a $\{\lambda_1, \ldots, \lambda_m\}$ son los mismos que establece la definición 4.9. Pero si T es la forma δ-clausal de $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$, por el corolario 4.23 hemos establecido que las *abducciones minimales* de $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$ coinciden con las δ-cláusulas que forman parte de T^δ.

Por tanto, las soluciones abductivas planas y minimales de $\langle \Theta, \phi \rangle$ se obtienen saturando por δ-resolución la forma δ-clausal de $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$. Sin embargo no todas las soluciones planas serán consistentes y explicativas, aunque lo recíproco sí se dé, pues el requisito de abducción plana es común con todos los tipos de solución abductiva. De modo que de entre las abducciones planas debemos seleccionar las que cumplen los requisitos para ser abducciones explicativas.

4.3.1. Requisito de consistencia

En cuanto al requisito de consistencia, establece

$$\Theta \cup \{\lambda_1 \wedge \ldots \wedge \lambda_m\} \nvDash \bot \tag{4.19}$$

lo que equivale a

$$FND((\theta_1 \wedge \ldots \wedge \theta_n) \wedge (\lambda_1 \wedge \ldots \wedge \lambda_m)) \nvDash \bot \tag{4.20}$$

Pero

$$\models \theta_1 \wedge \ldots \wedge \theta_n \leftrightarrow FND(\theta_1 \wedge \ldots \wedge \theta_n) \tag{4.21}$$

y por (4.21) transformamos (4.20) en

$$FND(FND(\theta_1 \wedge \ldots \wedge \theta_n) \wedge (\lambda_1 \wedge \ldots \wedge \lambda_m)) \not\models \bot \tag{4.22}$$

Pero si

$$FND(\theta_1 \wedge \ldots \wedge \theta_n) = (\gamma_{1_1} \wedge \ldots \wedge \gamma_{1_{j_1}}) \vee \ldots \vee (\gamma_{k_1} \wedge \ldots \wedge \gamma_{k_{j_k}}) \tag{4.23}$$

tenemos, que por (4.23) se puede transformar (4.22) en

$$FND((\gamma_{1_1} \wedge \ldots \wedge \gamma_{1_{j_1}}) \vee \ldots \vee (\gamma_{k_1} \wedge \ldots \wedge \gamma_{k_{j_k}})) \wedge (\lambda_1 \wedge \ldots \wedge \lambda_m)) \not\models \bot \tag{4.24}$$

pero aplicando sucesivas veces la propiedad distributiva, lo anterior se transforma en

$$\bigvee_{i=1}^{k} (\gamma_{i_1} \wedge \ldots \wedge \gamma_{i_{j_i}} \wedge \lambda_1 \wedge \ldots \wedge \lambda_m) \not\models \bot \tag{4.25}$$

Con lo que el requisito de consistencia sólo se cumple para los $\{\lambda_1, \ldots, \lambda_m\}$ tales que

$$\bigvee_{i=1}^{k} (\gamma_{i_1} \wedge \ldots \wedge \gamma_{i_{j_i}} \wedge \lambda_1 \wedge \ldots \wedge \lambda_m) \tag{4.26}$$

sea consistente, es decir, haya algún valor de i para el que

$$\gamma_{i_1} \wedge \ldots \wedge \gamma_{i_{j_i}} \wedge \lambda_1 \wedge \ldots \wedge \lambda_m \tag{4.27}$$

no contenga literales complementarios.

Todo lo anterior demuestra que

Proposición 4.24 *Para que $\{\lambda_1, \ldots, \lambda_m\}$ sea una solución consistente, debe haber alguna conjunción elemental $\gamma_{i_1} \wedge \ldots \wedge \gamma_{i_{j_i}}$ en la forma normal disyuntiva de la teoría que, además de ser satisfactible —no debe contener literales complementarios—, no contenga ninguno de los literales $\{\overline{\lambda_1}, \ldots, \overline{\lambda_m}\}$.*

4.3.2. Requisito explicativo

El requisito explicativo, aplicado a $\{\lambda_1, \ldots, \lambda_m\}$ exige

$$\lambda_1 \wedge \ldots \wedge \lambda_m \not\models \phi \tag{4.28}$$

lo cual equivale a

$$\neg\phi \wedge (\lambda_1 \wedge \ldots \wedge \lambda_m) \not\models \bot \tag{4.29}$$

y también a

$$FND(\neg\phi \wedge (\lambda_1 \wedge \ldots \wedge \lambda_m)) \not\models \bot \tag{4.30}$$

Pero obsérvese que (4.30) difiere de (4.20) en que en lugar de $\theta_1 \wedge \ldots \wedge \theta_n$ ahora nos referimos a $\neg\phi$, por lo que el requisito queda igual, pero referido a $\neg\phi$ en vez de a Θ. Por tanto,

Proposición 4.25 *Para que $\{\lambda_1, \ldots, \lambda_m\}$ sea una solución explicativa, debe haber alguna conjunción elemental $\gamma_{i_1} \wedge \ldots \wedge \gamma_{i_{j_i}}$ en la forma normal disyuntiva de la negación de la observación que, además de ser satisfactible —no debe contener literales complementarios—, no contenga ninguno de los literales $\{\overline{\lambda_1}, \ldots, \overline{\lambda_m}\}$.*

4.3.3. Procedimiento abductivo

Con todos los elementos que hemos introducido en este capítulo podemos definir un primer procedimiento de abducción explicativa usando δ-resolución, de forma que dadas una teoría

$$\Theta = \{\theta_1, \ldots, \theta_n\}$$

y una observación ϕ —véase que ahora no se ha impuesto que deba ser un literal— se procede de la siguiente manera:

1. Se obtiene A, forma δ-clausal de $\theta_1 \wedge \ldots \wedge \theta_n \rightarrow \phi$

2. Se satura A por δ-resolución obteniendo A^δ. Si $A^\delta = \{\Box\}$ entonces por el corolario 4.22 es que $\vDash \theta_1, \ldots, \theta_n \rightarrow \phi$ y por tanto $\Theta \vDash \phi$. En otro caso,

3. De entre los elementos de A^δ se seleccionan sólo los que cumplen los requisitos de consistencia y explicativo. Esto se puede realizar:

   a) Obteniendo $FND(\theta_1 \wedge \ldots \wedge \theta_n)$ y $FND(\neg\phi)$.

   b) Eliminando los elementos de A^δ que no cumplan el requisito de la proposición 4.24.

   c) Eliminando los elementos de A^δ que no cumplan el requisito de la proposición 4.25.

4.4. Implementación

A partir de la página 271 ofrecemos una implementación del procedimiento abductivo mediante δ-resolución que hemos presentado en la sección anterior. A continuación comentamos las características principales mientras mostramos la salida que produce el programa para el problema abductivo

$$\langle\{(p \to q) \wedge (t \to s) \wedge (m \to p \wedge t)\}, q \wedge s\rangle$$

En este caso, la observación no es un literal, pues como ya hemos visto, a diferencia de lo que ocurría con el sistema de Aliseda, mediante δ-resolución pueden buscarse explicaciones a fórmulas más complejas. El primer paso que realiza el programa es obtener la forma δ-clausal que servirá como punto de partida para la búsqueda abductiva. Esto se lleva a cabo obteniendo la forma δ-clausal de la implicación que tiene a la teoría como antecedente y la observación como consecuente, y seleccionando sólo las δ-cláusulas satisfactibles. La transformación a forma δ-clausal se realiza mediante el predicado —definido en `libreria.pl` (página 282)— `d_clausulas(+Form,?Cls)` que, dada la fórmula `Fml`, devuelve su forma δ-clausal `Cls`. Emplea los dos siguientes pasos:

1. Construye la forma normal disyuntiva de `Form` mediante `fnd(+Form,?Fnd)`. El predicado `fnd/2` está definido a partir de la transformación a forma normal conjuntiva que aparece en [AB02], teniendo en cuenta la dualidad entre ambas formas normales.

2. A partir de `Fnd`, se obtiene la forma δ-clausal correspondiente mediante `d_clausulas_FND(+Fnd,?Cls)`.

Antes de guardar cada una de las δ-cláusulas obtenidas, se comprueba su satisfactibilidad, desechando las δ-cláusulas no satisfactibles, que serán las que contengan literales complementarios. La salida en pantalla de esta parte del proceso es la siguiente:

```
?- abduce_dresol((p => q) & (t => s) & (m => p & t),q & s).

PROBLEMA ABDUCTIVO:
   Teoría:       (p=>q)& (t=>s)& (m=>p&t)
   Observación:  q&s

   Forma d-clausal de Teo => Obs:
     Satisfactible:   [p, -q]
     Satisfactible:   [t, -s]
     Satisfactible:   [m, -p]
     Satisfactible:   [m, -t]
     Satisfactible:   [q, s]
```

A continuación viene la parte más importante del proceso. Se trata de la saturación por δ-resolución del conjunto de δ-cláusulas que se ha obtenido. La implementación que hemos hecho sigue los mismos pasos que Otter, (acrónimo de Organized Techniques for Theorem-proving and Effective Research), un demostrador para primer orden basado en resolución, programado en ANSI C en el Argonne National Laboratory de Chicago. Hemos tomado la implementación que J.A. Alonso y J. Borrego hacen de Otter en Prolog [AB02] y la hemos adaptado —dualizándola— para que realice saturación por δ-resolución de un conjunto de δ-cláusulas dado. El resultado aparece en el fichero `libreria.pl` —página 282, y tiene como predicado principal

```
d_resol_anotada(+Usable, +Soporte, +Subsumidas, ?UltUsab)
```

tal que, dadas las listas de δ-cláusulas `Usable` y `Soporte` —usamos los mismos nombres que Otter— y la lista de δ-cláusulas `Subsumidas`, que son las que ya

han sido subsumidas —en principio `Usable` y `Subsumidas` son la lista vacía, y `Soporte` la lista de δ-cláusulas que quiere saturarse— construye la lista de δ-cláusulas `UltUsab` que es:

- El conjunto de δ-cláusulas correspondiente a la saturación por δ-resolución de `Usable` $\cup$ `Soporte`, en caso de que tal saturación no contenga $\square$.
- Si la saturación de `Usable` $\cup$ `Soporte` contiene $\square$, entonces el conjunto inicial de δ-cláusulas es universalmente válido. En tal caso, `UltUsab` contiene todas las δ-cláusulas necesarias para alcanzar $\square$ desde `Usable` $\cup$ `Soporte`. Entonces, mediante `escribe_prueba(UltUsab)` se reconstruye la prueba de $\square$ y se muestra en pantalla.

Cada δ-cláusula se representa como `N*H*Lista`, siendo `N` un número que identifica dicha δ-cláusula, `H` la *historia* de la δ-cláusula, es decir, una lista con las δ-cláusulas de que proviene, y `Lista` es la lista de literales en que propiamente consiste la δ-cláusula.

El proceso que subyace a la saturación por δ-resolución que hemos implementado mediante `d_resol_anotada/2` es, como hemos comentado, dual al que sigue Otter. Parte de dos listas de δ-cláusulas, *usable* y *soporte*. La lista *usable* contiene en cada momento las δ-cláusulas a las que puede aplicarse la regla de δ-resolución, y la lista *soporte* reúne todas las demás. Mientras que el *soporte* no sea vacío y no se haya encontrado la δ-cláusula vacía, se hace:

1. Se selecciona como *δ-cláusula actual* la menos pesada del *soporte*. Las δ-cláusulas se ordenan por pesos mediante `ordenada_por_peso/2`, de forma que el peso de una δ-cláusula es igual al número de literales que contiene. Al tomar la menos pesada, por tanto, se elige la que tiene menos literales.
2. Se pasa la δ-cláusula actual desde *soporte* hasta `usable`.
3. Se calcula `Res`, la lista que contiene todas las δ-resolventes de la *δ-cláusula actual* con δ-cláusulas de *usable*. Esto se realiza mediante `d_resolventes(+C, +Usable, ?Res)`.

4. Se procesa cada una de las δ-cláusulas de `Res`, mediante `procesa(+Usable, +Soporte, +Res, ?Res2)`.

5. Se añaden al *soporte* todas las δ-cláusulas de `Res2`, que son las que han superado el procesamiento anterior.

El procedimiento para *procesar* una δ-cláusula `C` —realizado por `procesa/4`— es el siguiente:

1. Se escribe la δ-resolvente mediante `escribe_dresolvente(+C)`.

2. Si `C` es subsumida por alguna δ-cláusula de *usable* o *soporte* —mediante `subsumida(+C, +Conj)` se comprueba si `C` está subsumida por alguna δ-cláusula de `Conj`—, se descarta y se termina el proceso.

3. Si `C` es una δ-cláusula contradictoria —lo que se comprueba con `es_dclausula_contradictoria(+C)`, que busca si existen literales complementarios en `C`—, también se descarta y termina el proceso.

4. Si no se ha descartado `C`, se *retiene* y se numera mediante `numera(+C, -C1)`, obteniendo la δ-cláusula numerada `C1`.

5. Si `C1` tiene una solo literal, se le aplica *δ-resolución unitaria*, es decir, se busca la δ-cláusula que tenga el literal complementario en *usable* o *soporte* y se obtiene $\square$. Esto se realiza mediante `d_resol_unitaria/4`.

6. Si no se alcanzó $\square$ en el paso anterior, se aplica a `C1` *eliminación unitaria*, que consiste en eliminar de `C1` todos los literales tales que hay una δ-cláusula unitaria complementaria en *usable* o *soporte*. El predicado que lo realiza es `eliminacion_unitaria(+Usable, +Soporte, +C1, ?C2)`, obteniendo la δ-cláusula `C2`, que será igual a `C1` si no se logró aplicar *eliminación unitaria*.

7. El resultado de los pasos anteriores es la δ-cláusula *retenida* `C2`, que ha superado el procesamiento. Se escribe con `escribe_retenida(+C2)`, y se guarda. Según sea o no `C2` la δ-cláusula vacía se siguen procesando otras δ-resolventes o se detiene la búsqueda.

En nuestro ejemplo, hemos omitido la salida en pantalla correspondiente a la saturación por δ-resolución, debido a su extensión. Sin embargo, sí mostramos cuál es el soporte del que parte el proceso y cuál es el último usable —cuando el soporte se ha vaciado, sin encontrar la δ-cláusula vacía— con el que acaba el proceso de saturación.

```
Soporte para la búsqueda abductiva:
  1 [] [p, -q]
  2 [] [t, -s]
  3 [] [m, -p]
  4 [] [m, -t]
  5 [] [q, s]
```

[omitido el proceso de δ-resolución]

```
Último usable:
  1 [] [p, -q]
  2 [] [t, -s]
  5 [] [q, s]
  8 [5, 2] [q, t]
  9 [5, 1] [s, p]
  13 [8, 1] [t, p]
  15 [10, 7] [m]
```

Una vez alcanzado el último usable, que contiene las soluciones abductivas planas minimales, lo único que resta es seleccionar las explicativas. El predicado principal aquí es `busca_explic(+Planas, +Teo, +Obs, ?Explic)` que dadas las abducciones planas que forman parte de `Planas`, `Explic` toma sólo las que son abducciones explicativas, siendo `Teo` la lista de δ-cláusulass satisfactibles de la teoría, y `Obs` las de la negación de la observación. Antes de llamar a `busca_explic/4`, deben obtenerse las formas δ-clausales de la teoría y de la negación de la observación.

```
Forma d-clausal de Teo:
  Satisfactible:   [-m, -p, -t]
  Satisfactible:   [s, -m, -p]
  Contradictoria: [p, t, -p, -t]
  Contradictoria: [p, s, t, -p]
  Satisfactible:   [q, -m, -t]
```

```
      Satisfactible:   [q, s, -m]
      Contradictoria: [p, q, t, -t]
      Satisfactible:   [p, q, s, t]

    Forma d-clausal de -Obs:
      Satisfactible:   [-q]
      Satisfactible:   [-s]

    Abducciones explicativas:
      8 [5, 2] [q, t]
      9 [5, 1] [s, p]
      13 [8, 1] [t, p]
      15 [10, 7] [m]

RECURSOS EMPLEADOS:
   Tiempo (milisegundos):  20
   Espacio (bytes):        56136
   Inferencias:            9357
```

En cuanto a la eficiencia de esta implementación, el cuadro 4.3 recoge los recursos empleados durante la resolución de los problemas abductivos que aparecen en el apéndice C. También en este caso, como con el sistema de Aliseda —ver la información del cuadro 3.1, en la página 79— hay problemas que han quedado sin resolverse, pues se ha desbordado la memoria asignada a la pila global —también ahora de 25 megabytes—.

Un dato significativo es que en los problemas de las clases **conj**, **anidado** e **incon**, los recursos que este sistema invierte son, para las instancias más bajas, mayores que los que necesita el de Aliseda, pero la progresión es más baja, con lo que este sistema logra resolver más instancias. Esto se aprecia claramente en el caso de los problemas **incon(1)** e **incon(2)**. Mientras que en el primero de ellos el sistema de Aliseda gasta menos de la mitad de recursos que los que emplea la δ-resolución, en la siguiente instancia la eficiencia de la δ-resolución es del orden de 20 veces superior. Estos datos nos sugieren que la complejidad del cálculo de δ-resolución para la generación de abducciones explicativas es menor que la de las tablas semánticas.

	Instancia	*Milisegundos*	*Bytes*	*Inferencias*
implic	5	9	43212	6046
	10	1100	2746652	419254
	15	—	—	—
conj	3	0	14260	2064
	4	11	64376	8973
	5	120	457244	70285
	6	2161	4335676	712036
	7	—	—	—
anidado	3	10	31284	5348
	4	21	104836	20947
	5	149	314220	87663
incon	1	11	50232	7593
	3	20	120724	18851
	4	30	162908	25769
	5	50	211420	33767

Cuadro 4.3: Resultados para δ-resolución (primera versión)

Sin embargo, en las instancias mayores de los problemas **implic** y **conj** fracasa este sistema. El desbordamiento de la pila global se produce cuando el programa trata de seleccionar, de entre las abducciones planas, aquellas que sean explicativas. Concretamente, ocurre durante el cálculo de la forma normal disyuntiva de la teoría.

Los puntos débiles de este sistema son, por tanto, la complejidad de la transformación a forma normal disyuntiva —se requiere hacer tres transformaciones a lo largo de todo el proceso— y de nuevo, como en el sistema de Aliseda, la gran cantidad de operaciones conjuntistas que deben realizarse durante la selección de las abducciones explicativas, pues debe comprobarse, para cada una de las abducciones planas, si es consistente con las formas δ-clausales de la teoría y de la negación de la observación.

Tanto el sistema de Aliseda como el que acabamos de presentar realizan la selección de las abducciones explicativas después de generar todas las abducciones planas. Esto es algo que, como hemos visto en los dos últimos capítulos, no resulta nada ventajoso en cuanto a la eficiencia. Además, si no encontramos una manera de hacer que la *selección* deje de ser un paso —sumamente costoso— que se realiza tras la *generación*, no será fácil dar una respuesta positiva en cuanto a la posibilidad de integración de ambos procesos. Por ello, en el siguiente capítulo estudiamos cómo llevar a cabo esta tarea dentro del cálculo de δ-resolución. El haber logrado este objetivo nos parece que es el mayor éxito que podía tener nuestro cálculo de δ-resolución y, desde luego, una razón poderosa para poder considerarlo un *cálculo abductivo*.

Capítulo 5

Búsqueda de abducciones explicativas mediante δ-resolución

Al final del capítulo anterior llamamos la atención sobre la necesidad de integrar los procesos de *generación* y *selección* de hipótesis explicativas dentro de un mismo procedimiento abductivo. Esta es la misión que llevaremos a cabo, a través del cálculo de δ-resolución, en este capítulo. En la primera sección realizamos el desarrollo formal de los teoremas fundamentales que nos servirán para componer el proceso abductivo, que será presentado en la segunda sección. Por último, en la tercera sección se comenta la implementación que hemos realizado y se compara su eficiencia con la de los programas anteriormente presentados. Los comentarios sobre el proceso abductivo los posponemos para el siguiente capítulo, cuando presentemos algunas optimizaciones.

5.1. Estudio formal

Aunque en el capítulo 2 convenimos el uso de letras griegas minúsculas para referirnos a fórmulas de $\mathcal{L}_p$, así como de mayúsculas griegas para conjuntos de fórmulas, en lo sucesivo emplearemos Θ para hacer referencia a una fórmula que ocupa el rol de la teoría en un problema abductivo. Como ya explicamos al co-

mentar el teorema 2.26, Θ puede ser la conjunción —finita— de todas las fórmulas de la teoría. De esta manera, el estudio formal gana en claridad.

Definición 5.1 (Conjunto $\mathcal{A}b\delta(\Theta,\phi)$) *Dadas las fórmulas $\Theta,\phi \in \mathcal{L}_p$, definimos $\mathcal{A}b\delta(\Theta,\phi)$ como el conjunto de δ-cláusulas $\{\Sigma_1,\ldots,\Sigma_n\}$ tales que para cada i, $1 \leq i \leq n$ se verifican las tres condiciones siguientes:*

- *Σ_i es una abducción minimal de $\Theta \rightarrow \phi$, en el sentido de la definición 4.9.*
- $\Sigma_i \not\models \neg\Theta$
- $\Sigma_i \not\models \phi$

Teorema 5.2 *Dadas las fórmulas $\Theta,\phi \in \mathcal{L}_p$, $\mathcal{A}b\delta(\Theta,\phi)$ es el conjunto de soluciones abductivas explicativas minimales al problema abductivo $\langle\Theta,\phi\rangle$.*

Demostración:

Dadas Θ y ϕ, demostraremos que los requisitos para que una δ-cláusula Σ pertenezca a $\mathcal{A}b\delta(\Theta,\phi)$ son los mismos requisitos exigidos para que Σ sea una solución abductiva explicativa minimal al problema $\langle\Theta,\phi\rangle$. En primer lugar, el requisito primero de la definición 5.1 exige que Σ sea una abducción minimal de $\Theta \rightarrow \phi$. Pero esto, por la definición 4.9, conlleva que

$$\Sigma \models \Theta \rightarrow \phi$$

lo cual equivale a

$$\Theta, \Sigma \models \phi$$

que es el requisito para que Σ sea una solución abductiva plana al problema $\langle\Theta,\phi\rangle$. Además, la definición 4.9 exige que Σ sea minimal, es decir, que para toda $\Sigma' \subset \Sigma$ se cumpla $\Sigma' \not\models \Theta \rightarrow \phi$ y por tanto $\Theta, \Sigma' \not\models \phi$. También, por la definición 4.9, Σ debe ser satisfactible. De este modo, el primer requisito de la definición 5.1 se corresponde con que Σ sea una solución abductiva plana y minimal. Además, el segundo requisito exige

$$\Sigma \not\models \neg\Theta$$

que equivale a

$$\Theta, \Sigma \not\models \bot$$

se decir, que $\Theta \cup \Sigma$ sea satisfactible. Pero este es el requisito de consistencia para Σ. Igualmente, el tercer requisito de la definición 5.1 exige

$$\Sigma \not\models \phi$$

que es el requisito explicativo.

Por tanto, los tres requisitos de la definición 5.1 para que se cumpla $\Sigma \in \mathcal{A}b\delta(\Theta, \phi)$ son los mismos que los que se exigen para que Σ sea una solución abductiva minimal y explicativa al problema abductivo $\langle \Theta, \phi \rangle$. ■

Teorema 5.3 *Dadas las fórmulas $\alpha, \beta \in \mathcal{L}_p$, tales que A y B son respectivamente sus formas δ-clausales, si C es la forma δ-clausal de $\alpha \vee \beta$ se cumple:*

$$(A \cup B)^\delta = C^\delta$$

Demostración:

En primer lugar, probaremos que $A \cup B$ es equivalente a $\alpha \vee \beta$. Sea una valoración v que satisface $A \cup B$. Entonces, por la definición 4.2, v satisface alguna δ-cláusula de $A \cup B$, lo cual significa que v satisface alguna δ-cláusula de A o de B. Pero, por el teorema 4.3, A es equivalente a α y B a β, por ello v satisface α o β, con lo que, por evaluación del disyuntor, v satisface $\alpha \vee \beta$. Del mismo modo, si v satisface $\alpha \vee \beta$, por evaluación del disyuntor, v satisface α o β. Ello conlleva, por el teorema 4.3, que v satisface A o B, lo que por la definición 4.2 implica que v satisface alguna δ-cláusula de A o de B. En cualquier caso, se trata de una δ-cláusula de $A \cup B$, por lo que, de nuevo por la definición 4.2, v satisface $A \cup B$. De modo que $\alpha \vee \beta$ y $A \cup B$ son equivalentes.

Además, por el teorema 4.3, C y $\alpha \vee \beta$ son equivalentes, lo que, unido a lo que acabamos de probar, conlleva que $A \cup B$ y C son equivalentes. Pero por el teorema 4.14 esto implica que los δ-resolventes de ambos conjuntos son iguales, lo que, por el teorema 4.21, implica

$$(A \cup B)^\delta = C^\delta$$

que es lo que queríamos demostrar. ■

Corolario 5.4 *Dadas las fórmulas $\neg\alpha, \beta \in \mathcal{L}_p$, con formas δ-clausales A y B, respectivamente, si C es la forma δ-clausal de $\alpha \rightarrow \beta$, entonces*

$$C^\delta = (A \cup B)^\delta$$

Demostración:

Sea D la forma δ-clausal de $\neg\alpha \vee \beta$. Entonces, como $\vDash (\alpha \rightarrow \beta) \leftrightarrow (\neg\alpha \vee \beta)$, por el teorema 4.14 tenemos que C y D tendrán los mismos δ-resolventes minimales, lo que, por el teorema 4.21, hace que $C^\delta = D^\delta$. A su vez, por el teorema 5.3, tenemos que $(A \cup B)^\delta = D^\delta$. Por tanto, uniendo resultados,

$$(A \cup B)^\delta = C^\delta$$

■

Corolario 5.5 *Dadas las fórmulas $\Theta, \phi \in \mathcal{L}_p$, si N_Θ es la forma δ-clausal de $\neg\Theta$ y O la de ϕ, entonces $(N_\Theta \cup O)^\delta$ es el conjunto de soluciones abductivas minimales planas para $\langle\Theta, \phi\rangle$.*

Demostración:

Sea T la forma δ-clausal de $\Theta \rightarrow \phi$. Entonces T^δ, por el teorema 4.21, será el conjunto de δ-resolventes minimales de T, y por el corolario 4.12, también es el conjunto de abducciones minimales de $\Theta \rightarrow \phi$. Pero como vimos en la demostración del teorema 5.2, el conjunto de abducciones minimales de $\Theta \rightarrow \phi$ es igual al conjunto de soluciones abductivas minimales planas para el problema abductivo $\langle\Theta, \phi\rangle$. De modo que T^δ es el conjunto de soluciones abductivas minimales planas para el problema abductivo $\langle\Theta, \phi\rangle$.

Para acabar la demostración basta considerar que, por el corolario 5.4, se cumple

$$(N_\Theta \cup O)^\delta = T^\delta$$

por lo que $(N_\Theta \cup O)^\delta$ es el conjunto de soluciones abductivas minimales planas para $\langle\Theta, \phi\rangle$. ■

Teorema 5.6 *Sean $\alpha, \beta \in \mathcal{L}_p$ tales que sus formas δ-clausales sean, respectivamente, A y B. Además, $\Sigma \in (A \cup B)^\delta$. Entonces $\Sigma \models \alpha$ syss $\Sigma \in A^\delta$.*

Demostración:

Sea $\Sigma \in (A \cup B)^\delta$. Probaremos primero que si $\Sigma \in A^\delta$ entonces $\Sigma \models \alpha$, y a continuación lo recíproco.

Si $\Sigma \in A^\delta$ entonces, por el teorema 4.21, Σ es un δ-resolvente minimal de A, por el corolario 4.11, Σ es una abducción minimal de α, y por la definición 4.9, $\Sigma \models \alpha$.

Si $\Sigma \models \alpha$, demostremos que Σ es una abducción minimal de α. Las tres condiciones que deben darse para ello, por la definición 4.9, son:

1. Que Σ sea satisfactible. Sabemos que así ocurre, puesto que $\Sigma \in (A \cup B)^\delta$, y todas las δ-cláusulas no satisfactibles son eliminadas durante el proceso de saturación, como indica la definición 4.18.

2. Que $\Sigma \models \alpha$. Este requisito se cumple por hipótesis.

3. Que no exista ninguna δ-cláusula $\Sigma' \subset \Sigma$ tal que $\Sigma' \models \alpha$. No es posible, pues si existiera tal Σ', entonces si $\Sigma' \models \alpha$, por evaluación del disyuntor $\Sigma' \models \alpha \vee \beta$, y por la definición 4.9, Σ no podría ser una abducción minimal de $\alpha \vee \beta$. Pero Σ es una abducción minimal de $\alpha \vee \beta$, ya que $\Sigma \in (A \cup B)^\delta$, y por el teorema 5.3, si C es la forma δ-clausal de $\alpha \vee \beta$, entonces $C^\delta = (A \cup B)^\delta$, y por tanto $\Sigma \in C^\delta$. Pero por el teorema 4.21 sabemos que C^δ es el conjunto de δ-resolventes minimales de C y por el corolario 4.12 también es el conjunto de abducciones minimales de $\alpha \vee \beta$, así que Σ es una abducción minimal de $\alpha \vee \beta$.

■

Corolario 5.7 *Sean $\Theta, \phi \in \mathcal{L}_p$ tales que N_Θ es la forma δ-clausal de $\neg\Theta$ y O la de ϕ. Entonces se verifica,*

$$\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta \cup O)^\delta - (N_\Theta^\delta \cup O^\delta)$$

Demostración:

En primer lugar, por el teorema 5.2, $\mathcal{A}b\delta(\Theta, \phi)$ es el conjunto de soluciones abductivas minimales explicativas para el problema abductivo $\langle\Theta, \phi\rangle$. Demostremos que también lo es $(N_\Theta \cup O)^\delta - (N_\Theta^\delta \cup O^\delta)$.

Por el corolario 5.5, $(N_\Theta \cup O)^\delta$ es el conjunto de soluciones abductivas minimales y planas para el problema $\langle\Theta, \phi\rangle$. De modo que sólo nos queda demostrar que todas las soluciones abductivas minimales de $(N_\Theta \cup O)^\delta$ que no sean explicativas pertenecen a $(N_\Theta^\delta \cup O^\delta)$. En primer lugar, cada Σ que no sea consistente con la teoría cumple $\Theta, \Sigma \vDash \bot$, que equivale a $\Sigma \vDash \neg\Theta$. En este caso, por el teorema 5.6 tenemos que $\Sigma \in N_\Theta^\delta$. También cada Σ que no sea explicativa cumple $\Sigma \vDash \phi$. Entonces, también por el teorema 5.6 tenemos que $\Sigma \in O^\delta$. De modo que de entre las soluciones abductivas minimales planas de $(N_\Theta \cup O)^\delta$, las que no son consistentes pertenecen a N_Θ^δ y las no explicativas a O^δ. Por tanto, las que no sean consistentes o no sean explicativas pertenecen a $(N_\Theta^\delta \cup O^\delta)$, por lo que el conjunto de soluciones abductivas explicativas para el problema abductivo $\langle\Theta, \phi\rangle$ —que ya habíamos probado que era también $\mathcal{A}b\delta(\Theta, \phi)$— es

$$(N_\Theta \cup O)^\delta - (N_\Theta^\delta \cup O^\delta)$$

■

Teorema 5.8 *Para todo conjunto de δ-cláusulas A se cumple*

$$\vDash A \leftrightarrow A^\delta$$

Demostración:

Sea una valoración v que satisface A. Entonces, por la definición 4.2, v satisface alguna δ-cláusula $\Sigma \in A$ tal que $\Sigma = \{\lambda_1, \ldots, \lambda_n\}$. Por la definición 4.1 v satisface cada λ_i, $1 \leq n \leq n$. Ocurrirá uno de los siguientes casos:

1. Que $\Sigma \in A^\delta$. Entonces, por la definición 4.2, v satisface A^δ.

2. Que $\Sigma \notin A^\delta$. Como $\Sigma \in A$ y durante la saturación sólo se eliminan las δ-cláusulas no satisfactibles o subsumidas —según se especifica en la definición 4.18— y Σ es satisfactible puesto que v la satisface, entonces Σ estará subsumida por otra δ-cláusula Σ' de A^δ. Por la definición 4.16, esto significa que Σ' está compuesta por algunos de los literales λ_i, $1 \leq i \leq n$. Pero v satisface todos estos literales, así que por la definición 4.1 v satisface Σ' y por la definición 4.2, v satisface A^δ.

Consideremos ahora que v satisface A^δ. Entonces v satisface alguna δ-cláusula Σ de A^δ. Por el teorema 4.21, Σ es un δ-resolvente minimal de A, lo cual conlleva, por la definición 4.8 que $A \vdash_\delta \Sigma$, y por el teorema 4.6, que $\Sigma \vDash A$. De modo que v satisface A. ■

Corolario 5.9 *Dados dos conjuntos cualesquiera de δ-cláusulas A y B, se cumple*

$$(A^\delta \cup B^\delta)^\delta = (A \cup B)^\delta$$

Demostración:

Por el teorema 5.8, A es equivalente a A^δ y B a B^δ. Probemos que $A \cup B$ es también equivalente a $A^\delta \cup B^\delta$. Sea una valoración v que satisface $A \cup B$; entonces v satisface alguna δ-cláusula de A o de B, por la definición 4.2, y del mismo modo, v satisface A o B. Pero por ser A y A^δ equivalentes, así como B y B^δ, —teorema 5.8—, v satisface A^δ o B^δ, lo que por la definición 4.2 implica que v satisface alguna δ-cláusula de $A^\delta \cup B^\delta$, por lo que v satisface $A^\delta \cup B^\delta$.

Del mismo modo, si v satisface $A^\delta \cup B^\delta$, v satisface alguna δ-cláusula de A^δ o de B^δ —definición 4.2—, con lo que v satisface A^δ o B^δ. Pero por la equivalencia de A con A^δ y de B con B^δ —teorema 5.8—, v satisface A o B. Ello significa, por la definición 4.2, que v satisface alguna δ-cláusula de $A \cup B$, por lo que v satisface $A \cup B$.

Pero si $A \cup B$ y $A^\delta \cup B^\delta$ son equivalentes, por el teorema 4.14 sus δ-resolventes minimales son los mismos; de modo que, por el teorema 4.21,

$$(A^\delta \cup B^\delta)^\delta = (A \cup B)^\delta$$

■

Corolario 5.10 *Sean $\Theta, \phi \in \mathcal{L}_p$ dos fórmulas tales que N_Θ es la forma δ-clausal de $\neg\Theta$ y O la de ϕ. Entonces se verifica*

$$\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta)$$

Demostración:

Por el corolario 5.7, tenemos que

$$\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta \cup O)^\delta - (N_\Theta^\delta \cup O^\delta)$$

pero por el corolario 5.9 sabemos

$$(N_\Theta \cup O)^\delta = (N_\Theta^\delta \cup O^\delta)^\delta$$

de modo que uniendo ambos resultados obtenemos lo que queríamos probar. ■

Teorema 5.11 *Dadas $\alpha, \beta \in \mathcal{L}_p$, cuyas formas δ-clausales son, respectivamente, A y B, se verifica que $\beta \vDash \alpha$ syss*

$$(A \cup B)^\delta = A^\delta$$

Demostración:

En primer lugar, supongamos que $\beta \vDash \alpha$. Demostremos que $(A \cup B)^\delta = A^\delta$.

Sea $\Sigma \in A^\delta$. Entonces, por el teorema 4.21, Σ es un δ-resolvente minimal de A, y por el corolario 4.12, Σ es una abducción minimal de α que cumple los requisitos que establece la definición 4.9, por lo que,

1. Σ es satisfactible.
2. $\Sigma \vDash \alpha$
3. No existe $\Sigma' \subset \Sigma$ tal que $\Sigma' \vDash \alpha$

Veamos que Σ también es una abducción minimal de $\alpha \vee \beta$, es decir, cumple los siguientes requisitos:

1. Σ es satisfactible. Lo es, como hemos establecido en la anterior enumeración.
2. $\Sigma \vDash \alpha \vee \beta$. Puesto que, como hemos establecido, $\Sigma \vDash \alpha$, por evaluación del disyuntor $\Sigma \vDash \alpha \vee \beta$.
3. No existe $\Sigma' \subset \Sigma$ tal que $\Sigma' \vDash \alpha \vee \beta$. Si existiese tal Σ', por evaluación del disyuntor, $\Sigma' \vDash \alpha$ o bien $\Sigma' \vDash \beta$. Como $\beta \vDash \alpha$, en cualquier caso se cumpliría $\Sigma' \vDash \alpha$. Pero esto va contra lo que se ha establecido al ser Σ abducción minimal de α, por lo que no puede existir tal Σ'.

Así que Σ es una abducción minimal de $\alpha \vee \beta$. Por el corolario 4.12, Σ es un δ-resolvente minimal de C, siendo C la forma δ-clausal de $\alpha \vee \beta$. Por el teorema 4.21, $\Sigma \in C^\delta$. Pero por el corolario 5.3, $(A \cup B)^\delta = C^\delta$, por lo que $\Sigma \in (A \cup B)^\delta$.

Del mismo modo, si $\Sigma \in (A \cup B)^\delta$, tal como hemos visto, $\Sigma \in C^\delta$ —corolario 5.3—, con lo que Σ es un δ-resolvente minimal de C —teorema 4.21—, y por tanto una abducción minimal de $\alpha \vee \beta$ —corolario 4.12—. Por la definición 4.9 tenemos que $\Sigma \vDash \alpha \vee \beta$. De forma que toda valoración v que satisfaga Σ satisface o bien α o bien β. Pero si v satisface β también satisface α, puesto que $\beta \vDash \alpha$, con lo que toda valoración que satisfaga Σ satisface α, por lo que $\Sigma \vDash \alpha$. Entonces, por el teorema 5.6, $\Sigma \in A^\delta$.

Ya hemos probado que si $\beta \vDash \alpha$ entonces $(A \cup B)^\delta = A^\delta$. Probemos ahora lo recíproco.

Si $(A \cup B)^\delta = A^\delta$, y C es la forma δ-clausal de $\alpha \vee \beta$, por el teorema 5.3, $C^\delta = A^\delta$, lo que significa que C y A tienen los mismos δ-resolventes minimales

—teorema 4.21—, y que por tanto α y $\alpha \vee \beta$ tienen las mismas abducciones minimales —corolario 4.12—. Son abducciones minimales de $\alpha \vee \beta$ todas las δ-cláusulas Σ que cumplen:

1. Σ es satisfactible.
2. $\Sigma \vDash \alpha \vee \beta$
3. No existe Σ' tal que $\Sigma' \vDash \alpha \vee \beta$

Además, todas las δ-cláusulas Σ que cumplen los requisitos anteriores, al ser también abducciones minimales de α, cumplirán $\Sigma \vDash \alpha$.

Ahora, supongamos que $\beta \nvDash \alpha$. Entonces existe una valoración v tal que v satisface β, pero v no satisface α. Entonces, v satisface $\beta \wedge \neg\alpha$. Veamos que entonces habrá una δ-cláusula Ω^* que cumple los requisitos de la definición 4.9 para ser una abducción minimal de $\beta \wedge \neg\alpha$. Sea Ω la δ-cláusula $\{\lambda_1, \ldots, \lambda_n\}$ de tal modo que para las n variables proposicionales p_i, $1 \leq i \leq n$, que ocurren en $\beta \wedge \neg\alpha$, si v satisface p_i, entonces $\lambda_i = p_i$, y en otro caso $\lambda_i = \neg p_i$. Entonces resulta trivial que $\Omega \vDash \beta \wedge \neg\alpha$. Ahora bien, sea $\Omega^* \subseteq \Omega$ tal que $\Omega^* \vDash \beta \wedge \neg\alpha$ y no exista ninguna δ-cláusula $\Omega^\omega \subset \Omega^*$ tal que $\Omega^\omega \vDash \beta \wedge \neg\alpha$. Entonces Ω^* es una abducción minimal de $\beta \wedge \neg\alpha$, pues el único requisito de la definición 4.9 que queda por probar es su satisfactibilidad, asegurada al ser $\Omega^* \subseteq \Omega$ y Ω, por como se ha construido, no contiene literales complementarios. Pero si $\Omega^* \vDash \beta \wedge \neg\alpha$ también ocurre, por evaluación del conjuntor, $\Omega^* \vDash \beta$ y, por evaluación del disyuntor, $\Omega^* \vDash \beta \vee \alpha$. Por tanto, existe alguna δ-cláusula $\Omega^u \subseteq \Omega^*$ que es una abducción minimal de $\beta \vee \alpha$. Pero como el conjunto de abducciones minimales de $\beta \vee \alpha$ es igual que el de α, Ω^u es una abducción minimal de α. Por tanto, $\Omega^u \vDash \alpha$, y como $\Omega^u \subseteq \Omega$, entonces $\Omega \vDash \alpha$, pues cada valoración que satisfaga Ω satisface igualmente todos los literales de Ω^u y por tanto satisface α. Pero llegamos a una contradicción, pues, como antes establecimos, $\Omega \vDash \beta \wedge \neg\alpha$, y por evaluación del conjuntor, $\Omega \vDash \neg\alpha$; y esto no puede ocurrir, ya que Ω es satisfactible por construcción. Por tanto, negamos el supuesto con el que comenzamos este párrafo, y concluimos que $\beta \vDash \alpha$.

De modo que se cumple $\beta \vDash \alpha$ syss

$$(A \cup B)^\delta = A^\delta$$

■

Corolario 5.12 *Sean $\alpha, \beta \in \mathcal{L}_p$ cuyas formas δ-clausales son, respectivamente, A y B. Entonces se cumple que $\beta \vDash \alpha$ syss cada δ-cláusula $\Sigma \in B^\delta$ está subsumida por alguna $\Sigma' \in A^\delta$.*

Demostración:

Por el teorema 5.11, sabemos que $\beta \vDash \alpha$ syss

$$(A \cup B)^\delta = A^\delta$$

Además, por el corolario 5.9, tenemos que

$$(A \cup B)^\delta = (A^\delta \cup B^\delta)^\delta$$

Uniendo estos dos resultados, basta que demostremos que ocurre

$$(A^\delta \cup B^\delta)^\delta = A^\delta$$

syss cada δ-cláusula $\Sigma \in B^\delta$ está subsumida por alguna $\Sigma' \in A^\delta$.

En primer lugar, supongamos que toda δ-cláusula $\Sigma \in B^\delta$ está subsumida por alguna $\Sigma' \in A$. Entonces, al construir $(A^\delta \cup B^\delta)^\delta$ todas las δ-cláusulas de B^δ pueden eliminarse por ser subsumidas, por lo que nos queda $(A^\delta)^\delta$. Pero como A^δ ya esta saturado, y no puede aplicarse más la regla de δ-resolución, por la definición 4.18, y ninguna de sus δ-cláusulas es no satisfactoria ni subsumida, entonces, $(A^\delta)^\delta = A^\delta$, y por tanto, $(A^\delta \cup B^\delta)^\delta = A^\delta$.

Además, si $(A^\delta \cup B^\delta)^\delta = A^\delta$, ello significa que durante la saturación de $A^\delta \cup B^\delta$ las δ-cláusulas de B^δ que no pertenecen a A^δ se han eliminado. La definición 4.18 establece que sólo se eliminan, durante la saturación, las δ-cláusulas no satisfactibles o subsumidas. Pero las δ-cláusulas de B^δ no pueden ser

no satisfactibles, por la construcción de B^δ, por lo que cada una de ellas debe estar subsumida por alguna δ-cláusula de las que finalmente queda. Pero tales δ-cláusulas son las de A^δ. Por tanto, cada δ-cláusula de B^δ está subsumida por alguna de A^δ. ■

5.2. Integración de los procesos de generación y selección a través del cálculo de δ-resolución

El corolario 5.10 nos dice que dado un problema abductivo $\langle \Theta, \phi \rangle$, si N_Θ es la forma δ-clausal de $\neg\Theta$ y O la de ϕ, entonces

$$\boxed{\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta)}$$

es decir, podemos obtener las soluciones abductivas explicativas y minimales para el problema $\langle \Theta, \phi \rangle$ construyendo N_Θ^δ, en segundo lugar O^δ y, por último $(N_\Theta^\delta \cup O^\delta)^\delta$, lo que se hace saturando la unión de los conjuntos anteriormente obtenidos. Serán soluciones abductivas explicativas minimales todas las δ-cláusulas que pertenezcan a $(N_\Theta^\delta \cup O^\delta)^\delta$ pero no a $N_\Theta^\delta \cup O^\delta$. Esto nos da una idea del proceso abductivo que vamos a definir en esta sección. Proceso que consta, para cualesquiera teoría Θ y observación ϕ —sean o no un problema abductivo, pues dentro del propio proceso se incluye la comprobación—, de los siguientes pasos:

1. **Análisis de la teoría**. Este paso nos informa de si la teoría Θ es universalmente válida, no satisfactible o contingente. Sólo en este último caso continúa el proceso, ya que si la teoría es no satisfactible resulta trivial que cualquier ϕ será consecuencia lógica suya y, a su vez, si es universalmente válida ninguna explicación α podrá ser explicativa, pues si $\Theta, \alpha \vDash \phi$, entonces toda valoración que satisfaga Θ y α satisface ϕ, pero si Θ es universalmente válida, las valoraciones que satisfagan Θ y α son exactamente las que satisfacen α, por lo que $\alpha \vDash \phi$.

2. **Análisis de la observación**. Este paso nos informa de si la observación ϕ es universalmente válida, no satisfactible o contingente. Sólo continúa el proceso si la observación es contingente, pues si ϕ es universalmente válida, se trata de una tautología que no requiere explicación, y si ϕ fuera no satisfactible no puede nunca ser explicada por ninguna teoría satisfactible. De otro modo, sea α una explicación de ϕ en la teoría Θ. Entonces $\Theta, \alpha \vDash \phi$, lo que significa que toda valoración que satisfaga Θ y α debe satisfacer ϕ. Pero si $\Theta \cup \{\alpha\}$ fuera satisfactible, habría una valoración v que satisface $\Theta \cup \{\alpha\}$ y por tanto ϕ. Por ello, si ϕ no es satisfactible, no puede ser explicada en ninguna teoría satisfactible.

3. **Búsqueda de refutaciones**. En este paso se comprueba si es el caso o no de $\Theta \vDash \neg\phi$. Si ocurre que $\neg\phi$ es consecuencia lógica de Θ, entonces la observación ϕ es una refutación de Θ, pues es la negación de una consecuencia lógica de Θ. Sólo en caso de que no se encuentre la refutación continúa el proceso.

4. **Búsqueda de explicaciones**. Es el último paso del proceso, donde se comprueba si ocurre o no que $\Theta \vDash \phi$. En caso de que ϕ sea consecuencia lógica de la teoría Θ, no hay nada que explicar. En otro caso, se devuelven las abducciones explicativas que son solución de $\langle \Theta, \phi \rangle$.

En los siguientes apartados, nos detendremos en cada una de las fases del proceso abductivo.

5.2.1. Análisis de la teoría

Definición 5.13 (Análisis de la teoría) *Dado un par de fórmulas* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$ *definimos el* análisis de la teoría *como el proceso que consiste en:*

1. *Obtener* N_Θ*, forma* δ*-clausal de* $\neg\Theta$*.*

2. *Obtener* N_Θ^δ*.*

Llamamos a N_Θ^δ *el* conjunto de refutables *de* Θ*.*

Teorema 5.14 *Para toda forma δ-clausal A, la saturación A^δ sólo puede ser uno de los siguiente conjuntos:*

- $A^\delta = \emptyset$*, lo que ocurre syss A es no satisfactible.*
- $A^\delta = \{\square\}$*, lo que ocurre syss A es universalmente válida.*
- *Otro conjunto no vacío de δ-cláusulas que no incluye la δ-cláusula vacía, syss A es contingente.*

Demostración:

Sea A cualquier conjunto de δ-cláusulas, y A^δ su saturación por δ-resolución. Entonces necesariamente ocurre que o bien $\square \in A^\delta$ o que $\square \notin A^\delta$. Si $\square \in A^\delta$, entonces, como $\square$ subsume a cualquier otra δ-cláusula, que se eliminará durante la saturación, no puede haber en A^δ más que la δ-cláusula vacía. En caso de que $\square \notin A^\delta$, sólo pueden ocurrir dos cosas, que $A^\delta = \emptyset$ o que A^δ contenga al menos una δ-cláusula, pero como en este caso $\square \notin A^\delta$, ninguna de las δ-cláusulas de A^δ podrá ser la vacía. Por tanto los tres casos que pueden ocurrir son los que establece el teorema 5.14. Veamos ahora que cada uno de estos casos se da syss se cumplen las condiciones que se establecen.

En cuanto al primer caso, sea $A^\delta = \emptyset$. Entonces, por la definición 4.2, A^δ no es satisfactible, y por el teorema 5.8, tampoco lo será A. Del mismo modo, si A no es satisfactible, entonces veamos que A^δ debe ser el conjunto vacío. Si no lo fuera, y hubiese alguna δ-cláusula $\Sigma \in A^\delta$, entonces, por el teorema 4.21, Σ es un δ-resolvente minimal de A y por la definición 4.8, $A \vdash_\delta \Sigma$, y por el teorema 4.6, $\Sigma \vDash A$, por lo que, como por la definición 4.18, Σ debe ser satisfactible, pues en otro caso no estaría en A^δ, hay al menos una valoración que satisface Σ y por tanto A. Pero esto contradice el supuesto de que A no es satisfactible. Por tanto, negamos nuestro último supuesto y concluimos que $A^\delta = \emptyset$.

En el segundo caso, si $A^\delta = \{\square\}$ entonces por la definición 4.2, toda valoración que satisfaga $\square$ satisface A^δ. Pero por la definición 4.1, la δ-cláusula vacía es universalmente válida, con lo que toda valoración satisface A^δ, por lo que es

universalmente válido. Pero por el teorema 5.8, también A es universalmente válido. Del mismo modo, sea A universalmente válido. Entonces, por el teorema 4.7, $A \vdash_\delta \square$, por lo que en la construcción de A^δ debe aparecer $\square$, que como subsume todas las demás δ-cláusulas, resulta que $A^\delta = \{\square\}$.

El último caso, que dice que A^δ es un conjunto no vacío de δ-cláusulas que no incluye la δ-cláusula vacía syss A es contingente, resulta trivial a partir de los dos casos ya probados. ■

Corolario 5.15 *Para todo par de fórmulas proposicionales* $\langle \Theta, \phi \rangle$*, el* conjunto de refutables *de* Θ *sólo puede ser uno de entre los siguientes:*

1. *El conjunto vacío, en cuyo caso* Θ *es universalmente válida.*
2. *Un conjunto que contiene sólo la* δ*-cláusula vacía, en cuyo caso* Θ *es no satisfactible.*
3. *Un conjunto con al menos una* δ*-cláusula no vacía, en cuyo caso* Θ *es contingente.*

Demostración:

Sea N_Θ la forma δ-clausal de $\neg\Theta$, por lo que N_Θ^δ será el *conjunto de refutables* de Θ. Por el teorema 5.14 ocurrirá uno y sólo uno de los siguientes casos:

1. $N_\Theta^\delta = \emptyset$, syss N_Θ es no satisfactible, lo que por el teorema 4.3 conlleva que $\neg\Theta$ es no satisfactible, y por tanto Θ es universalmente válida.
2. $N_\Theta^\delta = \{\square\}$, syss N_Θ es universalmente válida, lo que por el teorema 4.3 conlleva que $\neg\Theta$ es universalmente válida, y por tanto Θ es no satisfactible.
3. Un conjunto con al menos una δ-cláusula no vacía, syss N_Θ es contingente, lo que lo que por el teorema 4.3 conlleva que $\neg\Theta$ es igualmente contingente, por lo que también lo será Θ.

■

5.2.2. Análisis de la observación

Definición 5.16 (Análisis de la observación) *Dado el par* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$ *definimos el* análisis de la observación *como el proceso que consiste en:*

1. *Obtener* O*, forma* δ*-clausal de* ϕ*.*
2. *Obtener* O^δ*.*

Llamamos a O^δ *el* conjunto de mínimos abductivos *de* ϕ*.*

Corolario 5.17 *Para todo par de fórmulas* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$*, el* conjunto de mínimos abductivos *de* ϕ *sólo puede ser uno de entre los siguientes:*

1. *El conjunto vacío, en cuyo caso* ϕ *es no satisfactible.*
2. *Un conjunto que contiene sólo la* δ*-cláusula vacía, en cuyo caso* ϕ *es universalmente válida.*
3. *Un conjunto con al menos una* δ*-cláusula no vacía, en cuyo caso* ϕ *es contingente.*

Demostración:

Si O es la forma δ-clausal de ϕ, por la definición 5.16, O^δ será el *conjunto de mínimos abductivos* de ϕ. Por el teorema 5.14, sólo ocurrirá una de entre las siguientes posibilidades:

1. $O^\delta = \emptyset$, syss O es no satisfactible, lo que por el teorema 4.3 conlleva que ϕ es no satisfactible.
2. $O^\delta = \{\square\}$, syss O es universalmente válido, lo que por el teorema 4.3 conlleva que ϕ es universalmente válida.
3. Un conjunto con al menos una δ-cláusula no vacía, en cuyo caso O es contingente, por lo que por el teorema 4.3 también lo será ϕ. ■

5.2.3. Búsqueda de refutaciones

Definición 5.18 (Búsqueda de refutaciones) *Dado el par* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$, *si* N_Θ^δ *es el conjunto de refutables de* Θ *y* O^δ *el conjunto de mínimos abductivos de* ϕ, *definimos la* búsqueda de refutaciones *para* $\langle \Theta, \phi \rangle$ *como el proceso que sigue los siguientes pasos:*

1. *Se define* $\mathcal{S}ubs(\Theta, \phi)$, *que será el conjunto de* δ*-cláusulas de* O^δ *subsumidas por alguna* δ*-cláusula de* N_Θ^δ.

2. *Se define* $\mathcal{R}ef(\Theta, \phi) = O^\delta - \mathcal{S}ubs(\Theta, \phi)$

3. *Se define* $\mathcal{B}ase(\Theta, \phi) = \mathcal{R}ef(\Theta, \phi) \cup N_\Theta^\delta$.

al conjunto $\mathcal{R}ef(\Theta, \phi)$ *lo llamamos* refutación *de* $\langle \Theta, \phi \rangle$, *y de* $\mathcal{B}ase(\Theta, \phi)$ *diremos que es su* base abductiva.

Corolario 5.19 *Para todo par* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$, *se cumple que* $\mathcal{R}ef(\Theta, \phi) = \emptyset$ *syss* $\Theta \models \neg\phi$.

Demostración:

Sea N_Θ la forma δ-clausal de $\neg\Theta$ y O la de ϕ. Por la definición 5.13, N_Θ^δ será el conjunto de refutables de Θ, y por la definición 5.16, O^δ el conjunto de mínimos abductivos de ϕ. Por tanto, $\mathcal{R}ef(\Theta, \phi)$, la *refutación* de $\langle \Theta, \phi \rangle$, contendrá, por la definición 5.18, las δ-cláusulas de O^δ no subsumidas por ninguna δ-cláusula de N_Θ^δ.

