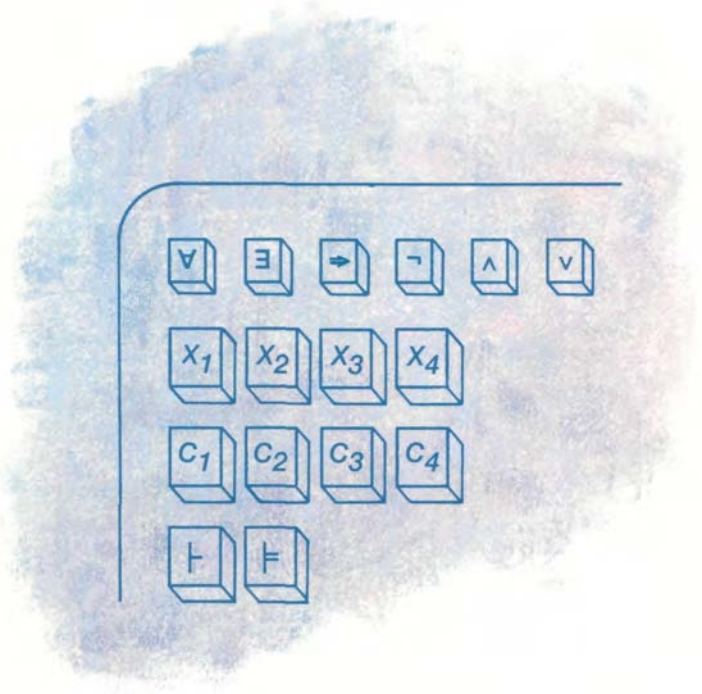
Lógica Matemática

Julio Ernesto Solís Daun
Yolanda Torres Falcón



Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

U N I D A D I Z T A P A L A P A

Julio Ernesto Solís Daun. Matemático, egresado de la Universidad Autónoma de Yucatán (1985). Cursó la Maestría en Matemáticas en la UAM-Iztapalapa (1989). Actualmente, es alumno del Doctorado en Ciencias por la misma universidad (1994). Otros estudios: Laboratorista Químico en la UADY (1985), y pasante de la Maestría de Filosofía de la Ciencia (área de ciencias formales) en la UAM-I (1994). Profesor Titular de tiempo completo del Departamento de Matemáticas de la UAM-I. Areas de interés: teoría de control, ecuaciones diferenciales y lógica matemática.

Lógica matemática

Julio Ernesto Solís Daun
Depto. de Matemáticas, C.B.I.

Yolanda Torres Falcón
Depto. de Filosofía, C.S.H.

Primera Edición 1995

UNIDAD IZTAPALAPA
Av. Michoacán y La Purísima
Iztapalapa, 09340, México D.F.

ISBN: 970-620-600-0

Impreso en México


vigésimo
aniversario
Casa abierta al tiempo
Tiempo abierto al cambio

Lógica matemática

Prefacio

Este texto fue escrito pensando en el curso de lógica que se imparte en la División de CBI a los alumnos de computación y de matemáticas aplicadas.

Dado que éste es el único curso de lógica contemplado en los programas de estudio de estas licenciaturas, resulta importante cubrir, en la medida de lo posible, todo el material que el alumno va a necesitar durante su carrera.

Existen muchos textos de lógica matemática, pero no conocemos ninguno apropiado para este curso: los de enfoque filosófico se concentran en problemas diferentes y no tienen ejemplos ni ejercicios adecuados; los de enfoque matemático cubren muchos temas que van más allá de las necesidades del curso, como recursividad, teoría de modelos o teoría de la demostración, y en consecuencia el material que nos interesa viene dado escuetamente. En ambos casos falta relacionar los teoremas y métodos de lógica matemática con problemas en ciencias computacionales.

Recientemente se han publicado algunos libros de computación con enfoque a la inteligencia artificial que tocan temas de lógica matemática, pero sólo enuncian lo necesario para entrar en materia.

Hace falta un texto que cubra adecuadamente la sintaxis y la semántica, tanto para la lógica proposicional como la de primer orden; que tenga ejemplos resueltos, muchos ejercicios y que relacione la lógica con algunos temas de computación.

Este texto es nuestra respuesta a tal necesidad. Tiene las siguientes características:

1. Contextualiza la lógica por medio de una introducción sobre argumentos y un resumen de su desarrollo histórico (Capítulo 1).

2. Es autocontenido, pues en el Capítulo 2 se definen todos los conceptos necesarios de teoría de conjuntos, a la vez que se presenta con detalle el método de demostración por inducción matemática, que es esencial en lógica.

3. Hace una presentación exhaustiva e intuitiva de los temas del programa:

   - Lenguajes y sistemas formales.

Estos conceptos se introducen en el Capítulo 3 por medio de ejemplos sencillos y amenos.

• Semántica para la lógica proposicional.
Se trabaja en el Capítulo 4 de tres maneras: por tablas de verdad, con valuaciones y con árboles semánticos. El primer enfoque es el tradicional y se incluyó por ser el más fácil y conocido por la mayoría de los alumnos. El segundo viene en muy pocos libros, es una generalización natural del primero y es más elegante. Nos sirve para demostrar el teorema de compacidad y muchos teoremas sobre nociones semánticas básicas. El tercer enfoque es más moderno y es un método de demostración algorítmico. Estos tres enfoques se desarrollan de manera tal que el alumno note que son tres maneras distintas de atacar el mismo problema.

• Sintaxis para la lógica proposicional.
Se desarrollan principalmente dos sistemas: uno axiomático y uno de deducción natural. Al final se interrelacionan por medio de los teoremas de validez y completud. Se presenta también un tercer enfoque, el de demostración automática de teoremas. Estos tres enfoques representan distintos niveles de mecanización del procedimiento de prueba. Se ayuda al alumno por medio de numerosos ejemplos resueltos, acompañados de comentarios sobre las ideas subyacentes en la resolución.

• Semántica para la lógica de primer orden.
La definición de satisfacibilidad de Tarski ha demostrado ser de importancia crucial en el desarrollo de la lógica comtemporánea. A pesar de ser una definición difícil de entender cuando se ve por primera vez, en general no se motiva ni se explica con detalle en la literatura. Aquí se introduce el tema con ejemplos sobre los números naturales y se hace ver que es una extrapolación natural de las valuaciones para la lógica proposicional, tomando en cuenta que se tienen distintas categorías semánticas básicas.

• Sintaxis para la lógica de primer orden.
Se desarrollan dos sistemas, uno axiomático y otro de deducción natural, extensiones de los correspondientes para la lógica proposicional. Se demuestran los metateoremas básicos de la lógica de primer orden: deducción, completud y compacidad, con algunas de sus consecuencias, como el teorema de Löwenheim-Skolem y la existencia de modelos no estándares de la aritmética.

4. Cubre gran cantidad de material para otros cursos:

   • *Diseño Lógico.* En el Capítulo 4 se da una interpretación de las fórmulas en términos de circuitos (Lógica combinacional).

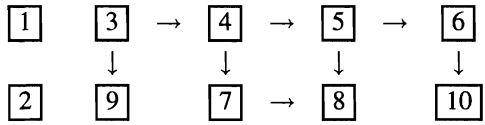
   • *Teoría Matemática de la Computación.* El Capítulo 9 está dedicado a lenguajes y autómatas, con particular énfasis en los autómatas finitos y lenguajes regulares. Se introducen primero gramáticas y lenguajes formales, y después autómatas, de manera tal que el teorema de Kleene sobre lenguajes regulares hace las veces de un teorema de completud y validez bajo la interpretación: "un autómata finito $\mathscr{M}$ acepta una palabra α" si y sólo si "α es cierta para $\mathscr{M}$". El Capítulo 10 es sobre máquinas de Turing, definición y construcción de una máquina universal. Estos dos capítulos cubren más del 50% del programa para este curso.

   • *Análisis y Diseño de Algoritmos.* En el Capítulo 6 se introduce el tema de los problemas $\mathscr{NP}$-completos en su relación con satisfacibilidad. Esto se retoma al final del Capítulo 10 en su relación con máquinas de Turing. Éste es uno de los temas más interesantes y complicados en análisis de algoritmos.

5. Contiene gran cantidad de ejemplos resueltos y muchos ejercicios adecuados al nivel y formación de los alumnos.

6. Puede ser utilizado para muchos cursos diferentes, pues el material no puede ser cubierto en un trimestre de 11 semanas como los de la UAM.

El diagrama siguiente muestra la interrelación entre los capítulos; en donde, los Capítulos 1 y 2 se muestran disconexos por el carácter general de sus contenidos, y el Capítulo 10 depende del 6 sólo en la última sección.



El texto consta de 10 Capítulos divididos en secciones. Los teoremas aparecen numerados por capítulos. Los símbolos ■, □ y $*$ denotan el final de una prueba o su ausencia, el final de un ejemplo y los ejercicios difíciles, respectivamente.

Deseamos manifestar nuestro agradecimiento al Prof. José A. Amor por sus comentarios y revisión del presente manuscrito.

Los autores

Junio de 1994

Contenido

Capítulo 1
Introducción

La pregunta es ésta: ¿Es NO la respuesta correcta a esta pregunta?
—¿A qué pregunta?— preguntó Alicia.
— ¡Pues a la pregunta que acabo de hacerte!
— respondió Humpty Dumpty.

Raymond Smullyan

1.1 Argumentos

La lógica se ocupa de las argumentaciones válidas. Las argumentaciones ocurren cuando se quiere justificar una proposición con base en otras asegurando que la primera es consecuencia necesaria de las últimas.

Un *argumento* es una lista de proposiciones o enunciados. El último enunciado es la conclusión del argumento y los otros son las premisas o hipótesis.

Cuando se afirma que un argumento es *válido* o *correcto* se sostiene que las premisas y la conclusión están relacionadas de tal manera que la conclusión se sigue de las premisas por necesidad estricta, en otras palabras, que afirmar las premisas y negar la conclusión sería contradictorio.

Ejemplos:

Ejemplo 1 El detective Sherlock Holmes entra en posesión de un viejo sombrero de fieltro, a partir del cual infiere ciertas cosas acerca de su propietario, sin

conocerlo. Entre sus conclusiones está la de que el propietario es muy intelectual. Al comunicárselo al Dr. Watson, éste pide a Holmes que la justifique. En guisa de respuesta, Holmes se caló el sombrero en la cabeza. Lo bajó más abajo de la frente y se le asentó sobre el puente de la nariz. 'Es cuestión de capacidad cúbica', dijo: 'un individuo de tamaño cerebro ha de tener algo en él'. Con esto Holmes da por demostrada su conclusión. Hagamos explícito el argumento de Holmes:

1. Este sombrero es grande.
2. Los propietarios de sombreros grandes tienen cabezas grandes.
3. La gente de cabeza grande tiene grande el cerebro.
4. La gente de cerebro grande es muy intelectual.

Conclusión:

5. El propietario de este sombrero es muy intelectual.

Ejemplo 2

1. Todos los mamíferos son mortales.
2. Todos los perros son mortales.

Conclusión:

3. Todos los perros son mamíferos.

En el ejemplo (1) la conclusión no está justificada por las premisas porque la verdad de las premisas no está demostrada. Sin embargo, *si* se aceptara sin reserva la verdad de las premisas, *entonces* la verdad de la conclusión tendría que ser aceptada también. Por su forma lógica, el argumento es correcto, pero para que la conclusión quede totalmente justificada se tiene que probar la verdad de las premisas.

El caso del ejemplo (2) es distinto. Aunque tanto las premisas como la conclusión son verdaderas, la relación entre ellas no es tal que la conclusión se siga necesariamente de las premisas. Consideremos el siguiente argumento:

1. Todos los franceses son europeos.
2. Todos los italianos son europeos.

Conclusión:

3. Todos los italianos son franceses.

Este argumento tiene exactamente la misma forma que el argumento del ejemplo (2) y sin embargo tiene premisas verdaderas y conclusión falsa.

A la lógica le interesa la *forma* de las proposiciones que integran un argumento, no su verdad o falsedad de hecho. Cuando un argumento es correcto, lo es en virtud de la forma de las proposiciones que lo componen. A lo largo del libro estudiaremos proposiciones en distintos lenguajes, las analizaremos y caracterizaremos todos los argumentos correctos en esos lenguajes.

1.2 Falacias

Hay argumentos que parecen válidos pero que en realidad no lo son. Se llaman *falacias,* y aunque no las estudiaremos en este libro, su estudio también cae dentro del dominio de la lógica. Aquí nos limitaremos a dar algunos ejemplos de falacias comunes.

Las falacias pueden ser divididas en tres tipos: falacias de *ambigüedad,* falacias de *presunción* y falacias de *relevancia.*

Las falacias de *ambigüedad* engañan por la naturaleza confusa del lenguaje en el que se expresan los argumentos.

Ejemplos:

1. El control de la natalidad es un suicidio de raza, pues si no nacen niños la raza desaparecerá.

Aquí una palabra clave en el argumento cambia de significado durante él, este tipo de falacia se llama de *equivocación.*

2. Las palomas mensajeras están prácticamente extintas. Esa es una paloma mensajera y por tanto está prácticamente extinta.

Esta falacia surge de aplicar lo que es verdadero para un todo a cada parte del todo. Es una falacia de *división.*

Las falacias de *presunción* engañan por su semejanza a formas válidas de argumentación.

Ejemplos:

Mal uso de las generalizaciones:

1. Herir a las personas es malo; por lo tanto los dentistas son malos.
2. La ciencia no debe ser tomada en serio. No ha podido explicar el origen de la vida.

Bifurcación (presume que una cierta clasificación es exhaustiva):

3. Jesucristo: "Quien no está conmigo está contra mí".

Petición de principio (se supone lo que se quiere justificar):

4. El Corán es infalible, pues fue compuesto por Mahoma, profeta de Dios.
5. Todo ser humano debería ser libre, pues la libertad es un derecho universal de la humanidad.

Falsa causalidad (se sugiere que ciertos eventos están conectados causalmente sin ninguna justificación):

6. Cada vez que se hacen pruebas atómicas se registran tormentas serias. Es obvio que se deben cesar estas pruebas, si no queremos alterar el clima del planeta.

Tesis irrelevante (se aduce a hechos irrelevantes para justificar la conclusión):

7. —Tienes que comerte la espinaca, hijito. Ya sabes cuántos niños se mueren de hambre en el mundo.

Las falacias de *relevancia* engañan a través de emociones.

Ejemplos:

Falacia genética (se condena una conclusión aduciendo a sus orígenes):

1. La religión se originó con la magia y el animismo. Por lo tanto no tiene sentido.
2. Esta ley está diseñada para explotar a los pobres: fue escrita por el senador más rico.

Falacia *ad hominem* (se argumenta atacando a la persona que sostiene lo contrario):

3. Se rechaza la sugerencia del Sr. X de aumentar la eficiencia de las universidades. Dado que se trata de un empresario, no se le puede pedir que entienda que nuestro propósito es educar a la juventud, no obtener ganancias.

1.3 Lenguajes y metalenguajes

Al estudiar lógica estaremos estudiando proposiciones en ciertos lenguajes y las relaciones entre ellas. Para estudiar estas proposiciones tendremos que utilizar un lenguaje, como el español, por ejemplo. Tendremos, por lo tanto, varios niveles de lenguaje: el lenguaje en el que están escritos los argumentos bajo objeto de estudio y el lenguaje utilizado para estudiarlos. Este último es el *metalenguaje.*

Esta diferencia de niveles se da en el habla cotidiana, pero el contexto nos ayuda a identificar el nivel en el que se está hablando. Consideremos, por ejemplo, las dos oraciones siguientes:

1. George Sand fue amante de Chopin.
2. George Sand era el seudónimo de Aurora Dupin.

En el primer caso se está afirmando algo de una persona, "George Sand" denota a una persona y se dice que la expresión se está *usando.* En el segundo se afirma algo de una expresión, "George Sand" se está *mencionando.*

En lógica a veces no es tan claro cuándo se está usando una expresión y cuándo se está mencionando. Para evitar confusiones se ha convenido en entrecomillar una expresión cuando se esté mencionando. Por ejemplo, con esta convención se escribe:

París es la capital de Francia y "París" tiene cinco letras.

1.4 Resumen histórico

Tradicionalmente se ha distinguido entre la *lógica deductiva,* cuyos principios se usan para obtener conclusiones de premisas dadas, y la *lógica inductiva,* que saca conclusiones generales a partir de hechos particulares que sirven de evidencia para

ellas. Esta distinción es obsoleta porque los problemas de inducción se tratan ahora en la metodología de las ciencias naturales. Para nosotros, entonces, lógica quiere decir lógica deductiva.

En un sentido estricto la lógica deductiva se divide en dos: la lógica de proposiciones y la lógica de predicados.

La lógica de proposiciones se llama así porque en ella las proposiciones o enunciados forman la única categoría semántica básica. Algunas proposiciones son simples y no se analizan, otras son compuestas y se analizan descomponiéndolas por medio de conectivos proposicionales (y, o, no, si . . . entonces) en proposiciones más simples. (Esta lógica se estudia con detalle en los capítulos 4 y 5).

En la lógica de predicados las proposiciones simples se descomponen en partes más simples, que forman así una segunda categoría semántica: la categoría de los nombres. Los nombres aparecen en las proposiciones unidos a predicados (de ahí el nombre de lógica de predicados), que expresan propiedades y relaciones, funcionando como "verbos". (Esta lógica se estudia en los capítulos 7 y 8).

En un sentido más amplio la lógica también comprehende varias teorías del lenguaje como sintaxis lógica y semántica lógica. Las *lógicas modales* (que estudian los conceptos de necesidad, posibilidad, contingencia, etc.) y el estudio de las paradojas y falacias también entran en este aspecto amplio de la lógica.

Aunque la teoría de la lógica proposicional es anterior desde un punto de vista lógico, a la lógica de predicados, esta última antecedió a la primera históricamente.

El primer sistema de la lógica de predicados fue creado por Aristóteles en el siglo IV a.c., en su monumental obra *Organon*, título que refleja el punto de vista de que la lógica es una herramienta para afinar el pensamiento.

En su obra, Aristóteles describió y clasificó *silogismos* válidos (*i.e.*, aquéllos en los que la conclusión en efecto se sigue de las premisas) y demostró por medio de contraejemplos la invalidez de ciertos silogismos. También inició el desarrollo de la lógica modal.

Una de las características más importantes de la obra de Aristóteles es que pudo dar a sus silogismos la forma de principios lógicos gracias a que, por primera vez en la historia de la lógica, hizo uso de variables o letras para representar proposiciones arbitrarias.

La lógica moderna empezó en el siglo XVII con Leibniz y desde entonces su desarrollo ha estado estrechamente relacionado con las matemáticas.

El programa de Leibniz era la construcción de un lenguaje universal, un cálculo general del razonamiento y una metodología general. Él aplicó con éxito métodos

matemáticos para la interpretación de la silogística aristotélica, y su visión acerca de lenguajes artificiales y la reducción del razonamiento a cálculos aritméticos fructificó en el trabajo de Gödel y en la emergencia de las ciencias computacionales.

Durante la segunda mitad del siglo XIX se gestaron los cimientos para el gran desarrollo que la lógica ha tenido desde entonces.

El matemático inglés George Boole publicó en 1854 un trabajo titulado *An Investigation into the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities*. Con este trabajo se progresó considerablemente al continuar los intentos de Leibniz de un cálculo algebraico para las leyes del pensamiento. El álgebra booleana tiene interpretaciones tanto en la lógica proposicional como en la de predicados.

El enfoque del matemático alemán Gottlob Frege era distinto. Él estaba interesado en el concepto de número. Pensaba que la noción de número natural se podía reducir a conceptos lógicos y que, por lo tanto, se podría demostrar que la aritmética era una parte de la lógica.

Una reducción formal de la aritmética a la lógica la dio Frege en *Grundgesetze der Arithmetik Begriffsschriftlich* (1893).

Otro aspecto en el estudio de la lógica y los fundamentos de la matemática empezó con la creación de la teoría de conjuntos por Georg Cantor, como una nueva disciplina matemática.

La teoría de Cantor no era deductiva, como la de Frege, sino que estaba, por así decirlo, en estado pre-axiomático. Para Cantor un conjunto era "una colección de objetos distintos, definidos, de nuestra percepción o nuestro pensamiento". Un conjunto está determinado unívocamente por sus elementos. Con estos conceptos Cantor creó una de las teorías matemáticas más profundas y bellas, con la que dio inicio el estudio de los llamados cardinales transfinitos.

En 1902 Bertrand Russell descubrió una paradoja que atacaba tanto al sistema de Frege como al de Cantor. Es la llamada *Paradoja de Russell* que consiste en definir el conjunto R cuyos elementos son precisamente todos los conjuntos que no son elementos de sí mismos. ¿Es R un elemento de sí mismo o no? Si lo fuera, entonces, por definición, tendría que satisfacer la propiedad que lo define y por lo tanto no sería elemento de sí mismo. Pero si no es elemento de sí mismo entonces satisface la propiedad que define a R y en consecuencia sería un elemento de R, esto es, sería un elemento de sí mismo. Esta es una contradicción.

La paradoja de Russell no era la única paradoja que había sido descubierta a fines del siglo pasado y a principios de éste. Había paradojas en teoría de conjuntos que

involucraban el concepto de cardinalidad y había paradojas semánticas, algunas que databan desde el tiempo de los griegos, como la paradoja del mentiroso y algunas recién descubiertas.

Paradoja del Mentiroso. Un hombre dice: "Estoy mintiendo". Si está mintiendo entonces lo que dice es verdadero y por lo tanto no está mintiendo. Si no está mintiendo, entonces lo que dice es verdadero, luego está mintiendo.

Paradoja de Berry (1906). Hay sólo un número finito de sílabas en español. Por lo tanto hay sólo un número finito de expresiones en español que tienen menos de cuarenta sílabas. Hay, por tanto, sólo un número finito de números naturales denotados por una expresión en español con menos de cuarenta sílabas. Sea k el *mínimo número natural no denotado por una expresión en español de menos de cuarenta sílabas.* La expresión en itálicas denota a k y tiene menos de cuarenta sílabas.

Paradoja de Grelling (1908). Un adjetivo se llama *autológico* si la propiedad denotada por el adjetivo es satisfecha por el adjetivo mismo. Un adjetivo es *heterológico* si la propiedad denotada por el adjetivo no se aplica al adjetivo mismo. Por ejemplo, "azul" es heterológico mientras que "polisilábico" es autológico. Considérese el adjetivo "heterológico". Si es heterológico entonces no satisface la propiedad denotada por él mismo y por tanto no es heterológico, si no es heterológico entonces satisface la propiedad denotada por él mismo y, por tanto, es heterológico.

El análisis de las paradojas condujo a varias propuestas para eliminarlas. Las tres propuestas principales, a principios de este siglo, fueron las siguientes:

1. La propuesta logicista

Encabezada por Bertrand Russell. A pesar de haber encontrado contradicciones en la teoría de Frege, Russell siguió creyendo que la aritmética se podía derivar de la lógica y que, en consecuencia, toda la matemática podría ser fundamentada en la lógica. En su intento por demostrar esto produjo, en colaboración con Whitehead, *Principia Mathematica* (1910–1913). Este trabajo se convirtió pronto en un clásico de la lógica.

Con respecto a las paradojas, Russell argumentó que surgen de un círculo vicioso que consistía en suponer totalidades ilegítimas. Notó que la auto-referencia

está presente en todas las paradojas y sugirió estratificar al lenguaje para evitar que una expresión predique sobre sí misma. Esta es la llamada *teoría de tipos.*

Esta teoría influyó mucho en Zermelo, quien en 1908, produjo una teoría axiomática para la teoría de conjuntos de Cantor que eliminó todas las paradojas conocidas de la teoría.

2. La propuesta intuicionista

Un enfoque más radical fue adoptado por Brouwer y su escuela intuicionista. Ellos creían que la raíz de las paradojas estaba en el concepto del infinito, y que el problema estaba en generalizar del caso finito al caso infinito. Para ellos no tenía sentido hablar de totalidades infinitas. También rechazaron la universalidad de ciertas leyes lógicas, como la ley del tercero excluso: P o no P. Heyting hizo un estudio sistemático de los principios lógicos clásicos que los intuicionistas aceptaban y esta escuela dio origen a un tipo de lógica no-clásica conocida como *lógica intuicionista.*

3. La propuesta formalista

Es la propuesta del matemático alemán David Hilbert y su escuela. Hilbert estaba tan interesado como Frege en el método axiomático, pero, a diferencia de él, no le daba ninguna importancia a la interpretación de los símbolos de un formalismo. Para él la matemática era una colección de sistemas formales sin significado y la tarea del matemático era demostrar que estos sistemas eran consistentes, es decir, sin contradicciones. La disciplina que se ocuparía de la demostración de la consistencia de las teorías matemáticas no sería la matemática propiamente, sino una nueva disciplina que él llamó *metamatemática.*

Su proyecto era, pues, el desarrollo de un sistema lógico-matemático dentro del cual estuvieran inmersas todas las matemáticas y que fuera consistente.

Este programa recibió un fuerte golpe a manos del matemático austriaco Kurt Gödel, quien en 1931 demostró que cualquier sistema formal lo suficientemente fuerte como para contener a la aritmética o es incompleto (es decir, existen verdades no demostrables en el sistema) o es inconsistente (*i.e.* contradictorio).

Pero aunque el trabajo de Gödel destruyó el programa de Hilbert, ayudó al desarrollo de un campo descuidado en matemáticas: el de determinar qué métodos son válidos en la resolución de problemas.

Gödel en ese trabajo desarrolló el concepto de *funciones recursivas,* las cuales fueron posteriormente propuestas como la contraparte formal de la noción vaga e intuitiva de *función calculable.*

Otros matemáticos y lógicos estaban tratando de hacer precisamente esto, dar una respuesta satisfactoria a la pregunta sobre qué queremos decir cuando afirmamos que una función es efectivamente calculable. De aquí surgieron varios conceptos, aparte del de las funciones recursivas de Gödel: las *funciones* λ-*definibles* de Church y las *funciones Turing-computables* de Turing.

En 1936, se demostró que todos estos conceptos, aunque superficialmente diferentes, eran equivalentes.

Con la proliferación de las computadoras, los estudios en lógica y teoría de algoritmos han adquirido nuevo ímpetu.

Originado por estudios para modelar el funcionamiento del cerebro, surgió el concepto de *autómata*; y posteriormente se halló su interrelación con los lenguajes formales. La versión más general de autómata es la *máquina de Turing.*

De un tiempo a la fecha se han desarrollado otros vínculos con la lógica dentro del campo de la denominada *inteligencia artificial*, como son la *demostración automática de teoremas*, la *programación lógica*, etc.

Capítulo 2
Preliminares de teoría de conjuntos

Una falacia de ambigüedad:
Si todo fuera expresable en la teoría de conjuntos entonces el conjunto vacío tendría la cualidad de omnipresencia, pues está en todo conjunto; de lo cual se sigue que Dios, siendo el único Ser omnipresente, sería el vacío. Por lo tanto, Dios no existe.

Cultura matemática popular

Este es un capítulo de referencia, cuyo objetivo es uniformizar terminología y notación en todo el texto. Una exposición intuitiva y detallada sobre estos temas se puede leer en el libro de Halmos [Ha].

2.1 Definiciones básicas

Podemos pensar en los conjuntos como colecciones de objetos totalmente determinadas por sus elementos. Generalmente denotaremos a los conjuntos con letras mayúsculas y a sus elementos con letras minúsculas.

La relación básica es la de *pertenencia*. Si x es un elemento de un conjunto S decimos que x *pertenece* a S y escribimos $x \in S$. De no ser así, escribimos $x \notin S$.

Dos conjuntos son iguales si y sólo si tienen exactamente los mismos elementos. Hay dos formas de describir a los conjuntos:

1. Por extensión. Dando una lista de todos los elementos del conjunto. Así, $A = \{s_1, \ldots, s_n\}$ quiere decir que A es el conjunto cuyos elementos son $s_1, \ldots, s_n$ y sólo ellos.

2. Por comprehensión. Dando una propiedad satisfecha por todos los elementos del conjunto y sólo por ellos. Si P es una propiedad, $A = \{x : P(x)\}$ quiere decir que A es el conjunto de todos aquellos objetos que tienen la propiedad P.

Ejemplos:

a. $a \in \{a\}$

b. $\{x, y\} = \{y, x\} = \{x, x, y\}$

c. $\{2, 3, 5\} = \{x : x \text{ es primo y } 1 < x < 7\} = \{x : x^3 - 10x^2 + 31x - 30 = 0\}$.

Al conjunto que no tiene elementos se le conoce como *conjunto vacío* y se denota por $\varnothing$. Una manera de definirlo por comprehensión es $\varnothing = \{x : x \neq x\}$.

> **Definición.** Sean A y B dos conjuntos. Decimos que A *está contenido* en B o que A es un *subconjunto* de B si y sólo si todo elemento de A es a su vez un elemento de B. Notación: $A \subseteq B$.

Proposición 2.1. *Para cualesquiera dos conjuntos A y B se tiene que $A = B$ si y sólo si $A \subseteq B$ y $B \subseteq A$.*

Notación. Si $A \subseteq B$ pero A no es igual a B entonces escribimos $A \subset B$, y decimos que A es un subconjunto *propio* de B.

Proposición 2.2. *Sean A, B y C conjuntos arbitrarios. Entonces*

1. $\varnothing \subseteq A$.

2. $A \subseteq A$.

3. *Si $A \subseteq B$ y $B \subseteq C$ entonces $A \subseteq C$.*

Ejercicios

1. Demuestre la proposición 2.1.
2. Demuestre la proposición 2.2.
3. Pruebe que el conjunto vacío es único.

2.2 Operaciones con conjuntos

Definición. Si A es un conjunto entonces el *conjunto potencia* de A es el conjunto $\mathscr{P}(A) = \{X : X \subseteq A\}$. Es decir, que para toda X, $X \in \mathscr{P}(A)$ si y sólo si $X \subseteq A$. En particular $\varnothing \in \mathscr{P}(A)$ y $A \in \mathscr{P}(A)$.

Definición. Sean A y B dos conjuntos arbitrarios. Se definen los siguientes conjuntos:

$A \cup B := \{x : x \in A \text{ o } x \in B\}$ (la *unión* de A y B)
$A \cap B := \{x : x \in A y x \in B\}$ (la *intersección* de A y B).
$A \setminus B := \{x : x \in A \text{ y } x \notin B\}$ (la *diferencia*)
$A \Delta B := (A \cup B) \setminus (A \cap B)$ (la *diferencia simétrica*)

En general, si $\mathscr{F}$ es una familia de conjuntos se definen $\bigcup \mathscr{F}$ y $\bigcap \mathscr{F}$ como:

$$\bigcup \mathscr{F} := \{x : x \in B \text{ para algún } B \in \mathscr{F}\}$$

y

$$\bigcap \mathscr{F} := \{x : x \in B \text{ para todo } B \in \mathscr{F}\}.$$

Definición. Dos conjuntos A y B son *ajenos* si y sólo si $A \bigcap B = \varnothing$.

Muchas veces es conveniente introducir un conjunto fijo $\mathscr{U}$ tal que todo conjunto considerado sea subconjunto de $\mathscr{U}$. A este conjunto se le llama el *conjunto universal.* Entonces se puede hablar del *complemento* de un conjunto A, denotado por A^c o por A', que es el conjunto de todos los elementos (de $\mathscr{U}$) que no pertenecen a A. Esto es, $A' = \mathscr{U} \setminus A$.

Teorema 2.3. *Sean A y B dos conjuntos contenidos en algún conjunto $\mathcal{U}$. Entonces:*

$A \cup (B \cup C) = (A \cup B) \cup C$, $A \cap (B \cap C) = (A \cap B) \cap C$	(Asociatividad)
$A \cup B = B \cup A$, $A \cap B = B \cap A$	(Conmutatividad)
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$, $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	(Distributividad)
$A \setminus B = A \cap B'$	
$A \Delta \mathcal{U} = A'$	
$(A')' = A$	(Idempotencia)
$(A \cup B)' = A' \cap B'$	(De Morgan)
$(A \cap B)' = A' \cup B'$	(De Morgan)

Demostración.

Ejercicio para el lector. □

2.3 Relaciones

El *par ordenado* (a, b) se define como $\{\{a\}, \{a, b\}\}$. Para $n \in \mathbb{N}$, la *n-ada ordenada* $(a_1, \ldots, a_{n-1}, a_n) := ((a_1, \ldots, a_{-1}), a_n)$ (aquí, (a_1) se define como a_1). Esta definición está dada por *recursión*, véase sección 2.5.

La definición dada (usando el lenguaje de la teoría de conjuntos) se debe a Kuratowski. Es posible dar otras definiciones, a condición de que rescaten la idea de sucesión ordenada, expresada en la proposición siguiente.

Proposición 2.4. *$(a_1, \ldots, a_n) = (b_1, \ldots, b_n)$ si y sólo si para toda i, $1 \leq i \leq n$, se tiene que $a_i = b_i$.*

El *producto cartesiano* de dos conjuntos A y B, denotado $A \times B$, es el conjunto $A \times B := \{(a, b) : a \in A \text{ y } b \in B\}$. $A \times A$ también se denota por A^2. También por recursión se puede definir, para $n \in \mathbb{N}$, el conjunto $A^n := A^{n-1} \times A = \{(a_1, \ldots, a_n) : a_1, \ldots, a_n \in A\}$.

Definición. Sean A y B dos conjuntos. Una *relación* de A en B es un subconjunto de $A \times B$. Notación: Si R es una relación escribimos aRb en lugar de $(a, b) \in R$.

Definición. Si R es una relación de A en B el *dominio* de R es el conjunto $\text{dom}\, R := \{x \in A : \exists y \in B((x, y) \in R)\}$, el *rango* de R es el conjunto $\text{rang}\, R := \{y \in B : \exists x \in A((x, y) \in R)\}$ y el *campo* de R, $\text{cam}\, R := \text{dom}\, R \cup \text{rang}\, R$.

Una *relación n-aria* en un conjunto A es un subconjunto de A^n.

Ejemplos:

1. $\varnothing$ es una relación n-aria en A para todo A.
2. La identidad en A, $I_A := \{(x, x) : x \in A\}$ es una relación binaria en A.
3. A^n es una relación n-aria en A.
4. Las relaciones unarias en A son los subconjuntos de A.

Definiciones. Sea R una relación binaria en A, decimos que R es:

- **a.** *Reflexiva,* si y sólo si $\forall a \in A\ (aRa)$
- **b.** *Antirreflexiva,* si y sólo si $\forall a \in A\ ((a, a) \notin R)$
- **c.** *Simétrica,* si y sólo si $\forall a, b \in A\ (aRb \Rightarrow bRa)$
- **d.** *Antisimétrica,* si y sólo si $\forall a, b \in A\ (aRb \text{ y } bRa \Rightarrow a = b)$
- **e.** *Transitiva,* si y sólo si $\forall a, b, c \in A\ (aRb \text{ y } bRc \Rightarrow aRc)$
- **f.** *Conexa*, si y sólo si $\forall a, b \in A\ (aRb \text{ o } bRa \text{ o } a = b)$.

Definición. Sea A un conjunto. Un *orden parcial* en A es una relación reflexiva, antisimétrica y transitiva en A. Un *orden total* en A es un orden parcial conexo. Notación: $\leq_A$ o simplemente $\leq$.

Dado un orden parcial en A, $\leq$, podemos definir una relación binaria $<$ en A como $a < b$ si y sólo si $a \leq b$ y $a \neq b$. $<$ es un orden *estricto.*

Los órdenes parciales pueden ser representados gráficamente mediante árboles (Figura 2.1).

El diagrama anterior representa un orden parcial en el conjunto $A = \{a, b, c, d, e\}$ tal que a es el elemento máximo, d y c son incomparables, b es incomparable con d y c, pero es mayor que e y menor que a.

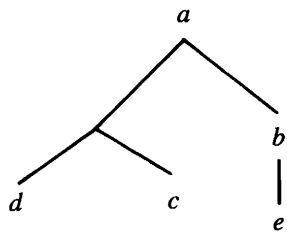


Figura 2.1

Las ramas de un árbol representan órdenes totales, pues en una misma rama todos los elementos son comparables entre sí. Un orden total se representa como un árbol con una sola rama.

Ejercicios

1. Probar la proposición 2.4, usando la definición dada.

***2.** Analice la definición siguiente de n-ada ordenada:

$$\begin{aligned}(a_1, \ldots, a_n) &:= \{\{a_1, \ldots, a_n\}, \{a_1, \ldots, a_{n-1}\}, \ldots, \{a_1\}\} \\ &= (\{a_1, \ldots, a_n\}, (a_1, \ldots, a_{n-1}))\end{aligned}$$

3. Definir un orden parcial pero no total en $\mathbb{N}$, el conjunto de los números naturales.

4. Definir seis relaciones binarias en algún conjunto A de forma tal que cada relación satisfaga únicamente una de las condiciones (a)–(f) de las definiciones de la página anterior.

Definición. Una relación binaria en un conjunto A es una *relación de equivalencia* (*releq*) si y sólo si es una relación reflexiva, simétrica y transitiva.

Definición. Si A es un conjunto y R es una relación de equivalencia en A, para cada $a \in A$ definimos la *clase de equivalencia* de a bajo R ($[a]_R$) como el conjunto $[a]_R := \{x \in A : xRa\}$. Si $a, b \in A$ entonces aRb si y sólo si $[a]_R = [b]_R$.

Las clases de equivalencia bajo R constituyen una *partición* de A, esto es:

1. Cada clase de equivalencia es no vacía.
2. Dos clases de equivalencia distintas son ajenas.
3. La unión de todas las clases de equivalencia es A.

Ejercicio

Probar esta última afirmación.

Ejemplos:

1. La identidad en A es una releq en A. Cada clase de equivalencia contiene un único elemento.

2. $A \times A$ es una releq en A, que tiene una sola clase de equivalencia, a saber, todo A.

3. En $\mathbb{N}$ la relación de congruencia módulo n es una releq que tiene exactamente n clases de equivalencia.

2.4 Funciones y cardinalidad

Definiciones. Sean A y B dos conjuntos. Una *función* f de A en B es una relación de A en B tal que $\text{dom } f = A$ y para cada $a \in A$ existe un único $b \in B$ tal que $(a, b) \in f$. Notación: $f: A \to B$. Si $(a, b) \in f$, escribimos $f(a) = b$.
Si $f: A \to B$ y $g: B \to C$ entonces se define la *composición de* f y g, $g \circ f: A \to C$ por medio de la regla $g \circ f(a) = g(f(a))$, para $a \in A$.

Definición. Una *operación n-aria* sobre un conjunto A es una función de A^n en A.

Definiciones. Sea $f: A \to B$ una función. Entonces decimos que:

a. f es *inyectiva* o 1-1 si y sólo si $\forall a_1, a_2 \in A(f(a_1) = f(a_2) \Rightarrow a_1 = a_2)$
b. f es *suprayectiva* o *sobre* si y sólo si $\forall b \in B\, \exists a \in A(f(a) = b)$
c. f es *biyectiva* si y sólo si f es inyectiva y sobre.

Si existe una función biyectiva entre A y B entonces los elementos de A están en correspondencia con los elementos de B de forma tal que a cada elemento de A le corresponde uno de B y viceversa y por tanto tienen el "mismo número de elementos".

> **Definiciones.** Se dice que dos conjuntos tienen la misma *cardinalidad* si y sólo si existe una función biyectiva entre ellos. Un conjunto A es *finito* si y sólo si es vacío o existe $n \in \mathbb{N}$ tal que $\{1, \ldots, n\}$ tiene la misma cardinalidad que A. Un conjunto que no es finito es *infinito*.

No todos los conjuntos infinitos tienen la misma cardinalidad. Un conjunto es *numerable* si y sólo si es biyectable con $\mathbb{N}$. Un conjunto es *contable* si y sólo si es finito o numerable.

Ejemplos:

1. El conjunto de los enteros, $\mathbb{Z}$, es numerable.
2. $\mathbb{Q}$, el conjunto de los números racionales es numerable.
3. $\mathbb{R}$, el conjunto de los números reales no es numerable.

Proposición 2.5. *La unión de una familia numerable de conjuntos numerables es numerable. Cf. [Ha].*

Ejercicios

1. Probar que la composición de funciones inyectivas (resp. suprayectivas, biyectivas) es inyectiva (resp. suprayectiva, biyectiva).
2. Probar que si $A \subseteq B$ y A es infinito entonces B es infinito.

2.5 Inducción matemática

A fines del siglo XIX, cuando se trataba de fundamentar la matemática por medio de sistemas axiomáticos, Giuseppe Peano (1858–1932) formuló un sistema axiomático para los números naturales. Los *conceptos primitivos* (es decir, no definidos) de su teoría eran los siguientes: "conjunto", "sucesor" y "pertenece a".

Los 5 axiomas de Peano son los siguientes:

1. 0 es un *número natural.*[1]

2. 0 no es el *sucesor* de ningún *número natural.*

3. Todo *número natural* tiene un *sucesor.*

4. Dos *números naturales* con el mismo *sucesor* son iguales.

5. Si S es un *conjunto* de *números naturales* tal que el 0 *pertenece* a S y cada vez que un *número natural pertenece a* S también su *sucesor* entonces S contiene a todos los *números naturales.*

El quinto axioma de Peano se conoce como el *Principio de Inducción Matemática*, y puede expresarse de la siguiente manera:

Principio de inducción matemática

Sea $\mathbb{N}$ el conjunto de los números naturales. Sea P una propiedad de números arbitraria y sea $S = \{n \in \mathbb{N} : P(n)\}$. Supongamos que:

(i) $0 \in S$ y

(ii) $\forall n \in \mathbb{N} \quad (n \in S \Rightarrow n + 1 \in S)$.

Entonces $S = \mathbb{N}$.

El principio de inducción matemática proporciona un método para demostrar que una propiedad P es satisfecha por todos los números naturales. En efecto, si queremos probar que todos los naturales satisfacen una cierta propiedad P basta con probar:

(i) Que 0 satisface P. (Base inductiva)

(ii) Que cada vez que un número n satisface P también $n+1$ satisface P. (Paso inductivo)

Si definimos a $S := \{n \in \mathbb{N} : P(n)\}$, por el principio de inducción matemática tendremos que $S = \mathbb{N}$, esto es, todo número natural satisface P.

Cuando se está demostrando algo por inducción, para probar (ii) se supone que un número arbitrario n satisface P (ésta es la hipótesis de inducción o H.I.) y a partir de esta suposición se demuestra que $n + 1$ también satisface P.

[1] De manera indistinta se puede tomar 0 o 1 como primer elemento, nosotros, por convención, elegiremos al 0.

Ejemplo Probar que para todo $n \in \mathbb{N}$, $n^3 + 2n$ es divisible entre 3.

Base inductiva.

Tenemos que probar que la propiedad es satisfecha por 0, esto es, que $0^3 + 2(0)$ es divisible entre 3. Pero $0^3 + 2(0) = 0$.

Paso inductivo.

Suponemos que la afirmación es verdadera para algún número natural arbitrario m, esto es:

H.I. $m^3 + 2m$ es divisible entre 3.

A partir de esta hipótesis tenemos que probar que $(m + 1)^3 + 2(m + 1)$ es divisible entre 3.

$$\begin{aligned}(m+1)^3 + 2(m+1) &= m^3 + 3m^2 + 3m + 1 + 2m + 2 \\ &= m^3 + 2m + 3m^2 + 3m + 3 \\ &= (m^3 + 2m) + 3(m^2 + m + 1)\end{aligned}$$

Por H.I. el primer sumando es divisible entre 3 y por tanto la propiedad es verdadera para $m + 1$. Esto concluye la prueba. □

En lógica se utiliza más otra versión del método de demostración por inducción matemática, que está basado en el siguiente teorema.

Teorema 2.6. *(Principio de Inducción Matemática Fuerte).*
Sea $S \subseteq \mathbb{N}$ tal que:

(i) $0 \in S$

(ii) Para $m \in \mathbb{N}$ arbitrario, si $k \in S$ para toda $k < m$ entonces $m \in S$.

Entonces $S = \mathbb{N}$

Demostración.

Supongamos que existe un conjunto S que satisface las hipótesis del teorema pero no la conclusión, es decir $S \subset \mathbb{N}$. Entonces $\mathbb{N} \setminus S$ no es vacío. Sea m el menor elemento de $\mathbb{N} \setminus S$. Por (i), $m > 0$ y además, si $n < m$ entonces $n \in S$ (por elección de m). La hipótesis (ii) implica que $m \in S$. Esta contradicción concluye la prueba. ■

Nótese que el paso crucial en la demostración anterior fue el hecho de asegurar la existencia del menor elemento de $\mathbb{N} \setminus S$ a partir de que este último conjunto es no vacío. Esto se debe a que los naturales están bien ordenados. De hecho, se puede demostrar que el buen orden de $\mathbb{N}$ es equivalente al principio de inducción. Ver [Ha] para una demostración de esto.

Para probar que una propiedad es satisfecha por todos los números naturales usando el principio de inducción matemática fuerte se procede exactamente igual que para demostraciones por inducción normal, la única diferencia es que la hipótesis de inducción en el paso inductivo es distinta:

H.I. Supongamos que toda $k < m$ satisface P

A partir de H.I. se prueba que m satisface P.

El principio de inducción matemática también puede usarse para *definir* funciones con dominio $\mathbb{N}$. Este tipo de definición se llama por *recursión.*

Proposición 2.7. *Si se quiere definir una función f con dominio $\mathbb{N}$ es suficiente con:*

1. *Dar una regla para calcular $f(0)$ y*
2. *Dar una regla para calcular $f(n)$ en términos de $\{f(m) : m < n\}$.* ■

Nota: El principio de inducción enunciado en esta sección puede ser modificado para demostrar que una propiedad P es satisfecha por todo número natural mayor o igual a un cierto $n \in \mathbb{N}$. Para hacer esto basta probar, como base de la inducción, que n satisface a P.

A lo largo del texto se encontrarán muchos ejemplos de demostraciones por inducción y definiciones por recursión.

Ejercicios

1. En el Teorema 2.6 la hipótesis (i) es innecesaria. ¿Por qué?

2. ¿Qué está mal en la prueba siguiente?

(i) 0 es un número interesante;

(ii) Supongamos que n es un número interesante, entonces $n + 1$ también lo es; pues en caso contrario, éste sería el primer número que no es interesante, lo cual lo convertiría en un número realmente interesante.

Por tanto, por el principio de inducción matemática, "todos los números naturales son interesantes".

3. Demuestre que cada ser humano es un descendiente de Adán y Eva. Para esto, use inducción matemática fuerte y especifique qué se entiende por la relación de "descendencia" sobre la raza humana. (Sugerencia: defina cláusulas para descendencia, como "todos, excepto Adán y Eva, tienen padres", etc.; y asigne un rango a cada persona).

4. Pruebe por inducción matemática que para todo $n \in \mathbb{N}$, $n^4 - 4n^2$ es divisible entre 3.

5. ¿Qué está mal en la prueba siguiente?

Teorema. *Para todo* $n \in \mathbb{N}$, $a^n = 1$.

Prueba. Denotemos con $h(k)$ a la expresión: "si $n \leq k$, $a^n = 1$."

Base inductiva: para $k = 0$, $a^0 = 1$.

Paso inductivo: Supongamos que la afirmación es válida para un $k \in \mathbb{N}$. Entonces,

$$a^{k+1} = \frac{a^k \times a^{k-1}}{a^{k-2}} = \frac{1 \times 1}{1} = 1$$

Capítulo 3
Lenguajes y sistemas formales

Una interrogante de la mayor importancia será el que si es teóricamente posible igualar el nivel de nuestras capacidades mentales, a través del empleo de algún sistema formal.

Douglas R. Hofstadter

3.1 Introducción

La lógica matemática moderna tiene sus orígenes en el sueño de Leibniz de un cálculo simbólico universal que comprehendiera toda la actividad mental de naturaleza lógica rigurosa, en particular todas las matemáticas. Para Leibniz, este cálculo simbólico universal sería una ciencia limitada únicamente por la necesidad de obedecer las leyes de la lógica. Esta ciencia general proveería, antes que nada, un lenguaje racional universal que se adaptaría al pensamiento. Sus conceptos, simplificados en conceptos primitivos y distintos, se podrían combinar de una manera casi mecánica. También pensó que un simbolismo sería necesario para evitar que la mente se confundiera. Este sueño fue demasiado ambicioso para que Leibniz lo realizara. Sin embargo, Boole, Frege, Peano, Russell, Hilbert, Skolem, Tarski y otros, con métodos abstractos más poderosos y motivados, algunos de ellos, por problemas en los fundamentos de la matemática, lograron realizar una parte significativa del sueño de Leibniz.

Durante muchos siglos los *Elementos* de Euclides fueron considerados como el paradigma del pensamiento riguroso en matemáticas. Euclides intentó derivar

todos los teoremas conocidos de la geometría a partir de un número relativamente pequeño de nociones comunes (proposiciones fundamentales verdaderas para todas las ciencias) y postulados (proposiciones geométricas evidentes). Y si bien es cierto que no logró cabalmente su propósito, pues en algunos lugares supuso proposiciones que no estaban entre los postulados ni se seguían de ellos, su intento fue magistral y los *Elementos* quedaron como el mejor ejemplo de un sistema deductivo.

Cuando surgió el problema de la fundamentación de la matemática a fines del siglo pasado y principios de éste, el ideal de muchos matemáticos fue el de reescribir todas las teorías matemáticas conocidas como sistemas deductivos, siguiendo el ejemplo de los *Elementos*. Muchos matemáticos empezaron a reconsiderar la relación entre la matemática y la lógica, y algunos de ellos incluso llegaron a sugerir que la matemática se podría fundamentar en la lógica. Fue a partir de esta idea que la lógica simbólica fue desarrollada como un sistema deductivo por Russell y Whitehead (*Principia Mathematica*), principalmente. La lógica matemática fue, pues, en un principio un modelo matemático del pensamiento deductivo. Pero, al igual que muchas disciplinas, ha crecido más allá de las circunstancias de su nacimiento.

3.2 Lenguajes formales

La matemática siempre ha utilizado símbolos particulares para expresar sus resultados: $+$ para representar a la suma, $\int$ para la integral, $\in$ para denotar la pertenencia a un conjunto, etc. Este lenguaje particular de la matemática es un lenguaje semiformalizado, que toma de los lenguajes naturales (como el español o el inglés) lo que necesita y agrega símbolos para hacer los resultados más precisos. Pero estos símbolos tienen "reglas gramaticales" precisas, de tal modo que "$3 + 4 = 7$" es una expresión que tiene sentido, mientras que "$+++ = 8-$" no lo tiene. En el intento de formalizar la lógica se estudiaron estos lenguajes semiformalizados de la matemática y surgió el concepto de *lenguaje formal*. Un lenguaje formal está dado por un conjunto de símbolos que se combinan entre sí para formar *expresiones bien formadas* mediante *reglas de formación* especificadas de antemano. Las expresiones bien formadas son todas las expresiones "gramaticalmente correctas" del lenguaje formal.

Hay muchos lenguajes formales, cada uno con símbolos y reglas de formación distintos. Cada teoría matemática requiere de un lenguaje formal propio, con símbolos adecuados para expresar los teoremas de la teoría. Pero es importante notar que los símbolos de un lenguaje formal carecen de significado. Se les puede asignar un significado, si se quiere, pero no tienen ningún significado fijo de antemano. Las manipulaciones de símbolos para formar expresiones bien formadas son puramente mecánicas.

La lógica matemática se dedica, entre otras cosas, al estudio de los lenguajes formales. Para estudiarlos y hablar sobre ellos se requiere, evidentemente, de un metalenguaje, que puede ser el español o algún lenguaje semiformalizado (véase sección 1.3).

En los capítulos siguientes tendremos oportunidad de estudiar varios lenguajes formales, algunos capaces de formalizar el pensamiento deductivo. Por el momento nos conformaremos con dar un ejemplo de un lenguaje formal sencillo al que llamaremos $\mathscr{L}_M$ (*cf.* [Ho]).

Ejemplo.

Símbolos de $\mathscr{L}_M$: las letras M, I, U.

Reglas de formación de $\mathscr{L}_M$:

Sólo una regla, R: Toda sucesión finita de símbolos de $\mathscr{L}_M$ es una expresión bien formada de $\mathscr{L}_M$.

Con estos dos elementos, los símbolos y la regla de formación, tenemos perfectamente definido a $\mathscr{L}_M$. Como ejemplos de expresiones bien formadas de $\mathscr{L}_M$, tenemos:

$$UUUU, \quad U, \quad I, \quad M, \quad MII, \quad MIU, \quad MUU, \quad \text{etc.}$$

Evidentemente, éste es un lenguaje formal que no parece tener mucha utilidad para estudiar estructuras matemáticas, pero es un lenguaje formal bien definido. □

Supongamos ahora que deseamos buscar expresiones que puedan representar integrales de funciones.

Ejemplo. Consideremos el alfabeto $\mathscr{A} = \{\int, a, f, x, d,), (, \circ\}$. Entonces las reglas siguientes nos permiten obtener *expresiones bien formadas* (ebf) sobre $\mathscr{A}$, que de ser interpretadas tendrán sentido para nosotros:

R1: $\int x\,dx$ y $\int f(x)\,dx$ son ebf's.

R2: Si α[1] es una ebf, entonces $\int \alpha\,dx$ es una ebf.

R3: Si α es una ebf, entonces $\int a\alpha\,dx$ y $a\int \alpha\,dx$ son ebf's.

R4: Si α es una ebf, entonces $\int f\circ\alpha\,dx$ y $\int f(\alpha)\,dx$ son ebf's.

R5: Sólo son ebf's aquéllas construidas con base en R1–R4.

Así, serán expresiones bien formadas las siguientes:

a) $\int f(x)\,dx$

b) $\int\int f(x)\,dxdx$

c) $a\int\int f(x)\,dxdx$

d) $\int f(\int x\,dx)\,dx$, etcétera.

Pero no lo serán las expresiones: $)f\int(dxa$, $\int f(x)d, dx\int\int\int)af$, etcétera. □

A partir de estos ejemplos procedemos a dar una definición de un lenguaje formal en general.

Definición. Un *alfabeto* $\mathscr{A}$ es un conjunto contable de símbolos.

Definición. Una *expresión* es cualquier sucesión finita de símbolos construida sobre un alfabeto $\mathscr{A}$, incluyendo a la palabra vacía, Λ. Así, si $\mathscr{A}^* = \{\text{expresiones sobre } \mathscr{A}\}$, $\mathscr{A}^* = \bigcup_{i\in\mathbb{N}}\mathscr{A}^i$, donde $\mathscr{A}^i = \mathscr{A}\times\ldots\times\mathscr{A}$ (i veces) y $\mathscr{A}^0 = \{\Lambda\}$.[2]

E.g., si $\mathscr{A} = \{a, b\}$, entonces $\mathscr{A}^2 = \{aa, ba, ab, bb\}$, etc.

Definición. Un *lenguaje formal* $\mathscr{L}$ es una pareja ordenada $(\mathscr{A}, \mathscr{E})$, donde $\mathscr{A}$ es un alfabeto y $\mathscr{E}\subseteq\mathscr{A}^*$ es el conjunto de *expresiones bien formadas* (ebf) sobre $\mathscr{A}$.

[1] Observe que α no es un símbolo del lenguaje, sino del metalenguaje.

[2] En este caso, por simplicidad, se identifica la pareja ordenada (a, b) con ab.

3.3 Cálculos asociativos y el problema de las palabras

La primera impresión, de plantear un lenguaje formal para el que toda cadena de símbolos de su alfabeto sea "gramaticalmente correcta" (*i.e.*, una ebf), puede parecer un tanto ocioso (*e.g.*, el lenguaje dado por $\mathscr{L}_M$). En esta sección, consideramos precisamente "sistemas" cuyos lenguajes formales son de este tenor trivial. Estos sistemas constituyen *cálculos* dado que están provistos de ciertas reglas que permiten obtener "nuevas expresiones" a partir de expresiones dadas de antemano. La razón para estudiar estos cálculos surgió de la necesidad de dar precisión al concepto intuitivo y vago de *algoritmo*. Originalmente planteado por Thue (1914), el *problema de las palabras* se convirtió en el punto de partida por medio del cual varios matemáticos (Markov, Post y Novikov) se abocaron a esta tarea de formalizar lo que significa un algoritmo. Desde una perspectiva intuitiva, un *algoritmo* es un procedimiento mediante el cual obtener una solución a un problema específico partiendo de un conjunto de datos (entradas) y a través de un número finito y determinado de pasos[3]. Esta noción de algoritmo es satisfactoria si lo que se pretende es dar una respuesta positiva a si determinado problema es soluble algorítmicamente. Para el caso, basta con exhibir un algoritmo que lo resuelva. Sin embargo, cuando no exista tal procedimiento, esta versión no es adecuada y se requiere de una definición formal. Esto es, porque dar una respuesta negativa significa, no sólo que no se ha hallado el algoritmo que resuelve el problema sino, ¡que jamás podrá encontrarse, pues no existe tal algoritmo!

En esta sección, si consideramos un lenguaje formal $\mathscr{L}$, éste constará de un alfabeto finito $\mathscr{A}$ y el conjunto de sus expresiones bien formadas serán todas las expresiones construidas a partir de $\mathscr{A}$, *i.e.*, $\mathscr{E} = \mathscr{A}^*$. Por tanto, podemos obviar al lenguaje formal $\mathscr{L}$ y hablar sólo del conjunto generado sobre su alfabeto, $\mathscr{A}^*$. A las cadenas de símbolos de $\mathscr{A}$ las denominamos *palabras*.

> **Definición.** Si una palabra α es parte de una palabra β, *i.e.*, la sucesión de signos de α es una subsucesión de la sucesión β, decimos que en β hay una *ocurrencia* de α.

[3]En contrapartida, un procedimiento que pueda llevar a una solución de un problema, pero sin garantía de que la halle (termine) se denomina *heurístico*

Las reglas para las transformaciones de una palabra dada en otra se darán mediante las siguientes sustituciones, que se llaman *sustituciones admisibles.*

> **Definición.** Sean α, β y γ tres palabras de $\mathscr{A}^*$. La *sustitución dirigida* (denotada) $\alpha \rightarrow \beta$ en la palabra γ consiste en sustituir con β alguna de las ocurrencias de α en γ (siempre y cuando α ocurra en γ). La *sustitución no dirigida* (o simplemente *sustitución*) α — β en γ consiste en sustituir α por una ocurrencia de β en γ o viceversa.

Ejemplo. Consideremos el alfabeto $\mathscr{A} = \{a, b, c\}$. La sustitución ac — bac podemos aplicarla a la palabra $bbaccb$ de varias maneras:

1. $bb\,\underline{ac}\,cb \mapsto bb\,\underline{bac}\,cb$
2. $b\,\underline{bac}\,cb \mapsto b\,\underline{ac}\,cb$

y de éstas, segundas aplicaciones darán:

3. $bbb\,\underline{ac}\,cb \mapsto bbb\,\underline{bac}\,cb$
4. $\underline{bac}\,cb \mapsto \underline{ac}\,cb$

respectivamente, etc. No así la palabra $ccba$, que no admite aplicación alguna de esta regla. □

> **Definición.** Un *cálculo asociativo* es un conjunto $\mathscr{A}^*$ formado con todas las palabras sobre un alfabeto $\mathscr{A}$, provisto de alguna colección finita de sustituciones admisibles.

A continuación, ilustraremos en qué consiste el problema de las palabras.

> **Definición.** Dos palabras α y β sobre $\mathscr{A}$ se denominan *adyacentes* si pueden transformarse una en la otra aplicando una sola sustitución admisible.

> **Definición.** Una *cadena deductiva* de palabras de α_1 hasta α_n lo constituye toda sucesión de la forma:
>
> $$\alpha_1, \alpha_2, \ldots, \alpha_n$$
>
> tal que α_i es adyacente a α_{i+1}, para $i = 1, \ldots, n-1$.

> **Definición.** Dos palabras α y β son *equivalentes* si y sólo si existe una cadena deductiva con la propiedad de que $\alpha_1 = \alpha$ y $\alpha_n = \beta$. Notación: $\alpha \sim \beta$.

Teorema 3.1. *La relación definida por la equivalencia entre palabras es, en efecto, una relación de equivalencia.* ■

Para realizar un cálculo deductivo es de particular interés el resultado siguiente.

Teorema 3.2. *Sea $\beta \sim \gamma$; entonces si β ocurre en una palabra π, al aplicar en π la sustitución $\beta \to \gamma$ se obtiene una palabra equivalente a π.*

Demostración.

Sea π la palabra $\alpha\beta\omega$ (donde α y ω pueden ser vacías, y si ambas lo son, el resultado es trivial), y probaremos que es equivalente a $\alpha\gamma\omega$.

Como $\beta \sim \gamma$, existe una cadena deductiva:

$$\beta = \beta_1, \beta_2, \ldots, \beta_n = \gamma.$$

Consideremos ahora la sucesión:

$$\alpha\beta_1\omega, \alpha\beta_2\omega, \ldots, \alpha\beta_n\omega$$

la cual es evidentemente una cadena deductiva (dado que cualesquiera dos palabras consecutivas son adyacentes) que parte desde $\pi = \alpha\beta\omega$ hasta la palabra transformada $\alpha\gamma\omega$, que es equivalente a π. ■

Del ejemplo anterior, tenemos que $accb \sim bbbbaccb$, mientras que la palabra $ccba$ no tiene ninguna otra palabra equivalente a ella, siendo así el único elemento de su clase de equivalencia.

De esta forma, dado un cálculo asociativo podemos plantear su *problema de equivalencia de palabras* o, como es usualmente denominado, *problema de las palabras*:

(PP) *Dada una pareja de palabras cualquiera en el cálculo, determinar si son o no equivalentes.*

Para un cálculo existe un conjunto infinito de palabras posibles, y, por ende, toda una gama infinita de problemas de esta índole. La solución del PP se presenta en la forma de un algoritmo que *decide* la equivalencia o no de cualquier pareja de palabras.

Bajo la apariencia de ser un acertijo artificial (¡tal vez sea más interesante resolver un crucigrama o ver T.V.!) y de que resolverlo carezca de importancia, según Trakhtenbrot [Tr], "*nada más lejos de la verdad —el problema es bastante común y tiene importancia tanto teórica como práctica, que justifica por completo el esfuerzo desarrollado para hallar un algoritmo*".

Para algunos cálculos asociativos es posible dar con un algoritmo que resuelva su PP asociado. Para ello, una técnica radica en construir un algoritmo auxiliar: el *algoritmo de reducción.* Éste consiste en transformar cualquier palabra en una palabra equivalente de una forma particular: su *palabra reducida,* mediante aplicaciones de una colección ordenada de sustituciones *dirigidas.* De tal manera, se tiene que para α, una palabra arbitraria dada, se le aplica la primera sustitución de la colección, al resultado (que puede ser α misma, si no fue posible aplicarle esta sustitución), se procede con la segunda sustitución, etc. Eventualmente, se obtendrá una palabra a la cual ninguna de las sustituciones resulta aplicable: es la palabra reducida. Así, y permitiendo ahora que las sustituciones sean *no dirigidas,* tendremos que dos palabras serán equivalentes si y sólo si tienen la misma palabra reducida, haciendo ésta las veces de representante de la clase de equivalencia. Para que este procedimiento sea válido, sólo restaría probar que, en efecto, las palabras reducidas no son equivalentes. De [Tr], tenemos el cálculo asociativo siguiente, cuyo PP asociado se puede resolver vía el algoritmo recién expuesto.

Ejemplo. Consideremos el cálculo asociativo con alfabeto $\mathscr{A} = \{a, b, c\}$ y cuyas sustituciones admisibles son:

$$\begin{array}{ll} (1)\ b \ \text{—} \ acc & (3)\ aa \ \text{—} \ \Lambda \\ (2)\ ca \ \text{—} \ accc & (4)\ cccc \ \text{—} \ \Lambda \end{array}$$

donde Λ representa la palabra vacía.

De tomar las sustituciones dirigidas (leídas de izquierda a derecha) a partir de (1)–(4), resultan como únicas palabras reducidas las ocho siguientes:

$$\Lambda, c, cc, ccc, a, ac, acc \quad \text{y} \quad accc.$$

de las cuales, ninguna pareja es equivalente. □

Lo interesante de este ejemplo en particular, reside en que si le añadimos la operación de *concatenación* entre palabras puede dársele una interpretación

geométrica en términos de *automorfismos*[4] sobre un cuadrado. Aquí los símbolos adquieren los siguientes significados:

$\Lambda \longleftrightarrow$ *identidad*

$a \longleftrightarrow$ *reflexión sobre un eje vertical que pasa por 0.*

$b \longleftrightarrow$ *reflexión sobre un eje horizontal que pasa por 0.*

$c \longleftrightarrow$ *rotación de 90° en torno a 0 en sentido antihorario.*

donde, 0 es el centro del cuadrado en cuestión.

De esta manera, la concatenación viene a ser un *producto (composición)* entre estas transformaciones sobre el cuadrado, operación que, aunque no resulta conmutativa, provee al cálculo asociativo de la *estructura algebraica de grupo,* facilitando así la decisión sobre su PP (*cf.* [Tr]-[Se]).

Una manera alterna (y a la vez equivalente, [Po]-[K11]) de resolver un problema de palabras es seleccionando un conjunto determinado de palabras, a las cuales se les denominará *axiomas,* y limitar la aplicabilidad a sustituciones dirigidas admisibles, cuya colección se denomina *diccionario,* a actuar sobre este conjunto de axiomas. El objetivo es caracterizar las palabras *producidas* a partir de los axiomas. Este enfoque fue el adoptado por Post para abordar el PP, mientras que el de los cálculos asociativos se debe a Markov. Los resultados de estos dos matemáticos, vía una necesaria precisión del concepto de algoritmo, condujeron a que el PP (o en términos más técnicos, *el problema de las palabras para semigrupos*) "*es irresoluble; de hecho existen un alfabeto particular y un diccionario de tal forma que no existe algoritmo alguno para decidir si dadas dos palabras (formadas sobre el alfabeto) son equivalentes (por ese diccionario)*"[5]. Por consiguiente, el problema general es irresoluble.

Un ejemplo sencillo para ilustrar un *sistema de producción de Post* (que así se llaman estos sistemas) viene presentado en [Ho][6] Este sistema tiene por alfabeto al conjunto $\{M, I, U\}$, su único axioma es la palabra MI, y contempla 4 sustituciones o *reglas de producción*:

R1: $\alpha I \rightarrow \alpha IU$

R2: $M\alpha \rightarrow M\alpha\alpha$

[4]Transformaciones geométricas que transforman al cuadrado en sí mismo.

[5]Cita de Kleene [K11]

[6]Bajo la forma de un acertijo con fin de motivar los temas siguientes de su libro.

R3: $\alpha III\omega \rightarrow \alpha U\omega$

R4: $\alpha UU\omega \rightarrow \alpha\Lambda\omega = \alpha\omega$

El acertijo que propone el autor es "*¿Puede usted producir* MU?"[7]

Ejercicios

1. Pruebe el Teorema 3.1.

2. Resuelva el problema de palabras para el cálculo asociativo con alfabeto $\mathscr{A} = \{a, b\}$ y sustituciones admisibles: bb — a y bbb — Λ.

***3.** Resuelva el problema de palabras para el cálculo asociativo con alfabeto $\mathscr{A} = \{a, b, c\}$ y sustituciones admisibles:

$$\begin{array}{ll} (1)\ b \text{ — } acc & (3)\ aa \text{ — } \Lambda \\ (2)\ ca \text{ — } accc & (4)\ cccc \text{ — } \Lambda \end{array}$$

4. Considere el sistema de producción de Post dado arriba. Verifique si se pueden producir las palabras: a. $UIIMI$, b. $MIIUIIU$, c. $MUIU$, d. $MIUII$.

3.4 Sistemas formales

Como dijimos anteriormente, la tarea de reescribir las teorías matemáticas como teorías deductivas fue muy importante a principios de siglo. En esta tarea nos pueden ayudar los lenguajes formales. Dado un lenguaje formal, con sus símbolos, reglas y expresiones bien formadas, podemos empezar a construir teorías formales en ese lenguaje. Para obtener una *teoría formal* en un lenguaje formal dado se seleccionan, de entre las expresiones del lenguaje, algunas que serán los *axiomas*. Se especifican también las llamadas *reglas de inferencia*, que nos permiten deducir ebf's nuevas a partir de ebf's anteriores. A las ebf's así deducidas se les denomina *teoremas*. Dicho de manera más breve, una teoría formal $\mathscr{T}$ para un lenguaje formal $\mathscr{L}$ está dada cuando se especifican los axiomas y las reglas de

[7] Se invita al lector leer simultáneamente el libro de [Ho] con el presente texto.

inferencia. Intuitivamente, los axiomas representan enunciados cuya verdad no se cuestiona; y las reglas de inferencia representan maneras correctas de inferir nuevas afirmaciones de afirmaciones que ya se tienen. Pero debe quedar claro que esto es sólo una ayuda heurística, que los símbolos del lenguaje formal carecen de significado en sí, y por lo tanto los teoremas de una teoría formal, también carecerán de significado, serán fórmulas a las que llegamos por medio de una manipulación mecánica de símbolos.

Las teorías formales también son objeto de estudio de la lógica. Veremos en capítulos siguientes que es posible construir teorías formales, relativamente simples, que rescaten el pensamiento deductivo.

Como un primer ejemplo de una teoría formal contamos con el sistema de producción de Post, MIU, descrito en la sección anterior. De esta teoría formal ya mencionamos que su lenguaje es trivial al considerar como ebf's a todas las expresiones sobre $\{M, I, U\}$.

Definición. Una *teoría* o *sistema formal* es una estructura matemática definida por la terna $\langle \mathscr{L}, \mathscr{A}x, \mathscr{R} \rangle$, donde

(i) $\mathscr{L} = (\mathscr{A}, \mathscr{E})$, es el *lenguaje formal* sobre $\mathscr{A}$, con $\mathscr{E} = \{$expresiones bien formadas$\}$;

(ii) $\mathscr{A}x \subseteq \mathscr{E}$, es llamado el conjunto de *axiomas* del sistema; y

(iii) $\mathscr{R}$, es la colección de *reglas de inferencia (derivación, deducción* o *producción*).

A continuación presentaremos otros sistemas formales relativamente sencillos, para los cuales sus lenguajes formales no son triviales.

Ejemplos:

1. El sistema formal $\mathscr{P}_{pq}$ (debido a [Ho]). Consideremos a α, β, γ como cadenas que constan sólo de guiones. El lenguaje formal $\mathscr{L} = (\mathscr{A}, \mathscr{E})$, donde $\mathscr{A} = \{-, p, q\}$ y $\mathscr{E}$ está constituido con todas las expresiones generadas por la regla de formación siguiente:

   RF: Las ebf's son las expresiones de la forma $\alpha p \beta q \gamma$.

   Aquí tenemos un único axioma:

A: $\alpha p - q\alpha -$;

y una única regla de inferencia:

RI: Si $\alpha p\beta q\gamma$ es un teorema en $\mathscr{S}_{pq}$, entonces $\alpha p\beta - q\gamma -$ es un teorema en $\mathscr{S}_{pq}$.

Una interpretación para $\mathscr{S}_{pq}$ viene dada mediante las asignaciones a sus símbolos:

$$\begin{aligned} p &\longleftrightarrow \textit{la operación de suma} : + \\ q &\longleftrightarrow \textit{la relación de igualdad} \\ - &\longleftrightarrow \textit{uno} \\ -- &\longleftrightarrow \textit{dos} \\ &\cdots \end{aligned}$$

En otras palabras, el sistema formal $\mathscr{S}_{pq}$ ¡simplemente nos enseña a sumar!

2. El sistema formal $\mathscr{S}_{tq}$ (de Hofstadter [Ho]). De nueva cuenta, consideremos que α, β y γ son cadenas que constan exclusivamente de guiones. Como lenguaje formal tenemos a $\mathscr{L} = (\mathscr{A}, \mathscr{E})$, donde $\mathscr{A} = \{-, t, q\}$ y $\mathscr{E}$ se genera mediante la regla de formación:

RF: Las ebf's son las expresiones de la forma $\alpha t\beta q\gamma$.

Su único axioma es:

A: $\alpha t - q\alpha$

y a manera de regla de inferencia:

RI: Si $\alpha t\beta q\gamma$ es un teorema en $\mathscr{S}_{tq}$, entonces $\alpha t\beta - q\gamma\alpha$ es un teorema en $\mathscr{S}_{tq}$.

Bajo la interpretación de los símbolos:

$$\begin{aligned} t &\longleftrightarrow \textit{la operación de multiplicación} : \cdot \\ q &\longleftrightarrow \textit{la relación de igualdad} \\ - &\longleftrightarrow \textit{uno} \\ -- &\longleftrightarrow \textit{dos} \\ &\cdots \end{aligned}$$

tenemos que con este sistema formal, $\mathscr{S}_{tq}$, se ha aprehendido, y el lector con su uso habrá aprendido, el concepto de multiplicación.

3. Consideremos ahora un sistema formal (debido a Quine [Qu]) para representar a la resta. El lenguaje formal consta del alfabeto $\mathscr{A}$ formado con:

   i. las letras con o sin subíndices: $x, y, z, \ldots, x_1, y_1, z_1, \ldots$ Las denominamos variables.

   ii. los símbolos: $-, \approx$

   iii. paréntesis:), (

   y el conjunto de expresiones $\mathscr{E}$ se genera de la manera siguiente:

   RF1: toda variable es ebf;

   RF2: Si α y β son ebf's, así también lo son:

   i. $(\alpha - \beta)$

   ii. $\alpha \approx \beta$

   RF3: Algo es ebf si y sólo si se generó usando RF1 o RF2.

   Ahora se requieren de dos axiomas:

   A1: $x \approx x - (y - y)$

   A2: $x - (y - z) \approx z - (y - x)$

   y de dos reglas de inferencia:

   RI1: Si α es un teorema y β es el resultado de reemplazar una o varias ocurrencias de alguna variable en α por una ebf obtenida por RF1 y RF2i), entonces β es un teorema.

   RI2: Si α es un teorema y β es el resultado de reemplazar el lado derecho de α por el lado izquierdo de α, entonces β es un teorema.

   E.g., Aplicando RI1, con $x \to (z - x)$, $y \to z$, a A1, obtenemos:

$$(z - x) \approx (z - x) - (z - z)$$

   Mientras que una aplicación de RI2 sobre A1, sustituyendo su lado derecho por el izquierdo, produce el teorema:

$$x \approx x$$

Este sistema resulta más fuerte que el $\mathscr{P}_{pq}$ al poderse representar $x + y$ mediante la expresión $x - ((y - y) - y)$. □

Una propiedad interesante que posee este último sistema es que toda ecuación que pueda ser representada en su lenguaje y que sea verdadera bajo la interpretación resulta deducible en él. En este sentido, el sistema se dice que es *completo* [Qu].

Posteriormente volveremos a tratar con mayor amplitud esta propiedad de un sistema formal, la completud, en lo que respecta a la lógica. La completud viene a ser así una medida del grado de aprehensión de un sistema formal para representar el conocimiento motivo de su creación.

Otra propiedad importante a cuestionar sobre un sistema formal es la de su *decidibilidad.*

> **Definición.** Decimos que un sistema formal $\mathscr{S}$ es *decidible* si y sólo si existe un procedimiento efectivo (algoritmo) que decide en un número finito de pasos si una ebf es un teorema o no en $\mathscr{S}$.

