UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS
"MAT. LUIS MANUEL RIVERA GUTIERREZ"

COMPUTABILIDAD Y EQUIVALENCIAS AL PROBLEMA DE LA DETENCIÓN

Tesis para obtener el grado de Licenciado en Ciencias Físico Matemáticas elaborada por:

César Ismael Corral Rojas

Supervisado por:
Dr. David Meza Alcántara

Morelia, Michoacán
Enero, 2016.

Agradecimientos

Muchas personas son las que merecen mención en esta parte, sin embargo, para evitar omisiones y hacer de esto algo extenso y tedioso, me centro sólo en aquellas a las que simplemente no puedo omitir.

Primeramente quiero agradecer a mis padres por su apoyo incondicional tanto en lo puramente escolar como en lo externo que me permitió continuar con esto hasta el final.

A Thelma y a Matías por convertirse en mi inspiración y un punto de inflexión en mis estudios y en la vida.

A los que comenzaron como mis profesores y se han convertido en mis amigos en estos años en la facultad, principalmente a mi canas David, por ofrecerme y asesorarme en este trabajo, por aconsejarme siempre con entusiasmo en todas las dudas que me surgieron como estudiante y también por apoyarme y alentarme para salir a escuelas y congresos a conocer más de este mundo de las matemáticas. No puedo evitar tampoco agradecer particularmente a Malú, por formarme como matemático desde el principio hasta el final, y por ser la primera persona que me hizo ver y trabajar las matemáticas de la forma en que lo hago hoy.

Y por último pero no menos importante, a mis compañeros y amigos con los que compartí estos años, a los cuales no hace falta mencionar individuamente, ustedes saben quienes son.

Índice general

Resumen

En este trabajo se utilizará una herramienta matemática conocida como *máquina de Turing* para clasificar problemas dentro de la misma matemática a partir de su complejidad, es decir, un problema más dificil de resolver que otro tendrá un grado de complejidad mayor. Dicho grado de complejidad de un problema matemático será medido dependiendo de la comparación de dicho problema con ciertos conjuntos definidos a partir de lo que puede y no puede hacer una *máquina de Turing*. Las *máquinas de Turing* son el predecesor teórico de las computadoras, y trabajar con *máquinas de Turing*, al ser objetos teóricos, nos dan la ventaja de no tener las limitaciones físicas de las computadoras, tales como el tiempo de cómputo y la cantidad de memoria finita. Se desarrollará primeramente la teoría de las *máquinas de Turing* y hasta llegar a un problema de las conocido como el problema de la detención, a partir del cual se darán varias aplicaciones. Dentro de las aplicaciones se calculará el grado de complejidad de tres problemas matemáticos:

- Calcular la componente conexa de un vértice en un grafo computable
- Calcular un ideal maximal dentro de un anillo conmutativo computable
- El décimo problema de Hilbert

los cuales resultarán ser equivalentes en complejidad, pero ninguno de ellos soluble.

Palabras clave: computabilidad, Turing, detención, computable, aritmetización.

Abstract

In this proyect a tool named *Turing machine* is used for clasifing math problems throught their complexity, that is, one is harder than other if it has a major complexity degre. The complexity degree of a math problem is measured from its relation with some special sets defining from what a *Turing machine* can do or do not. The *Turing machines* are the theoric ancestors of the computers, but working with *Turing machines* give us the advantage of we do not have physics limitations as with the real ones.

Firstly, we will develop the theory of *Turing machines* until we get to the well-known halting problem, after that we give some applications of it.

In the applications we study the complexity degree of three important problems:

- Compute the connected component of a vertex in a computable graph.
- Compute a maximal ideal in a conmutative computable ring.
- Hilbert's tenth problem.

All of these three problems are complexity-equivalents but none of them is solvable.

Introducción

A principios del siglo XX, David Hilbert junto con otros matemáticos intentaban formalizar la matemática de tal maner a que existiera una forma algorítmica de decidir la veracidad o falsedad de cualquier proposición matemática, y en particular, se preguntaban si existía forma de determinar si cualquier fórmula del cálculo de predicados de primer orden sobre los enteros era verdadera o no (Entscheidungsproblem). Sin embargo en 1931 Kurt Gödel presentó uno de sus teoremas de incompletitud, en el cual se delinea el modo de construir, dado un sistema axiomático con cualidades deseables específicas, una fórmula que no puede ser demostrada ni refutada. Como consecuencia de este teorema se tuvo que ninguna teoría consistente y decidible que contuviera a la aritmética de Peano podía ser completa, es decir, que para cada fórmula, ella o su negación fueran demostrables.

Algunos años más tarde se trató de formalizar la idea de algoritmo, y con ello separar procesos que sí eran algorítmicos, y procesos que no lo eran. En 1936 Alan Turing introdujo la idea de las Máquinas de Turing intentando formalizar la idea de algoritmo, y a pesar de que había otros modelos con el mismo fin, como las funciones recursivas o las funciones $\lambda - definibles$, todos ellos resultaron equivalentes.

Alonzo Church propuso que toda función intuitivamente calculable era λ-definible (y por tanto recursiva general, pues eran conceptos equivalentes). Poco después de esto, Turing propuso otro tipo de funciones, las funciones Turing-computables que eran precisamente las que podían ser calculadas por Maquinas de Turing. En 1937 Turing demostró que las funciones Turing-computables eran exactamente las funciones λ-definibles y por ende, las funciones recursivas. El concepto de que toda función intuitivamente calculable es una función Turing-computable es conocido como la tesis de Church-Turing.

Desde entonces, las máquinas de Turing han tenido una gran cantidad de aplicaciones, desde áreas aplicadas como el analisis de algoritmos, donde se utilizan para la clasificación y el estudio de las clases $\mathcal{P}$, $\mathcal{NP}$, y $\mathcal{NP}$-Hard principalmente, hasta áreas más abstractas como la lógica matemática. En esta última se han utilizado incluso para dar

un orden de «dificultad» a muchos de los problemas matemáticos, e incluso se les da una estructura de orden parcial construida en base a las máquinas de Turing.

En nuestro caso, utilizaremos dichas máquinas para encontrar problemas indecidibles dentro de las matemáticas, recurriendo principalmente al llamado problema de la detención, un problema indecidible concerniente a las mismas máquinas de Turing. Los problemas serán de distintas áreas de la matemática, como la combinatoria, la teoría de números o el álgebra.

El último de los problemas que demostraremos equivalente al problema de la detención, es el conocido como «El décimo problema de Hilbert», un problema referente a la existencia de un algoritmo general, para decidir si una ecuación diofántica, con cualquier cantidad de variables, tiene solución o no. Este problema planteado junto con los otros problemas de Hilbert en el año 1900, tardó 70 años en ser resuelto, hasta que se le dió una respuesta negativa por parte de Yuri Matiyasevich, tras más de 20 años de trabajo por parte de Matiyasevich y otros tres matemáticos más: Martin Davis, Julia Robinson y Hilary Putnam. El trabajo lo comenzó Martin Davis en 1949, y en años siguientes se fuernos conjeturando condiciones para la solubilidad del problema. Las conjeturas se fueron probando hasta llegar a la conclusión en 1970, con una publicación de Matiyasevich, de que los conjuntos diofanticos y los conjuntos recursivamente enumerables, eran exactamente los mismos, y con esto y algunos resultados de computabilidad que veremos más adelante, se concluyó que tal algoritmo no existía.

Aquí, como dijimos anteriormente, utilizaremos dicha demostración no solo para dar una respuesta negativa al décimo problema de Hilbert, sino también para demostrar que el problema de encontrar un algoritmo que nos responda si dada cualquier ecuación diofántica con cualquier cantidad de variables, tiene solución o no, es equivalente al problema de la detención.

Capítulo 1

Conceptos Básicos

Para hablar acerca de Máquinas de Turing, necesitamos primero hablar de lenguajes, y para ello, definiremos ciertos conceptos que nos llevarán a una definición formal. Intuitivamente un lenguaje es un conjunto de «palabras» formadas con ciertos símbolos de cierta forma que «tengan sentido», como las palabras en español o las expresiones matemáticas.

1.1. Alfabetos, cadenas y lenguajes

DEFINICIÓN 1.1. Un *alfabeto* es un conjunto Σ, que es finito y no vacío. A los elementos de Σ los llamaremos símbolos o caracteres.

DEFINICIÓN 1.2. Sea $n \in \mathbb{N}$. Una *cadena de longitud* n sobre Σ, es una n-tupla de elementos de Σ. A la cadena de longitud 0 la denotamos por ϵ. Al conjunto de todas las cadenas de largo n sobre Σ lo denotamos por Σ^n. Entonces definimos el conjunto de cadenas sobre un alfabeto Σ por:

$$\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$$

A una cadena de longitud n, $(a_1, a_2, ..., a_n)$, la esribiremos simplemente yuxtaponiendo sus caracteres, es decir, si c es dicha cadena, $c = a_1 a_2 ... a_n$.

DEFINICIÓN 1.3. Sean $a = a_1 a_2 ... a_n$ y $b = b_1 b_2 ... b_m$ dos cadenas. Definimos la *concatenación* de a y b por: $ab = a_1 a_2 ... a_n b_1 b_2 ... b_m$. Definimos también de forma recursiva $a^0 = \epsilon$, y $a^k = a a^{k-1}$

DEFINICIÓN 1.4. Un *lenguaje* $\mathcal{L}$ sobre Σ es un subconjunto de Σ^*.