Pero el corolario 5.12 nos dice que $\alpha \models \neg\Theta$ syss cada δ-cláusula de O^δ está subsumida por alguna δ-cláusula de N_Θ^δ. Por tanto, $\alpha \models \neg\Theta$ syss $\mathcal{R}ef(\Theta, \phi) = \emptyset$. Pero $\alpha \models \neg\Theta$ syss $\Theta, \alpha \models \bot$, syss $\Theta \models \neg\alpha$. De modo que $\Theta \models \neg\alpha$ syss $\mathcal{R}ef(\Theta, \phi) = \emptyset$. ■

5.2.4. Búsqueda de explicaciones

Definición 5.20 (Búsqueda de explicaciones) *Dado el par* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$*, tal que su base abductiva es* $\mathcal{B}ase(\Theta, \phi)$*, definimos la búsqueda de explicaciones de* $\langle \Theta, \phi \rangle$ *como el proceso que consta de los dos pasos siguientes:*

1. *Obtención de* $\mathcal{B}ase(\Theta, \phi)^\delta$.
2. *Creación del conjunto* $\mathcal{E}xpl(\Theta, \phi) = \mathcal{B}ase(\Theta, \phi)^\delta - \mathcal{B}ase(\Theta, \phi)$

Corolario 5.21 *Para cualquier par de fórmulas proposicionales* $\langle \Theta, \phi \rangle$*, se verifica*

$$\mathcal{E}xpl(\Theta, \phi) = \mathcal{A}b\delta(\Theta, \phi)$$

Demostración:

Sea el par de fórmulas proposicionales $\langle \Theta, \phi \rangle$ tal que N_Θ es la forma δ-clausal de $\neg\Theta$ y O la de ϕ. Además, sea $\mathcal{B}ase(\Theta, \phi)$ la base abductiva. Por el corolario 5.10 tenemos:

$$\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta) \tag{5.1}$$

Además, $\mathcal{B}ase(\Theta, \phi)$ es, por la definición 5.18, la unión de todas las δ-cláusulas de N_Θ^δ con las δ-cláusulas de O^δ no subsumidas por alguna δ-cláusula de N_Θ^δ. Por tanto, $\mathcal{B}ase(\Theta, \phi)$ tiene todas las δ-cláusulas de $N_\Theta^\delta \cup O^\delta$ excepto algunas de las δ-cláusulas de O^δ subsumidas por ciertas δ-cláusulas de N_Θ^δ. Pero por el teorema 4.17 sabemos que como $\mathcal{B}ase(\Theta, \phi)$ resulta de quitar a $N_\Theta^\delta \cup O^\delta$ sólo δ-cláusulas subsumidas, ambos conjuntos tendrán los mismos δ-resolventes minimales, y por el teorema 4.21, $\mathcal{B}ase(\Theta, \phi)^\delta$ será igual a $(N_\Theta^\delta \cup O^\delta)^\delta$. Uniendo esto a (5.1), tenemos:

$$\mathcal{A}b\delta(\Theta, \phi) = \mathcal{B}ase(\Theta, \phi)^\delta - (N_\Theta^\delta \cup O^\delta) \tag{5.2}$$

Ahora probemos que para toda δ-cláusula $\Sigma \in \mathcal{B}ase(\Theta, \phi)^\delta$, se cumple que $\Sigma \in \mathcal{B}ase(\Theta, \phi)$ syss $\Sigma \in N_\Theta^\delta \cup O^\delta$.

Sea $\Sigma \in \mathcal{B}ase(\Theta, \phi)^\delta$. Si $\Sigma \in \mathcal{B}ase(\Theta, \phi)$ resulta trivial que $\Sigma \in N_\Theta^\delta \cup O^\delta$, puesto que por la definición 5.18 se cumple que $\mathcal{B}ase(\Theta, \phi) \subseteq N_\Theta^\delta \cup O^\delta$.

Supongamos que $\Sigma \in N^{\delta}_{\Theta} \cup O^{\delta}$ y probemos que $\Sigma \in \mathcal{B}ase(\Theta, \phi)$. Supongamos que no ocurre así. Entonces por construcción de $\mathcal{B}ase(\Theta, \phi)$ —definición 5.18—, Σ es una δ-cláusula de O^{δ} subsumida por alguna δ-cláusula Σ' de N^{δ}_{Θ} tal que $\Sigma' \in \mathcal{B}ase(\Theta, \phi)$. Pero entonces, por la definición 4.18, no puede ocurrir que $\Sigma \in \mathcal{B}ase(\Theta, \phi)^{\delta}$, puesto que si apareciese durante la saturación sería subsumida por Σ' o por alguna otra Σ'' que haya subsumido a Σ'. Pero esto es contradictorio con el supuesto de que $\Sigma \in \mathcal{B}ase(\Theta, \phi)^{\delta}$, por lo que negamos nuestro último supuesto y concluimos que $\Sigma \in \mathcal{B}ase(\Theta, \phi)$.

De modo que todas las δ-cláusulas de $\mathcal{B}ase(\Theta, \phi)^{\delta}$ están en $N^{\delta}_{\Theta} \cup O^{\delta}$ syss están en $\mathcal{B}ase(\Theta, \phi)$. Por tanto, el resultado de eliminar de $\mathcal{B}ase(\Theta, \phi)^{\delta}$ las δ-cláusulas de $N^{\delta}_{\Theta} \cup O^{\delta}$ es el mismo que el de suprimir de $\mathcal{B}ase(\Theta, \phi)^{\delta}$ las δ-cláusulas de $\mathcal{B}ase(\Theta, \phi)$. Aplicando esto a (5.2) tenemos:

$$\mathcal{A}b\delta(\Theta, \phi) = \mathcal{B}ase(\Theta, \phi)^{\delta} - \mathcal{B}ase(\Theta, \phi) \tag{5.3}$$

Pero por la definición 5.20, esto se transforma en:

$$\mathcal{A}b\delta(\Theta, \phi) = \mathcal{E}xpl(\Theta, \phi) \tag{5.4}$$

■

Corolario 5.22 *Para cualquier problema abductivo $\langle \Theta, \phi \rangle$, $\mathcal{E}xpl(\Theta, \phi)$ es el conjunto de sus soluciones abductivas explicativas minimales.*

Demostración:

Se trata de una consecuencia inmediata del teorema 5.2 y del corolario 5.21.

■

5.2.5. Procedimiento completo

Los resultados que hemos obtenido en los epígrafes anteriores sirven para definir el siguiente *procedimiento abductivo*, así como para probar su corrección.

Se trata del primero de los procesos abductivos que usan el cálculo de δ-resolución que definiremos en este trabajo. Posteriormente se introducirán ciertas modificaciones orientadas a mejorar su eficiencia.

Definición 5.23 (Proceso abductivo por δ-resolución) *Para cualquier par de fórmulas $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$ se define el proceso de generación de abducciones explicativas mediante δ-resolución según los siguientes pasos:*

1. ***Análisis de la teoría**. Se obtiene N_Θ, forma δ-clausal de $\neg\Theta$. Mediante saturación por δ-resolución se crea —como indica la definición 5.13— N_Θ^δ, conjunto de* refutables de la teoría Θ. *Entonces,*

   - *Si $N_\Theta^\delta = \emptyset$ el proceso acaba, y se devuelve un mensaje informando de que la teoría Θ es universalmente válida.*
   - *Si $N_\Theta^\delta = \{\square\}$ el proceso acaba, y se devuelve un mensaje informando de que la teoría Θ es no satisfactible.*
   - *En otro caso, se continúa con el paso siguiente.*

2. ***Análisis de la observación**. Se obtiene O, forma δ-clausal de ϕ. Mediante saturación por δ-resolución se crea —como indica la definición 5.16— O^δ, conjunto de* mínimos abductivos *de ϕ. Entonces,*

   - *Si $O^\delta = \emptyset$ el proceso acaba, y se devuelve un mensaje informando de que la observación ϕ es no satisfactible.*
   - *Si $O^\delta = \{\square\}$ el proceso acaba, y se devuelve un mensaje informando de que la observación ϕ es universalmente válida.*
   - *En otro caso, se continúa con el paso siguiente.*

3. ***Búsqueda de refutaciones**. A partir de N_Θ^δ y O^δ se realiza la búsqueda de refutaciones —definición 5.18— para $\langle \Theta, \phi \rangle$, creando los conjuntos $\mathcal{R}ef(\Theta, \phi)$,* refutación *de $\langle \Theta, \phi \rangle$ y $\mathcal{B}ase(\Theta, \phi)$, su* base abductiva. *Entonces,*

   - *Si $\mathcal{R}ef(\Theta, \phi) = \emptyset$, el proceso acaba y devuelve un mensaje que informa de que $\Theta \models \neg\phi$.*
   - *En otro caso, se continúa con el paso siguiente.*

4. ***Búsqueda de explicaciones***. *A partir de* $\mathcal{B}ase(\Theta, \phi)$, *se construye el conjunto* $\mathcal{E}xpl(\Theta, \phi)$ *—definición 5.20—. Entonces,*

   - *Si* $\mathcal{E}xpl(\Theta, \phi) = \{\Box\}$ *el proceso acaba y devuelve un mensaje que informa de que* $\Theta \models \phi$.
   - *Si* $\mathcal{E}xpl(\Theta, \phi) = \emptyset$ *el proceso acaba y devuelve un mensaje que informa de que no existen abducciones conjuntivas explicativas.*
   - *En otro caso, se devuelve* $\mathcal{E}xpl(\Theta, \phi)$ *e informa mediante un mensaje de que son todas las soluciones abductivas que son conjuntivas, explicativas y minimales.*

Corolario 5.24 *El procedimiento que se muestra en la definición 5.23 es correcto, es decir, para cualquier par de fórmulas* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$, *los mensajes que devuelve son siempre verdaderos.*

Demostración:

Veamos que para todo par de fórmulas $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$, el mensaje que devuelve el proceso de la definición 5.23 es correcto. Los mensajes que emite durante el proceso de análisis de la teoría —si N_Θ^δ es igual a $\emptyset$ o a $\{\Box\}$— son correctos, por el corolario 5.15. Del mismo modo lo son los mensajes que se emiten durante el proceso de análisis de la observación, por el corolario 5.17, así como los mensajes producidos durante la fase de búsqueda de refutaciones —corolario 5.19— y los que se devuelven durante la fase de búsqueda de explicaciones, por el corolario 5.22. ■

5.3. Implementación

En la página 273 mostramos un programa que implementa los cuatro pasos del proceso abductivo presentado en la sección anterior. El análisis de la teoría lo realiza el predicado `analiza_teo(+Teo, ?UsTeo)` que dada la teoría `Teo` procede a su análisis y, si la teoría es universalmente válida o no satisfactible, avisa con

un mensaje y unifica `UsTeo` con `fin`. En otro caso, avisa de que la teoría es contingente y unifica `UsTeo` con su *conjunto de refutables*. El proceso que sigue consta de los siguientes pasos:

1. Tras avisar de que comienza el análisis de la teoría, construye el conjunto de δ-cláusulas `ClTeo`, forma δ-clausal de `-Teo`. Para ello llama a `d_clausulas(-Teo,ClTEo)`.

2. Revisa cada una de las δ-cláusulas de `ClTeo`, elimina las que son contradictorias mediante `elimina_contradicciones(ClTeo, ClSatTeo)` y obtiene `ClSatTeo`, conjunto de δ-cláusulas satisfactibles de `-Teo`.

3. Si `ClSatTeo` es vacío, entonces avisa de que la teoría es válida y unifica `UsTEo` con `fin`.

4. En otro caso, numera las δ-cláusulas de `ClSatTeo` —mediante `anotado(ClSatTeo, SopTeo)`—, las ordena por peso —`ordenada_por_peso(SopTeo, SopTeo1)`— y comienza la saturación por δ-resolución con `d_resol_anotada([], SopTeo1, [], UsTeoProv)`.

5. Si ha llegado a la δ-cláusula vacía en la saturación, avisa de que la teoría es inconsistente, emplea `UsTeoProv` para escribir la prueba de $\square$, lo que realiza con `escribe_prueba(UsTeoProv)`, y unifica `UsTeo` con `fin`.

6. En otro caso, avisa de que la teoría es contingente, muestra el conjunto de refutables de `Teo` llamando a `muestra_usable(UsTeoProv)` y unifica `UsTeo`, resultado del proceso de análisis de la teoría, con `UsTeoProv`.

A continuación mostramos el código correspondiente a la fase de análisis de la teoría. El ciertas líneas aparecen números que se corresponden con cada uno de los pasos de esta fase, según se acaba de explicar.

```
analiza_teo(Teo,UsTeo) :-
        format('~N~n1. ANÁLISIS DE LA TEORÍA:~n',[]),
        d_clausulas(-Teo,ClTeo),                          % 1
        format('~N~n   d-cláusulas de la teoría:~n~n',[]),
        elimina_contradicciones(ClTeo,ClSatTeo),          % 2
```

```
( ClSatTeo = [] ->                                    % 3
    format('~N~n   *** TEORÍA VÁLIDA ***~n',[]),
    UsTeo = fin
; % ClSatTeo no es vacío.                            % 4
    format('~N~n   Soporte de la teoría:',[]),
    anotado(ClSatTeo,SopTeo),
    ordenada_por_peso(SopTeo,SopTeo1),
    d_resol_anotada([],SopTeo1,[],UsTeoProv),
    ( UsTeoProv = [_*_*[]|_] ->                       % 5
        format('~N~n   *** TEORÍA INCONSISTENTE ***~n',[]),
        escribe_prueba(UsTeoProv),
        UsTeo = fin
    ; % No se encuentra d-cláusula vacía             % 6
        format('~N~n   *** TEORÍA CONTINGENTE ***~n',[]),
        format('~N~n   Refutables de la teoría:~n',[]),
        muestra_usable(UsTeoProv),
        UsTeoProv = UsTeo)).
```

El análisis de la observación se realiza mediante una llamada al predicado `analiza_obs(+Obs, ?UsObs)`, que dada la observación `Obs` procede a su análisis y, si es universalmente válida o no satisfactible, avisa con un mensaje y unifica `UsObs` con `fin`. En otro caso, avisa de que la observación es contingente y unifica `UsObs` con su *conjunto de mínimos abductivos*. El proceso que sigue es muy similar al realizado para el análisis de la teoría, por lo que remitimos al apéndice B, donde puede consultarse.

En cuanto a la búsqueda de refutaciones, el predicado encargado de esta fase es `busca_ref(+UsTeo, +UsObs, ?Sop)` que, dados los conjuntos de δ-cláusulas `UsTeo` —refutables de la teoría— y `UsObs` —refutables de la observación— devuelve `Sop` que será igual a `fin`, en caso de que la observación refute la teoría, o bien la base abductiva. Procede de la siguiente manera:

1. Tras avisar con un mensaje de que comienza la fase de búsqueda de refutaciones, realiza una llamada a `elimina_subsumidas_var(UsTeo, UsObs, UsTeo, NoSubs, SopProv)`, que produce las listas `NoSubs`, de δ-cláusulas de `UsObs` no subsumidas por δ-cláusulas de `UsTeo` —se trata del conjunto

refutación de la definición 5.18—, y `SopProv`, que es la unión de `UsTeo` y `NoSubs` —conjunto *base* de la definición 5.18—.

2. Si `NoSubs` es vacía, entonces avisa con un mensaje de que la observación refuta la teoría y unifica `Sop` con `fin`.

3. En otro caso, unifica `Sop` con `SopProv`, avisa de que no hay refutación y muestra la base abductiva `Sop`.

El código correspondiente a esta fase es el que aparece a continuación. También se han numerado ciertas líneas que se corresponden con los pasos explicados en la enumeración anterior.

```
busca_ref(UsTeo,UsObs,Sop) :-
        format('~N~n3. BÚSQUEDA DE REFUTACIONES:~n',[]),
        elimina_subsumidas_var(UsTeo,UsObs,UsTeo,NoSubs,SopProv), % 1
        ( NoSubs = [] ->                                          % 2
            format('~N~n   *** LA OBSERVACIÓN REFUTA LA TEORÍA ***',
                   []),
            Sop = fin
        ; % No hay refutación                                     % 3
            SopProv = Sop,
            format('~N~n   *** NO HAY REFUTACIÓN ***~n',[]),
            format('~N~n   Base abductiva:~n',[]),
            muestra_usable(Sop)).
```

Por último, la fase de búsqueda de explicaciones es realizada por el predicado `busca_abd(+Sop)`, que toma como entrada `Sop`, la base abductiva, y procede según los siguientes pasos:

1. Tras avisar de que comienza la fase de búsqueda de explicaciones, ordena por peso las δ-cláusulas de `Sop` mediante `ordenada_por_peso(Sop, Sop1)`.

2. Llama a `valor_global(n_clausulas_retenidas, Min1)`, que tiene como objetivo guardar en `Min1` el número de δ-cláusulas que han sido retenidas a lo largo de todo el proceso abductivo, lo que servirá para determinar cuáles, de entre las δ-cláusulas que aparezcan en la saturación por δ-resolución de `Sop`, no estaban en `Sop`.

3. Satura mediante δ-resolución `Sop` haciendo una llamada al predicado `d_resol_anotada([], Sop1, [], Ref)` y obtiene `Ref`.

4. Si la δ-cláusula vacía está en `Ref`, entonces avisa de que la observación está explicada por la teoría y escribe la prueba de $\square$ desde `Sop` mediante `escribe_prueba(Ref)`.

5. En otro caso, vuelve a obtener el número de δ-cláusulas que han sido retenidas, que ahora será `Max`, y reúne en `Explic` todas las δ-cláusulas `N*H*C` que pertenezcan a `Ref` y que tengan como `N` un número comprendido entre `Min+1` y `Max`. Por tanto, `Explic` es el conjunto de δ-cláusulas que pertenecen a `Ref`, la saturación de `Sop`, pero no a `Sop`. Es decir, `Explic` es el conjunto de soluciones abductivas explicativas minimales.

6. Si `Explic` es vacío, se avisa de que no hay abducciones explicativas.

7. En otro caso, se muestra `Explic`, conjunto de abducciones explicativas.

También ahora mostramos el código correspondiente a la búsqueda de explicaciones.

```
busca_abd(Sop) :-
        format('~N~n4. BÚSQUEDA DE EXPLICACIONES:~n',[]),
        ordenada_por_peso(Sop,Sop1),                          % 1
        valor_global(n_clausulas_retenidas,Min1),             % 2
        d_resol_anotada([],Sop1,[],Ref),                      % 3
        ( Ref = [_*_*[]|_] ->                                 % 4
            format('~N~n   *** OBSERVACIÓN EXPLICADA POR TEORÍA ***',
                   []),
            escribe_prueba(Ref)
        ; % la observación no se deriva de la teoría          % 5
            valor_global(n_clausulas_retenidas,Max),
            Min is Min1+1,
            findall(N*H*C,(between(Min,Max,N),
                           memberchk(N*H*C,Ref)),Explic),
            ( Explic = [] ->                                  % 6
              format('~N~n   *** NO HAY ABDUCCIONES EXPLICATIVAS ***',
                     [])
            ; % hay abducciones explicativas                  % 7
```

```
        format('~N~n   Las abducciones explicativas son:',[]),
        muestra_usable(Explic))).
```

Las cuatro fases del proceso de búsqueda abductiva están coordinadas mediante el predicado `abduce_proc(+Teo, +Obs)` que, tras cada fase, si el argumento de respuesta aparece unificado con `fin` termina el proceso y devuelve los datos sobre los recursos —tiempo, memoria e inferencia— consumidos. En otro caso, pasa a la fase siguiente. Terminamos la explicación del programa mostrando la salida en pantalla para un problema muy sencillo.

```
?- abduce_proc(b => a, a).
PROBLEMA ABDUCTIVO:
   Teoría:       b=>a
   Observación:  a
1. ANÁLISIS DE LA TEORÍA:
   d-cláusulas de la teoría:
     Satisfactible:   [b, -a]
   Soporte de la teoría:
     1 [] [b, -a]
   d-cláusula actual #1: 1 [] [b, -a]
   *** TEORÍA CONTINGENTE ***
   Refutables de la teoría:
     1 [] [b, -a]
2. ANÁLISIS DE LA OBSERVACIÓN:
   d-cláusulas de la observación:
     Satisfactible:   [a]
   Soporte de la observación:
     2 [] [a]
   d-cláusula actual #2: 2 [] [a]
   *** OBSERVACIÓN CONTINGENTE ***
   Mínimos abductivos de la observación:
     2 [] [a]
3. BÚSQUEDA DE REFUTACIONES:
   *** NO HAY REFUTACIÓN ***
   Base abductiva:
     1 [] [b, -a]
     2 [] [a]
4. BÚSQUEDA DE EXPLICACIONES:
   d-cláusula actual #3: 2 [] [a]
   d-cláusula actual #4: 1 [] [b, -a]
```

```
    0 [1, 2] [b]
   ** RETENIDA: 3 [1, 2] [b]
    3 subsume a 1
  d-cláusula actual #5: 3 [1, 2] [b]
  Las abducciones explicativas son:
    3 [1, 2] [b]
RECURSOS EMPLEADOS:
  Tiempo (milisegundos):  0
  Espacio (bytes):        5332
  Inferencias:            688
```

	Instancia	*Milisegundos*	*Bytes*	*Inferencias*
implic	5	10	37088	4935
	10	91	159740	62418
	15	451	803748	363918
conj	3	9	9596	1141
	4	0	14092	1631
	5	11	19564	2253
	6	9	26072	3030
	7	11	33676	3988
anidado	3	9	26372	3281
	4	20	50796	7297
	5	20	92524	15640
incon	1	11	16936	2047
	3	20	37220	5344
	4	10	49164	7784
	5	20	63444	11139

Cuadro 5.1: Resultados para δ-resolución (segunda versión)

En el cuadro 5.1 mostramos los resultados de eficiencia para esta implementación, usando los problemas abductivos del apéndice C. Por primera vez, nos encontramos con una implementación capaz de resolver todos los problemas que le hemos planteado. Además, aunque las observaciones de estos problemas sean siempre literales, este programa, tal como el anterior, es capaz de explicar fórmulas más complejas. El aumento de eficiencia resulta obvio, si se compara con la primera implementación de la δ-resolución, cuyos resultados de eficiencia

aparecen en la página 116. El ahorro de recursos se debe principalmente a que en la primera implementación que hicimos de la δ-resolución había que obtener la forma δ-clausal de $\neg\Theta \rightarrow \phi$, después saturarla, y luego comprobar para cada δ-resolvente minimal si era o no una solución explicativa, lo que requería obtener las formas δ-clausales de Θ y de $\neg\phi$ y realizar una buena cantidad de operaciones conjuntistas. Sin embargo, esta implementación requiere:

- Obtener las formas δ-clausales de $\neg\Theta$ y de ϕ, que son respectivamente N_Θ y O. El esfuerzo total de obtener ambas formas δ-clausales es el mismo que el de obtener la forma δ-clausal de $\Theta \rightarrow \phi$, pues en este caso la fórmula se transforma en $\neg\Theta \vee \phi$, lo que hace que el conjunto de δ-cláusulas resultante sea igual que $N_\Theta \cup O$.
- Saturar N_Θ, O y $N_\Theta^\delta \cup O^\delta$. Pero el esfuerzo total de estas saturaciones no es, por término medio, mucho mayor que el que requiere saturar $N_\Theta \cup O$, que viene a ser la saturación que realiza la primera implementación de la δ-resolución.

Además, esta implementación no necesita hacer grandes operaciones conjuntistas para obtener las abducciones explicativas, sino simplemente seleccionar las δ-cláusulas de $(N_\Theta^\delta \cup O^\delta)^\delta$ que no están en $N_\Theta^\delta \cup O^\delta$, lo que puede hacerse con gran eficiencia si se acude a la numeración de las δ-cláusulas, como se ha comentado.

Así se explica que esta implementación del cálculo de δ-resolución sea mucho más eficiente que la anterior. Comparada con la implementación del sistema de Aliseda, cuyos datos de eficiencia aparecen en la página 79, observamos algo parecido a lo que ya comentamos al final del capítulo 4. Para los problemas **conj** y **anidado** el sistema de Aliseda rinde mejor en la primera instancia, mientras que para la instancia siguiente no le bastan los 25 megabytes de la pila global. Sin embargo esta implementación, aunque emplea más recursos para las primeras instancias, soluciona las últimas con un gasto de memoria que no llega a cuatro veces la que necesitó para las primeras. Esto indica que la progresión en coste computacional es muchísimo más baja, lo cual es indicio de una complejidad menor. En los problemas **incon** este sistema logra una mayor eficiencia que el de Aliseda incluso en la primera instancia, debido a la mayor complejidad de

estos problemas, que hace que el método de las tablas, por la gran cantidad de operaciones conjuntistas que requiere, no resulte eficiente. Por el contrario, en los problemas **implic** sigue sin superarse la eficiencia del sistema de Aliseda, que trabaja mejor con problemas abductivos más sencillos, en que la tabla de la teoría extendida con la negación de la observación tiene menos ramas abiertas —sólo una rama abierta, en caso de los problemas **implic**— lo que hace que la cantidad de operaciones conjuntistas que deben hacerse sea menos costosa que los procesos de δ-resolución.

Capítulo 6

Integración de las tablas semánticas con el cálculo de δ-resolución

En este capítulo proponemos dos modificaciones en el proceso abductivo definido en el capítulo anterior. La primera de ellas integra el método de las tablas semánticas con la búsqueda por δ-resolución. La segunda optimiza el proceso abductivo para observaciones literales. Comenzamos realizando un estudio formal, donde se demuestran los teoremas fundamentales que hacen posibles estas dos modificaciones. A continuación, la sección segunda está dedicada a explicar los cambios que se introducen en el proceso abductivo. La tercera sección comenta la implementación del nuevo proceso abductivo, que aparece en la página 277. La cuarta sección contiene ciertos comentarios y reflexiones suscitados por el proceso abductivo mediante δ-resolución, así como algunas comparaciones con el sistema de Aliseda. Por último, en la quinta sección se discute la posibilidad de generar, mediante δ-resolución, explicaciones que no sean necesariamente conjunciones de literales.

6.1. Estudio formal

Definición 6.1 (Conjunto $\mathcal{C}\delta$ de una tabla) *Dado un conjunto de fórmulas $\Gamma \subset \mathcal{L}_p$ cuya tabla semántica, $\mathcal{T}(\Gamma)$, tiene n ramas abiertas, $0 \leq n$, definimos $\mathcal{C}\delta(\mathcal{T}(\Gamma))$ como el conjunto de δ-cláusulas $\{\Sigma_1, \ldots, \Sigma_n\}$, tal que, para cada i, $0 \leq i \leq n$, Σ_i es el conjunto de literales de la i-ésima rama abierta de $\mathcal{T}(\Gamma)$.*

Teorema 6.2 *Dado un conjunto de fórmulas $\Gamma \subset \mathcal{L}_p$, el conjunto de δ-cláusulas $\mathcal{C}\delta(\mathcal{T}(\Gamma))$ es equivalente a la conjunción de todas las fórmulas de Γ.*

Demostración:

Sea $\Gamma \subset \mathcal{L}_p$ y $\mathcal{T}(\Gamma)$ la tabla semántica de Γ, de forma que $\mathcal{C}\delta(\mathcal{T}(\Gamma)) = \{\Sigma_1, \ldots, \Sigma_n\}$, $0 \leq n$. Debemos demostrar que toda valoración v que satisfaga todas las fórmulas de Γ, satisface también $\mathcal{C}\delta(\mathcal{T}(\Gamma))$, así como lo recíproco.

En primer lugar, sea v una valoración que satisface cada una de las fórmulas de Γ. Remitimos a la demostración del teorema 3.13, donde se probó que v debe satisfacer todas las fórmulas de al menos una cierta rama de $\mathcal{T}(\Gamma)$, y que dicha rama debe ser abierta. Supongamos que se trata de la j-ésima rama abierta. Pero si v satisface todas sus fórmulas, debe satisfacer también todos sus literales, por lo que satisface cada uno de los elementos de Σ_j, $1 \leq j \leq n$. Pero entonces, por la definición 4.1, v satisface Σ_j, y por la definición 4.2, v satisface $\mathcal{C}\delta(\mathcal{T}(\Gamma))$.

Ahora, supongamos que v satisface $\mathcal{C}\delta(\mathcal{T}(\Gamma))$. Por la definición 4.2, ello significa que v satisface alguna Σ_j para $1 \leq j \leq n$. Por la definición 4.1, ello implica que v satisface todos los literales de Σ_j, y por la definición 6.1 que v satisface todos los literales de la j-ésima rama abierta de $\mathcal{T}(\Gamma)$. Veamos que v satisface también todas las fórmulas de dicha rama, y por tanto todas las fórmulas de Γ. La prueba, muy similar a la que dimos para el teorema 3.12, procede por inducción sobre el grado lógico de cada una de las fórmulas de la j-ésima rama abierta de $\mathcal{T}(\Gamma)$. En cuanto a las fórmulas de grado 0, es decir, literales positivos, son satisfechas por v, puesto que pertenecen a Σ_j —por construcción, según la definición 6.1—. También los literales negativos son satisfechos por v, por la misma razón. Supongamos que v satisface todas las fórmulas de la j-ésima rama abierta

hasta las de grado menor o igual a m. Veamos qué ocurre con las fórmulas de grado $m+1$. Como vimos en la observación 3.3, cada una de tales fórmulas podrá ser —excluyendo los literales, que ya hemos probado que son satisfechos por v— sólo de uno de los siguientes tipos:

1. $\neg\neg\phi$. Entonces, durante la construcción de la tabla, según indica la definición 3.10, se debió aplicar la regla de doble negación al nodo $\neg\neg\phi$ añadiendo ϕ a cada una de las ramas que compartieran dicho nodo, por lo que si $\neg\neg\phi$ está en la rama j, también lo estará ϕ. Pero ϕ es de grado lógico menor a m, por lo que, por hipótesis de inducción, v satisface ϕ y, por evaluación del negador, v satisface $\neg\neg\phi$.

2. Una fórmula θ de tipo α de componentes ϕ y ψ. Entonces, durante la construcción de la tabla, como indica la definición 3.10, se debió aplicar la regla α al nodo θ añadiendo los nodos ϕ y ψ a cada una de las ramas de la tabla que comparten el nodo θ, por lo que si θ aparece en la rama j, también deben aparecer ϕ y ψ. Por hipótesis de inducción, v satisface ϕ y ψ, puesto que el grado lógico de cada una de ellas debe ser menor o igual a m. Por la observación 3.2, esto implica que v satisface θ.

3. Una fórmula θ de tipo β de componentes ϕ y ψ. Entonces, por la definición 3.10, antes de completar la tabla se debió aplicar la regla β al nodo θ separando en dos cada una de las ramas en las que θ aparece, y añadiendo a una de ellas el nodo ϕ, y a otra, ψ. Por tanto, como θ aparece en la rama j, también debe aparecer o bien ϕ o ψ, por lo que por hipótesis de inducción v satisface a alguna de ellas, puesto que ambas tienen grado lógico menor o igual a m. Entonces, por la observación 3.2, esto implica que v satisface θ.

De modo que si v satisface $\mathcal{C}\delta(\mathcal{T}(\Gamma))$, satisface todas las fórmulas de al menos una rama de $\mathcal{T}(\Gamma)$, pero como todas las fórmulas de Γ pertenecen a cada rama de $\mathcal{T}(\Gamma)$, v satisface todas las fórmulas de Γ. ■

Corolario 6.3 *Dada una fórmula ϕ, el conjunto $\mathcal{C}\delta(\mathcal{T}(\phi))$ es su forma δ-clausal.*

Demostración:

Dado que, por el teorema 6.2, $\mathcal{C}\delta(\mathcal{T}(\phi))$ es una forma δ-clausal equivalente a ϕ, por la definición 4.2 podemos considerar a $\mathcal{C}\delta(\mathcal{T}(\phi))$ la forma δ-clausal de ϕ.

■

Teorema 6.4 *Sea el par* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$, *tal que* N_Θ *es la forma* δ*-clausal de* $\neg\Theta$ *y* ϕ *es un literal. Entonces,*

$$\mathcal{A}b\delta(\Theta, \phi) = \{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N_\theta^\delta, \ \ \overline{\phi} \in \Sigma\}$$

Demostración:

Dado que al ser ϕ un literal, su forma δ-clausal será $\{\{\phi\}\}$, por el corolario 5.10 se verifica

$$\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta^\delta \cup \{\{\phi\}\}^\delta)^\delta - (N_\Theta^\delta \cup \{\{\phi\}\}^\delta) \tag{6.1}$$

Pero además, resulta trivial, a partir de la definición 4.18, que $\{\{\phi\}\}^\delta = \{\{\phi\}\}$, por lo que aplicando esta igualdad a (6.1) resulta

$$\mathcal{A}b\delta(\Theta, \phi) = (N_\Theta^\delta \cup \{\{\phi\}\})^\delta - (N_\Theta^\delta \cup \{\{\phi\}\}) \tag{6.2}$$

Así que debemos demostrar lo siguiente

$$(N_\Theta^\delta \cup \{\{\phi\}\})^\delta - (N_\Theta^\delta \cup \{\{\phi\}\}) = \{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N_\theta^\delta, \ \ \overline{\phi} \in \Sigma\} \tag{6.3}$$

Sea $\Lambda \in (N_\Theta^\delta \cup \{\{\phi\}\})^\delta - (N_\Theta^\delta \cup \{\{\phi\}\})$. Entonces, se cumple que $\Lambda \in (N_\Theta^\delta \cup \{\{\phi\}\})^\delta$, $\Lambda \notin N_\Theta^\delta$ y $\Lambda \notin \{\{\phi\}\}$. Queremos probar que $\Lambda \in \{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N_\theta^\delta, \ \overline{\phi} \in \Sigma\}$. Para ello, basta con demostrar los dos siguientes objetivos: $\Lambda \cup \{\overline{\phi}\} \in N_\Theta^\delta$ y $\overline{\phi} \notin \Lambda$.

Como $\Lambda \in (N_\Theta^\delta \cup \{\{\phi\}\})^\delta$, entonces, por el teorema 4.21, y por la definición 4.8, sabemos que $N_\Theta^\delta \cup \{\{\phi\}\} \vdash_\delta \Lambda$. Llamemos $\mathcal{D}em$ a la prueba de Λ a partir de $N_\Theta^\delta \cup \{\{\phi\}\}$, y construyamos otra prueba, a la que llamaremos $\mathcal{D}em'$, paralela a $\mathcal{D}em$, y que transcurre igual a ésta, pero sólo a partir de δ-cláusulas de

N^{δ}_{Θ}, de modo similar a como se hizo en la prueba del teorema 4.10. La diferencia entre ambas pruebas es que cuando en $\mathcal{D}em$ se aplica la regla de δ-resolución a alguna δ-cláusula que no aparece en $\mathcal{D}em'$, en ésta no se hace nada. Como en $\mathcal{D}em$ la única intervención que puede hacer la δ-cláusula $\{\phi\}$ es eliminar literales $\neg\phi$ de las δ-cláusulas que aparecen, y la única diferencia entre $\mathcal{D}em$ y $\mathcal{D}em'$ es que en ésta no se usa la δ-cláusula $\{\phi\}$, la última δ-cláusula a la que se llega en $\mathcal{D}em'$ sólo puede ser una de las dos siguientes posibilidades:

- La propia Λ. En tal caso, pues, $N^{\delta}_{\Theta} \vdash_{\delta} \Lambda$. Pero como N^{δ}_{Θ} ya está saturado, todas las δ-cláusulas a las que puede llegarse desde él por δ-resolución son, por la definición 4.18, o bien no satisfactibles, o subsumidas por otras δ-cláusulas de N^{δ}_{Θ}, o bien δ-cláusulas del propio conjunto N^{δ}_{Θ}. Pero ninguna de estas tres posibilidades puede ocurrir. Veamos,

  - Λ no puede ser no satisfactible, pues tenemos que $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, y una δ-cláusula no satisfactible no puede pertenecer a ninguna saturación, por la misma definición 4.18.
  - Tampoco Λ puede estar subsumida por otra δ-cláusula de N^{δ}_{Θ}, pues en tal caso también estaría subsumida por una δ-cláusula de $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$, y no aparecería en la saturación de este último conjunto —definición 4.18—, como sin embargo ocurre.
  - Tampoco puede ser que $\Lambda \in N^{\delta}_{\Theta}$, pues hemos partido de que esto no ocurre.

  Por tanto, sólo puede ocurrir que $\mathcal{D}em'$ llegue a

- $\Lambda \cup \{\overline{\phi}\}$. En este caso, $N^{\delta}_{\Theta} \vdash_{\delta} \Lambda \cup \{\overline{\phi}\}$. Pero, por ser N^{δ}_{Θ} un conjunto de δ-cláusulas saturado, como en el anterior caso, sólo puede ocurrir, por la definición 4.18, que $\Lambda \cup \{\overline{\phi}\}$ sea no satisfactible, o que esté subsumida por alguna δ-cláusula de N^{δ}_{Θ}, o que pertenezca a este conjunto. Veamos cada caso,

  - No puede ser que $\Lambda \cup \{\overline{\phi}\}$ no sea satisfactible, como ahora veremos. Sabemos que $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, por lo que al pertenecer Λ a una saturación no puede contener literales complementarios —por la defi-

nición 4.18—. Por tanto, si $\Lambda \cup \{\overline{\phi}\}$ no fuera satisfactible es que $\phi \in \Lambda$. Caben entonces dos posibilidades:

- $\Lambda = \{\phi\}$. Pero esto no puede ocurrir, pues sabemos $\Lambda \notin \{\{\phi\}\}$.
- $\Lambda = \{\phi\} \cup \Pi$ para algún conjunto de literales Π no vacío y distinto de $\{\phi\}$. Pero tampoco puede ocurrir esto, pues entonces Λ sería subsumida por $\{\phi\}$ —definición 4.16—, y no pertenecería a la saturación de $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$ —definición 4.18[1]—, pero tenemos que $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$.

Por tanto, no puede ser que $\Lambda \cup \{\overline{\phi}\}$ sea no satisfactible.

- Supongamos que $\Lambda \cup \{\overline{\phi}\}$ está subsumida por otra δ-cláusula $\Lambda' \in N^{\delta}_{\Theta}$ tal que $\Lambda' \subset \Lambda$. Sabemos que como $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, Λ no puede estar subsumida por tal Λ', a no ser que $\Lambda' = \Lambda$, de modo que Λ' no puede ser un subconjunto propio de Λ. Caben por tanto dos posibilidades:
  - $\Lambda' = \Lambda$. No puede ser, puesto que $\Lambda \notin N^{\delta}_{\Theta}$, y hemos partido de que $\Lambda' \in N^{\delta}_{\Theta}$.
  - $\Lambda' \neq \Lambda$. Pero como Λ' no es subconjunto propio de Λ pero sí de $\Lambda \cup \{\overline{\phi}\}$, debe ocurrir que $\overline{\phi} \in \Lambda'$. Como hemos supuesto $\Lambda' \in N^{\delta}_{\Theta}$, al hacer la saturación $(N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$ puede aplicarse la regla de δ-resolución a las δ-cláusulas Λ' y $\{\phi\}$, y obtenerse como δ-resolvente $\Lambda' - \{\overline{\phi}\}$. Pero como $\Lambda' \subset \Lambda \cup \{\overline{\phi}\}$, también ocurrirá $\Lambda' - \{\overline{\phi}\} \subset \Lambda$, con lo que Λ estaría subsumida por una de las δ-cláusulas que aparece durante la saturación de $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$. Pero esto contradice lo que tenemos de partida: $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$.

  En ambos casos llegamos a alguna contradicción, por lo que negamos nuestro último supuesto, y concluimos que $\Lambda \cup \{\overline{\phi}\}$ no está subsumida por ninguna otra δ-cláusula de N^{δ}_{Θ} que sea subconjunto propio suyo.
- $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$. Es la única posibilidad que queda sin descartar, por lo que debemos considerarla probada.

Por tanto, ya hemos probado el primer objetivo: $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$.

[1]Dada la extensión de la presente prueba, en adelante omitimos las referencias a las definiciones, teoremas y corolarios que se emplean, pues son los mismos que hemos usado hasta ahora.

En cuanto al segundo objetivo, $\overline{\phi} \notin \Lambda$, lo probaremos por reducción al absurdo. Supongamos que $\overline{\phi} \in \Lambda$. Entonces, como $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, tenemos que $N^{\delta}_{\Theta} \cup \{\{\phi\}\} \vdash_{\delta} \Lambda$, por lo que, al ser $\Lambda - \{\overline{\phi}\}$ un δ-resolvente de Λ —que hemos supuesto que contiene $\overline{\phi}$— y $\{\phi\}$, se cumple que $N^{\delta}_{\Theta} \cup \{\{\phi\}\} \vdash_{\delta} \Lambda - \{\overline{\phi}\}$, y como $\Lambda - \{\overline{\phi}\} \subset \Lambda$, entonces no podría ser $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, al estar Λ subsumida por una δ-cláusula que es subconjunto propio suyo y que también se obtiene desde $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$. Pero esto contradice nuestro punto de partida, por el que $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$.

De modo que para toda $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta} - (N^{\delta}_{\Theta} \cup \{\{\phi\}\})$ se cumple que $\Lambda \in \{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N^{\delta}_{\theta},\ \ \overline{\phi} \in \Sigma\}$. A continuación veremos lo recíproco.

Supongamos cierta δ-cláusula Λ tal que $\Lambda \in \{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N^{\delta}_{\theta},\ \ \overline{\phi} \in \Sigma\}$. Por tanto, tenemos que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$ y que $\overline{\phi} \notin \Lambda$. Para probar que $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta} - (N^{\delta}_{\Theta} \cup \{\{\phi\}\})$, deben demostrarse los tres siguientes objetivos: $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, $\Lambda \notin N^{\delta}_{\Theta}$ y $\Lambda \notin \{\{\phi\}\}$.

En cuanto al primer objetivo, $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta}$, exige a su vez tres requisitos:

- Λ es satisfactible. Debe serlo, puesto que partimos de que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$, y como las δ-cláusulas no satisfactibles son eliminadas durante la saturación, $\Lambda \cup \{\overline{\phi}\}$ debe ser satisfactible, y por tanto también Λ.

- $N^{\delta}_{\Theta} \cup \{\{\phi\}\} \vdash_{\delta} \Lambda$. También ocurre, puesto que Λ es un δ-resolvente de $\Lambda \cup \{\overline{\phi}\}$ y $\{\phi\}$, y ambas δ-cláusulas pertenecen a $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$, ya que partimos de que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$.

- No existe ninguna $\Lambda' \subset \Lambda$ tal que $N^{\delta}_{\Theta} \cup \{\{\phi\}\} \vdash_{\delta} \Lambda'$. Supongamos que sí existe. Entonces, sea $\mathcal{D}em$ la prueba de Λ' desde $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$. Igual que hicimos antes, construimos otra prueba $\mathcal{D}em'$ paralela a $\mathcal{D}em$ pero sólo a partir de δ-cláusulas de N^{δ}_{Θ}. La δ-cláusula a la que llega $\mathcal{D}em'$ puede ser simplemente Λ' o bien $\Lambda' \cup \{\overline{\phi}\}$. Veamos cada caso:

  - $N^{\delta}_{\Theta} \vdash_{\delta} \Lambda' \cup \{\overline{\phi}\}$. Pero como $\Lambda' \subset \Lambda$ y $\overline{\phi} \notin \Lambda$, entonces $\Lambda' \cup \{\overline{\phi}\} \subset \Lambda \cup \{\overline{\phi}\}$. Así que como $\Lambda' \cup \{\overline{\phi}\}$, que es δ-resolvente de N^{δ}_{Θ}, es subconjunto propio

de la δ-cláusula $\Lambda \cup \{\overline{\phi}\}$, y por tanto la subsume, no podría ocurrir $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$, lo que de hecho ocurre.

- $N^{\delta}_{\Theta} \vdash_{\delta} \Lambda'$. Pero como $\Lambda' \subset \Lambda$, también $\Lambda' \subset \Lambda \cup \{\overline{\phi}\}$, con lo que al ser Λ' demostrable a partir de N^{δ}_{Θ}, y cumplirse que $\overline{\phi} \notin \Lambda$, no puede ser que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$, lo que hemos visto que sí ocurre.

Por tanto, en ambos casos llegamos a una —la misma— contradicción con uno de nuestros supuestos de partida. Así que negamos nuestro último supuesto, concluyendo que no existe ninguna $\Lambda' \subset \Lambda$ tal que $N^{\delta}_{\Theta} \cup \{\{\phi\}\} \vdash_{\delta} \Lambda'$, lo cual era el objetivo que debíamos probar.

En cuanto al segundo objetivo, $\Lambda \notin N^{\delta}_{\Theta}$, supongamos que no es cierto, y que por tanto $\Lambda \in N^{\delta}_{\Theta}$, de lo que tenemos $N^{\delta}_{\Theta} \cup \{\{\phi\}\} \vdash_{\delta} \Lambda$. Entonces, como partimos de que $\overline{\phi} \notin \Lambda$, tenemos también $\Lambda \subset \Lambda \cup \{\overline{\phi}\}$; pero, al ser $\Lambda \cup \{\overline{\phi}\}$ subsumida por su subconjunto propio Λ, siendo Λ demostrable por δ-resolución desde $N^{\delta}_{\Theta} \cup \{\{\phi\}\}$, no es posible que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$, pero esto sí ocurre, pues es una consecuencia de nuestro punto de partida. Por tanto, negamos nuestro último supuesto, y concluimos que $\Lambda \notin N^{\delta}_{\Theta}$.

Para terminar, sólo resta probar $\Lambda \notin \{\{\phi\}\}$. Procedemos también por reducción al absurdo, suponiendo que $\Lambda \in \{\{\phi\}\}$, por lo que $\Lambda = \{\phi\}$. Pero sabemos que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$, y como también $\Lambda \cup \{\overline{\phi}\} = \{\phi, \overline{\phi}\}$, entonces $\Lambda \cup \{\phi\}$ es una δ-cláusula no satisfactible, al contener dos literales complementarios. Esto se contradice con que $\Lambda \cup \{\overline{\phi}\} \in N^{\delta}_{\Theta}$, pues en una saturación no puede haber δ-cláusulas no satisfactibles. Por tanto, tenemos que negar nuestro último supuesto y concluir que $\Lambda \notin \{\{\phi\}\}$.

Con los tres objetivos que acabamos de demostrar, ya tenemos que $\Lambda \in (N^{\delta}_{\Theta} \cup \{\{\phi\}\})^{\delta} - (N^{\delta}_{\Theta} \cup \{\{\phi\}\})$, con lo que, unido a lo que ya habíamos probado, se demuestra la identidad expresada en (6.3), y con ella el teorema 6.4. ■

Teorema 6.5 *Dada una fórmula satisfactible $\Theta \in \mathcal{L}_p$, tal que la forma δ-clausal de $\neg\Theta$ es N_{Θ}, y el literal $\phi \in \mathcal{L}_p$, se cumple que $\Theta \models \phi$ syss $\{\overline{\phi}\} \in N^{\delta}_{\Theta}$.*

Demostración:

Sea Θ una fórmula satisfactible tal que la forma δ-clausal de $\neg\Theta$ es N_Θ, y ϕ un literal. Si $\overline{\phi}$ es el literal complementario de ϕ, su forma δ-clausal será $\{\{\overline{\phi}\}\}$, y además $\{\{\overline{\phi}\}\}^\delta = \{\{\overline{\phi}\}\}$. Entonces se cumple $\Theta \models \phi$ syss $\Theta, \overline{\phi} \models \bot$, syss $\overline{\phi} \models \neg\Theta$, pero por el corolario 5.12 tenemos que esto ocurre syss cada δ-cláusula $\Sigma \in \{\{\overline{\phi}\}\}$ está subsumida por alguna δ-cláusula $\Sigma' \in N^\delta_\Theta$. Recapitulando, tenemos que $\Theta \models \phi$ syss cada δ-cláusula $\Sigma \in \{\{\overline{\phi}\}\}$ está subsumida por alguna δ-cláusula $\Sigma' \in N^\delta_\Theta$. Por tanto, para terminar la prueba sólo queda comprobar si se cumple que cada δ-cláusula $\Sigma \in \{\{\overline{\phi}\}\}$ está subsumida por alguna δ-cláusula $\Sigma' \in N^\delta_\Theta$ syss $\{\overline{\phi}\} \in N^\delta_\Theta$.

En primer lugar, supongamos que cada δ-cláusula $\Sigma \in \{\{\overline{\phi}\}\}$ está subsumida por alguna δ-cláusula $\Sigma' \in N^\delta_\Theta$. Ello significa que hay en N^δ_Θ una δ-cláusula que subsume $\{\overline{\phi}\}$. Pero tal δ-cláusula, por la definición 4.16, sólo puede ser $\Box$ o la propia $\{\overline{\phi}\}$. Si $\Box$ perteneciera a N^δ_Θ tendríamos, por el teorema 4.21, que $\Box$ es un δ-resolvente minimal de N_Θ, y por la definición 4.8, $N_\Theta \vdash_\delta \Box$, y por el teorema 4.11, que toda valoración que satisfaga $\Box$ satisface N_Θ. Y como, por la definición 4.1, $\Box$ es universalmente válida, también lo sería $\neg\Theta$, por lo que Θ sería no satisfactible. Pero esto contradice nuestro punto de partida, pues hemos supuesto que Θ es satisfactible. De modo que $\Box$ no puede estar en N^δ_Θ. Por tanto, debe cumplirse que $\{\overline{\phi}\} \in N^\delta_\Theta$.

Igualmente, supongamos que $\{\overline{\phi}\} \in N^\delta_\Theta$. Entonces resulta trivial que cada δ-cláusula de $\{\{\overline{\phi}\}\}$ está subsumida por una δ-cláusula de N^δ_Θ, puesto que por la definición 4.16, cada δ-cláusula Σ está subsumida por ella misma, ya que $\Sigma \subseteq \Sigma$. ■

6.2. Transformaciones en el proceso abductivo

En esta sección explicamos las dos modificaciones que introducimos en el proceso abductivo —definición 5.23— con el objeto de mejorar su eficiencia. La primera modificación simplemente afecta al modo en que se obtendrán las formas

δ-clausales, que ahora será mediante tablas semánticas. Es la segunda modificación la que cambia la estructura del proceso abductivo cuando las observaciones son literales, caso por lo demás bastante común. El proceso abductivo resultante, que se recogerá en la definición 6.6, reduce la búsqueda abductiva —sólo si la observación es un literal— a sólo dos pasos. Más adelante, en la página 174, veremos que aún es posible otra modificación que afecta a la forma en que se guardan las representaciones de las teorías, de forma que reduce este proceso a sólo el segundo de tales pasos si la observación es un literal y a los tres últimos pasos si no lo es.

6.2.1. Obtención de formas δ-clausales mediante tablas semánticas

Esta transformación no afecta a la estructura del proceso abductivo, que permanece como en la definición 5.23, pues simplemente consiste en obtener las formas δ-clausales, en vez de por conversión a forma normal disyuntiva, mediante la creación de una tabla semántica.

La idea para esta modificación la hemos tomado de Arnon Avron [Avr93], quien estudia la dualidad que existe entre las tablas semánticas y el cálculo común de resolución y sugiere cómo puede combinarse el uso de las tablas semánticas con la resolución para obtener una mayor eficiencia. Así, pueden usarse tablas semánticas para transformar las fórmulas a forma clausal, y desde ahí aplicar resolución a las cláusulas obtenidas.

Sin embargo, al usar resolución común, la forma clausal que A. Avron emplea es la estándar, basada en formas normales conjuntivas. Nosotros necesitamos una forma δ-clausal que parte de una forma normal disyuntiva, por ello hemos establecido el corolario 6.3, que nos dice que dada una fórmula α, podemos obtener su forma δ-clausal construyendo su tabla semántica $\mathcal{T}(\alpha)$, tal como vimos en la definición 3.10, y partir de ella, creando el conjunto $\mathcal{C}\delta(\mathcal{T}(\alpha))$, que según la definición 6.1 es simplemente un conjunto de δ-cláusulas tal que cada δ-cláusula suya es el conjunto de los literales de una rama abierta de $\mathcal{T}(\alpha)$.

Por tanto, el proceso abductivo sigue siendo el que presentamos en la definición 5.23, con la única salvedad de que, dadas $\langle \Theta, \phi \rangle$, las formas δ-clausales de $\neg\Theta$ y de ϕ, es decir, N_Θ y O, se obtienen creando, respectivamente, los conjuntos $\mathcal{C}\delta(\mathcal{T}(\neg\Theta))$ y $\mathcal{C}\delta(\mathcal{T}(\phi))$.

6.2.2. Optimización del proceso abductivo para observaciones literales

Al introducir la definición 5.23 comentamos que el proceso abductivo que entonces definíamos experimentaría ciertas modificaciones orientadas a mejorar su eficiencia. Ya hemos visto que la obtención de las formas δ-clausales mediante tablas semánticas puede resultar bastante ventajosa. En la siguiente definición veremos que también, cuando la observación es un literal, puede mejorarse mucho el proceso abductivo, al reducirse a sólo dos pasos de los cuatro originales.

Definición 6.6 (Proceso abductivo para literales) *Dados el par de fórmulas $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$, tal que ϕ es un literal, se define el* proceso abductivo para literales *como el procedimiento que transcurre según los siguientes pasos:*

1. ***Análisis de la teoría**. Se construye N_Θ, forma δ-clausal de $\neg\Theta$, y a partir de ahí N_Θ^δ, mediante saturación por δ-resolución desde N_Θ. Entonces,*

   - *Si $N_\Theta^\delta = \emptyset$, el proceso termina y emite un mensaje que informa de que la teoría Θ es universalmente válida.*
   - *Si $N_\Theta^\delta = \{\square\}$, el proceso termina y emite un mensaje que informa de que la teoría Θ es no satisfactible.*
   - *En otro caso, emite un mensaje que informa de que la teoría Θ es contingente, y continúa con el siguiente paso.*

2. ***Búsqueda de refutaciones y explicaciones**, donde*

   - *Si $\{\phi\} \in N_\Theta^\delta$, entonces el proceso termina y emite un mensaje que informa de que $\Theta \vDash \neg\phi$, es decir, la observación refuta la teoría.*

- *Si* $\{\overline{\phi}\} \in N^{\delta}_{\Theta}$*, entonces el proceso termina y emite un mensaje que informa de que* $\Theta \vDash \phi$*, de decir, la observación se encuentra explicada por la teoría.*

- *En otro caso, se construye* $\{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N^{\delta}_{\Theta}, \ \ \overline{\phi} \in \Sigma\}$*, conjunto que se devuelve acompañado de un mensaje que informa de que se trata del conjunto de soluciones explicativas minimales.*

Corolario 6.7 *El procedimiento descrito en la definición 6.6 es correcto, es decir, cada uno de los mensajes que emite, para cualesquiera* $\langle \Theta, \phi \rangle$*, es verdadero.*

Demostración:

Veamos primero los mensajes que se emiten durante el análisis de la teoría. En primer lugar, si $N^{\delta}_{\Theta} = \emptyset$, por el teorema 5.14, N_{Θ} no es satisfactible, y por el teorema 4.3 tampoco lo es $\neg\Theta$, con lo que ninguna valoración satisface $\neg\Theta$, por tanto toda valoración satisface Θ, de modo que Θ es universalmente válida, con lo que el mensaje que emite la definición 6.6 es correcto.

Además, si $N^{\delta}_{\Theta} = \{\Box\}$, ello significa, por el teorema 5.14, que N_{Θ} es universalmente válida, y por el teorema 4.3 también lo es $\neg\Theta$, de modo que Θ no es satisfactible. Por tanto, el mensaje que emite el procedimiento de la definición 6.6 también es correcto en este caso.

Por último, si N^{δ}_{Θ} no es ninguno de los dos conjuntos anteriores, por el teorema 5.14, N_{Θ} es contingente, y por el teorema 4.3 también lo es $\neg\Theta$, de modo que Θ es igualmente contingente. Con lo que también el último de los mensajes que se pueden emitir durante el análisis de la teoría es verdadero.

En cuanto a los mensajes producidos durante la fase de búsqueda de refutaciones y explicaciones, como la definición 6.6 sólo alcanza esta fase si se da el tercer caso de entre los posibles en el análisis de la teoría, podemos partir, como ya hemos probado, de que cuando se alcanza la segunda fase se verifica que Θ es contingente, y por tanto satisfactible.

Además, ya que tanto ϕ como $\overline{\phi}$ son literales, por el teorema 6.5 se cumple que $\Theta \models \overline{\phi}$ syss $\{\phi\} \in N^{\delta}_{\Theta}$, por lo que resulta verdadero el primero de los mensajes que se producen durante esta fase.

Igualmente, por el teorema 6.5 se cumple que $\Theta \models \phi$ syss $\{\overline{\phi}\} \in N^{\delta}_{\Theta}$, por lo que también es verdad el segundo mensaje.

