Título de la Tesis

"JERARQUÍA ARITMÉTICA DE KLEENE"

TESIS

Sometida para optar al título de Maestría en Matemática

Vicerrectoría de Investigación y Postgrado

Facultad de Ciencias Naturales, Exactas y Tecnología

APROBADO POR

Doctor Jaime Gutiérrez
Presidente

Doctor Eduardo Piza Volio
Miembro

Profesor Julio Vallarino
Miembro

REFRENDADO POR:

REPRESENTANTE DE LA VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO

FECHA: 25 de julio de 2015

UNIVERSIDAD DE PANAMÁ
VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO
FACULTAD DE CIENCIAS NATURALES, EXACTAS Y TECNOLOGÍA

PROGRAMA DE MAESTRÍA

JERARQUÍA ARITMÉTICA DE KLEENE

IVETH V. MARTÍNEZ V.

TESIS PRESENTADA COMO UNO DE LOS REQUISISTOS PARA OPTAR AL GRADO DE MAGISTER EN CIENCIAS CON ESPECIALIZACIÓN EN MATEMATICA PURA

PANAMÁ, REPÚBLICA DE PANAMÁ
2015

AGRADECIMIENTOS

Agradezco primero a Dios, por darme la oportunidad de compartir con amigos, colegas y familiares que de una u otra forma enriquecen, en muchos aspectos, mi vida personal y profesional En especial al **Dr. Eduado Piza Volio**, de la Universidad de Costa Rica, por su asesoría, motivación, confianza y enseñanza durante las distintas etapas del trabajo Al **Dr. Jaime Gutiérrez** por sus atinados consejos, disposición y orientación en los momentos que lo necesitaba, además al profesor **Edis Flores** por inscribirme en la maestria

RESUMEN

Una de las aplicaciones de la Teoría de Recursión que permite explicar los problemas indecidibles, es la Jerarquía Aritmética de Kleene el cual clasifica conjuntos recursivos, así como recursivamente irresolubles por su grado de irresolubilidad. Encontraremos una clasificación dentro de esta Jerarquía para los conjuntos: K, K_0, K_1, Inf, Fin, Rec, Con, Cof, Ext, Subset y Tot , y ponemos especial interés en el grado de irresolubilidad de los conjuntos Crea, Simp y Comp.

SUMMARY

Kleene's Arithmetical Hierarchy allows to classify recursive and not recursive sets by their degree of unsolvability, and this is one of the application of Recusion Theory to the problem of undecidability. In thiss work we find a classification in the Klenne's Hierarchy for the sets: K, K_0, K_1, Inf, Fin, Rec, Con, Cof, Ext, Subset and Tot, and study the degrees of unsolvability of the sets: Crea, Simp and Comp.

Introducción

Los modelos abstractos de computación tienen su origen en los años 30, antes de que existieran las computadoras modernas, en los trabajos de los lógicos Church, Gódel, Kleene, Post, y Turing El punto de partida de estos primeros aportes surgieron de las interrogantes presentadas en 1928 por Hilbert, cuya meta era crear un sistema matemático formal "completo" y "consistente", en el que todas las aseveraciones pudieran plantearse con precisión, es decir, encontrar un algoritmo que determinara la verdad o falsedad de cualquier proposición en el sistema formal A este problema se le conoce como *Entscheidungsproblem* (*problema de la decidibilidad*)

La teoría de la recursión, también llamada teoria de la computabilidad, trata fundamentalmente sobre el estudio de los problemas de decisión en todos los campos de la Matemática La teoria busca dilucidar cuáles problemas de decisión pueden ser resueltos en forma algorítmica, clasificando los problemas de decisión algoritmicamente no resolubles de acuerdo a sus dificultades y circunstancias intrínsecas de irresolubilidad

Una de las aplicaciones de la Teoría de Recursión que permite explicar los problemas indecidibles, es la Jerarquía Aritmética de Kleene, punto central de este trabajo Nuestro objetivo es estudiar el grado de irresolubilidad de algunos conjuntos no triviales y las relaciones entre los mismos

Este trabajo está organizado en tres capítulos, en el primer capitulo, revisamos algunos conceptos fundamentales como las funciones primitivas, utilizadas por Godel en 1930 en su Teorema de Incompletitud, la noción de funciones primitiva-recursivas, funciones básicas, predicados primitivo-recursivos, la enumeración de Godel importante para el desarrollo del trabajo y las funciones recursivas en algunos casos llamada funciones computables

Fuertemente asociado con las funciones recursivas, están los conceptos de los conjuntos recursivamente enumerables y los conjuntos recursivos revisados en el segundo capítulo Además definimos la reducibilidad de conjuntos, revisamos las relaciones entre los conjuntos completos, productivos, creativos, cilindros, inmunes y simples Se

presenta los conjuntos de índices que corresponden a problemas naturales indecidibles relacionados con las funciones recursivas K, K_0, K_1, Inf, Fin, Rec, Con, Cof, Ext, Subset y Tot y un interés en el grado de irresolubilidad de los conjuntos Crea, Simp y Comp

La clasificación de los conjuntos definidos en el segundo capítulo, es desarrollada en el tercer capítulo, momento en que la lógica juega un papel importante para el logro del mismo Además se establece la noción de Turing-reducible y los grados de irresolubilidad asociados Importante para este tema es el estudio de los metodos "*finite-infinite injury priority*", creado por Shoenfield (1961) y Sacks (1963-1964) de forma independientemente

Índice

CAPITULO I

CONCEPTOS FUNDAMENTALES

I. Conceptos Fundamentales

En este capítulo caracterizaremos formalmente algunos conceptos básicos e importantes para el estudio de temas que son puntos centrales de este trabajo

Las nociones de computabilidad y sus funciones asociadas (computables o recursivas) se remontan desde la época de antiguos griegos y los egipcios que mostraron que tenían una buena comprensión de la computación "metodos" Como es el caso del científico persa Al-Khwarizmi en el año 825 escribió un libro titulado "Sobre el calculo con números hindúes", que contenía la descripción de varios procedimientos que ahora podrían ser llamados algoritmos Su nombre parece ser el origen de la palabra "algoritmo" ya que al ser traducido al latín, se tituló "Algoritmi de Numero Indorum" Un algoritmo, lo podemos describir como un procedimiento secuencial para la solución de un problema el cual posee las siguientes características es determinista, posee un número finito de pasos bien definidos y produce un resultado sin importar la entrada.

La computabilidad moderna tiene sus raíces en trabajos matemáticos, realizados al inicio del siglo XX, con el propósito de formalizar el concepto de "procedimiento efectivo" (algoritmo) sin referirse a algún lenguaje de programación ni a dispositivos computacionales físicos En la década de 1930 los lógicos, en particular Alan Turing y Alonzo Church, estudiaron el significado de cómputo como un proceso mental abstracto y empezaron a diseñar dispositivos teóricos para modelar el mismo, como un proceso de cálculo que podría ser utilizado para expresar algoritmos, así como algunos calculos que no tiene solucion alguna (o no terminan) De esta forma nace la noción de una funcion parcial que generaliza la de un algoritmo, descrito anteriormente, considerando aquellos procesos de calculo que no siempre conducen a un resultado Es por ello que introduciremos el concepto de una funcion "computable" o "recursiva" y las terminologías básicas que se describen en esta área de la Matemática

1.1 Funciones y Algoritmos

Consideremos los conjuntos $X_1, X_2, \quad, X_n$ e Y distintos del vacío y denotemos $X = X_1 \times X_2 \times \quad \times X_n$, entonces $f\ X \to Y$ es la función de n variables, del cual se desprende las definiciones siguientes

Definición 1.1.1 Se dice que una función $f\ X \to Y$ es una *función parcial* si f se define como un subconjunto de $X_1 \times X_2 \times \quad \times X_n \times Y$ de manera que para cada selección de $x_1 \in X_1, \quad, x_n \in X_n$ existe al menos un elemento $y \in Y$ para el cual

$$f(x_1, x_2, \quad, x_n) = y$$

De forma similar, $f\ X \to Y$ es una *función total* si f se define como un subconjunto de $X_1 \times X_2 \times \quad \times X_n \times Y$ de manera que para cada seleccion de $x_1 \in X_1, \quad, x_n \in X_n$ existe exactamente un elemento $y \in Y$para el cual $f(x_1, x_2, \quad, x_n) = y$

Podemos decir que una función total es aquella cuyo dominio es $D_f = X = X_1 \times X_2 \times \quad \times X_n$

Definición 1.1.2 Sea $f\ X \to Y$, si $(x_1, x_2, \quad, x_n) \in D_f$ decimos que f está definida para todos los argumentos y se denota $f(x_1, x_2, \quad, x_n)\downarrow$, en caso contario la función no está definida para todos los argumentos y se denota $f(x_1, x_2, \quad, x_n)\uparrow$

Ejemplo 1 Supongamos que $f\ X \to Y$, es la función definida por el conjunto $\{(0,0,a),(0,1,b),(1,0,c),(1,1,,b)\}$ donde $X = \{0,1\} \times \{0,1\}, Y = \{a,b,c\}$ Observemos que f es una función total de dos

s, si

$$f(0,0) = a, f(0,1) = b, f(1,0) = c \text{ y } f(1,1) = b$$

Ejemplo 2 Sean $f\ X \to Y$ y $X = \{0,1,2\} \times \{0,1,2\}, Y = \{a,b,c,d\}$ Si f se define como $\{(0,0,b),(1,2,d),(0,1,c),(1,1,b),(2,0,c)\}$ vemos que $D_f \neq X$ por lo que la

función no es total y además $f(0,0) = b, f(0,1) = c, f(1,1) = b, f(2,0) = c$ y $f(1,2) = d$ pero $f(1,0)\uparrow, f(0,2)\uparrow$ y así sucesivamente para otras combinaciones

Es muy conocido que una de las operaciones para combinar funciones es la composición de las mismas Para la composición de funciones parciales o de funciones con más de una variable utilizaremos la teoría de conjuntos, es decir, definiremos la composicion de funciones deseada como un conjunto, especificando exactamente los elementos que contiene

Definición 1.1.3 Sea h una función parcial de m variables y sean $g_1, g_2, \ldots, g_m$ funciones parciales de n variables Entonces la composición de h con $g_1, g_2, \ldots, g_m$ es una función f de n variables tal que $f(x_1, x_2, \ldots, x_n) = z$ si y sólo si existes elementos $y_1, y_2, \ldots, y_m$ tal que

$$g_1(x_1, x_2, \ldots, x_n) = y_1$$

$$g_m(x_1, x_2, \ldots, x_n) = y_m \quad y$$

$$h(y_1, y_2, \ldots, y_m) = z$$

se denota $f = h \circ (g_1, g_2, \ldots, g_m)$ De manera general el valor de f relacionado a los valores de h y $g_1, g_2, \ldots, g_m$ se escribe de la forma

$$f(x_1, x_2, \ldots, x_n) = h\big(g_1(x_1, x_2, \ldots, x_n), \ldots, g_m(x_1, x_2, \ldots, x_n)\big)$$

Podemos afirmar que la composición es una función $f\ X \to Z$ con $h\ Y_1 \times \cdots \times Y_m \to Z$ y $g_i\ X \to Y_1 \times Y_2 \times \cdots \times Y_m$ De esta forma que si las funciones g_i, con $i = 1,2, \ldots, m$, y h son funciones totales entonces la función composición es una función total En caso contario la función f será no definida

En el estudio que nos concierne, sólo nos interesa las funciones cuyos dominios y rangos están en el conjunto de números naturales incluyendo el cero, a estas funciones se le conoce como *funciones aritméticas*

1.2 Funciones Básicas

Existen clases de funciones especiales, fundamentales para el desarrollo de la teoría que revisaremos posteriormente

Definición 1.2.1 A la función aritmética $C\ \mathbb{N}^n \to \mathbb{N}$ se le llama función constante de n variables si para todo $x_1, x_2, \quad, x_n \in \mathbb{N}, \quad C_m^{(n)}(x_1, x_2, \quad, x_n) = m$

Definición 1.2.2 Sea $P\ \mathbb{N}^n \to \mathbb{N}$ una función aritmetica, para cada $n \in \mathbb{N}$ y cada $\iota \leq n$, se define la ι-esima función proyección (o identidad) de n variables, para todo $x_1, x_2, \quad, x_n \in \mathbb{N}$ a la función $P_\iota^{(n)}(x_1, x_2, \quad, x_n) = x_\iota$

Otra función importante es la función vacía de $n$variables, definida a continuación

Definición 1.2.3 Se le llama función vacía de n variables Φ a la función aritmética que cumple que para todo $x_1, x_2, \quad, x_n \in \mathbb{N}$, $\Phi^{(n)}(x_1, x_2, \quad, x_n) \uparrow$

La composicion de la función vacía con otras funciones produce una función vacía, sin embargo la funcion vacía puede ser obtenida de la composición de una función no total con una apropiada funcion constante

La nocion de función parcial es importante para las técnicas de programación moderna Podemos decir que cada programa define una función parcial, pero nuestro interés desde un punto de vista abstracto es si un problema tiene solución por medio de una función que sigue un procedimiento efectivo En este sentido un algoritmo puede ser interpretado como una función parcial de n variables, para todo entero positivo n, y este eventualmente termina si al aplicar la $n-$tupla de números naturales, el valor de la función es un número natural En caso que la función no sea definida, es decir que su resultado no represente un número natural, indica que el algoritmo falla Daremos una definición formal que asocie lo descrito anteriormente

Definición 1.2.4 Para todo $n \in \mathbb{N}$, una función aritmética de n variables es *computable* (efectivamente calculable) si se puede calcular a través de un procedimiento mecánico finito (algoritmo)

Se describe un conjunto enumerable de objetos abstractos $A = \{A_0, A_1, A_2, \ \}$ que llamaremos algoritmos Por la definicion 1 2 4, se le asocia exactamente una función parcial $\left\{\varphi_0^{(n)}, \varphi_1^{(n)}, \varphi_2^{(n)}, \ \right\}$ de n variables para n entero positivo Si tomamos un alfabeto no vacío Σ y el conjunto de todas las cadenas del alfabeto Σ^* (que incluye la cadena de longitud 0, llamada cadena vacía λ), para cada $w_\iota \in \Sigma^*$ se le asocia un algoritmo A_ι con $\iota \in \mathbb{N}$, entonces $\varphi_\iota^{(n)}$ es la función computada por el algoritmo A_ι y w_ι es además, un índice de la función $\varphi_\iota^{(n)}$ Así podemos afirmar que una función F es computable si existe un índice $e = w_\iota$, para algún ι, tal que $F^{(n)} = \varphi_e^{(n)}$ y generalizando la función aritmética que es algorítmica (o computable) de n variables la denotaremos como $F_\iota^{(n)}$ asociada al algoritmo A_ι para un índice natural ι

Ejemplo 3 Sea A_1 la familia de algoritmos en el cual la función de n variables computada por el ι −ésimo algoritmo que está definido por

$$F_\iota^{(n)}(x_1, \ \ , x_n) = x_1 + \ \ + x_n + \iota$$

Así $F_3^{(1)}(5) = 8$ $F_3^{(2)}(7,6) = 16$ $F_3^{(3)}(1,0,4) = 8$

La familia de algoritmos cuenta con propiedades útiles para la práctica computacional, la primera propiedad reúne las funciones definidas anteriormente con otra que en conjunto se le conoce como funciones iniciales (o básicas), descrita a continuación

Propiedad 1 Toda familia de algoritmos A contiene algoritmos que compute las siguientes funciones aritméticas

a. La función constante,

b. La función proyección, y

c. La función de una variable S, llamada *función sucesor* donde $S(x) = x + 1$

En la computación práctica es muy común la utilización de subrutinas, es decir utilizar el resultado de una o más computaciones como argumento de una nueva computación. Observemos que esta aplicación genera una situación interesante en el cual los miembros de la familia de algoritmo son componibles uno con el otro y esta composición es cerrada bajo la composición funcional, mostrado formalmente en la propiedad siguiente.

Propiedad 2: Toda familia de algoritmos A, contiene algoritmos que evalúan la función de m variables h y las funciones de n variables $g_1, g_2, \dots, g_m$. Contiene, también, algoritmos que evalúan la función composición $f = h \circ (g_1, g_2, \dots, g_m)$.

De las funciones computables hay una que requiere una atención especial, ya que es capaz de simular el comportamiento de cualquier función algorítmica para alguna entrada $n \in \mathbb{N}$. A esta función la llamaremos *función universal*.

Definición 1.2.5: Sean A una familia de algoritmos y $F_i^{(n)}$ la función de n variables evaluada por i −ésimo algoritmo de A. Entonces para cada entero positivo n, definimos como función universal para la familia A, a la función de $(n + 1)$ variables de la forma

$$U^{(n+1)}(i, x_1, \dots, x_n) = F_i^{(n)}(x_1, \dots, x_n)$$

En la práctica aplicando $U^{(n+1)}$ al índice i y a las entradas $x_1, \dots, x_n$ es el resultado que podríamos obtener de igual forma si se aplica el i −ésimo algoritmo a los argumentos $x_1, \dots, x_n$. Esto nos sugiere que el lenguaje universal se considera como un programa *intérprete*[1] para un lenguaje dado, por lo que escribir un intérprete para el

[1] Un intérprete es un programa informático capaz de analizar y ejecutar otros programas. Realiza la traducción instrucción por instrucción.

lenguaje en sí mismo es una familia cuyas funciones universales son computables por los miembros de la familia Esto se plantea en la siguiente propiedad llamada *propiedad de enumeración*

Propiedad 3: Una familia de algoritmos A satisface esta propiedad si, para cada entero positivo n, la familia de algoritmos contiene un algoritmo que evalúa la función universal $U^{(n+1)}$

Al algoritmo que evalua a la función de universal se le llama *algoritmo universal para n variables* Para que se cumpla la propiedad 3, se requiere de la existencia de un algoritmo universal para cada valor de n, que a diferencia de las propiedades 1 y 2 requiere sólo del conjunto de funciones que son evaluadas por algún miembro de la familia A

Teorema 1.2.1 Sea A una familia de algoritmos para cual se cumpla las propiedades 1, 2 y 3 Entonces no todos los miembros del A computa funciones totales, es decir que algun miembro de A debe computar una función de una variable no total

Demostración Supongamos que las funciones de una variable son computadas por los miembros de A y además que son totales Por hipótesis tenemos que $P_1^{(1)}$, S y $U^{(2)}$ son funciones algorítmicas en A

Como A es cerrado bajo la composición entonces la función de una variable

$$f = S\left(U^{(2)}\left(P_1^{(1)}, P_1^{(1)}\right)\right)$$

es algorítmica en A

Tomemos $w \in \mathbb{N}$ un índice de f , es decir $f = F_w^{(1)}$ Entonces

$$F_w^{(1)}(x) = f(x) = S\left(U^{(2)}\left(P_1^{(1)}, P_1^{(1)}\right)\right)$$
$$= S\left(U^{(2)}(x,x)\right)$$
$$= \begin{cases} F_w^{(1)}(x) + 1, & si\ F_w^{(1)}(x) \downarrow \\ 1 \quad , & si\ F_w^{(1)}(x) \uparrow \end{cases}$$

Así $F_w^{(1)}(x)$ debe ser una funcion total y en particular definida para w por lo que se obtiene la contradicción $F_w^{(1)}(w) = F_w(w) + 1$ □

El corolario que se muestra a continuación se desprende del teorema 1 2 1

Corolario 1.2.1 Sea A una familia de algoritmo que satisface las propiedades 1, 2 y 3 Entonces para cualquier entero positivo n, la función vacía $\Phi^{(n)}$ es algorítmica en A

Demostración Por el teorema 1 2 1 algunos miembros de A computa funciones no totales de una variable Tomemos una función $f(a)\uparrow$, para cualquier argumento a Como A satisface las propiedades 1 y 2, la composición $h = f \circ \left(C_a^{(n)}\right)$ debe ser algorítmica en A Asi para toda selección de $x_1, x_2, \quad , x_n$, tenemos que

$$h(x_1, x_2, \quad , x_n) = f \circ \left(C_a^{(n)}(x_1, x_2, \quad , x_n)\right) = f(a)$$

vemos que la función h, no está definida para todos los valores, por lo que

$$h(x_1, x_2, \quad , x_n) = \Phi^{(n)}(x_1, x_2, \quad , x_n) \qquad \square$$

Al manejar datos u obtener resultados intermedios en el proceso de computacion, es importante tener a mano un mecanismo para fundamentar una decisión En el estudio abstracto de computación es de interés un algoritmo que evalúe varias funciones de decisión

De manera general podemos referirnos a una función de decisión como la funcion de cuatro variables Λ que llamaremos función de selección tal que

$$\Lambda(x, y, s, t) = \begin{cases} s, & si\ x = y \\ t, & si\ x \neq y \end{cases}$$

La condición de función algorítmica, lo establece la siguiente propiedad conocida como la propiedad de la selección

Propiedad 4 Una familia de algoritmos A satisface la condición de selección, si A contiene un algoritmo que evalúa la función de selección de cuatro variables definida Λ

La propiedad 4 en conjunto con las propiedades 1 y 2 permiten que muchas otras funciones de selección sean algoritmicas

Ejemplo 4 Sea A una familia de algoritmos que satisface las propiedades 1, 2 y 4 Consideremos la función de una variable, definida por

$$f(x) = \begin{cases} 0, & si\ x = 0 \\ 1, & si\ x \neq 0 \end{cases}$$

Solo necesitamos escribir a f como la composicion de funciones algorítmicas, asi

$$f(x) = \Lambda \circ \left(P_1^{(1)}, C_0^{(1)}, C_0^{(1)}, C_1^{(1)}\right)$$

Definitivamente f es algorítmica en A

Un problema que puede surgir en términos de la familia de algoritmos, es saber si algún algoritmo no se detenga Dados los argumentos, si el algoritmo termina podremos determinar el valor de la función computada En caso contrario podriamos realizar la computación de una función de manera indefinida sin estar seguro de que termina en un tiempo finito

Por tal razon es importante contar con un procedimiento para determinar si una función es definida o no para cualquier valor de argumentos Esto se formaliza a continuacion

Definición 1.2.6 Sea A una familia de algoritmos y para cada entero positivo n, definimos la función total de $(n+1)$ variables, llamada *función dominio* de la familia A como

$$D^{(n+1)}(\iota, x_1, \quad, x_n) = \begin{cases} 1, & si \quad F_\iota^{(n)}(x_1, \quad, x_n) \downarrow \\ 0, & si \quad F_\iota^{(n)}(x_1, \quad, x_n) \uparrow \end{cases}$$

Aunque la función permite determinar si una función es definida o no para las $n-$tupla $(x_1, \quad, x_n)$, la función $D^{(n+1)}(\iota, x_1, \quad, x_n)$ no es algorítmica en una familia de algoritmos A

Teorema 1.2.2 Sean A alguna familia de algoritmos que cumple las propiedades de 1, 2, 3 y 4 Entonces sin valor de n, A no contiene un algoritmo que evalúa la funcion dominio $D^{(n+1)}(\iota, x_1, \quad, x_n)$

Demostración

La prueba se realiza para un caso particular, $n = 1$, supongamos que

$$D^{(2)}(\iota, x) = \begin{cases} 1, & si\ F_\iota(x) \downarrow \\ 0, & si\ F_\iota(x) \uparrow \end{cases}$$

es algoritmica

Tomemos un índice a de $C_0^{(1)}$ y un indice b de $\Phi^{(1)}$ ambas algoritmicas (por la prop 1 y corolario 1 2 1) Definamos una función $\widehat{D}$ por medio de la composición de funciones algoritmicas,

$$\widehat{D}(x) = U^{(2)}\left(\Lambda\left(D^{(2)}(x,x), 0, a, b\right), x\right) = \begin{cases} U^{(2)}(a,x), si \quad D^{(2)}(x,x) = 0 \\ U^{(2)}(b,x), si \quad D^{(2)}(x,x) \neq 0 \end{cases}$$
$$= \begin{cases} 0, & si\ D^{(2)}(x,x) = 0 \\ \uparrow, & si\ D^{(2)}(x,x) \neq 0 \end{cases}$$

Entonces $\widehat{D}(x)$ es algoritmica

Consideremos una índice w de la funcion D, entonces

$$\widehat{D}(x) = \begin{cases} 0, & si\ \tilde{D}^{(2)}(x,x) = 0 \\ \uparrow, & si\ D^{(2)}(x,x) \neq 0 \end{cases}$$
$$= \begin{cases} 0, & si\ F_w^{(1)}(w)\uparrow \\ \uparrow, & si\ F_w^{(1)}(w)\downarrow \end{cases}$$
$$= \begin{cases} 0, & si\ \widehat{D}(x)\uparrow \\ \uparrow, & si\ \widehat{D}(x)\downarrow \end{cases}$$

lo que es una contradicción, por lo que $D^{(2)}$ no puede ser algoritmica El mismo procedimiento se realiza para el caso general, lo que se demuestra que $D^{(n+1)}$ no es algorítmica □

Una versión restringida de la función dominio y que además es no algorítmica en una familia de algoritmos que cumple las propiedades 1, 2, 3 y 4 es la función de dominio diagonal para la familia A

Definición 1.2.7 Llamaremos la función de dominio diagonal para una familia de algoritmos A, a la función de una variable dada por

$$d(x) = \begin{cases} 1, & si\ F_x^{(1)}(x)\downarrow \\ 0, & si\ F_x^{(1)}(x)\uparrow \end{cases}$$

La característica de finitud, de un algoritmo, es un limitante importante Es posible que una familia de algoritmos contenga un numero contable de miembros que por el contrario el número de funciones aritmeticas de n variables es incontable (no-numerable) por lo que hay más funciones que algoritmos en la familia dada

Ejemplo 5 Consideremos la función aritmética de una variable

$$f(x) = \begin{cases} 1, & si\ una\ ejecución\ consecutiva\ de\ exactamente\ 5\ x, \\ & ocurre\ en\ la\ expasión\ decimal\ de\ \pi \\ 0, & de\ otra\ forma \end{cases}$$

Hasta el momento no se conoce un algoritmo que calcule f

Observemos que existen funciones no algorítmicas, para el cual dada una familia de algoritmos $A = \{A_i\}_{i=1}^{n}$ es no computable por los miembros de esta familia Una de las técnicas para especificar esto se conoce como *diagonalización*[2]

En la técnica de diagonalización las funciones computables se manejan como objetos, los atributos que definen a una funcion seran los resultados que dicha función asocie a cada uno de sus argumentos Su objetivo es especificar para una familia de algoritmos dada A, una función de una variable g que no sea computable en A y que difiera de algún modo de cada una de las funciones que son computables en A Esto es definir g de manera que para cada número natural i, el valor de g difiere del valor de la función $F_i^{(1)}$ para algun argumento i Así definiremos a g en términos del valor encontrado a lo largo de la diagonal principal de una matriz En general las funciones pueden tener cualquier número de argumentos, pero al aplicar la técnica conviene reducirla a funciones de una variable para simplificar la construcción de la función diagonal Observemos en la figura 1 que la j-ésima fila y la k-ésima columna es el valor $F_j^{(1)}(k)$

	$F_0^{(1)}$	$F_1^{(1)}$	$F_2^{(1)}$	$F_3^{(1)}$
0	$F_0^{(1)}(0)$	$F_1^{(1)}(0)$	$F_2^{(1)}(0)$	$F_3^{(1)}(0)$
1	$F_0^{(1)}(1)$	$F_1^{(1)}(1)$	$F_2^{(1)}(0)$	$F_3^{(1)}(1)$
2	$F_0^{(1)}(2)$		$F_2^{(1)}(2)$	$F_3^{(1)}(2)$
3	$F_0^{(1)}(3)$			$F_3^{(1)}(3)$

Figura 1
Matriz de la técnica de diagonalización

[2] La tecnica de diagonalización es la más básica y bastante conocida, Cantor la aplico por primera vez en 1873 en la Teoría de Conjuntos para demostrar la existencia de conjuntos no numerables

Ejemplo 6 Definamos la función g no computable, como sigue

$$g(x) = \begin{cases} x+1, & si\ F_x^{(1)}(x)+1 > 3 \\ 0 \quad , & si\ F_x^{(1)}(x)+1 \leq 3 \end{cases}$$

Supongamos que $g(x)$ es computable, así para un valor de x, $F_x^{(1)}(x) > 2$ basta probar que $g(x) > 0$ Definamos una función diagonal d distintas de todas las funciones computables, por lo que para un valor de x resulta que

- Si $g(x) > 0$ entonces $F_x^{(1)}(x) > 2$ obteniendo para todo valor de x que $d(x) \neq F_x^{(1)}(x)$, es decir que $d(x) \leq 2$ Para esto tomemos en particular $d(x) = 0$
- Si $g(x) = 0$ entonces $F_x^{(1)}(x) \leq 2$, significa que $d(x) > 2$ Tomemos en particular $d(x) = 5$

Para el análisis, los posibles resultados se representan en la tabla dada en la figura siguiente

x	d	0	1	2	3
0	5	$F_0^{(1)}(0) \leq 2$	?	?	?
1	0	?	$F_1^{(1)}(1) > 2$	?	?
2	5	?	?	$F_2^{(1)}(2) \leq 2$	?
3	0	?	?	?	$F_3^{(1)}(3) > 2$

Figura 2
Tabla de resultados de g(x)

La tabla muestra en la parte izquierda, la columna con los valores dados de $d(x)$ para distinguirse de las otras y de cada una de las funciones de la tabla y esencialmente de las funciones que recorren la diagonal principal En la parte superior se puede deducir los valores que puede tomar $g(x)$ Con estos datos y suponiendo que $g(x)$ es computable, tenemos la función computable

$$d(x) = \begin{cases} 5, & si\ g(x) = 0 \\ 0, & si\ g(x) > 0 \end{cases}$$

Por tal motivo tomemos un índice e de la función d tal que $d = F_e$ y se deduce

✓ Si $d(e) = 0 \Longrightarrow F_e(e) = 0 \Longrightarrow F_e(e) \leq 2 \Longrightarrow g(e) = 0 \Longrightarrow d(e) = 5 \Rightarrow d(e) \neq 0$

✓ Si $d(e) \neq 0 \Longrightarrow d(e) = 5 \Longrightarrow F_e(e) > 2 \Longrightarrow g(e) = e + 1 \Longrightarrow g(e) > 0 \Rightarrow d(e) = 0$

Lo que es un absurdo, demostrando que $g(x)$ es no computable

Una característica[3] que tienen los miembros de una familia de algoritmos es que las funciones evaluadas, por los mismos, pueden ser utilizadas para modificar o crear otras nuevas funciones Esto nos lleva a preguntarnos ¿bajo qué condiciones estas nuevas funciones son computables? ¿Tendrían las mismas características que las originales?

