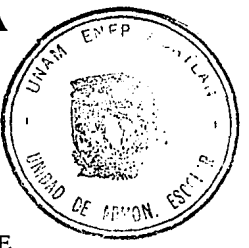
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ACATLAN

COMPUTABILIDAD Y LOGICA



TESIS QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

PRESENTA

JOSE MANUEL GOMEZ SOTO

ASESOR: DR. SERGIO V. CHAPA VERGARA

Marzo de 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

13
2 fjs.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLAN"
DIVISION DE MATEMATICAS E INGENIERIA
PROGRAMA DE ACTUARIA Y M.A.C.

SR. JOSE MANUEL GOMEZ SOTO
Alumno de la carrera de Matemáticas
Aplicadas y Computación
Presente.

De acuerdo a su solicitud presentada con fecha 7 de febrero de 1994, me complace notificarle que esta Jefatura tuvo a bien asignarle el siguiente tema de tesis: "COMPUTABILIDAD Y LOGICA", el cual se desarrollará como sigue:

Introducción
I.- Introducción a la Computabilidad y a la Lógica
II.- Máquina de Turing
III.- El cálculo de Lambda de Church y las funciones Recursivas
IV.- El teorema de Gödel
Conclusiones
Bibliografía

Asimismo fue designado como Asesor de Tesis el DR. SERGIO V. CHAPA VERGARA, profesor de esta Escuela.

Ruego a usted tomar nota que en cumplimiento de lo especificado en la Ley de Profesiones, deberá prestar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar examen profesional, así como de la disposición de la Coordinación de la Administración Escolar en el sentido de que se imprima en lugar visible de los ejemplares de la tesis el título del trabajo realizado. Esta comunicación deberá imprimirse en el interior de la tesis.

E.N.E.P. ACATLAN

Atentamente
"POR MI RAZA HABLARA EL ESPIRITU"
Acatlán, Edo. de Méx., [illegible] de marzo de 1994.

ACT. LAURA MA. RIVERA BECERRA
Jefe del Programa de Actuaría y M.A.C.

JEFATURA DE ACTUARIA Y APLICADAS Y COMPUTACION

Este trabajo lo quiero dedicar a un ser extraordinario, a la Maestra Enedina Soto Gutierrez ... Gracias por ser mi madre.

AGRADECIMIENTOS:

A mi Madre y a mis hermanos: Héctor, Oscar y Aidé...por su eterna confianza.

Quiero agradecer de una manera muy especial al Dr. Sergio V. Chapa Vergara por su gran apoyo y entusiasmo durante el desarrollo de esta tesis; por las maratónicas pláticas llenas de estímulos intelectuales, de "viajes recursivos" y de "metaconceptos"... fue un placer y un honor trabajar con usted.

A Martha por esa "lluvia" de ternura y apoyo, por esos "senderos" llenos de anhelos y quimeras ... por hacer de este trabajo un sueño compartido.

A Blanca Elena del Pozo le estoy profundamente agradecido por su generoso apoyo y paciencia, por sus valiosisimos comentarios y por ofrecerme ese rigor académico que siempre le ha caracterizado ... Gracias Helen.

Al Dr. Ulises Beltrán por el gran apoyo que me brindó ... por ese entusiasmo que tiene por los proyectos académicos.

A a la señora María Torres, mi eterno agradecimiento, por su gran sentido de amor y solidaridad, por haberme ofrecido un espacio en su corazón y en esta ciudad ... gracias por todo Doña Mary.

A la Sra Rosemary Olson por haber brindado el privilegio de ser una más de las generaciones de estudiantes a las que tanto ha ayudado ... mi profundo agradecimiento y reconocimiento a usted y a la "Quinta Restaumex" mi segunda Alma Mater.

A Lauro, por esos grandes momentos, por tu apoyo y la guerra de "porras"... por tu amistad.

A Javier por tu gran apoyo, por darme 'asilo' y compartime tu ajuar computacional en estos días tan decisivos.

A Poncho por tus acertados comentarios y consejos; a Lupita López, Alejandro Cruz, Roberto Banchik, Jesus, Norma, Mónica Tinajero por su apoyo y colaboración.

A mis compañeros de trabajo de la Asesoría Técnica, en especial a la gente de cómputo por su apoyo y constantes "porras".

En general mi agradecimeinto a toda la gente que de una u otra manera me ayudó durante el desarrollo de mi tesis.

Quiero concluir expresando mi más sentido reconocimiento y agradecimiento a la Universidad Nacional Autónoma de México y al pueblo que hace posible que existan instituciones como esta.

Ella está en el horizonte -dice Fernando Birri-. Me acerco dos pasos, ella se aleja dos pasos. Camino diez pasos y el horizonte se corre diez pasos más allá. Por mucho que yo camine, nunca la alcanzaré. ¿Para que sirve la utopía? Para eso sirve: para caminar.

Eduardo Galeano.

CONTENIDO

CONTENIDO

INTRODUCCION

Introducción:

En la Teoría de la Computación un concepto fundamental es la Máquina de Turing, cuya noción de la computabilidad se basa en el problema de la detención. La computabilidad tiene sus aproximaciones desde dos enfoques equivalentes a la caracterización de la Máquina de Turing. El primero corresponde a es la indecibilidad de la lógica mediante el Teorema de Gödel, que es una prueba de insolubilidad del problema de la detención. El otro enfoque es el que se deriva de la indecibilidad de la lógica mediante funciones recursivas de Church, ya que son equivalentes a las funciones Abacus y a las funciones Turing.

Estos enfoques y el de la propia máquina de Turing pueden ser vistos como equivalentes y encontrar consecuencias directas dentro del campo de la Teoría de la Computación, siendo la esencia misma de las diversas áreas. De hecho estos enfoques nacen al tratar de resolver. un problema filosófico-matemático propuesto por David Hilbert, el cual se esquematiza en la tabla 0.

El vínculo de la Computabilidad y la Lógica ha presentado uno de los problemas fundamentales e interesantes cuya solución incursiona el campo de la Filosofía, las Matemáticas Puras y Aplicadas. El objetivo central de este trabajo es presentar estos enfoques, que son las piedras angulares de la Computabilidad y la Lógica. Asimismo, el motivo de esta tesis es la de presentar los planteamientos de Turing-Church-Gödel; eslabonarlos en sus diversas respuestas; mostrar los conceptos que surgen en cada planteamiento, así como sus isomorfismos y equivalencias, cuya riqueza de ideas es en si misma objeto de estudio y de amplia reflexión.

Un objetivo adicional es presentar a los estudiosos del área un panorama general de estos problemas de Computabilidad y Lógica dentro del área de Teoría de la Computación. Con frecuencia el alumno se pierde en complicadas y largas demostraciones, proyectos computacionales y no ve el sentido de los problemas fundamentales y sus relaciones entre sí.

Tabla 0. Equivalencia entre los enfoques de Turing, Gödel y Church.

Hilbert		
¿Existe un sistema matemático en general que esa consistente y completo, en el que se pueda decidir la verdad o falsedad de todas las proposiciones?		
Gödel	**Turing**	**Church**
¿Existe un sistema general que sea consistente y completo, en el que se pueda decidir la verdad o falsedad de todas las proposiciones?	¿Existe un procedimeinto mecánico general que pudiera resolver todos los problemas matemáticos?	¿Existen funciones para las que siempre existirá un valor para cualquier argumento?
No existe tal sistema, y jamás se podrá construir uno	No existe un procedimeinto mecánico que resuelva todos los problemas matemáticos	No todas las funciones son computables
Teorema de la Incompletitud de Gödel	**Problema de la detención**	**Funciones computables**
Habrá proposiciones para las que no se pueda saber si son verdaderas o falsas	No se sabe si se detendrá o no la máquina de Turing	Existen funciones indecidibles

Creo que esta tesis puede alcanzar estas metas por el tipo de estructura y el lenguaje que se ofrece. De esta forma, la organización del trabajo se presenta bajo el siguiente esquema:

En el primer capítulo se hace una síntesis histórico-teórica de los conceptos que dieron forma a la computabilidad. Estos conceptos abarcan desde la teoría de conjuntos, la enumerabilidad, la diagonalización de Cantor, hasta llegar al problema de la axiomatización de las matemáticas, en el que se plasman los grandes intentos de axiomatizar, como el sistema de Frege, los Principia Matemática de Russell y el programa de Hilbert, que fue el punto de partida de los enfoques a su solución por parte de Gödel, Turing y de Church. Este capítulo representa la guía del trabajo, en él se introduce al lector al problema de Hilbert, a la filosofía del planteamiento por parte

de Gödel,Turing y Church, se habla de su repercusión e importancia, de las interrelaciones entre cada uno de ellos, así como de los isomorfismos que los hacen equivalentes.

En el segundo capítulo se muestra el enfoque que consiste en la creación del modelo matemático conocido como la máquina de Turing. Se hace una descripción del modelo, donde se plantea el problema de Hilbert en términos de procedimientos mecánicos, y el problema de la detención de la máquina de Turing como el punto de decisión para determinar si un problema tiene o no solución. Se realiza la demostración, en términos de la máquina de Turing, en donde se determina que el problema planteado por Hilbert no tiene solución. En este capítulo sobresale el concepto de computabilidad, planteado en el sentido en que un problema es computable, si al tratar de resolverlo con máquina de Turing ésta llegara a detenerse. Se presenta la perspectiva filosófica y matemática del problema de la indecibilidad como el problema de la detención de la máquina de Turing, y las operaciones lógicas y matemáticas como procesos mecánicos.

En el tercer capítulo se muestra otro de los grandes enfoques que dio lugar al concepto de la computabilidad: el Cálculo Lambda de Church. Se hace un desarrollo de como se contruye un sistema en términos del Cálculo lambda y como a partir del manejo de estas funciones, se pueden concluir resultados del sistema dado. La filosofía matemática del concepto de la computabilidad es mostrado en términos de funciones. Se definen las funciones Recursivas, Abacus, Turing y como estas son equivalentes, lo cual de manera indirecta demuestra la indecibilidad del problema de Hilbert. Se deja ver la estrecha relación entre los conceptos de computabilidad y recursividad, la equivalencia entre ellos, la creación de sistemas mediante la definición recursiva de funciones, y su gran importancia como conceptos inherentes a los procesos computables. De manera sucinta se plasman los conceptos que unen a los tres enfoques.

En el cuarto y último capítulo se plantea el teorema de la Incompletitud de Gödel que es la respuesta Gödeliana al "Entscheidungsproblem" y, en general, a todo el programa de Hilbert. Gödel había llegado a la conlusión de que no es posible llevar a cabo dicho programa. Para llegar a este importante resultado se da una introdución de los enfoques de semántica y sintáxis y a la lógica de primer orden. Se hace un planteamiento más detallado del programa de Hilbert, cómo Gödel aborda dicho programa, así cómo la argumentación de su teorema. Durante el desarrollo del planteamiento de Gödel una vez más se dejan ver cómo los conceptos de computabilidad y recursividad son inherentes a dicho planteamiento, la computabilidad en su famosa enumeracion de Gödel, el reflejo de sus proposiciones metamatemáticas en la aritmética y la recursividad en la misma definición de su modelo, es decir la aritmetización definida en términos recursivos.

CAPITULO I.- INTRODUCCION A LA COMPUTABILIDAD Y A LA LOGICA

I.- INTRODUCCION A LA COMPUTABILIDAD Y A LA LOGICA

1.1 LOGICA Y TEORIA DE CONJUNTOS

La lógica surge como un intento de formalizar los procesos "inteligentes" del razonamiento. Siempre se ha dicho que la capacidad humana de razonar es lo que distingue al hombre de otras especies; resulta entonces un tanto paradójico, a primera vista, mecanizar el razonamiento que es lo más humano que tenemos. Los antiguos griegos ya planteaban al razonamiento como un proceso sujeto a esquemas, y que en parte está gobernado por leyes perfectamente formulables. Aristóteles codificó los silogismos y Euclides codificó la geometría; convirtiéndose entonces en los precursores de la deducción formal y el análisis matemático, áreas de estudio que conforman el complicado desarrollo de la Ciencia Matemática. Durante mucho tiempo estas dos corrientes se desarrollaron por separado, (desde los griegos hasta aproximadamente el año 1600 y 1700), reconciliandose en la época de Newton y Leibnitz cuando se dio la invención del cálculo. Cabe señalar que el desarrollo de la lógica no fue sostenido y desde la época griega no se realizaron más estudios, por lo que tuvieron que pasar muchos siglos para que pudiera registrarse un avance en el estudio del razonamiento axiomático.

Fue hasta el siglo XIX cuando los lógicos ingleses George Boole y Augustus De Morgan intentaron dar forma final y definitiva a lo que ahora se tiene como la deducción formal y sometieron los esquemas estrictamente deductivos de razonamiento a una codificación que deja muy atrás la codificación aristotélica.[1] A partir de esta fecha filósofos y matemáticos han contribuido con el fin de resolver problemas fundamentales que interrelacionan la matemática y la lógica. En la figura 1.1 se intenta mostrar una síntesis cronológica de las aportaciones que dieron los principales matemáticos para el desarrollo de los pilares de la lógica y la matemática.

[1] Aristóteles ya había realizado reglas explícitas de deducción, pero las empezó a hacer en lengua natural. Boole quería algo más poderoso y desarrolló un sistema puramente simbólico.

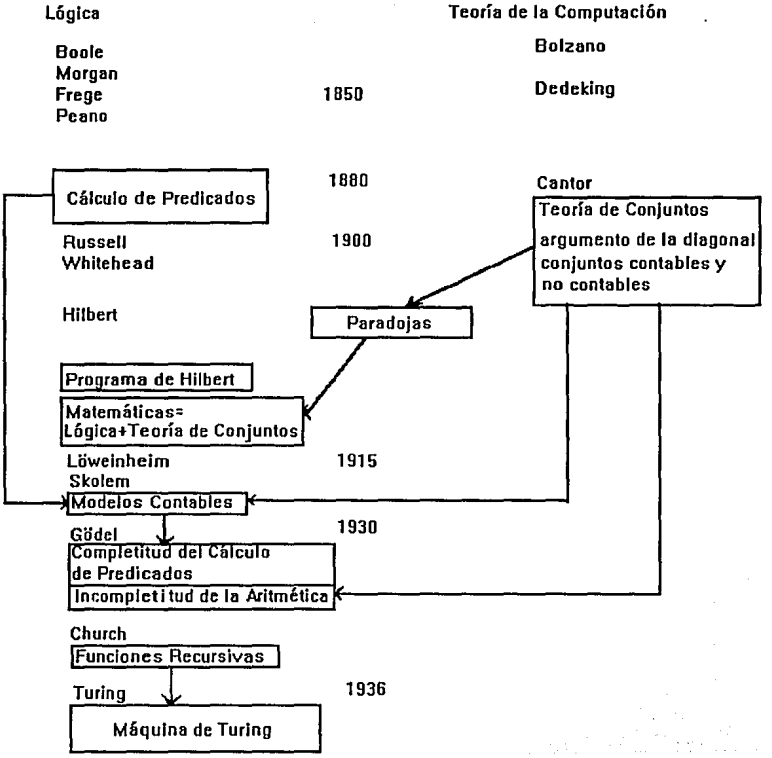


Figura 1.1 Síntesis Cronológica de los aportes en desarrollo de la Lógica y la Teoría de Conjuntos.

Por un lado, Gottlob Frege en Jena y Giuseppe Peano en Turín se dieron a la tarea de combinar el razonamiento formal con el estudio de conjuntos y números. En Gottingen, David Hilbert elaboró formalizaciones de geometría más estrictas que las de Euclides. Por otro lado, en el desarrollo del análisis matemático, surgieron controversias sobre el significado de los conceptos de derivada e integral de Newton porque hablaba de infinitesimales. Sin embargo, Cantor y otros demostraron que para llevar a cabo derivadas e integrales de una manera adecuada se deberían considerar

conjuntos infinitos de una forma muy precisa. No había forma de evitar el conjunto de infinitos. Esto representó el origen de la teoría de conjuntos.[2] La importancia de esto fue que Cantor llegó a la teoría de conjuntos a partir de un problema de análisis. El no trataba de definir los números naturales o planteamientos utilizados en la teoría de conjuntos, sino que buscaba el análisis de conjuntos infinitos de números reales. Todo esto parecía indicar que con mucha paciencia y suficientes definiciones, cualquier campo de las matemáticas era posible definirlo en términos de lógica y teoría de conjuntos, pudiendo realizar todas las demostraciones en dichos campos con el cálculo de predicados.

En cuanto a la Teoría de conjuntos Cantor iba a la vanguardia y, de alguna manera, fue más allá cuando trató de resolver problemas de análisis; él estaba interesado en los conjuntos por los conjuntos mismos y descubrió lo fascinantes que eran. Es importante dar al menos dos demostraciones de los resultados de Cantor porque los argumentos que usó fueron totalmente revolucionarios y estos se han extendido en la totalidad de la lógica desde entonces. De hecho, la mayoría de los teoremas pueden ser presentados en uno u otro argumento de Cantor. Al considerar a los conjuntos infinitos, Cantor concluyó que unos pocos de estos conjuntos infinitos eran similares al conjunto de los números naturales (en su totalidad) en el sentido de que los elementos de estos correspondían uno a uno con los números naturales, es decir que eran enumerables. Su primer gran descubrimiento fue demostrar que los números racionales[3] corresponden uno a uno con los números naturales. Esto sorprendió mucho a la comunidad pues, como se sabe, los números racionales se pueden poner densamente sobre una línea; en otras palabras, entre cualesquiera dos números siempre existe un número racional, por lo tanto, era inconcebible que pudieran enumerarse. El método de Cantor afirmaba que los números racionales pueden arreglarse como se muestra en la figura 2. En el primer renglón se colocan todos las fracciones con denominador igual a 1, en el segundo renglón todas las fracciones con denominador igual a 2, en el tercero aquellos con igual a 3 y así sucesivamente. Esta tabla se puede construir con todos los números racionales. Enseguida se enumeran comenzando en el extremo superior izquierdo y se zig-zagea, para obtener la secuencia: $\frac{1}{1}, \frac{2}{1}, \frac{1}{2}, \frac{1}{3}, \frac{2}{2}, \frac{3}{1}, \frac{4}{1}, \frac{3}{2}, \ldots$

[2] A partir del enlace implícito que se da entre el análisis y la lógica surge la Teoría de Conjuntos y con ella la inquietud de axiomatizar y ordenar estos conceptos. Frege fue el primero en aportar elementos para esto, y su trabajo influiría en todos los esfuerzos posteriores que se encaminaron en este sentido. Russell fue uno de los matemáticos que estaba profundamente interesado en el trabajo de Frege y a partir de esta inquietud se dedicó a las proposiciones que en matemáticas fueran realmente lógicas. Russell demostró que las matemáticas eran la lógica y la teoría de conjuntos

[3] Los números racionales son fracciones $\left(\frac{p}{q}\right)$ donde, p y q son números naturales, con $q \neq 0$

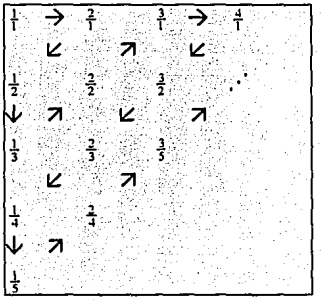


figura 2. Enumeración de los números racionales.

Mediante este método no se deja fuera ningún número racional y a cualquiera de estos puede asignarse un número natural que denota el lugar del número racional en la lista.

$$\begin{array}{cccccccccc} (& \frac{1}{1}, & \frac{2}{1}, & \frac{1}{2}, & \frac{1}{3}, & \frac{2}{2}, & \frac{3}{1}, & \dots &) \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & & \\ (& 1, & 2, & 3, & 4, & 5, & 6, & \dots &) \end{array}$$

El segundo descubrimiento, para sorpresa de muchos, fue que los números reales no cumplían con esto, después de ver el extraordinario hecho de que un conjunto tan denso sobre una línea como los racionales se pudiera contar, se podría esperar que cualquier conjunto infinito pudiera contarse también.