Respecto al último de los mensajes, sólo se produce, como vemos, si la teoría Θ es contingente y si además no es el caso de que $\Theta \models \phi$ ni $\Theta \models \neg\phi$. Por tanto, $\langle \Theta, \phi \rangle$ es un problema abductivo. Por el teorema 5.2, sus soluciones abductivas explicativas minimales son las que componen el conjunto $\mathcal{A}b\delta(\Theta, \phi)$, pero por el teorema 6.4,

$$\mathcal{A}b\delta(\Theta, \phi) = \{\Sigma - \{\overline{\phi}\} \mid \Sigma \in N^{\delta}_{\theta}, \quad \overline{\phi} \in \Sigma\}$$

y como este es el conjunto que devuelve la definición 6.6, acompañado del mensaje que informa de que se trata del conjunto de soluciones explicativas minimales, también en este caso es correcto. ■

Como se puede apreciar en la definición 6.6, esta modificación en el proceso abductivo simplifica bastante los pasos necesarios en la búsqueda de explicaciones cuando la observación ϕ es un literal. El segundo paso de la definición 5.23 ya no es necesario, pues todos los literales son siempre contingentes. También, para buscar refutaciones basta con comprobar si $\{\phi\}$ pertenece a N^{δ}_{Θ}, con lo que no hay que comprobar si hay en N^{δ}_{Θ} alguna δ-cláusula que subsuma a $\{\phi\}$, y se evitan así ciertas operaciones conjuntistas que, aunque en casos de δ-cláusulas con un solo elemento no son demasiado complejas, sí que tienen un coste mayor que el comprobar simplemente la pertenencia de un elemento a un conjunto, que es lo que ahora se requiere. Lo mismo ocurre cuando se trata de comprobar si $\Theta \models \phi$; en la definición 5.23 requiere una búsqueda por δ-resolución que, si ϕ es un literal, se trata de una búsqueda sencilla, pero desde luego no tan sencilla como buscar $\{\overline{\phi}\}$ en N^{δ}_{Θ}, que es lo único que exige la definición 6.6. Por último, la búsqueda de explicaciones también es ahora más simple al no requerir, como en la definición 5.23, una saturación por δ-resolución que, aunque sería la misma saturación que se hace para comprobar si $\Theta \models \phi$ y no resultaría muy compleja si

ϕ es un literal, tiene un coste mayor que buscar todas δ-cláusulas $\Sigma - \{\overline{\phi}\}$ tales que $\Sigma \in N^{\delta}_{\Theta}$ y $\overline{\phi} \in \Sigma$.

Por tanto, aunque no se espera una enorme disminución de la complejidad, sí que el proceso de la definición 6.6 debe ser notablemente más eficiente que el que aparece en la definición 5.23, cuando se enfrente a problemas abductivos en que la observación sea un literal.

Al igual que hemos abreviado el proceso para observaciones literales, podría hacerse algo parecido si la observación es una disyunción contingente de literales —o si es equivalente a alguna, como es el caso de $a \rightarrow b$, que es equivalente a $\neg a \vee b$—. En tal caso, dado el par de fórmulas $\langle \Theta, \phi \rangle$, tendríamos para la forma δ-clausal de ϕ, $O = \{\{\lambda_1\}, \ldots, \{\lambda_n\}\}$, siendo cada λ_i, $1 \leq i \leq n$, un literal. Si N_{Θ} es la forma δ-clausal de $\neg\Theta$, entonces se puede demostrar

- Que ϕ refuta Θ syss $O \subseteq N^{\delta}_{\Theta}$
- Que ϕ se encuentra explicado por Θ syss existe una δ-cláusula $\Sigma \in N^{\delta}_{\Theta}$ tal que $\Sigma \subseteq \{\overline{\lambda_1}, \ldots, \overline{\lambda_n}\}$
- Que si Θ es contingente, entonces

  $\mathcal{A}b\delta(\Theta, \phi) = \{\Sigma - \{\{\overline{\lambda_1}\}, \ldots, \{\overline{\lambda_n}\}\} \mid \Sigma \in N^{\delta}_{\Theta}, \quad \Sigma \cap \{\{\overline{\lambda_1}\}, \ldots, \{\overline{\lambda_n}\}\} \neq \emptyset\}$

Sin embargo, no vamos a desarrollar una versión del proceso abductivo para observaciones disyuntivas. Ni siquiera demostraremos lo que acabamos de enunciar, aunque por otra parte las pruebas son similares a las que hemos presentado para observaciones literales. Las razones para ello son, por un lado, lo poco habituales que resultan los problemas abductivos con observación disyuntiva, al contrario de lo que ocurre con las observaciones literales, que son mayoritarias. Otra razón es que el proceso abductivo resultante no sería tan ventajoso, en cuanto a reducción de complejidad, como el presentado en la definición 6.6 para observaciones literales, debido a las operaciones conjuntistas que ahora sería preciso hacer. Su complejidad estaría, pues, más cercana a la del proceso de la definición 5.23, y además caeríamos en algo que tratamos de evitar en este trabajo, donde nos hemos propuesto realizar abducción de forma directa, y no a través de procedimientos inversos.

6.3. Implementación

En la página 277 aparece el programa que integra las dos modificaciones que hemos comentado en la sección anterior. El predicado `abduce_mod(+Teo, +Obs)` implementa el procedimiento principal. Dadas la teoría `Teo` y la observación `Obs` realiza un proceso abductivo, de forma que

- Si `Obs` es un literal, implementa el proceso de búsqueda abductiva de la definición 6.6.
- En otro caso, implementa el proceso de la definición 5.23, aunque emplea tablas semánticas para la obtención de la forma δ-clausal, tal como permite el corolario 6.3.

En cuanto a la explicación de literales, el programa comienza con una cláusula del predicado `abduce_mod(+Teo, +Obs)` que realiza lo siguiente:

1. Sólo se usa esta cláusula si la observación, `Obs`, es un literal. En otro caso, se pasa a la siguiente cláusula de `abduce_mod/2`, que es igual que la cláusula de `abduce_proc/2` que se explicó en el capítulo anterior, aunque como veremos emplea tablas semánticas para obtener las formas δ-clausales.
2. Se toman medidas de espacio, inferencias y tiempo, para realizar posteriormente el cómputo de recursos empleados, mediante una llamada al predicado `medidas(Esp, Inf)`.
3. Se emiten dos mensajes, uno que devuelve la teoría y la observación del problema planteado, usando `comienzo(Teo, Obs)`, y otro que avisa de que la observación es un literal.
4. Procede al análisis de la teoría llamando al predicado `analiza_teo(Teo, UsTeo)`, cuya definición no ha cambiado desde que se explicó en el capítulo anterior, aunque ahora usa tablas semánticas, como se ha comentado.
5. Si `UsTeo` es `fin`, lo que como ya vimos significa que la teoría es universalmente válida o no satisfactible, entonces termina el proceso, y se devuelven los recursos consumidos mediante `terminar(Esp, Inf)`.

6. En otro caso, cuando la teoría es contingente, se emite un mensaje que avisa de que se pasa a la fase de búsqueda de refutaciones y explicaciones. A continuación, se obtiene el literal complementario de `Obs`, al que llamamos `NObs`, y se llama al predicado `refuta_y_abduce([Obs], NObs, UsTeo, [], Res)`, cuya definición veremos más adelante, que unifica `Res` con `refuta` si `[Obs]` pertenece a `UsTeo` —pues en este caso la observación refuta la teoría, como se vio en la definición 6.6—; unifica `Res` con `explica` si `[NObs]` pertenece a `UsTeo` —en este caso, la teoría explica la observación, también como se ve en la definición 6.6— y si no es el caso de que ni `[Obs]` ni `[NObs]` pertenezcan a `UsTeo`, entonces unifica `Res` con la lista formada por todas las δ-cláusulas tales que la unión de cada una de ellas y `[NObs]` pertenece a `UsTeo`.

7. Si `Res` es `refuta`, se emite un mensaje que informa de que la teoría ha sido refutada y se termina el proceso, midiendo los recursos consumidos con `terminar(Esp, Inf)`.

8. Si `Res` es `explica`, se emite un mensaje que informa de que la observación ya está explicada por la teoría y termina el proceso, midiendo los recursos consumidos con `terminar(Esp, Inf)`.

9. En otro caso, `Res3` reúne las δ-cláusulas de `Res` que no están subsumidas por alguna δ-cláusula de `UsTeo`. Por motivos de eficiencia, de `UsTeo`, conjunto de refutables de la teoría, no se eliminan ciertas δ-cláusulas subsumidas. Por ello es necesario comprobar, para mantener la corrección, que las δ-cláusulas que se devuelven como soluciones abductivas no sean subsumidas por ningún refutable de la teoría.

10. Si `Res3` es vacía, se emite un mensaje que informa de que no hay abducciones explicativas y termina el proceso, también midiendo los recursos consumidos con `terminar(Esp, Inf)`.

11. En otro caso, se devuelven todas las explicaciones de `Res3`, acompañadas de un mensaje que informa de que tales son las abducciones explicativas. El proceso termina, también, midiendo los recursos consumidos con `terminar(Esp, Inf)`.

A continuación, mostramos el código de esta primera cláusula del predicado `abduce_mod/2`, con una numeración en sus líneas que se corresponde con la de la explicación anterior.

```
abduce_mod(Teo,Obs) :-
  literal(Obs),!,                                          % 1
  medidas(Esp,Inf),                                        % 2
  comienzo(Teo,Obs),                                       % 3
  format('~N~n   *** OBSERVACIÓN LITERAL ***~n',[]),
  analiza_teo(Teo,UsTeo),                                  % 4
   ( UsTeo = fin ->                                        % 5
     terminar(Esp,Inf)
   ; % Teoría contingente                                  % 6
     format('~N~n2. BÚSQUEDA DE REFUTACIONES Y EXPLICACIONES:~n',[]),
     complementario(Obs,NObs),
     refuta_y_abduce([Obs],NObs,UsTeo,[],Res),
     ( Res = 'refuta' ->                                   % 7
       format('~N~n   *** TEORÍA REFUTADA ***~n',[]),
       terminar(Esp,Inf)
     ; % no hay refutación
       ( Res = 'explica' ->                                % 8
         format('~N~n   *** OBSERVACIÓN EXPLICADA ***~n',[]),
         terminar(Esp,Inf)
        ; % tampoco está explicada
         findall(X, (member(X,Res),                        % 9
                     member(_N*_H*Y,UsTeo),
                     subset(Y,X)), Res2),
         subtract(Res,Res2,Res3),
           ( Res3 = [] ->                                  % 10
             format('~N~n *** NO HAY ABDUCCIONES EXPLICATIVAS ***',[]),
             terminar(Esp,Inf)
            ; % hay explicaciones                          % 11
             format('~N~n Las abducciones explicativas son:',[]),
             muestra_conjunto(Res3),
             terminar(Esp,Inf))))).
```

El predicado `refuta_y_abduce(+[Lit], +CompLit, +ListaDCls, +Acum, ?Result)`, que desempeña un papel muy importante en la fase de búsqueda de refutaciones y explicaciones, se verifica si, siendo `+[Lit]` una δ-cláusula unitaria

con el literal `Lit`, `+CompLit` el literal complementario de `Lit`, `+ListaDCls` una lista de δ-cláusulas y `+Acum` un argumento que sirve de acumulador, entonces:

- Si `[Lit]` está en `ListaDCls`, `Result` unifica con `refuta`. En este caso, la observación refuta la teoría, como se vio en la definición 6.6.
- Si `[NObs]` está en `ListaDCls`, `Result` unifica con `explica`. Ahora, la teoría explica la observación, como muestra la definición 6.6.
- En otro caso, `Result` contiene todas las δ-cláusulas de `Acum` más todas las que resultan de quitar `NObs` a las δ-cláusulas de `ListaDCls` que lo contienen. `Result` será la lista de soluciones abductivas explicativas minimales, como muestra la definición 6.6[2].

Las cláusulas que definen este predicado son las siguientes:

```
refuta_y_abduce(_,_,[],Ac,Ac).
refuta_y_abduce(DcObs,NObs,[_N*_H*Dcl|NegTeo],Ac,Res) :-
        ( Dcl = DcObs ->
            Res = 'refuta'
        ; % no refuta Dcl
            ( Dcl = [NObs] ->
                Res = 'explica'
            ; % tampoco explica
                ( select(NObs,Dcl,Exp) ->
                    refuta_y_abduce(DcObs,NObs,NegTeo,[Exp|Ac],Res)
                ; % tampoco vale como explicación Dcl
                    refuta_y_abduce(DcObs,NObs,NegTeo,Ac,Res)))).
```

Las demás cláusulas que aparecen en el fichero mostrado en la página 277 sólo se diferencian respecto de las comentadas en el capítulo anterior en que los predicados `analiza_teo/2` y `analiza_obs/2` usan, para obtener las formas δ-clausales,

[2]En nuestra implementación, aún habrá que suprimir de `Result` las δ-cláusulas subsumidas por algún refutable de la teoría. Como hemos comentado más arriba, la razón para no eliminar durante la saturación del análisis de la teoría algunas de las δ-cláusulas que quedan subsumidas es que ello nos permite hacer el proceso un poco más eficiente, sin pérdida de la corrección siempre que luego se realice la comprobación que indicamos.

el predicado `tabla(+[LFmls, ?Tab])` que, como se explicó en la sección 3.2, dada la lista de fórmulas `LFmls`, devuelve `Tab`, que contiene las ramas abiertas de su tabla semántica. Como sólo se devuelven las ramas abiertas, no es necesario comprobar —como sí hacíamos en el capítulo anterior— si las δ-cláusulas obtenidas son satisfactibles, con lo que se reduce el coste computacional.

Al igual que en los capítulos anteriores, terminamos la explicación del programa con un ejemplo de la salida en pantalla que produce para un sencillo problema abductivo.

```
?- abduce_mod(a => b, b).
PROBLEMA ABDUCTIVO:
   Teoría:       a=>b
   Observación:  b
   *** OBSERVACIÓN LITERAL ***
1. ANÁLISIS DE LA TEORÍA:
   Soporte de la teoría:
     1 [] [-b, a]
   d-cláusula actual #1: 1 [] [-b, a]
   *** TEORÍA CONTINGENTE ***
   Refutables de la teoría:
     1 [] [-b, a]
2. BÚSQUEDA DE REFUTACIONES Y EXPLICACIONES:
 Las abducciones explicativas son:
      [a]
RECURSOS EMPLEADOS:
   Tiempo (milisegundos):  0
   Espacio (bytes):        992
   Inferencias:            289
Yes
```

Los resultados de eficiencia para esta implementación están recogidos en el cuadro 6.1, donde se muestran los recursos que el programa emplea para resolver cada uno de los problemas abductivos del apéndice C.

Comparando estos resultados con los que aparecen en la página 145 observamos que las dos modificaciones introducidas aumentan notablemente la eficiencia del sistema, pues la cantidad de recursos necesarios se encuentra ahora en torno

	Instancia	*Milisegundos*	*Bytes*	*Inferencias*
implic	5	0	15712	2826
	10	71	175660	53713
	15	420	470240	340508
conj	3	0	2240	345
	4	0	3804	590
	5	10	5792	928
	6	8	8252	1383
	7	0	11232	1982
anidado	3	9	8552	1507
	4	10	18112	3955
	5	10	34616	9125
incon	1	10	5308	729
	3	0	15536	2750
	4	10	22868	4544
	5	11	32200	7146

Cuadro 6.1: Resultados para la integración de δ-resolución y tablas semánticas

a la mitad o la tercera parte de la que requería aquel programa. Este ahorro se debe tanto a que la transformación a forma δ-clausal mediante tablas semánticas es más eficiente que el pasar las fórmulas a forma normal disyuntiva, como a la simplificación del proceso abductivo para observaciones literales, pues todos los problemas del apéndice C tienen como observación un literal.

El único de los sistemas anteriormente mostrados que supera en eficiencia a éste para algún tipo de problemas es el de Aliseda, cuyos resultados aparecen recogidos en la página 79. Aunque su eficiencia es menor para las demás clases, el sistema de Aliseda es mucho más eficiente que los demás en la resolución de los problemas **implic**. Como hemos comentado anteriormente, esto se debe a las peculiaridades que tienen las tablas semánticas que el sistema de Aliseda crea en estos problemas.

6.4. Comentarios

Si consideremos el proceso abductivo que en el capítulo anterior, en la definición 5.23, se ha propuesto y que ahora, en la definición 6.6, se ha optimizado para observaciones literales, vemos que supone un avance considerable en una de las direcciones que nos marcábamos como objetivo de este trabajo. Al contar con un *proceso abductivo* —y no sólo con procedimientos de generación de fórmulas que negadas cumplen los requisitos para ser explicaciones *planas*— disponemos también de un argumento fuerte a favor de la posibilidad de una *lógica abductiva*. El proceso definido es tan *lógico* como *abductivo*. Lógico porque está basado en una regla lógica, la de δ-resolución, a través de la cual se define un cálculo con el mismo nombre —definición 4.5—. Abductivo puesto que cada δ-resolvente que se obtiene es una hipótesis explicativa de las fórmulas de que se partió, como ya demostramos con el teorema 4.6, lo cual es sin duda una propiedad netamente abductiva. Por ello pensamos que son posibles las *lógicas abductivas*, una de las cuales es la definida por el cálculo de δ-resolución.

Otro de los logros del cálculo de δ-resolución es la integración que realiza de los procesos de *generación* y *selección* de las hipótesis explicativas. Otros sistemas

abductivos conocidos emplean procedimientos diferentes para generar cierto conjunto de explicaciones posibles y para comprobar cuáles de los elementos de tal conjunto cumplen los requisitos para ser una buena explicación. Así, el método de abducción mediante tablas semánticas emplea tablas semánticas para buscar explicaciones *planas*, y después una cantidad considerable de operaciones conjuntistas destinadas a determinar cuáles de entre las soluciones abductivas planas son consistentes, explicativas y minimales. Esta forma de proceder considera *generación* y *selección* como dos tareas bien diferentes, que se llevan a cabo mediante métodos distintos. Sin embargo, el procedimiento abductivo mediante δ-resolución —definición 5.23— trata ambos procesos de la misma forma, al basarse en el resultado del corolario 5.10,

$$\boxed{\mathcal{A}b\delta(\Theta, \phi) = (N^{\delta}_{\Theta} \cup O^{\delta})^{\delta} - (N^{\delta}_{\Theta} \cup O^{\delta})}$$

que establece que *selección* supone simplemente la eliminación de $(N^{\delta}_{\Theta} \cup O^{\delta})^{\delta}$ de todas las δ-cláusulas que aparezcan en $N^{\delta}_{\Theta} \cup O^{\delta}$. Además, la construcción de ambos conjuntos se realiza de igual forma, mediante saturación por δ-resolución, y no hay por qué construirlos por separado, pues basta con obtener, como ya hemos señalado, primero N^{θ}_{Θ} y O^{δ} y finalmente, a partir de los anteriores, $(N^{\delta}_{\Theta} \cup O^{\delta})^{\delta}$. Esta forma unificada de tratar *generación* y *selección* supone un aumento considerable de eficiencia, como queda claro si se comparan los resultados que aparecen en el cuadro 5.1 con los del cuadro 4.3, en que a pesar de usar δ-resolución no se integraron los procesos de *generación* y *selección*.

También queremos comentar algo respecto de la discusión sobre si la abducción debe considerarse simplemente como un *producto* —conjunto de fórmulas, con ciertas propiedades, que se obtienen para un problema abductivo dado— o más bien como un *proceso* en que cada uno de los pasos que se siguen es, de alguna manera, *abductivo*. Con el cálculo de δ-resolución hemos visto que, desde luego, se obtiene un producto que es *abductivamente correcto*, tal como muestran el corolario 5.5 para el caso de las abducciones planas y el corolario 5.10 para las abducciones explicativas. Pero este producto se alcanza, fundamentalmente, mediante la regla de δ-resolución que, como se ha comentado ya, es netamente abductiva, pues dado un conjunto de δ-cláusulas, cada aplicación de la regla de δ-resolución devuelve nuevas δ-cláusulas de las que el conjunto original es consecuencia lógica. Las otras reglas que intervienen en el proceso de saturación por

δ-resolución —definición 4.18—, que resulta de gran relevancia en el procedimiento abductivo, son las de eliminación de δ-cláusulas subsumidas y no satisfactorias. Ambas reglas pueden considerarse, desde el punto de vista procedimental, como *abductivas*. De hecho, la eliminación de δ-cláusulas subsumidas lo que hace es descartar como explicaciones las δ-cláusulas Σ para las que exista otra δ-cláusula $\Sigma' \subset \Sigma$ tal que Σ' sea una explicación más simple, que introduce menos hipótesis, por lo que esta última será siempre preferible. Igualmente, la eliminación de δ-cláusulas no satisfactibles se puede ver como la prohibición de que ocurran, entre posibles explicaciones, hipótesis contradictorias en sí mismas. Por tanto, podemos decir que el cálculo de δ-resolución unifica ambas concepciones: la *abducción como producto* y la *abducción como proceso*.

No sólo cada una de las reglas del proceso de saturación puede interpretarse en clave procedimentalmente abductiva, sino que podemos hacerlo también con los cuatro pasos que componen el procedimiento abductivo de la definición 5.23. Dado el par de fórmulas proposicionales $\langle \Theta, \phi \rangle$, veamos cada una de las cuatro fases.

Análisis de la teoría. En esta fase —donde puede acabar todo el proceso si se detecta que la teoría es universalmente válida o no satisfactible—, se obtiene el conjunto N^{δ}_{Θ}, por saturación de N_{Θ}, forma δ-clausal de $\neg\Theta$. El conjunto N^{δ}_{Θ}, al que hemos llamado *conjunto de refutables* de Θ, es el conjunto de *abducciones minimales* —según el sentido de la definición 4.9— de $\neg\Theta$. Es decir, se trata del conjunto de δ-cláusulas minimales que podrían refutar la teoría. Representan —de forma minimal— todas las formas de refutar la teoría. Es por tanto un modo muy popperiano de comenzar un proceso abductivo, determinando los puntos débiles de la teoría, aquellos conjuntos de fenómenos —literales— que si se dieran simultáneamente refutarían la teoría. Durante el resto del proceso abductivo no se trabajará más con la teoría sino con su *conjunto de refutables*. Esta es una peculiaridad de la δ-resolución que no se encuentra en otros procedimientos abductivos.

Análisis de la observación. Tras esta fase —donde también puede terminar el proceso si se detecta que la observación es universalmente válida o no satisfactible— se obtiene el conjunto O^{δ}, por saturación de O, forma δ-

clausal de ϕ. El conjunto O^δ, al que hemos llamado *conjunto de mínimos abductivos* de O, es el resultado de un análisis —en sentido cartesiano, si se quiere— donde se han determinado cuáles son los requisitos mínimos necesarios para explicar ϕ. De hecho, O^δ es el conjunto donde están representadas todas las posibles explicaciones de ϕ, pues cada una de sus δ-cláusulas contiene un conjunto de literales tal que si todos son satisfechos entonces ϕ lo será también. Como ϕ puede ser una fórmula con cierta complejidad, durante el resto del proceso se trabajará con O^δ.

Búsqueda de refutaciones. En esta fase se comprueba si cada δ-cláusula de O^δ está subsumida por alguna δ-cláusula de N^δ_Θ. Lo que se hace es determinar si cada *mínimo abductivo* de ϕ es un elemento del *conjunto de refutables* de Θ. Es decir, si cada posibilidad explicativa de la observación es una refutación de la teoría, en cuyo caso ϕ no puede ser explicada en Θ. Es más, ϕ refuta Θ. También es una rasgo característico de la δ-resolución el poder integrar la búsqueda de refutaciones dentro de la búsqueda abductiva, como un paso previo.

Búsqueda de explicaciones. Si la teoría no ha sido refutada, comienza la búsqueda de explicaciones a partir de N_Θ y aquellas δ-cláusulas de O^δ que no hayan sido subsumidas. Ninguna de las δ-cláusulas de partida sirve como abducción explicativa, pues algunas no son consistentes —las que pertenecen a N^δ_Θ— y otras no satisfacen el requisito explicativo —las de O^δ—; sin embargo, mediante sucesivas aplicaciones de la regla de δ-resolución se obtienen otras δ-cláusulas progresivamente mejores, que llegan a ser abducciones explicativas. Podemos ver la regla de δ-resolución como una navaja de Ockham, pues dadas dos explicaciones —posiblemente no muy buenas— $\Sigma_1 \cup \{\lambda\}$ y $\Sigma_2 \cup \{\neg\lambda\}$, genera otra —posiblemente más sencilla y mejor— $\Sigma_1 \cup \Sigma_2$ donde la variable proposicional λ no aparece. Así, poco a poco, se obtienen las abducciones explicativas.

A lo largo de esta sección vamos a ver otros comentarios al proceso abductivo mediante δ-resolución, especialmente referidos a posibles optimizaciones diferentes a las mostradas en este capítulo.

6.4.1. Criterios preferenciales

Una de las posibilidades que ofrece todo proceso abductivo es la de establecer criterios preferenciales que permitan ordenar las explicaciones según sean mejores o peores. Un primer criterio, muy común, es establecer alguna forma de determinar el *peso* de una explicación. Refiriéndonos a la δ-resolución, en la implementación que hemos hecho ordenamos frecuentemente las δ-cláusulas por pesos, siendo el peso de cada δ-cláusula igual al número de literales que contiene. Si se emplea este criterio para hacer un uso preferencial, se pueden elegir, de entre las explicaciones minimales de $\mathcal{A}b\delta(\Theta, \phi)$, aquellas de menor peso como preferibles, que serán las más cortas.

Un criterio similar puede establecerse asignando a cada par $\{\lambda, \neg\lambda\}$ de literales un *coeficiente de aceptabilidad* que será un número racional comprendido entre 0 y 1, y que determinará el grado en que aceptaríamos el que en una explicación cualquiera $\Sigma = \{\gamma_1, \ldots, \gamma_n\}$ apareciera alguno de los literales λ ó $\neg\lambda$. Cuanto más aceptable sea para nosotros una explicación donde aparezca alguno de tales literales, mayor será su *coeficiente de aceptabilidad*. Entonces podemos definir la *aceptabilidad* de la explicación Σ como el producto de los *coeficientes de aceptabilidad* de cada uno de sus literales γ_i, $1 \leq i \leq n$. Por tanto, la *aceptabilidad* favorece las explicaciones más cortas y las que contienen literales más aceptables.

Un ejemplo sobre el uso de este criterio podemos verlo en que si tratamos de explicar por qué llueve y nos parece más aceptable que las explicaciones elegidas contengan literales referidos a fenómenos atmosféricos que no a fenómenos extraterrestres, simplemente tendríamos que asignar un coeficiente de aceptabilidad mayor —seguramente mucho mayor— a los literales que se refieran a fenómenos atmosféricos que a los que se refieran a fenómenos extraterrestres.

Otro criterio preferencial —más natural aún— del cálculo de δ-resolución es ordenar las explicaciones según su *historia*, como ya hemos sugerido. Podemos definir la *historia* de una δ-cláusula Σ como el conjunto de todas las δ-cláusulas[3] que se han empleado hasta llegar a Σ. De esta forma, el criterio preferencial

[3]O sólo las dos últimas δ-cláusulas, como hemos hecho en nuestra implementación. A todos los efectos, ambas definiciones resultan equivalentes.

consistiría en elegir las δ-cláusulas con más historia. Como la historia de una δ-cláusula, en el proceso abductivo, nos dice cuáles de entre las δ-cláusulas de la teoría —de N^{δ}_{Θ}— y de la observación —de O^{δ}— se han usado hasta llegar a cada explicación, las δ-cláusulas con mayor historia serán —por lo general— las que empleen más parte de la teoría, las que requieran más postulados de la teoría para explicar la observación. En muchas ocasiones, las explicaciones con mayor historia serán las *últimas* explicaciones que permite la teoría, entendiendo por tales aquellas explicaciones que no pueden ser a su vez explicadas dentro de la teoría.

En el ejemplo que vimos en el cuadro 4.2, página 86, las historias de las explicaciones l y a contienen a las de z y c, y además son las explicaciones *últimas* que la teoría soporta. Por ello, l y a son las abducciones con mayor poder explicativo, pues dan cuenta no sólo de la observación, z, sino también de la otra abducción posible, c.

No siempre que una explicación Σ_1 tenga mayor historia que otra Σ_2 podremos decir que Σ_1 explica Σ_2 en la teoría; tampoco podemos decir de las explicaciones con mayor historia que serán siempre las últimas, pero sí que es un criterio que habitualmente funciona, razón que basta para tomarlo como criterio preferencial.

6.4.2. Representación compacta de teorías

Algo que podemos plantearnos para hacer más eficiente la búsqueda de soluciones abductivas mediante δ-resolución es la posibilidad de guardar, para cada teoría Θ con la que vaya a trabajarse, su *conjunto de refutables* N^{δ}_{Θ}, es decir, la *saturación por δ-resolución* de la forma δ-clausal de $\neg\Theta$.

Como es posible que se planteen diversos problemas abductivos para una misma teoría Θ, si se guarda N^{δ}_{Θ} no tendrá que obtenerse cada vez que se vaya a resolver un nuevo problema abductivo. Por tanto, la fase de análisis de la teoría no sería necesaria, con lo que se reduce considerablemente el coste computacional del procedimiento de la definición 5.23.

Pero el procedimiento que más se reduce con esta modificación es el de explicación de literales, que aparece en la definición 6.6. Como se ahorra el primer paso, sólo hay que comprobar, para un literal ϕ, si N^{δ}_{Θ} contiene $\{\phi\}$ —refutación— o $\{\overline{\phi}\}$ —la teoría ya explica ϕ—, o bien elegir todas las δ-cláusulas Σ de N^{δ}_{Θ} que contengan $\overline{\phi}$ y construir el conjunto de todas las $\Sigma - \{\overline{\phi}\}$. Entonces, la explicación de literales queda reducida a un procedimiento trivial.

Como en el teorema 5.8 se probó que para todo conjunto de δ-cláusulas A se cumple

$$\vDash A \leftrightarrow A^{\delta}$$

si se guarda N^{δ}_{Θ} lo que supone es guardar un conjunto de δ-cláusulas equivalente a N_{Θ}, y por el teorema 4.3, equivalente a $\neg\Theta$. Por tanto, es una forma muy compacta de guardar una representación de la teoría a través de su conjunto de refutables.

Por ejemplo, para la teoría $\Theta = (a \rightarrow c) \wedge (l \rightarrow c) \wedge (c \rightarrow z)$, que apareció en el ejemplo de la página 85, tenemos que

$$N^{\delta}_{\Theta} = \{\{a, \neg c\}, \{l, \neg c\}, \{c, \neg z\}, \{a, \neg z\}, \{l, \neg z\}\}$$

Aunque intuitivamente Θ nos resulte una representación más expresiva, a partir de N^{δ}_{Θ} es más sencillo construir explicaciones. De hecho, si queremos explicar z —como en el ejemplo citado—, inmediatamente podemos concluir que las soluciones explicativas minimales son c, a y l.

De modo que para cada teoría Θ, guardar N^{δ}_{Θ} puede no ser intuitivamente tan expresivo como guardar Θ, pero permite acortar los procesos abductivos al evitar tener que volver a calcularlo. Una solución intermedia que combina la expresividad intuitiva con la eficiencia es guardar ambas representaciones lo que, si bien es redundante, para teorías no muy grandes podría implementarse sin un gran gasto adicional de espacio.

6.4.3. Notas sobre el uso de resolución dual

Cuando al final de la sección 4.2 expusimos algunas de las razones por las que pensamos que el cálculo de δ-resolución tiene sentido en sí mismo, a pesar de ser completamente dual al cálculo común de resolución —con lo que no habría por qué definir la δ-resolución, pues en cierto sentido no permite obtener ningún resultado al que no pueda llegarse, de forma dual, por resolución— comentamos que la δ-resolución permitiría una integración eficiente con las tablas semánticas. Sin embargo, si hubiéramos usado resolución común la integración no habría sido tan sencilla. En tal caso, para resolver un problema abductivo $\langle \Theta, \phi \rangle$ de forma dual a como establece la definición 5.23, habría que comenzar obteniendo la forma clausal de Θ —dual a la forma δ-clausal de $\neg\Theta$— para saturarla por resolución. Pero dicha forma clausal supone calcular la forma normal conjuntiva de la teoría, como es habitual. Entonces la integración con tablas semánticas sería menos directa que la establecida en el corolario 6.3.

Hay dos formas posibles de obtener la forma normal conjuntiva de Θ mediante tablas semánticas. Una de ellas es construir la tabla semántica de $\neg\Theta$, $\mathcal{T}(\neg\Theta)$, y a partir de ahí obtener $\mathcal{C}\delta(\mathcal{T}(\neg\Theta))$, como indica la definición 6.1, que será el conjunto de δ-cláusulas

$$\{\{\lambda_{1_1}, \lambda_{1_2}, \ldots, \lambda_{1_{j_1}}\}, \ldots, \{\lambda_{i_1}, \lambda_{i_2}, \ldots, \lambda_{i_{j_i}}\}\} \tag{6.4}$$

para $i \geq 0$. Pero (6.4) que, por el teorema 6.2 es equivalente a $\neg\Theta$, también es equivalente, por las definiciones 4.1 y 4.2, a la fórmula proposicional

$$(\lambda_{1_1} \wedge \lambda_{1_2} \wedge \ldots \wedge \lambda_{1_{j_1}}) \vee \ldots \vee (\lambda_{i_1} \wedge \lambda_{i_2} \wedge \ldots \wedge \lambda_{i_{j_i}}) \tag{6.5}$$

De modo que si negamos (6.5) tendremos una fórmula equivalente a Θ, que será

$$\neg((\lambda_{1_1} \wedge \lambda_{1_2} \wedge \ldots \wedge \lambda_{1_{j_1}}) \vee \ldots \vee (\lambda_{i_1} \wedge \lambda_{i_2} \wedge \ldots \wedge \lambda_{i_{j_i}})) \tag{6.6}$$

Además, aplicando varias veces a (6.6) las equivalencias de De Morgan, así como las de doble negación, resulta

$$(\overline{\lambda_{1_1}} \vee \overline{\lambda_{1_2}} \vee \ldots \vee \overline{\lambda_{1_{j_1}}}) \wedge \ldots \wedge (\overline{\lambda_{i_1}} \vee \overline{\lambda_{i_2}} \vee \ldots \vee \overline{\lambda_{i_{j_i}}}) \tag{6.7}$$

Pero (6.7) es una fórmula en forma normal conjuntiva, que por ser equivalente a Θ, es la forma normal conjuntiva de Θ, que era lo que estábamos buscando.

Cada una de las disyunciones elementales de (6.7) —que se convertirá en una cláusula— es el conjunto de literales complementarios a los de una de las δ-cláusulas de (6.4). Por tanto, puede obtenerse la forma normal conjuntiva de Θ construyendo la tabla semántica $\mathcal{T}(\neg\Theta)$ y tomando como disyunciones elementales los conjuntos de literales complementarios a los de cada una de sus ramas abiertas. Este procedimiento es muy similar al que emplea A. Avron [Avr93].

Otra forma es usar tablas duales, que se construyen de forma diferente a como indica la definición 3.10. La diferencia es que ahora, al usar un nodo ϕ,

- Si ϕ es una fórmula de la clase α, se continúa la construcción de cada rama de la tabla que comparta el nodo ϕ añadiéndole el grafo de la figura 6.1.

Figura 6.1: Grafo modificado para fórmulas de tipo α

- Si ϕ es una fórmula de la clase β, se continúa la construcción de cada rama de la tabla que comparta el nodo ϕ añadiéndole el grafo de la figura 6.2.

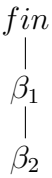

Figura 6.2: Grafo modificado para fórmulas de tipo β

Además, para construir una tabla dual para cierto conjunto Γ de fórmulas proposicionales, se comienza, desde el nodo raíz —que ahora no será una fórmula sino un símbolo elegido arbitrariamente— creando un hijo para cada una de las fórmulas de Γ, es decir, la tabla dual comienza dividiendo el nodo raíz en $|\Gamma|$ ramas. En todo lo demás, la construcción es igual a la definición 3.10.

Con estas tablas duales pueden probarse los siguientes teoremas, para cualquier conjunto de fórmulas proposicionales Γ:

- Γ es universalmente válido syss su tabla dual es cerrada.

- Si Γ no es universalmente válido, cualquier valoración v que satisfaga al menos un literal de cada rama abierta de la tabla dual de Γ satisface Γ.

Las demostraciones son duales a las que se hicieron para el cálculo de tablas semánticas en el capítulo 3. Además, de forma parecida a como probamos el teorema 6.1, puede demostrarse que dada la fórmula Θ, su forma normal disyuntiva puede obtenerse tomando como disyunciones elementales las formadas por los literales de cada una de las ramas abiertas de su tabla dual.

Así pues, hay dos formas de obtener la forma normal conjuntiva de Θ, que sería el primer paso para resolver problemas abductivos por medio del cálculo de resolución normal. Puede construirse la tabla semántica de $\neg\Theta$ y tomarse cada disyunción elemental como los literales complementarios a los de cada una de sus ramas abiertas, o bien construir la tabla dual de Θ y tomar cada disyunción elemental como los literales de cada una de sus ramas abiertas. En ambos casos, el procedimiento es más complejo que el que establece el corolario 6.3. En el primero, porque requiere construir el literal complementario de cada uno de los pertenecientes a ramas abiertas de la tabla semántica de $\neg\Theta$, lo que supone cierto coste computacional. En el segundo, porque requiere definir un nuevo cálculo de tablas duales, con reglas diferentes al cálculo habitual de tablas semánticas. Esto es un inconveniente de cara a integrar la búsqueda de soluciones abductivas con sistemas existentes de demostración automática, pues los cálculos para los que existen implementaciones eficientes suelen ser los habituales, y no sus formas duales.

Sin embargo, al usar δ-resolución podemos integrar mejor —y con mayor eficiencia— la búsqueda de explicaciones con implementaciones existentes, porque la construcción de las formas δ-clausales puede hacerse mediante tablas semánticas comunes y sin necesidad de negar los literales obtenidos. A continuación, como el proceso de saturación por δ-resolución es semánticamente dual al de resolución pero sintácticamente idéntico —pues las reglas de δ-resolución y de eliminación de δ-cláusulas subsumidas y contradictorias son sintácticamente idénticas a las

correspondientes del cálculo de resolución—, para la búsqueda abductiva pueden usarse igualmente implementaciones del cálculo común de resolución.

Otra razón para haber definido el cálculo de δ-resolución es que, mientras que si usáramos la resolución habitual para la búsqueda de explicaciones habría que negar las cláusulas obtenidas para obtener las abducciones, mediante la δ-resolución las δ-cláusulas obtenidas son ya explicaciones abductivas.

Además, muchos de los teoremas que se han demostrado a lo largo de los capítulos 4, 5 y 6 tienen un claro sentido abductivo debido precisamente a que se basan en el uso de la δ-resolución. Si se demostraran sus formas duales —nada habituales, puesto que el cálculo de resolución se suele emplear para buscar refutaciones y a este efecto tales teoremas no tienen sentido—, no se apreciarían tan claramente sus posibilidades abductivas.

6.4.4. Comparación de complejidad con el sistema de Aliseda

Aunque la complejidad de todo sistema de demostración en lógica proposicional es siempre exponencial para el caso peor, en este apartado vamos a comparar la complejidad que tiene, para el sistema de Aliseda y para el cálculo de δ-resolución, el paso principal en la resolución de problemas abductivos: la construcción de la tabla semántica de la teoría —o de su negación—, para cierta clase de teorías.

Nos centraremos en una extensión de las cláusulas de Horn. Concretamente, en teorías representables mediante una conjunción de fórmulas del tipo

$$\lambda_1 \wedge \ldots \wedge \lambda_{n-1} \rightarrow \lambda_n \tag{6.8}$$

para $n \geq 2$, siendo cada λ_i, $1 \leq i \leq n$, un literal —positivo o negativo, ésta es la diferencia con las cláusulas de Horn, donde los literales deben ser positivos—. Las teorías con esta forma son muy habituales en representación del conocimiento y constituyen la base de numerosos sistemas usados en Inteligencia Artificial.

El sistema de Aliseda comienza construyendo la tabla semántica de la teoría. Imaginemos una teoría con m fórmulas del tipo de (6.8), es decir,

$$(\lambda_{1_1} \wedge \ldots \wedge \lambda_{1_{j_1-1}} \rightarrow \lambda_{1_{j_1}}) \wedge \ldots \wedge (\lambda_{m_1} \wedge \ldots \wedge \lambda_{m_{j_m-1}} \rightarrow \lambda_{m_{j_m}}) \tag{6.9}$$

Entonces, como (6.9) es una conjunción, su tabla semántica comienza usando el nodo raíz —que es la propia teoría— y añadiendo sucesivamente, mediante aplicaciones de la regla α, cada una de las m fórmulas, $1 \leq i \leq m$, de tipo

$$\lambda_{i_1} \wedge \ldots \wedge \lambda_{i_{j_i-1}} \rightarrow \lambda_{i_{j_i}} \tag{6.10}$$

con lo que en cierto momento la tabla semántica —si se construye tal como estamos indicando— consta de una sola rama con $m+1$ nodos, el primero con (6.9) y después uno por cada una de las m fórmulas de tipo (6.10) que se obtienen desde (6.9) por aplicación de la regla α. Pero como las fórmulas de tipo (6.10) son de la clase β —definición 3.1—, a cada uno de los últimos m nodos hay que aplicar la regla β, con lo que cada rama se divide en dos, añadiendo a una de ellas el nodo

$$\neg(\lambda_{i_1} \wedge \ldots \wedge \lambda_{i_{j_i-1}}) \tag{6.11}$$

y a otra

$$\lambda_{i_{j_i}} \tag{6.12}$$

A su vez, como (6.11) es de la clase β, también hay que aplicar, sucesivamente, la regla β, produciendo como resultado que la rama en que ocurre (6.11) se divida en $j_i - 1$ nuevas ramas, cada una con un nodo de la forma $\overline{\lambda_{i_k}}$, $1 \leq k \leq j_i - 1$. Por tanto, cuando se han aplicado todas las reglas a cada uno de los m nodos de tipo (6.10), así como a las subfórmulas que se obtienen, el resultado es que cada una de las ramas en que tal nodo ocurre se divide en j_i ramas. Pero como inicialmente los m nodos de tipo (6.10) ocurren en la única rama que tiene la tabla, esto hace que la tabla se divida en $j_1 \times j_2 \times \ldots \times j_m$ ramas. Para un caso ideal, en que $j_i = l$, para todo i, $1 \leq i \leq m$, el número de ramas abiertas resultante —en el caso peor, suponiendo que ninguna rama se cierra antes— es l^m. Además, cada rama tiene un literal por cada una de las m fórmulas de tipo (6.10).

En cuanto a la δ-resolución, se construye la tabla semántica de la negación de la teoría. Partiendo de la misma teoría (6.9), si la negamos se transforma en una fórmula de la clase β —definición 3.1—, con lo que, al construir la tabla semántica,

el nodo superior —que es la negación de (6.9)— tiene como hijos —tras varias aplicaciones de la regla β— m nodos, $1 \leq i \leq m$, de tipo

$$\neg(\lambda_{i_1} \wedge \ldots \wedge \lambda_{i_{j_i-1}} \rightarrow \lambda_{i_{j_i}}) \tag{6.13}$$

Pero cada uno de los nodos tipo (6.13) es de la clase α, con lo que al usarlos con la regla α producen un nodo

$$\lambda_{i_1} \wedge \ldots \wedge \lambda_{i_{j_i-1}} \tag{6.14}$$

y otro

$$\overline{\lambda_{i_{j_i}}} \tag{6.15}$$

Pero el nodo (6.14) es también de la clase α, por lo que al aplicarle sucesivamente la regla α produce $j_i - 1$ nuevos nodos —en la misma rama— de tipo λ_{i_k}, $1 \leq k \leq j_i - 1$. Entonces, tras completar cada rama en que aparezca un nodo de tipo (6.13), se añaden a tal rama j_i nuevos nodos. De modo que tras acabar la tabla, las únicas divisiones en ramas que se han producido son las resultantes de usar el nodo raíz, negación de (6.9). Por tanto, la tabla final contiene sólo m ramas. Además, si como en el caso anterior hacemos que $j_i = l$ para cada i entre 1 y m, cada una de las m ramas contiene l literales.

Queda claro que en el primer paso de la resolución de un problema abductivo, el sistema de Aliseda y el nuestro muestran una gran diferencia de complejidad para teorías de la forma (6.9). Mientras que la tabla de la teoría —que es la que hace Aliseda— contiene l^m ramas, con m literales cada una, la tabla de la negación de la teoría —que es la que nosotros hacemos— sólo tiene m ramas, con l literales cada una. Si consideramos el tamaño de una tabla —espacio que ocupa su representación— como el número total de literales que hay entre todas sus ramas, tenemos que el tamaño de la tabla que hace al sistema de Aliseda es l^{m+1}, mientras que el tamaño de la tabla que hace nuestro sistema es $m \times l$. Por ello, la complejidad espacial de la primera tabla es exponencial y la de la segunda lineal.

Además, para mayor complejidad, el sistema de Aliseda requiere, para construir las explicaciones conjuntivas, tomar un literal de entre los conjuntos de cierres parciales de cada una de las ramas abiertas de la tabla de la teoría extendida con la negación de la explicación. Por tanto, como puede llegar a haber

hasta l^m ramas abiertas —siempre habrá alguna menos, pues la extensión debe ser semicerrada— con m literales cada una, en el caso peor pueden llegar a formarse m^{l^m} posibles explicaciones conjuntivas —de las cuales pueden ser diferentes hasta $2^{l \times m}$, pues hay $l \times m$ literales diferentes—, y es necesario comprobar para cada una de ellas, a través de costosas operaciones conjuntistas, si son soluciones abductivas explicativas o no, así como si son o no minimales.

Sin embargo, en el caso de la δ-resolución, para explicar literales —que es el caso que estamos considerando para que resulte posible hacer la comparación con el sistema de Aliseda— lo único que hay que realizar es saturar por δ-resolución un conjunto de m δ-cláusulas, saturación que difícilmente va a ser tan compleja como el procedimiento arriba comentado. Tras esta saturación sólo resta el segundo paso de la definición 6.6, que es bastante simple.

Las mismas conclusiones que hemos obtenido tras esta comparación informal de la complejidad de ambos sistemas se pueden corroborar a la vista de los datos que aparecen en los cuadros 3.1 y 6.1, en las páginas 79 y 168, respectivamente. Todas las teorías que constituyen los problemas abductivos del apéndice C son del tipo de (6.9). Por supuesto que siempre es posible encontrar casos de problemas en que por tener pocas ramas abiertas la tabla semántica de la teoría extendida con la negación de la observación, el sistema de Aliseda rinde mejor que la δ-resolución. Esto sólo ocurre en el caso de los problemas **implic**, como ya se ha comentado repetidas veces, pues dicha tabla tiene sólo una rama abierta y además sólo hay explicaciones atómicas, y son consistentes todas las inicialmente posibles. Sin embargo, en problemas de mayor complejidad, en que la tabla extendida tiene varias ramas abiertas, o bien en que hay posibles explicaciones no minimales, o incluso contradictorias con la teoría, el coste computacional se dispara ante instancias relativamente bajas. El uso de la δ-resolución, sin embargo, resulta muy apropiado para estos casos.

6.4.5. Comentarios sobre la no monotonía

En la sección 2.3 realizamos un análisis estructural del razonamiento abductivo, especialmente centrado en su carácter no monótono. A continuación volvemos

sobre la no monotonía de la *relación de consecuencia abductiva*, pero ahora nos vamos a centrar en el proceso de δ-resolución, de forma que descubramos en él rasgos no monótonos.

Consideremos el problema abductivo $\langle \Theta_1, \phi \rangle$, tal que $\mathcal{A}b\delta(\Theta_1, \phi) \neq \emptyset$. Sea Θ_2 una teoría que extiende Θ_1, es decir, $\Theta_1 \subset \Theta_2$. Supongamos que $\Theta_2 \nvDash \bot$ y $\Theta_2 \nvDash \phi$. Como en otros casos, trataremos las teorías como fórmulas de $\mathcal{L}_p$; por tanto, podemos considerar que $\Theta_2 = \Theta_1 \wedge \gamma$, siendo γ la conjunción de todas las fórmulas que Θ_2 añade a Θ_1.

Analizaremos cómo cambia el conjunto de soluciones abductivas explicativas al aumentar la teoría. Si N_{Θ_i} es la forma δ-clausal de $\neg\Theta_i$ y O la de ϕ, por el corolario 5.10 tenemos:

$$\mathcal{A}b\delta(\Theta_1, \phi) = (N^{\delta}_{\Theta_1} \cup O^{\delta})^{\delta} - (N^{\delta}_{\Theta_1} \cup O^{\delta})$$

y

$$\mathcal{A}b\delta(\Theta_2, \phi) = (N^{\delta}_{\Theta_2} \cup O^{\delta})^{\delta} - (N^{\delta}_{\Theta_2} \cup O^{\delta})$$

Pero tenemos que $\neg\Theta_2 = \neg(\Theta_1 \wedge \gamma)$, lo cual equivale a $\neg\Theta_1 \vee \neg\gamma$. Entonces, si C es la forma δ-clausal de $\neg\gamma$ podemos considerar que $N_{\Theta_2} = N_{\Theta_1} \cup C$, dadas las reglas de obtención de las formas δ-clausales y las equivalencias proposicionales que para ello se emplean. Entonces,

$$\mathcal{A}b\delta(\Theta_2, \phi) = ((N_{\Theta_1} \cup C)^{\delta} \cup O^{\delta})^{\delta} - ((N_{\Theta_1} \cup C)^{\delta} \cup O^{\delta})$$

Para estudiar cómo afecta al proceso abductivo mediante δ-resolución el carácter no monótono de la abducción, consideremos una solución abductiva Σ al problema abductivo $\langle \Theta_1, \phi \rangle$ y veamos si Σ aparece entre las soluciones al problema abductivo $\langle \Theta_2, \phi \rangle$. Como hemos considerado $\Theta_2 \nvDash \bot$ y $\Theta_2 \nvDash \phi$, despejamos los casos más triviales de no monotonía y aseguramos que $\langle \Theta_2, \phi \rangle$ sigue siendo un problema abductivo.

Entonces, sea $\Sigma \in \mathcal{A}b\delta(\Theta_1, \phi)$, lo cual por definición significa $\Sigma \in (N^{\delta}_{\Theta_1} \cup O^{\delta})^{\delta}$ y $\Sigma \notin (N^{\delta}_{\Theta_1} \cup O^{\delta})$. Pero como $\Sigma \in (N^{\delta}_{\Theta_1} \cup O^{\delta})^{\delta}$, debe existir una δ-cláusula $\Sigma' \subseteq \Sigma$ tal que $\Sigma' \in ((N_{\Theta_1} \cup C)^{\delta} \cup O^{\delta})^{\delta}$, ya que este último conjunto es

igual a $(N^{\delta}_{\Theta_1} \cup C^{\delta} \cup O^{\delta})^{\delta}$ —aplicando resultados obtenidos anteriormente— y $N^{\delta}_{\Theta_1} \cup C^{\delta} \cup O^{\delta} \vdash_{\delta} \Sigma$ —por los supuestos que hemos hecho—, de forma que si $\Sigma \notin (N^{\delta}_{\Theta_1} \cup C^{\delta} \cup O^{\delta})^{\delta}$ debe ser porque ha sido subsumida por Σ', ya que Σ es satisfactible.

De momento tenemos que para cada $\Sigma \in \mathcal{A}b\delta(\Theta_1, \phi)$ existe una δ-cláusula $\Sigma' \subseteq \Sigma$ tal que $\Sigma' \in (N^{\delta}_{\Theta_2} \cup O^{\delta})^{\delta}$. Ahora bien, puede que Σ' pertenezca o no a $\mathcal{A}b\delta(\Theta_2, \phi)$. Ello depende de cuál de las dos posibilidades siguientes se verifique:

- $\Sigma' \notin (N^{\delta}_{\Theta_2} \cup O^{\delta})$. Entonces $\Sigma' \in \mathcal{A}b\delta(\Theta_2, \phi)$, y:
  - Si $\Sigma' = \Sigma$, la explicación $\Sigma \in \mathcal{A}b\delta(\Theta_1, \phi)$ ha permanecido tras la extensión de la teoría.
  - Si $\Sigma' \subset \Sigma$, la explicación $\Sigma \in \mathcal{A}b\delta(\Theta_1, \phi)$ se ha refinado tras la extensión de la teoría. Esto significa que, por tener Θ_2 más información que Θ_1, las explicaciones necesitan añadir menos literales a la teoría para dar cuenta de las nuevas observaciones.
- $\Sigma' \in (N^{\delta}_{\Theta_2} \cup O^{\delta})$. En este caso, $\Sigma' \notin \mathcal{A}b\delta(\Theta_2, \phi)$. Analicemos cuándo ocurre esto. Como $N_{\Theta_2} = N_{\Theta_1} \cup C$, tenemos que $\Sigma' \in ((N_{\Theta_1} \cup C)^{\delta} \cup O^{\delta})$. Entonces, puesto que $\Sigma \notin (N^{\delta}_{\Theta_1} \cup O^{\delta})$, se cumple $\Sigma \notin O^{\delta}$ y debe ocurrir $\Sigma' \notin O^{\delta}$, ya que en otro caso tendría que ser $\Sigma' \subset \Sigma$ y no podría ser $\Sigma \in (N^{\delta}_{\Theta_1} \cup O^{\delta})^{\delta}$ —lo cual es uno de nuestros puntos de partida—, pues Σ sería subsumida por Σ'. Por tanto, $\Sigma' \in (N_{\Theta_1} \cup C)^{\delta}$, con lo que Σ' pertenece al conjunto de refutables de Θ_2. Es decir, como la teoría se hace más informativa también es más refutable, y la solución abductiva Σ, que era válida para el problema abductivo $\langle \Theta_1, \phi \rangle$, se ha perdido tras extender la teoría Θ_1 con γ, pues ahora refuta la teoría, al ser subsumida por Σ', uno de los refutables de Θ_2.

Recapitulando, para cada $\Sigma \in \mathcal{A}b\delta(\Theta_1, \phi)$ tenemos que al extender la teoría Θ_1 con γ y obtener Θ_2 se da uno de entre los siguientes casos:

- $\Sigma \in \mathcal{A}b\delta(\Theta_2, \phi)$, lo que significa que la explicación sigue siendo válida.
- Existe una δ-cláusula $\Sigma' \subset \Sigma$ tal que $\Sigma' \in \mathcal{A}b\delta(\Theta_2, \phi)$, y en este caso ocurre que la explicación se ha refinado.

- No existe ninguna δ-cláusula $\Sigma' \subseteq \Sigma$ tal que $\Sigma' \in \mathcal{A}b\delta(\Theta_2, \phi)$, lo que se debe a que la explicación es inconsistente con la nueva teoría.

Por supuesto, también puede ocurrir que en $\mathcal{A}b\delta(\Theta_2, \phi)$ aparezcan nuevas soluciones abductivas para las que no haya en $\mathcal{A}b\delta(\Theta_1, \phi)$ ninguna δ-cláusula similar. Vamos a ver cada uno de los casos posibles en el siguiente ejemplo.

Ejemplo 6.8 Sean $\Theta_1 = (a \rightarrow t) \wedge (b \rightarrow s) \wedge (f \rightarrow m) \wedge ((t \vee s) \wedge r \rightarrow m)$, $\phi = m$, $\gamma = \neg s \wedge (c \rightarrow a) \wedge r$ y $\Theta_2 = \Theta_1 \wedge \gamma$. Entonces:

$$\begin{aligned} \mathcal{A}b\delta(\Theta_1, \phi) &= \{\{r,t\},\{r,s\},\{r,a\},\{r,b\},\{f\}\} \\ \mathcal{A}b\delta(\Theta_2, \phi) &= \{\{t\},\{a\},\{c\},\{f\}\} \end{aligned}$$

Así pues, de las explicaciones de $\mathcal{A}b\delta(\Theta_1, \phi)$ tenemos que:

- **Desaparecen** $\{r, s\}$ y $\{r, b\}$ por ser inconsistentes con la nueva teoría Θ_2.
- **Se refinan** $\{r, t\}$ y $\{r, a\}$, puesto que en $\mathcal{A}b\delta(\Theta_2, \phi)$ aparecen $\{t\}$ y $\{a\}$.
- **Permanece** $\{f\}$, a la que no le afecta la extensión de la teoría.
- **Aparece** la nueva explicación $\{c\}$, que no era posible con la teoría Θ_1 y sí lo es con Θ_2.