Este proceso lo trataremos de formalizar, si consideramos una familia de algoritmos A que cumplen las propiedades 1 a 2, y que $F_t^{(2)}$ sea una de las funciones de dos variables computable por algún miembro de A Para cada $x \in \mathbb{N}$, tomemos una función de una variable f de manera que

$$f_x(y) = F_t^{(2)}(x, y) \qquad \text{para todo } y$$

donde la variable x no es un índice sino parte del nombre de f y

$$f_x(y) = F_t^{(2)} \circ \left(C_x^{(1)}, P_1^{(1)}(y)\right)$$

entonces f es algorítmica Además tiene uno o más índice y el valor de los mismos dependen de t y x, por lo que nos interesa una funcion total s, que sera como índice

Definición 1.2.8 Sean A una familia de algoritmos, a la función total s tal que

[3] Características que se pueden observar en los compiladores y sistemas operativos entre otros programas

$$F^{(1)}_{s(\iota,x)}(y) = f_x(y) = F^{(2)}_{\iota}(x,y)$$

para todo natural ι y x, la función s se le conoce como *funcion de índice* computable de f_x

De modo general, como mencionamos, para una selección arbitraria de las variables ι y x en s, f_x tendrá más de un índice Esto se formaliza con la siguiente propiedad llamada la propiedad s-m-n

Propiedad 5 Una familia de algoritmos A satisface la propiedad s-m-n, si para cada selección de enteros positivos m, n , la familia contiene un algoritmo que evalúe la función 1-1 y total s^m_n de $(m+1)$ variables tal que para todo $\iota, x_1, \quad , x_m, y_1, y_2, \quad , y_n$

$$\bar{F}^{(n)}_{s^m_n(\iota,x_1, \ ,x_m)}(y_1, y_2, \quad , y_n) = \bar{F}^{(m+n)}_{\iota}(x_1, \quad , x_m, y_1, y_2, \quad , y_n)$$

En esta propiedad sólo se requiere que la función de índice computable sea algorítmica, aunque no tienen algún interés particular para el estudio, cuando se cumple la propiedad 5 en conjunción con algunas otras propiedades discutidas previamente puede ser de gran utilidad para establecer la computabilidad de una amplia variedad de funciones de índice computables Como se demuestra en los siguientes teoremas

Teorema 1.2.3 Sean A una familia de algoritmos que cumple las propiedades de 1 a 5 Entonces debe existir una función algorítmica total k de una variable tal que

$$F^{(1)}_{k(x)} = C^{(1)}_x$$

Demostración Definamos la función de dos variables

$$g(x,y) = C^{(1)}_x(y) = x$$

Vemos que $g(x,y) = P^{(2)}_1$, por lo que g es algorítmica en A Así, tomemos w como un índice para g y por la propiedad 5 obtenemos

$$F^{(1)}_{s(w,x)}(y) = F^{(2)}_{w}(x,y) = g(x,y) = C^{(1)}_{x}(y)$$

Si hacemos $k = s \circ \left(C^{(1)}_{w}, C^{(1)}_{x}\right)$ tenemos la función de índice computable deseada, por lo que k es una función total □

De la misma forma podemos utilizar las funciones para caracterizar la propiedad de cerradura, como la composición de funciones algorítmicas

Teorema 1.2.4 Sean A una familia de algoritmos que cumple las propiedades de 1 a 5 Entonces existe una función total algorítmica de dos variables h tal que

$$F^{(1)}_{h(x_1,x_2)} = F^{(1)}_{x_1} \circ F^{(1)}_{x_2}$$

Demostración Consideremos la función g de tres variables definida de la forma

$$g(x_1,x_2,y) = F^{(1)}_{x_1}\left(F^{(1)}_{x_2}(y)\right) = U^{(2)}\left(x_1, U^{(2)}(x_2,y)\right)$$

Por los que la función g es algorítmica en A, ya que

$$g = U^{(2)} \circ \left(P^{(3)}_{1}, U^{(2)} \circ (P^{(3)}_{2}, P^{(3)}_{3})\right)$$

Entonces existe un índice w de g Por la propiedad s-m-n, se tiene una funcion algoritmica total s de tres variables, tal que

$$F^{(1)}_{s^2_1(w,x_1,x_2)}(y) = F^{(3)}_{w}(x_1,x_2,y) = g(x_1,x_2,y) = F^{(1)}_{x_1}\left(F^{(1)}_{x_2}(y)\right)$$

Haciendo $h = s \circ \left(C^{(1)}_{w}, P^{(3)}_{1}, P^{(3)}_{2}\right)$ que resulta ser la función de índice buscada □

1.3 Función Recursiva y Teorema de Recursión

En computación la noción de recursión[4] es fundamental Una definición recursiva de una función, de manera informal, es una definición en que los valores de la funcion, para argumentos dados, están directamente relacionados con los valores de la misma función para argumentos más simples o para valores simples de la función Por ejemplo la función f $\mathbb{N} \to \mathbb{N}$, definida por

$$f(0) = 1,$$
$$f(n+1) = n \quad f(n)$$

Las funciones primitiva-recursivas son ejemplos de amplias e interesantes clases de funciones que pueden ser obtenidas por una caracterización formal

Definición 1.3.1 Sean g una función aritmética total de n variables y h una función aritmetica total con $(n+2)$ variables Definimos la *función primitiva-recursiva* f de $(n+1)$ variables a partir de g y h, de la forma

$$f(x_1, \quad, x_n, 0) = g(x_1, \quad, x_n),$$
$$f(x_1, \quad, x_n, y+1) = h(x_1, \quad, x_n, y, f(x_1, \quad, x_n, y)) \ \textit{para todo } y$$

En la definición f es obtenida por recursión (primitivo) en y Los parámetros $x_1, \quad, x_n$ son referidos como parametros de la recursión (primitivo)

Observemos que la definición 1 3 1, puede ser vista como una regla de composicion para obtener una nueva función f de las funciones dadas h y g Esta regla de composición juega un papel importante para el estudio posterior de funciones efectivamente computables

[4] El concepto de recursión se deriva del verbo "to recur" que significa volver a un lugar o situación Antes del siglo XIX se conocía como función por inducción y Peano (1889-1891), utilizó el trabajo de Dedekind (1888) en su quinto axioma usando la definición por inducción que posteriormente se le llamó recursión primitiva

La existencia y unicidad de una función f en la definición 1 3 1, se formaliza en el siguiente teorema, llamado *teorema de recursión I*

Teorema 1.3.1 Sean g y h dos funciones aritmeticas totales de n y $(n+2)$ variables respectivamente Entonces existe una única función total de $(n+1)$ variables f tal que

$$f(x_1, \quad, x_n, 0) = g(x_1, \quad, x_n)$$

y

$$f(x_1, \quad, x_n, y+1) = h(x_1, \quad, x_n, y, f(x_1, \quad, x_n, y))$$

Demostración Por la complejidad y extensión, sólo demostraremos la unicidad de la función f, el resto se puede revisar en (Soare, 1987)

Supongamos que tenemos dos funciones f_1 y f_2 de $(n+1)$ variables que para todo $\vec{x} = x_1, \quad, x_n$ e y enteros positivos, satisfacen

$$f_1(\vec{x}, 0) = g(\vec{x}) = f_2(\vec{x}, 0),$$

$$f_1(\vec{x}, y+1) = h(\vec{x}, y, f_1(\vec{x}, y)),$$
$$f_2(\vec{x}, y+1) = h(\vec{x}, y, f_2(\vec{x}, y)),$$

Para n −tuplas $\vec{z} = z_1, \quad, z_n$, tomemos el conjunto

$$V = \{\vec{z} \; e \; y \; enteros \; positivos \quad f_1(\vec{z}, y) = f_2(\vec{z}, y)\}$$

Vemos que $y = 0, \vec{x} \in V$ Entonces

$$\begin{aligned} f_1(\vec{z}, y+1) &= h(\vec{z}, y, f_1(\vec{z}, y)) \\ &= h(\vec{x}, y, f_2(\vec{z}, y)) \\ &= f_2(\vec{z}, y+1) \end{aligned}$$

Lo cual implica que $(y+1) \in V$ y que $V = \mathbb{Z}^+$ Por inducción, las funciones son las mismas □

Las funciones primitiva-recursivas constituyen una clase muy amplia de funciones computables que contienen casi todas las funciones aritméticas que se encuentra comúnmente en las Matemáticas.

Definición 1.3.2: La clase de funciones primitiva-recursivas es la clase más pequeña $\mathcal{C}$ de funciones cerrada bajo el siguiente esquema:

i. La función sucesor S está en $\mathcal{C}$.

ii. Las funciones constante $C_i^{(n)}$ están en $\mathcal{C}$.

iii. La función proyección $P_i^{(n)}(x_1, x_2, \dots, x_i, \dots, x_n)$ está en $\mathcal{C}$.

iv. (Composición) Si $g_1, \dots, g_m, h \in \mathcal{C}$ son funciones de n y m variables, respectivamente, entonces

$$f(x_1, x_2, \dots, x_n) = h(g_1(x_1, x_2, \dots, x_n), \dots, g_m(x_1, x_2, \dots, x_n))$$

está en $\mathcal{C}$.

v. (Recursión primitiva) Para $n \geq 1$, si $g, h \in \mathcal{C}$ son funciones de n y $(n+2)$ variables, respectivamente, entonces $f \in \mathcal{C}$, donde

$$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$$

y

$$f(x_1, \dots, x_n, y+1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)).$$

Por lo tanto, una función es primitiva-recursiva si sólo si es una derivación formal de primitivo-recursivas. Es decir, existe una sucesión $f = f_1, \dots, f_k$ tal que, para cada $i \leq k, f_i$ es una función inicial (obtenida de (i), (ii) o (iii)), o es obtenida desde $\{f_j : j < i,\}$ por una aplicación de (iv) o (v).

Ejemplo 7: La función $suma(x, y) = x + y$ es definida por recursión primitivo, por las ecuaciones:

$$suma(x, 0) = x,$$
$$suma(x, y+1) = 1 + suma(x, y)$$

es decir

$$suma(x,0) = P_1^{(1)}(x),$$
$$suma(x,y+1) = S\left(P_3^{(3)}(x,y,suma(x,y))\right)$$

donde $g = P_1^{(1)}, h = S \circ P_3^{(3)}$

La noción de una derivación formal a menudo desempeña un papel clave para establecer las propiedades de las funciones primitiva-recursivas

Teorema 1.3.2 Toda función primitiva-recursiva es total

Demostración Tomemos una derivación formal de función primitiva-recursiva de la forma $f = f_1, \quad , f_m$ La demostración se realizará por inducción sobre la longitud de derivaciones de f

Para el caso $f = f_1$, la función debe ser la función constante, la función proyección o la función sucesor Cualquier caso f es una función total

Asumiremos que todas las funciones recursivas $f_1, \quad , f_k$ que tienen derivación cuya longitud es $k \leq m$, son funciones totales Si f es obtenida de funciones primitiva-recursivas $f_1, f_2, \quad , f_k$ y f_{k+1} por medio de la composición funcional, entonces las últimas funciones deben tener derivaciones de longitud menor o igual que m, lo que implica que $f_1, f_2, \quad , f_k$ y f_{k+1} son funciones totales (por la hipótesis de inducción) y por el teorema 1 3 1 f es una función total □

En forma similar, Kleene[5] mostró que todas las funciones aritméticas habituales son primitiva-recursivas, entre ellas, $x \ y$, x^y, $x!$ y la función conocida como sustracción propia

[5] Stephen Kleene (1909-1994) Matemático norteamericano que se especializó en la teoría de las funciones recursivas y Teoría de Autómatas Participó en el desarrollo del campo de la teoría de la

$$x-y=\begin{cases}x-y & si\ x\geq y\\ 0 & si\ x<y\end{cases}$$

En (Hennie, Introduction to Computability, 1977) se presenta una variedad de derivaciones, de manera informal, de funciones primitiva-recursivas de gran utilidad

Podemos entender fácilmente cómo una función puede ser construida a partir de funciones primitiva-recursivas, usando la composición, la adición y la multiplicación de funciones De esta forma es posible dar una definición de una nueva función con un operador conocido como el operador de *minimización acotada* Este operador busca el número mínimo que satisfaga una condición dada, en un intervalo dado y la condición se puede especificar utilizando un predicado recursivo primitivo, el cual analizaremos posteriormente Esta nueva función se define a continuación

Definición 1.3.3 Sea g una función total y sean $\vec{x}=x_1,x_2,\quad ,x_n$ e y números naturales, la minimización acotada que convierte la función g en la nueva función f de $(n+1)$ variables, para un valor de y, se define como

$$f(\vec{x},z)=min_{y\leq z}[g(\vec{x},y)=0]$$

En la expresión, dada en la definicion anterior, el valor de z se le conoce como cota (o límite) de la minimización Observemos que para el caso de que no exista tal valor de y, con $0\leq y\ \leq\ z$, el valor de la función es $z+1$ Ademas la funcion f debe ser necesariamente una función total ya que es definida de g por una minimización acotada, lo que se demostrará en el siguiente teorema, para esto primero presentamos el lema

Lema 1.3.1 Sea g una función primitiva-recursiva de $(n+1)$ variables entonces la funcion de $(n+1)$ variables h, es primitiva-recursiva si

recursividad en conjunto con Church, Gödel, Turing entre otros Escribió diferentes articulos y libros, destacando *Introduccion a la Metamatemática (1952)* y *Lógica Matematica (1967)*

$$h(\vec{x},k)=\begin{cases}1, & si\ g(\vec{x},y)\neq 0, para\ cada\ valor\ de\ z \\ & \quad con\ 0\leq z\leq k \\ 0, & en\ otro\ caso\end{cases}$$

Demostración Como g es una función primitiva-recursiva para cada $0 \leq y \leq k$ se puede definir por medio de la función $sig(g(\vec{x},y))$[6], de esta forma $h(\vec{x},k)$ tiene el valor de 1, entonces

$$h(\vec{x},k)=\prod_{y=0}^{k} sig\big(g(\vec{x},y)\big)$$

es obtenida del producto de funciones primitiva-recursivas, por lo que h es primitiva-recursiva. □

El lema anterior nos facilita deducir el siguiente teorema

Teorema 1.3.3 Sean g una función primitiva-recursiva de $(n+1)$ variables y f una función definida de g por minimizacion acotada, entonces f es primitiva-recursiva si

$$f(\vec{x},z)=min_{y\leq z}[g(\vec{x},y)=0]$$

Demostración Observemos que el valor de $f(\vec{x},z)$ puede ser el mínimo valor $y \leq z$ para el cual $g(\vec{x},y)=0$, o que no exista un valor y con $z+1$ En ambos casos el valor de $f(\vec{x},z)$ coincide sólo con el numero de valores de un $0 \leq k \leq z$ de manera que $g(\vec{x},y)\neq 0$ para todo $y \leq k$ Así podemos escribir

$$f(\vec{x},z)=\sum_{k=0}^{z} h(\vec{x},y)$$

Por el lema 1 3 1, h es una funcion primitiva-recursiva, lo que implica que f también lo es □

[6] $sig(x)=\begin{cases}1, si\ x>0\\ 0, si\ x=0\end{cases}$ es la función signo recursiva primitiva en Hennie[1977, p p 169]

1.4 Predicados primitivo-recursivos

En este punto introduciremos la noción de predicado como un medio para establecer ciertas relaciones o condiciones Los predicados están estrechamente relacionados a las funciones, ya que nos provee de métodos para describir y definir funciones especiales primitiva-recursivas Su estudio se desarrollará en el siguiente capítulo

Consideremos una $n-$tupla $\vec{x} = (x_1, x_2, \quad, x_n)$, un predicado es una función $P\ \mathbb{N}^n \to \{0,1\}$ de manera que, por extensión, tendremos el conjunto $\{\vec{x} \in \mathbb{N}\ P(\vec{x})\}$ es verdadero si $P(\vec{x})$ es verdadero, para una selección de argumentos $\vec{x}$

Definición 1.4.1 La función $\mathcal{X}_P(\vec{x})$ de un predicado $P\ n-$ ario, se le llama función característica de su extensión si

$$\mathcal{X}_P(\vec{x}) = \begin{cases} 1, & si\ P(\vec{x})^7 \\ 0, & si\ \neg P(\vec{x}) \end{cases}$$

Un predicado (relación) es primitivo-recursivo si su función característica es primitiva-recursiva

Ejemplo 8 Mostraremos que la relación $\mathcal{R} = \{x\ x\ es\ primo\}$ es recursiva primitivo Sea $p_0, p_1,$ los primeros numeros primos en orden creciente Cualquier $x \in \{0, 1, 2, 3, \}$ tiene una única representación de la forma

$$x = p_0^{x_0} p_1^{x_1} \quad p_n^{x_n}$$

Para un numero finito $x_i \neq 0$ Decimos que cada uno de los exponentes x_i, en esta factorizacion, está determinado únicamente por x por lo que podemos definir la funcion

[7] $P(\vec{x})$ indicará que el predicado $n-$ario es verdadero

$x(\iota)$ como el exponente x_ι que se produce en el ι − ésimo factor primo, lo que resulta ser una función total y consecuentemente primitivo-recursivo[8]

El trabajo que nos compete se desarrollará exclusivamente con predicados que denominaremos *predicados aritméticos*, que son aquellos predicados cuyo dominio es el conjunto de numero naturales, lo cual nos permitirá hacer uso de operadores aritméticos

Las expresiones $P(\vec{x})$ y $Q(\vec{x})$ representan enunciados, los cuales se le pueden aplicar las conectivas " $\wedge$ ", " $\vee$ " y "$\neg$" El símbolo $\Leftrightarrow$ se utiliza para indicar cuando dos predicados son equivalentes, lo que significa que tiene la misma extension, es decir

$$\{\vec{x} \in \mathbb{N}\ P(\vec{x})\} = \{\vec{x} \in \mathbb{N}\ Q(\vec{x})\}$$

y se denota

$$P(\vec{x}) \Leftrightarrow Q(\vec{x})$$

Para obtener nuevos predicados primitivos recursivos, se aplican las técnicas composición de funciones, conectivas lógicas y cuantificadores, las cuales analizaremos a continuacion

Teorema 1.4.1 Sea Q un predicado primitivo recursivo de m − ario y sea $g_1, g_2, \quad , g_m$ funciones primitiva-recursivas de n variables, entonces P es un predicado primitivo recursivo de n − ario si

$$P(\vec{x}) \Leftrightarrow Q\big(g_1(\vec{x}), \quad , g_m(\vec{x})\big)$$

Demostración

Sean $\mathcal{X}_P$ y $\mathcal{X}_Q$ las funciones características de los predicados P y Q, respectivamente Por la equivalencia de P, su función característica está dada por

[8] Prueba formal en (HEDMAN, 2006)

$$\mathcal{X}_P = \mathcal{X}_Q \circ (g_1, g_2, \quad , g_m)$$

Como $\mathcal{X}_Q$ y $g_1, g_2, \quad , g_m$ son primitiva-recursivas, $\hat{\mathcal{X}_P}$ es primitiva-recursiva y en consecuencia P lo es □

Mencionamos que un segundo camino para construir predicados recursivos a partir de otros es utilizando conectivas lógicas Esto se afirma en el siguiente teorema

Teorema 1.4.2 Sean P, Q predicados $n-$ arios primitivo-recursivos, entonces para una selección de $\vec{x}$ los predicados $P \vee Q$, $P \wedge Q$ y $\neg P$ son primitivo-recursivos

Demostracion Sean $\mathcal{X}_{P\vee Q}$, $\mathcal{X}_{P\wedge Q}$ y $\mathcal{X}_{\neg P}$, las funciones característica, respectivas de los predicados $P \vee Q$, $P \wedge Q$ y $\neg P$, entonces

$$\begin{aligned} \mathcal{X}_{P\vee Q} &= \mathcal{X}_P + \mathcal{X}_Q - \mathcal{X}_P \quad \mathcal{X}_Q, \\ \mathcal{X}_{P\wedge Q} &= \mathcal{X}_P \quad \mathcal{X}_Q, \\ \mathcal{X}_{\neg P} &\equiv (1 - \mathcal{X}_P) \end{aligned}$$

resultando primitiva-recursivas □

Para la aplicación del teorema 1 4 2, es necesario que los predicados tengan los mismos argumentos

El tercer camino para la construcción de funciones primitiva-recursivas es el uso de cuantificadores Así si tenemos el predicado $P(\vec{x}, y)$, $(n+1)-$ario, podemos definir nuevos predicado por medio un cuantificador existencial acotado $(\vee_y)$ y de cuantificador universal acotado $(\wedge_y)$, tal que, para el primero

$$\bigvee_y^z P(\vec{x}, y) \Leftrightarrow P(\vec{x}, 0) \vee P(\vec{x}, 1) \vee P(\vec{x}, 2) \vee \quad \vee P(\vec{x}, z)$$

nos indica que el predicado $(n+1)$ −ario es verdadero si y sólo si $P(\vec{x}, y)$ es verdadero para al menos un número $y \in \mathbb{N}$ tal que $0 \leq y \leq z$ Para el segundo cuantificador tenemos que

$$\bigwedge_{y}^{z} P(\vec{x}, y) \Leftrightarrow P(\vec{x}, 0) \wedge P(\vec{x}, 1) \wedge P(\vec{x}, 2) \wedge \quad \wedge P(\vec{x}, z)$$

se lee "para todo $y \in \mathbb{N}$ con $0 \leq y \leq z$ se cumple $P(\vec{x}, y)$"

Teorema 1.4.3 Sea $P(\vec{x}, y)$ un predicado $(n+1)-$ario primitivo-recursivo Entonces los predicados

$$\bigvee_{y=0}^{z} P(\vec{x}, y) \ y \ \bigwedge_{y=0}^{z} P(\vec{x}, y)$$

son primitivo-recursivos

Demostración Sea $\mathcal{X}_P$ la función característica del predicado P primitivo-recursivo y sea $\mathcal{X}_Q$ la función característica del predicado definido por

$$Q(\vec{x}, y) \Leftrightarrow \bigwedge_{y=0}^{z} P(\vec{x}, y)$$

entonces

$$\mathcal{X}_Q(\vec{x}, y) = \prod_{y=0}^{z} \mathcal{X}_P(\vec{x}, y)$$

es primitiva-recursiva lo que resulta que Q es primitivo-recursivo Igualmente consideremos

$$R(\vec{x}, y) \Leftrightarrow \bigvee_{y=0}^{z} P(\vec{x}, y)$$

y su función característica, dada por

$$\chi_R(\vec{x}, y) = sig\left(\sum_{y=0}^{z} \chi_P(\vec{x}, y)\right)$$

que es primitiva-recursiva, por lo que R es un predicado primitivo-recursivo □

El uso de predicados permite simplificar la definición de funciones de minimización acotada De esta forma, en lugar de buscar el número mínimo que satisface una condición dada, en un intervalo dado podemos pensar en definir el menor número natural en el cual el predicado dado sea verdadero

Definición 1.4.2 Sea $P(\vec{x}, z)$ un predicado $(n+1)-$ ario La expresión $min_{y \leq z}[P(\vec{x}, y)]$ denota el menor valor de y, con $0 \leq y \leq z$ para el cual el predicado $P(\vec{x}, z)$ es verdadero Una función total f de $(n+1)$ variables, es definida por un predicado de minimización acotada si

$$f(\vec{x}, z) = min_{y \leq z}[P(\vec{x}, y)]$$

Si no existe tal valor de y para el cual satisfaga la ecuación dada en la definición 1 4 2, entonces $f(\vec{x}, z) = 0$, por definición

Teorema 1.4.4 Sea $P(\vec{x}, y)$ un predicado primitivo-recursivo de $(n+1)-$ ario Entonces la función f de $(n+1)$ variables definida por

$$f(\vec{x}, z) = min_{y \leq z}[P(\vec{x}, y)]$$

es primitiva-recursiva

Demostración: La función f puede ser expresada en términos de la función característica de P. Sea χ_P la función característica de P, entonces

$$f(\vec{x}, z) = min_{y \leq z}[\chi_P(\vec{x}, y) = 1].$$

Por lo que f es primitiva-recursiva.