Muchos sistemas formales en matemáticas son indecidibles: el problema de las palabras, el problema de la identidad en teoría de grupos, la lógica de predicados, la aritmética formal, etc. Lo interesante del problema de las palabras reside en que fue el primer sistema formal fuera del ámbito de la lógica cuya indecidibilidad se probó. En su oportunidad (Capítulo 6) analizaremos éste y otros temas afines dentro del contexto de la lógica proposicional.

Ejercicio

Con base en el sistema formal para la resta, demuestre que:

a. $z - x \approx (y - x) - (y - z)$

b. $x - y \approx (z - z) - (y - x)$

c. $x + y \approx y + x$ (el símbolo "+" no es del lenguaje, sólo es abreviatura).

Capítulo 4
Lógica proposicional: enfoque semántico

Investigar las leyes fundamentales de las operaciones de la mente mediante las cuales el razonamiento es desempeñado, dar expresión de ellas en el lenguaje simbólico de un Cálculo, y bajo este fundamento establecer la ciencia de la Lógica y construir su método.

George Boole

4.1 Introducción

En este capítulo vamos a estudiar un lenguaje formal como los definidos en la sección precedente. Vamos a dar la lista de sus símbolos, sus reglas de formación y hemos de construir una teoría formal para ese lenguaje. También vamos a dar una interpretación para sus símbolos que nos ayudará para estudiar este lenguaje desde un punto de vista distinto al sintáctico, en el cual se estudian axiomas y reglas de inferencia. Este otro punto de vista es el llamado enfoque *semántico*, que es muy importante cuando uno estudia formalizaciones de teorías matemáticas.

Como hemos dicho anteriormente, un lenguaje formal puede ser estudiado como objeto abstracto, sin asignarle ningún significado a los símbolos, y estudiando a las teorías formales como sucesiones de expresiones de un lenguaje que obedecen ciertas reglas. Sin embargo, cuando se utilizan los lenguajes formales para

reescribir teorías matemáticas o de otro tipo, es conveniente construir un lenguaje formal que pueda ser interpretado de tal forma que sus fórmulas bien formadas expresen los enunciados de la teoría original. Así, un lenguaje formal tendrá una doble dimensión: la puramente sintáctica, sin significado, pero cuyo estudio nos proporciona más elementos para conocer a la teoría original; y la dimensión semántica, en la cual se tiene en mente el significado que se pretende dar a los símbolos, y cuyo estudio determina, de alguna manera, la teoría formal que se construirá en el lenguaje formal dado. Estos dos enfoques se complementan y se enriquecen mutuamente, como se verá en éste y el siguiente capítulo.

Recordemos que nuestro objetivo es la construcción de un modelo formal del pensamiento deductivo. El modelo que presentaremos en este capítulo es un primer intento, no rescata totalmente el pensamiento deductivo humano, pero tiene las características esenciales de modelos más sofisticados y es fácil de manejar, por eso lo presentaremos con cierto detalle.

Una buena manera de definir a la lógica es definirla como el estudio o análisis de los métodos correctos de razonamiento. El razonamiento deductivo se presenta en forma de argumentos: listas de proposiciones relacionadas de tal manera que la última, llamada conclusión del argumento se sigue de las anteriores, llamadas premisas del argumento. A un lógico no le interesa si las premisas o conclusión de un argumento son verdaderas o no, lo importante para un lógico es si la verdad de la conclusión se sigue de la verdad de las premisas. De modo que para un lógico los siguientes dos argumentos son correctos:

(1) *Todos los hombres son mortales*
Sócrates es hombre
Luego, Sócrates es mortal.

(2) *Todos los números son verdes*
El 5 es un número
Luego, el 5 es verde.

El argumento (2) es correcto aun cuando su conclusión sea falsa, pues si ambas premisas fueran verdaderas, estaríamos obligados a aceptar la verdad de la conclusión.

Antes de continuar es conveniente detenernos a pensar en lo que generalmente se entiende por "proposición". Una *proposición* es lo que se dice de algo. Lo esencial de una proposición es que expresa algo que puede ser verdadero o falso.

Si consideramos la siguiente expresión en español:

"Asómate, luz de mis ojos, para admirar tu belleza"

vemos que no le podemos asignar un valor de verdad, no tiene sentido afirmar que sea verdadera o falsa. Sin embargo, consideremos la siguiente expresión:

"México es la capital de China"

ésta es una oración de la cual podemos afirmar que es falsa, por tanto es una proposición.

Ejercicio

Determine si las oraciones siguientes son proposiciones o no:

i. Si una función es continua, entonces es derivable.
ii. Todo ser de nariz larga es Pinocho.
iii. En un lugar de la Mancha, de cuyo nombre no quiero acordarme.
iv. Robó, huyó y lo pescaron.
v. Yo miento.
vi. Esta oración es falsa.

4.2 Lenguaje formal de proposiciones

Las proposiciones pueden ser combinadas entre sí para obtener nuevas proposiciones. Así, si A es una proposición, No A también lo será; y si A y B son dos proposiciones, podemos combinarlas de muchas maneras para formar nuevas proposiciones, por ejemplo:

A y B
A y no B
Si A entonces B
Ni A, ni B
A o B

Definiremos a continuación un *lenguaje formal* que nos servirá para el análisis de ciertos tipos de argumentaciones correctas. A este lenguaje lo llamaremos $\mathscr{L}_0$, y consta de los siguientes símbolos:

1) Letras mayúsculas del alfabeto, con o sin subíndices:

$$A, B, C, \ldots, A_1, B_1, C_1, \ldots, A_2, B_2, C_2, \ldots, A_n, B_n, C_n, \ldots$$

A estos símbolos les llamamos *letras proposicionales.*

2) $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
A estos símbolos les llamamos *conectivos lógicos.*

3) Paréntesis:), (. Siendo éstos *símbolos de puntuación.*

Todos estos símbolos pueden combinarse para formar expresiones del lenguaje $\mathscr{L}_0$. Una *expresión* de $\mathscr{L}_0$ es una sucesión finita de símbolos de $\mathscr{L}_0$. Como ejemplos de expresiones, tenemos:

$$A_1,\ A_1A_2A_3,\ \neg A_1,\ P \Rightarrow Q),(P \Leftrightarrow Q)$$

Las reglas de formación para este lenguaje determinarán cuáles expresiones son fórmulas bien formadas de $\mathscr{L}_0$. Antes de dar estas reglas sería conveniente recordar que este lenguaje se está definiendo para dar un modelo de cierto tipo de argumentaciones, de forma tal que las fórmulas bien formadas "representen" proposiciones. Las letras proposicionales representan proposiciones arbitrarias y los conectivos serán utilizados para obtener proposiciones más complejas. El significado de los conectivos es el siguiente:

$\neg$	no
$\wedge$	y
$\vee$	o
$\Rightarrow$	implica
$\Leftrightarrow$	es equivalente a

Esto es, $\neg A$, representará a la negación de la proposición representada por A.

Como en el estudio de la lógica no nos interesa lo que las proposiciones dicen en sí, sino cómo se relacionan unas con otras, no asignaremos un significado específico

a las letras proposicionales, sólo pensaremos en ellas como proposiciones que pueden ser verdaderas o falsas.

Las reglas de formación para las *fórmulas bien formadas*, con esta interpretación en mente, son naturales:

1) Toda letra proposicional es una fórmula bien formada.

2) Si ϕ y ψ son fórmulas bien formadas arbitrarias, también lo son las siguientes expresiones:

$$(\neg\phi),\ (\phi \wedge \psi),\ (\phi \vee \psi), (\phi \Rightarrow \psi) \text{ y } (\phi \Leftrightarrow \psi).$$

3) Las únicas fórmulas bien formadas son aquéllas que se obtienen por medio de (1) o (2).

De aquí en adelante, debido a que las únicas fórmulas que hemos de tratar son las fórmulas bien formadas, nos referiremos a ellas simplemente como fórmulas o bien con su abreviatura fbf.

Las fórmulas con esta interpretación, representan a proposiciones simples o complejas. Las proposiciones más simples serán representadas por las letras proposicionales, mientras que las complejas se obtendrán aplicando la regla (2) para combinar letras proposicionales con conectivos. Las fórmulas *atómicas* son las letras proposicionales, las otras fórmulas se llaman *compuestas* o *moleculares*.

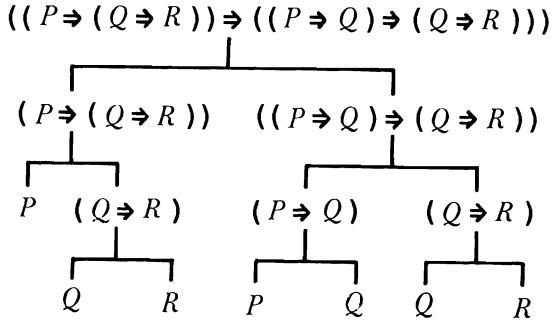
Resulta relativamente sencillo discernir dentro del conjunto de las expresiones de $\mathscr{L}_0$ las que son fórmulas de las que no. Para ello, dada α, verificamos primero si es una letra proposicional, si sí lo es, α es una fbf, y terminamos; caso contrario, identificamos al *conectivo principal* de la expresión (aquél que al eliminar los paréntesis externos concatena bien sea (i) otras dos expresiones, digamos α_{11} y α_{12}, o (ii) una sola, α_1 (si no hubiera paréntesis externos, α no sería fbf). En el caso (i), lo comparamos con: $\wedge$, $\vee$, $\Rightarrow$ ó $\Leftrightarrow$, mientras que en (ii) con $\neg$. Si no es alguno de estos casos, la expresión no era fbf, y terminamos. Si la respuesta es favorable, analizamos a su vez las expresiones α_{11} y α_{12} (por separado) o bien a α_1, (según el caso): procediendo de manera similar que para con α. Si el proceso es siempre favorable, debemos obtener eventualmente las letras proposicionales que ocurren en α, implicando que α es una fbf. Si esto no es así, α no es una fbf. Este proceso es representable mediante *árboles,* tal y como haremos a continuación.[1]

[1]El procedimiento aquí presentado es implementable como un *algoritmo recursivo.* Para justificar que está bien definido, *cf*. [En].

Ejemplos:

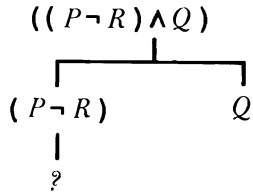
Analizar si las expresiones siguientes son fórmulas o no:

1. $\alpha = ((P \Rightarrow (Q \Rightarrow R)) \Rightarrow ((P \Rightarrow Q) \Rightarrow (Q \Rightarrow R)))$, entonces



luego sí es una fórmula.

2. $\beta = ((P \neg R) \wedge Q)$, entonces



Figuras 4.1

no es una fórmula, pues $\neg$ es un conectivo unario. □

Observación. Los paréntesis son símbolos a los que no les asignamos un significado. Sirven para evitar ambigüedades, pues una fórmula sin paréntesis como "$\neg P \Rightarrow Q$" se puede interpretar como $(\neg(P \Rightarrow Q))$ o como $((\neg P) \Rightarrow Q)$.

Ejercicios

Determinar si las expresiones siguientes son fórmulas o no:

i. $((A \Leftrightarrow B) \Leftrightarrow Q)$, ii. $((Q \vee P) \Leftrightarrow \neg P)$, iii. $(P \Leftrightarrow ((Q \neg R) \Rightarrow (S \wedge T)))$, iv. $(P \wedge (\neg Q)) \Rightarrow (\neg(\neg P) \Leftrightarrow Q)$, v. $(P \Leftrightarrow (R \vee S) \wedge \neg\neg(P \Rightarrow Q))$

4.3 Semántica de proposiciones

Para analizar si un argumento dado es correcto o no, lo que se verifica es si la verdad de la conclusión se sigue de la verdad de las premisas, por tanto debemos tener una manera precisa de saber cuándo una fórmula bien formada es verdadera. Si la fórmula bien formada es atómica, puede ser verdadera o falsa, ya que toda proposición en un lenguaje natural es verdadera o falsa. El valor de verdad de una fbf molecular se puede calcular a partir de las letras proposicionales que aparecen en ella por medio de las siguientes tablas:

P	$(\neg P)$
V	F
F	V

P	Q	$(P \wedge Q)$	$(P \vee Q)$	$(P \Rightarrow Q)$	$(P \Leftrightarrow Q)$
V	V	V	V	V	V
V	F	F	V	F	F
F	V	F	V	V	F
F	F	F	F	V	V

Tablas 4.1

En realidad estas *tablas de verdad* definen lo que vamos a entender por las palabras "no", "y", "o", "implica" y "es equivalente a".

La negación significa, para nosotros, un cambio de valor de verdad. Si una proposición es verdadera, su negación es falsa y viceversa.

Cuando se afirma una conjunción, se afirman ambas componentes de ella. Cabe mencionar que esta definición de conjunción no representa adecuadamente todos los casos que se presentan en el lenguaje natural, como en: "Mató y tuvo miedo", proposición que no resulta equivalente a "Tuvo miedo y mató", aquí la palabra "y"

tiene un sentido temporal y causal. Esta propiedad conmutativa sí resulta válida para la conjunción que hemos definido.

La tabla de verdad para la disyunción sólo es verdadera cuando ambas componentes son verdaderas. Éste no siempre es el caso en español, por ejemplo, cuando afirmamos que todo ser humano es hombre o mujer estamos excluyendo la posibilidad de que ambas opciones ocurran al mismo tiempo, a este uso de la palabra "o" se le denomina "exclusivo"; en lógica estamos trabajando con una "o" inclusiva, que en algunos documentos legales se escribe y/o. Esta elección de la "o" no representa una pérdida, como veremos más adelante. (*Cf.* sección 4.6). El símbolo $\vee$ empleado para la disyunción proviene de la palabra *vel* del latín que significa precisamente "o" inclusiva.

Quizás la tabla de verdad que más problemas presenta al principio es la tabla de la implicación o condicional. Si observamos los dos últimos renglones de dicha tabla para la implicación notamos que si el antecedente en una implicación es falso, la implicación es verdadera, sin importar el valor de verdad del consecuente. Así, las siguientes dos proposiciones son verdaderas:

$$\text{Si } 2+2=3 \text{ entonces } 2+2=4$$
$$\text{Si } 2+2=3 \text{ entonces } 4+1=0$$

Esto puede parecer contradictorio a primera vista, pero si analizamos lo que queremos decir con "si P entonces Q", vemos que estamos grarantizando que se da Q siempre y cuando se tenga P. Si no se da P, no nos hemos comprometido en nada respecto de la verdad o falsedad de Q. El hecho que estamos trabajando en una lógica bivalente (con sólo dos valores de verdad: V y F) nos obliga a decidir, dada una proposición, si es verdadera o falsa. Si en estos dos últimos renglones no le quisiéramos dar el valor V al condicional, tendríamos que darle el valor F, y esto sí sería erroneo. Imaginemos que un candidato a la presidencia afirma: "Si llego a ser electo presidente, reduciré todos los impuestos a la mitad". Si no resulta electo, ¿estaría justificado afirmar que dijo una falsedad?

El bicondicional "$P \Leftrightarrow Q$" es una manera de abreviar $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$, de modo que su tabla de verdad está determinada por las de implicación y conjunción.

Veamos ahora algunos ejemplos de cómo construir tablas de verdad para fórmulas con varios conectivos.

i. $\alpha = ((\neg P) \vee Q)$

P	Q	$(\neg P)$	$((\neg P) \vee Q)$
V	V	F	V
V	F	F	F
F	V	V	V
F	F	V	V

ii. $\alpha = (((\neg P) \vee Q) \wedge R)$

P	Q	R	$(\neg P)$	$((\neg P) \vee Q)$	$(((\neg P) \vee Q) \wedge R)$
V	V	V	F	V	V
V	V	F	F	V	F
V	F	V	F	F	F
V	F	F	F	F	F
F	V	V	V	V	V
F	V	F	V	V	F
F	F	V	V	V	V
F	F	F	V	V	F

Tablas 4.2

La construcción de las anteriores tablas de verdad, dependió de tres factores, dados a manera de convención.

1. De la forma de la fórmula pues, por ejemplo, la tabla de verdad de $(P \vee Q)$ no es igual que la de $(P \wedge Q)$. Sin embargo, sí se van a dar casos de fórmulas distintas que tengan la misma tabla de verdad.
2. Del número de letras proposicionales distintas que figuran en la fórmula. Así, si n es este número, la tabla de verdad constará de 2^n renglones.
3. Del orden en que se asigna a cada letra su valor de verdad. En la elaboración de las tablas anteriores hemos adoptado un orden lexicográfico.

Observación. Debido a que resulta equivalente representar con 1 al valor V y con 0 al valor F, introduciremos esta innovación a partir de aquí. La importancia de este reemplazo se hará patente en el curso de este capítulo.

Al construir tablas de verdad para fórmulas más complejas se hace evidente que los paréntesis de nuestro lenguaje son importantísimos. Las tablas para $((\neg P) \wedge Q)$

y $(\neg(P \wedge Q))$

P	Q	$((\neg P) \wedge Q)$	$(\neg(P \wedge Q))$
1	1	0	0
1	0	0	1
0	1	1	1
0	0	0	1

Tabla 4.3

son distintas, y por tanto, desde el punto de vista de la lógica, estas dos fórmulas tienen que ser diferentes y los paréntesis no se pueden quitar sin generar ambigüedades. Sin embargo, puede resultar incómodo escribir tantos paréntesis y hay convenciones para simplificar la notación. Las que adoptaremos aquí serán únicamente las siguientes.

1. Se pueden omitir los paréntesis externos de una fbf.
2. La negación es el conectivo más débil, de modo que si se aplica a una sola letra proposicional pueden omitirse los paréntesis correspondientes. Esto es, por ejemplo, en vez de $((\neg P) \vee Q)$ se puede escribir simplemente $\neg P \vee Q$.
3. Cuando en una fórmula sólo aparece un mismo conectivo binario y éste es $\wedge$ u $\vee$, se pueden omitir los paréntesis. Ejemplo: en vez de $((A \wedge B) \wedge C)$ se puede escribir $A \wedge B \wedge C$ y en vez de $(((\neg P) \vee Q) \vee R)$ se puede escribir $\neg P \vee Q \vee R$.

Otra cosa que es evidente después de haber construido varias tablas de verdad es que toda fbf de nuestro lenguaje tiene una única tabla de verdad. Este hecho es en realidad un teorema de lógica formal, pero su demostración rigurosa requiere de algunos teoremas fuertes de la teoría de conjuntos, y por tanto no lo demostraremos aquí.

Las letras proposicionales de nuestro lenguaje representan proposiciones concretas en algún lenguaje, pero ya hemos explicado que al lógico no le interesa lo que una proposición dice en sí, sino la estructura formal de los argumentos, y que para saber si un argumento es correcto o no, lo importante es determinar si de la verdad de las premisas se sigue la verdad de la conclusión. Por tanto, para interpretar las letras proposicionales, basta darles un valor de verdad, ya que al ser proposiciones, éstas serán verdaderas o falsas. De ahí la siguiente definición:

> **Definición.** Una *valuación* para el lenguaje formal $\mathscr{L}_0$ es una función $v: \mathscr{P} \to \{0, 1\}$, donde $\mathscr{P}$ es el conjunto de letras proposicionales de $\mathscr{L}_0$.

Esto es, una valuación asigna a cada letra proposicional un valor de verdad, 0 si es falsa, 1 si es verdadera.

Si tenemos una fórmula compleja y una valuación ν, siempre podremos calcular el valor de verdad de la fórmula dada, bajo esa valuación. Una valuación corresponde a algún renglón de la tabla de verdad para la fbf en cuestión. Este valor de verdad asignado a las fórmulas es único una vez fijada la valuación, pues sólo hay una manera de calcular los valores correspondientes en la tabla de verdad. Por ejemplo, supongamos que tenemos una valuación ν definida como sigue, si X es una letra proposicional,

$$\nu(X) = \begin{cases} 0 & \text{si } X \text{ no está indexada} \\ 1 & \text{si } X \text{ está indexada} \end{cases}$$

Con esta valuación fija, podemos calcular el valor de verdad de cualquier fbf bajo esta valuación, al que denotamos por $\bar{\nu}$:

$$\bar{\nu}(\neg A) = 1, \quad \text{ya que } \nu(A) = 0$$
$$\bar{\nu}(\neg A_1) = 0, \quad \text{ya que } \nu(A_1) = 1$$
$$\bar{\nu}(A \Leftrightarrow B) = 1, \quad \text{ya que } \nu(A) = \nu(B), \text{ etcétera.}$$

Para extender una función de valuación ν y sea aplicable a fórmulas moleculares, primero observamos que los conectivos lógicos pueden ser introducidos como operadores (funciones), pues requieren de fórmulas de entrada (inputs) para proporcionar una fórmula resultante (output). Así, si F es un operador lógico binario, por ejemplo $\wedge$, tenemos que F envía una pareja de fórmulas (α, β) en una nueva fórmula $\gamma = F(\alpha, \beta)$. (Nótese que aquí, si $F = \wedge$, por ejemplo, la fórmula $\gamma = \wedge(\alpha, \beta)$ está expresada en notación prefija y no en la infija, $\gamma = \alpha \wedge \beta$, que es la usual). De esta manera, la forma de extender una valuación ν radica en que el valor de verdad de la fórmula resultante puede ser determinado conociendo los valores de verdad de las proposiciones de entrada (α y β, en este caso) y de qué operador F está siendo empleado. Y esto es precisamente el propósito de una *tabla de verdad* o *función de verdad,* que denotaremos con f.

Notación. Sea $\Phi(\mathscr{P})$ el conjunto de fórmulas producidas a partir del conjunto de letras proposicionales $\mathscr{P}$ de $\mathscr{L}_0$.

De la discusión en curso, tenemos que si $F = \wedge: \Phi(\mathscr{P}) \times \Phi(\mathscr{P}) \to \Phi(\mathscr{P})$, es la conjunción, sus valores de verdad correspondientes pueden ser hallados utilizando

la operación numérica $f(x, y) = \min\{x, y\}$. Obsérvese que con la función min se sintetiza la tabla de verdad de la conjunción. Así, para $\alpha, \beta \in \Phi(\mathscr{P})$, si $\gamma = \wedge(\alpha, \beta)$, tenemos

$$\bar{\nu}(\gamma) = \bar{\nu}[\wedge(\alpha, \beta)] = \min\{\bar{\nu}(\alpha), \bar{\nu}(\beta)\}$$

De aquí, para obtener el valor de verdad de γ mediante la valuación extendida $\bar{\nu}$ se requiere conocer los valores de verdad de $\bar{\nu}(\alpha)$ y $\bar{\nu}(\beta)$; pero a su vez α y β pueden ser fórmulas moleculares, y por tanto los valores de $\bar{\nu}(\alpha)$ y $\bar{\nu}(\beta)$ se obtendrán en términos de sus fórmulas componentes, implicando así un proceso recursivo de valuaciones hasta llegar, en un número finito de pasos, a tener que evaluar las letras proposicionales que aparezcan en γ. Por ejemplo, consideremos la fórmula siguiente:

$$\gamma = (P \wedge Q) \wedge P$$

Entonces, $\bar{\nu}(\gamma) = \bar{\nu}((P \wedge Q) \wedge P) = \min\{\bar{\nu}(P \wedge Q), \bar{\nu}(P)\} = \min\{\min\{\nu(P), \nu(Q)\}, \nu(P)\}$, de donde, al asignar valores de verdad a $\nu(P)$ y $\nu(Q)$ obtendremos el valor correspondiente de $\bar{\nu}(\gamma)$. De esta manera, hemos obtenido una representación funcional para la tabla de verdad asociada a la fórmula γ:

$\nu(P)$	$\nu(Q)$	$\bar{\nu}(P \wedge Q)$	$\bar{\nu}((P \wedge Q) \wedge P)$
1	1	1	1
1	0	0	0
0	1	0	0
0	0	0	0

Tabla 4.4

Así, dados $\alpha, \beta \in \Phi(\mathscr{P})$, un operador lógico F y una valuación ν, la valuación de la fórmula $\gamma = F(\alpha, \beta)$ se obtendrá mediante la expresión $\bar{\nu}(F(\alpha, \beta)) = f(\bar{\nu}(\alpha), \bar{\nu}(\beta))$. Gráficamente, esto se interpreta como la conmutatividad del diagrama dado por la figura 4.2.

Si ahora la fórmula γ cuenta con algunos de los operadores lógicos: $\neg$, $\vee$, $\Rightarrow$ y $\Leftrightarrow$, para poder evaluar $\bar{\nu}(\gamma)$ se necesitan de otras funciones numéricas asociadas, que sinteticen apropiadamente las tablas de verdad correspondientes a estos operadores. El teorema siguiente nos garantiza que este enfoque funcional para obtener los valores de verdad de fórmulas moleculares a partir de los valores asignados a las letras proposicionales que en ellas aparezcan, siempre puede

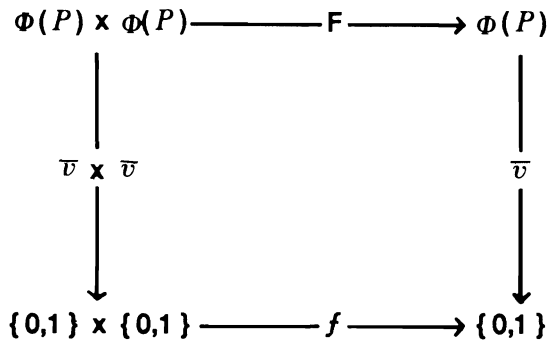


Figura 4.2

realizarse de manera recursiva y con un resultado unívocamente determinado para una valuación dada.

Teorema 4.1. *Sean* $\alpha, \beta \in \Phi(\mathscr{P})$ *y* ν *una valuación definida sobre* $\mathscr{P}$. *Entonces existe una única función* $\bar{\nu}\colon \Phi(\mathscr{P}) \to \{0, 1\}$ $\bar{\nu}$
(es una extensión de ν *al dominio* $\Phi(\mathscr{P})$*), tal que*

1. *Para toda* $P \in \mathscr{P}$, $\bar{\nu}(P) = \nu(P)$
2. $\bar{\nu}(\neg\alpha) = 1 - \bar{\nu}(\alpha)$
3. $\bar{\nu}(\alpha \vee \beta) = \max\{\bar{\nu}(\alpha), \bar{\nu}(\beta)\}$
4. $\bar{\nu}(\alpha \wedge \beta) = \min\{\bar{\nu}(\alpha), \bar{\nu}(\beta)\}$
5. $\bar{\nu}(\alpha \Rightarrow \beta) = 1 - \bar{\nu}(\alpha) + \bar{\nu}(\alpha)\bar{\nu}(\beta)$
6. $\bar{\nu}(\alpha \Leftrightarrow \beta) = \bar{\nu}(\alpha)\bar{\nu}(\beta) + (1 - \bar{\nu}(\alpha))(1 - \bar{\nu}(\beta))$.

El aspecto destacable de este teorema radica en que justifica una técnica alterna para hallar los valores de verdad de las fórmulas, transformando un problema del "mundo lógico" a un "mundo aritmético" que consiste del conjunto $\{0, 1\}$ y las operaciones numéricas correspondientes. Para la prueba de este teorema *cf*. [En]-[Ma].

Notación. Debido a la similitud que guarda la valuación ν con la función *valor absoluto*, la denotaremos con $|\cdot|$, siempre y cuando esto no cause confusiones.

Aun cuando existe una infinidad de valuaciones para $\mathscr{L}_o$, dado que hay una infinidad de letras proposicionales; para el caso de una fbf en particular, un número infinito de valuaciones coinciden en las letras que aparecen en la fbf, que son las relevantes para calcular su valor de verdad. Por esto la tabla de verdad de una fórmula cubre todas las posibilidades, dándonos así todos los valores de verdad de esa fórmula bajo todas las valuaciones. Además, haciendo una analogía con las tablas numéricas de las funciones reales no algebraicas, como son las tablas de logaritmos, trigonométricas, etc., tenemos que si contáramos con la tabla que nos diera los valores asociados a cualquier número en el dominio de la función, tendríamos perfectamente caracterizada la función. Sin embargo, dado que la cardinalidad de cualquier intervalo de los reales no es numerable, tal tabla no existe físicamente (el número de renglones sería no sólo infinito, sino no numerable), conformándonos con una distribución discreta de valores (suficiente para fines prácticos). A diferencia, como los valores asignados por valuaciones a las fórmulas son sólo 0 ó 1, sí tenemos, por ende, caracterizada la función de verdad de una fórmula mediante su tabla de verdad.

Ejemplos:

Determinemos las funciones de verdad de algunas fórmulas:

1. $$\begin{aligned} |P \Rightarrow (Q \Rightarrow P)| &= 1 - |P| + |P||Q \Rightarrow P| \\ &= 1 - |P| + |P|(1 - |Q| + |Q||P|) \\ &= 1 - |P| + |P| - |P||Q| + |P||P||Q| \\ &= 1 - |P||Q| + |P||Q| = 1 \end{aligned}$$

2. $$\begin{aligned} |A \vee (\neg R \Rightarrow Q)| &= \max\{|A|, |\neg R \Rightarrow Q|\} \\ &= \max\{|A|, 1 - |\neg R| + |\neg R||Q|\} \\ &= \max\{|A|, 1 - (1 - |R|) + (1 - |R|)|Q|\} \\ &= \max\{|A|, |R| + (1 - |R|)|Q|\} \end{aligned}$$

3. La ley de De Morgan $\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$. Para ésta, verificaremos que las fórmulas $\alpha = \neg(P \wedge Q)$ y $\beta = (\neg P \vee \neg Q)$ tienen el mismo valor de verdad bajo cualquier valuación. Para el efecto, usaremos las expresiones siguientes para determinar el máximo y el mínimo de dos números reales:

$$\max\{x, y\} = \frac{1}{2}(x + y + \|x - y\|) \quad \text{y} \quad \min\{x, y\} = \frac{1}{2}(x + y - \|x - y\|)$$

Para evitar confusiones, hemos denotado con $\|\cdot\|$ al valor absoluto. Así, por una parte

$$\begin{aligned}|\alpha| = |\neg(P \wedge Q)| &= 1 - |P \wedge Q| \\ &= 1 - \min\{|P|, |Q|\} \\ &= 1 - \frac{1}{2}(|P| + |Q| - \||P| - |Q|\|)\end{aligned}$$

Mientras que para β, tenemos

$$\begin{aligned}|\beta| = |\neg P \vee \neg Q| &= \max\{|\neg P|, |\neg Q|\} = \max\{1 - |P|, 1 - |Q|\} \\ &= \frac{1}{2}((1 - |P|) + (1 - |Q|) + \|(1 - |P|) - (1 - |Q|)\|) \\ &= \frac{1}{2}(2 - (|P| + |Q|) + \||P| - |Q|\|) \\ &= 1 - \frac{1}{2}(|P| + |Q| - \||P| - |Q|\|)\end{aligned}$$

□

Ejercicios

1. Supóngase que se quiere tener un nuevo conectivo $\vee$ que represente el uso exclusivo de la palabra "o" en español. Constrúyase una tabla de verdad que rescate ese significado.

2. Calcular las tablas de verdad para las fórmulas moleculares siguientes:

i. $((\neg P) \wedge P)$,
ii. $((P \wedge Q) \Rightarrow P)$,
iii. $(P \Rightarrow (Q \vee (\neg Q)))$,
iv. $(((Q \vee R) \wedge (\neg R)) \Rightarrow Q)$,
v. $(((A \Leftrightarrow C) \vee (\neg(A \Leftrightarrow Q))) \wedge (\neg Q))$,
vi. $((((\neg E) \vee R) \Leftrightarrow (K \vee E)) \Leftrightarrow E)$,
vii. $((R \Leftrightarrow S) \Leftrightarrow T)$,
viii. $(((\neg(C \Leftrightarrow (A \Leftrightarrow B))) \wedge (B \vee B)) \vee W)$

3. Usando las convenciones establecidas, restablezca los paréntesis en las expresiones siguientes para que sean fórmulas:

i. $(\neg Q \vee \neg R) \Rightarrow ((P \wedge Q) \vee S)$,

ii. $(\neg((P \wedge Q) \vee (R \wedge Q)) \Rightarrow \neg S) \Rightarrow \neg P$,

iii. $((P \Leftrightarrow Q) \Leftrightarrow R) \Leftrightarrow P$,

iv. $(\neg\neg\neg P \vee (Q \Rightarrow P)) \Leftrightarrow ((\neg\neg Q \Leftrightarrow T) \vee S)$,

v. $((Q \Rightarrow S) \Rightarrow (P \Rightarrow Q)) \wedge (\neg(R \wedge (T \vee Q)) \Leftrightarrow \neg P) \wedge \neg Q$

***4.** Pruebe que la tabla de verdad de una fórmula de n letras proposicionales tiene 2^n renglones. (Sugerencia: inducción sobre el número de letras de la fórmula).

5. Verifique que las expresiones (2)-(6) del Teorema 4.1 nos proporcionan las tablas de verdad de los operadores lógicos correspondientes.

6. Pruebe que $|P \vee Q| = |\neg P \Rightarrow Q|$ y $|P \wedge Q| = |\neg(P \Rightarrow \neg Q)|$. De aquí, obtenga expresiones más sencillas de operar que las funciones min y max dadas inicialmente.[2]

7. Determine las funciones de verdad para cada una de las fórmulas siguientes (puede usar los resultados del ejercicio 6):

i. $(\neg P \Rightarrow Q) \vee A$,

ii. $A \wedge \neg A$,

iii. $(A \vee B) \Leftrightarrow (B \vee A)$,

iv. $(P \Rightarrow (Q \vee R)) \Leftrightarrow ((P \Rightarrow Q) \vee (P \Rightarrow R))$,

v. $(A \wedge Q) \Rightarrow Q$,

vi. $((\neg\neg S \vee T) \Rightarrow Q) \wedge P$,

vii. $(((P \Rightarrow Q) \wedge (R \Rightarrow S)) \wedge (P \vee S)) \Rightarrow (Q \vee S)$.

8. Sea $\alpha = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n \in \Phi(\mathscr{P})$. Pruebe que $|\alpha| = \min_{1 \leq i \leq n}\{\alpha_i\}$ (use inducción matemática). Con base en el ejercicio 6, demuestre que $|\alpha| = \prod_{1 \leq i \leq n} |\alpha_i|$.

9. Pruebe a partir de los incisos (4) y (5) del Teorema 4.1, el (6) del mismo.

---

[2]La introducción de las funciones min y max asociadas a los operadores lógicos $\wedge$ y $\vee$, respectivamente, resultan necesarios cuando se consideran *lógicas polivalentes*.

4.4 Consecuencia tautológica, tautologías

Regresemos ahora a nuestro punto de partida: la lógica matemática es un modelo matemático del pensamiento deductivo. Ya tenemos un lenguaje formal para representar las proposiciones simples y complejas, sobre el cual estamos construyendo una teoría formal, como las descritas en el capítulo anterior. La relación que nos interesa rescatar es aquélla que se da entre un conjunto de proposiciones (premisas de un argumento) y otra proposición (conclusión del argumento) cuando esta última es una inferencia lógicamente válida de las anteriores, esto es, cuando el argumento es correcto. Con la definición siguiente se rescata la noción de un argumento correcto para nuestro lenguaje.

> **Definición.** Sean $\Gamma \subseteq \Phi(\mathscr{P})$, un conjunto de fórmulas, y ϕ una fbf. Decimos que ϕ es *consecuencia tautológica* de Γ si y sólo si para toda valuación $|\cdot|$ que hace verdaderas a todas las fórmulas de Γ se tiene que $|\phi| = 1$.

Nótese que esta definición sí rescata el concepto de argumentación correcta. En efecto, si ϕ es consecuencia tautológica de Γ, cada vez que los elementos de Γ (las premisas del argumento) sean verdaderos, también ϕ (la conclusión del argumento) tiene que serlo.

Notación. $\Gamma \vDash_T \phi$.

Si $\Gamma = \{\psi_1, \psi_2, \ldots, \psi\}$, se escribe $\psi_1, \psi, \ldots, \psi_n \vDash_T \phi$, en lugar de la expresión: $\{\psi_1, \psi, \ldots, \psi_n\} \vDash_T \phi$.

Ejemplos:

1. $P \Rightarrow \neg Q$, $Q \vDash_T \neg P$. En efecto, sea $|\cdot|$ cualquier valuación para la cual tengamos ambas premisas verdaderas, esto es $|P \Rightarrow \neg Q| = 1 = |Q|$; entonces $|\neg Q| = 0$ y por lo tanto $|P| = 0$, o sea, $|\neg P| = 1$.
2. $P \vee Q$, $\neg P \vDash_T Q$. Sea $|\cdot|$ una valuación arbitraria tal que $|P \vee Q| = 1$ y $|\neg P| = 1$, entonces, $|P| = 0$ y $|Q| = 1$.

> **Definición.** Dos fórmulas ϕ y ψ son *tautológicamente equivalentes* si y sólo si $\phi \vDash_T \psi$ y $\psi \vDash_T \phi$. Notación: $\phi \models\!\!\models_T \psi$.

Definición. Una fbf ψ se denomina una *tautología* o fórmula *válida* si y sólo si $\varnothing \vDash_T \psi$.

Notación. $\vDash \psi$.

Teorema 4.2. *Sea ϕ una fbf. Entonces $\vDash \phi$ si y sólo si para toda valuación $|\cdot|$ se tiene que $|\phi| = 1$.* ■

Definición. Una fórmula ϕ es *contradictoria* si y sólo si $\vDash \neg\phi$.

Teorema 4.3. *ϕ es una fórmula contradictoria si y sólo si para toda valuación $|\cdot|$ se tiene que $|\phi| = 0$.* ■

Teorema 4.4. *Sean $\alpha, \beta \in \Phi(\mathscr{P})$. Entonces, $\alpha \vDash_T \beta$ si y sólo si $\vDash (\alpha \Rightarrow \beta)$.*

Demostración.

$\Rightarrow$) Supongamos que $\alpha \vDash_T \beta$ y sea $|\cdot|$ una valuación arbitraria. Así, si $|\alpha| = 0$ entonces, $|\alpha \Rightarrow \beta| = 1$. Y si $|\alpha| = 1$, por hipótesis, $|\beta| = 1$, luego $|\alpha \Rightarrow \beta| = 1$. En ambos casos, $\vDash (\alpha \Rightarrow \beta)$.

$\Leftarrow$) Supongamos ahora que $\vDash (\alpha \Rightarrow \beta)$ y sea $|\cdot|$ una valuación arbitraria. Por lo tanto, $|\alpha \Rightarrow \beta| = 1$, *i.e.*, no es el caso que $|\alpha| = 1$ y $|\beta| = 0$, de donde $\alpha \vDash_T \beta$. ■

La importancia del teorema siguiente, generalización de anterior, radica en que permite traducir todo problema de argumentación (en el metalenguaje) en simplemente verificar si la fórmula que se obtiene es o no una tautología (*i.e.*, un problema en el lenguaje). En otros términos, transforma reglas del metalenguaje en fórmulas del lenguaje; razón a la que debe su nombre, como se hará patente en el próximo capítulo sobre la axiomática.

Teorema 4.5. (de la Deducción). *Sea $\Gamma \cup \{\alpha\} \subseteq \Phi(\mathscr{P})$, donde $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Entonces $\Gamma \vDash_T \alpha$ si y sólo si $\vDash (\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \Rightarrow \alpha$.* ■

Teorema 4.6. (*Modus Ponens*). *Sean $\alpha, \beta \in \Phi(\mathscr{P})$. Si $\vDash \alpha$ y $\vDash (\alpha \Rightarrow \beta)$, entonces $\vDash \beta$.*

Tabla de algunas *Leyes Lógicas* o tautologías especiales

identidad:	$P \Rightarrow P$
	$P \Leftrightarrow P$
el *tercero excluso:*	$P \vee \neg P$
no contradicción:	$\neg(P \wedge \neg P)$
doble negación:	$\neg\neg P \Rightarrow P$
asociatividad:	$(P \vee (Q \vee R)) \Leftrightarrow ((P \vee Q) \vee R)$
	$(P \wedge (Q \wedge R)) \Leftrightarrow ((P \wedge Q) \wedge R)$
conmutatividad:	$(P \vee Q) \Leftrightarrow (Q \vee P)$
	$(P \wedge Q) \Leftrightarrow (Q \wedge P)$
distributividad:	$(P \wedge (Q \vee R)) \Leftrightarrow ((P \wedge Q) \vee (P \wedge R))$
	$(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$
	$(P \Rightarrow (Q \vee R)) \Leftrightarrow ((P \Rightarrow Q) \vee (P \Rightarrow R))$
	$(P \Rightarrow (Q \wedge R)) \Leftrightarrow ((P \Rightarrow Q) \wedge (P \Rightarrow R))$
De Morgan:	$\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$
	$\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$
simplificación:	$P \Rightarrow (P \vee Q)$
	$(P \wedge Q) \Rightarrow P$
eliminación:	$((P \wedge Q) \vee Q) \Leftrightarrow Q$
	$((P \vee Q) \wedge Q) \Leftrightarrow Q$
transitividad:	$((P \Rightarrow Q) \wedge (Q \Rightarrow R)) \Rightarrow (P \Rightarrow R)$
	$((P \Leftrightarrow Q) \wedge (Q \Leftrightarrow R)) \Rightarrow (P \Leftrightarrow R))$
la implicación:	$(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$
	$(P \Rightarrow Q) \Leftrightarrow \neg(P \wedge \neg Q)$
el *dilema:*	$(((P \Rightarrow Q) \wedge (R \Rightarrow S)) \wedge (P \vee R)) \Rightarrow (Q \vee S)$
contrapositiva:	$(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$
reducción al absurdo:	$(Q \wedge \neg Q) \Rightarrow P$
afirmación del antecedente:	$P \Rightarrow (Q \Rightarrow P)$
exportación:	$((P \wedge Q) \Rightarrow R) \Leftrightarrow (P \Rightarrow (Q \Rightarrow R))$
modus ponens:	$((P \Rightarrow Q) \wedge P) \Rightarrow Q$
modus tollens:	$((P \Rightarrow Q) \wedge \neg Q) \Rightarrow \neg P$

Tabla 4.5

Demostración.

Sea $|\cdot|$ una valuación arbitraria. Por hipótesis tenemos $|\alpha| = |\alpha \Rightarrow \beta| = 1$, de donde, $|\beta| = 1$, luego, por definición, $\vDash \beta$. ■

Los siguientes dos teoremas nos permiten obtener nuevas tautologías a partir de las ya conocidas por medio de *i*) el principio de *sustitución uniforme* de expresiones dentro de fórmulas (Teorema 4.7) y *ii*) la denominada *regla de intercambio* (Teorema 4.8 b), de tal manera que podremos saber si una determinada fórmula es una tautología tan sólo apelando a su estructura (*cf*. [Me]-[Th]).

Teorema 4.7. *Sean α una tautología cuyas letras proposicionales son $P_1, P_2, \ldots, P_n$, y β una fórmula que se obtiene a partir de α sustituyendo $P_1, P_2, \ldots, P_n$ por las fórmulas $\alpha_1, \alpha_2, \ldots, \alpha_n$, respectivamente. Entonces β es una tautología. En otras palabras, la sustitución uniforme en una tautología proporciona otra tautología.*

Demostración.

Sea ν una valuación arbitraria. P.D. $\bar{\nu}(\beta) = 1$. Sea μ una asignación definida en $\{P_1, P_2, \ldots, P_n\}$ tal que $\mu(P_i) = \bar{\nu}(\alpha_i)$. Entonces, $\bar{\mu}(\alpha) = \bar{\nu}(\beta)$. Ahora, como $\vDash \alpha$, entonces $|\alpha| = 1$. Por tanto, $|\beta| = 1$, *i.e.*, $\vDash \beta$.

Gráficamente, se tiene el diagrama 4.3.

$$\begin{array}{ccc} \alpha = \alpha(P_1, \ldots, P_n) & \longrightarrow & \beta = \alpha(P_1/\alpha_1, \ldots, P_n/\alpha_n) \\ \downarrow & & \downarrow \\ \bar{\mu}(\alpha(P_1, \ldots, P_n)) \equiv 1 & \longrightarrow & \bar{\nu}(\beta(\alpha_1, \ldots, \alpha_n)) \end{array}$$

Figura 4.3

donde, P_i/α_i significa la sustitución de P_i por α_i, $1 \leq i \leq n$. La demostración de este teorema se traduce como la conmutatividad del diagrama 4.3. ■

Definición. Decimos que α es una *subfórmula* de una fórmula ϕ si y sólo si α es una parte de ϕ que es a su vez una fórmula.

Teorema 4.8. *Sean ϕ, ψ y β tres fórmulas y α una subfórmula de ϕ. Entonces:*

a. *Si ψ se obtiene de ϕ mediante la sustitución por β para una o varias ocurrencias de α en ϕ, entonces* $\vDash ((\alpha \Leftrightarrow \beta) \Rightarrow (\phi \Leftrightarrow \psi))$.

b. *Si, además,* $\alpha \underset{T}{\models\!\models} \beta$, *entonces* $\phi \underset{T}{\models\!\models} \psi$.

Demostración.

a. Consideremos una valuación arbitraria $|\cdot|$. Si $|\alpha| \neq |\beta|$, entonces $|\phi \Rightarrow \psi| = 0$, luego $|(\alpha \Leftrightarrow \beta) \Rightarrow (\phi \Leftrightarrow \psi)| = 1$. Por el contrario, si $|\alpha| = |\beta|$, entonces $|\phi| = |\psi|$, pues ψ difiere de ϕ sólo por contener β en algunos lugares donde ϕ contiene α. Así, en este caso $|\alpha \Leftrightarrow \beta| = 1$ y $|\phi \Leftrightarrow \psi| = 1$, de donde $|(\alpha \Leftrightarrow \beta) \Rightarrow (\phi \Leftrightarrow \psi)| = 1$.

b. Inmediato de la demostración de **a.** ■

Ejemplo. Sean $\phi = (\neg P \vee Q) \wedge R$, $\alpha = \neg P \vee Q$, y $\beta = P \Rightarrow Q$, luego $\psi = \phi(\alpha/\beta) = (P \Rightarrow Q) \wedge R$. Ahora como $\alpha \underset{T}{\models\!\models} \beta$, se sigue que $\phi \underset{T}{\models\!\models} \psi$.

Ejercicios

1. Pruebe que $\phi \vDash_T \psi$ para cada uno de los pares siguientes:

ϕ	ψ
$P \Rightarrow (Q \Rightarrow R)$	$P \wedge Q \Rightarrow R$
$(P \vee R) \wedge (Q \vee \neg R)$	$P \vee Q$
$P \wedge Q$	Q
P	$P \vee Q$

2. Pruebe que $\phi \underset{T}{\models\!\models} \psi$ si y sólo si $\vDash \phi \Leftrightarrow \psi$.

3. Pruebe que la equivalencia tautológica es una relación de equivalencia.

4. Pruebe los teoremas 4.2 y 4.3.

5. Verifique si el bicondicional es conmutativo y asociativo.

6. Sea $\Gamma \cup \{\alpha, \beta\} \subseteq \Phi(\mathscr{P})$. Pruebe o refute mediante un contraejemplo:

i. Si $\Gamma \models_T \alpha$ o $\Gamma \models \beta$, entonces $\Gamma \models_T (\alpha \vee \beta)$.

ii. Si $\Gamma \models_T (\alpha \vee \beta)$, entonces $\Gamma \models_T \alpha$ o $\Gamma \models_T \beta$.

iii. Si $\Gamma \models_T \neg\alpha$, entonces $\Gamma \not\models_T \alpha$.

7. Sea $\beta \in \Phi(\mathscr{P})$. Si para toda valuación $|\cdot|$, se tiene que $|\beta| = 0$, entonces para toda $\alpha \in \Phi(\mathscr{P})$, $\beta \models_T \alpha$.

8. Sea $\beta \in \Phi(\mathscr{P})$. Si para toda valuación $|\cdot|$, se tiene que $|\beta| = 1$, entonces para toda $\alpha \in \Phi(\mathscr{P})$, $\alpha \models_T \beta$.

***9.** Pruebe el Teorema 4.5.

4.5 Formas normales y el problema de síntesis

Definición. Una función de verdad n-aria (para $n \in \mathbb{N}$) es una función $f: \{0,1\}^n \to \{0,1\}$.

Ejercicio. Pruebe que para cada $n \in \mathbb{N}$, hay exactamente 2^{2^n} funciones de verdad distintas. (Sugerencia: Inducción sobre n).

Sean P y Q dos letras proposicionales y formemos las fórmulas siguientes, que denominaremos elementales:

$$\alpha_1 = P \wedge Q, \quad \alpha_2 = P \wedge \neg Q, \quad \alpha_3 = \neg P \wedge Q \quad \text{y} \quad \alpha_4 = \neg P \wedge \neg Q$$

Construyamos ahora todas las disyunciones posibles con estas fórmulas elementales en combinaciones de 4 a 4, 3 a 3, 2 a 2, 1 a 1 y 0 a 0, y calculemos sus valuaciones. Obtendremos los 16 casos ilustrados en la tabla siguiente (*cf.* [Pi]).

combinaciones	número	disyunciones	vector de valuaciones
4 a 4	1	$\alpha_1 \vee \alpha_2 \vee \alpha_3 \vee \alpha_4$	1 1 1 1
3 a 3	2	$\alpha_1 \vee \alpha_2 \vee \alpha_3$	1 1 1 0
	3	$\alpha_1 \vee \alpha_2 \vee \alpha_4$	1 1 0 1
	4	$\alpha_1 \vee \alpha_3 \vee \alpha_4$	1 0 1 1
	5	$\alpha_2 \vee \alpha_3 \vee \alpha_4$	0 1 1 1
2 a 2	6	$\alpha_1 \vee \alpha_2$	1 1 0 0
	7	$\alpha_1 \vee \alpha_4$	1 0 0 1
	8	$\alpha_1 \vee \alpha_3$	1 0 1 0
	9	$\alpha_2 \vee \alpha_4$	0 1 0 1
	10	$\alpha_2 \vee \alpha_3$	0 1 1 0
	11	$\alpha_3 \vee \alpha_4$	0 0 1 1
1 a 1	12	α_1	1 0 0 0
	13	α_2	0 1 0 0
	14	α_3	0 0 1 0
	15	α_4	0 0 0 1
0 a 0	16		0 0 0 0

Tabla 4.6

Observación. Existe una similitud entre la distribución (ocurrencia) de las α_i's en las disyunciones y el vector de las valuaciones correspondientes: donde aparece una α_i en la fórmula, aparece un "1" en el vector, mientras que su ausencia se corresponde con un "0".

Definición. Una fórmula es una *forma normal disyuntiva* (FND) si es una disyunción cuyas componentes consisten de conjunciones de *literales*, donde cada literal consta de una letra proposicional o su negación. Una FND es *completa* si ninguna componente contiene dos ocurrencias de una misma letra proposicional, y si una letra ocurre en una componente, ocurre en todas.

Ejemplos:

Las siguientes fórmulas son FND's:

1. $(P \wedge \neg Q \wedge R) \vee (\neg Q \wedge S)$, 2. $(W \wedge \neg Y \wedge \neg Q)$, 3. $(A \wedge \neg T) \vee (E \wedge \neg A \wedge \neg T)$,

4. $(Q \wedge T \wedge \neg P) \vee (T \wedge \neg Q \wedge P) \vee (\neg T \wedge \neg Q \wedge \neg P)$ es además completa.

Consideremos ahora una fórmula α con n letras proposicionales: $P_1, P_2, \ldots, P_n$. Entonces α determina una función de verdad n-aria, f_α, definida de la siguiente forma

$$f_\alpha(x_1, x_2, \ldots, x_n) = \bar{\nu}(\alpha)$$

donde $\bar{\nu}$ es la valuación tal que $\nu(P_i) = x_i$, para $1 \leq i \leq n$.

Las fórmulas con 2 letras proposicionales determinan funciones de verdad binarias. Del ejercicio anterior, hay $2^{2^2} = 16$ funciones tales. Estas funciones están representadas por los 16 casos de la tabla 4.6, por tanto toda fórmula con 2 letras proposicionales es equivalente a una FND. De aquí conjeturamos que dada cualquier función de verdad n-aria existe una *FND completa* que le es tautológicamente equivalente; por lo que cada forma normal viene a ser un representante de cada clase de equivalencia definida por la relación de equivalencia tautológica $\models\!\models_T$. Por consiguiente, aunque el número de fórmulas (aun considerando sólo las de dos letras) es infinito, el análisis se remite a unas cuantas fórmulas representativas.

Veamos los 16 casos de la Tabla 4.6:

1) es una fórmula válida.

2) es $P \vee Q$. Explicitando,

$(P \wedge Q) \vee (P \wedge \neg Q) \vee (\neg P \wedge Q)$,

$\models ((P \vee Q) \Rightarrow ((P \wedge Q) \vee (P \wedge \neg Q) \vee (\neg P \wedge Q)))$.

3) es $Q \Rightarrow P$.

4) es $P \Rightarrow Q$.

5) es $P|Q$, que significa "ni P ni Q". Esta es la operación de *incompatibilidad* o *negación alterna* (Sheffer 1913),

$\models ((P|Q) \Leftrightarrow (((P \wedge \neg Q) \vee (\neg P \wedge Q) \vee (\neg P \wedge \neg Q)))$.

Debido también que $\models ((P|Q) \Leftrightarrow (\neg(P \wedge Q)))$, es más frecuente llamarle operación *NAND*.

6) es P.

7) es $P \Rightarrow Q$.

8) es Q.

9) es $\neg Q$.

10) es $P \dot{\vee} Q$. Es la *disyunción exclusiva*, y equivale a $\neg(P \Leftrightarrow Q)$. Suele denominarse también *XOR*.

11) es $\neg P$.
12) es α_1 o bien $P \wedge Q$.
13) es α_2 o bien $\neg(P \Rightarrow Q)$.
14) es α_3 o bien $\neg(Q \Rightarrow P)$.
15) es α_4 o $P \downarrow Q$. Esta operación se conoce como *negación disjunta* (Pierce). Equivale a $\neg(P \vee Q)$, por lo que suele llamarse *NOR*.
16) es una fórmula contradictoria.

En un *problema de análisis,* una fbf α es dada y el objetivo es (viendo a α como una "caja negra") investigar la respuesta (output) de α bajo todos los posibles valores de verdad (inputs) asignados a las letras proposicionales que ocurren en α. Esto lo llevamos a efecto mediante la construcción de la tabla de verdad asociada a α (*i.e.*, hallamos la función de verdad f_α). De manera recíproca, la observación hecha respecto a la tabla 4.6 nos permite extraer un procedimiento para, dada la tabla de verdad, encontrar una fórmula α_f con los valores de verdad apropiados. Esto se denomina *problema de síntesis.*

El procedimiento para síntesis es el siguiente:

De la observación hecha resulta que sólo es necesario considerar las fórmulas α_i's que se corresponden con los valores "1"'s. Y como cada α_i es una conjunción de literales, α_i es verdadera sólo cuando todas sus literales lo son y viceversa. Finalmente, la disyunción de estas α_i's proporciona la fórmula α en FND completa deseada. Este proceso resulta susceptible de generalización, y tenemos:

Objetivo: asignar a toda función n-aria f una fórmula α_f tal que la función de verdad n-aria g asociada a la fórmula α_f, g_{α_f}, sea precisamente f, *i.e.*, $g_{\alpha_f} = f$.

En efecto, sea f una función de verdad n-aria, con $n \geq 1$. Tenemos dos casos:

i) Si $f \equiv 0$, *i.e.*, para toda n-ada $(x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$ se tiene $f(x_1, x_2, \ldots, x_n) = 0$, hacemos $\alpha_f = P \wedge \neg P$.

ii) Si $f \not\equiv 0$, sean $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^k$ una enumeración de todas aquellas sucesiones $\mathbf{x}^i = (x_1^i, x_2^i, \ldots, x_n^i) \in \{0, 1\}^n$ tales que $f(\mathbf{x}^i) = 1$, para $1 \leq i \leq k$. Así, para $1 \leq i \leq k$, sea $\alpha_i = x_1^i P_1 \wedge x_2^i P_2 \wedge \ldots \wedge x_n^i P_n$, donde escribimos

$$\begin{cases} 0 & P_j = \neg P_j \\ 1 & P_j = P_j \end{cases}, \text{ para } 1 \leq j \leq n,$$

y finalmente, definimos $\alpha_f = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_k$.

Ejemplo. Determinemos una fórmula α_f, dada la función de verdad f. En efecto, sea $f = \{(1, 1, 1; 0), (1, 1, 0; 1), \ldots, (0, 0, 0; 0)\}$ una función de verdad de aridad 3. En forma tabular, tenemos:

$\lvert P_1\rvert$	$\lvert P_2\rvert$	$\lvert P_3\rvert$	$f(\lvert P_1\rvert, \lvert P_2\rvert, \lvert P_3\rvert)$	
1	1	1	0	
1	1	0	1	$\longleftarrow \mathbf{x}^1 = (1, 1, 0)$
1	0	1	0	
1	0	0	1	$\longleftarrow \mathbf{x}^2 = (1, 0, 0)$
0	1	1	1	$\longleftarrow \mathbf{x}^3 = (0, 1, 1)$
0	1	0	0	
0	0	1	1	$\longleftarrow \mathbf{x}^4 = (0, 0, 1)$
0	0	0	0	

Tabla 4.7

Aquí, $n = 3$ y $k = 4$. Luego, si hacemos:

$$\alpha_1 = P_1 \wedge P_2 \wedge \neg P_3$$
$$\alpha_2 = P_1 \wedge \neg P_2 \wedge \neg P_3$$
$$\alpha_3 = \neg P_1 \wedge P_2 \wedge P_3$$
$$\alpha_4 = \neg P_1 \wedge \neg P_2 \wedge P_3$$

entonces definimos la FND completa buscada como $\alpha_f = \alpha_1 \vee \alpha_2 \vee \alpha \vee \alpha_4$. □

De este proceso de síntesis se sigue el siguiente teorema, cuya demostración, omitida por ser un tanto engorrosa, se reduce básicamente a probar que dos funciones de verdad son iguales.

Teorema 4.9. *La fórmula en FND completa α_f obtenida mediante este proceso de síntesis es tal que su función de verdad asociada es precisamente f.* ■

Aunque la forma α_f hallada con este procedimiento no suele ser mínima desde el punto de vista de su longitud, sí resulta normal (canónica) en el sentido de que el algoritmo empleado para hallarla siempre da el resultado deseado.

La figura 4.4 ilustra los procesos de análisis y síntesis:

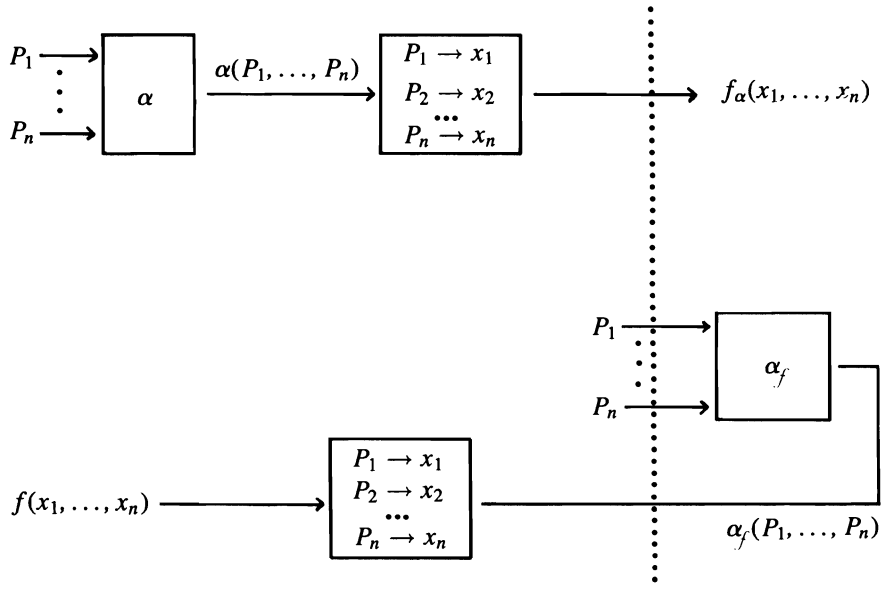


Figura 4.4

Definición. Una fórmula es una *forma normal conjuntiva* (FNC) si es una conjunción cuyas componentes consisten de disyunciones de literales. Una FNC es completa si ninguna componente contiene dos ocurrencias de una misma letra proposicional, y si una letra ocurre en una componente, ocurre en todas.

De lo expuesto en esta sección, se tiene un algoritmo para realizar síntesis, el cual puede ser usado para hallar la FND completa asociada a una fórmula: Dada α, se construye su tabla de verdad, y de ésta obtenemos la FND completa. Sin embargo, este proceso resulta ineficiente cuando el número letras que ocurren en la fórmula es grande. A continuación presentamos un procedimiento alterno al de las tablas de verdad conocido como *reducción a formas normales*, el cual se basa en la noción de equivalencia tautológica entre fórmulas.

Algoritmo para transformar fórmulas a las formas normales:

Paso 1. Use las leyes:

i) $\alpha \Leftrightarrow \beta \underset{T}{\models\!\models} (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

ii) $\alpha \Rightarrow \beta \underset{T}{\models\!\models} \neg\alpha \vee \beta$

Paso 2. Aplique cuantas veces sean necesarias las leyes de la doble negación y de De Morgan. Para llevar los signos de negación hasta las letras proposicionales.

Paso 3. Aplique repetidamente las leyes distributivas, así como las demás tautologías de la Tabla 4.5, para obtener la forma normal deseada.

Ejemplo. Obtengamos las FN's disyuntiva y conjuntiva para la fórmula $Q \wedge (Q \Rightarrow P)$:

$Q \wedge (Q \Rightarrow P) \underset{T}{\models\!\models} Q \wedge (\neg Q \vee P)$ es una FNC, mientras que

$Q \wedge (Q \Rightarrow P) \underset{T}{\models\!\models} Q \wedge (\neg Q \vee P) \underset{T}{\models\!\models} (Q \wedge \neg Q) \vee (Q \wedge P)$ es una FND. □

Observación. Las FN's obtenidas no son necesariamente únicas. No así las FN's completas que sí son únicas, salvo permutaciones de sus componentes o de sus literales.

Así, para el ejemplo anterior, como $(Q \wedge \neg Q) \vee (Q \wedge P) \underset{T}{\models\!\models} (Q \wedge P)$, entonces $(Q \wedge P)$ es la FND completa asociada.

El algoritmo para obtener las FN's completas se sigue del esbozado para FN's en general, anexando los pasos siguientes:

Paso 4. Las componentes que contengan fbf's contradictorias de la forma $P \wedge \neg P$ son eliminadas para las FND's, mientras que las que contengan tautologías en $P \vee \neg P$ lo serán de las FNC's.

Paso 5. Las componentes idénticas también se eliminan.

Paso 6. Las componentes se completan introduciendo los factores faltantes.

De la unicidad de las FN's completas se sigue que para verificar si dos fórmulas son tautológicamente equivalentes, podemos comparar si las FN's respectivas son idénticas.[3]

[3]Nótese la similitud entre este proceso y el *algoritmo de reducción* empleado en la sección 3.3.

Ejemplo. Probemos que para las fórmulas α y β siguientes $\alpha \underset{T}{\models\!\models} \beta$:

$$\alpha = P \wedge (\neg Q \Rightarrow P) \quad \text{y} \quad \beta = P$$

En efecto,

$$\begin{aligned} \alpha = P \wedge (\neg Q \Rightarrow P) &\underset{T}{\models\!\models} P \wedge (\neg\neg Q \vee P) \\ &\underset{T}{\models\!\models} P \wedge (Q \vee P) \underset{T}{\models\!\models} (P \vee (Q \wedge \neg Q)) \wedge (Q \vee P) \\ &\underset{T}{\models\!\models} ((P \vee Q) \wedge (P \vee \neg Q)) \wedge (Q \vee P) \\ &\underset{T}{\models\!\models} (P \vee \neg Q) \wedge (Q \vee P) \end{aligned}$$

Mientras que

$$\beta = P \underset{T}{\models\!\models} P \vee (Q \wedge \neg Q) \underset{T}{\models\!\models} (P \vee Q) \wedge (P \vee \neg Q).$$

Ejercicios

1. Halle una fórmula que corresponda a la tabla de verdad dada:

a)

$\lvert P\rvert$	$\lvert Q\rvert$	$\lvert R\rvert$	$f(\lvert P\rvert, \lvert Q\rvert, \lvert R\rvert)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	0
0	0	0	1

b)

$\lvert P\rvert$	$\lvert Q\rvert$	$\lvert R\rvert$	$f(\lvert P\rvert, \lvert Q\rvert, \lvert R\rvert)$
1	1	1	0
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

2. Proporcione un algoritmo para determinar la FNC completa que corresponda a una tabla de verdad dada.

3. Pruebe que para $\alpha \in \Phi(\mathscr{P})$, $\alpha \vee (P \wedge \neg P) \underset{T}{\models\!\models} \alpha$ y que $\alpha \wedge (P \vee \neg P) \underset{T}{\models\!\models} \alpha$.

4. Transforme a una FN cada una de las fórmulas siguientes:

i. $(Q \Leftrightarrow P) \wedge (\neg P \Rightarrow R)$

ii. $(P|\neg Q) \Rightarrow \neg(P \wedge Q \wedge R)$

iii. $(P \Leftrightarrow R) \vee ((\neg\neg Q \Rightarrow P) \wedge \neg R)$

iv. $((\neg P \vee R) \Leftrightarrow (P \Rightarrow (\neg Q \wedge Q))) \Rightarrow \neg(\neg P \vee \neg Q)$

***5.** Pruebe que para cada $n \in \mathbb{N}$, hay exactamente 2^{2^n} funciones de verdad distintas. (Sugerencia: Inducción sobre n).

***6.** Pruebe el Teorema 4.9.

4.6 Conjuntos funcionalmente completos de conectivos, lógica combinacional

De la sección anterior podemos concluir que toda fórmula α es tautológicamente equivalente a una FND completa, lo cual es traducible a que α la podemos representar usando sólamente tres conectivos lógicos: *negación, disyunción* y *conjunción.* Este número de conectivos es aun reducible a dos (uno unario y el otro binario) [Me].

> **Definición.** Un conjunto de conectivos es *funcionalmente completo* si y sólo si cualquier función de verdad se puede corresponder con una fórmula en la que sólo aparecen los conectivos del conjunto.

Corolario 4.10. *Las parejas* $\{\neg, \wedge\}$, $\{\neg, \vee\}$ *y* $\{\neg, \Rightarrow\}$ *son conjuntos funcionalmente completos de conectivos.*

Demostración.

Tenemos que $\models ((P \vee Q) \Leftrightarrow \neg(\neg P \wedge \neg Q))$, de donde por la parte b del Teorema 4.8, toda fbf en la que ocurren los conectivos $\neg$, $\wedge$ y $\vee$ es tautológicamente equivalente a una fórmula en la que ocurren sólo $\neg$ y $\wedge$ (obtenida reemplazando todas las expresiones $(\alpha \vee \beta)$ por $\neg(\neg\alpha \wedge \neg\beta)$). Los demás casos se siguen de las tautologías:

$$\models ((P \wedge Q) \Rightarrow \neg(\neg P \vee \neg Q))$$
$$\models ((P \vee Q) \Rightarrow (\neg P \Rightarrow Q))$$
$$\models ((P \wedge Q) \Rightarrow \neg(P \Rightarrow \neg Q))$$

■

Pero inclusive podemos ir más lejos y usar un solo conectivo lógico binario para para dar cuenta de todos los demás conectivos (operaciones), pudiendo, por tanto, realizar el proceso de síntesis con un único conectivo.

Corolario 4.11. *Los únicos conectivos binarios que pueden ser empleados sólos para la representación de todas las funciones de verdad son* $\downarrow$ *y* $|$.

Demostración.

Consideremos la tabla siguiente de equivalencias:

Conectivo	$P \vee Q$	$P \wedge Q$	$\neg P$
$\downarrow$		$(P \downarrow P) \downarrow (Q \downarrow Q)$	$P \downarrow P$
$\mid$	$(P \mid P) \mid (Q \mid Q)$		$P \mid P$

Tabla 4.8

De aquí que, por el corolario 4.10 (anterior), cualquier fórmula es tautológicamente equivalente a una que sólo involucre los conectivos $\downarrow$ ó $|$.

Para terminar la prueba resta demostrar que éstos son únicos. Para ello postulemos la existencia de otro conectivo binario con esta propiedad. Sea $H(P, Q)$ el conectivo adecuado y denotemos con $h(x_1, x_2)$ su función de verdad correspondiente. Así, si $h(1, 1) = 1$, entonces la fórmula contruida usando sólo H tomaría el valor de verdad 1 cuando todas sus letras proposicionales tomaran el valor 1 (*e.g.*, para la fórmula $\alpha = H(H(P_1, P_2), H(H(P_2, P_3), P_4))$, su función $f_\alpha(\mathbf{x}) = 1$, si $x_1 = x_2 = x_3 = x_4 = 1$). Pero de esta manera, $\neg P$ no sería definible en términos de H, luego $h(1, 1) = 0$. De manera análoga, $h(0, 0) = 1$. Así, la tabla de verdad para el conectivo H es hasta el momento:

x_1	x_2	$h(x_1 x_2)$
1	1	0
1	0	?
0	1	?
0	0	1

Tabla 4.9

Ahora, si el segundo y tercer renglones fuesen "?,?" = "0,0" ó "1,1", tendríamos que H es precisamente $\downarrow$ ó $|$, respectivamente. De no ser así, tenemos dos casos a

considerar:

i) "0,1" , y entonces $\models (H(P, Q) \Leftrightarrow \neg P)$

ii) "1,0" , y entonces $\models (H(P, Q) \Leftrightarrow \neg Q)$

En ambos casos, H sería definible en términos de la negación "¬"; pero ésta no es adecuada por sí sóla, porque las únicas fórmulas definibles con ella son la identidad "P" y la propia negación, mientras que una fórmula cuya función de verdad sea la constante 1 (*i.e.*, cualquier tautología) no sería definible. ■

Los métodos de análisis y síntesis suelen aplicarse en una técnica conocida como *lógica combinacional* [DG]. Para este efecto, los valores lógicos "0" y "1" son interpretados físicamente como dos voltajes diferentes en un circuito eléctrico, y los conectivos proposicionales por medio de dispositivos físicos conocidos como *compuertas* (*gates* en inglés). Estos dispositivos reconfiguran voltajes de acuerdo a la tabla de verdad del conectivo que simulan, adquiriendo así la misma denominación. Por ejemplo, una compuerta *AND* (y) requiere de dos voltajes de entrada, dando un único valor de salida, y se denota bien sea por el *diagrama de bloque* (Fig. 4.5a), el cual da la expresión lógica en forma proposicional, o bien el símbolo especial para circuitos (Fig. 4.5 b), el cual especifica operacionalmente a la compuerta en términos de los valores de verdad de entrada (input) y salida (output).

a) b)

Figura 4.5

La combinación de estos dispositivos da lugar a representaciones circuitales que constituyen realizaciones físicas de fórmulas de la lógica proposicional.

La Figura 4.6, ilustra los diagramas de bloques y los símbolos especiales correspondientes usualmente empleados en lógica combinacional.

Ejemplo. Denotemos la fórmula $\phi = ((P \vee Q) \Rightarrow R) \wedge S$ en diagrama de bloques y en símbolos circuitales.

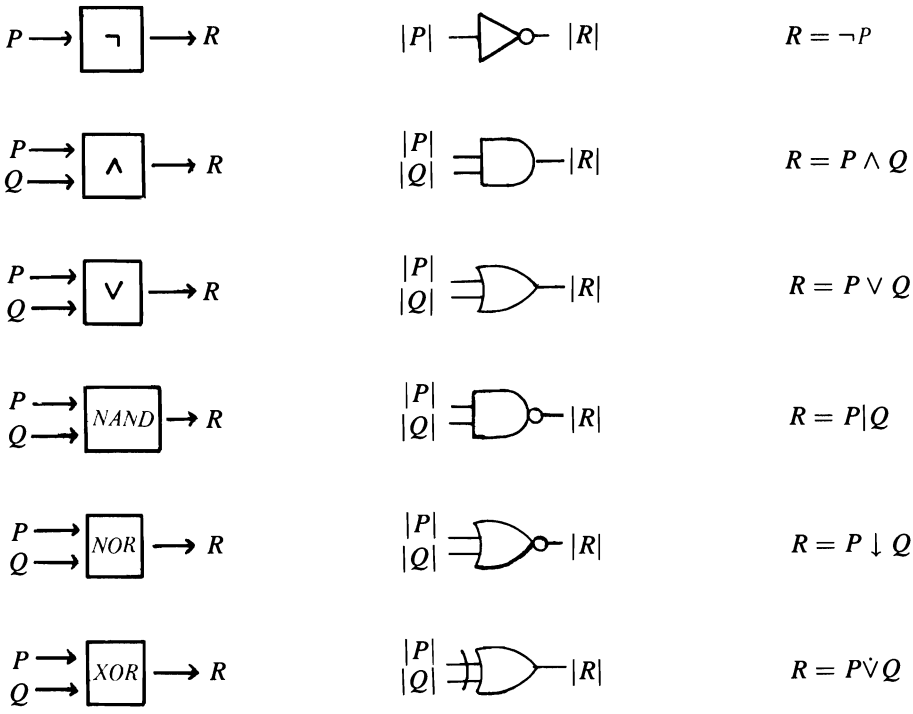


Figura 4.6

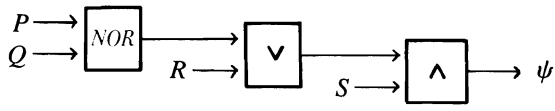
Solución:

Hagamos $\alpha = (P \vee Q) \Rightarrow R$, $\beta = \neg(P \vee Q) \vee R$ y $\psi = (\neg(P \vee Q) \vee R) \wedge S$. Así, ya que $\alpha \underset{T}{\models\!\models} \beta$, por el teorema 4.8 (parte b), entonces $\phi \underset{T}{\models\!\models} \psi$. Por lo tanto, su representación en diagrama de bloques está dada por la Fig. 4.7a), mientras que en símbolos especiales por la Fig. 4.7b).

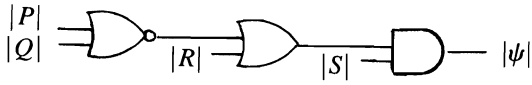
Ejercicios

1. Pruebe que los pares $\{\neg, \Leftrightarrow\}$ y $\{\vee, \Rightarrow\}$ no son funcionalmente completos.

2. Halle la FND que corresponde a la tabla de verdad siguiente y simplifique esta fórmula de modo que sólo aparezcan los conectivos lógicos $\wedge$ y $\vee$.



a)



b)

Figura 4.7

Explique.

$\|P\|$	$\|Q\|$	$f(\|P\|, \|Q\|)$
1	1	0
1	0	1
0	1	1
0	0	1

3. Represente mediante diagramas de bloques y símbolos circuitales las fórmulas siguientes:

i. $(P \wedge Q) \Rightarrow R$,

ii. $\neg Q \Leftrightarrow (\neg(P \Rightarrow P) \vee Q)$,

iii. $\neg\neg(\neg T \wedge S) \Rightarrow (P \vee \neg Q \vee \neg T)$,

iv. $P \wedge \neg P \wedge \neg R \wedge T$,

v. $(\neg F \vee T \vee Q \vee \neg T) \Leftrightarrow ((W \wedge \neg Q \wedge T) \Rightarrow \neg(\neg R \vee (\neg W \Rightarrow \neg F)))$

4. Construya dos circuitos representados mediante las fórmulas siguientes, bajo la restricción de que sólo se dispone de dos compuertas NAND y una NOR:

$$\alpha = (P \Rightarrow (\neg Q \wedge \neg P)) \vee \neg Q \quad \text{y} \quad \beta = \neg P \wedge \neg(P \vee \neg Q)$$

4.7 Satisfacibilidad

En la sección 4.1 definimos a la lógica como el estudio o análisis de los métodos de razonamiento correctos. Desde otra perspectiva veremos que en realidad no hay ninguna diferencia entre estos dos enfoques.