DEFINICIÓN 1.5. Sean $\mathcal{L} \subseteq \Sigma^*$ y $\mathcal{L}' \subseteq \Sigma^*$ dos lenguajes sobre Σ. Definimos las siguientes operaciones:

- La *concatenación* de $\mathcal{L}$ y $\mathcal{L}'$:
$$\mathcal{L} \circ \mathcal{L}' = \{xy : x \in \mathcal{L},\, y \in \mathcal{L}'\}$$
- La *potencia* de $\mathcal{L}$:
$$\mathcal{L}^0 = \{\epsilon\}$$
$$\mathcal{L}^n = \mathcal{L}\mathcal{L}^{n-1}$$
- La *claúsura de Kleene* de $\mathcal{L}$:
$$\mathcal{L}^* = \bigcup_{n \in \mathbb{N}} \mathcal{L}^n$$
- El *complemento* de $\mathcal{L}$:
$$\mathcal{L}^c = \Sigma^* \backslash \mathcal{L}$$

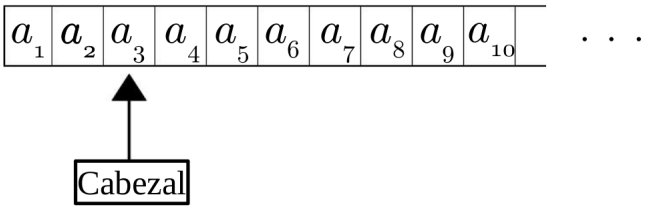
EJEMPLO 1.1. Como ejemplo de lenguaje más común, tenemos el lenguaje de las palabras en español; aquí el alfabeto Σ consta del alfabeto en castellano, y el lenguaje $\mathcal{E}$ consta de aquellas sucesiones de caracteres que tienen significado en español, así, *correr* es una palabra del lenguaje mientras que *aabbccdd* no lo es.

1.2. Máquinas de Turing

Una máquina de Turing es en cierta forma una computadora teórica, a pesar de que tiene sólo unos cuantos mecanismos primitivos, puede relizar cualquier computación que efectúe cualquier computadora real actual.

Informalmente, una máquina de Turing se puede visualizr como una cinta con borde por la izquierda pero infinita hacia la derecha, dividida en una infinidad de celdas y cada una conteniendo un caracter de un alfabeto, llamado alfabeto de la cinta. Además cuenta con un cabezal posicionado sobre una celda a la vez y que lee su caracter. La máquina de Turing cuenta con un conjunto finito de estados, y en cada instante se encuentra en uno y solo uno de ellos. Con estas herramientas, una máquina de Turing, dependiendo del estado en el que esté y del caracter que lee el cabezal, pasa a un nuevo estado, puede o no cambiar el caracter de la celda donde está colocado y finalmente desplazarse una celda a la izquierda o una celda a la derecha. Existen también dos estados especiales; un estado inicial donde comienza la computación, y un estado final, donde la máquina se detiene. Una máquina de Turing tambien cuenta con un alfabeto de entrada que es un subconjunto del alfabeto de cinta, empieza en el estado inicial con una cierta cadena sobre el alfabeto de entrada y define una función dependiendo de la transformación de dicha cadena hasta que la máquina se detiene.

FIGURA 1. Ejemplo gráfico de una máquina de Turing



Daremos ahora una definicion formal:[1]

DEFINICIÓN 1.6. Una *máquina de Turing* $\mathcal{M}$ es una 7-tupla

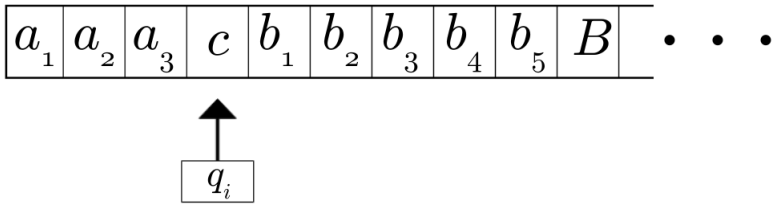
$$\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$$

donde:

- Q es el conjunto de estados de $\mathcal{M}$
- $q_0 \in Q$ es el estado inicial
- $q_f \in Q$ es el estado final
- Γ es el afabeto de la cinta
- $\Sigma \subset \Gamma$ es el alfabeto de entrada
- $B \in \Gamma \backslash \Sigma$ es un símbolo especial llamado «blanco»
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{I, D, N\}$ es una función parcial llamada función de transición, donde $\delta(q_1, s_1) = (q_2, s_2, m)$, indica que estando en el estado q_1 y leyendo el símbolo s_1, la máquina pasa al estado q_2, escribe el símbolo s_2 en lugar de s_1 y se mueve a la derecha si $m = D$, a la izquierda si $m = I$, o no se movió si $m = N$. Además $\delta(q_f, s)$ está indefinido para todo $s \in \Gamma$

[1]Esta es solo una de tantas definiciones de una máquina de Turing, en realidad es la definición de una máquina de Turing determinista. En otros textos se pueden encontrar definiciones con cinta infinita en ambas direcciones, máquinas de varias cintas y/o varios cabezales y tambien máquinas de Turing no deterministas, sin embargo, para nuestros fines, todas ellas son equivalentes. Una máquina de Turing no determinista es aquella donde δ es una relación que no necesariamente es una función, así δ puede tener varias opciones a elegir.

FIGURA 2. Visualización de la configuración (q_i, ucv) donde
$u = a_1a_2a_3$ y $v = b_1b_2b_3b_4b_5$



OBSERVACIÓN 1.1. Una diferencia esencial entre una máquina de Turing y una computadora real, es que una máquina de Turing tiene en cierta forma una memoria potencialmente infinita, es decir, aunque la cantidad de información que guarda en un instante dado es finita, esta cantidad no tiene una cota superior.

OBSERVACIÓN 1.2. En cierto modo, una máquina de Turing está definida solo en terminos de Q, Γ, Σ y δ, pues los demás terminos son constantes.

Vamos a hablar ahora de a qué nos referimos cuando hablamos de una máquina de Turing en un instante dado. En un momento específico, para que una máquina de Turing $\mathcal{M}$ lleve a cabo una computación, sólo se necesita conocer cierta información; el estado actual de $\mathcal{M}$, el contenido de la cinta y la posición en la que se encuentra el cabezal; para hacer esto, daremos explícitamente el estado de $\mathcal{M}$ y dividiremos el contenido de la cinta en tres: la cadena que está anterior al cabezal, el caracter que está bajo el cabezal, y la cadena que está a la derecha del cabezal, omitiendo en esta última la cadena que se encuentre a partir del primer B en la cinta para evitar cadenas infinitas. Formalizando esto tenemos:

DEFINICIÓN 1.7. Una *configuración* de una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$, es un elemento del conjunto de configuraciones posibles

$$C_{\mathcal{M}} = Q \times \Gamma^* \times \Gamma \times (\Gamma^* \backslash (\Gamma^* \circ \{B\}))$$

Regularmente escribiremos (q, uav) en lugar de (q, u, a, v).

Si $\mathcal{M}$ tiene una configuración de la forma (q_f, uav) diremos que la máquina entró en el estado de detención, o simplemente que esta detenida.

DEFINICIÓN 1.8. La relación *«llevar en un paso»* de una máquina de Turing $\mathcal{M}$, está definida como $\vdash_{\mathcal{M}} \subset C_{\mathcal{M}} \times C_{\mathcal{M}}$, donde

$$((q_1, u_1a_1v_1), (q_2, u_2a_2v_2)) \in \vdash_{\mathcal{M}} \iff \delta(q_1, a_1) = (q_2, x, m)$$

teniendo las siguientes opciones para x y m:

1. Si $m = I$, entonces $u_2a_2 = u_1$ y $v_2 = xv_1$
2. Si $m = D$, entonces $u_2 = u_1x$ y $a_2v_2 = v_1$
3. Si $m = N$, entonces $u_1 = u_2$, $v_1 = v_2$ y $a_2 = x$

Definimos tambien la *clausura reflexiva y transitiva* de $\vdash_{\mathcal{M}}$, como $\vdash^*_{\mathcal{M}}$, es decir,
$C \vdash^* C'$ si y solo si, existe una sucesión de configuraciones

$$\left(C = C_1, C_2, \ldots, C_{n-1}, C_n = C'\right)$$

donde para cada $1 \leq i < n$ se tiene que $C_i \vdash C_{i+1}$.

Se han omitido los subíndices, pues se sobreentiende que todas las configuraciones y relaciones son sobre una misma máquina de Turing dada, de ahora en adelante haremos esta omisión cuando no se preste a confusiones.

Nótese que esta sucesión puede tener un sólo elemento, de donde se obtiene la reflexividad.

A la relación $\vdash^*$ se le llamará «llevar en cero o más pasos»

Hasta ahora hemos definido todos los procesos que lleva a cabo una máquina de Turing, pero nos falta dar un punto de partida para estos procedimientos. Para poner en marcha una máquina de Turing, debemos alimentarla con una cadena c en su alfabeto de entrada Σ, esto lo haremos colocando un B en la celda que está más a la izquierda en la cinta, y escribiendo cada uno de los caracteres de c en las siguientes celdas, llenando la celdas restantes a la derecha de c, con símbolos B. La máquina comenzará en el estado inicial q_o, y el cabezal estara colocado en la celda más a la izquierda de la cinta.

Todo lo anterior nos dará una configuración inicial $(q_o, \epsilon Bc)$.

Daremos ahora algunas definiciones más.

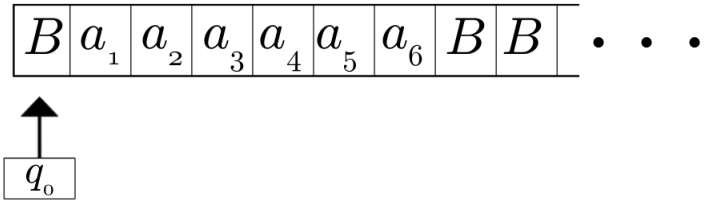
DEFINICIÓN 1.9. Sea $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ una máquina de Turing y sea $c \in \Sigma^*$, entonces decimos que $\mathcal{M}$ *computa* c *con salida* $d \in \Sigma^*$ si y solo si existe un configuración C tal que

$$(q_0, \epsilon Bc) \vdash^* C$$

donde $C = (q_f, \epsilon Bd)$.