6.5. Formas más complejas de explicación

Definición 6.9 (Conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$) *Dado un par de fórmulas proposicionales $\langle \Theta, \phi \rangle$, definimos el conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$ como el formado por todos los conjuntos de δ-cláusulas tal que para cada $A \in \mathcal{A}b\delta^*(\Theta, \phi)$, si $A = \{\Sigma_1, \ldots, \Sigma_n\}$, $n \geq 1$, se cumple:*

1. $\Theta, A \models \phi$
2. $\Theta, A \not\models \bot$

3. $A \nvDash \phi$

4. *Para cada* Σ_i, $1 \leq i \leq n$, *se cumple*

   a) Σ_i *es satisfactible.*

   b) *No existe ninguna* $\Sigma' \subset \Sigma_i$ *tal que* $\Theta, \Sigma' \vDash \phi$

En la definición 6.9, los requisitos 1–3 son los que hacen que cada $A \in \mathcal{A}b\delta^*(\Theta, \phi)$ sea una abducción explicativa para el problema abductivo $\langle \Theta, \phi \rangle$. El requisito $4a$ hace que dada cualquier $A \in \mathcal{A}b\delta^*(\Theta, \phi)$, no pueda ocurrir que $A \cup \{\Sigma\} \in \mathcal{A}b\delta^*(\Theta, \phi)$ para cualquier Σ contradictoria. Por último, el requisito $4b$ establece la minimalidad de cada δ-cláusula de A, ya que por 1 tenemos que para cada $\Sigma_i \in A$ se cumple $\Theta, \Sigma_i \vDash \phi$, pero si para cierto $\Sigma_j \in A$ hubiera una δ-cláusula $\Sigma' \subset \Sigma_j$ tal que $\Theta, \Sigma' \vDash \phi$, tendríamos que $(A \cup \{\Sigma'\}) - \{\Sigma_j\}$ sería una explicación más simple. Por tanto, la definición 6.9 se corresponde con el conjunto de formas δ-clausales que son, para el problema abductivo $\langle \Theta, \phi \rangle$, soluciones abductivas explicativas y, en cierto sentido, minimales. Decimos en cierto sentido pues existen otros criterios de minimalidad que en la definición 6.9 no hemos contemplado. Por ejemplo, es posible que ocurra que haya dos conjuntos de δ-cláusulas B y C en $\mathcal{A}b\delta^*(\Theta, \phi)$ tales que $B \subset C$. En tal caso, B sería minimal respecto de C en el sentido de contener menos δ-cláusulas, y a su vez C es minimal respecto de B desde un punto de vista semántico, ya que $B \vDash C$. El criterio de minimalidad que establece el requisito $4b$ simplemente exige que para cada $A \in \mathcal{A}b\delta^*(\Theta, \phi)$, aunque posiblemente $A - \{\Sigma\}$ pueda pertenecer a $\mathcal{A}b\delta^*(\Theta, \phi)$ para ciertos Θ, ϕ, A y Σ, nunca pueda ser el caso de que $(A - \{\Sigma\}) \cup \{\Sigma'\} \in \mathcal{A}b\delta^*(\Theta, \phi)$ para ninguna $\Sigma' \subset \Sigma$.

Como para cada fórmula puede obtenerse su forma δ-clausal —teorema 4.3—, el conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$ representa, para cada problema abductivo $\langle \Theta, \phi \rangle$, el conjunto de todas sus soluciones abductivas explicativas minimales —en el sentido comentado— que no son necesariamente conjunciones de literales. En el siguiente teorema vemos la forma en que podemos obtener el conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$, para cualquier $\langle \Theta, \phi \rangle$, mediante δ-resolución.

Teorema 6.10 *Para todo par de fórmulas* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$ *si* N_Θ *es la forma* δ*-clausal de* $\neg\Theta$ *y* O *la de* ϕ*, se cumple*

$$\mathcal{A}b\delta^*(\Theta, \phi) = \{A \mid A \subseteq (N_\Theta^\delta \cup O^\delta)^\delta, \quad A - N_\Theta^\delta \neq \emptyset, \quad A - O^\delta \neq \emptyset\}$$

Demostración:

En primer lugar, probaremos que todo conjunto de δ-cláusulas que pertenezca a $\mathcal{A}b\delta^*(\Theta, \phi)$ pertenece también al otro conjunto. Sea $B \in \mathcal{A}b\delta^*(\Theta, \phi)$. Entonces, si $B = \{\Sigma_1, \ldots, \Sigma_n\}$, sabemos, por la definición 6.9, que se cumple

$$\Theta, B \models \phi \tag{6.16}$$

$$\Theta, B \not\models \bot \tag{6.17}$$

$$B \not\models \phi \tag{6.18}$$

y además, cada Σ_i, $1 \leq i \leq n$ cumple

$$\Sigma_i \text{ es satisfactible} \tag{6.19}$$

$$\text{No existe } \Sigma' \subset \Sigma_i \text{ tal que } \Theta, \Sigma' \models \phi \tag{6.20}$$

Por otra parte, debemos probar los siguientes objetivos:

$$B \subseteq (N_\Theta^\delta \cup O^\delta)^\delta \tag{6.21}$$

$$B - N_\Theta^\delta \neq \emptyset \tag{6.22}$$

$$B - O^\delta \neq \emptyset \tag{6.23}$$

En cuanto al objetivo (6.21), supongamos que no es el caso de que ocurra. Entonces, debe haber alguna δ-cláusula Σ_j, $1 \leq j \leq n$ tal que $\Sigma_j \in B$ y $\Sigma_j \notin (N_\Theta^\delta \cup O^\delta)^\delta$. Pero por el corolario 5.9 tenemos que $(N_\Theta^\delta \cup O^\delta)^\delta = (N_\Theta \cup O)^\delta$, por lo que $\Sigma_j \notin (N_\Theta \cup O)^\delta$. Además, sea C la forma δ-clausal de $\Theta \rightarrow \phi$. Por el corolario 5.4 tenemos que $(N_\Theta \cup O)^\delta = C^\delta$, de donde $\Sigma_j \notin C^\delta$, de modo que, por el teorema 4.21, Σ_j no es un δ-resolvente minimal de C y, por el corolario 4.12, Σ_j no es una abducción minimal de $\Theta \rightarrow \phi$, en el sentido de la definición 4.9. Por tanto, hay algunas de las condiciones siguientes que no se cumplen:

- Σ_j es satisfactible. Pero, por (6.19), esto tiene que cumplirse necesariamente.

- $\Sigma_j \vDash \Theta \rightarrow \phi$. Esto también debe cumplirse necesariamente. Por (6.16) tenemos que $\Theta, B \vDash \phi$, que es equivalente a $B \vDash \Theta \rightarrow \phi$. Además, por la definición 4.2 toda valoración que satisfaga Σ_j satisface B, pero como toda valoración que satisface B satisface $\Theta \rightarrow \phi$ —tal como acabamos de establecer— tenemos que toda valoración que satisface Σ_j satisface $\Theta \rightarrow \phi$. Es decir, $\Sigma_j \vDash \Theta \rightarrow \phi$.

- No existe ninguna $\Sigma' \subset \Sigma_j$ tal que $\Sigma' \vDash \Theta \rightarrow \phi$. Esto último también debe verificarse, pues por (6.20) sabemos que para toda $\Sigma' \subset \Sigma_j$ se cumple $\Theta, \Sigma' \nvDash \phi$, pero esto equivale a $\Sigma' \nvDash \Theta \rightarrow \phi$.

De modo que hemos llegado a una contradicción, pues hemos establecido que Σ_j no es una abducción minimal de $\Theta \rightarrow \phi$, en el sentido de la definición 4.9, pero a la vez Σ_j cumple todos los requisitos para serlo. Por tanto, negamos nuestra última hipótesis, concluyendo que se cumple el objetivo (6.21).

Pasemos al objetivo (6.22). También procedemos por reducción al absurdo, suponiendo que $B - N^{\delta}_{\Theta} = \emptyset$. Entonces, debe darse que $B \subseteq N^{\delta}_{\Theta}$, con lo que cada $\Sigma_i \in N^{\delta}_{\Theta}$, $1 \leq i \leq n$. Ahora bien, sea v una valoración que satisface B. Entonces, por la definición 4.2, v satisface alguna Σ_j, $1 \leq j \leq n$. Pero como $\Sigma_j \in N^{\delta}_{\Theta}$, tenemos que, también por la definición 4.2, v satisface N^{δ}_{Θ}, con lo que concluimos que $B \vDash N^{\delta}_{\Theta}$. Pero por el teorema 5.8 tenemos que N^{δ}_{Θ} es equivalente a N_{Θ}, y esto último, por el teorema 4.3, a $\neg\Theta$. Por tanto, $B \vDash \neg\Theta$, de donde $\Theta, B \vDash \bot$. Pero esto contradice (6.17), por lo que negamos nuestra última hipótesis, concluyendo (6.22).

Por último, veamos lo que ocurre con (6.23). Supongamos ahora que $B - O^{\delta} = \emptyset$. Entonces, debe darse que $B \subseteq O^{\delta}$, con lo que cada $\Sigma_i \in O^{\delta}$, $1 \leq i \leq n$. Ahora bien, sea v una valoración que satisface B. Entonces, por la definición 4.2, v satisface alguna Σ_j, $1 \leq j \leq n$. Pero como $\Sigma_j \in O^{\delta}$, tenemos que, también por la definición 4.2, v satisface O^{δ}, con lo que concluimos que $B \vDash O^{\delta}$. Pero por el teorema 5.8 tenemos que O^{δ} es equivalente a O, y esto último, por el teorema 4.3, a ϕ. Por tanto, $B \vDash \phi$. Pero esto contradice (6.18), por lo que negamos nuestra última hipótesis, concluyendo (6.23).

A continuación, debemos probar que para todo conjunto de δ-cláusulas

$$B \in \{A \mid A \subseteq (N_\Theta^\delta \cup O^\delta)^\delta, \quad A - N_\Theta^\delta \neq \emptyset, \quad A - O^\delta \neq \emptyset\}$$

se cumple que $B \in \mathcal{A}b\delta^*(\Theta, \phi)$. En este caso, partimos de que se verifican (6.21), (6.22) y (6.23), y debemos probar que se cumplen (6.16), (6.17), (6.18), (6.19) y (6.20).

En primer lugar, partiendo de (6.21) tenemos que cada Σ_i, $1 \leq i \leq n$, cumple que $\Sigma_i \in (N_\Theta^\delta \cup O^\delta)^\delta$. Pero por el corolario 5.9 tenemos que $(N_\Theta^\delta \cup O^\delta)^\delta = (N_\Theta \cup O)^\delta$, por lo que $\Sigma_i \in (N_\Theta \cup O)^\delta$. Además, sea C la forma δ-clausal de $\Theta \rightarrow \phi$. Por el corolario 5.4 tenemos que $(N_\Theta \cup O)^\delta = C^\delta$, de donde $\Sigma_i \in C^\delta$, de modo que, por el teorema 4.21, Σ_i es un δ-resolvente minimal de C y, por el corolario 4.12, Σ_i es una abducción minimal de $\Theta \rightarrow \phi$, en el sentido de la definición 4.9. Esto significa que cada Σ_i, $1 \leq i \leq n$, cumple

- Σ_i es satisfactible. Con esto ya se ha probado (6.19).
- $\Sigma_i \vDash \Theta \rightarrow \phi$, para cada i entre 1 y n. Por tanto, sea cualquier valoración v que satisface B. Entonces, por la definición 4.2, v satisface alguna δ-cláusula Σ_j, $1 \leq j \leq n$. Pero como se cumple que $\Sigma_j \vDash \Theta \rightarrow \phi$, tenemos que v satisface $\Theta \rightarrow \phi$, con lo que ocurre $B \vDash \Theta \rightarrow \phi$, por lo que $\Theta, B \vDash \phi$, de modo que también hemos probado el objetivo (6.16).
- No existe ninguna $\Sigma' \subset \Sigma_i$ tal que $\Sigma' \vDash \Theta \rightarrow \phi$. Pero esto equivale a que no existe —para cada i entre 1 y n— ninguna $\Sigma' \subset \Sigma_i$ tal que $\Theta, \Sigma' \vDash \phi$. Pero este es el objetivo (6.20), que también resulta probado.

Ahora sólo nos quedan por probar los objetivos (6.17) y (6.18). Comencemos por el primero de ellos.

Por (6.22) sabemos que hay una δ-cláusula Σ_j, $1 \leq j \leq n$, tal que

$$\Sigma_j \notin N_\Theta^\delta \tag{6.24}$$

pero por (6.21) sabemos que

$$\Sigma_j \in (N_\Theta^\delta \cup O^\delta)^\delta \tag{6.25}$$

Por (6.25), usando los corolarios 5.9, 5.4 y 4.12, así como el teorema 4.21, tenemos que Σ_j es una abducción minimal de $\Theta \rightarrow \phi$, en el sentido de la definición 4.9. Ello implica que se verifican,

$$\Sigma_j \models \Theta \rightarrow \phi \tag{6.26}$$

$$\Sigma_j \text{ es satisfactible} \tag{6.27}$$

$$\text{No existe } \Sigma' \text{ tal que } \Sigma' \models \Theta \rightarrow \phi \tag{6.28}$$

A su vez, por (6.24), mediante un razonamiento semejante, tenemos que Σ_j no es una abducción minimal de $\neg\Theta$. Por tanto, debe ocurrir —por la definición 4.9— al menos una cosa de entre las siguientes:

- Σ_j no es satisfactible. Pero esto no puede ocurrir, pues contradice (6.27).
- Existe una δ-cláusula $\Sigma' \subset \Sigma_j$ tal que $\Sigma' \models \neg\Theta$. Si esto fuera así, tendríamos que no hay ninguna valoración que satisfaga a la vez Σ' y Θ, por lo que se daría $\Theta, \Sigma' \models \phi$, lo cual va en contra de (6.28).
- $\Sigma_j \not\models \neg\Theta$. Como no puede ocurrir ninguna de las dos opciones anteriores, debe verificarse esta última.

Como hay al menos una δ-cláusula $\Sigma_j \in B$ tal que $\Sigma_j \not\models \neg\Theta$, hay al menos una valoración que satisface Σ_j pero no satisface $\neg\Theta$. Pero como toda valoración que satisfaga Σ_j también satisface B, por la definición 4.2, tenemos que hay al menos una valoración que satisface B pero no satisface $\neg\Theta$. Por tanto, $B \not\models \neg\Theta$, lo cual equivale a $\Theta, B \not\models \bot$, que es el objetivo (6.17).

Sólo nos queda por probar (6.18). La demostración es muy similar a la que acabamos de hacer, por lo que omitimos la justificación de algunos pasos. Por (6.23) sabemos que hay una δ-cláusula Σ_j, $1 \leq j \leq n$, tal que

$$\Sigma_j \notin O^\delta \tag{6.29}$$

pero por (6.21) tenemos que se cumple (6.25).

Por (6.25), tenemos que Σ_j es una abducción minimal de $\Theta \rightarrow \phi$, en el sentido de la definición 4.9. Ello implica que se verifican las mismas condiciones (6.26), (6.27) y (6.28) anteriores.

Igualmente, por (6.29), mediante un razonamiento similar, tenemos que Σ_j no es una abducción minimal de ϕ. Por tanto, debe ocurrir —por la definición 4.9— al menos una cosa de entre las siguientes:

- Σ_j no es satisfactible. Pero esto no puede ocurrir, pues contradice (6.27).
- Existe una δ-cláusula $\Sigma' \subset \Sigma_j$ tal que $\Sigma' \models \phi$. Supongamos que esto ocurre, y que Σ'' es una δ-cláusula subconjunto de Σ_j tal que $\Sigma'' \models \phi$ y no existe ninguna $\Sigma^* \subset \Sigma''$ tal que $\Sigma^* \models \phi$. Entonces, como también Σ'' es satisfactible —al ser subconjunto de una δ-cláusula satisfactible— cumple los requisitos para ser una abducción minimal de ϕ, en el sentido de la definición 4.9, y por el corolario 4.12, Σ'' será igualmente un δ-resolvente minimal de O. Igualmente, por el teorema 4.21 tenemos que $\Sigma'' \in O^\delta$. Pero entonces, como $\Sigma'' \subset \Sigma_j$ es imposible que $\Sigma_j \in (N_\Theta \cup O^\delta)^\delta$, pues Σ_j está subsumida por Σ'', de modo que si apareciera durante la saturación de $N_\Theta \cup O^\delta$ sería eliminada, bien por estar subsumida por Σ'' o por alguna otra δ-cláusula que haya subsumido a Σ'', según indica la definición 4.18. Pero esto contradice (6.25), por lo que tenemos que negar nuestra última hipótesis, concluyendo que no puede ser el caso de que exista ninguna $\Sigma' \subset \Sigma_j$ tal que $\Sigma' \models \phi$.
- $\Sigma_j \not\models \phi$. Como no puede ocurrir ninguna de las dos opciones anteriores, debe verificarse esta última.

Dado que hay al menos una δ-cláusula $\Sigma_j \in B$ tal que $\Sigma_j \not\models \phi$, hay al menos una valoración que satisface Σ_j y no satisface ϕ. Pero como toda valoración que satisfaga Σ_j también satisface B, por la definición 4.2, tenemos que hay al menos una valoración que satisface B pero no satisface ϕ. Por tanto, $B \not\models \phi$, que es el objetivo (6.18). ■

Ejemplo 6.11 Sea el problema abductivo $\langle \Theta, \phi \rangle$ tal que $\Theta = a$ y $\phi = b$, siendo a y b dos variables proposicionales. Entonces, si N_Θ es la forma δ-clausal de $\neg a$ y O la de b, tenemos,

$$\begin{aligned} N_\Theta^\delta &= \{\{\neg a\}\} \\ O &= \{\{b\}\} \\ (N_\Theta^\delta \cup O^\delta)^\delta &= \{\{\neg a\}, \{b\}\} \end{aligned}$$

De forma que aplicando el resultado del teorema 6.10, tenemos que

$$\mathcal{A}b\delta^*(\Theta, \phi) = \{\{\{\neg a\}, \{b\}\}\}$$

La única solución, $\{\{\neg a\}, \{b\}\}$ es la forma δ-clausal de la fórmula $\neg a \vee b$, que es equivalente a $a \rightarrow b$.

Este sencillo ejemplo muestra que es posible emplear la δ-resolución para encontrar soluciones abductivas en forma de reglas, y no sólo de hechos. Si comparamos nuestro conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$ con el conjunto de explicaciones disyuntivas consistentes de Aliseda [Ali97] —ya que no explica con tanto detalle cómo construir las abducciones explicativas; de todos modos, como todas las abducciones explicativas son también consistentes, este conjunto englobará, pensamos, las explicativas—, vemos que éste se obtiene de la siguiente manera:

1. Se construyen todas las disyunciones que combinan de todas las formas posibles las abducciones atómicas y conjuntivas consistentes previamente obtenidas. Estas disyunciones serán todas consistentes.

2. Se comprueba cuáles de entre las anteriores disyunciones cierran la tabla semántica de $\Theta \cup \{\neg\phi\}$.

Aunque este procedimiento es correcto, no es completo, es decir, deja fuera muchas explicaciones disyuntivas que sin embargo sí pertenecen a $\mathcal{A}b\delta^*(\Theta, \phi)$. Así, en un ejemplo tan sencillo como el anterior, vemos que la abducción $\neg a \vee b$ no puede ser obtenida mediante tablas semánticas, ya que $\neg a$ no es una abducción consistente con la teoría a.

Veamos un ejemplo más complejo:

Ejemplo 6.12 Sea el problema abductivo $\langle \Theta, \phi \rangle$ tal que $\Theta = t \wedge (t \vee s \rightarrow b) \wedge (a \rightarrow m)$ y $\phi = m$, donde cada letra minúscula latina es una variable proposicional.

Entonces, si N_Θ es la forma δ-clausal de Θ y O la de ϕ, tenemos:

$$\begin{aligned} N_\Theta &= \{\{\neg t\}, \{t, \neg b\}, \{s, \neg b\}, \{a, \neg m\}\} \\ O &= \{\{m\}\} \\ N_\Theta^\delta &= \{\{\neg t\}, \{a, \neg m\}, \{\neg b\}\} \\ O^\delta &= \{\{m\}\} \\ (N_\Theta^\delta \cup O^\delta)^\delta &= \{\{\neg t\}, \{\neg b\}, \{m\}, \{a\}\} \\ \mathcal{A}b\delta(\Theta, \phi) &= \{\{a\}\} \end{aligned}$$

Como vemos, $\mathcal{A}b\delta(\Theta, \phi)$ tiene sólo un elemento. Sin embargo, es posible construir explicaciones más complejas. Aplicando el teorema 6.10, las siguientes formas δ-clausales son todas las que pertenecen a $\mathcal{A}b\delta^*(\Theta, \phi)$. A la derecha de cada una escribimos una fórmula equivalente —no necesariamente la única— de $\mathcal{L}_p$:

$$\begin{aligned} \{\{a\}\}, &\quad \text{equivalente a:} \quad a \\ \{\{m\}, \{a\}\}, &\quad \text{equivalente a:} \quad \neg a \rightarrow m \\ \{\{\neg b\}, \{a\}\}, &\quad \text{equivalente a:} \quad b \rightarrow a \\ \{\{\neg b\}, \{m\}\}, &\quad \text{equivalente a:} \quad b \rightarrow m \\ \{\{\neg b\}, \{m\}, \{a\}\}, &\quad \text{equivalente a:} \quad b \rightarrow a \vee m \\ \{\{\neg t\}, \{a\}\}, &\quad \text{equivalente a:} \quad t \rightarrow a \\ \{\{\neg t\}, \{m\}\}, &\quad \text{equivalente a:} \quad t \rightarrow m \\ \{\{\neg t\}, \{m\}, \{a\}\}, &\quad \text{equivalente a:} \quad t \rightarrow a \vee m \\ \{\{\neg t\}, \{\neg b\}, \{a\}\}, &\quad \text{equivalente a:} \quad t \wedge b \rightarrow a \\ \{\{\neg t\}, \{\neg b\}, \{m\}\}, &\quad \text{equivalente a:} \quad t \wedge b \rightarrow m \\ \{\{\neg t\}, \{\neg b\}, \{m\}, \{a\}\}, &\quad \text{equivalente a:} \quad t \wedge b \rightarrow m \vee a \end{aligned}$$

La primera de las formas δ-clausales anteriores se corresponde con la única solución que encontramos en $\mathcal{A}b\delta(\Theta, \phi)$. El resto, equivale a las abducciones disyuntivas que son explicativas y minimales, según el criterio elegido.

El siguiente corolario nos indica la relación que existe, dado cualquier problema abductivo $\langle \Theta, \phi \rangle$, entre las soluciones que se encuentran con $\mathcal{A}b\delta(\Theta, \phi)$ y las que se obtienen con $\mathcal{A}b\delta^*(\Theta, \phi)$.

Corolario 6.13 *Sea cualquier par de fórmulas* $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$. *Entonces, para cada* δ*-cláusula* $\Sigma \in \mathcal{A}b\delta(\Theta, \phi)$, *se cumple que* $\{\Sigma\} \in \mathcal{A}b\delta^*(\Theta, \phi)$.

Demostración:

Sea el par de fórmulas $\langle \Theta, \phi \rangle \in \mathcal{L}_p^2$ tal que N_Θ es la forma δ-clausal de $\neg\Theta$ y O la de ϕ, y Σ una δ-cláusula que pertenece a $\mathcal{A}b\delta(\Theta, \phi)$. Entonces, por el corolario 5.10 tenemos

$$\Sigma \in (N_\Theta^\delta \cup O^\delta)^\delta - (N_\Theta^\delta \cup O^\delta)$$

de forma que ocurre

$$\Sigma \in (N_\Theta^\delta \cup O^\delta)^\delta \tag{6.30}$$

$$\Sigma \notin N_\Theta^\delta \tag{6.31}$$

$$\Sigma \notin O^\delta \tag{6.32}$$

Por tanto, también ocurre:

1. $\{\Sigma\} \subset (N_\Theta^\delta \cup O^\delta)^\delta$, dado (6.30).
2. $\{\Sigma\} - N_\Theta^\delta \neq \emptyset$, dado (6.31).
3. $\{\Sigma\} - O^\delta \neq \emptyset$, dado (6.32).

Pero las tres relaciones anteriores aseguran que $\{\Sigma\} \in \mathcal{A}b\delta^*(\Theta, \phi)$. ■

De forma que el conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$ engloba todas las soluciones explicativas minimales para el problema abductivo $\langle \Theta, \phi \rangle$, tanto las que están representadas por conjuntos de una sola δ-cláusula, que son atómicas o conjuntivas, como las formas más complejas.

Querríamos hacer una observación sobre un criterio de minimalidad —del que más arriba hemos comentado algo— diferente del que establece la definición 6.9. Se trata del criterio de minimalidad semántica, por el que una explicación α es minimal syss no existe ninguna otra explicación β tal que $\alpha \vDash \beta$. Cuando las

explicaciones son simplemente δ-cláusulas, este criterio se corresponde con el que hemos usado, pues dadas dos δ-cláusulas Σ_1 y Σ_2, tenemos que $\Sigma_1 \vDash \Sigma_2$ syss $\Sigma_2 \subseteq \Sigma_1$. Pero cuando las explicaciones son formas δ-clausales este criterio de minimalidad semántica no favorece los conjuntos de δ-cláusulas más simples, sino precisamente los más grandes, puesto que $A \vDash A \cup \{\Sigma\}$ para cualquier conjunto de δ-cláusulas A y cualquier δ-cláusula Σ. Las autoras Marta Cialdea y Fiora Pirri [CP93], así como Aliseda [Ali97] llaman la atención sobre el hecho de que este criterio semántico de minimalidad convierte las explicaciones triviales en minimales. Veamos que esto también ocurre usando δ-resolución, pues como acabamos de comentar, este criterio hace minimales las formas δ-cláusulas con más elementos, de manera que del conjunto $\mathcal{A}b\delta^*(\Theta, \phi)$, según este criterio —y de acuerdo con el teorema 6.10—, sería minimal la explicación representada por el conjunto de δ-cláusulas $(N^\delta_\Theta \cup O^\delta)^\delta$, siempre y cuando tal conjunto contuviese al menos una δ-cláusula que no pertenezca a N^δ_Θ y otra que no pertenezca a O^δ, lo que por el teorema 6.10 ocurre siempre que $\mathcal{A}b\delta^*(\Theta, \phi) \neq \emptyset$. Pero el conjunto de δ-cláusulas $(N^\delta_\Theta \cup O^\delta)^\delta$, por el corolario 5.9, es igual a $(N_\Theta \cup O)^\delta$. Además, sea C la forma δ-clausal de $\Theta \rightarrow \phi$. Por el corolario 5.4 tenemos que $(N_\Theta \cup O)^\delta = C^\delta$. Por tanto, la abducción minimal en este sentido semántico es igual a C^δ, pero por el teorema 5.8 ello significa que se trata de un conjunto de δ-cláusulas equivalente a C, lo que implica, por el teorema 4.3, que la abducción minimal en sentido semántico sería equivalente a $\Theta \rightarrow \phi$, con lo que es completamente trivial.

Por esta razón no hemos elegido el criterio de minimalidad semántica para la definición 6.9. Tampoco hemos optado por el opuesto, que favorece los conjuntos más pequeños, pues al buscar formas complejas de explicación se busca más bien lo contrario, pues si no en muchos casos no habría ninguna diferencia sustantiva entre $\mathcal{A}b\delta(\Theta, \phi)$ y $\mathcal{A}b\delta^*(\Theta, \phi)$. Pensamos que el criterio de minimalidad que aparece en la definición 6.9 resulta mucho más adecuado.

Terminamos esta sección cuantificando las abducciones complejas que forman parte de $\mathcal{A}b\delta^*(\Theta, \phi)$. Volviendo al teorema 6.10 tenemos $\mathcal{A}b\delta^*(\Theta, \phi) = \{A \mid A \subseteq (N^\delta_\Theta \cup O^\delta)^\delta, \quad A - N^\delta_\Theta \neq \emptyset, \quad A - O^\delta \neq \emptyset\}$ de modo que $|\mathcal{A}b\delta^*(\Theta, \phi)|$ será igual a la cantidad de subconjuntos no vacíos de $(N^\delta_\Theta \cup O^\delta)^\delta$ con al menos un elemento que no pertenezca a N^δ_Θ y otro que no sea de O^δ.

La cantidad de subconjuntos no vacíos de $(N_\Theta^\delta \cup O^\delta)^\delta$ es

$$2^{|(N_\Theta^\delta \cup O^\delta)^\delta|} - 1 \tag{6.33}$$

que se corresponde con la cardinalidad del *conjunto partes*[4] de $(N_\Theta^\delta \cup O^\delta)^\delta$ menos una unidad, correspondiente al conjunto vacío. A la cantidad que aparece en (6.33) debemos restar la correspondiente a los subconjuntos no vacíos de $(N_\Theta^\delta \cup O^\delta)^\delta$ cuyos elementos sean todos de N_Θ^δ. Esta cardinalidad es

$$2^{|N_\Theta^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 1 \tag{6.34}$$

que es la correspondiente a todos los subconjuntos no vacíos de $N_\Theta^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta$, es decir, los subconjuntos no vacíos de $(N_\Theta^\delta \cup O^\delta)^\delta$, cuyos elementos son todos de N_Θ^δ. Por la misma razón, la cantidad de subconjuntos no vacíos de $(N_\Theta^\delta \cup O^\delta)^\delta$ cuyos elementos son todos de O^δ es

$$2^{|O^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 1 \tag{6.35}$$

Si restamos a la cantidad que aparece en (6.33) los subconjuntos cuantificados en (6.34) y (6.35) estamos restando dos veces todos los subconjuntos no vacíos de $N_\Theta^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta$ cuyos elementos son todos tanto de N_Θ^δ como de O^δ. La cantidad de tales subconjuntos es, de modo análogo a las anteriores,

$$2^{|N_\Theta^\delta \cap O^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 1 \tag{6.36}$$

Por tanto, obtenemos $|\mathcal{A}b\delta^*(\Theta, \phi)|$ restando de (6.33) las cantidades (6.34) y (6.35) y sumando (6.36). Es decir,

$$|\mathcal{A}b\delta^*(\Theta, \phi)| = \left(2^{|(N_\Theta^\delta \cup O^\delta)^\delta|} - 1\right) - \left(2^{|N_\Theta^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 1\right) - \left(2^{|O^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 1\right) + \left(2^{|N_\Theta^\delta \cap O^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 1\right)$$

o lo que es lo mismo,

$$|\mathcal{A}b\delta^*(\Theta, \phi)| = 2^{|(N_\Theta^\delta \cup O^\delta)^\delta|} + 2^{|N_\Theta^\delta \cap O^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 2^{|N_\Theta^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|} - 2^{|O^\delta \cap (N_\Theta^\delta \cup O^\delta)^\delta|}$$

[4] El *conjunto partes* de cierto conjunto C es el conjunto formado por todos los subconjuntos de C. Si la cardinalidad —número de elementos— de C es $|C|$, entonces la cardinalidad del conjunto partes de C es $2^{|C|}$

Como resulta fácilmente apreciable, la cardinalidad de $\mathcal{A}b\delta^*(\Theta, \phi)$ depende de un polinomio que es exponencial en la cardinalidad de $(N^{\delta}_{\Theta} \cup O^{\delta})^{\delta}$. Esto es así porque el número de abducciones complejas que son solución explicativa a cierto problema abductivo $\langle \Theta, \phi \rangle$ es siempre mucho mayor que el número de soluciones conjuntivas. Por ello, la complejidad de calcular $\mathcal{A}b\delta^*(\Theta, \phi)$ es exponencialmente mayor que la de calcular $\mathcal{A}b\delta(\Theta, \phi)$. Por esta razón, en las implementaciones nos hemos centrado en las soluciones abductivas conjuntivas.

Capítulo 7

Extensión a primer orden

En este capítulo presentamos una extensión del cálculo de δ-resolución a la lógica de primer orden. En la primera sección introducimos el lenguaje de predicados que posteriormente usaremos. A continuación, realizamos la presentación formal del cálculo de δ-resolución en primer orden, así como de sus teoremas fundamentales. Por último, en la tercera sección mostramos los detalles más importantes de la implementación que hemos realizado, incluida en el apéndice D.

En muchos aspectos, la extensión a primer orden del cálculo de δ-resolución es completamente natural a partir de la presentación, en los anteriores capítulos, de la δ-resolución proposicional. Esto ocurre con una buena parte de los teoremas que se presentan. Por ello, el desarrollo no será tan exhaustivo como en los capítulos dedicados al estudio proposicional, de modo que en ocasiones, cuando aparezcan teoremas que sean o bien suficientemente conocidos o extensiones naturales de otros ya probados, las demostraciones que proporcionaremos serán simples esquemas de prueba. Así podremos centrar la atención en los aspectos más importantes del cálculo de δ-resolución de primer orden.

7.1. Elementos de lógica de primer orden

En esta sección presentamos el lenguaje de primer orden que emplearemos en el resto del capítulo, así como su semántica. Seguiremos, en parte, a Nepomuceno [Nep03]. Comenzamos definiendo el lenguaje $\mathcal{L}$.

Definición 7.1 *Se define $\mathcal{L}$, lenguaje formal para la lógica de predicados de primer orden como:*

$$\mathcal{L} = \langle VOC(\mathcal{L}), FOR(\mathcal{L}) \rangle$$

El vocabulario *de $\mathcal{L}$, $VOC(\mathcal{L})$, consta de:*

1. *Un conjunto, posiblemente infinito, de* variables individuales*: x_1, x_2, ..., y_1, y_2, ...*

2. *Un conjunto, posiblemente vacío, de* constantes individuales *o* parámetros*: a_1, a_2, ..., b_1, b_2, ...*

3. *Para cada $n \geq 1$, un conjunto, posiblemente infinito —también puede ser vacío—, de* signos predicativos *o* predicados *n-ádicos —o de aridad n— P^1_1, P^1_2, ..., P^2_1, P^2_2, ..., en los que el índice superior indica la aridad. Así P^i_j es el signo predicativo j-ésimo de aridad $i \geq 1$. Cuando ello no pueda producir ambigüedades, usaremos letras mayúsculas para referirnos a los signos predicativos, sin indicar la aridad.*

4. *Para cada $n \geq 1$, un conjunto, que puede ser vacío o infinito, de* signos de función *o* functores *de aridad n, a los que nos referiremos con f^n, g^n, h^n, etc., indicando en el índice superior la aridad, y usando un índice inferior cuando sea necesario. Cuando no puedan producirse ambigüedades, se omitirá el índice superior, de forma que la aridad se determine por el número de argumentos.*

5. *El conjunto de las* constantes lógicas*, que comprende los operadores lógicos proposicionales y además* $\exists$ —cuantificador existencial— *y* $\forall$ —cuantificador universal—.

6. *Un conjunto finito de signos auxiliares: paréntesis, corchetes, etc.*

El conjunto de términos de $\mathcal{L}$, $TER(\mathcal{L})$, es el más pequeño que verifica lo siguiente:

1. *Todas las variables y todas las constantes individuales de $VOC(\mathcal{L})$ son elementos de $TER(\mathcal{L})$.*
2. *Si $t_1, t_2, \ldots, t_n \in TER(\mathcal{L})$ y f es un functor de aridad $n \geq 1$, entonces $f(t_1, t_2, \ldots, t_n) \in TER(\mathcal{L})$.*

Por último, el conjunto de fórmulas de $\mathcal{L}$, $FOR(\mathcal{L})$, es el más pequeño que verifica:

1. *Si $t_1, t_2, \ldots, t_n \in TER(\mathcal{L})$ y R es un predicado de aridad $n \geq 1$, entonces $Rt_1, t_2, \ldots, t_n \in FOR(\mathcal{L})$.*
2. *Si $\alpha, \beta \in FOR(L)$, entonces*

$$\neg(\alpha), (\alpha) \vee (\beta), (\alpha) \wedge (\beta), (\alpha) \rightarrow (\beta), (\alpha) \leftrightarrow (\beta) \in FOR(\mathcal{L})$$

3. *Si $\alpha \in FOR(\mathcal{L})$ y x es una variable individual, entonces*

$$\exists x(\alpha), \forall x(\alpha) \in FOR(\mathcal{L}).$$

En cuanto a las convenciones que adoptamos en el capítulo 2 para $\mathcal{L}_p$, las podemos extender a $\mathcal{L}$ de forma natural. Por ejemplo, dados $t_1, \ldots, t_n \in TER(\mathcal{L})$ y un predicado R de aridad n, llamamos a las fórmula $Rt_1, \ldots, t_n$ y $\neg Rt_1, \ldots, t_n$ *literales*. Además, del primero de ellos decimos que es *positivo* y, del segundo, *negativo*. Igualmente, por ser uno de ellos la negación del otro, decimos que son *complementarios*. Dado el literal —positivo o negativo— γ, representamos con $\overline{\gamma}$ el literal —negativo o positivo, respectivamente— complementario a γ.

Decimos que una variable ocurre *libre* en una fórmula cuando no cae bajo el alcance de ningún cuantificador que la tenga como sufijo; en otro caso, ocurre *ligada*. Una fórmula sin variables libres es llamada *sentencia*. Si β es una fórmula, mediante $\beta(x/t)$ representamos la fórmula igual a β en todo excepto en que cada ocurrencia de x se ha sustituido por el término t —con las restricciones

habituales—. Como es habitual, con $\sigma = \{x_1/t_1, x_2/t_2, \ldots, x_n/t_n\}$, $n \geq 1$, representamos una sustitución múltiple de cada una de las variables x_i, $1 \leq i \leq n$, por el término t_i correspondiente, con la restricción de que x_i no ocurra en t_i[1]. Mediante $\beta\sigma$, ó $\beta(x_1, x_2, \ldots, x_n/t_1, t_2, \ldots, t_n)$, nos referimos al resultado de aplicar a la fórmula β la sustitución σ.

También, con $\forall x_1 x_2 ... x_n \beta$ y $\exists x_1 x_2 ... x_n \beta$ abreviaremos $\forall x_1(\forall x_2(\ldots \forall x_n \beta)\ldots)$ y $\exists x_1(\exists x_2(\ldots \exists x_n \beta)\ldots)$, respectivamente.

Definición 7.2 *Definimos una* $\mathcal{L}$-estructura *como* $M = \langle \mathcal{D}, \Im \rangle$, *siendo* $\mathcal{D}$ *un conjunto no vacío llamado* dominio *o* universo de discurso *e* $\Im$ *la* función interpretación, *que debe cumplir los siguientes requisitos*[2]*:*

1. *Si* a *es una constante individual,* $\Im(a) \in \mathcal{D}$.
2. *Si* f *es un functor de aridad* $n \geq 1$, $\Im(f)$ *es una función de* $\mathcal{D}^n$ *en* $\mathcal{D}$.
3. *Si* P *es un predicado de aridad* $n \geq 1$, $\Im(P) \in \mathcal{P}(\mathcal{D}^n)$.

Definición 7.3 *Dada una* $\mathcal{L}$*-estructura* $M = \langle \mathcal{D}, \Im \rangle$, *definimos recursivamente la noción de satisfacción:*

1. *Dados el predicado* P *de aridad* n *y los términos sin variables —puede haber repeticiones—* $t_1, \ldots, t_n$; $M \vDash Pt_1, \ldots, t_n$ *syss* $\langle \Im(t_1), \ldots, \Im(t_n) \rangle \in \Im(P)$
2. $M \vDash \neg\beta$ *syss* $M \nvDash \beta$
3. $M \vDash \alpha \vee \beta$ *syss* $M \vDash \alpha$ *ó* $M \vDash \beta$
4. $M \vDash \alpha \wedge \beta$ *syss* $M \vDash \alpha$ *y* $M \vDash \beta$
5. $M \vDash \alpha \rightarrow \beta$ *syss* $M \vDash \neg\alpha$ *ó* $M \vDash \beta$
6. $M \vDash \alpha \leftrightarrow \beta$ *syss* $(M \vDash \alpha$ *y* $M \vDash \beta)$ *o bien* $(M \vDash \neg\alpha$ *y* $M \vDash \neg\beta)$

[1] Esta restricción sirve para que las sustituciones no produzcan términos infinitos, como ocurriría, por ejemplo, si se permitiera sustituir x por $f(x)$.

[2] Mediante $\mathcal{D}^n$ representamos el conjunto de n-tuplas de elementos de $\mathcal{D}$. Con $\mathcal{P}(\mathcal{D}^n)$ representamos el conjunto *partes* de dicho conjunto.

7. $M \vDash \exists x\beta$ *syss* $M' \vDash \beta(x/t)$ *para alguna* $\mathcal{L}$*-estructura* $M' = \langle \mathcal{D}, \Im' \rangle$ *que coincide con* M *excepto, a lo sumo, en cuanto a la interpretación del término* t*, que no debe aparecer en* β *ni contener variables. Es decir,* $\Im'(t)$ *puede tener un valor diferente a* $\Im(t)$*, pero en todo lo demás ambas funciones son idénticas. En adelante, representamos mediante* $M' =_{t_1,\ldots,t_n} M$ *que las* $\mathcal{L}$*-estructuras* M' *y* M *difieren a lo sumo en la interpretación de los términos —siempre nuevos y sin variables—* $t_1, \ldots, t_n$*.*

8. $M \vDash \forall x\beta$ *syss* $M' \vDash \beta(x/t)$ *para toda* $\mathcal{L}$*-estructura* M' *tal que* $M' =_t M$*.*

Definición 7.4 *Dada una* $\mathcal{L}$*-estructura* M*, cada fórmula denota un valor de verdad. Si* $\{0, 1\}$ *es el conjunto de valores de verdad —*0 *para lo falso y* 1 *para lo verdadero— entonces, dada* $\alpha \in \mathcal{L}$*:*

1. *Si* $M \vDash \alpha$*, entonces* α *es* verdadera *en* M*, lo que representamos como* $M(\alpha) = 1$*.*

2. *Si* $M \nvDash \alpha$*, entonces* α *es* falsa *en* M*, lo que representamos como* $M(\alpha) = 0$*.*

La semántica que hemos definido sólo asigna valores de verdad a las *sentencias*, que son las fórmulas sin variables libres. Con esta semántica se pueden obtener ciertos resultados interesantes, como veremos a continuación. Pero antes, introducimos varias nociones, algunas de las cuales son extensiones de las que se presentaron para lógica proposicional. En ciertos teoremas se omiten las demostraciones, por tratarse de resultados habituales.

Definición 7.5 *Decimos que una sentencia* ϕ *es* satisfactible *syss existe alguna* $\mathcal{L}$*-estructura* M *tal que* $M \vDash \phi$*. Si no existe ninguna* $\mathcal{L}$*-estructura que satisfaga* ϕ*, decimos que* ϕ *es* no satisfactible*. Si dado* $\Gamma \subset \mathcal{L}$*, para cada sentencia* $\alpha \in \mathcal{L}$ *se cumple* $M \vDash \alpha$*, escribiremos* $M \vDash \Gamma$*, y diremos que* M satisface simultáneamente Γ*. Si para toda* $\mathcal{L}$*-estructura* M *definida en un dominio* $\mathcal{D}$ *se verifica* $M \vDash \phi$*, decimos que* ϕ *es válida en* $\mathcal{D}$*. Si para toda* $\mathcal{L}$*-estructura* M *se cumple* $M \vDash \phi$*, decimos que* ϕ *es universalmente válida, y escribimos* $\vDash \phi$*.*

Teorema 7.6 *Para cada sentencia* $\phi \in \mathcal{L}$*,* $\vDash \phi$ *syss* ϕ *es válida en todo dominio* $\mathcal{D}$ *no vacío.*

Teorema 7.7 *Para cada sentencia $\phi \in \mathcal{L}$, $\vDash \phi$ syss $\neg\phi$ es no satisfactible.*

Teorema 7.8 *Dados el conjunto de sentencias Γ y la sentencia ϕ, se cumple $\Gamma \vDash \phi$ syss $\Gamma \cup \{\neg\phi\}$ es no satisfactible.*

Teorema 7.9 *Dado el conjunto $\{\gamma_1, \ldots, \gamma_n\}$ de sentencias de $\mathcal{L}$ y la sentencia $\alpha \in \mathcal{L}$, se cumple $\{\gamma_1, \ldots, \gamma_n\} \vDash \alpha$ syss $\vDash \gamma_1 \wedge \ldots \wedge \gamma_n \rightarrow \alpha$.*

El siguiente teorema resultará fundamental para la transformación de las fórmulas a forma prenexa, proceso necesario para el cálculo de δ-resolución.

Teorema 7.10 *Dadas las fórmulas $\alpha, \beta \in \mathcal{L}$, se verifican las siguientes equivalencias, siempre que se respeten las restricciones indicadas. La restricción* "i" *impone que la variable y no caiga bajo el alcance de ningún cuantificador que tenga a x como sufijo. La restricción* "ii" *requiere que x no ocurra libre en β.*

$$\vDash \forall x\alpha \leftrightarrow \forall y\alpha(x/y) \quad \textit{restricción} \text{ i} \tag{7.1}$$

$$\vDash \exists x\alpha \leftrightarrow \exists y\alpha(x/y) \quad \textit{restricción} \text{ i} \tag{7.2}$$

$$\vDash \forall x(\alpha \wedge \beta) \leftrightarrow (\forall x\alpha \wedge \forall x\beta) \tag{7.3}$$

$$\vDash \exists x(\alpha \vee \beta) \leftrightarrow (\exists x\alpha \vee \exists x\beta) \tag{7.4}$$

$$\vDash \forall x(\alpha \wedge \beta) \leftrightarrow (\forall x\alpha \wedge \beta) \quad \textit{restricción} \text{ ii} \tag{7.5}$$

$$\vDash \forall x(\alpha \vee \beta) \leftrightarrow (\forall x\alpha \vee \beta) \quad \textit{restricción} \text{ ii} \tag{7.6}$$

$$\vDash \exists x(\alpha \wedge \beta) \leftrightarrow (\exists x\alpha \wedge \beta) \quad \textit{restricción} \text{ ii} \tag{7.7}$$

$$\vDash \exists x(\alpha \vee \beta) \leftrightarrow (\exists x\alpha \vee \beta) \quad \textit{restricción} \text{ ii} \tag{7.8}$$

$$\vDash \neg\forall x\alpha \leftrightarrow \exists x\neg\alpha \tag{7.9}$$

$$\vDash \neg\exists x\alpha \leftrightarrow \forall x\neg\alpha \tag{7.10}$$

$$\vDash \forall x(\beta \rightarrow \alpha) \leftrightarrow (\beta \rightarrow \forall x\alpha) \quad \textit{restricción} \text{ ii} \tag{7.11}$$

$$\vDash \forall x(\alpha \rightarrow \beta) \leftrightarrow (\exists x\alpha \rightarrow \beta) \quad \textit{restricción} \text{ ii} \tag{7.12}$$

$$\vDash \exists x(\beta \rightarrow \alpha) \leftrightarrow (\beta \rightarrow \exists x\alpha) \quad \textit{restricción} \text{ ii} \tag{7.13}$$

$$\vDash \exists x(\alpha \rightarrow \beta) \leftrightarrow (\forall x\alpha \rightarrow \beta) \quad \textit{restricción} \text{ ii} \tag{7.14}$$

Demostración:

Por ser conocidas las equivalencias, no proporcionamos la demostración de cada una de ellas. Simplemente, para ilustrar cómo se lleva a cabo la evaluación de signos lógicos en primer orden con la semántica sin asignaciones de valores a las variables libres, veremos cómo se demuestra la equivalencia (7.12).

Sea $M \in MOD(\mathcal{L})$. Entonces $M(\forall x(\alpha \rightarrow \beta)) = 1$ syss para toda $M' =_t M$, se verifica $M'((\alpha \rightarrow \beta)(x/t)) = 1$. Esto equivale a $M'(\alpha(x/t) \rightarrow \beta(x/t)) = 1$ y, a su vez, como x no ocurre libre en β y entonces $\beta(x/t) = \beta$, y por evaluación de la implicación, equivale a $M'(\alpha(x/t)) = 0$ ó $M'(\beta) = 1$. Como lo anterior es válido para toda M' tal que $M' =_t M$, equivale, por evaluación de $\exists$, a que $M(\exists x\alpha) = 0$ ó $M(\beta) = 1$; pero esto último, por evaluación de la implicación, es equivalente a $M(\exists x\alpha \rightarrow \beta) = 1$. ■

El siguiente teorema es una versión para primer orden del teorema de intercambio, que demostramos en el capítulo 2 para lógica proposicional. Su demostración, que omitimos para abreviar, es una extensión de la del teorema 2.14.

Teorema 7.11 *Sean las sentencias $\phi_\alpha \in \mathcal{L}$, con α como una de sus subfórmulas, y ϕ_β, que difiere de la anterior sólo en que la subfórmula α está reemplazada por β. Entonces, si $\vDash \alpha \leftrightarrow \beta$, también $\vDash \phi_\alpha \leftrightarrow \phi_\beta$.*

Definición 7.12 *Una fórmula $\alpha \in \mathcal{L}$ está en* forma prenexa *syss tiene la forma, para $n \geq 0$,*

$$Q_1x_1Q_2x_2 \ldots Q_nx_n\beta,$$

donde, para cada $i \leq n$, $Q_i \in \{\forall, \exists\}$, x_i es la variable sufijo de Q_i, y en β —llamada matriz *de la fórmula— no ocurre ningún cuantificador. Cuando $n = 0$, se trata de una fórmula sin cuantificación, que también consideramos en forma prenexa.*

Definición 7.13 *Dada cualquier sentencia $\alpha \in \mathcal{L}$, definimos el procedimiento de transformación de α a forma prenexa, que consta de los siguientes pasos, que se iterarán cuantas veces sea necesario:*

1. *Eliminación de dobles implicaciones, aplicando la equivalencia* (2.15).

2. *Renombramiento de variables, mediante las equivalencias* (7.1) *y* (7.2)*, hasta que cada variable aparezca como índice de a lo sumo un cuantificador.*

3. *Interiorización de negadores, aplicando las equivalencias* (7.9) *y* (7.10)*, hasta que ningún negador tenga bajo su alcance un cuantificador.*

4. *Exteriorización de cuantificadores, aplicando las restantes equivalencias del teorema 7.10 hasta que la fórmula resultante no tenga ningún cuantificador bajo el alcance de otros operadores que no sean cuantificadores.*

Teorema 7.14 *Para cada sentencia $\alpha \in \mathcal{L}$ se puede hallar otra sentencia $\beta \in \mathcal{L}$ que está en forma prenexa tal que $\models \alpha \leftrightarrow \beta$.*

Demostración:

Como cada transformación de las que se emplean en la definición 7.13 se hace siguiendo una equivalencia, por el teorema 7.11 tenemos que la fórmula obtenida, en forma prenexa, es equivalente a α. ■

Para obtener las formas δ-clausales que usaremos en el cálculo de δ-resolución necesitamos normalizar las fórmulas. Para ello emplearemos una transformación dual a la forma normal de Skolem. A continuación presentamos ambas formas normales. Para la presentación de la forma normal de Skolem seguimos, en parte, a J. Cuena [Cue85].

Definición 7.15 *Decimos que una fórmula $\alpha \in \mathcal{L}$ está en* forma normal de Skolem *syss:*

1. *α está en forma prenexa*[3]*.*

2. *En α no aparecen cuantificadores existenciales.*

Definición 7.16 *Dada cualquier sentencia $\alpha \in \mathcal{L}$, definimos el procedimiento de* skolemización*, o de transformación de α a su forma normal de Skolem, que llamaremos $sko(\alpha)$, de la siguiente manera:*

[3]Es habitual exigir, además, que la matriz esté en forma normal conjuntiva. Sin embargo, para nuestros propósitos, no resulta necesario este requisito.

1. *A partir de α, obtenemos su forma prenexa, que llamaremos α_p, según la definición 7.13.*

2. *Se suprimen los cuantificadores existenciales de α_p, sustituyendo las variables cuantificadas existencialmente por* términos de Skolem, *de la siguiente forma:*

   a) *Los cuantificadores existenciales que no tengan a la izquierda ningún cuantificador universal se eliminan y se sustituye cada ocurrencia de la variable que cuantifican por un parámetro que no ocurra en la fórmula.*

   b) *Los cuantificadores existenciales que tengan a la izquierda algún cuantificador universal se eliminan igualmente, pero en este caso se sustituye cada ocurrencia de la variable que cuantifican por un término formado por un functor —que no debe haberse usado antes— que tiene como argumentos todas las variables que aparecen cuantificadas universalmente a la izquierda del cuantificador existencial que se elimina.*

El resultado de seguir los pasos anteriores es $sko(\alpha)$, la forma normal de Skolem de α.

Para cualquier sentencia $\alpha \in \mathcal{L}$, $sko(\alpha)$ tiene ciertas propiedades, la más importante de las cuales es que α es satisfactible syss $sko(\alpha)$ lo es. Sin embargo, más que presentar formalmente estos resultados, nos interesa mostrar un procedimiento dual a la transformación a forma normal de Skolem, que Samuel R. Buss [Bus95] llama *herbrandización*, y que tiene también una propiedad dual a la de la forma normal de Skolem. Ahora, dada cualquier sentencia $\alpha \in \mathcal{L}$, tendremos que α es universalmente válida syss el resultado de transformar α a la *forma normal de Herbrand*, $her(\alpha)$, es también una fórmula universalmente válida. Como se puede observar, las definiciones son duales a las que presentamos para la forma normal de Skolem.

Definición 7.17 *Decimos que una fórmula $\alpha \in \mathcal{L}$ está en* forma normal de Herbrand *syss:*

1. *α está en forma prenexa.*

2. *En α no aparecen cuantificadores universales.*

Definición 7.18 *Dada cualquier sentencia $\alpha \in \mathcal{L}$, definimos el procedimiento de transformación de α a su forma normal de Herbrand, que llamaremos $her(\alpha)$, de la siguiente manera:*

1. *A partir de α, obtenemos su forma prenexa, que llamaremos α_p, según la definición 7.13.*
2. *Se suprimen los cuantificadores universales de α_p comenzando por el que está más a la izquierda y continuando sucesivamente hacia la derecha, observando estas reglas:*
   a) *Los cuantificadores universales que no tengan a la izquierda ningún cuantificador existencial se eliminan y se sustituye cada ocurrencia de la variable que cuantifican por un parámetro que no ocurra en la fórmula.*
   b) *Los cuantificadores universales que tengan a la izquierda algún cuantificador existencial se eliminan igualmente, pero en este caso se sustituye cada ocurrencia de la variable que cuantifican por un término que consiste en un functor —que no debe haberse usado antes— que tiene como argumentos todas las variables que aparecen cuantificadas existencialmente a la izquierda del cuantificador universal que se elimina.*

*El resultado de seguir los pasos anteriores es $her(\alpha)$, la forma normal de Herbrand de α. A los términos que se introducen en los pasos 2*a *y 2*b *los llamaremos, como en la definición 7.16, términos de Skolem.*

Teorema 7.19 *Dada cualquier sentencia $\alpha \in \mathcal{L}$, se cumple $\alpha \vDash her(\alpha)$.*

Demostración:

Sea $\alpha \in \mathcal{L}$ cualquier sentencia, y $M = \langle \mathcal{D}, \Im \rangle$ una $\mathcal{L}$-estructura que satisface α. Probemos que también se cumple $M \vDash her(\alpha)$. El primer paso de transformación a forma normal de Herbrand es la obtención de la forma prenexa de α, α_p. Por el teorema 7.14 se verifica $\vDash \alpha \leftrightarrow \alpha_p$. Por tanto, $M \vDash \alpha_p$.

Tras obtener α_p, para alcanzar $her(\alpha)$ sólo resta eliminar de α_p todos los cuantificadores universales, según indica el paso 2 de la definición 7.18.