Los números reales corresponden a los puntos de una línea (todos los puntos de una línea forman un continuo) y pueden expresarse mediante expansiones de decimales infinitos y en general la expansión decimal tiende a ser infinita (por ejemplo $\sqrt{2}$ es un decimal infinito; y no hay una representación finita para él). Para demostrar que los números reales no son contables, se puede hacer por *reductio ad absurdum*. Supóngase que el resultado que se trata de establecer es falso, es decir, que el conjunto de los números reales es contable. Luego entonces los números reales entre el 0 y el 1 serian ciertamente contables, y se generaría una lista que relacionara a cada número de los reales con los números naturales por parejas:

Números Naturales		Números Reales
1	⇔	0.10357627183...
2	⇔	0.14329806115...
3	⇔	0.02166095213...
4	⇔	0.43005357779...
5	⇔	0.92550489101...
6	⇔	0.59210343297...
7	⇔	0.63667910457...
8	⇔	0.87050074193...
9	⇔	0.04311737804...
10	⇔	0.78635081150...
11	⇔	0.40916738891...
⋮	⋮	⋮

Para la lista de los números reales que se presenta, se han marcado los dígitos de la diagonal en "negritas": 1,4,1,0,0,3,1,4,8,5,1,.... Esta diagonal persigue construir un número real (entre el 0 y el 1) cuyo decimal de expansión (después del punto decimal) difiera de estos dígitos en cada lugar que le corresponda. Por definición vamos a decir que el dígito será 1 cuando el dígito de la diagonal sea diferente de 1, y 2 cuando el dígito de la diagonal sea igual a 1.

Luego entonces, en este caso el número real que nos dará es: 0.21211121112. Este número real no puede aparecer en la lista debido a que difiere del primer número

en el primer lugar del decimal (después del punto decimal), del segundo número del segundo lugar después del punto decimal, del tercero en el tercer lugar. y así sucesivamente.

Esta es una contradicción porque se supone que la lista contiene a todos los números reales entre el 0 y el 1. Por reducción al absurdo se ha demostrado entonces, en pocas palabras, que no existe correspondencia uno a uno entre los números reales y los números naturales, por lo que se ha descubierto un conjunto infinito no contable.

Pero la idea no iba a quedar ahí, y el mismo Cantor, haría un gran desarrollo de este argumento: describió brevemente que tomando cualquier conjunto S no se le puede poner en correspondencia uno a uno con el conjunto de todos sus subconjuntos, esto es, el conjunto $\{T: T \subseteq S\}$. Este argumento es en realidad el mismo que el anterior, pero lo demuestra desde un punto de vista diferente. Hizo lo que llamaremos superficialmente una lista de miembros de S. Es posible que sea un conjunto que no se puede listar, pero solo imagine que de alguna manera se trata de empatar a los miembros S con los subconjuntos de S, y sea T_s el subconjunto que se hizo corresponder con el miembro de S. Inmediatamente después Cantor construye un subconjunto U que no está en la lista, que consiste de miembros s que están empatados con un subconjunto al que no pertenecen. Se pueden hacer diferentes subconjuntos U a partir de cualquier subconjunto T_s ya que T_s o bien tiene una s o no la tiene. De cualquiera forma hacemos a U diferente de T_s con respecto al elemento s. Si s está dentro de T_s, se deja a s fuera de U; si no está dentro de T_s lo ponemos dentro de U. Por lo tanto, hay más subconjuntos de S que pueden estar relacionados uno a uno con los elementos de S; cualquier relación que se haga omite al conjunto U tal como ya se describió.

Existe una consecuencia más profunda sobre este argumento. Suponga que S es el conjunto de todos los conjuntos del universo. Este argumento parece decir que si se comienza con todos los conjuntos del universo y se forman todos sus subconjuntos, estos serían más, ¡pero, no se pueden obtener más conjuntos que todos los conjuntos del universo¡. Russell abordó este problema que dejó manifiesto en su célebre paradoja.

> Las clases pueden formularse en dos grupos, las que se contienen a sí mismas y las que no. Llamaremos NORMAL a una clase si y sólo si no se contiene a sí misma como miembro. En otro caso es NO NORMAL.
>
> Admitamos que por definición "N" es el nombre de todas las clases normales. Y preguntémonos sí N misma es una clase Normal. Sí N es normal, es un miembro de sí misma, pues por definición de "N", N contiene todas las clases normales; pero entonces N es también no normal, puesto

que, por definición de "NO NORMAL", las clases no normales son las que se contienen así mismas como miembros.

A la inversa, si N es no normal, entonces es miembro de sí misma, por definición de "no normal"; pero es también NORMAL porque pertenece a N, que es la clase de todas las clases normales.

Dicho de otro modo: N es no normal si y sólo si es normal. (ver [Newm56]).

A partir de esta paradoja se comprendió que los conjuntos no pueden ser miembros de sí mismos; esta regla tiene por misión evitar la autorreferencia, es decir, la definición de algo a partir de sí mismo. Si un conjunto se contiene a sí mismo, entonces ¿qué es un conjunto? (El trabajo de Frege y muchos otros estaban llenos de autorreferencias por lo que se presentaban diversas paradojas en sus teorías.)

La autorreferencia era el problema que según Russell y Whitehead no permitía llevar a cabo la axiomatización. Es decir, los "Bucles extraños" inmersos en las matemáticas eran el enemigo a vencer. Había que "extirparlos" de donde se encontraran y plantear esquemas y definiciones que los evitaran. Esta tarea no ha sido tan simple como se creía, porque puede ser difícil saber dónde exactamente está ocurriendo una autorreferencia. Sin embargo, tanto Russell como Whitehead se dieron a esta tarea en su gran obra llamada *"Principia Mathematica"* (P.M.) que es un esfuerzo por dejar limpios de autorreferencias a la lógica, la teoría de conjuntos y la teoría de números.

Este objetivo se logró con la Teoría de Tipos, que tenía como idea básica que un conjunto de un tipo dado no puede abarcar sino conjuntos de tipo más bajo, además de objetos. Dado esto, como cada conjunto pertenece a un tipo específico, es claro que ningún conjunto puede contenerse a sí mismo, porque entonces tendría que pertenecer a un tipo más alto que su propio tipo. Este enfoque introduce sin duda una jerarquización un tanto artificial, que si bien era adecuada para aquellos cuyos intereses no rebasaban el campo de la teoría de conjuntos, para los que querían eliminar de una manera general las paradojas, los sumergía en una gran inquietud que se resume en dos grandes interrogantes:¿Toda la matemática quedaba realmente englobada en los métodos diseñados por Russell y Whitehead? y ¿Esos métodos eran coherentes consigo mismos? Esto podría ser posible planterlo en una sola pregunta: ¿era absolutamente claro que siguiendo los métodos de Russell y Whitehead ningún matemático del mundo podría llegar nunca a resultados contradictorios?; esta cuestión particular se venía haciendo con el planteamiento del "Entscheidungsproblem" dado por David Hilbert.

1.2 COMPUTABILIDAD

EL "ENTSCHEIDUNGSPROBLEM " DE HILBERT

A principios de este siglo el distinguido matemático alemán David Hilbert planteó su décimo problema que consistía en de encontrar un sistema formal que generara todas las oraciones matemáticas verdaderas, y sólo estas. El reto era construir un sistema que fuera coherente (a salvo de contradicciones) y completo (que toda proposición válida de teoría de números pudiera desarrollarse dentro del armazón de dicho sistema).[4] Este problema era parte del "programa de Hilbert" y ocupó la cabeza de muchos de los mayores matemáticos del mundo durante los primeros treinta años del presente siglo.

El atractivo de este cuestionamiento era enorme: si por ejemplo el esquema de Frege se llevara a cabo por completo y, el método pudiera determinar la verdad o falsedad de cualquier oración en lógica formal, entonces el método podría también determinar la verdad o la falsedad de cualquier oración matemática sin importar qué tan complejo fuera. Y, por otro lado, si existía una respuesta afirmativa al programa de Hilbert, podrían reducirse las matemáticas a cálculos mecánicos. Estas grandes espectativas mantenian a mucho matemáticos trabajando; de hecho, grandes desarrollos de la lógica, computación y matemáticas le deben mucho al programa de Hilbert, ya que al intentar los problemas que este planteaba surgieron grandes conceptos matemáticos. Uno de ellos es el teorema de Gödel que es considerado uno de los más grandes descubrimientos de la Lógica.

EL TEOREMA DE GÖDEL.

En 1931 apareció en una revista científica alemana un artículo extraordinariamente difícil y brillante titulado "*Über formal unentscheidbare Satze der Principia Mathematica und verwandter Systeme*"[5] ("Sobre proposiciones formalmente indecibles de 'Principia Mathematica' y sistemas análogos"). El autor del artículo era Kurt Gödel, un joven matemático de 25 años de la Universidad de Viena, miembro en ese entonces del Advanced Study de Princeton. En este artículo se encontraba el Teorema de Gödel en el que se mostraba que estaba injustificado el supuesto implícito en el programa de Hilbert[6].

---

[4]El problema fue propuesto por primera vez en forma parcial en el Congreso Internacional de Matemáticos realizado en Paris y de una forma formal y completa en el Congreso de Bolonia en 1928.

[5]Publicado en 1931 en *Monatshefte fur Mathematik und Physik.*

[6]El supuesto de que podría existir algún método para distinguir oraciones verdaderas y falsas en lógica formal.

Gödel probó que cualquier sistema consistente de lógica formal lo bastante poderosa para formular oraciones en la teoría de números, puede incluir oraciones verdaderas pero no pueden ser demostradas, porque los sistemas axiomáticos consistentes tales como los realizados por Russell y Whithead no pueden abarcar todas las oraciones de la materia que ellos buscan formalizar; tales sistemas se dicen que son incompletos.

A Gödel se le ocurrió la idea de utilizar el razonamiento matemático para explorar el pensamiento matemático. Esa idea de hacer de las matemáticas una disciplina "introspectiva" resultó ser enormemente dinámica, y la más fecunda de sus implicaciones es una que él mismo encontró: el Teorema de la Incompletitud. Qué propone este teorema y cómo lo demuestra son dos cosas distintas. Podemos comparar el Teorema con una perla y el método de demostración con una ostra. La perla es estimada por su tersura y su sencillez; la ostra es un ser vivo y complejo de cuyas tripas brota esa gema misteriosamente simple.

El Teorema de Gödel aparece en la Proposición VI de un artículo suyo "Sobre proposiciones formalmente indecibles y sistemas análogos, y dice así: A cada clase *k* *w*-consistente y recursiva de *fórmulas* corresponden *signos de clase r* recursivos, de tal modo que ni *v* Gen *r* ni Neg (*v* Gen *r*) pertenecen a Flg (*k*) (donde *v es* la variable libre de *r*). Lo cual significa que toda formulación axiomática de teoría de los números incluye proposiciones indecibles.

Tal es la perla.

En esta perla es difícil ver una autorreferencia. Ello se debe a que tal, está sepultada en la ostra, o sea en la demostración. La demostración del Teorema de la Incompletitud de Gödel está trabada con la escritura de una proposición matemática auto-referencial, de la misma manera que la paradoja de Epiménides es una proposición lingüística auto-referencial ("Todos los cretenses son mentirosos", Epiménides era cretense). Pero servirse del lenguaje para hablar acerca del lenguaje es cosa simple, mientras que no es nada fácil ver cómo una proposición relativa a números puede hablar acerca de sí misma. Hizo falta un genio para esto tan simple: conectar la idea de las proposiciones autorreferenciables con la teoría de los números. En el momento en que Gödel tuvo la intuición de que esa

proposición podía crearse, dejó ya atrás el principal de los obstáculos. La hechura misma de la proposición no fue sino la elaboración de su espléndido chispazo intuitivo.(ver [Hofs79]).

Es preciso, en primer lugar, que quede absolutamente claro en dónde está la dificultad. Las proposiciones matemáticas -limitémonos a la teoría de números- se refieren a las propiedades de los números enteros. Los números enteros no son proposiciones, ni tampoco lo son sus propiedades. Una proposición de teoría de números no habla *acerca de* una proposición de teoría de los números; *es* sólo una proposición de teoría de los números. Este es el problema; pero Gödel se dio cuenta que algo "bullía" por dentro.

Gödel intuyó que una proposición de teoría de números podía hablar acerca de una proposición de una teoría de números (inclusive, quizá, acerca de sí misma) a condición, simplemente, de hacer que los números cumplieran las funciones de las proposiciones. Dicho de otro modo, en la médula de su construcción está la idea del *código*. En el código de Gödel, llamado por lo común "numeración de Gödel", se hace que los números cumplan las funciones de símbolos y de secuencias de símbolos. De esa manera, siendo una secuencia de símbolos especializados, cada proposición de teoría de los números adquiere un "número de Gödel", al cual uno puede referirse. Este recurso de codificación permite que las proposiciones de la teoría de los números se entiendan en dos niveles distintos: como teoría de los números y como **proposiciones acerca de proposiciones de teoría de los números.**

Una vez con este esquema de codificación, Gödel tuvo que elaborar detalladamente una manera de transportar la paradoja de Epiménides a un formalismo de teoría de los números. Su transplante final de la paradoja de Epiménides no decía "Esta proposición de teoría de los números es falsa", sino que "Esta proposición de Teoría de los números no tiene demostración". De aquí pueden originarse no pocas confusiones a causa de que la gente no entiende en general el concepto de "demostración" sino en forma bastante vaga. De hecho la obra de Gödel se inscribe como episodio del largo esfuerzo de los matemáticos por explicarse a si mismos qué cosa son las demostraciones.[7]

Dicho sea de paso, esta aseveración de Gödel no es el teorema de Gödel, tal como la aseveración de Epiménides no es la aseveración de que "la observación de Epiménides es una paradoja". Ahora podemos precisar cuál es el efecto del descubrimiento de Gödel. Mientras que la aseveración de Epiménides crea una

[7]El hecho importante que hay que tener en cuenta es que las demostraciones son pruebas *dentro de sistemas fijos* de proposiciones. En el caso de la obra de Gödel, el sistema de razonamiento teórico-numérico a que se refiere la palabra "demostración" es el de los *Principia Mathematica.*, gran obra de Bertrand Russell y Alfred North Whitehead, publicada entre 1910 y 1913. Por lo tanto la aseveración G de Gödel debería escribirse más adecuadamente así: Esta aseveración de teoría de los números no tiene ninguna demostración en el sistema de los *Principia Mathematica.*

paradoja, puesto que no es verdadera ni falsa, la aseveración G de Gödel no es demostrable dentro de los *Principia .Mathematica* (P.M.), pero es verdadera. ¿Y cuál es la conclusión de todo esto? la conclusión es:

el sistema de los *Principia Mathematica* es "incompleto"; hay proposiciones verdaderas de teoría de los números para cuya demostración resulta demasiado débil el método de los P.M.

Los *Principia Mathemática* fueron la primera víctima de este golpe pero ciertamente no la única. Las palabras "y sistemas afines" que se leen en el título del trabajo de Gödel son muy elocuentes: en efecto, si Gödel se hubiera limitado a señalar un defecto en la obra de Russell y Whitehead, no habrían faltado otros matemáticos dispuestos a mejorar el sistema de los P.M. y sobreponerse al Teorema de Gödel. Pero esto no era posible: la demostración de Gödel se aplicaba a cualquier sistema axiomático cuyo propósito fuera lograr las metas que Whitehead y Russell se habían fijado. Un solo sistema básico bastaba para hacerse cargo de cada uno de estos sistemas. En suma, lo que Gödel demostró fue que la demostrabilidad es un concepto más endeble que la verdad, independientemente del sistema axiomático de que se trate.

De esta forma, el teorema central demostrado por Gödel destruía prejuicios muy arraigados acerca del método matemático y terminaba con las grandes esperanzas suscitadas por decenios de investigación acerca de los fundamentos de la matemática. Gödel mostraba que el método axiomático, explotado por matemáticos con creciente poder y rigor desde los días de Euclides, tiene ciertas limitaciones inherentes cuando se aplica a sistemas lo suficientemente complejos: de hecho cuando se aplica a sistemas relativamente tan sencillos como la aritmética corriente de los números cardinales.

También demostró que es imposible mostrar la consistencia interna (no-contrariedad) de tales sistemas, si no es mediante el uso de principios de inferencia tan complejos que su consistencia interna es tan susceptible de duda como la de los sistemas mismos.

El trabajo de Gödel puso fin al programa de Hilbert. No puede existir un método para decidir cuando algunas oraciones arbitrarias en matemáticas son falsas o verdaderas. Si existiera el método podría constituir la demostración de todas las oraciones verdaderas, y Gödel ha demostrado que tal prueba es imposible con un sistema axiomático consistente. En este contexto continúa una simple pregunta análoga a la de Hilbert que no ha sido resuelta: ¿Existe un método en donde todos las probables oraciones en matemáticas puedan ser demostradas a partir de un conjunto de axiomas lógicos?.

Pero el artículo de Gödel no era, de todos modos, sólo relevante en lo negativo. Introducía en la fundamentación de la matemática una nueva técnica de análisis que es comparable en fecundidad al poder del método algebráico introducido por Descartes en el estudio de la geometría. El método dado por Gödel sugirió e inició nuevos problemas y ramas de la investigación lógico-matemática. Provocó una apreciación crítica, aún no completa, de filosofías muy difundidas del conocimiento en general, y de las filosofías de las matemáticas en particular.

Otro de los frutos de "Entscheidungsproblem" consistió en dar origen a otra de las formas para expresar una de las ideas más fecundas de las matemáticas: La Computabilidad. Esta forma es conocida como las Funciones de Church.

EL CALCULO LAMBDA DE CHURCH

Un año después de que Gödel hiciera su demostración, Alonzo Church de la Universidad de Princeton, desarrolló un lenguaje consistente formal llamado el cálculo lambda[8] (λ). Este es útil para razonar acerca de las funciones matemáticas, tal como la raíz cuadrada, las logarítmicas y cualquier función más compleja que pueda ser definida.

El argumentaba que si una función matemática puede ser computada, es decir, que ésta pueda ser evaluada para cualquier número en el dominio de definición, entonces la función puede ser definida en el cálculo lambda. El trabajo de Church demostró que si existiera algo parecido a una función matemática que se pudiera expresar en el cálculo lambda y que no fuera computable, no habría ningun método para determinar si una expresión matemática se pudiera demostrar, ya no digamos determinar si tal expresión era verdadera o no.

La última hipótesis sobreviviente del programa de Hilbert sería descartada en abril de 1936; Church publicó una fórmula lógica que no es computable en su propio sistema.

Uno de los frutos importantes de estos hallazgos fue el concepto de computabilidad el cual nace como una importante idea matemática. (El poder de esta idea estriba particularmente en el hecho de). Estipula algunas operaciones aún bien definidas en matemáticas no son computables.

Dado que la computabilidad es un genuino y absoluto concepto matemático, después se dieron otras formas de expresar las ideas de computabilidad. Por ejemplo Turing trabajó independientemente a Church y también usó la conexión lógica técnica

[8]También conocido por Notación Lambda.

entre el problema de Hilbert y la idea de una función computable.[9] Así, los dos partiendo de nociones diferentes de sistemas lógicos llegaron a resutados análogos sobre la indecibilidad y universalidad. [10]

LA MAQUINA DE TURING

Turing fue otro de los grandes matemáticos que estaban muy interesados por el problema planteado por Hilbert. Sin embargo, para atacar el problema en una forma mucho más directa y concreta que Church, creó un simple pero preciso modelo del proceso de la computación, y las máquinas de Turing fueron diseñadas para cumplir esa necesidad. Turing definió el concepto de máquina. El dio argumentos para sugerir que cualquier computación que una persona pudiera hacer podría ser realizada por una máquina. Analizó todos los pasos que una persona hace cuando hace cálculos (computaciones): escribir cosas, buscar bloques de símbolos, hacer notas, regresar a cosas que se hicieron antes, hacer listas, etc, y planeó un tipo de máquina que pudiera hacer todo este tipo de operaciones de una forma muy simple. La máquina sólo necesitaba ser capaz de colocarse en cuadros sobre una cinta e identificar los símbolos que hay en los cuadros; moviéndose un cuadro hacia la izquierda o derecha se podía cambiar de símbolo. Esta máquina tiene un programa que decide qué hacer si se encuentra con cierto símbolo, y es un programa que tiene un final.

El concepto que da la máquina de Turing coincide con el concepto de Gödel el cual estaba definido en términos del lenguaje de la aritmética. De la misma forma el concepto de Church también había definido a las funciones computables por un método diferente. Los tres llegaron al mismo concepto. El resultado es que el vago concepto de computabilidad tuviera una definición tan precisa. Por primera vez se podía probar qué cosas no tienen solución.

¿Qué significa tener un algorítmo o una manera de resolver un problema?

Por ejemplo, se sabe que el algorítmo para resolver ecuaciones cuadráticas está dado a traves de:

La ecuación $ax^2 + bx + c = 0$ tiene soluciones

[9] Como veremos más tarde, las funciones computables son equivalentes a las funciones recursivas y abacus.