También diremos que $\mathcal{M}$ simplemente se detiene sin especificar su salida si $C = (q_f, uav)$, con $u, v \in \Gamma^*$ y $a \in \Gamma$.

FIGURA 3. Comienzo de una computación de una máquina de Turing, alimentada con una cadena $c = a_1a_2a_3a_4a_5a_6$



Escribiremos $\mathcal{M}\downarrow(c) = d$ o simplemente $\mathcal{M}(c) = d$, para decir que $\mathcal{M}$ se detiene con entrada c y salida d, también pondremos $\mathcal{M}\downarrow(c)$ si $\mathcal{M}$ se detiene con entrada c, sin especificar su salida, y por último escribiremos $\mathcal{M}\uparrow(c)$, para decir que $\mathcal{M}$ no se detiene con entrada c.

Veamos ahora algunos ejemplos de ciertas máquinas de Turing muy útiles para fijar ideas.

EJEMPLO 1.2. La máquina borradora $\mathcal{B} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$, es una máquina que recibiendo cualquier cadena de entrada, se detiene después de haberla borrado de la cinta, es decir,

$$\forall w \in \Sigma^* (\mathcal{B}\downarrow(w) = \epsilon)$$

o equivalentemente,

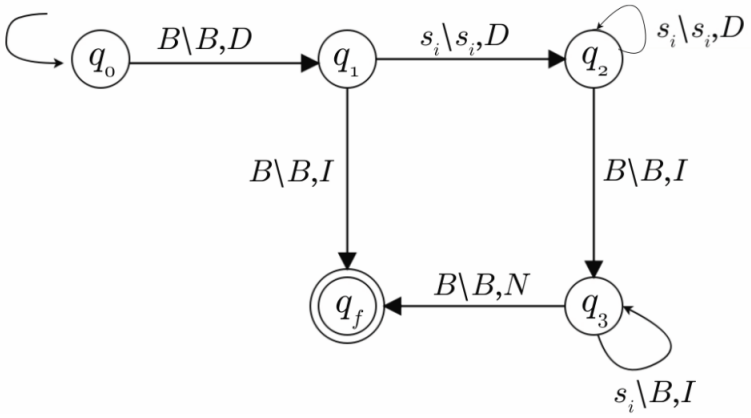
$$\forall w \in \Sigma^* ((q_0, \epsilon Bw) \vdash_{\mathcal{B}}^* (q_f, \epsilon B\epsilon))$$

Construyamos entonces dicha máquina. Primero que nada definamos las componentes de $\mathcal{B}$; $\mathcal{B}$ funciona sobre cualquier alfabeto, así que sean $\Sigma = \{s_{1,\dots}, s_n\}$ un alfabeto cualquiera, $\Gamma = \Sigma \cup \{B\}$, $Q = \{q_0, q_1, q_2, q_3, q_f\}$ y δ esta dada como sigue:

δ	s_i	B
q_0	Indefinida	(q_1, B, D)
q_1	(q_2, s_i, D)	(q_f, B, I)
q_2	(q_2, s_i, D)	(q_3, B, I)
q_3	(q_3, B, I)	(q_f, B, N)
q_f	Indefinida	Indefinida

Entonces lo anterior, define a la máquina de Turing borradora $\mathcal{B}$.

También podemos expresar esta máquina de Turing mediante un diagrama como el siguiente:



Donde cada flecha de q_i a q_j, etiquetada con una tripleta $a_1\backslash a_2, m$, indica que se pasó del estado q_i leyendo a_1, al estado q_j, escribiendo a_2 y moviendose como indica m; en pocas palabras $\delta(q_i, a_1) = (q_2, a_2, m)$.

EJEMPLO 1.3. La máquina copiadora $\mathcal{C} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ también es independiente del alfabeto de entrada que escojamos, y al ser alimentada con una cadena, escribe dicha cadena nuevamente inmediatamente después de la primera y después se detiene. Formalmente

$$\forall c \in \Sigma^* \left(\mathcal{C} \downarrow (c) = cc\right)$$

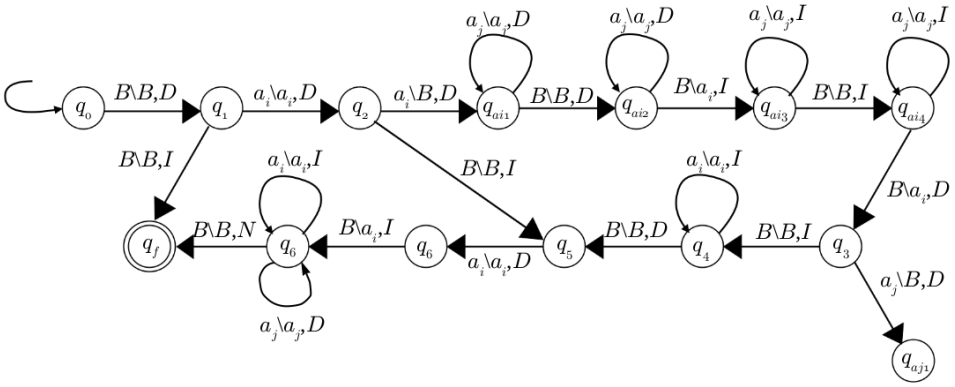
Entonces la máquina $\mathcal{C}$ con alfabeto de entrada $\Sigma = \{a_1, \ldots, a_n\}$ consta de un conjunto

$$Q = \{q_0, \ldots, q_6, q_{a_1}, q_{a_1}1, \ldots, q_{a_1}4, q_{a_2}, q_{a_2}1, \ldots, q_{a_2}4, \ldots, q_{a_n}, q_{a_n}1, \ldots, q_{a_n}4, q_f\}$$

de estados que depende del alfabeto de entrada, y la función de transición δ esta dada por la siguiente tabla:

δ	B	a_j
q_0	(q_1, B, D)	Indefinida
q_1	(q_f, B, I)	(q_2, a_j, D)
q_2	(q_5, B, I)	$\left(q_{a_j}1, B, D\right)$
q_3	(q_4, B, I)	$\left(q_{a_j}, B, D\right)$
q_4	(q_5, B, D)	(q_4, a_j, I)
q_5	Indefinida	$\left(q_{a_j}, a_j, D\right)$
q_6	(q_f, B, N)	(q_6, a_j, I)
q_{a_i}	(q_6, a_i, I)	(q_{a_i}, a_j, D)
$q_{a_i}1$	$(q_{a_i}2, B, D)$	$(q_{a_i}1, a_j, D)$
$q_{a_i}2$	$(q_{a_i}3, a_i, I)$	$(q_{a_i}2, a_j, D)$
$q_{a_i}3$	$(q_{a_i}4, B, I)$	$(q_{a_i}3, a_j, B)$
$q_{a_i}4$	(q_3, a_i, D)	$(q_{a_i}4, a_j, I)$
q_f	Indefinida	Indefinida

Esta máquina se puede visualizar con la siguiente figura:



En la máquina $\mathcal{C}$ se utilizan estados indexados con los caracteres del alfabeto para «memorizar» un cierto caracter, y ya que el alfabeto es finito, también los estados lo son. Esta técnica proveé a las máquinas de Turing de memoria en cierto modo y se puede utilizar esto para memorizar cualquier secuencia finita de caracteres, utilizando estados indexados con dicha secuencia.

Las máquinas de Turing también pueden combinarse o modificarse sin que esto presente un problema. Por ejemplo, podemos crear una máquina de Turing que dada una cadena la copie hasta dejarla escrita cuatro veces consecutivas en la cinta, simplemente cambiando el estado final de la máquina $\mathcal{C}$ y «pegándolo» con una copia de $\mathcal{C}$ nuevamente. También podemos hacer una máquina que primero duplique la cadena de entrada y luego la borre (aunque esto sería de una utilidad nula) simplemente cambiando el estado final de $\mathcal{C}$ por el estado inicial de $\mathcal{B}$. Regularmente usaremos máquinas de Turing ya definidas como hasta ahora lo están $\mathcal{B}$ y $\mathcal{C}$ pero con algunas modificaciones, ya sea agregando algunos blancos "B", pegándolas unas con otras o cambiando algunos estados.

Como un comentario final sobre las máquinas de Turing; si una máquina se trata de mover a la izquierda cuando está en la celda más a la izquierda de la cinta, la máquina no podría continuar con la computación pero tampoco entrará en el estado final, entonces se dice que la máquina se «cuelga». Por otro lado, también puede entrar en un ciclo infinito y nunca detenerse, y aunque a veces se puede identificar con facilidad cuando esto sucede, no siempre es trivial hacerlo.

Capítulo 2

Tipos de lenguajes

Una tarea común dentro de las matemáticas es la de clasificar los objetos de estudio de cierta área; las máquinas de Turing nos pueden ayudar para detectar ciertas propiedades dentro de lenguajes sobre un alfabeto e incluso sobre funciones y conjuntos muy especiales sobre los números naturales.

2.1. Lenguajes computables y aceptables

Hay una forma muy natural de relacionar funciones parciales con máquinas de Turing, esto se hace con la siguiente definición.

DEFINICIÓN 2.1. Una función parcial $f\,;\,\Sigma_1^* \longmapsto \Sigma_2^*$ se dice que es *Turing-computable* o simplemente *computable* si existe una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$, tal que $\Sigma_1 \cup \Sigma_2 \subseteq \Sigma$ y se tiene además que:

$$\forall c \in \Sigma_1^* \left(f(c) = \begin{cases} d & \iff \mathcal{M}(c) \downarrow = d \\ \text{indefinido} & \iff \mathcal{M}(c) \uparrow \end{cases} \right)$$

Podemos extender esta definición a una función parcial con varios parámetros $g\,;\,A_1^* \times \ldots \times A_n^* \longmapsto R^*$ tomando $A_1 \cup \ldots \cup A_n \cup R \subseteq \Sigma$, y si $c = (a_1, \ldots, a_n) \in A_1^* \times \ldots \times A_n^*$, entonces la representamos con la cadena $c = a_1 S a_2 S \ldots S a_{n-1} S a_n$, donde $S \in \Gamma \backslash \left(\left(\bigcup_{i=1}^{n} A_i \right) \cup R \cup \{B\} \right)$ funciona como un separador, y diferenciando la cadena c de la tupla c por el contexto.