La lógica también puede ser pensada como el estudio de *conjuntos consistentes* de enunciados. Pero la palabra "consistencia" en lógica tiene un significado muy preciso, el tipo de consistencia que nos interesa en lógica es la *compatibilidad de enunciados*. Cuando decimos, por ejemplo, que una persona que predica una cosa y hace otra es inconsistente, o que alguien que apoya a un partido político en una elección y a otro en la siguiente es inconsistente, en realidad estamos hablando de sinceridad o lealtad. Cuando en lógica decimos que un conjunto de enunciados es consistente estamos afirmando que los enunciados del conjunto son compatibles entre sí, esto es, que es posible para todos los enunciados del conjunto ser verdaderos al mismo tiempo en alguna situación. Veamos algunos ejemplos.

Supongamos que alguien dice: "No importa que haya programas violentos en la televisión porque la televisión no afecta el comportamiento de los jóvenes, pero debería haber más programas educativos para que los jóvenes se interesaran en los libros". Esta persona está afirmando dos enunciados que no son consistentes entre sí, pues bajo ninguna circunstancia se podría dar que la televisión afectara y no afectara el comportamiento juvenil.

Además, supongamos que alguien dice: "Yo sé todo lo que tengo que saber para pasar mis exámenes; todo lo que me han enseñado lo he entendido y aprendido; pero en todos los exámenes he tenido muy mala suerte y por eso los he reprobado todos". Estos enunciados constituyen un conjunto consistente. Es posible (aunque extremadamente improbable) que sean todos verdaderos.

Analizar si un conjunto de enunciados es consistente o no, para el lenguaje que hemos estado estudiando es muy fácil, pues ya tenemos todas las situaciones posibles: las valuaciones. Dado un conjunto de fórmulas bien formadas, que son las expresiones que representan a los enunciados, decimos que es consistente si existe alguna valuación para las letras proposicionales bajo la cual todas las fórmulas del conjunto sean verdaderas.

Para demostrar que un conjunto de fórmulas es consistente basta exhibir una valuación bajo la cual todas las fórmulas del conjunto sean verdaderas. Si por el contrario, queremos demostrar que un conjunto de fórmulas no es consistente

tendremos que hacer una demostración general de que ninguna valuación hace verdaderas a todas las fórmulas del conjunto.[4]

A modo de ejemplo probaremos que el siguiente conjunto de fórmulas no es consistente: $\{P \wedge Q, P \Rightarrow R, \neg R\}$. Supongamos que existiera alguna valuación $|\cdot|$ para las letras que satisface a todas las fórmulas del conjunto, esto es, tal que $|P \wedge Q| = |P \Rightarrow R| = |\neg R| = 1$. Entonces, de la primera fórmula, se tiene que $|P| = |Q| = 1$; de la segunda, como el antecedente es verdadero, se obtiene $|R| = 1$, pero la tercera implica que $|R| = 0$. Esto es una contradicción, por lo que concluímos que tal valuación no puede existir y por tanto el conjunto es inconsistente.

Como la palabra consistencia tiene otro significado en lógica, para evitar ambigüedades de ahora en adelante llamaremos *satisfacibles* a los conjuntos consistentes en el sentido que acabamos de ver.

Definición. Sea $\Gamma \subseteq \Phi(\mathscr{P})$. Decimos que Γ es *satisfacible* si y sólo si existe una valuación $|\cdot|$ que satisface a Γ, *i.e.*, para toda $\alpha \in \Gamma$, $|\alpha| = 1$.

Definición. Γ es *insatisfacible* si y sólo si no existe valuación $|\cdot|$ alguna que satisfaga a todas las fórmulas de Γ al mismo tiempo, *i.e.*, dada cualquier $|\cdot|$, existe al menos una $\alpha \in \Gamma$ tal que $|\alpha| = 0$.

Observaciones:

1) Una fórmula α es insatisfacible si y sólo si α es una fórmula contradictoria.
2) $\models \alpha$ si y sólo si $\neg\alpha$ es insatisfacible.

Teorema 4.12. *Γ es satisfacible si y sólo si existe $\alpha \in \Phi(\mathscr{P})$ tal que $\Gamma \nvdash_T \alpha$ (i.e., Γ no implica cualquier fórmula).*

Demostración.

Supongamos que Γ es satisfacible, y consideremos cualquier fórmula contradictoria, por ejemplo, $\alpha = P \wedge \neg P$, entonces $\Gamma \nvdash_T \alpha$, pues $|\alpha| = 0$, para cualquier valuación que satisfaga a Γ.

Ahora si $\Gamma \nvdash_T \alpha$, el resultado es obvio, ya que esta expresión significa que existe una valuación que satisface a Γ, pero no a α; luego Γ es satisfacible. ■

[4]Esta es la razón de que a los conjuntos consistentes de fórmulas se les llama también *satisfacibles.*

Lema 4.13. *Sea* $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, *para algún* $n \in \mathbb{N}$. *Entonces,* Γ *es satisfacible si y sólo si la fórmula* $\beta = \alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n$ *es satisfacible.*

Demostración.

Γ es satisfacible si y sólo si existe una valuación $|\cdot|$ tal que para toda $\alpha_i \in \Gamma$ se tiene que $|\alpha_i| = 1$, para $1 \leq i \leq n$, si y sólo si (por el ejercicio 8, secc. 4.3) $|\beta| = \min\{|\alpha_1|, |\alpha_2|, \ldots, |\alpha_n|\} \leq |\alpha_i| = 1$, para $1 \leq i \leq n$. ■

Definición. Consideremos un argumento $\Gamma \vDash_T \alpha$, entonces, el *conjunto contraejemplo* está dado por $\Gamma \cup \{\neg\alpha\}$, *i.e.*, es el conjunto formado por las premisas del argumento, Γ, y la negación de la conclusión, $\neg\alpha$.

Teorema 4.14. $\Gamma \vDash_T \alpha$ *si y sólo si* $\Gamma \cup \{\neg\alpha\}$ *es insatisfacible. En particular, si* Γ *está dado por* $\Gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, $\Gamma \vDash \alpha$ *si y sólo si* $\alpha_1 \wedge \ldots \wedge \alpha_n \wedge \neg\alpha$ *es contradictoria.*

Demostración.

Para el caso de Γ insatisfacible, el resultado se sigue de la definición.

$\Gamma \vDash_T \alpha$ significa que para toda valuación $|\cdot|$ que satisface a Γ, se tiene también que $|\alpha| = 1$, o sea el conjunto $\Gamma \cup \{\neg\alpha\}$ es insatisfacible, ya que $|\neg\alpha| = 0$. Supongamos ahora que $\Gamma \cup \{\neg\alpha\}$ es insatisfacible y que Γ es satisfacible bajo una valuación $|\cdot|$, luego $|\neg\alpha|$ es 0, y por tanto, $|\alpha| = 1$, *i.e.*, $\Gamma \vDash_T \alpha$.

Si $\Gamma = \{\alpha_1, \ldots, \alpha_n\}$, entonces, $\Gamma \vDash_T \alpha$ si y sólo si $\Gamma \cup \{\neg\alpha\}$ es insatisfacible, y esto último equivale, por el lema 4.13, a que $\alpha_1 \wedge \ldots \wedge \alpha_n \wedge \neg\alpha$ es contradictoria. ■

Observación. Este teorema implica que el conjunto contraejemplo de un argumento es insatisfacible cuando es imposible de que todos los enunciados del conjunto sean verdaderos, esto es, siempre que todas las premisas del argumento son verdaderas, la conclusión también lo es, y esto se da si y sólo si el argumento es correcto. El método de demostración que se basa en la implicación "*si* $\Gamma \cup \{\alpha\}$ es *insatisfacible entonces* $\Gamma \vDash_T \neg\alpha$" se denomina *reducción al absurdo*.

Ejercicio

Pruebe la (in) satisfacibilidad de los conjuntos de fórmulas siguientes

i. $\Gamma = \{P \Rightarrow Q, \neg P \Rightarrow R, \neg Q \Rightarrow \neg R, \neg Q\}$

ii. $\Gamma = \{(\neg P \Rightarrow Q) \Rightarrow R, \neg R, (\neg Q \Rightarrow P), \neg S\}$

iii. $\Gamma = \{A \wedge \neg V, (\neg B \wedge F) \Rightarrow \neg Q, \neg(Q \wedge F) \Rightarrow S, \neg V \Rightarrow (A \Rightarrow \neg B), \neg S\}$

4.8 Técnicas semánticas de argumentación

Como objetivo de esta sección tenemos la tarea de proveer técnicas que nos permitan verificar argumentaciones correctas, *i.e.*, para $\Gamma \cup \{\alpha\} \subseteq \Phi(\mathscr{P})$ dados, Γ finito, si $\Gamma \models \alpha$. Para esto, los Teoremas 4.5 y 4.14 nos proporcionan *modos equivalentes* de expresar la consecuencia tautológica de una conclusión α a partir de un conjunto Γ de premisas. Ahora bien, la manera de verificar que estamos en posesión de alguno de estos modos equivalentes es mediante las técnicas que damos a continuación.

a. Uso de tablas de verdad y de la definición del condicional

En esta técnica se contemplan dos casos:

1°) *Encadenamiento hacia delante* (forward chaining). Se verifican todas las instancias en las que las premisas son verdaderas. Si de aquí tenemos que la conclusión es siempre verdadera bajo estas instancias, entonces $\Gamma \models_T \alpha$.

2°) *Encadenamiento hacia atrás* (backward chaining). Si revisando todas las instancias en las que α es falsa, tenemos que siempre alguna de las premisas es también falsa, entonces $\Gamma \models_T \alpha$.

Estos casos deben sus nombres al hecho de que en el 1° se comienza examinando los valores de verdad de las premisas y de aquí verificamos los de la conclusión; mientras que en el 2° es lo contrario, "vamos" de la conclusión a las premisas.

Observación. De manera implícita hacemos uso del Teorema 4.5 (de deducción).

Ejemplos:

Verifique si $\Gamma \models_T \alpha$ o no para.

1. $\Gamma = \{\alpha_1, \alpha_2, \alpha_3\}$, con $\alpha_1 = \neg P \Rightarrow R$, $\alpha_2 = R \Rightarrow Q$, $\alpha_3 = \neg P$ y $\alpha = Q$.

$\lvert P\rvert$	$\lvert Q\rvert$	$\lvert R\rvert$	$\lvert\alpha_1\rvert$	$\lvert\alpha_2\rvert$	$\lvert\alpha_3\rvert$	$\lvert\alpha\rvert$	
1	1	1	1	1	0	1	
1	1	0	1	0	0	1	
1	0	1	1	1	0	0	$\longleftarrow$ 2º
1	0	0	1	1	0	0	$\longleftarrow$ 2º
0	1	1	1	1	1	1	$\longleftarrow$ 1º
0	1	0	0	1	1	1	
0	0	1	1	0	1	0	$\longleftarrow$ 2º
0	0	0	0	1	1	0	$\longleftarrow$ 2º

Tabla 4.10

Para el 1er caso, tenemos:

siempre que $\lvert\alpha_1\rvert = \lvert\alpha_2\rvert = \lvert\alpha_3\rvert = 1$, entonces $\lvert\alpha\rvert = 1$.

Mientras que para el 2º:

cuando $\lvert\alpha\rvert = 0$, se tiene que $\lvert\alpha_i\rvert = 0$ para al menos algún $i = 1, 2$ o 3.

Así, de ambos casos podemos concluir que $\Gamma \models_T \alpha$.

2. $\Gamma = \{\alpha_1, \alpha_2\}$, con $\alpha_1 = P \wedge Q$, $\alpha_2 = \neg P \vee Q$ y $\alpha = \neg Q$.

$\lvert P\rvert$	$\lvert Q\rvert$	$\lvert\alpha_1\rvert$	$\lvert\alpha_2\rvert$	$\lvert\alpha\rvert$	
1	1	1	1	0	$\longleftarrow$ 1º y 2º
1	0	0	0	1	
0	1	0	1	0	
0	0	0	1	1	

Tabla 4.11

Aquí, tanto el primer como el segundo casos fallan, pues $\lvert\alpha_1\rvert = \lvert\alpha_2\rvert = 1$, pero $\lvert\alpha\rvert = 0$, y viceversa. Por lo tanto $\Gamma \not\models_T \alpha$, y una interpretación que falsea esta implicación es precisamente la dada. □

b. Método algebraico

Esta técnica se basa en transformar la argumentación a analizar bien sea a una FNC y aplicar así el Teorema 4.5, o bien a una FND y entonces aplicar el 4.14; para la transformación en cuestión hacemos uso del algoritmo presentado en la sección 4.5 y de los resultados de la Tabla 4.12, que proporcionamos a continuación.

Notación. El símbolo **1** (**0**) representa la función de verdad de cualquier tautología (fbf contradictoria), y por abuso de notación, las identificaremos.

Sea $\alpha \in \Phi(\mathscr{P})$,

$\alpha \vee \neg\alpha \models_T \mathbf{1}$	$\alpha \vee \mathbf{1} \models_T \mathbf{1}$	$\alpha \wedge \mathbf{1} \models_T \alpha$
$\alpha \wedge \neg\alpha \models_T \mathbf{0}$	$\alpha \vee \mathbf{0} \models_T \alpha$	$\alpha \wedge \mathbf{0} \models_T \mathbf{0}$

Tabla 4.12

Consideremos los dos casos:

1°) Usamos el Teorema 4.5 y transformamos $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \Rightarrow \alpha$ a una FNC. Si al final de las simplificaciones obtenemos **1**, diremos que $\models (\alpha_1 \wedge \ldots \wedge \alpha_n) \Rightarrow \alpha$, *i.e.* $\Gamma \models_T \alpha$.

2°) Se usa el teorema 4.14 sobre la insatisfacibilidad de $\Gamma \cup \{\neg\alpha\}$. Aquí, se transforma $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \wedge \neg\alpha$ a una FND. Si después de las simplificaciones obtenemos un **0**, entonces $(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \wedge \neg\alpha$ será una fórmula contradictoria, de donde, $\Gamma \cup \{\neg\alpha\}$ será insatisfacible.

Observación. Este método debe su nombre (algebraico) a la similitud que guardan los conectivos $\wedge$ y $\vee$ respecto a la multiplicación y suma algebraicas, hecha explícita por la Tabla 4.12 y algunas de las tautologías de la Tabla 4.5.

Ejemplo. Probemos que $\Gamma \models_T \alpha$, si $\Gamma = \{P \Rightarrow Q, \neg Q\}$ y $\alpha = \neg P$.

Por el 1er caso, tenemos:

$$
\begin{aligned}
((P \Rightarrow Q) \wedge \neg Q) \Rightarrow \neg P &\underset{T}{\models} \neg((P \Rightarrow Q) \wedge \neg Q) \vee \neg P \\
&\underset{T}{\models} \neg((\neg P \vee Q) \wedge \neg Q) \vee \neg P \\
&\underset{T}{\models} \neg(\neg P \vee Q) \vee Q \vee \neg P \\
&\underset{T}{\models} \neg(\neg P \vee Q) \vee (\neg P \vee Q) \\
&\underset{T}{\models} \mathbf{1} \\
&\therefore \models ((P \Rightarrow Q) \wedge \neg Q) \Rightarrow \neg P.
\end{aligned}
$$

Ahora, por el 2do,

$$
\begin{aligned}
((P \Rightarrow Q) \wedge \neg Q) \wedge \neg(\neg P) &\underset{T}{\models} ((\neg P \vee Q) \wedge \neg Q) \wedge P \\
&\underset{T}{\models} (\neg P \vee Q) \wedge (\neg Q \wedge P) \\
&\underset{T}{\models} (\neg P \wedge \neg Q \wedge P) \vee (Q \wedge \neg Q \wedge P) \\
&\underset{T}{\models} (\mathbf{0} \wedge \neg Q) \vee (\mathbf{0} \wedge P) \\
&\underset{T}{\models} \mathbf{0} \vee \mathbf{0} \underset{T}{\models} \mathbf{0} \\
&\therefore \models ((P \Rightarrow Q) \wedge \neg Q) \wedge P \text{ es insatisfacible.}
\end{aligned}
$$

c. Árboles semánticos

Los árboles semánticos constituyen un método para determinar si un conjunto de enunciados de un lenguaje proposicional es satisfacible o no.[5]

Supongamos que tenemos un conjunto de enunciados Γ y que queremos ver si es satisfacible o no. Para probar que es satisfacible tenemos que exhibir una situación posible en la que todos los enunciados de Γ sean verdaderos. Trataremos de describir esta situación utilizando enunciados tan pequeños como sea posible. Un primer intento para describir esta situación es Γ mismo, lo escribimos y así empieza nuestro árbol.

[5]Los árboles semánticos podemos atribuirlos a Smullyan. Los denominaba *tableaux analíticos* y con ellos construyó un cálculo tipo *deducción natural* [Sm].

A continuación seleccionamos algún enunciado de Γ, digamos Y, y tratamos de describir alguna situación en la que Y sea verdadero. Si, por ejemplo, descubrimos que Y es verdadero cuando otros dos enunciados, digamos Q y R son verdaderos, entonces debajo de Γ escribimos Q y R. Nuestro árbol en este caso se vería como la Fig. 4.8a). Si en cambio descubrimos que Y es verdadero precisamente en el caso en que alguno de dos enunciados, digamos Q y R sean verdaderos, entonces escribimos Q y R debajo de Γ, pero en diferentes ramas, ya que cada una representa una situación posible distinta. Nuestro árbol en este caso sería la Fig. 4.8b).

Figura 4.8

Después continuamos la operación con otro enunciado de Γ, haciendo lo mismo hasta que no podamos continuar. Nuestro árbol se podría ver en la figura 4.9.

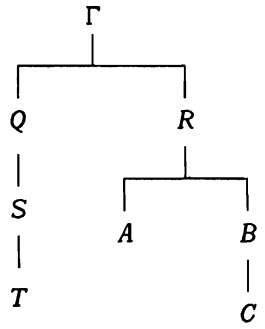


Figura 4.9

Cada rama representa una situación posible, los enunciados son tan pequeños que dentro de una misma rama es fácil verificar si hay inconsistencias, pues éstas siempre se presentarán cuando en la misma rama aparezcan enunciados de forma A y $\neg A$. Cuando esto ocurra dibujaremos una línea horizontal al final de la rama para indicar que esa posibilidad está cerrada. Si al terminar el árbol

queda alguna rama abierta esto indicará que existe esa posibilidad y que en esa situación todos los enunciados del conjunto original son verdaderos. Con esto quedará probada la satisfacibilidad del conjunto. Si, por otro lado, todas las ramas quedan cerradas, esto indicará que no hay ninguna situación en la que todos los enunciados del conjunto original sean verdaderos. Esto demostrará que el conjunto es insatisfacible.

Ejemplo. Determinemos si el conjunto de enunciados siguiente es satisfacible o no: $\Gamma = \{P \vee Q, R \Rightarrow P, Q \Rightarrow R\}$

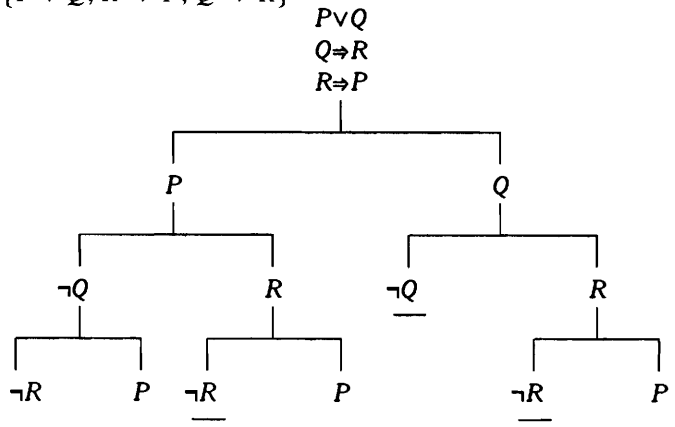


Figura 4.10

En este árbol se escribieron en primer lugar los tres enunciados del conjunto cuya satisfacibilidad se desea verificar. A continuación se abrieron dos ramas que corresponden a las dos posibilidades para que el primer enunciado de la lista sea verdadero. El siguiente nivel se obtuvo considerando las dos posibilidades para que el segundo enunciado de la lista sea verdadero. Aquí tuvimos que cerrar una rama, que contenía a los enunciados Q y $\neg Q$. El tercer nivel se obtuvo considerando las posibilidades para que el tercer enunciado del conjunto fuera verdadero. Aquí también tuvimos que cerrar dos ramas, que contenían a los enunciados R y $\neg R$. Cualquiera de las ramas abiertas define una situación en la que los tres enunciados del conjunto son verdaderos, por tanto el conjunto es satisfacible. ■

El método de árboles semánticos también puede ser utilizado para verificar si una fórmula bien formada de algún lenguaje proposicional es una tautología o no.

Si se tiene una fórmula ϕ y se desea verificar si es tautología o no basta construir un árbol semántico con la fórmula $\neg\phi$ en la parte superior. Si se cierran todas las ramas quiere decir que no existe situación posible en la que $\neg\phi$ sea verdadera y por lo tanto ϕ es una tautología. Si alguna rama queda abierta entonces es posible que $\neg\phi$ sea verdadera, en consecuencia ϕ no es una tautología.

Si recordamos la definición que se dio de argumento correcto, vemos que un argumento es correcto precisamente cuando su conjunto contraejemplo es insatisfacible. Este resultado fue presentado como el Teorema 4.14. De esta manera, el método de árboles semánticos puede ser utilizado también para verificar si una fórmula α es consecuencia tautológica de un conjunto Γ de fórmulas o no. Se inicia el árbol listando todos los elementos de Γ seguidos de $\neg\alpha$. Si quedan ramas abiertas entonces α no es consecuencia tautológica de Γ y si todas se cierran entonces α es consecuencia tautológica de Γ.

Observaciones:

1) Ver a la lógica como el estudio de las argumentaciones correctas o como el de satisfacibilidad de conjuntos de enunciados son, pues, dos enfoques equivalentes.

2) En virtud de que el conjunto $\{\neg, \vee, \wedge\}$ es un conjunto funcionalmente completo de conectivos, siempre se puede construir el árbol semántico de cualquier conjunto de fórmulas.

Debe notarse al construir árboles semánticos, que se pueden dar reglas para su construcción. Estas reglas pueden ser ejecutadas de manera mecánica y nos proporcionan un algoritmo para determinar si una fbf ϕ es tautología o no. No todos los sistemas formales tienen esta propiedad, en capítulos posteriores estudiaremos un sistema formal para la lógica para el cual no existe ningún algoritmo que decida en un número finito de pasos si una fórmula del lenguaje es lógicamente verdadera o no. Así, en la figura 4.11 damos reglas para la construcción de árboles semánticos. Se puede ver, analizando las tablas de verdad de las fórmulas involucradas, que en efecto satisfacen las condiciones dadas al principio de la sección para la construcción de los árboles semánticos. Aquí, ϕ y ψ representan fórmulas bien formadas arbitrarias.

Ejemplo. Determinemos si $\Gamma \models_T \alpha$ para $\Gamma = \{P \Rightarrow \neg Q, \neg(R \wedge T) \Rightarrow Q, P, (R \wedge T) \Rightarrow \neg S\}$ y $\alpha = \neg S$

De la figura 4.12, como todas las ramas se cerraron, concluimos que $\Gamma \models_T \alpha$. □

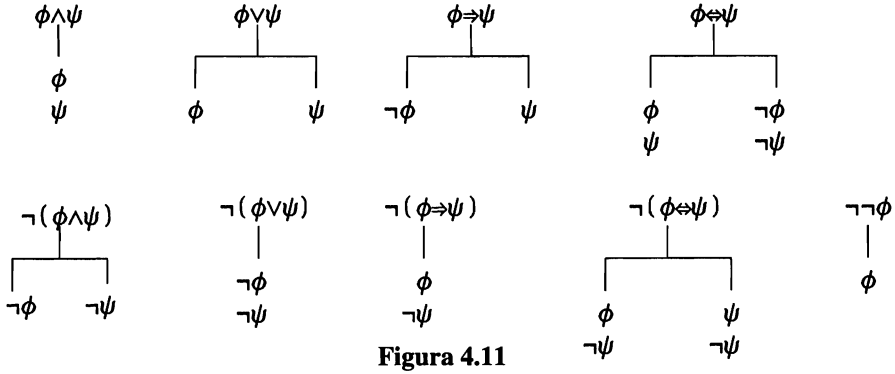


Figura 4.11

Sugerencias para reducir el tamaño de un árbol semántico:

1) Introduzca de primera instancia $\neg\alpha$. Porque no necesariamente todas las premisas en Γ son requeridas para implicar α.

2) Procure introducir justo después de $\neg\alpha$ (de ser posible) todas las premisas que no den lugar a ramas diferentes (*e.g.*, conjunciones, dobles negaciones, etc.), pues así disminuimos el número de situaciones a considerar en los pasos subsiguientes.

3) De las premisas en Γ, elija aquéllas que compartan letras con α, caso de que sean negaciones unas de otras. Así cerramos algunas ramas alternas.

Ejercicios

1. Muestre que $\Gamma \vDash_T \alpha$ en cada caso, empleando las técnicas recién presentadas:

i. $\Gamma = \{P \Rightarrow Q, Q \Rightarrow R\}$, $\alpha = P \Rightarrow R$
ii. $\Gamma = \{P \Leftrightarrow Q, Q \Rightarrow \neg R, R\}$, $\alpha = \neg P$
iii. $\Gamma = \{P, Q \vee R, \neg\neg R \wedge P\}$, $\alpha = (P \wedge Q) \vee (P \wedge R)$
iv. $\Gamma = \{P \Rightarrow Q, R \Rightarrow S, \neg Q \vee \neg S\}$, $\alpha = \neg P \vee \neg R$
v. $\Gamma = \{P \Rightarrow Q, R \Rightarrow S, P \vee R\}$, $\alpha = Q \vee S$
vi. $\Gamma = \{P \Rightarrow (R \wedge S), (Q \Rightarrow R) \Rightarrow P, R \wedge Q\}$, $\alpha = Q \Rightarrow P$
vii. $\Gamma = \{P \Rightarrow \neg R, (S \wedge T) \Rightarrow R, \neg S \Rightarrow Q, \neg(P \Rightarrow Q)\}$, $\alpha = \neg T$

2. Halle una interpretación que invalide que $\Gamma \vDash_T \alpha$, si:

i. $\Gamma = \{(P \wedge Q) \Rightarrow R, \neg R \wedge S, \neg Q\}$, $\alpha = \neg P$
ii. $\Gamma = \{P \Rightarrow \neg Q, Q \Rightarrow R, R \Rightarrow \neg S\}$, $\alpha = \neg S \vee \neg P$

Ejemplo.- Determinemos si $\Gamma \models_T \alpha$ para $\Gamma = \{P \Rightarrow \neg Q, \neg(R \wedge T) \Rightarrow Q, P, (R \wedge T) \Rightarrow \neg S\}$ y $\alpha = \neg S$

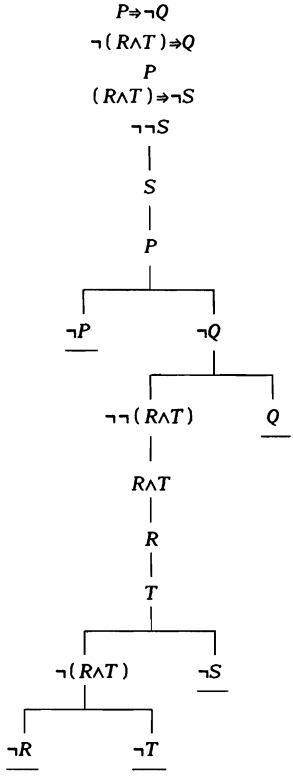


Figura 4.12

Como todas las ramas se cerraron, concluímos que $\Gamma \models_T \alpha$. □

Capítulo 5
Lógica proposicional: enfoque sintáctico

A primera vista puede pensarse que ... es posible afirmar Q siempre que P sea verdadero y que implique Q. Pero ese enigma planteado en "Lo que la tortuga dijo a Aquiles"[1] *muestra que no es así ... Necesitamos en realidad la noción de* **por tanto,** *que es muy diferente de la noción de* **implica,** *y que vale entre entidades diferentes.*

Bertrand Russell

5.1 Introducción

En el capítulo anterior vimos que el estudio de la argumentación correcta en el lenguaje de proposiciones se puede reducir al estudio de las tautologías, ya que un argumento con premisas $P_1, P_2, \ldots, P_n$ y con conclusión C es correcto si y sólo si la fórmula $(P_1 \wedge P \wedge \ldots \wedge P_n) \Rightarrow C$ es una tautología.

En este capítulo definiremos un sistema formal, como los presentados en el Capítulo 3, que sirva como un modelo formal del pensamiento deductivo correcto. Escogeremos ciertas fórmulas bien formadas del lenguaje proposicional para que

[1] Puede consultarse una reedición de ese texto debido a Lewis Carroll en [Ho], p. 51.

sean axiomas de nuestro sistema, daremos una regla de inferencia y una definición rigurosa de lo que quiere decir demostración y teorema. Posteriormente se probará que este sistema es adecuado para rescatar todas las tautologías del lenguaje.

En las secciones 5.4-5, daremos otro sistema formal, que no es una teoría formal en sentido estricto, pero que tiene la ventaja de ser mucho más manejable que la primera, es más fácil demostrar teoremas en ella.

5.2 Una teoría del cálculo proposicional

La teoría formal que veremos en esta sección se debe a Elliot Mendelson [Me].

Sea $\mathscr{L}$ el lenguaje formal siguiente:

1. Los símbolos de $\mathscr{L}$ son $\neg$, $\Rightarrow$, (,) y las letras P_i donde i es un número natural: $P_1, P_2, P_3, \ldots$ Los símbolos $\neg$ y $\Rightarrow$ son los conectivos, las letras P_i son las letras proposicionales y los paréntesis son símbolos de puntuación.

2. Las reglas de formación para las fórmulas bien formadas en $\mathscr{L}$ son las siguientes:

- **i.** Toda letra proposicional es una fórmula bien formada.
- **ii.** Si α y β son fórmulas bien formadas arbitrarias, entonces también lo son las expresiones $(\neg\alpha)$ y $(\alpha \Rightarrow \beta)$.
- **iii.** Una expresión es fórmula bien formada si y sólo si se puede generar a partir de letras proposicionales aplicando (ii).

Al igual que en el capítulo anterior, eliminaremos paréntesis cuando esto no dé lugar a confusión.

En $\mathscr{L}$ definimos una teoría formal, llamada el cálculo proposicional o de enunciados, y a la que denotaremos CE.

Axiomas de CE

Si α, β y γ son fórmulas de $\mathscr{L}$ entonces las fórmulas siguientes de $\mathscr{L}$ son axiomas de CE:

$$\begin{aligned} &\text{A1} \quad \alpha \Rightarrow (\beta \Rightarrow \alpha) \\ &\text{A2} \quad (\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)) \\ &\text{A3} \quad (\neg\beta \Rightarrow \neg\alpha) \Rightarrow ((\neg\beta \Rightarrow \alpha) \Rightarrow \beta) \end{aligned}$$

Regla de inferencia de CE

La única regla de inferencia es el *modus ponens* (MP):

$$\beta \text{ es consecuencia de } \alpha \text{ y } \alpha \Rightarrow \beta.$$

Nótese que la teoría CE tiene una infinidad de axiomas, cada vez que se sustituyan α, β o γ por fórmulas de $\mathscr{L}$ específicas en A1, A2 o A3 se obtendrán axiomas. A1–A3 son, pues, *esquemas axiomáticos.*

> **Definición.** Una *demostración* o *prueba* en el CE es una lista finita de fórmulas de $\mathscr{L}$ cada una de las cuales es un axioma de CE o es consecuencia de anteriores por MP. La fórmula que aparece al final de una demostración en el CE es un *teorema* de CE.

Observación. En un sentido estricto deberíamos denotar el hecho de que ϕ es un teorema de CE por medio de $\vdash_{\text{CE}} \phi$, sin embargo, para simplificar la notación, escribiremos simplemente $\vdash \phi$, sobreentendiendo el sistema formal en el que se está trabajando.[2]

Aquí puede parecer que surge cierta ambigüedad cuando usamos por un lado la palabra "teorema" para establecer propiedades sobre el sistema formal (*i.e.*, en el metalenguaje), y por otro, para designar a las fórmulas derivadas en él (dentro del lenguaje). Esto en realidad no ocurre, pues un teorema es una fbf del lenguaje y un metateorema se enuncia en español.

> **Definición.** Si Γ es un conjunto de fórmulas de $\mathscr{L}$ y ϕ es una fórmula de $\mathscr{L}$, decimos que ϕ es *demostrable* o *derivable* en CE a *partir de* Γ si y sólo si existe una sucesión finita $\alpha_1, \alpha_2, \ldots, \alpha_n$ de fórmulas de $\mathscr{L}$ tales que $\alpha_n = \phi$ y para cada i, α_i es un axioma de CE, o es un miembro de Γ o es consecuencia de anteriores por MP.

Cuando ϕ es demostrable en el CE a partir de Γ, denotamos este hecho por medio de $\Gamma \vdash \phi$. En particular, si $\Gamma = \{\alpha_1, \ldots, \alpha_n\}$, escribimos $\alpha_1, \ldots, \alpha_n \vdash \phi$ en lugar de $\{\alpha_1, \ldots, \alpha_n\} \vdash \phi$. Nótese que $\varnothing \vdash \phi$ si y sólo si ϕ es un teorema del CE.

[2]Esta misma convención la aplicaremos a los demás sistemas que consideremos, explicitando sólo cuando se cambie de sistema formal.

Teorema 5.1. *Si* $\Delta \vdash \phi$ *y para cada* $\alpha \in \Delta$, $\Gamma \vdash \alpha$ *entonces* $\Gamma \vdash \phi$.

Demostración.

En la demostración de ϕ a partir de Δ sustitúyase cada ocurrencia de los elementos de Δ por su demostración a partir de Γ. El resultado es una demostración de ϕ a partir de Γ. ■

Antes de probar algunos teoremas de esta teoría observemos que en virtud de los resultados sobre conjuntos completos de conectivos (sección 4.6), no se está perdiendo generalidad al considerar un lenguaje proposicional con $\neg$ y $\Rightarrow$ como únicos conectivos. Podemos introducir los otros conectivos por medio de las siguientes definiciones:

$$(\alpha \wedge \beta) \text{ es una abreviación de } \neg(\alpha \Rightarrow \neg\beta)$$
$$(\alpha \vee \beta) \text{ es una abreviación de } \neg\alpha \Rightarrow \beta$$
$$(\alpha \Leftrightarrow \beta) \text{ es una abreviación de } (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$

Teorema 5.2. *Para toda fórmula* ϕ, $\phi \Rightarrow \phi$.[3]

Demostración.

Damos a continuación una demostración de $\phi \Rightarrow \phi$ en CE. A la derecha de cada fórmula que aparezca en la demostración escribiremos la razón por la que tiene derecho a aparecer en ella.

(1) $(\phi \Rightarrow ((\phi \Rightarrow \phi) \Rightarrow \phi)) \Rightarrow ((\phi \Rightarrow (\phi \Rightarrow \phi)) \Rightarrow (\phi \Rightarrow \phi))$	A2
(2) $\phi \Rightarrow ((\phi \Rightarrow \phi) \Rightarrow \phi)$	A1
(3) $(\phi \Rightarrow (\phi \Rightarrow \phi)) \Rightarrow (\phi \Rightarrow \phi)$	1, 2 MP
(4) $\phi \Rightarrow (\phi \Rightarrow \phi)$	A1
(5) $\phi \Rightarrow \phi$	3,4 MP

■

Teorema 5.3. *Para toda fórmula* ϕ, $(\neg\phi \Rightarrow \phi) \Rightarrow \phi$.

[3] De aquí en adelante, como sólo trabajaremos con el lenguaje $\mathscr{L}$, cada vez que se hable de fórmulas se entenderá fórmulas de $\mathscr{L}$.

Demostración.

La siguiente lista de fórmulas es una demostración de $(\neg\phi \Rightarrow \phi) \Rightarrow \phi$ en CE.

(1) $(\neg\phi \Rightarrow \neg\phi) \Rightarrow ((\neg\phi \Rightarrow \phi) \Rightarrow \phi)$	A3
(2) $\neg\phi \Rightarrow \neg\phi$	Teorema 5.2
(3) $(\neg\phi \Rightarrow \phi) \Rightarrow \phi$	1, 2 MP

■

Observación. Aquí es conveniente remarcar que la lista dada no es, en sentido estricto, una demostración de $(\neg\phi \Rightarrow \phi) \Rightarrow \phi$ en el CE, puesto que en el paso 2 se introdujo una fórmula que no es axioma ni consecuencia de anteriores por MP. Sin embargo esta lista se puede transformar en una demostración en el CE si sustituimos el paso 2 por una demostración de $\neg\phi \Rightarrow \neg\phi$ como la dada en el Teorema 5.2. (Usar el resultado de un teorema, previamente demostrado, en la prueba de un nuevo teorema puede considerarse, parafraseando la terminología computacional, como el *llamado de una subrutina* (el teorema previo) dentro de un *programa principal* (la prueba del teorema en curso).)

Teorema 5.4. *Si α, β y γ son tres fórmulas arbitrarias entonces $\alpha \Rightarrow \beta$, $\beta \Rightarrow \gamma \vdash \alpha \Rightarrow \gamma$.*

Demostración.

(1) $\alpha \Rightarrow \beta$	hipótesis
(2) $\beta \Rightarrow \gamma$	hipótesis
(3) $(\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow (\beta \Rightarrow \gamma))$	A1
(4) $\alpha \Rightarrow (\beta \Rightarrow \gamma)$	2, 3 MP
(5) $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$	A2
(6) $(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$	4, 5 MP
(7) $\alpha \Rightarrow \gamma$	1, 6 MP

■

Cuando uno se enfrenta a la tarea de demostrar que la fórmula $\alpha \Rightarrow \beta$ es un teorema del CE es muy tentador suponer α y a partir de esta suposición probar β. Esto es lo que se hace en matemáticas y cuando se prueban metateoremas sobre teorías formales. Las reglas de CE no permiten hacer esto de manera directa, pero el siguiente teorema, probado por Herbrand en 1930, justifica este procedimiento dentro de la teoría CE.

Teorema 5.5 (de la Deducción). *Sea* Γ *un conjunto de fórmulas de* $\mathscr{L}$, α *y* β *fórmulas de* $\mathscr{L}$ *y supóngase que* $\Gamma, \alpha \vdash \beta$. *Entonces* $\Gamma \vdash \alpha \Rightarrow \beta$.

Demostración.

Si $\Gamma, \alpha \vdash \beta$ entonces existe una demostración de β a partir de $\Gamma \cup \{\alpha\}$, digamos que $\beta_1, \beta_2, \ldots, \beta_n$, con $\beta_n = \beta$.

Por inducción sobre i probaremos que para toda $i \in \{1, \ldots, n\}$ se tiene $\Gamma \vdash \alpha \Rightarrow \beta_i$. Si $i = 1$ entonces hay tres posibilidades para β_1: β_1 es axioma de CE o está en Γ o es igual a α.

Caso 1. β_1 es axioma de CE.

Considérese la siguiente lista de fórmulas:

(1) β_1	axioma
(2) $\beta_1 \Rightarrow (\alpha \Rightarrow \beta_1)$	A1
(3) $\alpha \Rightarrow \beta_1$	1, 2 MP

Esta es una demostración de $\alpha \Rightarrow \beta_1$ en el CE y por tanto $\Gamma \vdash \alpha \Rightarrow \beta_1$.

Caso 2. β_1 está en Γ.

La misma demostración del caso anterior sirve, sólo que la justificación en el primer paso es que $\beta_1 \in \Gamma$.

Caso 3. $\beta_1 = \alpha$

En este caso el Teorema 5.2 nos asegura que $\vdash \alpha \Rightarrow \alpha$ y por lo tanto $\Gamma \vdash \alpha \Rightarrow \beta_1$.

Supongamos ahora que $\Gamma \vdash \alpha \Rightarrow \beta_k$ para toda $k < i$ y consideremos β_i.

Si β_i es axioma o está en Γ o es α se procede de la misma manera que en el caso $i = 1$. El único caso que falta por considerar es cuando β es consecuencia de fórmulas anteriores por MP. En este caso existen dos fórmulas anteriores a β_i, β_j y β_m, con $j, m < i$ y tales que β_m es de la forma $\beta_j \Rightarrow \beta_i$. Por hipótesis de inducción tenemos que $\Gamma \vdash \alpha \Rightarrow \beta_j$ y $\Gamma \vdash \alpha \Rightarrow (\beta_j \Rightarrow \beta)$. Concatenando las demostraciones a partir de Γ de estas dos fórmulas obtenemos una deducción en el CE a partir de Γ, que podemos completar de la siguiente forma:

$\ldots$	
(p)	$\alpha \Rightarrow \beta_j$
$\ldots$	
(q)	$\alpha \Rightarrow (\beta_j \Rightarrow \beta_i)$

(q+1) $(\alpha \Rightarrow (\beta_j \Rightarrow \beta_i)) \Rightarrow ((\alpha \Rightarrow \beta_j) \Rightarrow (\alpha \Rightarrow \beta_i))$	A2
(q+2) $(\alpha \Rightarrow \beta_j) \Rightarrow (\alpha \Rightarrow \beta_i)$	q, q + 1 MP
(q+3) $\alpha \Rightarrow \beta_i$	p, q + 2 MP

Por lo tanto $\Gamma \vdash \alpha \Rightarrow \beta_i$.

Con esto queda completa la inducción y tomando $i = n$ se obtiene la conclusión del teorema. ■

Analizando la demostración del Teorema de la Deducción se puede ver que ésta nos proporciona un método para construir una deducción de $\alpha \Rightarrow \beta$ a partir de Γ en el CE basándose en una deducción dada de β a partir de $\Gamma \cup \{\alpha\}$.

También se puede ver que los únicos axiomas que se usaron para demostrar el Teorema de la Deducción son A1 y A2. El Teorema de la Deducción es una consecuencia del significado que tiene el condicional; este significado ha sido determinado, desde el punto de vista semántico, por la tabla de verdad del condicional, y desde el punto de vista sintáctico, por los Axiomas A1 y A2. El Axioma A3 sirve para determinar el comportamiento de la negación en este sistema formal.

El Teorema de la Deducción es muy útil porque reduce el trabajo para demostrar teoremas dentro del CE. Como ejemplo veamos una prueba del Teorema 5.4 usando el Teorema de la Deducción:

(1) $\alpha \Rightarrow \beta$	hipótesis
(2) $\beta \Rightarrow \gamma$	hipótesis
(3) α	hipótesis
(4) β	1, 3 MP
(5) γ	2, 4 MP

Por tanto, $\alpha \Rightarrow \beta, \beta \Rightarrow \gamma, \alpha \vdash \gamma$, y aplicando el Teorema de la Deducción se obtiene que $\alpha \Rightarrow \beta, \beta \Rightarrow \gamma \vdash \alpha \Rightarrow \gamma$.

Teorema 5.6. *Para cualesquiera fórmulas α y β, las siguientes son teoremas del CE:*

(a) $\neg\neg\alpha \Rightarrow \alpha$

(b) $\alpha \Rightarrow \neg\neg\alpha$

(c) $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$

(d) $(\neg\beta \Rightarrow \neg\alpha) \Rightarrow (\alpha \Rightarrow \beta)$

(e) $(\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$

(f) $\alpha \Rightarrow (\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta))$

(g) $(\alpha \Rightarrow \beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \beta)$

Demostraciones:

(a) $\neg\neg\alpha \Rightarrow \alpha$

(1)	$(\neg\alpha \Rightarrow \neg\neg\alpha) \Rightarrow ((\neg\alpha \Rightarrow \neg\alpha) \Rightarrow \alpha)$	A3
(2)	$\neg\alpha \Rightarrow \neg\alpha$	Teorema 5.2
(3)	$(\neg\alpha \Rightarrow \neg\neg\alpha) \Rightarrow \alpha$	1, 2 Ejercicio 4, al final de esta sección
(4)	$\neg\neg\alpha \Rightarrow (\neg\alpha \Rightarrow \neg\neg\alpha)$	A1
(5)	$\neg\neg\alpha \Rightarrow \alpha$	3, 4 Teorema 5.4

(b) $\alpha \Rightarrow \neg\neg\alpha$

(1)	$(\neg\neg\neg\alpha \Rightarrow \neg\alpha) \Rightarrow ((\neg\neg\neg\alpha \Rightarrow \alpha) \Rightarrow \neg\neg\alpha))$	A3
(2)	$\neg\neg\neg\alpha \Rightarrow \neg\alpha$	inciso anterior
(3)	$(\neg\neg\neg\alpha \Rightarrow \alpha) \Rightarrow \neg\neg\alpha$	1, 2 MP
(4)	$\alpha \Rightarrow (\neg\neg\neg\alpha \Rightarrow \alpha)$	A1
(5)	$\alpha \Rightarrow \neg\neg\alpha$	3, 4 Teorema 5.4

(c) $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$

(1)	$\neg\alpha$	hipótesis
(2)	α	hipótesis
(3)	$\alpha \Rightarrow (\neg\beta \Rightarrow \alpha)$	A1
(4)	$\neg\alpha \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$	A1
(5)	$\neg\beta \Rightarrow \alpha$	2, 3 MP
(6)	$\neg\beta \Rightarrow \neg\alpha$	1, 4 MP
(7)	$(\neg\beta \Rightarrow \neg\alpha) \Rightarrow ((\neg\beta \Rightarrow \alpha) \Rightarrow \beta)$	A3
(8)	$(\neg\beta \Rightarrow \alpha) \Rightarrow \beta$	6, 7 MP
(9)	β	5, 8 MP

Por tanto $\neg\alpha, \alpha \vdash \beta$; aplicando el teorema de la deducción, $\neg\alpha \vdash \alpha \Rightarrow \beta$. Con una segunda aplicación del Teorema de la Deducción se obtiene el resultado.

(d) $(\neg\beta \Rightarrow \neg\alpha) \Rightarrow (\alpha \Rightarrow \beta)$

(1)	$\neg\beta \Rightarrow \neg\alpha$	hipótesis

(2) α	hipótesis
(3) $(\neg\beta \Rightarrow \neg\alpha) \Rightarrow ((\neg\beta \Rightarrow \alpha) \Rightarrow \beta)$	A3
(4) $\alpha \Rightarrow (\neg\beta \Rightarrow \alpha)$	A1
(5) $\neg\beta \Rightarrow \alpha$	2, 4 MP
(6) $(\neg\beta \Rightarrow \alpha) \Rightarrow \beta$	1, 3 MP
(7) β	5, 6 MP

Hemos probado que $\neg\beta \Rightarrow \neg\alpha, \alpha \vdash \beta$. Aplicando el Teorema de la Deducción dos veces obtenemos el resultado deseado.

(e) $(\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$

(1) $\alpha \Rightarrow \beta$	hipótesis
(2) $\neg\neg\alpha \Rightarrow \alpha$	inciso (a)
(3) $\neg\neg\alpha \Rightarrow \beta$	1, 2 Teorema 5.4
(4) $\beta \Rightarrow \neg\neg\beta$	inciso (b)
(5) $\neg\neg\alpha \Rightarrow \neg\neg\beta$	3, 4 Teorema 5.4
(6) $(\neg\neg\alpha \Rightarrow \neg\neg\beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$	inciso (d)
(7) $\neg\beta \Rightarrow \neg\alpha$	5, 6 MP

Por tanto $\alpha \Rightarrow \beta \vdash \neg\beta \Rightarrow \neg\alpha$, aplicando el Teorema de la Deducción obtenemos (e).

(f) $\alpha \Rightarrow (\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta))$

(1) α	hipótesis
(2) $\alpha \Rightarrow \beta$	hipótesis
(3) β	1, 2 MP

Por tanto $\alpha, \alpha \Rightarrow \beta \vdash \beta$. Usando el Teorema de la Deducción dos veces obtenemos que $\vdash \alpha \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow \beta)$. Por el inciso (e) se obtiene $\vdash ((\alpha \Rightarrow \beta) \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta))$. Aplicando el Teorema 5.4 a estas dos últimas afirmaciones se obtiene el resultado deseado.

(g) $\vdash (\alpha \Rightarrow \beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \beta)$

(1) $\alpha \Rightarrow \beta$	hipótesis
(2) $\neg\alpha \Rightarrow \beta$	hipótesis
(3) $(\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$	inciso (e)
(4) $\neg\beta \Rightarrow \neg\alpha$	1, 3 MP

(5) $(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\neg\alpha)$	inciso (e)
(6) $\neg\beta \Rightarrow \neg\neg\alpha$	2, 5 MP
(7) $(\neg\beta \Rightarrow \neg\neg\alpha) \Rightarrow ((\neg\beta \Rightarrow \neg\alpha) \Rightarrow \beta)$	A3
(8) $(\neg\beta \Rightarrow \neg\alpha) \Rightarrow \beta$	6, 7 MP
(9) β	4, 8 MP

Con esta lista, junto con dos aplicaciones del Teorema de la Deducción, se obtiene (g). ■

Las demostraciones de los teoremas anteriores pueden dejar la impresión en el lector de haber sido "sacadas de la manga". Esto resulta así, debido a que al ser las más cortas que son directamente derivables de los axiomas y MP (con uso de pocos teoremas previos), se pierde la "intuición" con las que se hicieron.

A continuación discutiremos las pruebas de tres teoremas con la finalidad de proporcionar sugerencias para guiar la "intuición" y dar así cierta habilidad en la demostración de teoremas en CE.

Ejemplos:

Demostremos formalmente las leyes siguientes:

(a) *Ley de Pierce*: $(\alpha \Rightarrow \beta) \Rightarrow \alpha \vdash \alpha$

Primera versión.

(1) $(\alpha \Rightarrow \beta) \Rightarrow \alpha$	hipótesis
(2) $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$	Teorema 5.6(c)
(3) $\neg\alpha \Rightarrow \alpha$	1, 2 Teorema 5.4
(4) $\alpha \Rightarrow \neg\neg\alpha$	Teorema 5.6(b)
(5) $\neg\alpha \Rightarrow \neg\neg\alpha$	3,4 Teorema 5.4
(6) $(\neg\alpha \Rightarrow \neg\neg\alpha) \Rightarrow ((\neg\alpha \Rightarrow \neg\alpha) \Rightarrow \alpha)$	A3
(7) $(\neg\alpha \Rightarrow \neg\alpha) \Rightarrow \alpha$	5, 6 MP
(8) $\neg\alpha \Rightarrow \neg\alpha$	Teorema 5.2
(9) α	7, 8 MP

Sin embargo, una versión más corta sería:

Segunda versión.

(4) $(\alpha \Rightarrow \alpha) \Rightarrow ((\neg\alpha \Rightarrow \alpha) \Rightarrow \alpha)$	Teorema 5.6(g)
(5) $\alpha \Rightarrow \alpha$	Teorema 5.2

(6) $(\neg\alpha \Rightarrow \alpha) \Rightarrow \alpha$	4, 5 MP
(7) α	3, 6 MP

(b) *Ley distributiva*: $(\alpha \Rightarrow \beta) \vee (\alpha \Rightarrow \gamma) \vdash \alpha \Rightarrow (\beta \vee \gamma)$.

Recuerde que $\vee$ no es un símbolo del lenguaje $\mathscr{L}$, sino sólo se tiene en calidad de abreviatura, por lo que hay que "traducir" las fórmulas en términos de $\neg$ y $\Rightarrow$. Así, hemos de probar que: $\neg(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma) \vdash \alpha \Rightarrow (\neg\beta \Rightarrow \gamma)$

Primera versión.

(1) $\neg(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$	hipótesis
(2) α	hipótesis
(3) $\neg\beta$	hipótesis
(4) $\alpha \Rightarrow (\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta))$	Teorema 5.6(f)
(5) $\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta)$	2, 4 MP
(6) $\neg(\alpha \Rightarrow \beta)$	3, 6 MP
(7) $\alpha \Rightarrow \gamma$	1, 6 MP
(8) γ	2, 7 MP

El resultado se sigue de aplicar dos veces el Teorema de la Deducción.

Segunda versión.

(1) $\neg(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$	hipótesis
(2) α	hipótesis
(3) $\alpha \Rightarrow (\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta))$	Teorema 5.6(f)
(4) $\neg\beta \Rightarrow \neg(\alpha \Rightarrow \beta)$	2, 3 MP
(5) $\neg\beta \Rightarrow (\alpha \Rightarrow \gamma)$	1,5 Teorema 5.4
(6) $\neg\beta \Rightarrow \gamma$	2,5 Ejercicio 4, esta sección

Una aplicación del Teorema de la Deducción nos proporciona el resultado deseado.

(c) *Ley de exportación*: $(\alpha \wedge \beta) \Rightarrow \gamma \vdash \alpha \Rightarrow (\beta \Rightarrow \gamma)$

De nueva cuenta, $\wedge$ no es del lenguaje; "traduciendo" vamos a probar que: $\neg(\alpha \Rightarrow \neg\beta) \Rightarrow \gamma \vdash \alpha \Rightarrow (\beta \Rightarrow \gamma)$

Primera versión.

(1) $\neg(\alpha \Rightarrow \neg\beta) \Rightarrow \gamma$	hipótesis

(2) α	hipótesis
(3) $\alpha \Rightarrow (\neg\neg\beta \Rightarrow \neg(\alpha \Rightarrow \neg\beta))$	Teorema 5.6(f)
(4) $\neg\neg\beta \Rightarrow \neg(\alpha \Rightarrow \neg\beta)$	2, 3 MP
(5) $\neg\neg\beta \Rightarrow \gamma$	1, 4 Teorema 5.4
(6) $\beta \Rightarrow \neg\neg\beta$	Teorema 5.6(b)
(7) $\beta \Rightarrow \gamma$	5, 6 Teorema 5.4

Así, como $\neg(\alpha \Rightarrow \neg\beta) \Rightarrow \gamma, \alpha \vdash (\beta \Rightarrow \gamma)$, aplicando una vez el Teorema de la Deducción, se obtiene la prueba.

A continuación, y "traduciendo" a los conectivos $\neg$ y $\Rightarrow$, probaremos que:

$$\neg(\alpha \Rightarrow \neg\beta) \Rightarrow \gamma, \alpha \vdash \beta \Rightarrow \gamma$$

Segunda versión.

(1) $\neg(\alpha \Rightarrow \neg\beta) \Rightarrow \gamma$	hipótesis
(2) α	hipótesis
(3) $\alpha \Rightarrow (\beta \Rightarrow \neg(\alpha \Rightarrow \neg\beta)$	Ejercicio 6(b), esta secc.
(4) $\beta \Rightarrow \neg(\alpha \Rightarrow \neg\beta)$	2, 3 MP
(5) $\beta \Rightarrow \gamma$	1, 4 Teorema 5.4

Aplicando una vez el Teorema de la Deducción, el resultado se sigue. ■

Al comparar las distintas versiones de las demostraciones de los teoremas dados, podemos extraer las sugerencias siguientes:

1. Como la única regla de inferencia que disponemos es MP, procure reconstruir "en sentido contrario" parte de la prueba, observando que la conclusión, digamos β, proviene del consecuente de una implicación de la forma $\alpha \Rightarrow \beta$. Así, la consigna es hallar un α idóneo que guarde cierta conexión con las premisas.

2. Mientras más premisas se disponga, es "heurísticamente" más fácil hacer la demostración. Sin embargo, esto puede redundar en una mayor longitud de la prueba, *e.g.*, la primera versión para el ejemplo (b).

3. Demasiadas aplicaciones del MP, así como el uso de instancias de los axiomas pueden incidir sobre la extensión de una demostración. Confronte la segunda versión de prueba para el ejemplo (a).

4. Por el contrario, la utilización adecuada de instancias de teoremas ya probados suele "agilizar" el desarrollo de una prueba. Las segundas versiones

de las demostraciones presentadas parecen ser más "elegantes" respecto de las primeras.

5. Los Teoremas 5.6(f) y 5.6(g) pueden ser particularmente útiles para las pruebas de fórmulas abreviantes que involucren a los conectivos $\wedge$ y $\vee$.

Observaciones:

(i) Las sugerencias expuestas tienen un carácter más bien *heurístico* que *algorítmico* (procedimiento mecánico), pues la intuición para saber qué instancia de axioma o de teorema es la adecuada en un paso determinado de una prueba no es aprehensible por regla (o receta) alguna; depende de la habilidad de quien lleve a efecto la prueba.

(ii) De manera implícita se ha venido manejando el converso del Teorema de la Deducción (Ejercicio 5), al considerar que si se tiene $\Gamma \vdash \alpha \Rightarrow \beta$, podemos contar con que $\Gamma, \alpha \vdash \beta$, *i.e.*, disponemos de tantas premisas como sean necesarias, obtenidas del (los) antecedente(s) α de la conclusión $\alpha \Rightarrow \beta$.

(iii) La moraleja que podemos tener de la sugerencia (4) es que si disponemos de suficientes teorema previos, éstos aunados al teorema de la deducción hacen las veces de "nuevas reglas de inferencia", facilitando así la prueba que esté en turno. Este punto es incluso implementable, tal y como se hará en la sección 5.4.

Ejercicios

1. Probar que si $\Delta \subseteq \Sigma$ y $\Delta \vdash \phi$ entonces $\Sigma \vdash \phi$.

2. Probar que $\Sigma \vdash \phi$ si y sólo si existe $\Delta \subseteq \Sigma$, Δ finito, tal que $\Delta \vdash \phi$.

3. Sean α, β y γ fórmulas arbitrarias de $\mathscr{L}$. Probar, sin usar el Teorema de la Deducción, que:

a. $\alpha \Rightarrow (\beta \Rightarrow \gamma) \vdash \beta \Rightarrow (\alpha \Rightarrow \gamma)$

b. $\vdash (\neg\beta \Rightarrow \neg\alpha) \Rightarrow (\alpha \Rightarrow \beta)$

4. Utilizar el teorema de la deducción para probar que $\alpha \Rightarrow (\beta \Rightarrow \gamma)$, $\beta \vdash \alpha \Rightarrow \gamma$.

5. Probar que si $\Gamma \vdash \alpha \Rightarrow \beta$, entonces $\Gamma, \alpha \vdash \beta$.

6. Demostrar que para α, β y γ fórmulas de $\mathscr{L}$, las siguientes fórmulas son teoremas del CE:

a. $((\alpha \Rightarrow \beta) \Rightarrow \alpha) \Rightarrow \alpha$
b. $\alpha \Rightarrow (\beta \Rightarrow (\alpha \wedge \beta))$
c. $(\alpha \wedge \beta) \Rightarrow \beta$
d. $\alpha \Rightarrow (\alpha \vee \beta)$
e. $\neg(\alpha \Rightarrow \beta) \Rightarrow \alpha$
f. $((\alpha \Rightarrow \beta) \wedge \neg\beta) \Rightarrow \neg\alpha$
g. $(\alpha \Rightarrow \beta) \Rightarrow (\neg\neg\alpha \Rightarrow \neg\neg\beta)$
h. $(\alpha \vee \beta) \Rightarrow (\beta \vee \alpha)$
i. $\neg(\alpha \wedge \beta) \Rightarrow (\neg\alpha \vee \neg\beta)$
j. $(\neg\alpha \vee \neg\beta) \Rightarrow \neg(\alpha \wedge \beta)$
k. $(\neg\alpha \wedge \neg\beta) \Rightarrow \neg(\alpha \vee \beta)$
l. $\neg(\alpha \vee \beta) \Rightarrow (\neg\alpha \wedge \neg\beta)$
m. $(\alpha \vee \beta) \Rightarrow ((\neg\beta \vee \gamma) \Rightarrow (\alpha \vee \gamma))$
n. $(\alpha \Rightarrow \beta) \Rightarrow ((\alpha \wedge \gamma) \Rightarrow (\beta \wedge \gamma))$
ñ. $((\alpha \Rightarrow \beta) \wedge (\alpha \Rightarrow \gamma)) \Rightarrow (\alpha \Rightarrow (\beta \wedge \gamma))$
o. $((\alpha \Rightarrow \gamma) \vee (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \wedge \beta) \Rightarrow \gamma)$
p. $((\alpha \vee \beta) \Rightarrow \gamma) \Rightarrow ((\alpha \Rightarrow \gamma) \wedge (\beta \Rightarrow \gamma))$
q. $((\alpha \vee \beta) \vee \gamma) \Rightarrow (\alpha \vee (\beta \vee \gamma))$

5.3 Validez y completud para CE

El propósito de esta sección es demostrar que la teoría CE es adecuada para rescatar a todas las tautologías del lenguaje proposicional en el cual está formulada. Probaremos que una fórmula de $\mathscr{L}$ es teorema del CE si y sólo si es una tautología.

Teorema 5.7 (de Validez de CE). *Todo teorema del CE es una tautología.*

Demostración.

Sea ϕ un teorema del CE, procedemos por inducción sobre la longitud de la prueba de ϕ en CE.

Si la demostración de ϕ tiene longitud 1 entonces ϕ es un axioma de CE. Utilizando cualquiera de los métodos semánticos del capítulo anterior es fácil verificar que los tres axiomas son tautologías.

Para el paso inductivo es suficiente con probar que la regla de inferencia MP preserva tautologías, lo cual es cierto en virtud del Teorema 4.6. ■

La demostración de la otra parte es más compleja y requiere de un lema preliminar.

Lema 5.8. *Sean ϕ una fórmula de $\mathscr{L}$ y $P_1, \ldots, P_n$ las letras proposicionales que aparecen en ϕ. Sea $|\cdot|$ una asignación de valores de verdad fija y arbitraria para las P_i. Para cada $i \in \{1, \ldots, n\}$ definimos la fórmula P_i' de la siguiente manera:*

$$P_i' = P_i \text{ si } |P_i| = 1 \quad y \quad P_i' = \neg P_i \text{ si } |P_i| = 0.$$
$$\text{Sea } \phi' = \phi \text{ si } |\phi| = 1 \quad y \quad \phi' = \neg\phi \text{ si } |\phi| = 0.$$

Entonces $P_1, \ldots, P_n \vdash \phi'$.

Demostración.

Por inducción en el número de ocurrencias de conectivos de ϕ. (Se supone que ϕ está escrita sin abreviaciones).

Si $n = 0$ entonces ϕ es una letra P_i. Si $|P_i| = 1$ entonces $\phi' = P_i' = P_i$. En este caso el lema se reduce a demostrar que $P_i \vdash P_i$. Si $|P_i| = 0$ entonces $\phi' = P_i' = \neg P_i$. En este caso el lema se reduce a demostrar que $\neg P_i \vdash \neg P_i$.

Supongamos ahora el lema verdadero para toda fórmula con menos de n conectivos. Sea ϕ una fórmula con n conectivos. Probaremos que el lema vale para ϕ.

Caso 1. ϕ es $\neg\chi$.

Subcaso 1a. $|\chi| = 1$. Entonces $|\phi| = 0$ y por lo tanto $\chi' = \chi$ y $\phi' = \neg\phi = \neg\neg\chi$. Por hipótesis de inducción se tiene que $P_1', \ldots, P_n' \vdash \chi'$, esto es, $P_1', \ldots, P_n' \vdash \chi$. Por el Teorema 6(a), tenemos que $\vdash \chi \Rightarrow \neg\neg\chi$, por lo tanto $P_1', \ldots, P_n' \vdash \phi'$.

Subcaso 1b. $|\chi| = 0$. Entonces $|\phi| = 1$ y por lo tanto $\chi' = \neg\chi$ y $\phi' = \phi = \neg\chi$. Por hipótesis de inducción $P_1', \ldots, P_n' \vdash \neg\chi$, esto es justamente lo que queremos.

Caso 2. ϕ es de la forma $\alpha \Rightarrow \beta$. En este caso tanto α como β tienen menos de n conectivos, por lo que la hipótesis de inducción garantiza que $P_1', \ldots, P_n' \vdash \alpha'$ y $P_1', \ldots, P_n' \vdash \beta'$.

Subcaso 2a. $|\beta| = 1$. Entonces $|\phi| = 1$ y por lo tanto $\beta' = \beta$ y $\phi' = \phi = \alpha \Rightarrow \beta$. Por hipótesis de inducción $P_1', \ldots, P_n' \vdash \beta$. Aplicando la instancia del axioma 1, $\beta \Rightarrow (\alpha \Rightarrow \beta)$, y MP se obtiene que $P_1', \ldots, P_n' \vdash \phi'$.

Subcaso 2b. $|\alpha| = 0$. Entonces $|\phi| = 1$ y por tanto $\alpha' = \neg\alpha$ y $\phi' = \phi = \alpha \Rightarrow \beta$. Por hipótesis de inducción $P_1', \ldots, P_n' \vdash \neg\alpha$. Aplicando el Teorema 5.6(c) y MP obtenemos el resultado.

Subcaso 2c. $|\alpha| = 1$ y $|\beta| = 0$. En este caso $|\phi| = 0$ y por tanto, $\alpha' = \alpha$, $\beta' = \neg\beta$ y $\phi' = \neg\phi = \neg(\alpha \Rightarrow \beta)$. La hipótesis de inducción nos garantiza que $P_1', \ldots, P_n' \vdash \alpha$ y que $P_1', \ldots, P_n' \vdash \neg\beta$. Aplicando el Teorema 5.6(f) y MP dos veces el resultado se sigue. ■

Ahora sí estamos en posición de demostrar el metateorema principal sobre el sistema CE. La demostración que presentamos a continuación se debe a Kalmár.

Teorema 5.9 (de Completud de CE). *Toda tautología en el lenguaje $\mathscr{L}$ de CE es un teorema de CE.*

Demostración.

Sea ϕ una tautología y sean $P_1, \ldots, P_n$ las letras proposicionales que aparecen en ϕ. Para cualquier asignación de verdad $|\cdot|$, el Lema 5.8 asegura que $P_1', \ldots, P_n' \vdash \phi$. (Sabemos que $\phi' = \phi$ siempre porque ϕ es una tautología.) Sea $|\cdot|$ una asignación tal que $|P_n| = 1$; por el lema 5.8 se tiene que $P_1', \ldots, P_n \vdash \phi$. Sea μ una asignación que coincide con $|\cdot|$, con la salvedad de que $\mu(P_n) = 0$; por el Lema 5.8 se tiene que $P_1', \ldots, \neg P_n \vdash \phi$. Aplicando el Teorema de la Deducción a estas dos pruebas en el CE obtenemos que $P_1', \ldots, P_{n-1}' \vdash P_n \Rightarrow \phi$ y $P_1', \ldots, P_{n-1}' \vdash \neg P_n \Rightarrow \phi$. Entonces, por el Teorema 5.6(g) de la sección anterior, tenemos que $P_1', \ldots, P_{n-1}' \vdash \phi$. De la misma manera podemos tomar otras dos asignaciones de verdad iguales con la excepción de que una haga verdadera a P_{n-1} y la otra la haga falsa. Otra vez utilizando el Teorema de la Deducción podemos eliminar a P_{n-1} de entre las hipótesis. Después de n pasos habremos eliminado todas las hipótesis y tendremos que $\vdash \phi$. ■

Observación. Toda tautología en el lenguaje extendido $\mathscr{L}_0$ usado en el capítulo anterior es un teorema del CE, pues sólo tenemos que escribirlo de manera equivalente usando sólo los símbolos $\neg$ y $\Rightarrow$. El resultado será una tautología del lenguaje del CE y por lo tanto, en virtud del Teorema de Completud, un teorema del CE.

Para enunciar el siguiente corolario del Teorema de Completud para el CE necesitamos definir lo que quiere decir que una teoría formal sea consistente.

> **Definición.** Una teoría formal $\mathscr{T}$ es *consistente* si no existe ninguna fórmula ϕ de su lenguaje tal que tanto ella como su negación sean ambas teoremas de $\mathscr{T}$.

Corolario 5.10. *El CE es una teoría consistente.*

Demostración.

Si el CE fuera inconsistente existiría una fórmula ϕ tal que $\vdash \phi$ y $\vdash \neg\phi$. Por el Teorema de Validez tendríamos que tanto ϕ como $\neg\phi$ son tautologías. Esto es imposible. ■

Ejercicios

1. Verifique en cada caso si la fórmula es o no un teorema:

a. $\alpha \Rightarrow (\alpha \Rightarrow \alpha)$

b. $(\alpha \Rightarrow \alpha) \Rightarrow \alpha$

c. $(\alpha \Rightarrow (\beta \Rightarrow \alpha)) \Rightarrow (\beta \Rightarrow \alpha)$

d. $(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$

e. $(\alpha \Rightarrow \gamma) \Rightarrow (\neg\alpha \Rightarrow \beta)$

f. $((\alpha \Rightarrow \gamma) \vee (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \vee \beta) \Rightarrow \gamma)$

2. Pruebe que la consistencia de CE (Corolario 5.10) equivale a que existe una fórmula α de $\mathscr{L}$ tal que $\nvdash \alpha$.

3. Suponga que CE fuera inconsistente. Pruebe que bajo este supuesto, en el CE podría demostrarse cualquier fórmula de $\mathscr{L}$. (El sistema sería inútil, pues serían demostrables todas las fórmulas y no sólo las que son válidas).

5.4 Un sistema de deducción natural

Aunque el sistema formal que acabamos de estudiar es excelente, en el sentido de que con tres esquemas axiomáticos y una única regla de inferencia se obtienen todas las tautologías, tiene un inconveniente: es demasiado rígido y por tanto es muy difícil demostrar teoremas en él.

En esta sección desarrollaremos un sistema para la lógica proposicional llamado *deducción natural.* No vamos a obtener una teoría formal en sentido estricto, pues no tiene axiomas, sólo tiene reglas de inferencia que nos permiten deducir ciertas fórmulas a partir de otras.[4]

Veremos que es un sistema que rescata más la manera de razonar en matemáticas y tiene la ventaja adicional de que también tiene la propiedad de que sus teoremas son precisamente las tautologías del lenguaje de proposiciones.

---

[4]En [Sm] se tiene un cálculo tipo deducción natural basado en la técnica de árboles semánticos (*cf.* la nota 4 de la sección 4.8), y cuyo sistema de reglas de inferencia semeja al ilustrado por la Figura 4.11 para la construcción de árboles.

Como no hay axiomas en el sistema, si se quiere demostrar una fórmula de la forma $\phi \Rightarrow \tau$, se procede exactamente de la manera que uno procedería en matemáticas: se supone ϕ y a partir de ϕ se trata de llegar a τ. τ "depende"[5] de ϕ, pero la fórmula $\phi \Rightarrow \tau$ no dependerá de ϕ.

En este sistema tenemos, pues, el derecho de introducir premisas adicionales, pero debemos tener cuidado a la hora de derivar, de llevar un control sobre qué conclusiones dependen de qué premisas adicionales.

Definamos el sistema de deducción natural para el cálculo de enunciados, denotado CEN. El lenguaje en el que se definirá esta teoría será el lenguaje proposicional completo, esto es, con los cinco conectivos lógicos: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ y $\Leftrightarrow$. En toda esta sección $\mathscr{L}_0$ denotará este lenguaje y a menudo se omitirá mencionarlo, sobreentendiéndose que cada vez que se hable de fórmulas se trata de fórmulas de $\mathscr{L}_0$.

Definición. Una *demostración del CEN* es una sucesión finita de fórmulas bien formadas de $\mathscr{L}_0$, cada una de las cuales tiene asignado un conjunto de números (llamados números de premisa) y tal que la sucesión ha sido construida de acuerdo con las siguientes reglas (siendo α, β, γ y δ fórmulas de $\mathscr{L}_0$):

P (*Introducción de premisas*)

Cualquier fórmula puede ser introducida en una línea, tomando el número de esa línea como único número de premisa.

MP (*Modus Ponens*)

β puede ser introducida en una línea si α y $\alpha \Rightarrow \beta$ aparecen en líneas anteriores; como números de premisa de la nueva línea se toman todos los números de premisa de esas líneas anteriores.

MT (*Modus Tollens*)

α puede ser introducida en una línea si β y $\neg\alpha \Rightarrow \neg\beta$ aparecen en líneas anteriores; como números de premisa de la nueva línea se toman todos los números de premisa de esas líneas anteriores.

C (*Condicionalización*)

$\alpha \Rightarrow \beta$ puede ser introducida en una línea si β aparece en una línea anterior; como números de premisa de esta nueva línea se toman todos los números de

[5] El sentido exacto de la palabra "depende" se verá más adelante, cuando se defina el sistema.

premisa de β con excepción (si se desea) del número de línea correspondiente a la línea de α.

D (*Intercambio definicional*)

Si β se obtiene a partir de α reemplazando una ocurrencia de una fórmula γ en α por una fórmula δ tal que γ y δ son definicionalmente equivalentes, y si α aparece en la lista, entonces puede introducirse β en la lista; como números de premisa, esta nueva línea llevará los números de premisa de α.

Fórmulas definicionalmente equivalentes:

$$(\alpha \vee \beta) := (\neg\alpha \Rightarrow \beta)$$
$$(\alpha \wedge \beta) := \neg(\alpha \Rightarrow \neg\beta)$$
$$(\alpha \Rightarrow \beta) := (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$$

Definición. Si se tiene una demostración en el CEN cuya última fórmula es ϕ y si Γ es el conjunto de fórmulas que aparecen en las líneas numeradas con los números de premisa de ϕ, entonces se dice que ϕ es *derivable a partir de* Γ en el CEN.

Deberíamos escribir: $\Gamma \vdash_{\text{CEN}} \phi$, para precisar el sistema formal CEN; sin embargo, a fin de simplificar haremos abuso de la notación y escribiremos $\Gamma \vdash \phi$.[6]

Definición. Una fórmula ϕ es un *teorema del CEN* si y sólo si es derivable en el CEN a partir de $\varnothing$. Notación $\vdash \phi$.[7]

Ejemplos de derivaciones en el CEN

En los ejemplos que siguen α, β y γ son fórmulas arbitrarias.

1. *Principio del silogismo*

$\vdash (\alpha \Rightarrow \beta) \Rightarrow ((\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \gamma))$

$\{1\}$	(1) $\alpha \Rightarrow \beta$	P

[6] Confróntese la nota 2, dada anteriormente.
[7] Idem.

$\{2\}$	(2) $\beta \Rightarrow \gamma$	P
$\{3\}$	(3) α	P
$\{1, 3\}$	(4) β	1, 3 MP
$\{1, 2, 3\}$	(5) γ	2, 4 MP
$\{1, 2\}$	(6) $\alpha \Rightarrow \gamma$	3, 5 C
$\{1\}$	(7) $(\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \gamma)$	2, 6 C
$\varnothing$	(8) $(\alpha \Rightarrow \beta) \Rightarrow ((\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \gamma))$	

Observaciones:

1) Notemos que esta lista no es una derivación en el CEN en sentido estricto, pues α, β y γ no son fórmulas del lenguaje. Pero es evidente que si fueran sustituidas uniformemente por fórmulas específicas (*cf.* el Teorema 4.7) el resultado sería una derivación en el CEN.