[10] Resultados que pronto mostraron que sus sistemas eran equivalentes

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Si se sustituyen los números a,b,c la respuesta se da mecánicamente.

Se puede tener una máquina para hacer este trabajo y esta máquina podría resolver infinitamente las ecuaciones cuadráticas, todas en forma mecánica.

Sin embargo, hay problemas para los cuales no se conoce ninguna solución mecánica. Obviamente no se cuenta con un método para decir cuando una oración de matemáticas es verdadera o falsa.

El problema de Hilbert que abordó Turing (*El Entscheidungsproblem*) iba más allá de cualquier formulación Matemática en términos de sistemas axiomáticos. La cuestión era:

> ¿Hay algún procedimiento mecánico general que pudiera, en principio, resolver todos los problemas de matemáticas (de una clase adecuadamente bien definida) uno después de otro?[11]

Turing llegaría a resolver esta cuestión, haciendo una brillante conexión entre la idea de una función computable y los resultados de un trabajo matemático realizado por el alemán George Cantor hace unos cincuenta años .

Como se sabe, Cantor argumentaba que aunque no existe un número tan grande, cualquier conjunto infinito de objetos que pueda ser contado o relacionado con números enteros positivos, es un conjunto que tiene el mismo tamaño que el conjunto de los números enteros. Debido a que cualquier máquina de Turing se puede expresar como una cadena de caracteres de longitud finita, todas las posibles máquinas de Turing y, con ellas, todas las funciones computables pueden ser listadas en orden numérico o alfabético; de manera que también pueden ser relacionadas uno a uno con el conjunto antes mencionado. No hay por supuesto, un límite superior fijo del tamaño de la máquina de Turing, de forma que no hay un límite en el número de posibles máquinas de Turing.[12]

---

[11] Parte de la dificultad de responder esta cuestión fue la de decidir lo que se entendía por un "procedimiento mecánico". El concepto estaba más allá de las ideas matemáticas de la época. Para entenderlo, Turing trató de imaginar cómo el concepto de "máquina" podría ser formalizado, y esta operación la dividió en términos elementales. De esta forma este tipo de cuestiones estaban abiertas por primera vez para demostrarlas o para afirmar que no tienen demostración, porque ya se sabía lo que era un método mecánico.

[12] Sin embargo, el análisis de Cantor muestra que el conjunto de todas las posibles funciones computables tienen el mismo tamaño que el de todos los números enteros; ambos son llamados conjuntos contables ó enumerables.

En base a las demostraciones de Cantor sobre los conjuntos infinitos que no son contables, son más grandes que los conjuntos contables; y suponiendo que las funciones son elementos de un conjunto, podemos ver que existen más funciones que números enteros. Por lo tanto no todas las funciones son computables. Luego entonces demostraba la insolubilidad al problema de Hilbert.[13]

[13]Church también demostró que no existía un método mecánico para decidir la verdad ó falsedad en el cálculo de predicados. Este fue llamado el teorema de la indecibilidad para el cálculo de predicados. Desde entonces se ha encontrado un número de resultados no-decidibles en todas las matemáticas.

CAPITULO II.- MAQUINAS DE TURING

II. MAQUINAS DE TURING

2.1. LA MAQUINA DE TURING

La formulación del *concepto de algoritmo general* data desde Euclides. Tratando de hacer precisiones se han dado varias descripciones alternativas de este concepto, todas en la década de los treinta. La más directa y persuasiva, también la históricamente más importante, es mediante el concepto de la Máquina de Turing. (M.T.). Este modelo matemático fue introducido en 1935 por Alan Turing con el objeto de resolver el problema de "Entscheidungsproblem" propuesto por David Hilbert en 1900.[1]

Una máquina de Turing consiste de una cinta ilimitada y control de estados finito. La cinta almacena los datos, un alfabeto. La máquina tiene un conjunto pequeño de operaciones propias: leer, escribir, e interpretar operaciones sobre la cinta. La operación de lectura no es una operación de datos, pero provee de ramas condicionales para un estado de control en función del dato que está debajo de la cabeza lectora (ver figura 2.1).

Como se sabe, este modelo contiene la esencia de todas las computadoras en términos de lo que ellas pueden hacer, aunque otras computadoras con diferentes memorias y operaciones pueden llevar a cabo los mismos cálculos con diferentes requerimientos de espacio y tiempo.

Modelo Físico

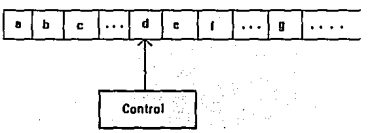


Figura 2.1. Modelo de la Máquina de Turing.

Un movimiento de la Máquina de Turing depende del símbolo que se encontró en la cinta y el estado de control. Las operaciones básicas que realiza son:

1. Cambia de estado.
2. Imprime un símbolo sobre la celda localizada, remplazando lo que está escrito en la celda.
3. Mueve su cabeza de lectura de izquierda a derecha sobre la celda.

[1] Hilbert preguntaba nada menos que sobre la existencia de un procedimiento algorítmico general que resolviera cuestiones matemáticas o, en otras palabras, por una respuesta a la cuestión de cuándo tal procedimiento podría en principio existir o no.

Modelo Matemático

Formalmente una Máquina de Turing se denota por una héptupla, (ver [Hopc79]) $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$. Donde:

Q:es el conjunto finito de estados.
Σ :es el conjunto de los *símbolo de entrada* (un subconjunto de Γ, que no incluye B),
Γ: es el conjunto de símbolos disponibles en la cinta.
δ : es la próxima función de movimiento, un mapeo de $Q\times\Gamma$ *a* $Q\times\Gamma\times\{L,R\}$ (δ puede estar indefinido para algunos argumentos.)
q_0: es el estado inicial de Q.
B: es un símbolo de Γ , es el blanco.
$F\subseteq Q$ es el conjunto de estados finales.

Así pues, uno puede definir máquinas de Turing que ejecuten aritmética o simples operaciones lógicas; máquinas que ejecuten tareas complicadas de naturaleza algorítmica. Una máquina de este tipo puede ejecutar cualquier operación mecánica. Matemáticamente definiríamos a una operación mecánica como un evento que puede llevar a cabo tal tipo de máquinas, en términos como: algoritmo, computabilidad, recursividad.

De esta manera un procedimiento claro y mecánico, podrá ser ejecutado por una máquina de Turing.[2]

> *¿Se podría seguir con la interrogante de cuándo el concepto de una máquina de Turing incorpora realmente operaciones lógicas o matemáticas que se pudieran considerar como mecánicas?*

Turing creyó necesario incluir un apreciable detalle, fortalecía su argumento encontrando soportes adicionales en el hecho de que (un poco antes y de manera independiente) el lógico americano Alonzo Church (con ayuda de S. C. Kleene) había creado un esquema -el cálculo lambda- que también ayudaba a resolver el "Entscheidungsproblem" de Hilbert. Aunque éste era mucho menos obvio que el esquema mecánico completamente comprensible de Turing, tenía la ventaja de tener la economía que brinda la estructura matemática. De manera independiente hubo otras propuestas para resolver el problema de Hilbert; particularmente la del lógico Polaco-Norteamericano Emil Post que las presentó un poco después de Turing, pero con ideas más afines a él que a Church.

Poco después se encontró que todos estos esquemas eran completamente equivalentes. Esto aportaba mucha fortaleza al punto de vista que después sería conocido

[2]Un procedimiento claro y mecánico es aquel que puede ser descrito sin ambigüedades en sus mínimas partes, donde las mínimas partes, es el tamaño de la descripción que se puede ejecutar.

como la Tesis de Church-Turing y que consistía en el concepto de la máquina de Turing y la definición matemática del significado de un procedimiento (efectivo, recursivo o mecánico) algorítmico.

Los números que pueden generarse mediante una máquina de Turing son llamados computables (Turing 1937). Aquellos que no se pueden producir (la vasta mayoría) son llamados no computables. Esto tiene relevancia en relación a la cuestión de cuándo un objeto físico (por ejemplo un cerebro humano) puede, de acuerdo a nuestras teorías físicas, ser descrito adecuadamente en términos de estructuras matemáticas computables.

Se pueden tener máquinas de Turing las cuales operen directamente sobre fórmulas matemáticas tales como expresiones algebráicas o trigonométricas, o las cuales lleven a cabo manipulaciones formales de cálculo. Todo lo que se necesita es precisar de alguna forma la codificación de la secuencia de ceros y unos, de todos los símbolos que estén involucrados, y entonces se puede aplicar el concepto de una máquina de Turing. Esto, después de todo era lo que Turing tenía en mente al atacar el "Entscheidungsproblem", enfocado a encontrar un procedimiento algorítmico que diera respuesta a cuestiones matemáticas de naturaleza general.

MAQUINA UNIVERSAL DE TURING

La idea básica de una máquina universal de Turing es codificar la lista de instrucciones de una máquina de Turing T con 0's y 1's para poder ser representado en la cinta. Esta cinta se usa como la parte inicial de la entrada de alguna máquina particular de Turing U - llamada una máquina universal de Turing- la cual actúa sobre el resto de la entrada, de la misma forma en que una máquina de Turing T lo haría. La parte inicial de la cinta le da a la máquina universal de Turing la información completa que necesita para imitar a cualquier máquina T exactamente. De esta forma la máquina de Turing es una generalización de todas las máquinas T de Turing.

Todas las computadoras modernas son en efecto, máquinas universales de Turing. Esto no significa que el diseño lógico de tales computadoras necesariamente adopte de una manera estrecha el tipo de descripción de una máquina universal de Turing dada. El punto es que simplemente supliendo cualquier máquina universal de Turing mediante el programa apropiado (la parte inicial de la cinta), se puede imitar el comportamiento de ésta o cualquier máquina de Turing.

2.2. COMPUTABILIDAD Y MAQUINAS DE TURING

Turing llegó a conceptos tales como algoritmo, computabilidad, recursividad al intentar solucionar el problema de Hilbert. Como se sabe Hilbert en su problema cuestionaba si había algún procedimiento mecánico que pudiera responder a todos los problemas matemáticos, tan extensos como se quisieran.

La forma en cómo Turing planteó una respuesta a dicha cuestión fue en términos de resolver el problema de decidir cuándo la n-ésima máquina de Turing podría pararse o

no cuando actuaba sobre el número m. Este problema se conoce como el problema de la detención ("Halting problem"). Es fácil construir una lista de instrucciones para las cuales la máquina no se detiene para cualquier número m. También existen muchas listas de instrucciones para las cuales la máquina siempre se detiene, cualquiera que sea el número dado; y algunas máquinas podían parar para algunos números pero para otros no. Se podría decir que un algoritmo no es de gran utilidad cuando éste "corre" siempre sin pararse. De hecho no es un algoritmo realmente, siendo lo importante la capacidad de saber, cuando t_n aplicado a m dará una respuesta o no.

Sería de una grandísima importancia para las matemáticas el ser capaces de saber cuando una máquina se detiene. Por ejemplo, cuando se está procesando la siguiente ecuación.

$$(x+1)^{w+3} + (y+1)^{w+3} = (z+1)^{w+3}{}_3$$

Esta ecuación en particular es uno de los famosos problemas en matemáticas a los que no se les había encontrado solución[4] (quizás el más famoso de todos). El problema es el siguiente: ¿existe algún conjunto de números w,x,y,z para los cuales la ecuación se satisface?. La famosa aseveración conocida como "El último teorema de Fermat", hecha por Pierre Fermat (1601- 1665); a la cual él dijo haber encontrado la demostración de que la ecuación nunca se satisface, pero que no escribió por que el margen del libro era muy estrecho para contenerla.

Es claro que dado un cuádruple de números (w,x,y,z) sea de interés para la computación decidir cuando la ecuación se cumple o no. Entonces se podría imaginar un algoritmo por computadora el cual corra a través de todas las combinaciones de cuádruples de números uno después de otro, y se detenga solamente cuando la ecuación se satisfaga. Si se establece que este algoritmo termina o que no terminará, entonces se tendrá la prueba de la aseveración de Fermat.[5]

De una manera muy similar es posible establecer muchos problemas matemáticos sin solución en términos del problema de que pueda detenerse una máquina de Turing.

---

[3]La razón por la que se escribió '$x+1$' , '$w+3$', '$y+1$','$z+1$', en lugar que la forma más familiar ($x^w+y^w=z^w; x,y,z>0, w>2$) de la aseveración de Fermat, es que con ello se permiten considerar a todos los números enteros para x,w, etc., empezando desde cero.

[4]La demostración del Ultimo Teorema de Fermat se acaba de hacer a mediados de 1993 dicha demostración la logró relizar el matemático inglés Andrew Wiles. Vale la pena mencionar que los mayores genios lo habían intentado sin exito y el teorema tuvo que esperar más de tres siglos para que por fin se pudiera escribir QED (Quod Erat Demonstrandum)

[5]Sabemos que hay formas de codificar conjuntos finitos de números en una forma computable, sobre la cinta, es decir, simplemente como números, de tal manera que se pueda "correr a través de ellos" todos los cuádruples siguiendo precisamente el orden natural de los números.

Surge una pregunta natural: ¿Cómo saber cuándo puede detenerse o no cualquier máquina de Turing en particular (con una entrada suficientemente especificada)? Para muchas máquinas de Turing esta cuestión podría no ser tan difícil de responder, aunque ocasionalmente como se vio con anterioridad, la respuesta podría involucrar la solución de un gran problema matemático. De la misma forma se podría cuestionar: ¿hay procedimientos algorítmicos para responder cualquier problema en general? - El problema de la detención- de una forma completamente automática, Turing mostró que no la hay.

Su argumento fue esencialmente el siguiente. Primero se supone lo contrario, es decir que tal algoritmo existe, entonces debe existir una máquina de Turing H la cual "sabe" cuándo se detiene o no la n-ésima máquina de Turing, cuando actúa sobre un número m. Digamos que la salida de la cinta sea un 0 cuando no se detiene y un 1 cuando sí lo hace. Si se imagina un arreglo infinito, el cual lista todas las salidas de todas las posibles máquinas de Turing que actúan sobre todas las diferentes entradas, el n-ésimo renglón del arreglo despliega la salida de la n-ésima máquina de Turing, cuando se aplica a varias entradas: 0,1,2,3,4,5,6...:

$m \rightarrow$	0	1	2	3	4	5	6	7	8	. . .
$n \downarrow$										
0	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	. . .
1	0	0	0	0	0	0	0	0	0	. . .
2	1	1	1	1	1	1	1	1	1	. . .
3	0	2	0	2	0	2	0	2	0	. . .
4	1	1	1	1	1	1	1	1	1	. . .
5	0	⊕	0	⊕	0	⊕	0	⊕	0	. . .
6	0	⊕	1	⊕	2	⊕	3	⊕	4	. . .
7	0	1	2	3	4	5	6	7	8	. . .
8	⊕	1	⊕	⊕	1	⊕	⊕	⊕	1	. . .
.	.				.				.	. . .
.	.				.				.	
.	.				.				.	
197	2	3	5	7	11	13	17	19	23	. . .
.	.				.				.	
.	.				.				.	
.	.				.				.	

a n-ésima máquina de Turing.

El símbolo ⊕ se usa para denotar los cálculos indeterminados para la n-ésima máquina de Turing. Las ocurrencias de ⊕ podrían causar dificultades si se tratara de calcular el arreglo, ya que no se sabe con seguridad cuando poner un ⊕ en alguna posición donde esos cálculos simplemente corren indefinidamente.

Sin embargo, se podría proveer un procedimiento de cálculo para generar la tabla, si se permite el uso de H, de tal forma que H se active cuando ocurría el ⊕, pero en lugar de eso, se va a usar H para eliminar cualquier ⊕ reemplazando su ocurrencia por 0. Esto se logra precediendo la acción de T_n sobre m por el cálculo $H(n;m)$; de forma que se permita a T_n actuar sobre m solamente si $H(n;m)=1$, es decir solamente si el cálculo de $T_n(m)$ da una respuesta, y simplemente escribir 0 si $H(n;m)=0$ es decir, si $T_n(m)=\oplus$. Se puede escribir este nuevo procedimiento (es decir la obtención de preceder a $T_n(m)$ mediante la acción de $H(n;m)$) como :

$$T_n(m) \times H(n;m)^{6}$$

La tabla quedará ahora de esta manera:

[6] Aquí se está usando una convención matemática común acerca del orden de las operaciones matemáticas: el término que está más a la derecha es el que termina primero. Téngase en mente que simbólicamente $\oplus \times 0 = 0$.

n ↓ / m →	0	1	2	3	4	5	6	7	8	.	.	.
0	0	0	0	0	0	0	0	0	0			
1	0	0	0	0	0	0	0	0	0			
2	1	1	1	1	1	1	1	1	1			
3	0	2	0	2	0	2	0	2	0			
4	1	1	1	1	1	1	1	1	1			
5	0	0	0	0	0	0	0	0	0			
6	0	0	1	0	2	0	3	0	4			
7	0	1	2	3	4	5	6	7	8			
8	0	1	0	0	1	0	0	0	1	.	.	.
.	.				.				.			
.	.				.				.			
.	.				.				.			

Asumiendo que H existe, los renglones de esta tabla están en una **secuencia computable.** Es decir, es una secuencia finita cuyos valores sucesivos pueden ser generados por un algoritmo; es decir, existe una máquina de Turing la cual, cuando se aplica a los números naturales $m = 0, 1, 2, 3, 4, 5, \ldots$, genera los números sucesivos de dicha secuencia.

Se pueden hacer notar dos hechos acerca de esta tabla. En primer lugar, cualquier secuencia computable de números naturales puede aparecer algunas veces (quizá muchas) a través de los renglones. Esta propiedad también era válida para la tabla original con los $\oplus$. Se han agregado simplemente algunos renglones para remplazar las máquinas de Turing "dudosas" (es decir las que producen al menos un $\oplus$).

En segundo lugar, el supuesto que se ha hecho de que la máquina de Turing H realmente existe, y la tabla ha sido generada de una forma computable (es decir, generada por un algoritmo definido), por el procedimiento $T_n(m) \times H(n;m)$. Esto es, que existe una máquina de Turing Q la cual, cuando procede sobre un par de números (n,m), produce el elemento apropiado en la tabla; por esto, se puede codificar n y m sobre la cinta de Q de la misma forma que H lo hizo, y entonces se tiene

$$Q(n;m) = T_n(m) \times H(n;m).$$

Ahora se aplicará una variante de una ingeniosa y poderosa herramienta; "la diagonal de George Cantor". Considérese los elementos de la diagonal principal, aquellos elementos de un tamaño ligeramente mayor al resto.

0	0	0	0	0	0	0	0	0
0	**0**	0	0	0	0	0	0	0
1	1	**1**	1	1	1	1	1	1
0	2	0	**2**	0	2	0	2	0
1	1	1	1	**1**	1	1	1	1
0	0	0	0	0	**0**	0	0	0
0	0	1	0	2	0	**3**	0	4
0	1	2	3	4	5	6	**7**	8
0	1	0	0	1	0	0	0	**1**
.				.				.
.				.				.
.				.				.

Estos elementos tienen la secuencia 0,0,1,2,1,0,3,7,1,...; a cada uno de los términos se le suma 1 Entonces se tiene: 1,1,2,3,2,1,4,8,2,.... Esto es claramente un procedimiento computable y, dado que la tabla fue generada de una forma computable, esto otorga una nueva secuencia computable. En efecto, con la secuencia $1+Q(n;m)$, es decir.

$$1+T_n(n)\times H(n;n)$$

(donde la diagonal corresponde a las posiciones en que $m=n$). Como la tabla contiene cualquier secuencia computable, la nueva secuencia debe estar en cualquier lugar de la tabla; sin embargo esto no se da. La nueva secuencia difiere del primer renglón en el primer elemento, del segundo renglón en el segundo elemento, del tercer renglón en el tercer elemento, y así sucesivamente. Esto es claramente una contradicción. Esta es la condición que establece lo que se ha tratado de probar, y es el hecho de que ¡la máquina de Turing H no existe!. Es decir: **¡No existe un algoritmo universal para saber si la máquina de Turing se detendrá o no!.**

Otra manera analítica de llegar a esto es el hecho de notar que, sobre el supuesto de que H existe, hay alguna máquina de Turing numerada con k, para el algoritmo $1+Q(n;m)$, de forma que

$$1+T_n(n)\times H(n;n)=T_k(n).$$

Si se sustituye $n=k$ en la relación, que significa lo mismo que el proceso de la diagonal

$$1 + T_k(k) \times H(k;k) = T_k(k)$$

Lo cual es una contradicción pues si $T_k(k)$ se detiene se llega a una relación imposible

$$1 + T_k(k) = T_k(k)$$

(Dado que $T_k(k)$ se detiene bajo $H(k;k) = 1$),

Por otro lado, si $T_k(k)$ no se detiene $(H(k;k) = 0)$ se tiene una desigualdad inconsistente:

$$1 + 0 = \oplus$$

La cuestión de cuándo se detiene o no una máquina de Turing en particular es una pieza de las matemáticas. (Ya hemos visto que varias interrogantes con significado matemático pueden ponerse en términos de la detención de las máquina de Turing). De esta manera se muestra que un algoritmo existe, decidiendo la cuestión de la detención de las máquinas de Turing. Turing mostró (como Church lo hizo, usando su propio y diferente tipo de enfoque) que no hay un algoritmo general para decidir cuestiones matemáticas. **¡El Entscheidungsproblem de Hilbert no tiene solución!**

Esto no quiere decir que en cualquier caso individual no se pueda ser capaz de decidir la verdad o lo contrario de algunas cuestiones matemáticas en particular; o decidir si alguna máquina de Turing pudiera llegar a detenerse o no. Mediante el ingenio, o simplemente del sentido común, se podrá ser capaz de decidir si se detiene o no en un caso dado. Por ejemplo, si la lista de instrucciones de una máquina de Turing no contiene órdenes de detenerse, o contiene sólo órdenes de detenerse, entonces el sólo sentido común es suficiente para decir si se detiene o no. Pero no existe un algoritmo que trabaje para todas las cuestiones matemáticas o, en otros términos, para todos las máquinas de Turing y todos los números sobre los que ellos actúan.