Inversamente también podemos definir la función f a partir de la máquina $\mathcal{M}$; en este caso diremos que f es la función computada por $\mathcal{M}$. Denotaremos la función computada por una máquina de Turing $\mathcal{M}$ por $\Phi_{\mathcal{M}}$.

Ver cuáles funciones son Turing computables será uno de nuestros principales objetivos en este y los siguientes capítulos. A estas funciones las llamaremos simplemente funciones computables.

Veamos ahora algunas propiedades sobre lenguajes.

DEFINICIÓN 2.2. Un lenguaje $\mathcal{L}$ sobre un alfabeto Σ se dice que es *computable* si existe una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$, donde en Σ hay al menos dos elementos distintos s y n tales que:

$$\forall c \in \Sigma^* \left(\mathcal{M}(c) = \begin{cases} s & \iff c \in \mathcal{L} \\ n & \iff c \notin \mathcal{L} \end{cases} \right)$$

A los lenguajes computables se les conoce también por lenguajes recursivos o por lenguajes decidibles, pues desde cierto punto de vista, la máquina de Turing $\mathcal{M}$ puede decidir para cada cadena c, si c pertenece o no al lenguaje $\mathcal{L}$.

DEFINICIÓN 2.3. Un lenguaje $\mathcal{L}$ sobre un alfabeto Σ se dice que es *aceptable* si existe una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ tal que:

$$\forall c \in \Sigma^* \left(\mathcal{M} \downarrow (c) \iff c \in \mathcal{L} \right)$$

A este tipo de lenguajes también se les conoce como lenguajes semidecidibles o reconocibles.

2.2. Conjuntos computables y recursivamente enumerables en $\mathbb{N}$

Un tipo muy especial e importante de máquinas de Turing serán aquellas tales que su función computada esté definida de los naturales a los naturales, sin embargo, no podríamos tener en un alfabeto a todos los números naturales, así que codificaremos a los naturales con cadenas sobre un alfabeto específico.

Primero identificaremos a cada número natural n con su representación 1-aria, es decir, un número natural n será representado por una cadena de n+1 símbolos 1 consecutivos, y a esta cadena la denotaremos por $\overline{n}$. Tambien podemos representar una k-tupla de números naturales por una cadena que contenga a cada una de las representaciones 1-arias de los elementos de la tupla, separadas cada una por un símbolo 0, colocándolo concatenado entre las representaciones individuales; analogamente al caso de un número natural, $\overline{(n_1, \ldots, n_k)}$ denotará la codificación de la tupla $(n_1, \ldots, n_k)$, así entonces tendremos que una función $f : \mathbb{N}^r \longmapsto \mathbb{N}$ podría ser computada por una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ con alfabetos $\Sigma = \{0, 1\}$ y $\Gamma = \{0, 1, B\}$, donde las cadenas de entrada pertenecen al lenguaje

FIGURA 1. Codificación del número 3 en una MT

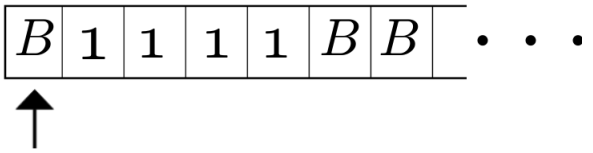
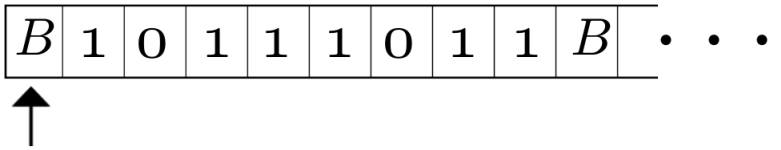


FIGURA 2. Codificación de la 3-tupla $(0, 2, 1)$ en una MT



$\mathcal{L} = 1\{1\}^* 01\{1\}^* 0 \ldots 01\{1\}^*$ tal que en $\mathcal{L}$ se tienen $r-1$ ceros que indican la separación entre dos números de la r-tupla.

Daremos entonces dos tipos de subconjuntos especiales basados en los lenguajes computables y recursivamente enumerables. Para ello definiremos antes un tipo especial de máquinas de Turing.

DEFINICIÓN 2.4. Una *máquina de Turing binaria* es una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ tal que $\Gamma = \{0, 1, B\}$ y $\Sigma = \{0, 1\}$

DEFINICIÓN 2.5. Un conjunto $A \subseteq \mathbb{N}$ se dice que es computable, si el lenguaje $\mathcal{L} = \{\overline{n} \in 1\{1\}^* : n \in A\}$ es 2-computable, es decir, computable por una máquina de Turing binaria. Analogamente se definen los subconjuntos aceptables de $\mathbb{N}$.

También podemos decir que una función $f : N^r \longmapsto \mathbb{N}$ es computable si lo es en el sentido de la definición 2.1. De esta forma, un subconjunto $A \subseteq \mathbb{N}$ es computable si, y solo si, su función caracteristica χ_A lo es.

Las máquinas de Turing pueden contener a la aritmética de Peano en el sentido de que las operaciones sucesor, suma y producto, vistas como funciones, son 2-computables. Antes que nada, notemos también que cada número natural puede ser escrito por una máquina de Turing a partir de una configuración que inicia con la cinta vacía (llena de símbolos B). De esta forma decimos que cada número natural es computable visto como una función sin parámetros.

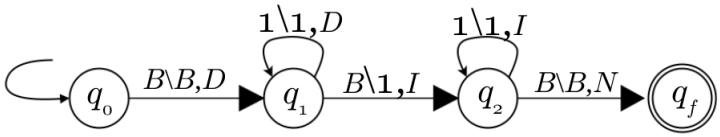
OBSERVACIÓN 2.1. Cualquier función constante es 2-computable.

Basta aplicar la máquina borradora y usar el hecho de que cada número natural es computable.

Observación 2.2. La operación sucesor es 2-computable.

Simplemente observemos que la máquina sucesor $\mathcal{Z}$ solo necesita recibir de entrada la codificación de un número natural y escribir un símbolo 1 más al final de la cadena. Al ser $\mathcal{Z} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ una máquina de Turing binaria, solo es necesario especificar su conjunto de estados y su función de transición. Entonces tenemos $Q = (q_0, q_1, q_2, q_f)$ y δ dada por:

δ	B	0	1
q_0	(q_1, B, D)	Indefinida	Indefinida
q_1	$(q_2, 1, I)$	Indefinida	$(q_1, 1, D)$
q_2	(q_f, B, N)	Indefinida	$(q_2, 1, I)$
q_f	Indefinida	Indefinida	Indefinida

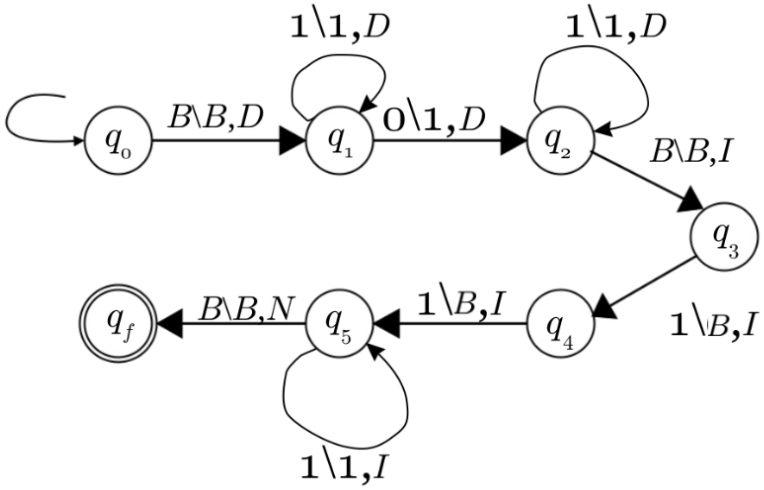


Observación 2.3. La operación suma es 2-computable.

Como la operación suma recibe dos parametros, recibira una cadena de la forma $1\{1\}^* 01\{1\}^*$, así, si recibe el numeral $\overline{(n,m)}$, cambiamos el símbolo 0 de enmedio por un símbolo 1 y entonces tendremos $(n+1)+(m+1)+1 = n+m+3$ símbolos 1, así que solo borramos dos símbolos 1 al final de la cadena y obtenemos el numeral de $n+m$.

Entonces la máquina binaria Suma $\mathcal{S} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$, está determinada por su conjunto de estados $Q = (q_0, q_1, q_2, q_3, q_4, q_5, q_f)$ y su función de transición δ dada por:

δ	B	0	1
q_0	(q_1, B, D)	Indefinida	Indefinida
q_1	Indefinida	$(q_2, 1, D)$	$(q_1, 1, D)$
q_2	(q_3, B, I)	Indefinida	$(q_2, 1, D)$
q_3	Indefinida	Indefinida	$(q_4, 0, I)$
q_4	Indefinida	Indefinida	$(q_5, 0, I)$
q_5	(q_f, B, N)	Indefinida	$(q_5, 1, I)$
q_f	Indefinida	Indefinida	Indefinida



La operación suma puede extenderse a una suma de varios parametros e incluso puede construirse una máquina de Turing que sume todas las entradas que se le den independientemente de cuantas sean (obviamente solo pueden ser un número finito de sumandos). También utilizando la máquina borradora y con un procedimiento similar se puede computar la resta.

Observación 2.4. La operación producto es 2-computable.