2) A la izquierda de la lista de fórmulas, entre llaves, se escriben los números de premisa de cada fórmula; aparte se escriben los números de línea. No hay que confundirlos, los números de línea van aumentando de 1 en 1, los números de premisa representan los supuestos de los cuales dependen las fórmulas de las líneas correspondientes.

3) A la derecha de la lista de fórmulas se pone su justificación para facilitar la verificación de que en efecto se trata de una derivación en el sistema.

Es importante hacer notar que la regla C, al permitirnos quitar números de premisas en una derivación, es el equivalente en nuestro sistema del teorema de la deducción de la teoría formal CE.

(2) $\alpha \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow \beta)$

$\{1\}$	(1) α	P
$\{2\}$	(2) $\alpha \Rightarrow \beta$	P
$\{1, 2\}$	(3) β	1, 2 MP
$\{1\}$	(4) $(\alpha \Rightarrow \beta) \Rightarrow \beta$	2, 3 C
$\varnothing$	(5) $\alpha \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow \beta)$	1, 4 C

(3) $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$

$\{1\}$	(1) $\alpha \Rightarrow (\beta \Rightarrow \gamma)$	P
$\{2\}$	(2) $\alpha \Rightarrow \beta$	P
$\{3\}$	(3) α	P

$\{2, 3\}$	(4) β	2, 3 MP
$\{1, 3\}$	(5) $\beta \Rightarrow \gamma$	1, 3 MP
$\{1, 2, 3\}$	(6) γ	4, 5 MP
$\{1, 2\}$	(7) $\alpha \Rightarrow \gamma$	3, 6 C
$\{1\}$	(8) $(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$	2, 7 C
$\varnothing$	(9) $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$	1, 8 C

(4) $\alpha \Rightarrow \alpha$

$\{1\}$	(1) α	P
$\varnothing$	(2) $\alpha \Rightarrow \alpha$	1 C

(5) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

$\{1\}$	(1) α	P
$\{1\}$	(2) $\beta \Rightarrow \alpha$	1 C
$\varnothing$	(3) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$	1, 2 C

En los dos últimos ejemplos aparecen aplicaciones un poco raras de la regla C, pero si se leen las reglas con detenimiento se notará que para poder aplicar la regla C no se requiere que ambas componentes del condicional aparezcan con anterioridad en la lista, sólo se requiere del consecuente. Cuando (y generalmente ocurre esto) aparece también el antecedente con anterioridad, entonces tenemos la ventaja de poder quitar su número de línea del conjunto de números de premisa, si así se quiere.

En el Ejemplo 4, α fue tomada como antecedente y como consecuente de la implicación, por lo que su número de premisa fue eliminado al escribir los números de premisa de $\alpha \Rightarrow \alpha$.

En el Ejemplo 5, como β no aparecía en la derivación, al aplicar la regla C a α para obtener $\beta \Rightarrow \alpha$, los números de premisa de α y de $\beta \Rightarrow \alpha$ son los mismos.

(6) $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$

$\{1\}$	(1) $\neg\alpha$	P
$\{2\}$	(2) α	P
$\{1\}$	(3) $\neg\beta \Rightarrow \neg\alpha$	1 C
$\{1, 2\}$	(4) β	2, 3 MT
$\{1\}$	(5) $\alpha \Rightarrow \beta$	2, 4 C
$\varnothing$	(6) $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$	1, 5 C

(7) $\alpha \Rightarrow (\neg\alpha \Rightarrow \beta)$

La demostración es análoga a la del Ejemplo 6.

(8) $\neg\neg\alpha \Rightarrow \alpha$

$\{1\}$	(1) $\neg\neg\alpha$	P
$\{2\}$	(2) $\neg\alpha$	P
$\{1\}$	(3) $\neg\neg\neg\neg\alpha \Rightarrow \neg\neg\alpha$	1 C
$\{1, 2\}$	(4) $\neg\neg\neg\alpha$	2, 3 MT
$\{1\}$	(5) $\neg\alpha \Rightarrow \neg\neg\neg\alpha$	2, 4 C
$\{1\}$	(6) α	1,5 MT
$\varnothing$	(7) $\neg\neg\alpha \Rightarrow \alpha$	1, 6 C

Es claro que si se tiene una derivación en el CEN para una fórmula bien formada ϕ, cualquier otra fórmula χ que sea una instancia de sustitución de ϕ podrá ser derivada de la misma forma que lo fue ϕ, sólo necesitamos recorrer paso a paso la derivación de ϕ y efectuar las sustituciones requeridas para transformar ϕ en χ, el resultado será una derivación de χ en el CEN.

Si una fórmula fue introducida en la derivación por la regla P, cualquier instancia de sustitución de la fórmula estará justificada por regla P para aparecer en la lista. Las reglas MP, MT, C y D preservan sustituciones uniformes, esto es, si χ se obtuvo de ϕ y $\phi \Rightarrow \chi$ por MP entonces χ' se obtiene de ϕ' y $\phi' \Rightarrow \chi'$ por MP, donde ϕ' y χ' son instancias de sustitución de ϕ y χ tales que las mismas fórmulas en ambas fueron sustituidas por fórmulas iguales.

De este modo, si se empezó con una derivación en el CEN, al efectuar sustituciones uniformes en toda la derivación se termina con una derivación en el CEN.

Esta observación nos permite introducir la siguiente regla al sistema.

Regla TE

Cualquier fórmula de $\mathscr{L}_0$ que sea una instancia de sustitución de un teorema del CEN puede ser introducida en una línea, con el conjunto vacío como su conjunto de números de premisa. Más generalmente, χ puede ser introducido en una línea si $\phi_1, \ldots, \phi_n$ aparecen en líneas anteriores y el condicional $(\phi_1 \Rightarrow (\phi_2 \Rightarrow \ldots (\phi_n \Rightarrow \chi)\ldots))$ es una instancia de sustitución de un teorema del CEN ya probado; como números de premisa de esa nueva línea se toman todos los de esas líneas anteriores.

La regla TE no es una regla como las reglas básicas porque cualquier deducción que se haga utilizándola también se puede hacer sin ella, utilizando sólo las reglas básicas. El siguiente teorema ilustra el uso de esta nueva regla.

(9) $\alpha \Rightarrow \neg\neg\alpha$

$\varnothing$	(1) $\neg\neg\neg\alpha \Rightarrow \neg\alpha$	TE Ejemplo 8
$\{2\}$	(2) α	P
$\{2\}$	(3) $\neg\neg\alpha$	1, 2 MT
$\varnothing$	(4) $\alpha \Rightarrow \neg\neg\alpha$	2, 3 C

Ejercicios

Demostrar que si α, β y γ son fórmulas del lenguaje del CEN entonces las siguientes fórmulas son teoremas del CEN:

a. $(\neg\alpha \Rightarrow \neg\beta) \Rightarrow (\beta \Rightarrow \alpha)$
b. $(\alpha \Rightarrow \neg\beta) \Rightarrow (\beta \Rightarrow \neg\alpha)$
c. $(\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$
d. $(\neg\alpha \Rightarrow \alpha) \Rightarrow \alpha$
e. $(\alpha \Rightarrow \neg\alpha) \Rightarrow \neg\alpha$
f. $\alpha \Rightarrow (\beta \Rightarrow (\alpha \wedge \beta))$
g. $\alpha \Rightarrow (\alpha \vee \beta)$
h. $\alpha \vee \neg\alpha$
i. $\neg(\alpha \wedge \neg\alpha)$
j. $((\alpha \Rightarrow \gamma) \wedge (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \vee \beta) \Rightarrow \gamma)$

5.5 Validez y completud para CEN

En esta sección demostraremos para el CEN los teoremas demostrados para el CE en la sección 5.3.

Teorema 5.11. (de Completud del CEN). *Toda tautología del lenguaje del CEN es un teorema del CEN.*

Demostración.

Sea ϕ una tautología. Sin pérdida de generalidad podemos suponer que en ϕ los únicos conectivos que aparecen son $\Rightarrow$ y $\neg$, ya que los otros pueden ser eliminados por medio de equivalencias tautológicas. En virtud del Teorema de Completud para el CE, es suficiente con demostrar que todo teorema del CE es un teorema de CEN.

Como el Modus Ponens es una regla de inferencia de CEN, bastará con probar que los tres esquemas axiomáticos de CE son teoremas de CEN. Los axiomas 1

y 2 de CE son teoremas de CEN en virtud de los ejemplos 3 y 5 de la sección anterior, sólo nos resta demostrar que el axioma 3 de CE es un teorema de CEN.

La siguiente lista de fórmulas es una demostración en el CEN de $(\neg\alpha \Rightarrow \neg\beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha)$ cuando α y β son fórmulas bien formadas del lenguaje:

$\{1\}$	(1) $\neg\alpha \Rightarrow \neg\beta$	P
$\{2\}$	(2) $\neg\alpha \Rightarrow \beta$	P
$\{3\}$	(3) $\neg\alpha$	P
$\{2,3\}$	(4) β	2, 3 MP
$\{1,2,3\}$	(5) α	1, 4 MT
$\{1,2\}$	(6) $\neg\alpha \Rightarrow \alpha$	3, 5 C
$\{1\}$	(7) $(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\alpha \Rightarrow \alpha)$	2, 6 C
$\varnothing$	(8) $(\neg\alpha \Rightarrow \alpha) \Rightarrow \alpha$	TE Ejercicio d secc. anterior
$\{1\}$	(9) $(\neg\alpha \Rightarrow \beta) \Rightarrow \alpha$	7, 8 Ejemplo 1
$\varnothing$	(10) $(\neg\alpha \Rightarrow \neg\beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha)$	1, 9 C

Dada una tautología en el lenguaje del CEN, se puede transformar en una tautología en el lenguaje del CE utilizando las definiciones que se dieron al enunciar la regla del intercambio definicional, el resultado es una tautología en el CE y como toda tautología es un teorema del CE y hemos demostrado que los tres esquemas axiomáticos del CE son demostrables en el CEN, la prueba en CE se puede reproducir en CEN. Por lo tanto toda tautología del lenguaje del CEN es un teorema del CEN. ■

Teorema 5.12 (de Validez del CEN). *Todo teorema del CEN es una tautología.*

Demostración.

Sea ϕ una fórmula que aparece al final de alguna derivación en el CEN, probaremos por inducción sobre la longitud de la derivación que ϕ es consecuencia tautológica de las premisas ϕ. (Las premisas de ϕ son las fórmulas que tienen como número de línea algún número de premisa de ϕ.)

Base: ϕ aparece en la primera línea.

Entonces ϕ fue introducida por la regla P y ϕ es su única premisa.

Hipótesis de inducción: Supongamos que para toda fórmula cuya derivación conste de menos de k pasos se tiene que es consecuencia tautológica de sus premisas.

Sea ϕ una fórmula cuya derivación consta de k pasos.

Caso 1. Si ϕ fue introducida por regla P entonces ϕ es su única premisa.

Caso 2. Si ϕ fue introducida por MP entonces en la derivación de ϕ aparecen dos fórmulas anteriores χ y $\chi \Rightarrow \phi$ y las premisas de ϕ son todas las premisas de χ y $\chi \Rightarrow \phi$. Por hipótesis de inducción, tanto χ como $\chi \Rightarrow \phi$ son consecuencia tautológica de sus respectivas premisas, por lo tanto, como ϕ es consecuencia tautológica de χ y $\chi \Rightarrow \phi$, ϕ es consecuencia tautológica de sus premisas.

Caso 3. Si ϕ fue introducida por MT, el argumento es exactamente igual al del caso anterior.

Caso 4. Si ϕ fue introducida por regla C entonces ϕ es de la forma $\chi \Rightarrow \psi$, donde ψ aparece en una línea anterior. Las premisas de ϕ son las premisas de ψ con la posible excepción de χ. Sean $P_1, \ldots, P_n$, χ las premisas de ψ. Por hipótesis de inducción sabemos que $P_1, \ldots, P_n, \chi \vDash \psi$ y lo que queremos probar es que $P_1, \ldots, P_n \vDash \chi \Rightarrow \psi$. Sea $|\cdot|$ una asignación de valores de verdad para las letras proposicionales tal que $|P_i| = 1$ para $i = 1, \ldots, n$. Si $|\chi| = 0$ entonces $|\chi \Rightarrow \psi| = 1$, y si $|\chi| = 1$ entonces por hipótesis de inducción tenemos que $|\chi \Rightarrow \psi| = 1$.

Caso 5. Si ϕ fue introducida por la regla D entonces aparece una fórmula anterior a ϕ en la derivación que es definicionalmente equivalente, y por tanto tautológicamente equivalente a ϕ, y que tiene las mismas premisas que ϕ. Por hipótesis de inducción se tiene que esa fórmula es consecuencia tautológica de sus premisas, de lo que se sigue que ϕ también lo es.

Si ϕ es un teorema del CEN entonces aparece en una derivación con conjunto de premisas vacío. Por lo que acabamos de ver, entonces ϕ es consecuencia tautológica de $\varnothing$, esto es, ϕ es una tautología. ■

5.6 Teorema de Compacidad

En esta sección demostraremos el Teorema de Compacidad[8] para la lógica de proposiciones y algunos de sus corolarios. Este teorema es en realidad un caso particular del Teorema de Compacidad para la lógica de primer orden, la cual estudiaremos en los Capítulos 7 y 8. El Teorema de Compacidad para la lógica de

[8]Para aquellos lectores con un conocimiento en matemáticas básicas, cabe mencionar que el Teorema de Compacidad debe su nombre a que, en términos de una formulación topológica, significa que un cierto espacio topológico es compacto [Am]-[Eb]-[Ma].

primer orden es uno de los teoremas de la lógica que más aplicaciones ha tenido en otras ramas de la matemática.

Recordemos primero que un conjunto Σ de fórmulas bien formadas de un lenguaje proposicional es *satisfacible* si existe una asignación $|\cdot|$ para las letras proposicionales que aparecen en Σ bajo la cual todas las fórmulas de Σ son verdaderas.

Sea Σ un conjunto de fórmulas. El Teorema de Compacidad nos asegura que: "Σ es satisfacible si y sólo si todo subconjunto finito de Σ lo es". Antes de demostrar este teorema queremos hacer notar que si Σ es un conjunto finito de fórmulas entonces el teorema es una trivialidad, puesto que $\Sigma \subseteq \Sigma$. El caso interesante es cuando Σ es infinito, en cuyo caso el teorema asegura que si para cada subconjunto finito de Σ existe una asignación que satisface a todas sus fórmulas entonces se puede encontrar una asignación que satisfaga a todas las fórmulas de Σ. Nótese que las asignaciones que existen para cada subconjunto finito de Σ no tienen por qué coincidir en las letras que aparezcan en la intersección de los dominios de las asignaciones, de manera que no se puede tomar la unión de todas las asignaciones que satisfacen a los subconjuntos finitos de Σ. Por tanto, la demostración resulta un poco más complicada.

> **Definición.** Llamemos a un conjunto Σ *finitamente satisfacible* si y sólo si todo subconjunto finito de Σ es satisfacible.

Lema 5.13. *Sea Σ un conjunto finitamente satisfacible de fórmulas. Entonces, para cualquier fórmula α, alguno de los conjuntos $\Sigma \cup \{\alpha\}$ o $\Sigma \cup \{\neg\alpha\}$ es finitamente satisfacible.*

Demostración.

Supongamos que $\Sigma \cup \{\neg\alpha\}$ no es finitamente satisfacible. Entonces existe un subconjunto finito de $\Sigma \cup \{\neg\alpha\}$ digamos $\{\sigma_1, \ldots, \sigma_n, \neg\alpha\}$ que no es satisfacible. (Sabemos que $\neg\alpha$ tiene que pertenecer al conjunto porque por hipótesis Σ es finitamente satisfacible.) Entonces $\sigma_1, \ldots, \sigma_n \vDash_T \alpha$ y $\{\sigma_1, \ldots, \sigma_n\}$ es satisfacible.

Tomemos ahora un subconjunto finito arbitrario de $\Sigma \cup \{\alpha\}$ y llamémosle Γ. Si $\alpha \notin \Gamma$ entonces $\Gamma \subseteq \Sigma$ y es, por lo tanto, satisfacible. Si $\alpha \in \Gamma$ entonces Γ es de la forma $\{\gamma_1, \ldots, \gamma_m, \alpha\}$, con $\gamma_1, \ldots, \gamma_m \in \Sigma$. En este caso el conjunto $\{\gamma_1, \ldots, \gamma_m, \sigma_1, \ldots, \sigma_n\}$ es un subconjunto finito de Σ y es satisfacible. Como $\sigma_1, \ldots, \sigma_n \vDash_T \alpha$, cualquier asignación que satisfaga a $\{\gamma_1, \ldots, \gamma_m, \sigma_1, \ldots, \sigma_n\}$

también hará verdadera a α, por lo que Γ es satisfacible. Esto prueba que $\Sigma \cup \{\alpha\}$ es finitamente satisfacible. ■

El Teorema de Compacidad se puede entonces enunciar de la siguiente forma:

Teorema 5.14 (de Compacidad). *Un conjunto de fórmulas Σ es satisfacible si y sólo si es finitamente satisfacible.*

Demostración.

Evidentemente, si Σ es satisfacible todo subconjunto finito de él lo es, la asignación que satisface a todos los elementos de Σ satisface a cualquier subconjunto finito de Σ.

Supongamos ahora que Σ es un subconjunto de fórmulas finitamente satisfacible. Primero vamos a extender a Σ a un conjunto Γ finitamente satisfacible y que sea maximal con esta propiedad. (Esto quiere decir que si existe algún otro conjunto Γ' tal que $\Sigma \subseteq \Gamma'$ y Γ' es finitamente satisfacible, entonces $\Gamma' \subseteq \Gamma$.)

Primero enumeramos todas las fórmulas bien formadas de $\mathscr{L}$ y obtenemos una lista (fija) $\phi_1, \phi_2, \ldots, \phi_n, \ldots$ Esto se puede hacer porque el conjunto de fórmulas bien formadas de $\mathscr{L}$ es numerable. Con esta lista vamos a construir una cadena de conjuntos de fórmulas, por recursión sobre los números naturales.

$$\Gamma_0 = \Sigma$$

$$\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{\phi_{n+1}\} & \text{si es finitamente satisfacible} \\ \Gamma_n \cup \{\neg\phi_{n+1}\} & \text{si no lo es.} \end{cases}$$

Entonces cada Γ_n es finitamente satisfacible, por el Lema 5.13. Sea $\Gamma = \bigcup_n \Gamma_n$. Así, $\Sigma \subseteq \Gamma$ y Γ es finitamente satisfacible, puesto que cualquier subconjunto finito de Γ es subconjunto de alguna Γ_n y es por tanto satisfacible. Para ver que Γ es maximal es suficiente con notar que dada cualquier fórmula ϕ se tiene $\phi \in \Gamma$ o $\neg\phi \in \Gamma$ y por tanto cualquier extensión propia de Γ es insatisfacible.

Ahora vamos a definir una asignación $|\cdot|$ de valores de verdad para las letras de la siguiente manera: $|P| = 1$ si y sólo si $P \in \Gamma$, donde P es una letra proposicional arbitraria.

Para completar la demostración probaremos por inducción sobre el número de símbolos de α, que para toda fórmula α, $|\alpha| = 1$ si y sólo si $\alpha \in \Gamma$.

Si α es una letra proposicional el resultado es justamente la definición de $|\cdot|$.

Supongamos que la afirmación es verdadera para toda fórmula con menos de k símbolos y sea α una fórmula con k símbolos.

Caso 1. $\alpha = \neg\beta$. Entonces $|\alpha| = 1$ si y sólo si $|\beta| = 0$ si y sólo si (por hipótesis de inducción) $\beta \notin \Gamma$ si y sólo si $\neg\beta \in \Gamma$ (por ser Γ maximal).

Caso 2. $\alpha = (\beta \Rightarrow \gamma)$. Si $|\alpha| = 0$ entonces $|\beta| = 1$ y $|\gamma| = 0$, por hipótesis de inducción, entonces $\beta \in \Gamma$ y $\gamma \notin \Gamma$. Como Γ es maximal, $\neg\gamma \in \Gamma$ y $\{\beta, \neg\gamma, \beta \Rightarrow \gamma\}$ es insatisfacible. Por lo tanto $(\beta \Rightarrow \gamma) \notin \Gamma$.

Por otro lado, si $\alpha \notin \Gamma$ entonces $\neg\alpha \in \Gamma$, esto es, $\neg(\beta \Rightarrow \gamma) \in \Gamma$. Como Γ es maximal y tanto β como $\neg\gamma$ son consecuencias de $\neg(\beta \Rightarrow \gamma)$ se tiene que $\beta \in \Gamma$ y $\neg\gamma \in \Gamma$. Aplicando la hipótesis de inducción obtenemos que $|\beta| = 1$ y $|\gamma| = 0$, por lo que $|\alpha| = 0$. ■

Corolario 5.15 *Si* $\Sigma \vDash_T \alpha$ *entonces existe* $\Sigma_o \subseteq \Sigma$, Σ_o *finito, tal que* $\Sigma_o \vDash_T \alpha$. ■

Con base en el Teorema de Compacidad podemos proporcionar una versión más fuerte que la expuesta en la sección 5.3 para CE, para relacionar las nociones primordiales obtenidas bajo los aspectos semántico y sintáctico:

$$\Gamma \vDash_T \alpha \text{ (implicación tautológica) } vs. \ \Gamma \vdash_{CE} \alpha \text{ (derivación).}$$

Corolario 5.16 (Teorema de Completud Fuerte para el CE). *Sea* Γ *un conjunto de fórmulas y* α *una fórmula cualquiera. Entonces* $\Gamma \vDash_T \alpha$ *si y sólo si* $\Gamma \vdash_{CE} \alpha$.

Demostración.

$\Gamma \vDash_T \alpha$ si y sólo si $\Gamma \cup \{\neg\alpha\}$ es insatisfacible, por el Teorema 4.14. Por el Teorema de Compacidad, esto pasa si y sólo si existen $\gamma_1, \ldots, \gamma_n \in \Gamma$ tales que $\{\gamma_1, \ldots, \gamma_n, \neg\alpha\}$ es insatisfacible, y esto es cierto, nuevamente por el Teorema 4.14, si y sólo si $\gamma_1, \ldots, \gamma_n \vDash_T \alpha$ si y sólo si $\vDash (\gamma_1 \Rightarrow \ldots \Rightarrow (\gamma_n \Rightarrow \alpha)\ldots)$. Aplicando ahora el Teorema 5.9 (de Completud) para el CE, podemos concluir que esto sucede si y sólo si $\vdash_{CE} (\gamma_1 \Rightarrow \ldots \Rightarrow (\gamma_n \Rightarrow \alpha)\ldots)$ lo que implica que $\gamma_1, \ldots, \gamma_n \vdash_{CE} \alpha$ y por lo tanto $\Gamma \vdash_{CE} \alpha$.

Si $\Gamma \vdash_{CE} \alpha$ entonces existen $\gamma_1, \ldots, \gamma_n \in \Gamma$ tales que $\gamma_1, \ldots, \gamma_n \vdash_{CE} \alpha$, ya que las pruebas en el CE son finitas. Aplicando el Teorema de la Deducción n veces obtenemos que $\vdash_{CE} (\gamma_1 \Rightarrow \ldots \Rightarrow (\gamma_n \Rightarrow \alpha)\ldots)$, y por el argumento del párrafo anterior se obtiene que $\Gamma \vDash_T \alpha$. ■

Una versión equivalente al corolario anterior viene dada por el resultado siguiente, el cual establece el nexo entre los conceptos de satisfacibilidad (semántico) y el de consistencia (sintáctico).

Corolario 5.17. *Sea* Γ *un conjunto de fórmulas de* $\mathscr{L}$. *Entonces,* Γ *es satisfacible si y sólo si* Γ *es consistente.* ■

De haber probado directamente el Teorema de Completud fuerte (ya sea el Corolario 5.17 o el 5.18) resulta relativamente fácil obtener de éste el Teorema de Compacidad, dado que la versión sintáctica del mismo se sigue de la noción de derivabilidad.

Teorema 5.18. Γ *es consistente si y sólo si cada subconjunto finito de* Γ *es consistente.*

Demostración.

Si Γ es consistente, entonces todo subconjunto de él lo es, en particular todos los finitos.

Supongamos ahora que todo $\Delta \subseteq \Gamma$, Δ finito, es consistente y que Γ es inconsistente, luego existe β en $\mathscr{L}$ tal que $\Gamma \vdash_{CE} \beta$ y $\Gamma \vdash_{CE} \neg\beta$. Pero entonces, por el Ejercicio 2 de la secc. 5.2, existe $\Delta' \subseteq \Gamma$, Δ' finito, tal que $\Delta' \vdash_{CE} \beta$ y $\Delta' \vdash_{CE} \neg\beta$, *i.e.*, Δ' es inconsistente; contradiciendo así la hipótesis. Por lo tanto, Γ es consistente. ■

Otros resultados sintácticos que tienen análogos semánticos vía compacidad y completud son los siguientes:

Teorema 5.19. Γ *es consistente si y sólo si existe* β *en* $\mathscr{L}$ *tal que* $\Gamma \nvdash_{CE} \beta$. ■

Teorema 5.20. $\Gamma \vdash_{CE} \alpha$ *si y sólo si* $\Gamma \cup \{\neg\alpha\}$ *es inconsistente.* ■

Ejercicios

1. Demostrar el Corolario 5.15.
2. Demostrar que el Corolario 5.15 es equivalente al Teorema de Compacidad.

3. Suponer el Teorema de Completud fuerte y demostrar el Teorema de Compacidad.

4. Pruebe que el Corolario 5.16 es equivalente al 5.17.

5. Pruebe sintácticamente (*i.e.*, sin el aval del Teorema de Completud fuerte):

a. el Teorema 5.19.

b. el Teorema 5.20.

c. (Una versión del Teorema de la Deducción). $\Gamma \vdash_{\mathrm{CE}} \alpha$ si y sólo si existe $\Delta \subseteq \Gamma$, Δ finito, digamos $\Delta = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$, tal que $\vdash_{\mathrm{CE}} (\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) \Rightarrow \alpha$.

Capítulo 6
Lógica proposicional: enfoque algorítmico

Un hombre tras una gran penitencia religiosa y en pleno éxtasis, consigue al fin una cita con El Ser Supremo:
Señor. ¿Qué es para Vos un milenio?
— ¡Tan solo un segundo!
— ¿Y un millón de dólares?
— Un simple centavo.
— Señor. ¡Concédame un millón de dólares!
— ¡Espérame un segundo!

Anónimo

6.1 Introducción

El sistema formal de la lógica proposicional tiene la propiedad adicional de ser *decidible.* Esto significa que hay un procedimiento mecánico con el cual se determina en un número finito de pasos si una fórmula dada es o no un teorema. Dada una fórmula α, para averiguar si $\vdash \alpha$, preguntamos si $\models \alpha$, la respuesta a esto se obtiene vía las tablas de verdad (que nos proporcionan el procedimiento mecánico aludido), y al resultado hallado se le aplican los Teoremas de Completud y Validez, para entonces dar el fallo.

Ahora bien, la condición de "en un número finito de pasos" puede resultar muy relativa: sólo compare la "finitud" de un milenio con la vida del hombre del cuento.

El problema de determinar si una cadena de símbolos dada del lenguaje formal de proposiciones $\mathscr{L}_0$ es o no una fórmula bien formada es más bien

trivial. De hecho, pueden darse *algoritmos* que resuelven *eficientemente* este problema (*cf.* la sección 4.2, y para mayor información [En]-[Fr]). Considerando el *tiempo computacional* como equivalente al número de "operaciones elementales" realizadas bajo una entrada de longitud n, por *eficiencia* se entiende cuando el tiempo empleado por un algoritmo en resolver un problema determinado es de un *orden polinomial,* $\mathcal{O}(p(n))$. Por otra parte, el problema de *decidir* ahora si una fbf en $\mathscr{L}_0$ es satisfacible o es una tautología puede involucrar un gran número de manipulaciones (*i.e.*, "talacha"). De la sección 4.8, podemos afirmar que, *e.g.*, la técnica "por tablas" resulta claramente ineficiente, pues para decidir que $\Gamma \vDash_T \alpha$ o no, si ocurren n letras proposicionales (una entrada de longitud n) en $\Gamma \cup \{\alpha\}$, entonces se requiere construir una tabla de 2^n renglones. El algoritmo así derivado es de un tiempo de *orden exponencial,* $\mathcal{O}(2^n)$.

En términos generales, un problema es *tratable* si existe algún algoritmo asociado eficiente; caso contrario se dice *intratable.*

Ahora bien, cabría preguntarse si existen algoritmos eficientes tanto para la satisfacción como para la validez de las fórmulas. Sin embargo, la respuesta respecto de la satisfacibilidad hasta el momento ha sido elusiva, al probarse que este problema pertenece a una "familia" de problemas "equivalentes entre sí" conocida como *problemas $\mathcal{NP}$-completos. Grosso modo*, los problemas $\mathcal{NP}$-completos son aquéllos para los que hallar un algoritmo eficiente es un gran problema abierto, pues esta familia cuenta con la cualidad adicional que de probarse la existencia de un algoritmo eficiente para alguno de ellos repercutiría en que todos estos problemas tendrían soluciones eficientes, mientras que una respuesta negativa, de que el algoritmo óptimo es necesariamente de un tiempo de orden no polinomial para un problema particular, implicaría que todos son intratables. De esta manera, el descubrimiento de un algoritmo general y eficiente para el problema de decisión sobre la satisfacibilidad de las fórmulas es más que difícil, si no imposible. En cuanto al problema de la validez de las fórmulas, todo parece indicar que es un problema aun más difícil que el de satisfacción, dado que es un problema abierto el que si la validez siquiera está en $\mathcal{NP}$ (menos ha de estar en los $\mathcal{NP}$-completos, siendo éste un subconjunto propio de $\mathcal{NP}$). Por consiguiente, a pesar de que el cálculo proposicional sea decidible, desde un punto de vista práctico, su problemática es intratable.

Aunque este panorama resulta desalentador (computacionalmente hablando), existen algoritmos eficientes para el problema de satisfacibilidad en ciertos conjuntos de fórmulas, *e.g.* las *claúsulas de Horn.* Un algoritmo tal (de complejidad

de orden cuadrática) lo proporciona la denominada técnica de *resolución.* Esta técnica procede mediante una clase especial de *prueba* denominada *refutación*: una fórmula se prueba verificando que su negación es insatisfacible.

El hecho que el problema de la satisfacibilidad dentro de un cierto conjunto de las fórmulas sea algoritmicamente tratable hace de la *deducción automática de teoremas* un hecho, y no sólo un aspecto teórico. Esta característica extrapolada a la lógica de predicados (considerada como extensión de la proposicional) y aunado al gran poder expresivo de ésta en la representación del conocimiento, dio pauta a la concepción de la lógica (entiéndase la de predicados) como un lenguaje de programación: la *programación lógica.* (*Cf.* [CL]-[Ga]-[Th] para un estudio de estos temas.)

6.2 Análisis de técnicas semánticas

Anteriormente (secciones 3.3-3.4) se hizo mención de una versión intuitiva de algoritmo, así como su conexión con la noción de decidibilidad. A este respecto, se concibe un *algoritmo* como un procedimiento *efectivo* que se realiza en un número *finito* de pasos bien *definidos* para resolver un problema específico, y que consta además de dos conjuntos: *las entradas* (datos iniciales de los casos particulares del problema) y *las salidas* (las respuestas por obtener de los casos particulares distintos del problema).

Sin embargo, esta definición ha de incurrir en sí en su propia falta: la *definibilidad*; pues por *efectividad*[1] suele entenderse que todas las operaciones a realizar en el algoritmo deben ser lo suficientemente básicas como para ser efectuadas de manera exacta en un lapso finito de tiempo por el procesador (o dispositivo) que ejecute el algoritmo. Este concepto no tiene la precisión necesaria para ser aplicable en todos los casos. Una definición precisa de la *efectividad* viene dada mediante el concepto de *recursividad.* La relación establecida entre la noción (intuitiva) de algoritmo con la (formal) de recursividad conforma la denominada *Tesis de Church.* Ésta no es un teorema en sí, sino más bien una cuestión de fe, dado que relaciona lo que se entiende (informal e intuitivamente) por algoritmo con el proceso matemáticamente formal de recursión (*cf.* [En]-[Th]).

[1]Para evitar confusiones, usaremos el calificativo *efectivo* para referirnos a nociones que involucren la propiedad definida arriba, mientras que el de *eficiente* será cuando el tiempo requerido no sólo es finito, sino además "razonable".

La noción intuitiva de algoritmo es satisfactoria para dar respuestas positivas a si determinado problema es algorítmicamente soluble, pues de ser así, basta simplemente con exhibir un algoritmo que lo resuelva. Para esta sección, esto es lo más que necesitaremos. Para el caso que no exista tal procedimiento efectivo para abordar un problema, esta versión intuitiva ya no es adecuada, y sí resulta entonces indispensable la definición formal.

Concretando en las técnicas semánticas de la sección 4.8, podemos decir que el método por tablas tiene la desventaja de que para dar su respuesta se tiene que explorar todos los casos posibles (2^n renglones, para n letras proposicionales), aunque no todos los renglones sean necesarios. Cuenta como único punto a favor el que evita construir una columna, en virtud del significado del condicional: se requiere al menos tantas columnas como premisas hayan, más una columna adicional para la conclusión. Así, esta técnica es obviamente ineficiente, y su implementación algorítmica consume un tiempo de orden exponencial.

Ahora cabría preguntarse si el método algebraico resulta mejor al tener "atajos" en el proceso de reducción a formas normales. Esto es sólo aparente. Su problema reside precisamente en que el algoritmo para reducir a FN's puede involucrar demasiadas manipulaciones.

Otras dos técnicas que presentaremos aquí son los métodos de Quine [Qu] y de reducción al absurdo. Estos dos métodos comparten la "versatilidad" que tiene el algebraico al compararlos con el de tablas. Pero aun así, no son eficientes, y presentaremos un ejemplo, después de exponer en qué consisten, para despejar cualquier duda en el lector.

Método de Quine

En este procedimiento se consideran interpretaciones parciales de las fórmulas, procediendo según algún orden sobre las letras proposicionales. La estrategia radica en que de obtenerse el mismo valor de verdad para una fórmula al aplicar todas las posibles extensiones de una valuación parcial, entonces resulta irrelevante construir la rama (o subárbol) que brote del nodo correspondiente a esta asignación parcial. Este proceso guarda cierta semejanza con el de árboles semánticos, diferenciándose en que se aplica en "directo" y no por insatisfacción de fórmulas. Ilustrémoslo con un caso.

Ejemplo. Analicemos la validez de la fbf:

$$\alpha = ((P \Rightarrow R) \wedge (Q \Rightarrow R)) \Rightarrow ((P \vee Q) \Rightarrow R)$$

Primero establecemos un orden lexicográfico en las letras: $P < Q < R$, y procedemos de menor a mayor. Consideremos una valuación $|\cdot|$ tal que $|P| = 1$, fijemos el valor 1, e identifiquémoslo con la función de verdad **1**, a fin de, bajo abuso de notación, sustituirlo en la fórmula (tal y como se hizo en el método algebraico). Así, tenemos:

$$((\mathbf{1} \Rightarrow R) \wedge (Q \Rightarrow R)) \Rightarrow ((\mathbf{1} \vee Q) \Rightarrow R) \underset{T}{\models\!\models} (R \wedge (Q \Rightarrow R)) \Rightarrow R$$

De donde, si $|Q| = 1$, obtenemos $(R \wedge (\mathbf{1} \Rightarrow R)) \Rightarrow R \underset{T}{\models\!\models} R \Rightarrow R$, mientras que si $|Q| = 0$, entonces $(R \wedge (\mathbf{0} \Rightarrow R)) \Rightarrow R \underset{T}{\models\!\models} (R \wedge \mathbf{1}) \Rightarrow R \underset{T}{\models\!\models} R \Rightarrow R$, ocurriendo en ambas la fórmula válida $R \Rightarrow R$.

Supongamos ahora que $|P| = 0$, luego

$$((\mathbf{0} \Rightarrow R) \wedge (Q \Rightarrow R)) \Rightarrow ((\mathbf{0} \vee Q) \Rightarrow R) \underset{T}{\models\!\models} (\mathbf{1} \wedge (Q \Rightarrow R)) \Rightarrow (Q \Rightarrow R)$$
$$\underset{T}{\models\!\models} (Q \Rightarrow R) \Rightarrow (Q \Rightarrow R)$$

siendo ésta una tautología.

Este análisis revela que la fbf es válida. Gráficamente, el proceso es representado con el árbol dado por la figura 6.1.

Método por reducción al absurdo (RAA)

Éste no es más que una aplicación del Teorema 4.14. Resulta particularmente útil si la fbf contiene varias ocurrencias del condicional; en cuyo caso se procede de manera tal que se produzca una eventual contradicción (absurdo) al suponer que el consecuente es falso y el antecedente es verdadero, correspondientes a la implicación que sea el conectivo principal. En otros términos, si la fbf α por analizar es de la forma $\alpha = \beta \Rightarrow \gamma$, y suponemos que para una valuación $|\cdot|$ arbitraria, $|\gamma| = 0$, entonces se debe conseguir que $\beta \wedge \neg\gamma$ es insatisfacible (es una fórmula contradictoria). Ilustrémoslo con la misma fbf del ejemplo anterior.

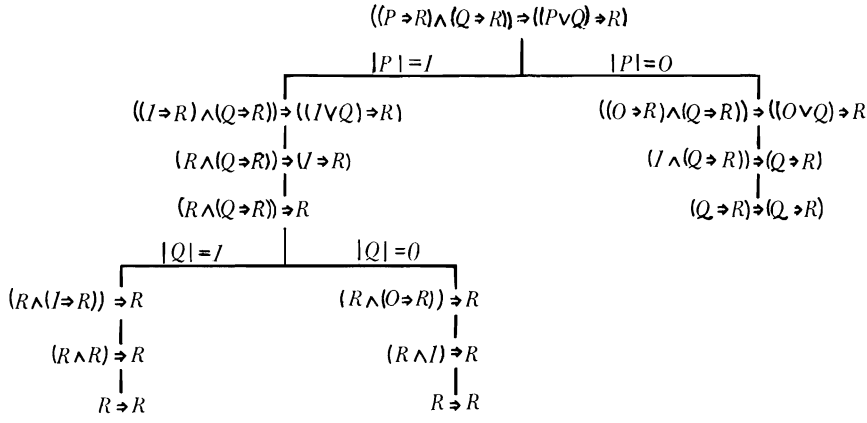


Figura 6.1

Ejemplo. Consideremos la fórmula:

$$\alpha = ((P \Rightarrow R) \wedge (Q \Rightarrow R)) \Rightarrow ((P \vee Q) \Rightarrow R)$$

Sea $|\cdot|$ una valuación tal que $|\alpha| = 0$, es decir,

$$|(P \Rightarrow R) \wedge (Q \Rightarrow R)| = 1 \quad \text{y} \quad |(P \vee Q) \Rightarrow R| = 0$$

Así, dado que $|(P \vee Q) \Rightarrow R| = 0$, se tiene que $|P \vee Q| = 1$ y $|R| = 0$, *i.e.*, $|P| = 1$ o $|Q| = 1$, pero $|R| = 0$. Por otra parte, debido a que $|(P \Rightarrow R) \wedge (Q \Rightarrow R)| = 1$ se obtiene cierta "incoherencia", pues siendo $|R| = 0$, ambos $|P| = |Q| = 0$ para que se satisfaga el antecedente, mientras que por la misma valuación, al menos $|P| = 1$ o $|Q| = 1$. De esta manera, es imposible que exista tal valuación $|\cdot|$ bajo la cual $|\alpha| = 0$, *i.e.*, α es válida. □

Presentemos ahora sí el ejemplo prometido para exhibir la ineficiencia de estos métodos. En efecto, consideremos

$$\Gamma = \{V \Rightarrow \neg P, \neg(\neg A \vee P), \neg(F \Rightarrow B) \Rightarrow \neg Q, \neg(Q \wedge F) \Rightarrow S, \neg V \Rightarrow (A \Rightarrow \neg B)\}$$
$$\text{y} \quad \alpha = R \Rightarrow S.$$

y la consigna es verificar que $\Gamma \models \alpha, \ldots$

Bueno, ¡aclaremos que sólo nos comprometimos a presentar, mas no a desarrollar un ejemplo! ¡En lógica, hay que ser cautos y precisos en el uso de las palabras; y le correspondrá al lector con reservas a que se "aviente la talacha", si sus dudas quiere despejar! □

De los métodos presentados, algunos pueden tener dificultades algorítmicas de aplicarlos a fórmulas que no sean teoremas. De hecho, al emplear la técnica por reducción al absurdo pueden darse casos en que el procedimiento nunca termine, pues la "instrucción de paro" es activada precisamente tan pronto como se obtenga una contradicción.

Algoritmo de Wang

Para concluir esta sección, esbozaremos un último procedimiento, cuya virtud radica en que sí es un algoritmo, pues termina en un número finito de pasos dando el fallo sobre la correción o no de una argumentación y además, permite "automatizar" la demostración de los teoremas de la lógica proposicional. El mismo se debe a Hao Wang, "Towards Mechanical Mathematics", (1960). (*Cf.* [DG]-[TM].)

Consideremos una argumentación de la forma:

$$\beta_1, \beta_2, \ldots, \beta_n \models_T \alpha \tag{1}$$

en la que todas las fórmulas sólo contengan $\neg$, $\wedge$ y $\vee$. Esto siempre puede realizarse con reducir todas las expresiones a las formas normales.

Paso 1. Las comas del lado izquierdo de (1) corresponden a conjunciones y viceversa. De esta manera, (1) es equivalente a la expresión:

$$\beta_1 \wedge \beta_2 \wedge \ldots \wedge \beta_n \models_T \alpha \tag{2}$$

Ahora, supongamos que la fbf α se representa en la forma $\alpha = \alpha_1 \vee \alpha_2 \vee \ldots \vee \alpha_m$ (cosa siempre realizable en virtud de las FND's), con lo cual tenemos la siguiente propiedad, análoga al paso 1.

Paso 2. Las disyunciones del lado derecho de (1) se corresponden con comas y viceversa. De donde, la expresión (1) queda:

$$\beta_1, \beta_2, \ldots, \beta_n \models_T \alpha_1, \alpha_2, \ldots, \alpha_m \tag{3}$$

Observación. Si redujéramos las fórmulas β_i's (o las α_j's) a FNC's (resp., a FND's), podríamos aplicarles la propiedad 1 (resp., 2), y obtener así un número mayor de fórmulas (las componentes de las β_i's (resp., de las α_j's)) y, obviamente, más comas.

La propiedad siguiente se sigue de una reformulación del teorema de la deducción que contempla el empleo de los conectivos lógicos $\neg$, $\wedge$ y $\vee$.

Paso 3. Consideremos la argumentación (3). Entonces, para cualesquiera fórmulas β_i, α_j, con $1 \leq i \leq n$ y $1 \leq j \leq m$, se tiene

$$\begin{gathered}\beta_1, \beta_2, \ldots, \beta_i, \ldots, \beta_n \vDash_T \alpha_1, \alpha_2, \ldots, \alpha_j, \ldots, \alpha_m \\ \text{si y sólo si} \\ \beta_1, \beta_2, \ldots, \neg\alpha_j, \ldots, \beta_n \vDash_T \alpha_1, \alpha_2, \ldots, \neg\beta_i, \ldots, \alpha_m \end{gathered} \tag{4}$$

En otras palabras, toda fórmula de un miembro de una argumentación se "traslada" negada al otro miembro. En particular, esta propiedad se empleará procurando que la traslación tenga el sentido que elimine negaciones.

Paso 4. Instauremos como una regla (usando el teorema de la deducción) la tautología:

$$\vDash ((\gamma_1 \vee \gamma_2) \Rightarrow \delta) \Rightarrow ((\gamma_1 \Rightarrow \delta) \wedge (\gamma_2 \Rightarrow \delta)),$$

es decir,

$$\text{Si } \alpha, \gamma_1 \vee \gamma_2, \omega \vDash_T \delta \text{ entonces } \alpha, \gamma_1, \omega \vDash_T \delta \quad \text{y} \quad \alpha, \gamma_2, \omega \vDash_T \delta \tag{5}$$

Observación. Esta regla no es más que la conversa (*i.e.*, con el condicional invertido, $\Leftarrow$) del ejemplo dado para ilustrar los métodos de Quine y RAA.

Paso 5. Introduzcamos ahora como regla, la tautología (ley distributiva):

$$\vDash (\gamma \Rightarrow (\delta_1 \wedge \delta_2)) \Rightarrow ((\gamma \Rightarrow \delta_1) \wedge (\gamma \Rightarrow \delta_2)),$$

es decir,

$$\text{Si } \gamma \vDash_T \alpha, \delta_1 \wedge \delta_2, \omega \text{ entonces } \gamma \vDash_T \alpha, \delta_1, \omega \quad \text{y} \quad \gamma \vDash_T \alpha, \delta_2, \omega \tag{6}$$

Observación. En los pasos 4 y 5, los símbolos sin subíndices: α, γ, δ, y ω, representan sucesiones de fórmulas, *i.e.*, cadenas de símbolos constituidos de fbf's con o sin comas.

Resumiendo, la aplicación recursiva de los pasos 1–5 nos lleva finalmente a un conjunto (o una conjunción) de $p \times q$ expresiones de la forma:

$$\gamma_i \Rightarrow \delta_j, \text{ para } 1 \leq i \leq p \text{ y } 1 \leq j \leq q \tag{7}$$

donde, $\gamma_i = \ell_1^i \wedge \ldots \wedge \ell_r^i$ y $\delta_j = m_1^j \vee \ldots \vee m_s^j$, siendo las ℓ^i's y m^j's literales.

Sin pérdida de generalidad, podemos considerar que todas las literales son letras proposicionales, pues en caso contrario, simplemente aplicamos el paso 3 las veces necesarias.

Por lo tanto, obtenemos un conjunto de $p \times q$ expresiones *básicas* de la forma:

$$P_1^i \wedge \ldots \wedge P_r^i \Rightarrow Q_1^j \vee \ldots \vee Q_s^j \tag{8}$$

mismas que son fórmulas válidas si y sólo si al menos una misma letra aparece en ambos lados del condicional. Esto es una sencilla consecuencia de combinar las leyes de simplificación: $(P \wedge Q) \Rightarrow P$ y $P \Rightarrow (P \vee R)$, y generalizarlas.

Por consiguiente, contamos así con el siguiente teorema.

Teorema 6.1. *La argumentación:* $\beta_1, \beta_2, \ldots, \beta_n \vDash_T \alpha_1, \alpha_2, \ldots, \alpha_m$ *es correcta si y sólo si en cada una de las* $p \times q$ *argumentaciones básicas:*

$$P_1^i, \ldots, P_r^i \vDash_T Q_1^j, \ldots, Q_s^j \tag{9}$$

ocurre al menos una misma letra proposicional en ambos miembros de la expresión.

Demostración.

Se sigue del planteamiento anterior, validando el paso 3, y aplicando los pasos 1 y 2, así como terminar la justificación al "si y sólo si", en (8). ■

Observaciones:

1. La ineficiencia del algoritmo se deriva de utilizar recursivamente los pasos 4 y 5: para un total de k disyunciones y conjunciones que ocurran en los lados izquierdo y derecho del símbolo, respectivamente, entonces se obtiene 2^k nuevas expresiones.

2. Las fórmulas básicas (8), siendo tautologías (en el caso en que se satisfaga (3)), pueden usarse en calidad de axiomas, y construir un cálculo al implementar

los pasos 1–5 en reglas de inferencia. Para agilizar este cálculo, se incluyen otras reglas de inferencia que involucren a los conectivos lógicos $\Rightarrow$ y $\Leftrightarrow$, efectuando así la reducción a las formas normales en forma implícita. De esta suerte, resulta un cálculo deductivo cuyos axiomas son todas las expresiones de la forma (8) en las que ocurra al menos una misma letra proposicional en ambos lados del condicional, y que participa de la versatilidad de un cálculo tipo deducción natural, al estar provisto de un buen número de reglas de inferencia con las que se indica el manejo (y significado) de los cinco conectivos lógicos: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ y $\Leftrightarrow$. Además, el cálculo resultante se implementa fácilmente como un algoritmo (*cf.* [TM]). En un sentido estricto, este cálculo pertenece a los cálculos de secuencias (*sequent calculi*) al permitirse que la conclusión sea un conjunto y no una sola fórmula, como sucede en la deducción natural.

3. Para recuperar sintácticamente a la lógica proposicional, hemos recorrido un orden creciente de "mecanización" del procedimiento de prueba con detrimento de la participación intelectual, realizado a través de tres cálculos: 1) un *sistema axiomático* tipo Hilbert (Mendelson), 2) un *sistema de deducción natural* tipo Gentzen (Mates) y, finalmente, 3) un *sistema para demostración automática de teoremas* (Wang). El sistema de Mendelson, con su economía en recursos (sólo 2 conectivos lógicos, 3 (esquemas de) axiomas y una regla de inferencia), exige de mayor habilidad por parte del usuario para obtener un resultado; mientras que, en el extremo opuesto, tenemos al sistema de Wang completamente "mecanizado" (un algoritmo), donde los procesos mentales (intuición) se ven relegados, pudiendo, por tanto, prescindirse del usuario humano, reemplazándolo con la "máquina". En estos términos, el sistema de deducción natural es el más "humano" (a lo cual debe su nombre), situándose en una posición intermedia (algo así como "ni tanto que queme al santo, ni tanto que no lo alumbre").

Ejemplo. Verifiquemos si $\Gamma \vDash_T \alpha$ o no, para $\Gamma = \{\neg P \vee Q, P \vee R, Q \vee \neg R\}$ y $\alpha = Q \vee \neg S$.

En efecto, y usando 2,

$$\neg P \vee Q,\ P \vee R,\ Q \vee \neg R \vDash_T Q, \neg S$$

por 3,

$$S, \neg P \vee Q, P \vee R, Q \vee \neg R \vDash_T Q$$

por 4,

$$S, \neg P, P \vee R, Q \vee \neg R \vDash_T Q, \quad \text{y} \quad S, \boxed{Q}, P \vee R, Q \vee \neg R \vDash_T \boxed{Q}$$

por 3,

$$S, P \vee R, Q \vee \neg R \vDash_T Q, P$$

por 4,

$$S, \boxed{P}, Q \vee \neg R \vDash_T Q, \boxed{P} \quad \text{y} \quad S, R, Q \vee \neg R \vDash_T Q, P$$

por 4,

$$S, R, \boxed{Q} \vDash_T \boxed{Q}, P \quad \text{y} \quad S, R, \neg R \vDash_T Q, P$$

por 3,

$$S, \boxed{R} \vDash_T Q, P, \boxed{R}$$

Por lo tanto, por el Teorema 6.1, $\Gamma \vDash_T \alpha$. □

Ejercicios

***1.** Proporcione un algoritmo para analizar si dada una expresión en $\mathscr{L}_0$ es o no una fbf de la lógica proposicional.

2. Construya un algoritmo que genere tablas de verdad para fbf's.

3. Complete la demostración del Teorema 6.1.

4. Pruebe si $\Gamma \vDash_T \alpha$ o no, usando los métodos de Quine, reducción al absurdo y el algoritmo de Wang, para:

a. $\Gamma = \{P\}$, $\alpha = \neg P \Rightarrow Q$
b. $\Gamma = \{P, \neg P, Q\}$, $\alpha = R$
c. $\Gamma = \{P \vee Q, P \Rightarrow R, Q \Rightarrow S\}$, $\alpha = S \vee R$
d. $\Gamma = \{R \Rightarrow Q, Q \Rightarrow \neg P, P, R \vee (T \wedge S)\}$, $\alpha = T \wedge S$
e. $\Gamma = \{B \Rightarrow \neg C, C \vee Q, \neg(T \Rightarrow B) \wedge Q, \neg\neg T \Rightarrow (S \vee C), B\}$, $\alpha = T$

6.3 Problemas NP-completos y satisfacibilidad

En esta sección tendremos como objetivo inmediato responder a ¿qué significa decir que "un algoritmo F es más difícil de computar que uno G"? Para ello, se requiere una *medida* del grado de *complejidad* de los algoritmos. Entre las medidas de complejidad, parece ser claro que el *tiempo* y el *espacio*—principalmente el tiempo— son de las más importantes. A la cantidad de memoria usada por un algoritmo durante el proceso se denomina el *espacio* requerido por el algoritmo.

El tiempo resulta ser un mejor parámetro para medir la bondad de un algoritmo, ya que el tiempo de ejecución es equivalente al costo económico por uso-máquina, entre otras cosas. Ahora, como el tiempo real requerido por una computadora es proporcional al número de operaciones básicas realizadas por ella, se suele llamar *tiempo de computación* no al físico (real), sino al número de operaciones elementales. (Aquí se supone que todas las operaciones básicas toman el mismo tiempo.)

Presentaremos a continuación dos algoritmos que resuelven un mismo problema a fin de calcular y comparar sus tiempos de ejecución.

Problema (de búsqueda): "Dada una lista de palabras $L = \{a_1, a_2, \ldots, a_n\}$, averiguar si una palabra X pertenece o no a L. En caso afirmativo, dé además su localización en L."

Solución 1. Mejorando otros métodos de *búsqueda secuencial*, daremos una interpretación sobre la salida (respuesta) del algoritmo (dada por la localización j del casillero a_j en L): si $j = 0$, significa que la palabra X no se encuentra en L; mientras que, $j \neq 0$ nos reporta no sólo una respuesta afirmativa, sino además la posición de X en L. Sobre esta base, extendemos L a una nueva lista L' que cuente con la localidad a_o, *i.e.*, $L' = \{a_o, a_1, a_2, \ldots, a_n\}$. Así, el algoritmo es el siguiente:

Algoritmo 1.

PROC BUSQSEC(a, n, X, j)
1. HAZ ($a_o \leftarrow X$; $j \leftarrow n$).
2. SI ($X = a_j$) ENTONCES (RESPONDE j). FIN.
3. HAZ ($j \leftarrow j - 1$); Y VOLVER al paso 2.

Notación. El símbolo "⟵" representa la asignación del valor situado en la derecha en la variable de la izquierda, *e.g.*, la expresión "$j \longleftarrow j - 1$" significa un decremento de la variable j. Aparte de los conectivos lógicos "Y" y "SI... ENTONCES", hemos empleado las instrucciones "HAZ (...)", como orden, "VOLVER", como retorno incondicionado, "RESPONDE", como mecanismo de salida, y "FIN", como paro.

Análisis de complejidad: Considerando solamente el tiempo empleado en el paso 2, en el *peor* de los casos ($a_1 = X$), tenemos qué comparar la palabra X con las n entradas de la lista antes de que la búsqueda termine. Esto se traduce en una búsqueda exhaustiva hasta dar con la palabra buscada; para una lista de, *e.g.*, 10^6 palabras se requerirían así 10^6 comparaciones (unidades de tiempo) como máximo.

Solución 2. Supongamos ahora que la lista L se nos proporciona ordenada, digamos $a_1 < a_2 < \ldots < a_n$. Podemos explotar esta condición adicional para diseñar un "mejor" algoritmo que el anterior. (Un conjunto con estructura provee de mayor información: "El todo es mayor que la suma de sus partes"[2].) La estrategia es: 1. comparar X con la entrada situada a la mitad de L, a_m; 2. si X es (numérica o alfabéticamente) menor que a_m, se descarta (¡para fines de búsqueda subsecuente!) la mitad de los elementos de L: $a_m, a_{m+1}, \ldots, a_n$; 3. procediendo análogamente con el caso de X mayor que a_m, se descartan entonces las primeras m entradas de esta lista. De esta forma, una sola comparación ha reducido el problema de búsqueda a la mitad. Aplicando esta misma técnica con la mitad remanente de la lista, y así sucesivamente, el algoritmo eventualmente dará su fallo sobre la localización o ausencia de la palabra X en L.

Algoritmo 2.

PROC BUSQBIN (a, n, X, j, l, r).
1. HAZ $(l \leftarrow 0; r \leftarrow n + 1)$.
2. HAZ $(j \leftarrow \lfloor (l + r)/2 \rfloor)$ || aquí, $\lfloor\ \rfloor$ es la función mayor entero menor que ||
3. SI $(j = l)$ ENTONCES (RESPONDE 0). FIN.
4. SI $(X = a_j)$ ENTONCES (RESPONDE j). FIN.
5. SI $(X < a_j)$ ENTONCES (HAZ $(r \leftarrow j)$); Y VOLVER al paso 2.
6. SI $(X > a_j)$ ENTONCES (HAZ $(l \leftarrow j)$); Y VOLVER al paso 2.

[2]Frase debida a Aristóteles, reinstaurada actualmente por los teóricos de sistemas.

Notación. Idéntica al algoritmo 1, salvo "$\| \ldots \|$," que significa un comentario dirigido sólo al usuario, no a la máquina.

Análisis de complejidad: Ahora el análisis es ligeramente más complicado. El algoritmo procede eliminando la mitad restante de la lista después de la ejecución del paso 4. En el peor caso ($X = a_1$ o $X = a_n$), el máximo número de ejecuciones del paso 4 será k, donde k es el menor entero tal que 2^k es mayor que n. De aquí que la función de tiempo, $T(n)$, sea proporcional a $\log_2(n)$, *i.e.*, puesto que $2^k > n$, entonces $k = T(n) = c \log_2(n)$, con $c > 0$, una constante. Con este algoritmo, para una lista de 10^6 palabras sólo se requerirán 20 comparaciones como máximo (si $c = 1$). □

Un análisis comparativo temporal de los dos métodos respecto a una lista ordenada se "inclina" en favor de la búsqueda binaria. Todo el análisis se centró siempre sobre el denominado *peor caso*; existen otros conceptos para tratar la complejidad de los algoritmos, como son los casos *mejor* y *promedio.* (Para un estudio más extenso de este tema y otros afines, consulte *e.g.* [HS].)

Analizar así un algoritmo se reduce a un problema de combinatoria. Sin embargo, como puede resultar difícil hacer el cálculo del número de operaciones realizadas, el análisis se lleva vía el comportamiento de la función tiempo, $T(n)$, en lugar de su expresión exacta. El comparar algoritmos sólo resulta ventajoso si el volumen de datos en grande. De aquí las definiciones que hemos de usar.

> **Definición.** Sean $f, g: \mathbb{N} \to \mathbb{R}$ dos funciones. Decimos que g *domina asintóticamente* a f si y sólo si existen constantes $k \geq 0$ y $m \geq 0$ tales que para toda $n \geq m$, se verifica que $|f(n)| \leq k|g(n)|$. Notación: $f \prec g$.

En términos de los algoritmos F y G a comparar, si f y g son las funciones respectivas de tiempo, tendremos que el algoritmo F tardará k veces más que el G en resolver un mismo problema.

> **Definición.** El *orden de* f es el conjunto de todas las funciones dominadas por f y lo denotamos $\mathcal{O}(f) := \{g: \mathbb{N} \to \mathbb{R} \text{ y } g \prec f\}$. Si $g \in \mathcal{O}(f)$, decimos que g es de orden f.

Observaciones:

1) Se tiene que $g \in \mathcal{O}(f)$, aun cuando k sea una fracción propia.

2) La notación $\mathcal{O}(f)$ es un medio para medir el desempeño de un algoritmo, siendo una *cota superior* para el crecimiento de la función tiempo.

Ejemplo. Consideremos las funciones: $\log(n)$ (en base 2), n, n^2, 2^n y n^n. De la figura 6.2, se observa que existe un valor de n a partir del cual 2^n y n^n son siempre mayores que $\log(n)$, n y n^2. Estas últimas funciones tienen un crecimiento más lento que las primeras dos. Además, en términos de los órdenes de complejidad, no es difícil probar que se dan las contenciones propias:

$$\mathcal{O}(\log(n)) \subset \mathcal{O}(n) \subset \mathcal{O}(n^2) \subset \mathcal{O}(2^n) \subset \mathcal{O}(n^n)$$

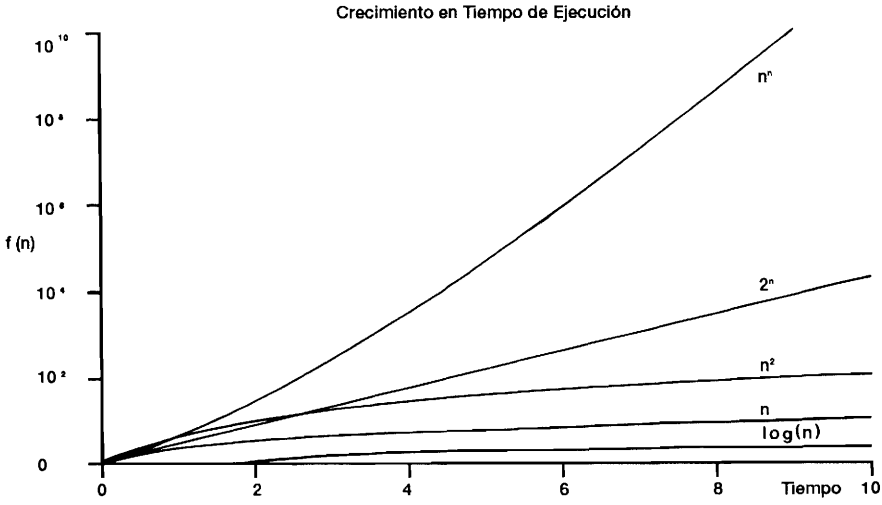


Figura 6.2

Generalizando la comparación anterior a todas las funciones, dio pauta para escindir al conjunto de las funciones en dos grupos:

1) las funciones de orden polinomial, constituido por todos los polinomios y funciones de órdenes menores (bajo contención, $\subset$); y

2) las funciones de orden exponencial, todas aquellas funciones de orden mayor o igual al exponencial.

Ahora, si las funciones representan las funciones del tiempo de ejecución asociadas a los algoritmos, tenemos la definición siguiente.

> **Definición.** Un algoritmo es *eficiente* si y sólo si su función de tiempo es de orden polinomial. Un algoritmo cuya función de orden es exponencial se dice *ineficiente*.

Otra cota no menos importante es la *cota inferior*.

> **Definición.** $f \in \Omega(g)$, y se lee "f pertenece a omega de g", si y sólo si existen constantes $k > 0$ y $m \geq 0$ tales que para toda $n > m$, $|f(n)| \geq k|g(n)|$.

Con base en los tiempos de crecimiento, es razonable considerar que un problema con una solución algorítmica eficiente sea un problema *tratable*, mientras que uno con una solución ineficiente sea *intratable*. Esta (controversial) división parece sustentarse como un hecho más bien empírico, dado que aunque cualquiera dudaría que un algoritmo de un orden $n^{1\,000}$ sea mejor a uno de orden $2^{0.001n}$ (al menos para valores de $n < 24{,}549{,}171$), este último orden no surge en un problema natural. Otro punto a favor para esta división, son ciertos problemas para los que sus cotas inferiores rebasan por mucho la capacidad de cualquier computadora, aun las no construidas todavía, para resolverlos.

> **Definición.** La clase de los problemas $\mathscr{P}$ (*determinísticos Polinomiales*) la constituyen todos los problemas de decisión tratables, *i.e.*, aquéllos que cuentan con algoritmos deterministicos eficientes.

Hallar una cota superior para la complejidad temporal de un problema es relativamente fácil de obtener: dar un algoritmo F que resuelva el problema y de éste determinar $\mathscr{O}(f)$, donde f es la función de tiempo de F. Es más, resulta común asegurar que un algoritmo F es mejor que otro G, si $\mathscr{O}(f) \subset \mathscr{O}(g)$. Sin embargo, hallar cotas inferiores es muy difícil en general (hay muchos problemas abiertos

al respecto). La dificultad reside en sí en que una cota inferior requiere considerar *todos* los algoritmos que resuelven el problema, mientras que una cota superior se obtiene construyendo un algoritmo particular y analizando su desempeño temporal. Decimos que un problema algorítmico es *cerrado* cuando sus cotas superior e inferior coinciden. Caso contrario se habla de un *brecha algorítmica*, *i.e.*, cuando la mínima (mejor) cota superior conocida difiere de la máxima (mejor) cota inferior conocida. De aquí se derivó la necesidad de concebir los problemas $\mathcal{NP}$, clase para los cuales la brecha algorítmica es muy grande: algunos de estos problemas tienen $\Omega(n)$ *vs.* $\mathcal{O}(c^n)$, con $c > 1$. Así, la meta fijada es minimizar esta brecha.

Hemos corroborado que probar la satisfacibilidad de una fórmula es una tarea *non grata*, conforme el número de letras proposicionales se incrementa. Sin embargo, supongamos que nos dan de antemano una valuación asegurándonos que sí satisface una fórmula (*e.g.*, al copiar en el examen); por si las dudas sería sencillo verificar que sí es la respuesta correcta: retomando, *e.g.*, el algoritmo de Quine, sólo habría que recorrer un camino en el árbol asociado para saber la respuesta, en lugar de proceder por "ensayo y error" para dar con ella. Así, al exhibir una respuesta, ésta *certifica* la satisfacibilidad de la fórmula (problema), relegando la tarea de verificarlo a un proceso que sólo toma un tiempo de orden polinomial.

Resulta un hecho el que hay muchos problemas importantes de muy diversas áreas e intereses, para los cuales el proceso: "si contáramos con la respuesta correcta, corroborar que en efecto lo es *no lleva tiempo*", es común a todos ellos. Pero, aunque suele ser "bonito soñar", ¿quién nos proporcionará la respuesta en cuestión?:

> ¿La obtendremos por gracia de la Divina Providencia? ¡un milagro! o
> ¿invocaremos al Diablo?, y ¡haremos magia negra!

Independientemente de las preferencias, para el caso es lo mismo, necesitamos de la *magia* para dar *pronósticos* acertados. En términos técnicos, requerimos de *algoritmos no determinísticos.* Al calcular el tiempo empleado por un algoritmo no determinístico para completar su objetivo, se *omite* el tiempo usado para generar el pronóstico.

> **Definición.** El *tiempo* requerido por un algoritmo no determinístico para proceder sobre una entrada dada es el mínimo número de pasos que se necesitan para alcanzar una ejecución exitosa, en el supuesto caso de que

exista una sucesión de decisiones que den lugar a tal ejecución. Ahora, un algoritmo no determinístico F es de orden $f(n)$ si y sólo si existen ciertas constantes $k \geq 0$ y $m \geq 0$ tales que para todas las entradas de longitud n que producen una ejecución exitosa, donde $n > m$, el tiempo empleado es a lo más $kf(n)$.

Definiendo el tiempo de esta manera, se captura la situación de un "adivino" que siempre acierta y que siempre toma la mejor respuesta (el pronóstico que permite a la parte determinista del algoritmo resolver el problema más rápidamente).

Definición. La clase de los problemas $\mathcal{NP}$ (*No determinísticos Polinomiales*) la constituye todos aquellos problemas de decisión para los cuales existen algoritmos no determinísticos de orden polinomial.

Debido a que los algoritmos determinísticos (D) son un caso particular de los no determinísticos (ND) (¿Por qué?), concluimos que $\mathcal{P} \subseteq \mathcal{NP}$. Lo que hasta la fecha se ignora, y, parafrasendo a Horowitz-Sahni [HS],

"ha venido a ser el más famoso problema abierto en la ciencia computacional es si

$$\mathcal{P} = \mathcal{NP} \quad \text{o} \quad \mathcal{P} \neq \mathcal{NP}."$$

Hasta ahora, todo parece indicar que $\mathcal{NP}$ no está incluido en $\mathcal{P}$, *i.e.*, es poco probable que existan algoritmos determinísticos efectivos, aun no descubiertos, para la clase de los $\mathcal{NP}$, esto en virtud del enorme (e infructuoso) esfuerzo invertido en encontrarlos. Sin embargo, una prueba de que $\mathcal{P} \neq \mathcal{NP}$ ha resultado también tan elusiva, que debe requerir de técnicas todavía inexistentes.

Verifiquemos ahora que el problema de satisfacibilidad está en $\mathcal{NP}$. Para ello, simplemente exhibimos el siguiente algoritmo ND de orden polinomial:

Algoritmo ND *para satisfacibilidad.*

PROC SAT(E, n)
1. BOOLEANA $x(n)$
2. PARA $i = 1$ HASTA n
3. HAZ ($x_i \leftarrow$ELIGE$\{1, 0\}$)
4. REPITE
5. SI ($E(x_1, x_2, \ldots, x)$) ES 1) ENTONCES (RESPONDE "ÉXITO"). FIN
6. RESPONDE "FALLO"
7. FIN

Notación. La instrucción "BOOLEANA" es para declarar que la variable $x(n)$ sólo toma los valores 0 o 1; la instrucción "PARA $i = 1$ HASTA n ... REPITE" constituye lo que se denomina un *ciclo,* e indica que lo que se representa con "..." debe realizarse n veces; "ELIGE" viene a ser nuestra instrucción no determinística (¡para hacer magia!); $E(x_1, x_2, \ldots, x_n)$ es la fórmula a evaluar; las demás instrucciones fueron mencionadas anteriormente.

Análisis de complejidad: Con este algoritmo se asigna una instancia al vector booleano $(x_1, x_2, \ldots, x_n)$ mediante un ciclo, y se evalúa determinísticamente la fórmula $E(x_1, x_2, \ldots, x_n)$. El tiempo no determinístico es $\mathcal{O}(n) : \mathcal{O}(n)$ para el ciclo $+\mathcal{O}(|E|)$, donde $|E|$ = longitud de E; los tiempos para las instrucciones ELIGE y RESPONDE son tomados como $\mathcal{O}(1)$.

A la pregunta de cómo son los problemas más difíciles en $\mathcal{NP}$, S. Cook ("The Complexity of Theorem Proving Procedures", (1971)) probó que el problema de satisfacibilidad es tan difícil como cualquier otro problema en $\mathcal{NP}$. Para aclarar esta afirmación, necesitamos de algunos conceptos previos.

> **Definición.** Sean P_1 y P_2 dos problemas. Decimos que P_1 se *reduce* a P_2, y se denota $P_1 \propto P_2$, si y sólo si hay una manera de resolver P_1 mediante un algoritmo determinístico efectivo usando para ello un algoritmo determinístico efectivo que resuelve a P_2.

Esta definición implica que de resolverse P_2 con un algoritmo polinomial, también P_1 se resuelve con un algoritmo polinomial.

Observación. La relación dada por $\propto$ es transitiva, *i.e.*, si $P_1 \propto P_2$ y $P_2 \propto P_3$, entonces $P_1 \propto P_3$.

> **Definición.** Un problema P_1 es *$\mathcal{NP}$-duro* si y sólo si todo problema P_2 en $\mathcal{NP}$ se reduce a P_1 ($P_2 \propto P_1$). Un problema P_1 es *$\mathcal{NP}$-completo* si y sólo si $P_1 \in \mathcal{NP}$ y P_1 es *$\mathcal{NP}$-duro.*

De esta suerte, la clase de los problemas $\mathcal{NP}$-completos conforman una especie de élite dentro de la clase $\mathcal{NP}$, por la propiedad de que si uno de ellos tuviera un algoritmo eficiente, entonces todo problema en $\mathcal{NP}$ podría resolverse eficientemente.

El problema de satisfacibilidad (SAT) fue el primero en probarse $\mathcal{NP}$-completo (Cook, 1971); este hecho, debido al gran poder expresivo de la lógica proposicional

(como lenguaje) para representar los problemas de decisión, revela la importancia del mismo. Para este efecto, Cook demostró que la computación realizada por un algoritmo ND, A, en cada entrada puede codificarse como un problema de satisfacción, y que la complejidad del problema de satisfacción resultante está polinomialmente relacionado con el tiempo requerido por A para resolver el problema. El teorema de Cook establece que:

"El problema de satisfacibilidad está en $\mathscr{P}$ si y sólo si $\mathscr{P} = \mathscr{NP}$"

Dado que el problema SAT está en $\mathscr{NP}$, la implicación "Si $\mathscr{P} = \mathscr{NP}$ entonces SAT está en $\mathscr{P}$" es más bien trivial. El converso se basa en cómo obtener de cualquier algoritmo ND que esté en $\mathscr{NP}$, A, y una entrada I, una fórmula $\alpha(A, I)$ tal que $\alpha(A, I)$ es satisfacible si y sólo si A tiene una ejecución exitosa con I.

Veamos un ejemplo sencillo, pero no trivial, para ilustrar cómo se realiza el proceso de reducción entre problemas.

> **Definición.** Una fórmula α está en *k-forma normal conjuntiva* (k-FNC) si y sólo si es una FNC cuyas componentes tienen a lo más k literales. El *problema de k-satisfacibilidad* (k-SAT) consiste en decidir si una fórmula en k-FNC es satisfacible.

Ejemplo. Reduciremos el denominado *problema de los tres colores* (3-COL) al problema 3-SAT. El problema *cromático* general consiste en colorear un mapa de países con un número fijo de colores de tal manera que dos países contiguos no compartan el mismo color. Para el caso de dos colores, es fácil decidir: se puede siempre que en el mapa no hayan puntos donde un número impar de países concurran. El problema de cuatro colores siempre tiene solución positiva, por lo que ni siquiera es necesario mirar al mapa. (Problema clásico resuelto por K. Appel & W. Haken en 1976, *cf*. [SK].) Sin embargo, para el problema de tres, no se ha hallado ningún algoritmo eficiente que lo resuelva. Este problema está en $\mathscr{NP}$, pues una coloración correcta del mapa certifica una respuesta afirmativa. Probemos entonces que 3-COL $\propto$ 3-SAT. En efecto, describiremos un algoritmo A, cuya entrada sea un mapa M, del que obtendremos una fórmula α, tal que M puede colorearse con tres colores si y sólo si α es satisfacible. Además, el proceso se realiza eficientemente, lo que implica que el número de símbolos en α es una función polinomial del número de países de M. Consideremos los tres colores: A-azul, R-rojo y V-verde; y un mapa constituido con los países $P_1, P_2, \ldots, P_n$.

La expresión "P_i-es-A", digamos, representa la proposición "el país P_i es azul". La fórmula en cuestión, se construye con la conjunción de dos subfórmulas: la primera establece que cada país P_i tiene exactamente un color:

$$\bigwedge_{1\leq i\leq n} (((P_i\text{-es-A}) \wedge \neg(P_i\text{-es-R}) \wedge \neg(P_i\text{-es-V})) \vee (\neg(P_i\text{-es-A}) \wedge (P_i\text{-es-R}) \wedge \neg(P_i\text{-es-V})) \vee (\neg(P_i\text{-es-A}) \wedge \neg(P_i\text{-es-R}) \wedge (P_i\text{-es-V})))$$

y la segunda, afirma que dos países cualesquiera no pueden compartir el mismo color:

$$\bigwedge_{\substack{ij \\ P_i \text{ contiguo } P_j}} (\neg((P_i\text{-es-A}) \wedge (P_j\text{-es-A})) \vee \neg((P_i\text{-es-R}) \wedge (P_j\text{-es-R})) \vee \neg((P_i\text{-es-V}) \wedge (P_j\text{-es-V})))$$

La longitud de la fórmula es así $\mathcal{O}(n^2)$: pues por la primera expresión es de orden lineal en n, *i.e.*, $\mathcal{O}(n)$; mientras que por la segunda es $\mathcal{O}(n)$, debido a las combinaciones de las parejas P_i y P_j de los países adyacentes.