Ahora demostraremos que, para cualquier sentencia $\phi \in \mathcal{L}$ en que ocurra al menos un cuantificador universal, si $M \vDash \phi$, entonces $M \vDash \phi'$, siendo ϕ' es resultado de eliminar de ϕ el cuantificador universal que se encuentra más a la izquierda, de la forma que indica el paso 2 de la definición 7.18. Sea $n \geq 0$ el número de cuantificadores existenciales que ocurren a la izquierda del primer cuantificador universal de ϕ —si $n = 0$ se trata del caso $2a$ de la definición 7.18; si $n > 0$, estamos en el caso $2b$—, con lo que ϕ es la fórmula $\exists x_1, \ldots, x_n \forall y \beta$. Entonces, como $M \vDash \exists x_1, \ldots, x_n \forall y \beta$ tenemos, por n evaluaciones del cuantificador existencial, que existe una $\mathcal{L}$-estructura $M' =_{t_1,\ldots,t_n} M$, tal que $M' \vDash \forall y \beta(x_1, \ldots, x_n/t_1, \ldots, t_n)$. Sea t el término de Skolem que se introduce en sustitución de la variable y cuando se elimina su cuantificación. Se trata de un término que originalmente contenía como argumentos las variables $x_1, \ldots, x_n$, pero al haber sido sustituidas, respectivamente, por $t_1, \ldots, t_n$, ahora es un término sin variables libres. Entonces, por evaluación del cuantificador universal tenemos que para toda $M'' =_t M'$ se cumple $M'' \vDash \beta(x_1, \ldots, x_n, y/t_1, \ldots, t_n, t)$. Ahora bien, como es trivial que $M' =_t M'$, entonces $M' \vDash \beta(x_1, \ldots, x_n, y/t_1, \ldots, t_n, t)$. Por último, por n evaluaciones del cuantificador existencial, tenemos que $M \vDash \exists x_1, \ldots, x_n \beta(y/t)$. Pero esta fórmula es el resultado de eliminar en ϕ el primer cuantificador universal y reemplazar la variable cuantificada por el término o constante de Herbrand t. Por tanto, $M \vDash \phi'$.

De modo que si $M \vDash \alpha_p$, entonces M satisface el resultado de ir eliminando, sucesivamente, cada cuantificador universal de α_p, del modo que indica la definición 7.18. Por tanto, también $M \vDash her(\alpha)$.

Así que si $M \vDash \alpha$ también ocurre $M \vDash her(\alpha)$, por lo que $\alpha \vDash her(\alpha)$. ■

Teorema 7.20 *Dada cualquier sentencia $\alpha \in \mathcal{L}$, se verifica*

$$\vDash \alpha \textit{ syss } \vDash her(\alpha)$$

Demostración:

Sea α una sentencia de $\mathcal{L}$ cuya forma normal de Herbrand es $her(\alpha)$. Por el teorema 7.19, tenemos que si α es universalmente válida, también lo será $her(\alpha)$ por ser consecuencia lógica de α. Por tanto, sólo nos queda demostrar que si $her(\alpha)$ es universalmente válida, también lo será α. Lo probaremos por contraposición, es decir, que si α no es universalmente válida, entonces tampoco lo es $her(\alpha)$.

Sea $\alpha = \exists x_1, \ldots, x_n \forall y \beta$, y $her(\alpha) = \exists x_1, \ldots, x_n \beta(y/h(x_1, \ldots, x_n))$, para $n \geq 0$. Téngase en cuenta que si $n = 0$, $h(x_1, \ldots, x_n)$ será una constante, y en otro caso se tratará de un término complejo. La demostración que presentamos es válida para ambos casos.

Si $\not\models \alpha$, existe entonces una $\mathcal{L}$-estructura $M = \langle \mathcal{D}, \Im \rangle$ tal que

$$M \not\models \exists x_1, \ldots, x_n \forall y \beta \tag{7.15}$$

Como h no aparece en α, pues por ser $h(x_1, \ldots, x_n)$ un término de Skolem su functor debe ser nuevo, podemos, provisionalmente, considerar sin especificar la definición de $\Im(h)$ en M, ya que no afecta a la satisfacción de α en M.

Por (7.15) tenemos que, por n evaluaciones del cuantificador existencial, para todo $M' = \langle \mathcal{D}, \Im' \rangle$, si $M' =_{t_1,\ldots,t_n} M$, entonces,

$$M' \not\models \forall y \beta(x_1, \ldots, x_n / t_1, \ldots, t_n) \tag{7.16}$$

Por evaluación del cuantificador universal tenemos, desde (7.16), que para cierto parámetro c que no ocurre en α, existe algún $M'' = \langle \mathcal{D}, \Im'' \rangle$ tal que $M'' =_c M'$ y

$$M'' \not\models \beta(x_1, \ldots, x_n, y / t_1, \ldots, t_n, c) \tag{7.17}$$

Ahora es cuando vamos a definir $\Im(h)$ como la función constante, de modo que para toda n-upla de términos $u_1, \ldots, u_n$, $\Im(h(u_1, \ldots, u_n)) = \Im''(c)$. Entonces, como M, M'' y M' sólo difieren, a lo sumo, en la interpretación que hacen de los términos $t_1, \ldots, t_n, c$, tenemos que para toda n-upla de términos $u_1, \ldots, u_n$ se verifica $\Im(f(u_1, \ldots, u_n)) = \Im'(f(u_1, \ldots, u_n)) = \Im''(f(u_1, \ldots, u_n)) = \Im''(c)$. Por

tanto, desde (7.17) tenemos

$$M'' \not\models \beta(x_1, \ldots, x_n, y/t_1, \ldots, t_n, h(x_1, \ldots, x_n)) \tag{7.18}$$

Téngase en cuenta que como $x_1, \ldots, x_n$ han sido sustituidas, respectivamente, por $t_1, \ldots, t_n$, entonces $h(x_1, \ldots, x_n)$ denota, en (7.18), $h(t_1, \ldots, t_n)$. Pero $M'' =_c M'$, y el parámetro c no aparece en (7.18), pues cuando se introdujo en (7.17) era nuevo. Por tanto,

$$M' \not\models \beta(x_1, \ldots, x_n, y/t_1, \ldots, t_n, h(x_1, \ldots, x_n)) \tag{7.19}$$

Como el razonamiento que hemos hecho es para todo $M' =_{t_1,\ldots,t_n} M$, por n evaluaciones del cuantificador existencial, tenemos

$$M \not\models \exists x_1, \ldots, x_n \beta(y/h(x_1, \ldots, x_n)) \tag{7.20}$$

Pero (7.20) es $M \not\models her(\alpha)$, por lo que $her(\alpha)$ no es universalmente válida. De modo que si α no es universalmente válida tampoco lo es $her(\alpha)$. Por tanto, si $her(\alpha)$ es universalmente válida, también lo será α.

De forma que α es universalmente válida si y sólo si lo es $her(\alpha)$. ■

7.2. Cálculo de δ-resolución en primer orden

En esta sección presentamos el cálculo de δ-resolución para el lenguaje $\mathcal{L}$. En las siguientes definiciones volvemos a emplear muchos de los términos que hemos usado en capítulos anteriores al desarrollar el cálculo de δ-resolución proposicional: δ-cláusula, forma δ-clausal, regla de δ-resolución, etc. Como ahora estamos centrados en lógica de primer orden, no debe producirse ninguna ambigüedad entre los significados de estos conceptos en lógica proposicional y los que ahora tendrán.

Definición 7.21 *Una δ-cláusula Σ es un conjunto de literales de $\mathcal{L}$,*

$$\Sigma = \{\lambda_1, \ldots \lambda_n\}$$

Si $\{x_1, \ldots, x_m\}$, $m \geq 0$, es el conjunto de variables libres que aparecen en Σ entonces, dada una $\mathcal{L}$-estructura M, consideramos que M satisface Σ syss $M \vDash \exists x_1, \ldots, x_m(\lambda_1 \wedge \ldots \wedge \lambda_n)$. Entonces escribimos $M \vDash \Sigma$, como una abreviatura de lo anterior, ya que la semántica que hemos definido no contempla la interpretación de fórmulas con variables libres. Si $M \nvDash \exists x_1, \ldots, x_m(\lambda_1 \wedge \ldots \wedge \lambda_n)$ escribimos —también como abreviatura— $M \nvDash \Sigma$, y decimos que M no satisface Σ.

En adelante, nos referiremos a las δ-cláusulas mediante letras griegas mayúsculas. Con $\square$ nos referimos a la δ-cláusula vacía, universalmente válida.

Definición 7.22 *Una forma δ-clausal A es un conjunto de δ-cláusulas*

$$A = \{\Sigma_1, \ldots, \Sigma_n\}$$

Dada una $\mathcal{L}$-estructura M, consideramos que M satisface A, lo que escribimos $M \vDash A$ —en caso contrario, $M \nvDash A$— syss $M \vDash \Sigma_i$, para alguna $\Sigma_i \in A$.

En adelante, nos referiremos a las formas δ-clausales mediante letras mayúsculas latinas. La forma δ-clausal vacía es no satisfactible.

Definición 7.23 *Dada una sentencia $\alpha \in \mathcal{L}$, definimos el procedimiento de transformación de α a A, su forma δ-clausal, que consta de los siguientes pasos:*

1. *A partir de α obtenemos su forma normal de Herbrand, $her(\alpha)$.*

2. *Empleamos las equivalencias que aparecen en el teorema 2.13 para transformar la matriz de $her(\alpha)$ a forma normal disyuntiva[4], según se explica en el teorema 2.18. El resultado, que llamaremos $her(\alpha)_d$ tendrá la forma*

$$\exists x_1, \ldots, x_m((\lambda_{1_1} \wedge \ldots \wedge \lambda_{1_{j_1}}) \vee \ldots \vee (\lambda_{n_1} \wedge \ldots \wedge \lambda_{n_{j_n}}))$$

siendo n el número de conjunciones elementales de su matriz, y cada λ_{i_k} el k-ésimo literal, $1 \leq k \leq j_i$, de la i-ésima conjunción elemental, $1 \leq i \leq n$.

[4]Aunque estemos ante una fórmula de primer orden, siguen siendo válidas las equivalencias proposicionales, ya que la semántica de $\mathcal{L}$ es una extensión de la de $\mathcal{L}_p$.

3. *A partir de $her(\alpha)_d$, construimos directamente A, de la forma:*

$$A = \{\{\lambda_{1_1}, \ldots, \lambda_{1_{j_1}}\}, \ldots, \{\lambda_{n_1}, \ldots, \lambda_{n_{j_n}}\}\}$$

es decir, lo que hacemos es quitar los cuantificadores iniciales y construir una δ-cláusula por cada una de las conjunciones elementales que aparecen en la matriz de $her(\alpha)_d$.

Teorema 7.24 *Dada cualquier sentencia $\alpha \in \mathcal{L}$ su forma δ-clausal, A, es equivalente a $her(\alpha)$.*

Demostración:

Sea α una sentencia de $\mathcal{L}$ y A su forma δ-clausal, obtenida como indica la definición 7.23. El primer paso de esta definición es la construcción de la forma normal de Herbrand de α, por lo que hasta entonces se cumple el enunciado del teorema. Veamos que los dos pasos siguientes no alteran la satisfactibilidad de la fórmula, y el resultado es equivalente a $her(\alpha)$.

El paso 2 de la definición 7.23 se obtiene completamente a partir de equivalencias, por lo que $her(\alpha)_d$, resultado de este paso, sigue siendo equivalente a $her(\alpha)$.

Ahora veamos que el último paso también puede darse mediante ciertas equivalencias. Si $her(\alpha)_d$ es

$$\exists x_1, \ldots, x_m((\lambda_{1_1} \wedge \ldots \wedge \lambda_{1_{j_1}}) \vee \ldots \vee (\lambda_{n_1} \wedge \ldots \wedge \lambda_{n_{j_n}})) \tag{7.21}$$

A partir de (7.21) puede aplicarse repetidamente la equivalencia (7.4) para obtener

$$\exists x_1, \ldots, x_m(\lambda_{1_1} \wedge \ldots \wedge \lambda_{1_{j_1}}) \vee \ldots \vee \exists x_1, \ldots, x_m(\lambda_{n_1} \wedge \ldots \wedge \lambda_{n_{j_n}}) \tag{7.22}$$

De modo que (7.22) es equivalente a (7.21). Como demostraremos, también (7.22) es equivalente a A, forma δ-clausal de α. Para ello, por la definición 7.22, basta con probar que dada una $\mathcal{L}$-estructura cualquiera, para cada $1 \leq i \leq n$,

$$M \vDash \{\lambda_{i_1}, \ldots, \lambda_{i_{j_i}}\} \tag{7.23}$$

se verifica syss

$$M \vDash \exists x_1, \ldots, x_m(\lambda_{n_1} \wedge \ldots \wedge \lambda_{n_{j_n}}) \tag{7.24}$$

Como $x_1, \ldots, x_m$ son exactamente las variables libres que ocurren en (7.23), ya que α es una sentencia, por la definición 7.21 tenemos que se cumple (7.23) syss se cumple

$$M \vDash \exists y_1, \ldots, y_m(\lambda_{n_1} \wedge \ldots \wedge \lambda_{n_{j_n}}) \tag{7.25}$$

siendo $\langle y_1, \ldots, y_m \rangle$ las mismas variables $\langle x_1, \ldots, x_m \rangle$, pero tal vez ordenadas de otra forma. Si estuvieran ordenadas de la misma forma, entonces (7.25) es igual a (7.24), y tenemos lo que queríamos probar.

Supongamos que las variables $\langle y_1, \ldots, y_m \rangle$ están ordenadas de diferente manera que las $\langle x_1, \ldots, x_m \rangle$. Entonces, por m evaluaciones del cuantificador existencial tendríamos que (7.25) se cumple syss existe una $\mathcal{L}$-estructura $M' =_{t_1,\ldots,t_m} M$ tal que

$$M' \vDash (\lambda_{n_1} \wedge \ldots \wedge \lambda_{n_{j_n}})(x_1, \ldots, x_m/t_1, \ldots, t_m) \tag{7.26}$$

pero de nuevo, por otras m evaluaciones del cuantificador existencial, tenemos que (7.26) es equivalente a (7.24), que es lo que queríamos probar.

De forma que el último paso de la obtención de la forma δ-clausal no altera la equivalencia con $her(\alpha)$, por lo que para toda sentencia de $\mathcal{L}$, su forma δ-clausal es equivalente a su forma normal de Herbrand. ■

Corolario 7.25 *Sea $\alpha \in \mathcal{L}$ una sentencia cuya forma δ-clausal es A. Entonces, $\vDash A$ syss $\vDash her(\alpha)$.*

Demostración:

Este corolario es una consecuencia inmediata de los teoremas 7.20 y 7.24. Dado que α es válida syss $her(\alpha)$ lo es, y que A es equivalente a $her(\alpha)$, entonces α es válida syss A lo es. ■

A continuación introducimos las nociones de *unificador* y *unificador de máxima generalidad*, que serán fundamentales en el cálculo de δ-resolución. Para más detalles, remitimos a J. Cuena [Cue85], donde se encuentra también el algoritmo para la obtención del unificador de máxima generalidad.

Definición 7.26 *Dado $T = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, un conjunto de literales positivos de $\mathcal{L}$ tal que cada λ_i, $1 \leq i \leq n$, tiene la misma letra de predicado, decimos que una sustitución σ es un* unificador *de T syss*

$$\lambda_1\sigma = \lambda_2\sigma = \ldots = \lambda_n\sigma$$

Definición 7.27 *Dado $T = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, un conjunto de literales positivos de $\mathcal{L}$ decimos que es* unificable *syss existe un unificador para el mismo.*

Definición 7.28 *Dado $T = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, un conjunto unificable de literales positivos, decimos que σ es un* unificador de máxima generalidad *de T si para cualquier otro unificador ρ existe una sustitución ϵ tal que para cada fórmula $\alpha \in \mathcal{L}$, se cumple que*

$$(\alpha\ \sigma)\ \epsilon = \alpha\ \rho$$

Definición 7.29 *Dada la δ-cláusula Σ cuyas variables son $\{x_1, \ldots, x_n\}$, definimos el procedimiento de* renombramiento de variables *de Σ como la obtención de $\Sigma(x_1, \ldots, x_n/y_1, \ldots, y_n)$, siendo cada y_i, $1 \leq i \leq n$, una variable que no aparece en Σ.*

Definición 7.30 *Dadas $\Sigma_1 \cup \{\lambda\}$ y $\Sigma_2 \cup \{\neg\gamma\}$, dos δ-cláusulas cuyos conjuntos de variables son disjuntos, y además σ es un unificador de máxima generalidad de $\{\lambda, \gamma\}$, se define la* regla de δ-resolución *de la siguiente forma:*

$$\frac{\Sigma_1 \cup \{\lambda\} \qquad \Sigma_2 \cup \{\neg\gamma\}}{\Sigma_1\sigma \cup \Sigma_2\sigma}$$

De la δ-cláusula $\Sigma_1\sigma \cup \Sigma_2\sigma$ decimos que es un δ-resolvente de las δ-cláusulas $\Sigma_1 \cup \{\lambda\}$ y $\Sigma_2 \cup \{\neg\gamma\}$.

Definición 7.31 *Decimos que la δ-cláusula Λ puede demostrarse mediante δ-resolución a partir de las δ-cláusulas $\Sigma_1, \ldots, \Sigma_n$, lo que representamos como*

$$\Sigma_1, \ldots, \Sigma_n \vdash_\delta \Lambda$$

syss existe una secuencia de δ-cláusulas, a la que llamamos demostración mediante δ-resolución *de Λ a partir de $\Sigma_1, \ldots, \Sigma_n$, que cumple los siguientes requisitos:*

- *Cada una de las δ-cláusulas de la secuencia es:*

1. *Una de las* Σ_i, $1 \leq i \leq n$, *o bien*
2. *Una cláusula* $\Sigma\sigma$ *obtenida por renombramiento de variables de otra* δ*-cláusula* Σ *anterior, o bien*
3. *Un* δ*-resolvente de* δ*-cláusulas anteriores.*

- *La secuencia termina con la* δ*-cláusula* Λ.

La definición que hemos hecho del cálculo de δ-resolución emplea sólo la regla de δ-resolución y el renombramiento de variables, que se usará para lograr que las δ-cláusulas a las que se aplica la regla de δ-resolución tengan diferentes conjuntos de variables libres. Sin embargo, hay otras técnicas que pueden incorporarse al proceso de δ-resolución, y que sí hemos usado en la implementación que aparece en el apéndice D. Algunas de estas técnicas son las siguientes:

Eliminación de δ-cláusulas subsumidas. Una δ-cláusula Σ_1 está subsumida por otra δ-cláusula diferente Σ_2 si existe una sustitución σ tal que $\Sigma_2\sigma \subseteq \Sigma_1$. Cuando Σ_1 y Σ_2 aparecen en una prueba por δ-resolución, Σ_1 —la δ-cláusula subsumida— puede eliminarse.

Factores binarios. Si la δ-cláusula Σ contiene dos literales λ_1 y λ_2 unificables por la sustitución σ, es decir, $\lambda_1\sigma = \lambda_2\sigma$, entonces $\Sigma\sigma$ es un factor binario de Σ. Los factores binarios de las δ-cláusulas que aparecen en una demostración por δ-resolución pueden añadirse a la prueba.

Eliminación de δ-cláusulas contradictorias. Una δ-cláusula Σ es una contradicción si contiene dos literales complementarios[5]. Cada vez que aparezca una δ-cláusula contradictoria en una prueba por δ-resolución puede eliminarse.

Simplificación de factores. Si en una demostración por δ-resolución aparece una δ-cláusula Σ que contiene dos literales λ_1 y λ_2 para los que existe una sustitución σ tal que $\lambda_1\sigma = \lambda_2\sigma$ y $\Sigma\sigma \subset \Sigma$, entonces Σ puede reducirse a $\Sigma\sigma$ por simplificación de factores.

[5]Para que la δ-cláusula sea contradictoria, deben aparecer un literal λ y su negación $\neg\lambda$. Una δ-cláusula en que haya un literal γ y otro $\neg\eta$ tales que $\{\gamma, \eta\}$ son unificables, no es una δ-cláusula contradictoria.

Teorema 7.32 *Dada cualquier δ-cláusula Σ y cualquier sustitución σ, se verifica*[6]

$$\Sigma\sigma \vDash \Sigma$$

Demostración:

Sea $\Sigma = \{\lambda_1, \ldots, \lambda_h\}$ una δ-cláusula con las variables libres $\{x_1, \ldots, x_n\}$. Sea $\sigma = \{x_1, \ldots, x_n/t_1, \ldots, t_n\}$, de forma que cuando σ no modifica el valor de alguna x_i entonces $t_i = x_i$.

Sea $M = \langle \mathcal{D}, \Im \rangle$ tal que $M \vDash \Sigma\sigma$. Entonces, si $\{y_1, \ldots, y_m\}$ son las variables libres de $\Sigma\sigma$, tenemos que por la definición 7.21

$$M \vDash \exists y_1, \ldots, y_m \left((\lambda_1 \wedge \ldots \wedge \lambda_h)(x_1, \ldots, x_n/t_1, \ldots, t_n)\right) \tag{7.27}$$

Desde (7.27), por m evaluaciones del cuantificador existencial tenemos que existe $M' = \langle \mathcal{D}, \Im' \rangle$, tal que $M' =_{u_1,\ldots,u_m} M$, siendo $u_1, \ldots, u_m$ términos nuevos sin variables, de forma que:

$$M' \vDash \left((\lambda_1 \wedge \ldots \wedge \lambda_h)(x_1, \ldots, x_n/t_1, \ldots, t_n)\right)(y_1, \ldots, y_m/u_1, \ldots, u_m) \tag{7.28}$$

Ahora bien, sea π una sustitución tal que para $1 \leq i \leq n$,

$$(x_i\ \sigma)(y_1, \ldots, y_m/u_1, \ldots, u_m) = x_i\ \pi$$

por lo que $\pi = \{x_1, \ldots, x_n/v_1, \ldots, v_n\}$ siendo $\{v_1, \ldots, v_n\}$ términos sin variables. Entonces, por (7.28) tenemos:

$$M' \vDash (\lambda_1 \wedge \ldots \wedge \lambda_h)(x_1, \ldots, x_n/v_1, \ldots, v_n) \tag{7.29}$$

Por otra parte, para que se verifique $M \vDash \Sigma$, debe ocurrir, por la definición 7.21,

$$M \vDash \exists x_1, \ldots, x_n(\lambda_1 \wedge \ldots \wedge \lambda_h) \tag{7.30}$$

[6]La expresión $\Sigma\sigma \vDash \Sigma$ es, según lo que se indicó en la definición 7.21, una abreviatura de:

$$\exists y_1, \ldots, y_m((\lambda_1 \wedge \ldots \wedge \lambda_h)\sigma) \vDash \exists x_1, \ldots, x_n(\lambda_1 \wedge \ldots \wedge \lambda_h)$$

siendo $\{y_1, \ldots, y_m\}$ las variables libres de $\Sigma\sigma$, $\{x_1, \ldots, x_n\}$ las de Σ y $\Sigma = \{\lambda_1, \ldots, \lambda_h\}$.

Pero, por evaluación del cuantificador existencial, (7.30) se verifica syss existe una $\mathcal{L}$-estructura $M'' = \langle \mathcal{D}, \Im'' \rangle$ tal que $M'' =_{c_1,\ldots,c_n} M$ y

$$M'' \vDash (\lambda_1 \wedge \ldots \wedge \lambda_h)(x_1, \ldots, x_n/c_1, \ldots, c_n) \tag{7.31}$$

Pero sabemos que existe tal M'', pues basta que se construya a partir de M y M' considerando que para toda $1 \leq i \leq n$ debe ocurrir $\Im''(c_i) = \Im'(v_i)$. Entonces, (7.31) se verifica, y por tanto también (7.30). Por tanto, si $M \vDash \Sigma\sigma$, entonces $M \vDash \Sigma$, por lo que $\Sigma\sigma \vDash \Sigma$. ■

Teorema 7.33 *Sea $\Sigma_1\sigma \cup \Sigma_2\sigma$ un δ-resolvente de $\Sigma_1 \cup \{\lambda\}$ y $\Sigma_2 \cup \{\neg\gamma\}$. Entonces, toda $\mathcal{L}$-estructura que satisfaga $\Sigma_1\sigma \cup \Sigma_2\sigma$ satisface o bien a $\Sigma_1 \cup \{\lambda\}$ o a $\Sigma_2 \cup \{\neg\gamma\}$.*

Demostración:

Sean $\Sigma_1 \cup \{\lambda\} = \{\eta_1^1, \ldots, \eta_1^{k_1}, \lambda\}$ y $\Sigma_2 \cup \{\neg\gamma\} = \{\eta_2^1, \ldots, \eta_2^{k_2}, \neg\gamma\}$. Sean $\{x_1, \ldots, x_n\}$ las variables libres de $\Sigma_1\sigma \cup \Sigma_2\sigma$, δ-resolvente de las dos δ-cláusulas anteriores. Por tanto, la sustitución σ es un unificador —de máxima generalidad— de λ, γ.

Supongamos que $M \vDash \Sigma_1\sigma \cup \Sigma_2\sigma$. Entonces, por definición,

$$M \vDash \exists x_1, \ldots, x_n((\eta_1^1 \wedge \ldots \wedge \eta_1^{k_1} \wedge \eta_2^1 \wedge \ldots \wedge \eta_2^{k_2})\sigma) \tag{7.32}$$

Entonces, por evaluación de los n cuantificadores existenciales de (7.32), existe $M' =_{b_1,\ldots,b_n} M$ tal que

$$M' \vDash ((\eta_1^1 \wedge \ldots \wedge \eta_1^{k_1} \wedge \eta_2^1 \wedge \ldots \wedge \eta_2^{k_2})\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n) \tag{7.33}$$

Pero desde (7.33), por evaluación del conjuntor tenemos

$$M' \vDash ((\eta_1^1 \wedge \ldots \wedge \eta_1^{k_1})\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n) \tag{7.34}$$

y

$$M' \vDash ((\eta_2^1 \wedge \ldots \wedge \eta_2^{k_2})\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n) \tag{7.35}$$

Pero por ser σ un unificador de $\{\lambda, \gamma\}$, entonces $\lambda\sigma = \gamma\sigma$, y por tanto también

$$(\lambda\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n) = (\gamma\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n)$$

y por tanto $M' \vDash \lambda$ ó $M' \vDash \neg\gamma$. Pero entonces, por (7.34) y (7.35) tenemos que

$$M' \vDash ((\eta_1^1 \wedge \ldots \wedge \eta_1^{k_1} \wedge \lambda)\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n)$$

o bien

$$M' \vDash ((\eta_2^1 \wedge \ldots \wedge \eta_2^{k_2} \wedge \neg\gamma)\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n)$$

Pero entonces, por la definición 7.21

$$M' \vDash ((\Sigma_1 \cup \{\lambda\})\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n)$$

o bien

$$M' \vDash ((\Sigma_2 \cup \{\neg\gamma\})\sigma)(x_1, \ldots, x_n/b_1, \ldots, b_n)$$

Entonces, por n evaluaciones del cuantificador existencial tenemos que $M \vDash (\Sigma_1 \cup \{\lambda\})\sigma$ o bien $M \vDash (\Sigma_2 \cup \{\neg\gamma\})\sigma$. Finalmente, por el teorema 7.32 tenemos que $M \vDash \Sigma_1 \cup \{\lambda\}$ o bien $M \vDash \Sigma_2 \cup \{\neg\gamma\}$. ■

Teorema 7.34 *Sea A un conjunto de δ-cláusulas y Σ una δ-cláusula cualquiera, tal que $A \vdash_\delta \Sigma$. Entonces $\Sigma \vDash A$.*

Demostración:

Sea A un conjunto de δ-cláusulas y Σ una δ-cláusula tal que $A \vdash_\delta \Sigma$. Entonces, por la definición 7.31 existe una secuencia de δ-cláusulas $\Omega = \langle \Theta_1, \ldots, \Theta_n \rangle$, $n \geq 1$, $\Theta_n = \Sigma$, que constituye la prueba por δ-resolución de Σ a partir de A. Probaremos que para cada δ-cláusula Θ_i, $1 \leq i \leq n$, de Ω se verifica

$$\Theta_i \vDash A$$

Procedemos por inducción sobre i. En el caso base, $i = 1$, resulta trivial que $\Theta_1 \vDash A$, ya que por la definición 7.31 ocurre que $\Theta_1 \in A$, y por la definición 7.22 toda $\mathcal{L}$-estructura que satisfaga al menos una δ-cláusula de A satisface A.

Como hipótesis de inducción, supongamos que A es consecuencia lógica de cada una de las δ-cláusulas $\Theta_1, \ldots, \Theta_m$, para $m \geq 1$. Probemos ahora que $\Theta_{m+1} \vDash A$. Por la definición 7.31, pueden ocurrir tres posibilidades:

1. $\Theta_{m+1} \in A$. Entonces, como en el caso base, resulta trivial que $\Theta_{m+1} \vDash A$, por la definición 7.22.

2. $\Theta_{m+1} = \Theta_k\sigma$ para $k \leq m$, es decir, Θ_{m+1} procede a partir de una δ-cláusula anterior por renombramiento de variables. Entonces, por el teorema 7.32 tenemos que $\Theta_{m+1} \vDash \Theta_k$, y como por hipótesis de inducción $\Theta_k \vDash A$, entonces, por la transitividad de la relación de consecuencia lógica clásica, $\Theta_{m+1} \vDash A$.

3. Θ_{m+1} procede de aplicar la regla de δ-resolución a dos δ-cláusulas anteriores Θ_{k_1} y Θ_{k_2}. Sea $M = \langle \mathcal{D}, \Im \rangle$ tal que $M \vDash \Theta_{m+1}$. Por el teorema 7.33 tenemos que $M \vDash \Theta_{k_1}$ o bien $M \vDash \Theta_{k_2}$. Por hipótesis de inducción tenemos que A es consecuencia lógica tanto de Θ_{k_1} como de Θ_{k_2}, por lo que $M \vDash A$. Por tanto, $\Theta_{m+1} \vDash A$.

Si cada Θ_i, $1 \leq i \leq n$ verifica $\Theta_i \vDash A$, entonces también $\Sigma \vDash A$, ya que $\Sigma = \Theta_n$. ■

Corolario 7.35 (Corrección de la δ-resolución) *Sea la fórmula $\alpha \in \mathcal{L}$ tal que su forma δ-clausal A verifica $A \vdash_\delta \square$. Entonces, $\vDash \alpha$.*

Demostración:

Supongamos que $A \vdash_\delta \square$. Entonces, por el teorema 7.34 tenemos que $\square \vDash A$, por lo que A es un conjunto universalmente válido de δ-cláusulas. Pero por ser A la forma δ-clausal de α, por el teorema 7.24 resulta que A es equivalente a $her(\alpha)$, con lo que $her(\alpha)$ es también universalmente válida. Por el teorema 7.20 esto implica que $\vDash \alpha$. ■

Teorema 7.36 (Completud de la δ-resolución) *Sea $\alpha \in \mathcal{L}$ una sentencia tal que $\vDash \alpha$. Entonces, si A es la forma δ-clausal de α se verifica $A \vdash_\delta \square$.*

Demostración:

Para probar directamente la completud del cálculo de δ-resolución en primer orden debemos primero demostrar un teorema dual al teorema de Herbrand, y proceder de forma análoga a la prueba de la completud del cálculo de resolución en primer orden. Sin embargo, para abreviar, recurriremos a la dualidad entre la resolución y la δ-resolución, que se mantiene en primer orden, y probaremos la completud de la δ-resolución a partir de la completud de la resolución.

Sea $\alpha \in \mathcal{L}$ una sentencia tal que $\models \alpha$, y A la forma δ-clausal de α. Entonces, por el teorema 7.20 tenemos $\models her(\alpha)$, y a su vez por el teorema 7.24 que $\models A$. Sea

$$A = \{\{\lambda_1^1, \ldots, \lambda_1^{k_1}\}, \ldots, \{\lambda_n^1, \ldots, \lambda_n^{k_n}\}\}$$

y sean $\{x_i^1, \ldots, x_i^{l_i}\}$ las variables libres de $\{\lambda_i^1, \ldots, \lambda_i^{k_i}\}$, para cada $1 \leq i \leq n$. Entonces,

$$\models \{\{\lambda_1^1, \ldots, \lambda_1^{k_1}\}, \ldots, \{\lambda_n^1, \ldots, \lambda_n^{k_n}\}\} \tag{7.36}$$

lo cual, por las definiciones 7.21 y 7.22, equivale a

$$\models \left(\exists x_1^1 \ldots x_1^{l_1}(\lambda_1^1 \wedge \ldots \wedge \lambda_1^{k_1})\right) \vee \ldots \vee \left(\exists x_n^1 \ldots x_n^{l_n}(\lambda_n^1 \wedge \ldots \wedge \lambda_n^{k_n})\right) \tag{7.37}$$

y a su vez a

$$\neg \left(\left(\exists x_1^1 \ldots x_1^{l_1}(\lambda_1^1 \wedge \ldots \wedge \lambda_1^{k_1})\right) \vee \ldots \vee \left(\exists x_n^1 \ldots x_n^{l_n}(\lambda_n^1 \wedge \ldots \wedge \lambda_n^{k_n})\right)\right) \models \bot \tag{7.38}$$

Pero aplicando sucesivamente equivalencias (7.38) se transforma en

$$\left(\forall x_1^1 \ldots x_1^{l_1}(\overline{\lambda_1^1} \vee \ldots \vee \overline{\lambda_1^{k_1}})\right) \wedge \ldots \wedge \left(\forall x_n^1 \ldots x_n^{l_n}(\overline{\lambda_n^1} \vee \ldots \vee \overline{\lambda_n^{k_n}})\right) \models \bot \tag{7.39}$$

siendo $\overline{\lambda_i^j}$, $1 \leq i \leq n$, $1 \leq j \leq k_i$ el literal complementario de λ_i^j.

Pero en el cálculo de resolución la fórmula que aparece en (7.39) es equivalente a la forma clausal

$$B = \{\{\overline{\lambda_1^1}, \ldots, \overline{\lambda_1^{k_1}}\}, \ldots, \{\overline{\lambda_n^1}, \ldots, \overline{\lambda_n^{k_n}}\}\} \tag{7.40}$$

que es no satisfactible, según (7.39). Por tanto, asumiendo la completud del cálculo de resolución en primer orden[7], desde B puede alcanzarse la cláusula vacía mediante la aplicación de repetidas sustituciones y de la regla de resolución.

[7]La prueba de la completud del cálculo de resolución en primer orden, basada en el teorema de Herbrand, es muy popular. Puede encontrarse, por ejemplo, en [Dav89].

Pero B es dual a A. La única diferencia es que los literales que aparecen en B son siempre complementarios a los de A. Además, la regla de δ-resolución y la operación de sustitución son, en el cálculo de δ-resolución, sintácticamente idénticas —aunque semánticamente duales— a la regla de resolución y a las sustituciones de la resolución clásica. Por tanto, para cada cláusula $\{\overline{\gamma_1}, \ldots, \overline{\gamma_m}\}$ que puede alcanzarse desde B por resolución clásica, la δ-cláusula $\{\gamma_1, \ldots, \gamma_m\}$ puede alcanzarse desde A mediante δ-resolución. Entonces, como desde B se alcanza por resolución la cláusula vacía, también $A \vdash_\delta \square$. ■

Definición 7.37 (δ-explicación) *Sea $\alpha \in \mathcal{L}$ una sentencia, cuya forma δ-clausal es A, tal que $\not\models \alpha$. Entonces, decimos que la sentencia $\beta \in \mathcal{L}$ es una δ-explicación de α si se cumplen las siguientes condiciones:*

1. *β tiene la forma $Q_1 x_1 \ldots Q_{n_1} x_{n_1}(\lambda_1 \wedge \ldots \wedge \lambda_{m_1})$, $n_1 \geq 0$, siendo cada λ_i, $1 \leq i \leq m_1$, un literal, y cada Q_j, $1 \leq j \leq n_1$, un cuantificador con la variable x_j como índice.*

2. *Es posible construir la forma normal de Skolem, $sko(\beta) = \forall y_1, \ldots, y_{n_2}(\lambda_1 \wedge \ldots \wedge \lambda_{m_1})\sigma$, $n_2 \leq n$, siendo σ la sustitución que asigna a cada una de las variables cuantificadas existencialmente en β un término nuevo, de modo que ninguno[8] de los functores ni constantes que aparecen en A pero no en α esté en $sko(\beta)$.*

3. *Existe una δ-cláusula $\Sigma = \{\gamma_1, \ldots, \gamma_{m_2}\}$ tal que:*

   a) *$A \vdash_\delta \Sigma$.*

   b) *Dada la forma normal de Skolem $sko(\beta) = \forall y_1, \ldots, y_{n_2}(\lambda_1 \wedge \ldots \wedge \lambda_{m_1})\sigma$ indicada en el punto 2 anterior, para cada $\gamma_k \in \Sigma$, $1 \leq k \leq m_2$ existe un $\lambda_i\sigma$, $1 \leq i \leq m_1$, tal que para alguna sustitución posible π_k se cumple $\lambda_i\sigma\pi_k = \gamma_k$.*

Mediante las δ-explicaciones construiremos explicaciones abductivas, tal como vemos en el siguiente ejemplo. Sea la fórmula correspondiente a la teoría

[8] Este requisito hace que ninguno de los términos de Skolem que se introdujeron en A al hacer $her(\alpha)$ ni sus functores correspondientes aparezcan ni en β ni en $sko(\beta)$.

$\forall x(\forall y(Rxy \wedge Ryx) \rightarrow Tx)$ y la correspondiente a la observación Ta. Entonces, como hacíamos en la δ-resolución proposicional obtenemos la forma δ-clausal de la implicación que tiene como antecedente la teoría y como consecuente la observación, es decir

$$(\forall x\,(\forall y\,(Rxy \wedge Ryx) \rightarrow Tx)) \rightarrow Ta$$

La forma prenexa correspondiente es

$$\exists x \forall y\,(((Rxy \wedge Ryx) \rightarrow Tx)) \rightarrow Ta$$

cuya forma normal de Herbrand es

$$\exists x\,(((Rxh(x) \wedge Rh(x)x) \rightarrow Tx)) \rightarrow Ta$$

Al poner la matriz en forma normal disyuntiva obtenemos

$$\exists x\,((Rxh(x) \wedge Rh(x)x \wedge \neg Tx) \vee Ta)$$

de donde sale la forma δ-clausal

$$\{\{Rxh(x), Rh(x)x, \neg Tx\}, \{Ta\}\}$$

Como sólo podemos hacer una aplicación de la regla de δ-resolución obtenemos la δ-cláusula $\{Rah(a), Rh(a)a\}$. A partir de ella pueden construirse ciertas δ-explicaciones que cumplen los requisitos de la definición 7.37, como por ejemplo $\forall x(Rax \wedge Rxa)$ y $\forall xyRxy$. Fácilmente puede comprobarse que Ta es consecuencia lógica de la teoría con cualquiera de las dos. Sin embargo, como $\forall xyRxy \vDash \forall x(Rax \wedge Rxy)$, la primera de las explicaciones es *minimal* con respecto a la segunda.

Para obtener δ-explicaciones resulta interesante el algoritmo que P.T. Cox y T. Pietrzykowski [CP84] proporcionan para realizar *skolemización inversa*. Así, dado un literal λ que contiene términos de Skolem y las variables libres $x_1, \ldots, x_n$, devuelve un conjunto C de literales cuantificados y sin variables libres con las siguientes propiedades:

Corrección: si $\sigma \in C$, entonces $\vDash \sigma \rightarrow \exists x_1, \ldots, x_n(\lambda)$.

Completud: Para cada sentencia $\alpha \in \mathcal{L}$, si $\models \alpha \rightarrow \lambda$, entonces existe un literal cuantificado $\sigma \in C$ tal que $\models \alpha \rightarrow \sigma$.

No redundancia: Si $\sigma, \alpha \in C$, entonces $\not\models \sigma \rightarrow \alpha$ y $\not\models \alpha \rightarrow \sigma$.

Mediante una adaptación de este algoritmo a conjunciones de literales es posible construir las δ-explicaciones. Además, la propiedad de *no redundancia* es deseable para evitar δ-explicaciones no minimales, como ocurría en el ejemplo anterior. M. Cialdea y F. Pirri [CP93] presentan una extensión del algoritmo de Cox y Pietrzykowski que puede aplicarse a conjunciones de literales, y por tanto a la construcción de δ-explicaciones. A continuación proporcionamos una adaptación de su definición.

Definición 7.38 (Skolemización inversa) *Sea $\alpha \in \mathcal{L}$ una fórmula en forma normal de Skolem y sea $ts(\alpha) = \{t_1, \ldots, t_k\}$ el conjunto de términos de Skolem y variables libres que aparecen en α. Sea $\langle t_{p_1}, \ldots, t_{p_k}\rangle$ un orden total de los elementos de $ts(\alpha)$ de forma que para cada i y j, si t_{p_i} ocurre en t_{p_j} entonces $i < j$. Entonces $Q_1 x_1 \ldots Q_k x_k \alpha'$, $Q_l \in \{\forall, \exists\}$, se obtiene desde α por* skolemización inversa *a partir de $\langle t_{p_1}, \ldots, t_{p_k}\rangle$ syss*

- *α' se obtiene desde α reemplazando cada término t_{p_i} por la variable nueva x_i, y además*
- *Para cada i, si t_{p_i} es una variable, entonces Q_i es un cuantificador existencial; en otro caso, si t_{p_i} es un término de Skolem, Q_i es un cuantificador universal.*

Al conjunto de todas las fórmulas que pueden obtenerse por skolemización inversa desde α lo llamaremos $desk(\alpha)$.

Siguiendo esta definición, para la δ-cláusula $\{Rah(a), Rh(a)a\}$ del ejemplo anterior podemos construir una δ-explicación tomando $\alpha = Rah(a) \wedge Rh(a)a$. Entonces $ts(\alpha) = \{h(a)\}$ y sólo es posible un orden total, $\langle h(a)\rangle$. Por tanto, sólo puede obtenerse, por skolemización inversa, la fórmula $\forall x(Rax \wedge Rxa)$, ya que $h(a)$ es un término de Skolem, por lo que debe cuantificarse universalmente.

Teorema 7.39 *Sean α, β dos sentencias de $\mathcal{L}$. Entonces, $\beta \vDash \alpha$ syss $sko(\beta) \vDash her(\alpha)$.*

Demostración:

Este teorema lo hemos tomado de S.R. Buss [Bus95], donde puede encontrarse la prueba con todo detalle; para simplificar, presentamos un esquema de su demostración. Partiendo de $\beta \vDash \alpha$ tenemos que por el teorema 7.19 se cumple $\beta \vDash her(\alpha)$. Mediante un teorema similar al 7.19 para formas normales de Skolem —cuya prueba sería dual a la del teorema presentado— tendríamos que $sko(\beta) \vDash \beta$ y por tanto $sko(\beta) \vDash her(\alpha)$.

En sentido inverso, partiendo de $sko(\beta) \vDash her(\alpha)$ podemos llegar a $\beta \vDash \alpha$ teniendo en cuenta que cada uno de los términos de Skolem introducidos es nuevo, y que por tanto los que ocurren en $sko(\beta)$ son diferentes a los que aparecen en $her(\alpha)$. Concretamente, desde $sko(\beta) \vDash her(\alpha)$ se justifica el paso a $sko(\beta) \vDash \alpha$ de forma análoga a como en el cálculo de secuentes se hace con la la introducción de $\forall$ en el consecuente. Desde ahí se pasa a $\beta \vDash \alpha$ de modo similar a la introducción de $\exists$ en el antecedente. ■

Teorema 7.40 (Corrección abductiva débil) *Sea $\alpha, \beta \in \mathcal{L}$ dos sentencias tales que $\nvDash \alpha$ y β es una δ-explicación de α. Entonces, $\beta \vDash \alpha$.*

Demostración:

Sean $\alpha, \beta \in \mathcal{L}$ dos sentencias tales que $\nvDash \alpha$ y β es una δ-explicación de α. Además, sea A la forma δ-clausal de α. Como β es una δ-explicación de α tenemos por la definición 7.37 que:

- $sko(\beta) = \forall x_1 \ldots x_n(\lambda_1 \wedge \ldots \wedge \lambda_m)$
- Existe $\Sigma = \{\gamma_1, \ldots, \gamma_k\}$ tal que $A \vdash_\delta \Sigma$.
- Para cada γ_j, $1 \leq j \leq k$, existe algún λ_i, $1 \leq i \leq m$, y una sustitución π_j tal que $\lambda_i \pi_j = \gamma_j$.

Entonces,

$$\neg sko(\beta) = \exists x_1 \ldots x_n(\overline{\lambda_1} \vee \ldots \vee \overline{\lambda_m})$$

y por la definición 7.22 tenemos que $\neg sko(\beta)$ equivale al conjunto de δ-clausales

$$B = \{\{\overline{\lambda_1}\}, \ldots, \{\overline{\lambda_m}\}\}$$

Pero para cada γ_j de Σ hay un λ_i tal que existe una sustitución π_j tal que $\lambda_i\pi_j = \gamma_j$. Por tanto cada γ_j de Σ puede eliminarse aplicando la regla de δ-resolución con una δ-cláusula $\{\lambda_i\}$ de B. Entonces,

$$B \cup \{\Sigma\} \vdash_\delta \square$$

Pero como $A \vdash_\delta \Sigma$ tenemos

$$B \cup A \vdash_\delta \square$$

lo que por el teorema 7.34 significa

$$\models B \cup A$$

De modo que como A es equivalente a $her(\alpha)$ —teorema 7.24— y B a $\neg sko(\beta)$, tenemos por la definición 7.22

$$\models \neg sko(\beta) \vee her(\alpha)$$

lo cual equivale a

$$sko(\beta) \models her(\alpha)$$

y por el teorema 7.39 llegamos a

$$\beta \models \alpha$$

■

Teorema 7.41 (Completud abductiva débil) *Sea $\alpha, \beta \in \mathcal{L}$ dos sentencias tales que:*

- $\not\models \alpha$
- $\beta \models \alpha$

- *β es una fórmula en forma prenexa cuya matriz es una conjunción de literales.*

Entonces, β es una δ-explicación de α.

Demostración:

Sean α y β dos sentencias de $\mathcal{L}$ tales que $\not\models \alpha$, $\beta \models \alpha$ y además β está en forma prenexa y su matriz es una conjunción de literales. Debemos probar que β es una δ-explicación de α.

Dado que

$$\beta \models \alpha$$

por el teorema 7.39 tenemos

$$sko(\beta) \models her(\alpha)$$

y a su vez

$$\models \neg sko(\beta) \vee her(\alpha)$$

Sea $sko(\beta) = \forall x_1 \ldots x_n(\lambda_1 \wedge \ldots \lambda_m)$. Entonces $\neg sko(\beta) = \exists x_1 \ldots x_n(\overline{\lambda_1} \vee \ldots \vee \overline{\lambda_m})$, lo cual es equivalente al conjunto de δ-cláusulas

$$B = \{\{\overline{\lambda_1}\}, \ldots, \{\overline{\lambda_m}\}\}$$

Además, sea A la forma δ-clausal de α, equivalente a $her(\alpha)$ por el teorema 7.24. Entonces, por la definición 7.22 tenemos:

$$\models B \cup A$$

y por el teorema 7.36 tenemos que

$$B \cup A \vdash_\delta \square$$

Ahora bien, sea Dem la secuencia de δ-cláusulas que constituye una demostración de $\square$ por δ-resolución a partir de $A \cup B$, tal como indica la definición 7.31. A partir de Dem podemos construir Dem', que se caracteriza porque

no contiene δ-cláusulas de B. Sin embargo, toda vez que en Dem se aplica por k-ésima vez la regla de δ-resolución entre alguna $\{\overline{\lambda_i}\}$, $1 \leq i \leq m$, y una δ-cláusula tipo $\Omega \cup \{\eta\}$ para obtener $\Omega\sigma_k$, en Dem' se aplica a la δ-cláusula correspondiente Ω' la misma sustitución σ_k. Entonces, si en Dem se hicieron t usos de la regla de δ-resolución con δ-cláusulas de B, al final de Dem' se alcanza una δ-cláusula $\Sigma = \{\gamma_1, \ldots, \gamma_t\}$.

Como en Dem' sólo se han usado δ-cláusulas de A, tenemos que $A \vdash_\delta \Sigma$. Además, cada literal γ_k de Σ resuelve con algún $\overline{\lambda_i}$ aplicando a $\overline{\lambda_i}$ la sustitución σ_k. Por tanto, como

$$sko(\beta) = \forall x_1 \ldots x_n(\lambda_1 \wedge \ldots \wedge \lambda_m)$$

para cada $\gamma_k \in \Sigma$ hay un literal λ_i y una sustitución σ_k tal que $\lambda_i\sigma_k = \gamma_k$. Pero este es el requisito fundamental para que β sea una δ-explicación de α. Los demás requisitos de la definición 7.37 se verifican igualmente, por encontrarse entre los supuestos de partida. ■

La construcción de un proceso abductivo mediante δ-resolución en primer orden presenta algunos problemas que no aparecían en lógica proposicional. El proceso que entonces presentamos en la definición 5.23 se basaba en la noción de saturación por δ-resolución, introducida en la definición 4.18. Sin embargo en lógica de primer orden, dada su indecidibilidad, la saturación de los conjuntos de δ-cláusulas no queda por lo general asegurada. Es más, la introducción de términos de Skolem durante la conversión a forma δ-clausal —al realizar la transformación a forma normal de Herbrand— impide ciertas unificaciones, con lo que no es posible, por lo general, reconocer durante el proceso abductivo situaciones en que la observación refuta la teoría, o bien en que las abducciones generadas —que serán δ-cláusulas a las que habrá que someter a *skolemización inversa* para obtener las δ-explicaciones correspondientes— no son explicativas.

Para ilustrar lo que acabamos de comentar, veamos un ejemplo muy simple. Sea la teoría $\Theta = \exists x Tx$ y la observación $\phi = \forall y \neg Ty$. En este caso, la observación refuta la teoría, puesto que $\Theta \models \neg\phi$. Recordemos que en el proceso abductivo proposicional —definición 5.23— esta situación se reconocía porque, tras obtener N^δ_Θ y O^δ mediante saturación por δ-resolución de las formas δ-clausales de $\neg\Theta$ y

ϕ, respectivamente, cada δ-cláusula de O^δ estaba subsumida por una de N^δ_Θ. Si trasladamos esto a nuestro ejemplo en primer orden, tenemos que $N^\delta_\Theta = \{\{\neg Th_1\}\}$ y $O^\delta = \{\{\neg Th_2\}\}$, siendo h_1 y h_2 los términos de Skolem introducidos durante la transformación a forma normal de Herbrand. Como se observa, estos términos impiden que $\{\neg Th_2\}$ pueda ser subsumida por $\{\neg Th_1\}$, con lo que no es posible constatar durante el proceso abductivo que $\Theta \models \neg\phi$.

Sin embargo, sí es posible verificarlo en un proceso independiente. De hecho,

$$\Theta \models \neg\phi$$

syss

$$\models \Theta \rightarrow \phi$$

syss

$$\models \{\{\neg Th_1\}, \{Ty\}\}$$

al ser $\{\{\neg Th_1\}, \{Ty\}\}$ la forma δ-clausal de $\Theta \rightarrow \phi$. Pero esto último se verifica, ya que con una sola aplicación de la regla de δ-resolución se alcanza $\square$.

Del mismo modo también se puede comprobar si las abducciones que se obtienen son explicativas, aunque siempre en procesos independientes, y con las limitaciones propias de la lógica de primer orden. Es decir, a cada δ-cláusula que se obtenga puede aplicarse skolemización inversa, y comprobar luego si las δ-explicaciones obtenidas cumplen los requisitos para ser consideradas abducciones explicativas.

La opción que hemos tomado en la implementación que a continuación presentaremos es adaptar el proceso de la definición 5.23 de la siguiente manera:

Análisis de la teoría. Dada la teoría Θ —conjunción de sentencias de $\mathcal{L}$— se obtiene la forma δ-clausal de $\neg\Theta$, que llamaremos N_Θ. Entonces se aplica δ-resolución a las δ-cláusulas de N_Θ. Si el proceso llega a saturarse —no es posible una aplicación más de la regla de δ-resolución en que aparezca una δ-cláusula que no está ya—, sea N^δ_Θ el conjunto obtenido. Entonces,

- Si $\square \in N^\delta_\Theta$, entonces se devuelve un mensaje que informa de que la teoría es no satisfactible. Ello es verdad puesto que por el teorema 7.35 tenemos que $\models \neg\Theta$.

- En otro caso, continúa el proceso.

Análisis de la observación. Dada la observación ϕ —sentencia de $\mathcal{L}$—, se obtiene su forma δ-clausal, O, a la que se aplica δ-resolución. Si al proceso se satura, sea O^δ el conjunto obtenido. Entonces,

- Si $\square \in O^\delta$ entonces se devuelve un mensaje que avisa de que la observación es universalmente válida. Esto se verifica igualmente por el teorema 7.35.
- En otro caso, continúa el proceso.

Búsqueda de refutaciones. Se eliminan de O^δ todas las δ-cláusulas subsumidas por alguna δ-cláusula de N^δ_Θ. Entonces,

- Si no queda ninguna δ-cláusula en O^δ se devuelve un mensaje que dice que la observación refuta la teoría. Puede verificarse que esto es cierto a partir del teorema 7.41, pues en este caso tendríamos que toda δ-explicación de ϕ es también una δ-explicación de $\neg\Theta$.
- En otro caso, continúa al proceso, siendo B el conjunto unión de las δ-cláusulas de N^δ_Θ y aquellas de O^δ no subsumidas.

Búsqueda de explicaciones. Se aplica δ-resolución a las δ-cláusulas de B. Si el proceso se satura, sea B^δ el conjunto obtenido. Entonces,

- Si $\square \in B^\delta$, se devuelve un mensaje que informa de que la observación es consecuencia lógica de la teoría, tal como garantiza el teorema 7.35.
- En otro caso, se devuelven aquellas δ-cláusulas de B^δ que no están en B como posibles abducciones explicativas. No sirven las de B puesto que todas ellas satisfacen a $\neg\Theta$ o a ϕ, con lo que no son explicativas.

A cada una de las δ-cláusulas obtenidas habrá que aplicar skolemización inversa para obtener las δ-explicaciones, y a su vez —para asegurar la corrección del proceso— verificar para cada δ-explicación si es realmente una abducción explicativa. Ello se realiza comprobando mediante δ-resolución —o por otro método de decisión— que para cada δ-explicación α no se verifican ni $\Theta \models \neg\alpha$ ni $\alpha \models \phi$. Si se elige la δ-resolución como método de decisión, esto supone verificar que no se

obtiene $\square$ en ninguna de las dos búsquedas que hay que hacer. Pero de nuevo, la indecidibilidad evita que ambas condiciones puedan comprobarse por lo general.

El proceso anterior se ha definido contando con que los procesos de δ-resolución se saturaban. Sin embargo, en ciertas ocasiones, como hemos explicado, puede que esto no ocurra. Entonces debe elegirse algún otro criterio para detener las búsquedas y pasar a la siguiente etapa del proceso abductivo. Si estamos ante una implementación, tal criterio puede ser el número máximo de δ-cláusulas generadas, o bien un límite en la memoria de trabajo, etc.

A pesar de las limitaciones comentadas, existen muchos problemas abductivos en primer orden que pueden resolverse con éxito mediante δ-resolución, como veremos en la siguiente sección.

7.3. Implementación

En esta sección presentamos la implementación que hemos realizado del cálculo abductivo de δ-resolución, y que como la versión proposicional se trata de una adaptación del demostrador Otter. Comentaremos simplemente sus características más importantes, así como las que difieren de las implementaciones proposicionales anteriores. El programa completo, con el código comentado, puede encontrarse en el apéndice D, página 301.