En este caso la máquina de Turing binaria $\mathcal{P} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ que computa el producto es mucho más laboriosa para dar una representación explícita, pero conceptualmente es igual de sencilla; solo tenemos que hacer uso de la máquina copiadora y modificarla para que al recibir una entrada $\overline{(n,m)}$, copie el numeral de n, m veces pero separadas con un símbolo 0, es decir, transforma $\overline{(n,m)}$ en la m-tupla $\overline{(n,\ldots,n)}$, y después de esto, solo aplicamos la máquina $\mathcal{S}$ y nos dará como salida $\overline{nm}$.

Análogo a la creación de la máquina $\mathcal{P}$ a partir de la máquina S, podemos crear la máquina exponente $\mathcal{E}$ que recibiendo de entrada un par $\overline{(a,x)}$ devuelva como salida a^x.

Por último tambien podemos hacer una máquina de Turing que compute el algoritmo de la división, esto es, una máquina $\mathcal{Q}$ que reciba de entrada $\overline{(n,m)}$ y regrese de resultado un numeral de la forma $\overline{(q,r)}$ donde q es el cociente y r es el residuo al dividir n entre m. Esto se hace alimentando la máquina multiplicadora con n y el numeral del 1, y checando que la resta m con el resultado de dicha multiplicación sea menor que n, de lo contrario borramos el resultado, aplicamos la

operación sucesor al numeral del 1 y repetimos este proceso hasta obtener un número menor que n en el resultado de la resta, así q será el número hasta el cual lleguemos aplicando la operación sucesor y r será el resultado de la resta. Solo debemos tener cuidado si $m < n$, en donde solo pondremos $q = 0$ y $r = m$.

OBSERVACIÓN 2.5. Ahora podemos ver una función $f : \mathbb{N}^2 \longmapsto \mathbb{N}$ como una función de $\mathbb{N}$ en $\mathbb{N}$, usando una biyección $b : \mathbb{N}^2 \longmapsto \mathbb{N}$, como por ejemplo $(a, b) \longmapsto 2^a (2b + 1) - 1$, que en vista de las anteriores máquinas, es computable. Repitiendo esta biyección podemos ver en general cualquier función $g : \mathbb{N}^r \longmapsto \mathbb{N}$, como una función de $\mathbb{N}$ en $\mathbb{N}$. Diremos tambien que una función $h : \mathbb{N}^r \longmapsto \mathbb{N}^s$ es computable si y solo si $h = (h_1, \ldots, h_s)$ y para cada i, h_i es computable.

De esta forma para estudiar las funciones entre $\mathbb{N}^r$ y $\mathbb{N}^s$ con $r, s \in \mathbb{N}$, nos bastará con estudiar las funciones de $\mathbb{N}$ en $\mathbb{N}$.

DEFINICIÓN 2.6. Un conjunto $X \subseteq A$ se dice que es *efectivamente enumerable* si $X = \emptyset$ o si existe una función total computable $f : \mathbb{N} \longmapsto X$ que es suprayectiva. En este caso tenemos que la sucesión: $f(0), f(1), f(2), \ldots$ enumera a los elementos de X. En particular si $A = \mathbb{N}$, diremos que X es *recursivamente enumerable (RE).* Más aún, podemos extender esta definición a subconjuntos de $\mathbb{N}^r$ con $r \in \mathbb{N}$, gracias a la función b de la observación anterior.

Notemos que un subconjunto A de $\mathbb{N}$ es aceptable si es el dominio de una función computada por una máquina de Turing, de la misma manera si A es RE entonces es la imagen de una función total computable y suprayectiva $f : \mathbb{N} \longmapsto A$. Veamos pues que estos dos conceptos son equivalentes.

TEOREMA 2.1. *Un subconjunto A de $\mathbb{N}$ es aceptable si, y solo si es RE.*

DEMOSTRACIÓN. ($\Longrightarrow$) Si A es vacío por definición tenemos que A es RE. Supongamos ahora que $A \neq \emptyset$ es aceptable, que $\mathcal{M}$ es la máquina de Turing que verifica esto, y sea $a \in A$; entonces definimos $h : \mathbb{N}^2 \longmapsto \mathbb{N}$ dada por:

$$h(i, n) = \begin{cases} n & \text{si } \mathcal{M} \text{ se detiene con entrada } n \text{ en a lo mas } i \text{ pasos} \\ a & \text{de otra forma} \end{cases}$$

con esto h es claramente computable y si tomamos la biyeccion computable b dada anteriormente, tenemos que $f = h \circ b^{-1} : \mathbb{N} \longmapsto A$ es una

función total computable. Además $x \in \mathrm{Im}(f)$ si, y solo si, $x = a$ o existe $i \in \mathbb{N}$ tal que $\mathcal{M}$ se detiene con entrada x en a lo más i pasos, y como en ambos casos $x \in A$, se sigue que A es RE.

$(\Longleftarrow)$ Supongamos que $\emptyset \neq A \subseteq \mathbb{N}$ es RE, entonces construyendo una máquina de Turing $\mathcal{M}$ tal que con entrada n, primero escriba en la cinta un cero, es decir, en la cinta aparezca la cadena $B\overline{n}0\overline{0}B\ldots = B\overline{n}01B\ldots$, después aplicando f la función que testifica que A es RE, cambiamos la cinta por $B\overline{n}0\overline{0}0\overline{f(0)}B\ldots$ y despues comprobamos si $n = f(0)$, si esto pasa detenemos $\mathcal{M}$, de lo contrario aplicamos la función sucesor (la cual ya vimos que es computable) al $\overline{0}$ y cambiamos $\overline{f(0)}$ por $\overline{f(1)}$, para llegar a un contenido de cinta $B\overline{n}0\overline{1}0\overline{f(1)}B\ldots$ y repetimos el proceso. Como A es RE, si $n \in A$, para algún $k \in \mathbb{N}$ se tendrá que $n = f(k)$, y en ese momento $\mathcal{M}$ se detendrá, así para todo $a \in A$, $\mathcal{M}(a)\downarrow$. Por otro lado si $n \notin A$, este procedimiento seguirá indefinidamente, y así $\mathcal{M}(a)\uparrow$. Por tanto A es aceptable.
Si $A = \emptyset$, simplemente damos una máquina que no se detenga nunca; como por ejemplo, la máquina de Turing binaria con $Q = \{q_0, q_f\}$ y $\delta(q_0, i) = (q_0, i, N)$ para $i \in \{0, 1, B\}$. □

En varias ocasiones hemos utilizado las relaciones $=$ y $<$ sobre los números naturales dentro de las máquinas de Turing sin tener la certeza de que esto sea posible, sin embargo probaremos que estas relaciones son también computables.

Definición 2.7. Una reación $R \subseteq \mathbb{N}^2$ se dice que es computable si el conjunto $\{(a, b) \in \mathbb{N}^2 : aRb\}$ es computable.

Proposición 2.1. *Las relaciones $=$ y $<$ sobre los números naturales son computables.*

Demostración. Daremos una máquina de Turing que con entrada (a, b) devuelva 1 si $a = b$ y 0 de otro modo.
La idea es la siguiente: al tener sobre la cinta la cadena $\overline{(a,b)} = B1\cdots101\cdots1BB\cdots$, donde la primer cadena tiene $a+1$ símbolos 1 y la segunda $b+1$, primero cambiamos el símbolo 1 de más a la izquierda $\overline{a}$ por un símbolo B, después hacemos lo mismo con el símbolo más a la derecha de $\overline{b}$, volvemos a hacer esto con el símbolo más a la izquierda de los símbolos 1 de lo que queda $\overline{a}$, repetimos esto con el símbolo 1 más a la derecha de lo que queda de $\overline{b}$ y continuamos con este proceso recursivamente. Al hacer esto, si $a = b$, entonces el último símbolo 1 sobre la cinta será el que está a la derecha del número 0, entonces si en algún momento cambiamos este número con por un símbolo B, basta comprobar que el símbolo a la izquierda del número 0 es un símbolo B, en cuyo caso borramos el número 0 y escribimos el

numeral del 1 en la cinta.
Si en algún momento no se puede seguir con el procedimiento recursivo de «tachar» los símbolos 1, o llegamos a «tachar» el 1 que está a la derecha del 0 pero aún hay un símbolo 1 a su izquierda, entonces damos la instrucción de borrar toda la cinta, escribir el numeral del 0 y detener la máquina.
Formalizando esto tenemos que la máquina $\mathcal{M}$ con conjunto de estados $Q\,\{q_0, q_f\}$ y la función δ dada por:

δ	0	1	B
q_0	indefinida	indefinida	$(q_1, 1, D)$
q_1	$(q_9, 0, D)$	(q_2, B, D)	indefinida
q_2	$(q_3, 0, D)$	$(q_2, 1, D)$	indefinida
q_3	indefinida	$(q_4, 1, D)$	(q_{10}, B, I)
q_4	indefinida	$(q_4, 1, D)$	(q_5, B, I)
q_5	indefinida	(q_6, B, I)	indefinida
q_6	(q_{13}, B, I)	$(q_7, 1, I)$	indefinida
q_7	$(q_8, 0, I)$	$(q_7, 1, I)$	indefinida
q_8	indefinida	indefinida	(q_1, B, D)
q_9	indefinida	(q_9, B, D)	(q_{10}, B, I)
q_{10}	(q_{11}, B, I)	indefinida	(q_{10}, B, I)
q_{11}	indefinida	(q_{12}, B, D)	(q_{11}, B, I)
q_{12}	indefinida	indefinida	$(q_f, 1, I)$
q_{13}	indefinida	(q_{14}, B, I)	(q_{15}, B, I)
q_{14}	indefinida	$(q_{14}, B, 1)$	(q_{12}, B, I)
q_{15}	indefinida	(q_{16}, B, D)	(q_{15}, B, I)
q_{16}	indefinida	indefinida	$(q_{17}, 1, D)$
q_{17}	indefinida	indefinida	$(q_{18}, 1, I)$
q_{18}	indefinida	$(q_{18}, 1, I)$	(q_f, B, N)
q_f	indefinida	indefinida	indefinida

computa la relación de igualdad (Se puede convencer de esto haciendo un diagrama como en las anteriores máquinas de Turing, o probando con algunas entradas).