Utilizando los métodos expuestos, es fácil establecer que las siguientes funciones y predicados son primitivo-recursivos:

1. Predicado de decibilidad: $y \,/\, x$ es verdadero si y sólo si y es divisor de x. Es primitivo-recursivo, ya que

$$y \,/\, x \Leftrightarrow \bigvee_{z=0}^{x}(x = yz).$$

2. Predicado de números primos: $Primo(x)$ es verdadero si y sólo si x es un número primo. Es primitivo-recursivo, pues

$$Primo(x) \Leftrightarrow (x > 1) \wedge \bigwedge_{y=0}^{x}[(y = 1) \vee (y = x) \vee \neg(y \,/\, x)].$$

3. Función codificadora de Cantor: $\pi: \mathbb{N}^2 \rightarrow \mathbb{N}$ es la biyección dada por

$$\pi(x, y) = \lfloor\{(x + y)^2 + 3x + y\} \,/\, 2\rfloor.$$

4. Funciones descodificadoras de Cantor: Se define como las funcione unarias σ_1 y σ_2 tal que $\pi(\sigma_1(z), \sigma_2(z)) = z$, para todo $z \in \mathbb{N}$ y

$$\sigma_1(z) = min_{x \leq z} \bigvee_{y=0}^{z}[z = \pi(x, y)], \qquad \sigma_2(z) = min_{y \leq z} \bigvee_{x=0}^{z}[z = \pi(x, y)].$$

1.5 Funciones Turing Computables.

Un concepto importante de la teoría de la computabilidad es la noción de una función efectivamente calculable o computable, visto en la definición 1 2 4 El trabajo realizado por A M Turing[9] en 1936, sobre el concepto de computabilidad, centra su interés en el proceso de computación mas que en la naturaleza de las funciones computables

Las funciones definidas por Turing se construyen a partir de funciones muy elementales que no se dude de su propiedad algorítmica Este modelo de dispositivo de cómputo se le conoce como Máquinas de Turing (MT)

Por su importancia sólo haremos una descripción del mecanismo de este dispositivo y como evalúan funciones aritméticas, así como su representación a una familia de algoritmos que satisfaga las cinco (5) propiedades expuestas en 1 2,

Definición 1.5.1 Una máquina de Turing la podemos definir como una estructura de la forma $MT = (\Sigma, S, \delta, \Gamma, \iota, h)$ en donde

- $\Sigma \neq \emptyset$ Es un conjunto finito de símbolos de entrada llamado alfabeto, distinto al símbolo en blanco (B)
- $S \neq \emptyset$ Es un conjunto finito llamado conjunto de estados
- δ Es la función de transición de la máquina
- Γ Es el conjunto finito de símbolo de la cinta de la máquina, incluyendo a Σ
- $\iota \in S$ Es el estado inicial
- $h \subseteq S$ Es el estado de parada o estado final

[9] A M Turing (1912-1954) Matemático inglés, en 1934 se graduó del King's College de la Universidad de Cambrige y fue miembro del mismo, por su investigación sobre *la funcion de error gaussiana* Se interesó en la lógica y en 1936 publica su artículo *sobre los numeros computables con una aplicacion al Entscheidungsproblem*, en donde introduce por primera vez las Máquinas de Turing

La MT, cuenta con una o varias cintas infinitas divididas en celda y con una cabeza lectora que escanea una celda de la cinta a la vez, como se observa en la figura 3 Del conjunto de estados $S = \{s_1, s_2, \quad , s_n\}$ contiene al menos dos estados, uno inicial y otro denominado estado de parada o final, dentro de sus acciones están el de leer y escribir en su medio de entrada (cinta) Puede escribir símbolos que no pertenecen al alfabeto y emplea marcas especiales llamadas simbolos de cinta de la máquina como B para indicar el espacio en blanco (en este trabajo se usará el cero (0)), que no pertenece al alfabeto Entendemos como *configuración de la cinta* de la MT a la representación de los símbolos presentes en la cinta, en una etapa dada, en una unica posición para analizar el simbolo a la derecha de la ubicación del lector

En un paso simple la máquina podría simultáneamente (1) reemplazar un símbolo de la cinta por otro y después cambiar de estado, (2) mover la cabeza a una celda a la derecha (R) o a una celda a la izquierda (L), y (3) pasar a un nuevo estado La operación de la MT es contralada por una aplicación parcial $\delta \ (S' \times \Gamma) \rightarrow S \times (\Gamma \cup \{L, R\})$, llamada transición o Turing programa (que podrá ser indefinida para algunos argumentos) La transicion $(s_i, p, s_j, q, X) \in \delta$ indica que la MT en el estado s_i, lee el simbolo p y cambia al estado s_j, reemplazando p por q y mueve el lector para una celda a la derecha si $X = R$ (o izquierda si $X = L$)

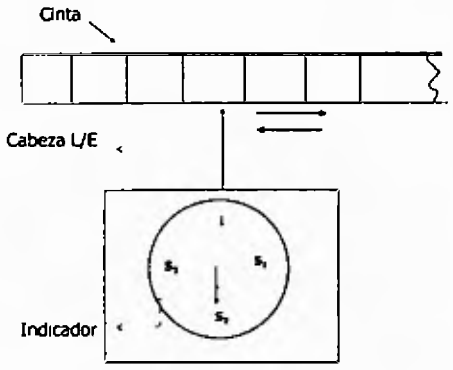

Figura 3
Mecanismo de Control

Si las máquinas de Turing se deben usar para evaluar las funciones aritméticas, se debe elegir un método para representar los números naturales y las n −tuplas de números naturales sobre la cinta de una MT Una de los métodos es el esquema unario descrito en (Hennie, Introduction to Computability, 1977) Recapitulando, en una representacion de esquema unario se utiliza el símbolo $\bar{n}$[10] para denotar *la representación unaria* para un $n \in \mathbb{N}$, que es la configuración de la cinta comprendido por cadenas consecutivas de 1 de longitud $n + 1$ De esta forma la k-tupla $(n_1, \quad , n_k)$ representará la configuracion de la cinta $\underline{0}\bar{n}_1 0 \bar{n}_2 0 \quad 0\bar{n}_k$

Definición 1.5.2 Decimos que una función aritmética de f de n variables, es una función Turing computable (o computable) si es evaluada por una máquina de Turing (MT) y se denota $F^{(n)}(x_1, \quad , x_n)$

Describiremos, brevemente, la manera que la máquina de Turing evalua una función aritmética Consideremos una máquina de Turing de una cinta que computa la función aritmética f de n variables Se requerirá que la evaluacion de la función $f(\bar{x})$ inicie con la configuración siguiente

$$\underline{0}\bar{x}_1 0 \bar{x}_2 0 \quad 0\bar{x}_n 0$$

Todas las celdas a la izquierda de $\bar{x}_1$ y a la derecha de $\bar{x}_n$ están en blanco Así, el lector de la máquina inicia en la celda a la izquierda de $\bar{x}_1$, como indica el patron de la cinta descrita arriba Si $f(\bar{x})$ es definida, la maquina se detiene (termina) con la configuración $\underline{0}\overline{f(x_1, \quad , x_n)}0$, sobre la cinta Si $f(\bar{x})$ no es definida, pueda que la máquina falle para finalizar (problema de parada) o que se detenga en una configuración que no contenga bloques de unos a la derecha de la celda del mismo

Una computación de Turing de acuerdo a un programa de Turing P con una entrada x es una sucesión de configuraciones $c_0, c_1, \quad , c_n$ tal que c_0 representa la máquina en el estado inicial s_1, leyendo el símbolo más a la izquierda de la entrada x, c_n

[10] $\bar{3}$ denota el patrón de cinta 1111

representa la máquina en el estado de parada s_n y la transición $c_i \rightarrow c_{i+1}$, para $i < n$, es dado por el programa de Turing P De esta forma cuando hablemos de computación nos referimos a una parada (detención), esto es, cálculo convergente Una función parcial de n variables es asociada con cada MT mediante la entrada $(x_1, \quad , x_n)$ por la configuración inicial de la cinta y análisis descrito anteriormente

El conjunto de funciones Turing computables es cerrado bajo la composición funcional De manera similar a los argumentos anteriores pueden ser utilizados para establecer el cierre bajo una variedad de reglas, definir nuevas funciones y resolver el problema de determinar cuáles funciones son Turing computables

Es posible implementar las máquinas de Turing como miembros de una familia de algoritmos que denotaremos $\mathcal{T}$ y así asignarle un índice i (visto en I 2), para referirse a la $i-$ésima máquina de Turing MT_i que computa a la i-ésima función aritmética de n variables, $F_i^{(n)}$ Esto nos indica que una función dada podría ser computada por diferentes máquinas y que dado una máquina podría tener muchas descripciones diferentes Por lo que una máquina como una funcion, puede tener diferentes índices

Definición 1.5.3 Decimos que una función aritmética es computada en la forma estándar si cumple

a. Para una combinación de argumentos dados, la función es definida y la máquina que realiza el cálculo se detiene para dichos valores

b Si la máquina se detiene, su cinta contendrá solamente bloques de 1's consecutivos que representan el valor apropiado de la función y el resto de las celdas deben estar en blanco

c La máquina inicia y finaliza su calculo con la misma configuración

d La cabeza de lectura no puede analizar celdas a la izquierda de la celda inicial establecida en la configuración de la cinta dada

Teorema 1.5.1 Sean $g_1, g_2, \quad , g_m$ funciones Turing computables de n variables, y sea h una función Turing computable de m variables Entonces la función composición $f = h \circ (g_1, g_2, \quad , g_m)$ es Turing computable

Demostración Consideremos la $n-$ tuplas $\vec{x} = (x_1, x_2, \quad , x_n)$ y la composición $h \circ (g_1, g_2, \quad , g_m) = h\big(g_1(\vec{x}), g_2(\vec{x}), \quad , g_m(\vec{x})\big)$, en que la máquina evalúa primero las funciones $g_i(\vec{x}), i = 1,2, \quad , m$, con la configuración inicial de la cinta

$$\underline{0}\bar{x}_1 0 \bar{x}_2 0 \quad 0\bar{x}_n 0$$

En (Hennie, Introduction to Computability, 1977) se demuestra que existe una MT que convierte patrones de la forma anterior a un nuevo patrón inicial de la forma con una nueva configuración de la cinta

$$0\bar{x}_1 0 \bar{x}_2 0 \quad 0\bar{x}_n \underline{0} \quad \overline{g_1(x)}0 \quad 0\overline{g_m(x)}0$$

La máquina computa seguidamente la función h en la forma estandar con la configuración

$$0\bar{x}_1 0 \bar{x}_2 0 \quad 0\bar{x}_n \underline{0} \quad \overline{h\big(g_1(x), g_2(x), \quad , g_m(x)\big)}0$$

De esta forma se evalúa la función f para los argumentos $\vec{x} = (x_1, x_2, \quad , x_n)$ □

Dada la familia $\mathcal{T}$, es fácil mostrar que la función constante, la funcion proyección y la función sucesor son funciones Turing computables, de igual manera la función selección Λ Entonces la familia $\mathcal{T}$ satisface las propiedades 1 y 4 El teorema 1 5 1, nos asegura que el conjunto de funciones Turing computables es cerrado bajo la composición de funciones lo que implica que la familia $\bar{\mathcal{T}}$ satisface la propiedad $\hat{2}$, todas dadas en la sección 1 2

Para verificar que la familia $\mathcal{T}$ satisface las propiedades 3 y 5, se requiere de un amplio análisis, el cual no es el propósito de este trabajo, pero es importante destacar que

el resultado de la propiedad 3 génera el Teorema de Enumeración para las máquinaa de Turing, que establece la existencia de una máquina Universal de Turing De forma similar, la propiedad 5 (propiedad s-m-n), demuestra que la familia $\mathcal{T}$ satisface a lo que se refiere como el Teorema s-m-n para Maquinas de Turing

La recursión primitivo permite encontrar una función f efectivamente computable siempre que g y h lo sean, resultado analizado en el siguiente teorema

Teorema 1.5.2 Toda función primitiva-recursiva es Turing computable

Demostración Sabemos que una familia $\mathcal{T}$ satisface la propiedad 1, lo que nos indica que existe un procedimiento (recursión primitivo) para derivar las funciones básicas lo que las hace Turing computables

Para completar la prueba, es suficiente mostrar que las funciones Turing computables son cerradas bajo la composición y la recursión primitivo El teorema 1 5 1, establece la cerradura de la composición de las funciones Turing computables, sólo queda mostrar que las funciones Turing computables son cerradas bajo la recursión primitivo, es decir, si una función f es definida por recursión primitivo de funciones totales Turing computables g y h entonces f es Turing computable

Sean g y h funciones totales Turing computables y sea f una funcion definida por

$$f(x_1, \quad, x_n, 0) = g(x_1, \quad, x_n),$$
$$f(x_1, \quad, x_n, y+1) = h\big(x_1, \quad, x_n, y, f(x_1, \quad, x_n, y)\big)$$

Como g y h son Turing computables, existen las máquinas de Turing G y H que computan respectivamente las funciones Entonces se construirá una maquina de composición F que compute la función f

De esta forma la composición de $f(x_1, \quad, x_n, y)$ inicia con la configuración de la cinta

$$\underline{0}\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{y}0$$

con el siguiente procedimiento

1. Un contador, con valor inicial 0, se escribe en la celda derecha contigua a la entrada. El contador es utilizado para registrar el valor de la variable recursiva en la computación. Copiando el patrón completo de manera que la nueva configuración es

$$0\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{y}0\underline{\bar{0}}\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0$$

2. La máquina G se ejecuta en el final n, produciendo

$$0\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{y}0\bar{0}0\overline{g(x_1, \ldots, x_n)}0$$

La computación de G, genera

$$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$$

3. La configuración de la cinta tiene la forma

$$0\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{y}0\bar{\imath}0\overline{f(x_1, \ldots, x_n, i)}0$$

Si $i = y$, la computación de $f(x_1, \ldots, x_n, y)$ está completa eliminando los $n + 2$ números iniciales de la cinta y trasladando el resultado a la posición inicial de la cinta (uno).

4. Si $i < y$ la configuración de la cinta para evaluar el próximo valor de f es

$$0\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{y}0\overline{i+1}0\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{\imath}0\overline{f(x_1, \ldots, x_n, i)}0$$

La máquina H se ejecuta al final de los $n + 2$ valores de la cinta, produciento

$$0\bar{x}_1 0\bar{x}_2 0 \cdots 0\bar{x}_n 0\bar{y}0\overline{i+1}0\overline{h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, i))}0;$$

donde el valor más a la derecha de la cinta es $f(x_1, \quad, x_n, \iota + 1)$ La evaluación continúa en el paso 3

1.6 Numeración de Gödel[11]

Una forma simple de representar una sucesión de números naturales es descomponerlo en potencia de sus factores primos Por ejemplo 600 se puede representar por el producto de los números naturales de la forma $600 = 2^3 3^1 5^2$ Analizaremos un método para codificar y descodificar las n −tuplas de números naturales que sea de gran utilidad para simplificar algunas funciones primitiva-recursivas y posteriormente para establecer la computabilidad para una amplia clases de funciones

Definición 1.6.1 Dada una sucesión de números naturales $x_0, x_1, \quad, x_n$, se llama *numeración de Godel* al numero formal descrito por una secuencia de la forma $p_0^{x_0} p_1^{x_1} \quad p_n^{x_n}$, donde p_ι denota el ι −ésimo numero primo y el numero de Godel se denota como $\langle x_0, x_1, \quad, x_n \rangle = 2^{x_0} 3^{x_1} 5^{x_2} \cdot k^{x_n}$, con k como n −ésimo número primo

Con esta representación es posible encontrar distintas secuencias que tiene el mismo número de Gódel como se muestra en el siguiente ejemplo

Ejemplo 9 Las sucesiones $(1,2,3), (1,2,3,0)$ y $(1,2,3,0,0)$ tienen como número de Gödel 2250

Es sabido que un número natural distinto de cero tiene una única representación de descomposición de factores primos Observemos que para que dos o más sucesiones, con longitudes distintas, poseen un mismo número de Gödel, se requiere que la secuencia inicial coincida exactamente en los números y posiciones mientras que las sucesiones finales de longitud distintas, sean ceros Como el ejemplo 9, la mínima representación del

[11] Kurt Godel (1906-1978), matemático austro-hungaro Mejor conocido por su demostración del teorema de la incompletitud, en su publicación *"Sobre las proposiciones formales indecidible de los Principios Matemáticos y Sistemas Conexos"* Con este resultado terminó bruscamente cientos de años de investigaciones que pretendían establecer una axiomatización completa de toda la Matemática Lo que es considerado, por muchos, como el acontecimiento más importante de las Matemáticas del siglo XX

número de Godel 2250 con longitud 3 y las sucesiones siguientes de mayor longitud terminan en ceros

Para un valor fijo de longitud n la representación $\langle x_0, x_1, \quad, x_n\rangle$, es obviamente una función primitiva-recursiva de $(x_0, x_1, \quad, x_n)$ En el caso de que la longitud n no sea fija, consideraremos una función primitiva-recursiva de una variable $f(\iota)$ que es el valor de x_ι para $\iota = 0,1, \quad$, y la computación del número de Gödel está dado por

$$\langle x_0, x_1, \quad, x_n\rangle = \prod_{\iota=0}^{n} p_\iota^{f(\iota)}(\iota)$$

que resulta ser una función primitiva-recursiva de n

Revisemos lo inverso, conocido como problema de descodificacion de un número de Gödel, que es obtener los valores de $x_0, x_1, \quad, x_n$ de un número natural

$$z = \langle x_0, x_1, \quad, x_n\rangle$$

Definición 1.6.2 Para un número natural $z > 0$, si z es un número de Godel de la sucesión $x_0, x_1, \quad, x_n$, llamaremos función descodificadora de Godel (o función extracción) a la funcion de dos variables E, definida por

$$E(\iota, z) = \begin{cases} x_\iota, & si\ 0 \le \iota \le n \\ 0, & si\ \iota > n \end{cases}$$

La función $E(\iota, z)$ representa el exponente del $\iota - ésimo$ primo de la descomposición y n es la longitud, que puede ser fija o no

Ejemplo 10 Consideremos el número de Godel z = 2250, descrito en el ejemplo 9 Utilizaremos la definición 1 6 2, obteniendo los resultados

$$E(0,2250) = 1,\ E(1,2250) = 2,\ E(2,2250) = 3, E(3,2250) = E(3,2250) = \quad = 0$$

Sabemos que $2250 = 2^1 3^2 5^3$ y cualquier potencia de primos consecutivos que sobrepasen la longitud tendrán como exponente cero

Teorema 1.6.1 La función E es primitiva-recursiva

Demostración Para un número entero $z > 0$, vemos que $E(\iota, z)$ es el numero natural x más grande tal que p_ι^x divide a z De la misma forma podemos indicar que $E(\iota, z)$ es el número natural x más pequeño tal que p_ι^{x+1}no divida a z Así, la última sentencia se cumple para todo natural z, y que x nunca excederá a z tal que

$$E(\iota, z) = min_{x \leq z}[div(p_n^{x+1}(\iota), z) = 0]$$

La función div es la que realiza la operación "divide a", la cual es primitiva-recursiva, puede verificarse en (Hennie, Introduction to Computability, 1977), y p_n el n −esimo primo que también es primitivo-recursivo, por lo que E lo es

1.7 Funciones Recursivas

Las funciones recursivas empleadas por Gödel como un concepto técnico auxiliar que le permitia enunciar en forma adecuada algunos resultados previos a sus teoremas de incompletitud[12] Este concepto se extendió, tomando interés por sí mismo ya que las funciones recursiva resultan ser exactamente las funciones calculables mediante un algoritmo (funciones computables)

Definición 1.7.1 Un función f es recursiva si existe una sucesión de funciones $f_1, f_2, \ldots, f_n$ tales que f_n es f y para todo $1 \leq \iota \leq n$, la función f_ι es primitiva-recursiva o bien esta definida por la composicion o recursion (minimizacion), definido en 1 3 1

Es claro observar, por la definición anterior, que toda función primitiva-recursiva es recursiva La diferencia entre ambos conceptos consiste en que las funciones

[12] Publicado en 1931 por K Godel que indica que "Todo sistema de primer orden consistente que contenga los teoremas de la aritmética y cuyo conjunto (numeros de Godel) de axiomas sea recursivo, no es completo"

recursivas[13] admiten la minimización, llamadas también funciones recursivas parciales, como técnica de definición lo que no es considerado por las funciones primitiva-recursivas

Revisamos, en el punto 1 4, distintas formas de caracterizar funciones primitiva-recursivas por medio de la recursividad primitiva, pero desafortunadamente no todas las funciones pueden determinarse por los esquemas expuestos, es posible construir funciones recursivas que no sean primitiva-recursivas Una de ellas es la función exponencial generalizada de Ackermann[14], que es una función f de tres variables cuya forma recursiva está dada por

$$\begin{aligned} f(0,0,y) &= y, \\ f(0,x+1,y) &= f(0,x,y)+1, \\ f(1,0,y) &= 0, \\ f(z+2,0,y) &= 1, \\ f(z+1,x+1,y) &= f(z,f(z+1,x,y),y), \end{aligned}$$

resultando ser Turing computable, pero no primitiva-recursiva

La función de Ackermann provee un ejemplo de una función total efectivamente computable que puede ser definida recursivamente, por medio de funciones primitiva-recursivas pero ella en sí misma no es primitiva-recursiva Observemos que la especificación recursiva mostrada difiere de las estudiadas anteriormente, ya que define nuevos valores de la función en términos de valores de los argumentos de la función dada, resultando ser propios de la misma función Para esto analizaremos las clases de funciones primitiva-recursivas que pueda ser ampliada de formas distintas, en una clase de funciones que llamaremos *funciones* μ *− recursivas*

Definición 1.7.2 Sea g una función total de $(n+1)$ variables La funcion f se dice que es obtenida de una función g por *minimización* (no acotada) si

[13] K Godel las llamó funciones recursivas a lo que hemos definido como funciones primitiva- recursivas El término fue introducido por Herband para realizar procedimientos de construcción más generales
[14] Nombrada por W Ackermann

$$f(x_1, \quad, x_n) = \mu z[g(x_1, \quad, x_n, z) = 0]$$

El nuevo valor de la función f depende de la existencia o no de un numero z de manera que se cumpla o no que $g(x_1, \quad, x_n, z) = 0$, es decir, si existe un número z tal que $g(x_1, \quad, x_n, z) = 0$, entonces el valor de $f(x_1, \quad, x_n)$ es el valor más pequeño de z En caso de que no existe tal valor de z, $f(x_1, \quad, x_n)$ es no definida De la misma manera, decimos que un predicado P satisface la condición de minimización no acotada si para toda n −tupla $x_1, \quad, x_n$ existe un z tal que $P(x_1, \quad, x_n, z)$ sea verdadero, es decir,

$$f(x_1, \quad, x_n) = \mu z[P(x_1, \quad, x_n, z)]$$

Por la definición 1 7 2, al usar minimización no acotada se puede producir funciones no totales, en contraste a lo requerido de que sea total para su aplicación Por lo que analizaremos la recursión de una función obtenida de funciones parciales por recursión primitiva, como composición de funciones, aplicado a funciones no totales así como a funciones totales De esta forma es posible definir la función μ − recursiva.