Así, tenemos que 3-COL $\propto$ 3-SAT. No es difícil probar la reducción recíproca 3-SAT $\propto$ 3-COL (ejercicio 6). Con base en esto último y de que 3-SAT es $\mathcal{NP}$-duro (ejercicio 7), por la transitividad de la relación $\propto$, se sigue que 3-COL es $\mathcal{NP}$-duro. Por lo tanto, 3-COL es $\mathcal{NP}$-completo, pues está en $\mathcal{NP}$. □

Por consiguiente, una manera equivalente de determinar que un problema P es $\mathcal{NP}$-duro es probando que el problema SAT se reduce a P. El problema SAT es así el *representante* por excelencia de la clase de los problemas $\mathcal{NP}$-completos. Ahora que, para probar que un problema P es $\mathcal{NP}$-duro, la mejor estrategia es reducir un problema $\mathcal{NP}$-duro, ya previamente probado, al problema P, y entonces aplicar la transitividad de $\propto$. De exhibir un algoritmo en $\mathcal{NP}$ que resuelva P, bastará entonces para demostrar que P es $\mathcal{NP}$-completo. A la clase de los $\mathcal{NP}$-completos pertenece más de mil problemas de gran interés tanto teórico como práctico, como son los problemas de decisión: *del agente viajero, del camino hamiltoniano, de la programación entera, de los tres colores,* etc.

Prácticamente, una prueba de que un problema es $\mathcal{NP}$-completo equivale a expedir un certificado sobre su intratabilidad, *i.e.*, es un argumento fuerte para abandonar cualquier esfuerzo ulterior por hallarle un algoritmo eficiente para su solución.

Por último, a diferencia de SAT, el problema de la validez (VAL) de una fórmula puede pensarse como más difícil, ya que para este problema ni siquiera se ha probado que está en $\mathcal{NP}$: cualquier algoritmo (determinístico o no) tiene que proceder con todas las valuaciones posibles para decidir sobre la validez. Sin embargo, VAL es importante por otras razones. Una de ellas, similar al teorema de Cook, establece que "Si VAL está en $\mathcal{P}$ entonces $\mathcal{P} = \mathcal{NP}$"; y otra es "Si VAL $\notin \mathcal{NP}$ entonces $\mathcal{P} \neq \mathcal{NP}$". De aquí se explica todo el esfuerzo invertido en analizar la complejidad del problema de validez de las fórmulas proposicionales.

Ejercicios

1. Dé una argumentación que respalde la optimalidad del algoritmo de búsqueda binaria. ¿De aquí se sigue que el problema de búsqueda es cerrado? Justifique.

2. Indique el sentido de la contención para cada pareja de complejidades:

a. $\mathcal{O}(2^{10 \log n})$, $\mathcal{O}(n^{11})$

b. $\mathcal{O}(4 \log(n^n))$, $\mathcal{O}(n^3)$

c. $\mathcal{O}((2n)!)$, $\mathcal{O}(2^{2^n})$

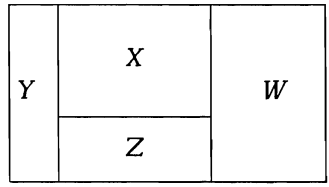
d. $\mathcal{O}(2^n n!)$, $\mathcal{O}((n^3)!)$

e. $\mathcal{O}\left(3^{(\log n)^4}\right)$, $\mathcal{O}\left(4^{(\log n)^3}\right)$

3. Explique por qué $\mathcal{P} \subset \mathcal{NP}$.

4. Pruebe que un mapa puede ser coloreado con dos colores cuando en cada vértice de frontera concurren un número par de países. (Sug. el 'resto del mundo' es considerado como un país más; y si un país toca un punto de frontera dos veces, se cuenta como doble).

5. Construya una fórmula satisfacible α que corresponda al mapa siguiente, en un problema 3COL:



6. Pruebe que 3-SAT $\propto$ 3-COL.

***7.** Pruebe que 3-SAT es $\mathcal{NP}$-completo.

Capítulo 7
Lógica de predicados: enfoque semántico

La lógica matemática es una rama de las matemáticas cuya relación con el análisis y crítica del pensamiento es la misma que la que tiene la geometría con la ciencia del espacio.

Haskell B. Curry

7.1 Introducción

Consideremos ahora el argumento clásico:

Todos los hombres son mortales
Sócrates es hombre
Luego, Sócrates es mortal.

Si tratamos de expresar este argumento en un lenguaje formal de proposiciones como los estudiados en los capítulos anteriores, vemos que la única manera de traducirlo es sustituyendo la primera premisa por una letra proposicional, digamos A, la segunda premisa como otra letra proposicional, digamos B y la conclusión como una tercera letra proposicional, C. Es evidente que C no es una consecuencia tautológica de A y B, por lo que tenemos un argumento correcto que no es rescatado por la lógica proposicional.

El modelo del pensamiento deductivo que hemos estado estudiando evidentemente no es lo suficientemente fino como para reconocer todos los argumentos correctos en español. Si tenemos un argumento en español y traducimos todo al lenguaje del cálculo de enunciados obtendremos un conjunto Σ de premisas y una fórmula ϕ que representa a la conclusión. Si ϕ es consecuencia tautológica de Σ entonces podemos asegurar que el argumento es correcto, pero si no lo es entonces no podemos estar seguros, podría tratarse de un argumento como el que dimos al principio, cuya corrección no es rescatada por la lógica de enunciados.

El problema con el argumento que dimos es que su corrección no está basada en la manera en que se relacionan, desde el punto de vista de funciones de verdad, las proposiciones involucradas en el argumento. Su corrección se basa en la estructura interna de las proposiciones y en el significado que tiene la palabra "todos".

Necesitamos, pues, refinar el lenguaje de proposiciones para construir lenguajes formales que sean lo suficientemente ricos para expresar, por ejemplo, que todos o algunos de los miembros de una cierta clase tienen una propiedad dada. Los lenguajes que vamos a obtener son los llamados *lenguajes de primer orden.* (El por qué se llaman de primer orden se verá más adelante.)

Queremos preservar todo lo que habíamos obtenido con los lenguajes proposicionales, de modo que los nuevos lenguajes formales serán extensiones de los lenguajes proposicionales. Para poder reflejar la estructura interna de las proposiciones y hablar de individuos y sus propiedades necesitamos introducir nuevos símbolos que representen individuos, propiedades de individuos, relaciones entre individuos y a los conceptos "todos" y "algunos".

Los individuos serán representados por letras minúsculas, las propiedades de individuos y las relaciones entre ellos serán representadas por letras mayúsculas y los conceptos "todos" y "algunos" serán representados por los símbolos $\forall$ y $\exists$ respectivamente.

Todo esto se hará más preciso en la siguiente sección, de momento sólo queremos motivar la definición de estos lenguajes y dar algunos ejemplos, para que su definición rigurosa no parezca totalmente arbitraria.

Regresemos al argumento del principio y tratemos de traducir las proposiciones que aparecen en él.

La segunda premisa es "Sócrates es hombre", que involucra a un individuo y a una propiedad. Para nombrar al individuo usamos una letra minúscula, digamos s; y para nombrar a la propiedad utilizamos una letra mayúscula, digamos H. Para expresar que el individuo denotado por "s" tiene la propiedad denotada por "H"

escribimos Hs. Si M denota la propiedad de ser mortal entonces la conclusión puede ser traducida por Ms.

Lo único que nos falta traducir es la primera premisa, para lo cual necesitamos el símbolo $\forall$. Decir "todos los hombres son mortales" es asegurar que todo individuo que tenga la propiedad de ser hombre también tiene la propiedad de ser mortal. Esto quedará traducido en nuestro lenguaje como $\forall x(Hx \Rightarrow Mx)$. Aquí la letra minúscula "x" representa individuos, pero no un individuo particular, como en el caso de "s", sino un individuo arbitrario. La letra "x" es una *variable individual*, mientras que la letra "s" es una *constante individual.*

Usando estas interpretaciones que hemos dado a las letras y la interpretación normal que se da a los conectivos de cualquier lenguaje de proposiciones podemos traducir las siguientes expresiones:

$\neg Ms$	Sócrates no es mortal
$\neg Hs$	Sócrates no es hombre
$\neg Ms \wedge \neg Hs$	Sócrates no es ni hombre ni mortal
$\neg\forall x(Hx \Rightarrow Mx)$	No todo hombre es mortal
$\exists x Mx$	Algún individuo es mortal
$\exists x(Hx \wedge \neg Mx)$	Algún hombre es inmortal
$\forall x(Hx \Rightarrow \neg Mx)$	Todo hombre es inmortal

En matemáticas se trabaja con lenguajes muy parecidos con los que trabajaremos en éste y el siguiente capítulo. Consideremos, por ejemplo, a los números naturales como un conjunto ordenado que tiene un primer elemento. Al orden se le denota tradicionalmente con el símbolo "$\leq$" y al primer elemento con el símbolo "0". Siguiendo las convenciones que hemos establecido para los lenguajes de primer orden, si estamos hablando de los números naturales los individuos del discurso serán justamente los números naturales, de modo que para denotar al 0 tendremos que usar alguna letra minúscula, digamos "c" y cuando se escriba $\forall x$ o $\exists x$ se entenderá "todo número natural" o "algún número natural", respectivamente. La relación de orden entre los números naturales es una relación binaria, de modo que la representaremos por una letra mayúscula "P" seguida de dos letras minúsculas de tal forma que "Pxy" se leerá: "x es menor o igual a y". Con estas convenciones podemos traducir las siguientes afirmaciones acerca de los números naturales como conjunto ordenado:

0 es menor o igual que cualquier número natural
$\forall x(Pcx)$

No existe un último número natural
$\neg\exists x\forall y(Pyx)$

Para todo número natural hay un natural mayor o igual a él
$\forall x\exists y(Pxy)$

La relación de orden en los naturales es transitiva
$\forall x\forall y\forall z((Pxy \wedge Pyz) \Rightarrow Pxz)$

La relación de orden en los naturales es reflexiva
$\forall x\, Pxx$

También podemos efectuar el proceso inverso, esto es, dada una expresión con los símbolos que hemos escogido, podemos traducirla al castellano. Por ejemplo:

$\exists x\, Pcx$
Hay algún natural mayor o igual a 0

$\forall x\forall y\, Pxy$
Cualquier número natural es menor o igual a cualquier otro

$\exists x\exists y\, Pxy$
Algún natural es menor o igual a algún otro

Aquí se puede observar que si se quisiera expresar la propiedad antisimétrica de la relación de orden se necesitaría una letra nueva para expresar la relación de igualdad. También sería conveniente poder hablar de la suma y el producto de números naturales, que no son relaciones entre números sino operaciones. Ambas cosas se pueden hacer, se puede introducir una letra que represente la relación de igualdad y puede considerar la suma y el producto como relaciones ternarias S y Q de tal modo que $(m, n, r) \in S$ si y sólo si $m + n = r$ y $(m, n, r) \in Q$ si y sólo si $mn = r$, donde $m, n, r \in \mathbb{N}$.

Sin embargo la relación de identidad y las operaciones son tan importantes en matemáticas que en la definición de lenguajes de primer orden que veremos en la siguiente sección vamos a introducir un símbolo especial para la identidad y vamos a introducir símbolos funcionales que representen funciones en el dominio de discurso.

7.2 Lenguajes de primer orden

Un *lenguaje de primer orden* $\mathscr{L}$ consta de los siguientes símbolos:

Símbolos lógicos

1. Un conjunto numerable de variables individuales:

$$x_1, x_2, \ldots, x_n, \ldots$$

2. Conectivos lógicos: $\neg$ y $\Rightarrow$.
3. Símbolo de igualdad (opcional): $\approx$.
4. Paréntesis:) y (.

Símbolos no lógicos o parámetros

1. Cuantificadores: $\forall$ y $\exists$.
2. Predicados: Para cada $n \in \mathbb{N}$, un conjunto (posiblemente vacío) de símbolos de predicado n-ario $P_1^n, P_2^n, \ldots$
3. Constantes individuales: Un conjunto (posiblemente vacío) de símbolos de constante $c_1, c_2, \ldots$
4. Símbolos funcionales: Para cada $n \in \mathbb{N}$, un conjunto (posiblemente vacío) de símbolos funcionales n-arios $f_1^n, f_2^n, \ldots$

Observaciones:

1. Los símbolos de $\mathscr{L}$ se dividen en lógicos y no lógicos porque desde el punto de vista semántico hay una diferencia entre ellos: los símbolos lógicos siempre serán interpretados de la misma manera mientras que los no lógicos podrán tener significados distintos de acuerdo con la interpretación en la cual se esté trabajando.

2. El símbolo de la igualdad, aunque es un predicado binario, se ha puesto como símbolo lógico porque se quiere que su interpretación sea siempre como la relación de identidad entre los individuos del dominio de discurso. Si se hubiera puesto como un símbolo de predicado binario cualquiera, aunque se especificaran axiomas tratando de rescatar las propiedades de la identidad, nunca se podría garantizar que se interpretaría como la identidad.

3. Se han elegido únicamente dos conectivos porque la lógica de primer orden será una extensión de la lógica de enunciados, de modo que los conectivos mantendrán el significado que se les dio allá (en términos de tablas de verdad). Por tanto las siguientes equivalencias entre fórmulas seguirán siendo válidas, para α y β fórmulas del (nuevo) lenguaje $\mathscr{L}$:

$$\begin{aligned} \alpha \wedge \beta &:= \neg(\alpha \Rightarrow \neg\beta) \\ \alpha \vee \beta &:= \neg\alpha \Rightarrow \beta \\ \alpha \Leftrightarrow \beta &:= (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha) \end{aligned}$$

De modo que no se está perdiendo nada al considerar únicamente $\neg$ y $\Rightarrow$ como conectivos lógicos.

4. Los cuantificadores $\forall$ y $\exists$ son parámetros porque su significado cambiará según la interpretación que se esté manejando. Recuérdese que en la sección anterior $\forall x$ quería decir en un caso "todo ser humano" y en el otro "todo número natural".

5. No se le exige a un lenguaje de primer orden que tenga símbolos de predicado, constantes individuales o símbolos funcionales. Tampoco se supone que tenga al símbolo de igualdad. Pero recordemos que los predicados son los símbolos que nos permiten hablar de propiedades de individuos y de relaciones de individuos entre sí. Si tuviéramos un lenguaje de primer orden sin predicados y sin igualdad no podríamos decir nada en él, por tanto supondremos que todo lenguaje de primer orden o tiene al símbolo de igualdad o tiene al menos un predicado (¡aquí la o es inclusiva!).

6. Para definir un lenguaje de primer orden se debe especificar si tiene símbolo de igualdad o no, y se tienen que enumerar sus predicados y símbolos funcionales, diciendo sus aridades respectivas, así como sus constantes individuales.

Damos a continuación algunos ejemplos de lenguajes de primer orden:

El lenguaje de la igualdad

Tiene símbolo de igualdad y no tiene predicados, constantes ni símbolos funcionales. Se denota $\mathscr{L} = \{\approx\}$.

El lenguaje de predicados puro

No tiene símbolo de igualdad ni símbolos funcionales, pero tiene un conjunto numerable de constantes individuales y, para cada $n \in \mathbb{N}$, un conjunto numerable de predicados n-arios. Se denota $\mathscr{L} = \{\{P_i^n\}_{i\in\mathbb{N}}, \{c_n\}\}_{n\in\mathbb{N}}$.

El lenguaje de la teoría de conjuntos

Tiene símbolo de igualdad y un predicado binario, $\in$, que denota la pertenencia. No tiene constantes individuales ni símbolos funcionales. Se denota $\mathscr{L} = \{\approx, \in\}$ o simplemente $\mathscr{L} = \{\in\}$.

Aquí es conveniente anotar que como casi todos los lenguajes de primer orden con los que se trabaja tienen al símbolo de igualdad, es muy común no especificar que se tiene igualdad. Más bien se aclara que no se tiene símbolo de igualdad cuando esto ocurre.

También se habrá notado que este último lenguaje tiene un símbolo, a saber $\in$, que no es estrictamente un símbolo de predicado, si se fuera riguroso se tendría que poner una letra mayúscula que representara la pertenencia, pero como un abuso de notación se toma a $\in$ como predicado binario.

El lenguaje de la teoría de grupos

Tiene símbolo de igualdad, un símbolo funcional binario, f, que representa a la operación del grupo, un símbolo funcional unario, g, que representa al inverso y una constante individual, c, que representa al elemento neutro del grupo.

Damos ahora las reglas de formación para cualquier lenguaje de primer orden $\mathscr{L}$. Nuevamente, una *expresión* de $\mathscr{L}$ es cualquier sucesión finita de símbolos de $\mathscr{L}$. De entre todas las expresiones de $\mathscr{L}$ vamos a seleccionar las *fórmulas bien formadas* de $\mathscr{L}$, pero este proceso no va ser tan sencillo como en el caso de lenguajes proposicionales, recuérdese que estamos tratando de reflejar la estructura interna de las proposiciones, de modo que antes de llegar a las fórmulas bien formadas tendremos que pasar por expresiones más simples que sean importantes para construir las dichas fórmulas. Estas expresiones son los *términos* de $\mathscr{L}$, y son las expresiones que denotan a individuos.

Evidentemente las variables individuales y las constantes individuales denotan individuos, pero hay otra manera de denotar individuos: por medio de los símbolos funcionales, ya que una función aplicada a individuos genera otro individuo (recuérdese el caso de la suma en $\mathbb{N}$).

Definición. Daremos la siguiente definición recursiva para los *términos* del lenguaje $\mathscr{L}$:

1. Una variable individual es un término de $\mathscr{L}$.
2. Una constante individual es un término de $\mathscr{L}$.
3. Si $t_1, t_2, \ldots, t_n$ son n términos de $\mathscr{L}$ y f es un símbolo funcional n-ario de $\mathscr{L}$, entonces $f t_1 \ldots t_n$ es un término de $\mathscr{L}$.
4. Una expresión de $\mathscr{L}$ es un término de $\mathscr{L}$ si y sólo si es un término en virtud de 1, 2 o 3.

Si $\mathscr{L}$ no tiene símbolos funcionales entonces los términos de $\mathscr{L}$ son las constantes y las variables individuales.

Las *fórmulas bien formadas* de $\mathscr{L}$ son las expresiones de $\mathscr{L}$ que afirman o niegan algo acerca de los individuos del dominio de discurso. Para hacer esto vamos a hacer uso de los predicados n-arios, que representan en el lenguaje formal propiedades de individuos y relaciones entre individuos. Las expresiones obtenidas serán las fórmulas más simples, es decir, las fórmulas atómicas. A partir de ellas formaremos fórmulas más complejas usando los conectivos y los cuantificadores.

Definición. Daremos la siguiente definición recursiva para las *fórmulas bien formadas* de $\mathscr{L}$.

Fórmulas atómicas

1. Si t_1 y t_2 son términos de $\mathscr{L}$ entonces la siguiente expresión de $\mathscr{L}$ es una fórmula atómica de $\mathscr{L}$: $t_1 \approx t_2$.
2. Si $t_1, t_2, \ldots, t_n$ son términos de $\mathscr{L}$ y P es un predicado n-ario de $\mathscr{L}$ entonces la siguiente expresión es una fórmula atómica de $\mathscr{L}$: $P t_1 \ldots t_n$.
3. Una expresión de $\mathscr{L}$ es una fórmula atómica de $\mathscr{L}$ si y sólo si es una fórmula atómica en virtud de 1 o 2.

Fórmulas bien formadas

1. Toda fórmula atómica es una fórmula bien formada de $\mathscr{L}$.
2. Si α y β son fórmulas bien formadas de $\mathscr{L}$, entonces también lo son las siguientes expresiones de $\mathscr{L}$: $(\neg\alpha)$ y $(\alpha \Rightarrow \beta)$.

3. Si α es una fórmula bien formada de $\mathscr{L}$ y x es una variable individual de $\mathscr{L}$ entonces las siguientes expresiones son fórmulas bien formadas de $\mathscr{L}$: $(\forall x\alpha)$ y $(\exists x\alpha)$.
4. Una expresión de $\mathscr{L}$ es una fórmula bien formada de $\mathscr{L}$ si y sólo si es una fórmula bien formada en virtud de 1, 2 o 3.

Observación. De ahora en adelante llamaremos fórmulas o fbf's a las fórmulas bien formadas de $\mathscr{L}$. A las fbf's no atómicas se les denomina *compuestas* o *moleculares*.

Observaciones:

1. Es posible, en un lenguaje de primer orden, tener predicados 0-arios, en este caso, si P es un predicado 0-ario, de acuerdo con las reglas de formación, P es una fórmula atómica de $\mathscr{L}$. P es una fórmula que no se refiere a individuos, en este sentido los lenguajes proposicionales están contenidos en los lenguajes de primer orden. Sin embargo, como ya se ha estudiado la lógica proposicional en detalle y lo que nos interesa es hacer de nuestros lenguajes formales instrumentos más finos, supondremos que todo predicado de $\mathscr{L}$ tiene aridad al menos 1.

2. También es posible para un lenguaje de primer orden tener símbolos funcionales 0-arios. Estos símbolos pueden ser identificados con las constantes individuales y de ahora en adelante supondremos que todo símbolo funcional es de aridad mayor o igual a 1.

Las convenciones adoptadas respecto a los paréntesis para los lenguajes proposicionales las seguiremos adoptando aquí. Por ejemplo, se omitirán los paréntesis externos de una fórmula, se usarán paréntesis cuadrados tanto como redondos para hacer las fórmulas más legibles y en general se omitirán paréntesis cuando no haya riesgo de ambigüedad. Se agrega la convención de que los cuantificadores se aplican a tan poco como sea posible, de tal modo que $\forall x\alpha \Rightarrow \beta$ se interpretará como $(\forall x\alpha) \Rightarrow \beta$ y no como $\forall x(\alpha \Rightarrow \beta)$.

Si se observa el caso (3) de la definición de fórmula bien formada de $\mathscr{L}$ se notará que los cuantificadores se aplican sólo a variables individuales, en estos lenguajes no se acepta como fórmula una expresión como $\forall P\alpha$, donde P es un predicado de $\mathscr{L}$. Por eso se llaman *lenguajes de primer orden.* Existen lenguajes de orden superior donde cuantificaciones sobre propiedades son permitidas, pero esos lenguajes no serán estudiados aquí.

Ejemplos:

Sea $\mathscr{L}$ el lenguaje de predicados puro. Entonces cualquier variable individual o constante individual es un término de $\mathscr{L}$, y éstos son los únicos términos de $\mathscr{L}$. Las siguientes expresiones son fórmulas atómicas de $\mathscr{L}$:

$$P_1^2 x_1 x_2,\ P_1^1 c_1,\ P_1^n c_1 c_2 \ldots c_n,\ P_1^n c_1 x_2 \ldots x_n.$$

Las siguientes expresiones son fórmulas moleculares de $\mathscr{L}$:

$$\neg P_1^2 x_1 x_2,\ P_1^2 x_1 x_2 \Rightarrow P_1^2 x_1 x_2,\ \forall x_1 P_1^2 x_1 x_2,\ \forall x_{20} P_1^2 x_1 x_2,$$
$$\forall x_1 P_1^n c_1 c_2 \ldots c_n,\ \exists x_2 P_1^1 c_1,\ \forall x_1 P_1^2 x_1 x_2 \Rightarrow \exists x_2 P_2^1 c_1,\ \forall x_1 (P_1^2 x_1 x_2 \Rightarrow \exists x_2 P_2^1 c_1).$$

Las siguientes expresiones no son fórmulas bien formadas de $\mathscr{L}$:

$P_1^1 c_1 c_2$, porque el predicado es unario y se escribieron a la derecha dos términos.

$\exists c_1 P_1^1 c_1$, porque a la derecha del cuantificador aparece una constante, y debe aparecer una variable individual.

Como se puede ver en el ejemplo anterior, puede resultar demasiado engorroso poner subíndices a todas las variables, constantes, predicados y símbolos funcionales que aparezcan en una cierta fórmula. Nuevamente abusando de la notación, se usarán las letras x, y, u, v, w como variables individuales; las letras a, b, c, d como constantes individuales; las letras P, Q, R, S como predicados y las letras f, g, h como símbolos funcionales.

Ejercicios

Sea $\mathscr{L}$ el lenguaje de primer orden que tiene un predicado unario P, un predicado binario Q, un predicado ternario R, un símbolo funcional unario f, un símbolo funcional binario g, dos constantes individuales, a y b y que tiene al símbolo de la igualdad.

1. Escriba 10 términos de $\mathscr{L}$, justificando en cada caso por qué son términos.

2. Escriba 10 fórmulas de $\mathscr{L}$, justificando en cada caso por qué son fórmulas.

3. Decir si las siguientes expresiones son términos de $\mathscr{L}$ o no. (Justificar su respuesta): fx, Px, fa, gab, $gfaa$, a, $gfxfy$, gz, ay.

4. Decir si las siguientes expresiones son fórmulas de $\mathscr{L}$ o no. (Justificar su respuesta): Pa, Pb, Qax, Qb, $\neg fa$, $\forall x(Px \Rightarrow Qx)$, $\forall x(Px \Rightarrow Qxx)$, $\exists x \forall x Pc$, $Px \Rightarrow gxx$, $\exists x Px \Rightarrow \forall x \neg Px$, $\forall x \exists y \forall z$.

7.3 Interpretaciones y satisfacibilidad

Para los lenguajes proposicionales teníamos asignaciones de verdad que nos permitían decidir cuáles fórmulas eran verdaderas y cuáles falsas. Para los lenguajes de primer orden vamos a necesitar interpretar (*i.e.* traducir) todos los símbolos no lógicos del lenguaje de manera que podamos decidir cuáles fórmulas se traducen como proposiciones verdaderas y cuáles como proposiciones falsas.

Realmente los dos procesos no son tan distintos como podría parecer a primera vista. En un lenguaje proposicional, para saber si una fórmula es verdadera o no se traducen las letras enunciativas que aparecen en la fórmula por proposiciones en algún lenguaje natural, luego se contrastan con la realidad para saber si las proposiciones obtenidas son verdaderas o falsas y finalmente se utiliza la definición de verdad (tablas de verdad) para los conectivos para obtener el valor de verdad de la fórmula bajo la interpretación dada. Desde el punto de vista de la lógica, cómo se contrastan las proposiciones de un lenguaje natural con la realidad para decidir sobre su verdad es irrelevante, lo importante es que cada letra enunciativa tiene, bajo una interpretación dada, un valor de verdad determinado, y por eso el primer paso del proceso se elimina y nos quedamos únicamente con la asignación de verdad, pues ésta nos proporciona toda la información requerida para encontrar el valor de verdad de cualquier fórmula.

Cuando se trabaja con lenguajes de primer orden se hace lo mismo: se traducen todos los símbolos no lógicos del lenguaje, se contrastan con la realidad (en un sentido que se hará preciso más adelante), se decide sobre la verdad o falsedad de las fórmulas atómicas y se aplican ciertas reglas para encontrar la verdad o falsedad de todas las fórmulas bajo una interpretación dada. Como muchas (en realidad casi todas) de las interpretaciones que se dan para estos lenguajes son de carácter matemático es necesario tener una teoría de la verdad que haga posible decidir cuándo una cierta proposición sobre una estructura abstracta es verdadera

y cuándo es falsa. La definición que vamos a dar aquí, y que es la definición "estándar" para la lógica de primer orden se debe a Alfred Tarski.[1]

Antes de dar la definición rigurosa vamos a considerar un ejemplo.

Sea $\mathscr{L}$ el lenguaje de primer orden con igualdad que tiene una constante individual c, un predicado binario P, un símbolo funcional unario f y dos símbolos funcionales binarios g y h. Vamos a dar significado a los símbolos del lenguaje para decidir sobre el valor de verdad de algunas fórmulas bajo esta interpretación. Podemos denotar a este lenguaje como $\mathscr{L} = \{P, f, g, h, c\}$.

Primero tenemos que decir cuáles son los individuos de quienes estamos hablando, es decir, tenemos que fijar el universo de discurso. Sea $\mathbb{N}$ el universo de discurso, esto quiere decir que cuando leamos $\forall x(\exists x)$ entenderemos "todo número natural" ("existe algún número natural"). Este paso equivale a darle significado a los cuantificadores, que son parámetros de $\mathscr{L}$.

Ahora damos significado a los demás parámetros de $\mathscr{L}$. La constante "c" tiene que ser interpretada como un elemento de $\mathbb{N}$, sea 0 la interpretación de c. El predicado binario P representa una relación binaria entre elementos de $\mathbb{N}$, interpretemos a P como la relación de orden estricto $<$. Los símbolos funcionales representan funciones en el dominio del discurso, de la aridad correspondiente. O sea que f deberá ser interpretada como una función de $\mathbb{N}$ en $\mathbb{N}$, y g y h deberán ser interpretadas como funciones de $\mathbb{N} \times \mathbb{N}$ en $\mathbb{N}$. Interpretemos a f como la función:

$$S: \mathbb{N} \to \mathbb{N} \text{ tal que } S(n) = n + 1 \text{ para } n \in \mathbb{N},$$

y traduzcamos a g y h como la suma y el producto en $\mathbb{N}$, respectivamente.

Esta interpretación se denota $\mathfrak{N} = \langle \mathbb{N}, <, S, +, \cdot, 0 \rangle$.

Bajo esta interpretación podemos calcular el valor de verdad de muchas fórmulas de $\mathscr{L}$. También es fácil ver que cualquier proposición que se haga en español sobre los números naturales y en la que se involucren únicamente las relaciones y funciones que aparecen en puede ser traducida como una fbf de $\mathscr{L}$.

[1] Al lector interesado en la justificación filosófica de esta definición de verdad se le aconseja leer *The Semantic Conception of Truth*, por Alfred Tarski, [Ta].

Ejemplos:

Primero traduciremos proposiciones sobre $\mathfrak{N}$ del español a $\mathscr{L}$.

(1) La suma en $\mathbb{N}$ es conmutativa
$\forall x \forall y (gxy \approx gyx)$

(2) 0 es el menor elemento de $\mathbb{N}$
$\forall x \, Pcx$

(3) La relación $<$ en $\mathbb{N}$ es transitiva
$\forall x \forall y \forall z ((Pxy \wedge Pyz) \Rightarrow Pxz)$

(4) La relación $<$ en $\mathbb{N}$ es antisimétrica
$\forall x \forall y (Pxy \Rightarrow \neg Pyx)$ o $\neg (\exists x \exists y (Pxy \wedge Pyx)$

(5) El sucesor de cualquier número es mayor que él
$\forall x (Pxfx)$

(6) $m(n + r) = mn + mr$ para $m, n, r \in \mathbb{N}$.
$\forall x \forall y \forall z (hxgyz \approx ghxyhxz)$

(7) En $\mathbb{N}$ no hay un último elemento
$\forall x \exists y \, Pxy$ o $\neg \exists x \forall y (Pyx \vee y \approx x)$

(8) 1 es neutro multiplicativo

No tenemos en $\mathscr{L}$ ninguna constante que represente al 1, sin embargo el término fc representa al sucesor de 0 en $\mathbb{N}$, que es precisamente 1. Así que podemos expresar (8) de la siguiente forma:

$$\forall x (hxfc \approx x)$$

Pasamos ahora al problema de calcular el valor de verdad de fórmulas de $\mathscr{L}$ bajo esta interpretación.

(1) Sea α la fórmula $c \approx c$
Esta fórmula se traduce como $0 = 0$, que es verdadera en $\mathfrak{N}$, este hecho lo escribimos así: $\mathfrak{N} \vDash \alpha$.

(2) Sea α la fórmula Pcc
Esta fórmula se traduce como $0 < 0$ que es falsa en $\mathfrak{N}$, hecho que se denota: $\mathfrak{N} \nvDash \alpha$.

(3) $\alpha = \forall x(c \approx x \vee Pcx)$
α se interpreta como "todo número natural es mayor o igual a cero", por tanto $\mathfrak{N} \vDash \alpha$.

(4) $\alpha = \exists x \exists y Pxy$
α dice que hay dos números naturales uno de los cuales es mayor que el otro, por tanto $\mathfrak{N} \vDash \alpha$.

(5) $\alpha = \forall x \forall y Pxy$
α dice que dados dos naturales cualesquiera, uno de ellos es mayor que el otro, esto es falso y por tanto $\mathfrak{N} \nvDash \alpha$.

(6) $\alpha = \forall x \exists y Pxy$
α dice que dado cualquier número natural existe otro mayor que él, por tanto $\mathfrak{N} \vDash \alpha$.

(7) $\alpha = \exists y \forall x Pxy$
α dice que hay un número natural mayor que todos y por tanto $\mathfrak{N} \nvDash \alpha$.

Los Ejemplos (6) y (7) nos muestran que el orden de los cuantificadores es de suma importancia para analizar fórmulas de $\mathscr{L}$.

(8) $\alpha = \exists x \forall y Pxy$
α dice que existe un número natural que es menor que todos lo cual es falso (¡0 *no* es menor que sí mismo!) y por lo tanto $\mathfrak{N} \nvDash \alpha$.

Si se observa, aunque las fórmulas de (7) y (8) son ambas falsas en $\mathfrak{N}$, sus significados son distintos. No sólo es importante el orden de los cuantificadores, también qué variables son afectadas por cada uno.

(9) $\alpha = Pxy$
α dice que el número natural representado por "x" es menor que el natural representado por "y", pero tanto x como y son variables, sin significado fijo en $\mathfrak{N}$, por lo que no podemos asignarle a α ningún valor de verdad bajo la interpretación $\mathfrak{N}$. Sin embargo, cada vez que a "x" y a "y" se le asignen significados dentro de $\mathbb{N}$, el valor de verdad de α (para esos significados) podrá ser determinado. Supongamos que a "x" le asignamos el número 3 y a "y" el 7, entonces α es verdadera en $\mathfrak{N}$. Esto se denota por $\mathfrak{N} \vDash \alpha[3, 7]$. Análogamente $\mathfrak{N} \nvDash \alpha[7, 3]$.

Tenemos, pues, que ciertas fórmulas de $\mathscr{L}$ tienen valores de verdad fijos bajo $\mathfrak{N}$ y otras necesitan que se especifiquen significados para las variables. La diferencia

entre las variables que aparecían en los ejemplos (1)–(8) y las que aparecen en (9) es que las primeras estaban afectadas por cuantificadores y las de (9) no. Las primeras variables están "*acotadas*" o "*ligadas*" en α, mientras que las de (9) están "*libres*" en α.

Damos a continuación una definición recursiva de lo que significa, para una variable individual x, decir que x *ocurre libre* en α, para α una fórmula de $\mathscr{L}$ cualquiera.

Definición. Sea α una fórmula de $\mathscr{L}$ y x una variable individual de $\mathscr{L}$. Se define recursivamente x *ocurre libre* en α de la manera siguiente:

1. Si α es atómica, x ocurre libre en α si y sólo si x ocurre en (esto es, x es un símbolo de) α.
2. Si $\alpha = (\neg\beta)$ entonces x ocurre libre en α si y sólo si x ocurre libre en β.
3. Si $\alpha = (\beta \Rightarrow \gamma)$ entonces x ocurre libre en α si y sólo si x ocurre libre en β o en γ.
4. Si $\alpha = \forall x_i \beta$ entonces x ocurre libre en β y x no es x_i.

Definición. Si x ocurre en α, pero no ocurre libre, decimos que x está *acotada* o *ligada* en α.

Esta definición, que puede parecer un poco obscura, se puede ver de otra manera, definiendo las ocurrencias acotadas de x en vez de las libres.

Para un cuantificador $\forall$ o $\exists$ se define su *alcance* dentro de una fórmula como la fórmula bien formada inmediatamente a la derecha de él. Así, por ejemplo, el alcance de $\forall x$ en $\forall x(\alpha \Rightarrow \beta)$ es $\alpha \Rightarrow \beta$, mientras que el alcance de $\forall x$ en $\forall x\alpha \Rightarrow \beta$ es α. El alcance de $\forall x$ en $\forall x \exists y \alpha$ es $\exists y \alpha$.

Sea ahora x_i una variable que ocurre en una fórmula α de $\mathscr{L}$. Una ocurrencia de x_i en α está acotada en α si y sólo si x_i es la variable de algún cuantificador $\forall x_i$ o $\exists x_i$ en α, o está en el alcance de algún cuantificador $\forall x_i$ o $\exists x_i$ en α.

Ejemplo. Consideremos las siguientes fórmulas de $\mathscr{L}$:

(1) $P x_1 x_2$

(2) $\forall x_1 P x_1 x_2$

(3) $\exists x_3 P x_1 x_2$

(4) $\exists x_1 P x_1 x_2 \Rightarrow f x_1 \approx c$

En la primera fórmula las ocurrencias de x_1 y x_2 son libres pues no hay cuantificadores. En la segunda las dos ocurrencias de x_1 están acotadas, la primera porque forma parte de $\forall x_1$ y la segunda porque está en el alcance de $\forall x_1$, la ocurrencia de x_2 en la segunda fórmula está libre porque, aunque está en el alcance de un cuantificador, éste no tiene a la misma variable. En la tercera fórmula x_3 aparece ligada y las otras dos están libres. En la cuarta fórmula las primeras dos ocurrencias de x_1 aparecen acotadas, mientras que la ocurrencia de x_2 y la tercera de x_1 están libres.

Definición. Cuando en una fórmula α de $\mathscr{L}$ ninguna variable ocurre libre se dice que α es un *enunciado* de $\mathscr{L}$. Ejemplos de enunciados: Pcc, $fc \approx c$, $\forall x Pxc$, $\forall x \forall y Pxfy$.

Ejercicios

1. Traducir las siguientes proposiciones a fbf's de $\mathscr{L}$:

- **a.** El producto en $\mathbb{N}$ es conmutativo.
- **b.** El producto en $\mathbb{N}$ es asociativo.
- **c.** La suma en $\mathbb{N}$ es asociativa.
- **d.** La relación $<$ en $\mathbb{N}$ es antirreflexiva.
- **e.** Todo natural distinto del cero es sucesor de algún natural.
- **f.** El cero es neutro aditivo.
- **g.** El sucesor de cualquier número es el resultado de sumar 1 a ese número, esto es $S(n) = n + 1$, para $n \in \mathbb{N}$.
- **h.** Si $n < m$ entonces $n + r < m + r$ para cualesquiera $m, n, r \in \mathbb{N}$.

2. En este lenguaje, y con la interpretación $\mathfrak{N}$ que se ha estado manejando, decir si $\mathfrak{N} \models \alpha$ o no, para:

- **a.** $\alpha = \forall x \forall y (Pxy \Rightarrow Pfxfy)$
- **b.** $\alpha = (Pcc \Rightarrow c \approx c)$

c. $\alpha = \exists x\, Pcx$

d. $\alpha = \exists x\, Pxc$

e. $\alpha = (\forall x \forall y\, Pxy \Rightarrow \forall x \forall y\, Pfxfy)$

3 Con base en el ejercicio 2, supongamos ahora que la variable "x" se interpreta como 5, "y" como 2 y "z" como 1. Decidir si $\mathfrak{N} \vDash \alpha[5, 2, 1]$ o no, para:

f. $\alpha = gfzy \approx x$

g. $\alpha = \exists w(Pfwx)$

h. $\alpha = \exists w(Pfwy)$

i. $\alpha = hzz \approx y$

j. $\alpha = \forall whwz \approx w$

4. Analizar las ocurrencias de variables en las siguientes fórmulas. Decir cuál es el alcance de cada cuantificador que aparezca.

a. $\exists x \exists y \exists z(Pxy \vee Pyz)$

b. $\exists x\, Pcc$

c. $Pxfy \Rightarrow \forall x\, Pcx$

d. $\forall xfy \approx y \Rightarrow \exists z gxx \approx x$

7.4 Definición de verdad de Tarski

En esta sección se formalizará lo que se hizo intuitivamente en la sección anterior. Daremos una definición precisa de lo que es una interpretación para un lenguaje de primer orden $\mathscr{L}$ y de lo que significa que una fórmula sea verdadera bajo una interpretación.

Definición. Sea $\mathscr{L}$ cualquier lenguaje de primer orden. Una *estructura* para $\mathscr{L}$ (o *$\mathscr{L}$-estructura*) $\mathfrak{A}$ consta de:

1. Un conjunto no vacío A llamado el *dominio* o *universo* de $\mathfrak{A}$.
2. Una relación n-aria $R_i^{\mathfrak{A}} \subseteq A^n$ para cada predicado n-ario P_i de $\mathscr{L}$.
3. Una función n-aria $f_j^{\mathfrak{A}}: A^n \to A$ para cada símbolo funcional n-ario f_j de $\mathscr{L}$.
4. Un elemento $a_k^{\mathfrak{A}} \in A$ para cada constante a_k de $\mathscr{L}$.

Notación: $\mathfrak{A} = \langle A, \{R_i^{\mathfrak{A}}\}, \{f_j^{\mathfrak{A}}\}, \{a_k^{\mathfrak{A}}\}\rangle$

Una estructura $\mathfrak{A}$ asigna significados a todos los parámetros de $\mathscr{L}$. Es claro que $\mathfrak{N}$, como se definió en la sección anterior, satisface la definición de ser una $\mathscr{L}$-estructura para el correspondiente lenguaje de primer orden.

Daremos ahora una definición precisa de lo que significa que una fórmula ϕ de $\mathscr{L}$ sea verdadera en una $\mathscr{L}$-estructura $\mathfrak{A}$. Como se vio en la sección anterior, aparte de tener significados para los parámetros de $\mathscr{L}$ necesitaremos asignar significados a las variables, esto se hace por medio de una función s que a cada variable individual le asigne un elemento del universo de $\mathfrak{A}$. Si ϕ es verdadera en $\mathfrak{A}$ bajo s, este hecho será denotado por $\mathfrak{A} \vDash \phi[s]$.

Sea $\mathscr{L}$ cualquier lenguaje de primer orden, $\mathfrak{A}$ una $\mathscr{L}$-estructura con dominio A y $s\colon V \to A$ una función del conjunto V de las variables individuales de $\mathscr{L}$ en A.

Definiremos recursivamente lo que significa que una fórmula ϕ *es verdadera en* $\mathfrak{A}$ *bajo* s. Notación: $\mathfrak{A} \vDash \phi[s]$.

Al igual que en la definición de fórmula bien formada, tenemos que proceder en dos etapas, la primera para los términos y la segunda para las fórmulas.

Definición. Los términos de $\mathscr{L}$ van a denotar elementos del dominio A de la estructura $\mathfrak{A}$. La siguiente definición asigna a cada término t de $\mathscr{L}$ un elemento de A, denotado por $\bar{s}(t)$, que es el *individuo nombrado por t en* $\mathfrak{A}$ bajo s. La definición es recursiva.

1. Si $t = x_i$ para alguna variable individual x_i de $\mathscr{L}$, entonces $\bar{s}(t) = s(x_i)$.
2. Si $t = c_k$ para alguna constante individual c_k de $\mathscr{L}$, entonces $\bar{s}(t) = c_k^{\mathfrak{A}}$.
3. Si $t = ft_1 \ldots t_n$ donde f es un símbolo funcional n-ario de $\mathscr{L}$ y $t_1, \ldots, t_n$ son términos de $\mathscr{L}$, entonces $\bar{s}(t) = f^{\mathfrak{A}}(\bar{s}(t_1), \ldots, \bar{s}(t_n))$.

Es fácil ver que esta definición rescata la manera intuitiva con que hemos decidido qué elemento está nombrado por qué término en la sección anterior: la sucesión s sirve para dar sentido a las variables, las constantes mantienen el mismo significado bajo cualquier función s, a saber, el que les fue asignado por la estructura $\mathfrak{A}$; por último, para ver cómo se interpreta un término complejo, se interpretan primero los más simples y a las interpretaciones obtenidas se les aplican las funciones correspondientes en $\mathfrak{A}$.

Una última observación: el elemento $\bar{s}(t)$ depende no sólo de s y de t, sino también de $\mathfrak{A}$, pero sería demasiado engorroso mencionar a $\mathfrak{A}$, se tendría que agregar un subíndice o un superíndice; esto sólo se hace cuando se están manejando varias estructuras a la vez y hay riesgo de ambigüedad.

Definición. Ahora definimos, para toda fórmula ϕ de $\mathscr{L}$, lo que significa que ϕ *es verdadera en* $\mathfrak{A}$ *bajo* s o $\mathfrak{A}$ *satisface a* ϕ *con* s. Lo haremos por recursión sobre la complejidad de ϕ.

1. ϕ es atómica

Caso 1a. ϕ es de la forma $t_1 \approx t_2$, con t_1 y t_2 términos de $\mathscr{L}$.

Entonces $\mathfrak{A} \vDash \phi[s]$ si y sólo si $\bar{s}(t_1) = \bar{s}(t_2)$.

Caso 1b. ϕ es de la forma $Pt_1 \ldots t_n$, donde P es un predicado n-ario de $\mathscr{L}$ y $t_1, \ldots, t_n$ son términos de $\mathscr{L}$.

Entonces $\mathfrak{A} \vDash \phi[s]$ si y sólo si $(\bar{s}(t_1), \ldots, \bar{s}(t_n)) \in P^{\mathfrak{A}}$.

2. ϕ es de la forma $(\neg\psi)$

Entonces $\mathfrak{A} \vDash \phi[s]$ si y sólo si $\mathfrak{A} \nvDash \psi[s]$.

3. ϕ es de la forma $(\psi \Rightarrow \xi)$

Entonces $\mathfrak{A} \vDash \phi[s]$ si y sólo si $\mathfrak{A} \nvDash \psi[s]$ o $\mathfrak{A} \vDash \xi[s]$.

4. ϕ es de la forma $\forall x\psi$

Entonces $\mathfrak{A} \vDash \phi[s]$ si y sólo si para toda $a \in A$, $\mathfrak{A} \vDash \psi[s(x/a)]$, donde $s(x/a)\colon V \to A$ es la siguiente función:

$$s(x/a)(y) = \begin{cases} s(y) & \text{si } y \neq x \\ a & \text{si } y = x \end{cases}$$

5. ϕ es de la forma $\exists x\psi$

Entonces $\mathfrak{A} \vDash \phi[s]$ si y sólo si existe $a \in A$ tal que $\mathfrak{A} \vDash \psi[s(x/a)]$, donde $s(x/a)$ es la misma función definida en el inciso anterior.

Nuevamente se puede ver que esta definición en efecto formaliza la manera intuitiva en que calculamos el valor de verdad de una fórmula en la estructura $\mathfrak{A}$. La primera cláusula establece que para calcular el valor de verdad de una fórmula

atómica en una estructura hay que traducir los términos y verificar si los individuos denotados por esos términos están en la relación denotada por los predicados del lenguaje. El caso $1a$. asegura que el predicado $\approx$ siempre se interpreta como la igualdad en cualquier estructura. Las cláusulas 2 y 3 establecen que los conectivos tienen el mismo significado que en la lógica proposicional. Las cláusulas 4 y 5 definen a los cuantificadores.

La cláusula 4 podría ser interpretada de la siguiente manera: Supongamos que agregamos al lenguaje $\mathscr{L}$ una nueva constante a para cada elemento a de A, entonces una fórmula $\forall x\phi(x)$ es verdadera si $\phi(a)$ es verdadera para todo elemento a de A. En este sentido el cuantificador universal $\forall$ funciona como una abreviación de una conjunción (generalmente infinita), pues $\forall x\phi$ es equivalente a $\bigwedge_{a\in A}\phi(a)$. Análogamente se puede pensar en $\exists x$ como una disyunción.

Otro hecho que es claro a partir de los ejemplos es que para saber si una fórmula es verdadera en $\mathfrak{A}$ bajo s, la única información de s que es relevante es el valor de s en las variables que ocurren libres en ϕ. En particular, si ϕ es un enunciado, s es irrelevante. Esto es consecuencia del siguiente teorema, cuya demostración, que se hace por inducción, omitimos.

Teorema 7.1. *Supongamos que s_1 y s_2 son dos funciones de V en A que coinciden en todas las variables que ocurren libres en ϕ. Entonces se tiene que $\mathfrak{A} \vDash \phi[s_1]$ si y sólo si $\mathfrak{A} \vDash \phi[s_2]$.* ■

Corolario 7.2. *Sea ϕ un enunciado. Entonces $\mathfrak{A} \vDash \phi[s]$ para toda s o $\mathfrak{A} \nvDash \phi[s]$ para toda s.* ■

Ejemplos:

Aunque lo más importante es ver que la definición rigurosa lo único que hace es formalizar el procedimiento natural que ya habíamos hecho en la sección 7.3, damos a continuación algunos ejemplos de cómo utilizar la definición para hacer una justificación más formal de que una cierta fórmula es verdadera en una estructura bajo alguna sucesión.

Retomamos el lenguaje $\mathscr{L} = \{P, f, g, h, c\}$, donde P es un predicado binario, f un símbolo funcional unario, g y h son símbolos funcionales binarios y c es una constante individual. Sea $\mathfrak{N} = \langle \mathbb{N}, <, S, +, \cdot, 0\rangle$, que es una $\mathscr{L}$-estructura. Sea $s\colon V \to \mathbb{N}$ la función tal que $s(x_i) = i$. Usaremos la definición formal de

satisfacibilidad en una estructura para decidir si $\mathfrak{N} \vDash \phi[s]$ para algunas fórmulas ϕ de $\mathscr{L}$.

(1) ϕ es Pcc
Primero notemos que $\bar{s}(c) = 0$. Pero como $(0, 0) \notin P^{\mathfrak{N}}$, ya que 0 no es menor que 0, concluimos que $\mathfrak{N} \nvDash \phi[s]$.

(2) ϕ es Px_1fx_1
$\bar{s}(x_1) = 1$ y $\bar{s}(fx_1) = f^{\mathfrak{N}}(\bar{s}(x_1)) = S(1) = 2$. Como $(1, 2) \in P^{\mathfrak{N}}$ ya que $1 < 2$, concluimos que $\mathfrak{N} \vDash \phi[s]$.

(3) ϕ es $\forall x_1 Pcx_1$
Por la cláusula 4, $\mathfrak{N} \vDash \phi[s]$ si y sólo si para todo $n \in \mathbb{N}$, $\mathfrak{N} \vDash Pcx_1[s(x/n)]$ si y sólo si para todo $n \in \mathbb{N}$, $(0, n) \in P^{\mathfrak{N}}$ si y sólo si para todo $n \in \mathbb{N}$, $0 < n$. Esto último es falso para $n = 0$ y por tanto $\mathfrak{N} \nvDash \phi[s]$.

(4) ϕ es $\exists x \exists y Pxy$
Por cláusula 5 de la definición, $\mathfrak{N} \vDash \phi[s]$ si y sólo si existen dos números naturales n y m tales que $\mathfrak{N} \vDash \phi[s(x/n, y/m)]$ si y sólo si existen $n, m \in \mathbb{N}$ tales que $(n, m) \in P^{\mathfrak{N}}$ si y sólo si existen $n, m \in \mathbb{N}$ tales que $n < m$. Esto es claramente cierto y por tanto $\mathfrak{N} \vDash \phi[s]$.

Definición. Sean Σ un conjunto de fórmulas y ϕ una fórmula de un lenguaje de primer orden $\mathscr{L}$. Decimos que Σ *implica lógicamente* a ϕ, o que ϕ es *consecuencia lógica* de Σ si y sólo si para toda $\mathscr{L}$-estructura $\mathfrak{A}$ y toda función $s: V \to A$, si todos los elementos de Σ son verdaderos en $\mathfrak{A}$ bajo s entonces también lo es ϕ. Notación: $\Sigma \vDash \phi$.

Igual que para la lógica proposicional, se escribe $\phi \vDash \psi$ en lugar de $\{\phi\} \vDash \psi$; y se dice que ϕ y ψ son *lógicamente equivalentes*, denotado $\phi \models\!\!\vDash \psi$, si y sólo si $\phi \vDash \psi$ y $\psi \vDash \phi$.

Definición. Una fórmula ϕ de $\mathscr{L}$ es *universalmente válida* si es verdadera en cualquier $\mathscr{L}$-estructura bajo cualquier sucesión s.

Ejemplos:

(1) $\forall x_1 Px_1 \vDash \exists x_1 Px_1$

Sea $\mathfrak{A}$ una estructura arbitraria, s una función de V en el dominio de $\mathfrak{A}$ tal que $\mathfrak{A} \vDash \forall x_1 P x_1[s]$. Entonces para todo $a \in A$, $\mathfrak{A} \vDash P x_1[s(x_1/a)]$. Sea $a \in A$ cualquiera (aquí se está usando el hecho de que A no es vacío), como $\mathfrak{A} \vDash P x_1[s(x_1/a)]$, podemos concluir que $\mathfrak{A} \vDash \exists x_1 P x_1$.

(2) $\exists x \forall y P x y \vDash \forall y \exists x P x y$

Sean $\mathfrak{A}$ y s tales que $\mathfrak{A} \vDash \exists x \forall y P x y[s]$; para ver que $\mathfrak{A} \vDash \forall y \exists x P x y[s]$ consideremos un elemento $a \in A$ arbitrario. Lo que queremos es probar que $\mathfrak{A} \vDash \exists x P x y[s(y/a)]$, es decir, que existe $b \in A$ tal que $\mathfrak{A} \vDash P x y[s(y/a, x/b)]$. Sea b el elemento de A tal que $\mathfrak{A} \vDash \forall y P x y[s(x/b)]$, entonces $\mathfrak{A} \vDash P x y[s(y/a, x/b)]$.

Ejercicios

1. Sean $\mathscr{L}$, $\mathfrak{N}$ y s como en los ejemplos anteriores. Encontrar $\bar{s}(t)$ para los siguientes términos:

$$c,\ g c x_3,\ x_{20},\ h f c g x_1 x_2,\ g x_4 c.$$

2. Decir si $\mathfrak{N} \vDash \phi[s]$ para las siguientes fórmulas:

a. $\forall x_1 (x_1 \approx x_1)$

b. $\exists x (P x x \Rightarrow \neg P x x)$

c. $P f c c \Rightarrow \forall x P f x x$

d. $\exists x P x x \Rightarrow \forall x P x x$

***3.** Probar el Teorema 7.1.

4. Probar las siguientes afirmaciones:

a. $\exists x P x \nvDash P x$

b. $P x \vDash \exists x P x$

c. $\forall y \exists x P x y \nvDash \exists x \forall y P x y$

d. $\forall x \forall y P x y \vDash \forall y P x y$

e. Si $\vDash \alpha$ entonces $\vDash \forall x \alpha$

f. $\exists x P x \models\!\!\dashv \neg \forall x \neg P x$

g. $\forall x P x \models\!\!\dashv \neg \exists x \neg P x$

Capítulo 8
Lógica de predicados: enfoque sintáctico

La lógica es capaz de justificar las matemáticas en no mayor grado que la biología es capaz de justificar la vida.

Yuri Manin

8.1 Introducción

En este capítulo se construye una teoría formal para lenguajes de primer orden cuyos teoremas sean precisamente las fórmulas universalmente válidas. En la lógica proposicional la existencia de un cálculo es un lujo, pues se tiene un método de decisión para verificar si una fórmula dada es una tautología o no. Aquí la situación es diferente, puesto que tal algoritmo no existe; la existencia de una teoría axiomática que demuestre en un número finito de pasos cualquier fórmula universalmente válida tiene pues, una mayor importancia en este contexto.

Presentaremos dos sistemas formales para la lógica de primer orden: uno axiomático y uno de deducción natural. Cada uno de ellos extiende el correspondiente sistema presentado en el Capítulo 5. No haremos un estudio detallado de estos sistemas, simplemente los definiremos, daremos algunos ejemplos y probaremos algunos metateoremas básicos.

8.2 Un cálculo de predicados

El sistema que presentamos en esta sección se debe a Mendelson [Me].

En todo el capítulo $\mathscr{L}$ denotará un lenguaje de primer orden cuyos símbolos son los siguientes:

Un conjunto numerable de variables individuales

$$x_1, x_2, \ldots, x_n, \ldots$$

Para cada $n \in \mathbb{N}$, un conjunto no vacío de predicados n-arios,

$$P_1^n, P_2^n, \ldots, P_i^n, \ldots$$

Para cada $n \in \mathbb{N}$, un conjunto no vacío de símbolos funcionales n-arios

$$f_1^n, f_2^n, \ldots, f_j^n, \ldots$$

Un conjunto no vacío de constantes individuales

$$c_1, c_2, \ldots, c_k, \ldots$$

Conectivos lógicos

$$\neg \quad \text{y} \quad \Rightarrow$$

Cuantificador universal

$$\forall$$

Símbolos de puntuación

$$) \quad \text{y} \quad ($$

El lector observará que se han hecho algunas modificaciones en la definición de lenguaje de primer orden.

En primer lugar se ha anulado el predicado binario $\approx$. La razón para hacer esto es que desde el punto de vista sintáctico este predicado no tiene propiedades especiales, puede ser cualquier predicado binario; cuando se esté trabajando en un lenguaje de primer orden con igualdad y se quiera utilizar algún resultado sintáctico se podrá hacer tomando a la igualdad como cualquier predicado binario.

También se han eliminado algunos conectivos y el cuantificador existencial. Esto se debe a que, como se probó en capítulos anteriores, los siguientes pares de fórmulas son lógicamente equivalentes:

$$\begin{aligned} \alpha \wedge \beta \quad & \text{y} \quad \neg(\alpha \Rightarrow \neg\beta) \\ \alpha \vee \beta \quad & \text{y} \quad \neg\alpha \Rightarrow \beta \\ \alpha \Rightarrow \beta \quad & \text{y} \quad (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha) \\ \exists x\alpha \quad & \text{y} \quad \neg\forall x\neg\alpha \end{aligned}$$

Recuérdese también que en un lenguaje de primer orden los predicados (con excepción de un predicado binario para la igualdad), los símbolos funcionales y las constantes individuales son opcionales; aquí los hemos introducido para hacer nuestra teoría lo más general posible.

Notación:

1) Se escribe $\phi(x_1, \ldots, x_n)$ para indicar que las variables libres de la fórmula ϕ están en el conjunto $\{x_1, \ldots, x_n\}$.

2) $\phi(t_1, \ldots, t_n)$ es el resultado de sustituir en ϕ las ocurrencias libres (si las hay) de x_i por t_i, para $1 \leq i \leq n$.

La teoría del cálculo de predicados (CP) tiene los siguientes axiomas, para α, β y γ fórmulas de $\mathscr{L}$:

A1 $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

A2 $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$

A3 $(\neg\beta \Rightarrow \neg\alpha) \Rightarrow ((\neg\beta \Rightarrow \alpha) \Rightarrow \beta)$

A4 $\forall x_j \alpha(x_j) \Rightarrow \alpha(t)$, donde $\alpha(x_j)$ es una fórmula de $\mathscr{L}$ y t es un *término* de $\mathscr{L}$ *libre para* x_j en $\alpha(x_j)$.

A5 $\forall x_i(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \forall x_i \beta)$, donde α es una fórmula de $\mathscr{L}$ que no contiene ocurrencias libres de x_i.

Las reglas de inferencia de CP son la siguientes:

Modus Ponens (MP):
β se sigue de α y $\alpha \Rightarrow \beta$

Generalización (Gen):

$\forall x_i \alpha$ se sigue de α

Igual que en el CE, una fórmula ϕ es un teorema de CP si y sólo si existe una lista finita de fórmulas de $\mathscr{L}$ cada una de las cuales es una axioma de CP o es consecuencia de anteriores por MP o Gen. Notación: $\vdash \phi$. Análogamente se define $\Gamma \vdash \phi$.

Observación. Los primeros tres axiomas tienen la misma forma que los axiomas del CE y el *Modus Ponens* es una regla de CP, lo que implica que si tomamos predicados 0-arios en $\mathscr{L}$ y los interpretamos como letras proposicionales (*cf.* sección 7.2) se puede ver a CP como una extensión de CE. Es importante notar, sin embargo, que las instancias de A1-A3 en el cálculo de predicados *no* son fórmulas proposicionales, aquí α, β y γ pueden ser sustituidas por cualesquiera fórmulas de $\mathscr{L}$, las siguientes fórmulas son instancias de A1-A3:

$$\forall x Px \Rightarrow (Pc \Rightarrow \forall x Px)$$
$$(Px \Rightarrow (Qxy \Rightarrow Pc)) \Rightarrow ((Px \Rightarrow Qxy) \Rightarrow (Px \Rightarrow Pc))$$
$$(\neg \forall x Qxfx \Rightarrow \neg Pz) \Rightarrow ((\neg \forall x Qxfx \Rightarrow Pz) \Rightarrow \forall x Qxfx)$$

Se puede verificar fácilmente que toda instancia de una tautología de la lógica proposicional es una fórmula universalmente válida, ya que la definición de verdad de Tarski conserva el significado de los conectivos lógicos. (Una *instancia* de una tautología es el resultado de reemplazar uniformemente, en una tautología, cada letra proposicional por una fórmula bien formada de $\mathscr{L}$.)

El Axioma 4 define al cuantificador universal: si $\forall x \alpha(x)$ es verdadera también debe serlo $\alpha(t)$ para cualquier término t siempre y cuando $\alpha(t)$ afirme de t lo mismo que $\alpha(x)$ afirma de x. Esta última condición es rescatada por la restricción impuesta al Axioma 4. Consideremos, por ejemplo, la siguiente fórmula:

$$\phi(x) = \exists y(x \not\approx y),$$

$\phi(x)$ "dice" que algún individuo es distinto de x. En este caso tenemos que:

$$\forall x \phi(x) = \forall x \exists y(x \not\approx y),$$

$\forall x \phi(x)$ "dice" que existen por lo menos dos individuos.

Sea t el término y, entonces $\phi(y) = \exists y(y \not\approx y)$, lo cual es falso siempre. Cualquier estructura con dos o más elementos satisface $\forall x\phi(x)$ pero no $\phi(y)$, con lo cual se demuestra que la fórmula $\forall x\phi(x) \Rightarrow \phi(y)$ no es universalmente válida en general.

El problema aquí es que mientras $\phi(x)$ "dice" que hay algún elemento distinto de x, $\phi(y)$ "dice" que hay algún elemento distinto de sí mismo. ¿Por qué cambió de significado ϕ? Porque ϕ tenía un cuantificador $\exists y$ que dejaba a x libre, pero cuando x fue sustituida por y, el cuantificador $\exists y$ alcanzó a y, la cual quedó acotada. Para evitar esto se restringe el Axioma 4 de acuerdo con la siguiente definición.

> **Definición.** Sea ϕ una fórmula de $\mathscr{L}$ y t un término de $\mathscr{L}$. Se dice que t es *libre para* x_i en ϕ si ninguna ocurrencia libre de x_i en ϕ está en el alcance de un cuantificador $\forall x_j$, donde x_j es una variable de t.

Es claro que al pedir en el Axioma 4 que t sea libre para x en ϕ se está impidiendo que pase lo que pasó en nuestro ejemplo. Nótese que sólo se habla del cuantificador $\forall$ en la definición, pues siempre que en una fórmula aparezca el cuantificador $\exists$ ésta se sustituirá, mediante equivalencias lógicas, por otra fórmula en la cual sólo el cuantificador universal aparezca.

Ejemplos:

1. x_1 es libre para x_2 en $\forall x_3 P x_2$, pero no en $\forall x_1 P x_2$.
2. $f c x_4$ es libre para x_1 en $\forall x_4 P x_2 \Rightarrow P x_1$, pero no en $\forall x_4(P x_2 \Rightarrow P x_1)$.
3. Para toda fórmula ϕ y toda variable x_i, x_i es libre para x_i en ϕ.
4. Si ϕ es un enunciado entonces t es libre para x en ϕ, para todo t y toda x_i.
5. Si t es un término sin variables entonces t es libre para x_i en ϕ, cualesquiera ϕ y x_i.

La restricción del Axioma 5 también es necesaria, ya que si no pidiéramos que x_i no ocurriera libre en α podríamos tener axiomas del siguiente tipo:

$\forall x(P x \Rightarrow P x) \Rightarrow (P x \Rightarrow \forall x P x)$ (aquí α y β fueron sustituidas por $P x$ y x ocurre libre en $P x$).

Sea $\mathfrak{A}$ una estructura tal que $P^{\mathfrak{A}}$ no es vacío ni todo A, es decir, existen a, $b \in A$ tales que $a \in P^{\mathfrak{A}}$ y $b \notin P^{\mathfrak{A}}$. Entonces $\mathfrak{A} \models \forall x(Px \Rightarrow Px)$, $\mathfrak{A} \not\models \forall x Px$ y $\mathfrak{A} \models Px[a]$. Por tanto $\mathfrak{A} \not\models \forall x(Px \Rightarrow Px) \Rightarrow (Px \Rightarrow \forall x Px)[a]$.

Esta situación no puede ocurrir si x no ocurre libre en α ya que en este caso $\mathfrak{A} \models \alpha[s]$ si y sólo si $\mathfrak{A} \models \forall x\alpha[s]$ para cualquier estructura $\mathfrak{A}$ y sucesión s. (*Cf.* Teorema 7.1).

La regla de Gen puede sorprender un poco a primera vista: tal parecería que se está permitiendo el paso de lo particular a lo general. Pero debemos tomar en cuenta que no estamos afirmando que la fórmula $\alpha \Rightarrow \forall x\alpha$ sea una teorema de CP, sino que *si* α es una teorema de CP *entonces* $\forall x\alpha$ también lo es. Estas dos afirmaciones no son equivalentes, en la próxima sección veremos que el Teorema de la Deducción para el CP requiere de ciertas restricciones.

La regla Gen, a diferencia de MP, no preserva la verdad, pues es posible que α sea verdadera en alguna estructura $\mathfrak{A}$ bajo alguna sucesión s sin que $\forall x\alpha$ lo sea. Sin embargo Gen preserva validez universal, es decir, si α es universalmente válida, también lo es $\forall x\alpha$. Esto es suficiente para nosotros, pues si el CP va a servir de modelo del pensamiento deductivo correcto, todos sus teoremas serán fórmulas universalmente válidas (este hecho se demostrará en la sección 8.4).

Teorema 8.1. *Sea ϕ una fbf de $\mathscr{L}$. Si ϕ es instancia de una tautología, entonces* $\vdash \phi$.

Demostración.

Sea α la tautología que genera a ϕ, esto es, ϕ se obtuvo de α sustituyendo uniformemente las letras proposicionales de α por fórmulas de $\mathscr{L}$. Por el Teorema de Completud para el cálculo de enunciados, α es un teorema de CE. En la prueba de α sustitúyanse las letras proposicionales de α que aparezcan en la prueba por las fórmulas de $\mathscr{L}$ que se usaron para obtener ϕ y las demás letras proposicionales por alguna fórmula de $\mathscr{L}$ arbitraria. El resultado es una demostración de ϕ en CP. ■

8.3 El Teorema de la Deducción

El Teorema de la Deducción tal y como se enunció en el Capítulo 5 (Teo. 5.5) no va a ser verdadero en general para el CP, pues aplicando Gen se tiene que $Px \vdash \forall x Px$

y sin embargo, $\not\vdash Px \Rightarrow \forall x Px$ ya que no es una fórmula universalmente válida. No obstante, con ciertas restricciones, el Teorema de la Deducción se puede demostrar para el CP.

Definición. Sea Γ un conjunto de fórmulas de $\mathscr{L}$ y $\phi \in \Gamma$. Sea $\phi_1, \ldots, \phi_n$ una deducción en el CP a partir de Γ, junto con la justificación de cada paso. Se dice que ϕ_i *depende de* ϕ en esta deducción si y sólo si:

(a) $\phi_i = \phi$ y la justificación para ϕ_i es que pertenece a Γ, o

(b) ϕ_i se justificó en la deducción como consecuencia de fórmulas anteriores por MP o Gen y alguna de esas fórmulas anteriores depende de ϕ en la deducción.

Ejemplo. $\beta \vdash \forall x(\alpha \Rightarrow \beta)$

(1)	β	hipótesis
(2)	$\beta \Rightarrow (\alpha \Rightarrow \beta)$	A1
(3)	$\alpha \Rightarrow \beta$	1, 2 MP
(4)	$\forall x(\alpha \Rightarrow \beta)$	3, Gen

En esta deducción todas las fórmulas, excepto la segunda, dependen de β.

Teorema 8.2. *Si* $\Gamma, \alpha \vdash \beta$ *y en esta deducción* β *no depende de* α, *entonces* $\Gamma \vdash \beta$.

Demostración.

Este teorema justifica el uso de la palabra "depende" en la definición anterior. Si β no depende de α en una deducción, esto quiere decir que α era irrelevante y por tanto se puede obtener β aun si eliminamos a α del conjunto de las hipótesis.

Sea $\beta_1, \ldots, \beta_n = \beta$ una deducción de β a partir de $\Gamma \cup \{\alpha\}$ en la cual β no depende de α. Probaremos que $\Gamma \vdash \beta$ por inducción sobre n.

Base inductiva: $n = 1$

En este caso $\beta = \beta_1$ y por tanto β es un axioma de CP o $\beta \in \Gamma$ (β no puede ser α porque β no depende de α en la deducción). En ambos casos se tiene que $\Gamma \vdash \beta$.

Supongamos que el teorema es verdadero para toda deducción de menos de n pasos y supongamos que la deducción de β tiene n pasos.

Si β es axioma o está en Γ, entonces $\Gamma \vdash \beta$. Si β es consecuencia de anteriores por MP o Gen entonces, por definición, esas fórmulas tampoco dependen de α y por hipótesis de inducción se pueden deducir a partir de Γ. Por tanto $\Gamma \vdash \beta$. ■

Teorema 8.3 (de la Deducción). *Supóngase que* $\Gamma, \alpha \vdash \beta$ *y que en la deducción, si se aplica* Gen *a una fórmula que depende de* α, *la variable cuantificada en la regla no aparece libre en* α. *Entonces* $\Gamma \vdash \alpha \Rightarrow \beta$.

Demostración.

Sea $\beta_1, \ldots, \beta_n = \beta$ una deducción de β a partir de $\Gamma \cup \{\alpha\}$ que satisface las hipótesis del teorema. Probamos por inducción sobre n que $\Gamma \vdash \alpha \Rightarrow \beta_i$, para toda $i \in \{1, \ldots, n\}$.

Base inductiva: $n = 1$

Entonces β_1 es axioma o está en Γ o es α.

Si β_1 es axioma o está en Γ entonces, como $\beta_1 \Rightarrow (\alpha \Rightarrow \beta_1)$ es una instancia de A1, aplicando MP se obtiene $\Gamma \vdash \alpha \Rightarrow \beta_1$.

Si $\beta_1 = \alpha$ entonces $\alpha \Rightarrow \beta_1$ es $\alpha \Rightarrow \alpha$, que es un teorema de CP por ser instancia de tautología y en consecuencia $\vdash \alpha \Rightarrow \alpha$.

Supongamos ahora el teorema cierto para toda $i < n$ y consideremos a β_n.

Si β_n es axioma, está en Γ o es α se procede de la misma manera que para el caso $n = 1$.

Supongamos que β_n es consecuencia de dos fórmulas anteriores por MP. Entonces existen $j, k < n$ tales que $\beta_j = \beta_k \Rightarrow \beta_n$. Por hipótesis de inducción $\Gamma \vdash \alpha \Rightarrow \beta$ y también $\Gamma \vdash \alpha \Rightarrow (\beta_k \Rightarrow \beta_n)$. Aplicando A2 y MP se obtiene que $\Gamma \vdash \alpha \Rightarrow \beta_n$.

El último caso es cuando β_n es consecuencia de alguna fórmula β_j por Gen, para alguna $j < n$. Entonces $\beta_n = \forall x_i \beta_j$. La hipótesis de inducción garantiza que $\Gamma \vdash \alpha \Rightarrow \beta_j$ y las hipótesis del teorema garantizan que β_j no depende de α en la deducción o que x_i no aparece libre en α.

Si β_j no depende de α en la deducción entonces, por el Teorema 8.2, $\Gamma \vdash \beta_j$. Aplicando Gen se obtiene $\Gamma \vdash \forall x_i \beta_j$, es decir, $\Gamma \vdash \beta_n$. Como $\beta_n \Rightarrow (\alpha \Rightarrow \beta_n)$ es una instancia de A1, aplicando MP se obtiene que $\Gamma \vdash \alpha \Rightarrow \beta_n$.

Supongamos que x_i no ocurre libre en α. Entonces, por hipótesis de inducción tenemos que $\Gamma \vdash \alpha \Rightarrow \beta_j$ y aplicando Gen obtenemos $\Gamma \vdash \forall x_i(\alpha \Rightarrow \beta)$. Como x_i no ocurre libre en α, podemos aplicar A5 y MP para obtener $\Gamma \vdash \alpha \Rightarrow \forall x_i \beta_j$, es decir, $\Gamma \vdash \alpha \Rightarrow \beta_n$. ■

Obsérvese que si α es un enunciado entonces las hipótesis adicionales del Teorema de la Deducción son siempre satisfechas y tenemos, por tanto, el siguiente corolario.

Corolario 8.4. *Si α es un enunciado y $\Gamma, \alpha \vdash \beta$ entonces $\Gamma \vdash \alpha \Rightarrow \beta$.* ■

Ejemplos:

1. $\vdash \forall x \forall y \alpha \Rightarrow \forall y \forall x \alpha$

(1) $\forall x \forall y \alpha$	hipótesis
(2) $\forall x \forall y \alpha \Rightarrow \forall y \alpha$	A4
(3) $\forall y \alpha$	1,2 MP
(4) $\forall y \alpha \Rightarrow \alpha$	A4
(5) α	3,4 MP
(6) $\forall x \alpha$	5, Gen
(7) $\forall y \forall x \alpha$	6, Gen

Los pasos 1-7 demuestran que $\forall x \forall y \alpha \vdash \forall y \forall x \alpha$, como ni x ni y aparecen libres en $\forall x \forall y \alpha$, podemos aplicar el Teorema de la Deducción y obtener el resultado deseado.

2. $\vdash Ax \Rightarrow \exists x Ax$

(1) Ax	hipótesis
(2) $Ax \Rightarrow \neg\neg Ax$	instancia de tautología
(3) $\neg\neg Ax$	1,2 MP
(4) $\forall x \neg Ax \Rightarrow \neg Ax$	A4
(5) $(\forall x \neg Ax \Rightarrow \neg Ax) \Rightarrow (\neg\neg Ax \Rightarrow \neg \forall x \neg Ax)$	instancia de tautología
(6) $\neg\neg Ax \Rightarrow \neg \forall x \neg Ax$	4,5 MP
(7) $\neg \forall x \neg Ax$	3,6 MP

Los pasos 1-7 demuestran que $Ax \vdash \exists x Ax$ (recuérdese que cuando se quiere probar una fórmula con símbolos que no están en el lenguaje del CP se utilizan las equivalencias dadas en la primera sección de este capítulo para transformarla en una fórmula del lenguaje del CP); como en la deducción no se aplicó Gen, podemos aplicar el teorema de la deducción y concluir que $\vdash Ax \Rightarrow \exists x Ax$.

Ejercicios

Probar los siguientes teoremas de CP:

a. $\forall x(\alpha \Rightarrow \beta) \Rightarrow (\forall x\alpha \Rightarrow \forall x\beta)$

b. $\forall x\alpha \Rightarrow \exists x\alpha$

c. $\exists x\exists y\alpha \Rightarrow \exists y\exists x\alpha$

d. $\forall x(\alpha \Rightarrow \beta) \Rightarrow \forall x(\neg\beta \Rightarrow \neg\alpha)$

e. $\forall x(\alpha \Rightarrow \beta) \Rightarrow (\exists x\alpha \Rightarrow \exists x\beta)$

f. $\forall x(\alpha \wedge \beta) \Rightarrow (\forall x\alpha \wedge \forall x\beta)$

g. $\exists x(\alpha \wedge \beta) \Rightarrow (\exists x\alpha \wedge \exists x\beta)$

h. $\exists x(Px \vee Qx) \Rightarrow (\exists xPx \vee \exists xQx)$

8.4 Validez y completud para CP

En esta sección demostraremos para el cálculo de predicados los teoremas equivalentes a 5.7 y 5.9.