El mismo razonamiento se puede utilizar para computar la relación $<$, simplemente viendo por «cuál lado se llega primero» al símbolo 0 sobre la cinta. □

Como consecuencia de este teorema tenemos entonces varias formas de ver un conjunto RE.

COROLARIO 2.1. *Sea $A \subseteq \mathbb{N}$, entonces las siguientes afirmaciones son equivalentes:*

1. A es aceptable.
2. A es RE.
3. A es el dominio de una función $f : \mathbb{N} \longmapsto \mathbb{N}$ computable.
4. A es la imagen de una función $f : \mathbb{N} \longmapsto \mathbb{N}$ computable.

DEMOSTRACIÓN. $1 \iff 3$ y $2 \iff 4$ son exactamente las deficiniones y $1 \iff 2$ es precisamente el teorema 2.1 □

Veamos entonces algunos ejemplos de conjuntos computables:

EJEMPLO 2.1. Los conjuntos $\mathbb{N}$ y $\emptyset$ son computables.

EJEMPLO 2.2. El cojunto de los números pares es computable.

Basta simplemente con crear una máquina de Turing $\mathcal{T}_2$ que con entrada $\overline{n}$ utilice $\mathcal{Q}$ con entrada $\overline{(n,2)}$ y por último compruebe si $q = 0$. Este mismo procedimiento puede hacerse para probar que $\forall a \in \mathbb{N}$ el conjunto de todos los multiplos de a es computable por una máquina similar $\mathcal{T}_a$.

EJEMPLO 2.3. El conjunto de los números primos es computable.

Sabiendo que para todo $a \in \mathbb{N}$ el lenguajes de los multiplos de a es computable, podemos crear una máquina que con entrada $\overline{n}$ corra sucesivamente la máquina $\mathcal{Q}$ con entradas $\overline{(n,2)}, \overline{(n,3)}, \ldots, \overline{(n,n-1)}$ y vaya comprobando si en algun momento el residuo r_i al correr $\mathcal{Q}$ con entrada $\overline{(n,i)}$ fué igual a 0.

Para terminar esta sección, veamos algunas propiedades inmediatas de los lenguajes computables y aceptables.

PROPOSICIÓN 2.2. *Sea $\mathcal{L}$ un lenguaje sobre un alfabeto Σ (ó $\mathcal{L}$ subconjunto de $\mathbb{N}$), entonces se tiene lo siguiente:*

1. $\mathcal{L}$ computable $\implies \mathcal{L}^c$ computable
2. $\mathcal{L}$ computable $\implies \mathcal{L}$ aceptable
3. $\mathcal{L}$ y $\mathcal{L}^c$ aceptables $\iff \mathcal{L}$ computable

DEMOSTRACIÓN. Sea $\mathcal{L}$ computable, entonces existe una máquina de Turing $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ y dos elementos $s, n \in \Sigma$ tal que

$$\forall c \in \Sigma^* \left(\mathcal{M}(c) = \begin{cases} s & \iff c \in \mathcal{L} \\ n & \iff c \notin \mathcal{L} \end{cases} \right).$$

Para ver (1) cambiamos el estado final de $\mathcal{M}$ por un estado interno y simplemente agregamos estados para cambiar s por n y viceversa, así esta máquina modificada $\mathcal{M}'$ a partir de $\mathcal{M}$ computa $\mathcal{L}^c$.

Para ver (2) simplemente cambiamos también el estado final pero en este caso agregamos un estado que compruebe si la salida es s, y en tal caso pasamos al estado final, de lo contrario colgamos la máquina o la ponemos en un ciclo infinito, ya sea moviendo el cabezal siempre hacia la izquierda o dando una asignación a δ como sigue: $\delta(q_i, a_i) = (q_i, a_i, N)$, donde a_i puede ser cualquier símbolo incluyendo B.

Para (3) ya tenemos la demostración de la vuelta ($\Longleftarrow$) utilizando los puntos 1 y 2. Para la otra parte supongamos que $\mathcal{M}_1$ es la máquina que acepta $\mathcal{L}$ y que $\mathcal{M}_2$ es la máquina que acepta $\mathcal{L}^c$, entonces podemos construir una máquina $\mathcal{M}$ que con entrada c y comenzando desde $i = 1$ corra la máquina $\mathcal{M}_1$ y pruebe si se detiene en i pasos, si lo hace entonces devolvemos s como salida, de lo contrario, pasamos a probar si $\mathcal{M}_2$ se detiene en i pasos, y si lo hace devolvemos n como salida, si no lo hace entonces repetimos el proceso con $i + 1$ y como $\mathcal{L}$ y $\mathcal{L}^c$ son aceptables, para alguna $i \in \mathbb{N}$, $\mathcal{M}_1$ o $\mathcal{M}_2$ se detendrá. De este modo $\mathcal{M}$ decide $\mathcal{L}$. □

Con la anterior proposición podemos encontrar otros conjuntos computables fácilmente, como por ejemplo los números impares, por ser el complemento de los pares; y en general $\forall l, n \in \mathbb{N}$ el conjunto $\mathcal{A} = \{m \in \mathbb{N} : m \equiv n \pmod{l}\}$ es computable.

Por otro lado, preguntarse si un lenguaje aceptable es computable, es una cuestión más interesante, pues no siempre es así; más adelante veremos algunos de estos lenguajes, y de hecho, usaremos conjuntos RE que no son computables para desarrollar la mayor parte de la teoría.

2.3. Aritmetización

Los alfabetos al ser finitos (o incluso si fueran numerables) nos dan la ventaja de que el número de cadenas que pueden construirse a partir de un alfabeto dado no será demasiado grande, de hecho, será numerable, y así nosotros podemos ordenar las cadenas para trabajar más facilmente. Esto también nos servirá para centrarnos solo en el estudio de las máquinas de Turing binarias y los conjuntos computables y recursivamente enumerables, pues todo lenguaje sobre cualquier alfabeto y toda máquina de Turing pueden reducirse a subconjuntos de $\mathbb{N}$ y a máquinas de Turing binarias respectivamente.

LEMA 2.1. *Sea Γ un alfabeto; entonces el conjunto de cadenas Γ^* es numerable.*

DEMOSTRACIÓN. Por definición tenemos que $\Gamma^* = \bigcup_{n \in \mathbb{N}} \Gamma^n$, donde la unión es ajena, asi que si $|\Gamma| = m$ se tiene que $|\Gamma^n| = |\Gamma|^n = m^n$, así Γ

es la unión numerable y ajena de conjuntos finitos, de donde se sigue que

$$|\Gamma^*| = |\bigcup_{n\in\mathbb{N}} \Gamma^n| = \aleph_0.$$

□

COROLARIO 2.2. *Todo lenguaje $\mathcal{L}$ es a lo más numerable.*

DEMOSTRACIÓN. Supongamos que $\mathcal{L}$ es un lenguaje sobre un alfabeto Σ, entonces como $\mathcal{L} \subseteq \Sigma^*$, se tiene que $|\mathcal{L}| \leq |\Sigma^*| = \aleph_0$ □

Ahora que ya tenemos que el conjunto de cadenas sobre un afabeto Σ es numerable podemos proceder a ordenar todas las cadenas posibles. Esto podemos hacerlo ordenando por longitud, y a las cadenas de la misma longitud las ordenamos lexicográficamente, así si tenemos que $\Sigma = \{a_1, \ldots, a_n\}$ tendremos las cadenas ordenadas de la siguiente manera:

$$(\epsilon\,,\, a_1\,,\, \ldots\,,\, a_n\,,\, a_1a_1\,,\, a_1a_2\,,\, \ldots\,,\, a_1a_n\,,\, a_2a_1\,,\, \ldots)$$

De esta forma podemos pensar $\Sigma^* = \{w_i\}_{i\in\mathbb{N}}$ y si $\mathcal{L}$ es un lenguaje sobre Σ,
entonces $\mathcal{L} = \{w_i\}_{i\in A}$, con $A \subseteq \mathbb{N}$, y así el problema de ver si una cadena c está en $\mathcal{L}$, se reduce a saber si i está en A, un problema sobre los naturales.
Veamos ahora que la función que transforma una cadena w_i en $\bar{i}$ es computable.

LEMA 2.2. *La función f que dada una cadena c devuelve $|c|$, es decir, calcula el largo de una cadena, es computable.*

DEMOSTRACIÓN. Construiremos una máquina $\mathcal{G}$ que compute dicha función; claramente el alfabeto de $\mathcal{G}$ debe contener el símbolo 1 para devolver $\overline{|c|}$. Simplemente $\mathcal{G}$ irá leyendo cada símbolo de la cadena y sustituyéndolo por un símbolo 1 al igual que el primer blanco inmediatamente después de la cadena. Así, $\mathcal{G}$ estará dada por la siguiente función de transición δ:

δ	s_i	B
q_0	Indefinida	(q_1, B, D)
q_1	$(q_2, 1, B)$	$(q_f, 1, I)$
q_2	$(q_2, 1, D)$	$(q_3, 1, I)$
q_3	$(q_3, 1, I)$	(q_f, B, N)
q_f	Indefinida	Indefinida

para todo $s_i \neq B$. □

DEFINICIÓN 2.8. Sea $\Sigma = \{s_1, \ldots, s_k\}$ indexado con los primeros k números naturales exceptuando al 0. Definimos la función $h : \Sigma \longmapsto \mathbb{N}$ que dado un caracter devuelve su indice por:

$$h(c) = k \iff c = s_k$$

Claramente h es computable pues es una función finita.