Definición 1.7.3 La familia de funciones (predicados) μ − recursiva se define como sigue

i) Las funciones básicas son μ − recursivas

ii) Si h es una funcion de n variables μ − recursiva y $g_1, g_2, \quad, g_n$ son funciones de m variables μ − recursivas, entonces

$$f = h \circ (g_1, g_2, \quad, g_n)$$

es μ − recursiva

iii) Si g y h son funciones μ − recursivas de $(n + 2)$ variables, entonces la función f definida de g y h por recursión primitivo es μ − recursiva

iv) Si $P(x_1, \quad, x_n, y)$ es un predicado total μ − recursivo, entonces

$$f(x_1, \quad, x_n) = \mu z[P(x_1, \quad, x_n, z)]$$

es μ – recursivo

v) Una función es μ – recursiva si sólo si puede ser obtenida de la condición (i) por un número finito de aplicaciones de las reglas (ii), (iii) y (iv)

La condiciones (i), (ii) y (iii) implican que todas las funciones primitiva-recursivas son μ – recursivas Notemos que la minimización no acotada no está definida para todos los predicados, solamente para aquellos predicados μ – recursivos totales

Es claro que todas las funciones primitiva-recursivas son funciones μ – recursivas, además por su definición son funciones efectivamente computables, originando una clase de funciones que llamaremos clase de funciones μ – recursiva o clase de funciones recursivas parciales, como vemos en la figura 4

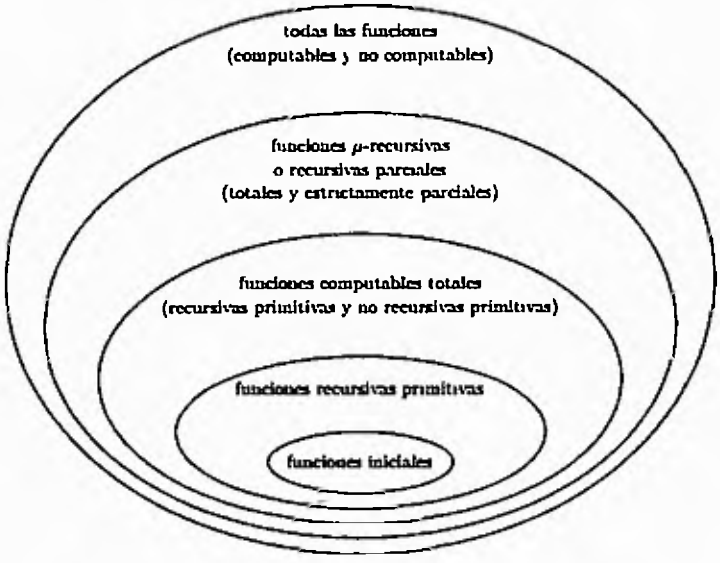

Figura 4 Jerarquía de las clases de funciones,
segun la Teoria de funciones recursivas

Ahora estableceremos la relación entre las funciones μ – recursivas y las funciones Turing computables El primer paso lo establece el siguiente teorema (una extensión del teorema 1 5 2)

Teorema 1.7.1 Toda función μ – recursiva es Turing computable

Demostración Es conocido que las funciones básicas son Turing computables, la prueba consiste en demostrar que las funciones parciales son cerradas bajo la composición, recursión primitiva y la minimización no acotada. Los teoremas 1 5 1 y 1 5 2, dan evidencia de la cerradura con la diferencia de que g y h deben ser funciones no totales Estas maquinas, además establecen la cerradura para funciones parciales ya que un valor no definido en uno de los cálculos constituye una causa para que la computación continua indefinidamente (no se detenga)

Para completar la prueba, se muestra que la minimización no acotada de un predicado total Turing computable es Turing computable Sea

$$f(x_1, \quad, x_n) = \mu z[P(x_1, \quad, x_n, z)]$$

donde $P(x_1, \quad, x_n, y)$ es un predicado total Turing computable Consideremos una maquina P que computa el predicado P con la configuracion inicial de la cinta,

$$0\bar{x}_1 0\bar{x}_2 0 \quad 0\bar{x}_n 0$$

A partir de la máquina P se construira una que evalue a f

1 La representación del numero cero es adicionado al extremo derecho de la entrada La busqueda especificada por el operador minimizacion inicia con la configuración de la cinta

$$0\bar{x}_1 0\bar{x}_2 0 \quad 0\bar{x}_n 0\bar{0}0$$

El numero a la derecha de la entrada se le llamará índice para el operador minimización j

2 Realizando una copia del patron inicial, con el indice j tenemos

$$0\bar{x}_1 0\bar{x}_2 0 \quad 0\bar{x}_n 0\bar{j}0\bar{x}_1 0\bar{x}_2 0 \quad 0\bar{x}_n 0\bar{j}0$$

3 La máquina P se ejecuta con una copia del patrón inicial y el índice j, obteniendo

$$0\bar{x}_1 0\bar{x}_2 0 \quad 0\bar{x}_n 0\bar{j}0\overline{P(x_1, \quad, x_n, j)}0$$

4 Si $P(x_1, \quad, x_n, j) = 1$, el valor de minimización de P es j De otra forma, $P(x_1, \quad, x_n, y)$ es eliminado y j se incrementa, continuando con la computación en el paso $\bar{2}$

Una computación termina para el primer valor del índice j que $P(x_1, \quad, x_n, 1)$ sea verdadero De lo contrario tal valor no existe produciendo que el proceso se mantenga en un ciclo indefinidamente, lo que implica que f sería una función no definida □

El teorema anterior establece que toda función $\mu -$ recursiva es Turing computable, pero nos preguntamos lo opuesto ¿toda función Turing computable es una función $\mu -$ recursiva? Para mostrar esto, se designa una función aritmetica para simular el comportamiento de una máquina de Turing La construccion de la función que simula, requiere mover el dominio de las máquinas al dominio de los números enteros El proceso de trasladar máquinas computacionales a funciones se conoce como aritmetización de las maquinas de Turing, tema que no desarrollaremos en este trabajo, pero el lector lo puede revisar en (Sudkamp, 2006) Con la aritmetización de las máquinas de Turing se concluye con la afirmación de que toda función Turing computable es $\mu -$ recursiva

Como hemos visto las funciones Turing computables, $\mu -$ recursiva y la recursiva general representan tres posibles extensiones de las clases de funciones primitiva-recursivas El hecho es que estas tres extensiones resultan ser idénticas y proporcionan las bases para la definición formal de computabilidad y los temas que revisaremos en los capitulos siguientes

II. Fundamentos de la teoría de la recursión.

En el capítulo anterior, se analizó que las funciones aritmética evaluadas por una máquina de Turing satisfacen de forma intuitiva la noción de efectivamente computable, ver definición 1 2 4 , y que la clase de funciones Turing computables son las clases más pequeña que incluyen, sino a todas, las funciones efectivamente computables De manera similar se analizó el concepto de función μ −recursiva, fundamentada sobre un conjunto de funciones básicas como la función constante, la funcion proyeccion y la función sucesor, las cuales satisfacen la noción de efectivamente computable Cada una de las operaciones de composición funcional, recursión primitiva y minimización pueden ser implementadas por un procedimiento efectivo en la que todas las funciones μ −recursivas, son efectivamente computables Se concluyó que la clase de funciones μ −recursiva la clase de funciones Turing computables son identicas

En general los miembros de estos conjuntos se le llaman funciones recursivas, sinónimo de funciones computables Las funciones recursivas incluyen todas las funciones primitiva-recursivas, Turing computables y las funciones recursivas generales De manera similar los predicados recursivos, como se plasmó, son aquellos cuyas funciones características son funciones recursivas

Las formas comunes de modelar las funciones efectivamente computables, estudiadas en la sección 1, son la μ −recursividad y la Turing computabilidad En esta seccion analizaremos otras posibilidades de definir y evaluar funciones, los problemas de decisión, se introducirá algunos tipos de conjuntos recursivos, recursivamente enumerables y se analizará sus propiedades y las relaciones existentes entre ellos

2.1 Computabilidad Efectiva

Un hecho sorprendente es que los distintos intentos de formalizar el concepto de algoritmo o de función computable, revisados anteriormente, dan lugar a conjuntos extensionalmente coincidentes Esto hace que tales nociones sean equivalentes, por lo

CAPITULO II

FUNDAMENTO DE LA TEORÍA DE RECURSIÓN

II. Fundamentos de la teoría de la recursión.

En el capítulo anterior, se analizó que las funciones aritmética evaluadas por una máquina de Turing satisfacen de forma intuitiva la noción de efectivamente computable, ver definición 1 2 4 , y que la clase de funciones Turing computables son las clases más pequeña que incluyen, sino a todas, las funciones efectivamente computables De manera similar se analizó el concepto de función μ −recursiva, fundamentada sobre un conjunto de funciones básicas como la función constante, la funcion proyección y la función sucesor, las cuales satisfacen la noción de efectivamente computable Cada una de las operaciones de composición funcional, recursión primitiva y minimización pueden ser implementadas por un procedimiento efectivo en la que todas las funciones μ −recursivas, son efectivamente computables Se concluyó que la clase de funciones μ −recursiva la clase de funciones Turing computables son idénticas

En general los miembros de estos conjuntos se le llaman funciones recursivas, sinónimo de funciones computables Las funciones recursivas incluyen todas las funciones primitiva-recursivas, Turing computables y las funciones recursivas generales De manera similar los predicados recursivos, como se plasmó, son aquellos cuyas funciones caracteristicas son funciones recursivas

Las formas comunes de modelar las funciones efectivamente computables, estudiadas en la sección 1, son la μ −recursividad y la Turing computabilidad En esta sección analizaremos otras posibilidades de definir y evaluar funciones, los problemas de decisión, se introducirá algunos tipos de conjuntos recursivos, recursivamente enumerables y se analizara sus propiedades y las relaciones existentes entre ellos

2.1 Computabilidad Efectiva

Un hecho sorprendente es que los distintos intentos de formalizar el concepto de algoritmo o de funcion computable, revisados anteriormente, dan lugar a conjuntos extensionalmente coincidentes Esto hace que tales nociones sean equivalentes, por lo

que parece que cualquiera de ellas es una adecuada formalización del concepto intuitivo de "función computable" Esta afirmación se le conoce como la *Tesis Church-Turing*[15]

Tesis de Church-Turing Toda función aritmética efectivamente computable es una función recursiva (computable) y viceversa

Como podemos observar lo planteado equipara la noción informal de efectivamente computable con la noción formal de recursividad

La tesis de Church-Turing no es considerada un teorema y a pesar de que no está sujeta a prueba formal, existen razones para aceptarla tales como

1 Basada en la experiencia acumulada todos los algoritmos conocidos son recursivos, mencionado por (GANDY, 1988)

2 Basada en la equivalencia entre los tres conceptos introducidos para establecer la noción de algoritmos

3 El análisis que Turing realizó del concepto de computabilidad, enfocándose en el propio proceso de computación en lugar de la naturaleza de las funciones computables y así construyendo las funciones a partir de funciones elementales de la cual no cabe duda su carácter algoritmico

4 El argumento presentado por Church, similar al presentado por Turing

5 El fracaso del argumento de la diagonal[16]

Es importante destacar que la tesis de Church-Turing, se refiere solamente a funciones aritmeticas Para superar esta limitante es posible utilizar un esquema de codificación de numeros naturales, de manera que las cadenas de símbolos puedan ser representados por medios de funciones aritméticas

[15] No es un enunciado matemático susceptible de demostración, ya que involucra la noción intuitiva de *algoritmo*

[16] Proceso introducido por Cantor para demostrar que $\wp(w)$ no es biyectiva con w, lo que permite señalar conjuntos que no han sido incluidos en una enumeración, pretendidamente completa de un determinado dominio

Llamaremos F a la función obtenida al implementar el esquema de codificación de una función no aritmética, de forma que resultado de F pueda ser reformulado como una función aritmética f Esto nos indica que existe un procedimiento efectivo para la conversión de ida y vuelta entre los argumentos originales y los valores de F y su representación numérica asociadas De acuerdo a la tesis de Church-Turing, esto significa que F será efectivamente computable si y sólo si f es recursiva

Al generalizar la tesis de Church-Turing, se hace una distinción entre los términos de funciones aritméticas y funciones no aritméticas, ya que es común referir a una todas las funciones efectivamente computables como funciones recursivas o computables De esta forma se reserva la palabra recursiva para funciones aritméticas y se utiliza computable como término más general para alguna función efectivamente computable, sea o no aritmética Lo que significa que para la función no aritmética sea considerada computable debe ser posible convertirla, por medio de una máquina de Turing apropiada, a una función aritmética

Aclarado los conceptos, ahora caracterizaremos las funciones recursivas como una familia Sea $n \in \mathbb{N}$, la ι −ésima función recursiva parcial de n variables que es evaluada por la ι −esima máquina de Turing se denota $\varphi_\iota^{(n)}$

Definición 2.1.1 Una función algorítmica parcial la cual es definida sobre todo los argumentos (es decir total) es llamada recursiva o computable

Escribiremos $\varphi_{\iota,s}(x) = y$ si $x, y, \iota < s$, además y es la resultado de $\varphi_\iota(x)$ en menos de s pasos en la ι −ésima Máquina de Turing Podemos decir que si tal y existe, la función $\varphi_{\iota,s}(x)$ converge, es decir $\varphi_{\iota,s}\downarrow$, de lo contrario diverge Similarmente escribiremos $\varphi_\iota(x)\downarrow$ si $\varphi_{\iota,s}(x)\downarrow$ para algún s y $\varphi_{\iota,s}(x)\downarrow = y$ si $\varphi_\iota(x)\downarrow$ y $\varphi_{\iota,s}(x) = y$

Teorema 2.1.1

(1) El conjunto $\{(\iota, x, s)\ \varphi_{\iota,s}(x)\downarrow\}$ es recursivo

(ii) El conjunto $\{(\iota, x, s)\ \varphi_{\iota,s}(x) = y\}$ es recursivo

Demostración Se aplica la tesis de Church, hasta obtener un resultado o hasta completar los s primero pasos

Definición 2.1.2 Sea x, y, ι y $s \neq 0$ números enteros positivos, La función recursiva $\varphi_{\iota,s}(x) = y$ se puede definir como

1) $\varphi_{\iota,s}(x) = y \Longrightarrow \iota, x, y < s$ ó

2) $(\forall s)(\exists\ al\ menos\ un\ (\iota, x, y))[\varphi_{\iota,s}(x) = y \wedge \varphi_{\iota,s-1}(x)\ \uparrow]$

Definición 2.1.3 Sea $\varphi_\iota^{(n)}$ la ι −ésima función recursiva parcial de n variables La clase de las funciones recursivas se denota por

$$\Re = \left\{\varphi_\iota^{(n)}, \iota \in \mathbb{N}, n \in \mathbb{N}\backslash\{0\}\right\}$$

Ejemplo 11 Lo siguiente son ejemplos de funciones recursivas

1 Todas las funciones primitiva-recursivas son recursivas

2 La función de Ackermann es recursiva pero no es primitiva-recursiva, visto en el capítulo 1

La clase de funciones recursivas (parciales o totales) es la menor clase de funciones que contiene la función constante, la funcion sucesor y la función proyeccion y es cerrado por sustitución, recursion primitivo y minimización implicando esto, que satisface las cinco propiedades básicas estudiadas en el capítulo anterior

2.2 Predicados semi-computables y sus propiedades

Señalamos que una computabilidad efectiva nos indica que existe un procedimiento efectivo para llegar a la solución de un problema en un numero finito de pasos Pero en algunos casos, es posible lograr ese procedimiento sólo de forma parcial,

para ciertos datos, y en otros casos no existe procedimiento alguno para que el algoritmo se detenga Informalmente podemos indicar que aquellos problemas que pueden expresarse mediante predicados se denominan *problemas de decisión* Aunque no todos los problemas interesantes son de este tipo, lo que sí podemos decir es que todos los problemas relevantes para la Teoría de la Computabilidad tienen asociado un problema de decisión

Definición 2.2.1 Un predicado P n-ario se le llama semi-computable (o semi-decidible) si existe una función recursiva $f\ \mathbb{N}^n \to \mathbb{N}$ de n variables, cuyo dominio coincide con la extensión[17] del predicado P, esto es

$$dom\ f = \{\vec{x} \in \mathbb{N}\ P(\vec{x})\}$$

Si f es una función recursiva parcial de n variables, el predicado n-ario P es semi-computable si

$$P(\vec{x}) \Leftrightarrow f(\vec{x}) \downarrow$$

Al predicado P se le conoce también como recursivamente enumerable si su definición esta asociada a una función parcial recursiva y como semi-computable si su definición se fundamenta en una función computable

Teorema 2.2.1 Todo predicado recursivo es semi-computable

Demostración Sea $P \subset \mathbb{N}^k$ un predicado k −ario recursivo, entonces

$$A = \mathbb{N}^k \setminus P = \overline{P}$$

es recursivo Entonces, existe una función característica χ_A que es recursiva Se define una funcion $f\ \mathbb{N}^k \longrightarrow \mathbb{N}$, tal que

[17] La extensión del predicado se conoce como el conjunto $(x_1, \ , x_n)$ para el cual $P(x_1, \ , x_n)$ es verdadero

$$f(\vec{x}) = \mu y[\chi_A(\vec{x}) = 0]$$

Observemos que la variable y no tiene incidencia en la expresion principal de μ −minimización Entonces se puede presentar lo siguiente

1 Si $P(\vec{x})$ se cumple, entonces $\chi_A(\vec{x}) = 0$ específicamente $f(\vec{x})\downarrow$

2 Si $A(\vec{x})$ se cumple, entonces $\chi_A(\vec{x}) = 1$ por lo que no existe un y tal que

$$\chi_A(\vec{x}) = 0$$

Así $f(\vec{x})\uparrow$ y se obtiene

$$P(\vec{x}) \Leftrightarrow f(\vec{x})\downarrow \Leftrightarrow \vec{x} \in dom(f)$$

por lo que $P = dom\, f$ es semi-computable □

El reciproco del teorema anterior es falso Antes de probar este hecho veremos que si anteponemos un cuantificador existencial a los predicados semi-computables, los mismos se obtienen a partir de predicados recursivos

Un resultado importante de la teoría de funciones recursivas es la *Forma normal de Kleene*[18] que expone lo siguiente

Teorema 2.2.2 Para cada entero positivo n, existe un predicado primitivo recursivo $(n+2)$ −ario T_n tal que, si f es una función recursiva de n variables, entonces existe un número natural w_f tal que

$$f(\vec{x}) = \rho\big(\mu y[T_n(w_f, \vec{x}, y)]\big)$$

donde ρ[19] es una función primitiva-recursiva fija

[18] La forma normal de Kleene se refiere a que una cadena de caracteres que puede ser expresada con operaciones recursivas

El teorema de la forma normal de Kleene, establece que para toda función recursiva general (parcial) es $\mu-$ recursiva y viceversa, además se cumple si reemplazamos una función total por una función recursiva parcial

Teorema 2.2.3 Sea $R \subset \mathbb{N}^n$ un predicado semi-computable si y solamente si existe un predicado recursivo $P \subset \mathbb{N}^{n+1}$, tal que

$$R(\vec{x}) \Leftrightarrow \bigvee_y P(\vec{x}, y)$$

Demostración Como R es un predicado semi-computable, existe una función recursiva f tal que

$$dom\, f = \{\vec{x} \in \mathbb{N}\ \ R(\vec{x})\}$$

Por el teorema 2 2 2, existe un numero natural w_f de manera que

$$f(\vec{x}) = \rho\big(\mu y[T_n(w_f, \vec{x}, y)]\big)$$

Se tiene que

$$dom\, f = \left\{\vec{x}\ \bigvee_y T_n(w_f, \vec{x}, y)\right\}$$

Por lo que $R(\vec{x}) \Leftrightarrow \mathrm{V}_y\, P(\vec{x}, y)$

Por otro lado si P es un predicado recursivo $(n+1)$-ario, existe una función caracteristica χ de P Consideremos el conjunto $\{\vec{x}\ \ R(\vec{x})\}$como el dominio de la función f de n variables definida de la manera siguiente

$$f(\vec{x}) = min_y[1-\chi(\vec{x}, y) = 0])$$

[19] A la función ρ se le conoce como función codificadora de Kleene, que al igual que la forma exacta de los predicados T_n de Kleene, depende de la aritmetización empleada por las funciones recursivas, lo que implica que no es unica

Por lo tanto R es un predicado semi-computable

El teorema 2 2 3, nos indica que se puede asignar a cada predicado semi-computable (recursivamente enumerable) R, un número natural llamado w dependiendo de la aritmetización definido para funciones recursivas Este resultado se ampliara con el siguiente teorema, llamado teorema de enumeración de Kleene

Teorema 2.2.4 Sea $R \subset \mathbb{N}^n$ un predicado semi-computable Entonces existe un número natural w_R tal que

$$R(\vec{x}) \Leftrightarrow \bigvee_y T_n(w_R, \vec{x}, y)$$

La relación existente entre los conceptos semi-computable y recursividad se establece con el siguiente resultado

Teorema 2.2.5 Un predicado $R \subset \mathbb{N}^n$ es recursivo si y solo si tanto R y $\neg R$ son semi-computables

Demostración Si R es un predicado recursivo, tambien los es $\neg R$, entonces ambos son semi-computable, por el teorema 2 2 1

Por otro lado, supongamos que R y $\neg R$ son semi-computables entonces por el teorema 2 2 3, existen predicados recursivos $P, Q \subset \mathbb{N}^{n+1}$, tales que

$$R(\vec{x}) \Leftrightarrow \bigvee_y P(\vec{x}, y), \qquad \neg R(\vec{x}) \Leftrightarrow \bigvee_y Q(\vec{x}, y)$$

Para algun valor de $\vec{x}$, seleccionado, existe un valor de y para la cual algunos de los predicados $P(\vec{x}, y)$ o $Q(\vec{x}, y)$ sea verdadero Por lo que se define la funcion

$$h(\vec{x}) = \mu y[P(\vec{x}, y) \vee Q(\vec{x}, y)]$$

es recursiva y total Entonces tendremos que $R(\vec{x}) \Leftrightarrow P(\vec{x}, h(\vec{x}))$ por lo tanto R es recursivo

El resultado anterior determina que un predicado es recursivo si y sólo si, el propio predicado y su negación sean semi-computables Retomando el teorema 2 2 1, veremos que el recíproco del mismo no es cierto mostrado en el siguiente teorema

Teorema 2.2.7 El predicado unario $\forall_y T(x, x, y)$ es semi-computable, pero no recursivo

Demostración Tomemos el predicado $\neg \forall_y T(x, x, y)$ semi-computable Entonces existe un número natural w (por el teo 2 2 3) tal que

$$\neg \bigvee_y T(x, x, y) \Leftrightarrow \bigvee_y T(w, x, y)$$

si se cumple para cualquier número natural es posible que $x = w$, resultando una contradicción Por lo que el predicado unario T es no recursivo

2.3 Problema de decisión

Con los resultados obtenidos, es posible estudiar con mayor detalle los problemas de decisión Prestaremos especial atención a la existencia de un procedimiento efectivo (algoritmo), que nos permita decidir la verdad o falsedad de toda una clase completa de afirmaciones Se le conoce como *solución positiva* a la demostración de la existencia de un algoritmo para resolver un problema dado En el caso de que se demuestre, sin duda, que no existe un algoritmo para resolver el problema en cuestión, se le llama *solución negativa* y se dice que problema de decisión es *indecidible* (irresoluble algorítmicamente)

Definición 2.3.1 Sea $R(\vec{x})$ un predicado $n-$ario El problema de decisión para el predicado R se dice recursivamente resoluble (o efectivamente decidible) si el predicado R es recursivo De lo contrario diremos que es recursivamente irresoluble

En la teoría de recursión, uno de los problemas de decisión relacionados es la clase de las funciones recursivas que son recursivamente irresolubles, definición 2 1 3, como se muestra en el siguiente ejemplo

Ejemplo 12 Los siguientes problemas de decisión, asociados a las clases de funciones recursivas, son recursivamente irresolubles

(a) El problema de decidir, para algún x e y, si $\varphi_x(y)$ es definida

(b) El problema de decidir, para algún x , si $\varphi_x^{(1)}$ es total

(c) El problema de decidir, para algún x e y, si $\varphi_x = \varphi_y$

En este contexto el teorema 2 2 7 , se puede reescribir como el siguiente corolario

Corolario 2.3.1 El problema de decisión para el predicado de Kleene $V_y T(x,x,y)$ es recursivamente irresoluble

El siguiente teorema, debido a Rice, establece automáticamente la irresolubilidad de una amplia clase de problemas de decisión

Teorema 2.3.1 (Teorema de Rice) Sea P un subconjunto propio no vacío del conjunto de funciones recursivas de una variable Entonces el conjunto $A = \left\{x \ \varphi_x^{(1)} \in P\right\}$ no es recursivo

Demostración Primero consideremos el caso en el cual la función vacia $\Phi^{(1)} \notin P$ Como P es no vacio, debe existir al menos una función recursiva de una variable f que pertenece a P En el camino usual, podemos establecer la existencia de una funcion recursiva total r tal que

$$\varphi_{r(x)}^{(1)}(y) = \begin{cases} f(y) & si \ \varphi_x^{(1)} \downarrow \\ \uparrow & si \ \varphi_x^{(1)} \uparrow \end{cases}$$

Si $\varphi_x^{(1)}$ es definida, $\varphi_{r(x)}^{(1)}$ es la función f y en caso contrario es la función $\Phi^{(1)}$ Sea χ_A la función característica del conjunto $A = \left\{x\ \varphi_x^{(1)} \in P\right\}$ Si χ_A fuera recursiva, la composición funcional $\chi_A \circ (r)$ también sería recursiva, resultando la función de dominio diagonal (sobre la cual se sabe que no es recursiva), ya que

$$\chi_A\big(r(x)\big) = \begin{cases} 1 & si\ \ \varphi_{r(x)}^{(1)} \in P \\ 0 & si\ \ \varphi_{r(x)}^{(1)} \notin P \end{cases} = \begin{cases} 1 & si\ \varphi_x^{(1)} \downarrow \\ 0 & si\ \ \varphi_x^{(1)} \uparrow \end{cases}$$

Entonces χ_A no es recursiva por lo que el conjunto A no es recursivo
Finalmente en el caso de que $\Phi^{(1)} \in P$, simplemente reemplazamos P por su complemento $\overline{P}$, en el argumento anterior lo que muestra que $\overline{A}$ no es recursivo Implicando que A no es recursivo □

Invocando la definición de resolubilidad recursiva, el teorema de Rice se puede reescribir como el corolario a continuación

Corolario 2.3.2 Sea P un predicado cuyo dominio son las funciones recursivas de una variable Supongamos que P es verdadero para al menos una función recursiva y P es falso para al menos otra función recursiva Entonces, el problema de determinar, para cualquier índice x dado, si el predicado P es verdadero para la función $\varphi_x^{(1)}$, es recursivamente irresoluble

Informalmente, el corolario 2 3 2 , nos indica que no existe un algoritmo para determinar, sobre la base de un unico índice, si una función satisface una propiedad no trivial dada Como se había mencionado el teorema de Rice establece automáticamente la irresolubilidad recursiva de una gran clase de problemas de decisión, algunos de estos podemos ver en el ejemplo siguiente

Ejemplo 13 Algunos problemas de decisión

I El problema de determinar, para cada x, si $\varphi_x^{(1)}$ es total

2 El problema de determinar, para cada x, si $\varphi_x^{(1)}$ es una función constante

3 El problema de determinar, para cada x, si el rango de $\varphi_x^{(1)}$ es infinito

4 El problema de determinar, para cada x, si $\varphi_x^{(1)}$ incluye a 17 dentro de su rango

El teorema de Rice puede extenderse de manera natural a predicados que no sean unarios Según (Hennie, Introduction to Computability, 1977), no se puede pensar que todos los problemas de decision recursivamente irresolubles relacionados con las funciones recursivas caen víctimas del teorema de Rice o sus extensiones En particular, es importante entender que el predicado referido en el teorema de Rice debe ser un predicado de funciones solamente, y no un predicado sobre los índices, o las funciones y sus indices al mismo tiempo

Ejemplo 14 El teorema de Rice no es aplicable al problema de determinar que para una función específica total y recursiva f y una x seleccionada arbitrariamente se tiene

$$\varphi_x^{(1)} = \varphi_{f(x)}^{(1)}$$

Se ha demostrado, hasta el momento, que la clase de predicados semi-computables no es cerrada bajo la operación de la negación Presentaremos algunas operaciones en que la clase, mencionada, es cerrada

Teorema 2.3.2 Sea $P(\vec{x}, y)$ un predicado $(n+1)-$ ario semi-computable y sea

$$Q(\vec{x}) \Leftrightarrow \bigvee_y P(\vec{x}, y)$$

Entonces, Q es semi-computable

Demostración Por la condición necesaria del teorema 2 2 3, existe un predicado $R \subset \mathbb{N}^{n+2}$, tal que $P(\vec{x}, y) \Leftrightarrow \mathrm{V}_z R(\vec{x}, y, z)$ Entonces

$$Q(\vec{x}) \Leftrightarrow \bigvee_y \bigvee_z R(\vec{x}, y, z) \Leftrightarrow \bigvee_t R(\vec{x}, \sigma_1(t), \sigma_2(t))$$

donde σ_1 y σ_2 son las funciones descodificadoras de Cantor (descrita en el capitulo 1) De manera que, aplicando nuevamente el teorema 2 2 3, Q es semi-computable

Una generalización del teorema anterior, fue presentada por Andrezj Mostowski[20] cuya demostración se puede ver en (Piza Volio, 2001)

Teorema 2.3.3 Sea $P(\vec{x}, y)$ un predicado $(n+1)-$ ario semi-computable y sea

$$Q(\vec{x}, z) \Leftrightarrow \bigwedge_{y=0}^{z} P(\vec{x}, y)$$

Entonces, Q es semi-computable

Teorema 2.3.4 Sea $P(\vec{x}, y)$ un predicado $(n+1)-$ ario semi-computable y sea f una función recursiva de n variables Entonces, el predicado $(n+1)-$ario $P\big(\vec{x}, f(\vec{x})\big)$ es semi-computable

Demostración Como P es un predicado semi-computable, por el teorema 2 2 3, existe el predicado recursivo $R(\vec{x}, y, z)$ tal que

$$P(\vec{x}, y) \Leftrightarrow \bigvee_{z} R(\vec{x}, y, z)$$
$$P\big(\vec{x}, f(\vec{x})\big) \Leftrightarrow \bigvee_{z} R(\vec{x}, f(\vec{x}), z)$$

Queda demostrado

2.4 Conjuntos recursivos y recursivamente enumerables

La recursividad y la enumerabilidad recursiva juegan un papel importante en el estudio de las funciones computables y los problemas de decisión En los puntos

[20] Andrezj, Mostowski (1913-1975) Matemático polaco, conocido por el recordado lema del colapso Sus trabajos de investigación estuvieron fundamentalmente relacionados con la teoría de recursión e indecibilidad, así como la lógica de primer orden y la teoría de modelos

anteriores se estudió los predicados semi-computables que resulta ser el estudio de los dominios de las funciones recursivas. Al estudiar los rangos de las funciones recursivas, obtenemos fundamentalmente los mismos resultados. Estos dos conceptos no son equivalentes pero están estrechamente relacionados.

Definición 2.4.1: A es un conjunto recursivo si posee una función característica recursiva.