CAPITULO III.- CALCULO LAMBDA DE CHURCH Y FUNCIONES RECURSIVAS

III. CALCULO LAMBDA DE CHURCH Y FUNCIONES RECURSIVAS

Este capítulo tiene como objetivo mostrar el enfoque del cálculo lambda de Church y cómo la computabilidad esta relacionada con él.

Para esto, se presentan tres conceptos básicos derivados de los trabajos de Church y Turing:

a) Las Funciones Recursivas (*R*)

b) Las Funciones Abacus (*A*)

c) Las Funciones Computables o de Turing (*T*)

En términos generales, como su nombre lo indica las **funciones recursivas** se definen como las funciones que siguiendo un patrón recursivo[1] pueden representar a un sistema matemático[2], Las **funciones abacus** se definen en términos de diagramas de flujo Abacus[3].Las **funciones de Turing** tiene como base de construcción el modelo de la Máquina de Turing.

Estos tres conceptos básicos permiten mostrar cómo las funciones del cálculo lambda de Church, que son las mismas que las funciones recursivas, pueden representarse en términos de procesos computables y, por lo tanto el propio sistema matemático creado a través de las funciones recursivas puede también ser computable.

Es decir, que las funciones recursivas pueden representar a un sistema matemático, y a su vez los términos en que se representa el sistema matemático se pueden trasladar a los diagramas de flujo que definen a las funciones Abacus; diagramas que pueden ser expresados en términos de la máquina de Turing y, por lo tanto procesados por estas[4].

El desarrollo de estos planteamientos está organizado de la siguiente manera:

- Primero, se muestra el cálculo lambda de Church que es el antecedente que define a las funciones recursivas, y

---

[1]Que estan definidas básicamente por el cálculo lambda y la recursividad que planteó Church

[2]Por sistema matemático se considerará a la aritmética de los números naturales o cualquier otro.

[3]Una función Abacus es un programa que se encuentra representado por un diagrama de flujo, conocido como diagrama de flujo Abacus.

[4]En otras palabras: las funciones recursivas representan a un sistema en términos que pueden ser representados por lenguajes de programación y procesados por computadoras.

- Segundo se muestra cómo las funciones recursivas se pueden definir en términos de las funciones Abacus ($R \subseteq A$) y estas, a su vez, en términos de las funciones de Turing ($A \subseteq T$)[5].

En la primera parte se muestra como se representa un sistema matemático en términos del cálculo lambda de Church, en este caso se consideran los números enteros no negativos 0,1,2,3,... y algunas operaciones entre ellos. A nivel de ejemplificación, se presenta la definición de las operaciones tales como: el incremento de 1 a un número; la multiplicación por dos de un número; la suma de dos números; la multiplicación de dos números; y la exponenciación.

En la segunda parte se presenta una definición de las funciones recursivas, Abacus y Turing y, a partir de estas definiciones, se muestran las forma en que es posible transformar, por un lado, a la función recursiva en Abacus y, por el otro, a la función Abacus en Turing.

3.1. EL CALCULO LAMBDA DE CHURCH

El procedimiento de Church es completamente diferente y distintivamente más abstracto que el de Turing. Es importate dar una breve descripción del esquema de Church no solamente porque este esquema enfatiza que la computabilidad es una idea matemática, sino también porque ilustra el poder de las ideas abstractas en las matemáticas.

En este esquema se utiliza "universo" de objetos, denotado por: $a,b,c,d,\ldots,z,a',b',\ldots,z',a'',b'',\ldots,a''',\ldots,a''',\ldots$

cada uno de los cuales representa una operación matemática o función, y los argumentos de estas funciones son argumentos del mismo tipo, es decir, también son funciones. A su vez el resultado de una de estas funciones que actúa sobre otra será también una función. Hay, dentro de esto, una maravillosa economía de conceptos.

En el Sistema de Chuch se escribe:

$a = bc$[6]

---

[5] De hecho estos tres conceptos son equivalentes ($R \cong A \cong T$) pues, además de que $R \subseteq A$ y $A \subseteq T$, también las funciones Turing se pueden definir en términos de las funciones recursivas $T \subseteq R$. Si se desea ver como $T \subseteq R$ consultar [Bool89].

[6] Una notación más familiar es haber escrito esto como $a = b(c)$, pero este paréntesis en particular no es realmente necesario y es mejor omitirlo. Si se incluye, por consistencia nos llevaría a crear fórmulas como $(f(p))(q)$ y $((f(p))(q))(r)$, en lugar de $(fp)q$ y $((fp)q)r$ respectivamente.

que indica que el resultado de la función b que actúa sobre la función c es la función a.

Para una función f de dos variables p y q, se puede escribir $(fp)q$ que es el resultado de la función fp aplicado a q; para una función de tres variables se considera: $((fp)q)r$ y así sucesivamente.

El Cálculo lambda de Church está definido por $\lambda x.[fx]$. Se usa la letra griega λ (lambda) seguida de una letra que hace las veces de una función de Church y en seguida se denota cualquier ocurrencia de la variable x mediante una expresión encerrada entre corchetes.

Entonces: $(\lambda x.[fx])a = fa$ y $\lambda x.[fx] = f$.

Por ejemplo la función seno definida en términos del cálculo lambda es: $\lambda x.[sen\, x] = sen$. Y como

$$\text{sen}\, x = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \ldots$$

entonces se puede definir: $sen = \lambda x\left[x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \ldots\right]$.

Ahora se va a mostrar cómo se definen los enteros no negativos a partir de las funciones de Church. Como se acaba de ver $\lambda x.[fx]$ permite definir funciones de x, donde λx indica la variable que se sustituye en la función que está en términos de x. Entonces, para definir funciones en términos de funciones se establece a la función de Church como $\lambda f.[f(fx)]$.. En este caso, λf indica la función que se va a sustituir en la función que está en términos de funciones. Por lo tanto, si tenemos que g es una función que actúa doblemente sobre un argumento x, se tiene que $(\lambda f.[f(fx)])g = g(gx)$.

De la misma manera, se puede definir a λ tanto para funciones como para variables (λfx) lo cual da $\lambda f.[\lambda x.[f(fx)]]$ o, de manera abreviada, $\lambda fx.[f(fx)]$. Esto permite reemplazar una función en la función definida en términos de funciones y sustituir la variable x en las funciones que están en términos de esta variable. Esto representa la misma doble actuación de una función sobre su argumento, con la diferencia de que x también es reemplazable en caso de que se requiera. A su vez, también se puede definir una función de Church que representa tres veces la función sobre un argumento y así indefinidamente.

Es sobre la base que se definen los números enteros no negativos. El cero se define como la función de Church en la que no actúa ninguna función sobre el argumento; el uno se define como la función en la que actúa una función sobre su argumento; el dos lo define la función de Church que actúa doblemente sobre su argumento (la función de la función de x); el número tres se define como la función de Church en notación lambda de Church que actúa tres veces sobre su argumento (la función de la función de la función de x), y así sucesivamente:

$$0 = \lambda fx.[x]$$
$$1 = \lambda fx.[fx]$$
$$2 = \lambda fx.[f(fx)]$$
$$3 = \lambda fx.[f(f(fx))]$$
$$\vdots$$
etc.

Entonces, dado que el '2' de Church es "el doble ", '3' es "triple", etc. Entonces, la acción de 3 sobre una función, digamos $3f$, es la operación de 'iterar tres veces f'.

La acción de $3f$ sobre "y" podría ser por lo tanto $(3f)y = f(f(f(y)))$.

Una vez que se vio la definición de los números enteros no negativos, se verá como se definen las operaciones entre ellos. En concreto se mostrará la operación de incrementar en uno y la operación de multiplicar por dos. En cada uno de estos se hará el desarrollo para ilustrar cómo operan y, finalmente, sólo se mostrará cómo se definen las operaciones de suma entre dos números, la multiplicación y la exponenciación en términos del cálculo lambda.

La operación aritmética de incrementar en uno a un número entero positivo, se expresa en el esquema de Church de la siguiente manera:

$$S = \lambda abc.[b((ab)c)].$$

Supóngase que quiere sumarse un 1 al 3, entonces

$$S3 = \lambda\, abc.\left[b((ab)c)\right]3$$

$$S3 = \lambda\, bc.\left[b((3b)c)\right] \text{ y como } 3b = b(b(bc))$$

$$S3 = \lambda\, bc.\left[b(b(b(bc)))\right] = 4$$

La operación aritmética de multiplicar un número entero por dos se expresa en el esquema de Church como:

$$D = \lambda\, abc.\left[(ab)((ab)c)\right],$$

Supóngase que quiere multiplicar al número 3 por 2, entonces:

$$D = \lambda\, abc.\left[(ab)((ab)c)\right]3 = \lambda\, bc.\left[(3b)((3b)c)\right]$$

$$D = \lambda\, bc.\left[(3b)(b(b(bc)))\right] = \lambda\, bc.\left[b(b(b(b(b(bc)))))\right] = 6.$$

La definición de las operaciones de la adición, multiplicación y exponenciación en términos de las funciones de Church para los números naturales (donde m y n son este tipo de funciones) se definen como:

Adición: (Am)n=m+n donde

$$A = \lambda\, fgxy.\left[((fx)(gx))y\right]$$

Multiplicación: (Mm)n=mxn donde

$$M = \lambda\, fgx.\left[f(gx)\right]$$

Exponenciación: (Em)n= n^m donde

$$E = \lambda\, fg.\left[fg\right]$$

De esta misma manera, se pueden definir todas las operaciones que sean necesarias y por lo tanto representar un sistema matemático.

3.2.. FUNCIONES RECURSIVAS, ABACUS Y COMPUTABLES

Una vez que ya se vio cómo se representa un sistema matemático en términos de funciones lambda de Church, se mostrará cómo se pueden transformar las funciones recursivas en su equivalente a funciones Abacus ($R \subseteq A$) y éstas a su vez en su equivalente a funciones de Turing ($A \subseteq T$).

LAS FUNCIONES RECURSIVAS

Aquí se presenta la definición explícita de las funciones recursivas (R), y se mostrará que todas las funciones en R pertenecen a las funciones Abacus computables (A), de tal manera que : $R \subseteq A$.

Las **funciones recursivas** se definen como un grupo inicial de funciones que pertenecen a la clase R, las que sujetas a dos operaciones (composición y recursividad) dan como resultado funciones que pertenecen también a la clase R. Nada pertenece a la clase R a menos que se encuentre en el grupo inicial de funciones, o que se hayan obtenido a partir de una secuencia finita de aplicaciones de estas dos operaciones sobre el grupo inicial de funciones, sin importar el orden en que se apliquen.

El grupo inicial de funciones consiste en la función cero, la función sucesora y la función identidad. La función cero, z, es una función de un argumento. Para cualquier número natural que se tenga como argumento, da el valor de cero: $z(x) = 0$.

Por lo que $z(0) = z(1) = z(2) = \ldots = 0$.

y la notación con funciones recursivas es:

$$z(0) = z(s(0)) = z(s(s(0))) = \ldots = 0.$$

La función sucesora, s, que tiene como un argumento a un elemento, dá como resultado al sucesor de su argumento, de manera que: $s(0) = 1$, $s(1) = 2$, $s(2) = 3$,...en notación de funciones recursivas: $s(0) = 1$, $s(s(0)) = 2$, $s(s(1)) = s(s(s(0))) = 3$...[7]

[7]Recuerdese que los números 0,1,2,3,... etc. están definidos por 0, $s(0)$, $s(s(0))$, $s(s(s(0)))$,... etc

La función identidad está definida por:

$$\mathrm{id}_i^n(x_1,\ldots,x_i,\ldots,x_n) = x_i.$$

donde para cualquier número entero positivo, se toman los valores del primero, segundo, ... , n-ésimo argumento.

A partir de este grupo de funciones iniciales, se pueden formar nuevas funciones recursivas, mediante operaciones de dos tipos: la composición y la recursividad.

El primer tipo de operación, denominado **composición**, se define de la siguiente manera: Si f es una función de m argumentos y, a su vez, cada uno de sus argumentos $g_1,\ldots,g_m$, son funciones de n argumentos, entonces la función h,

$$h(x_1,\ldots,x_n) = f(g_1(x_1,\ldots,x_n),\ldots,g_m(x_1,\ldots,x_n)) \qquad \text{(Cn)}$$

es una función que se obtiene a partir de $f,g_1,\ldots,g_m$ mediante la composición. Esto puede indicarse como: $h = \mathrm{Cn}[f,g_1,\ldots,g_m]$

Como un ejemplo para mostrar esta operación, supóngase que la función h es una función de tres argumentos que da como resultado al sucesor de su tercer argumento. Entonces, esta función está definida a partir de la composición de un par de funciones: la función sucesora s y la función identidad: id_3^3.

La función $h = h[s, \mathrm{id}_3^3]$ en términos del formato Cn donde n=3 y m=1, se define como:

$h(x_1, x_2, x_3) = f(g_1(x_1, x_2, x_3))$, donde $f = s$, $g_1 = \mathrm{id}_3^3$

El segundo tipo de operación llamado **recursividad**, se define como:

$$h(x,0) = f(x), \qquad h(x,s(y)) = g(x,\ y,\ h(x,y)) \qquad \text{(Re)}$$

donde se dice que h se define por recursividad a partir de las funciones f y g. También se puede expresar como: $h = Re[f,g]$

El siguiente ejemplo muestra cómo trabaja esta operación.

La definición de la suma, donde $x+0=0,\ x+s(y)=s(x+y)$ puede expresarse en términos completamente funcionales reemplazando a '+' por la función '*suma*' de manera que se tendrá $suma(x,0)=x,\ suma(x,s(y))=s(suma(s,y))$.

Esto a su vez se puede transferir al formato de la operación de recursividad, en donde se definen las funciones f y g para las que $f(x)=x,\ g(x,y,-)=s(-)$, para todos los números enteros no negativos x y y -. El símbolo '-' se usa para denotar cualquier función.

Las funciones f y g se definen como: $f=\mathrm{id}, y\ g=\mathrm{Cn}[s,id_3^3]$, de donde f y g son recursivas. La función f es recursiva dado que está definida directamente a partir de la función recursiva inicial id, y g es recursiva dado que está definida a partir de la composición de dos funciones recursivas iniciales: la función s y la función id. Por lo tanto, dado que la suma está definida en términos de dos funciones recursivas f y g entonces la suma es a su vez recursiva. Así la *suma* entonces planteada en términos de (Re) se expresa como:

$$suma(x,0)=id(x),\ suma(x,s(y))=Cn[s,id_3^3](x,y,suma(x,y))$$

o $suma=Re[\mathrm{id},\ Cn[s,\mathrm{id}_3^3]]$.

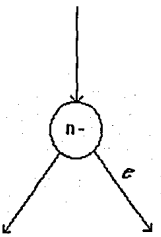
LAS FUNCIONES RECURSIVAS SON ABACUS .

A continuación se verificará que las funciones recursivas son Abacus. Para mostrar esto primero se define que es una función Abacus y a partir de ello se verá como se pueden representar en términos de las funciónes Abacus a las funciones Recursivas.

Una función Abacus es un programa que se encuentra representado por un diagrama de flujo, en el que se indica la secuencia de los pasos que se siguen para llegar a un resultado. Estos pasos se llevan a cabo mediante el auxilio de registros o cajas de almacenamiento en donde se almacenan los datos que se procesan y el resultado de los procesos. A partir de una función Abacus se puede definir cualquier tipo de operación. Para ilustrar esto se verá como se establecen algunas operaciones en términos de funciones Abacus.

Las funciones Abacus elementales son:

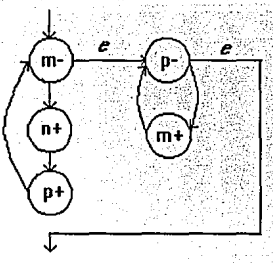
La cual indica la suma de 1 a el argumento de la caja *n*. Y



Significa que si la caja n esta vacía, el flujo continúa en la flecha que tiene a '*e*' como etiqueta; en caso contrario se vacía la caja *n*.

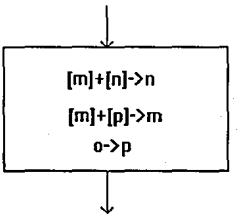
Otras operaciones son:

Vaciar la caja *m* en la caja *n* sin perder el valor de *m*. Este programa tiene una caja auxiliar *p* que se inicializa con 0.

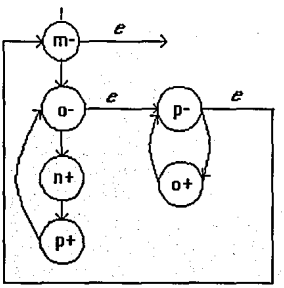


Se llenan las cajas *n* y *p* hasta que *m* se vacía, *m* se vuelve a llenar auxiliándose de la variable *p*, quedando al final *p* vacía.

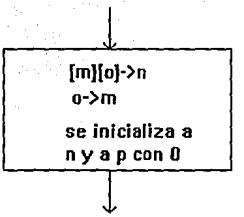
El diagrama de bloques que representa a este programa es:



La multiplicación de dos cajas, dígase *m* y o se define como:



cuyo diagrama de bloques correspondiente es:



en donde m y o son las cajas que tienen los números a multiplica; n es la caja en donde se almacenará el resultado; y p es una caja auxiliar de almacenamiento temporal.

Este programa presenta una estructura anidada donde n va acumulando el resultado de la multiplicación hasta que se vacia m, la variable p inicializa a o con su valor inicial, de manera que n va acumulando o tantas veces como el valor de m lo determina.

Una vez que se definieron las funciones Abacus y se dieron algunos ejemplos de operaciones con estas, se procederá a mostrar que $R \subseteq A$.

Esta prueba consiste en expresar las funciones recursivas iniciales: la función cero, la función sucesora, la función de identidad, así como a las operaciones de composición y recursividad en términos de los diagramas Abacus.

Primero se definirá qué es una función Abacus y después se hará la representación de las funciones recursivas en términos de Abacus.

Para mostrar cómo se expresan las funciones recursivas en términos de funciones Abacus se tienen dos consideraciones:

1. Al calcular una función de n argumentos sobre un Abacus, se especifican n registros o cajas en las que los argumentos se almacenan inicialmente y además una caja en la que el valor de la función aparecerá al final del cálculo.

2. Para facilitar la posterior comparación con los cálculos de la máquina de Turing, se asumirá que los argumentos aparecen en las cajas de la $1,...,n$ y el valor de la función aparecerá en la caja $n+1$.

Supóngase que al inicio del proceso todas las cajas están vacías, excepto las primeras n. Entonces, el cálculo de las funciones básicas está definido de la siguiente manera:

La función z (función cero) se calcula mediante un programa 'bobo' (ver Figura 3.1) el cual deja la caja siempre vacia[8]

Figura 3.1 La Función cero

La función *s* se calcula con el auxilio de dos cajas donde al inicio del proceso, la caja [1]=*x* tiene almacenado el argumento y la caja [2]=0 está vacía; al final del proceso la caja [2]=*s*(0) tiene el resultado de la función (ver figura 3.2):

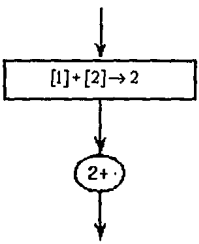


Figura 3.2 La Función Sucesora

La función identidad se calcula con el auxilio de una serie de cajas que van de [1], ... ,[*n*] y la caja [*n*+1]. Al inicio las cajas [1]=x_1, ... , [*n*]=x_n tiene los valores de los argumentos de la caja y la caja [*n*+1] tiene el valor del argumento del resultado de la función (ver figura 3.3).