PROPOSICIÓN 2.3. *La función* $C : \Sigma^* \longmapsto \mathbb{N}$ *tal que*

$$\forall w_i \in \Sigma^* \left(f(w_i) = \bar{i} \right)$$

es computable.

DEMOSTRACIÓN. Esta proposición se puede demostrar de manera análoga a la interpretación de la expresión decimal de un número, es decir, suma de potencias de 10 por un elemento del conjunto $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \}$. Sea $\Sigma = \{s_1, \ldots, s_k\}$ y sea $w \in \Sigma^*$, entonces aplicamos $\mathcal{G}$ del lema anterior, y así $w = c_1 \ldots c_{\mathcal{G}(w)}$, de donde

$$\bar{i} = \sum_{j=1}^{\mathcal{G}(w)} k^{\mathcal{G}(w)-j} h(c_j)$$

en donde $h(c_j)$ juega el papel del elemento de D y $n^{\mathcal{G}(w)-j}$ es el equivalente a la expresión 10^r en expansión decimal. Como todas las operaciones involucradas son computables se tiene que C es computable también. □

OBSERVACIÓN 2.6. No es dificil ver que la función C es biyectiva gracias a la forma de expresar $\bar{i}$ en la demostración anterior y a que $|\Sigma^*| = \aleph_0$. Entonces podemos hacer lo mismo con la función C^{-1}.

AFIRMACIÓN 2.1. La función $f : \mathbb{N} \longmapsto \mathbb{N}$ dada por $f(n) = \min \{r : n < k^r\}$ es computable.

Esto es fácil de ver, pues para empezar, el conjunto $\{r : n < k^r\}$ es no vacío, entonces podemos decidir computablemente si un número i cumple que $n < k^i$, si lo hace, devolvemos i como resultado, de lo contrario cambiamos i por su sucesor y repetimos el proceso. Como el conjunto es no vacío este proceso terminará en una cantidad finita de pasos. Además, si comenzamos esto con $i = 0$, el resultado será $l = \min \{r : n < k^r\}$.

PROPOSICIÓN 2.4. *La función* C^{-1} *es computable*

DEMOSTRACIÓN. Sean $n \in \mathbb{N}, l = \min\{r : n < k^r\}$ y $\Sigma = \{s_{1,\ldots}, s_k\}$, entonces definimos $f : \mathbb{N} \longmapsto \Sigma^*$ dada por $f(n) := w$ con $|w| = l$, y tal que si $w = c_1 \ldots c_l$, entonces:

$$c_l = s_j \iff n \equiv h(s_j) \ (mod\, k)$$

y definimos recursivamente los demas caracteres por

$$c_{l-i} = s_r \iff \frac{n - \sum_{m=0}^{i-1} h(c_{l-m}) \cdot k^m}{k^i} \equiv r \ (mod\, k)$$

De esta forma claramente f es computable y también es fácil (aunque laborioso) ver que $C^{-1} = f$. □

Las dos proposiciones anteriores nos dan la libertad de trabajar sólo con máquinas de Turing que tengan como lenguajes de entrada a los numeros naturales (en su expresión unaria), sin embargo, esto no es suficiente para que nos baste trabajar con las máquinas de Turing binarias, pues deberíamos tener que el alfabeto de cinta fuera $\{0, 1, B\}$, para eso necesitamos el siguiente teorema.

TEOREMA 2.2. *Sea $\mathcal{M}$ una máquina de Turing cuyo alfabeto de entrada Σ contiene al menos dos elementos, entonces existe una máquina de Turing $\mathcal{N}$ tal que para todo $c \in \Sigma^*$, $\mathcal{M}(c) = \mathcal{N}(c)$ y $\mathcal{N}$ tiene alfabeto de cinta $\Gamma = \Sigma \cup \{B\}$.*

DEMOSTRACIÓN. Gracias a las proposiciones 2.3 y 2.4 podemos suponer sin perdida de generalidad $\Sigma = \{0, 1\}$. Sea entonces $\Gamma_{\mathcal{M}} = \{0, 1, B, c_1, \ldots, c_k\}$ el alfabeto de cinta de $\mathcal{M}$ y sea $n \in \mathbb{N}$ tal que $2^n > k+2$. Entonces definimos una función $cod : \Gamma \longmapsto \{0, 1, B\}^n$ dada por:

$$cod(x) = \begin{cases} 1^n & \iff x = 1 \\ 1^{n-1}0 & \iff x = 0 \\ B^n & \iff x = B \\ \bar{i}^2 & \iff x = c_i \end{cases}$$

donde $\bar{i}^2$ representa i en su forma binaria. A partir de *cod* construimos otra función $Cod : \Gamma^* \longmapsto \{0, 1, B\}^*$ dada por:

$$Cod(w) = \begin{cases} cod(w) & \iff w \in \Gamma \\ cod(x)\, Cod(w') & \iff w = xw' \wedge x \in \Gamma \wedge w' \in \Gamma^* \setminus \{\epsilon\} \end{cases}$$

De esta forma si nosotros comenzamos una computación sobre la cinta de $\mathcal{M}$ con una entrada de la forma $Bc_{i_1} \ldots c_{i_r}B$, al tener esta

misma entrada en la cinta de $\mathcal{N}$ podemos transformarla en una entrada de la forma $cod(B)\, cod(c_{i_1}) \ldots cod(c_{i_r})\, cod(B)$.

La demostración de que las funciones Cod y Cod^{-1} son computables es en escencia lo que hicimos en las proposiciones 2.3 y 2.4 para el caso binario, así que daremos esto por hecho. Ahora basta con poder simular el comportamiento de la máquina $\mathcal{M}$ dentro de $\mathcal{N}$ para que si $\mathcal{M} \downarrow (c) = d$ entonces con entrada c, $\mathcal{N}$ aplica Cod y después transforma $Cod(c)$ en $Cod(d)$, para finalmente aplicar Cod^{-1} y tener $\mathcal{N} \downarrow (c) = d$ y análogamente si $\mathcal{M} \uparrow (c)$ tendremos que $\mathcal{N} \uparrow (c)$; en pocas palabras $\mathcal{N}$ tratará a la cadena $Cod(w)$ como $\mathcal{M}$ trataría a w.

Para empezar hagamos que el conjunto de estados $Q_{\mathcal{N}}$ de $\mathcal{N}$ contenga al conjunto de estados $Q_{\mathcal{M}}$ de $\mathcal{M}$ y simulemos cada acción de $\mathcal{M}$; para esto solo necesitamos simular la lectura y escritura de un caracter, y los movimientos a la derecha o izquierda del cabezal de $\mathcal{M}$ dentro de $\mathcal{N}$. Diremos que $\mathcal{N}$ esta en una configuración normal si $\mathcal{N}$ se encuentra en un estado $q_i \in Q_{\mathcal{M}}$ y si cumple que si el cabezal se encuentra sobre la cadena $cod(s)$ entonces esta sobre el caracter más a la izquierda de $cod(s)$. Es claro que si alimentamos a $\mathcal{N}$ con una cadena c, aplicamos $Cod(Bc)$, movemos el cabezal hasta el símbolo B más a la izquierda en la cinta y llegamos al estado $q_0 \in Q_{\mathcal{M}}$ entonces estamos en una configuración normal que simula la configuración inicial de $\mathcal{M}$ al recibir c como entrada.

Entonces ahora simularemos el comportamiento de $\delta_{\mathcal{M}}$ dentro de $\mathcal{N}$. Supongamos que estamos en una configuración normal de $\mathcal{N}$, en un estado q_i y sobre la cadena $cod(s)$, entonces primero debemos interpretar el caracter s para simular $\delta_{\mathcal{M}}(q_i, s)$. Para hacer esto supongamos que $s = a_1 \ldots a_n$ con $a_i \in \{0, 1, B\}$, y utilicemos una sucesión de estados para memorizar cada caracter hasta llegar a un estado $q_{i,s}$, es decir, $\delta_{\mathcal{N}}(q_i, a_1) = (q_{i,a_1}, a_1, D)$ después de esto, $\delta_{\mathcal{N}}(q_{i,a_1}, a_2) = (q_{i,a_1a_2}, a_2, D)$, y en general para $k < n$; $\delta_{\mathcal{N}}\left(q_{i,a_1 \ldots a_{k-1}}, a_k\right) = (q_{i,a_1 \ldots a_k}, a_k, D)$. Para $k = n$ hacemos lo mismo pero sin realizar el movimiento a la derecha, es decir $\delta_{\mathcal{N}}\left(q_{i,a_1 \ldots a_{n-1}}, a_n\right) = (q_{i,s} = q_{i,a_1 \ldots a_n}, a_n, N)$. De esta forma si estamos en el estado $q_{i,s}$ dentro de $\mathcal{N}$ simularemos el comportamiento de $\delta_{\mathcal{M}}(q_i, s)$ independientemente del caracter que estemos leyendo.

Ahora supongamos que $\delta_{\mathcal{M}}(q_i, s) = (q_j, r, m)$; lo siguiente será cambiar $cod(s)$ por $cod(r)$, pero seguiremos el mismo procedimiento que el anterior, y suponiendo que $cod(r) = b_1 \ldots b_n$, entonces para toda $a \in \{0, 1, B\}$, hacemos primero $\delta_{\mathcal{N}}(q_{i,s}, a) = (q_{i,s,0}, a, N)$ y para $0 \leq j < n - 1$, ponemos $\delta_{\mathcal{N}}(q_{i,s,j}, a) = (q_{i,s,j+1}, b_{n-j}, I)$ para terminar con $\delta_{\mathcal{N}}(q_{i,s,n-1}, a) = (q_{i,s,m}, b_1, N)$.

De esta forma ya habremos escrito $cod\,(r)$ y si $m = N$, entonces simplemente hacemos $\delta_{\mathcal{N}}\,(q_{i,s,n}, a) = (q_j, a, N)$ y habremos terminado de simular la transición $\delta_{\mathcal{M}}\,(q_i, s) = (q_j, r, N)$ y acabaremos de nuevo en una configuración normal. Si $m = I$ entonces simplemente nos movemos n pasos a la izquierda sobre la cinta con la ayuda de n estados auxiliares y análogamente para $m = D$.