Intuitivamente, A es recursiva si existe un procedimiento efectivo para decidir, dado algún x, si x pertenece o no al conjunto A.

Ejemplo 15: Los siguientes conjuntos son recursivos

i. El conjunto de enteros pares $\{0, 2, 4, \dots\}$

ii. $\mathbb{N}$ y $\emptyset$

iii. Cualquier conjunto finito

iv. Cualquier conjunto con complemento finito

Ejemplo 16: Tenemos el conjunto $E = \{x: \varphi_x(x) \downarrow\}$ y su función característica dada por

$$\chi_E = \begin{cases} 1 & si\ \varphi_x(x) \downarrow \\ 0 & si\ \varphi_x(x) \uparrow \end{cases}$$

Observemos que la función característica es justamente la función de dominio diagonal para las clases de funciones recursivas, la cual no es una función recursiva por ende el conjunto E no es recursivo.

Definición 2.4.2: El conjunto A es recursivamente enumerable (r.e.) si $A = \emptyset$ o existe una función recursiva total de una variable tal que $A = rang\ f$ (rango de la función f).

Intuitivamente, un conjunto es recursivamente enumerable si existe un procedimiento efectivo para enumerar los miembros del conjunto (permitiendo la

repetición). Existen $\aleph_0$ conjuntos recursivamente enumerables. Por cardinalidad, deben existir conjuntos que no son recursivamente enumerable, dado que hay $2^{\aleph_0}$ subconjuntos de $\mathbb{N}$, particularmente entre estos el conjunto $\{x: \varphi_x\ total\}$.

En la literatura, de la teoría de la recursión, el término semi-computable es, algunas veces, utilizada como sinónimo de recursivamente enumerable, hecho aclarado anteriormente. En este sentido es posible replantear los predicados semi-computables en términos de conjuntos recursivamente enumerable con la definición siguiente.

Definición 2.4.3: Para cada $n \in \mathbb{N}$, tenemos

$$W_n = \left\{ x: \bigvee_y T(n,x,y) \right\}$$

donde, el conjunto W_n es el dominio de la n-ésima función de una variable recursiva $\varphi_n^{(1)}$.

De la expresión en la definición 2.4.3 se deduce que

$$W_n = dom\ \varphi_n = \{x: \varphi_n(x)\downarrow\} = \{x: (\exists y)T(n,x,y)\}$$

donde n es el índice recursivamente enumerable o número de Gödel para el conjunto recursivamente enumerable W_n. Además si $W_{n,s} = dom\ \varphi_{n,s}$ podemos decir que si $x \in W_{n,s}$ entonces $x, n < s$.

De la definición 2.1.2, se puede establecer una relación ente el conjunto W_n y la función φ_n como sigue: si $\varphi_n(x) = y$ si y sólo si $(\exists s)[\varphi_{n,s}(x) = y]$ y $x \in W_n$ si y sólo si $(\exists s)[x \in W_{n,s}]$

Miraremos una conexión inmediata entre recursividad y enumerabilidad recursiva en el siguiente teorema.

Teorema 2.4.1: A es recursiva entonces A es recursivamente enumerable

Demostración: Sea A un conjunto recursivo entonces

Caso (i) Si $A = \emptyset$, por definición es recursivamente enumerable

Caso (ii): A es finito y distinto del vacío. Sean $A = \{n_0, n_1, \ldots n_k\}$ y la función definida por

$$f(x) = \begin{cases} n_x, & para\ x \leq k \\ n_k, & para\ k < x \end{cases}$$

f es una función total.

Caso (iii): A es infinito y $\mathcal{X}_A$ su función característica, entonces se define f por recursión primitivo

$$f(0) = \mu y[\mathcal{X}_A(y) = 1],$$
$$f(x+1) = \mu y[\mathcal{X}_A(y) = 1 \wedge f(x) < y].$$

En los casos (i) y (ii) f es recursiva, y $x \in ran\, f$ si y sólo si $x \in A$ lo que se deduce que A es recursivamente enumerable

El inverso de este teorema no es cierto y su demostración no es constructiva. Sin embargo un conjunto y su complemento son recursivamente enumerables, entonces el conjunto debe ser recursivo, argumento formalmente demostrado en el teorema siguiente.

Teorema 2.4.2: Un conjunto A es recursivo si y sólo si A y $\overline{A}$ ambos son recursivamente enumerable

Demostración: ($\Longrightarrow$) Si A es recursivo entonces $\overline{A}$ es recursivo, resultado inmediato del teorema 2.2.5.

($\Longleftarrow$): Si $A = \emptyset$ o $\overline{A} = \emptyset$, A es inmediatamente recursivo. Si $A \neq \emptyset$ y $\overline{A} \neq \emptyset$, entonces, existen dos funciones totales recursivas f y g tal que $A = rang\, f$ y $\overline{A} = rang\, g$. Consideremos la función auxiliar, h, definida por:

$$h(x) = \mu z[(f(z) = x) \vee (g(z) = x)]$$

De esta manera que h es una función recursiva Para un valor de x dado, $x \in ran\, f$ o $x \in ran\, g$ lo que se deduce que h es una función total Como el valor de $h(x)$, depende de cualquiera de los dos procesos que generen a x, podemos definir la función característica de A de la forma

$$\chi_A = \begin{cases} 1, & si\ f(h(x)) = x \\ 0, & si\ f(h(x)) \neq x \end{cases}$$

Entonces χ_A es una función recursiva, como consecuencia A debe ser un conjunto recursivo

Veremos que algunos conjuntos son recursivamente enumerables pero no recursivos

Teorema 2.4.3 El conjunto $K = \{x \vee_y T(x,x,y)\} = \{x\ \varphi_x(x)\downarrow\}$ de Godel es recursivamente enumerable pero no es recursivo

Demostración Es una consecuencia del teorema 2 2 7

Entenderemos, informalmente, que un *problema de decisión* para un conjunto $A \subset \mathbb{N}$, es el problema de determinar, para cada $n \in \mathbb{N}$, si n pertenece o no al conjunto A, es decir, si existe un procedimiento efectivo para responder si ¿ $n \in \mathbb{N}$?

Definición 2.4.4: *El problema de decisión* para el conjunto $A \subset \mathbb{N}$ se dice que es recursivamente resoluble o irresoluble si A es o no un conjunto recursivo

Una consecuencia del teorema 2 4 3 es el corolario siguiente

Corolario 2.4.1 El conjunto K de Godel tiene un problema de decisión recursivamente irresoluble

Definición 2.4.5 Un conjunto A es recursivamente enumerable en *orden creciente* si existe una función total y recursiva f, tal que $A = rang\ f$ y f es una función creciente

Obviamente si un conjunto es recursivamente enumerable en orden creciente es recursivamente enumerable en orden decreciente en el sentido irrestricto Pero lo contrario, sin embargo, no es cierto

Teorema 2.4.4 Sea A un conjunto Entonces, son equivalentes

(a) A es recursivo e infinito

(b) A recursivamente enumerable en orden creciente

Demostración $(a) \Longrightarrow (b)$ Inmediato por las funciones construidas en los casos (i) y (ii) de la prueba del teorema 2 4 1

$(b) \Longrightarrow (a)$ Consideremos los siguientes casos

(i) Si A es finito entonces A es recursivo

(ii) Si A es infinito y recursivamente enumerable en orden creciente, sea f la función recursiva total y creciente que enumera a A en orden creciente, así $A = \{f(n)\ n \in \mathbb{N}\}$ Entonces la función característica de A coincide con la funcion característica del predicado recursivo $V_{n=0}^{x}(f(n) = x)$, por lo que A es recursivo

Vemos que en el caso de conjunto infinitos, la recursividad es equivalente a recursividad enumerable en orden creciente

Teorema 2.4.5 Todo conjunto infinito recursivamente enumerable tiene un subconjuto infinito y recursivo

Demostración Sea A un conjunto infinito y recursivamente enumerable Sea f la función recursiva y total de manera que $A = rang\, f$ Definamos la función recursiva g de la forma siguiente

$$g(0) = f(0),$$
$$g(x+1) = f(\mu y[f(y) > g(x)])$$

Sea $B = rang\, g$ Entonces g es una función recursiva y total que enumera a B en orden creciente, por el teorema 2 4 4, B es un conjunto recursivo e infinito y claramente $B \subset A$

Este concepto, de conjunto recursivamente enumerable, puede extenderse naturalmente a conjunto n −étuplos, definido a continuación

Definición 2.4.6 Sea $A \subset \mathbb{N}^n$ un conjunto de $n-$ étuplos Decimos que A es recursivamente enumerable si su conjunto asociado

$$A^* = \{\pi^{(n)}(\vec{x})\ \ \vec{x} \in A\},$$

es recursivamente enumerable, donde $\pi^{(n)}$ denota la función codificador n −etuplos de Cantor

Aunque es posible su extensión, esto no le agrega ninguna caracteristica especial a la teoría de los conjuntos recursivamente enumerables debido a que tanto las funciones codificadoras y decodificadoras asociadas de Cantor son primitiva-recursivas

El siguiente teorema nos señala las condiciones en que un conjunto de n −étuplos es recursivamente enumerable

Teorema 2.4.6 Sea $A \subset \mathbb{N}^n$ un conjunto de n −etuplos con $A \neq \emptyset$ Entonces, A es un conjunto recursivamente enumerable si y sólo si A es el dominio de alguna funcion recursiva de n variables

Demostración ($\Longrightarrow$) Supongamos que A es recursivamente enumerable, por definición el conjunto asociado $A^* = \{\pi^{(n)}\ \vec{x} \in A\}$ es recursivamente enumerable, entonces existe un índice $\omega \in \mathbb{N}$ tal que $A^* = dom\ \varphi_\omega^{(1)}$

Definamos la funcion composición de n variables $\varphi_\omega^{(1)} \circ \pi^{(n)}$ que es recursiva, por lo que existe un índice e, de esta función tal que

$$\varphi_\omega^{(1)}\left(\pi^{(n)}(\vec{x})\right) = \varphi_e^{(n)}(\vec{x}) \quad \forall \vec{x} \in \mathbb{N}^n$$

Si $\vec{x} \in A \Leftrightarrow \pi^{(n)}(\vec{x}) \in A^* \Leftrightarrow \varphi_\omega^{(1)}\left(\pi^{(n)}(\vec{x})\right)\downarrow \Leftrightarrow \vec{x} \in dom\ \varphi_e^{(n)}$, lo que se deduce que

$$A = dom\ \varphi_e^{(n)}$$

($\Longleftarrow$) Supongamos que $A = dom\ \varphi_e^{(n)}$ de manera que

$$A^* = \left\{\pi^{(n)}(\vec{x})\ \varphi_e^{(n)}(\vec{x})\downarrow\right\} = \left\{t\ \varphi_e^{(n)}\left(\sigma_1^{(n)}(t), \quad , \sigma_n^{(n)}(t)\right)\downarrow\right\}$$

Entonces A^* es el dominio de la funcion recursiva unaria $\varphi_e^{(n)} \circ \left(\sigma_1^{(n)}, \quad , \sigma_n^{(n)}\right)$

2.5 Conjuntos 1-1 y m reducibles

En el punto 2 3, presentamos la definición de problemas resolubles y recursivamente irresolubles Para el proposito de la teoría de recursión tales problemas son representados como conjuntos de enteros De esta forma el problema es resoluble si el respectivo conjunto de enteros es recursivo

En este punto incorporaremos la noción de reducibilidad, utilizado para establecer ciertos resultados indecidibles Informalmente podemos decir que un problema es irreducible a otro si un metodo para resolver el segundo problema (reducido) produce un método para resolver el problema principal En vista al uso de conjuntos para representar los problemas, formularemos y estudiaremos la reducibilidad como una relación entre

conjuntos enteros Así veremos algunas técnicas que permite reducir el grado de complejidad de algunos conjuntos en otros

Iniciaremos definiendo el conjunto básico

Definición 2.5.1 Sea $K_0 = \{\pi(x,y)\ x \bar{\in} W_y\}$

El siguiente teorema, dentro del contexto de conjuntos recursivamente enumerable, es la forma abstracta del problema de parada de una máquina de Turing

Teorema 2.5.1 K_0 es recursivamente enumerable, pero no es recursivo

Demostración Sea $t = \pi(x,y)$, entonces por definición

$$K_0 = \{t\ \sigma_1(t) \in W_{\sigma_2(t)}\}$$
$$= \left\{t\ \bigvee_z T(\sigma_2(t), \sigma_1(t), z)\right\},$$

K_0 es recursivamente enumerable Por otro lado si $x \in K$ entonces $\pi(x,y) \in K_0$, de manera que si K_0 tiene una función característica recursiva entonces K también lo tendría lo que resulta una contradicción, por el teorema 2 4 3

Es frecuente, para mostrar que un problema de decisión es indecidible, relacionarlo a otro problema que se conoce que sea indecidible Este enfoque se refiere en reducir un problema a otro

Definición 2.5.2 Sean $A \subseteq \mathbb{N}$ y $B \subseteq \mathbb{N}$ dos conjuntos, diremos que

a) A es m −reducible a B si existe una función recursiva f tal que para todo $x \in A$ si y solo si $f(x) \in B$ Se denota

$$A \leq_m B$$

b) A es 1 −reducible[21] a B si existe una función 1-1 (función inyectiva) recursiva f tal que para todo $x \in A$ si y sólo si $f(x) \in B$, se denota

$$A \leq_1 B$$

Para que B sea m −reducible a A, es pertinente que la función f no sólo debe relacionar todos los elementos de B con algún elemento de A, además debe relacionar todos los miembros de $\overline{B}$ con algun miembro de $\overline{A}$ por la misma función, es decir

$$f(A) \subseteq B \ y \ f(\overline{A}) \subseteq \overline{B}$$

Ejemplo 17 Sean $B = \{x \ \varphi_x(x) \downarrow\}$ y $A = \left\{x \ \varphi_x^{(1)} \ es \ total\right\}$ Para que B sea m −reducible a A necesitamos, solamente, recordar que existe una función total recursiva f, de la forma

$$\varphi_{f(x)}^{(1)} = \begin{cases} C_0^{(1)} & si \ \varphi_x(x) \downarrow \\ \Phi^{(1)} & si \ \varphi_x(x) \uparrow \end{cases}$$

Como $C_0^{(1)}$ es una función total y $\Phi^{(1)}$ no lo es, $f(x) \in A$ si y solo si $x \in B$

Teorema 2.5.3

a) $\leq_1$ y $\leq_m$ son reflexiva y transitivas

b) $A \leq_1 B$ entonces $A \leq_m B$

c) $A \leq_1 B$ entonces $\overline{A} \leq_1 \overline{B}$

d) $A \leq_m B$ entonces $\overline{A} \leq_m \overline{B}$

e) $A \leq_m B$ y B es un conjunto recursivo entonces A es un conjunto recursivo

---

[21] El término abreviado m −reducible proviene del inglés *many-one* reducible y 1 − reducible del término inglés *one-one* reducible

f) $A \leq_m B$ y B es un conjunto recursivamente enumerable entonces A es un conjunto recursivamente enumerable

Demostración Las pruebas de (a), (b), (c) y (d) son inmediatas

(e) Si $A \leq_m B$ por medio de una función total y recursiva f, consideremos las funciones características $\mathcal{X}_A$ y $\mathcal{X}_B$ de A y B respectivamente, tal que[22]

$$\mathcal{X}_A(x) = \mathcal{X}_B(f(x)) = \begin{cases} 1 & si\ f(x) \in B \\ 0 & si\ f(x) \notin B \end{cases} = \begin{cases} 1 & si\ x \in A \\ 0 & si\ x \notin A \end{cases}$$

por lo que si $\mathcal{X}_B$ es recursiva entonces $\mathcal{X}_A$ también, de manera que A es un conjunto recursivo

(f) Si $A \leq_m B$ por medio de una función total y recursiva f y $B = rang\ f$ entonces $A = f^{-1}(B)$ es recursivamente enumerable

La parte (a) del teorema anterior establece una relación de equivalencia $\equiv_1$ o $\equiv_m$ Su clases de equivalencias son llamados $1-$grados de irresolubilidad respecto a 1-reducible y m $-$grados de irresolubilidad con respecto a m $-$reducible

Definición 5.2.3 Sean $A \subseteq \mathbb{N}$ y $B \subseteq \mathbb{N}$ dos conjuntos, definimos las relaciones de equivalencias $\equiv_1$ y $\equiv_m$ como los grados de equivalencias $deg_1(A)$ y $deg_m(A)$ mediante

(a) $A \equiv_m B$ si y sólo si $A \leq_m B\ y\ B \leq_m A$

(b) $A \equiv_1 B$ si y sólo si $A \leq_1 B\ y\ B \leq_1 A$

(c) $deg_m(A) = \{B\ \ A \equiv_m B\}$

(d) $deg_1(A) = \{B\ \ A \equiv_1 B\}$

[22] Se utiliza composición de funciones

Los resultados presentados proporcionan una técnica para demostrar la irresolubilidad de numerosos problemas, tal como lo ilustraremos

Definición 2.5.4 Definamos el conjunto $Tot = \left\{x \ \varphi_x^{(1)} \ es\ una\ función\ total\right\}$

Teorema 2.5.4 $K \leq_1 Tot$

Demostración Consideremos la siguiente función, que la llamaremos función auxiliar

$$\psi(x,y) = \begin{cases} 1 \ si\ x \in K \\ \uparrow \ si\ x \notin K \end{cases}$$

La función se puede escribir de la forma

$$\psi = C_1^{(1)} \circ \left(U^{(2)} \circ \left(P_1^{(2)}, P_1^{(2)}\right)\right)$$

la cual es recursiva Tomemos w como el índice para ψ, de manera que $\psi = \varphi_w^{(2)}$, por la propiedad 5 del capítulo 1, existe una función inyectiva y total s tal que

$$\varphi_{s_x^w}^{(1)}(y) = \psi(x,y) = \varphi_w^{(2)}(x,y)$$

Tomando $f = s_x^w = s \circ \left(C_w^{(1)}, P_1^{(1)}\right)$ es recursiva, 1-1 y total por lo que satisface la relación siguiente

$$\varphi_{f(x)}^{(1)} = \begin{cases} C_1^{(1)} \ si\ x \in K \\ \Phi^{(1)} \ si\ x \notin K \end{cases}$$

Vemos que si $x \in K \Leftrightarrow \varphi_{f(x)}^{(1)}\ es\ total \Leftrightarrow f(x) \in Tot$

2.6 Conjuntos recursivamente isomorfos y Teorema de Recursión de Kleene

Los conceptos de 1-grados e isomorfismo recursivo coinciden, este un resultado debido a Myhill en 1955, cuyo contenido y prueba se presenta en el teorema que lleva su nombre y que está estrechamente relacionado con el teorema de Cantor-Scröder-Bernstein para numeros cardinales Además se analizará el teorema de la Recursión de Kleene conocido como teorema del punto fijo, que establece la recursividad de ciertas funciones definidas en forma implícita

Definición 2.6.1 A las biyecciones recursivas de $\mathbb{N}$ las llamaremos *permutaciones recursivas* Un predicado de conjunto se dice ser recursivamente invariante si es invariante bajo todas las permutaciones recursivas

Proposición 2.6.1 Las permutaciones recursivas con la operación de composicion forman un grupo

Definición 2.6.2 Sean A y B subconjuntos de $\mathbb{N}$ Se dice que son recursivamente isomorfos, se denota $A \equiv B$, si existe una permutación recursiva p tal que

$$x \in A \Leftrightarrow p(x) \in B$$

Significa que $p(A) = B$ Las clases de equivalencias bajo la relación " $\equiv$ " son llamadas tipos recursivamente isomórficos

Teorema 2.6.1 (Isomorfismo de Myhill) $A \equiv B \Leftrightarrow A \equiv_1 B$

Demostración ($\Longrightarrow$) Es inmediata por definición

($\Longleftarrow$) Supongamos que $A \leq_1 B$ por medio de la función f y $B \leq_1 A$ vía la funcion g Construiremos una permutación h por etapas, de tal manera que $x \in A \Leftrightarrow h(x) \in B$ Consideremos una sucesión de funciones recursivas (no totales) $h_0 \subset h_1 \subset h_2 \subset$ con $h = \bigcup_s h_s$ tal que $h_0 = \emptyset$ y h_s es la porción de h definida al final de la etapa s dada En h_s, cada paso, añadiremos a los sumo un par más de manera de podamos comprobar que

en la etapa h_{2s+1} la pertenencia a $dom\, h_s$ y en la etapa h_{2s+2} que pertenezca al $rango\, h_s$, conjuntos que supondremos finitos

Etapa $s+1 = 2x+1$ Asumamos que h_s es inyectiva, $dom\, h_s$ es finito y

$$y \in A \Leftrightarrow h_s(y) \in B$$

para todo $y \in dom\, h_s$ Si $h_s(x)$ es definida no hay nada que hacer En caso contrario, enumeremos el conjunto

$$\{f(x), f(h_s^{-1} \circ f(x)), \quad , f((h_s^{-1} \circ f(x))^n), \quad \}$$

hasta que encontremos un elemento y tal que $y \notin rango\, h_s$ Como f y h_s son funciones inyectivas y $x \notin dom\ h_s$, hacemos $h_{s+1}(x) = y$ por lo que si $x \in A \Leftrightarrow y \in B$

Etapa $s+1 = 2x+2$ Definamos $h^{-1}(x)$ de forma similar, reemplazando f, h_s, $rang$ y dom, respectivamente

El teorema nos proporciona una forma rápida y conveniente para demostrar isomorfismos recursivos, además observemos en la demostración anterior, que la construcción no depende de A ni de B sino unicamente de f y g

De esto se deduce el siguiente corolario

Corolario 2.6.1 $K \equiv K_0$

Teorema 2.6.2 (de la recursión de Kleene) Para toda función recursiva f $\mathbb{N} \to \mathbb{N}$, existe un índice n (llamado punto fijo para f) tal que

$$\varphi_n^{(1)} = \varphi_{f(n)}^{(1)}$$

Demostración Definamos la función recursiva diagonal $d(u)$[23] mediante

$$\varphi_{d(u)}(z) = \begin{cases} \varphi_{\varphi(u)}(z) & si\ \varphi_u(u)\downarrow \\ \uparrow & si\ \varphi_u(u)\uparrow \end{cases}$$

con d 1-1 y recursiva, independiente de f Sea v un índice tal que

$$\varphi_v = f \circ d$$

Si $n = d(v)$ es un punto fijo de f Primero observemos que al ser f total implica que $f \circ d = \varphi_v$ es total, de manera que $\varphi_v(v)$ es convergente y por tanto $\varphi_{d(v)} = \varphi_{\varphi_v(v)}$ Entonces $\varphi_n = \varphi_{d(v)} = \varphi_{\varphi_v(v)} = \varphi_{f(d(v))} = \varphi_{f(n)}$

Teorema 2.6.3 (recursión por parámetros, Kleene) Sea f una función binaria recursiva Entonces existe una función recursiva $\eta(y)$ tal que $\varphi_{\eta(y)} = \varphi_{f(\eta(y),y)}$

Demostración Definimos la función d por medio

$$\varphi_{d(x,y)}(z) = \begin{cases} \varphi_{\varphi_x(x,y)}(\bar{z}), & si\ \varphi_x(x,y)\downarrow \\ \uparrow & si\ \varphi_x(x,y)\uparrow \end{cases}$$

Sea v un índice de la función binaria y recursiva $f(d(x,y),y)$, esto es,

$\varphi_v(x,y) = f(d(x,y),y)$ Entonces $\eta(y) = d(v,y)$ será un punto fijo ya que

$$\varphi_{d(v\,y)} = \varphi_{\varphi_v(v,y)} = \varphi_{f(d(v,y)\,y)} = \varphi_{f(\eta(y),y)}$$

2.7 Conjuntos Completos, Productivos y Creativos

La $m-$reducibilidad es un preorden Consideremos $\sum_1$ como la familia de conjuntos recursivamente enumerables y a los elementos maximales se le denominan completos Los problemas correspondiente a los conjuntos completos son los mas difíciles entre los $\sum_1$

[23] La existencia y propiedades de la función d definida en el cap I, se debe al teorema s-m-n

Definición 2.7.1 A es 1 −completo con respecto a $\leq_1$ si

(i) A es recursivamente enumerable y

(ii) Para todo conjunto recursivamente enumerable W, $W \leq_1 A$

A es m −completo con respecto a $\leq_m$ si

(i) A es recursivamente enumerable y

(ii) Para todo conjunto recursivamente enumerable W, $W \leq_m A$

Teorema 2.7.1 Los conjunto K, K_0 y $K_1 = \{x \ W_x \neq \emptyset\}$ son 1 −completos

Demostración Los tres conjuntos son recursivamente enumerables

Como $K_0 = \{\pi(x,y) \ x \in W_y\}$ y $x \in W_y \Leftrightarrow \pi(x,y) \in K_0$ con W_y recursivamente enumerable, entonces K_0 es 1 −completo, es decir

$$W_y \leq_1 K_0$$

Consideremos un conjunto recursivamente enumerable cualquiera B, y una función recursiva y total $g(x)$ tal que

$$\varphi_{g(x)}(y) = \begin{cases} 1, & si\ x \in B \\ \uparrow, & si\ x \notin B \end{cases}$$

con lo que

$$W_{g(x)} = \begin{cases} \mathbb{N}, & si\ x \in B \\ \emptyset, & si\ x \notin B \end{cases}$$

Vemos que $x \in B$ si y sólo si $g(x) \in K$ por lo que $B \leq_1 K$ De igual manera se cumple si $g(x) \in K_1$ entonces $B \leq_1 K_1$

La noción de conjunto *productivo*[24] tiene cierta similitud con la de conjunto enumerable Para ver que un conjunto A no es enumerable basta ver que para cada conjunto numerable N con $N \subset A$, es posible encontrar un elemento $x \in A \setminus N$ De la misma forma un conjunto A no es recursivamente enumerable si para cualquier B recursivamente enumerable con $B \subset A$, existe un valor $x \in A \setminus B$ Es así que la noción de conjunto *productivo* impone que este elemento pueda encontrarse efectivamente

Definición 2.7.2 $A \subset \mathbb{N}$ Diremos que A es un conjunto productivo, si existe una función recursiva $\psi(x)$, que llamaremos función productiva de A, tal que

$$(\forall x)\left[W_x \subset A \Rightarrow [\psi(x) \downarrow \wedge (\psi(x) \in A \setminus W_x)]\right]$$

Por definición un conjunto A no es recursivamente enumerable, ya que si A fuese productivo y recursivamente enumerable no sería posible encontrar una función recursiva que esté en $A \setminus W_x$

Ejemplo 18

$K = \{x \; x \in W_x\}$ no es productivo por ser recursivamente enumerable

$\overline{K} = \{x \; x \notin W_x\}$ es productivo La identidad $(\psi(x) = x)$ es una funcion de producción de $\overline{K}$

Veremos ahora, conjuntos recursivamente enumerables cuyos complementos son productivos, como el ejemplo anterior Tales conjuntos se le conocen como *creativos*[25]

Definición 2.7.3 A es creativo si

(i) A es recursivamente enumerable y

(ii) $\overline{A}$ es productivo

[24] El término "productivo" es debido a Dekker
[25] Los conjuntos fueron llamados "creativos" por Post en 1944

Ejemplo 19 K es creativo, por el ejemplo 18

Un conjunto productivo es un conjunto para el que la prueba de no ser recursivamente enumerable se realiza recursivamente Un conjunto creativo es un conjunto recursivamente enumerable para el que la prueba de ser no recursivo se realiza recursivamente por medio de una función productiva de su complemento

Teorema 2.7.2 Todo conjunto productivo posee una función productiva 1-1, total y recursiva.