[8]Las figuras que aparecen sólo de aquí hasta el final del capítulo se tomaron del libro [Bool89].

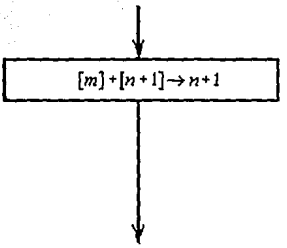
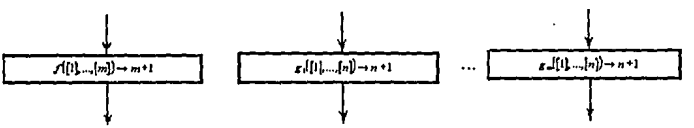


Figura 3.3 La Función identidad

Una vez que se han representado las funciones básicas se define la operación de composición. La representación de la función de composición se divide en dos partes.

1. Se definen los diagramas Abacus para las funciones $f, g_1, \ldots, g_m$ (ver figura 3.4)



La función f Las funciones $g_1, \ldots, g_m$

Figura 3.4 Las funciones f y $g_1, \ldots, g_m$

2. El diagrama Abacus que calcula sus composiciones, $h = C^n[f, g_1, \ldots, g_m]$ se muestra en la figura 3.5:

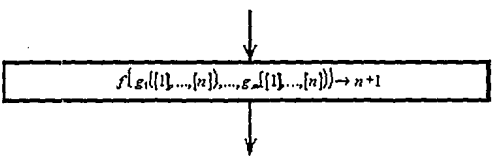
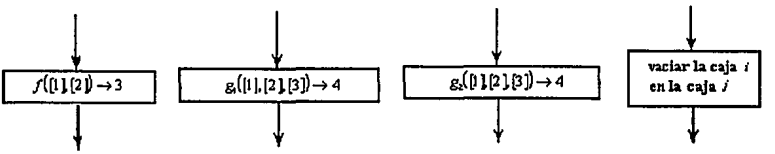


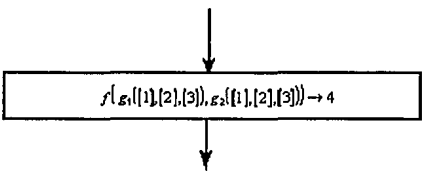
Figura 3.5 La función de composición

Para ilustrar la operación de la composición en términos de las funciones Abacus, considérese el siguiente ejemplo.

Supóngase que m=2 y n=3, entonces se tiene una función f y dos funciones g_1 y g_2 con tres argumentos cada una, por lo que se tienen tres diagramas que definen la función f a las funciones g_1 y g_2 y además un diagrama auxiliar para realizar movimientos de contenidos de cajas.



A partir de los cuales se obtiene el diagrama Abacus de la Composición.



Entonces la función de composición opera de la siguiente manera:

1. 1.Primero, se crean 9 áreas de almacenamiento (cajas): 5 temporales "p_1, p_2, q_1, q_2, q_3"; y 4 permanentes "[1],[2],[3],[4]" de estas últimas, las tres primeras almacenan los argumentos de la función y el cuarto almacenará el resultado de la función; la [4] al inicio estará vacía.

2. En la caja 4 se almacena el resultado de la función g_1 y se vacía en la caja p_1[9].

3. Se calcula la función g_2 y se almacena el resultado en la caja 4 y se vacía en la caja p_2.

4. Se vacían los argumentos [1],[2] y [3] en los registros temporales q_1, q_2, q_3.

5. Se vacían los valores de los registros temporales p_1 y p_2 en la caja [1] y [2], respectivamente.

[9]Vaciar la caja 4 en la caja p_1 quiere decir pasar lo que almacena la caja 4 a la caja p_1 e inicializar en 0 la caja 4.

6. Se evalúa la función *f* con los argumentos de las cajas [1] y [2] y el resultado se almacena en la caja [3].

7. Se vacía la caja 3 en la caja 4, se inicializa la caja 1 y la caja 2.

8. Los argumentos iniciales vuelven a ponerse en las cajas de origen, es decir [1], [2] y [3].

Esto genera al final los argumentos en [1], [2] y [3] y el resultado en la caja [4].

La Figura 3.6 muestra el diagrama detallado de la operación de composición.

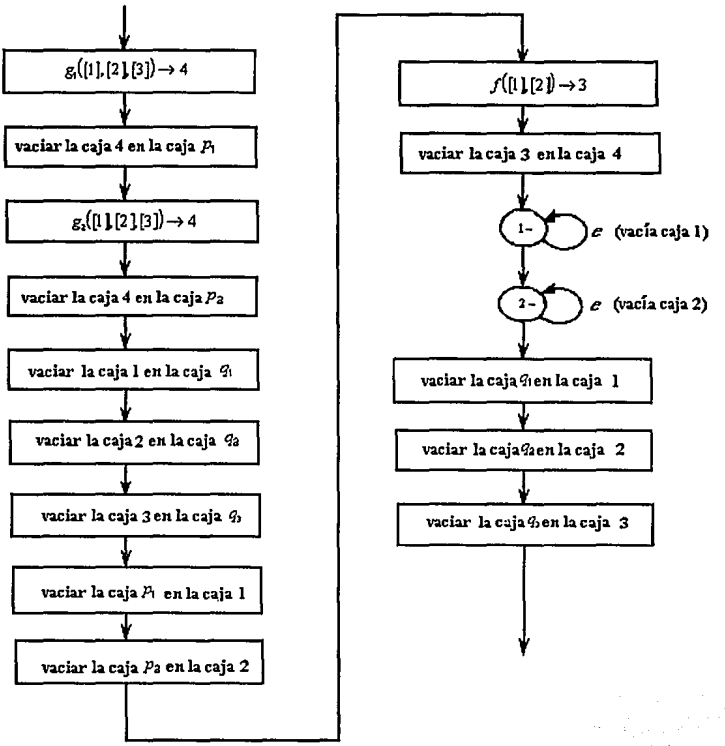
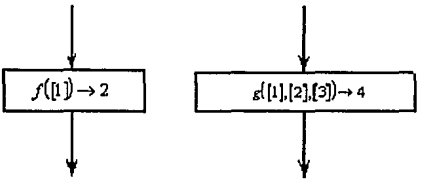


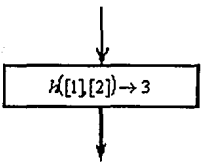
Figura 3.6 Diagrama Abacus detallada para el caso m=2 y n=3 de la operación de composición.

A continuación se verá la representación de la operación de recursividad en términos de funciones Abacus. Para ilustrar de una manera directa la representación de la operación de recursividad en términos de la función Abacus, se hará mediante un ejemplo.

Supóngase que h tiene dos argumentos[10], dígase que f y g donde f es una función de un argumento $f([1])$ y g una función de tres argumentos $g([1],[2],[3])$; estas funciones están representadas por las siguientes funciones Abacus:



y a $h = \mathrm{Re}[f, g]$ lo representa la gráfica



cuyo diagrama Abacus esta dado por la Figura 3.7.

Entonces la función de recursividad opera de la siguiente manera:

Las cajas [1]=x, [2]=y se inicializan con x y y, y el resto con 0 [3]=[4]=...=0. El registro p se crea para que cumpla la función de un contador de manera que determine el fin del proceso. Si $[p]$=0 el proceso se concluye; si $[p]\neq 0$ se le resta 1 a p y se procesa otra fase; p se inicializa con y.

Entonces:

1. Primero se cálcula $f(x)$.

2. Si $y = 0$ entonces $h(x, y) = h(x, 0) = f(x)$ y el cálculo se concluye con el resultado en la caja 3.

[10]Más de dos argumentos son tratados de una manera similar.

3. Si $y \neq 0$, se calcula sucesivamente a $h(x,1)$, $h(x,2)$,... hasta que $y = 0$, con el resultado de $h(x,y)$ en la caja 3.

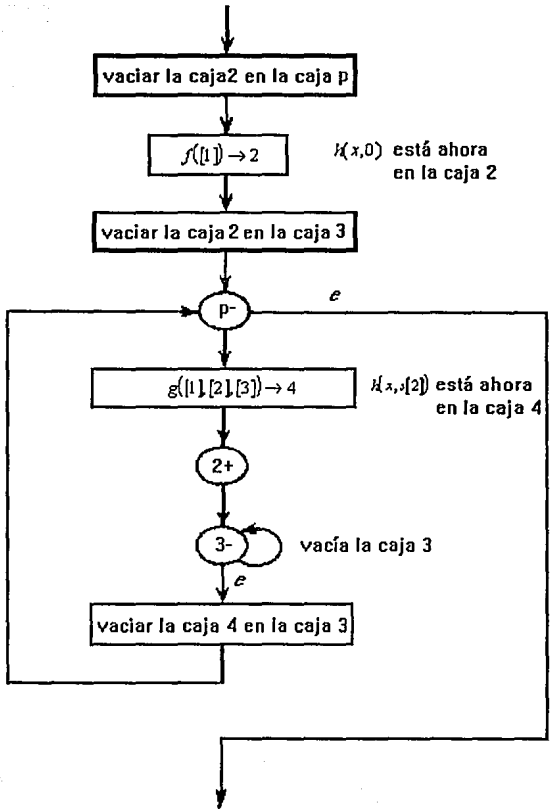


Figura 3.7 Función Abacus de la operación de recursividad.

Como se ha visto, es posible expresar las funciones recursivas primitivas básicas así como los la operacion de composición y recursividad en términos de cálculos abacus, de ahí que se concluye:

Todas las funciones recursivas son abacus.

LAS FUNCIONES ABACUS SON TURING

En esta parte se va a mostrar por que las funciones Abacus se pueden expresar en términos de las funciones Turing. La forma en que se desarrolla esto es:

1. Se define qué es una función Turing y
2. Se argumenta porque $A \subseteq T$.

Las funciones Turing son las funciones que pueden calcularse con Máquinas de Turing de acuerdo con las siguientes características:

1. Los números se representan mediante bloques de 1s, separados por un espacio en blanco[11], y el valor (el resultado del cálculo) se representa por un simple bloque de 1s en una cinta limpia, cuando la máquina se detiene.

2. La máquina comienza y se detiene solamente en la posición estándar, es decir en el 1 que al inicio del proceso estaba más hacia la izquierda de la cinta.

3 A través del proceso la máquina no puede avanzar más de dos cuadros hacia la izquierda del 1 que se encontraba al inicio del proceso más hacia la izquierda de la cinta. Por lo que la cinta es infinita en una sola dirección: hacia la derecha.

4 La máquina no lee ni escribe otros símbolos que no sean B's y 1's.

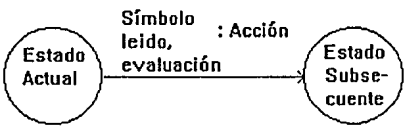
5 Según lo descrito en el capítulo anterior, la héptupla de la Máquina de Turing se forma de: dos componentes para describir la dirección del movimiento: I (izquierda) y D (derecha); 2 componentes para denotar los símbolos posibles de leer y escribir en la cinta B (espacios) y 1 (unos); y tres componentes para indicar los estados de la máquina; q_1, q_2 y q_3. Así , la héptupla estará dada por $(I, D, B, 1, q_1, q_2, q_3)$.

5 El resultado quedará en la misma posición en donde empezó a trabajar la máquina, es decir, en el 1 más a la izquierda en el bloque de 1s que representa el valor (o resultado) de la función.

[11] En computación un espacio en blanco es realmente un caracter: el caracter B.

Las características 2 y 4 señaladas antes no son indispensables. Sin embargo, la segunda es un supuesto sencillo que permite definir sin mayor complicación el inicio y término de la máquina; mientras que la cuarta -notación monádica- facilita la operación en una Máquina de Turing, en contraste con el uso de notación decimal o cualquier otro tipo de notación.

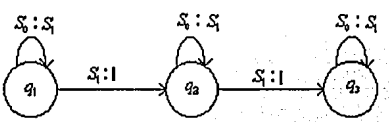
Las operaciones que se realizan mediante una máquina de Turing tiene varias formas de representación, una de ellas es mediante diagramas. El diagrama se define básicamente con la siguente estructura:



Así, los estados son son la consecuencia de posiciones e instrucciones que se realizan considerando el símbolo accesado. Las instrucciones pueden ser escribir un símbolo, moverse un cuadro a la derecha o a la izquierda, o detenerse.

Los símbolos están representados por $q_1, q_2, q_3, \ldots$ etc; el símbolo accesado en la cinta se representa con $S_0, S_1, S_2, S_3, \ldots$ etc; y el movimiento hacia la izquierda por I y a la derecha por D.

Por ejemplo la gráfica suguiente:



representa una Máquina de Turing que escribe en una cinta tres veces el símbolo S_1. La gráfica se lee así:

1. Se inicia en el estado q_1; si el símbolo de la cinta es S_0 (cualquiera que no sea S_1) se escribe S_1; si el símbolo en la cinta es S_1, se hace un movimiento en la cinta a la izquierda según se especifica en la gráfica. y se procede con q_2.

2. En el estado q_2, si el símbolo de la cinta es S_0 se escribe S_1; si el símbolo en la cinta es S_1, se hace un movimiento en la cinta a la izquierda de nuevo y se procede con q_3.

3. En el estado q_3, si el símbolo de la cinta es S_0 se escribe S_1; si el símbolo en la cinta es S_1, la máquina escibe el símbolo S_1 y se detiene.

Así el proceso completo de la operación está dado por las siguientes étapas, en dónde los siete símbolos representan la cinta y el número de abajo representa la posición en la cinta en cada estado):

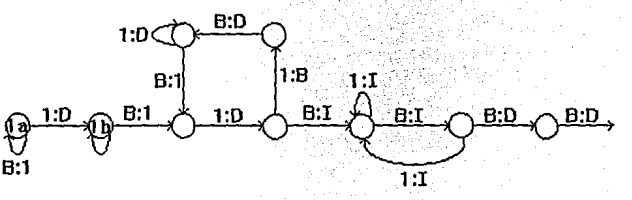
```
0 0 0 0 0 0 0
        1

0 0 0 0 1 0 0
        1

0 0 0 0 1 0 0
      2

0 0 0 1 1 0 0
      2

0 0 0 1 1 0 0
    3

0 0 1 1 1 0 0
    3
```

El diagrama que representa a una función Turing es una gráfica como la que representa a una Máquina de Turing pero con las características antes expuestas antes expuestas.

Los diagramas de las funciones Abacus se pueden representar en términos de los diagramas de las funciones de Turing. Para mostrarlo se verá un caso particular de la operación Abacus s+, para cuando s=1[12].

El diagrama Abacus cuando s=1, es:

cuyo diagrama equivalente para la función de Turing es:t



Este diagrama hace tres cosas:

1. Agrega un 1 al 1 encontrado.
2. Recorre los 1 hacia la izquierda del 1 afectado si es necesario.
3. Regresa al 1 más hacia la izquierda de la cinta.

De la misma manera en que esta operación Abacus tiene su equivalente en las funciones de Turing, todas las operaciones Abacus también lo tienen. Lo único que hay que agregar a la conversión de gráficas es un programa que convierta la notación monádica a la decimal y viceversa.

[12]El ejemplo presentado sólo pretende ilustrar la representación de una función Turing en términos de una función Abacus, y de ninguna manera hacer una generalización de todas las funciones Abacus ya que, además de ser muy elaborado, no permitirfa generalizar nada fuera de los límites del ejemplo mismo.Para más detalles se puede ver [Bool89].

Por lo tanto, la obtención del diagrama de flujo de Turing a partir de un diagrama Abacus, y el hecho de que el diagrama resultante calcule la misma función que calculaba el diagrama Abacus, permite establecer que:

Las funciones Abacus son Turing.

Con todo lo anterior se concluye que las funciones de Church o recursivas son computables; además que existe una equivalencia entre las funciones recursivas , Abacus y Turing. Es decir, que podemos representar un sistema matemático en términos computables, mediante funciones recursivas, diagramas Abacus (programas) y funciones Turing (computadoras).

CAPITULO IV.- TEOREMA DE GÖDEL

Queda pues entendido que para demostrar un teorema no es necesario, ni siquiera conveniente, saber lo que quiere decir. El geómetra podría sustituirse por el "piano lógico" imaginado por Stanley Jevons; o, si se prefiere, podría imaginarse una máquina en la que las premisas se introdujeran por un lado y los teoremas salieran por el otro...El matemático no necesita saber que hace más que esas máquinas.

Henri Poincaré.

Tortuga: Ah, sí. Bueno, mire, el cangrejo vino un día a visitarme. Debe usted saber que él siempre tuvo debilidad por los artefactos llamativos, y en ese momento estaba apasionado, más que por ninguna otra cosa, por los dispositivos de pasar discos. Acababa de comprar el primero de su vida y, como es un poco crédulo, había creído todo lo que le dijo el vendedor: en especial, que ese aparato era capaz de reproducir cualquier clase de sonido. En fin, el Cangrejo estaba convencido de poseer un fonógrafo perfecto.

Aquiles: Naturalmente, supongo que usted habrá intentado disuadirlo.

Tortuga: Claro que sí, pero él no escuchó mis razones. Sostenía empecinadamente que su adquisición podía reproducir el sonido que fuera. Ya que no podía convencerlo de lo contrario, deje de insistir y poco tiempo después le devolví su visita, llevando conmigo el disco de una canción que yo había compuesto. La canción se llamaba "No puedo ser escuchada mediante el fonografo 1".

Aquiles: Un tanto inusual. ¿Era un regalo para el Cangrejo?

Tortuga: De ningún modo. Yo sugerí que la pusiéramos en su nuevo fonográfo, y él tuvo mucho agrado en complacerme. Pero, lamentablemente, después de unas cuantas notas, el fonógrafo empezó a vibrar con bastante fuerza y luego con un sonoro "plop", se deshizo en un sinnúmero de pedacitos, que quedaron esparcidos por toda la habitación. Ni decir que también el disco se destruyó por completo.....

Tortuga:...El cangrejo no quiso creer que la falla estuviera en su aparato, de modo que tercamente, fué y compró un nuevo fonografo, más caro que el anterior, y esta vez el vendedor prometió devolverle el doble del precio si el cangrejo encontraba algún sonido que el equipo no pudiera reproducir con exactitud. Todo esto me vino a relatar exitadamente mi amigo, y yo le prometí ir a verlo de nuevo para ver su flamante modelo.

Aquiles: Corríjame si me equivoque: seguramente usted, antes de volver a la caja del cangrejo, consultó al fabricante otra vez, y compuso y grabó una canción llamada "No puede ser escuchada mediante el fonógrafo 2", basada en el diseño del nuevo equipo.

Tortuga: ¡Brillantísima deducción, Aquiles! Ha captado usted la situación de un modo perfecto.

Aquiles:¿Y qué ocurrió ahora?

Tortuga:Lo que tenía que ocurrir, exactamente de la misma manera: el fonografo se hizo mil pedazos, y el disco quedó deshecho.

Tortuga:Nuestro jueguito siguió adelante, en el mismo estilo, durante algunos asaltos más, y finalmente nuestro amigo trató de actuar con mucho mayor ingenio. Se inspiró en el mismo principio por el cual yo grababa mis discos, e intentó adelantárseme. Escribió a los fabricantes, enviándoles la descripción de su equipo de su propia invención para que se lo construyeran. Lo llamó "Fonógrafo Omega", y era mucho más complejo que un fonógrafo comun....En primer lugar el Fonógrafo Omega tenía una cámara de Televisión cuya finalidad era examinar cada disco, antes de hacerlo escuchar. Esta camara estaba conectada a una pequeña computadora construida a propósito, encargada de determinar con exactitud la naturaleza del sonido mediante el análisis de los surcos del disco.

Aquiles: Sí, hasta aquí muy bien, pero ¿Qué hacia el Fonógrafo Omega con esa información?