Para representar las variables libres de las δ-cláusulas se han usado variables de Prolog. Por tanto, es posible usar la unificación de Prolog para comprobar si dos literales unifican; esto se hace, por ejemplo, al aplicar la regla de δ-resolución. Sin embargo, para asegurar la corrección, ya que en el lenguaje que usamos aparecen functores, es preciso que la unificación sea correcta[9], por lo que usamos `unify_with_occurs_check/2`. Haciendo uso de este predicado, veamos cómo queda la definición de `d_resolvente(+C1,+C2,-C3)` que se verifica si `C3` es un δ-resolvente de las δ-cláusulas `C1` y `C2`:

[9]Para más detalles sobre el problema de la unificación correcta de Prolog véase el apéndice A, así como las referencias que allí proporcionamos.

```
d_resolvente(N1*_*CA,N2*_*CB,_*[N1,N2]*Resolvente) :-
        copy_term(CA,C1),                       % 1
        copy_term(CB,C2),                       % 2
        select(L1,C1,R1),                       % 3
        ( (L1 = -Neg; -L1 = Neg) ->             % 4
            select(L2,C2,R2),                   % 5
            unify_with_occurs_check(L2,Neg),    % 6
            append(R1,R2,Resolvente)).          % 7
```

Las dos llamadas que se hacen a `copy_term/2` en las líneas 1 y 2 sirven para renombrar las variables libres que aparecen en `CA` y `CB`, conjuntos de literales de las δ-cláusulas a las que se aplica la regla de δ-resolución, y así asegurar que sus conjuntos de variables sean disjuntos. Se obtienen, pues, `C1` y `C2`. A continuación —línea 3— se toma un literal, `L1`, de `C1`, y se deja como resto `R1`. En la línea 4 se recoge en `Neg` el literal complementario a `L1`. En la línea 5 se toma un literal `L2` de `C2`, y se deja el resto R2. La línea 6 es la que comprueba, mediante unificación correcta, si `L2` unifica con `Neg` —complementario de `L1`—, es decir, si `L1` y `L2` pueden resolverse. Además, de ser posible, aplica una unificación de máxima generalidad entre ambos literales. Por último, la línea 7 se encarga de reunir en `Resolvente` todos los literales de `R1` y R2, cuyas variables también han sido afectadas por el unificador que se aplicó anteriormente.

Como en las implementaciones proposicionales, también ahora se eliminarán las δ-cláusulas subsumidas. Sin embargo, la definición de subsunción en primer orden difiere de la que dimos para lógica proposicional. Como indicamos más arriba, la δ-cláusula Σ_1 está subsumida por Σ_2 si existe una sustitución σ tal que $\Sigma_2\sigma$ es un subconjunto de Σ_1. Para comprobar esto, no basta con aplicar simplemente unificación correcta entre los literales de ambas δ-cláusulas. En primer lugar, la sustitución σ indicada debe aplicarse sólo a la δ-cláusula Σ_2, por lo que no sirve cualquier unificador entre literales de ambas δ-cláusulas. En segundo lugar, la δ-cláusula Σ_2 no debe quedar finalmente afectada por σ.

Para afrontar estos dos problemas hemos definido, en primer lugar, `unifica_con(+L1, +L2)`, que se verifica si existe una sustitución que transforma el literal `L1` en `L2`. Además, aplica dicha sustitución a `L1`. El literal `L2` quedará afec-

tado, a lo sumo, por un renombramiento de sus variables libres. Su definición es la siguiente:

```
unifica_con(L1,L2) :-
        free_variables(L2,Free1),              % 1
        unify_with_occurs_check(L1,L2),        % 2
        free_variables(L2,Free2),              % 3
        unify_with_occurs_check(Free1,Free2). % 4
```

La clave de la definición anterior está en que antes de la unificación que se hace en la línea 2 entre `L1` y `L2` se toman en `Free1` las variables libres de `L2`. Además, tras la unificación, vuelven a recogerse —línea 3— las variables libres de `L2` en `Free2`. Finalmente, se comprueba que ambos conjuntos de variables libres, unifican entre sí. Veamos que esta definición evita que en la unificación que se lleva a cabo en la línea 2 unifique alguna variable de `L2` con algún otro término, más allá del renombramiento de variables, que resulta inofensivo. Informalmente los dos casos que podrían ocurrir son:

- Que una variable de `L2` unifique en la línea 2 con un término sin variables. Entonces, en `Free2` habrá un elemento menos que en `Free1` y no sería posible la unificación de la línea 4.
- Que una variable de `L2`, por ejemplo `X`, unifique en la línea 2 con un término con variables, como `f(Y)`. Entonces, en `Free1`, que se incluyó `X`, tras la unificación de 2 estará `f(Y)`, mientras que en `Free2` aparecerá `Y`. Sin embargo, entre `Y` y `f(Y)` no puede haber una unificación correcta.

Mediante `unifica_con/2` podemos definir `subsume(+C1, +C2)`, que se verifica si la δ-cláusula `C1` subsume a `C2`:

```
subsume(_*_*E1,_*_*E2):-          %  1
        copy_term(E1,E3),         %  2
        copy_term(E2,E4),         %  3
        subsume_aux(E3,E4),       %  4
        unifica_con(E4,E2).       %  5
subsume_aux([],_).                %  6
```

```
subsume_aux([E1|Resto],Cl) :-       %  7
        member(Lit,Cl),             %  8
        unifica_con(E1,Lit),        %  9
        subsume_aux(Resto,Cl).      % 10
```

Las líneas 2 y 3 emplean `copy_term/2` para hacer copias de `E1` y `E2` con nuevos nombres de variables libres, y logran así que sus conjuntos de variables libres sean disjuntos y que las variables libres de las δ-cláusulas originales puedan ser instanciadas. En la línea 4 se comprueba si para cada literal de `E3` hay una sustitución que lo convierte en un literal de `E4`. Para ello se emplea `subsume_aux/2`, que como puede observarse acude al predicado `unifica_con/2`. Un problema que puede ocurrir en este paso es que alguna de las variables de `E3` que haya unificado con una variable de `E4` vuelva a unificar con otro término. En tal caso, el resultado sería que se ha producido una sustitución en las variables de `E4`. Para verificar que esto no ocurre, la línea 5 comprueba que `E4` pueda unificar con `E2`, lo que resultaría imposible si en `E4` se hubieran producido sustituciones que fueran más allá del simple renombramiento de variables libres.

En la página 301 se encuentran los demás detalles de implementación del programa. Ahora, veamos algunos ejemplos de su funcionamiento. En primer lugar, es posible detectar mediante δ-resolución la validez universal de una fórmula de primer orden. Para ello podemos definir el siguiente predicado:

```
prueba(Fml) :-
      inicia,                          % 1
      d_clausulas(Fml, Cs),            % 2
      anotado(Cs,CsAn),                % 3
      d_resol_anotada([],CsAn,[],Us),  % 4
      escribe_prueba(Us).              % 5
```

La línea 1 inicia los valores de las variables globales que se administran los contadores de δ-cláusulas retenidas y analizadas. A continuación, en la línea 2 se obtiene el conjunto de δ-cláusulas correspondiente a la fórmula `Fml`, y en 3 se

anotan dichas cláusulas. La línea 4 comienza la búsqueda de $\square$ y la 5, de haberse encontrado $\square$, escribe su prueba. Veamos un sencillo ejemplo[10]:

```
?- prueba(all(X, p(X) v q(X)) & -q(a) => p(a)).
     1 [] [-p(_G168), -q(_G168)]
     2 [] [q(a)]
     3 [] [p(a)]
   d-cláusula actual #1: 1 [] [-p(_G168), -q(_G168)]
   d-cláusula actual #2: 2 [] [q(a)]
     0 [2, 1] [-p(a)]
    ** RETENIDA: 4 [2, 1, 3] []
   -----> D-CLÁUSULA VACÍA: 4 [2, 1, 3] []
  -------- DEMOSTRACIÓN --------
     1 [] [-p(_G811), -q(_G811)]
     2 [] [q(a)]
     3 [] [p(a)]
     4 [2, 1, 3] []
  --- fin de la demostración ---
```

El proceso abductivo está implementado por el predicado `abduce(+Teo, +Obs)`. Veamos un ejemplo de su uso, considerando que la teoría es la fórmula $\forall x(Px \vee Qx \rightarrow Rx)$ y la observación Ra. Veamos el comportamiento del programa:

```
?- abduce(all(X, p(X) v q(X) => r(X)), r(a)).
PROBLEMA ABDUCTIVO:
   Teoría:       all(_G168, p(_G168)v q(_G168)=>r(_G168))
   Observación:  r(a)
1. ANÁLISIS DE LA TEORÍA:
   d-cláusulas de la teoría:
                    [-r(_G168), p(_G168)]
                    [-r(_G168), q(_G168)]
   Soporte de la teoría:
     1 [] [-r(_G168), p(_G168)]
     2 [] [-r(_G168), q(_G168)]
```

[10]En la implementación se ha usado la notación `all(X, F)` para cuantificar universalmente la variable `X` sobre la fórmula `F` y `ex(X, F)` para cuantificarla existencialmente. Además, se han usado letras minúsculas para representar tanto functores como predicados, con los argumentos a continuación, encerrados entre paréntesis.

```
   d-cláusula actual #1: 1 [] [-r(_G557), p(_G557)]
   d-cláusula actual #2: 2 [] [-r(_G708), q(_G708)]
   Último usable de la teoría:
     1 [] [-r(_G557), p(_G557)]
     2 [] [-r(_G708), q(_G708)]
2. ANÁLISIS DE LA OBSERVACIÓN:
   d-cláusulas de la observación:
                    [r(a)]
   Soporte de la observación:
     3 [] [r(a)]
   d-cláusula actual #3: 3 [] [r(a)]
   Último usable de la observación:
     3 [] [r(a)]
3. BÚSQUEDA DE REFUTACIÓN:
   *** NO SE ENCUENTRA REFUTACIÓN ***
   Soporte para búsqueda abductiva:
     1 [] [-r(_G557), p(_G557)]
     2 [] [-r(_G708), q(_G708)]
     3 [] [r(a)]
4. BÚSQUEDA DE EXPLICACIONES:
   d-cláusula actual #4: 3 [] [r(a)]
   d-cláusula actual #5: 2 [] [-r(_G1460), q(_G1460)]
     0 [2, 3] [q(a)]
    ** RETENIDA: 4 [2, 3] [q(a)]
     4 subsume a 2
   d-cláusula actual #6: 4 [2, 3] [q(a)]
   d-cláusula actual #7: 1 [] [-r(_G1966), p(_G1966)]
     0 [1, 3] [p(a)]
    ** RETENIDA: 5 [1, 3] [p(a)]
     5 subsume a 1
   d-cláusula actual #8: 5 [1, 3] [p(a)]
     Pueden ser abducciones explicativas:
     (comprobar tras deseskolemizar)
     4 [2, 3] [q(a)]
     5 [1, 3] [p(a)]
RECURSOS EMPLEADOS:
   Tiempo (milisegundos):  0
   Espacio (bytes):        8808
   Inferencias:            1526
```

Al final, el programa nos devuelve `[q(a)]` y `[p(a)]` como posibles abducciones explicativas, con la advertencia de comprobarlo tras *deseskolemizar*, es decir, tras la *skolemización inversa*. Como ni `q(a)` ni `p(a)` contienen términos de Skolem ni variables libres, las δ-explicaciones finales son `q(a)` y `p(a)`, que son realmente abducciones explicativas.

Por mostrar un ejemplo algo más complejo, consideremos el caso en que la teoría es $\forall x((\exists y Rxy) \rightarrow Fx)$ y la observación Fa. Veamos lo que devuelve el programa. Por brevedad, sólo mostramos la entrada y las explicaciones que el programa devuelve como salida:

```
?- abduce(all(X, ex(Y, r(X,Y)) => f(X)), f(a)).
     Pueden ser abducciones explicativas:
     (comprobar tras deseskolemizar)
     3 [1, 2] [r(a, _G1513)]
```

La explicación que encuentra es `[r(a, _G1513)]`, siendo `_G1513` una variable libre. Si sometemos este literal a skolemización inversa, obtenemos $\exists z Raz$, y tras realizar las comprobaciones oportunas, a saber, $\exists z Raz \not\models Fa$ y $\forall x((\exists y Rxy) \rightarrow Fx), \exists z Raz \not\models \bot$, concluimos que se trata de una abducción explicativa.

Capítulo 8

Consideraciones finales

Con la definición del cálculo de δ-resolución hemos logrado uno de los objetivos principales que nos propusimos al comienzo de este trabajo: crear un *cálculo abductivo* que proceda de forma *directa* y no haciendo un uso *inverso* de cálculos deductivos. Como hemos visto, el cálculo de δ-resolución cuenta, en su versión más simple, con una sola regla, la de δ-resolución —introducida en la definición 4.4—, que es claramente abductiva, pues, dado un conjunto de δ-cláusulas A, se verifica que para cada δ-cláusula Σ, si $A \vdash_\delta \Sigma$ entonces $\Sigma \vDash A$ —teorema 4.6—. Además, haciendo uso de la operación de saturación por δ-resolución —definición 4.18— hemos obtenido el resultado que se expresa en el corolario 5.10,

$$\boxed{\mathcal{A}b\delta(\Theta, \phi) = (N^\delta_\Theta \cup O^\delta)^\delta - (N^\delta_\Theta \cup O^\delta)}$$

con el que hemos desarrollado, exclusivamente a partir de operaciones de δ-resolución, el procedimiento abductivo de la definición 5.23, que, dado un problema abductivo $\langle \Theta, \phi \rangle$, devuelve el conjunto de sus abducciones explicativas minimales. En capítulos posteriores se ha refinado en varios sentidos este proceso abductivo, con una optimización que hace más eficiente la obtención de formas δ-clausales, pues emplea tablas semánticas para ello, y otra que modifica el proceso cuando se trata de explicar literales. Además, hemos explorado su extensión a primer orden.

El punto de partida en nuestra investigación fue el ensayo de una versión dual al cálculo de resolución, que ha hecho posibles todos los resultados que

acabamos de mencionar. Hasta ahora, la resolución siempre ha estado ligada a la búsqueda de contradicciones, y se ha empleado como un método muy eficiente de demostración automática de teoremas. También se habían explorado sus posibilidades abductivas, pero siempre a través de procedimientos *inversos*. Sin embargo, el uso de la versión dual, la δ-resolución, nos ha proporcionado ventajas en varios sentidos. En primer lugar, hemos obtenido resultados muy significativos abductivamente, como el citado corolario 5.10 o los procesos abductivos que se han definido. Las versiones duales de estos resultados, que sería posible obtener usando el cálculo habitual de resolución, no tienen ni el mismo valor abductivo ni ningún valor deductivo equiparable. También la integración con el método de tablas semánticas, que ha resultado computacionalmente muy ventajosa, es más sencilla que con el cálculo de resolución habitual. Por último, nos ha servido para explorar las relaciones que existen entre la deducción y la abducción. El propio Peirce se refiere en ocasiones a la abducción como *retroducción*, con lo que se sugiere cierta dualidad con respecto a la deducción. Con la δ-resolución, esta dualidad se pone de manifiesto, y se hace posible situarla como el objeto de una investigación formal profunda.

Otro de los objetivos originales era tratar de superar las dicotomías habituales entre *generación* y *selección* y entre *producto* y *proceso*. En el proceso abductivo que hemos mostrado en la definición 5.23 encontramos que la *generación* y la *selección* se llevan a cabo mediante el mismo tipo de operaciones, esencialmente a través de la saturación por δ-resolución. En esto se diferencia la δ-resolución de otros sistemas abductivos, que llevan a cabo la *generación* mediante ciertos cálculos —tablas semánticas, o incluso resolución— y para la *selección* —donde se determinan cuáles, de entre las posibles abducciones generadas, son válidas— emplean otros sistemas, habitualmente operaciones conjuntistas o combinatorias. Por otra parte, en la polémica sobre si la abducción es ante todo un *producto* —lo importante sería entonces que las fórmulas generadas cumpliesen ciertas condiciones, y no cómo se han obtenido— o más bien un *proceso* —en el que importa que el proceder sea abductivo, de igual forma que el proceder de la regla de *modus ponens*, por ejemplo, es deductivo— la δ-resolución arroja también cierta luz. De hecho, no sólo los productos son buenos, como demuestran los análisis formales que se han presentado, en los que se muestra cómo obtener todas y sólo las abducciones explicativas minimales para cada problema abductivo; además, el propio

proceder del cálculo de δ-resolución es abductivo, como ya hemos comentado, pues la regla de δ-resolución —definición 4.4—,

$$\frac{\Sigma_1 \cup \{\lambda\} \qquad \Sigma_2 \cup \{\neg\lambda\}}{\Sigma_1 \cup \Sigma_2}$$

no es más que la generación de una nueva explicación, $\Sigma_1 \cup \Sigma_2$, a partir de otras dos: $\Sigma_1 \cup \{\lambda\}$ y $\Sigma_2 \cup \{\neg\lambda\}$. En la nueva explicación no aparecen ni el literal λ ni su complementario $\neg\lambda$, con lo que en muchas ocasiones podrá decirse que se trata de una explicación mejor que las anteriores, en el sentido de más simple. De este modo, que recuerda el principio de la navaja de Ockham, transcurre todo el proceso abductivo mediante δ-resolución.

En la sección 6.4 realizamos un análisis filosófico de los cuatro pasos del proceso abductivo mediante δ-resolución, y subrayamos el paralelismo de cada uno de ellos con ideas provenientes de la Filosofía de la Ciencia. Ahora, queremos evaluar el carácter abductivo de la δ-resolución de un modo más concreto, a la luz de las cuatro tesis de Kapitan-Hintikka [Hin98], que en la sección 1.3 propusimos como los cuatro rasgos fundamentales del razonamiento abductivo, y que por tanto debe recoger un sistema formal que pretenda captar este tipo de razonamiento. Recordemos las cuatro tesis:

Tesis inferencial. La abducción es, o incluye, un *proceso* inferencial.

Tesis de objetivo. El propósito de la abducción científica es doble: en primer lugar generar nuevas hipótesis y posteriormente seleccionar las mejores para su análisis.

Tesis de comprensión. La abducción científica incluye todas las operaciones por las que se engendran las teorías.

Tesis de autonomía. La abducción es un tipo de razonamiento irreductible tanto a la deducción como a la inducción.

Comencemos por la *tesis inferencial*. Resulta innegable que el cálculo de δ-resolución es un *proceso inferencial*, además caracterizable formalmente, y del que pueden demostrarse, como hemos visto, importantes propiedades metalógicas.

En cuanto a la *tesis de objetivo*, que otorga a la abducción la doble tarea de *generar nuevas hipótesis* y de *seleccionar las mejores*, también es satisfecha por el cálculo de δ-resolución, pues como hemos indicado un poco más arriba, ambas tareas pueden llevarse a cabo dentro de un mismo proceso abductivo, y con una misma herramienta, como es la saturación por δ-resolución. Respecto a la *tesis de comprensión*, si la δ-resolución recoge o no todas las operaciones por las que se engendran las teorías científicas, se trata de uno de los objetivos —y sin duda el más ambicioso— que quedan abiertos tras la realización de este trabajo. Si no para modelar la formulación de nuevas teorías científicas, al menos sí confiamos en que la δ-resolución muestre su utilidad para afrontar ciertos problemas abductivos reales que surgen en ámbitos tan diversos como son el Procesamiento del Lenguaje Natural, la asimilación de conocimientos, o problemas propios de la Inteligencia Artificial como la planificación o el razonamiento por defecto. Finalmente, la *tesis de autonomía* se ve sin duda satisfecha por el cálculo de δ-resolución, ya que el motivo para formularlo fue el poder abducir de modo directo, sin usar sistemas deductivos. Aunque es dual a un cálculo deductivo —la resolución— la δ-resolución no es ni deductiva ni inductiva, sino propiamente abductiva.

Un logro computacional de la δ-resolución, con respecto a los sistemas de abducción basados en tablas semánticas —como son los de A. Aliseda [Ali97] o M. Cialdea y F. Pirri [CP93]—, es que la eficiencia se ha mejorado notablemente, sobre todo en los problemas más complejos de entre los incluidos en el apéndice C. Además, como hemos comentado en anteriores capítulos, aunque pudiera realizarse una adaptación de los sistemas de abducción basados en tablas semánticas para explicar fórmulas más complejas —pagando el correspondiente coste computacional, elevado en este caso—, en principio están definidos para explicar literales. Sin embargo, en el cálculo de δ-resolución es posible explicar observaciones sintácticamente más complejas, sin que ello redunde en un drástico aumento de complejidad. Es más, ni siquiera habría que modificar los procesos abductivos definidos. Eso sí, cuando la observación es un literal, pueden modificarse y ganar considerablemente en eficiencia.

Una de las puertas que quedan abiertas tras la realización de este trabajo se ha comentado en el capítulo anterior, a propósito de las repercusiones abductivas del problema de la decisión en lógica de primer orden. Como se vio entonces, es

posible adaptar el proceso abductivo para lógica de primer orden, pero a riesgo de perder corrección —por ejemplo, puede que se generen explicaciones no válidas por no ser consistentes con la teoría— y completud —algunas explicaciones puede que no aparezcan—. Esto es debido a que la saturación mediante δ-resolución en lógica de primer orden no es una operación cuyo final quede siempre asegurado en un número finito de pasos. De todos modos, aunque pueda perderse en propiedades metalógicas —corrección y completud— se gana en similitud con las situaciones reales en que aparecen problemas abductivos —desde el razonamiento de sentido común hasta la práctica científica—, donde no siempre resulta posible demostrar la corrección de las explicaciones generadas. En esta línea, una posible extensión del cálculo de δ-resolución en primer orden sería la adaptación de las soluciones parciales de decisión que se han aportado, por diferentes autores, en ámbitos deductivos. Una de las más prometedoras es la que se deriva de los estudios de las posibilidades decisorias de las tablas semánticas por parte de G. Boolos [Boo84], E. Díaz [Dia93] y N. Peltier [Pel03]. A. Nepomuceno y A. Aliseda [Nep02, AN] han estudiado las posibilidades abductivas de la modificación del método de las tablas semánticas que proponen los autores citados. Estos resultados pueden integrarse en el cálculo de δ-resolución en primer orden por medio de una modificación en el proceso abductivo, similar a la que presentamos en la sección 6.2.1 de este trabajo. En lógica de primer orden también resultaría posible obtener las formas δ-clausales por medio de tablas semánticas, que pueden realizarse según la modificación citada. Entonces el proceso abductivo resultaría decidible en dominios de cardinalidad acotada.

También resultaría interesante el estudio de las posibilidades de aplicación del cálculo de δ-resolución a lógicas no clásicas. En lógica clásica, la principal aportación de la δ-resolución es que constituye un método abductivo *directo*, y no *inverso*. De acuerdo con lo que comentamos al comienzo del capítulo 4, los sistemas *inversos* se basan en (4.2):

$$\Theta, \alpha, \neg\phi \vDash \bot$$

mientras que la δ-resolución acude a (4.4):

$$\alpha \vDash \Theta \rightarrow \phi$$

Pese a que ambas relaciones son equivalentes en lógica clásica, no ocurre lo mismo en las lógicas paraconsistentes, por ejemplo, o en otros sistemas no clásicos. Sin embargo, para aquellas lógicas que cuenten con pruebas constructivas de completud puede resultar sumamente interesante la definición de sistemas abductivos *directos*.

Otro aspecto donde aún puede ahondarse más es en la caracterización cognitiva de los procesos abductivos. También aquí encuentra utilidad el carácter *directo* de la abducción mediante δ-resolución. Consideramos poco creíble que, para explicar cierto hecho, la mente opere a partir de la negación del dato observado y busque una contradicción, tal como expresa (4.2). No se suele suponer la negación de algo evidente, la observación, a menudo más fiable que las teorías. Más bien son éstas las que podemos estar dispuestos a poner en duda.

Entre las teorías que explican las capacidades humanas de razonamiento hemos citado dos: los *Modelos Mentales*, de P.N. Johnson-Laird [JL83], y la *Lógica Mental*, encabezada por M.D.S. Braine. En cierto sentido, se puede considerar la δ-resolución como conciliadora de ambos enfoques. Para Johnson-Laird, un modelo mental se representa por los hechos que se verifican en él, al modo de una conjunción de literales. Por tanto, no sería raro comprender una δ-cláusula como la representación de un modelo mental. En tal caso, la regla de δ-resolución puede entenderse, tal vez, como una regla mental que opera con modelos mentales —las δ-cláusulas—. Esta comprensión cognitiva del cálculo de δ-resolución soluciona, por un lado, el problema de la proliferación de modelos mentales que aparece en la teoría de Johnson-Laird, donde debe considerarse un modelo diferente por cada posible estado del mundo. Además, en cuanto a la lógica mental, es posible regular la aplicación de la regla de δ-resolución de una forma más sencilla que las reglas propuestas por Braine.

Apéndice A

Introducción a Prolog

En este apéndice presentamos una pequeña introducción a Prolog, fundamentalmente basada en el manual de Flach [Fla94]. Más que un estudio en profundidad, lo que pretendemos es mostrar los rasgos más importantes de este lenguaje de programación. Para descripciones más detalladas se puede acudir a [DEDC96, BBS01, NM95], por ejemplo.

El nombre de Prolog hace referencia a la *programación lógica* —de hecho, es una abreviatura de *PROgramming in LOGic*—, un paradigma de programación que nace en los años 70 de la mano de Colmerauer y Kowalski, al sistematizar trabajos anteriores como los de Boyer y Moore, quienes desarrollaron algoritmos de unificación según las ideas de Robinson [Rob65].

En la programación lógica, los programas no se conciben como una descripción detallada de los algoritmos que la máquina tiene que seguir para solucionar cierto problema, como hacen los lenguajes tradicionales como C o Pascal, comúnmente llamados *imperativos* por esta razón. Aquí, la estructura del programa es la de una teoría, de forma que su ejecución consiste en la búsqueda de una prueba para cierto teorema que se propone. Por ello, de la programación lógica se dice que es *declarativa*. Así pues, si en la programación *imperativa* un programa especifica procedimentalmente *cómo* debe resolverse cierto problema, en la

programación *declarativa* el programa especifica *qué* propiedades debe tener la solución del problema.

A.1. Estructura de un programa lógico

Como ya hemos apuntado, un programa lógico consiste en una teoría, una descripción del mundo a partir de *hechos* y *reglas*. Para llevar esto a cabo se usan *cláusulas de Horn*, que son fórmulas de tipo

$$\lambda_1 \wedge \ldots \wedge \lambda_n \rightarrow \gamma \tag{A.1}$$

para $n \geq 0$, tal que cada λ_i, $1 \leq i \leq n$, es una fórmula atómica, así como γ. A los λ_i se les llama *literales negativos*, y a γ, *literal positivo*, ya que (A.1) es equivalente a

$$\neg\lambda_1 \vee \ldots \vee \neg\lambda_n \vee \gamma \tag{A.2}$$

A las cláusulas del tipo (A.1) en que $n = 0$ las llamaremos *hechos*, ya que equivalen a γ. Cuando $n > 0$ se las llama *reglas*, pues condicionan γ a que se verifiquen todos los literales λ_i, $1 \leq i \leq n$.

La representación en Prolog para hechos y reglas se realiza como en el ejemplo siguiente[1]:

```
l1.
l2 :- l1.
l3 :- l2, l4.
```

Las tres líneas de código anteriores constituyen lo que se conoce como *base de conocimientos* o descripción del mundo. La primera línea corresponde a un hecho, `l1`, y las otras dos a reglas. En cada regla, el literal que está a la izquierda

[1] A lo largo de este trabajo, reservamos una tipografía de máquina de escribir para los fragmentos de código Prolog. Además, cuando en un término Prolog ocurra un guión, se usará siempre el guión bajo "_" y se reservará el guión intermedio "-" para partir sílabas al final de línea.

de "`:-`" es el literal positivo, o *cabeza* de la regla, y los literales que aparecen a la derecha son los literales negativos, o *cuerpo* de la regla.

Como ya se ha comentado, la ejecución de un programa consiste en la demostración de teoremas dentro de una base de conocimientos. Así, para comprobar si el literal `lit` puede probarse en cierta base de conocimientos, se lanza a Prolog la pregunta "`?- lit.`", lo que producirá una respuesta: positiva si el objetivo —en este caso el literal `lit`— logra probarse y negativa en otro caso. En la base de conocimientos que propusimos en el ejemplo anterior, el objetivo "`?- l1.`" tendrá éxito, ya que "`l1`" pertenece a la base de conocimientos; también "`?- l2.`", puesto que la regla "`l2 :- l1.`" representa `l1`→`l2`, que junto con `l1` permite probar `l2`. Sin embargo, no puede probarse `l3`:

```
?- l1.
   Yes
?- l2.
   Yes
?- l3.
   No
```

Tanto las cláusulas de un programa como las preguntas pueden contener functores y variables. Veamos la siguiente base de conocimientos:

```
es_padre_de(juan,ana).                                    % 1
es_padre_de(sergio,maría).                                % 2
es_madre_de(maría,ana).                                   % 3
es_madre_de(carmen,juan).                                 % 4
es_madre_de(luisa,carmen).                                % 5

es_abuelo_de(X,Y) :- es_padre_de(X,Z), es_madre_de(Z,Y). % 6
es_abuelo_de(X,Y) :- es_padre_de(X,Z), es_padre_de(Z,Y). % 7

es_abuela_de(X,Y) :- es_madre_de(X,Z), es_madre_de(Z,Y). % 8
es_abuela_de(X,Y) :- es_madre_de(X,Z), es_padre_de(Z,Y). % 9
```

Como puede apreciarse, los functores se pueden usar para definir predicados tal como `es_padre_de(juan,ana)` que hemos usado para expresar que `juan` es

padre de `ana`[2]. Con las variables, que hemos llamado `X`, `Y`, `Z`, hemos establecido relaciones entre los predicados. Así, con

```
es_abuelo_de(X,Y) :- es_padre_de(X,Z), es_madre_de(Z,Y).
```

expresamos que `X` es abuelo de `Y` si `X` es padre de `Z` y `Z` es madre de `Y`. Como ocurre en el ejemplo anterior, un mismo predicado puede estar definido por varias cláusulas.

De las variables, debemos decir que las que aparecen en distintas cláusulas son independientes, aunque lleven el mismo nombre. Además, distinguimos entre las *variables universales*, que son las que aparecen en el literal positivo de una cláusula, y las existenciales, que son todas las demás. Así, en la cláusula anterior, `X` e `Y` son variables universales, mientras que `Z` es existencial. Esta denominación se puede comprender fácilmente a través de la equivalencia de la cláusula anterior con

$$\forall xyz\,(es\_padre\_de(x,z) \wedge es\_madre\_de(z,y) \rightarrow es\_abuelo\_de(x,y)) \qquad \text{(A.3)}$$

y a su vez con

$$\forall xy\,(\exists z\,(es\_padre\_de(x,z) \wedge es\_madre\_de(z,y)) \rightarrow es\_abuelo\_de(x,y)) \qquad \text{(A.4)}$$

Cuando en una cláusula quiere introducirse una variable sin importar a qué término represente —pues puede representar a cualquiera— se usa la variable anónima, que se escribe como "`_var`", siendo "`var`" cualquier cadena de caracteres.

[2]Obsérvese que escribimos los nombres de átomos y functores comenzando por minúsculas, y los de las variables por mayúsculas. Usamos los términos *functor*, *variable* y *predicado* en el sentido que tienen en Programación Lógica, no como se han empleado en el capítulo 7 en referencia al lenguaje $\mathcal{L}$ de lógica de primer orden. En Prolog, un *predicado* se define mediante un conjunto de cláusulas que comparten el mismo functor en su literal positivo. Así, en la base de conocimientos anterior, el predicado `es_padre_de/2` —el número 2 indica que es un predicado de dos argumentos— está definido por las cláusulas 1 y 3, que son las únicas que tienen tal functor en su literal positivo. Por la misma razón, el predicado `es_abuela_de/2` está definido por las cláusulas 8–9.

Veamos algunos ejemplos de cómo las variables pueden usarse también en las preguntas. Entonces, si la respuesta es afirmativa, viene acompañada de los términos con los que debe *unificar* cada variable.

```
?- es_padre_de(X,maría).
   X = sergio
   Yes
?- es_padre_de(X,luisa).
   No
?- es_abuelo_de(X,Y).
   X = sergio
   Y = ana
   Yes
?- es_abuela_de(X,Y).
   X = luisa
   Y = juan ;
   X = carmen
   Y = ana ;
   No
```

En el último ejemplo, en que se han pedido todos los pares $\langle x, y\rangle$ en que x es abuela de y, cuando el intérprete devolvió la primera respuesta $\langle luisa, juan\rangle$ le hemos pedido otra más usando ";", signo que tiene en Prolog un valor disyuntivo, y nos ha devuelto $\langle carmen, ana\rangle$. Al pedirle un tercer par, la respuesta ha sido negativa, pues en la base de conocimientos no existe ninguna otra solución a la pregunta propuesta.

A.2. La resolución SLD

Los programas Prolog son evaluados mediante un *intérprete*, un programa encargado de demostrar los objetivos que se le lanzan al formularle preguntas. En concreto, el intérprete usado en las implementaciones de este trabajo es SWI-Prolog, versión 5.4.3. El motor de prueba que emplean los intérpretes consiste en un procedimiento de demostración por refutación mediante resolución, conocido como *resolución SLD*: S por la regla de *selección*, L por usar resolución *lineal* —lo

cual se refiere a la forma de los árboles de prueba obtenidos— y D por realizarse sobre cláusulas *definidas*, que es como se llama a las cláusulas que forman parte de los programas Prolog.

También se usa la *unificación*, que es el mecanismo por el que se reemplazan las variables por términos, mediante una sustitución conocida como *unificador*. En concreto, siempre se buscará el *unificador más general*.

Veamos un ejemplo para mostrar cómo se combinan la resolución SLD y la unificación. Partiremos de la base de conocimientos que definimos en la página 247, y usaremos la numeración que acompaña a cada cláusula para referirnos a ella. Consideremos el objetivo "`?- es_abuela_de(Abuela,Nieto).`", cuyas respuestas ya conocemos. Como en toda prueba por refutación mediante resolución, se niega lo que quiere probarse y se aplica resolución hasta llegar a la cláusula vacía $\square$. La negación de nuestro objetivo se representa como "`:- es_abuela_de(Abuela, Nieto).`".

La siguiente tarea es elegir la cláusula con la que va a resolverse este objetivo. La regla de selección elige entonces una cláusula cuyo literal positivo unifique con `es_abuela_de(Abuela,Nieto)`. En este caso, hay dos cláusulas, 8 y 9, que cumplen este requisito. Esta situación se conoce como *punto de elección*. Entonces la regla de selección elige la primera de ellas, la número 8, pero guarda un registro de las demás alternativas de modo que si la prueba fallara con la primea opción —o bien si se piden más soluciones— se continúa por las demás. Al proceso de volver sobre las decisiones de la regla de selección para seguir otras alternativas se le llama *backtracking*, y es otro de los rasgos fundamentales de Prolog.

Volviendo a nuestro ejemplo, tras usar la cláusula 8, el objetivo que tenemos es

```
:- es_madre_de(Abuela,Z), es_madre_de(Z,Nieto).
```

En esta situación, el objetivo se compone de dos literales. La forma en que se abordan estos objetivos es ordenadamente, comenzando por el primero. Se trata de una *pila de objetivos*, de la que siempre se añaden y se extraen los objetivos

por un mismo extremo. Ahora vuelve a producirse un nuevo *punto de elección*, pues el primero de los objetivos puede resolverse con las cláusulas 3, 4 y 5. Al resolverse con 3, se produce el nuevo objetivo "`:- es_madre_de(ana,Nieto).`", unificando `Abuela` con `maría`. Pero este nuevo objetivo no puede resolverse con ninguna cláusula, por lo que se produce una *rama muerta* del árbol de prueba. Entonces por *backtracking* se vuelve al último punto de elección, y se usa esta vez la cláusula 4, pero también se produce otra rama muerta. Sin embargo, al usar la cláusula 5, se unifica `Abuela` con `luisa` y el nuevo objetivo resulta ser "`:- es_madre_de(carmen,Nieto).`". Este nuevo objetivo unifica con la cláusula 4, haciendo `Nieto` igual a `juan`, y se obtiene la cláusula vacía. Por tanto, se ha alcanzado la primera solución a la pregunta planteada, de forma que `Abuela = luisa` y `Nieto = juan`.

Esta primera respuesta es la misma que obtuvimos del intérprete en la sección anterior. En aquella ocasión pedimos una segunda solución. Entonces lo que se hace es volver sobre el último punto de elección del que aún queden alternativas por seguir. En nuestro caso se trata del primer punto de elección de la prueba, y la única alternativa abierta es usar la cláusula 9, como ya comentamos, que produce el objetivo

```
:- es_madre_de(Abuela,Z), es_padre_de(Z,Nieto).
```

Volvemos a tener un nuevo punto de elección, pues el primer objetivo se puede resolver con las cláusulas 3, 4 y 5. Si se resuelve con 3, el nuevo objetivo resulta ser "`:- es_padre_de(ana,Nieto).`", que no resuelve con ninguna cláusula, por lo que produce una nueva rama muerta. Pero si usamos 4 el objetivo, tras unificar `Abuela` con `carmen`, se convierte en "`:- es_padre_de(juan,Nieto).`". Este objetivo puede resolverse sólo con la cláusula 1, y se obtiene la cláusula vacía tras unificar `Nieto` con `ana`, con lo que se alcanza así la segunda solución del problema.

Si pedimos una tercera solución, tal como hicimos en la sección anterior, se vuelve al último punto de elección, del que aún queda por seguirse una rama, la producida al usar la cláusula 5. Entonces, el objetivo que resulta es "`:- es_padre_de(carmen,Nieto).`", tras unificar `Abuela` con `luisa`. Pero este

objetivo *falla*, por no poderse unificar con ningún literal positivo del programa. Entonces se busca otro punto de elección del que quede alguna alternativa sin explorar, pero ya no hay ninguno, por lo que la búsqueda falla —para la tercera solución—, lo que se corresponde con la respuesta negativa que se obtuvo en la sección anterior.

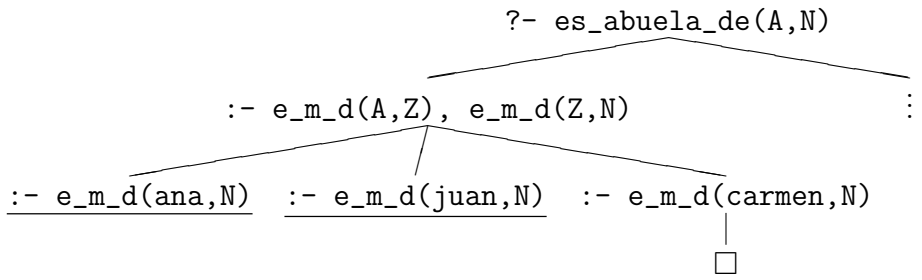

Figura A.1: Ejemplo de un árbol de resolución SLD

En la figura A.1 hemos representado parte del árbol de resolución SLD correspondiente al objetivo que acabamos de probar. Por reducir espacio, sólo hemos representado la parte correspondiente a la primera opción del primer punto de elección. Además, se han omitido las unificaciones y se han usado las siguientes abreviaturas: `A` por `Abuela`, `N` por `Nieto`, y `e_m_d` por `es_madre_de`. Las ramas muertas se indican subrayando el último objetivo que se alcanza, para el que ya no es posible encontrar una cláusula con la que pueda resolverse. Las ramas que tienen éxito se marcan con la cláusula vacía.

A.3. Programación en Prolog

A continuación mostramos algunos recursos elementales de programación en Prolog. Para más detalles se puede consultar la ayuda de SWI-Prolog, o bien el estándar [DEDC96].

A.3.1. Disyunción

Como hemos comentado más arriba, el operador ";" tiene un valor disyuntivo en Prolog, de forma que `A;B` se verificará siempre que se verifique `A` o bien `B`. Teniendo esto en cuenta, las cláusulas 6–9 de la base de conocimientos que aparece en la página 247 se pueden reescribir como

```
es_abuelo_de(X,Y) :-
      es_padre_de(X,Z),
      (es_madre_de(Z,Y);
       es_padre_de(Z,Y)).
es_abuela_de(X,Y) :-
      es_madre_de(X,Z),
      (es_madre_de(Z,Y);
       es_padre_de(Z,Y)).
```

Otra opción posible, también usando la disyunción, sería

```
es_antecesor_de(X,Y):-
      es_padre_de(X,Y);
      es_madre_de(X,Y).
es_abuelo_de(X,Y) :-
      es_padre_de(X,Z),
      es_antecesor_de(Z,Y).
es_abuela_de(X,Y) :-
      es_madre_de(X,Z),
      es_antecesor_de(Z,Y).
```

A.3.2. Corte

Al explicar el procedimiento de resolución SLD, vimos que Prolog siempre guarda información sobre los puntos de elección que aparecen en la búsqueda, de forma que si la prueba falla —o bien si se piden más soluciones—, sea posible volver sobre ellos mediante *backtracking*, recorriendo así nuevas ramas del árbol de prueba.

Sin embargo, hay situaciones en que al programador no le interesa conservar los puntos de elección, especialmente cuando se definen predicados que tienen a lo sumo una sola solución válida. Entonces, resulta necesario disponer de algún mecanismo de control que impida, en tales casos, que la búsqueda vuelva sobre puntos de elección inútiles, o de alguna forma no deseados. Esto es lo que Prolog realiza mediante el corte, "`!`", que bien empleado puede mejorar notablemente la eficiencia de los programas. Su efecto, cuando aparece en una cláusula

```
lit_1 :- lit_2, lit_3, !, lit_4.
```

es eliminar —una vez que se alcanza, es decir, probados `lit_2` y `lit_3`— todas las soluciones alternativas para los literales `lit_2` y `lit_3`. Además hace que no se busquen cláusulas alternativas para `lit_1`. Sin embargo, los puntos de elección que aparezcan al tratar de probar `lit_4` no son eliminados por este corte.

Haciendo uso del corte, se construye el condicional "`->`", de forma que "`A -> B`" se evalúa igual que "`A, !, B`", es decir, si tiene éxito la prueba de `A` se eliminan los puntos de elección de la cláusula en que aparece el condicional y se trata de probar `B`. Mediante la disyunción pueden anidarse los condicionales, tal como se hace en la siguiente cláusula:

```
lit_1 :-
    lit_2 -> lit_4;
    lit_3 -> lit_5;
    true  -> lit_6.
```

de forma que si se verifica `lit_2`, habrá que probar `lit_4` y no se evalúan más literales. En caso de que no se verifique `lit_2`, hay que probar `lit_3` y, si tiene éxito, se continúa por `lit_5`. Sólo si no tuvo éxito la prueba de `lit_3` se evalúa `lit_6`. El átomo `true` siempre se evalúa como verdadero por Prolog, así como `fail` siempre como falso. Esta construcción es sumamente útil para definir reglas.

Si bien el uso del corte —y por tanto del condicional— puede mejorar mucho la eficiencia de los programas, podando los árboles de prueba, también puede eliminar soluciones válidas. Por ello, debe usarse siempre con cuidado.

A.3.3. Negación por fallo

También usando el corte se construye la negación por fallo[3], `not/1`, de Prolog, de la siguiente forma

```
not(P):- P, !, fail.
not(P).
```

de modo que para evaluar `not(P)` primero se evalúa `P`. Si la prueba tiene éxito, se cortan los puntos de elección —impidiendo que tenga éxito `not(P)`— y la prueba falla. En caso de que no tenga éxito la prueba de `P`, tiene éxito —gracias a la segunda cláusula— la de `not(P)`.

Por tanto, la negación por fallo tiene éxito si falla la prueba de lo que se niega. Posteriormente, discutiremos las diferencias entre esta negación y la negación lógica. En muchos intérpretes, como es el caso de SWI-Prolog, `not(P)` puede escribirse también como `\+ P`.

El corte, el condicional y la negación por fallo introducen elementos procedimentales que a menudo pueden suponer una pérdida del sentido declarativo de Prolog. Algo que ocurre frecuentemente es que, para que los programas funcionen correctamente, deben imponerse restricciones a los argumentos de ciertos predicados. Estas restricciones pueden indicarse en las especificaciones de los predicados mediante los signos "`-`", "`+`" y "`?`", tal como explicamos a continuación. Por ejemplo, con `pred(-Arg1, +Arg2, ?Arg3)` se indica que para el predicado `pred`, el primer argumento, `Arg1`, debe ser una variable —que probablemente se instanciará durante la prueba, por lo que se trata de un argumento de *salida*—; el segundo, `Arg2`, un término sin variables —siendo, pues, un argumento de *entrada*—; y el tercero, `Arg3`, no tiene ninguna restricción.

La notación anterior es ampliamente usada en los comentarios de los programas. A propósito de los *comentarios*, se denomina así a toda cadena de caracteres que, o bien va desde un signo "`%`" hasta el siguiente final de línea, o se encuentra

[3]Mediante la notación `pred/n`, indicamos que el predicado `pred` tiene `n` argumentos, siendo `n` un número natural.

encerrada entre "/*" y "*/". Tales líneas, que no son leídas por el intérprete, se emplean para anotar los programas.

A.3.4. Aritmética en Prolog

En Prolog, la identidad "+A = +B" tiene éxito si pueden unificarse los términos A y B. Además, "+A == +B" tiene éxito sólo si A y B son el mismo término —sin permitir la unificación de variables— de forma que

```
?- p(a) = p(S).
   S = a
   Yes
?- p(a) == p(S).
   No
?- p(D) == p(S).
   No
?- p(S) == p(S).
   S = _G168
   Yes
?- 5 = 3+2.
   No
```

El término _G168 es la representación interna que SWI-Prolog emplea para la variable S en el tercer ejemplo. El último ejemplo muestra la diferencia que existe entre la unificación, que se realiza con "=", y la evaluación de términos aritméticos. Por ello, no tiene éxito el objetivo "?- 5 = 3+2.", ya que no es posible unificar el término atómico 5 con el compuesto 3+2. Para la evaluación de términos aritméticos, Prolog dispone del predicado is/2, de forma que "?A is +B" se verifica[4] si A es el número correspondiente a la evaluación aritmética de B. Veamos algunos ejemplos:

```
?- 5 is 3+2.
   Yes
```

[4]El predicado is/2 lo podemos escribir entre sus dos argumentos debido a su carácter *infijo*. Más adelante explicaremos de qué se trata.

```
?- X is (3*2)-1.
   X = 5
   Yes
?- X is (10/2)^3.
   X = 125
   Yes
```

A.3.5. Recursividad

Al definir un predicado en Prolog, es posible usar la recursividad, como vemos en el siguiente ejemplo[5].

```
/* factorial(+N,?Fac) se verifica si Fac es el factorial del número
   natural N */
factorial(0,1) :- !.          % Caso base
factorial(N,Fac) :-           % Recursión
        N1 is N-1,
        factorial(N1,F2),
        Fac is N*F2.
```

La primera cláusula se corresponde con el factorial de 0, mientras que la segunda define el factorial de n, para $n > 0$, como el producto de n por el factorial de $n - 1$.

A.3.6. Listas

Es muy frecuente el manejo de listas, que en Prolog se forman mediante una estructura recursiva, y se representan separando sus elementos por comas y ence-

[5]Las dos primeras líneas, que aparecen comentadas entre "`/*`" y "`*/`", constituyen la especificación del predicado `factorial/2`. Como en ellas se indica, el primero de los argumentos de este predicado, `N`, es de *entrada*, y se trata del número para el que se calcula el factorial. El segundo, `Fac`, que en cada llamada podrá ser un número o una variable, es el resultado de calcular el factorial de `N`. Como en Prolog existe bastante libertad en cuanto al orden de los argumentos, conviene aclarar mediante comentarios la especificación de cada predicado que se defina.

rrándolos entre los corchetes "[" y "]", de forma que `[a,b,c]` es la lista compuesta por lo elementos `a`, `b` y `c`, en este orden. Igualmente, `[X,d,Z]` representa cualquier lista de tres elementos que tenga a `d` en segunda posición. Los elementos de una lista sos términos de cualquier tipo, incluido listas. La lista más pequeña es la lista vacía, "`[]`", que no contiene ningún elemento.

Al primer elemento de una lista —de al menos un elemento— se le llama *cabeza* y al resto —que es una lista— *cola*. Mediante `[X|Y]` se representa una lista que tiene como cabeza el elemento `X` y como cola la lista `Y`. Siguiendo esta notación `[Pri,Seg,Ter|Rst]` representa una lista con al menos tres elementos, `Pri`, `Seg` y `Ter`, y de resto `Rst`. Cuando una lista tiene un solo elemento, como por ejemplo `[a]`, dicho elemento es su cabeza, y su cola, la lista vacía. Veamos algunos ejemplos.

```
?- [a,b,c,d,e] = [Pri,Seg,Ter|Rst].
   Pri = a
   Seg = b
   Ter = c
   Rst = [d, e]
   Yes
?- [a,b,c] = [Pri,Seg,Ter|Rst].
   Pri = a
   Seg = b
   Ter = c
   Rst = []
   Yes
?- [a,b] = [Pri,Seg,Ter|Rst].
   No
?- [a] = [Pri|Rst].
   Pri = a
   Rst = []
   Yes
```

A.3.7. Acumuladores

Como es habitual en otros lenguajes, también en Prolog las definiciones recursivas de predicados se optimizan si se emplean acumuladores. Veamos la siguiente

definición del predicado `longitud(?L,?Long)`, que se verifica si `Long` es igual al número de elementos[6] de la lista `L`.

```
longitud([],0).
longitud([_X|Y],N):-
        longitud(Y,M),
        N is M+1.
```

Observando las cláusulas anteriores fácilmente se constata que para calcular la longitud de una lista dada, primero hay que calcular la longitud de su cola para después sumarle una unidad. Por tanto, hay que esperar que acabe la llamada recursiva para hacer todavía una operación más. Esta forma de proceder resulta bastante costosa, especialmente en cuanto al gasto de memoria.

Sin embargo, empleando *acumuladores* se reduce mucho el coste, pues antes de pasar a un nuevo nivel de recursividad se hace todo el trabajo del nivel anterior. La definición del predicado anterior, empleando un argumento que sirve de acumulador, es

```
longitud2(Lista,Long) :-
        longitud(Lista,0,Long).
longitud([],Long,Long).
longitud([_Pri|Rst],Acum,Long):-
        Acum1 is Acum+1,
        longitud(Rst,Acum1,Long).
```

A.3.8. Definición de operadores

En Prolog, los functores y predicados pueden definirse como *operadores*. Esto puede hacerse llamando a `op(Precedencia,Tipo,Nombre)`, donde `Precedencia` es un número entre 0 y 1200 —cuanto mayor precedencia tenga un operador, más grandes son las subfórmulas que caen bajo su alcance—, `Tipo` es `fx` o `fy` para los operadores *prefijos* —que van delante de sus argumentos— `xfx`, `xfy` o `yfx` para

[6] Prolog dispone del predicado `length/2`, equivalente a nuestro `longitud/2`.

los operadores *infijos* —que van entre sus argumentos— y `xf` o `yf` para los *sufijos* —que se sitúan tras sus argumentos—. La asociatividad del operador se indica mediante la `x` —no asociativo— y la `y` —asociativo—. Cuando aparecen dos letras, una a la izquierda y otra a la derecha de la `f` —para operadores *infijos*— están indicando, respectivamente, la asociatividad por la izquierda y por la derecha del operador en cuestión. Por último, `Nombre` es la forma en que se escribe el propio operador.

```
:-      op(400,fy,-),       % Negación
        op(500,xfy,&),      % Conjunción
        op(600,xfy,v),      % Disyunción
        op(650,xfy,=>),     % Implicación
        op(700,xfy,<=>).    % Doble implicación
```

La cláusula anterior recoge la definición de los operadores lógicos tal como aparece en los siguientes apéndices. A continuación mostramos algunos ejemplos que muestran las funciones de la precedencia y la asociatividad.

```
?- (a v b & c) = (S v G).
   S = a
   G = b&c
   Yes
?- (a v b & c) = (S & G).
   No
?- (a => b => c) = (S => G).
   S = a
   G = b=>c
   Yes
```

A.4. Aspectos teóricos

La posibilidad de definir predicados y relaciones lógicas hace a Prolog un lenguaje de programación especialmente apropiado para la implementación de sistemas de razonamiento automático, tal como haremos en los siguientes apéndices.

Además, su carácter declarativo hace casi innecesaria la verificación de los programas lógicos, ya que los predicados pueden implementarse mediante cláusulas que son prácticamente una traducción a Prolog de sus definiciones formales, tal como más arriba ha ocurrido al definir `factorial/2` o `longitud/2`. Aunque existen trabajos sobre verificación formal de programas lógicos, como [LCLRC93], generalmente basta asumir la corrección de los intérpretes para asegurar que el comportamiento de los programas será correcto.

Sin embargo, la unificación de Prolog, tal como es realizada por defecto por los intérpretes, no es correcta. Como ejemplo, veamos el siguiente programa,

```
es_estrictamente_menor_que(N,sucesor(N)).
```

que simplemente contiene una cláusula que dice que todo número natural `N` es estrictamente menor que su sucesor. Si preguntamos si todo número `X` es estrictamente menor que sí mismo, la respuesta

```
?- es_estrictamente_menor_que(X,X).
   X = sucesor(**)
   Yes
```

es afirmativa, apareciendo la variable `X` unificada con el término `sucesor(**)`, que representa un término infinito de tipo `sucesor(sucesor(...(X)...))`. Sin embargo, se trata de una unificación incorrecta, pues no existe ninguna sustitución que haga a `X` y `sucesor(X)` idénticos, ya que los términos infinitos están excluidos del lenguaje de primer orden.

Esta incorrección de la unificación que hace Prolog se puede salvar usando un predicado que al unificar compruebe que cada variable que aparezca en la sustitución no ocurra en el término por el que se reemplaza. El predicado que implementa la unificación correcta es `unify_with_occurs_check/2`. Si reescribimos el programa anterior usando este predicado, el resultado es

```
es_estrictamente_menor_que(N,sucesor(M)) :-
        unify_with_occurs_check(N,M).
```

y ahora

```
?- es_estrictamente_menor_que(X,X).
   No
```

Combinando adecuadamente las dos unificaciones, se puede reservar la correcta para las ocasiones en que de otra forma podrían producirse incorrecciones —como al programar sistemas de razonamiento para lógica de primer orden—, y la estándar para los demás casos; así se gana en eficiencia, ya que la unificación común de Prolog es computacionalmente mucho menos costosa.

También pueden aparecer en Prolog problemas de incompletud. Consideremos el siguiente programa,

```
hermano_de(X,Y) :- hermano_de(Y,X).
hermano_de(juan,pablo).
```

una de cuyas consecuencias lógicas es `hermano_de(pablo,juan)`. Sin embargo, si lanzamos este literal como una pregunta al intérprete, el resultado es que se agota la memoria, tras entrar en una rama infinita del árbol SLD —la primera rama, pues la resolución SLD no cambia de rama hasta no completar la anterior—, ya que la regla de selección hace volver repetidamente sobre la primera cláusula. Simplemente cambiando el orden en que ambas cláusulas aparecen en el programa, este problema desaparece. Encontramos aquí una nueva limitación al carácter declarativo de Prolog, que habrá que tener en cuanta al programar, ya que el orden de las cláusulas que definen un mismo predicado no resulta indiferente.

Otro problema que aparece al programar en Prolog es la ausencia de la negación lógica. La negación por fallo, `not/1`, no es equivalente, ya que no es posible concluir $\neg\alpha$ del hecho de no poder demostrar α. El modo de razonamiento que se correspondería con la negación por fallo es el que hacemos, por ejemplo, cuando consideramos inocente a alguien por no poder demostrar su culpabilidad. Pero esta forma de razonar no siempre es válida. Para paliar la carencia de la negación lógica a menudo se emplea la *hipótesis del mundo cerrado* al construir las bases

de conocimientos, según la cual todo aquello que no se sepa verdadero se considera falso. Esta hipótesis se basa en el hecho de que en los contextos para los que se escriben bases de conocimientos generalmente hay más cosas falsas que verdaderas.

Cuando en los siguientes apéndices usemos Prolog para programar sistemas de razonamiento en lógica clásica, necesitamos que el comportamiento de los programas obtenidos sea correcto, completo, y que desde luego pueda manejar correctamente la negación lógica. Lograremos este objetivo gracias a la posibilidad de usar la resolución SLD para implementar sobre ella otros motores de inferencia —con las propiedades metalógicas deseadas—, como serán el cálculo de tablas semánticas o el de resolución.