Teorema 8.5 (de Validez del CP). *Todo teorema del CP es universalmente válido.*

Demostración.

Sea ϕ un teorema, probamos por inducción en el número de pasos de la demostración de ϕ en el CP que ϕ es universalmente válida. Para esto es suficiente con probar que toda instancia de A1–A5 es universalmente válida y que las reglas MP y Gen preservan validez universal.

Probamos que toda instancia de A1 es universalmente válida y dejamos A2 y A3 como ejercicio para el lector. Sea $\mathfrak{A}$ una $\mathscr{L}$-estructura, s una función de V en A, el dominio de $\mathfrak{A}$. Entonces $\mathfrak{A} \not\models \alpha \Rightarrow (\beta \Rightarrow \alpha)[s]$ si y sólo si

$$(1)\ \mathfrak{A} \vdash \alpha[s]$$

y

$$(2)\ \mathfrak{A} \not\models (\beta \Rightarrow \alpha)[s],$$

pero esto último sólo sucede si $\vDash \beta[s]$ y $\nvDash \alpha[s]$. Esta contradicción demuestra que no pueden existir $\mathfrak{A}$ y s tales.

Supongamos ahora, para probar que toda instancia de A5 es universalmente válida, que $\mathfrak{A} \nvDash \forall x(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \forall x\beta)[s]$ para alguna estructura $\mathfrak{A}$ y $s: V \to A$. Entonces

$$(1)\ \mathfrak{A} \vDash \forall x(\alpha \Rightarrow \beta)[s]$$

y

$$(2)\ \mathfrak{A} \nvDash \alpha \Rightarrow \forall x\beta[s].$$

Esto último implica que

$$(3)\ \mathfrak{A} \vDash \alpha[s]$$

y

$$(4)\ \mathfrak{A} \nvDash \forall x\beta[s].$$

Por tanto existe $a \in A$ tal que $\mathfrak{A} \nvDash \beta[s(x/a)]$ y como x no ocurre libre en α, las sucesiones s y $s(x/a)$ coinciden en todas las variables libres de α. Aplicando el Teorema 7.1, de (3) podemos concluir que $\mathfrak{A} \vDash \alpha[s(x/a)]$, por lo que, $\mathfrak{A} \nvDash \alpha \Rightarrow \beta[s(x/a)]$. Esto contradice (1).

Por último, sea $\phi = \forall x\alpha(x) \Rightarrow \alpha(t)$ una instancia de A4 y sean $\mathfrak{A}$ y s arbitrarias. Supongamos que $\mathfrak{A} \nvDash \phi[s]$. Entonces:

$$(1)\ \mathfrak{A} \vDash \forall x\alpha(x)[s]$$

y

$$(2)\ \mathfrak{A} \nvDash \alpha(t)[s]$$

Por (1) tenemos que para toda $a \in A$, $\mathfrak{A} \vDash \alpha[s(x/a)]$, en particular, esto es cierto para $a = \bar{s}(t)$. Para obtener el resultado basta con probar que si t es libre para x en α, entonces $\mathfrak{A} \vDash \alpha(t)[s]$ si y sólo si $\mathfrak{A} \vDash \alpha[s(x/\bar{s}(t)]$. Esto se hace por inducción sobre la complejidad de α.

Falta ver que las reglas de inferencia de CP preservan validez universal.

MP preserva verdad y por tanto validez universal.

Supongamos ahora que α es universalmente válida y probaremos que $\forall x\alpha$ también. Sean $\mathfrak{A}$ y s arbitrarias, entonces $\mathfrak{A} \vDash \forall x\alpha$ si y sólo si para toda $a \in A$ $\mathfrak{A} \vDash \alpha[s(x/a)]$, pero esto último es cierto porque α es universalmente válida.

■

Para probar el Teorema de Completud para el CP, necesitamos considerar teorías arbitrarias en algún lenguaje de primer orden $\mathscr{L}$.

> **Definición.** Sea $\mathscr{L}$ un lenguaje de primer orden, una *teoría de primer orden* o una *teoría* en $\mathscr{L}$ se obtiene agregando a los axiomas del CP, una lista de fórmulas de $\mathscr{L}$ que serán los *axiomas propios* de la teoría, también llamados *axiomas no lógicos.* Los axiomas del CP son los *axiomas lógicos.*

Obsérvese que el CP es, así definidas las cosas, la menor teoría de primer orden, en el sentido de que todo teorema del CP es un teorema de cualquier teoría de primer orden.

> **Definición.** Una teoría $\mathscr{T}$ es *consistente* si y sólo si no existe ninguna fórmula ϕ de su lenguaje tal que tanto ϕ como $\neg\phi$ son teoremas de $\mathscr{T}$.

Notación:

1) Si ϕ es un teorema de $\mathscr{T}$, se escribe $\mathscr{T} \vdash \phi$ o $\vdash_{\mathscr{T}} \phi$.

2) Si $\Gamma \cup \{\phi\}$ es un conjunto de fórmulas del lenguaje de $\mathscr{T}$, $\Gamma \vdash_{\mathscr{T}} \phi$ quiere decir que ϕ se puede deducir a partir de $\mathscr{T}$ si se aceptan como hipótesis adicionales a todos los elementos de Γ.

Corolario 8.6. *El CP es una teoría consistente.*

Demostración.

Si fuera inconsistente existiría una fórmula ϕ de $\mathscr{L}$ tal que $\vdash \phi$ y $\vdash \neg\phi$. Por el Teorema de Validez, tanto ϕ como $\neg\phi$ serían universalmente válidas, lo cual es una contradicción. ■

Teorema 8.7 (de Completud del CP). *Toda fórmula universalmente válida de $\mathscr{L}$ es un teorema del CP.*

La demostración del Teorema 8.7 es bastante más compleja que la del correspondiente 5.9, por lo que necesitamos algunos lemas y definiciones auxiliares.

> **Definición.** Sean x_i y x_j dos variables distintas de $\mathscr{L}$, ϕ una fórmula de $\mathscr{L}$. Decimos que $\phi(x_i)$ y $\phi(x_j)$ son *similares* si y sólo si x_j es libre para x_i en $\phi(x_i)$ y $\phi(x_i)$ no tiene ocurrencias libres de x_j. (Aquí suponemos que $\phi(x_j)$ se obtiene de $\phi(x_i)$ sustituyendo x_j por todas las ocurrencias libres de x_i en $\phi(x_i)$).

Lema 8.8. *Si $\phi(x_i)$ y $\phi(x_j)$ son similares, entonces* $\vdash \forall x_i \phi(x_i) \Leftrightarrow \forall x_j \phi(x_j)$.

Demostración.

Por A4 tenemos que $\vdash \forall x_i \phi(x_i) \Rightarrow \phi(x_j)$, y usando Gen se tiene que $\vdash \forall x_j(\forall x_i \phi(x_i) \Rightarrow \phi(x_j))$. Como $\phi(x_i)$ y $\phi(x_j)$ son similares, x_j no aparece libre en $\phi(x_j)$ y podemos aplicar A5, obteniendo $\vdash \forall x_i \phi(x_i) \Rightarrow \forall x_j \phi(x_j)$. Análogamente, se prueba que $\vdash \forall x_j \phi(x_j) \Rightarrow \forall x_i \phi(x_i)$. ■

Lema 8.9. *Sea ϕ un enunciado de $\mathscr{L}$ y sea $\mathscr{T}$ una teoría de primer orden tal que $\mathscr{T} \vdash \neg\phi$. Entonces la teoría $\mathscr{T}'$ que se obtiene al agregar ϕ a los axiomas de $\mathscr{T}$ es una teoría consistente.*

Demostración.

Supóngase que $\mathscr{T}'$ es inconsistente. Entonces existe una fórmula ψ tal que $\mathscr{T}' \vdash \psi$ y $\mathscr{T}' \vdash \neg\psi$. Entonces, por el Teorema 8.1, $\mathscr{T}' \vdash \psi \Rightarrow (\neg\psi \Rightarrow \neg\phi)$. Aplicando MP dos veces, tenemos que $\mathscr{T}' \vdash \neg\phi$ y por tanto $\phi \vdash_{\mathscr{T}} \neg\phi$. Como ϕ es un enunciado, podemos aplicar el Teorema de la Deducción y obtener $\mathscr{T} \vdash \phi \Rightarrow \neg\phi$. Otra vez, por el Teorema 8.1, $\mathscr{T} \vdash (\phi \Rightarrow \neg\phi) \Rightarrow \neg\phi$ y por MP $\mathscr{T} \vdash \neg\phi$. Esto contradice la hipótesis. ■

Observación. Análogamente, si ϕ es un enunciado de $\mathscr{L}$ y $\mathscr{T} \nvdash \phi$, entonces la teoría que se obtiene al agregar $\neg\phi$ a los axiomas de $\mathscr{T}$ es consistente.

La siguiente afirmación no la vamos a demostrar porque requiere un poco más de teoría de conjuntos que la que hasta ahora hemos utilizado, pero no es difícil de creer. Si el lenguaje $\mathscr{L}$ tiene un conjunto numerable de símbolos y las expresiones de $\mathscr{L}$ son sucesiones finitas de símbolos de $\mathscr{L}$, entonces el conjunto de expresiones no puede ser no numerable. (¡Recuérdese que la unión numerable de conjuntos numerables es numerable! *Cf.* la proposición 2.5)

Afirmación 8.10. *El conjunto de expresiones de $\mathscr{L}$ es numerable.*

Definición. Una teoría de primer orden $\mathscr{T}$ es *completa* si y sólo si, dado cualquier enunciado ϕ del lenguaje de la teoría, se tiene que $\mathscr{T} \vdash \phi$ o $\mathscr{T} \vdash \neg\phi$.

Lema 8.11 (de Lindenbaum). *Sea $\mathscr{T}$ una teoría de primer orden consistente. Entonces existe una teoría de primer orden $\mathscr{T}'$, consistente y completa, tal que:*

a. El lenguaje de $\mathscr{T}'$ es el mismo que el lenguaje de $\mathscr{T}$.

b. $\mathscr{T} \subseteq \mathscr{T}'$, es decir, para cualquier fórmula ϕ, si $\mathscr{T} \vdash \phi$, entonces $\mathscr{T}' \vdash \phi$.

Demostración.

Sea $\phi_1, \phi_2, \ldots, \phi_n, \ldots$ una enumeración de todos los enunciados del lenguaje de $\mathscr{T}$. Vamos a definir una sucesión de teorías por recursión, de la siguiente manera:

$$\mathscr{T}_o = \mathscr{T}$$

$$\mathscr{T}_{n+1} = \begin{cases} \mathscr{T}_n, & \text{si } \mathscr{T}_n \vdash \neg\phi_{n+1} \\ \mathscr{T}_n \cup \{\phi_{n+1}\} & \text{si } \mathscr{T}_n \nvdash \neg\phi_{n+1} \end{cases}$$

Aquí, $\mathscr{T}_n \cup \{\phi_{n+1}\}$ denota a la teoría obtenida de $\mathscr{T}_n$ agregando ϕ_{n+1} como axioma.

Obsérvese que para toda n se tiene que $\mathscr{T}_n \subseteq \mathscr{T}_{n+1}$, por definición. Ahora probaremos por inducción que cada $\mathscr{T}_n$ es consistente.

Base: $n = 0$.

$\mathscr{T}_o$ es consistente por hipótesis, pues $\mathscr{T}_o = \mathscr{T}$.

Supongamos que $\mathscr{T}_n$ es consistente. Entonces, si $\mathscr{T}_{n+1} = \mathscr{T}_n$, por H.I., $\mathscr{T}_{n+1}$ es consistente. Y si $\mathscr{T}_{n+1} = \mathscr{T}_n \cup \{\phi_{n+1}\}$, entonces $\mathscr{T}_n \vdash \neg\phi_{n+1}$ y por el Lema 8.9, $\mathscr{T}_{n+1}$ es consistente.

Definimos ahora $\mathscr{T}' = \bigcup_{n=o}^{\infty} \mathscr{T}_n$, esto es, $\mathscr{T}'$ tiene como axiomas a todos los axiomas de $\mathscr{T}_n$, para toda n.

Evidentemente, $\mathscr{T} \subseteq \mathscr{T}'$ y $\mathscr{T}'$ tiene el mismo lenguaje que $\mathscr{T}$. Tenemos que probar que $\mathscr{T}'$ es consistente y completa.

Si $\mathscr{T}'$ fuera inconsistente, habría una deducción en $\mathscr{T}'$ de $\phi \wedge \neg\phi$, para algún enunciado ϕ. Pero las deducciones son finitas y por tanto sólo un número finito de axiomas pueden aparecer en la deducción, esto significa que la deducción se puede hacer a partir de los axiomas de alguna $\mathscr{T}_n$, pero esto es imposible, pues ya probamos que cada $\mathscr{T}_n$ es consistente.

Para ver que $\mathscr{T}'$ es completa, sea ϕ cualquier enunciado del lenguaje. Entonces $\phi = \phi_{n+1}$ para alguna $n = 0, 1, 2, \ldots$ Ahora bien, $\mathscr{T}_n \vdash \neg\phi_{n+1}$ o $\mathscr{T}_n \nvdash \neg\phi_{n+1}$. Si $\mathscr{T}_n \vdash \neg\phi_{n+1}$ entonces $\mathscr{T}' \vdash \neg\phi_{n+1}$ (ya que $\mathscr{T}_n \subseteq \mathscr{T}'$).

Si $\mathscr{T}_n \nvdash \neg\phi_{n+1}$ entonces, por definición, $\mathscr{T}_{n+1} \vdash \phi_{n+1}$ y por tanto $\mathscr{T}' \vdash \phi_{n+1}$.

Por consiguiente, $\mathscr{T}' \vdash \neg\phi_{n+1}$ o $\mathscr{T}' \vdash \phi_{n+1}$ y $\mathscr{T}'$ es completa. ■

El siguiente teorema fue demostrado por Gödel en 1930, aunque la demostración que vamos a dar es la de Henkin (1949).

Teorema 8.12. *Toda teoría de primer orden consistente tiene un modelo, esto es, una estructura bajo la cual todos los teoremas de la teoría son verdaderos.*

Demostración.

Sea $\mathscr{T}$ una teoría de primer orden consistente, en el lenguaje $\mathscr{L}$. Agreguemos a $\mathscr{L}$ un conjunto numerable de nuevas constantes, $\{b_1, b_2, \ldots\}$. Denotemos al nuevo lenguaje $\mathscr{L}_0$ y a la teoría obtenida de $\mathscr{T}$ agregando los nuevos axiomas que resulten de considerar a todas las fórmulas y términos de $\mathscr{L}_0$ por $\mathscr{T}_0$, de modo que los axiomas de $\mathscr{T}_0$ son todos los axiomas de $\mathscr{T}$ junto con los axiomas lógicos que involucren a las nuevas constantes.

Se afirma que $\mathscr{T}_0$ es consistente, pues si no lo fuera tendríamos que $\mathscr{T}_0 \vdash \phi \wedge \neg\phi$, para alguna fórmula ϕ de $\mathscr{L}_0$. Reemplacemos cada ocurrencia de alguna b_i en la demostración de $\phi \wedge \neg\phi$ por una nueva variable, de manera uniforme. Esto transforma axiomas en axiomas y mantiene la aplicación correcta de reglas de inferencia. La fórmula al final de esta nueva deducción es una contradicción en la que no intervienen las nuevas constantes y por tanto, es una deducción en $\mathscr{T}$. Esto contradice la consistencia de $\mathscr{T}$ y por lo tanto $\mathscr{T}_0$ es consistente.

Sea $\phi_1(x_{i_1}), \phi(x_{i_2}), \ldots, \phi_k(x_{i_k}), \ldots$ una enumeración de todas las fórmulas de $\mathscr{L}_0$ que tienen a lo más una variable libre. (Aquí, x_{i_k} es la variable libre de ϕ_k si ϕ_k tiene variables libres, y $x_{i_k} = x_1$ si ϕ_k es un enunciado).

Escogemos ahora una sucesión $b_{j_1}, b_{j_2} \ldots$ de entre las nuevas constantes individuales de tal forma que b_{j_k} no aparece en $\phi_1(x_{i_1}), \phi_2(x_{i_2}), \ldots, \phi_k(x_{i_k})$ y tal que $b_{j_k} \neq b_{j_1}, b_{j_2}, \ldots, b_{j_{k-1}}$.

Consideremos la siguiente fórmula de $\mathscr{L}_0$, para cada k:

$$\psi_k = \neg\forall x_{i_k} \phi_k(x_{i_k}) \Rightarrow \neg\phi_k(b_{j_k})$$

ψ_k explica el papel que juegan las constantes b_{j_k}: si ϕ_k no es "verdadera" para todos los individuos, entonces no es "verdadera" para b_{j_k}. b_{j_k} son conocidas como *testigos.*[1]

[1]Nótese que en realidad no tenemos derecho de hablar de "verdad" en una teoría formal, el comentario es simplemente de ayuda para entender lo que se hace.

Para cada número natural n, sea $\mathscr{T}_n$ la teoría que se obtiene a partir de $\mathscr{T}_o$ al agregar como axiomas a las fórmulas $\psi_1, \psi_2, \ldots, \psi_n$. Y sea $\mathscr{T}_\infty$ la teoría que se obtiene de $\mathscr{T}_o$ al agregar como axiomas a todas las ψ_i, $i = 1, 2, \ldots$

Al igual que en el Lema de Lindenbaum, para probar que $\mathscr{T}_\infty$ es consistente será suficiente con demostrar que cada $\mathscr{T}_n$ lo es. Probaremos pues, por inducción, que cada $\mathscr{T}_n$ es consistente.

Base: $n = 0$.

$\mathscr{T}_o$ es consistente, ya demostrado.

Paso inductivo:

Supongamos que $\mathscr{T}_{n-1}$ es consistente pero que $\mathscr{T}_n$ no lo es, para $n \geq 1$. En este caso, como $\phi_1 \Rightarrow (\neg\phi_1 \Rightarrow \phi_2)$ es una instancia de tautología para cualesquiera fórmulas ϕ_1 y ϕ_2, tenemos que cualquier fórmula es teorema de $\mathscr{T}_n$. En particular, $\mathscr{T} \vdash \neg\psi_n$.

Por lo tanto, $\psi_n \vdash_{\mathscr{T}_{n-1}} \neg\psi_n$, y como ψ_n es un enunciado, podemos aplicar el Teorema de la Deducción para obtener:

$$\mathscr{T}_{n-1} \vdash (\psi_n \Rightarrow \neg\psi_n)$$

Pero $(\psi_n \Rightarrow \neg\psi_n) \Rightarrow \neg\psi_n$ es una instancia de tautología, por lo que, por MP y el Teorema 8.1, tenemos que $\mathscr{T}_{n-1} \vdash \neg\psi_n$.

Sustituyendo, obtenemos:

$$\mathscr{T}_{n-1} \vdash \neg(\neg\forall x_{i_n}\phi_n(x_{i_n}) \Rightarrow \neg\phi_n(b_{j_n})),$$

esto es,

$$\mathscr{T}_{n-1} \vdash \neg\forall x_{i_n}\phi_n(x_{i_n}) \quad \text{y} \quad \mathscr{T}_{n-1} \vdash \phi_n(b_{j_n}) \tag{1}$$

(Esto último se obtuvo utilizando las tautologías siguientes: $\neg(A \Rightarrow B) \Rightarrow (A \wedge \neg B)$, $(A \wedge B) \Rightarrow A$, $(A \wedge B) \Rightarrow B$, $\neg\neg A \Rightarrow A$.)

Del hecho que b_{j_n} no ocurre en $\psi_1, \psi_2, \ldots, \psi_{n-1}$ y de que $\mathscr{T}_{n-1} \vdash \phi_n(b_{j_n})$, podemos concluir que $\mathscr{T}_{n-1} \vdash \phi_n(x_p)$, donde x_p es una variable individual que no ocurre en la demostración de $\phi_n(b_{j_n})$ a partir de $\mathscr{T}_{n-1}$, basta reemplazar b_{j_n} por x_p en la deducción.

Aplicando Gen obtenemos que $\mathscr{T}_{n-1} \vdash \forall x_p\phi_n(x_p)$, y como $\phi_n(x_p)$ y $\phi_n(x_{i_n})$ son similares, por el Lema 8.8 tenemos que $\mathscr{T}_{n-1} \vdash \forall x_{i_n}\phi_n(x_{i_n})$.

Esto último, junto con (1), contradice la consistencia de $\mathscr{T}_{n-1}$, y en consecuencia $\mathscr{T}_n$ es consistente.

Así, cada $\mathscr{T}_n$ es consistente y se sigue que $\mathscr{T}_\infty$ es consistente.

Sea $\mathscr{T}^*$ una extensión completa y consistente de $\mathscr{T}_\infty$. Vamos a construir un modelo para $\mathscr{T}^*$, y como $\mathscr{T} \subseteq \mathscr{T}_0 \subseteq \mathscr{T}_\infty \subseteq \mathscr{T}^*$, éste será un modelo para $\mathscr{T}$, con lo que terminará la demostración del teorema.

Definimos, pues, a la estructura $\mathfrak{A}$:

1) El dominio de $\mathfrak{A}$, A, es el conjunto de términos de $\mathscr{L}_0$ que no contienen variables.

2) Si c es una constante de $\mathscr{L}_0$, $c^{\mathfrak{A}} = c$.

3) Si f es un símbolo funcional n-ario de $\mathscr{L}_0$, $f^{\mathfrak{A}}: A^n \to A$ tal que

$$f^{\mathfrak{A}}(t_1, \ldots, t_n) = f t_1 \ldots t_n.$$

(Nótese que $f t_1 \ldots t_n \in A$, pues si $t_1, \ldots, t_n$ no tienen variables, $f t_1 \ldots t_n$ es un término de $\mathscr{L}_0$ que no tiene variables.)

4) Si P es un predicado n-ario de $\mathscr{L}_0$, $P^{\mathfrak{A}} \subseteq A^n$ es la relación n-aria definida como sigue:

$$(t_1, \ldots, t_n) \in P^{\mathfrak{A}} \text{ si y sólo si } \mathscr{T}^* \vdash P t_1 \ldots t_n.$$

(Esta definición es buena pues $\mathscr{T}^*$ es completa y consistente, y por consiguiente, dada una n-ada de términos $(t_1, \ldots, t_n)$, $\mathscr{T}^* \vdash P t_1 \ldots t_n$ o $\mathscr{T}^* \vdash \neg P t_1 \ldots t_n$, pero no ambas.)

Para probar que $\mathfrak{A} \models \mathscr{T}^*$, probaremos que para todo enunciado ϕ, $\mathfrak{A} \models \phi$, si y sólo si $\mathscr{T}^* \vdash \phi$. Esto lo haremos por inducción sobre la complejidad de ϕ.

Base: ϕ es un enunciado atómico.

En este caso ϕ es de la forma $P t_1 \ldots t_n$, donde P es un predicado n-ario de $\mathscr{L}_0$ y $t_1, \ldots, t_n$ son términos de $\mathscr{L}_0$ sin variables. Por definición, $\mathfrak{A} \models \phi$ si y sólo si $\mathscr{T}^* \vdash \phi$.

Hipótesis inductiva: Supongamos que para todo enunciado ϕ con menos de n conectivos y cuantificadores se tiene que $\mathfrak{A} \models \phi$ si y sólo si $\mathscr{T}^* \vdash \phi$.

Sea ψ con n conectivos y cuantificadores.

Caso 1. ψ es $\neg\phi$.

$\mathfrak{A} \models \psi$ si y sólo si $\mathfrak{A} \models \neg\phi$ si y sólo si $\mathfrak{A} \not\models \phi$ si y sólo si $\mathscr{T}^* \not\vdash \phi$, por H.I.

Pero $\mathscr{T}^*$ es completa y consistente, por tanto, $\mathscr{T}^* \not\vdash \phi$ si y sólo si $\mathscr{T}^* \vdash \neg\phi$. Por tanto, $\mathfrak{A} \models \psi$ si y sólo si $\mathscr{T}^* \vdash \psi$.

Caso 2. ψ es $\phi_1 \Rightarrow \phi_2$.

Como ψ es un enunciado, también lo son ϕ_1 y ϕ_2, y para ellos vale la H.I.

Supongamos que $\mathfrak{A} \nvDash \psi$. Entonces $\mathfrak{A} \vDash \phi_1$ y $\mathfrak{A} \nvDash \phi_2$. Por la H.I., tenemos que $\mathscr{T}^* \vdash \phi_1$ y $\mathscr{T}^* \nvdash \phi_2$; como $\mathscr{T}^*$ es completa, $\mathscr{T}^* \vdash \neg\phi_2$. Usando la tautología $A \Rightarrow (\neg B \Rightarrow \neg(A \Rightarrow B))$, obtenemos que $\mathscr{T}^* \vdash \neg(\phi_1 \Rightarrow \phi_2)$, y de la consistencia de $\mathscr{T}^*$, $\mathscr{T}^* \nvdash \psi$.

Conversamente, supongamos ahora que $\mathscr{T}^* \nvdash \psi$. Por ser completa, $\mathscr{T}^* \vdash \neg\psi$, esto es, $\mathscr{T}^* \vdash \neg(\phi_1 \Rightarrow \phi_2)$. Nuevamente, usando las tautologías adecuadas, tenemos $\mathscr{T}^* \vdash \phi_1$ y $\mathscr{T}^* \vdash \neg\phi_2$, y por consistencia, $\mathscr{T}^* \nvdash \phi_2$. Por H.I., tenemos que $\mathfrak{A} \vDash \phi_1$ y $\mathfrak{A} \nvDash \phi_2$, y por tanto, $\mathfrak{A} \nvDash \psi$.

Caso 3. ψ es $\forall x_n \phi$.

Como ψ es un enunciado, ϕ tiene a lo sumo una variable libre, y consecuentemente $\phi = \phi_k(x_{i_k})$ para alguna k.

Podemos suponer que $x_n = x_{i_k}$, ya que en caso contrario ϕ no tendría variables libres y por tanto $\mathfrak{A} \vDash \psi$ si y sólo si $\mathfrak{A} \vDash \phi$ y $\mathscr{T}^* \vdash \psi$ si y sólo si $\mathscr{T}^* \vdash \phi$. En este caso el resultado para ψ se seguiría inmediatamente del resultado para ϕ.

Supongamos, pues, que $x_n = x_{i_k}$ y que $\mathfrak{A} \vDash \psi$, pero $\mathscr{T}^* \nvdash \psi$. Por la completud de $\mathscr{T}^*$, $\mathscr{T}^* \vdash \neg\psi$, es decir, $\mathscr{T}^* \vdash \neg\forall x_{i_k} \phi_k(x_{i_k})$. Pero, $\mathscr{T}^* \vdash \psi_k$, ya que $\mathscr{T}_\infty \subseteq \mathscr{T}^*$, es decir, $\mathscr{T}^* \vdash \neg\forall x_{i_k} \phi_k(x_{i_k}) \Rightarrow \neg\phi_k(b_{i_k})$, y por tanto, $\mathscr{T}^* \vdash \neg\phi_k(b_{i_k})$.

Ahora bien, como $\mathfrak{A} \vDash \psi$, $\mathfrak{A} \vDash \forall x_{i_k} \phi_k(x_{i_k})$, y en consecuencia $\mathfrak{A} \vDash \phi_k(b_{i_k})$, y por la H.I. se tiene que $\mathscr{T}^* \vdash \phi_k(b_{i_k})$, contradiciendo la consistencia de $\mathscr{T}^*$. Por tanto, $\mathfrak{A} \vDash \psi$ si y sólo si $\mathscr{T} \vdash \psi$.

Para el converso, supongamos que $\mathscr{T}^* \vdash \psi$ y $\mathfrak{A} \nvDash \psi$.

Como $\mathfrak{A} \nvDash \forall x_{i_k} \phi_k(x_{i_k})$, existe $t \in A$ tal que $\mathfrak{A} \nvDash \phi_k(x_{i_k})[t]$ y por H.I., esto significa que $\mathscr{T}^* \nvdash \phi(t)$, donde t es un término de $\mathscr{L}_0$ sin variables. Como $\mathscr{T}^*$ es completa, se tiene que $\mathscr{T}^* \vdash \neg\phi_k(t)$.

Por otro lado, si $\mathscr{T}^* \vdash \forall x_{i_k} \phi_k(x_{i_k})$, entonces $\mathscr{T}^* \vdash \phi_k(t)$, por A4. Esto contradice la consistencia de $\mathscr{T}^*$.

Hemos probado que para todo enunciado ϕ, si $\mathscr{T}^* \vdash \phi$ entonces $\mathfrak{A} \vDash \phi$.

Si $\mathscr{T} \vdash \phi$, como $\mathscr{T} \subseteq \mathscr{T}^*$, entonces $\mathscr{T}^* \vdash \phi$ y por lo tanto, $\mathfrak{A} \vDash \phi$.

Por consiguiente $\mathfrak{A} \vDash \mathscr{T}$. ■

Corolario 8.13. *Si $\mathscr{T}$ es una teoría consistente en un lenguaje de primer orden numerable, entonces $\mathscr{T}$ tiene un modelo numerable.*

Demostración.

Directo de la demostración del Teorema 8.12. ■

Ahora sí estamos en posición de demostrar el Teorema de Completud para el CP, que volvemos a enunciar.

Teorema 8.7 (de Completud del CP). *Toda fórmula universalmente válida de $\mathscr{L}$ es un teorema del CP.*

Demostración.

Es suficiente con considerar enunciados, pues ϕ es universalmente válida si y sólo si $\forall x_{i_1}\forall x_{i_2}\ldots\forall x_{i_n}\phi$ lo es, donde $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ son las variables libres de ϕ, y CP $\vdash \phi$ si y sólo si CP $\vdash \forall x_{i_1}\forall x_{i_2}\ldots\forall x_{i_n}\phi$.

Sea pues, ϕ, un enunciado universalmente válido y supongamos que CP $\nvdash \phi$. entonces, por el Lema 8.9, la teoría $\mathscr{T}$ cuyo único axioma no lógico es $\neg\phi$ es consistente.

Por el Lema 8.12, $\mathscr{T}$ tiene un modelo $\mathfrak{A}$ y consecuentemente $\mathfrak{A} \models \neg\phi$, pero como ϕ es universalmente válido, $\mathfrak{A} \models \phi$, y esto no es posible. ■

Concluimos esta sección con algunas consecuencias del Teorema de Completud.

Corolario 8.14. *Sea $\mathscr{T}$ una teoría de primer orden en un lenguaje numerable.*

(a) Si ϕ es un enunciado verdadero de $\mathscr{T}$, entonces $\mathscr{T} \vdash \phi$.

(b) Si para todo $\mathfrak{A} \models \mathscr{T}$, para todo $s\colon V \to A$, se tiene que $\mathfrak{A} \models \Gamma[s]$ implica que $\mathfrak{A} \models \phi[s]$, donde ϕ es una fórmula y Γ es un conjunto de fórmulas, entonces $\Gamma \vdash_{\mathscr{T}} \phi$.

(c) Si $\Gamma \models \phi$ entonces $\Gamma \vdash_{\mathscr{T}} \phi$.

(d) Si $\phi \models \psi$ entonces $\phi \vdash_{\mathscr{T}} \psi$.

Demostración.

(a) Si $\mathscr{T} \nvdash \phi$, entonces $\mathscr{T}' = \mathscr{T} \cup \{\neg\phi\}$[2] es consistente y por el Corolario 8.13 tiene un modelo numerable, $\mathfrak{A}$. Entonces $\mathfrak{A} \models \mathscr{T}$ y $\mathfrak{A} \models \neg\phi$, esto no es posible, pues contradice la hipótesis, por tanto, $\mathscr{T} \vdash \phi$.

[2]Aquí $\mathscr{T} \cup \Gamma$ es la teoría que se obtiene agregando a los axiomas de $\mathscr{T}$, todos los elementos de Γ.

(b) Considérese la teoría $\mathscr{T}'$[3] y obsérvese que ϕ es verdadera en todo modelo de $\mathscr{T} \cup \Gamma$.

(c) y (d) (Ejercicio 6). ■

Observaciones:

1. Nótese que se pueden dar los axiomas de los números reales como un campo ordenado en un lenguaje de primer orden numerable, y por el Corolario 8.13 tenemos que esta teoría tiene un modelo numerable.
2. El inciso (a) del Corolario 8.14 garantiza que si una afirmación es verdadera en todo grupo, por ejemplo, entonces se puede demostrar a partir de los axiomas de teoría de grupos.

Ejercicios

1. Probar que todas las instancias de los axiomas A2 y A3 son universalmente válidas.

2. Probar que si $\mathscr{T}$ es una teoría que no es consistente, entonces para toda fórmula ϕ del lenguaje de $\mathscr{T}$ se tiene que $\mathscr{T} \vdash \phi$.

3. Sea $\mathscr{T}$ una teoría de primer orden y ϕ una fórmula del lenguaje de $\mathscr{T}$ sin variables libres. Entonces $\mathscr{T} \vdash \phi$ si y sólo si $\mathscr{T} \vdash \forall x\phi$, donde x es cualquier variable individual.

4. Probar que si $\phi(x_i)$ y $\phi(x_j)$ son similares, entonces x_i es libre para x_j en $\phi(x_j)$ y $\phi(x_j)$ no tiene ocurrencias libres de x_i.

5. Probar que si $\phi(x_i)$ y $\phi(x_j)$ son similares, entonces $\vdash \exists x_1\phi(x_i) \Leftrightarrow \exists x_j\phi(x_j)$.

6. Sea $\mathscr{T}$ una teoría de primer orden en un lenguaje numerable. Sea $\Gamma \cup \{\phi\}$ un conjunto de fórmulas. Entonces,

i. Si $\Gamma \vDash \phi$, entonces $\Gamma \vdash_{\mathscr{T}} \phi$.

ii. Si $\phi \vDash \psi$ entonces $\phi \vdash_{\mathscr{T}} \psi$.

[3] *Idem.*

8.5 Formas normales prenexas

Definición. Una fórmula de la forma $Q_1x_1Q_2x_2\ldots Q_nx_n\phi$, donde cada Q_ix_i es $\exists x_i$ o $\forall x_i$, $x_i \neq x_j$ si $i \neq j$ y ϕ no contiene cuantificadores está en *forma normal prenexa.* (Nota: se acepta $n = 0$, cuando la fórmula no tiene cuantificadores).

En esta sección vamos a demostrar que toda fórmula es equivalente a una forma normal prenexa.

Note que en virtud del Teorema de Completud para CP, los símbolos $\models$ y $\vdash$ son intercambiables, de modo que si $\phi \models \psi$ y se tiene ϕ en alguna línea de alguna demostración, se puede introducir ψ, dado que existe una prueba de ψ a partir de ϕ.

Lema 8.15. *Sea ϕ y ψ fórmulas de $\mathscr{L}$. Entonces,*

1. $\vdash (\forall x\phi x \Rightarrow \psi) \Leftrightarrow \exists y(\phi y \Rightarrow \psi)$, *si y no es libre en ψ, $\phi(x)$ y $\phi(y)$ son similares.*
2. $\vdash (\exists x\phi x \Rightarrow \psi) \Leftrightarrow \forall y(\phi y \Rightarrow \psi)$, *si y no es libre en ψ, $\phi(x)$ y $\phi(y)$ son similares.*
3. $\vdash (\psi \Rightarrow \forall x\phi x) \Leftrightarrow \forall y(\psi \Rightarrow \phi y)$, *si y no es libre en ψ, $\phi(x)$ y $\phi(y)$ son similares.*
4. $\vdash (\psi \Rightarrow \exists x\phi x) \Leftrightarrow \exists y(\psi \Rightarrow \phi y)$, *si y no es libre en ψ, $\phi(x)$ y $\phi(y)$ son similares.*
5. $\vdash \neg\forall x\phi \Leftrightarrow \exists x\neg\phi$.
6. $\vdash \neg\exists x\phi \Leftrightarrow \forall x\neg\phi$.

Demostración.

Haremos la primera en detalle, dejando las demás como ejercicios.

(1) $\forall x\phi x \Rightarrow \psi$	hipótesis
(2) $\neg\exists y(\phi y \Rightarrow \psi)$	hipótesis
(3) $\neg\neg\forall y\neg(\phi y \Rightarrow \psi)$	2, definición de $\exists y$
(4) $\forall y\neg(\phi y \Rightarrow \psi)$	3, tautología $\neg\neg A \Rightarrow A$
(5) $\neg(\phi y \Rightarrow \psi)$	4, A4 y MP
(6) ϕy	5, tautología $\neg(A \Rightarrow B) \Rightarrow A$

(7)	$\neg\psi$	5, tautología $\neg(A \Rightarrow B) \Rightarrow \neg B$
(8)	$\forall y \phi y$	6, Gen
(9)	$\forall x \phi x$	8, ya que $\forall y \phi y \vDash \forall x \phi x$ si ϕx y ϕy son similares
(10)	ψ	1, 9 MP
(11)	$\psi \wedge \neg\psi$	7, 10 tautología

Por tanto,

$$\forall x \phi x \Rightarrow \psi, \neg\exists y(\phi y \Rightarrow \psi) \vdash \psi \wedge \neg\psi,$$

y por el Teorema de la Deducción,

$$\forall x \phi x \Rightarrow \psi \vdash \neg\exists y(\phi y \Rightarrow \psi) \Rightarrow (\psi \wedge \neg\psi).$$

Utilizando la tautología $(A \Rightarrow (B \wedge \neg B)) \Rightarrow \neg A$, obtenemos

$$\forall x \phi x \Rightarrow \psi \vdash \exists y(\phi y \Rightarrow \psi),$$

y aplicando nuevamente el Teorema de la Deducción, obtenemos

$$(a) \qquad \vdash (\forall x \phi x \Rightarrow \psi) \Rightarrow \exists y(\phi y \Rightarrow \psi)$$

Probaremos ahora la otra implicación:

(1)	$\exists y(\phi y \Rightarrow \psi)$	hipótesis
(2)	$\forall x \phi x$	hipótesis
(3)	ψ	1,2 MP, usando el hecho de que $\exists y(\phi y \Rightarrow \psi), \forall x \phi x \vDash \psi$, si ϕy y ϕx son similares, e y no es libre en ψ.

En consecuencia, $\exists y(\phi y \Rightarrow \psi), \forall x \phi x \vdash \psi$.

Aplicando el Teorema de la Deducción dos veces, obtenemos

$$(b) \qquad \vdash \exists y(\phi y \Rightarrow \psi) \Rightarrow (\forall x \phi x \Rightarrow \psi)$$

De (a) y (b), se tiene que: $\vdash (\forall x \phi x \Rightarrow \psi) \Leftrightarrow \exists y(\phi y \Rightarrow \psi)$. ■

Teorema 8.16. *Para toda fórmula ϕ de un lenguaje de primer orden $\mathscr{L}$, existe una fórmula ψ de $\mathscr{L}$ en forma normal prenexa tal que $\vdash_{CP} \phi \Leftrightarrow \psi$.*

Demostración.

Por inducción sobre el número k de conectivos y cuantificadores de ϕ.

Base: $k = 0$.

ϕ no tiene cuantificadores, y puede concluirse que está en forma normal prenexa. Sea $\psi = \phi$.

H.I. Supongamos la afirmación cierta para toda fórmula con menos de k conectivos y cuantificadores.

Sea ϕ con k conectivos y cuantificadores.

Caso 1. ϕ es $\neg\chi$. Por H.I., existe ψ' en forma normal prenexa tal que $\vdash_{CP} \chi \Leftrightarrow \psi'$; por tanto, $\vdash_{CP} \phi \Leftrightarrow \neg\psi'$.

Aplicando 5 y 6 del Lema 8.15, obtenemos una fórmula ψ en forma normal prenexa tal que $\vdash_{CP} \neg\psi' \Leftrightarrow \psi$; de donde, $\vdash_{CP} \phi \Leftrightarrow \psi$.

Caso 2. ϕ es $(\chi_1 \Rightarrow \chi_2)$. Por H.I., existen ψ_1 y ψ_2 en forma normal prenexa tales que $\vdash_{CP} \chi_1 \Leftrightarrow \psi_1$ y $\vdash_{CP} \chi_2 \Leftrightarrow \psi_2$. Por tanto, $\vdash_{CP} \phi \Leftrightarrow (\psi_1 \Rightarrow \psi_2)$. Aplicando (1)–(4) del Lema 8.15, podemos mover todos los cuantificadores que aparecen en ψ_1 y ψ_2 al principio, obteniendo una fórmula ψ en forma normal prenexa tal que $\vdash_{CP} \phi \Leftrightarrow \psi$.

Caso 3. ϕ es $\forall x\chi$. Por H.I., existe una fórmula ψ' en forma normal prenexa tal que $\vdash_{CP} \chi \Leftrightarrow \psi'$. Entonces, $\vdash_{CP} \forall x\chi \Leftrightarrow \forall x\psi'$, y $\forall x\psi'$ está en forma normal prenexa; sea $\psi = \forall x\psi'$, $\vdash_{CP} \phi \Leftrightarrow \psi$. ■

Ejemplo. Sea $\phi = \forall x(Ax \Rightarrow \exists y Bxy) \Rightarrow \forall z Cz$. Entonces,

$$\phi \equiv \forall x(\exists w(Ax \Rightarrow Bxw)) \Rightarrow \forall z Cz \equiv \forall z \forall x \exists w((Ax \Rightarrow Bxw) \Rightarrow Cz).$$

cláusula (4) cláusula (3)

□

Ejercicios

1. Encontrar fórmulas en forma normal prenexa equivalentes a las siguientes fórmulas:

a. $\forall x Px \Rightarrow \neg\exists y Qy$, **b.** $\forall x(Px \Rightarrow \exists y \exists w Qxyw)$,
c. $\neg\forall x(Px \Rightarrow Qa)$.

2. Probar las claúsulas (2)-(6) del Lema 8.15.

8.6 Teorema de Compacidad para lenguajes de primer orden

Teorema 8.17 (de Compacidad, primera versión). *Sea $\Gamma \cup \{\phi\}$ un conjunto de fórmulas de un lenguaje de primer orden. Entonces $\Gamma \models \phi$ si y sólo si existe un subconjunto finito $\Gamma_o \subseteq \Gamma$, tal que $\Gamma_o \models \phi$.*

Demostración.

$\Leftarrow$) Obvio. Si $\Gamma_o \subseteq \Gamma$ y $\Gamma_o \models \phi$, entonces $\Gamma \models \phi$.

$\Rightarrow$) Si $\Gamma \models \phi$, entonces, por el Corolario 8.14 (c), $\Gamma \vdash \phi$, pero las deducciones son finitas y consecuentemente existe $\Gamma_o \subseteq \Gamma$, Γ_o finito, tal que $\Gamma_o \vdash \phi$. Puede concluirse que, $\Gamma_o \models \phi$. ■

Teorema 8.18 (de Compacidad, segunda versión). *Si Γ es un conjunto de fórmulas y todo subconjunto finito de Γ tiene modelo, entonces Γ tiene modelo.*

Demostración.

Si todo subconjunto finito de Γ tiene modelo, entonces todo subconjunto finito de Γ es consistente, ya que si $\Gamma_o \vdash \phi \wedge \neg\phi$ y $\mathfrak{A} \models \Gamma_o$, se tendría que $\mathfrak{A} \models \phi \wedge \neg\phi$, lo cual no es posible.

Tenemos, pues, que todo subconjunto finito de Γ es consistente, y como las deducciones son finitas, esto implica que Γ es consistente.

Por el Teorema 8.12, Γ tiene un modelo. ■

Damos a continuación algunas aplicaciones de los Teoremas de Completud y de Compacidad a teorías matemáticas específicas.

Corolario 8.19. *Si una teoría $\mathscr{T}$ tiene modelos finitos de cardinalidades arbitrariamente grandes, entonces tiene un modelo infinito.*

Demostración.

Sea $\mathscr{T}$ una teoría en el lenguaje de primer orden $\mathscr{L}$. Consideremos el lenguaje $\mathscr{L}'$ que se obtiene al agregar a los símbolos de $\mathscr{L}$ un conjunto numerable de nuevas constantes, todas distintas. $\mathscr{L}' = \mathscr{L} \cup \{c_n : n \in \mathbb{N}\}$.

Sea $\mathscr{T}'$ la teoría que se obtiene al agregar a los axiomas de $\mathscr{T}$ los siguientes axiomas: $\neg(c_i = c_j)$, si $i \neq j$, para $i, j \in \mathbb{N}$.

$\mathscr{T}'$ es una teoría en el lenguaje $\mathscr{L}'$ y $\mathscr{T} \subseteq \mathscr{T}'$, es decir, si $\mathscr{T} \vdash \phi$, entonces $\mathscr{T}' \vdash \phi$.

Para ver que $\mathscr{T}'$ tiene modelos, aplicamos el Teorema de Compacidad y tomamos un subconjunto finito de $\mathscr{T}'$, llamémosle $\mathscr{S}$. $\mathscr{S}$ sólo puede involucrar a un número finito de las nuevas constantes, digamos $c_o, \ldots, c_m$.

Como $\mathscr{T}$ tiene modelos finitos arbitrariamente grandes, podemos escoger un modelo $\mathfrak{A}$ de $\mathscr{T}$ que tenga por lo menos $m + 1$ elementos. Construimos una interpretación $\mathfrak{A}'$ para $\mathscr{L}'$ con el mismo universo de $\mathfrak{A}$, A, las mismas interpretaciones para los símbolos de $\mathscr{L}$ y tal que las constantes $c_o, \ldots, c_m$ se interpreten como elementos distintos de A.

Entonces, $\mathfrak{A}' \models \mathscr{S}$. Por compacidad, $\mathscr{T}'$ tiene un modelo, $\mathfrak{B}$, digamos. $\mathfrak{B}$ es infinito y $\mathfrak{B} \models \mathscr{T}$. ■

El teorema siguiente requiere de un cierto conocimiento de cardinales infinitos. El lector no familiarizado con estos temas puede omitir su lectura sin pérdida de continuidad.

Teorema 8.20 (de Löwenheim-Skolem). *Si una teoría $\mathscr{T}$ de primer orden tiene modelos infinitos, tiene modelos de cualquier cardinalidad infinita.*

Demostración.

Es muy similar al corolario anterior. Sea $\mathscr{L}$ el lenguaje de $\mathscr{T}$ y agreguemos, dado un cardinal infinito α arbitrario, un conjunto de α nuevas constantes, es decir, sea $\mathscr{L}' = \mathscr{L}\{c_\iota\}_{\iota<\alpha}$.

Sea $\mathscr{T}'$ la teoría que se obtiene al agregar a los axiomas de $\mathscr{T}$ la siguiente lista de axiomas: $\{\neg(c_\iota = c_\upsilon)\}_{\substack{\iota\neq\upsilon \\ \iota,\upsilon<\alpha}}$.

Cada subconjunto finito de $\mathscr{T}'$ tiene modelo, igual que en el corolario anterior, y por tanto, $\mathscr{T}'$ tiene un modelo de cardinalidad α y que es también modelo de $\mathscr{T}$. ■

Nota. En la demostración del teorema de Löwenheim-Skolem, los nuevos axiomas garantizan que la cardinalidad del modelo es al menos α. Que es α es consecuencia de la demostración del Teorema de Completud, donde se construye el modelo tomando como universo a los términos del lenguaje. Por eso se puede garantizar la existencia de un modelo de cardinalidad exactamente α.

Corolario 8.21. *Hay modelos no estándares de la aritmética.*

Demostración.

Los axiomas de Peano se pueden formular en un lenguaje de primer orden con los símbolos **0**, **S**, $+$, $\cdot$, donde **0** es una constante individual, **S** es un símbolo funcional unario, y $+$ y $\cdot$ son símbolos funcionales binarios.

Por el Teorema de Löwenheim-Skolem esta teoría tiene modelos no numerables y que, consecuentemente, no son modelos *isomorfos* a $\mathbb{N}$, éstos son los modelos no estándares de la aritmética. ■

Ejercicios

1. Pruebe que las dos versiones del Teorema de Compacidad (Teoremas 8.17 y 8.18) son equivalentes.

***2.** Un mapa es un par ordenado $M = (P, A)$, donde $P \neq \varnothing$, $A \subseteq P \times P$ y (x, y) está en A si y sólo si x "es adyacente a" y. Si M es un mapa y $k \in \mathbb{N}$, decimos que M es k-coloreable si y sólo si i) para toda $x \in P$, x tiene un y sólo uno de los k colores, y ii) para todo $x, y \in P$, si $(x, y) \in A$, entonces x e y tienen distinto color. Pruebe que si todo mapa finito es k-coloreable, entonces todo mapa (infinito) es k-coloreable.

8.7 Deducción natural para predicados

En esta sección esbozamos un cálculo de deducción natural para la lógica de predicados análogo al presentado en la sección 5.4 para la lógica proposicional. Este sistema se debe a Benson Mates [Mt]. A este sistema lo denotamos CPN.

Al igual que para la lógica proposicional, se puede demostrar que este sistema tiene los mismos teoremas que el sistema estudiado en la sección 8.2-3, y por lo tanto los Teoremas de Validez y Completud también se aplican al CPN.

Sea $\mathscr{L}$ un lenguaje de primer orden.

Definición. Una *demostración* en el CPN es una sucesión finita de enunciados de $\mathscr{L}$, cada uno de los cuales tiene asignado un conjunto de números (llamados números de premisa) y tal que la sucesión ha sido construida de acuerdo con las siguientes reglas (siendo α y β fórmulas de $\mathscr{L}$, x una variable individual cualquiera de $\mathscr{L}$ y c cualquier constante individual de $\mathscr{L}$):

P (Introducción de premisas)

Cualquier enunciado puede ser introducido en una línea, tomando el número de esa línea como único número de premisa.

T (Inferencia tautológica)

Cualquier enunciado puede ser introducido en una línea si es consecuencia tautológica de enunciados que aparecen en líneas anteriores; como números de premisa se toman todos los números de premisa de esas líneas anteriores.

C (Condicionalización)

El enunciado $\alpha \Rightarrow \beta$ puede ser introducido en una línea si β aparece en una línea anterior; como números de premisa de la nueva línea se toman todos los de la línea anterior, con excepción (si se desea) del número de línea correspondiente a la línea en que aparece α.

EU (Especificación universal)

El enunciado $\alpha(x/c)$ puede ser introducido en una línea si $\forall x\alpha$ aparece en una línea anterior; como números de premisa de esa nueva línea se toman los de esa línea anterior.

GU (Generalización universal)

El enunciado $\forall x\alpha$ puede ser introducido en una línea si $\alpha(x/c)$ aparece en una línea anterior y c no ocurre ni en α ni en ninguna premisa de esa línea anterior; como números de premisa de esa nueva línea se toman los de esa línea anterior.

E (Cuantificación existencial)

El enunciado $\exists x\alpha$ puede ser introducido en una línea si $\neg\forall x\neg\alpha$ aparece en una línea anterior, o viceversa; como números de premisa de esa nueva línea se toman los de esa línea anterior.

GE (Generalización existencial)

El enunciado $\exists x\alpha$ puede ser introducido en una línea si $\alpha(x/c)$ aparece en una línea anterior; como números de premisa de la nueva línea se toman los de esa línea anterior.

Definición. Si se tiene una demostración en el CPN cuya última fórmula es ϕ y Γ es el conjunto de fórmulas que aparecen en las líneas numeradas con los números de premisa de ϕ, se dice que ϕ *es derivable a partir de* Γ en el CPN. Notación: $\Gamma \vdash \phi$. Cuando $\Gamma = \varnothing$ decimos que ϕ es una *teorema* del CPN. Notación: $\vdash \phi$.

Observaciones sobre las reglas:

1. La regla *P* permite introducir el antecedente de una implicación que se quiere demostrar. De llegar al consecuente entonces, por medio de la regla C, se obtiene el condicional como teorema. Estas dos reglas rescatan el modo de demostrar en matemáticas.

2. La regla T asegura que el sistema CPN incluye a CEN (ver sección 5.4), pues todo teorema del CEN es una tautología (Teorema 5.12) y por lo tanto puede ser introducido en una demostración del CPN con el conjunto como conjunto de números de premisa.

3. La regla EU rescata el significado del cuantificador universal.

4. La regla GU refleja otro procedimiento común en matemáticas: si se quiere demostrar que una propiedad es satisfecha por todos los elementos de un cierto conjunto, se toma un elemento arbitrario y se prueba que tiene la propiedad dada.

5. La regla E enuncia la interdefinibilidad de los dos cuantificadores del lenguaje.

6. La regla GE define al cuantificador existencial. Esta regla no es independiente de las otras, se puede deducir a partir de EU, GU y E.

Ejemplos:

1. $\{\forall x(Fx \Rightarrow Gx), \forall x(Gx \Rightarrow Hx)\} \vdash \forall x(Fx \Rightarrow Hx)$

$\{1\}$	(1)	$\forall x(Fx \Rightarrow Gx)$	P
$\{2\}$	(2)	$\forall x(Gx \Rightarrow Hx)$	P
$\{3\}$	(3)	Fa	P
$\{1\}$	(4)	$Fa \Rightarrow Ga$	1, EU
$\{1,3\}$	(5)	Ga	3,4 T
$\{2\}$	(6)	$Ga \Rightarrow Ha$	2, EU
$\{1,2,3\}$	(7)	Ha	5,6 T

$\{1,2\}$	(8) $Fa \Rightarrow Ha$	3,7 C
$\{1,2\}$	(9) $\forall x(Fx \Rightarrow Hx)$	8, GU

Nota: En el paso 9 se aplicó GU para $\alpha = Fx \Rightarrow Hx$ y c =a.

2. $\vdash \forall x Fx \Rightarrow \exists x Fx$

$\{1\}$	(1) $\forall x Fx$	P
$\{1\}$	(2) Fa	1, EU
$\{1\}$	(3) $\exists x Fx$	3, GE
$\varnothing$	(4) $\forall x Fx \Rightarrow \exists x Fx$	1,3 C

3. $\forall x \forall y(Pxy \Rightarrow \neg Pyx) \vdash \forall x \neg Pxx$

$\{1\}$	(1) $\forall x \forall y(Pxy \Rightarrow \neg Pyx)$	P
$\{1\}$	(2) $\forall y(Pay \Rightarrow \neg Pya)$	1, EU
$\{1\}$	(3) $Paa \Rightarrow \neg Paa$	2, EU
$\{1\}$	(4) $\neg Paa$	3, T
$\{1\}$	(5) $\forall x \neg Pxx$	4, GU

Ejercicios

Demostrar los siguientes teoremas del CPN:

a. $\forall x(Px \wedge Qx) \Leftrightarrow (\forall x Px \wedge \forall x Qx)$

b. $\exists x(Px \vee Qx) \Leftrightarrow (\exists x Px \vee \exists x Qx)$

(*Sugerencia*: Demostrar cada implicación por separado y usar regla T para obtener el bicondicional)

c. $(\forall x Px \vee \forall x Qx) \Rightarrow \forall x(Px \vee Qx)$

d. $\exists x(Px \wedge Qx) \Rightarrow (\exists x Px \wedge \exists x Qx)$

e. $\exists x \forall y Pxy \Rightarrow \forall y \exists x Pxy$

Capítulo 9
Lenguajes y autómatas

Dicho con la jerga de la mecánica: la lógica es la resultante de dos componentes: la gramática y la verdad.

Willard V. Quine

9.1 Introducción

En este capítulo, atenderemos un tema donde han concurrido la lógica, la ligüística y la computación: la relación entre los *autómatas* y los conjuntos de símbolos (lenguajes) aceptados por los mismos. Desde el Capítulo 3 esbozamos algunos de los problemas relacionados con las palabras que conforman un lenguaje (*e.g.*, el denominado "problema de las palabras"); problemas que surgieron de concebir definiciones formales para el concepto de algoritmo y para los dispositivos teóricos encargados de sus ejecuciones: los *autómatas*. El autómata más general ideado es la máquina de Turing (siendo otros, la máquina de Post, la de Schönhage, RAM, etc., todas equivalentes). El estudio de estos autómatas lo aplazaremos hasta el capítulo siguiente. Debido a que las máquinas de Turing (MT) contemplan el uso de una cinta (o banda) de longitud infinita (lo cual torna estos dispositivos físicamente irrealizables) se han considerado versiones más restringidas para ser modelos de computadoras. El requerimiento de una banda infinita para una máquina de Turing proviene del hecho de que es imposible dar a priori una cota superior para la longitud de cinta que una MT empleará al realizar un cálculo (incluso algunos muy sencillos, *cf.* secc. 10.5). Este problema es indecidible. Por consiguiente,

una restricción natural a imponer es que la longitud de cinta sea finita, o bien, en forma equivalente, considerar máquinas que consten de sólo un número finito de *estados internos* a ser usados tanto para memoria como para cómputo.

De los modelos idealizados para la neurona por neurofisiologistas, uno de los primeros se debe a Mc Culloch y Pitts. Partiendo de este modelo, conocido como *red neural (nerve-net)*, Kleene investigó las capacidades y limitaciones de los autómatas construidos a partir de estos componentes idealizados. El estudio de Kleene llevó a la caracterización de los *lenguajes regulares*, que son los conjuntos de símbolos aceptados por estos *autómatas finitos*. Además, estableció condiciones en términos finitistas para definición de estos lenguajes empleando ciertas operaciones sobre los conjuntos finitos de símbolos, y *expresiones regulares* para representarlos. Todos estos resultados suelen resumirse en la literatura como el Teorema de Kleene para lenguajes regulares. Con la finalidad de obtener una máquina más poderosa, Rabin y Scott introdujeron la noción de *autómata finito indeterminista*, el cual, paradójicamente, resultó equivalente a su contraparte determinista, pero de representación más sencilla.

Otra manera de considerar a un lenguaje es como un conjunto generado bajo la aplicación de ciertas *reglas de producción* (sustituciones dirigidas admisibles) a un conjunto dado de palabras. Tal fue en parte el enfoque adoptado en el Capítulo 3. Procediendo sobre esta línea, los *lenguajes regulares* también pueden construirse empleando *gramáticas lineales*. *Grosso modo*, una gramática es una estructura en la cual se establece una división dentro del vocabulario entre símbolos "intermedios" en el proceso de producción denominados *variables sintácticas* y los *símbolos terminales*, siendo éstos los constituyentes de las ebf's del lenguaje generado por la gramática en cuestión. Los lenguajes que estudiaremos aquí fueron introducidos por Chomsky en su intento de hallar modelos para los lenguajes naturales.

Por ejemplo, consideremos un reducto del lenguaje natural, donde es posible construir una oración concatenando un sujeto y un predicado. Si a su vez, sabemos cómo construir sujetos y predicados, sabremos cómo generar algunas oraciones. Simbólicamente, una oración puede construirse a partir de la regla de producción o *regla de reescritura*:

$$(\text{oración}) \mapsto (\text{sujeto})(\text{predicado})$$

Los paréntesis son empleados aquí para indicar que su contenido debe ser considerado como una unidad. Es por esto que en lingüística a estas unidades

se les denomina *variables sintácticas.* Otras reglas de reescritura pueden ser:

(sujeto) $\mapsto$ (artículo)(nombre);
(sujeto) $\mapsto$ (artículo)(nombre)(adjetivo);
(nombre) $\mapsto$ Juan;
(adjetivo) $\mapsto$ pequeño; etcétera.

Nótese que en las dos últimas reglas se omitieron los paréntesis, porque es la palabra en sí la referida en lugar de su uso como parte de la variable sintáctica. Estas palabras son los *símbolos terminales.* A manera de ilustración, veamos la construcción de la oración "el pequeño Juan es un ladrón", mediante las reglas de reescritura:

(oración) $\mapsto$ (sujeto)(predicado);
(sujeto) $\mapsto$ (artículo)(adjetivo)(nombre);
(sujeto) $\mapsto$ (nombre);
(predicado) $\mapsto$ (verbo)(complemento directo);
(complemento directo) $\mapsto$ (artículo)(nombre);
(artículo) $\mapsto$ el; (artículo) $\mapsto$ un;
(nombre) $\mapsto$ Juan; (nombre) $\mapsto$ ladrón;
(adjetivo) $\mapsto$ pequeño; (verbo) $\mapsto$ es.

La oración es generada comenzando con el símbolo "(oración)" y aplicando una regla de reescritura a la vez a alguna variable sintáctica, hasta que se obtenga una cadena de palabras del lenguaje. Para la oración anterior, tal proceso está dado por:

(oración) $\mapsto$ (sujeto)(predicado)
$\Rightarrow$ (sujeto)(verbo)(complemento directo)
$\Rightarrow$ (sujeto) es (complemento directo)
$\Rightarrow$ (sujeto) es (artículo)(nombre)
$\Rightarrow$ (sujeto) es un (nombre)
$\Rightarrow$ (sujeto) es un ladrón
$\Rightarrow$ (artículo)(adjetivo)(nombre) es un ladrón

$\Rightarrow$ el (adjetivo)(nombre) es un ladrón
$\Rightarrow$ el pequeño (nombre) es un ladrón
$\Rightarrow$ el pequeño Juan es un ladrón.

El mismo conjunto de reglas de reescritura generará también las oraciones:

"un ladrón es el Juan",
"el pequeño ladrón es un Juan",
"Juan es un Juan", etcétera. □

Algunas observaciones pertinentes:

1. En las reglas de reescritura anteriores no se contempla el uso de espacios entre las palabras, de tal forma que las oraciones así construidas quedan como, *e.g.*, "unladróneselJuan", completamente yuxtapuestas.

2. Se emplean palabras como "Juan" considerándolas como símbolos cuando notamos que están conformadas a su vez de otros símbolos (letras), sin dar reglas ulteriores para su producción.

3. Todas las oraciones generadas pueden ser tomadas como bien formadas aun cuando sean falsas o carentes de sentido: las gramáticas sólo atienden la forma (sintaxis) más no al significado (semántica).

Las gramáticas (y por ende, sus lenguajes) se ven divididas (respectivamente conformando una jerarquía bajo contención de clases, $\subseteq$) de acuerdo con el tipo de reglas de producción con que estén provistas. Los lenguajes denominados *libres del contexto* han resultado particularmente importantes para analizar, ya no el lenguaje natural, sino ciertos lenguajes artificiales: *lenguajes de programación*. Retornando al tema de los autómatas, cabe entonces preguntarse sobre los tipos de autómatas que aceptan las palabras de las respectivas familias de lenguajes. Partiendo de los más simples, *autómatas finitos* $\longleftrightarrow$ *lenguajes regulares*, y ascendiendo en la jeraquía, tenemos las parejas: *autómatas de pila* $\longleftrightarrow$ *lenguajes libres del contexto*, *autómatas lineales acotados* $\longleftrightarrow$ *lenguajes sensibles al contexto,* y finalmente, MT $\longleftrightarrow$ *lenguajes (conjuntos) recursivamente enumerables*.

9.2 La jerarquía de Chomsky

Formalizando la idea de gramática dada anteriormente, tenemos que el conjunto de símbolos de su vocabulario $\mathcal{V}$ se particiona en $\mathcal{V} = \mathcal{V}_T \cup \mathcal{V}_N$, donde $\mathcal{V}_T$ es el *conjunto de símbolos terminales* y $\mathcal{V}_N$ el de *símbolos no terminales* o *variables sintácticas.* Por convención, usaremos letras minúsculas a, b, c, etc. para denotar a los símbolos terminales y mayúsculas A, B, C, etc. para los no terminales. Para las palabras (cadenas de símbolos) formadas con símbolos terminales y/o no terminales, emplearemos letras griegas minúsculas α, β, γ, etc. La longitud de una palabra α (número de símbolos) será denotada $|\alpha|$. (En un sentido estricto, los símbolos no terminales son elementos del metalenguaje, mientras que los terminales lo son del lenguaje.)

> **Definición.** Una *gramática* (*de estructura de frases*) $\mathcal{G}$ es una cuaterna dada por $\mathcal{G} = \langle \mathcal{V}_T, \mathcal{V}_N, E, \mathcal{R} \rangle$, donde $\mathcal{V}_T$ y $\mathcal{V}_N$ son los conjuntos de símbolos terminales y no terminales, respectivamente; E es un elemento distinguido de $\mathcal{V}_N$, denominado el *símbolo inicial*; y $\mathcal{R}$ es una relación ($\text{dom}\,\mathcal{R} \subset \mathcal{V}^* \times \mathcal{V}_N \times \mathcal{V}^*$ y $\text{ran}\,\mathcal{R} \subset \mathcal{V}^*$) que constituye un conjunto finito, el *conjunto de las reglas de producción.* Por convención, la pareja $(\alpha, \beta) \in \mathcal{R}$ se escribe $\alpha \mapsto \beta$, y denota una regla de producción.

Así, el ejemplo expuesto (de una parte del castellano) podemos reescribirlo con la presente terminología como:

$$\mathcal{G} = \langle \{j, p, e, u, l, f\}, \{O, S, P, A, N, H, C, V\}, O, \mathcal{R} \rangle,$$

donde

$$\mathcal{R} = \left\{ \begin{array}{l} O \mapsto SP, S \mapsto AHN, S \mapsto N, P \mapsto VC, C \mapsto AN, A \mapsto e, \\ A \mapsto u, N \mapsto j, N \mapsto l, H \mapsto pp, V \mapsto f \end{array} \right\} \quad \square$$

> **Definición.** Sean $\mathcal{G} = \langle \mathcal{V}_T, \mathcal{V}_N, E, \mathcal{R} \rangle$ una gramática y $\phi, \psi \in \mathcal{V}^* \setminus \{\Lambda\}$. Decimos que ψ es una *derivación directa* de ϕ, y lo denotamos $\phi \Rightarrow \psi$, si y sólo si existen cadenas $\alpha, \omega \in \mathcal{V}^*$ (posiblemente vacías) tales que $\phi = \alpha\beta\omega$, $\psi = \alpha\gamma\omega$, y $\beta \mapsto \gamma$ es una produción de $\mathcal{R}$.

Este concepto puede extenderse para producir una palabra ψ en un número finito de pasos a partir de ϕ.

Definición. Sea $\mathcal{G} = \langle \mathcal{V}_T, \mathcal{V}_N, E, \mathcal{R} \rangle$ una gramática. La palabra ϕ *produce* la palabra ψ, denotado $\phi \Rightarrow \psi$, si y sólo si existen palabras $\phi_1, \phi_2, \ldots, \phi_n$ (para $n > 0$) tales que $\phi = \phi_1 \Rightarrow \phi_2, \phi_2 \Rightarrow \phi_3, \ldots, \phi_{n-1} \Rightarrow \phi_n = \psi$. La relación $\Rightarrow^+$ es la *cerradura* (extensión) *transitiva* de la relación $\Rightarrow$. Si permitimos $n = 0$, podemos definir la *cerradura reflexiva* y *transitiva* de $\Rightarrow$, denotada $\Rightarrow^*$, como:

$$\phi \Rightarrow^* \psi \quad \text{si y sólo si} \quad \phi \Rightarrow^+ \psi \quad \text{o} \quad \phi = \psi.$$

Definición. Una *forma sentencial* es cualquier derivación a partir del símbolo inicial E. El *lenguaje* generado por una gramática $\mathcal{G}$, denotado $\mathcal{L}(\mathcal{G})$, es el conjunto de todas las formas sentenciales cuyos símbolos son terminales, *i.e.* el conjunto de las ebf's,

$$\mathcal{L}(\mathcal{G}) = \{\alpha \in \mathcal{V}_T^* : E \Rightarrow^* \alpha\}$$

Así, $\mathcal{L}(\mathcal{G}) \subset \mathcal{V}_T^*$. Un conjunto $\mathcal{L} \subset \mathcal{V}_T^*$ es un *lenguaje de estructura de frases* si y sólo si existe una gramática (de estructura de frases) $\mathcal{G}$ tal que $\mathcal{L}(\mathcal{G}) = \mathcal{L}$. La familia de todos los lenguajes de estructura de frases resulta ser idéntica a la de los *conjuntos recursivamente enumerables, i.e.* los conjuntos aceptados por las MT (*cf.* [Da]). Ahora bien, siendo que estos conjuntos han sido ampliamente estudiados, no parece haber ventaja alguna en concebirlos como lenguajes. De aquí que sea más prometedor considerar restricciones sobre las gramáticas para dar origen a otras clases de lenguajes. N. Chomsky clasificó las gramáticas en cuatro tipos atendiendo a las restricciones impuestas sobre sus reglas de producción. Las *gramáticas sin restricciones* son denominadas de *tipo* 0 y, como ya mencionamos, dan lugar a los conjuntos recursivamente enumerables. Estas gramáticas pueden tener reglas de producción de la forma $\phi A \psi \mapsto \phi \Lambda \psi$, que aplicándolas en una producción $\alpha \Rightarrow \beta$ dan como consecuencia una forma sentencial β de longitud menor que α, $|\beta| < |\alpha|$. Tales reglas se denominan *reglas de contracción.*

Ahora bien, motivado por que las gramáticas sirvan para generar modelos de los lenguajes naturales, Chomsky propuso como primer requisito que se excluyan de las mismas todas las reglas de contracción. (Este aspecto se confirma con el ejemplo que expusimos.)

Definición. Una gramática es de *tipo 1, sensible al contexto* o *dependiente del contexto,* si y sólo si todas sus producciones sólo son de la forma $\alpha \mapsto \beta$, donde $|\alpha| \leq |\beta|$ (*i.e.*, las producciones no reducen en longitud a las cadenas), pudiendo tener como única regla de contracción a $E \mapsto \Lambda$.

Nótese que si $E \mapsto \Lambda$ es una regla de producción, E no puede ocurrir en el miembro derecho de ninguna regla. Así, Λ es generada por una gramática dependiente del contexto si y sólo si se tiene la regla $E \mapsto \Lambda$.

El término "sensible al contexto" es debido a que las producciones que caracterizan estas gramáticas pueden reformularse equivalentemente en las siguientes: si $\alpha \mapsto \beta$ es una producción, entonces $\alpha = \phi_1 A \phi_2$, y $\beta = \phi_1 \gamma \phi_2$, donde ϕ_1 y ϕ_2 son posiblemente vacías y $\gamma \neq \Lambda$. Esta derivación puede pensarse como una regla de producción $A \mapsto \gamma$ dentro del "contexto" ϕ_1, ϕ_2. Estas gramáticas generan los *lenguajes sensibles al* (*dependientes del*) *contexto.*

Ejemplos:

1. El conjunto $\mathscr{L} = \{\alpha\alpha : \alpha \in \mathscr{V}_T^* \setminus \{\Lambda\}\} \subset \{a\}^*$ es el lenguaje formal generado por la gramática $\mathscr{G} = \langle \{a\}, \{E\}, E, \{E \mapsto aEa, E \mapsto aa\}\rangle$.
2. El conjunto $\mathscr{L} = \{a^n b^n c^n : n \geq 1\} \subset \{a, b, c\}^*$ es el lenguaje generado por la gramática $\mathscr{G} = \langle \{a, b, c\}, \{E, B, C\}, E, \mathscr{R}\rangle$, donde $\mathscr{R}$ consiste de las reglas:

$$\mathscr{R} = \left\{ \begin{array}{l} E \mapsto aEBC, E \mapsto aBC, CB \mapsto BC, aB \mapsto ab \\ bB \mapsto bb, bC \mapsto bc, cC \mapsto cc \end{array} \right\} \qquad \square$$

Imponiendo una restricción adicional se obtienen las gramáticas libres del contexto.

Definición. Una gramática es de *tipo 2, libre del contexto* o *independiente del contexto* si y sólo si todas sus producciones sólo son de la forma $\alpha \mapsto \beta$, donde $|\alpha| \leq |\beta|$, $\alpha \in \mathscr{V}_N$, pudiendo tener como única contracción a $E \mapsto \Lambda$.

Para tales gramáticas, la variable a reescribir (sustituir) en una forma sentencial se reescribe sin atender a los demás símbolos en su vecindad o "contexto". La clase correspondiente de lenguajes que generan son los *lenguajes libres* o *independientes*

del contexto. Aunque no poseen el poder de representar partes significativas del lenguaje natural, estas gramáticas deben su estudio a que han sido fructíferamente empleadas en lenguajes más simples en estructura, como algunos lenguajes de programación. Estas gramáticas no especifican si una determinada variable fue declarada cuando se empleó en alguna expresión de una proposición subsecuente de un programa fuente.

Ejemplos:

1. Cualquier subconjunto finito $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ de $\mathcal{V}_T^*$ es un lenguaje generado por la gramática $\mathcal{G} = \langle \mathcal{V}_T, \{E\}, E, \mathcal{R}\rangle$, donde $\mathcal{R} = \{E \mapsto \alpha_i\}$ para $i = 1, \ldots, n$.
2. $\mathcal{V}_T^*$ es un lenguaje generado por la gramática $\mathcal{G} = \langle \mathcal{V}_T, \{E\}, E, \mathcal{R}\rangle$, donde $\mathcal{R}$ está dado por $\mathcal{R} = \{E \mapsto Ea : a \in \mathcal{V}_T\} \cup \{E \mapsto \Lambda\}$.
3. El conjunto $\mathcal{L} = \{a^n b^n : n \geq 0\}$ es un lenguaje formal generado por la gramática $\mathcal{G} = \langle \{a, b\}, \{E\}, E, \mathcal{R}\rangle$, donde $\mathcal{R}$ está dado por $\mathcal{R} = \{E \mapsto aEb, E \mapsto ab\}$. □

Finalmente tenemos las gramáticas lineales.

Definición. Una gramática es de *tipo* 3 o *lineal derecha* (*izquierda*) si y sólo si todas sus producciones sólo son de la forma $\alpha \mapsto \beta$, con $|\alpha| \leq |\beta|$, donde $\beta \in \{b\gamma(\text{resp. } \gamma b), b\}$, $\alpha, \gamma \in \mathcal{V}_N$, y posiblemente se tenga $E \mapsto \Lambda$.

A los lenguajes generados por estas gramáticas se les denomina *lenguajes regulares*.

No es difícil notar que las cuatro clases de gramáticas están ordenadas (en términos de sus extensionalidades) de forma tal que las familias de lenguajes por ellas generados forman un *encaje* (bajo $\subseteq$):

Lengs. Regs. $\subset$ Lengs. Indeps. Contex. $\subset$ Lengs. Deps. Contex. $\subset$ Conjs. Rec. Enum.

siendo estas contenciones propias, pues se tienen ejemplos de lenguajes no generables por gramáticas de los otros tipos, a saber,

a. $\mathscr{L} = \{a^n b^n : n \geq 0\}$ es libre del contexto, pero no es regular; y

b. $\mathscr{L} = \{a^n b^n c^n : n \geq 0\}$ es sensible al contexto, pero no es libre.

Una de las razones para que esta jerarquía de familias de lenguajes haya permanecido desde que se instauró (robustez), obedece a sus propiedades de *invariancia* o *cerradura* bajo varias operaciones de conjuntos (unión, intersección, etc.), *homomorfismo* (invariancia de la estructura algebraica), etcétera.

Ejercicios

1. Considere las gramáticas $\mathscr{G}_i = \langle \{a, b\}, \mathscr{V}_N^i, E, \mathscr{R}_i \rangle$, donde

$$\mathscr{V}_N^1 = \{E, A\}, \quad \mathscr{R}_1 = \{E \mapsto \Lambda, E \mapsto A, A \mapsto aA, A \mapsto b\}$$
$$\mathscr{V}_N^2 = \{E, A, B\}, \quad \mathscr{R}_2 = \{E \mapsto \Lambda, E \mapsto bA, A \mapsto Ba, B \mapsto b\}$$
$$\mathscr{V}_N^3 = \{E, A, B\}, \quad \mathscr{R}_3 = \{E \mapsto A, A \mapsto aA, A \mapsto Bb, B \mapsto a\}$$
$$\mathscr{V}_N^4 = \{E, A, B\}, \quad \mathscr{R}_4 = \{E \mapsto \Lambda, A \mapsto aA, A \mapsto bA, A \mapsto A, B \mapsto b\}$$

a. Describa $\mathscr{L}(\mathscr{G}_i)$, para $i = 1, 2, 3$ y 4.

b. Para cada lenguaje, produzca una expresión de longitud 5.