Tortuga: A través de complicados procesos de cálculo la computadora establecía los efectos que el sonido podía ejercer sobre el fonógrafo. Si deducía que los sonidos serían de tal tipo que dañarían la configuración presente del aparato, realizaba entonces una acción muy astuta: el Primer Omega contenía un dispositivo de desarmar las partes componentes del fonógrafo y volver a armarlas de diferentes maneras, de forma tal que, realmente estaba en condiciones de modificar su propia estructura. Si los sonidos eran "peligrosos", era adoptada una configuración nueva alejada de la amenaza; su construcción la concretaba la subunidad reconstructora, bajo la dirección de la pequeña computadora. Unicamente después de toda esta operación, el Fonógrafo Omega hacía escuchar el disco.

Aquiles: ¡Ajá! Eso debe haber significado el fin de sus tretas, señora. Estoy seguro de que experimentó usted un leve contratiempo...

Tortuga: Es curioso que me diga eso... Me imagino que no conoce usted en profundidad el Teorema de la Incompletitud, de Gödel, ¿verdad?

ver

IV. EL TEOREMA DE GÖDEL

El objetivo de este capítulo es presentar el teorema de Gödel, así como el papel que desempeñó la computabilidad en el desarrollo del mismo.

Por ello, el capítulo consta de dos grandes apartados. El primero persigue plantear los aspectos que conforman el marco conceptual en que se desarrolla el trabajo de Gödel; mientras que el segundo contiene el teorema propiamente dicho, así como la forma en que el autor concluyó que el Programa de Hilbert no era completo.

El desarrollo del primer apartado se compone a su vez de dos partes: una presenta el enfoque semántico y sintáctico con el que se plantean los sistemas lógicos; y la segunda los principios de la Lógica de Primer Orden en el que se describe cómo está conformada, cómo se hacen deducciones a partir de ella, y la manera en que se da la conexión semántica y sintáctica de un sistema lógico.

4.1. SEMANTICA Y SINTAXIS

Un sistema lógico se define desde dos puntos de vista diferentes que son equivalentes: el punto de vista semántico y el punto de vista sintáctico.

La semántica se refiere a la interpretación, que consiste en la especificación de un conjunto no-vacío, en donde constantes y variables toman valores. Para que la semántica pueda cumplir su función se requiere tener lo que se conoce como Fórmula Bien Formada (FBF), que es una fórmula que surge del sistema lógico mediante una serie de reglas de deducción. La FBF está definida por las proposiciones simples (también llamadas átomos), o por proposiciones compuestas resultado de la composición de proposiciones simples mediante operadores lógicos como la negación, la conjunción, etc. (Más adelante se dará una definición más amplia de las FBF).

Entonces, un modelo de un conjunto de FBF's constituye una interpretación si todas las FBF's en el conjunto son verdaderas. Una FBF es verdadera si para la sustitución de todos los valores de verdadero (V) y falso (F), el resultado de la proposición es verdadero. Por lo tanto, existe un punto de vista semántico válido si todas las FBF del sistema son verdaderas.

El punto de vista sintáctico tiene que ver con la "demostración" de una FBF de un sistema lógico. La demostración debe hacer manifiesta la idea de 'validez deductiva'. El planteamiento es el siguiente: Sea el argumento "$p_1,\ldots, p_n$", entonces p es deductivamente correcta si y sólo si hay una demostración de p a partir de las premisas $p_1, p_2, p_3,\ldots, p_n$.

La demostración es una secuencia finita de pasos en donde el último paso es el propio teorema. Cada paso produce una premisa o una fórmula nueva que se justifica por una regla de deducción aceptable, luego entonces, es necesario señalar cuándo una regla de deducción es aceptable, lo que implica que estas reglas deben darse de una manera explícita a fin de que el concepto de demostración sea decidible.

Hay diferentes maneras de proceder para lograr una definición rigurosa de lo que es una demostración con reglas deductivamente aceptables. Una consiste en hacer un sistema axiomático deductivo (al estilo de la geometría de Euclides); otra forma, es presentar un "sistema de deducción natural". Este tipo de teoría de la demostración fue propuesto originalmente por Gerhard Gentzen en los 30's y mejorado por Federich en los 50's. En particular, se aplica a lo que se conoce com Lógica de Primer Orden.

4.2. LOGICA DE PRIMER ORDEN

Un sistema de deducción natural es muy parecido a la manera en que se suceden demostraciones (por ejemplo, en geometría). Como componentes esenciales estos sistemas utilizan:

a) Hipótesis de trabajo.

b) Unas cuantas reglas que explican las conexiones lógicas.

c) Una estructura de prueba principal y pruebas subordinadas a esta prueba principal. Esta estructura se va a representar mediante rayas verticales. Una subprueba (o prueba subordinada) comienza con una hipótesis de trabajo.

d) Toda fórmula que ha sido aceptada puede usarse más adelante en la misma prueba, o en pruebas subordinadas a ésta (reiteración).

Dado que este sistema de deducción natural se aplica a la Lógica de Primer Orden (LPO), es conveniente plantear los lineamientos generales en torno a esto.

La LPO se basa en proposiciones que cumplen los requisitos de la definición de FBF, cuya evaluación se basa en valores de verdad. El resultado permite interpretar la FBF e indicar si se generan fórmulas válidas.

Bajo este esquema, el Cálculo de predicados de Primer Orden es un sistema formal. Tiene un lenguaje objeto; un conjunto de esquemas y axiomas (axiomas de la lógica), y dos reglas de inferencia: *modus ponens y generalización.* Estas reglas de inferencia se aplican a un conjunto de FBF. La regla de inferencia *modus ponens* es la

operación que produce una FBF w_2 a partir de la FBF w_1 y de la FBF $w_1 \rightarrow w_2$. La regla de inferencia de la generalización (o Especialización universal) produce la FBF $w_2(A)$ de las FBF $(\forall x)[w_1(x) \rightarrow w_2(x)]$ y $w_1(A)$. Como cada nueva FBF se añade a los axiomas, el sistema formal que se obtiene se llama Teoría de Primer Orden.

Un modelo de una teoría es una interpretación en el cual todos los axiomas son verdaderos; los axiomas lógicos son, en efecto, elegidos como verdaderos en todas las interpretaciones. Una FBF w se deriva de un conjunto de FBF's W en una teoría $T(W+w)$ si y sólo si w es deducible de W y de los axiomas de T por una aplicación finita de las reglas de inferencia usando la regla del *modus ponens*, en la cual se establece que, de p y $p \rightarrow q$ es posible concluir q. Otras reglas de inferencia de *modus ponens y generalización* pueden usarse para derivar teoremas.

De manera más específica, se dice que la lógica proposicional trata con planteamientos declarativos, los cuales reciben el nombre de proposiciones y se evalúan para calificarlas de forma excluyente como verdaderas o falsas.

Ejemplos de proposiciones son: "la tierra es redonda", "la luna gira alrededor de la tierra", "Jaime Rangel tiene el grado de doctor". Las proposiciones en lógica se denotan por símbolos como P, Q y R que son llamados *fórmulas atómicas o átomos.* Dichas proposiciones pueden ser compuestas usando los conectores lógicos: ~ (negación), $\wedge$ (conjunción), $\vee$ (disyunción), $\Rightarrow$ (implicación), $\Leftrightarrow$ (si y sólo si), para dar como resultado las denominadas Fórmulas Bien Fórmadas (FBF) y que se definen recursivamente como sigue:.

1) Un átomo (o proposición simple) es una fórmula.

2) Si G es una fórmula, entonces la negación $(\tilde{}G)$ es una fórmula.

3) Si G y H son fórmulas, entonces: $(G \wedge H)$, $(G \vee H)$, $(G \Rightarrow H)$ y $(G \Leftrightarrow H)$ son fórmulas.

4) Todas las fórmulas se generan a partir de las reglas anteriores.

Las evaluaciones de Fórmulas Bien Formadas en lógica de proposiciones se basan en valores de verdad: verdadero y falso $\{V, F\}$, que ocurren en los átomos de las fórmulas según tablas de verdad, definidas como:

G	H	$\sim G$	$(G \wedge H)$	$(G \vee H)$	$G \Rightarrow H$	$G \Leftrightarrow H$
v	v	f	v	v	v	v
v	f	f	f	v	f	f
f	v	v	f	v	v	f
f	f	v	f	f	v	v

Dada una fórmula proposicional G compuesta de $A_1,...,A_n$ átomos, G tiene una interpretación según el valor de asignación que le dan $A_1,...,A_n$; donde a cada A_i se le asigna el valor verdadero o falso, pero no ambos. Por otro lado, una fórmula G se dice que es verdadera bajo una interpretación si y sólo si G es evaluada como V en la interpretación; de otra forma se dice que G es falsa bajo la interpretación.

En una fórmula con n diferentes átomos se tienen 2^n distintas interpretaciones para la fórmula que pueden ser mostradas en una tabla de verdad determinando que una fórmula es válida, (o una tautología) cuando es verdadera bajo todas sus interpretaciones.

Ejemplo. Considérese la fórmula siguiente: $F \equiv \big((P \Rightarrow Q) \wedge P\big) \Rightarrow Q$

La tabla de verdad para $\big((P \Rightarrow Q) \wedge P\big) \Rightarrow Q$ está dada por:

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

Una fórmula se dice que es válida si y sólo si es verdadera bajo todas sus interpretaciones. Una fórmula se dice que es inválida si y sólo si es no válida. Por otra parte, se dice que una fórmula es inconsistente (o no satisfactible) si y sólo si es falsa bajo todas sus interpretaciones. Por último, una fórmula se dice que es consistente (o satisfactible) si y sólo si es no inconsistente.

Ejemplo: Considérese la fórmula siguiente:

$$(P \Rightarrow Q) \wedge (P \wedge \sim Q)$$

La tabla de verdad asociada a $(P \Rightarrow Q) \wedge (P \wedge \sim Q)$ es:

P	Q	$(P \Rightarrow Q) \wedge (P \wedge \sim Q)$
V	V	F
V	F	F
F	V	F
F	F	F

En lógica es necesario transformar una fórmula en otra, para plantearla en lo que se denomina "formas normales". La transformación consiste de un proceso de reemplazo de una fórmula por alguna otra equivalente, dada por la siguiente definición: Dos fórmulas F y G son equivalentes $(F \equiv G)$ si y sólo si los valores de verdad de F son los mismos de G bajo cualesquier dominio de interpretación de F y de G. El proceso de transformación se lleva a cabo de acuerdo con algunas leyes o reemplazos de fórmulas que mantienen los valores de verdad en los dominios de interpretación, y que se muestran a continuación.

i) $F \Leftrightarrow G = (F \Rightarrow G) \wedge (G \Rightarrow F)$

ii) $F \Rightarrow G = \sim F \vee G$

iii) Ley conmutativa

$F \vee G = G \vee F$

$F \wedge G = G \wedge F$

iv) Ley asociativa

$$(F \vee G) \vee H = F \vee (G \vee H)$$

$$(F \wedge G) \wedge H = F \wedge (G \wedge H)$$

v) Ley distributiva

$$F \vee (G \wedge H) = (F \vee G) \wedge (F \vee H)$$

$$F \wedge (G \vee H) = (F \wedge G) \vee (F \wedge H)$$

vi) Leyes de Morgan

$$\sim (F \vee G) = \sim F \wedge \sim G$$

$$\sim (F \wedge G) = \sim F \vee \sim G$$

Ahora bien, una literal es un átomo o la negación de un átomo.

Por otro lado, una fórmula F se dice que puede estar en una forma normal conjuntiva o bien en una normal disyuntiva. La forma normal conjuntiva se da si y sólo si F tiene la forma $F \equiv F_1 \wedge ... \wedge F_n$, $n \geq 1$, donde cada $F_1,...,F_n$ es una disyunción de literales. La forma normal disyuntiva se obtiene si y sólo si F tiene la forma: $F \equiv F_1 \vee ... \vee F_n$, $n \geq 1$; donde cada $F_1,...,F_n$ es una conjunción de literales.

Hay dos teoremas intermedios que se usan con la finalidad de probar que una fórmula particular es una consecuencia lógica de un conjunto finito de fórmulas, equivalente a probar que una cierta fórmula relacionada es válida o inconsistente.

Así, dadas las fórmulas $F_1,...,F_n$ y una fórmula G, se dice que G es una consecuencia lógica de $F_1,...,F_n$ si y sólo si dada cualquier interpretación I en la cual $F_1 \wedge ... \wedge F_n$, es verdadera, G es también verdadera. $F_1, F_2, ... F_n$ se llaman axiomas de G. En segundo lugar, se dice que G es una consecuencia lógica de $F_1,...,F_n$, si y sólo si la fórmula $((F_1 \wedge ... \wedge F_n) \Rightarrow G)$ es *válida*. Y por último, G es una consecuencia lógica de $F_1,...,F_n$, si y sólo si la fórmula $(F_1 \wedge ... \wedge F_n \wedge \sim G)$ es *inconsistente*. Por lo tanto, si G es una consecuencia lógica de las fórmulas $F_1,...,F_n$, la fórmula $((F_1 \wedge ... \wedge F_n) \Rightarrow G)$ se llama un teorema, y G se denomina la *conclusión del teorema.*

El lenguaje del sistema lógico consiste de los siguientes símbolos primitivos:

- Variables y símbolos constantes
- Símbolos de función
- Símbolos de predicado
- Conectivos lógicos, negación($\sim$), implicación ($\Rightarrow$)
- Cuantificadores universales $\forall$ (para todo).

En general, se admite usar los siguientes cuatro tipos de símbolos para la construcción de un átomo:

i) **Símbolo Individual o Constante**: Los cuales son usualmente nombres de objetos.

ii) **Símbolo de variable:** Denotado con letras minúsculas $x,y,z,\ldots$

iii) **Símbolos de Función**: Se denotan con letras minúsculas $l,g,h,\ldots$ o bien, una cadena como PADRE_DE ( ); ( ) más ( ); etc.

iv) **Símbolo de Predicado**: Se usan letras mayúsculas $P,Q,R,\ldots$ o bien cadenas expresivas relacionales.

Cualquier símbolo de función o de predicado toma un número específico de argumentos, considerando que si tiene n argumentos es una función *n-aria:* similarmente, los predicados P toman n argumentos. Por ejemplo, La función PADRE_DE( ) es de un argumento, con un mapeo de elementos de una lista de constantes a constantes.

PADRE_DE (Juan) = Pedro

En lógica de primer orden, PADRE_DE (Juan) es un término. Los términos tienen la siguiente definición recursiva:

i) Una constante es un término

ii) Una variable es un término

iii) Si f es una función de n argumentos donde $t_1,\ldots,t_n$ son términos; entonces $f(t_1,\ldots,t_n)$ es también un término.

iv) Cualquier cosa generada mediante la aplicación de alguna de las reglas anteriores se denomina término.

Por otra parte, si P es un símbolo de predicados de grado n y $t_1,\ldots,t_n$ son términos, entonces $P(t_1,\ldots,t_n)$ es un átomo, y pueden operarse mediante el uso de los conectivos logicos: $\sim$(Negación), $\Rightarrow$ (Implicación), $\wedge$ (Conjunción), $\Leftrightarrow$ (Si y sólo si), $\vee$ (Disyunción)

Además, por el hecho de considerar variables se introducen dos simbolos especiales:

$\forall x$ (Cuantificador universal)

$\exists x$ (Cuantificador existencial)

Cada expresión se llama una fórmula y en cada una de ellas las variables son ligadas o libres. Por otro lado, se define el alcance de un cuantificador que ocurre en una fórmula, como la fórmula para la cual el cuantificador se aplica. Por ejemplo.

$(\forall x)(\exists y)$ MENOR_QUE(x,y) es: MENOR_QUE(x,y)

$(\forall x)(Q(x) \Rightarrow R(x))$ es: $(Q(x) \Rightarrow R(x))$

La ocurrencia de una variable en una fórmula es ligada si y sólo si la ocurrencia está en el alcance de un cuantificador empleado con la variable, o bien, es la ocurrencia en ese cuantificador. Una variable es libre en una fórmula si al menos una ocurrencia de ésta es libre en la fórmula. Una variable es ligada en una fórmula si al menos una ocurrencia es ligada. Por ejemplo:

$(\forall x)\, P(x,y)$ x es ligada y y es libre

$(\forall x)\, R(x,y)$ o $(\forall y)\, Q(y)$ y es libre y ligada, respectivamente.

Ahora bien, si se involucran planteamientos con variables x y cuantificadores, la definición de FBF está dada recursivamente como sigue:

i) Un átomo es una fórmula

ii) Si F y G son fórmulas, entonces $\sim F$; $(F \vee G)$; $(F \wedge G)$; $(F \Rightarrow G)$; y $(F \Leftrightarrow G)$ son también fórmulas.

iii) Si F es una fórmula y x es una variable libre en F entonces:

$(\forall x)\, F$ y $(\exists x)\, F$ son también fórmulas

iv) Las fórmulas se generan solamente por un número finito de aplicaciones de i), ii), y iii)

Uno de los ejemplos clásicos es:

Cualquier hombre es mortal, Turing es un hombre, por lo tanto Turing es mortal. Lo que en términos simbólicos se representa por :

Cualquier hombre es mortal $(\forall x)(\text{HOMBRE}(x) \Rightarrow \text{MORTAL}(x))$

Turing es un hombre HOMBRE (Turing)

Turing es mortal MORTAL (Turing)

$(\forall x)(\text{HOMBRE}(x) \Rightarrow \text{MORTAL}(x))$ o HOMBRE (Turing) $\Rightarrow$ MORTAL (Turing)

Una interpretación de una fórmula F en Lógica de Primer Orden (LPO) consiste de un dominio D no vacío y una asignación de valores para cada constante, símbolo de función o símbolos predicados que ocurren en F; cumpliéndose lo siguiente:

1. A cada constante se le asigna un elemento en D.

2. Para cada símbolo de función *n-aria* se asigna un mapeo de D^n a D, en donde $D^n=\{(X_1,...,X_n) / X_1 \in D,...,X_n \in D\}$

3. Para cada símbolo de predicado *n-ario*, se asigna un mapeo de D^n a (V,F).

Para cualquier interpretación de una fórmula sobre un dominio D, la fórmula puede ser evaluada como V o F de acuerdo a:

1) Si las fórmulas F y G se evalúan lógicamente, entonces las fórmulas: $\sim F$, $(F \wedge G)$, $(F \vee G)$, $(F \Rightarrow G)$ y $(F \Leftrightarrow G)$, pueden evaluarse bajo la consideración de la primera tabla de verdad presentada en este capútulo.

2) Una fórmula F es evaluada como verdadera sobre todo el dominio de interpretación, si es verdadera para toda x $(\forall x\, F[x])$.

3) El cuantificador existencial aplicado a una fórmula F da por resultado una evaluación verdadera, si existe al menos uno que evalúa a F como verdadera.

Con todo lo anterior se ha cubierto lo que se considera importante conocer como marco de referencia del trabajo de Godel. A continuación se expone el Teorema propiamente dicho, antecedido de una presentación del Programa de Hilbert dada la estrecha relación que guarda con el tema central de este capútulo. Así, a continuación se presenta , de una manera general, lo que intentaba el Programa de Hilbert, así como las

definiciones que permiten plantear cuándo un sistema formal es completo, inconsistente, o bien consistente o inconsistente.

4.3 EL TEOREMA DE GÖDEL

EL PROGRAMA DE HILBERT

La preocupación fundamental de Hilbert fue garantizar la formalización de la matemática clásica. Esto debido a las deficiencias que presentaba hasta ese entonces el desarrollo de la matemática sustentada en los *Principia Mathematica.* Para ello, Hilbert desarrolló sus aportaciones a través de plantear sus conceptos mediante signos gráficos; plasmó sus ideas a través de "hileras de signos", es decir, fórmulas. El razonamiento lo trabajó por la mera manipulación combinatoria de las ideas y, por último, la demostración por la deducción formal conforme a reglas mecánicas.

El objetivo central de todos estos elementos era demostrar que no había contradicción alguna en el sistema y que, dado que todo tenía una referencia en un contexto finito numerable, era posible restringir todos los pensamientos a este ámbito y, por tanto, olvidar el contenido transfinito presuntamente problemático de la matemática clásica.

Por ello, el programa formalista de Hilbert requería:

1. Construir sistemas formales completos para las principales tareas teóricas de la matemática clásica, y

2. Probar la consistencia de dichos problemas.

Se buscaba entonces desarrollar un sistema formal que contara, en primer lugar, con un conjunto numerable de signos primitivos que determinara el conjunto de sus hileras o secuencias finitas de signos (con posibles repeticiones). En segundo lugar, que tuviera ciertas reglas combinatorias que determinara cuales hileras eran fórmulas; el conjunto de las fórmulas constituirían el lenguaje formal del sistema. En tercer lugar, tener otras reglas combinatorias, que determinaran cuáles secuencias de fórmulas constituirían deducciones.