Demostración Sea A un conjunto productivo a través de la función productiva $\psi(x)$ Construiremos una sucesión finita de funciones productivas de A distintas, a partir de las cuales construiremos una función productiva estrictamente creciente, y por tanto inyectiva.

Por el teorema s-m-n, existe una función g recursiva total y 1-1 tal que

$$\varphi_{g(x)}(y) = \begin{cases} 1, & si\ \varphi(x)\downarrow \vee y = \psi(x) \\ \uparrow, & de\ lo\ contrario \end{cases}$$

Podemos ver facilmente que

$$W_x \subset W_{g(x)} \subset W_{g(g(x))} \subset \ \subset A$$

$$W_{g(x)} = W_x \cup \{\psi(x)\} \subset A$$

$$W_{g(g(x))} = W_{g(x)} \cup \{\psi(g(x)) \subset A\}$$

Denotaremos como g^j las j-veces que se realiza la composición de g en g Vemos que si $W_x \subset A$ todos los elementos del conjunto $S = \{\psi(x), \psi(g(x)), \psi(g^2(x)),\ \}$ son distintos por ser ψ la función productiva de A Entonces para cualquier $k \in \mathbb{N}$, si $W_x \subset A$, existe en S entre los primeros $k+1$ elementos, algún elemento mayor que k (que no pertenece al conjunto)

De esta manera todas las $\psi(g^j)$ son funciones productoras de A Así,

$$W_x \subset A \Longrightarrow W_{g^j(x)} \subset A \Longrightarrow \psi\left(g^j(x)\right) \in A \setminus W_x$$

Definiendo h de forma primitiva-recursiva para $n \in \mathbb{N}$, obtenemos,

$$h(0) = 0$$
$$h(n+1) = \begin{cases} \psi(g^r(n+1)), & si\ \psi(g^j(n+1)) > h(n)\ existe \\ h(n)+1, & en\ otro\ caso \end{cases}$$

Para $r = \mu j \leq h(n)$, la función h es recursiva total y estrictamente creciente y por tanto inyectiva Además es función productiva de A, ya que, como se puede observar en a figura 5,

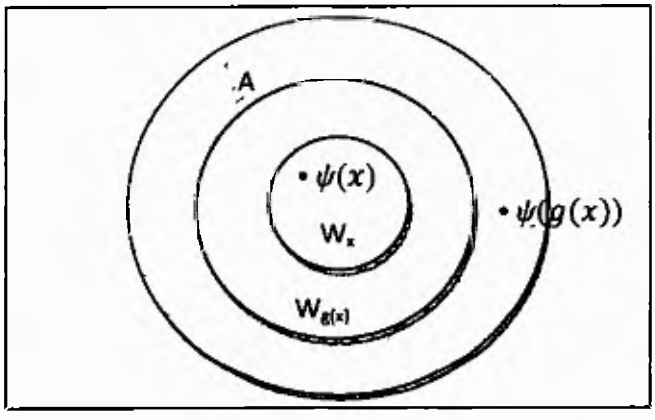

Figura 5
Ilustración del teorema 2 7 2

si $W_x \subset A$ al ser los elementos de S todos distintos existirán j tal que

$$h(x) = \psi(g^j(x)) \subset A \setminus W_x$$

El teorema nos indica que si $W_x \subset A$ y ψ la función productiva de A, $W_x \cup \{\psi(x)\}$ es un conjunto recursivamente enumerable contenido en A y por tanto un cierto $W_{g(x)} \subset A$, con lo que $\psi(g(x)) \in A \setminus W_{g(x)}$

Teorema 2.7.3 Sea $A \subset \mathbb{N}$ un conjunto,

(a) A productivo entonces A no es recursivamente enumerable

(b) A productivo, entonces A contiene un conjunto infinito recursivamente enumerable W

(c) A productivo y $A \leq_m B$ entonces B es productivo

Demostración

(a) Inmediato por definición

(b) Consideremos algún índice $n \in \mathbb{N}$, tal que $W_n = \emptyset$ y sea h la función total recursiva tal que $W_{h(x)} = W_x \cup \{f(x)\}$, donde f es la funcion productiva de A Entonces tenemos el conjunto

$$W = \{f(n), f(h(x)), f(h^2(x)), \quad \}$$

que es infinito y recursivamente enumerable

(c) Sea A reducible a B vía f y sea g la función productiva de A Supongamos que $W_x \subset B$ Sea h una función recursiva tal que $f^{-1}(W_x) = W_{h(x)}$ Entonces $W_{h(x)} \subset A$ por lo tanto $g \circ h(x) \in A \setminus W_{h(x)}$ y $f \circ g \circ h(x) \in B \setminus W_x$ Así $f \circ g \circ h$ es la función productiva de B

La parte c) del teorema anterior, nos da una forma más conveniente de exhibir otros conjuntos productivos A, mostrando que $\overline{K} \leq_m A$ y como sabemos $\overline{K}$ es un conjunto productivo Mostraremos que todos los conjuntos productivos tienen esta

propiedad y que todos los conjuntos creativos son 1-completos y por lo tanto recursivamente isomorfos a K, en el siguiente teorema

Teorema 2.7.4 Sean A y C son subconjuntos de $\mathbb{N}$

(a) Si A es productivo, entonces $\overline{K} \leq_1 A$

(b) Si C es creativo, entonces C es 1-completo y $C \equiv K$

Demostración (a) Sea p una función total, 1-1 y productiva de A Sea f una función recursiva definida mediante

$$W_{f(x,y)} = \begin{cases} \{p(x)\} & si\ y \in K \\ \emptyset & si\ y \notin K \end{cases}$$

Por el teorema 2 6 3, existe una función recursiva 1-1, $\eta(y)$ tal que

$$W_{\eta(y)} = W_{f(\eta(y),y)} = \begin{cases} \{p(\eta(y))\} & si\ y \in K \\ \emptyset & si\ y \notin K \end{cases}$$

Si $y \in K \Longrightarrow W_{\eta(y)} = \{p(\eta(y))\} \Longrightarrow W_{\eta(y)} \nsubseteq A \Longrightarrow p(\eta(y)) \in \overline{A}$

y además si $y \in \overline{K} \Longrightarrow W_{\eta(y)} = \emptyset \Longrightarrow W_{\eta(y)} \subseteq A \Longrightarrow p(\eta(y)) \in A$

Ambas implicaciones nos lleva a que $\overline{K} \leq_1 A$ por medio de la función $p \circ \eta$

(b) Como C es creativo, $\overline{C}$ es productivo entonces de (a) $\overline{K} \leq_1 \overline{C}$ Ademas C es recursivamente enumerable y K creativo tenemos que $C \leq_1 K$, por el teorema 2 5 3 se deduce que $\overline{C} \leq_1 \overline{B}$, por lo tanto C es $1-$completo y aplicando el teorema de Myhill

$C \equiv K$

Los siguientes corolarios resumen los resultados anteriores

Corolario 2.7.1 Las siguientes afirmaciones son equivalentes

(a) A es productivo

(b) $\overline{K} \leq_1 A$

(c) $\overline{K} \leq_m A$

Corolario 2.7.2 Las siguientes afirmaciones son equivalentes

(a) C es creativo

(b) C es 1 −completo

(c) C es m −completo

Por los resultados presentados podemos ver que las propiedades de isomorfismos a K, 1 −completo y m −completo coinciden

2.8 Cilindros, conjuntos simples e inmunes

Para cada conjuntos A y B, $A \equiv_1 B \Longrightarrow A \equiv_m B$ De esta forma cada m −grado podria ser visto como conjunto de uno o más 1 −grados, es decir, tipos de isomorfismos Del cual surge la pregunta ¿Podemos especificar más, la estructura simple de un m −grado en términos de sus componentes 1 −grado?

Definición 2 8.1 Un conjunto A es un cilindro si $A \equiv B \times \mathbb{N}$, para algun conjunto $B \subset \mathbb{N}$

Esto significa que el proceso de asociar a cada cilindro aritmetico un numero natural de A es el conjunto

$$A \times \mathbb{N} = \{(x, n)\ x \in A\}$$

llamado proceso de *cilindrificación*[26]

[26] Dado un numero natural v la cilindrificación de C_v se define como

$$D_{C_v} = \{\pi(n,k)\ n \in D_v\}$$

$$C_v(\pi(n,k)) = v(i),$$

donde $\pi(n,k)$ es la función codificadora de Cantor

Observemos en la definición el uso de la relación " $\equiv$ " en lugar de la relación " $=$ ", lo que conlleva a que el concepto de cilindro sea recursivamente invariante El siguiente teorema muestra algunas propiedades entre las $1-$ reducciones y las m $-$reducciones

Teorema 2.8.1 Sean A y B dos subconjuntos de $\mathbb{N}$ Entonces

(a) $A \leq_1 A \times \mathbb{N}$ (por lo tanto $A \leq_m A \times \mathbb{N}$)

(b) $A \times \mathbb{N} \leq_m A$ (por lo tanto $A \equiv_m A \times \mathbb{N}$)

(c) A es un cilindro $\Leftrightarrow$ $(\forall B)[B \leq_m A \Leftrightarrow B \leq_1 A]$

(d) $A \leq_m B \Leftrightarrow A \times \mathbb{N} \leq_1 B \times \mathbb{N}$

Demostración (a) $A \leq_1 A \times \mathbb{N}$ vía la función recursiva $f(x) = \pi(x, x)$

(b) $A \times \mathbb{N} \leq_m A$ es inmediato vía la función recursiva $f(x, y) = x$

(c) ($\Longrightarrow$) Asumiremos que A es un cilindro, entonces existe un conjunto $C \subset B$ tal que $A \equiv C \times \mathbb{N}$ y supongamos que $B \leq_m A$, entonces por (b) obtenemos

$$B \leq_m A \equiv C \times \mathbb{N} \leq_m C$$

Así, $B \leq_m C$ vía alguna función recursiva f Sea $g(x) = (f(x), x)$ una función inyectiva y recursiva por lo que $B \leq_1 C \times \mathbb{N}$ vía g Entonces por el isomorfismo de Myhill, $B \leq_1 A$

($\Longleftarrow$) Supongamos que $(\forall B)[B \leq_m A \Leftrightarrow B \leq_1 A]$ Consideremos el conjunto $A \times \mathbb{N}$ y por (b) $A \times \mathbb{N} \leq_m A$ De la hipotesis tenemos que $A \times \mathbb{N} \leq_1 A$ y por (a) $A \leq_1 A \times \mathbb{N}$ de esta forma $A \equiv_1 A \times \mathbb{N}$ Del isomorfismo de Myhill concluimos que $A \equiv A \times \mathbb{N}$, demostrando que A es un cilindro

(d) ($\Longrightarrow$) Supongamos que $A \leq_m B$, aplicando (a) y (b) obtenemos

$$A \times \mathbb{N} \leq_m A \leq_m B \leq_1 B \times \mathbb{N}$$

Por lo que $A \times \mathbb{N} \leq_m B \times \mathbb{N}$, utilizando (c) de tiene que $A \times \mathbb{N} \leq_1 B \times \mathbb{N}$

($\Leftarrow$) Supongamos que $A \times \mathbb{N} \leq_1 B \times \mathbb{N}$ Entonces,

$$A \leq_1 A \times \mathbb{N} \leq_1 B \times \mathbb{N} \leq_m B$$

concluyendo que $A \leq_m B$

Las partes (a), (b) y (c) del teorema 2 8 1 muestran que cada m −grado contiene a lo sumo un 1 −grado, y este 1 −grado es posible obtenerlo de cualquier conjunto en el m −grado por medio del proceso de *cilindrificación* En la parte (d) la estructura del ordenamiento m − reducible se refleja naturalmente en el 1 −ordenamiento, esto es que existe un homomorfismo canónico entre el m −ordenamiento y el 1 −ordenamiento

En el siguiente corolario se asocian otras propiedades cuya demostración es inmediata

Corolario 2.8.1 Es equivalente

(a) A es un cilindro

(b) $A \times \mathbb{N} \leq_1 A$

(c) $A \equiv A \times \mathbb{N}$

Lema 2.8.1 Sea un conjunto A recursivamente enumerable Supongamos que para todo conjunto B recursivamente enumerable, se tiene que $B \leq_m A \Rightarrow B \leq_1 A$ Entonces A es un cilindro

Demostración De la parte (b) del teorema 2 8 1 tenemos que $A \times \mathbb{N} \leq_m A$ Como $A \times \mathbb{N}$ es recursivamente enumerable, por hipótesis $A \times \mathbb{N} \leq_1 A$ Por el corolario 2 8 1 se deduce que A es un cilindro

Corolario 2.8.2 Sea A un conjunto recursivamente enumerable tal que $K \leq_m A$ Entonces, A es un cilindro

Demostración como vimos K es m −completo y $K \leq_m A$ (por hipótesis) Se deduce que A es m −completo, del corolario 2 7 2, entonces A es 1 −completo Por lo que para todo conjunto recursivamente enumerable B, tenemos que $B \leq_m A$ y $B \leq_1 A$, concluyendo que A es un cilindro

De lo analizado hasta el momento quedan interrogantes abiertas como

1 ¿Todos los conjuntos recursivamente enumerable y no-recursivos, serán creativos?

2 ¿Todos los conjuntos recursivamente enumerable y no-recursivos, serán m −completos?

3 ¿las relaciones $\leq_m$ y $\leq_1$, coincidirán sobre los conjuntos recursivamente enumerable y no-recursivos?

La íntima relación entre el método de diagonalización y la irresolubilidad, podria sugerir una respuesta positiva a la pregunta 1 Por otra parte, los conjuntos recursivamente enumerables y no-recursivos presentados en los ejemplos nos sugiere una respuesta afirmativa para la respuesta 2 y finalmente para la pregunta 3, el corolario 2 7 2 nos podria dar una respuesta positiva

Otra pregunta que surge, propuesta por (Post, 1944), es ¿Si existe algún conjunto infinito que no contenga algun subconjunto infinito recursivamente enumerable? La misma dio origen al término de *conjunto simple* definido a continuación

Definición 2.8.2 $A \subseteq \mathbb{N}$, es un conjunto simple si

(i) A es recursivamente enumerable

(ii) $\overline{A}$ es infinito

(iii) $(\forall B)\big[[B\ infinito\ \wedge B\ r\ e\ ] \Longrightarrow B \cap A \neq \emptyset\big]$

Un conjunto recursivamente enumerable es simple si su complemento es infinito y no contiene subconjuntos infinitos recursivamente enumerable

Teorema 2.8.2 Si A es simple entonces

(i) A no es recursivo

(ii) A no es creativo

(iii) A no es m −completo (eso es $K \not\leq_m A$)

Demostración (i) $\overline{A}$ no es recursivamente enumerable, por definición

(ii) El complemento de un conjunto creativo contiene un conjunto infinito recursivamente enumerable por el teorema 2 7 3 De manera que A no es creativo

(iii) Si $K \leq_m A$, entonces A es creativo por corolario 2 7 2

Teorema 2.8.3 Existe un conjunto simple S

Demostración Sea ψ la funcion recursiva definida

$$\psi(e) = \sigma_1(min_t T(e, \sigma_1(t), \sigma_2(t)) \wedge \sigma_1(t) > 2e)$$

Claramente el predicado $P \leftrightarrow T(e, \sigma_1(t), \sigma_2(t)) \wedge (\sigma_1(t) > 2e)$ es primitivo-recursivo, por lo que ψ es una función parcial recursiva Sea $S = rang\ \psi$ Vamos a demostrar que S es simple, pero antes expliquemos cuál es el significado intuitivo de S para cada $e \in \mathbb{N}$ enumeramos el conjunto W_e y buscamos el primer natural $x = \sigma_1(t)$ que pertenece que a W_e y satisfice $x > 2e$ Si existe, lo ponemos en S Esto es,

$$\psi(e) = min_x[x \in W_e \wedge x > 2e]$$

Empleando la "la enumeración diagonal", de W_e para buscar el valor de x

(1) S es recursivamente enumerable, pues es el rango de una función recursiva

(2) $\overline{S}$ es infinito Razonamos por contero sea $f(u)$el número de elementos de S que son menores o iguales a u Busquemos una estimación de $f(2n+2)$, esto es, cuántos números $\psi(e)$ hay tales que $\psi(e) \leq 2n+2$ Para este propósito, podemos restringirnos a valores $e \leq n$, debido a que para $e \leq n+1$ sabemos que $\psi(e) > 2e \geq 2n+2$ Luego, a lo sumo habrá $n+1$ de tales números $\psi(0), \psi(1), \ , \psi(n)$ Por consiguiente, $f(2n+2) \leq n+1$ Esto significa que a lo sumo $n+1$ de los primeros $2n+2$ números naturales pertenecen a S Se deduce entonces que al menos $n+1$ de los primeros $2n+2$ números naturales pertenecen a $\bar{S}$ Por lo tanto $\bar{S}$ es infinito

(3) Veamos que $\bar{S}$ no contiene ningún subconjunto recursivamente enumerable e infinito W_e es infinito, entonces existe algún $x \in W_e$ tal que $x > 2e$ El menor de estos número $\psi(e)$, pertenecerá entonces a $W_e \cap S$ Luego W_e no es subconjunto de $\overline{S}$

Otros conjuntos, presentados por Dekker, con la propiedad atribuida al complemento de un conjunto simple, se les conocen como *conjuntos inmunes*

Definición 2.8.3 $A \subset \mathbb{N}$ es un conjunto inmune si

(i) A es infinito

(ii) $(\forall B)\left[[B \ infinito \ \wedge B \ r \ e \ ] \Rightarrow B \cap A \neq \emptyset\right]$

La familia de conjuntos inmunes es similar en algunos aspectos a la familia de conjuntos finitos

Teorema 2.8.4 Existen $2^{\aleph_0}$ conjuntos A tales que ambos A y $\overline{A}$ son conjuntos inmunes

Demostración Sean $x_0, x_1,$ elementos de $\{x \ W_x \ es \ infinito\}$ en orden creciente Definamos la sucesión de pares $\{y_0, z_0\}, \{y_1, z_1\},$ donde $\{y_0, z_0\}$ representa los dos

elementos menores de W_{x_0}, para $y_0 < z_0$, y de forma sucesiva $\{y_{k+1}, z_{k+1}\}$ son los dos menores elementos de W_{x_k} que son mayores que y_k y z_k, con $y_{k+1} < z_{k+1}$ Formamos un conjunto A, seleccionando un entero de cada elemento que forman las parejas de la sucesión Claramente hay $2^{\aleph_0}$ distintas maneras de hacer esto Tanto A como $\overline{A}$ intersecan a cada conjunto recursivamente enumerable e infinito Por consiguiente tanto A como $\overline{A}$ son inmunes

¿Será cierto que todo conjunto recursivamente enumerable y no-recursivo es creativo o simple? Esta pregunta es respondida de forma negativa por el siguiente teorema

Teorema 2.8.4 Existe un conjunto recursivamente enumerable que no es recursivo, ni simple, ni creativo

Demostración Sea A un conjunto simple y consideremos el conjunto $A \times \mathbb{N}$ que recursivamente enumerable puesto que A lo es $A \times \mathbb{N}$ no es recursivo pues $A \leq_1 A \times \mathbb{N}$, por el teorema 2 8 1 Además $A \times \mathbb{N}$ no es simple debido a que es un cilindro Finalmente es fácil demostrar que si $A \times \mathbb{N}$ es creativo entonces A sería creativo, hecho que sería falso bajo el supuesto que A es simple

Corolario 2.8.3 Existen conjuntos que no son recursivamente enumerables, ni productivos, ni inmunes

Demostración Basta con tomar $\overline{A \times \mathbb{N}}$, en la prueba del teorema anterior

2.9. Conjunto de índices e indecidibilidad

En la demostración del teorema 2 5 4, el conjunto K podría ser reemplazado por cualquier otro conjunto recursivamente enumerable, de manera que si A es ese conjunto se tendría que $A = W_\iota = dom\, \varphi_\iota^{(1)}$, para algún índice $\iota \in \mathbb{N}$ Pero esto no se cumple para conjuntos recursivamente no-enumerable El método en la demostración se puede

aplicar a otros conjuntos distintos de Tot siempre y cuando ese conjunto sea caracterizado por la propiedad de sus funciones y no del índice de algunas de las mismas, esto es a condición de que A sea un conjunto de índices, concepto que formalizamos a continuación

Definición 2.9.1 Llamaremos al conjunto $A \subseteq \mathbb{N}$, conjunto de índices si

$$\left[x \in A \wedge \varphi_x^{(1)} = \varphi_y^{(1)}\right] \Rightarrow y \in A$$

para todo $x, y \in \mathbb{N}$

Teorema 2.9.1 Si A es un conjunto de índices no trivial[27] entonces $K \leq_1 A$ o bien $K \leq_1 \overline{A}$

Demostración Consideremos un índice w para la función nula $\phi^{(1)}$ de manera que $\phi^{(1)} = \varphi_w^{(1)}$ Si $w \in \overline{A}$ entonces veremos que $K \leq_1 A$, de manera analoga si $w \in A$ se obtiene que $K \leq_1 \overline{A}$

Supongamos que $w \in \overline{\overline{A}}$ Como $A \neq \emptyset$, podemos seleccionar un índice $e \in A$ y como A es un conjunto de índice, entonces $\varphi_e^{(1)} \neq \varphi_w^{(1)}$ Por el teorema s-m-n, existe una función recursiva, 1-1 y total f tal que

$$\varphi_{f(x)}^{(1)} = \begin{cases} \varphi_e^{(1)}(y), & si\ x \in K \\ \uparrow, & si\ x \notin K \end{cases}$$

Ahora,

$$x \in K \Rightarrow \varphi_{f(x)}^{(1)} \Rightarrow \varphi_e^{(1)} \Rightarrow f(x) \in A$$
$$x \in \overline{K} \Rightarrow \varphi_{f(x)}^{(1)} \Rightarrow \varphi_t^{(1)} \Rightarrow f(x) \in \overline{A}$$

[27] A es un conjunto no trivial si $A \neq \emptyset$ y $A \neq \mathbb{N}$

De la primera afirmación se deduce que $K \leq_1 A$ La última implicación de cada línea se sigue porque A es un conjunto de indices

Corolario 2.9.1 (Rice[28]) Sea $\mathcal{C}$ una clase de funciones recursivas parciales de una variable Entonces el conjunto $\left\{n \ \varphi_x^{(1)} \in \mathcal{C}\right\}$ es recursivo si y sólo si $\mathcal{C} = \emptyset$ o $\mathcal{C}$ es la clase de todas las funciones parciales recursivas de una variable

Demostración Sea $A = \left\{n \ \varphi_x^{(1)} \in \mathcal{C}\right\}$ que es un conjunto de indices

($\Longrightarrow$) Supongamos que $A \neq \emptyset$ y $A \neq \mathbb{N}$ entonces por el teorema 2 9 1, se tiene que $K \leq_1 A$ o $K \leq_1 \overline{A}$ En ambos casos K no puede ser recursivo, por el teorema 2 4 3, lo que es una contradicción Entonces $\mathcal{C} = \emptyset$ o $\mathcal{C}$ es la clase de todas las funciones parciales recursivas de una variable

($\Longleftarrow$) Por otro lado si $\mathcal{C} = \emptyset$ entonces $A = \emptyset$ y si $\mathcal{C}$ es la clase de todas las funciones parciales recursivas de una variable entonces $A = \mathbb{N}$ En ambos casos A es recursivo

Es posible que tanto $K \leq_1 A$ y $K \leq_1 \overline{A}$ por un conjunto de índices A, como el caso de $A = Tot$ Se puede probar que ni Tot ni $\overline{Tot}$ son recursivamente enumerables en (Piza Volio, 2001) Existen otros conjuntos, además de los analizados k, k_0 y Tot, con problemas de irresolubilidad dados en definicion siguiente

Definición 2.9.2 Se define los siguientes conjuntos de índice, que corresponden a problemas naturales indecidibles relacionados con las funciones recursivas

$$K_1 = \{x \ W_x \neq \emptyset\},$$

$$Fin = \{x \ W_x \ es \ finito\},$$

[28] El teorema de Rice es un teorema enunciado por Henry Gordon Rice, lógico y matematico que fue profesor en la Universidad de New Hampshire En 1960 fue empleado por la Corporacion de Ciencias Computacionales en El Segundo California Este teorema se generalizó en conjunto con John Myhill y Norman Shapiro posteriormente

$$Inf = \{x \ W_x \ es \ infinito\} = \mathbb{N} \setminus Fin,$$

$$Con = \{x \ \varphi_x \ es \ total \ y \ constante\},$$

$$Cof = \{x \ W_x \ es \ cofinito\},$$

$$Rec = \{x \ W_x \ es \ recursivo\},$$

$$Ext \equiv \left\{x \ \varphi_x^{(1)} \ es \ extendible \ a \ una \ función \ total\right\},$$

$$Subset = \{\langle x, y\rangle \ W_x \subseteq W_y\}$$

Los conjuntos descritos, son conjuntos de índices no recursivos por el teorema de Rice, entre ellos los conjuntos k, k_0 y k_1 son recursivamente enumerables y por lo tanto tienen el mismo 1-grado En el teorema 2 6 1 se comprobó que estos conjuntos son recursivamente isomorfos

Definición 2.9.3 Un conjunto A recursivamente enumerable es 1 −completo si $W_n \leq_1 A$ para todo un conjunto recursivamente enumerable W_n

Debido a que $x \in W_n$ si y sólo si $\pi(x, n) \in k_0$ es claro que K_0 es 1 −completo (corolario 2 6 1) Igualmente se puede comprobar que K y K_1son también 1 −completo Los conjuntos de índices restantes, en la definición 2 9 2, y sus complementos no son recursivamente enumerable como se puede ver en (Soare, 1987) Además en (Soare, 1987) se muestra que

$$Inf \equiv_1 Tot \equiv_1 Con \ , Cof \equiv_1 Rec \equiv_1 Ext$$

Propiedades que serán de gran utilidad para el estudio de la clasificacion de conjuntos en el capitulo consecuente

CAPITULO III

CLASIFICACIÓN DE CONJUNTOS EN LA JERARQUÍA ARITMÉTICA DE KLEENE

III. Clasificación de conjuntos en la Jerarquía Aritmética de Kleene

En el capítulo anterior, se analizaron los conceptos de grado e irreducibilidad para clasificar conjuntos de enteros Estudiaremos una clasificación relacionada y amplia de conjuntos la cual da una idea añadida de la teoría precedente y se muestra la conexión entre la teoría de las funciones recursivas y la lógica La Jerarquía Aritmética, conocida como la Jerarquía de Kleene[29], nos da una medida precisa para clasificar conjuntos de índices, produciendo mayor información que los métodos primitivos, los cuales se fundamenta en el teorema s-m-n Esta clasificación constará de conjuntos construidos a partir de conjuntos sencillos, por aplicaciones estrictamente alternantes de las operaciones de proyección y/o complementación (negación) Una forma más atractiva de presentar la jerarquía es, sin embargo, el uso de los cuantificadores existencial ($\exists$) y universal ($\forall$), es decir, reemplazamos la complementación por el cuantificador universal y la proyección por el cuantificador existencial

3.1 Relación de las funciones recursivas con la lógica de primer orden

Iniciamos con una caracterización en términos de la proyección de relaciones recursivas

Definición 3.1.1 La proyección de una relación R, a los largo de la $\iota-$ésima coordenada, es el conjunto

$$A = \{(x_1, \quad, x_{\iota-1}, x_{\iota+1}, \quad, x_k) / (\exists x_\iota)\ R(x_1, \quad, x_k)\}$$

Llamaremos A la proyección de una relación $R \subseteq \mathbb{N}^k$

Enunciaremos dos teoremas que en conjunto se le conocen como el teorema de Proyección o teoremas del Cuantificador Existencial, los cuales se utilizan en la demostración de la enumerabilidad recursiva

[29] Steve Kleene (1943) quien fue el primero en estudiar la Jerarquía Aritmética Su trabajo en la teoría de la recursión estableció los fundamentos teóricos de las ciencias de la computación al proveer métodos para determinar cuáles problemas eran resolubles

Teorema 3.1.1 Si R es recursivamente enumerable, entonces existe una relación recursiva S tal que R es una proyección de S Específicamente, para una relación k −aria R recursivamente enumerable entonces existe una relación $(k+1)$ −aria recursiva S, tal que

$$R = \{(x_1, \quad, x_k) \,/\, (\exists x_{k+1})\ S(x_1, \quad, x_k, x_{k+1})\}$$

Demostración Sea $S = \pi^k(R)$, S es un conjunto recursivamente enumerable Por el teorema 2 4 6, existe una función parcial recursiva y un índice fijo ω de la función φ tal que $S = dom\ \varphi_\omega$ Consideremos P_ω un conjunto correspondiente de instrucciones (de MT) para la funcion φ, de manera que

$$\begin{aligned} R(x_1, \quad, x_k) &\Leftrightarrow (x_1, \quad, x_k) \in S \\ &\Leftrightarrow \varphi((x_1, \quad, x_k)) \downarrow \\ &\Leftrightarrow (\exists x_{k+1})[P_\omega \textit{ con entrada } (x_1, \quad, x_k) \textit{ y} \\ &\textit{salida en un número de pasos menor que } x_{k+1}] \end{aligned}$$

Como produce una salida, por la Tesis de Church, se define una relación recursiva sobre $\mathbb{N}^k$, lo cual demuestra el teorema □

Corolario 3.1.1 La proyeccion de una relación recursivamente enumerable es recursivamente enumerable

Establecida la proyección de una relación y las condiciones para que un conjunto sea recursivamente enumerable, revisaremos los conjuntos en una Jerarquía Aritmética.