2. Construya una gramática para generar cada uno de los lenguajes siguientes:

a. $\mathscr{L}_1 = \{\alpha\alpha : \alpha \in \mathscr{V}_T^* \setminus \{\Lambda\}\} \subset \{a, b\}^*$

b. $\mathscr{L}_1 = \{m\alpha\alpha : \alpha \in \mathscr{V}_T^* \setminus \{\Lambda\}\} \subset \{m, i, u\}^*$

c. $\mathscr{L}_1 = \{\alpha a \alpha : \alpha \in \{a, b\}^*\}$

3. Pruebe que el lenguaje generado por $\mathscr{G} = \langle \{E, A, B\}, \{a, b\}, E, \mathscr{R} \rangle$, con $\mathscr{R}$ dado por:

$$\mathscr{R} = \left\{ \begin{array}{l} E \mapsto aAb, bB \mapsto a, Ab \mapsto EBb \\ Aa \mapsto EaB, B \mapsto EA, B \mapsto ab \end{array} \right\}$$

es vacío.

9.3 Lenguajes regulares

Si un lenguaje es finito, su especificación puede reducirse a una enumeración exhaustiva de todas sus ebf's. De aquí que la representación finitista se torna importante cuando el lenguaje es infinito. Una manera para representar un lenguaje es mediante una cadena finita (construida en algún lenguaje) que lo especifique de manera unívoca. Nótese que un lenguaje formal $\mathscr{L} = (\mathscr{A}, \mathscr{E})$ puede considerarse simplemente como un subconjunto de $\mathscr{A}^*$. Así, para un alfabeto $\mathscr{A}$ dado, tenemos que aunque el conjunto de todas las palabras $\mathscr{A}^*$ sea numerable, el conjunto $\mathscr{P}(\mathscr{A}^*)$ de todos los subconjuntos de $\mathscr{A}^*$ es no numerable (para todo conjunto C, $|C| < |\mathscr{P}(C)|$). Por consiguiente, hay incontables lenguajes[1] (tantos como números irracionales en $\mathbb{R}$) que no podrán ser especificados mediante un número contable de representantes. Así, lo mejor a hacer es hallar familias de lenguajes que admitan representaciones finitas. Tal es el caso, de la familia de los lenguajes regulares, la cual fue concebida como la menor clase construida a partir de conjuntos finitos de símbolos por aplicación de un número finito de operaciones (Kleene).

Definición. Sean $\mathscr{L}_1, \mathscr{L}_2 \subseteq \mathscr{A}^*$. El *producto* o *concatenación* $\mathscr{L}_1 \circ \mathscr{L}_2$ se define como $\mathscr{L}_1 \circ \mathscr{L}_2 = \{\alpha : \alpha = \beta\gamma, \text{ donde } \beta \in \mathscr{L}_1 \text{ y } \gamma \in \mathscr{L}_2\}$.

En otros términos, cada palabra de $\mathscr{L}_1 \circ \mathscr{L}_2$ se forma concatenando una palabra de $\mathscr{L}_1$ seguida de una de $\mathscr{L}_2$. Para simplificar la notación, omitiremos el símbolo "$\circ$". Es fácil notar que este producto no es conmutativo $\mathscr{L}_1\mathscr{L}_2 \neq \mathscr{L}_2\mathscr{L}_1$. Sin embargo, sí es asociativo: para cualesquiera $\mathscr{L}_1, \mathscr{L}_2, \mathscr{L}_3 \subseteq \mathscr{A}^*$, $(\mathscr{L}_1\mathscr{L}_2)\mathscr{L}_3 = \mathscr{L}_1(\mathscr{L}_2\mathscr{L}_3)$.

Definición. La *cerradura* o *estrella de Kleene*[2] de un conjunto $\mathscr{L}$, denotada $\mathscr{L}^*$, es el conjunto que consta de la palabra vacía Λ y de todas las palabras formadas por concatenaciones finitas de las palabras de $\mathscr{L}$. En otros términos, $\mathscr{L}^* := \mathscr{L}^0 \cup \mathscr{L}^1 \cup \mathscr{L}^2 \cup \mathscr{L}^3 \cup \ldots$, donde $\mathscr{L}^o = \{\Lambda\}$, y $\mathscr{L}^i = \mathscr{L}^{i-1}\mathscr{L}$, para $i > 0$.

[1] Por simplicidad, hemos supuesto que todo elemento de $\mathscr{P}(\mathscr{A}^*)$ es un lenguaje.
[2] En su artículo original, Kleene presenta esta operación como si fuese binaria.

Ejemplos:

1. Si $\mathscr{L} = \{a, bc\}$, entonces $\mathscr{L}^* = \{\Lambda, a, bc, aa, abc, bca, bcbc, \ldots\}$.
2. Nótese que el empleo de $\mathscr{A}^*$ para denotar al conjunto de todas las palabras sobre un alfabeto $\mathscr{A}$ es consistente con la notación de cerradura.
3. $\varnothing^* = \{\Lambda\}$. □

Definición. La familia de los *conjuntos regulares* (CR) sobre un alfabeto $\mathscr{A}$ es la menor clase de conjuntos que contiene al $\varnothing$, a los conjuntos unitarios $\{a\}$, para cada $a \in \mathscr{A}$, y es cerrada bajo las operaciones de unión, producto y cerradura. En otros términos, esta familia se define recursivamente mediante las cláusulas siguientes:

(a) $\varnothing \in$ CR; y si $a \in \mathscr{A}$, entonces $\{a\} \in$ CR;

(b) Si $\mathscr{L}_1, \mathscr{L}_2 \in$ CR, entonces $\mathscr{L}_1 \cup \mathscr{L}_2 \in$ CR, $\mathscr{L}_1\mathscr{L}_2 \in$ CR y $\mathscr{L}_1^* \in$ CR; y

(c) Sólo son CR's aquellos conjuntos construidos con base en (a) y (b).

En lo que resta de esta sección y en la siguiente presentaremos cuatro formas equivalentes para especificar a los conjuntos regulares. Un subconjunto de $\mathscr{A}^*$ será un conjunto regular si y sólo si:

(1) puede ser representado mediante una *expresión regular*;

(2) es un *lenguaje regular* (generado por una gramática lineal);

(3) es el conjunto de palabras (símbolos en una cinta) aceptado por un *autómata finito*;

(4) es el conjunto de palabras aceptado por un *autómata finito indeterminista.*

La interrelación entre: un conjunto regular—el conjunto representable con una expresión regular—el conjunto aceptable por un autómata finito, constituye el mencionado Teorema de Kleene para conjuntos regulares [Kl]. El resultado del inciso (2) fue presentado originalmente por Chomsky y Miller. Y la equivalencia con el inciso (4) se debe a Rabin y Scott [RS].

Consideremos un alfabeto finito $\mathscr{A}$, al cual extendemos con los símbolos $\emptyset$, $\mathbf{\Lambda}$, +, •,), (y denotémoslo con $\bar{\mathscr{A}}$; *i.e.*, $\bar{\mathscr{A}} = \mathscr{A} \cup \{\emptyset, \mathbf{\Lambda}, +, \bullet, *,), (\}$.

Definición. El conjunto de las *expresiones regulares* (ER) sobre $\mathscr{A}$ se define recursivamente de la manera siguiente:

(i) $\emptyset$ y $\mathbf{\Lambda}$ son ER;

(ii) Si $a \in \mathscr{A}$, entonces $\mathbf{a}$ es una ER;

(iii) Si $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$ son ER, también los son $(\boldsymbol{\alpha}+\boldsymbol{\beta})$, $(\boldsymbol{\alpha} \bullet \boldsymbol{\beta})$ y $(\boldsymbol{\alpha}^*)$;

(iv) Sólo son ER aquéllas construidas con base en (i)–(iii).

Ejemplo. Si $\mathscr{A} = \{a, b\}$, entonces $(((\boldsymbol{\alpha}+\boldsymbol{\beta})^* \cdot \boldsymbol{\alpha})+(\boldsymbol{\beta})^*)$ es una ER sobre $\mathscr{A}$. □

Cada expresión regular $\boldsymbol{\alpha}$ describe unívocamente (es la representante de un único) conjunto regular, $\mathscr{L}(\boldsymbol{\alpha})$, definido recursivamente:

(a) $\emptyset$ representa al conjunto vacío $\varnothing$;

(b) $\mathbf{\Lambda}$ representa al conjunto $\mathscr{L}(\mathbf{\Lambda}) = \{\Lambda\}$, que consiste de la palabra vacía;

(c) $\mathbf{a}$ representa al conjunto $\mathscr{L}(\mathbf{a}) = \{a\}$, para cada $a \in \mathscr{A}$;

(d) Si $\boldsymbol{\alpha}$ representa a $\mathscr{L}_1$ y $\boldsymbol{\beta}$ representa a $\mathscr{L}_2$, se tiene que: $(\boldsymbol{\alpha} + \boldsymbol{\beta})$, $(\boldsymbol{\alpha} \cdot \boldsymbol{\beta})$ y $(\boldsymbol{\alpha})^*$ representan a $\mathscr{L}_1 \cup \mathscr{L}_2$, $\mathscr{L}_1\mathscr{L}_2$, y $\mathscr{L}_1^*$, respectivamente.

Algunas convenciones para la eliminación de paréntesis pueden facilitar la escritura de las expresiones regulares. Las que adoptaremos aquí, se siguen de manera análoga a la jerarquía usual (en términos del *alcance*) de las operaciones numéricas: + (suma), • (producto) y * ("exponenciación"). También eliminaremos los (símbolos que representan a los) paréntesis externos. Como simplificación adicional, omitiremos el símbolo • . De esta forma, **a + ba** se escribe en lugar de **(a + (b • a))**; **(a* + b)* a + a** en lugar de **((((a) + b)* • a) + a)**, etcétera.

Ejemplos:

Consideremos el alfabeto $\mathscr{A} = \{a, b, c\}$. Las ER's escritas a la izquierda denotan a los conjuntos correspondientes de la derecha:

$\boldsymbol{\alpha}$	$\mathscr{L}(\boldsymbol{\alpha})$
c*(ab)	Todas las palabras sobre $\mathscr{A}$ que constan de c's seguidas de ab
(a* + b*) + c*	Todas las palabras con sólo a's, sólo b's o sólo c's
((a + b) + c)*	$\mathscr{A}^*$

(a + b)*(c(a + b)*)	Todas las palabras en las que ocurre exactamente una *c*
b*(a + cb*)*	Todas las palabras en las que no ocurre la palabra *ab*

□

El proceso recursivo presentado permite obtener *el* conjunto regular $\mathscr{L}(\boldsymbol{\alpha})$ a partir de *una* expresión regular $\boldsymbol{\alpha}$ dada. Por tanto, este proceso implica que $\mathscr{L}(\boldsymbol{\alpha})$ denota a una función de un conjunto de ER's sobre $\mathscr{A}$ en $\mathscr{P}(\mathscr{A})^*$ (correspondencia hecha explícita con el ejemplo anterior). También contamos con el recíproco: dado un conjunto regular $\mathscr{L}$ existe una expresión regular $\boldsymbol{\alpha}$ tal que $\mathscr{L} = \mathscr{L}(\boldsymbol{\alpha})$. Así, este recíproco equivale a que $\mathscr{L}(\boldsymbol{\alpha})$ es una función sobreyectiva con rango CR's.

Proposición 9.1. *Para todo subconjunto finito $\mathscr{L}$ de $\mathscr{A}^*$, existe una expresión regular $\boldsymbol{\alpha}$ sobre $\mathscr{A}$ tal que $\mathscr{L} = \mathscr{L}(\boldsymbol{\alpha})$.*

Demostración.

Procederemos por inducción matemática sobre la cardinalidad de los conjuntos finitos.

Base: si $\mathscr{L} = \varnothing$, entonces $\mathscr{L} = \mathscr{L}(\emptyset)$, y si $\mathscr{L} = \{\Lambda\}$, entonces $\mathscr{L} = \mathscr{L}(\boldsymbol{\Lambda})$. Consideremos ahora que $\mathscr{L}$ es un unitario, $\mathscr{L} = \{\alpha\}$, donde $\alpha = a_1 a_2 \ldots a_k \in \mathscr{A}^*$, entonces $\mathscr{L} = \mathscr{L}(\boldsymbol{\alpha})$, con $\boldsymbol{\alpha} = \mathbf{a_1(a_2(\ldots a_k)\ldots)}$. De esta manera, tenemos representaciones para conjuntos con un y sin elementos.

Hipótesis inductiva: supongamos ahora que el resultado es válido para todos los conjuntos en $\mathscr{P}(\mathscr{A}^*)$ con n elementos.

Sea $\mathscr{L}$ un conjunto con $n + 1$ elementos. Luego $\mathscr{L}$ puede escribirse como $\mathscr{L} = \mathscr{L}_1 \cup \{\beta\}$, con $\beta \in \mathscr{A}^*$ y $\mathscr{L}_1$ conteniendo n elementos. Por la hipótesis inductiva, existe una ER $\boldsymbol{\alpha}$ tal que $\mathscr{L}_1 = \mathscr{L}(\boldsymbol{\alpha})$. Considerando el caso unitario recién tratado, existe una ER $\boldsymbol{\beta}$ tal que $\mathscr{L}(\boldsymbol{\beta}) = \{\beta\}$. Así,

$$\mathscr{L}(\boldsymbol{\alpha}+\boldsymbol{\beta}) = \mathscr{L}(\boldsymbol{\alpha}) \cup \mathscr{L}(\boldsymbol{\beta}) = \mathscr{L}_1 \cup \{\beta\} = \mathscr{L}$$ ■

Teorema 9.2. *Para todo conjunto regular $\mathscr{L} \subseteq \mathscr{A}^*$, existe una expresión regular $\boldsymbol{\alpha}$ sobre $\mathscr{A}$ tal que $\mathscr{L}(\boldsymbol{\alpha}) = \mathscr{L}$.*

Demostración.

Por la proposición anterior el resultado es cierto para todos los conjuntos finitos. Sea $\mathscr{L}$ un conjunto regular arbitrario. Por su definición, $\mathscr{L}$ se obtuvo a partir de ciertos conjuntos regulares finitos aplicándoles un número finito de veces las operaciones de $\cup$, $\circ$ y $*$. Empezando con las ER's que representan a estos conjuntos, podemos construir una ER para $\mathscr{L}$, simplemente indicando cada uso de las operaciones $\cup$, $\circ$ y $*$, escribiendo $+$, $\bullet$ y *, respectivamente y los símbolos de puntuación) y (con) y (. ■

La función $\mathscr{L}(\boldsymbol{\alpha})$, aunque sobre, no es inyectiva: $\boldsymbol{\alpha}$ y $\boldsymbol{\alpha} + \emptyset$ representan al mismo conjunto; así como también $(\boldsymbol{\alpha} + \boldsymbol{\beta}) + \boldsymbol{\gamma}$ y $\boldsymbol{\alpha} + (\boldsymbol{\beta} + \boldsymbol{\gamma})$ (asociatividad de $+$).

Definición. Dos expresiones regulares $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$ sobre $\mathscr{A}$ son *equivalentes,* y lo denotamos $\boldsymbol{\alpha} \simeq \boldsymbol{\beta}$, si y sólo si $\mathscr{L}(\boldsymbol{\alpha}) = \mathscr{L}(\boldsymbol{\beta})$.

La equivalencia entre ER's puede ser establecida con el auxilio de identidades (tal como la asociatividad). Algunas identidades importantes (varias de las cuales fueron obtenidas en [Kl]) son las siguientes:

Identidades básicas para expresiones regulares

Sean $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ expresiones regulares sobre un alfabeto $\mathscr{A}$. Entonces

1. $\boldsymbol{\alpha} + \emptyset \simeq \boldsymbol{\alpha}$, $\boldsymbol{\alpha} + \boldsymbol{\alpha} \simeq \boldsymbol{\alpha}$.
2. $\boldsymbol{\alpha} + \boldsymbol{\beta} \simeq \boldsymbol{\beta} + \boldsymbol{\alpha}$.
3. $(\boldsymbol{\alpha} + \boldsymbol{\beta}) + \boldsymbol{\gamma} \simeq \boldsymbol{\alpha} + (\boldsymbol{\beta} + \boldsymbol{\gamma})$.
4. $(\boldsymbol{\alpha}\boldsymbol{\beta})\boldsymbol{\gamma} \simeq \boldsymbol{\beta}(\boldsymbol{\alpha}\boldsymbol{\gamma})$ (Por lo que el producto podemos escribirlo como $\boldsymbol{\alpha}\boldsymbol{\beta}\boldsymbol{\gamma}$).
5. $\boldsymbol{\alpha}\boldsymbol{\Lambda} \simeq \boldsymbol{\Lambda}\boldsymbol{\alpha} \simeq \boldsymbol{\alpha}$, $\boldsymbol{\alpha}\emptyset \simeq \emptyset\boldsymbol{\alpha} \simeq \emptyset$.
6. $(\boldsymbol{\alpha} + \boldsymbol{\beta})\boldsymbol{\gamma} \simeq \boldsymbol{\alpha}\boldsymbol{\gamma} + \boldsymbol{\beta}\boldsymbol{\gamma}$.
7. $\boldsymbol{\alpha}(\boldsymbol{\beta} + \boldsymbol{\gamma}) \simeq \boldsymbol{\alpha}\boldsymbol{\beta} + \boldsymbol{\alpha}\boldsymbol{\gamma}$.
8. $\boldsymbol{\alpha}^* \simeq \boldsymbol{\alpha}^*\boldsymbol{\alpha}^* \simeq \boldsymbol{\alpha}^*)^* \simeq (\boldsymbol{\Lambda} + \boldsymbol{\alpha})^*$, $\emptyset^* \simeq \boldsymbol{\Lambda}^* \simeq \boldsymbol{\Lambda}$.
9. $\boldsymbol{\alpha}^* \simeq \boldsymbol{\alpha}^*\boldsymbol{\alpha}$.
10. $\boldsymbol{\alpha}^* \simeq \boldsymbol{\Lambda} + \boldsymbol{\alpha} + \boldsymbol{\alpha}\boldsymbol{\alpha} + \boldsymbol{\alpha}^3 + \cdots + \boldsymbol{\alpha}^n\boldsymbol{\alpha}^*$, para $n \geq 1$.
11. $\boldsymbol{\alpha}^* \simeq \boldsymbol{\Lambda} + \boldsymbol{\alpha}\boldsymbol{\alpha}^*$.
12. $(\boldsymbol{\alpha} + \boldsymbol{\beta})^* \simeq (\boldsymbol{\alpha}^* + \boldsymbol{\beta}^*)^* \simeq (\boldsymbol{\alpha}^*\boldsymbol{\beta}^*)^* \simeq (\boldsymbol{\alpha}^*\boldsymbol{\beta})^* \simeq (\boldsymbol{\alpha}^*\boldsymbol{\beta})^*\boldsymbol{\alpha}^* \simeq \boldsymbol{\alpha}^*(\boldsymbol{\beta}\boldsymbol{\alpha}^*)^*$.

13. $\boldsymbol{\alpha(\beta\alpha)^* \simeq (\alpha\beta)^* \alpha}$.
14. $\boldsymbol{(\alpha^*\beta)^* \simeq \Lambda + (\alpha + \beta)^*, (\alpha\beta^*)^* \simeq \Lambda + \alpha(\alpha + \beta)^*}$.

Veamos ahora cómo generar a los lenguajes regulares mediante gramáticas y su interrelación con los conjuntos regulares. Primero retomemos la definición de gramática lineal presentada en la sección precedente.

Definición. Una *gramática lineal derecha* (resp. *izquierda*) es una gramática independiente del contexto caracterizada por producciones de la forma:

$$A \mapsto bB \text{ (resp. } A \mapsto Bb) \text{ o } A \mapsto b; \text{ y posiblemente } E \mapsto \Lambda.$$

Ejemplos:

1. La gramática lineal derecha $\mathscr{G} = \langle \{a, b\}, \{E, A\}, E, \mathscr{R} \rangle$, donde $\mathscr{R}$ viene dado por las producciones $\mathscr{R} = \{E \mapsto \Lambda, E \mapsto aA, A \mapsto bA, A \mapsto a\}$, claramente genera el lenguaje representado por $\mathbf{ab^*a + \Lambda}$.
2. La gramática lineal izquierda $\mathscr{G} = \langle \{a, b\}, \{E\}, E, \mathscr{R} \rangle$, donde $\mathscr{R}$ está dado por las producciones:

$$\mathscr{R} = \left\{ \begin{array}{l} E \mapsto Ab, E \mapsto Ba, A \mapsto Ba, \\ B \mapsto Ab, A \mapsto a, B \mapsto b \end{array} \right\}$$

genera el lenguaje de cadenas alternadas de a's y b's, representado por la expresión regular $\mathbf{(\Lambda + b)(ab)^* + (\Lambda + a)(ba)^*}$. □

Definición. Dos gramáticas $\mathscr{G}$ y $\mathscr{G}'$ son *equivalentes* si y sólo si los lenguajes generados por ellas son iguales, *i.e.*, $\mathscr{L}(\mathscr{G}) = \mathscr{L}(\mathscr{G}')$.

La familia de los lenguajes generados bien sea por gramáticas lineales derechas o por las izquierdas es la misma.

Proposición 9.3. *Para cada gramática lineal derecha $\mathscr{G}$, existe una gramática lineal izquierda $\mathscr{G}'$ equivalente.*

Demostración.

Sea $\mathcal{G} = \langle \mathcal{V}_T, \mathcal{V}_N, E, \mathcal{R} \rangle$ una gramática lineal derecha. Asociemos a $\mathcal{R}$ un grafo dirigido de manera tal que sus vértices sean todos los símbolos no terminales más la palabra vacía Λ, y toda producción de la forma $A \mapsto cB$ le correspondemos un arco $\overrightarrow{AB}$ etiquetado con c. (Toda producción $A \mapsto c$ se reescribe como $A \mapsto c\Lambda$.) Así, para $\mathcal{R}$ tenemos,

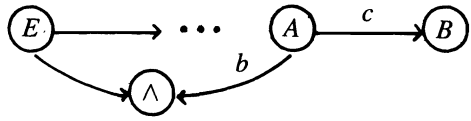


Figura 9.1

Para construir una gramática lineal izquierda $\mathcal{G}'$, notamos que ésta debe generar el lenguaje de derecha a izquierda. De aquí que si intercambiamos E por Λ e invertimos el sentido de los arcos del grafo asociado a $\mathcal{R}$, el grafo obtenido,

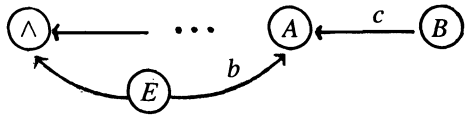


Figura 9.2

corresponde a una gramática lineal izquierda $\mathcal{G}'$ cuyas reglas de producción son de la forma $B \mapsto Ac$ o $C \mapsto b$ o $E \mapsto \Lambda$; la cual genera (¿por qué?) el mismo lenguaje que $\mathcal{G}$. ■

Ejemplo. Consideremos la gramática lineal derecha $\mathcal{G} = \langle \{a, b\}, \{E, A, B\}, E, \mathcal{R} \rangle$, donde $\mathcal{R} = \{E \mapsto \Lambda, E \mapsto aB, B \mapsto bB, B \mapsto aA, A \mapsto b\}$, que podemos representar con el grafo dirigido dado en la figura 9.3.

Entonces, la gramática lineal izquierda $\mathcal{G}' = \langle \{a, b\}, \{E, A, B\}, E, \mathcal{R}' \rangle$, donde, de la figura, $\mathcal{R}' = \{E \mapsto \Lambda, E \mapsto Ab, A \mapsto aB, B \mapsto bB, B \mapsto a\}$ es equivalente a $\mathcal{G}$. □

La representación con grafos dirigidos de las gramáticas lineales nos provee de una técnica heurística para "visualizar" (construir) el lenguaje generado por una

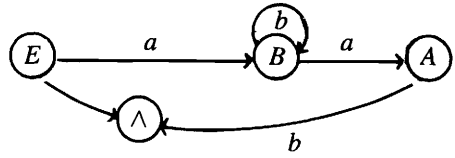


Figura 9.3

gramática, observando que las aplicaciones de las reglas se traducen en "flujos" sobre el grafo asociado.

Ejemplo. Consideremos la gramática lineal $\mathscr{G} = \langle\{a, b, c\}, \{E, A, B, C\}, E, \mathscr{R}\rangle$, donde $\mathscr{R} = \{E \mapsto cA, A \mapsto aA, A \mapsto bB, A \mapsto cC, B \mapsto aB, B \mapsto c, C \mapsto b\}$, que podemos representar con el grafo dirigido siguiente

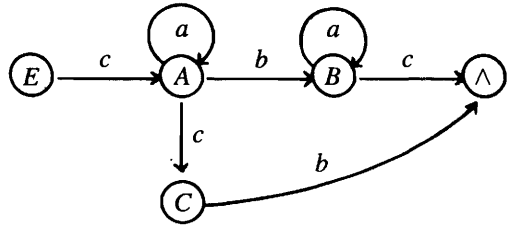


Figura 9.4

Del grafo, tenemos que toda $\alpha \in \mathscr{L}(\mathscr{G})$ debe constar de: 1) una c seguida de a's en un número arbitrario (posiblemente cero), una b, seguida también de arbitrarias (incluso cero) a's, y finalmente terminar en una c; o bien 2) una c seguida de a's en un número arbitrario (posiblemente cero), otra c, y finalmente de una b. Es decir, $\mathscr{L}(\mathscr{G})$ es el conjunto regular representado la expresión regular $\alpha =$ **ca*(ba*c + cb)**. □

Después de un momento de reflexión, podemos también concebir un recíproco heurístico para este resultado, *i.e.*, obtener una gramática lineal a partir de un conjunto regular. Esto se lleva a efecto analizando el proceso de formación de alguna expresión regular para el mismo.

Notación. Denotemos con $A_i \Rightarrow^* \alpha A_j$ la producción realizada para obtener $\alpha \in \mathscr{V}_T^*$ a partir de todas las reglas de la forma $A_i \mapsto aA$, empleando repetidamente

todas las reglas en $\mathscr{R}$ a condición de que la última regla utilizada permita aplicar alguna regla de la forma $A_j \mapsto bB$. En otros términos, si $\alpha = a_i \ldots a_j \in \mathscr{V}_T^*$, entonces $A_i \Rightarrow^* \alpha A_j$ significa que α se generó a partir de A_i mediante la aplicación sucesiva de reglas de la forma $A_i \mapsto a_i A_k$, $A_k \mapsto a_{i+1} A_m, \ldots, A_l \mapsto a_j A_j$, o sea al límite de la sucesión $A_i \Rightarrow a_i A_k$, $A_i \Rightarrow^* a_i \ldots a_j A_j = \alpha A_j$. Esta notación concuerda con la usada para definir a $\mathscr{L}(\mathscr{G})$, $E \Rightarrow^* \alpha$, pues esto se puede representar como $E \Rightarrow^* \alpha\Lambda$. Aquí, basta agregar Λ a los símbolos no terminales, de forma tal que si $A \mapsto b \in \mathscr{R}$, entonces lo expresamos $A \mapsto b\Lambda$.[3]

Estos procedimientos heurísticos son implementables como algoritmos, mecanismo que se sigue de la demostración del teorema siguiente. Con lo cual las expresiones $\mathscr{L}(\boldsymbol{\alpha})$ y $\mathscr{L}(\mathscr{G})$, denotando con la primera a un conjunto (representado por $\boldsymbol{\alpha}$) y con la segunda un lenguaje (generado por $\mathscr{G}$) determinarán así a los mismos objetos.

Teorema 9.4. *Un conjunto $\mathscr{L} \subseteq \mathscr{A}^*$ es regular si y sólo si es un lenguaje regular.*

Demostración.

En primera instancia, hagamos $\mathscr{V}_T = \mathscr{A}$.

$\Rightarrow$) Veamos cómo concebir a los conjuntos regulares como lenguajes regulares, usando inducción matemática sobre su formación.

(i) Para $\varnothing$, tenemos que es un lenguaje regular generado por la gramática cuya única regla de producción es $E \mapsto aE$.

(ii) El conjunto $\{\Lambda\}$ es un lenguaje regular generado, *e.g.*, por la gramática con regla de producción $E \mapsto \Lambda$.

(iii) Todo conjunto finito $\{a_1, a_2, \ldots, a_n\}$ es un lenguaje regular generado por la gramática con reglas de la forma $E \mapsto a_i$, para $i = 1, \ldots, n$.

(iv) Sean $\mathscr{L}_1$ y $\mathscr{L}_2$ dos conjuntos regulares tales que, por hipótesis inductiva, también son lenguajes regulares. Entonces los conjuntos siguientes son lenguajes regulares: $\mathscr{L}_1 \cup \mathscr{L}_2$, $\mathscr{L}_1 \mathscr{L}_2$ y $\mathscr{L}_1^*$. (Ejercicio 7).

Por consiguiente, partiendo de su definición, todo conjunto regular es un lenguaje regular.

[3] *Cf.* con los conceptos de *palabra adyacente* y *cadena deductiva* introducidos en la sección 3.3.

$\Leftarrow$) Sin pérdida de generalidad, consideremos un lenguaje $\mathscr{L}(\mathscr{G})$ generado por una gramática lineal derecha $\mathscr{G} = \langle \mathscr{A}, \mathscr{V}_N, E, \mathscr{R} \rangle$, con $\mathscr{V}_N = \{A_1, A_2, \ldots, A_n\}$, donde $A_1 = E$ (por la Proposición 9.3).

Probaremos que $\mathscr{L}(\mathscr{G})$ es un conjunto regular, expresándolo como una unión finita de ciertos conjuntos regulares obtenidos a partir de las reglas de producción de $\mathscr{G}$. Denotemos con $\bar{\mathscr{G}}$ a la gramática obtenida de $\mathscr{G}$ tal que $\bar{\mathscr{G}} = \langle \mathscr{A}, \mathscr{V}_N \cup \{A_{n+1}\}, E, \bar{\mathscr{R}} \rangle$, donde $A_{n+1} = \lambda (\cong \Lambda)$, y $\bar{\mathscr{R}}$ se obtiene de $\mathscr{R}$ sustituyendo cada producción $A_i \mapsto a$ por $A_i \mapsto a\lambda$. Bajo esta modificación, se tiene que $\mathscr{L}(\bar{\mathscr{G}}) = \mathscr{L}(\mathscr{G})$. Para $i, j = 1, \ldots, n$, y $k = 1, \ldots, n+1$, definamos a R_{ij}^k como el conjunto de todas las palabras $\alpha \in \mathscr{A}^*$ tales que $A_i \Rightarrow^* \alpha A_j$, con la restricción de que no se haya aplicado ninguna regla en la que ocurra alguna A_m, con $m \geq k$. Formalmente,

$$R_{ij}^k = \left\{ \alpha \in \mathscr{A}^* \;\middle|\; \begin{array}{l} A_i \Rightarrow^* \alpha A_j, \text{ y} \\ \text{si } A_i \Rightarrow^* \beta A_m \text{ y } A_m \Rightarrow^* \omega A_j, \text{ para algunas } \beta, \omega \in \mathscr{A}^*, \\ \text{entonces } \alpha = \beta\omega \text{ y } m < k \end{array} \right\}.$$

Obviamente, si $k = n + 1$, se sigue que $R_{1j}^{n+1} = \{\alpha \in \mathscr{A}^* : E \Rightarrow^* \alpha A_j\}$. Por tanto,

$$\alpha \in \mathscr{L}(\bar{\mathscr{G}}) \text{ si y sólo si } E \Rightarrow^* \alpha\lambda \text{ si y sólo si}$$
$$\alpha \in R_{1n+1}^{n+1} = \bigcup_j \{R_{1j}^{n+1} : A_j \mapsto a_k \in \mathscr{R}\}.$$

Probaremos (por inducción matemática sobre k) que cada conjunto R_{ij}^k es regular, y por consiguiente $\mathscr{L}(\mathscr{G})$ también.

Base: para $k = 1$, tenemos lo siguiente,

$$R_{ij}^1 = \begin{cases} \{a \in \mathscr{A} : A_i \mapsto aA_j\}, & \text{si } i \neq j \\ \{\Lambda\} \cup \{a \in \mathscr{A} : A_i \mapsto aA_j\}, & \text{si } i = j \end{cases} \tag{1}$$

pues, si $k = 1 \leq i, j$, las producciones son directas. Además, como estos conjuntos son finitos, entonces son regulares.

H.I.: Supongamos ahora que para $k = 1, \ldots, n$, todos los conjuntos R_{ij}^k son regulares.

De aquí se sigue que el conjunto

$$D_{ij}^{k+1} = R_i^k \cup R_{ik}^k (R_{kk}^k)^* R_{kj}^k \tag{2}$$

es un conjunto regular. La identidad $D_{ij}^{k+1} = R_{ij}^{k+1}$ se sigue de observar que para generar α en una producción $A_i \Rightarrow^* \alpha A_j$ sin emplear reglas en las que ocurran alguna A_m, con $m \geq k$, se debe cumplir que:

1. $\alpha \in R_{ij}^k$, *i.e.*, α se produjo usando todas las reglas en las que ocurren sólo A_l's con $l < k$, o bien,

2. $\alpha \in R_{ik}^k (R_{kk}^k)^* R_{kj}^k$, *i.e.*, α se produjo por las etapas:

- **a.** mediante producciones $A_i \Rightarrow^* \beta A_k$; luego,
- **b.** repetidamente de producciones $A_k \Rightarrow^* \gamma A_k$; y por último,
- **c.** por producciones $A_k \Rightarrow^* \omega A_j$,

usando en cada caso reglas en las que sólo ocurren A_l's con $l < k$. ■

La demostración del teorema anterior, siendo constructiva, nos provee además de un algoritmo para determinar el lenguaje generado por una gramática lineal; lenguaje que, en virtud del Teorema 9.2, podemos describir con una expresión regular. En lugar de calcular todos los conjuntos R_{ij}^k, el algoritmo en cuestión se inicia con la ecuación (2) para $k = n + 1$, y por medio de retroceso se obtiene el caso base $k = 1$, usando recursivamente a (2).

Ejemplo. Determinemos el lenguaje $\mathscr{L}$ generado por la gramática lineal derecha $\mathscr{G}$, $\mathscr{G} = \langle \{a, b\}, \{E, A\}, E, \mathscr{R} \rangle$, donde $\mathscr{R} = \{E \mapsto aE, E \mapsto bA, A \mapsto a\}$. Así, $\bar{\mathscr{G}}$ viene dada por $\bar{\mathscr{G}} = \langle \{a, b\}, \{E, A, \lambda\}, E, \bar{\mathscr{R}} \rangle$, donde $\bar{\mathscr{R}} = \{E \mapsto aE, E \mapsto bA, A \mapsto a\lambda\}$

Calculemos entonces $R_{13}^3 = \mathscr{L}(\bar{\mathscr{G}}) = \mathscr{L}(\mathscr{G})$.

Por la ecuación (2):

$$R_{13}^3 = R_{13}^2 \cup R_{12}^2 (R_{22}^2)^* R_{23}^2.$$

Ahora necesitamos de cuatro elementos para $k = 2$:

$$R^2_{12} = R^1_{12} \cup R^1_{11}(R^1_{11})^* R^1_{12}.$$
$$R^2_{13} = R^1_{13} \cup R^1_{11}(R^1_{11})^* R^1_{13}.$$
$$R^2_{22} = R^1_{22} \cup R^1_{21}(R^1_{11})^* R^1_{12}.$$
$$R^2_{23} = R^1_{23} \cup R^1_{21}(R^1_{11})^* R^1_{13}.$$

Por último, obtenemos los seis elementos para $k = 1$, empleando la ecuación (1) (el caso base):

$$R^1_{11} = \{\Lambda\} \cup \{a\} \equiv \mathbf{\Lambda + a} \simeq \mathbf{a}.$$
$$R^1_{12} = \{b\} \equiv \mathbf{b}.$$
$$R^1_{13} = R^1_{22} = R^1_{21} = \varnothing \equiv \emptyset.$$
$$R^1_{23} = \{a\} \equiv \mathbf{a}.$$

Sustituyendo estas expresiones recursivamente en los conjuntos para $k > 1$, y empleando las identidades para expresiones regulares, obtenemos:

$$R^2_{12} \equiv \mathbf{b + aa^*b} \simeq \mathbf{b + a^*b} \simeq \mathbf{(\Lambda + a^*)b} \simeq \mathbf{a^*b}.$$
$$R^2_{13} = R^2_{22} = \varnothing \equiv \emptyset.$$
$$R^2_{23} \equiv \mathbf{a + \emptyset a^* \emptyset} \simeq \mathbf{a}.$$

Y finalmente, $\mathscr{L}(\mathscr{G}) = R^3_{13} \equiv \mathbf{\emptyset + a^*b\ \emptyset^* a} \simeq \mathbf{a^*\ b\Lambda a} \simeq \mathbf{a^*ba}$. □

Ejercicios

1. Pruebe las identidades básicas para expresiones regulares 1–14.
2. Sea $\mathscr{A} = \{0, 1\}$. Usando las identidades básicas 1-14, pruebe la equivalencia de las expresiones:
   a. **(10)*1+(10)*(11+0)(0+1(10)*(11+0))*1(10)*1 ≃ (10+(11+0)0*1)*1.**
   b. **((1*0)*01*)* ≃ Λ + 0(0 + 1)* + (0 + 1)*00(0 + 1)***
3. Construya una gramática lineal que genere todas las palabras de ceros (0) y unos (1) teniendo el mismo número impar de ceros que de unos.

4. Pruebe que para toda gramática lineal derecha $\mathscr{G}$ existe una gramática lineal derecha equivalente $\mathscr{G}'$ tal que $\mathscr{R}'$ contiene producciones de la forma $B \mapsto \Lambda$, en donde $B \neq E$ y también de la forma $A \mapsto aE$.

5. Sea $\mathscr{G} = \langle \{a, b\}, \{E\}, E, \mathscr{R} \rangle$, donde $\mathscr{R} = \{E \mapsto aE, E \mapsto aE, E \mapsto b, E \mapsto a\}$. Describa (lo mejor posible) el lenguaje generado por ésta.

6. Sea $\mathscr{G} = \langle \{a, b\}, \{E, A, B\}, E, \mathscr{R} \rangle$, donde $\mathscr{R}$ está definido por las producciones

$$\mathscr{R} = \left\{ \begin{array}{l} E \mapsto aE, E \mapsto bA, A \mapsto aA \\ A \mapsto bB, B \mapsto aB, B \mapsto bE \end{array} \right\}$$

   Construya (con base en el Teorema 9.4) una expresión regular para $\mathscr{L}(\mathscr{G})$.

7. Sean $\mathscr{L}_1$ y $\mathscr{L}_2$ dos lenguajes regulares. Entonces los conjuntos siguientes son también lenguajes regulares: $\mathscr{L}_1 \cup \mathscr{L}_2$, $\mathscr{L}_1 \mathscr{L}_2$ y $\mathscr{L}_1^*$.

8. Sea $\mathscr{L} = \{\alpha \in \{a, b\}^* : \alpha \neq \Lambda$, y bb no es una subpalabra de $\alpha\}$
   a. Pruebe que $\mathscr{L}$ es un lenguaje regular.
   b. Halle una expresión regular $\boldsymbol{\alpha}$ tal que $\mathscr{L} = \mathscr{L}(\boldsymbol{\alpha})$.

9.4 Autómatas finitos

Hemos presentado los lenguajes regulares como conjuntos generados bien sea por operaciones sobre los conjuntos finitos de símbolos o por medio de las gramáticas lineales. Ahora vamos a definirlos como aquellas cadenas de símbolos que, impresos en una cinta, son aceptados por un tipo de dispositivo: el *autómata finito.* Desde esta perspectiva, un autómata puede considerarse como un *dispositivo* o *máquina para el reconocimiento de lenguajes*, es decir, un *algoritmo* diseñado para responder preguntas de la clase: dado un lenguaje $\mathscr{L}$, ¿la cadena $\alpha \in \mathscr{L}$? Así concebido, un autómata es una "caja negra" que proporciona respuestas SI/NO al alimentarlo con cintas impresas con cadenas de símbolos tomados de algún alfabeto preasignado. Podemos imaginarlo provisto con una "cabeza lectora" la cual lee un casillero (símbolo) de la cinta a la vez, y sólo entonces avanza sobre la cinta al próximo casillero, digamos el derecho. El dispositivo deja de operar tan pronto se complete la lectura de la cadena de símbolos bajo análisis, dando entonces su respuesta. La concepción interna suele simplificarse, y al mismo

tiempo rendir la suficiente generalidad, usando la noción de *estado interno.* Una ayuda gráfica nos la proporciona la figura siguiente, donde las letras q_i denotan a los estados internos. Nótese la restricción para la lectura de la cinta: sólo de izquierda a derecha (prohibido todo retroceso).

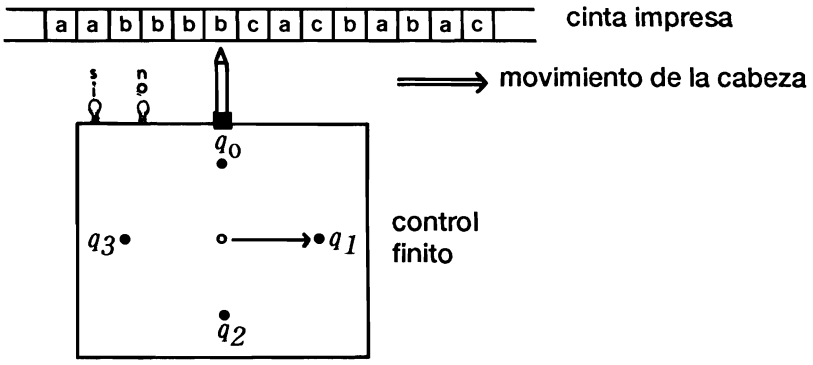


Figura 9.5

Contrario a las computadoras, un autómata finito carece de *memoria auxiliar,* sólo comparte con éstas el contar con un "procesador central" de capacidad finita fija, de acuerdo con el diseño original. Debido a que un autómata finito tiene sólo una oportunidad de leer un símbolo de la cinta, bajo el movimiento de izquierda a derecha, no hay ventaja alguna en proveer a la máquina con una "cabeza para escritura". Aun cuando se permitiera el movimiento en ambos sentidos, el dispositivo obtenido (*autómata finito de doble sentido*) no es más poderoso que el primero [RS]. Los autómatas finitos son particularmente importantes dado que sirven para controlar dispositivos más complejos, como las máquinas de Turing. Otra razón para emprender su estudio es su aplicabilidad para el diseño de algoritmos y programas comúnmente usados en computación, tal como analizadores sintácticos para compiladores (que suelen basarse en la "simulación" en un autómata).

A continuación describiremos la operación de un autómata finito a mayor detalle.

Definición. Un *autómata finito* $\mathcal{M}$ es una quíntupla $\mathcal{M} = \langle I, Q, q_0, \delta, F \rangle$, donde I es un conjunto finito de símbolos, denominado *alfabeto de cinta*, Q es un conjunto finito de *estados internos*, $q_0 \in Q$ es el *estado inicial* de la máquina, $\delta: Q \times I \to Q$ es una función, llamada *función de transición de estados*, y $F \subseteq Q$ es un conjunto de *estados finales*.

El autómata es "alimentado" con una cinta impresa, la cual lee de izquierda a derecha en forma secuencial (sin "brincos" ni retrocesos). Al comenzar la lectura de una cinta, el dispositivo se inicializa al estado q_o. La interpretación para la asignación $\delta(q_i, a) = q_j$, con $q_i, q_j \in Q$ y $a \in I$ es que estando la máquina en el estado q_i, lee el símbolo a, y luego mueve la cabeza lectora un casillero a la derecha cambiando su estado interno a q_j. El proceso se repite hasta completar la cadena de símbolos (palabra) de la cinta.

Definición. Un autómata finito $\mathcal{M}$ *acepta* una palabra de cinta si y sólo si al terminar de leerla se encuentra en alguno de sus estados finales. Caso contrario, decimos que *rechaza* la palabra.

Ejemplo. Consideremos el autómata finito $\mathcal{M} = \langle \{a, b\}, \{q_o, q_1\}, q_0, \delta, \{q_o\} \rangle$, donde δ está definida por la siguiente tabla:

Estado q	letra leída α	$\delta(q, \alpha)$
q_o	a	q_o
q_o	b	q_1
q_1	a	q_o
q_1	b	q_1

De aquí, es fácil verificar que las palabras:

$$\alpha_1 = \boxed{a}\boxed{a}\boxed{a}\boxed{a}\boxed{a}$$
$$\alpha_2 = \boxed{a}\boxed{b}\boxed{b}\boxed{a}$$
$$\alpha_3 = \boxed{b}\boxed{b}\boxed{b}\boxed{b}\boxed{a}$$

son todas aceptadas por el autómata. De hecho, $\mathcal{M}$ sólo acepta el conjunto de todas las palabras terminadas en "a". Dentro del contexto de lenguajes, sería el lenguaje regular $\mathcal{L}$**((a + b)*a)**. □

La *configuración* de un autómata $\mathcal{M}$ viene dada por la historia de sus *status* de control, cabeza lectora y cinta de entradas. Debido a la restricción sobre el sentido de lectura de la cinta, la porción de ésta una vez leída no puede influir sobre la operación futura del autómata. La configuración depende exclusivamente del estado actual y de la parte aún no leída (procesada) de la palabra de cinta En otros términos, la *configuración de un autómata finito* es cualquier elemento $(q, \alpha) \in Q \times I^*$.

Para representar el proceso de que un autómata finito $\mathcal{M}$ pase a una configuración a partir de la actual, contamos con las definiciones siguientes.

> **Definición.** Sean $\mathcal{M} = \langle I, Q, q_o, \delta, F\rangle$ un autómata finito y $(q, \alpha), (q', \alpha')$ dos configuraciones de $\mathcal{M}$. Decimos que $\mathcal{M}$ *mueve* o *transforma directamente* (o en *un sólo paso*) (q, α) en (q', α'), y lo denotamos $(q, \alpha) \models\!\!\Rightarrow (q', \alpha')$, si y sólo si $\alpha = a\alpha'$, para algún $a \in I$, y $\delta(q, a) = q'$.

Notemos que $\models\!\!\Rightarrow$ puede considerarse como una función $\models\!\!\Rightarrow: Q \times I^+ \to Q \times I^*$. Podemos extender la función $\models\!\!\Rightarrow$ para que esté definida en $Q \times I^*$, haciendo simplemente que sea constante ante configuraciones del tipo (q, Λ) (*i.e.*, al alimentar a $\mathcal{M}$ con una cinta no impresa).

Las aplicaciones sucesivas de $\models\!\!\Rightarrow$ sobre una configuración fija (q, α) producen un decremento paulatino en la (porción de) palabra α (*i.e.*, ésta va siendo leída) hasta conducir eventualmente a la palabra vacía Λ, dándose por terminada la lectura de α: sea $\alpha = a_o \ldots a_n \in I^*$, y denotemos con ${}_i\alpha_j$ a la subcadena $a_i a_{i+1} \ldots a_j$, para $i \leq j$, entonces tenemos la sucesión,

$$(q_o, \alpha) \models\!\!\Rightarrow (q_{1,1}\alpha_n) \models\!\!\Rightarrow (q_{2,2}\alpha_n) \models\!\!\Rightarrow \cdots \models\!\!\Rightarrow (q_n, a_n) \models\!\!\Rightarrow (q_{n+1}, \Lambda) \models\!\!\Rightarrow \cdots \models\!\!\Rightarrow (q_{n+1}, \Lambda).$$

Esta sucesión la podemos representar como $(q_o, \alpha) \models\!\!\Rightarrow^* (q_{n+1}, \Lambda)$, *i.e.*, la *clausura reflexiva* y *transitiva* de $\models\!\!\Rightarrow$.

> **Definición.** Dadas dos configuraciones (q, α) y (q', α') de un autómata $\mathcal{M}$, decimos que (q, α) se *transforma* en (q', α'), y lo denotamos $(q, \alpha) \models\!\!\Rightarrow^* (q', \alpha')$, si y sólo si (q', α') se obtiene de (q, α) en un número finito (posiblemente cero) de transformaciones directas.

Notación. Representaremos (bajo abuso de notación) con $q_i \models\!\!\Rightarrow^* \alpha q_j$ a la transformación $(q_i, \alpha) \models\!\!\Rightarrow^* (q_j, \Lambda)$, *i.e.*, el proceso completo realizado por el autómata $\mathscr{M}$ en leer la palabra α iniciando en el estado q_i. La relevancia de esta notación se hará patente posteriormente.

La aceptación de palabras en esta terminología se traduce como sigue.

Definición. Una palabra $\alpha \in I^*$ es *aceptada* por un autómata finito $\mathscr{M}$ si y sólo si existe un estado $q_f \in F$ tal que $q_o \models\!\!\Rightarrow^* \alpha q_f$.

Definición. El *lenguaje aceptado* por un autómata finito $\mathscr{M}$, $\mathscr{L}(\mathscr{M})$, es el conjunto de todas las palabras de cinta de I^* aceptadas por $\mathscr{M}$, *i.e.*

$$\mathscr{L}(\mathscr{M}) = \{\alpha \in I^* : \exists q_f \in F, q_0 \models\!\!\Rightarrow^* \alpha q_f\}$$

Una formulación alternativa para dar cuenta del procesamiento de palabras por un autómata finito se obtiene en términos de aplicaciones iteradas de la función de transición. Esto conlleva a una extensión de la función de transición δ para operar sobre palabras en I^*, definida recursivamente. En efecto, consideremos una palabra $\alpha = a_0 \ldots a_{n-1} \in I^*$. Si ahora $\mathscr{M}$ se halla en el estado inicial q_o, renombremos con q_1 el estado resultante de estar $\mathscr{M}$ en q_o y leer a_o, *i.e.*,

$$(q_o, \alpha) \models\!\!\Rightarrow (\delta(q_o, a_o), {}_1\alpha_{n-1}) = (\delta_1(q_o, a_o), {}_1\alpha_{n-1}) = (q_1, {}_1\alpha_{n-1})$$

Nótese que también renombramos a δ como $\delta_1 : Q \times I \to Q$. Repitamos el proceso para obtener q_2 a partir de (q_1, a_1) mediante δ, renombrando una vez más a δ:

$$\begin{aligned}(q_1, {}_1\alpha_{n-1}) = (q_1, a_1 \cdot {}_2\alpha_{n-1}) \models\!\!\Rightarrow (\delta(q_1, a_1), {}_2\alpha_{n-1}) &= (\delta(\delta_1(q_o, a_o), a_1), {}_2\alpha_{n-1}) \\ &:= (\delta_2(q_o, a_o a_1), {}_2\alpha_{n-1}) \\ &= (q_2, {}_2\alpha_{n-1})\end{aligned}$$

donde $\delta_2 : Q \times I^2 \to Q$. Procediendo según este orden, obtendremos finalmente una función $\delta_n : Q \times I^n \to Q$, dada por:

$$q_n = \delta_n(q_o, \alpha_{n-1}) = \delta_n(q_o, \alpha) = \delta(\delta_{n-1}(q_o, {}_o\alpha_{n-2}), a_{n-1}).$$

Suprimiendo subíndices (lo cual no causa ambigüedades), podemos concebir la extensión de δ: $Q \times I \to Q$ a la función $\hat{\delta}$: $Q \times I^* \to Q$ definida por recursión como:

$$\hat{\delta}(q, \Lambda) = q$$
$$\hat{\delta}(q, \alpha a) = \delta(\hat{\delta}(q, \alpha), a), \text{ para todo } q \in Q, \alpha \in I^* \text{ y } a \in I$$

Obsérvese que como extensión, restringida a elementos de I se reduce a δ:

$$\hat{\delta}(q, a) = \hat{\delta}(q, \Lambda a) = \delta(\hat{\delta}(q, \Lambda), a) = \delta(q, a)$$

Es usual identificar ambas funciones y renombrar a $\hat{\delta}$ como δ, sin problema de confusiones.

En esta versión, la aceptación de palabras se traduce como sigue.

Definición. Una palabras $\alpha \in I^*$ es *aceptada* por un autómata $\mathscr{M}$ si y sólo si existe $q_f \in Q$ tal que $q_f = \delta(q_o, \alpha) \in F$.

Definición. El *lenguaje aceptado* por un autómata $\mathscr{M}$, $\mathscr{L}(\mathscr{M})$, está dado por el conjunto

$$\mathscr{L}(\mathscr{M}) = \{\alpha \in I^* : \exists q_f \in F, q_f = \delta(q_o, \alpha)\}$$

El significado de $\delta(q, \alpha)$ es sencillo: consiste en aquel estado de la máquina $\mathscr{M}$ obtenido de comenzar en el estado q e ir leyendo toda la palabra α símbolo a símbolo, cambiando de estados de acuerdo con la tabla dada de movimientos (δ).

Retomemos el autómata finito del ejemplo anterior.

Ejemplo. Si $\mathscr{M}$ es alimentado con la palabra $\alpha = aabba$, su configuración inicial es $(q_o, aabba)$. De aquí se sigue el proceso siguiente:

$$\begin{aligned}(q_o, aabba) &\models (q_o, abba)\\ &\models (q_o, bba)\\ &\models (q_1, ba)\\ &\models (q_1, a)\\ &\models (q_o, \Lambda), \text{ con } q_o \in F\end{aligned}$$

Por lo que $q_o \models\!\!\Rightarrow^* aabbaq_o$, y la palabra se acepta. Bajo el enfoque recursivo, el proceso es:

$$\begin{aligned}
\delta(q_o, aabba) &= \delta(\delta(q_o, aabb), a) \\
&= \delta(\delta(\delta(q_o, aab), b), a) \\
&= \delta(\delta(\delta(\delta(q_o, aa), b), b), a) \\
&= \delta(\delta(\delta(\delta(\delta(q_o, a), a), b), b), a) \\
&= \delta(\delta(\delta(\delta(q_o, a), b), b), a) \\
&= \delta(\delta(\delta(q_o, b), b), a) \\
&= \delta(\delta(q_1, b), a) \\
&= \delta(q_1, a) = q_o \in F
\end{aligned}$$ □

Desde una perspectiva computacional, tenemos que el procedimiento (algoritmo) para el reconocimiento de palabras mediante autómatas finitos puede considerarse tanto iterativo ($\models\!\!\Rightarrow^*$) como recursivo ($\hat{\delta}$), obteniendo iguales resultados.

La representación tabular para la función de transición δ puede no ser la más clara. Es usual, y más conveniente, emplear una representación gráfica para δ en términos de grafos dirigidos, denominada *diagrama de transición de estados.*

> **Definición.** Sea $\mathcal{M}$ un autómata finito. El *diagrama de transición de estados* de $\mathcal{M}$ es un grafo dirigido en el que cada estado es representado por un vértice, y los arcos se etiquetan con elementos de I de forma tal que si hay un arco de q a q', su etiqueta es a si y sólo si $\delta(q, a) = q'$. Los estados finales se indican con círculos dobles y el inicial se señala con el símbolo $>$.

Ejemplos:

1. Retomando nuevamente el autómata anterior, su diagrama de transición de estados correspondiente es:

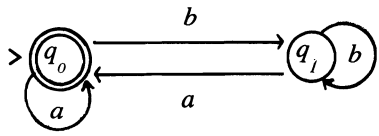


Figura 9.6

2. Sea $\mathscr{M} = \langle \{a, b\}, \{q_o, q_1, q_2\}, q_o, \delta, \{q_2\} \rangle$ un autómata finito, donde δ viene representada por el siguiente diagrama de transición de estados:

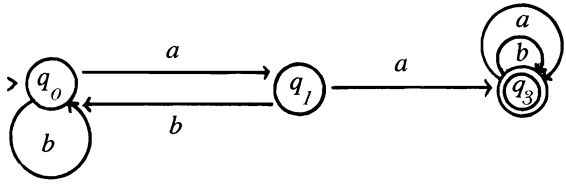


Figura 9.7

Este autómata acepta todas las palabras sobre $\{a, b\}$ con exactamente dos a's, *i.e.*, el lenguaje regular $\mathscr{L}(\mathscr{M}) = \mathscr{L}(\mathbf{(a + b)*aa(a + b)*})$. □

Procedamos ahora con el enfoque para el reconocimiento de los lenguajes regulares vía los autómatas finitos. Consideremos un autómata finito $\mathscr{M}$ dado, el cual acepta un lenguaje $\mathscr{L}(\mathscr{M})$. Con base en $\mathscr{M}$ obtendremos una gramática $\mathscr{G}$ tal que $\mathscr{L}(\mathscr{M}) = \mathscr{L}(\mathscr{G})$; es decir, las ebf's producidas por la gramática serán precisamente las palabras aceptadas por el autómata $\mathscr{M}$.

Sea $\mathscr{M} = \langle I, Q, q_o, \delta, F \rangle$ un autómata finito. El algoritmo para determinar la gramática asociada a $\mathscr{M}$ es el siguiente

Algoritmo: para el conjunto de símbolos terminales $\mathscr{V}_T$, hacemos $\mathscr{V}_T = I$; para el conjunto de símbolos $\mathscr{V}_N$ asociamos al estado inicial q_o el símbolo E, y a todos los demás estados de $\mathscr{M}$, q_i, los símbolos A_i, respectivamente, *i.e.*, $\mathscr{V}_N = \{E, A_1, \ldots A_n\}$; y el conjunto de reglas de producción viene dado por:

$$\mathscr{R} = \begin{cases} A_i \mapsto a_{ij} A_j, & \text{si } q_i \models\!\Rightarrow a_{ij} q_j \\ A_j \mapsto \Lambda, & \text{si } q_j \in F \end{cases}$$

Es fácil ver que esta gramática es lineal derecha.[4]

Ejemplo. Retomemos el autómata finito dado en el ejemplo 2) anterior:

$$\mathscr{M} = \langle \{a, b\}, \{q_o, q_1, q_2\}, q_o, \delta, \{q_2\} \rangle,$$

[4]Nótese que hemos agregado otras producciones a las gramáticas, basándonos en el Ejercicio 4 de la sección anterior.

con δ representada por el grafo de la figura 9.7.

Entonces la gramática lineal derecha correspondiente es: $\mathscr{G} = \langle \{a, b\}, \{E, A_1, A_2\}, E, \mathscr{R}\rangle$, donde $\mathscr{R}$ son las reglas de producción siguientes

$$\mathscr{R} = \left\{ \begin{array}{lll} (1)\ E \mapsto bE, & (4)\ A_1 \mapsto aA_2, & (7) A_2 \mapsto \Lambda \\ (2)\ E \mapsto aA_1, & (5)\ A_2 \mapsto aA_2, & \\ (3)\ A_1 \mapsto bE, & (6)\ A_2 \mapsto bA_2, & \end{array} \right\}$$

Así, el reconocimiento y producción correspondiente de la expresión $\alpha = abbaab$ es:

Transformaciones de $\mathscr{M}$	Producciones en $\mathscr{G}$	Regla empleada de $\mathscr{R}$
$(q_o, abbaab) \models\!\!\Rightarrow (q_1, bbaab)$	$E \Rightarrow aA_1$	(2)
$\models\!\!\Rightarrow (q_o, baab)$	$\Rightarrow abE$	(1)
$\models\!\!\Rightarrow (q_o, aab)$	$\Rightarrow abbE$	(1)
$\models\!\!\Rightarrow (q_1, ab)$	$\Rightarrow abbaA_1$	(2)
$\models\!\!\Rightarrow (q_2, b)$	$\Rightarrow abbaaA_2$	(4)
$\models\!\!\Rightarrow (q_2, \Lambda)$	$\Rightarrow abbaabA_2$	(6)
	$\Rightarrow abbaab\Lambda$	(7)
	$= abbaab$	
Por tanto, como $q_o \models\!\!\Rightarrow^* aabbaq_2$, con $q_2 \in F$, entonces, $abbaab \in \mathscr{L}(\mathscr{M})$	En consecuencia, como $E \Rightarrow^* aabba$, entonces, $abbaab \in \mathscr{L}(\mathscr{G})$	

□

Es un hecho, que todo estado en una computación depende sólo de la porción de palabra ya leída y no de la porción por procesar. Esto se describe formalmente con el lema siguiente.

Lema 9.5. *Sea $\mathscr{M} = \langle I, Q, q_o, \delta, F\rangle$ un autómata finito. Sean $q_i, q_j \in Q$ y α, $\beta \in I^*$. Entonces, $q_i \models\!\!\Rightarrow^* \alpha\beta q_j$ si y sólo si para algún $q_k \in Q$, $q_i \models\!\!\Rightarrow^* \alpha q_k$ y $q_k \models\!\!\Rightarrow^* \beta q_j$.*

Demostración. (Ejercicio 1). ■

Observación. Del ejemplo anterior, notamos que para una palabra (o una porción de ésta) $\omega \in I^*$ siendo procesada por el autómata, las transformaciones

$q_i \models\!\Rightarrow^* \omega q$, se corresponden con producciones del tipo $A_i \Rightarrow^* \omega A_j$, *e.g.*, $A_1 \Rightarrow^* abA_2 \longleftrightarrow q_1 \models\!\Rightarrow^* abq_2$.

A partir de esta observación y el lema anterior se sigue el resultado siguiente.

Proposición 9.6. *Sea* $\mathscr{M} = \langle I, Q, q_o, \delta, F\rangle$ *un autómata finito. Entonces,* $q_o \models\!\Rightarrow^* \alpha q_j$ *si y sólo si* $E \Rightarrow^* \alpha A_j$, *para toda* $\alpha \in I^*$.

Demostración. Por inducción matemática sobre la longitud de α.

Base: $\alpha = \Lambda$. Entonces, $q_o \models\!\Rightarrow^* \Lambda q_j$ si y sólo si $q_o = q_j$, o sea, $E \Rightarrow^* \Lambda E = E$.

Hipótesis inductiva: Supongamos válido el resultado para toda $\beta \in I^*$ tal que $|\beta| < |\alpha|$, *i.e.*, $q_o \models\!\Rightarrow^* \beta q_i$ si y sólo si $E \Rightarrow^* \beta A_i$.

$\Leftarrow$) Consideremos que $E \Rightarrow^* \beta a_k A_j$. De la hipótesis inductiva, supongamos cierto $E \Rightarrow^* \beta A_i$. Por lo tanto, $\mathscr{R}$ debe tener una regla de producción $A_i \mapsto a_k A_j$. Luego, por definición, tenemos que $q_i \models\!\Rightarrow^* a_k q_j$, y por la hipótesis inductiva, $q_o \models\!\Rightarrow^* \beta q_i$. Así, $q_o \models\!\Rightarrow^* \beta a_k q_j$, por el lema anterior.

$\Rightarrow$) Consideremos ahora que $q_o \models\!\Rightarrow^* \beta a_k q_j$. Entonces, por el lema anterior, existe $q_i \in Q$ tal que $q_o \models\!\Rightarrow^* \beta q_i$ y $q_i \models\!\Rightarrow^* a_k q_j$. Por la H. I., $E \Rightarrow^* \beta A_i$, y por definición, tenemos la regla $A_i \mapsto a_k A_j \in \mathscr{R}$. Por lo tanto, $E \Rightarrow^* \beta a_k A_j$. ■

Del ejemplo anterior, se obtuvo también que:

$$q_o \models\!\Rightarrow^* aabbaq_2, \text{ con } q_2 \in F \longleftrightarrow E \Rightarrow^* aabba$$

i.e., la palabra *aabba* es aceptada por el autómata $\mathscr{M}$ si y sólo si es producida por la gramática derivada de $\mathscr{M}$.

Este resultado, genérico para todos los autómatas finitos, se debe a Chomsky y Miller.

Teorema 9.7. *Sea* $\mathscr{M} = \langle I, Q, q_o, \delta, F\rangle$ *un autómata finito. Si* $\mathscr{L}(\mathscr{M}) \subseteq I^*$ *es el lenguaje aceptado por* $\mathscr{M}$, *entonces* $\mathscr{L}(\mathscr{M})$ *es un lenguaje regular.*

Demostración. Anteriormente dimos un algoritmo para hallar una gramática lineal derecha $\mathcal{G}$ a partir de un autómata finito ($n + 1$ estados): $\mathcal{G} = \langle I, \{E, A_1, \ldots A_n\}, E, \mathcal{R}\rangle$. Por lo tanto, sólo resta probar para el lenguaje generado por esta gramática, $\mathcal{L}(\mathcal{G})$, que $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{M})$. Para esto, simplemente observamos que $\alpha \in \mathcal{L}(\mathcal{G})$, *i.e.*, $E \Rightarrow^* \alpha$ si y sólo si $E \Rightarrow^* \alpha A_j$ para algún A_j y una regla de producción $A_j \mapsto \Lambda$, *i.e.*, $q_j \in F$. Ahora, por la Proposición 9.6, $E \Rightarrow^* \alpha A_j$ si y sólo si $q_o \models^* \alpha q_j$. Así, $E \Rightarrow^* \alpha$ si y sólo si $q_o \models^* \alpha q_j$ y $q_j \in F$. En otros términos, $\alpha \in \mathcal{L}(\mathcal{G})$ si y sólo si $\alpha \in \mathcal{L}(\mathcal{M})$. ■

Corolario 9.8. *El Lenguaje aceptado por un autómata finito es un conjunto regular.* ■

Uno podría en primera instancia concebir un proceso inverso para dada una gramática lineal (derecha) $\mathcal{G}$, construir un autómata finito $\mathcal{M}$ tal que $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{M})$. Atendiendo sólo a la interrelación producción-transformación, tenemos que si $\mathcal{G}$ cuenta con la regla de producción $A_i \mapsto bA_j$, uno asociaría el arco $\overrightarrow{q_i q_j}$ etiquetándolo con b en el diagrama de transición de estados en construcción. Sin embargo, la definición de gramática lineal incluye producciones como $A_i \mapsto bA_k$. Así, el autómata $\mathcal{M}$ al estar en el estado q_i y leer b tendría dos posibles estados por tomar: q_j ó q_k. De esta manera, se llega al *indeterminismo* de los autómatas. Por esto no debe entenderse la introducción de la aleatoriedad o probabilidad en la teoría. El autómata no elige un estado al azar entre los posibles, sino que elige alguno que lleve a la respuesta correcta. El indeterminismo radica en que uno ignora cuál es la sucesión de configuraciones que realiza un autómata finito indeterminista para dar con la respuesta correcta: en cierta forma, "hace magia". Al dotar con indeterminismo a estos autómatas, paradójicamente no resultan máquinas más poderosas que los autómatas deterministas (los estudiados hasta ahora). El resultado es que la familia de lenguajes reconocibles por ambos tipos de autómatas es la misma: lenguajes regulares. Los autómatas indeterministas son más sencillos de representar que los deterministas, y usualmente se emplean como un artificio intermedio para construir a estos últimos.

Ejemplo. Consideremos la gramática $\mathcal{G} = \langle \{a, b\}, \{E, A_1, A_2\}, E, \mathcal{R}\rangle$, con las reglas de producción:

$$\mathscr{R} = \begin{Bmatrix} (1)\ E \mapsto aA_1, & (4)\ A_2 \mapsto aA_1, & (7) A_2 \mapsto bE \\ (2)\ E \mapsto bA_1, & (5)\ A_2 \mapsto a, & \\ (3)\ A_1 \mapsto aA_2, & (6)\ A_2 \mapsto bA_2, & \end{Bmatrix}$$

Su grafo correspondiente el siguiente:

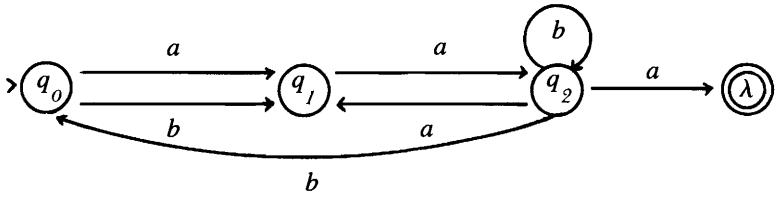


Figura 9.8

el cual guarda serias diferencias con respecto al diagrama de transición de estados de un autómata finito:

1. El diagrama está incompleto, pues el vértice q_1 sólo tiene un arco que emana de él, etiquetado con a. Ante la configuración $(q_1, b\omega)$ el autómata "no sabría" cómo proceder. Esto puede corregirse añadiendo un vértice adicional como se muestra en la Figura 9.9. Esta modificación no altera al conjunto de palabras aceptadas por $\mathscr{M}$.

2. El vértice q_2 "padece" de indeterminismo, teniendo dos arcos etiquetados con "b" y dos con "a" emanando de él.

> **Definición.** Un *autómata finito indeterminista* $\mathscr{M}$ es una quíntupla $\mathscr{M} = \langle I, Q, q_o, \Delta, F\rangle$, donde I es un conjunto finito, denominado el *alfabeto de cinta*, Q es un conjunto finito de *estados internos*, $q_o \in Q$ es el *estado inicial*, Δ es un subconjunto finito de $Q \times I^* \times Q$ llamado *relación de transición de estados*, y $F \subseteq Q$ es el conjunto de *estados finales*.

Nótese que para este autómata se tiene una relación para la transición de estados. La terna $(q_i, \alpha, q_j) \in \Delta$ si y sólo si $\mathscr{M}$ estando en el estado q_i y leyendo la cadena $\alpha \in I^*$ entra en el estado q_j. Gráficamente le corresponde el arco $\overrightarrow{q_i q_j}$ etiquetado

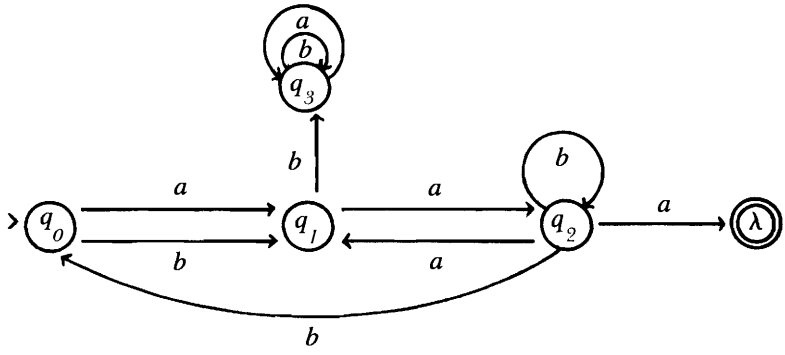


Figura 9.9

con α. Bajo la restricción de finitud de los dispositivos, se consideró Δ un conjunto finito, aún cuando $Q \times I^* \times Q$ sea infinito.

De manera análoga que los autómatas finitos deterministas, una configuración de $\mathcal{M}$ es simplemente un elemento $(q, \alpha) \in Q \times I^*$. La definición para la transición entre configuraciones es como sigue.

Definición. Sean $\mathcal{M} = \langle I, Q, q_o, \Delta, F \rangle$ un autómata finito indeterminista y (q, α), (q', α') dos configuraciones de $\mathcal{M}$. Decimos que (q, α) se *transforma directamente* (o en *un sólo paso*) en (q', α'), y lo denotamos $(q, \alpha) \models\!\!\Rightarrow (q', \alpha')$, si y sólo si existe $\beta \in I^*$ tal que $\alpha = \beta\alpha'$ y $(q, \beta, q') \in \Delta$.

Nótese que en este caso, $\models\!\!\Rightarrow$ no es necesariamente una función.

Definición. Denotamos con $\models\!\!\Rightarrow^*$ a la *clausura reflexiva* y *transitiva* de $\models\!\!\Rightarrow$.

Notación. A semejanza con el caso determinista, representaremos (bajo abuso de notación) con $q_i \models\!\!\Rightarrow^* \alpha q_j$ a la transformación $(q_i, \alpha) \models\!\!\Rightarrow^* (q_j, \Lambda)$.

Nótese que, a diferencia de un autómata finito determinista, uno indeterminista puede hacer tranformaciones ante cintas no impresas, *i.e.*, $(q_i, \Lambda) \models\!\!\Rightarrow^* (q_j, \Lambda)$, (o bien $q_i \models\!\!\Rightarrow^* q_j$) con $q_i \neq q_j$, o sea "saltos entre estados sin lectura".

Definición. Una palabra de cinta $\alpha \in I^*$ es *aceptada* por un autómata finito indeterminista $\mathcal{M}$ si y sólo si existe un estado $q_f \in F$ tal que $q_o \models\!\!\Rightarrow^* \alpha q_f$.

El *lenguaje aceptado* por $\mathscr{M}$, denotado $\mathscr{L}(\mathscr{M})$, está definido por

$$\mathscr{L}(\mathscr{M}) = \{\alpha \in I^* : \exists q_f \in F, q_o \models^* \alpha q_f\}$$

Ejemplo. Consideremos el autómata finito indeterminista $\mathscr{M} = \langle\{a, b\}, \{q_o, q_1\}, q_o, \Delta, \{q_1\}\rangle$, con Δ definida por el diagrama siguiente:

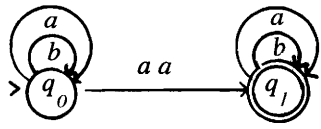


Figura 9.10

Este autómata sólo admite a las palabras que contienen la subpalabra "*aa*", *i.e.*, acepta el mismo lenguaje que el autómata determinista con diagrama de transición de estados mostrado por la Figura 9.7. □

La versión indeterminista de parte del Lema 9.5 es la siguiente.

Lema 9.9. *Sea* $\mathscr{M} = \langle \mathscr{A}, Q, q_o, \Delta, F\rangle$ *un autómata finito indeterminista; y sean* q_i *y* q_j *estados y* $\alpha, \beta \in \mathscr{A}^*$. *Entonces* $q_i \models^* \alpha\beta q_j$, *si para algún* $q_k \in Q$, $q_i \models \alpha q_k$ *y* $q_k \models \beta q_j$.

Demostración.

$\Rightarrow$) Supongamos que para algún $q_k \in Q$, $q_i \models^* \alpha q_k$ y $q_k \models^* \beta q_j$. Por definición, $q_i \models^* \alpha q_k$ significa que existen $n \geq 0$, $q_{i_o}, q_{i_1}, \ldots, q_{i_n} \in Q$, y α_o, $\alpha_1, \ldots, \alpha_n \in \mathscr{A}^*$, tales que

$$(q_i, \alpha) = (q_{i_o}, \alpha_o) \models (qi_1, \alpha_1) \models \cdots \models (q_{i_n}, \alpha_n) = (q_k, \Lambda)$$

Debido a que $(q_{i_m}, \alpha_m) \models (q_{i_{m+1}}, \alpha_{m+1})$, por definición de $\models$, existe una $\beta_m \in \mathscr{A}^*$ tal que $\alpha_m = \beta_m \alpha_{m+1}$ y $(q_{i_m}, \beta_m, q_{i_{m+1}}) \in \Delta$, para $m \geq 1$. Además,

como $\alpha_m\beta = \beta_m\alpha_{m+1}\beta$, se sigue que $(q_{i_m}, \alpha_m\beta) \models\!\Rightarrow (q_{i_{m+1}}, \alpha_{m+1}\beta)$. Por tanto,

$$(q_i, \alpha\beta) = (q_{i_o}, \alpha_o\beta) \models\!\Rightarrow (q_{i_1}, \alpha_1\beta) \models\!\Rightarrow \cdots \models\!\Rightarrow (q_{i_n}, \alpha_n\beta) = (q_k, \beta)$$

de donde, $(q_i, \alpha\beta) \models\!\Rightarrow^* (q_k, \beta)$. Ahora, como por hipótesis $q_k \models\!\Rightarrow^* \beta q_j$, de la transitividad de $\models\!\Rightarrow^*$, se obtiene que $q_i \models\!\Rightarrow^* \alpha\beta q_j$. ■

Proposición 9.10. *Sea $\mathcal{M} = \langle I, Q, q, \Delta, F\rangle$ un autómata finito indeterminista. Entonces, $q_o \models\!\Rightarrow^* \alpha q_j$ si y sólo si $E \Rightarrow^* \alpha A_j$, para toda $\alpha \in I^*$.*

Demostración. (Ejercicio 3). ■

Con este dispositivo indeterminista, podemos contar con un recíproco parcial para la interrelación entre los lenguajes regulares y los conjuntos aceptados por los autómatas finitos.