Para ello, se necesitaba contemplar aspectos básicos. Así, se concibió una proposición como una fórmula sin variables libres. A su vez, una proposición era

deducible si constituía el último miembro de (una secuencia de fórmulas) una deducción. El conjunto de las proposiciones deducibles constituiría una teoría formalizada.

Fue necesario asimismo, definir varias características de un sistema formal. Así, se definió que:

a) Un sistema formal *S* es **completo** si y sólo si para cada proposición φ de su lenguaje formal ocurre que φ es deducible en *S* o que no-φ también es deducible en S. Así pues, un sistema formal completo no deja pregunta sin respuesta. Con su ayuda pueden decidirse todas las cuestiones pertinentes acerca de un sistema. Basta con realizar un proceso deductivo hasta llegar a φ o a no-φ, puesto que es seguro que se llegará a alguna de las dos. Así, todos los problemas planteados en un sistema formal completo son **decidibles**.

b) Un sistema formal *S* es **incompleto** si y sólo si hay alguna proposición φ, tal que ni φ es deducible en *S*, ni tampoco lo es no-φ. Por lo tanto, hay problemas planteables en *S* para los que S no ofrece solución, es decir, hay problemas **indecidibles** en *S*.

c) Un sistema formal *S* es **consistente** si y sólo si hay alguna proposición de su lenguaje formal que no es deducible en *S* (o, equivalentemente, si para ninguna proposición φ ocurre que tanto φ como no-φ sean deducibles en *S*).

d) Un sistema formal S es **inconsistente** o contradictorio si y sólo si toda proposición del lenguaje formal de *S* es deducible en *S* (o, equivalentemente, si hay alguna proposición φ del lenguaje formal de S, tal que tanto φ como no-φ son deducibles en S).

Una teoría definida semánticamente (como el conjunto de las proposiciones verdaderas en un modelo determinado) es siempre completa y consistente. Y si se pretende que un sistema formal (como la aritmética formalizada) refleje perfectamente una teoría definida semánticamente (como la aritmética natural), entonces es preciso que el sistema formal sea completo y consistente.

Se tenía pues, un interés fundamental en determinar si un sistema axiomático era completo, ya que en realidad uno de los motivos más poderosos de varias de las ramas de las matemáticas había sido el deseo de especificar un conjunto suficiente de supuestos iniciales de los que fueran deducibles todos los enunciados verdaderos de algún campo de análisis.

Muchos matemáticos suponían como un hecho obvio que siempre podía indicarse un conjunto axiomático completo para cualquier rama de la matemática. En particular, parece haberse creido universalmente que los axiomas propuestos para la aritmética de los matemáticos del siglo XIX eran efectivamente completos o, en el peor de los casos, podían completarse mediante la adición de un número finito de nuevos axiomas.

La Lógica de Primer Orden es un ejemplo del sistema matemático para el cual pueden alcanzarse plenamente los objetivos de la teoría Hilbertiana de la demostración.

Pero, ese cálculo no codifica más que un fragmento de la lógica formal, y ni su vocabulario ni su aparato formal bastan para desarrollar en su marco ni siquiera la aritmética elemental.

EL TEOREMA DE GÖDEL

Sobre la base de la inquietud de Hilbert, quien cuestionaba si podía mostrarse la consistencia de un sistema formal, en cuanto a que pudiera expresarse toda la aritmética, y no sólo un fragmento de ella, en 1931 se publicó el artículo más famoso de Gödel y quizá de la historia entera de la lógica. Sus resultados mostraban la imposibilidad de llevar a cabo el programa de Hilbert.

En primer lugar, Gödel probaba que todos los sistemas formales de la matemática clásica (incluidos el de P.M., la aritmética formal de Peano, la teoría axiomática de conjuntos etc., y en general, cualquier sistema formal que cumpliese ciertas condiciones de aceptabilidad) eran incompletos, es decir que para cada uno de ellos podía efectivamente construirse una proposición indecidible (tal que ni ella ni su negación fueran deducibles). Además esta incompletitud no tenía solución. Incluso, por muchos axiomas que se añadieran, los sistemas formales seguían siendo incompletos. En segundo lugar, Gödel demostraba que era imposible probar la consistencia de un sistema formal (que cumpliera ciertas condiciones mínimas de aceptabilidad) como el de la matemática clásica, aun utilizando todos los recursos de razonamientos incorporados en el sistema, es decir, que era imposible demostrar la consistencia de un sistema formal dentro del mismo. Naturalmente, seguía siendo posible probar su consistencia desde una teoría más potente que el propio sistema formal, lo cual sería de dudosa utilidad; pues se mata un dragón sólo para engendrar otro.

Gödel llevó a cabo sus pruebas planteando y resolviendo los problemas metamatemáticos dentro de la aritmética. Por un ingenioso procedimiento, Gödel asignó números naturales a las hileras (y secuencias de ideas) del sistema formal y relaciones numéricas a las relaciones metamatemáticas. Estableció así un isomorfismo entre el sistema formal y cierto sistema numérico. Gödel se movió con gran habilidad entre ambos, jugando con el doble hecho de que, por un lado, toda afirmación metamatemática sobre el sistema formal tenía una relación numérica y de que, por otro lado, toda afirmación numérica pertinente podía ser expresada por una fórmula del sistema formal. Así, era posible construir una proposición φ que, naturalmente interpretada, decía que un cierto número n tenía cierta propiedad P. Pero ese número era el número correspondiente a la fórmula φ, y esa propiedad era la propiedad numérica correspondiente a la propiedad metamatemática de no ser deducible. Por tanto, la proposición φ, interpretada naturalmente, afirmaba su propia indecibilidad. La proposición φ era a la vez verdadera e indeducible. Gödel probaba que ni φ ni no-φ podían ser deducibles en el sistema formal, que, por tanto, era incompleto. De modo análogo probó que era imposible deducir la fórmula que, naturalmente interpretada, afirmaba la consistencia del sistema.

El estudio que llevó a cabo Gödel es muy complejo. Antes de llegar a los principales resultados es necesario comprender y dominar perfectamente 46 definiciones preliminares. Aquí se emprenderá un camino más fácil, intentando mostrar la escencia de la demostración.

En primer lugar, Gödel demostró que es posible asignar un único número a cada signo elemental, a cada fórmula o sucesión de signos -esta asignación la hizo para el sistema formal de los P.M.- y a cada prueba (o sucesión finita de fórmulas). Este número tiene como función identificar el símbolo y recibe el nombre **número de Gödel** del signo, fórmula o prueba. A todo este proceso se le conoce como **la enumeración de Gödel.**

El vocabulario fundamental está formado por símbolos elementales divididos en dos clases: los símbolos **constantes y los símbolos variables.** Se supondrá que existen diez símbolos constantes, a los que se asocian, como números de Gödel, los números enteros que van del 1 al 10. (vease tabla 4.1)

Tabla 4.1

Símbolos Constantes

Símbolos constantes	Número de Gödel	Significado
~	1	no
∨	2	o
⊃	3	Si... entonces...
∃	4	Existe un
=	5	igual a
0	6	cero
s	7	el sucesor inmediato de
(	8	Signo de puntuación
)	9	Signo de puntuación
,	10	Signo de puntuación

Los símbolos **variables** del vocabulario fundamentalmente están divididos en tres clases: Los símbolos de **variables numéricas**, '*x*', '*y*', '*z*', etc., con los que se puede sustituir a los numerales y a las expresiones numéricas; los símbolos de **variables proposicionales** '*p*', '*q*', '*r*', etc.,con los que se pueden sustituir a las fórmulas; y los símbolos de **variables predicativas**, 'P', 'Q', 'R', etc. con los que se pueden sustituir los predicados tales como 'primo' o 'mayor que', según el ejemplo de la sección anterior.

A estas tres clases se les asigna el número de Gödel mediante la siguiente regla: a las variables numéricas se les asocia un número primo mayor que 10 (ver la tabla 4.2), a las variables proposicionales se les asocia el cuadrado de un número primo mayor que 10 (ver la tabla 4.3), y a las variables predicativas se les asigna un número primo mayor que 10 elevado al cubo (ver tabla 4.4).

Tabla 4.2

Símbolos de Variables Numéricas

Símbolo de Variable numérica	Número de Gödel	Posible sustitución
x	11	0
y	13	s0
z	17	y

Tabla 4.3

Símbolos de Variables Proposicionales

Símbolos de variable proposicional	Número de Gödel	Posible sustitución
p	11^2	$0 = 0$
q	13^2	$(\exists x)(x = sy)$
r	17^2	$p \supset q$

Tabla 4.4

Símbolos de Variables Predicativas

Símbolos de Variable Predicativa	Número de Gödel	Posible sustitución
P	11^3	Primo
Q	13^3	Compuesto
R	17^3	Mayor que

Considérese ahora una fórmula del sistema, por ejemplo, '$(\exists x)(x = sy)$'. (que dice : 'Existe una x, tal que x es el sucesor inmediato de y', es decir, que todo número tiene un sucesor). Los números asociados a los diez símbolos elementales que lo forman son, 8,4,11,9,8,11,5,7,13,9, lo que se observa en el siguiente esquema:

(	∃	x	)	(	x	=	s	y	)
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
8	4	11	9	8	11	5	7	13	9

En seguida, para obtener el número de Gödel (llámese m), se multiplican los números primos elevados al número asociado al signo, es decir:

$$2^8 \times 3^4 \times 5^{11} \times 7^9 \times 11^8 \times 13^{11} \times 17^5 \times 19^7 \times 23^{13} \times 29^9$$

Considérese ahora, la siguiente sucesión de fórmulas (que incluso pueden ser parte de una demostración):

$(\exists x)(x = sy)$

$(\exists x)(x = s0)$

La segunda fórmula, que significa '0 tiene un sucesor inmediato', es derivable de la primera fórmula sustituyendo a la variable numérica 'y' por el numeral '0'.

Ya se conoce el número de Gödel de la primera fórmula, que es m.Supóngase que el número de Gödel de la segunda fórmula es n. El número de Gödel que se le asigna a la sucesión de dos fórmulas está dado por los dos números primos en orden creciente

elevados a una potencia igual al número de Gödel de la fórmula correspondiente (Como quizá ya se intuyó, se puede ver que la asignación de los números de Gödel se puede definir de una manera recursiva).

Entonces, el número de Gödel asociado a esta sucesión será: $k = 2^m \times 3^n$

Por este procedimiento de condensación se puede obtener un número para cada serie de fórmulas, de manera que toda expresión contenida en el sistema (trátese de un signo elemental, una sucesión de símbolos o una sucesión de sucesiones), debe llevar asignado un único número de Gödel.

Lo que se ha hecho hasta aquí es establecer un método para **aritmetizar** completamente al cálculo formal. El método consiste esencialmente en un conjunto de reglas para establecer una correspondencia biunívoca entre las expresiones del cálculo y una cierta subclase de los números enteros (no todo número entero es un número Gödel. Por ejemplo, 100 no es un número Gödel por que no cumple con las reglas de asinación de la numeración de Gödel). Dada una expresión, puede calcularse unívocamente el número de Gödel que corresponde a ella. Pero además, dado un número, se puede determinar si es un número de Gödel, y, si lo es, la expresión que representa puede ser exactamente "restablecida" y analizada, pues como el número se ha construido como producto de números primos, es posible factorizarlo en dichos primos.[1]

Los siguientes pasos muestran cómo se puede investigar, a partir de un número de Gödel, qué expresión representa y viceversa:

❶. Supóngase que 243,000,000 es un número de Gödel

❷. Factorizando $64 \times 243 \times 15625$

❸. Expresado en potencias del mínimo factor primo equivale a: $2^6 \times 3^5 \times 5^6$

[1]Todo número menor o igual a 10, es también el número de Gödel correspondiente de un signo constante elemental; por ello, el signo puede ser unívocamente identificado. Si el número es mayor que 10, puede ser descompuesto en sus factores primos de una forma precisa. Si es un número primo mayor que 10, o la segunda o tercera potencia de un número primo que reuna esa cualidad, es el número de Gödel de una variable identificable. Si es el producto de números primos sucesivos, elevando cada uno de ellos a alguna potencia, puede ser el número de Gödel o de una fórmula o de una sucesión de fórmulas. En cualquier caso, puede determinarse exactamente la expresión a que corresponde.

❹. Las potencias de los factores anteriores, según la correspondencia

$$\begin{array}{ccc} 6 & 5 & 6 \\ \downarrow & \downarrow & \downarrow \\ 0 & = & 0 \end{array}$$

de las tablas anteriores equivale a:

❺. Por lo tanto, la expresión que representa el número de Gödel 243,000,000 es:'0 = 0'

Así, los pasos seguidos en el orden: ❶,❷,❸,❹ y ❺ muestran cómo a partir del número de Gödel se obtiene la expresión que representa. Y, visceversa, si los pasos se siguen en el orden: ❺,❹,❸,❷ y ❶, pues muestran como a partir de una expresión se obtiene su número de Gödel.

El paso siguiente de Gödel es una ingeniosa aplicación de la idea de representación.[2]. Demostró que todas las propiedades metamatemáticas acerca de las propiedades estructurales de las expresiones contenidas en el cálculo, pueden ser reflejadas adecuadamente dentro del cálculo mismo. La idea básica que subyace en su procedimiento es la siguiente: puesto que toda expresión del cálculo está asociado a un número Gödel, puede construirse una proposición metamatemática acerca de las expresiones y de sus relaciones aritméticas recíprocas. De esta manera, queda completamente **aritmetizada** la metamatemática.

Cada proposición metametamática se encuentra representada por una única fórmula dentro de la aritmética, y por las relaciones numéricas de dependencia entre sus correspondientes fórmulas aritméticas.

En particular, toda **caracterización metamatemática**, de la estructura de expresiones en el sistema y todo lo que se diga acerca de ellas, queda reflejado en una función aritmética de enteros; y todo **enunciado metamatemático acerca de las relaciones entre fórmulas** queda reflejado en una relación aritmética entre enteros.[3]

---

[2]Este paso deja sentir un poco más la brillante idea de utilizar el razonamiento matemático para explorar el pensamiento matemático.

[3]Por función aritmética se entiende una expresión como 2+3 o (5 x 7)-8, etc; Por una relación aritmética se entiende una proposición -verdadera o falsa- tal como 5=7, 7>4, etc.

¡Es decir, que tanto la caracterización matemática como el enunciado metamatemático acerca de las relaciones entre fórmulas, son COMPUTABLES !

Por lo tanto, la importancia de esta aritmetización de la metamatemática se debe al hecho de que, puesto que cada uno de sus enunciados puede representarse en el sistema formal por una y sólo una expresión asociada con un número de Gödel, las relaciones de dependencia lógica entre enunciados metamatemáticos pueden explorarse mediante el procedimiento de examinar relaciones entre enteros y sus factores.

Considérese el primer axioma del cálculo proposicional que es, a su vez, además, un axioma del sistema formal sujeto a examen: '$(p \vee p) \supset p$'. Su número de Gödel es $2^8 \times 3^{11^2} \times 5^2 \times 7^{11^2} \times 11^9 \times 13^3 \times 17^{11^2}$, que se designará con la letra '*a*'. Considérese también la fórmula '$(p \vee p)$', cuyo número de Gödel es $2^8 \times 3^{11^2} \times 5^2 \times 7^{11^2} \times 11^9$, designado por la letra '*b*'. La proposición metamatemática de que la fórmula '$(p \vee p)$' es una parte del axioma. ¿A qué fórmula aritmética del sistema formal corresponde? Es evidente que la fórmula más pequeña '$(p \vee p)$' puede ser una parte inicial de la fórmula mayor, que es el axioma, si y solamente si, el número de Gödel *b*, que representa a la primera, es un factor del numero de Gödel *a*, que representa a la segunda. Bajo el supuesto de que la expresión 'factor de' esté convencionalmente definida en el sistema aritmético formalizado, la única fórmula aritmética que corresponde a la declaración metamatemática antes enumerada es: '*b* es un factor de *a*'. Además, si esta fórmula es verdadera, esto es, si *b* es un factor de *a*, entonces es cierto que '$(p \vee p)$' es una parte inicial de '$(p \vee p) \supset p$'.

Ahora considérese el enunciado metamátematico: 'La sucesión de fórmulas cuyo número de Gödel es *x*, es una demostración de la fórmula cuyo número de Gödel es *z*'. Este enunciado está representado en el cálculo aritmético (o reflejado en él) por una determinada fórmula del cálculo, la cual expresa una relación puramente aritmética entre *x* y *z*.[4] Se escribe esta relación aritmética entre *x* y *z* como la fórmula '*Dem(x,z)*'; para recordar el enunciado metamatemático 'La sucesión de fórmulas cuyo número de Gödel es *x* es una demostración de la fórmula cuyo número de Gödel es *z*'; y a su vez el enunciado metamatemático: 'la sucesión de fórmulas cuyo número de Gödel *x* no es una

[4] En el ejemplo anterior, en el cual se asigna el número de Gödel *k* a una demostración (o parte de una demostración) se encontró que $k = 2^m \times 3^n$. Una breve reflexión muestra que existe una determinada relación aritmética, aunque bastante complicada, entre *k*, el número de Gödel de la demostración, y el número de Gödel *n* de la conclusión.

demostración de la fórmula cuyo número de Gödel es *z*' será también representado por una fórmula del formalismo matemático. Se escribirá esta fórmula del modo siguiente:

'$\sim Dem(x,z)$' .

Es necesario agregar un poco más de esta notación especial para exponer el punto clave del argumento de Gödel. Por ejemplo, la fórmula '$(\exists x)(x = sy)$' tiene como número de Gödel a *m*, mientras que el número de Gödel de la variable '*y*' es 13. En dicha fórmula sustitúyase la variable del número Gödel 13 (o sea '*y*') por el numeral *m*. El resultado es la fórmula '$(\exists x)(x = sm)$', que dice literalmente que existe un número *x* tal que *x* es el sucesor inmediato de *m*, tiene también su número Gödel, que puede calcularse fácilmente. Pero, en lugar de hacer el cálculo, se puede identificar el número mediante una inequívoca caracterización metamatemática: es el número Gödel de la fórmula que se obtiene a partir de la fórmula del número de Gödel *m*, sustituyendo la variable de número Gödel 13 por el numeral de *m*. Esta caracterización metamatemática determina unívocamente un número definido, que es cierta función aritmética de los números *m* y 13, en la que puede ser expresada la función misma dentro del sistema formalizado. El número puede, por consiguiente, ser designado dentro del cálculo. Esta designación será escrita como 'sust (*m*,13,*m*)' que tiene como finalidad recordar la caracterización metamatemática que representa, 'el número de Gödel de la fórmula obtenida a partir de la fórmula de número de Gödel *m*, sustituyendo la variable de número Gödel 13 por el numeral *m*'. En términos generales, se puede tener la expresión 'sust (y,13,y)' que es la imagen reflejada dentro del cálculo aritmético formalizado de la caracterización metamatemática 'el número de Gödel de la fórmula que se obtiene a partir de la fórmula del número Gödel y, sustituyendo la variable del número Gödel 13 por el numeral de y'. Se observará también que cuando se sustituye 'y' por un número definido en 'sust (*y*,13,*y*)' -por ejemplo, el número de *m* o el numeral de doscientos cuarenta y tres millones-, la expresión resultante designa un número entero definido, que es el número de Gödel de una determinada fórmula.[5]

En este momento se tienen todos los elementos para mostrar la argumentación de Gödel. Primeramente se mostrará el argumento de manera general y, después, se entrará en los detalles en cada uno de los pasos de la argumentación.

---

[5] ¿Qué entero se designa por 'sust (*y*,13,*y*)' si por casualidad la fórmula cuyo número de Gödel *y* no contiene la variable del número de Gödel 13 -esto es, si la fórmula no contiene a la variable 'y'? .La respuesta es que, puesto que *y* es '$0 = 0$' y *y* no contiene a esta variable, no puede hacerse ninguna sustitución, o, lo que es lo mismo, que la fórmula obtenida de '$0 = 0$' es esta misma fórmula. Por lo tanto, el número designado por 'sust (243 000 000,13, 243 000 000)' es 243 000 000.

❶. Primero Gödel mostró cómo construir una fórmula aritmética G que representa la declaración metamatemática 'La fórmula G no es demostrable'.[6]

❷. En segundo lugar Gödel demostró que G no es formalmente demostrable.

❸. Gödel demostró después que la fórmula G que no es formalmente demostrable, es una fórmula aritmética verdadera (**¡es decir, es Computable!**).[7]

❹. Entonces dado que G es al mismo tiempo verdadera y formalmente indecidible, los axiomas de la aritmética son incompletos.[8] y siempre serán incompletos.