Definición 3.1.2 Una relación R está en la Jerarquia Aritmética si R es recursivo, o si existe una relación recursiva S tal que R pueda ser obtenida de S por alguna sucesion finita de operaciones de proyección y/o complementación

Observemos que una relación recursivamente enumerable está en la Jerarquía Aritmética si toda relación recursivamente enumerable se puede obtener por alguna relación recursiva por medio de la proyección, teorema 3 1 1 El complemento de alguna

relación recursivamente enumerable está, también, en la jerarquía ya que puede ser obtenido de una relación recursiva por medio de las operaciones de proyección y complementación, en ese orden

Definición 3.1.3 Sea un A un conjunto R está en la Jerarquía Aritmética en A si Res A −recursiva o si existe una relación A −recursiva S tal que R pueda ser obtenido de S por alguna sucesión finita de operaciones complementación y/o proyección

En (Rogers, 1992), se verifica que todo conjunto recursivamente enumerable en un conjunto A, está en la Jerarquía Aritmética en A

Es importante utilizar una notación para indicar las operaciones de proyección y complementación, esto nos lo proporciona la lógica, pues la proyección puede ser indicada por el cuantificador existencial y la complementación por la negación, representada por el cuantificador universal, Esto se debe a la equivalencia siguiente

$$\forall x[\phi U] \Leftrightarrow \neg\exists x[\neg\phi U]$$

Ejemplo 20 Sea R una relación n −aria El complemento de R es el conjunto

$$\{(x_1, \quad, x_n) \mid \neg R(x_1, \quad, x_n)\}$$

y una de las n posibles proyecciones de R es

$$\{(x_1, \quad, x_n) \mid (\exists x_n) R(x_1, \quad, x_n)\}$$

además, de la aplicación sistemática de proyección, proyección, complementación, proyección y complementación a R, se obtiene

$$\{(x_1, \quad, x_n) \mid \neg\, (\exists x_{n-2}) \neg (\exists x_n)(\exists x_{n-1}) R(x_1, \quad, x_n)\}$$

Se puede reescribir como

$$\{(x_1, \quad, x_n) \mid (\forall x_{n-2})(\exists x_n)(\exists x_{n-1}) R(x_1, \quad, x_n)\}$$

De manera general, utilizaremos Q como una notación conveniente para denotar el cuantificador $\exists$ ó $\forall$

Teorema 3.1.2 Una relación n −aria R está en la Jerarquia Aritmética si y sólo si, R es recursiva o para algún $m \in \mathbb{N}$, puede ser expresada como

$$\{(x_1, \quad, x_n) | (Q_1 y_1) \quad (Q_m y_m) S(x_1, \quad, x_n, y_1, \quad, y_m)\}$$

donde Q_i es $\exists$ o $\forall$ para $1 \leq i \leq m$, y S es una relación $(n+m)$ −aria recursiva

Demostración Dada una sucesión de operaciones de proyección y complementación aplicada a una relación recursiva, las reglas de la lógica pueden ser usadas y las coordenadas de la relación recursiva (o sus complementos) pueden ser reordenadas para producir la expresión deseada con cuantificadores

Inversamente, dada una secuencia de cuantificadores, el hecho de que el cuantificador $\forall$ es equivalente al cuantificador $\neg\exists\neg$, se puede utilizar para reemplazar todos los cuantificadores universales por simbolos de negación y cuantificadores existenciales, produciendo una sucesión de operaciones de proyeccion y complementación □

El teorema que antecede nos presenta una expresión que relaciona cuantificadores, símbolos de relación y relaciones recursivas sobre coordenadas distintas, esto en su conjunto se le llama *formas predicativas*

Ejemplo 21 Sea S una relación recursiva 4 −aria, entonces

$$(\exists x_3)(\forall x_1) S(x_1, x_2, x_3, x_4)$$

junto con la relación S es una forma predicativa, y

$$\{(x_2, x_4) | (\exists x_3)(\forall x_1) S(x_1, x_2, x_3, x_4)\}$$

es la relación expresada por esta forma

Definición 3.1.4 Sea A un conjunto dado Si en la forma predicativa se reemplaza la relación recursiva por una relación A −recursiva, al resultado se le llama una A −*forma*

Corolario 3.1.2 R está en la Jerarquía Aritmética si y sólo si existe una forma predicativa que exprese R Y por otro lado, R está en la Jerarquía Aritmética en A si y sólo si existe una A −forma que exprese R

Definición 3.1.5 Llamaremos *prefijo* a la secuencia de cuantificadores que antecede un predicado

Ejemplo 22 El prefijo del predicado $\exists(x_1)\exists(x_2)\forall(x_3)R(x_1,x_2,x_3,x_4)$ tiene el prefijo $\exists\exists\forall$

Definición 3.1.6 El número de alternancias en un prefijo es el número de cambios de un cuantificador a otro

En el caso del prefijo $\exists\exists\forall\exists$, el número de alternancias es 2

3.2 Niveles de recursividad en la Jerarquía Aritmética

Definiremos los conjuntos $\sum_n$ y $\prod_n$, que clasifican las formas de los predicados (y las relaciones que expresan) de acuerdo al número de alternancias en sus prefijos Esta clasificación será de significado fundamental en la Jerarquía Aritmética

Definición 3.2.1 Para $n > 0$, un $\sum_n$ −prefijo es un prefijo que inicia con el cuantificador $\exists$ y si inicia con el cuantificador $\forall$ se le llama $\prod_n$ − prefijo, ambas con $(n-1)$ alternancias

Definición 3.2.2

(i) Un conjunto B esta en $\sum_0$ $(\prod_0)$ si y si solo si B es recursivo

(ii) Para $n \geq 1$, B está en $\sum_n$, $B \in \sum_n$, si existe una relación recursiva $R(x, y_1, \quad , y_n)$ tal que

$$x \in B \Leftrightarrow (\exists y_1)(\forall y_2)(\exists y_3) \quad (Qy_n)\, R(x, y_1, \quad , y_n)$$

donde Q es el cuantificador $\exists$, si n es impar y el cuantificador $\forall$ si n es par De la misma forma B está en $\prod_n$, $B \in \prod_n$, si

$$x \in B \Leftrightarrow (\forall y_1)(\exists y_2)(\forall y_3) \quad (Qy_n)\, R(x, y_1, \quad , y_n),$$

donde Q puede ser los cuantificadores $\exists$ o $\forall$ si n es par o impar, respectivamente

(iii) B está en Δ_n si $B \in \sum_n \cap \prod_n$

(iv) B está en la Aritmética si $B \in \bigcup_n (\sum_n \cap \prod_n)$

Notemos que B está en la Jerarquia si y sólo si se puede obtener a partir de relaciones recursivas por un número finito de aplicaciones de proyección y complementación

Definición 3.2.3 Sea A un conjunto fijo Si reemplazamos en la definicion 3 2 2, recursiva por A −recursiva entonces B pertenece a $\sum_n$ en A, (ó $B \in \sum_n^A$) , B pertenece a $\prod_n$ en A, (ó $B \in \prod_n^A$), $B \in \Delta_n^A$ y B esta en la Aritmética de A

Por el corolario 3 1 2, una relación está en la Jerarquía Aritmética si y solo si, para algún entero positivo n, es miembro de $\sum_n$ o de $\prod_n$ Esto se cumple de igual manera para la Jerarquía Aritmética en un conjunto A Así $\bar{\sum}_0$ ($\bar{\prod}_0$) es la clase de todas las relaciones recursivas Revisaremos posteriormente que, $\sum_1$ es la clase de todas las relaciones recursivamente enumerables Igual para $\sum_0^A$ ($\prod_0^A$) es la clase de todas relaciones A −recursivas y $\sum_1^A$ es la clase de todas las relaciones recursivamente enumerables en A En (Soare, 1987), se muestra que las clases $\sum_0, \sum_1, \sum_2,$ forman una sucesión estrictamente creciente

El siguiente teorema establece la equivalencia entre los conjuntos recursivamente enumerables y las funciones parciales recursivas

Teorema 3.2.1 (Teorema de la forma normal para conjuntos recursivamente enumerables) Un conjunto A es recursivamente enumerable si y sólo si A está en $\sum_1$

Demostración ($\Longrightarrow$) Si A es un conjunto recursivamente enumerable, entonces $A = W_e = dom\ \varphi_e$ para algún índice e Puesto que

$$x \in W_e \Leftrightarrow (\exists s)[x \in W_{e,s}] \Leftrightarrow (\exists s)T(e,x,s)$$

por el predicado $T(e,x,s)$, es recursivo primitivo

($\Longleftarrow$) Sea $A = \{x\ \ (\exists y)R(x,y)\}$, donde R es recursivo Entonces para alguna funcion recursiva ψ, $A = dom\ \psi$ donde $\psi(x) = (\mu y)R(x,y)$ por la definición 2 1 2

Teorema 3.2.2 Si existe una relación recursiva $R \subseteq \mathbb{N}^{n+1}$ y

$$A = \{x\ \ (\exists y_1)(\exists y_2)\quad (\exists y_n)\bar{R}(x,\vec{y})\ \}$$

entonces A está en $\sum_1$

Demostración Sea $S \subseteq \mathbb{N}^2$ una relación recursiva definida por

$$S(x,z) \Leftrightarrow R(x,(z)_1,(z)_2,\quad ,(z)_n)$$

donde $z = p_1^{(z)_1}p_2^{(\bar{z})_2}\quad p_k^{(z)_k}$ es la descomposicion de z, visto en el capítulo 1, así

$$\begin{aligned}(\exists z)S(x,z) &\Leftrightarrow (\exists z)R(x,(z)_1,(z)_2,\quad ,(z)_n)\\ &\Leftrightarrow (\exists z)R(x,y_1,y_2,\quad ,y_n)\end{aligned}$$

En el capítulo 2, se demostró que los conjuntos K, K_0 y K_1 son recursivamente enumerables y 1 −completos, teorema 2 7 1, por lo tanto estos conjuntos están en $\sum_1$

Es un hecho que una relación definible dentro de la lógica de cuantificadores de relaciones recursivas, está en la Jerarquia Aritmética. De esto se establece las siguientes afirmaciones

1 Si una relación es definida por combinaciones de sentencias lógicas de relaciones recursivas, entonces es recursiva

2 Si F y G son expresiones tal que G no contiene ninguna ocurrencia no cuantificada de un simbolo a, entonces los siguientes pares de expresiones son equivalentes

$(\exists a)F \vee G$	$(\exists a)[F \vee G]$
$(\forall a)F \vee G$	$(\forall a)[F \vee G]$
$(\exists a)F \wedge G$	$(\exists a)[F \wedge G]$
$(\forall a)F \wedge G$	$(\forall a)[F \wedge G]$
$(\exists a)F \Rightarrow G$	$(\exists a)[F \Rightarrow G]$
$(\forall a)F \Rightarrow G$	$(\forall a)[F \Rightarrow G]$
$G \Rightarrow (\exists a)F$	$(\exists a)[G \Rightarrow F]$
$G \Rightarrow (\forall a)F$	$(\forall a)[G \Rightarrow F]$

3 Si $F(a)$ es una expresión definida en a, y si b es un variable que no está en $F(a)$ $F(b)$ es el resultado de reemplazar b por a en todas las ocurrencias no cuantificadas en $F(a)$ Entonces las siguientes parejas de expresiones son equivalentes

$(\exists a)F(a)$	$(\exists b)F(b)$
$(\forall a)F(a)$	$(\forall b)F(b)$

4 Si F y G son cualquieras expresiones, entonces las siguientes parejas de expresiones son equivalentes

$$\neg(\exists a)F \qquad (\forall a)\neg F$$

$$\neg(\forall a)F \qquad (\exists a)\neg F$$

$$\neg\neg F \qquad F$$

$$F \Leftrightarrow G \qquad [F \Rightarrow G] \wedge [G \Rightarrow F]$$

Los puntos descritos, conllevan a realizar la computación de una expresión en dos etapas Con el punto 1 se obtiene una definición del conjunto desde una relación recursiva y con los puntos 2, 3 y 4 se forma una cadena de equivalencias principales proporcionando una forma llamada *forma prenexa*

Definición 3.2.4 Una forma de lógica de predicado tiene la forma normal prenexa (fnp), si está escrita como una cadena de cuantificadores seguida por una parte sin cuantificador

Familiarizados con las reglas usual para la manipulación de cuantificadores de la lógica elemental, es posible convertir una fórmula a una equivalente en la forma normal prenexa Usando estas reglas podemos mostrar los hechos que se utilizan con frecuencia para demostrar que un conjunto particular esté en $\sum_n$ o $\prod_n$

Definición 3.2.5 Un cuantificador acotado es aquel que tiene de la forma $(Qx \leq y)F$, en donde

$$(\forall x \leq y)F, \quad \text{por } (\forall x)[x \leq y \Rightarrow F]$$

$$(\exists x \leq y)F, \quad \text{por } (\exists x)[x \leq y \wedge F]$$

en ambos casos x e y son símbolos de variables distintas

Para los cuantificadores acotados podemos tomar en cuenta los hechos siguientes

1 Un cuantificador puede ser movido a la derecha más allá de un cuantificador ordinario, si es posible realizar los cambios apropiados en las relaciones recursivas operado sobre este cuantificador

2 La aplicación de un cuantificador acotado a una relación recursiva produce una relación recursiva

Estos puntos se aclaran con el siguiente ejemplo

Ejemplo 23 Para la observación (1) consideremos $(\forall x \leq y)(\exists z)(\forall w)R(x,y,z,w)$ que es equivalente a $(\exists z)(\forall x \leq y)(\forall w)R\left(x,y,\pi_x^y(z),w\right)$, y es conocido que π_x^y es una función recursiva

Para la observación (2), si R es una relación recursiva 3 −aria, entonces

$$\{(x,u)|(\forall y \leq u)R(x,y,u)\}$$

es una relación recursiva binaria

Teorema 3.2 3

(i) $A \in \sum_n \Leftrightarrow \overline{A} \in \prod_n$,

(ii) $A \in \sum_n(\prod_n) \Rightarrow (\forall m > n)[A \in \sum_m \cap \prod_m]$,

(iii) $A, B \in \sum_n(\prod_n) \Rightarrow A \cup B, A \cap B \in \sum_n(\prod_n)$,

(iv) $[R \in \sum_n \wedge\ n > 0\ \wedge\ A = \{x\ (\exists y)R(x,y)\}] \Rightarrow A \in \sum_n$,

(v) $[B \leq_m A\ \wedge\ A \in \sum_n] \Rightarrow B \in \sum_n$,

(vi) Si $R \in \sum_n(\prod_n)$, y A y B están definidos por

$$\langle x,u\rangle \in A \Leftrightarrow (\forall z < y)R(x,y,z)$$

y

$$\langle x,u\rangle \in B \Leftrightarrow (\exists z < y)R(x,y,z)$$

entonces

$$A, B \in \sum_n(\prod_n)$$

Demostración

(i) Si $A = \{x \ (\exists y_1)(\forall y_2) \quad [R(x, \vec{y})]\}$ entonces $\overline{A} = \{x \ (\forall y_1)(\exists y_2) \quad [\neg R(x, \vec{y})]\}$

(ii) Consideremos el conjunto $A = \{x \ (\exists y_1)(\forall y_2)[R(x, y_1, y_2)]\}$, lo podemos expresar como $A = \{x \ (\exists y_1)(\forall y_2)(\exists y_3)[R(x, y_1, y_2) \wedge y_3 = y_3]\}$

(iii) Sea $A = \{x \ (\exists y_1)(\forall y_2) \quad [R(x, \vec{y})]\}$ y $B = \{x \ (\exists z_1)(\forall z_2) \quad [S(x, \vec{z})]\}$ entonces

$$\begin{aligned} x \in A \cup B &\Leftrightarrow (\exists y_1)(\forall y_2) \quad R(x, \vec{y}) \vee (\exists z_1)(\forall z_2) \quad S(x, \vec{z}) \\ &\Leftrightarrow (\exists y_1)(\exists z_1)(\forall y_2)(\forall z_2) \quad [R(x, \vec{y}) \vee S(x, \vec{z})] \\ &\Leftrightarrow (\exists u_1)(\forall u_2) \quad [R(x, (u_1)_0, (u_2)_0, \quad) \vee S(x, (u_1)_1, (u_2)_1, \quad)] \end{aligned}$$

donde $(u_1)_0$ es de la forma visto en el ejemplo 8 De igual manera se demuestra para $A \cap B$

(iv) Es inmediata por la aplicacion de cuantificador contraccion en (iii) y el teorema 3 2 2

(v) Sea $A = \{x \ (\exists y_1)(\forall y_2) \quad R(x, \vec{y})\}$ y $B \leq_m A$ vía f Entonces

$$B = \{x \ (\exists y_1)(\forall y_2) \quad R(f(x), \vec{y})\}$$

(vi) La prueba es por inducción sobre n Si $n = 0$, es claro que A y B son recursivos Para $n > 0$, supongamos que $R \in \sum_n$ se cumple para todo $m < n$ Entonces por (iv) $B \in \sum_n$

Existe $S \in \prod_{n-1}$ tal que

$$\begin{aligned}\langle x,y\rangle \in A &\Leftrightarrow (\forall z < y)R(x,y,z)\\ &\Leftrightarrow (\forall z < y)(\exists u)S(x,y,z,u)\\ &\Leftrightarrow (\exists \sigma)(\forall z \leq y)R(x,y,z,\sigma(z))\end{aligned}$$

donde σ oscila sobre $\mathbb{N}^{<\mathbb{N}}$ Tenemos por la hipótesis de inducción que $(\forall z < y)S \in \prod_{n-1}$ por hipótesis de inducción, de donde $A \in \sum_n$ Para el caso $R \in \prod_n$ se sigue de (1), cuando $R \in \sum_n$ □

Las propiedades analizadas hasta el momento nos permitirán clasificar algunos conjuntos, definidos en el capítulo 2

Proposición 3.2.1 $Fin \in \sum_2$

Demostración

$$\begin{aligned}Si\ x \in Fin &\Leftrightarrow W_x\ es\ finito\\ &\Leftrightarrow (\exists s)(\forall t)\Big[[t \leq s] \vee \big[W_{xt} = W_{x,s}\big]\Big]\end{aligned}$$

La relación, en los corchetes, de x, s, t es claramente recursiva

Se puede deducir de la proposición 3 2 1, que el conjunto $Inf \in \prod_2$

Proposición 3.2.2 $Cof \in \sum_3$

Demostración

$$\begin{aligned}Si\ x \in Cof &\Leftrightarrow \overline{W}_x\ es\ finito\\ &\Leftrightarrow (\exists y)(\forall z)\big[[z \leq y] \vee [z \in W_x]\big]\\ &\Leftrightarrow (\exists y)(\forall z)(\exists s)\Big[[z \leq y] \vee \big[z \in W_{xs}\big]\Big]\end{aligned}$$

Proposición 3.2.3 $Subset \in \prod_2$

Demostración

$$\begin{aligned} W_y \subseteq W_x &\Leftrightarrow (\forall z)[z \in W_y \Rightarrow z \in W_x] \\ &\Leftrightarrow (\forall z)[z \notin W_y \vee z \in W_x] \\ &\Leftrightarrow (\forall z)[\forall \vee \exists] \\ &\Leftrightarrow \forall\forall\exists[\quad] \\ &\Leftrightarrow \forall\exists[\quad] \end{aligned}$$

Corolario 3.2.1 $Tot = \left\{x \ \varphi_x^{(1)} es\ una\ función\ total\right\} = \{x \ W_x = \mathbb{N}\} \in \prod_2$

Demostración Considerando el conjunto con $W_y = \mathbb{N}$ en la proposición 3 2 4, queda demostrado

Proposición 3.2.4 $Rec \in \sum_3$

Demostración

$$\begin{aligned} Si\ x \in Rec &\Leftrightarrow W_x\ es\ recursivo \\ &\Leftrightarrow (\exists y)[W_x = \overline{W}_y] \\ &\Leftrightarrow (\exists y)\left\{[W_x \cap W_y = \emptyset] \wedge [W_x \cup W_y = \mathbb{N}]\right\} \\ &\Leftrightarrow \exists[\forall \wedge \forall\exists] \\ &\Leftrightarrow \exists\forall\exists[\quad] \end{aligned}$$

Proposición 3.2.5 $Simp = \{x \ W_x\ es\ simple\} \in \prod_3$

Demostración $Si\ x \in Simp \Leftrightarrow W_x\ es\ simple$

$$\begin{aligned} &\Leftrightarrow \left[\overline{W}_x\ es\ infinito \wedge (\forall y)[W_y\ es\ infinito \wedge W_x \cap W_y \neq \emptyset]\right] \\ &\Leftrightarrow [(\forall z)(\exists y)[y > x \wedge y \notin W_x] \wedge (\forall y)[(\forall u)(\exists v)[v > u \wedge v \in W_y] \\ &\qquad \Rightarrow (\exists w)[w \in W_y \wedge w \in W_z]]] \end{aligned}$$

La última expresión se puede abreviar, de la forma

$$\forall\exists\left[\wedge\,\forall[\forall\bar{\exists}(\wedge\,\bar{\exists})]\Longrightarrow\bar{\exists}[\bar{\exists}\wedge\bar{\exists}]\right]$$
$$\forall\exists\forall\wedge\forall[\forall\exists\Longrightarrow\exists]$$
$$\forall\exists\forall\wedge\forall\exists[\forall\exists\Longrightarrow]$$
$$\forall\exists\forall\wedge\forall\exists\exists\forall$$
$$\forall\exists\forall$$

Por lo tanto $Simp \in \prod_3$

Proposición 3.2.6 $Ext = \{x \ \ \varphi_x \ es \ extendible \ a \ una \ función \ total\} \in \sum_3$

Demostración

$$Si\ x \in Ext \Leftrightarrow (\exists y)\left[[\varphi_x \subseteq \varphi_y] \wedge [W_y = \mathbb{N}]\right]$$
$$\Leftrightarrow (\exists y)(\forall z)(\exists s)(\exists t)\left[\left[[z \notin W_{x,s}] \vee [\varphi_{x,s}(z) = \varphi_{y,s}(z)]\right] \wedge z \in W_{y\,t}\right]$$

Como los prefijos son $\exists\forall\exists$ se obtiene que $Ext \in \sum_3$

Proposición 3.2.7 $Crea = \{x \ W_x es\ creativo\} \in \sum_3$

Demostración

$$Si\ x \in Crea \Leftrightarrow \overline{W_x}\ \ es\ \ productivo$$
$$\Leftrightarrow (\exists y)(\forall z)\left[W_z \subseteq \overline{W_x} \Rightarrow \left[[\varphi_y(z)\downarrow \wedge \varphi_y(z)] \in \overline{W_x} \setminus W_z\right]\right]$$
$$\Leftrightarrow (\exists y)(\forall z)\left[W_z \cap W_x = \emptyset \Rightarrow \left[[\varphi_y(z)\downarrow \wedge \varphi_y(z)] \notin W_x \cup W_z\right]\right]$$
$$\Leftrightarrow (\exists y)(\forall z)\left[W_z \cap W_x \neq \emptyset \vee \left[[\varphi_y(z)\downarrow \wedge \varphi_y(z)] \notin W_x \cup W_z\right]\right]$$

Ahora $W_z \cap W_x \neq \emptyset$ si y sólo si

$$(\exists s)[W_{z,s} \cap W_{x,s} \neq \emptyset]$$

y para $[\varphi_y(z)\downarrow \wedge \varphi_y(z)] \notin W_x \cup W_z$ si y solo si

$$(\exists s)[z \in W_{y,s}] \wedge (\forall s)\left[[z \notin W_{y,s} \vee \varphi_{y,s}(z)] \notin W_{x,s} \cup W_{z,s}\right]$$

Utilizando la propiedades de la lógica de primer orden se obtiene un prefijo del tipo $\exists\forall\exists$, probando que $Crea \in \sum_3$

3.3 Teorema de jerarquía estricta y grados

Estudiaremos otra metodología para clasificar algunos conjuntos en donde relacionaremos el concepto de grado y reducibilidad con la lógica Introduciremos un concepto llamado *operador salto* definido por Kleen y Post que corresponde a la noción de la lógica infinita con respecto al número de cuantificadores en un predicado como una variable Para el mejor entendimiento denotaremos como **0** el grado de todas las funciones recursivas así, se puede revisar en (Soare, 1987), en el capítulo XIII, que el operador salto provee una jerarquía de grados $\mathbf{0} < \mathbf{0}' <$ $< \mathbf{0}^{(n)} <$, cuyos conceptos es utilizados en el teorema de Post, en los conjuntos $\sum_n -$ completos

Definición 3.3.1

(i) Llamaremos al conjunto $K^A = \{x\ \varphi_x^A(x)\downarrow\} = \{x\ x \in W_x^A\}$ el salto de A y se denota A'