Teorema 9.11. *Sea $\mathcal{G} = \langle \mathcal{V}_T, \mathcal{V}_N, E, \mathcal{R}\rangle$ una gramática lineal derecha cuyo lenguaje es $\mathcal{L}(\mathcal{G})$. Entonces existe un autómata finito indeterminista $\mathcal{M} = \langle I, Q, q_o, \Delta, F\rangle$ tal que $\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{G})$.*

Demostración. Construyamos $\mathcal{M}$ a partir de la gramática $\mathcal{G}$ de la forma siguiente:

Identifiquemos $I = \mathcal{V}_T$, y asociemos $E \to q_o$ y para $i = 1, \ldots, n-1$, $A_i \to q_i$. Entonces,

$$\begin{aligned} Q &:= \{q_o, q_1, \ldots, q_n\} \cup \{\lambda\} \\ \Delta &:= \{(q_i, a_k, q_j) \colon A_i \mapsto a_k q_j \in \mathcal{R}\} \cup \{(q_i, a_k, \lambda) \colon A_i \mapsto a_k \in \mathcal{R}\} \\ F &:= \{q_j \colon A_j \mapsto \Lambda \in \mathcal{R}\} \cup \{\lambda\}. \end{aligned}$$

Sólo requerimos probar que $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{M})$, *i.e.*, para toda $\alpha \in \mathcal{V}_T^*$, $E \Rightarrow^* \alpha$ si y sólo si $q_o \models\!\Rightarrow^* \alpha q_j$, con $q_j \in F$.

$\Rightarrow$) Sea $\alpha \in \mathcal{V}_T^*$, $E \Rightarrow^* \alpha$. Entonces tenemos los casos:

1. $E \Rightarrow^* \alpha A_j \Rightarrow \alpha$, *i.e.*, $A_j \mapsto \Lambda \in \mathcal{R}$. Luego, $q_j \in F$, por definición, y $q_o \models\!\Rightarrow^* \alpha q_j$ por la proposición anterior. Por tanto, $\alpha \in \mathcal{L}(\mathcal{M})$.

2. $E \Rightarrow^* \beta A_i \Rightarrow \beta a_k = \alpha$, *i.e.*, $A_i \mapsto a_k \in \mathscr{R}$. Entonces $q_o \models\!\Rightarrow^* \beta q_i$ y $q_i \models\!\Rightarrow^* a_k\lambda$ y $\lambda \in F$. En consecuencia, como $q_o \models\!\Rightarrow^* \beta a_k\lambda = \alpha\lambda$, con $\lambda \in F$, se tiene $\alpha \in \mathscr{L}(\mathscr{M})$.

$\Leftarrow$) Sea $\alpha \in \mathscr{V}_T^*$ tal que $\alpha \in \mathscr{L}(\mathscr{M})$, *i.e.*, $q_o \models\!\Rightarrow^* \alpha q_j$, con $q_j \in F$. Se contemplan dos casos:

1. $q_j \neq \lambda$. Entonces, por la proposición anterior, $E \Rightarrow^* \alpha A_j$. Ahora, como $q_j \in F$ se tiene la regla $A_j \mapsto \Lambda \in \mathscr{R}$, de donde, $E \Rightarrow^* \alpha$.

2. $q_j = \lambda$. Así, $q_o \models\!\Rightarrow^* \alpha\lambda$, y por tanto, $\alpha \neq \Lambda$ (pues $q_o \neq \lambda$), digamos $\alpha = \beta a_k$, con $a_k \in \mathscr{V}$. Entonces existe $q_i \in Q$ tal que $q_o \models\!\Rightarrow^* \beta q_i$ y $q_i \models\!\Rightarrow a_k\lambda$. De aquí que, por la proposición, $E \Rightarrow^* \beta A_i$, y $A_i \mapsto a_k \in \mathscr{R}$. Se concluye que, $E \Rightarrow^* \beta a_k = \alpha$. ■

Ahora pasemos al cuestionamiento sobre si los autómatas finitos indeterministas son más "poderosos" que sus contrapartes deterministas. El término "poderoso" se explicita con la pregunta: Al usar el artificio de la indeterminación, ¿las máquinas concebidas podrán aceptar palabras que de otra forma sería imposible?

> **Definición.** Dos autómatas $\mathscr{M}, \mathscr{M}'$ son *equivalentes* si y sólo si $\mathscr{L}(\mathscr{M}) = \mathscr{L}(\mathscr{M}')$.

Así, los autómatas correspondientes a las Figuras 9.7 y 9.10 son equivalentes.

Por su propia definición, todo autómata finito determinista es indeterminista, pues $\delta := \Delta$, si la relación de transición de estados Δ es la gráfica de una función. El recíproco de este resultado trivial también es cierto.

Teorema 9.12. *Para todo autómata finito indeterminista $\mathscr{M}$, existe uno determinista $\mathscr{M}'$ equivalente.*

Demostración.

Sea $\mathscr{M} = \langle I, Q, q_o, \Delta, F\rangle$ un autómata finito indeterminista. A fin de obtener un autómata finito determinista $\mathscr{M}'$ equivalente a $\mathscr{M}$, debemos empezar por convertir las transformaciones múltiples $(q, \alpha, s) \in \Delta$ con $|\alpha| > 1$, en simples $(q, a, s) \in \Delta$, donde $a \in I$: gráficamente, los arcos de Δ etiquetados con $\alpha \in I^*$ en arcos etiquetados con $a \in I$. Para esto, basta introducir estados adicionales cuyos arcos se etiqueten con los símbolos que constituyen α, *i.e.*, si

$\alpha = a_1 a_2 \ldots a_k$, entonces agregamos los estados nuevos $q_1, \ldots, q_{k-1} \in Q \setminus F$ (no finales), de forma tal que $(q, a_1 \ldots a_k, s) \in \Delta$ se traduce en la sucesión de tranformaciones $(q, a_1, q_1), (q_1, a_2, q_2), \ldots, (q_{k-1}, a_k, s)$ en Δ. Es obvio que esta modificación no afecta al conjunto de palabras aceptadas por el autómata. Por lo tanto, renombremos con $\mathcal{M}$ al autómata así obtenido.

Ahora procederemos a dar cuenta del caso cuando $\alpha = \Lambda$ ("saltos entre estados sin lectura"). Primero, debemos hacer notar que un autómata finito indeterminista puede concebirse como un dispositivo que se halla en cada momento no en único estado, sino en un conjunto de estados. Así, la construcción de $\mathcal{M}'$ comienza por definir su conjunto de estados Q' como $\mathcal{P}(Q)$, *i.e.*, el conjunto potencia de los estados de $\mathcal{M}$. Ahora, la función de transición de estados δ' de $\mathcal{M}'$, se define mediante *simulación, i.e.*, $\mathcal{M}'$ debe "imitar" el comportamiento de $\mathcal{M}$ de tal forma que todo movimiento de $\mathcal{M}'$ ante la lectura de un símbolo $a \in I$ equivalga al movimiento realizado por $\mathcal{M}$ ante el símbolo a, seguido de todos los posibles "saltos entre estados sin lectura" de $\mathcal{M}$. Formalmente, tenemos,

Definición. Sea $q \in Q$. El conjunto de todos los *estados alcanzables* desde q, mediante "saltos entre estados sin lectura", se define como[5]

$$E(q) = \{s \in Q : q \vdash\!\!\!\Rightarrow_M^* s\}$$

Debido a que estas transformaciones se realizan sin "alimentar" a $\mathcal{M}$, el proceso no depende de la palabra que se lea. Así, sea $\alpha \in I^*$ arbitraria, entonces

$$E(q) = \{s \in Q : (q, \alpha) \vdash\!\!\!\Rightarrow_M^* (s, \alpha)\}$$

De esta forma, el autómata finito determinista $\mathcal{M}' = \langle I, Q', q_o, \delta', F' \rangle$ obtenido a partir de $\mathcal{M}$ está dado por:

$$\begin{aligned} Q' &= \mathcal{P}(Q), \\ q_o' &= E(q_o), \\ F' &= \{K \subseteq Q : K \cap F \neq \varnothing\} \end{aligned}$$

y donde δ' se define para cada $K \subseteq Q$ y $a \in I$, como

$$\delta'(K, a) = \bigcup_q \{E(q) : q \in Q \text{ y } (s, a, q) \in \Delta \text{ para algún } s \in K\}.$$

[5] Aquí sí se requiere indicar qué máquina es la que realiza las transformaciones: $\vdash\!\!\!\Rightarrow_M^*$.

El determinismo de $\mathscr{M}'$ se sigue del hecho de que δ' es una función.

Ahora sólo resta probar que $\mathscr{M}$ y $\mathscr{M}'$ son equivalentes, *i.e.*, $\mathscr{L}(\mathscr{M}) = \mathscr{L}(\mathscr{M}')$. Pero para esto, necesitamos de la siguiente afirmación. (Ejercicio 4.)

Afirmación. *Sean* $\alpha \in I^*$ y $s, q \in K$. *Entonces, para algún conjunto* S *que contenga* s

$$q \models\!\!\Rightarrow^*_M \alpha p \text{ si y sólo si } E(q) \models\!\!\Rightarrow^*_{M'} \alpha S$$

Así, sea $\alpha \in I^*$. Entonces, $\alpha \in \mathscr{L}(\mathscr{M})$ si y sólo si existe $q_f \in F$ tal que $q_o \models\!\!\Rightarrow^*_M \alpha q_f$ (por definición) lo cual, por la afirmación, equivale a que $E(q_o) \Rightarrow^*_{M'} \alpha S$, para algún S que contenga a q_f, *i.e.*, $q'_o \models\!\!\Rightarrow^*_{M'} \alpha S$, para algún $S \in F'$; en otros términos, $\alpha \in \mathscr{L}(\mathscr{M}')$. ■

Corolario 9.13 (Teorema de Kleene). *Un conjunto es regular si y sólo si es el lenguaje aceptado por un autómata finito determinista.* ■

Cabe recalcar que el resultado anterior fue probado por Kleene sin el auxilio de gramáticas lineales ni el artificio del indeterminismo. La interrelación de estos últimos con los autómatas y sus lenguajes es posterior. De esta manera, el algoritmo contenido en la demostración del Teorema 9.4 es parte del resultado original, y nos permite obtener: i) ya sea como un conjunto (o su expresión) regular al lenguaje aceptado por un autómata finito (determinista) dado; o bien, ii) el autómata finito (determinista) correspondiente a un conjunto (o su expresión) regular. Ahora bien, como ya mencionamos anteriormente, suele resultar más conveniente utilizar autómatas indeterministas en los pasos intermedios a la construcción de un autómata finito determinista.

Ejemplos:

1. Consideremos el autómata finito $\mathscr{M} = \langle \{a, b\}, \{q_o, q_1\}, q_o, \delta, \{q_1\} \rangle$, donde δ está definida por el diagrama de transición de estados siguiente:

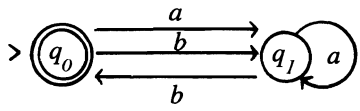


Figura 9.11

Con base en el algoritmo del Teorema 9.4, tenemos que $\mathscr{L}(\mathscr{M}) = R_{11}^3$. Ahora, por la ecuación (2):

$$R_{11}^3 = R_{11}^2 \cup R_{12}^2 (R_{22}^2)^* R_{21}^2$$

Requerimos así de tres conjuntos con $k = 2$

$$\begin{aligned} R_{11}^2 &= R_{11}^1 \cup R_{11}^1 (R_{11}^1)^* R_{11}^1 = R_{11}^1 \\ R_{12}^2 &= R_{12}^1 \cup R_{11}^1 (R_{11}^1)^* R_{12}^1 \\ R_{22}^2 &= R_{22}^1 \cup R_{21}^1 (R_{11}^1)^* R_{12}^1 \\ R_{21}^2 &= R_{21}^1 \cup R_{21}^1 (R_{11}^1)^* R_{11}^1. \end{aligned}$$

Por último, obtenemos cuatro conjuntos con $k = 1$, a partir de la ecuación (1):

$$\begin{aligned} R_{11}^1 &= \varnothing \equiv \emptyset \\ R_{12}^1 &= \{a, b\} \equiv \mathbf{a + b} \\ R_{22}^1 &= \{a\} \equiv \mathbf{a} \\ R_{21}^1 &= \{b\} \equiv \mathbf{b}. \end{aligned}$$

Sustituyendo estas expresiones en las anteriores, obtenemos:

$$\begin{aligned} R_{11}^2 &\equiv \emptyset \\ R_{12}^2 &\equiv \mathbf{(a + b) + \emptyset\emptyset^*(a + b)} \simeq \mathbf{a + b} \\ R_{22}^2 &\equiv \mathbf{a + b\emptyset^*(a + b)} \simeq \mathbf{a + b(a + b)} \\ R_{21}^2 &\equiv \mathbf{b + b\emptyset^*\emptyset} \simeq \mathbf{b}. \end{aligned}$$

Por tanto,

$$\mathscr{L}(\mathscr{M}) = R_{11}^3 \equiv \emptyset + \mathbf{(a + b)(a + b(a + b)^*b} \simeq \mathbf{(a + b)(a + b(a + b))^*b}. \quad \square$$

2. Consideremos la expresión regular:

$$\mathbf{(a + b)^*(aa + b)(a + b)^*}$$

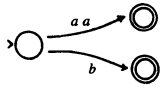
la cual describe al conjunto de todas las palabras de $\{a, b\}$ que contienen dos a's consecutivas o bien una b. Primero construiremos un autómata finito indeterminista que acepte este conjunto. Representaremos gráficamente las etapas de este proceso.

Paso 1.

$aa \mid b$

Paso 2.

$(aa + b)$



Paso 3.

$(a+b)^*$

Paso 4.

$(a+b)^*(aa+b)$

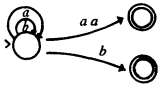


Figura 9.12

Paso 5.

$(a+b)^*(aa+b)(a+b)^*$

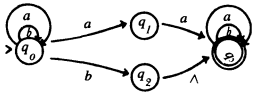


Figura 9.13

Siendo este último grafo el diagrama que representa a la relación de transición de estados Δ del autómata finito indeterminista $\mathscr{M}$.

Ahora construiremos el correspondiente autómata finito determinista, con base en el procedimiento expuesto en la demostración del Teorema 9.12. Debido a que $\mathscr{M}$ tiene 4 estados, $\mathscr{M}'$ deberá tener 16. Sin embargo, sólo los estados que son alcanzables desde q'_o son los que resultan relevantes.

Del diagrama, tenemos que $E(q_o) = \{q_o\}$, $E(q_1) = \{q_1\}$, $E(q_2) = \{q_2, q_3\}$, y $E(q_3) = \{q_3\}$.

Ahora, ya que $q'_o = E(q_o) = \{q_o\}$, las transformaciones (q_o, a, q_o) y (q_o, a, q_1) son todas las posibles de la forma (q, a, s) con $q_o \in q'_o$. De aquí se sigue que

$$\delta'(q'_o, a) = E(q_o) \cup E(q_1) = \{q_o, q_1\}.$$

Análogamente, (q_o, b, q_o) y (q_o, a, q_2) son todas las posibles de la forma (q, b, s) con $q \in q'_o$; de donde,

$$\delta'(q'_o, a) = E(q_o) \cup E(q_2) = \{q_o, q_2, q_3\}.$$

Repitiendo este cálculo con los estados recién creados, obtenemos

$$\begin{aligned}
\delta'(\{q_o, q_1\}, a) &= \{q_o, q_1, q_3\} & \delta'(\{q_o, q_1, q_3\}, b) &= \{q_o, q_2, q_3\} \\
\delta'(\{q_o, q_1\}, b) &= \{q_o, q_2, q_3\} & \delta'(\{q_o, q_2, q_3\}, a) &= \{q_o, q_1, q_3\} \\
\delta'(\{q_o, q_1, q_3\}, a) &= \{q_o, q_1, q_3\} & \delta'(\{q_o, q_2, q_3\}, b) &= \{q_o, q_2, q_3\}
\end{aligned}$$

Denotemos a los nuevos estados $q'_1 = \{q_o, q_1\}$, $q'_2 = \{q_o, q_2, q_3\}$ y $q'_3 = \{q_o, q_1, q_3\}$. De estos estados, $q'_2, q'_3 \in F'$, pues q_3 es el único elemento de F. De esta manera, el autómata finito determinista deseado tiene por diagrama de transición de estados el siguiente

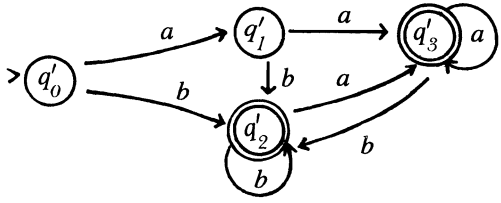


Figura 9.14

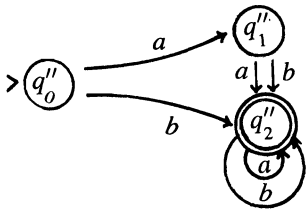


Figura 9.15

Este autómata puede simplificarse a sólo tres estados tal como se muestra en el diagrama 9.15.

Puede tomar un poco de tiempo en convencerse que este autómata en efecto acepta lenguaje representado por **(a + b)*(aa + b)(a + bb)***, mientras que con el diagrama de la Figura 9.13, *i.e.* la versión indeterminista, esto es obvio. □

El hecho de que los conjuntos regulares sean los lenguajes aceptados por los autómatas finitos, permite obtener pruebas más sencillas para algunas propiedades de estos conjuntos.

Teorema 9.14. *Si $\mathscr{L} \subseteq \mathscr{A}^*$ es un conjunto regular, entonces $\mathscr{A}^* \setminus \mathscr{L}$ es también regular.*

Demostración.

Sea $\mathscr{M} = \langle \mathscr{A}, Q, q_o, \delta, F \rangle$ una autómata finito determinista que acepta a $\mathscr{L}$, *i.e.*, $\mathscr{L} = \mathscr{L}(\mathscr{M})$. Entonces, el conjunto complemento $\mathscr{A}^* \setminus \mathscr{L}$ es aceptado por el autómata finito $\mathscr{M}^c = \langle \mathscr{A}, Q, q_o, \delta, Q \setminus F \rangle$, o sea, la única diferencia entre $\mathscr{M}^c$ y $\mathscr{M}$ radica en que se han intercambiado los estados finales con los no finales. ■

Teorema 9.15. *Si $\mathscr{L}_1$ y $\mathscr{L}_2$ son regulares, entonces $\mathscr{L}_1 \cap \mathscr{L}_2$ también lo es.*

Demostración.

Consideremos que $\mathscr{L}_1$, $\mathscr{L}_2 \subseteq \mathscr{A}^*$ son lenguajes regulares aceptados por los autómatas finitos $\mathscr{M}_1$ y $\mathscr{M}_2$ respectivamente, entonces por una identidad de conjuntos (De Morgan), tenemos que:

$$\mathscr{L}_1 \cap \mathscr{L}_2 = \mathscr{A}^* \setminus ((\mathscr{A}^* \setminus \mathscr{L}_1) \cup (\mathscr{A}^* \setminus \mathscr{L}_2))$$

de donde, $\mathscr{L}_1 \cap \mathscr{L}_2$ es regular en virtud del teorema anterior y de, por definición, la unión de conjuntos regulares es regular. ■

A continuación daremos respuesta a algunas cuestiones importantes relacionados con los autómatas finitos.

Proposición 9.16 [RS]. *Sea $\mathscr{M}$ un autómata finito determinista con n estados. Entonces $\mathscr{L}(\mathscr{M}) \neq \varnothing$ si y sólo si existe una palabra $\alpha \in \mathscr{L}(\mathscr{M})$ tal que $|\alpha| < n$.*

Demostración.

Sólo se requiere dar la prueba en un sentido.

$\Leftarrow$) Supongamos que $\mathscr{L}(\mathscr{M}) \neq \varnothing$ y que $\alpha \in \mathscr{L}(\mathscr{M})$ de longitud mínima, digamos r, tal que $n < r$. Se sigue entonces que existen dos números naturales, $k < m \leq r$, tales que $\delta(q_o, {}_o\alpha_k) = \delta(q_o, {}_o\alpha_m)$, donde ${}_o\alpha_k$ y ${}_o\alpha_m$ son los segmentos iniciales de α de longitudes k y m, respectivamente. Consideremos la palabra $\alpha' = {}_o\alpha_k \cdot_m \alpha_r$, la cual es más corta que α. Así, tenemos,

$$\begin{aligned}\delta(q_o, \alpha') = \delta(q_o, {}_o\alpha_{km}\alpha_r) &= \delta(\delta(q_o, {}_o\alpha_k), {}_m\alpha_r) \\ &= \delta(\delta(q_o, {}_o\alpha_m), {}_m\alpha_r) \\ &= \delta(q_o, {}_o\alpha_{mm}\alpha_r) \\ &= \delta(q_o, \alpha)\end{aligned}$$

pues, $\alpha = {}_o\alpha_m \cdot_m \alpha_r$. Por consiguiente, $\alpha' \in \mathscr{L}(\mathscr{M})$ también, y es más corta que α, lo cual es una contradicción. ■

Las propiedades de cerradura con las operaciones de conjuntos, de los lenguajes regulares, nos auxilian para responder a este tipo de preguntas.

Proposición 9.17. *Sean $\mathscr{M}_1$ y $\mathscr{M}_2$ autómatas finitos deterministas. Entonces, podemos dar respuesta a si:*

a. $\mathscr{L}(\mathscr{M}_1) = I^*$;

b. $\mathscr{L}(\mathscr{M}_1) = \mathscr{L}(\mathscr{M}_2)$.

Demostración.

(a) Simplemente se construye un autómata finito $\mathscr{M}'$ tal que $\mathscr{L}(\mathscr{M}') = I^* \setminus \mathscr{L}(\mathscr{M}_1)$, y se aplica la proposición anterior a si $\mathscr{L}(\mathscr{M}') = \varnothing$.

(b) Hagamos $\mathscr{L}_1 = \mathscr{L}(\mathscr{M}_1)$ y $\mathscr{L}_2 = \mathscr{L}(\mathscr{M}_2)$ tales que $\mathscr{L}_1, \mathscr{L}_2 \subseteq I^*$. Entonces el lenguaje

$$\mathscr{L}_3 = (\mathscr{L}_1 \cap (I^* \setminus \mathscr{L}_2)) \cup ((I \setminus \mathscr{L}_1) \cap \mathscr{L}_2)$$

es también regular (Proposiciones 9.14-9.15), por lo que podemos hallar un autómata finito determinista que lo acepte. Por propiedades de conjuntos, se sigue que $\mathscr{L}_3 = \varnothing$ si y sólo si $\mathscr{L}_1 = \mathscr{L}_2$; por lo que podemos aplicar la Proposición 9.16. ■

Ejercicios

1. Pruebe el Lema 9.5.
2. Halle un autómata finito indeterminista que acepte el lenguaje generado por la gramática lineal $\mathscr{G} = \langle \{a, b\}, \{E, A, B\}, E, \mathscr{R} \rangle$, donde $\mathscr{R}$ viene dado por

$$\mathscr{R} = \left\{ \begin{array}{ll} E \mapsto aA, & B \mapsto bA, \\ E \mapsto bB, & A \mapsto bA, \\ B \mapsto aB, & A \mapsto b \end{array} \right\}$$

3. Pruebe la Proposición 9.10.
4. Pruebe la afirmación siguiente:
   Sean $\alpha \in I^*$ y $s, q \in K$. Entonces, para algún conjunto S que contenga s

$$q \models\!\Rightarrow^*_M \alpha p \text{ si y sólo si } E(q) \Rightarrow^*_{M'} \alpha S.$$

5. Describa formalmente el conjunto de palabras aceptado por el autómata finito cuyo diagrama de transición de estados es el siguiente

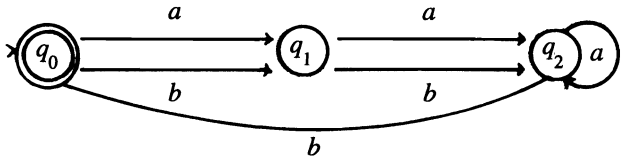


Figura 9.16

6. Para cada una de las expresiones regulares siguientes, halle autómatas finitos deterministas que acepten los conjuntos representados por éstas:

a. $a^*b + b$, **b.** $(a + b)^*bbb(a + b)^*$, **c.** $(a^* + b^*) + aba(a + b^*)$

Capítulo 10
Máquinas de Turing

Me propongo a considerar la pregunta '¿Pueden pensar las máquinas?'.

Alan Turing

10.1 Introducción

La situación en 1935 respecto de lo que intuitivamente se entendía por una función "calculable" era la siguiente:

Church y Kleene, de 1932 a 1935, consideraron una clase de funciones precisamente definida, las llamadas *funciones λ-definibles,* y encontraron que tenía propiedades que sugerían que las funciones λ-definibles abarcaban a todas las funciones que eran calculables, según la noción intuitiva y vaga que se tenía de *calculabilidad.* Otra clase de funciones calculables, llamadas *funciones generales recursivas*, definida por Gödel en 1934, tenía propiedades similares. Church y Kleene demostraron, en 1936, que estas dos clases de funciones son la misma, esto es, que toda función λ-definible es recursiva y viceversa. Bajo estas circunstancias, Church propuso *la tesis* de que todas las funciones que son intuitivamente calculables son λ-definibles, o, equivalentemente, generales recursivas. Esta es una tesis, no un teorema, pues propone identificar un concepto vago con un concepto formulado matemáticamente de manera precisa, y por tanto no se puede demostrar.

Un poco más tarde el matemático inglés Turing definió otra clase de funciones intuitivamente calculables, las *funciones Turing computables,* que son las funciones computables por medio de las llamadas *máquinas de Turing.* La misma tesis

se propuso con respecto a esta nueva clase de funciones, y se conoce como la *tesis de Turing.*

En 1937 Turing demostró que las funciones Turing computables son precisamente las λ-definibles, y consecuentemente, las generales recursivas. Así que las tesis de Church y de Turing son equivalentes.

El concepto de máquina de Turing surgió de analizar los procesos computacionales como los conocemos intuitivamente y descomponerlos en operaciones elementales. Turing argumentaba que cualquier computación posible podría ser llevada a cabo por medio de repeticiones de estas operaciones elementales. Por esta razón, la computabilidad de Turing sugiere la tesis de Church más fuertemente que las otras versiones. Turing describió una especie de computadora teórica que difiere de las computadoras humanas o digitales es dos aspectos:

1. Una máquina de Turing no puede cometer errores, es decir, obedece las instrucciones que se le dan sin desviarse.

2. Una máquina de Turing tiene una memoria potencialmente infinita, es decir, aunque la cantidad de información que guarda en cualquier momento es finita, no hay una cota superior para esta cantidad.

En estos dos aspectos se idealiza a las computadoras humanas y mecánicas quitándoles sus limitaciones prácticas.

10.2 Definición de una máquina de Turing

Las máquinas de Turing pueden ser descritas de la siguiente manera:

Hay una cinta, potencialmente infinita en ambas direciones, dividida en casilleros.

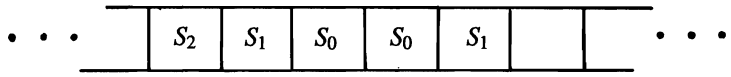


Figura 10.1

La cinta es potencialmente infinita en el siguiente sentido: aunque en cualquier momento su longitud es finita, siempre se pueden agregar casilleros adicionales a ambos lados.

Hay un conjunto finito de *símbolos de cinta* $\{S_0, S_1, \ldots, S_n\}$, llamado el *alfabeto de la máquina.* En cualquier momento, cada casillero de la cinta está ocupado por un símbolo a lo más. La máquina tiene también un conjunto finito de *estados internos* $\{q_o, q_1, \ldots, q_m\}$. En cualquier momento, la máquina está exactamente en alguno de estos estados. Finalmente, hay una cabeza lectora-escritora que está sobre algún casillero de la cinta en cualquier momento dado.

Si en algún momento t la cabeza lectora-escritora está sobre un casillero que tiene al símbolo S_i y la máquina está en el estado interno q_j, la acción de la máquina está determinada y puede hacer alguna de las siguientes cosas:

1. Puede borrar el símbolo S_i e imprimir S_k;
2. Puede moverse a la derecha;
3. Puede moverse a la izquierda;
4. Puede parar.

En los casos (1)–(3), la máquina entra en un nuevo estado interno q_r y está lista para operar otra vez en el momento $t + 1$.

Vamos a suponer que el símbolo S_o representa un espacio en blanco de tal modo que la cabeza lectora-escritora siempre está leyendo un símbolo. Las primeras tres acciones de la máquina pueden representarse por cuádruplas de la siguiente manera:

$$(1)\ q_j S_i S_k q_r, \qquad (2)\ q_j S_i D q_r, \qquad (3)\ q_j S_i I q_r$$

El primer símbolo representa el estado interno de la máquina al iniciar la acción, el segundo símbolo representa el símbolo del alfabeto que se está leyendo, el tercer símbolo representa la acción de la máquina (imprimir S_k, moverse a la derecha, moverse a la izquierda) y el cuarto símbolo representa el estado interno de la máquina cuando ya se realizó la acción.

Dada una máquina de Turing (MT), le podemos asociar el siguiente algoritmo en el alfabeto de MT. Llamamos $\mathscr{A}$ al alfabeto de MT y sea α una palabra ($\alpha \in \mathscr{A}^*$) de MT. Imprímase α de izquierda a derecha en una cinta vacía. Póngase a ésta en la máquina, con la cabeza lectora-escritora sobre el casillero más a la izquierda. Inicie la máquina en el estado inicial q_o. Si la máquina para en algún momento, la palabra de $\mathscr{A}$ que aparece en la cinta es el valor de α bajo el algoritmo. Este algoritmo que acabamos de describir es lo que se conoce como un *algoritmo de Turing.*

Nota: La palabra de $\mathscr{A}$ que aparece en la cinta se define como la sucesión de símbolos que comienza con el símbolo más a la izquierda y continúa hacia la derecha hasta llegar al símbolo más a la derecha. Recuerde que cuando se encuentre un casillero en blanco al realizar este movimiento se supone al símbolo S_o impreso.

Nótese que no hemos definido un mecanismo por medio del cual la máquina "sepa" cuando parar. La idea intuitiva es que la máquina para cuando no tiene instrucciones que le digan cómo seguir, por ejemplo, cuando está leyendo un símbolo S_i en un estado q_j y ninguna cuádrupla comienza con $q_j S_i$.

A continuación hacemos esto más preciso dando una definición más rigurosa de máquina de Turing.

De lo dicho anteriormente, vemos que una MT queda determinada de manera precisa por un conjunto finito de cuádruplas[1] de los siguientes tres tipos:

$$(1)\ q_j S_i S_k q, \qquad (2)\ q_j S_i D q, \qquad (3)\ q_j S_i I q_r \tag{1}$$

de tal forma que no hay dos cuádruplas distintas que coincidan en los primeros dos símbolos. Así es como vamos a definir formalmente una MT.

Sea $\mathscr{A} = \{S_o, S_1, \ldots, S_n\}$ un conjunto de símbolos de cinta y $\{q_o, q_1, \ldots, q_m\}$ un conjunto de símbolos (que representarán estados internos).

Definición. Una *máquina de Turing* $\mathscr{M}$ con alfabeto $\mathscr{A}$ es un conjunto de cuádruplas de los siguientes tres tipos:

$$(1)\ q_j S_i S_k q_r, \qquad (2)\ q_j S_i D q_r, \qquad (3)\ q_j S_i I q_r$$

tal que no hay dos cuádruplas distintas que coincidan en los primeros dos símbolos.

Se supone que q_o es un estado interno de cualquier máquina de Turing.

[1] A veces se definen las MT como conjuntos de quíntuplas de las formas $q_j S_i S_k D q_r$, $q_j S_i S_k I q_r$, $q_j S_i S_k F q_r$, donde se están efectuando dos operaciones: la sustitución de S_i por S_k y el movimiento sobre la cinta (F indica que no hay movimiento, la cabeza se queda fija). Claramente las dos formulaciones son equivalentes.

Definición. Sea $\mathcal{M}$ una máquina de Turing. Una *configuración* es una sucesión finita de símbolos α tal que:

1. Todos los símbolos de α, con excepción de uno, son símbolos de cinta;
2. El único símbolo de α que no es de cinta es un estado interno q_s;
3. q_s no es el último símbolo de α.

Observación. Una configuración especifica la condición de la máquina y de la cinta en un momento dado. Si se leen de izquierda a derecha, los símbolos de cinta de α representan los símbolos impresos en la cinta en ese momento. El estado interno q_s representa el estado interno de la máquina en ese momento y el símbolo de cinta que aparece inmediatamente a la derecha de q_s representa al símbolo que $\mathcal{M}$ está leyendo en ese momento.

Definición. Sean α y β dos configuraciones de la máquina de Turing $\mathcal{M}$. Decimos que $\mathcal{M}$ *transforma* o *mueve* a α en β (en símbolos, $\alpha \models\!\!\Rightarrow_M \beta$) si y sólo si ocurre alguna de las siguientes condiciones:

(a) α es de la forma Pq_jS_iQ, β es de la forma Pq_rS_kQ y $q_jS_iS_kq_r$ es una de las cuádruplas de $\mathcal{M}$; o

(b) α es de la forma $PS_sq_jS_iQ$, β es de la forma $Pq_rS_sS_iQ$, y $q_jS_iIq_r$ es una de las cuádruplas de $\mathcal{M}$; o

(c) α es de la forma q_jS_iQ, β es de la forma $q_rS_oS_iQ$, y $q_jS_iIq_r$ es una de las cuádruplas de $\mathcal{M}$; o

(d) α es de la forma $Pq_jS_iS_kQ$, β es de la forma $PS_iq_rS_kQ$, y $q_jS_iDq_r$ es una de las cuádruplas de $\mathcal{M}$; o

(e) α es de la forma Pq_jS_i, β es de la forma $PS_iq_rS_o$, y $q_jS_iDq_r$ es una de las cuádruplas de $\mathcal{M}$.[2]

Observación. Nótese que, según nuestra idea intuitiva, si $\mathcal{M}$ mueve a α en β y α describe la situación de $\mathcal{M}$ en un momento dado t, entonces β describe la situación de $\mathcal{M}$ en el momento $t+1$. Según las cláusulas (c) y (e), si la máquina está en algún extremo y la instrucción es moverse aún más, se le agrega a la cinta un nuevo casillero en blanco, representado por el símbolo S_o.

[2] P y Q representan sucesiones finitas de símbolos de cinta.

Definición. Sea α una configuración de la máquina $\mathscr{M}$. Decimos que $\mathscr{M}$ *para* en α si y sólo si no existe configuración β tal que $\alpha \models\!\!\Rightarrow_M \beta$.

Nota. $\mathscr{M}$ para en α si $q_j S_i$ ocurre en α pero ninguna cuádrupla de $\mathscr{M}$ tiene a $q_j S_i$ como sus primeros dos símbolos.

Definición. Una *computación* de una máquina de Turing es una sucesión finita de configuraciones $\alpha_o, \ldots, \alpha_m$ $(m \geq 0)$ tal que el estado interno que ocurre en α_o es q_o; para cada $i = 1, \ldots, m$, $\alpha_i \models\!\!\Rightarrow_M \alpha_{i+1}$ y $\mathscr{M}$ para en α_m. Esta computación se dice que *empieza* en α_o y *termina* en α_m.

Si se tiene un alfabeto $\mathscr{C}$ que contiene a $\mathscr{A}$, el alfabeto de la máquina $\mathscr{M}$, podemos definir un algoritmo, denotado por $\mathfrak{B}_{M,C}$, de la siguiente manera: si P y Q son dos palabras de $\mathscr{A}$, $\mathfrak{B}_{M,C}(P) = Q$ si y sólo si hay una computación de $\mathscr{M}$ que empieza con la configuración $q_o P$ y termina con una configuración de la forma $R_1 q_j R_2$, con $Q = R_1 R_2$.

Definición. Un algoritmo $\mathfrak{A}$ en un alfabeto $\mathscr{D}$ es *Turing-computable* si y sólo si existe una máquina de Turing $\mathscr{M}$ con alfabeto $\mathscr{A}$ y un alfabeto $\mathscr{C}$ que contiene a $\mathscr{A} \cup \mathscr{D}$ tal que $\mathfrak{A}$ y $\mathfrak{B}_{M,C}$ son equivalentes con respecto a $\mathscr{D}$, esto es, dadas P y Q palabras de $\mathscr{D}$, $\mathfrak{A}(P) = Q$ si y sólo si $\mathfrak{B}_{M,C}(P) = Q$.

En el contexto de funciones aritméticas, podemos usar esto para dar una definición de lo que quiere decir que una función sea Turing-computable.

Para denotar a los números naturales sólo necesitamos el símbolo 1, ya que el número m puede ser denotado por $1 \ldots 1$, $m+1$ veces, llamemos $\overline{m}$ a esta sucesión de unos.

Definición. Sea $\mathscr{A}$ un alfabeto que contiene a $\{1, \circ\}$, donde $\circ$ es la concatenación. (Suponemos pues, que 1 es S_1 y $\circ$ es S_2.) Sea $\mathscr{M}$ una máquina de Turing con alfabeto $\mathscr{A}$ y $f(x_1, \ldots, x_n)$ una función definida en un subconjunto de $\mathbb{N}$. Entonces decimos que $\mathscr{M}$ *computa* a f si y sólo si dados $k_1, \ldots, k_n \in \mathbb{N}$ y cualquier palabra de $\mathscr{A}$, $\mathfrak{B}_{M,A}(\overline{k_1} \circ \ldots \circ \overline{k_n}) = Q$ si y sólo si Q es $R_1 \overline{f(k_1, \ldots, k_n)} R_2$, donde R_1 y R_2 son palabras (posiblemente vacías) que consisten sólo de S_o's.

Aquí se permite que el resultado sea de la forma $R_1 \overline{f(k_1, \ldots, k_n)} R_2$ porque el símbolo S_o se interpreta como un espacio en blanco.

Definición. Una función f es *Turing-computable* si y sólo si existe una máquina de Turing que la computa.

La tesis de Church-Turing afirma que toda función calculable es Turing-computable.

Ejemplos:

1. La función *sucesor* es Turing-computable. Considérese la máquina de Turing $\mathscr{M}$ definida por las siguientes cuádruplas:

$$q_o 1 I q_1 \quad \text{y} \quad q_1 S_o 1 q_2$$

El alfabeto de $\mathscr{M}$ es $\mathscr{A} = \{1, S_o\}$. $\mathscr{M}$ transforma cualquier palabra que no empiece con 1 en sí misma. Cualquier palabra que empiece con 1 es de la forma $1P$, donde P es cualquier palabra del alfabeto, y la acción de $\mathscr{M}$ sobre $1P$ es la siguiente:

$$q_o 1P \models_M q_1 S_o 1P \models_M q_2 11P$$

En general, sea $k \in \mathbb{N}$ arbitrario, entonces $\overline{k}$ es una sucesión de k 1's y tenemos

$$q_o \overline{k} \models_M q_1 S_o \overline{k} \models q_2 \overline{k+1}$$

Por tanto $\mathfrak{B}_{M,A}(\overline{k}) = \overline{k+1}$.

2. Sea $\mathscr{M}$ la máquina de Turing definida por las siguientes cuádruplas:

$$q_o 1 I q_1 \quad \text{y} \quad q S_o 1 q_o$$

Si $\mathscr{M}$ empieza en una palabra cuyo primer símbolo no es 1, se para. Si el primer símbolo es 1, la máquina agrega 1's a la izquierda y nunca para.

3. Sea $\mathscr{M}$ la máquina de Turing dada por las siguientes cuádruplas:

$$q_o S_o D q_o, \quad q_o S_2 D q_o, \quad q_o S_3 D q_o, \quad \ldots, \quad q_o S_k D q_o, \quad q_o 11 q_1$$

El alfabeto de $\mathscr{M}$ es $\{S_o, S_1, \ldots, S_k\}$, donde S_o es un espacio en blanco y S_1 el 1. La máquina $\mathscr{M}$ se mueve a la derecha hasta que encuentra un 1, y entonces para.

4. La suma es Turing-computable.

Sea $\mathscr{M}$ la máquina con alfabeto $\{S_o, 1, \circ\}$, definida por las siguientes cuádruplas:

$$q_o 1 S_o q_o,\ q_o S_o D q_1,\quad q_1 1 D q_1,\ q_1 \circ 1 q_2,\ q_2 1 D q_2,\ q_2 S_o I q_3,\ q_3 1 S_o q_3$$

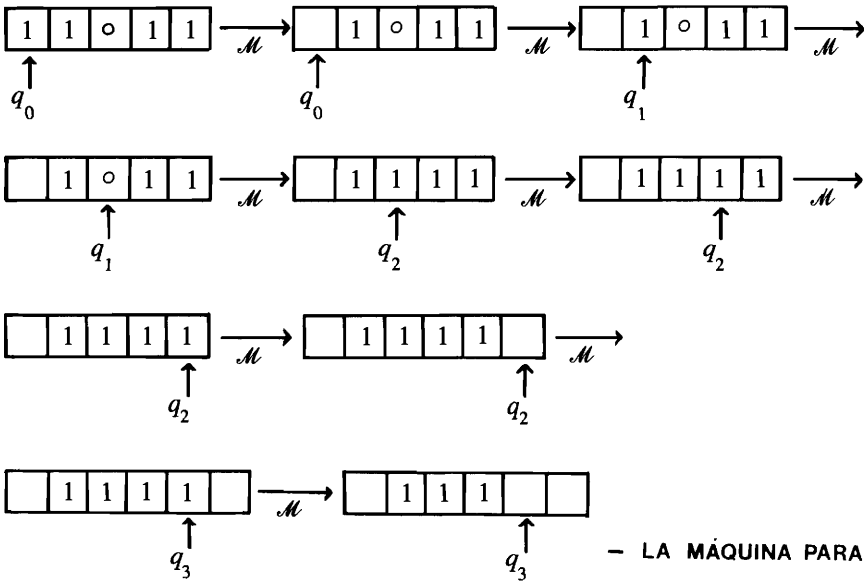
Antes de demostrar en general que en efecto esta máquina computa la suma, veamos un ejemplo sencillo. Verifiquemos que $1 + 1 = 2$, en el lenguaje de $\mathscr{M}$, esto equivale a ver que si "arrancamos" a $\mathscr{M}$ con la palabra $11 \circ 11$ llegamos a 111. (Recuérdese que $\overline{m} = 11\ldots1$, $m + 1$ veces).

Veamos entonces el proceso que sigue si inicia con $11 \circ 11$ (Fig. 10.2).

Hemos visto, con este ejemplo, que $\mathfrak{B}_{M,A}(\overline{1} \circ \overline{1}) = S_o \overline{1+1} S_o S_o$.

La demostración en general es exactamente la misma; para simplificar la notación, denotemos por 1^{m+1} a la sucesión $11\ldots1$, $m+1$ veces, *i.e.*, $\overline{m} = 1^{m+1}$. Sean m y n dos números naturales arbitrarios. Entonces,

$$\begin{aligned}
q_o \overline{m} \circ \overline{n} = q_o 1^{m+1} \circ 1^{n+1} &\models_M q_o S_o 1^m \circ 1^{n+1} \models_M S_o q_1 1^m \circ 1^{n+1} \\
&\models_M S_o 1 q_1 1^{m-1} \circ 1^{n+1} \models_M \cdots \\
&\models_M S_o 1^m q_1 \circ 1^{n+1} \models_M S_o 1^m q_2 1 1^{n+1} \\
&\models_M S_o 1^{m+1} q_2 1^{n+1} \models_M S_o 1^{m+1} 1 q_2 1^n \models_M \cdots \\
&\models_M S_o 1^{m+1} 1^{n+1} q_2 S_o \models_M S_o 1^{m+1} 1^n q_3 1 S_o \\
&\models_M S_o 1^{m+1} 1^n q_3 S_o S_o = S_o 1^{m+n+1} q_3 S_o S_o \\
&= S_o \overline{m+n} q_3 S_o S_o. \qquad \square
\end{aligned}$$



Figuras 10.2

Ejercicios

1. Construir una máquina de Turing que compute la siguiente función:

$$f(n) = \begin{cases} n-1 & \text{si } n \neq 0 \\ 0 & \text{si } n = 0. \end{cases}$$

2. Demostrar que la función $m - n$ es Turing-computable, donde:

$$m - n = \begin{cases} m - n & \text{si } m \geq n \\ 0 & \text{si } m < n. \end{cases}$$

10.3 Matrices funcionales para máquinas de Turing

En la sección anterior describimos a las máquinas de Turing como conjuntos de cuádruplas, donde cada cuádrupla representaba la acción de la máquina al leer un símbolo en cierto estado. Las acciones que la máquina podía ejecutar eran de tres tipos: reemplazar el símbolo S_i por S_j, moverse al casillero inmediato a la derecha o moverse al casillero inmediato a la izquierda.

Podemos combinar dos acciones dando en una sola instrucción un comando para reemplazar al símbolo S_i y otro para indicar qué movimiento hace la máquina después de haber efectuado la sustitución. Para esto necesitamos un nuevo símbolo, F, para indicar que la máquina queda fija después del reemplazo.

En este contexto, una máquina de Turing es un conjunto de quíntuplas de las formas: $q_i S_m S_n D q_j$, $q_i S_m S_n I q_j$, $q_i S_m S_n F q_j$, con los siguientes significados:

$q_i S_m S_n D q_j$ significa: si la máquina $\mathcal{M}$ está leyendo el S_m símbolo en el estado q_i, debe borrar S_m, imprimir S_n y moverse un casillero a la derecha. Al terminar, $\mathcal{M}$ está en el estado q_j.

Análogamente para los otros dos tipos de quíntuplas, sólo que I significa moverse un casillero a la izquierda y F significa no hacer ningún movimiento.

Es conveniente representar a la máquina $\mathcal{M}$ por medio de una tabla rectangular con una columna para cada símbolo de estado y un renglón para cada símbolo de alfabeto, y que tiene la terna de salida en la intersección del renglón y la columna de la pareja de entrada. Esta tabla es la *matriz funcional de* $\mathcal{M}$.

Ejemplo. Supongamos que $\mathcal{M}$ tiene el alfabeto $\{S_o, S_1, S_2\}$ y los estados q_o, q_1 y q_2. La siguiente matriz funcional:

	q_o	q_1	q_2
S_o	$S_o D q_1$	$S_o D q_2$	$S_o F q_2$
S_1	$S_2 I q_1$	$S_o I q_2$	$S_1 F q_2$
S_2	$S_1 I q_1$	$S_1 F q_2$	$S_2 F q_2$

representa a la máquina de Turing definida por las siguientes quíntuplas:

$$\mathcal{M} = \left\{ \begin{array}{lll} q_o S_o S_o D q_1, & q_o S_1 S_2 I q_1, & q_o S_2 S_1 I q_1 \\ q_1 S_o S_o D q_2, & q_1 S_1 S_o I q_2, & q_1 S_2 S_1 F q_2 \\ q_2 S_o S_o F q_2, & q_2 S_1 S_1 F q_2, & q_2 S_2 S_2 F q_2 \end{array} \right\} \qquad \square$$

Es claro que la operación de una máquina de Turing queda completamente determinada por su matriz funcional, de modo que dos máquinas de Turing que tengan la misma matriz funcional trabajan de manera idéntica, y sus algoritmos asociados serán iguales.

Si se analiza la matriz funcional dada, se puede ver que si la máquina está en el estado q_2 no hace nada, deja al símbolo que está leyendo igual, permanece en la misma posición y en el mismo estado. En otras palabras, se para. Esto se puede denotar, en la matriz funcional, del modo siguiente; agregando en símbolo "!" para indicar la condición de alto:

	q_o	q_1	q_2
S_o	$S_o D q_1$	$S_o D q_2$	!
S_1	$S_2 I q_1$	$S_o I q_2$	!
S_2	$S_1 I q_1$	$S_1 F q_2$	!

Una simplificación más se puede hacer si se adopta la convención que cuando no hay reemplazo de símbolo del alfabeto, o no hay cambio en el estado de la máquina, se omitirán de la terna de salida, también se omite el símbolo F que indica que no hay movimiento de la máquina.

Con estas convenciones, la matriz funcional para $\mathcal{M}$ queda como sigue:

	q_o	q_1	q_2
S_o	$D q_1$	$D q_2$	!
S_1	$S_2 I q_1$	$S_o I q_2$	!
S_2	$S_1 I q_1$	$S_1 q_2$	!

Nota. Otra forma de denotar esta matriz, con estas convenciones sería dejar la última columna totalmente vacía, indicando que no hay cambio de símbolo ni de estado, y que no hay movimiento.

A continuación presentaremos un ejemplo de una máquina de Turing construida en forma matricial.

Ejemplo. Transformar n en $n + 1$ en notación decimal.

El lenguaje de $\mathscr{M}$ es $\{S_o, 0, 1, \ldots, 9\}$, $\mathscr{M}$ tiene sólo un estado interno: q_o. La matriz funcional es la siguiente:

	q_o	
0	1	!
1	2	!
2	3	!
3	4	!
4	5	!
5	6	!
6	7	!
7	8	!
8	9	!
9	0	I
S_o	1	!

Dado un número n, escrito en notación decimal, un dígito por cada casillero de la cinta, se coloca la cabeza lectora-escritora de la máquina el dígito de la extrema derecha. Si el dígito es menor que 9, la máquina reemplaza este dígito por el dígito siguiente y para. Si el dígito es 9, lo cambia por 0 y se mueve al casillero de la izquierda.

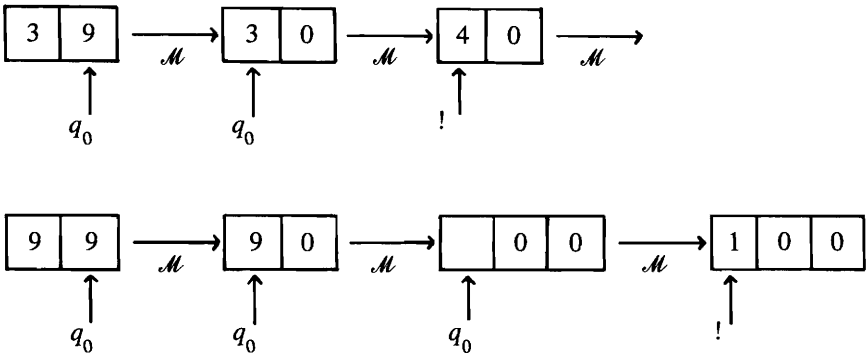
En la Fig 10.3 Vemos la acción de la máquina en dos casos particulares.

Ejercicio

Construir una máquina de Turing en forma matricial que transforme n en $n - 1$ para todo número natural $n \geq 1$.

10.4 Máquina de Turing universal

Hasta ahora hemos trabajado con la idea que cada algoritmo necesita una máquina de Turing particular, con su propia matriz funcional. Sin embargo, puede construirse una máquina de Turing universal, capaz de ejecutar, en cierto sentido, el trabajo de cualquier máquina de Turing.



Figuras 10.3

No daremos una demostración detallada de la construcción de una máquina de Turing universal, sólo vamos a dar un bosquejo general de cómo sería tal máquina.

Obsérvese que la máquina universal $\mathscr{U}$ debe tener un lenguaje finito fijo y lo que se debe lograr es que este lenguaje "contenga" todos los posibles lenguajes de máquinas de Turing particulares.

Por otro lado, $\mathscr{U}$ debe tener un mecanismo mediante el cual, dado un símbolo, sepa a qué máquina se refiere y pueda seguir las instrucciones del algoritmo de esa máquina. Esto se logra por medio de una codificación de los lenguajes de las máquinas de Turing en un único lenguaje, capaz de representar todas las matrices y configuraciones posibles.

Antes de hacerlo nótese que en un lenguaje particular, los símbolos S_o, $S_1, \ldots, S_n$ se pueden reemplazar por otros símbolos, digamos A_o, $A_1, \ldots, A_n$ sin que se altere el funcionamiento de la máquina.

En este punto, es conveniente volver a pensar en las máquinas de Turing como sucesiones finitas de quíntuplas de cierto tipo. (Esto es, debemos pensar en cada máquina como unidimensional, no como bidimensional, para poder codificarla en $\mathscr{U}$.)

El lenguaje de $\mathscr{U}$ sólo va a tener como símbolos al 0 y al 1.

Damos a continuación el código.

	Símbolo	Código o grupo cifrado	
	I	101	
	D	1001	
	F	10001	
Alfabeto	S_o	10001—4 ceros	un número par de ceros mayor que 2
	$\vdots$		
	S_k	$100\ldots001$ — $2(k+2)$ ceros	
Estados	q_o	100001—5 ceros	un número impar de ceros mayor que 3
	$\vdots$		
	q_m	$1000\ldots001$ — $2(m+2)+1$ ceros	

Observaciones:

1. Cada símbolo de algún lenguaje empieza con 1 y termina con 1.

2. Según el número de ceros entre cada par de 1's, la máquina $\mathscr{U}$ puede identificar si la sucesión de 0's y 1's representa un movimiento (I, D, F), un símbolo del alfabeto o un símbolo de estado.

3. Cada vez que se tenga una nueva máquina de Turing, siempre es posible codificarla, ya que los alfabetos y los conjuntos de estados son finitos, y siempre se pueden introducir nuevos ceros entre dos 1's para traducir nuevos símbolos.

Ejemplo Supongamos que tenemos una máquina de Turing $\mathscr{M}$, con alfabeto $\{S_o, S_1\}$ y estados $\{q_o, q_1\}$, definida por las siguientes quíntuplas:

$$q_o S_o S_1 F q_1, \qquad q_o S_1 S_1 F q_o$$

Con la codificación dada, esta máquina queda descrita por la siguiente palabra del lenguaje de $\mathscr{U}$:

1000001100001100000011000110000000110000011000000110000001100011000001

Si ahora queremos codificar, junto con ésta, otra máquina de Turing $\mathscr{S}$ con alfabeto $\{S_o, S_1\}$ y estados $\{q_o, q_1\}$, basta ahora codificar estos símbolos de

manera distinta, para evitar ambigüedades, por ejemplo:

S_o como 1000000001
S_1 como 100000000001
q_o como 10000000001
q_1 como 1000000000001. □

Estas sucesiones de 0's y 1's obtenidas se llaman *matrices funcionales cifradas* o *configuraciones cifradas*, según representen matrices funcionales o configuraciones.

Podemos ahora dar el algoritmo para $\mathcal{U}$:

Paso 1. Dada una configuración buscar el (único) grupo cifrado con un número impar (> 3) de ceros y examinar el grupo cifrado inmediatamente a la derecha.

Paso 2. Buscar en la matriz cifrada la pareja de grupos cifrados adyacentes idénticos a los mencionados en el paso 1.

Paso 3. Seguir las instrucciones dadas por la matriz encontrada en el paso 2. (Por ejemplo, si el siguiente grupo cifrado es 101, la máquina se moverá, en la configuración original, al grupo cifrado inmediatamente a la izquierda.)

Observaciones:

1. Cualquier operación realizada con grupos cifrados puede reducirse a una operación estándar de máquinas de Turing.

2. El lenguaje de $\mathcal{U}$ necesitará algunas letras más, por ejemplo, para separar la matriz funcional cifrada de la configuración cifrada, y letras que sirvan como marcadores provisionales mientras se examinan los 0's y los 1's.

Esperamos que este bosquejo haya sido suficiente para convencer al lector de que el algoritmo descrito puede ser expresado como una matriz funcional de una máquina de Turing. Para una exposición más detallada, referimos al lector a [Tu].

10.5 Una jerarquía para la complejidad computacional

Aun cuando las máquinas de Turing tienen un carácter general, existen problemas que éstas no pueden resolver, esto es, problemas *indecidibles* para la clase de las máquinas de Turing. A una de ellas puede llevarle algún tiempo en completar una computación; e incluso puede suceder que ésta nunca termine. Por lo que sería deseable disponer de un algoritmo para *decidir* si dadas una máquina $\mathcal{M}$ y una palabra P, el proceso terminará o no. Este problema es el denominado *problema de paro*. Sin embargo, se prueba que no existe un procedimiento para resolver este problema.

Para esclarecer en qué consiste la indecidibilidad del problema de paro, veamos el ejemplo siguiente.

Ejemplo. Consideremos los algoritmos:

Algoritmo $\mathfrak{A}$:
1. MIENTRAS ($n \neq 1$),
2. HAZ ($n \leftarrow n/2$);
3. REPITE.
4. FIN.

Algoritmo $\mathfrak{B}$:
1. MIENTRAS ($n \neq 1$),
2. HAZ (SI (n es par) ENTONCES (HAZ ($n \leftarrow n/2$));
3. OTRO HAZ ($n \leftarrow 3n + 1$));
4. REPITE
5. FIN.

Notación. La instrucción "MIENTRAS (*cond*) HAZ (...) REPITE", a semejanza de "PARA $i = 1 \ldots$", es empleada para realizar iteraciones. Semánticamente, lo representado con "..." se repite hasta que *cond* sea falsa, saliéndose así del ciclo. (Aquí empleamos las "instrucciones" de la secc. 6.3.)

Para el caso del algoritmo $\mathfrak{A}$, tenemos que es fácil decidir si termina o no: simplemente, cuando n es una potencia de 2 entonces para, mientras que si no lo es se producirá una interminable sucesión de números racionales que converge al

0. Ahora que para el algoritmo $\mathfrak{B}$, la situación es diferente: si n es par, el número n se reduce a la mitad, mientras que si es impar, se incrementa por un factor de tres más 1. Para todos los valores de n que han sido introducidos hasta la fecha, resulta que siempre converge a 1, pero el por qué sucede así, es algo que nadie a probado aún. (Para mayor información sobre este problema y otros afines, *cf.* [La].) En cierta forma, el comportamiento de este algoritmo semeja al del sistema formal MIU, ya presentado en la sección 3.3, donde se tienen reglas de producción que acortan y otras que alargan las longitudes de las ebf's, tornando su problema de decisión un tanto difícil, aunque no insoluble (*cf.* el Capítulo 9 de [Ho]). □

Ahora nos corresponde relacionar la problemática de la complejidad computacional abordada desde la sección 6.3, con el concepto de máquina de Turing, siendo que éstas constituyen una formalización de la noción de algoritmo.[3]

Dado que hay problemas insolubles para las MT's (aparte del problema anterior), parecería natural que con hallar un algoritmo para resolver un problema sea más que suficiente. Sin embargo, para muchos problemas, la solución puede considerarse como irrealizable o impráctica, tomando todo el tiempo que tomaría para llevar a efecto el algoritmo encontrado, *e.g.*, "las torres de Hanoi". En vista de esto, se procede a escindir las clases de los problemas "bien planteados" en insolubles *versus* solubles. A su vez, estos últimos se ven jerarquizados conforme al tiempo que toman en dar su respuesta, en las clases: $\mathscr{P}$, $\mathscr{NP}$, $\mathscr{NP}$-completos y problemas intratables.

La mencionada tesis de Church-Turing da lugar a una identificación entre las nociones de algoritmo y de máquina de Turing, de forma tal que podemos emplearlas como sinónimas. Así, podemos introducir al indeterminismo en las MT's. Ahora bien, es un hecho probado que éste no es más que un artificio técnico que no influye sobre la potencialidad de las mismas [HU]-[Re], por lo que nos es permitido usarlas indistintamente, salvo cuando se proceda a implementarlas en la "práctica", tal como se explicitará más adelante.

Se puede verificar que la contención: $\mathscr{NP}$-completos $\subset$ $\mathscr{NP}$-duros es propia. Para esto, que no todo problema de decisión $\mathscr{NP}$-duro sea $\mathscr{NP}$-completo, tenemos *el problema de paro.* Como vimos anteriormente, éste consiste en determinar para un algoritmo determinístico arbitrario $\mathfrak{A}$ y una entrada $\mathscr{I}$, si el algoritmo $\mathfrak{A}$ con entrada $\mathscr{I}$ siempre termina o entra en un ciclo infinito. Dado que este problema es indecidible, no puede estar en $\mathscr{NP}$. Sin embargo, para probar que es $\mathscr{NP}$-duro, o sea que SAT $\propto$ problema de paro, requerimos construir un algoritmo $\mathfrak{A}$ cuya

[3]El material que sigue presupone del conocimiento de la sección 6.3

entrada sea una fórmula α. Si en α ocurren n letras entonces $\mathfrak{A}$ ensaya con todas las 2^n valuaciones posibles y verifica que α es satisfacible. Así, dependiendo de si α sea satisfacible o no, el algoritmo para o entra en un ciclo infinito. De haber un algoritmo eficiente para el problema de paro entonces podríamos resolver el de satisfacibilidad eficientemente, usando $\mathfrak{A}$ y α como entrada para el algoritmo del problema de paro. Por lo tanto, el problema de paro es $\mathcal{NP}$-duro, pero no es $\mathcal{NP}$-completo.

En la Figura 10.4 damos una representación gráfica de esta jerarquía.

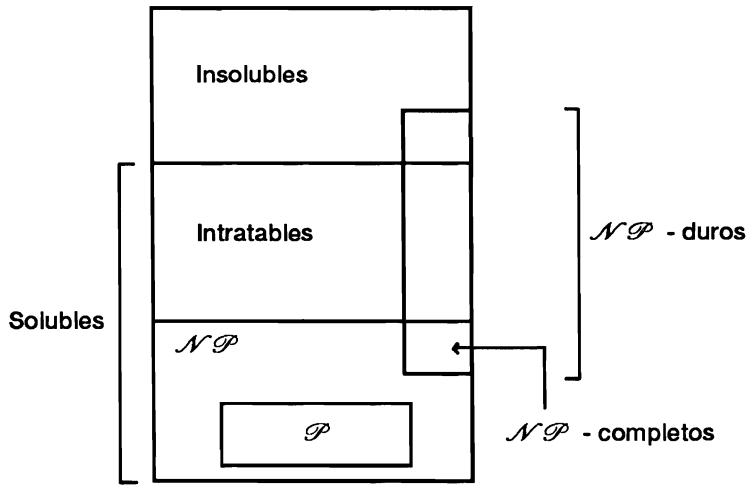


Figura 10.4

Analicemos ahora si las clases de complejidad son invariantes respecto al modelo de computadora o lenguaje de programación.

Sobre la base de la tesis de Church-Turing, tenemos que *la clase de los problemas computables* (efectiva o algorítmicamente solubles) *es robusta*, entendiéndose por esto que es invariante a cambios en el modelo de computadora o lenguaje de programación.

Aunque las máquinas de Turing son comúnmente usadas en la literatura, no son los únicos modelos de computadora (*e.g.*, *acceso aleatorio de memoria* (RAM), *la máquina de Schönhage*, etc.), pues nada nos obliga a tomar sólo modelos con cinta lineal y acceso secuencial. De aquí, y cuestionando sobre lo que constituye un

"paso" en una computación, se llega a la pregunta de ¿cuál es el modelo correcto de computadora para medir el tiempo de ejecución de un algoritmo?, o en otros términos, si es posible una teoría independiente del modelo de computadora. A pesar de que no hay una respuesta sencilla a ¿qué es un paso?, afortunadamente, muchos de los modelos no son muy diferentes en el tiempo computacional: en general, la simulación de uno en otro no toma más del cuadrado del tiempo de ejecución. De esta manera, *la clase de los problemas tratables es* también *robusta*. Sin embargo, esta versión refinada de la tesis de Church-Turing no se satisface para cualquier modelo (*e.g.*, *recurrencia*), por este motivo se le conoce como la *tesis computacional secuencial*. Bajo este formalismo, la controversia $\mathcal{P}$ *vs* $\mathcal{NP}$ toma el carácter siguiente:

"Si las máquinas indeterministas de Turing satisfacen el criterio de secuencialidad, entonces la tesis refinada implica que, en efecto, $\mathcal{P} = \mathcal{NP}$*"*.

Consecuentemente, las máquinas indeterministas de Turing no son secuenciales, usan "magia" para resolver los problemas, pues sin magia tendrían que ensayar muchas posibilidades simultáneamente para determinar la mejor solución. Una forma de describir estas máquinas (o algoritmos) es permitiendo libre *retroseguimiento* (*backtraking*). Si una máquina indeterminista de Turing $\mathcal{M}'$ alcanza un punto en el que varias elecciones son posibles, cualquiera puede tomarse, pero si el camino elegido da lugar a un "callejón sin salida", la máquina $\mathcal{M}'$ puede "saltar" hacia atrás (retroceder) al nodo previamente elegido, sin costo alguno en el tiempo computacional invertido en un camino erróneo. Simulando $\mathcal{M}'$ mediante una máquina determinística, $\mathcal{M}$, $\mathcal{M}$ debe generar cada pronóstico y usar el método de la parte determinística de $\mathcal{M}'$ para "checar" la correción del pronóstico. De esta manera, $\mathcal{M}$ requiere trazar los caminos indeterministas explícitamente (usando, *e.g.*, una pila) y emplear la técnica por retroseguimiento para tratar con las computaciones fallidas. Concretando, este método exhaustivo devendría en un proceso exponencial. Aunque sea terriblemente ineficiente, esta máquina $\mathcal{M}$ corresponde muy de cerca a los métodos usados en la práctica para resolver los problemas más difíciles en $\mathcal{NP}$, pues ¡los algoritmos indeterministas simplemente no existen!

Resumiendo, la problemática $\mathcal{P}$ *vs* $\mathcal{NP}$, bajo este contexto formal, ¡queda igual que como lo fue en un principio!, y para resolver el problema de manera negativa (*i.e.*, mostrar que $\mathcal{P} \neq \mathcal{NP}$), sólo bastaría con probar que las máquinas de Turing no pueden resolver un problema $\mathcal{NP}$-completo en un tiempo menor al exponencial, ¡tarea por demás difícil, si no imposible!

Ejercicios

1. Codifique en un lenguaje de programación los algoritmos $\mathfrak{A}$ y $\mathfrak{B}$ (en torno al problema de paro). Ensaye para varios valores de $n \in \mathbb{N}$. ¿Puede seguirse alguna pauta para las respuestas de $\mathfrak{B}$?

2. Haga un breve esbozo sobre *recurrencia* y *paralelismo*. ¡Investigue!

Bibliografía

[Am] José A. Amor M., *Compacidad en la Lógica de Primer Orden y su Relación con el Teorema de Completud,* UNAM (1993).

[Bo] G. Boole, *An Investigation of the Laws of Thought on Which are Founded the Mathematical Theories of Logic and Probabilities* (1854). Reedición Dover (1958).

[Cr] J. Crossley *et al., ¿Qué es la Lógica Matemática?,* Técnos (1983).

[Cu] H. B. Curry, *Foundations of Mathematical Logic,* Dover (1977).

[CL] Ch.L. Chang and R. Ch. T. Lee, *Symbolic Logic and Mechanical Theorem Proving,* Academic Press (1973).

[Da] M. Davis, *Computability and Unsolvability,* The Math. Assoc. Am. (1973), Reedición Dover (1982).

[DW] M. Davis and E. Weyuker, *Computability, Complexity and Languages: Fundamentals of Theoretical Computer Science,* Academic Press (1983).

[DG] E.G. Dougherty and Ch.R. Giardina, *Mathematical Methods for Artificial Intelligence and Autonomous Systems,* Prentice-Hall (1988).

[Eb] H.D. Ebbinghaus, J. Flum and W. Thomas, *Mathematical Logic,* Springer-Verlag (1984).

[En] H.B. Enderton, *Una Introducción Matemática a la Lógica,* UNAM (1987).

[El] S.M. Engel, *Analyzing Informal Fallacies,* Prentice-Hall (1980).

[Er] E. Engeler, *Introduction to the Theory of Computation,* Academic Press (1973).

[Fr] R. Frost, *Bases de Datos y Sistemas Expertos,* Díaz de Santos, SA (1989).

[Ga] J.H. Gallier, *Logic for Computer Science. Foundations of Automatic Theorem Proving,* John Wiley & Sons (1987).

[Ha] P.L. Halmos, *Teoría Intuitiva de los Conjuntos,* CECSA (1972).

[Hr] D. Harel, *Algorithmics: The Spirit of Computing,* Addison-Wesley 2a. Ed. (1992).

[HA] D. Hilbert y W. Ackermann, *Elementos de Lógica Teórica,* Técnos 2a. Ed. (1975).

[Ho] D.R. Hofstadter, *Gödel, Escher, Bach: Una Eterna Trenza Dorada,* CONACYT (1982).

[Hp] J. Hopcroft, *Turing Machines,* Scientific American **250,** (5) (May. 1984), pp. 70–80.

[HU] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation,* Addison-Wesley (1979).

[HS] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms,* Computer Science Press (1978).

[Kl] S.C. Kleene, *Representation of Events in Nerve Nets and Finite Automata,* Proj. RAND Res. Mem. RM-704 (Dec. 1951). Reedit. en C. Shannon and J. Mc Carthy (Eds.) *Automata Studies,* Princeton Univ. Press (1956), pp. 3–41.

[Kl1] S. Kleene, *Introduction to Metamathematics,* Wolters-Noordhoof & North-Holland (1974).

[Kn] D. E. Knuth, *Algorithms*, Scientific American, (April 1977), pp. 63–80.

[Ko] R. R. Korfhage, *Lógica y Algoritmos,* Limusa (1974).

[La] J. C. Lagarias, *The* $3x + 1$ *Problem and its Generalizations,* Am. Math. Monthly, **92,** (1) (Jan. 1985), pp. 3–23.

[LP] H.R. Lewis and Ch.H. Papadimitriou, *The Efficiency of Algorithms,* Scientific American, **238,** (1) (Jan. 1978), pp. 96–109.

[LP1] H.R. Lewis and Ch.H. Papadimitriou, *Elements of the Theory of Computation,* Prentice-Hall (1981).

[Ma] Yu. Manin, *A Course in Mathematical Logic,* Springer-Verlag (1977).

[Mn] Z. Manna, *Mathematical Theory of Computation,* Mc Graw-Hill (1974).

[Mt] B. Mates, *Lógica Matemática Elemental,* Técnos (1971).

[Me] E. Mendelson, *Introduction to Mathematical Logic,* Wadsworth & Brooks/ Cole Advanced Books and Software 3a. Ed (1987).

[Pi] J. Piaget *et al., Tratado de Lógica y Conocimiento Científico,* vol. 2 *Lógica,* Paidós (1979).

[Po] E. Post, *Recursive Unsolvability of a Problem of Thue*, J. Symbolic Logic (1947). Reedit. en M. Davis, *The Undecidable,* Raven Press (1965), pp. 292–303.

[Qu] W.V. Quine, *Los Métodos de la Lógica,* Ariel (1981).

[RS] M.O. Rabin and D. Scott, *Finite Automata and Their decision Problems*, IBM J. Res. Develop., **3,** 2(April 1959), pp. 114–125.

[Re] G. Révész, *Introduction to Formal Languages*, Dover (1983).

[Ru] B. Russell, *Los Principios de la Matemática*, Espasa-Calpe (1983).

[SM] T. Saaty and P. Kainen, *The Four-Color Problem: Assaults and Conquest*, Dover (1986).

[Se] S. Serrano, *Elementos de Ligüística Matemática,* Anagrama (1973).

[Sm] R. Smullyan, *Lógica de Primer Orden,* Cuadernos Teorema (1983).

[Sm1] R. Smullyan, *Alicia en el País de las Adivinanzas,* Cátedra (1986).

[Sp] E. Spanier, *Grammars and Languages,* Am. Math. Monthly, **76,** (4) (1969), pp. 335–342.

[Ta] A. Tarski, *La Concepción Semántica de la Verdad y los Fundamentos de la Semántica,* en Mario Bunge (Comp.), *Antología Semántica*, Nueva Visión (1972).

[Th] A. Thayse *et al., From Standard Logic to Logic Programming,* John Wiley & Sons (1988).

[Tr] B.A. Trakhtenbrot, *Algoritmos y Computadoras,* Limusa (1974).

[TM] J.P. Tremblay y R. Manohar, *Discrete Mathematical Structures with Applications to Computer Science*, McGraw-Hill (1987).

[Tu] A. Turing, *On Computable Numbers with an Application to the Entscheidungsproblem,* Proc. London Math. Soc., **42,** (2) (1936–7), pp. 230–265; Correc., Ibid., **43,** (1937), pp. 544–546. Reedit. en Davis, *The Undecidable*, Raven Press (1965), pp. 116-154.

Índice

Las palabras se dan por orden de aparición (principalmente, por su definición), no considerando, necesariamente, todas sus ocurrencias a lo largo del texto; el símbolo "↓" denota que la palabra en cuestión aparece en un pie de página; y la abreviatura "*ss.*" significa "*sucesivas*"

Ch

D

E

F

G

Q

R

S

Lógica matemática se terminó de imprimir el
25 de julio de 1995 en Editorial Marsa, S.A. de C.V.
La edición consta de 1000 ejemplares

Yolanda Torres Falcón. Matemática egresada de la Facultad de Ciencias de la UNAM. Obtuvo el grado de Maestra en Lógica y Método Científico en la London School of Economics (University of London). Realizó estudios de Doctorado en Lógica en Bedford College y London School of Economics (University of London).

Desde 1983 es Profesora de Tiempo Completo en el Departamento de Filosofía de la UAM Iztapalapa. De 1984 a 1986 fue Jefa del Area de Lógica y Filosofía de la Ciencia.

Ha participado en congresos y simposios nacionales y publicado varios artículos sobre Lógica.

Actualmente es candidata a Doctor por el Departamento de Matemáticas del CINVESTAV y elabora su tesis sobre Lógica Matemática.

Este es un libro dirigido a estudiantes de matemáticas y de ciencias de la computación. Para los estudiantes de matemáticas presenta un tratamiento riguroso de los temas principales de la materia. Para los estudiantes de computación, cubre material de teoría de matemática de la computación y de análisis de algoritmos, así como herramientas básicas para diseño lógico.

Características especiales del libro:

• Contextualiza la lógica por medio de una introducción sobre argumentos y falacias.
•Se introducen lenguajes y sistemas formales por medio de ejemplos sencillos.
•Hace una presentación exhaustiva de la lógica proposicional desde tres puntos de vista: semántico, sintáctico y algorítmico.
•Se desarrolla la lógica de primer orden sintáctica y semánticamente, esto último con mayor detalle de lo acostumbrado en la literatura.
•Incluye temas poco usuales en la literatura menos espcializada como el teorema de compacidad, el de Lowenheim-Skolen y la existencia de modelos no estándares de la aritmética.
•Cubre material útil para las ciencias de la computación: compuertas lógicas, lenguajes y autómatas, máquinas de Turing y problemas NP - completos.
• Contiene gran cantidad de ejemplos resueltos y ejercicios.