- Además demostró que la aritmética es esencialmente incompleta.[9]

❺. Por último, Gódel describió cómo construir una fórmula aritmética A que represente a la proposición metamatemática: 'la aritmética es consistente'.

- Demostró que la fórmula '$A \supset G$' es formalmente demostrable.
- Finalmente demostró que la fórmula A no es demostrable.

Por lo que la consistencia de la aritmética no puede ser establecida por un argumento que pueda hallarse representado en el cálculo aritmético formal, en términos de sí misma.

A continuación se verá con detalle el desarrollo de cada uno de los pasos de la argumentación de Gödel:

---

[6] La fórmula G dice de sí misma que no es demostrable.

[7] Es verdadera en el sentido de que afirma que todo número entero posee una cierta propiedad aritmética que puede ser exactamente definida y presentada en cualquier número entero que se examine, es decir que es computable.

[8] Si los axiomas de la aritmética son incompletos ⇨ no se pueden deducir todas las verdades aritméticas de los axiomas.

[9] Esencialmente incompleta significa que aunque se incluyeran axiomas al sistema para que la fórmula verdadera G fuera formalmente derivable, podría construirse otra fórmula verdadera pero formalmente indecidible.

❶ Construcción de la fórmula aritmética *G* que representa la declaración "la fórmula *G* no es demostrable"

1. Sea '~*Dem*(*x*,*z*)'[10]

2. Anteponiendo el prefijo'(*x*)' ⇨:'(*x*)~*Dem*(*x*,*z*)' [11] que representa la proposición metamatemática 'la fórmula con número de Gödel *z* no es demostrable'.

3. Gödel demostró que un caso particular de '(*x*)~*Dem*(*x*,*z*)', no es demostrable

4. Sea Π '(*x*)~*Dem*(*x*,sust(*y*,13,*y*))' la fórmula que representa la proposición metamatemática 'la fórmula de número de Gödel sust(*y*,13,*y*) no es demostrable'.[12]. Supóngase que el número de Gödel de la Π es *n*.

5. Entonces si se sustituye en la fórmula Π a la variable de número de Gödel 13 (es decir la variable 'y') por el numeral *n*. Se tiene la fórmula que se quería construir:

G '(*x*)~*Dem*(*x*,sust(*n*,13,*n*))'

[10] La sucesión de fórmulas con número de Gödel *x* no es una prueba de la fórmula con número de Gödel *z*.

[11] Para todo *x*, la sucesión de fórmulas con número de Gödel *x* no son una prueba de la fórmula con número de Gödel *z*.

[12] Como se recordará la expresión sus(y,13,y) designa a un número

6. Como se puede ver G es la fórmula que se obtiene a partir de la fórmula de número de Gödel *n* (es decir la fórmula II), sustituyendo la variable 'y' por el numeral *n*. Y sust(*n*,13,*n*) es el número de Gödel de la fórmula que se obtiene a partir de la fórmula de número de Gödel *n* (la fórmula II) , sustituyendo la variable 'y' por el numeral *n*. Entonces, sust(*n*,13,*n*) es el número de Gödel de G.

7. Pero G es la imagen reflejada dentro del cálculo aritmético de la proposición metamatemática 'La fórmula de número de Gödel sust(*n*,13,*n*) no es demostrable'.

8. Y dado que la fórmula de número de Gödel sust(*n*,13,*n*) es (*x*)~*Dem*(*x*,sust(*n*,13,*n*)) entonces la fórmula aritmética '(*x*)~*Dem*(*x*,sust(*n*,13,*n*))' representa en la proposición metamatemática: 'la fórmula '(*x*)~*Dem*(*x*,sust(*n*,13,*n*))' no es demostrable'.

Es decir que G está construida afirmando de sí misma que no es demostrable

❷

1) Si G es demostrable, entonces ~*G* es demostrable.[13]

Para probar que Si *G* es demostrable, entonces ~*G* es también demostrable, considérese:

* Si *G* es demostrable entonces tendría que haber una sucesión de fórmulas dentro de la aritmética que constituyesen una prueba para *G*. Sea k el número de Gödel para dicha prueba. La relación aritmética '~*Dem(x,z)*' debe darse entre k y el número de Gödel de *G*, es decir,'sust (*n*,13,*n*)'

'*Dem (k, sust(n,13,n))*' tiene que ser una fórmula aritmética verdadera..

[13]De hecho aquí Gödel hizo la demostración de que la fórmula G es demostrable si y solamente si ~*G* es demostrable. Pero para nuestra hipótesis es suficiente G es demostrable si ~*G* es demostrable.

* Y dado que la relación aritmética '*Dem (k, sust(n,13,n))*' se da entre un par definido de números, entonces la fórmula que expresa este hecho es demostrable.

* Entonces como *G* además de ser una fórmula aritmética verdadera es formalmente demostrable, *G* es un teorema.

* Entonces, dado que *G* es un teorema se puede derivar de *G* la fórmula

~G (~(x)~Dem (x, sust(n,13,n))'

* ∴ *G* es demostrable, su negación formal *~G* también es demostrable.

2) Pero si G y *~G* son formalmente demostrables, los axiomas que los derivaron (axiomas de la aritmética) son inconsistentes.

3) De lo que se concluye que si los axiomas del sistema formalizado de la aritmética son consistentes, entonces ni la fórmula G ni su negación son demostrables.

∴

Si la aritmética es consistente ⇨ G no es formalmente demostrable[14].

❸

Gödel demostró después que la fórmula G que no es formalmente demostrable, es una fórmula aritmética verdadera (¡es decir, es Computable!)

1. La proposición metamatemática 'la fórmula '(*x*)~*Dem*(*x*.sust(*n*,13,*n*))' no es demostrable es verdadera bajo la hipótesis de la aritmética consistente.

2. Esta proposición está representada dentro de la aritmética por la misma fórmula que menciona la proposición.

3. Las proposiciones metamatemáticas verdaderas corresponden a fórmulas aritméticas verdaderas, ya que las proposiciones metamatemáticas han sido representadas en el fórmalismo aritmético.

4. Si G corresponde a una proposición metamatemática verdadera, entonces G es verdadera.

Por lo tanto, *G* es verdadera y formalmente indecidible

[14] O formalmente indecidible. En el sentido de que ni *G* ni *~G* pueden deducirse formalmente de los axiomas.

❹ Entonces, dado que *G* es al mismo tiempo verdadera y formalmente indecidible, los axiomas de la aritmética son incompletos. Además, la aritmética es escencialmente incompleta.

1. Dado que G es verdadera e indecidible, los axiomas de la aritmética son incompletos. Es decir: no se pueden deducir todas las verdades aritméticas de los axiomas, y no importa cuántos axiomas se agreguen siempre será incompleto.

- Los axiomas de un sistema son completos si todas las proposiciones verdaderas que pueden expresarse en el sistema son formalmente deducibles de los axiomas.

- Pero dado que G es una fórmula verdadera de la aritmética no deducible dentro de ella, por lo tanto los axiomas de la aritmética son incompletos. Esto es bajo la hipótesis de que sean consistentes.

2. La aritmética será siempre necesariamente incompleta. Aunque al sistema se le agregara cualquier número de axiomas (incluido G), aún podría construirse otra fórmula aritmética verdadera, que fuera indecidible; de la misma manera en que se construyó G. Esto siempre se dará no importa cuantas veces se amplie el sistema dado.

Por lo tanto, existe una limitación fundamental en la eficacia del método axiomático. Es decir, que la verdad aritmética no puede ser reducido a un orden sistemático basadode de una vez para siempre en un conjunto de axiomas del que pueda derivarse formalmente toda proposición aritmética verdadera.

❺ Finalmente, se puede llegar a establecer la proposición metamatemática ' Si la aritmética, es consistente, es incompleta'. Esta proposición condicional tomada como un todo, esta representada por una fórmula demostrable dentro de la aritmética formalizada.

1 Por un lado, la proposición metamatemática 'la aritmética es consistente' es equivalente a la proposición 'existe al menos una fórmula de la aritmética que no es demostrable', la cual se representa por :

A) $(\exists x)\,(x)\,\sim \mathrm{Dem}(x,y)$

Traducida dice que 'Existe por lo menos un número *y* tal que para todo *x*, *x* no se mantiene en la relación *Dem* a *y*, que interpretada metamatemáticamente :'Existe por lo

menos una fórmula de la aritmética para la cual ninguna sucesión de fórmulas constituye una prueba.'

Por lo tanto, A representa a 'la aritmética es consistente'

2. Por otro lado, la proposición metamatemática 'la aritmética es incompleta' procede de 'existe una proposición aritmética verdadera que no es formalmente demostrable' que es la fórmula G.

Por lo tanto, de 1 y 2, la proposición metamatemática condicional 'Si la aritmética es consistente, es incompleta' está representada por la fórmula:

$(\exists x)\,(x)\,\sim\mathrm{Dem}(x,y)\supset(x)\sim\mathrm{Dem}(x,\mathrm{sust}(n,13,n))$

Es decir $A\supset G$

Ahora sólo falta demostrar que A no es demostrable.

- Suponiendo que A es demostrable, puesto que $A\supset G$ es demostrable ⇨ G es demostrable.
- Pero si la aritmética es consistente, G es formalmente indecidible.(G es indecidible a menos que el cálculo sea inconsistente.)
- lo que implica que si G es no demostrable A también lo es.

Por lo tanto, si la aritmética es consistente, la fórmula A no es demostrable.

Resumiendo:

i) La fórmula A representa la proposición metamatemática 'la aritmética es consistente'

ii) Si esta proposición pudiera ser demostrada con una argumentación susceptible de ser plasmada en una sucesión de fórmulas que constituyera una prueba en cálculo aritmético, la propia fórmula sería demostrable.

iii) pero A es no demostrable si la aritmética es consistente.

iv) Si la aritmética es consistente, su consistencia no puede ser demostrada por ningún razonamiento metamatemático capaz de ser representado dentro del formalismo de la aritmética.

Por lo tanto,r no hay posibilidad de que una prueba de la consistencia sea reflejada sobre deducciones formales de la aritmética.

CONCLUSIONES

Conclusiones:

Las conclusiones que a continuación se presentan corresponden a tres enfoques: el teórico, el práctico, y el relacionado con la cuestión formativa de los alumnos de carreras de Computación; en particular de estudiantes de Matemáticas Aplicadas y Computación. Estos enfoques guardan una relación tan estrecha que no fue posible hacer una separación tajante entre ellos.

Para los estudiantes de la carrera de Matemáticas Aplicadas y Computación es fundamental conocer los conceptos teóricos que sustentan a la computadora. El tener estos conceptos en mente da otra visión en la solución de los problemas; esta visión establece qué problemas son tratados en términos computacionales y cómo llevar a cabo su tratamiento.

Generalmente, la computación se concibe como una herramienta aislada del campo de la ciencia; algo así como una brillante invención que la tecnología ha hecho realidad, sin conocer el transfondo de los planteamientos filosóficos y matemáticos que hay detrás de todo esto.

El tener una concepción teórico-matemática de la computación permite determinar sus alcances y limitaciones.

La computadora, concebida sobre principios de modelos matemáticos, requiere tener bases muy sólidas sobre teoría de esta área del saber humano, pues ello permite plantear diseños, desarrollos, alcances y limitaciones reales de la computabilidad.

Luego entonces, el aporte mínimo de la computabilidad está en la posibilidad de plantear que algo computable asegura una solución, es decir, una respuesta. Esta Tesis no intenta dar elementos para determinar cuándo algo es computable; únicamente señala que si algo es computable entonces tendrá solución.

De este trabajo se observa que la Lógica de pronto se convierte en la metaciencia de la computación; es decir, que todos los resultados en la Lógica tienen una repercusión directa en la Computación.

La computabilidad es un concepto universal que aparece en los enfoques de Turing, Church y Gödel. Turing expresa la computabilidad en términos de la detención de la máquina de Turing. En el cálculo lambda de Church, la computabilidad está definida, por un lado, en el sentido de que las funciones pueden ser evaluadas para cualquier argumento dado; y por otro lado, en la correspondencia entre el cálculo lambda de Church y las funciones Turing. En el teorema de Gödel, la representación de la computabilidad está dada por la propia númeración de Gödel. La necesidad de la

correspondencia entre el número de Gödel y todos los elementos que forman parte del lenguaje del sistema define a esta relación como computable.

Así, para Turing la computabilidad se manifiesta a través de si la máquina se detiene o no; para Church mediante la existencia o ausencia del valor de la función; y para Gödel la computabilidad se da mediante la existencia de los números creados por él.

Se observa entonces que la computabilidad resulta ser uno de los grandes vínculos entre la teoría matemática y la teoría de la computación. Permite analizar problemas matemáticos desde enfoques computacionales y viceversa.

BIBLIOGRAFIA

BIBLIOGRAFIA

[Aho 74] Aho A. V., J. E. Hopcroft, and J.D. Ullman [1974]
The Design and Analysis of Computer Algorithms
Addison Wesley, Reading, Mass.

Uno de los libros de una serie de clásicos que ha escrito el gran trio Aho, Hopcroft y Ullman, en el que se hace un estudio riguroso de de los Algoritmos.

[Aho 83] Aho A. V., J. E. Hopcroft, and J.D. Ullman [1983]
Data Structures and Algorithms
Addison Wesley, Reading, Mass.

Otro clásico, continuación del anterior con la diferencia de que este le añade al estudio de los algorítmos su gran complemento: la estructura de Datos.

[Aho 86] Aho A. V., R. Sethi, and J.D. Ullman [1986]
Compiler Design: Principles, Techniques, and Tools
Addison Wesley, Reading, Mass.

Bajo la filosofía de Aho, Ullman y la colaboración de Sethi, en este libro se plasman todas las herramientas para la creación de compiladores, claro esta, sustentado bajo el rigor del estudio de los lenguajes y el estudio de los autómatas.

[Bool89] Boolos, Jeffrey, [1989]
Computability and Logic
Third Edition Cambridge University Press

Libro que expone conceptos computacionales en términos estrictamente matemáticos, llegando a importantes resultados.

[Chap91] Chapa V. S, Garcia F. J.. [1991]
Lógica Matematica y Aplicaciones a la Demostración de Teoremas
CINVESTAV IPN México (Reporte Técnico-Depto Ingenieria Eléctrica)

Un escrito muy completo que expone los aspectos teóricos de la lógica, hasta llevarlos a una de las aplicaciones más interesantes, que es la demostración de teoremas.

[Cros72]
Crossley, Ash, Brickhill, Williams, [1972]
What is mathematical Logic
Oxford University Press

Una buena descripción tanto teórica como histórica del desarrollo de la lógica que abarca desde los primeros intentos de axiomatización hasta la hipótesis de la continuidad y el teorema de la elección.

[Davi58]
Davis Martin [1958]
Computability and Unsolvability
Mc. Graw-Hill, New York

Sin duda un libro de lectura obligada, considerado uno de los clásicos en el estudio de la computabilidad.

[Harr75] Harrison M. A. [1975]
Introduction to switching and Automata Theory
Mc. Graw-Hill, New York

Libro que trata sobre el estudio del diseño de compuertas lógicas enfocandolo desde el punto de vista de teoría de autómatas.

[Hofs79]
Hofstadfter Douglas R. [1979]
Gödel, Escher, Bach: An Eternal Golden Braid
Basic Books, New York

Una fascinante obra, que destaca por su profundidad filosófica, matemática y poética, en ella se entrelazan a tres grandes figuras Gödel, Escher y Bach, todos ellos "bailando" al ritmo del teorema de Gödel, sobra decir que es un libro indispensable para el alumno de computación.

[Hofs85] Hofstadfter Douglas R. [1985]
Metamagical Themas: Questing for the Essence of Mind and Pattern
Basic Books, New York

Con la gran exposición a que nos tiene acostumbrados Douglas, hace un libro de una recopilación de escritos, la mayoría de ellos publicados en la revista "Scientific American" cuando "heredó" la columna de Martin Gardner, estos ecritos tratan de diversos temas todos ellos relacionados con el estudio de la Inteligencia Artificial. También se recomienda su lectura.

[Hopc79] Hopcroft, J.D. Ullman [1979]
Introduction to Automata Theory, Languajes, and Computation
Addison Wesley, Reading, Mass.

Sin duda un clásico en el estudio de la teoría de la computación, libro que sienta las bases teóricas para el diseño de lenguajes y compiladores.

[Knut73] Knuth, D. E. [1973]
The Art of Computer Programming. Vol. III: Sorting and Searching
Addison Wesley, Reading, Mass.

Un libro que se recomienda por sí mismo, el clásico *ipso facto,* entre las muchas y brillantes exposiciones que tiene, el concepto de recursividad es uno de los agraciados por el Sr Knuth

[McIn77] McIntosh Harold V. [1977]
Lógica Matemática, Notas de Clase
(ESFM-IPN), México

Notas de clase de McIntosh, sobre la lógica, un excelente material tanto por su claridad como por su rigidez matemática.

[Mins67] Minsky Marvin [1967]
Computation: Finite and Infinite Machines
Prentice-Hall, New York

Uno de los clásicos realizado por uno de los pensadores más profundos de la Inteligencia Artificial, en el que se trata el concepto de de computabilidad con un rigor y claridez dignas de su autor.

[Newm56] Newman J. R., Nagel E. [1958]
Gödel's Proof
New York University Press

Una de las exposiciones más claras del teorema de Gödel, una prueba de que los conceptos más abstractos de la matemática pueden ser tratados y expuestos con una gran maetría pedagógica.

[Newm56] Newman J. R., Nagel E. [1956]
The World of Mathematics
First Edition Simon and Shuster, New York

Una gran obra que reune obra, vida conceptos, trabajos y hallazgos del facinante mundo de las matemáticas algunos de ellos escritos por los propios protagonistas.

[Penr89] Penrose R. [1989]
The Emperor's New Mind
First Edition Oxford University Press

Libro escrito por uno de los excépticos mas serios de la Inteligencia Artificial (uno de los físico-matemáticos más creativos del mundo, estrecho colaborador en los descubrimientos de Stephen Hawking), donde se exponen con una maestría conceptos de matemáticas y computación. Penrose conjunta una serie de tópicos matemáticos , computacionales, de mecanica cuantica, de física, etc. para ofrecer su punto de vista sobre la Inteligencia Artificial.

[Most89] Mosterin J [1989]
Obras Completas de Gödel
Segunda Edición, Alianza Editorial, España.

Un gran esfuerzo por recopilar todo el material que dejó la obra de Gödel, entre ellas el teorema de la incompletitud. Además de su valor por constituir toda la obra de Gödel, este libro tiene acompañado en cada escrito de Gödel una brillante explicación del mismo, hecho que enriquece al trabajo.

[Galt90] Galton A. [1990]
Logic for Information Technology
John Wiley & Sons, England.
Un libro hecho a la medida de los estudiantes de computación que quieran introducirse al estudio de la lógica, hecho que es inevitable dada la estrecha relación entre la lógica matemática y la ciencia de la computación.

Hemerografía

[McCa60] McCarthy John [1960]
Recursive Functions of Symbolic Expresión and Their Computation by Machine, Part I.
Communication de ACM Vol 3, Num 4 pp 183-195 MIT Cambridge

Este libro representa el esquema formal en términos de funciones recursivas del lenguaje Lisp, sin duda nos lleva a reflexionar sobre los conceptos matemáticos que modelan la creación de un lenguaje.

[Newe76] Newell A., Simon H.A. [1976]
Computer Science As Empirical Inquiry: Symbols and Search
Communication of ACM Vol. 19 Num. 3 pp. 113-126

El artículo con el que se les dio el premio de Turing a dos grandes pioneros de la Inteligencia Artificial., este artículo presenta la argumentación de defensa de la I.A. que les permitió ser parte de la vanguardia en los trabajos de Inteligencia Artificial.

[Knut74] Knuth E.D. [1974]
Computer Programming As an Art
Communication of ACM Vol. 17 Num. 12 pp. 667-673

Artículo con el que se le dió reconocimiento mediante el premio de Turing, a uno de los grandes maestros de la computación; en el Knuth plasma una parte de lo que se considera ahora como clásicos de la computación, la serie de libros bajo el titulo The Art of Computer Programming.

[Back78] Backus J. [1978]
Can Programming be Liberted from the Von Newman Style? A funtional Style and its algebra of programs
Communications od ACM Vol. 21 Num. 8 pp. 613-641

En este artículo, también premio Turing de Backus se distute el paradigma de la programación funcional y la programación procedural, un artículo de gran interés dado el planteamiento filosófico matemático que hace sobre la creación de los lenguajes.