Apéndice B

Implementaciones proposicionales

B.1. Abducción con tablas semánticas

```
/* Fichero: aliseda.pl Implementación de la abducción con tablas se-
   mánticas, a partir de Aliseda, ‘‘Seeking Explanations: Abduction
   in Logic, Philosophy of Science and Artificial Intelligence’’ */

:-use_module(libreria,[tabla/2,medidas/2,terminar/2,complementario/2,
                       añadir_fórmula_tabla/3,muestra_conjunto/1]).

/* Operadores lógicos */
:-      op(400,fy,-),
        op(500,xfy,&),
        op(600,xfy,v),
        op(650,xfy,=>),
        op(700,xfy,<=>).

/* Predicados principales */
/* abduce_estad(+Teo,+Obs) comienza la resolución del problema abducti-
   vo para la teoría Teo y la observación Obs. Al final, muestra los
   recursos empleados en tiempo, inferencias, y memoria */
abduce_estad(Teo,Obs) :-
        medidas(Esp,Inf),
        abduce_tab(Teo,Obs),
        terminar(Esp,Inf).
```

```
/* abduce_tab(+Teo,+Obs) realiza un proceso abductivo dada la teoría
   Teo (una fórmula) y la observación Obs (literal). Sigue los pasos:
   1. Realiza la tabla semántica de la teoría (TabTeo).
   2. Extiende TabTeo con el literal complementario a Obs (TabExt).
   3. Si TabExt es cerrada (igual a la lista vacía), entonces la exten-
   sión es cerrada y la observación está explicada por la teoría. En
   otro caso, pasa a 4.
   4. Si la extensión es abierta, entonces no son posibles las explica-
   ciones consistentes. Si no, pasa a 5.
   5. Va realizando los cierres parciales y totales, de ramas y tablas
   tanto de TabTeo como de TabExt.
   6. Construye las explicaciones atómicas y las muestra, quitando
   previamente (si estuviera) el literal que quiere explicarse.
   7. Si todas las ramas de TabExt tienen cierres parciales, entonces
   son posibles explicaciones conjuntivas, y va al paso 8. En otro
   caso, no hay explicaciones conjuntivas.
   8. Se construyen las explicaciones conjuntivas y se muestran */
abduce_tab(Teo,Obs) :-
        format('~N~N   Teoría: ~w',[Teo]),
        format('~N~N   Observación: ~w',[Obs]),
        tabla([Teo],TabTeo),                                    % 1
        format('~N~n   Tabla de la teoría (TabTeo):',[]),
        muestra_conjunto(TabTeo),
        complementario(Obs,NegObs),
        format('~N~n   Complementario del hecho: ~w~n',[NegObs]),
        añadir_fórmula_tabla(TabTeo,NegObs,TabExt),             % 2
        ( TabExt = [] ->                                        % 3
          format('~N~n Extensión cerrada:',[]),
          format('~N~n ** HECHO EXPLICADO POR LA TEORÍA **',[])
        ; % No es extensión cerrada
          format('~N~n   Tabla extendida (TabExt):',[]),
          muestra_conjunto(TabExt),
          length(TabTeo,N),
          length(TabExt,M),
          ( N = M ->                                            % 4
            format('~N~n   Extensión abierta:',[]),
            format('~N~n  ** NO HAY EXPLICACIONES CONSISTENTES **',[])
          ; % Extensión semicerrada
            cierres_totales_ramas(TabTeo,BTCTeo),               % 5
            format('~N~n   Cierres totales de ramas de TabTeo:',[]),
```

```
            muestra_conjunto(BTCTeo),
            cierres_totales_tabla(BTCTeo,TTCTeo),
            format('~N~n   Cierres totales de tabla de TabTeo:',[]),
            muestra_conjunto(TTCTeo),
            cierres_parciales_ramas(BTCTeo,TTCTeo,BPCTeo),
            format('~N~n   Cierres parciales de ramas de TabTeo:',[]),
            muestra_conjunto(BPCTeo),
            cierres_parciales_tabla(BPCTeo,TPCTeo),
            format('~N~n   Cierres parciales de tabla de TabTeo:',[]),
            muestra_conjunto(TPCTeo),
            cierres_totales_ramas(TabExt,BTCUn),
            format('~N~n   Cierres totales de ramas de TabExt:',[]),
            muestra_conjunto(BTCUn),
            cierres_totales_tabla(BTCUn,TTCUn),
            format('~N~n   Cierres totales de tabla de TabExt:',[]),
            muestra_conjunto(TTCUn),
            cierres_parciales_ramas(BTCUn,TTCUn,BPCUn),
            format('~N~n   Cierres parciales de ramas de TabExt:',[]),
            muestra_conjunto(BPCUn),
            explicaciones_atómicas(TTCUn,TPCTeo,ExpAtoProv),     % 6
            subtract(ExpAtoProv,[Obs],ExpAto),
            muestra_explicaciones('atómicas',ExpAto),
            length(TabExt,N2),
            ( length(BPCUn,N2) ->                                % 7
              explicaciones_conjuntivas(BPCUn,BTCTeo,ExpCon),    % 8
              muestra_explicaciones('conjuntivas',ExpCon)
            ; % hay ramas sin cierres parciales
              format('~N~n   No hay explicaciones conjuntivas',[]),
              ExpCon = [])).

/* Construcción de explicaciones abductivas */
/* explicaciones_atómicas(+TTC,+TPC,-Explic) se verifica si Explic es
   el conjunto de explicaciones atómicas, siendo TTC el conjunto de
   cierres totales de la tabla de la teoría con la negación de la ob-
   servación y TPC el conjunto de cierres parciales de la tabla de la
   teoría. Se consigue haciendo la intersección de ambos conjuntos.
   Es necesario suprimir el literal que se quiere explicar, si es que
   aparece, lo que se hará en el procedimiento principal */
explicaciones_atómicas(TTC,TPC,Explic) :-
        intersection(TTC,TPC,Explic).
```

```
/* explicaciones_conjuntivas(+BPC,+BTC,-Explic) se verifica si Explic
   es el conjunto de explicaciones conjuntivas, siendo BPC el conjunto
   de los cierres parciales de todas las ramas de la tabla de la teo-
   ría con la negación de la observación, y BTC el conjunto de cie-
   rres totales de la tabla de la teoría. Explic se obtiene:
   1. Se hacen todas las combinaciones posibles que toman un literal
   de cada uno de los conjuntos de BPC y se eliminan los repetidos y
   los no minimales.
   2. Se seleccionan, de las combinaciones obtenidas, las que no cie-
   rran BTC */
explicaciones_conjuntivas(BPC,BTC,Explic) :-
        findall(Exp,es_explicación_conjuntiva(BPC,Exp),Explic0),
        quita_repet(Explic0,Explic1),
        findall(Exp2,(member(Exp2,Explic1),
                     exp_consistente(Exp2,BTC)),Explic).

/* es_explicación_conjuntiva(+BPC,?Exp) se verifica si Exp es una lista
   con un literal de cada una de las listas de literales que forman
   BPC, eliminando las repeticiones */
es_explicación_conjuntiva(BPC,Exp) :-
        uno_de_cada(BPC,Exp0),
        sort(Exp0,Exp).

/* uno_de_cada(+ListaDeListas,?Lista) se verifica si Lista contiene un
   elemento de cada una de las listas que forman ListaDeListas */
uno_de_cada([],[]).
uno_de_cada([P|R],[EP|ER]) :-
        member(EP,P),
        uno_de_cada(R,ER).

/* exp_consistente(+Exp,+Cierres) se verifica si hay al menos una lista
   que forma parte de Cierres cuya intersección con Exp es vacía */
exp_consistente(Exp,[R1|OR]) :-
        (intersection(Exp,R1,[]);
            exp_consistente(Exp,OR)),!.

/* quita_repet(+CIni,?CFin) se verifica si CFin es el resultado de qui-
   tar de CIni todas las explicaciones conjuntivas repetidas o no mini-
   males */
quita_repet(A,B) :-
        findall(N-Ex,(member(Ex,A),length(Ex,N)),Conj),
```

```
        keysort(Conj,Conj2),
        findall(P,member(_-P,Conj2),Orden),
        quita_repet_aux(Orden,[],B).
quita_repet_aux([],F,F).
quita_repet_aux([Mayor|Resto],ConjProv,Final) :-
        member(Menor,ConjProv),
        subset(Menor,Mayor),!,
        quita_repet_aux(Resto,ConjProv,Final).
quita_repet_aux([P|Resto],ConjProv,Final) :-
        quita_repet_aux(Resto,[P|ConjProv],Final).

/* Cierres parciales y totales de ramas y tablas */
/* cierres_totales_ramas(+Tabla, -BTC) ve verifica si, dada una Tabla,
   BTC es el conjunto de los cierres totales de cada rama. Se compone
   de todos los literales complementarios de los que aparecen en cada
   rama */
cierres_totales_ramas([B|Resto],[C1|CResto]):-
        cierres_totales_ramas_aux(B,C1P),
        sort(C1P,C1),
        cierres_totales_ramas(Resto,CResto).
cierres_totales_ramas([],[]).
cierres_totales_ramas_aux([A|R],[NA|CR]) :-
        complementario(A,NA),
        cierres_totales_ramas_aux(R,CR).
cierres_totales_ramas_aux([],[]).

/* cierres_totales_tabla(+BTC,-TTC) se verifica si TTC es el conjunto
   de los cierres totales de una tabla cuyo conjunto de cierres tota-
   les de cada rama es BTC. Se obtiene por la intersección de todos
   los cierres totales de ramas */
cierres_totales_tabla([C1|BTC],TTC):-
        cierres_totales_tabla_aux(BTC,C1,TTC).
cierres_totales_tabla_aux([],TTC,TTC).
cierres_totales_tabla_aux([C1|Resto],TP,TTC) :-
        intersection(C1,TP,TTC1),
        cierres_totales_tabla_aux(Resto,TTC1,TTC).

/* cierres_parciales_ramas(+BTC,+TTC,-BPC) se verifica si BPC es el
   conjunto de los cierres parciales de cada rama de una tabla cuyo
   conjunto de cierres parciales de ramas es BTC y cuyo conjunto de
   cierres totales de tabla es TTC. Se obtiene quitando a cada con-
```

```
   junto de BTC aquellos literales que aparecen en TTC */
cierres_parciales_ramas([],_TTC,[]).
cierres_parciales_ramas([R1|Otras],TTC,[[E1|R1PC]|OtrasPC]):-
        subtract(R1,TTC,[E1|R1PC]),!,
        cierres_parciales_ramas(Otras,TTC,OtrasPC).
cierres_parciales_ramas([_|Otras],TTC,BPC):-
        cierres_parciales_ramas(Otras,TTC,BPC).

/* cierres_parciales_tabla(+BPC,-TPC) se verifica si TPC es el conjunto
   de cierres parciales de una tabla cuyo conjunto de cierres parciales
   de rama es BPC */
cierres_parciales_tabla(BPC,TPC) :-
        cierres_parciales_tabla_aux(BPC,[],TPC).
cierres_parciales_tabla_aux([],TPC,TPC).
cierres_parciales_tabla_aux([B1|RB],Acum,TPC):-
        union(B1,Acum,Acum2),
        cierres_parciales_tabla_aux(RB,Acum2,TPC).

/* Predicados auxiliares */
/* muestra_explicaciones(+Tipo,+Conj) sirve para imprimir en pantalla
   todas las explicaciones de Tipo 'atómico' o 'conjuntivo' del con-
   junto Conj */
muestra_explicaciones(Tipo,[]) :-!,
        format('~N~n   ** No hay explicaciones ~w **~n',[Tipo]).
muestra_explicaciones(Tipo,Exp) :-
        format('~N~n   Las explicaciones ~w son:~n',[Tipo]),
        muestra_explicaciones_aux(Tipo,Exp).
muestra_explicaciones_aux(_,[]) :- !.
muestra_explicaciones_aux('conjuntivas',[C1|Resto]) :-
        format('~N      ',[]),
        escribe_conjunción(C1),
        muestra_explicaciones_aux('conjuntivas',Resto).
muestra_explicaciones_aux('atómicas',[A1|Resto]) :-
        format('~N      ~w~n',[A1]),
        muestra_explicaciones_aux('atómicas',Resto).

/* escribe_conjunción(+Lista) muestra una conjunción de los literales
   de Lista */
escribe_conjunción([L1,L2|Resto]) :-
        format('~w & ',[L1]),
        escribe_conjunción([L2|Resto]).
```

```
escribe_conjunción([L1]) :-
        format('~w~n',[L1]).
```

B.2. Abducción mediante δ-resolución (primera versión)

```
/* Fichero: dresol1.pl Abducción mediante d-resolución (primera
   versión) */

:-use_module(libreria,[d_clausulas/2,anotado/2,ordenada_por_peso/2,
                       d_resol_anotada/4,medidas/2,comienzo/2,
                       terminar/2,elimina_contradicciones/2,
                       muestra_usable/1]).

/* Operadores lógicos */
:-      op(400,fy,-),
        op(500,xfy,&),
        op(600,xfy,v),
        op(650,xfy,=>),
        op(700,xfy,<=>).

/* abduce_dresol(+Teo,+Obs) comienza un proceso abductivo con la teoría
   Teo (conjunción de fórmulas proposicionales) y la Observación Obs.
   Los pasos principales del proceso son:
   1.- Obtención de ClsSat, forma d-clausal de Teo => Obs.
   2.- Obtención de UltUsab, último usable de la búsqueda por d-resolu-
   ción desde ClsSat. Si UltUsab contiene la d-cláusula vacía, el pro-
   ceso termina, pues Obs es consecuencia de Teo. En otro caso,
   3.- Se obtienen las formas d-clausales de Teo y de -Obs, que son,
   respectivamente, ClsTeoSat y ClsObsSat.
   4.- Se seleccionan de UltUsab las abducciones explicativas */
abduce_dresol(Teo,Obs) :-
        medidas(Esp,Inf),
        comienzo(Teo,Obs),
        d_clausulas(Teo => Obs,Cls),
        format('~N~n   Forma d-clausal de Teo => Obs:~n',[]),
        elimina_contradicciones(Cls,ClsSat),                    % 1
        format('~N~n   Soporte para la búsqueda abductiva:~n',[]),
```

```
        anotado(ClsSat,Sop),
        ordenada_por_peso(Sop,Sop1),
        d_resol_anotada([],Sop1,[],UltUsab),                    % 2
        ( UltUsab = [_*_*[]|_] ->
            format('~N~n   *** Obs es consecuencia de Teo ***~n',[])
        ; % Obs no es consecuencia lógica de Teo
            format('~N~n   Último usable:~n',[]),
            muestra_usable(UltUsab),
            d_clausulas(Teo,ClsTeo),
            format('~N~n   Forma d-clausal de Teo:~n',[]),
            elimina_contradicciones(ClsTeo,ClsTeoSat),
            d_clausulas(- Obs,ClsObs),
            format('~N~n   Forma d-clausal de -Obs:~n',[]),
            elimina_contradicciones(ClsObs,ClsObsSat),
            busca_explic(UltUsab,ClsTeoSat,ClsObsSat,Explic),
            format('~N~n   Abducciones explicativas:~n',[]),
            muestra_usable(Explic)),
            terminar(Esp,Inf),!.

/* busca_explic(+Planas,+Teo,+Obs,?Explic) se verifica si de las expli-
   caciones planas que forman parte de Planas, Explic es el conjunto de
   las explicativas, siendo Teo las d-cláusulas satisfactibles de la
   teoría, y Obs las de la negación de la observación */
busca_explic(Planas,Teo,Obs,Explic) :-
        findall(N*H*E,(member(N*H*E,Planas),
                    member(T,Teo),
                    append(E,T,ET),
                    satisfactible(ET),
                    member(O,Obs),
                    append(E,O,EO),
                    satisfactible(EO)),
                Explic),!.

/* satisfactible(+DCl) se verifica si la d-cláusula DCl es satisfacti-
   ble, es decir, no contiene literales complementarios */
satisfactible(DCl) :-
        \+ (member(-A,DCl),
            member(A,DCl)).
```

B.3. Abducción mediante δ-resolución (segunda versión)

```
/* Fichero: dresol2.pl Abducción mediante d-resolución. Segunda ver-
   sión, que integra los procesos de generación y selección */

:-use_module(libreria,[d_clausulas/2,anotado/2,ordenada_por_peso/2,
                       d_resol_anotada/4,medidas/2,comienzo/2,
                       terminar/2,elimina_contradicciones/2,
                       muestra_usable/1,elimina_subsumidas_var/5,
                       valor_global/2]).

/* Operadores lógicos */
:-      op(400,fy,-),
        op(500,xfy,&),
        op(600,xfy,v),
        op(650,xfy,=>),
        op(700,xfy,<=>).

/* abduce_proc(+Teo,+Obs) se verifica si Teo es una fórmula proposicio-
   nal, correspondiente a la teoría, y Obs otra, correspondiente a la
   observación. Entonces procede a la resolución del problema abduc-
   tivo a través de cuatro pasos:
   - ANÁLISIS DE LA TEORÍA. Sirve para determinar si:
   * La teoría es universalmente válida (se detiene).
   * La teoría es inconsistente (se detiene).
   * La teoría es contingente (va al paso siguiente).
   - ANÁLISIS DE LA OBSERVACIÓN. Donde determina si:
   * La observación es inconsistente (se detiene).
   * La observación es universalmente válida (se detiene).
   * La observación es contingente (va al paso siguiente).
   - BÚSQUEDA DE REFUTACIONES. Determina si:
   * La observación contradice la teoría (se detiene).
   * Teoría y observación son satisfactibles (paso siguiente).
   - BÚSQUEDA DE EXPLICACIONES. Realiza entonces la búsqueda de solu-
   ciones abductivas. Pueden ocurrir tres cosas:
   * La observación se deriva de la teoría (avisa y para).
   * Devuelve abducciones explicativas minimales (y para).
   * No hay abducciones explicativas (lo avisa y para).
   Al final de la ejecución, devuelve información sobre la memoria y
```

```
   tiempo empleados, así como del número de inferencias */
abduce_proc(Teo,Obs) :-
        medidas(Esp,Inf),
        comienzo(Teo,Obs),
        analiza_teo(Teo,UsTeo),
        ( UsTeo = fin ->
            terminar(Esp,Inf)
        ; % Teoría contingente
            analiza_obs(Obs,UsObs),
            ( UsObs = fin ->
                terminar(Esp,Inf)
            ; % Teoría y hecho contingentes
                busca_ref(UsTeo,UsObs,Sop),
                ( Sop = fin ->
                    terminar(Esp,Inf)
                ; % Pueden hacerse explicaciones
                    busca_abd(Sop),
                    terminar(Esp,Inf)))),!.

/* analiza_teo(+Teo,-UsTeo) realiza el primer paso de la resolución de
   un problema abductivo, el análisis de la teoría. Procede de la si-
   guiente manera:
   1.- Obtiene la forma d-clausal de -Teo, y elimina las d-cláusulas
   contradictorias, obteniendo el conjunto ClSatTeo. Entonces, si
   ClSatTeo es vacío, es que -Teo es no satisfactible, por lo que Teo
   es universalmente válida. En este caso, avisa con un mensaje y de-
   tiene la búsqueda, unificando UsTeo con 'fin', lo que indica al pre-
   dicado principal que la búsqueda ha terminado. En otro caso,
   2.- Aplica d-resolución, tomando como soporte ClSatTeo. Si se llega
   a la d-cláusula vacía, entonces es que -Teo es válida, por lo que
   Teo es no satisfactible. En tal caso, avisa con un mensaje y se de-
   tiene (unificando también UsTeo con 'fin'). Si no se llegó a la
   d-cláusula vacía tras saturar la búsqueda, es que la teoría es con-
   tingente, y guarda en UsTeo el conjunto de refutables de Teo */
analiza_teo(Teo,UsTeo) :-
        format('~N~n1. ANÁLISIS DE LA TEORÍA:~n',[]),
        d_clausulas(-Teo,ClTeo),
        format('~N~n   d_cláusulas de la teoría:~n~n',[]),
        elimina_contradicciones(ClTeo,ClSatTeo),
        ( ClSatTeo = [] ->
            format('~N~n   *** TEORÍA VÁLIDA ***~n',[]),
```

```
            UsTeo = fin
        ; % ClSatTeo no es vacío.
            format('~N~n   Soporte de la teoría:',[]),
            anotado(ClSatTeo,SopTeo),
            ordenada_por_peso(SopTeo,SopTeo1),
            d_resol_anotada([],SopTeo1,[],UsTeoProv),
            ( UsTeoProv = [_*_*[]|_] ->
                format('~N~n   *** TEORÍA INCONSISTENTE ***~n',[]),
                escribe_prueba(UsTeoProv),
                UsTeo = fin
            ; % No se encuentra d-cláusula vacía
                format('~N~n   *** TEORÍA CONTINGENTE ***~n',[]),
                format('~N~n   Refutables de la teoría:~n',[]),
                muestra_usable(UsTeoProv),
                UsTeoProv = UsTeo)).

/* analiza_obs(+Obs,-UsObs) realiza el segundo paso de la resolución de
   un problema abductivo, el análisis de la observación. Procede de
   la siguiente manera:
   1.- Obtiene la forma d-clausal de Obs, y elimina las d-cláusulas
   contradictorias, obteniendo el conjunto ClSatObs. Entonces, si
   ClSatObs es vacío, es que Obs es no satisfactible. En este caso,
   avisa con un mensaje y detiene la búsqueda, unificando UsObs con
   'fin', lo que indica al predicado principal que la búsqueda ha ter-
   minado. En otro caso,
   2.- Aplica d-resolución, tomando como soporte ClSatObs. Si se llega
   a la d-cláusula vacía, entonces es que Obs es válida. En tal caso,
   avisa con un mensaje y se detiene (unificando también UsObs con
   'fin'). Si no se llegó a la d-cláusula vacía tras saturar la bús-
   queda, es que la observación es contingente, y guarda en UsObs el
   conjunto de mínimos abductivos de Obs */
analiza_obs(Obs,UsObs) :-
        format('~N~n2. ANÁLISIS DE LA OBSERVACIÓN:~n',[]),
        d_clausulas(Obs,ClObs),
        format('~N~n   d-cláusulas de la observación:~n~n',[]),
        elimina_contradicciones(ClObs,ClSatObs),
        ( ClSatObs = [] ->
            format('~N~n   *** OBSERVACIÓN INCONSISTENTE ***~n',[]),
            UsObs = fin
        ; % Hay d-cláusulas no contradictorias
            format('~N~n   Soporte de la observación:',[]),
```

```
            anotado(ClSatObs,SopObs),
            ordenada_por_peso(SopObs,SopObs1),
            d_resol_anotada([],SopObs1,[],UsObsProv),
            ( UsObsProv = [_*_*[]|_] ->
                format('~N~n   *** OBSERVACIÓN VÁLIDA ***~n',[]),
                escribe_prueba(UsObsProv),
                UsObs = fin
            ; % No se encuentra d-cláusula vacía
            format('~N~n   *** OBSERVACIÓN CONTINGENTE ***~n',[]),
            format('~N~n  Mínimos abductivos de la observación:~n',[]),
            muestra_usable(UsObsProv),
            UsObsProv = UsObs)).

/* busca_ref(+UsTeo,+UsObs,-Sop) realiza el tercer paso de la resolu-
   ción de un problema abductivo, la búsqueda de refutaciones, es
   decir, comprueba si la observación contradice la teoría. Para ello,
   comprueba si cada d-cláusula de UsObs está subsumida por una d-cláu-
   sula de UsTeo. Si ello es así, es que la negación de la observación
   se deriva de la teoría, por lo que Obs es inconsistente con Teo. En
   este caso, avisa con un mensaje y detiene la búsqueda (ahora, unifi-
   ca Sop con 'fin'). En otro caso, es posible comenzar la búsqueda de
   explicaciones, tomando como soporte, Sop, conjunto unión de UsTeo
   con las d-cláusulas de UsObs no subsumidas por las de UsTeo */
busca_ref(UsTeo,UsObs,Sop) :-
        format('~N~n3. BÚSQUEDA DE REFUTACIONES:~n',[]),
        elimina_subsumidas_var(UsTeo,UsObs,UsTeo,NoSubs,SopProv),
        ( NoSubs = [] ->
            format('~N~n   *** LA OBSERVACIÓN REFUTA LA TEORÍA ***',
                   []),
            Sop = fin
        ; % No hay refutación
            SopProv = Sop,
            format('~N~n   *** NO HAY REFUTACIÓN ***~n',[]),
            format('~N~n   Base abductiva:~n',[]),
            muestra_usable(Sop)).

/* busca_abd(+Sop) realiza el cuarto y último paso de la búsqueda de
   soluciones abductivas. Toma Sop, la base abductiva, y aplica d-reso-
   lución. Si se llega a la d-cláusula vacía, entonces la observación
   es consecuencia de la teoría. En tal caso avisa y se para la búsque-
   da. En otro caso, tras saturar la búsqueda, toma todas las d-cláusu-
```

```
   las que forman parte del conjunto usable final pero no estaban en el
   soporte inicial, formando el conjunto Explic de abducciones explica-
   tivas y minimales. Si Explic es vacío, es que no existen abducciones
   explicativas, de lo que avisa con un mensaje. En otro caso, devuelve
   los elementos de Explic */
busca_abd(Sop) :-
        format('~N~n4. BÚSQUEDA DE EXPLICACIONES:~n',[]),
        ordenada_por_peso(Sop,Sop1),
        valor_global(n_clausulas_retenidas,Min1),
        d_resol_anotada([],Sop1,[],Ref),
        ( Ref = [_*_*[]|_] ->
            format('~N~n   *** OBSERVACIÓN EXPLICADA POR TEORÍA ***',
                   []),
            escribe_prueba(Ref)
        ; % la observación no se deriva de la teoría
            valor_global(n_clausulas_retenidas,Max),
            Min is Min1+1,
            findall(N*H*C,(between(Min,Max,N),
                           memberchk(N*H*C,Ref)),Explic),
            ( Explic = [] ->
              format('~N~n   *** NO HAY ABDUCCIONES EXPLICATIVAS ***',
                     [])
            ; % hay abducciones explicativas
                format('~N~n   Las abducciones explicativas son:',[]),
                muestra_usable(Explic))).
```

B.4. Integración de tablas semánticas y δ-resolución

```
/* Fichero: dresol3.pl Abducción mediante d-resolución. Versión
   que emplea tablas semánticas y optimiza el proceso al explicar
   literales */

:-use_module(libreria,[anotado/2,ordenada_por_peso/2,d_resol_anotada/4,
                       medidas/2,comienzo/2,terminar/2,
                       muestra_usable/1,elimina_subsumidas_var/5,
                       valor_global/2,literal/1,tabla/2,
                       complementario/2,escribe_prueba/1,
```

```
                        muestra_conjunto/1]).

/* Operadores lógicos */
:-      op(400,fy,-),
        op(500,xfy,&),
        op(600,xfy,v),
        op(650,xfy,=>),
        op(700,xfy,<=>).

/* El predicado abduce_mod/2 se comporta como abduce_proc/2 del fiche-
   ro 'dresol2.pl', a diferencia de que cuando la observación es un
   literal abrevia el proceso.

   El predicado refuta_y_abduce/5 implementa en un solo paso la búsque-
   da de refutaciones y de explicaciones cuando la observación es un
   literal. Es la modificación principal de este fichero.

   El resto de predicados se comporta como los que aparecen en el fi-
   chero 'dresol2.pl', a excepción de que emplean tablas semánticas
   para la transformación a forma d-clausal */

/* Para explicar literales */
abduce_mod(Teo,Obs) :-
  literal(Obs),!,
  medidas(Esp,Inf),
  comienzo(Teo,Obs),
  format('~N~n   *** OBSERVACIÓN LITERAL ***~n',[]),
  analiza_teo(Teo,UsTeo),
   ( UsTeo = fin ->
     terminar(Esp,Inf)
   ; % Teoría contingente
     format('~N~n2. BÚSQUEDA DE REFUTACIONES Y EXPLICACIONES:~n',[]),
     complementario(Obs,NObs),
     refuta_y_abduce([Obs],NObs,UsTeo,[],Res),
     ( Res = 'refuta' ->
       format('~N~n   *** TEORÍA REFUTADA ***~n',[]),
       terminar(Esp,Inf)
     ; % no hay refutación
       ( Res = 'explica' ->
         format('~N~n   *** OBSERVACIÓN EXPLICADA ***~n',[]),
         terminar(Esp,Inf)
```

```
        ; % tampoco está explicada
           findall(X, (member(X,Res),
                       member(_N*_H*Y,UsTeo),
                       subset(Y,X)), Res2),
           subtract(Res,Res2,Res3),
           ( Res3 = [] ->
             format('~N~n *** NO HAY ABDUCCIONES EXPLICATIVAS ***',[]),
             terminar(Esp,Inf)
           ; % hay explicaciones
             format('~N~n Son abducciones explicativas minimales:',[]),
             muestra_conjunto(Res3),
             terminar(Esp,Inf))))).

/* Para explicar fórmulas complejas */
abduce_mod(Teo,Obs) :-
        medidas(Esp,Inf),
        comienzo(Teo,Obs),
        analiza_teo(Teo,UsTeo),
        ( UsTeo = fin ->
            terminar(Esp,Inf)
        ; % Teoría contingente
            analiza_obs(Obs,UsObs),
            ( UsObs = fin ->
                terminar(Esp,Inf)
            ; % Teoría y hecho contingentes
                busca_ref(UsTeo,UsObs,Sop),
                ( Sop = fin ->
                    terminar(Esp,Inf)
                ; % Pueden hacerse explicaciones
                    busca_abd(Sop),
                    terminar(Esp,Inf)))),!.

analiza_teo(Teo,UsTeo) :-
        format('~N~n1. ANÁLISIS DE LA TEORÍA:~n',[]),
        tabla([-Teo],ClSatTeo),
        ( ClSatTeo = [] ->
            format('~N~n   *** TEORÍA VÁLIDA ***~n',[]),
            UsTeo = fin
        ; % ClSatTeo no es vacío.
            format('~N~n   Soporte de la teoría:',[]),
            anotado(ClSatTeo,SopTeo),
```

```
            ordenada_por_peso(SopTeo,SopTeo1),
            d_resol_anotada([],SopTeo1,[],UsTeoProv),
            ( UsTeoProv = [_*_*[]|_] ->
                format('~N~n   *** TEORÍA INCONSISTENTE ***~n',[]),
                escribe_prueba(UsTeoProv),
                UsTeo = fin
            ; % No se encuentra d-cláusula vacía
                format('~N~n   *** TEORÍA CONTINGENTE ***~n',[]),
                format('~N~n   Refutables de la teoría:~n',[]),
                muestra_usable(UsTeoProv),
                UsTeoProv = UsTeo)).

analiza_obs(Obs,UsObs) :-
        format('~N~n2. ANÁLISIS DE LA OBSERVACIÓN:~n',[]),
        tabla([Obs],ClSatObs),
        ( ClSatObs = [] ->
            format('~N~n   *** OBSERVACIÓN INCONSISTENTE ***~n',[]),
            UsObs = fin
        ; % Hay d-cláusulas no contradictorias
            format('~N~n   Soporte de la observación:',[]),
            anotado(ClSatObs,SopObs),
            ordenada_por_peso(SopObs,SopObs1),
            d_resol_anotada([],SopObs1,[],UsObsProv),
            ( UsObsProv = [_*_*[]|_] ->
                format('~N~n   *** OBSERVACIÓN VÁLIDA ***~n',[]),
                escribe_prueba(UsObsProv),
                UsObs = fin
            ; % No se encuentra d-cláusula vacía
            format('~N~n   *** OBSERVACIÓN CONTINGENTE ***~n',[]),
            format('~N~n  Mínimos abductivos de la observación:~n',[]),
            muestra_usable(UsObsProv),
            UsObsProv = UsObs)).

busca_ref(UsTeo,UsObs,Sop) :-
        format('~N~n3. BÚSQUEDA DE REFUTACIONES:~n',[]),
        elimina_subsumidas_var(UsTeo,UsObs,UsTeo,NoSubs,SopProv),
        ( NoSubs = [] ->
            format('~N~n   *** LA OBSERVACIÓN REFUTA LA TEORÍA ***',
                   []),
            Sop = fin
        ; % No hay refutación
```

```
            SopProv = Sop,
            format('~N~n   *** NO HAY REFUTACIÓN ***~n',[]),
            format('~N~n   Base abductiva:~n',[]),
            muestra_usable(Sop)).

busca_abd(Sop) :-
        format('~N~n4. BÚSQUEDA DE EXPLICACIONES:~n',[]),
        ordenada_por_peso(Sop,Sop1),
        valor_global(n_clausulas_retenidas,Min1),
        d_resol_anotada([],Sop1,[],Ref),
        ( Ref = [_*_*[]|_] ->
            format('~N~n   *** OBSERVACIÓN EXPLICADA POR TEORÍA ***',
                   []),
            escribe_prueba(Ref)
        ; % la observación no se deriva de la teoría
            valor_global(n_clausulas_retenidas,Max),
            Min is Min1+1,
            findall(N*H*C,(between(Min,Max,N),
                           memberchk(N*H*C,Ref)),Explic),
            ( Explic = [] ->
              format('~N~n   *** NO HAY ABDUCCIONES EXPLICATIVAS ***',
                     [])
            ; % hay abducciones explicativas
                format('~N~n   Las abducciones explicativas son:',[]),
                muestra_usable(Explic))).

/* refuta_y_abduce(+[Lit],+CompLit,+ListaDCls,+Acum,?Result) se verifi-
   ca si siendo +[Lit] una d-cláusula unitaria con el literal Lit,
   +CompLit el literal complementario de Lit, +ListaDCls una lista de
   d-cláusulas y +Acum un argumento que sirve de acumulador, entonces:
   - Si [Lit] está en ListaDCls, Result unifica con 'refuta'.
   - Si [CompLit] está en ListaDCls, Result unifica con 'explica'.
   - En otro caso, Result contiene todas las d-cláusulas de Acum más
   las d-cláusulas Cl1,...,Cln tales que Cli+{CompLit} pertenece a
   ListaDCls */
refuta_y_abduce(_,_,[],Ac,Ac).
refuta_y_abduce(DcObs,NObs,[_N*_H*Dcl|NegTeo],Ac,Res) :-
        ( Dcl = DcObs ->
            Res = 'refuta'
        ; % no refuta Dcl
            ( Dcl = [NObs] ->
```

```
            Res = 'explica'
        ; % tampoco explica
            ( select(NObs,Dcl,Exp) ->
                refuta_y_abduce(DcObs,NObs,NegTeo,[Exp|Ac],Res)
            ; % tampoco vale como explicación Dcl
                refuta_y_abduce(DcObs,NObs,NegTeo,Ac,Res))).
```

B.5. Librería de predicados de uso general

```
/* Fichero: libreria.pl  Predicados proposicionales de uso general */

:-module(libreria,[tabla/2,medidas/2,terminar/2,añadir_fórmula_tabla/3,
                   complementario/2,d_clausulas/2,anotado/2,inicia/0,
                   ordenada_por_peso/2,d_resol_anotada/4,comienzo/2,
                   elimina_contradicciones/2,muestra_usable/1,prueba/2,
                   literal/1,elimina_subsumidas_var/5,valor_global/2,
                   escribe_prueba/1,muestra_conjunto/1,asigna_global/2,
                   incrementa_global/2,antecesor/3,escribe_actual/1,
                   d_resolventes/3,escribe_dresolvente/1,subsumida/2,
                   numera/2,escribe_retenida/1,elimina_subsumidas/5,
                   padre/3,escribe_linea/1]).

/* Operadores lógicos */
:-      op(400,fy,-),
        op(500,xfy,&),
        op(600,xfy,v),
        op(650,xfy,=>),
        op(700,xfy,<=>).

/* Creación de tablas semánticas */
/* tabla(+ListaFml,-Tab) se verifica si, dada la lista de fórmulas
   ListaFml, se cumple que Tab es su tabla semántica, representada
   como una lista de sus ramas abiertas, siendo la representación de
   cada rama la lista de sus literales (eliminando repeticiones) */
tabla(LisFml,Tabla) :-
        construye_tabla(LisFml,[[]],Tabla).

/* construye_tabla(+ListaFml,+TablaProv,-Tab) se verifica si Tab es la
   tabla semántica que se obtiene a partir de TablaProv, incorporando
```

```
   todas las fórmulas de la lista de fórmulas ListaFml */
construye_tabla([X|R],TabProv,Res) :-
        añadir_fórmula_tabla(TabProv,X,TResult),
        construye_tabla(R,TResult,Res).
construye_tabla([],R,R).

/* añadir_fórmula_tabla(+Tab0,+Fml,-Res) se verifica si Res es la tabla
   resultante de añadir a la tabla Tab0 la fórmula Fml */
añadir_fórmula_tabla([B|R],Form,Res) :-
        añadir_fórmula_rama(B,Form,NB),
        añadir_fórmula_tabla(R,Form,Res0),
        union(NB,Res0,Res).
añadir_fórmula_tabla([],_,[]).

/* añadir_fórmula_rama(+Rama,+Fml,-Res) se verifica si tras añadir la
   fórmula Fml a la rama Rama, el resultado es Res, que será una lis-
   ta con las ramas resultantes. Res puede tener una o más ramas, de-
   pendiendo de si en Fml ocurre o no una subfórmula de tipo beta. Se
   distinguen cuatro casos según sea Fml:
   1. Doble negación. Entonces hay que añadir la fórmula sin las dos
   negaciones.
   2. Fórmula de tipo alfa de componentes A y B. Entonces se añade A y
   al resultado se añade B. Téngase en cuenta que el resultado puede
   contener más de una rama, por lo que se trata como una tabla.
   3. Fórmula de tipo beta de componentes A y B. Entonces se hacen dos
   ramas. A una se añade A y a la otra B. Finalmente, se unen los re-
   sultados.
   4. Literal. Si su complementario está en la rama, se cierra, lo que
   se representa por la lista vacía. En otro caso, se trata de un lite-
   ral cuyo complementario no está en la rama, por lo que se añade sólo
   si tampoco está el propio literal */
añadir_fórmula_rama(R,-(-F),Res) :- !,            % 1
        añadir_fórmula_rama(R,F,Res).
añadir_fórmula_rama(R,Alfa,Res):-                 % 2
        alfa(Alfa,A,B),!,
        añadir_fórmula_rama(R,A,Res1),
        añadir_fórmula_tabla(Res1,B,Res).
añadir_fórmula_rama(R,Beta,Tab) :-                % 3
        beta(Beta,A,B),!,
        añadir_fórmula_rama(R,A,R1),
        añadir_fórmula_rama(R,B,R2),
```

```
        union(R1,R2,Tab).
añadir_fórmula_rama(Branch,Lit,NewBranch):-        % 4
        ( busca(Lit,Branch,R) ->
            ( R = n -> NewBranch = []
            ; % R = i
                NewBranch = [Branch])
        ; % no están ni F ni su complem.
            NewBranch = [[Lit|Branch]]).

/* Clasificación de tipos de fórmulas */
/* alfa(+Alfa,?A1,?A1) se verifica si Alfa es una fórmula de tipo alfa,
   y A1 y A2 son sus dos componentes */
alfa(A & B, A, B).
alfa(-(A v B), -A, -B).
alfa(-(A => B), A, -B).

/* beta(+Beta,?B1,?B2) se verifica si Beta es una fórmula de tipo beta,
   y B1 y B2 son sus dos componentes */
beta(A <=> B, A & B, -A & -B).
beta(-(A <=> B), A & -B, -A & B).
beta(A v B, A, B).
beta(-(A & B), -A, -B).
beta(A => B, -A, B).

/* Proceso de d-resolución proposicional
   Adaptado a partir de la implementación de Otter que realizan
   J.A. Alonso y J. Borrego en ‘‘Deducción Automática’’ */

/* Manejo de las variables globales */
/* 'inicia' asigna a las variables globales el valor 0. Se trata de
   n_clausulas_analizadas (que cuenta las d-cláusulas analizadas) y
   n_clausulas_retenidas (que cuenta las d-cláusulas retenidas) */
inicia :-
        asigna_global(n_clausulas_retenidas,0),
        asigna_global(n_clausulas_analizadas,0).

/* asigna_global(+A,+T) asigna al átomo A (de forma global) el
   valor T */
asigna_global(A,T) :-
        flag(A,_,T).
```

```
/* incrementa_global(+A,-N) incrementa en 1 el valor global de A, que
   pasa a valer N */
incrementa_global(A,N) :-
        valor_global(A,M),
        N is M+1,
        asigna_global(A,N).

/* valor_global(+A,-T) se verifica si T es el valor global del
   átomo A */
valor_global(A,T) :-
        var(T),
        flag(A,T,T).

/* Anotación de conjuntos de d-cláusulas */
/* anotado(+S1,-S2) se verifica si S2 es el conjunto de d-cláusulas
   anotadas correspondiente al conjunto de d-cláusulas S1 (además,
   escribe cada d-cláusula anotada) */
anotado([],[]).
anotado([C|S1],[N*[]*C|S2]) :-
        incrementa_global(n_clausulas_retenidas,N),
        format('~N     ~w [] ~w~n',[N,C]),
        anotado(S1,S2).

/* Ordenación de d-cláusulas por peso */
/* ordenada_por_peso(+S1,-S2) se verifica si S2 es el conjunto de las
   d-cláusulas anotadas del conjunto S1 ordenadas por peso */
ordenada_por_peso(S1,S2) :-
        findall(L1-N1*P1*C1,
                (member(N1*P1*C1,S1),
                    length(C1,L1)),
                S1a),
        keysort(S1a,S2a),
        findall(CA2,member(_-CA2,S2a),S2).

/* d-resolución en conjuntos de d-cláusulas anotadas y soporte
   ordenado */
/* d_resol_anotada(+Usable,+Soporte,+Subsumidas,-Ref) opera realizando
   una búsqueda mediante d-resolución de la d-cláusula vacía en el con-
   junto de d-cláusulas formado por Usable y Soporte. Subsumidas con-
   tiene las d-cláusulas que aparecieron pero fueron subsumidas. Además
   escribe la búsqueda. Si llega a la d-cláusula vacía, Ref contiene
```

```
   su prueba, y será un conjunto de d-cláusulas con la vacía en primer
   lugar. En otro caso, Ref será el último usable al que llegó */
d_resol_anotada(Usable,[],_,Usable).
d_resol_anotada(Usable,[N*H*[]|Soporte],Sub,Ref) :-
        findall(C,(member(C,Usable);member(C,Soporte);member(C,Sub)),
                S),
        prueba([N*H*[]|S],Ref), !.
d_resol_anotada(Usable,[C|Soporte],Sub,Ref) :-
        escribe_actual(C),
        d_resolventes(C,Usable,S1),
        procesa(Usable,Soporte,S1,S2),
        ( memberchk(_*_*[],S2) ->
            append(S2,Soporte,Soporte1),
            d_resol_anotada([C|Usable],Soporte1,Sub,Ref)
        ; % \+ memberchk(_*_*[],S2) ->
            elimina_subsumidas(S2,[C|Usable],Sub,Usable1,Sub1),
            elimina_subsumidas(S2,Soporte,Sub1,Soporte1,Sub2),
            append(Soporte1,S2,Soporte2),
            ordenada_por_peso(Soporte2,Soporte3),
            d_resol_anotada(Usable1,Soporte3,Sub2,Ref)).

/* Construcción de la prueba */
/* prueba(+S,-Dem) se verifica si Dem es la prueba contenida en el con-
   junto de d-cláusulas anotadas S. Para facilitar la identificación de
   los conjuntos que codifican pruebas, la d-cláusula vacía ocupará el
   primer lugar. El resto de d-cláusulas estarán ordenadas */
prueba(S,[N*H*[]|Dem]) :-
        member(N*H*[],S),
        setof(CA,antecesor(S,N*H*[],CA),Dem).

/* antecesor(+S,+CA1,-CA2) se verifica si CA2 es un antecesor de la
   d-cláusula anotada CA1 en el conjunto de d-cláusulas anotadas S */
antecesor(S,CA1,CA2) :-
        padre(S,CA1,CA2).
antecesor(S,CA1,CA2) :-
        padre(S,CA1,CA3),
        antecesor(S,CA3,CA2).

/* padre(+S,+CA1,+CA2) se verifica si CA2 es un padre en S de la
   d-cláusula anotada CA1 */
padre(S,CA1,CA2) :-
```

```
        CA1 = _*H*_,
        member(N,H),
        CA2 = N*_*_,
        member(CA2,S).

/* escribe_actual(N*P*C) incrementa el número de d-cláusulas analizadas
   y escribe la d-cláusula anotada N*P*C */
escribe_actual(N*P*C) :-
        incrementa_global(n_clausulas_analizadas,M),
        format('~N~n   d-cláusula actual #~w: ~w ~w ~w~n',[M,N,P,C]).

/* Cálculo de d-resolventes anotados */
/* d_resolventes(+C,+S1,-S2) se verifica si S2 es el conjunto formado
   por todos los d-resolventes de C con d-cláusulas de S1 */
d_resolventes(C,S1,S2) :-
        findall(C2,(member(C1,S1),d_resolvente(C,C1,C2)),S2).

/* d_resolvente(+C1,+C2,-C3) se verifica si C3 es un d-resolvente de
   las d-cláusulas anotadas C1 y C2 */
d_resolvente(N1*_*C1,N2*_*C2,_*[N1,N2]*C) :-
        member(L1,C1),
        complementario(L1,L2),
        member(L2,C2),
        delete(C1,L1,C1P),
        delete(C2,L2,C2P),
        append(C1P,C2P,C3),
        list_to_set(C3,C).

/* Procesamiento de d-resolventes */
/* procesa(+Usable,+Soporte,+D_resolventes,-Retenidas) se verifica si
   Retenidas son las d-cláusulas de D_resolventes retenidas al proce-
   sarlas con Usable y Soporte (además escribe mensajes del procesa-
   miento) */
procesa(_,_,[],[]).
procesa(Usable,Soporte,[C|S1],S2) :-
        escribe_dresolvente(C),
        ( (subsumida(C,Usable);
              subsumida(C,Soporte);
              es_dclausula_contradictoria(C)) ->
            procesa(Usable,Soporte,S1,S2)
        ; % C es retenida ->
```

```
            numera(C,C1),
            ( d_resol_unitaria(C1,Usable,Soporte,C2) ->
                S2 = [C2,C1]
            ; % C1 no es unitaria con complementaria en Usable o
              % Soporte ->
                eliminacion_unitaria(Usable,Soporte,C1,C2),
                escribe_retenida(C2),
                ( C2 = N*H*[] ->
                  format('~N~n   -----> D-CLÁUSULA VACÍA: ~w ~w []',
                         [N,H]),
                  S2 = [C2]
                ; % C2 no es la cláusula vacía ->
                    procesa(Usable,[C2|Soporte],S1,S3),
                    S2 = [C2|S3]))).

/* escribe_dresolvente(+C) escribe el d-resolvente procesado C */
escribe_dresolvente(C) :-
        C = _*P*C1,
        format('~N     0 ~w ~w~n',[P,C1]).

/* Eliminación de d-cláusulas subsumidas */
/* subsumida(+C,+S) se verifica si existe una d-cláusula del conjunto S
   que subsume a la d-cláusula C (en cuyo caso escribe un mensaje indi-
   cativo) */
subsumida(C,S) :-
        member(D,S),
        subsume(D,C),
        D = N*_*_,
        format('~N       Subsumida por ~w~n',[N]).

/* subsume(+C,+D) se verifica si la d-cláusula C subsume a D */
subsume(_*_*C1,_*_*C2) :-
        subset(C1,C2).

/* Eliminación de d-cláusulas contradictorias */
/* es_dclausula_contradictoria(+C) se verifica si la d-cláusula anotada
   C es una contradicción (no satisfactible) */
es_dclausula_contradictoria(_*_*C) :-
        member(-A,C),
        memberchk(A,C).
```

```
/* elimina_contradicciones(+C1,?C2) se verifica si C2 es el conjunto de
   d-cláusulas obtenido tras eliminar del conjunto de d-cláusulas C1
   todas las d-cláusulas contradictorias. Aquí, las d-cláusulas no
   están aún anotadas. Cuando trata cada d-cláusula, la imprime con un
   mensaje que indica la satisfactibilidad de la misma */
elimina_contradicciones([],[]).
elimina_contradicciones([C1|R],ER) :-
        member(-A,C1),memberchk(A,C1),!,
        format('~N     Contradictoria: ~w~n',[C1]),
        elimina_contradicciones(R,ER).
elimina_contradicciones([C1|R],[C1|ER]) :-
        format('~N     Satisfactible:   ~w~n',[C1]),
        elimina_contradicciones(R,ER).

/* Numeración de d-cláusulas retenidas */
/* numera(C1,C2) se verifica si incrementa el número de d-cláusulas re-
   tenidas y C2 se obtiene numerando la d-cláusula retenida C1 */
numera(_*H*C,N*H*C) :-
        incrementa_global(n_clausulas_retenidas,N).

/* Procesamiento de d-cláusulas unitarias */
/* d_resol_unitaria(+C1,+U,+S,-C2) se verifica si C1 es una d-cláusula
   unitaria cuya complementaria C pertenece a U ó S y S2 es la d-resol-
   vente de C1 y C (además, escribe un mensaje indicativo) */
d_resol_unitaria(N1*H1*[L1],Usa,Sop,M*[N1,N2]*[]) :-
        complementario(L1,L2),
        ( memberchk(N2*_H*[L2],Usa)
        ; memberchk(N2*_H*[L2],Sop) ),
        escribe_retenida(N1*H1*[L1]),
        M is N1+1,
        format('~N~n   ----> D-RESOLUCIÓN UNITARIA ~w [~w,~w] []~n',
               [M,N1,N2]).

/* escribe_retenida(+C) escribe la d-cláusula retenida C */
escribe_retenida(N*H*C) :-
        format('~N    ** RETENIDA: ~w ~w ~w',[N,H,C]).

/* Eliminación unitaria */
/* eliminacion_unitaria(+Usable,+Soporte,+C1,+C2) se verifica si C2 es
   el resultado de eliminar los literales de la d-cláusula anotada C1
   tales que la d-cláusula unitaria complementaria está en Usable o en
```

```
   Soporte y actualizar la historia de C1 con los números de las
   d-cláusulas unitarias usadas en la eliminación */
eliminacion_unitaria(_U,_S,N*H*[],N*H*[]).
eliminacion_unitaria(U,S,N*H*[L|C],N*H1*C1) :-
        complementario(L,L1),
        (member(M*_*[L1],U) ; member(M*_*[L1],S)), !,
        append(H,[M],H2),
        eliminacion_unitaria(U,S,N*H2*C,N*H1*C1).
eliminacion_unitaria(U,S,N*H*[L|C],N*H1*[L|C1]) :-
        eliminacion_unitaria(U,S,N*H*C,N*H1*C1).

/* Subsunción hacia atrás */
/* elimina_subsumidas(+S,+S1,+T1,-S2,-T2) se verifica si S2 es el con-
   junto de d-cláusulas de S1 no subsumidas por d-cláusulas de S y T2
   es la unión de T1 y las d-cláusulas de S1 subsumidas por d-cláusu-
   las de S (además, escribe el mensaje correspondiente) */
elimina_subsumidas(_,[],T,[],T).
elimina_subsumidas(S,[N1*H1*C1|S1],T1,S2,[N1*H1*C1|T2]) :-
        member(N*_*C,S),
        subsume(N*_*C,N1*_*C1), !,
        format('~N     ~w subsume a ~w~n',[N,N1]),
        elimina_subsumidas(S,S1,T1,S2,T2).
elimina_subsumidas(S,[C1|S1],T1,[C1|S2],T2) :-
        elimina_subsumidas(S,S1,T1,S2,T2).

/* elimina_subsumidas(+S,+S1,+T1,-S2,-T2) se verifica si S2 es el con-
   junto de d-cláusulas de S1 no subsumidas por d-cláusulas de S y T2
   es la unión de T1 y S2 (escribe el mensaje correspondiente) */
elimina_subsumidas_var(_,[],T,[],T).
elimina_subsumidas_var(S,[N1*_*C1|S1],T1,S2,T2) :-
        member(N*_*C,S),
        subsume(N*_*C,N1*_*C1), !,
        format('~N     ~w subsume a ~w~n',[N,N1]),
        elimina_subsumidas_var(S,S1,T1,S2,T2).
elimina_subsumidas_var(S,[C1|S1],T1,[C1|S2],[C1|T2]) :-
        elimina_subsumidas_var(S,S1,T1,S2,T2).

/* Escritura de la prueba */
/* escribe_prueba(+Dem) escribe la prueba contenida en Dem. Aunque la
   d-cláusula vacía aparece en primer lugar, a efectos de la identifi-
   cación de la lista como prueba, se escribe al final */
```

```
escribe_prueba([LineaVacia|Dem]) :-
        format('~N~n  -------- DEMOSTRACIÓN --------~n',[]),
        checklist(escribe_linea,Dem),
        escribe_linea(LineaVacia),
        format('~N  --- fin de la demostración ---~n',[]).
escribe_linea(N*H*C) :-
        format('~N     ~w ~w ~w',[N,H,C]).

/* Escritura de conjuntos de d-cláusulas */
/* muestra_usable(+F) se emplea para mostrar el conjunto de d-cláusulas
   usables F. En primer lugar, las ordena. A continuación, las va mos-
   trando, una por linea */
muestra_usable(F) :-
        sort(F,FS),
        muestra_usable_aux(FS).
muestra_usable_aux([]).
muestra_usable_aux([U|R]):-
        escribe_linea(U),
        muestra_usable_aux(R).

/* Transformación a forma d-clausal */
/* d_clausulas(+F,?S) se verifica si S es un conjunto de d-cláusulas
   equivalente a la fórmula F */
d_clausulas(F,S) :-
        fnd(F,F1),
        d_clausulas_FND(F1,S).

/* Forma normal negativa */
/* fnn(+F,?G) se verifica si G es una forma normal negativa de la
   fórmula F */
fnn(F,G) :-
        elimina_equivalencias(F,F1),
        elimina_implicaciones(F1,F2),
        interioriza_negaciones(F2,G).

/* elimina_equivalencias(+F,?G) se verifica si G es la fórmula
   obtenida eliminando las equivalencias de la fórmula F */
elimina_equivalencias(A <=> B,(A1 & B1) v (-A1 & -B1)) :- !,
        elimina_equivalencias(A,A1),
        elimina_equivalencias(B,B1).
elimina_equivalencias(A,B):-
```

```
        A =.. [Op|L1], !,
        maplist(elimina_equivalencias,L1,L2),
        B =.. [Op|L2].
elimina_equivalencias(A,A).

/* elimina_implicaciones(+F,?G) se verifica si G es la fórmula
   obtenida eliminando las implicaciones de la fórmula F */
elimina_implicaciones(A => B, -A1 v B1) :- !,
        elimina_implicaciones(A,A1),
        elimina_implicaciones(B,B1).
elimina_implicaciones(A,B) :-
        A =.. [Op|L1], !,
        maplist(elimina_implicaciones,L1,L2),
        B =.. [Op|L2].
elimina_implicaciones(A,A).

/* interioriza_negaciones(+F,?G) se verifica si G es la fórmula obteni-
   da interiorizando las negaciones de la fórmula F (que no tiene => ni
   <=>) de forma que las negaciones se apliquen sólo sobre símbolos
   proposicionales */
interioriza_negaciones(-(A & B), A1 v B1) :- !,
        interioriza_negaciones(-A,A1),
        interioriza_negaciones(-B,B1).
interioriza_negaciones(-(A v B), A1 & B1) :- !,
        interioriza_negaciones(-A,A1),
        interioriza_negaciones(-B,B1).
interioriza_negaciones((-(-A)), A1) :- !,
        interioriza_negaciones(A,A1).
interioriza_negaciones(A,B) :-
        A =.. [Op|L1], !,
        maplist(interioriza_negaciones,L1,L2),
        B =.. [Op|L2].
interioriza_negaciones(A,A).

/* Forma normal disyuntiva */
/* fnd(+F,?G) se verifica si G es una forma normal disyuntiva de la
   fórmula F */
fnd(F,G) :-
        fnn(F,F1),
        interioriza_conjunciones(F1,G).
```

```
/* interioriza_conjunciones(+F,?G) se verifica si G es la fórmula
   obtenida desde F, interiorizando las conjunciones mediante la
   propiedad distributiva */
interioriza_conjunciones(A & (B v C), AB v AC) :- !,
        interioriza_conjunciones(A & B, AB),
        interioriza_conjunciones(A & C, AC).
interioriza_conjunciones((A v B) & C, AC v BC) :- !,
        interioriza_conjunciones(A & C, AC),
        interioriza_conjunciones(B & C, BC).
interioriza_conjunciones(A, B) :-
        A =.. [Op|L],
        maplist(interioriza_conjunciones,L,L1),
        ( L1 = L ->
          B =.. [Op|L1]
        ; % not(L1 = L) ->
          B1 =.. [Op|L1],
          interioriza_conjunciones(B1,B)).
interioriza_conjunciones(A,A).

/* d_clausula(+F,-C) se verifica si C es una d-cláusula equivalente
   a la conjunción elemental F */
d_clausula(F1 & F2, S) :- !,
        d_clausula(F1, S1),
        d_clausula(F2, S2),
        append(S1, S2, S3),
        sort(S3, S).
d_clausula(L,[L]).

/* d_clausulas_FND(+F,?S) se verifica si S es un conjunto de d-cláu-
   sulas equivalente a la fórmula en forma normal disyuntiva F */
d_clausulas_FND(A1 v A2, S) :- !,
        d_clausulas_FND(A1, S1),
        d_clausulas_FND(A2, S2),
        union(S1, S2, S).
d_clausulas_FND(A,[S]) :-
        d_clausula(A, S).

/* Predicados auxiliares */
/* busca(+L,+Lits,-R) tiene éxito si, siendo L un literal y Lits una
   lista de literales, Lits contiene o bien L (unificando R con 'i'
   de 'idéntico', o su complementario (en cuyo caso R será 'n' de 'ne-
```

```
   gación'). En caso de que Lits no contenga ni L ni su complementario,
   falla */
busca(L,[L|_],i):-!.
busca(-L,[L|_],n):-!.
busca(L,[-L|_],n):-!.
busca(L,[_|R],X):-
        busca(L,R,X).

/* medidas(-Esp,-Inf) devuelve la memoria (Esp) e inferencias (Inf)
   realizadas. Para el tiempo, activa un contador */
medidas(Esp,Inf) :-
        statistics(runtime,[_,_]),
        statistics(globalused,Esp),
        statistics(inferences,Inf).

/* comienzo(+Teo,+Obs) emite un mensaje de comienzo de la resolución
   de un problema abductivo para la teoría Teo y la observación Obs */
comienzo(Teo,Obs) :-
        inicia,
        format('~N~nPROBLEMA ABDUCTIVO:~n~n',[]),
        format('~N   Teoría:       ~w~n',[Teo]),
        format('~N   Observación:  ~w~n',[Obs]).

/* terminar(+Esp,+Inf) devuelve información de tiempo, espacio e infe-
   rencias de la ejecución del programa, teniendo en cuenta que al co-
   mienzo de la ejecución los valores consumidos eran Esp e Inf. Para
   el tiempo, se activó el contador 'runtime' */
terminar(Esp,Inf) :-
        statistics(runtime,[_,Tiempo]),
        statistics(globalused,Esp2),
        statistics(inferences,Inf2),
        Espacio is Esp2 - Esp,
        Inferencias is Inf2 - Inf,
        format('~N~nRECURSOS EMPLEADOS:~n',[]),
        format('~N   Tiempo (milisegundos):  ~w',[Tiempo]),
        format('~N   Espacio (bytes):        ~w',[Espacio]),
        format('~N   Inferencias:            ~w',[Inferencias]).

/* complementario(+Lit,-LitCompl) se verifica si Lit es un literal y
   LitCompl su complementario */
complementario(-A,A) :-!.
```

```
complementario(A,-A).

/* literal(+C) se verifica si C es un literal */
literal(-C) :-!,
        atom(C).
literal(C) :-
        atom(C).

/* muestra_conjunto(+Conj) sirve para imprimir en pantalla línea a
   línea cada uno de los elementos del conjunto Conj */
muestra_conjunto([]).
muestra_conjunto([C1|Resto]) :-
        format('~N      ~w',[C1]),
        muestra_conjunto(Resto).
```

Apéndice C

Comparación de eficiencia

A continuación mostramos varias familias de problemas abductivos, de distinta complejidad, que empleamos para evaluar los diferentes sistemas que recogemos en este trabajo. El predicado con el que están definidos es `problema_abd(?Nom, ?Teo, ?Obs)`, donde `Nom` es el nombre del problema, `Teo` una fórmula que representa la teoría, y `Obs` la observación.