(ii) Llamaremos $A^{(n)}$ al n −ésimo salto de A, que es obtenida iterando el salto n veces, esto es $A^{(0)} = A$, $A^{(n+1)} = \left(A^{(n)}\right)'$

Algunas propiedades básicas acerca del operador A' sobre conjuntos se recopilan en el siguiente teorema El operador salto parece depender de una selección de numeración de Godel

Teorema 3.3.1

(i) $B \leq_T A \Leftrightarrow B$ y $\bar{B}$ son recursivamente enumerable en A

(ii) A' es recursivamente enumerable en A

(iii) $A' \not\leq_T A$

(iv) B es recursivamente enumerable en A si y sólo si $B \leq_1 A'$

(v) Si A es recursivamente enumerable en B y si $B \leq_T C$ entonces A es recursivamente enumerable en C

(vi) $B \leq_T A$ si y sólo si $B' \leq_1 A'$

(vii) Si $B \equiv_T A$ entonces $B' \equiv_1 A'$ $(por\ lo\ tanto\ B' \equiv_T A')$

(viii) A es recursivamente enumerable en B si y sólo si A es recursivamente enumerable en $\bar{B}$

Demostración

(i) Del teorema 2 4 1

(ii) Si A' es recursivamente enumerable, la función parcial $f(x) = \varphi_x^A(x)$ es A −recursiva y $dom\ f = A'$ Entonces A' es recursivamente enumerable en A

(iii) Supongamos que A' es A −recursiva entonces $\bar{A}'$ es A − recursivo enumerable Por lo que existe un índice e tal que

$$W_e^A = \bar{A}' = \{x \ x \notin W_x^A\}$$

Entonces $e \in A'$si y sólo si $e \in \bar{W_e^A}$ lo que es una contradicción

(iv) ($\Longrightarrow$) Existe un índice e, tal que $B = W_e^A = dom\ \varphi_e^A$ Entonces $x \in B$ sii $\varphi_e^A \downarrow$ sii $\langle x, e\rangle \in K_0^A$ Como $g(x) = \langle x, e\rangle$ es 1-1, entonces $B \leq_1 K_0^A$ Además $K_0^A \equiv_1 K^A = A'$

($\Leftarrow$) Supongamos que f es una función recursiva tal que $x \in B$ sii $f(x) \in A'$ Entonces $x \in B$ sii $\varphi^{A}_{f(x)}(f(x)) \downarrow$ La función parcial $g(x) = \varphi^{A}_{f(x)}(f(x))$ es recursiva en A, entonces su dominio es recursivamente enumerable en A

(v) Si $A \neq \emptyset$ entonces es el rango de alguna funcion recursiva en B, y por lo tanto de alguna funcion recursiva en C, de donde $B \leq_T C$

(vi) ($\Rightarrow$) Si $B \leq_T A$ entonces B', por (v), es recursivamente enumerable en A entonces por (ii) B' es recursivamente enumerable en B De modo que $B' \leq_1 A'$ por (iv)

($\Leftarrow$) Si $B' \leq_1 A'$ entonces, por (iv), B y $\bar{B}$ son recursivamente enumerables en A (ya que $B, \bar{B} \leq_1 B'$) Por lo que $B \leq_T A$ por (i)

(vii) Directo de (vi)

(viii) Directo de (v)

Denotaremos $a, b, c,$ como grados y $\mathbf{0}$ los grados recursivos En (Rogers, 1992) si a es recursivamente enumerable en b ($a \leq b$), significa que existe un A en a tal que para algún B en b, A es recursivamente enumerable en B Además si A es recursiva en b, se refiere a que A es recursivo en B, para algún $B \in b$ Debido a esta afirmación, se tiene por ejemplo que $\emptyset = \{e \ e \in W_e\} \in \mathbf{0}$, $K \in \mathbf{0}'$, $(a \leq a')$, $(a' \not\leq a)$ [30] y además $(\forall n)[\emptyset^{(n)} \in \mathbf{0}^{(n)}]$

Definición 3.3.1 Un conjunto $A \in \sum_n$ es $\sum_n$ −completo si $B \leq_1 A$ para todo $B \in \sum_n$

Definición 3.3.2 Un conjunto $A \in \prod_n$ es $\prod_n$ −completo si $B \leq_1 A$ para todo $B \in \prod_n$

[30] Si a y b son grados, $a \leq b$ significa que a es T-reducible a b, y $a \not\leq b$ significa que son incoparables

Notemos que si A es $\sum_1$ −completo si y sólo si A es $1-$ completo, definición 2 7 1, por lo tanto K es $\sum_1$ −completo y $\bar{K}$ es $\prod_1$ −completo El siguiente teorema relaciona la jerarquía de salto de grados con la Jerarquia Aritmética.

Teorema 3.3.2 (Teorema de Post) Para todo $n \geq 0$

(i) $B \in \sum_{n+1} \Leftrightarrow \bar{B}$ es recursivamente enumerable en algún conjunto $\prod_n \Leftrightarrow B$ es recursivamente enumerable en algún conjunto $\sum_n$, parte (viii) del teorema 3 3 1,

(ii) $\emptyset^{(n)}$ es $\sum_n$ −completo para $n > 0$,

(iii) $B \in \sum_{n+1} \Leftrightarrow B$ es recursivamente enumerable en $\emptyset^{(n)}$,

(iv) $B \in \Delta_{n+1} \Leftrightarrow B \leq_T \emptyset^{(n)}$

Demostración

(i) ($\Longrightarrow$) Sea $B \in \sum_{n+1}$, entonces $x \in B \Leftrightarrow (\exists y)R(x,y)$ para algun $R \in \prod_n$ De esta forma B está $\sum_1$ en R y por lo tanto recursivamente enumerable en R, por el teorema 3 2 1

($\Longleftarrow$) Supongamos que B es recursivamente enumerable en algun $\prod_n$ en un conjunto C, entonces para algun índice e,

$$x \in B \Leftrightarrow x \in W_e^C$$
$$\Leftrightarrow (\exists s)(\exists \sigma)\left[\sigma \subset C \wedge \underbrace{x \in W_{e,s}^{\sigma}}_{recursiva}\right]$$

Basta probar que $\sigma \subset C$está en $\sum_{n+1}$ Ahora

$$\sigma \subset C \Leftrightarrow (\forall y < |\sigma|)[\sigma(y) = C(y)]$$

$$\Leftrightarrow (\forall y < |\sigma|)\left[\left[\underbrace{\sigma(y) = 1 \wedge y \in C}_{\prod_n}\right] \vee \left[\underbrace{\sigma(y) = 0 \wedge y \notin C}_{\sum_n}\right]\right]$$

Sabemos que $\prod_n, \sum_n \subseteq \sum_{n+1}$ por lo que $C \in \sum_{n+1}$

(ii) Por inducción para $n > 0$ Es claro para $n = 1$ Supongamos que $\emptyset^{(n)}$ es $\sum_n$ −completo, entonces $\overline{\emptyset^{(n)}}$ es $\prod_n$ −completo

$$B \in \sum_{n+1} \Leftrightarrow B \text{ es r e en algún conjunto} \sum_n \text{ por } (\iota)$$
$$\Leftrightarrow B \text{ es r e en } \emptyset^{(n)} \text{ por h } \iota$$
$$\Leftrightarrow B \leq_1 \emptyset^{(n+1)}, \quad (vii) \text{ teorema } 3\,3\,1$$

Por lo que $\emptyset^{(n+1)}$ es $\sum_{n+1}$ −completo

(iii) Por (i) y (ii) tenemos que $\overline{\emptyset^{(n)}}$ es $\prod_n$ −completo

(iv) $B \in \Delta_{n+1} \Leftrightarrow B, \bar{\bar{B}} \in \sum_{n+1} \Leftrightarrow B, \bar{\bar{B}}$ son recursivamente enumerable en $\emptyset^{(n)}$, por (iii), $\Leftrightarrow B \leq_T \emptyset^{(n)}$

Corolario 3.3.1 (Teorema de Jerarquia) $(\forall n > 0)[\Delta_n \subset \sum_n y\, \Delta_n \subset \prod_n]$

Demostración Si $\emptyset^{(n)} \in \sum_n$ Supongamos que $\emptyset^{(n)} \in \Delta_n$, por (iv) del teorema 3 3 2 $\emptyset^{(n)} \leq_T \emptyset^{(n-1)}$, lo que contradice (ii) del teorema 3 3 2 Entonces $\emptyset^{(n)} \in (\sum_n - \Delta_n)$ Similarmente $\overline{\emptyset^{(n)}} \in (\prod_n - \Delta_n)$

El teorema 3 3 2 y en el corolario 3 3 1 se cumple si la recursividad es en el operador de salto $A^{(n)}$, reemplazando $\sum_{n+1}(\prod_{n+1})$ por la notación $\sum_{n+1}^{A}(\prod_{n+1}^{A})$, conocido como *forma relativizada* De este modo tenemos el siguiente corolario, conocido como el teorema de completitud

Corolario 3.3.2 Para $n > 0$, algun conjunto B y A dado,

$$
\begin{aligned}
&1\ B \in \textstyle\sum_n^A \Leftrightarrow B \leq_1 A^{(n)},\\
&2\ B \in \textstyle\prod_n^A \Leftrightarrow B \leq_1 \overline{A^{(n)}},\\
&3\ B \in \textstyle\sum_n \Leftrightarrow B \leq_1 \emptyset^{(n)},\\
&4\ B \in \textstyle\prod_n \Leftrightarrow B \leq_1 \overline{\emptyset^{(n)}},\\
&5\ \textit{Si}\ A \leq_T B \Rightarrow \textstyle\sum_n^A \subseteq \sum_n^B,\\
&6\ \textit{Si}\ A \leq_T B \Rightarrow \textstyle\prod_n^A \subseteq \prod_n^B,\\
&7\ \textstyle\sum_{n+1} = \sum_n^{\emptyset'},\\
&8\ \textstyle\prod_{n+1} = \prod_n^{\emptyset'}
\end{aligned}
$$

Demostración (1) a (4) Inmediata del teorema 3 1 1, (5) y (6) triviales Para (7), $A \in \sum_{n+1}$ si sólo si A es recursivamente enumerable en $\emptyset^{(n)}$ Por (iii) del teorema 3 3 2, $A \in \sum_n^{\emptyset'}$ si y sólo si A es recursivamente enumerable en $(\emptyset')^{(n-1)} = \emptyset^{(n)}$ De igual forma para (8)

Ejemplo 24 Se mostró que el conjunto $Inf \in \prod_2$, debido al corolario 3 3 2 se obtiene que $Inf \leq_1 \overline{\emptyset^{(2)}}$

Definición 3.3.3 Para $n > 0$, $(\sum_n, \prod_n) \leq_m (C, D)$[31] si $(A, \bar{A}) \leq_m (C, D)$ para algun conjunto A $\sum_n$ −completo De forma similar para " $\leq_1$ "

La expresión del definición dada se puede escribir $\sum_n \leq_m C$ y $\prod_n \leq_m D$

Teorema 3.3.3 $(\sum_2, \prod_2) \leq_1 (Fin, Tot)$ Así Fin es $\sum_2$ − completo, Inf y Tot son $\prod_2$ −completo, y $Inf \equiv Tot$

Demostración En la proposición y corolario 3 2 1, se demostró, respectivamente, que $Fin \in \sum_2$, $(Inf \in \prod_2)$, y $Tot \in \prod_2$ Fijemos $A \in \sum_2$, de manera que $\bar{A} \in \prod_2$, por lo que existe una relación recursiva R tal que

$$x \in \bar{A} \Leftrightarrow (\forall y)(\exists z)[R(x, y, z)]$$

Por el teorema s-m-n existe una función recursiva 1-1 f, definida por

[31] La notación se justifica ya que $\sum_n$ y $\prod_n$ son clases en lugar de conjuntos

$$\varphi_{f(x)}(u) = \begin{cases} 0 \text{ si } (\forall y \leq u)(\exists z)R(x,y,z) \\ \uparrow \quad \textit{de otra mabera} \end{cases}$$

Ahora

$$x \in \bar{A} \Longrightarrow W_{f(x)} = \mathbb{N} \Longrightarrow f(x) \in Tot$$

pero si

$$x \in A \Longrightarrow W_{f(x)} \textit{ es finito} \Longrightarrow f(x) \in Fin$$

Definición 3.3.4: $Comp = \{x \ W_x = K\}$ esto es el conjunto de indices de un conjunto recursivamente enumerable completo

Teorema 3.3.4 $(\sum_3, \prod_3) \leq_1 (Cof, Comp) \leq_1 (Rec, Comp)$

Debido a su extensión y complejidad la prueba del teorema se puede revisar en (Soare, 1987)

Corolario 3.3.5 Cof es $\sum_3$ −completo

Demostración Se demuestra aplicando la proposición 3 2 2 y el teorema 3 3 4

Corolario 3.3.6 Rec es $\sum_3$ −completo

Demostración Por la proposición 3 2 4 y teorema 3 3 4 y el hecho de que

$$Rec \cap Comp = \emptyset$$

Introduciremos un conjunto importante y propiedades para demostrar que conjunto $Comp$ es $\sum_4 -$ completo Para esta demostración, realizada por Yates[32], combino la técnica de conjunto de indices con la llamada construcción $Injury$ infinito[33],

[32] Mike Yates, profesor honorario de la Escuela de Ciencias Computacionales de *Prifysgal Bangor University* Obtuvo con honor el Ph D en la Universidad de Manchester con su trabajo sobre los grados de irresolubilidad de Turing

[33] Infinity Injury Constructions

este último tema, es de utilidad como parte de la prueba pero no forma parte del desarrollo del estudio

Definición 3.3.5 Sea A un conjunto recursivamente enumerable, entonces el conjunto de índices, $G(A)$, se define como

$$G(A) = \{W_x \equiv_T A\}$$

Lema 3.3.1 Si A es recursivamente enumerable, entonces $G(A) \in \sum_3^A$

Demostración

$$\begin{aligned} Si\ x \in G(A) &\Leftrightarrow (\exists e)(\exists \iota)\left[[W_x = \varphi_e^A] \wedge \left[A = \varphi_\iota^{W_x}\right]\right] \\ &\Leftrightarrow (\exists e)(\exists \iota)\left[[\varphi_e^A\ total] \wedge \left[A = \varphi_\iota^{\varphi_e^A} total\right]\right. \\ \wedge (\forall y)&\left.\left[[y \in W_x \Leftrightarrow \varphi_e^A(y) = 1] \wedge [y \in A \Leftrightarrow \varphi_\iota^{W_x}(y) = 1]\right]\right] \end{aligned}$$

Ahora reemplazaremos la expresión $[y \in A \Leftrightarrow \varphi_\iota^{W_x}(y) = 1]$ por la siguiente

$$Si\ y \in A \Leftrightarrow (\exists s)(\exists \sigma)\left[\left[\varphi_{\iota,s}^{W_x}(y) = 1\right] \wedge \left(\forall z < lh(\sigma)^{34}\right)[\varphi_s^A(z) \downarrow = \sigma(x)]\right]$$

la nueva expresión está en la forma $\sum_1^A$ por lo que $G(A) \in \sum_3^A$

Los siguientes teoremas, demostrados por Yates, se puede revisar en (Soare, 1987)

Teorema 3.3.5 (Teorema de representación) Sea A un conjunto recursivamente enumerable Para algún conjunto $S \in \sum_3^A$, existe un conjunto $\{V_k\}_{k\in N}$ uniformemente recursivo enumerable en A, tal que para todo k,

$$k \in S \Rightarrow (\exists e_0)\left[(\forall e \geq e_0)\left[V_k^{[e]} \equiv_T A\right] \wedge (\forall e \geq e_0)\left[V_k^{[e]} recursivo\right]\right]$$

[34] $lh(\sigma)$ es la longitud de σ, donde σ es la función característica de un conjunto A, si la función es total $lh(\sigma) = |dom\ \sigma|$

y $k \notin S \Longrightarrow (\forall e)\left[V_k^{[e]} recursivo\right]$

Teorema 3.3.6 (Teorema de conjunto de índices) Dados dos conjunto recursivamente enumerable C y D, tal que $D \leq_T C$ y $S \in \sum_3^C$, existe una función recursiva, tal que para todo k,

(i) $D \leq_T W_{g(k)} \leq_T C$, y

(ii) $k \in S \Leftrightarrow W_{g(k)} \equiv_T C$

Corolario 3.3.7 Si C es un conjunto recursivamente enumerable entonces $G(C)$ es $\sum_3^C$ −completo

Demostración Por el Lema 3 3 1 se tiene que $G(C)$ es $\bar{\sum}_3^C$ Además si C es recursivo entonces aplicando, en conjunto, el corolario 3 3 6 y el teorema 3 3 6 para $D = \emptyset$, queda demostrado que $G(C)$ es $\sum_3^C$ −completo

Corolario 3.3.8 *Comp* es $\sum_4$ −completo

Demostración Por el corolario 3 3 6 con $C = \emptyset'$, $G(\emptyset') = \{W_x \equiv_T \emptyset'\}$ es $\sum_3^{\emptyset'}$ −completo Por el teorema el teorema de Post, (Teorema 3 3 2) $\sum_3^{\emptyset'} = \sum_4$

El corolario 3 3 8, nos provee la clasificación, deseada en el Teorema 3 3 4, probando que el conjunto *Comp* es $\sum_4$ −completo, asimismo numerosos conjunto de índices de conjuntos recursivamente enumerables se pueden clasificar en varios niveles más altos de la Jerarquía Aritmética

En la figura 6, se muestra la clasificación en la Jerarquía Aritmética para los conjuntos enteros dados

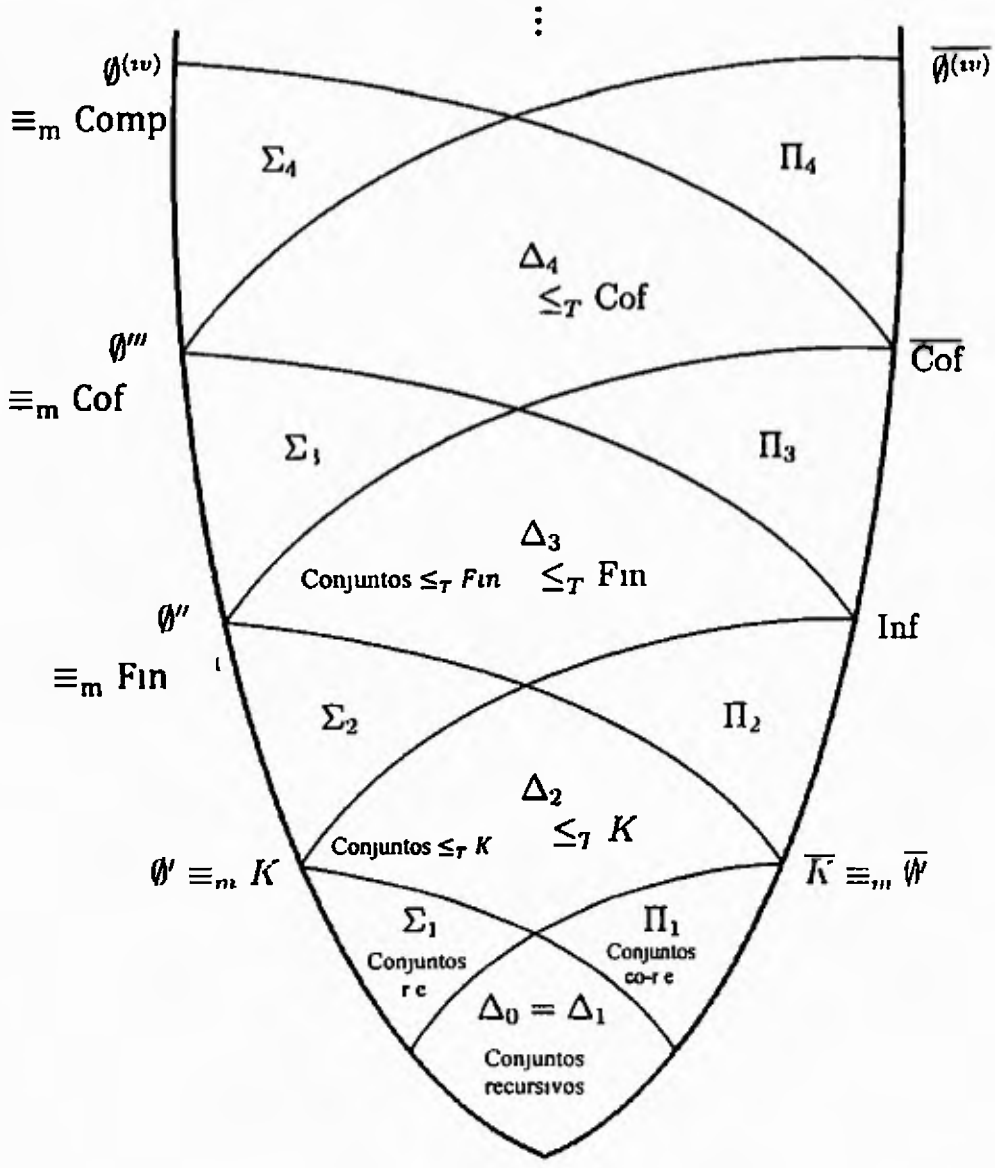

Figura 6
Jerarquía Aritmética de los conjunto de enteros

IV. Conclusiones y recomendaciones

Uno de los aspectos importantes, en la Teoría de Recursión, está relacionado con los problemas de decisión y la clase de las funciones recursivas que sean recursivamente irresolubles, es decir, indecidible Tales problemas son representados como conjuntos de enteros, de manera que su problema de decisión se reduce en determinar que dado un numero entero, si el mismo pertenece o no al conjunto

El teorema de Rice provee un suministro incontable de conjuntos no recursivos en que cada uno de estos, es un conjunto de índice no trivial Por Rice, cada uno es no recursivo y de hecho ningún conjunto es recursivo enumerable Mientras que cada uno de estos conjuntos es reducible al conjunto K, pero K no es reducible a estos conjuntos lo que significa que son más complicados para analizar

Al abordar el problema indecidibilidad de los conjuntos establecidos, la tecnica de diagonalización y el concepto de grado de irreducibilidad resultaron insuficientes, en la mayoría de los casos fue necesario el uso de la reducibilidad entre conjuntos, el cual nos permitió relacionar la indecidibilidad de unos conjuntos con otros estableciendo una jerarquia por medio de grados de reducibilidad llamada de forma general $\mathbf{0}^{(n)}$

Los conjuntos recursivamente enumerables establecen una Jerarquía Aritmética infinita, que corresponde al grado de irresolubilidad de los problemas de decisión asociados con estos conjuntos El método de clasificación combina la teoría de recursión con la lógica, en que la forma normal prenexa con el uso de los cuantificadores nos permitió agruparlos en conjuntos de referencias asociados a las operaciones de proyeccion y/o complementación, los cuales denotamos $\sum_n$ y $\prod_n$ Con estos conjuntos obtuvimos niveles de indecibilidad

Dentro de la Jerarquía Artimética de Kleene, los conjuntos recursivos resultan ser los más "sencillos" posibles, quedando por encima de los recursivamente enumerables que no son recursivos Vemos que dentro de los conjuntos recursivamente enumerables,

también existen conjuntos que podríamos decir que son más o menos "difíciles" y así sucesivamente

La clasificación presentada, es la mejor posible ya que nos indica que existen conjuntos que son más indecidible que otros, mostrando un indicio que es posible encontrar un grado de indecidibilidad tan grande como se desee y ademas es posible conocer que un conjunto es $\sum_n$ −completo probando que es 1-reducible a otro

Son diversas las aplicaciones la Jerarquía Aritmética de Kleene en problemas teóricos dentro de la Computación Matemática, como es el caso del conocido último teorema de Fermat el cual se demostró que está en $\prod_1$

En la actualidad existen problemas abiertos y conocidos con respecto a su complejidad en la Jerarquia Aritmética podemos mencionar el estudio del Teorema de los Cuatro Colores que está en $\prod_1$, el análisis de la Conjetura del Jacobiano, se refiere al comportamiento de los polinomios con coeficientes complejos, entre otros

V. Referencias Bibliográficas

AMBOS-SPIES, K , & FEJER, P A (20 de marzo de 2006) Degrees of Unsolvability Recuperado el 09 de enero de 2013, de http //www cs umb edu/~fejer/articles/History_of_Degrees pdf

CASTILLO, C I (15 de septiembre de 2013) Lógica y Teoría de Conjuntos Recuperado el 18 de junio de 2014, de http //www uv es/~ivorra/Libros/Logica pdf

DOWEK, G (2011) **Proofs and Algorith: An Introduction to Logic and Computability** Nueva York, Estados Unidos Springer

FERNÁNDEZ, M (2009) **Models of Computation: An Introduction to Computability Theory** Londres, Inglaterra Springer

FIGUIERA, S (2010) Teoria de la Computabilidad Recuperado el 03 de 03 de 2015, de http //www glyc dc uba ar/teocomp/2010/clases/teoricasTC pdf

FU, Y (2013) XIV Arithmetic Hierarchy Recuperado el 5 de 3 de 2015, de http //basics sjtu edu cn/~yuxi/teaching/computability2013/slides/14 %20Arithmetic%20Hierarchy pdf

GALLEGO CASTAÑO, E (Mayo de 2001) Tesis Doctoral **Técnicas de Demostración de Indecidibilidad e Inseparabilidad en Teorías Formal.** España, Madrid Tesis

GANDY, R (1988) **The confluence of ideas in 1936** A half-century survey on The Universal Turing Machine, 55-111

HEDMAN, S (2006) **A First Course in Logic: A Introduccion to Model Theory, Proof Theory, Computability, and Complexity** En S Hedman, A First Course in Logic A Introduccion to Model Theory, Proof Theory, Computability, and Complexity (pág 316) Oxford OXFORD, university Press

HENNIE, F (1977) **Introduction to Computability** Masschusetts Addison-Wesley Publishing

MONTALBAN, A (23 de enero de 2013) Matemática ComputabLe Recuperado el 22 de julio de 2014, de http //premat fing edu uy/papers/2013/151 pdf

PIZA VOLIO, E (2001) **Aritmética Recursvisa y algunas aplicaciones** Costa Rica Editorial Cimpa

POST, E L (mayo de 1944) Recuperado el 08 de 01 de 2015, de Recursively Enumerable Sets of Positive Integers and Their Decision Problems http //isites harvard edu/fs/docs/icb topic1237261 files/Post%201944%20Annotated pdf

ROGERS, H (1992) **Theory of Recursive Functions and Effective Computability** Estados Unidos McGraw-Hill

ROSENFIELD, R , & IRAZABAL, J (2013) **Computabilidad, complejidad computacional y verificación de programas** Buenos Aires, Argentina Editorial de la Universidad de La Plata

SOARE, R I (1987) **Recursively Enumerable Sets and Degrees**. Alemania Springer-Verlag

SOARE, R I (s f) Computability and Recursion Recuperado el 04 de febrero de 2014, de http //www people cs uchicago edu/~soare/History/compute pdf

SUDKAMP, T (2006) Languages and Machines An Introduction to the Theory of Computer Science En T Sudkamp, **Languages and Machines. An Introduction to the Theory of Computer Science** (pag 415) Addison-Wesley

VIDAL, J C (s f) Teoría de la Recurción y el Primer Teorema de Gödel Recuperado el 15 de febrero de 2015, de http //www uv es/~jkliment/Documentos/Godel pc pdf