La primera serie de problemas es la que llamamos $implic(n)$, que se caracteriza porque las teorías consisten en una serie de implicaciones entre literales que se encadenan de la forma

$$(t_1 \rightarrow t_2) \wedge (t_2 \rightarrow t_3) \wedge \ldots \wedge (t_{n-1} \rightarrow t_n)$$

y la observación es el literal n. En estos problemas, las soluciones abductivas explicativas son los literales t_i, $1 \leq i < n$. Usamos tres instancias:

```
problema_abd(implic(5),(t1 => t2) & (t2 => t3) & (t3 => t4) &
            (t4 => t5), t5).
problema_abd(implic(10),(t1 => t2) & (t2 => t3) & (t3 => t4) &
            (t4 => t5) & (t5 => t6) & (t6 => t7) & (t7 => t8) &
            (t8 => t9) & (t9 => t10), t10).
problema_abd(implic(15),(t1 => t2) & (t2 => t3) & (t3 => t4) &
            (t4 => t5) & (t5 => t6) & (t6 => t7) & (t7 => t8) &
            (t8 => t9) & (t9 => t10) & (t10 => t11) & (t11 => t12)
            & (t12 => t13) & (t13 => t14) & (t14 => t15),t15).
```

En la serie $conj(n)$, de mayor complejidad, el problema será determinar cuáles, de entre todas las explicaciones posibles —que siempre serán conjunciones de literales— son las minimales. En todos los casos la observación es m, y las teorías siempre contienen $p \wedge q \rightarrow m$, siendo $p \wedge q$ la única abducción explicativa minimal. Pero además, las teorías contienen otra serie de implicaciones de la forma

$$p \wedge q \wedge l_3 \wedge \ldots \wedge l_n \rightarrow n$$

que producen explicaciones no minimales. Cuanto mayores son las instancias de $conj(n)$, más implicaciones hay —un total de $n-1$— con un mayor número de literales —de 2 a n—. Las instancias que usamos en las pruebas son las siguientes:

```
problema_abd(conj(3),(p & q => m) & (p & q & r => m),m).
problema_abd(conj(4),(p & q => m) & (p & q & r => m) &
            (p & q & r & s => m),m).
problema_abd(conj(5),(p & q => m) & (p & q & r => m) &
            (p & q & r & s => m) & (p & q & r & s & t => m),m).
problema_abd(conj(6),(p & q => m) & (p & q & r => m) &
            (p & q & r & s => m) & (p & q & r & s & t => m) &
            (p & q & r & s & t & u => m),m).
problema_abd(conj(7),(p & q => m) & (p & q & r => m) &
            (p & q & r & s => m) & (p & q & r & s & t => m) &
            (p & q & r & s & t & u => m) &
            (p & q & r & s & t & u & v => m),m).
```

En cuanto a la serie $anidado(n)$, su complejidad también reside en la elección de las conjunciones que constituyen abducciones explicativas minimales. Los problemas tienen siempre el literal m como observación y una teoría que consiste en una conjunción de las n primeras implicaciones de entre las siguientes

$$\begin{aligned} a_1 &\rightarrow m \\ a_2 \wedge a_3 &\rightarrow a_1 \\ a_4 \wedge a_5 &\rightarrow a_2 \\ a_6 \wedge a_7 &\rightarrow a_4 \\ a_8 \wedge a_9 &\rightarrow a_6 \\ &\vdots \end{aligned}$$

de forma que para cada literal a_i que forma parte de una explicación puede haber una conjunción de otros dos literales a_j y a_k que a su vez lo explican. También a_j o a_k podrán ser explicados por otros literales, y así sucesivamente, por lo que se trata de determinar cuáles de entre las conjunciones de literales son explicaciones minimales. Usamos las siguientes instancias:

```
problema_abd(anidado(3),(a1 => m) & (a2 & a3 => a1) &
            (a4 & a5 => a2),m).
problema_abd(anidado(4),(a1 => m) & (a2 & a3 => a1) & (a4 & a5 => a2) &
            (a6 & a7 => a4),m).
problema_abd(anidado(5),(a1 => m) & (a2 & a3 => a1) & (a4 & a5 => a2) &
            (a6 & a7 => a4) & (a8 & a9 => a6),m).
```

Los problemas $incon(n)$, por último, tienen el literal m como observación y una teoría que se obtiene por la conjunción de las siguientes implicaciones:

$$\begin{aligned} t &\rightarrow m \\ d \wedge s &\rightarrow m \\ d &\rightarrow f_1 \vee \ldots \vee f_n \\ s &\rightarrow \neg(f_1 \vee \ldots \vee f_n) \end{aligned}$$

con lo que hay siempre una explicación consistente, t, y otra inconsistente, $d \wedge s$. Sin embargo, cuanto mayores valores toma n, más difícil es comprobar la inconsistencia de esta última, al ser mayor la disyunción $f_1 \vee \ldots \vee f_n$. Usamos las siguientes instancias:

```
problema_abd(incon(1),(d => f1) & (s => -f1) &
            (d & s => m) & (t => m),m).
problema_abd(incon(3),(d => (f1 v f2 v f3)) & (s => -(f1 v f2 v f3)) &
            (d & s => m) & (t => m),m).
problema_abd(incon(4),(d => (f1 v f2 v f3 v f4)) &
            (s => -(f1 v f2 v f3 v f4)) & (d & s => m) &
            (t => m),m).
problema_abd(incon(5),(d => (f1 v f2 v f3 v f4 v f5)) &
            (s => -(f1 v f2 v f3 v f4 v f5)) & (d & s => m) &
            (t => m),m).
```

Con cada uno de los programas que, a lo largo de este trabajo, hemos creado para resolver problemas abductivos proposicionales, se han tratado de resolver todos los problemas abductivos anteriormente presentados. Ello se ha llevado a cabo usando SWI-Prolog 5.4.3, y dando a cada una de las pilas —*local*, *global* y *tail*— un espacio de 25 megabytes. La máquina usada es un IBM NetVista 6269-P2G, equipada con un procesador Intel Pentium III a 800 MHz. En cuanto al sistema operativo, se empleó SuSE Linux 9.1 con el kernel 2.6. Los resultados se recogen en los cuadros que aparecen en las páginas 79 —sistema de Aliseda—, 116 —primera versión que usa δ-resolución—, 116 —segunda versión que usa δ-resolución— y 116 —versión que integra δ-resolución con tablas semánticas—. Además de los cuadros, se comentan los resultados obtenidos para cada sistema, en comparación con los demás.

Apéndice D

Implementación de la δ-resolución en primer orden

```
/* Fichero: dresol_1ord.pl  Abducción mediante d-resolución en primer
   orden */

:-use_module(libreria,[medidas/2,comienzo/2,terminar/2,inicia/0,
                       asigna_global/2,anotado/2,incrementa_global/2,
                       valor_global/2,prueba/2,antecesor/3,padre/3,
                       escribe_actual/1,d_resolventes/3,subsumida/2,
                       complementario/2,escribe_dresolvente/1,
                       numera/2,escribe_retenida/1,escribe_prueba/1,
                       elimina_subsumidas/5,elimina_subsumidas_var/5,
                       muestra_usable/1,escribe_linea/1]).

/* Operadores lógicos */
:-      op(400,fy,-),
        op(500,xfy,&),
        op(600,xfy,v),
        op(650,xfy,=>),
        op(700,xfy,<=>).

/* abduce(+Teo,+Obs) se verifica si Teo es una fórmula de primer orden,
   correspondiente a la teoría, y Obs otra, correspondiente a la obser-
   vación. Entonces procede a la resolución del problema abductivo a
   través de cuatro pasos:
```

```
   - ANÁLISIS DE LA TEORÍA. Sirve para determinar si:
   * La teoría en inconsistente (se detiene).
   * En otro caso, guarda el último usable y va a,
   - ANÁLISIS DE LA OBSERVACIÓN. Donde determina si:
   * La observación es universalmente válida (se detiene).
   * En otro caso, guarda el último usable y va a,
   - BÚSQUEDA DE REFUTACIONES. Determina si:
   * Encuentra una refutación (puede haberla y no encontrarla).
   * En otro caso, determina el soporte para,
   - BÚSQUEDA DE EXPLICACIONES. Realiza entonces la búsqueda de solu-
   ciones abductivas. Pueden ocurrir tres cosas:
   * La observación se deriva de la teoría (avisa y para).
   * Devuelve supuestas abducciones explicativas (para).
   * No hay abducciones explicativas (lo avisa y para).
   NOTA: El procedimiento varía respecto del proposicional al no ser
   posible en primer orden determinar, por d-resolución, la insatisfac-
   tibilidad de un cierto conjunto de d-cláusulas.
   Al final de la ejecución, devuelve información sobre la memoria y
   tiempo empleados, así como del número de inferencias */
abduce(Teo,Obs) :-
        medidas(Esp,Inf),
        comienzo(Teo,Obs),
        analiza_teo(Teo,UsTeo),
        ( UsTeo = fin ->
            terminar(Esp,Inf)
        ; % Teoría contingente
            analiza_obs(Obs,UsObs),
            ( UsObs = fin ->
                terminar(Esp,Inf)
            ; % Teoría y hecho contingentes
                busca_ref(UsTeo,UsObs,Sop),
                ( Sop = fin ->
                    terminar(Esp,Inf)
                ; % Pueden hacerse explicaciones
                    busca_abd(Sop),
                    terminar(Esp,Inf)))).

/* analiza_teo(+Teo,-UsTeo) realiza el primer paso de la resolución de
   un problema abductivo, el análisis de la teoría. Procede de la si-
   guiente manera:
   1.- Obtiene la forma d-clausal de -Teo, elimina las d-cláusulas
```

```
   contradictorias, y forma el conjunto ClSatTeo. Entonces, si
   ClSatTeo es vacío, es que -Teo es no satisfactible, por lo que Teo
   es universalmente válida. En este caso, avisa con un mensaje, de-
   tiene la búsqueda y unifica UsTeo con 'fin', lo que indica al
   predicado principal que la búsqueda ha terminado. En otro caso,
   2.- Aplica d-resolución, tomando como soporte ClSatTeo. Si se llega
   a la d-cláusula vacía, entonces es que -Teo es válida, por lo que
   Teo es no satisfactible. En tal caso, avisa con un mensaje y se de-
   tiene (unificando también UsTeo con 'fin'). Si no se llegó a la
   d-cláusula vacía tras saturar la búsqueda guarda en UsTeo el últi-
   mo usable de la búsqueda */
analiza_teo(Teo,UsTeo) :-
        format('~N~n1. ANÁLISIS DE LA TEORÍA:~n',[]),
        d_clausulas(-Teo,ClTeo),
        format('~N~n   d-cláusulas de la teoría:~n~n',[]),
        elimina_contradicciones(ClTeo,ClSatTeo),
        ( ClSatTeo = [] ->
            format('~N~n   *** TEORÍA VÁLIDA ***~n',[]),
            UsTeo = fin
        ; % ClSatTeo no es vacío.
            format('~N~n   Soporte de la teoría:',[]),
            anotado(ClSatTeo,SopTeo),
            ordenada_por_peso(SopTeo,SopTeo1),
            d_resol_anotada([],SopTeo1,[],UsTeoProv),
            ( UsTeoProv = [_*_*[]|_] ->
                format('~N~n   *** TEORÍA INCONSISTENTE ***~n',[]),
                escribe_prueba(UsTeoProv),
                UsTeo = fin
            ; % No se encuentra d-cláusula vacía
                format('~N~n   Último usable de la teoría:~n',[]),
                muestra_usable(UsTeoProv),
                UsTeoProv = UsTeo)).

/* analiza_obs(+Obs,-UsObs) realiza el segundo paso de la resolución de
   un problema abductivo, el análisis de la observación. Procede de la
   siguiente manera:
   1.- Obtiene la forma d-clausal de Obs, elimina las d-cláusulas
   contradictorias, y forma el conjunto ClSatObs. Entonces, si
   ClSatObs es vacío, es que Obs es no satisfactible. En este caso,
   avisa con un mensaje, detiene la búsqueda y unifica UsObs con
   'fin', lo que indica al predicado principal que la búsqueda ha ter-
```

```
   minado. En otro caso, va al paso siguiente.
   2.- Aplica d-resolución, tomando como soporte ClSatObs. Si se llega
   a la d-cláusula vacía, entonces es que Obs es válida. En tal caso,
   avisa con un mensaje y se detiene (unificando también UsObs con
   'fin'). Si no se llegó a la d-cláusula vacía tras saturar la bús-
   queda guarda en UsObs el último usable */
analiza_obs(Obs,UsObs) :-
        format('~N~n2. ANÁLISIS DE LA OBSERVACIÓN:~n',[]),
        d_clausulas(Obs,ClObs),
        format('~N~n   d-cláusulas de la observación:~n~n',[]),
        elimina_contradicciones(ClObs,ClSatObs),
        ( ClSatObs = [] ->
           format('~N~n   *** OBSERVACIÓN INCONSISTENTE ***~n',[]),
           UsObs = fin
        ; % Hay d-cláusulas no contradictorias
            format('~N~n   Soporte de la observación:',[]),
            anotado(ClSatObs,SopObs),
            ordenada_por_peso(SopObs,SopObs1),
            d_resol_anotada([],SopObs1,[],UsObsProv),
            ( UsObsProv = [_*_*[]|_] ->
                format('~N~n   *** OBSERVACIÓN VÁLIDA ***~n',[]),
                escribe_prueba(UsObsProv),
                UsObs = fin
            ; % No se encuentra d-cláusula vacía
              format('~N~n   Último usable de la observación:~n',[]),
              muestra_usable(UsObsProv),
              UsObsProv = UsObs)).

/* busca_ref(+UsTeo,+UsObs,-Sop) realiza el tercer paso de la resolu-
   ción de un problema abductivo, la búsqueda de refutaciones, es de-
   cir, comprueba si la observación contradice la teoría. Para ello,
   comprueba si cada d-cláusula de UsObs está subsumida por una d-cláu-
   sula de UsTeo. Si ello es así, es que la negación de la observación
   se deriva de la teoría, por lo que Obs es inconsistente con Teo. En
   este caso, avisa con un mensaje y detiene la búsqueda (ahora, uni-
   fica Sop con 'fin'). En otro caso, es posible comenzar la búsqueda
   de explicaciones, tomando como soporte, Sop, conjunto unión de UsTeo
   con las d-cláusulas de UsObs no subsumidas por d-cláusulas de UsTeo.
   Podría haber refutación pero que el programa no la encuentre */
busca_ref(UsTeo,UsObs,Sop) :-
        format('~N~n3. BÚSQUEDA DE REFUTACIÓN:~n',[]),
```

```
        elimina_subsumidas_var(UsTeo,UsObs,UsTeo,NoSubs,SopProv),
        ( NoSubs = [] ->
            format('~N~n   *** LA OBSERVACIÓN REFUTA LA TEORÍA ***',
                   []),
            Sop = fin
        ; % No hay refutación
            SopProv = Sop,
            format('~N~n   *** NO SE ENCUENTRA REFUTACIÓN ***~n',[]),
            format('~N~n   Soporte para búsqueda abductiva:',[]),
            muestra_usable(Sop)).

/* busca_abd(+Sop) realiza el cuarto y último paso de la búsqueda de
   soluciones abductivas. Toma Sop, el soporte de la búsqueda, y
   aplica d-resolución. Si se llega a la d-cláusula vacía, entonces
   la observación es consecuencia de la teoría. En tal caso avisa y
   se para la búsqueda. En otro caso, tras saturar la búsqueda, toma
   todas las d-cláusulas que forman parte del conjunto usable final
   pero no estaban en el soporte inicial, formando el conjunto Explic
   de abducciones explicativas y minimales. Si Explic es vacío, es
   que no existen abducciones explicativas, de lo que avisa con un
   mensaje. En otro caso, devuelve los elementos de Explic */
busca_abd(Sop) :-
        format('~N~n4. BÚSQUEDA DE EXPLICACIONES:~n',[]),
        ordenada_por_peso(Sop,Sop1),
        valor_global(n_clausulas_retenidas,Min1),
        d_resol_anotada([],Sop1,[],Ref),
        ( Ref = [_*_*[]|_] ->
            format('~N~n   *** OBSERVACIÓN EXPLICADA POR TEORÍA ***',
                   []),
            escribe_prueba(Ref)
        ; % la observación no se deriva de la teoría
            valor_global(n_clausulas_retenidas,Max),
            Min is Min1+1,
            findall(N*H*C,(between(Min,Max,N),
                           memberchk(N*H*C,Ref)),Explic),
            ( Explic = [] ->
              format('~N~n   *** NO HAY ABDUCCIONES EXPLICATIVAS ***',
                     [])
            ; % hay abducciones explicativas
            format('~N~n     Pueden ser abducciones explicativas:',[]),
            format('~N     (comprobar tras deseskolemizar)',[]),
```

```
            muestra_usable(Explic))).

/* ordenada_por_peso(+S1,-S2) se verifica si S2 es el conjunto de las
   d-cláusulas anotadas del conjunto S1 ordenadas por peso */
ordenada_por_peso(S1,S2) :-
        findall(L1-N1*P1*C1,
                (member(N1*P1*C1,S1),
                    peso_cl(C1,L1)),
                S1a),
        keysort(S1a,S2a),
        findall(CA2,member(_-CA2,S2a),S2).
peso_cl([],0).
peso_cl([L1|Lits],N) :-
        peso(L1,P1),
        peso_cl(Lits,P2),
        N is P1 + P2.
peso(Lit,P) :-
        ( (var(Lit); functor(Lit,_,0)) ->
            P = 1;
            Lit =.. [_|As],
            peso_aux(As,PA),
            P is 1+PA).
peso_aux([],0).
peso_aux([A1|R],Peso):-
        peso(A1,P1),
        peso_aux(R,P2),
        Peso is P1+P2.

/* d_resol_anotada(+Usable,+Soporte,+Subsumidas,-Ref) opera realizando
   una búsqueda por d-resolución de la d-cláusula vacía en el conjunto
   de d-cláusulas formado por Usable y Soporte, siendo Subsumidas las
   d-cláusulas que aparecieron pero fueron subsumidas. Además, va es-
   cribiendo la búsqueda. Si llega a la d-cláusula vacía, Ref contiene
   su prueba, y será un conjunto de d-cláusulas con la vacía en primer
   lugar. En otro caso, Ref será el último usable al que se llegó */
d_resol_anotada(Usable,[],_,Usable).
d_resol_anotada(Usable,[N*H*[]|Soporte],Sub,Ref) :-
        findall(C,(member(C,Usable);member(C,Soporte);member(C,Sub)),
                S),
        prueba([N*H*[]|S],Ref), !.
d_resol_anotada(Usable,[C|Soporte],Sub,Ref) :-
```

```
        escribe_actual(C),
        d_resolventes(C,Usable,S1),
        procesa(Usable,Soporte,S1,S2),
        ( memberchk(_*_*[],S2) ->
            append(S2,Soporte,Soporte1),
            d_resol_anotada([C|Usable],Soporte1,Sub,Ref)
        ; % \+ memberchk(_*_*[],S2) ->
            elimina_subsumidas(S2,[C|Usable],Sub,Usable1,Sub1),
            elimina_subsumidas(S2,Soporte,Sub1,Soporte1,Sub2),
            append(Soporte1,S2,Soporte2),
            factores_binarios_conj(S2,Factores),
            ( Factores = [_|_] -> % hay algún factor binario...
                procesa(Usable1,Soporte2,Factores,Factores2),
                (memberchk(_*_*[],Factores2) ->
                  append(Factores2,Soporte2,Soporte3),
                  d_resol_anotada(Usable1,Soporte3,Sub2,Ref)
                 ; % \+ memberchk(_*_*[],S2) ->
                 elimina_subsumidas(Factores2,Usable1,Sub2,Usable2,Sub3),
                 elimina_subsumidas(Factores2,Soporte3,Sub3,Soporte4,Sub4),
                          append(Soporte4,Factores2,Soporte5),
                  ordena_por_peso(Soporte5,Soporte6),
                  d_resol_anotada(Usable2,Soporte6,Sub4,Ref))
            ; % no hay factores:
            ordenada_por_peso(Soporte2,Soporte3),
            d_resol_anotada(Usable1,Soporte3,Sub2,Ref))).

/* d_resolvente(+C1,+C2,-C3) se verifica si C3 es un d-resolvente de
   las d-cláusulas anotadas C1 y C2 */
d_resolvente(N1*_*CA,N2*_*CB,_*[N1,N2]*Resolvente) :-
        copy_term(CA,C1), % renombramiento de variables
        copy_term(CB,C2), % renombramiento de variables
        select(L1,C1,R1),
        ( (L1 = -Neg; -L1 = Neg) ->
            select(L2,C2,R2),
            unify_with_occurs_check(L2,Neg), % unif. correcta
            append(R1,R2,Resolvente)).

/* procesa(+Usable,+Soporte,+D_resolventes,-Retenidas) se verifica si
   Retenidas son las d-cláusulas de D_resolventes retenidas al proce-
   sarlas con Usable y Soporte (escribe mensajes del procesamiento) */
procesa(_,_,[],[]).
```

```
procesa(Usable,Soporte,[C|S1],S2):-
        escribe_dresolvente(C),
        numera(C,C1),
        eliminacion_unitaria(Usable,Soporte,C1,C2),
        factoriza(C2,C3),
        ( (subsumida(C3,Usable);
              subsumida(C3,Soporte);
              es_dclausula_contradictoria(C3)) ->
            procesa(Usable,Soporte,S1,S2)
        ; % C es retenida ->
            escribe_retenida(C3),
            C3 = N*H*Cuerpo,
            ( Cuerpo = [] ->
                format('~N~n   -----> D-CLÁUSULA VACÍA: ~w ~w []',
                         [N,H]),
                S2 = [C3]
            ;
                ( (Cuerpo = [_],
                      d_resol_unitaria(C3,Usable,Soporte,N2*H2*[])) ->
                    format('~N~n   -----> D-CLÁUSULA VACÍA: ~w ~w []',
                           [N2,H2]),
                    S2 = [N2*H2*[]]
                ;
                    procesa(Usable,[C3|Soporte],S1,S3),
                    S2 = [C3|S3]))).

/* Simplificación de factores */
/* factoriza(+ClAnot,-Factor) se verifica si, dada la d-cláusula anota-
   da ClAnot, Factor es el resultado de aplicar simplificación de fac-
   tores */
factoriza(N*H*ClIni,N*[fc|H]*ClFin):-
        copy_term(ClIni,ClControl),
        select(E1,ClIni,Resto),
        select(E2,Resto,_),
        unifica_con(E1,E2),
        subsume(_*_*Resto,_*_*ClControl),!,
        factoriza(_*_*Resto,_*_*ClFin),
        format('~N     ~w ~w ~w~n',[N,[fc|H],ClFin]).
factoriza(E,E).

/* Factores binarios */
```

```
/* factores_binarios_conj(+ConjCls,-Factores) se verifica si dado el
   conjunto de d-cláusulas ConjCls, Factores es el conjunto de factores
   binarios de los elementos de ConjCls */
factores_binarios_conj([C1|Resto],Factores):-
        factores_binarios(C1,Facs1),
        factores_binarios_conj(Resto,Facs2),
        append(Facs1,Facs2,Factores).
factores_binarios_conj([],[]).
factores_binarios(N*_*Cl,Fac):-
        findall(_*[N]*F,es_factor_binario(Cl,[F]),Fac).
es_factor_binario(Cl,Factores) :-
        setof(Fac,factor_binario(Cl,Fac),Factores).
factor_binario(Cl,Fac) :-
        factor(Cl,Fac);
        ( factor(Cl,FP),
            factor_binario(FP,Fac)).
factor(Cl,Factor) :-
        select(Lit1,Cl,FactorP),
        select(Lit2,FactorP,_Resto2),
        unify_with_occurs_check(Lit1,Lit2),
        sort(FactorP,Factor).

/* Eliminación de d-cláusulas subsumidas */
/* subsume(+C,+D) se verifica si la d-cláusula C subsume a D */
subsume(_*_*E1,_*_*E2):-
        copy_term(E1,E4),
        copy_term(E2,E3),
        subsume_aux(E4,E3),
        unifica_con(E3,E2).
subsume_aux([],_).
subsume_aux([E1|Resto],Cl) :-
        member(Lit,Cl),
        unifica_con(E1,Lit),
        subsume_aux(Resto,Cl).

/* unifica_con(+X,+Y) se verifica si hay una sustitución que
   transforma X en Y. Además, aplica dicha sustitución a X */
unifica_con(X,Y) :-
        free_variables(Y,Free1),
        unify_with_occurs_check(X,Y),
        free_variables(Y,Free2),
```

```
        unify_with_occurs_check(Free1,Free2).

/* Eliminación de d-cláusulas contradictorias */
/* es_dclausula_contradictoria(+C) se verifica si la d-cláusula anotada
   C es una contradicción (contiene un literal y su complementario) */
es_dclausula_contradictoria(_*_*Cl) :-
        member(-E1,Cl),
        member(E2,Cl),
        ( E2 = -_ ->
            fail
        ; % E2 es positivo
            E1 == E2).

/* elimina_contradicciones(+C1,?C2) se verifica si C2 es el conjunto de
   d-cláusulas obtenido tras eliminar del conjunto de d-cláusulas C1
   todas las d-cláusulas contradictorias. Aquí, las d-cláusulas no
   están aún anotadas. Cuando encuentra una d-cláusula contradictoria,
   lo avisa. En las que no detecta contradicción, no da ningún mensaje,
   ya que podrían serlo tras la skolemización inversa */
elimina_contradicciones([],[]).
elimina_contradicciones([C1|R],ER) :-
        es_dclausula_contradictoria(_*_*C1),!,
        format('~N     Contradictoria: ~w~n',[C1]),
        elimina_contradicciones(R,ER).
elimina_contradicciones([C1|R],[C1|ER]) :-
        format('~N                     ~w~n',[C1]),
        elimina_contradicciones(R,ER).

/* Procesamiento de d-claúsulas unitarias */
/* d_resol_unitaria(+C1,+U,+S,-C2) se verifica si C1 es una d-cláusula
   unitaria cuya complementaria C pertenece a U ó S y S2 es un d-resol-
   vente de C1 y C (además, escribe un mensaje indicativo) */
d_resol_unitaria(N1*H1*[L1],Usa,Sop,M*[N1,N2]*[]) :-
        complementario(L1,L2),
        ( memberchk(N2*_H*[L3],Usa)
        ; memberchk(N2*_H*[L3],Sop) ),
        copy_term(L3,L4),
        unify_with_occurs_check(L2,L4),
        escribe_retenida(N1*H1*[L1]),
        M is N1+1,
        format('~N~n   ----> D-RESOLUCIÓN UNITARIA ~w [~w,~w] []~n',
```

```
                [M,N1,N2]).

/* Eliminación unitaria */
/* eliminacion_unitaria(+Usable,+Soporte,+C1,+C2) se verifica si C2 es
   el resultado de eliminar los literales de la d-cláusula anotada C1
   tales que la d-cláusula unitaria complementaria está en Usable o
   en Soporte y actualizar la historia de C1 con los números de las
   d-cláusulas unitarias usadas en la eliminación */
eliminacion_unitaria(_U,_S,N*H*[],N*H*[]).
eliminacion_unitaria(U,S,N*H*[L|C],N*H1*C1) :-
        complementario(L,L1),
        (member(M*_*[L2],U) ; member(M*_*[L2],S)),
        copy_term(L2,L3),
        unifica_con(L3,L1),!,
        append(H,[M],H2),
        eliminacion_unitaria(U,S,N*H2*C,N*H1*C1).
eliminacion_unitaria(U,S,N*H*[L|C],N*H1*[L|C1]) :-
        eliminacion_unitaria(U,S,N*H*C,N*H1*C1).

/* Transformación a d-cláusulas */
/* d_clausulas(+F,?S) se verifica si S la forma d-clausal de la
   fórmula F */
d_clausulas(F,Clausulas) :-
        reescr_implic(F,F1),
        negac_dentro(F1,F2),
        forma_prenexa(F2,F3),
        skolemizar_inv(F3,F4),
        forma_normal_disy(F4,F5),
        d_clausulas_fnd(F5,Clausulas).

/* reescr_implic(+F1,?F2) se verifica si F2 es una fórmula de primer
   orden sin implicaciones, equivalente a F1 */
reescr_implic(A,B):-
        reescr_implic([],A,B).
reescr_implic(VL,A => B, -C v D):-!,
        reescr_implic(VL,A,C),
        reescr_implic(VL,B,D).
reescr_implic(VL,A <=> B,(A2 & B2) v (-A3 & -B3)):-!,
        reescr_implic(VL,A,A2),
        reescr_implic(VL,B,B2),
        copy_term((VL,A2,B2),(VL,A3,B3)).
```

```
reescr_implic(VL,A & B, C & D):-!,
        reescr_implic(VL,A,C),
        reescr_implic(VL,B,D).
reescr_implic(VL,A v B, C v D):-!,
        reescr_implic(VL,A,C),
        reescr_implic(VL,B,D).
reescr_implic(VL,-A,-C):-!,
        reescr_implic(VL,A,C).
reescr_implic(VL,all(X,B), all(X,D)):-!,
        reescr_implic([X|VL],B,D).
reescr_implic(VL,ex(X,B), ex(X,D)):-!,
        reescr_implic([X|VL],B,D).
reescr_implic(_,A,A).

/* negac_dentro(+F1,?F2) se verifica si F2 es una fórmula de primer
   orden equivalente a F1, pero donde las negaciones sólo afectan a
   literales */
negac_dentro(-(A & B),C v D):-!,
        negac_dentro(-A,C),
        negac_dentro(-B,D).
negac_dentro(-(A v B),C & D):-!,
        negac_dentro(-A,C),
        negac_dentro(-B,D).
negac_dentro(-(-A),B):-!,
        negac_dentro(A,B).
negac_dentro(-ex(X,A),all(X,B)):-!,
        negac_dentro(-A,B).
negac_dentro(-all(X,A),ex(X,B)):-!,
        negac_dentro(-A,B).
negac_dentro(A & B,C & D):-!,
        negac_dentro(A,C),
        negac_dentro(B,D).
negac_dentro(A v B, C v D):-!,
        negac_dentro(A,C),
        negac_dentro(B,D).
negac_dentro(ex(X,A),ex(X,B)):-!,
        negac_dentro(A,B).
negac_dentro(all(X,A),all(X,B)):-!,
        negac_dentro(A,B).
negac_dentro(A,A).
```

```
/* forma_prenexa(+F1,?F2) se verifica si F2 es una fórmula de primer
   orden equivalente a F1, pero en forma normal prenexa */
forma_prenexa(F,FNP):-
        fnp(F,FNP,B,B).
fnp(all(X,F),all(X,Cuants),V,Cuerpo):-!,
        fnp(F,Cuants,V,Cuerpo).
fnp(ex(X,F),ex(X,Cuants),V,Cuerpo):-!,
        fnp(F,Cuants,V,Cuerpo).
fnp(A & B,Cuants,V,CuerpoA & CuerpoB):-!,
        fnp(A,Cuants,CuantsB,CuerpoA),
        fnp(B,CuantsB,V,CuerpoB).
fnp(A v B,Cuants,V,CuerpoA v CuerpoB):-!,
        fnp(A,Cuants,CuantsB,CuerpoA),
        fnp(B,CuantsB,V,CuerpoB).
fnp(A,V,V,A).

/* skolemizar_inv(+F1,?F2) se verifica si F2 es el resultado de
   sustituir todas las variables cuantificadas universalmente por
   términos de Skolem, y las existenciales por variables libres */
skolemizar_inv(F1,F2):-
        skolemizar_x(F1,[],F2).
skolemizar_x(ex(X,F1),ListaVar,F2):-!,
        skolemizar_x(F1,[X|ListaVar],F2).
skolemizar_x(all(X,F1),ListaVar,F2):-!,
        termino_skolem(X,ListaVar),
        skolemizar_x(F1,ListaVar,F2).
skolemizar_x(F,_,F).
termino_skolem(X,ListaVar):-
        flag(skol,N,N),
        N1 is N+1,
        flag(skol,_,N1),
        name(N,Var),
        name(Functor,[115,107|Var]),
        X =.. [Functor|ListaVar].

/* forma_normal_disy(+F1,?F2) se verifica si F2 es el resultado de
   poner en forma normal disyuntiva F1 */
forma_normal_disy(A,A) :-
        conj_elemental(A).
forma_normal_disy((A v B) & C, D v E ):-!,
        forma_normal_disy(A & C,D),
```

```
        forma_normal_disy(B & C,E).
forma_normal_disy(A & (B v C), D v E ):-!,
        forma_normal_disy(A & B,D),
        forma_normal_disy(A & C,E).
forma_normal_disy(A v B,C v D):-!,
        forma_normal_disy(A,C),
        forma_normal_disy(B,D).
forma_normal_disy(A & B,E):-!,
        forma_normal_disy(A,C),
        forma_normal_disy(B,D),
        forma_normal_disy(C & D,E).

/* conj_elemental(+Conj) se verifica si Conj es una conjunción elemen-
   tal. Conj debe estar en forma normal negativa skolemizada */
conj_elemental(A & B) :-!,
        conj_elemental(A),
        conj_elemental(B).
conj_elemental(_A v _B):-!,
        fail.
conj_elemental(_A):-!.

/* d_clausula(+F,-C) se verifica si C es una d-cláusula equivalente a
   la conjunción elemental F */
d_clausula(F1 & F2, S) :- !,
        d_clausula(F1, S1),
        d_clausula(F2, S2),
        append(S1, S2, S3),
        sort(S3, S).
d_clausula(L,[L]).

/* d_clausulas_fnd(+Form,?Conj) se verifica si Conj es la forma
   d-clausal de la fórmula Form */
d_clausulas_fnd(A1 v A2, S) :- !,
        d_clausulas_fnd(A1, S1),
        d_clausulas_fnd(A2, S2),
        append(S1, S2, S).
d_clausulas_fnd(A,[S]) :-
        d_clausula(A, S).
```

Bibliografía

[AB02] José Antonio Alonso y Joaquín Borrego. *Deducción automática (Vol. 1: Construcción lógica de sistemas lógicos)*. Kronos, Sevilla, 2002.

[Abr99] Paulo Abrantes. "Analogical reasoning and modelling in the sciences". *Foundations of Science*, 4:237–270, 1999.

[Ace04] Juan José Acero. "Los estados mentales de las máquinas de Turing". En Vicente et al. [VFC+04], páginas 296–299.

[Ala04] Mario Alai. "A.I., scientific discovery and realism". *Minds and Machines*, 14:21–42, 2004.

[Ali97] Atocha Aliseda. *Seeking Explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence*. Institute for Logic, Language, and Computation, Amsterdam, 1997.

[Ali98] Atocha Aliseda. "La abducción como cambio epistémico: C.S. Peirce y las teorías epistémicas en inteligencia artificial". *Analogía Filosófica*, XII(1):125–144, 1998.

[AM] José Alfredo Amor y Raymundo Morado. "Demostración Automática". Publicación electrónica en http://minerva.filosoficas.unam.mx/~morado/LogicaHoy/morado.html.

[AMO95] Carlos E. Alchourrón, José M. Méndez, y Raúl Orayen, editores. *Lógica*. Volumen 7 de la Enciclopedia Iberoamericana de Filosofía. Ed. Trotta y CSIC, Madrid, 1995.

[AN] Atocha Aliseda y Ángel Nepomuceno. "Abduction in first order semantic tableaux". Pendiente de publicación.

[Avr93] Arnon Avron. "Gentzen-type systems, resolution and tableaux". *Journal of Automated Reasoning*, 10(2):265–281, 1993.

[BBS01] Patrick Blackburn, Johan Bos y Kristina Striegnitz. "Learn Prolog now!" Informe técnico, University of the Saarland, 2001. Publicación electrónica en http://www.coli.uni-sb.de/~kris/learn-prolog-now/.

[BDP96] Christophe Bourely, Gilles Défourneaux y Nicolas Peltier. "Building proofs or counterexamples by analogy in a resolution framework". En José Júlio Alferes, Luís Moniz Pereira y Ewa Orlowska, editores, *JELIA*, volumen 1126 de *Lecture Notes in Computer Science*, páginas 34–49. Springer, 1996.

[Bel] John Bell. "Inductive, abductive and pragmatic reasoning". En *IJCAI'97 Workshop on Abduction and Induction in AI*, páginas 7–12.

[Ben03] Johan van Benthem. "Logic and the dynamics of information". *Minds and Machines*, 13:503–519, 2003.

[Bet69] Evert Willem Beth. *Entrañamiento semántico y derivabilidad formal*. Cuadernos Teorema, 18, 1969.

[BFZ93] Matthias Baaz, Christian G. Fermüller y Richard Zach. "Dual systems of sequents and tableaux for many-valued logics". *Bulletin of the EATCS*, (49):192–197, 1993.

[BG96] Bernhard Beckert y Rajeev Goré. "Free variable tableaux for propositional modal logics". Interner Bericht 41/96, Universität Karlsruhe, Fakultät für Informatik, 1996.

[Boo84] George Boolos. "Trees and finite satsifactibility". *Notre Dame Journal of Formal Logic*, (25):110–115, 1984.

[BP94] Bernhard Beckert y Joachim Posegga. "Logic programming as a basis for lean deduction: Achieving maximal efficiency from minimal means". En N. E. Fuchs y G. Gottlob, editores, *Proceedings, 10th Logic Programming Workshop, Zürich, Switzerland*. Institut für Informatik der Universität Zürich, 1994.

[BP95] Bernhard Beckert y Joachim Posegga. "leanTAP: Lean tableau-based deduction". *Journal of Automated Reasoning*, 15(3):339–358, 1995.

[Bra78] M.D.S. Braine. "On the relation between the natural logic of reasoning and standard logic". *Psychological Review*, 85:1–21, 1978.

[BS01] Debra T. Burhans y Stuart C. Shapiro. "Abduction and question answering". En *Proceedings of the IJCAI-01 Workshop on Abductive Reasoning*. Seattle, 2001.

[Bun99] Alan Bundy. "A survey of automated deduction". En Michael J. Wooldridge y Manuela Veloso, editores, *Artificial Intelligence Today. Recent Trends and Developments*, volumen 1600 de *Lecture Notes in Computer Science*, páginas 153–174. Springer, 1999.

[Bus95] Samuel R. Buss. "On Herbrand's theorem". *Lecture Notes in Computer Science*, 960:195–209, 1995.

[Car58] Rudolf Carnap. *Introduction to Symbolic Logic and its Applications*. Dover Publications, 1958.

[Cos89] Leda Cosmides. "The logic of social exchange: Has natural selection shaped how humans reason? studies with the wason selection task". *Cognition*, 31:187–276, 1989.

[CP] Charles S. Peirce. *Collected Papers of Charles Sanders Peirce*. Volúmenes 1–6 editados por C. Hartshorne, P. Weiss. Cambridge, Harvard University Press, 1931–1935; volúmenes 7–8 editados por A.W. Burks. Cambridge, Harvard University Press, 1958.

[CP84] Philip T. Cox y Tomasz Pietrzykowski. "A complete, nonredundant algorithm for reversed skolemization". *Theoretical Computer Science*, 28:239–261, 1984.

[CP93] Marta Cialdea Mayer y Fiora Pirri. "First order abduction via tableau and sequent calculi". *Bulletin of the IGPL*, 1:99–117, 1993.

[Cue85] José Cuena. *Lógica informática*. Alianza-Informática, 1985.

[Dav89] Ruth E. Davis. *Truth, Deduction, and Computation. Logic and Semantics for Computer Science*. Computer Science Press, 1989.

[Dav01] Martin Davis. "The early history of automated deduction", 2001.

[Deb98] Guy Debrock. "El ingenioso enigma de la abducción". *Analogía Filosófica*, XII(1):21–41, 1998.

[DEDC96] Pierre Deransart, AbdelAli Ed-Dbali y Laurent Cervoni. *Prolog, The Standard: Reference Manual*. Springer, 1996.

[Dia93] Emilio Díaz. "Árboles semánticos y modelos mínimos". En *Actas del I Congreso de la Sociedad de Lógica, Metodología y Filosofía de la Ciencia en España*, página 40. Universidad Complutense de Madrid, 1993.

[DP97] Gilles Défourneaux y Nicolas Peltier. "Analogy and abduction in automated deduction". En *IJCAI'97 Workshop on Abduction and Induction in AI*, páginas 216–225. 1997.

[DS98] Marc Denecker y Danny De Schreye. "SLDNFA: An abductive procedure for abductive logic programs". *Journal of Logic Programming*, 34(2):111–167, 1998.

[DV01] Anatoli Degtyarev y Andrei Voronkov. "The inverse method". En Alan Robinson y Andrei Voronkov, editores, *Handbook of Automated Reasoning*, volumen 1, capítulo 1, páginas 179–272. Elsevier Science and MIT Press, 2001.

[Dyc97] Roy Dyckhoff. "Some benchmark formulae for intuitionistic propositional logic". Informe técnico, University of St. Andrews, 1997. Publicación electrónica en http://www.dcs.st-and.ac.uk/~rd/logic/marks.html.

[End04] Herbert B. Enderton. *Una introducción matemática a la lógica*. Traducción de José A. Amor. Instituto de Investigaciones Filosóficas de la UNAM, México, 2004.

[EP] Nathan Houser y Christian Kloesel, editores. *The Essential Peirce. Selected Philosophical Writings*. Volúmenes 1–2. Indiana University Press, Bloomington, 1992–1998.

[ES83] Umberto Eco y Thomas A. Sebeok. *The Sign of Three: Dupin, Holmes, Peirce*. Indiana University Press, Bloomington, IN, 1983.

[Fit98] Melvin Fitting. "leanTAP revisited". *Journal of Logic and Computation*, (8):33–47, 1998.

[Fla94] Peter Flach. *Simply Logical. Intelligent Reasoning by Example*. John Wiley, 1994.

[Fla95] Peter Flach. *Conjectures: An Inquiry Concerning the Logic of Induction*. Tesis Doctoral, Katholieke Universiteit, Brabant, 1995.

[GG01] D. Gabbay y F. Guenthner, editores. *Handbook of Philosophical Logic*. 13 volúmenes. Kluwer Academic Publishers, 2ª edición, 2001–2005.

[Gen96] Gonzalo Génova. "Los tres modos de inferencia". *Anuario Filosófico*, XXIX(3):1249–1265, 1996.

[Gom96] Antoni Gomila. "Peirce y la ciencia cognitiva". *Anuario Filosófico*, XXIX(3):1345–1369, 1996.

[Gri97] Ralph Grimaldi. *Matemáticas discreta y combinatoria*. Addison-Wesley, 1997.

[Her92] Carmen Hernández. *Lógica y racionalidad del descubrimiento (acerca del* Método *de Arquímedes*. Secretariado de Publicaciones de la Universidad de Sevilla, 1992.

[Hin98] J. Hintikka. "What is abduction? The fundamental problem of contemporary epistemology". *Transactions of the Charles S. Peirce Society*, 34(3):503–533, 1998.

[Hof98] Michael Hoffmann. "¿hay una lógica de la abducción?" *Analogía Filosófica*, XII(1):41–57, 1998.

[Hof03] Douglas R. Hofstadter. *Gödel, Escher, Bach*. Tusquets, 8ª edición, 2003.

[Ino01] Katsumi Inoue. "Induction, abduction, and consequence-finding". En *Proceedings of the 11th International Conference on Inductive Logic Programming*, páginas 65–79. Springer, 2001.

[JL83] P.N. Johnson-Laird. *Mental models*. Cambridge University Press, 1983.

[KKT98] Antonis Kakas, Robert Kowalski y Francesca Toni. "The role of abduction in logic programming". En *Handbook of logic in Artificial Intelligence and Logic Programming*, páginas 235–324. Oxford University Press, 1998.

[Kle74] Stephen Cole Kleene. *Introducción a la metamatemática*. Tecnos, 1974.

[Kra02] Peter Krause. "Presupposition justification by abduction and quantified presuppositions". En Graham Katz, Sabine Reinhard y Philip Reuter, editores, *Sinn & Bedeutung VI, Proceedings of the Sixth Annual Meeting of the Gesellschaft für Semantik*. University of Osnabrück, 2002.

[Kui] Theo A.F. Kuipers. "Inference to the best theory rather than inference to the best explanation. Kinds of abduction and induction". En F. Stadler, editor, *Induction and Deduction in the Sciences*. Springer, 2004, páginas 25–51.

[Kun96] Kenneth Kunen. "The semantics of answer literals". *Journal of Automated Reasoning*, 17(1):83–95, 1996.

[LCLRC93] Baudouin Le Charlier, Christophe Leclère, Sabina Rossi y Agostino Cortesi. "Automated verification of Prolog programs". *The Journal of Logic Programming*, 1993.

[LD94] Nada Lavrac y Saso Dzeroski. *Inductive Logic Programming. Techniques and Applications*. Ellis Horwood, New York, 1994.

[Lip] Peter Lipton. *Inference to the Best Explanation*. Routledge, New York.

[Lus92] Ewing E. Lusk. "Controlling redundancy in large search spaces: Argonne-style theorem proving through the years". En *LPAR'92*. Springer, 1992.

[LY02] Fangzhen Lin y Jia-Huai You. "Abduction in logic programming: A new definition and an abductive procedure based on rewriting". *Artificial Intelligence*, 140(1/2):175–205, 2002.

[Man89] María Manzano. *Teoría de Modelos*. Alianza, Madrid, 1989.

[Mar95] Pascual F. Martínez-Freire. *La nueva filosofía de la mente*. Gedisa, Barcelona, 1995.

[McI98] Sheila A. McIlraith. "Logic-based abductive inference". Informe técnico, Knowledge Systems Laboratory, julio 1998.

[Mer01] László Mérö. *Los azares de la razón. Fragilidad humana, cálculos morales y teoría de juegos*. Paidós, Barcelona, 2001.

[Mit97] Tom M. Mitchell. *Machine Learning*. McGraw-Hil, 1997.

[MS] Charles S. Peirce. *The Charles S. Peirce Papers*. 32 rollos de microfilms de los manuscritos conservados en la Houghton Library. Cambridge, Harvard University Library, Photographic Service, 1966.

[MV98] Erica Melis y Manuela M. Veloso. "Analogy in problem solving". En L.Farinas del Cerro, D. Gabbay y H.J. Ohlbach, editores, *Handbook of Practical Reasoning Computational and Theoretical Aspects*. Oxford University Press, 1998.

[Nag68] Ernest Nagel. *La estructura de la ciencia*. Paidós, 1968.

[Nep02] Ángel Nepomuceno. "Scientific explanation and modified semantic tableaux". En *Logical and Computational Aspects of Model-Based Reasoning*, Applied Logic Series, páginas 181–198. Kluwer Academic Publishers, 2002.

[Nep03] Ángel Nepomuceno. *El método de las tablas semánticas*. Kronos, Sevilla, 2003.

[Nep04] Ángel Nepomuceno. "La lógica y las ciencias de la mente". En Nepomuceno et al. [NSS04], páginas 133–161.

[NM95] Ulf Nilsson y Jan Maluszynski. *Logic, Programming and Prolog*. John Wiley, 1995.

[NS04] Ángel Nepomuceno y Fernando Soler. "Abducción y razonamiento por defecto". En Vicente et al. [VFC+04], páginas 385–387.

[NSS04] Ángel Nepomuceno, Francisco José Salguero y Fernando Soler, editores. *Bases biológicas, lingüísticas, lógicas y computacionales para la conceptualización de la mente*. Mergablum, Sevilla, 2004.

[Nub98] Jaime Nubiola. "Walker Percy y Charles S. Peirce: Abducción y lenguaje". *Analogía Filosófica*, XII(I):87–97, 1998.

[Paa04] Sami Paavola. "Abduction as a logic and methodology of discovery: the importance of strategies". *Foundations of Science*, 9(3):267–283, 2004.

[Pal02] Gladys Dora Palau. *Introducción filosófica a las lógicas no clásicas*. Gedisa, 2002.

[Pel86] Francis Jeffry Pelletier. "Seventy-five problems for testing automatic theorem provers". *Journal of Automated Reasoning*, 2:191–216, 1986.

[Pel03] Nicolas Peltier. "A more efficient tableau procedure for simultaneous search for refutations and finite models". En Marta Cialdea Mayer y Fiora Pirri, editores, *TABLEAU'03 (International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, páginas 181–195. Springer, 2003.

[PMG98] David Poole, Alan Mackworth y Randy Goebel. *Computational Intelligence: A Logical Approach*. Oxford University Press, 1998.

[Pop62] Karl R. Popper. *La lógica de la investigación científica*. Tecnos, 1962.

[Pop67] Karl R. Popper. *El desarrollo del conocimiento científico*. Paidós, 1967.

[Pop74] Karl R. Popper. *Conocimiento objetivo*. Tecnos, 1974.

[Pos93] Joachim Posegga. "Compiling proof search in semantic tableaux". En *Seventh International Symposium on Methodologies for Intelligent Systems*, LNAI. Springer, Trondheim, junio 1993.

[PS99] Joachim Posegga y P. H. Schmitt. "Implementing semantic tableaux". En Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle y Joachim Posegga, editores, *Handbook of Tableau Methods*, páginas 581–629. Kluwer Academic Publishers, 1999.

[Rip83] L.J. Rips. "Cognitive processes in propositional reasoning". *Psychological Review*, 90:38–71, 1983.

[Rip94] L.J. Rips. *The Psychology of Proof*. MIT Press, 1994.

[RN96] Stuart Russell y Peter Norvig. *Inteligencia Artificial: un enfoque moderno*. 1996.

[Rob65] John Alan Robinson. "A machine-oriented logic based on the resolution principle". *Journal of the ACM*, 12:23–41, 1965.

[Rus02] Bertrand Russell. *El conocimiento humano*. Ediciones Folio, Barcelona, 2002.

[SAMP97] Michael Schroeder, Iara de Almeida Mora y Luis Moniz Pereira. "A deliberative and reactive diagnosis agent based on logic programming". En *Agent Theories, Architectures, and Languages*, páginas 293–307. 1997.

[Sch98] Michael Schroeder. *Model-based Diagnosis Agents*. Kluwer Academic Publishers, 1998.

[Sch03a] Viola Schiaffonati. "A framework for the foundation of the philosophy of artificial intelligence". *Minds and Machines*, 13:537–552, 2003.

[Sch03b] Ute Schmid. "Analogical reasoning and generalization". *Lecture Notes in Computer Science*, 2654:279–290, 2003.

[Sol93] Daniel Solow. *Cómo entender y hacer demostraciones en Matemáticas*. Limusa, México, 1993.

[Sol04a] Fernando Soler. "Cálculo de δ-resolución proposicional". En Vicente et al. [VFC$^+$04], páginas 392–395.

[Sol04b] Fernando Soler. "leanTAP, un sistema de representación y razonamiento para lógica de primer orden". En Nepomuceno et al. [NSS04], páginas 175–197.

[Tha93] Paul Thagard. *Computational Philosophy of Science*. The MIT Press, 1993.

[VDNMP94] Carlos Viegas Damásio, Wolfgang Nejdl y Luís Moniz Pereira. "REVISE: An extended logic programming system for revising knowledge bases". En *Proceedings of the International Conference on Knowledge Representation and Reasoning*. Bonn, 1994.

[VFC$^+$04] Agustín Vicente, Patricia de la Fuente, Cristina Corredor, Juan Barba y Alfredo Marcos, editores. *Actas del IV Congreso de la Sociedad de Lógica, Metodología, y Filosofía de la Ciencia en España*. Sociedad de Lógica, Metodología, y Filosofía de la Ciencia en España, 2004.

[Win92] Patrick H. Winston. *Artificial intelligence*. Addison-Wesley, 3ª edición, 1992.