Si por otro lado nos encontramos en algún momento con una transición de tal forma que $\delta_{\mathcal{M}}\,(q_i, s)$ este indefinido; entonces hacemos que $\delta_{\mathcal{N}}\,(q_{i,s}, a)$ quede también indefinido para toda $a \in \{0, 1, B\}$. Así para todo $c \in \Sigma^*$ se tiene que $\mathcal{M}\,(c) = \mathcal{N}\,(c)$ pues en escencia tienen los mismos estados y la misma función de transición, solo basta aplicar Cod a la entrada de $\mathcal{N}$ al principio de la computación, y Cod^{-1} a la cadena sobre la cinta cuando se llegue al estado $q_f \in Q_{\mathcal{M}}$, además como $\Gamma_{\mathcal{N}} = \{0, 1, B\}$, se tiene demostrado el teorema. □

Con todo lo dicho a lo largo del capítulo llegamos entonces al resultado deseado.

COROLARIO 2.3. *Sea* $\mathcal{L} = \{w_i\}_{i \in A}$, *entonces*

$$\mathcal{L} \text{ es computable (aceptable)} \iff A \text{ es computable (RE)}$$

DEMOSTRACIÓN. ($\Longrightarrow$). Supongamos que $\mathcal{L}$ es computado por una máquina $\mathcal{M}$; entonces para cada $n \in \mathbb{N}$, aplicamos C^{-1} y despues $\mathcal{M}$, de esta forma tenemos que $\mathcal{N} = \mathcal{M} \circ C^{-1}$ computa A.

Analogamente hacemos ($\Longleftarrow$) suponiendo que A es computable por una máquina de Turing $\mathcal{M}$ y tomando $\mathcal{N} = \mathcal{M} \circ C$ como la máquina que computa $\mathcal{L}$.

Prácticamente la misma prueba vale para los conjuntos y lenguajes aceptables. □

A partir de ahora entonces podemos tratar solo con máquinas de Turing binarias y conjuntos computables y recursivamente enumerables, así que de aqui en adelante cuando hablemos de máquinas de Turing nos referiremos a máquinas de Turing binarias a menos que se diga lo contrario. Además, también estamos en libertad de agregar cualquier cantidad finita de símbolos al alfabeto de cinta de una máquina de Turing binaria, sabiendo siempre que hay una máquina equivalente en el sentido de que computan la misma función y que solo usa alfabeto de cinta $\Gamma = \{0, 1, B\}$, es decir, que sigue siendo binaria. Este último resultado nos será muy util para memorizar posiciones, caracteres y separar cadenas en una máquina binaria sin preocuparnos por usar solo símbolos 0, 1 y B.

Capítulo 3

El problema de la detención ($\mathcal{H}$).

Hasta ahora hemos utilizado máquinas de Turing que hacen una cierta tarea específica, sin embargo, podemos considerar una máquina universal de Turing (MUT), la cual puede recibir como entrada una máquina de Turing $\mathcal{M}$ y un número $\bar{i}$ y devolver como salida $\mathcal{M}\left(\bar{i}\right)$ si $\mathcal{M}$ se detiene; en cierta forma, la máquina universal $\mathcal{U}$ simula el funcionamiento de otra máquina de Turing previamente construida. Por los resultados del capítulo anterior basta con tratar máquinas de Turing binarias y estas están determinadas gracias a su conjunto de estados y a su función de transición, pues todas tienen los mismos alfabetos de entrada y de cinta, por tanto basta con que la máquina $\mathcal{U}$ tenga la información de todas las tuplas que pertenecen a la función de transición.

Lo anterior tiene un inconveniente y es que una máquina de Turing solo puede tener alfabetos de entrada y de cinta finitos y previamente especificados, así que si $\mathcal{U}$ tuviera un alfabeto de cinta de tamaño n, no podríamos aceptar como entrada una máquina de Turing con, por ejemplo, m estados y $n < m$; además, también querremos que $\mathcal{U}$ sea una máquina de Turing binaria.

Por este motivo necesitaremos primero codificar de alguna manera las máquinas de Turing tal y como se hizo con los lenguajes para poder simular toda máquina de Turing construible con una máquina universal de Turing binaria $\mathcal{U}$.

3.1. Aritmetización de máquinas de Turing y la máquina universal.

Para codificar una máquina de Turing, en realidad debemos codificar los símbolos que aparecen en la función de transición, es decir, sus estados, los caracteres de su alfabeto de cinta que siempre será el conjunto $\{0, 1, B\}$, y los símbolos que representan movimientos que siempre serán I, D, N. Para los estados, aunque el número de estados que se necesitarán codificar para cada máquina de Turing es finito, no tenemos una cota superior para el número de estados que puede tener

una máquina de Turing, así que codificaremos los elementos del conjunto $Q_\infty = \{q_f, q_o, q_1, q_2, \ldots\}$, aunque en cada máquina de Turing solo usaremos una cantidad finita de estos estados.

Definiremos la codificación que usaremos para las máquinas de Turing y para ello codificaremos primero cada uno de los símbolos que mencionamos arriba.

DEFINICIÓN 3.1. Definimos la función

$$\varphi : Q_\infty \cup \{0, 1, B\} \cup \{I, D, N\} \longmapsto \{1\}^*$$

que codifica los símbolos de una función de transición por:

x	$\varphi(x)$
0	1
1	1^2
B	1^3
I	1^4
D	1^5
N	1^6
q_f	1^7
q_i	1^{i+8}

donde 1^i se entenderá como una cadena, por ejemplo: $1^3 = 111$.

Ya que tenemos codificados los símbolos que usa la función de transición podemos proceder a codificar la función en sí.

DEFINICIÓN 3.2. Sea δ la función de transición de una máquina de Turing $\mathcal{M}$, entonces si $\delta(q_i, s) = (q_j, r, m)$ donde q_i, q_j son estados de $\mathcal{M}$; $s, r \in \{0, 1, B\}$ y $m \in \{I, D, N\}$, definimos

$$A_{i,s} = 0\varphi(q_i)\,0\varphi(s)\,0\varphi(q_j)\,0\varphi(r)\,0\varphi(m)\,0$$

en caso de que $\delta(q_i, s)$ esté indefinida, entonces hacemos

$$A_{i,s} = 0\varphi(q_i)\,0\varphi(s)\,0$$

De esta forma podemos al fin dar la codificación de una máquina de Turing en base a su función de transición.

DEFINICIÓN 3.3. Sea $\mathcal{M}$ una máquina de Turing con un conjunto de estados $Q_\mathcal{M}$, entonces definimos la función ρ que la codifica por

$$\rho(\mathcal{M}) = 0A_{f,0}0A_{f,1}0A_{f,B}0A_{0,0}0A_{0,1}0A_{0,B}0A_{1,0}\ldots 0A_{|Q_\mathcal{M}|,1}0A_{|Q_\mathcal{M}|,B}0$$

donde las $A_{i,j}$ son las descritas en la definición 3.2 en base a la función de transición de $\mathcal{M}$.

Ahora ya estamos en condiciones de definir la máquina universal de Turing formalmente, y posteriormente a construir dicha máquina, aunque no lo haremos a detalle pues aunque conceptualmente es muy simple, explicarla detalladamente conlleva muchos aspectos técnicos.

DEFINICIÓN 3.4. Definimos la *máquina universal de Turing* $\mathcal{U}$, como la máquina que al ser alimentada con una cadena $c = B\rho(\mathcal{M})\,00B\overline{w}$ donde $\mathcal{M} = (Q, q_0, q_f, \Gamma, \Sigma, B, \delta)$ y $\overline{w} \in \Sigma^*$ hace lo siguiente:

1. Si $\mathcal{M}(\overline{w}) \downarrow = \overline{k}$ entonces $\mathcal{U}(c) \downarrow = \overline{k}$
2. Si $\mathcal{M}(\overline{w}) \uparrow$ entonces $\mathcal{U}(c) \uparrow$

Notemos que $\mathcal{M}(\overline{w}) \uparrow$, puede pasar por dos motivos: por un lado puede ser que la máquina $\mathcal{M}$ con entrada $\overline{w}$ llegue a una configuración (q_i, uav) y $\delta(q_i, a)$ esté indefinida, o por otro lado, puede que $\mathcal{M}$ con entrada $\overline{w}$ se cuelgue. En ambos casos, la máquina $\mathcal{U}$ llegará al mismo resultado con entrada $(\mathcal{M}, \overline{w})$.

Procedamos entonces a dar una construcción superficial de dicha máquina.

Supongamos que tenemos una entrada de la forma $c = B\rho(\mathcal{M})\,00B\overline{w}$ en $\mathcal{U}$, primero notemos que el único lugar donde la cinta de $\mathcal{U}$ tiene cuatro símbolos 0's consecutivos es entre $\rho(\mathcal{M})$ y $B\overline{w}$, también tenemos cadenas de exactamente tres 0's consecutivos entre cada par de tuplas codificadas de $\delta_{\mathcal{M}}$.

Antes que nada movemos el cabezal a la derecha hasta que encontremos una cadena de cuatro 0's consecutivos, y sustituimos esta cadena por un símbolo especial $S \notin \{0, 1, B\}$ que nos servira para separar $\rho(\mathcal{M})$ de $\overline{w}$, y también para visualizar la cinta de $\mathcal{M}$ a la derecha de S. Así, procedemos a mover el cabezal a la derecha hasta encontrar un símbolo B, el cual será el que este a la izquierda de $\overline{w}$ cambiamos dicho símbolo por un símbolo especial $\sharp \notin \{0, 1, B\} \cup \{S\}$ que nos servirá para recordar la posición del cabezal de $\mathcal{M}$, también entramos en un estado especial indexado con B para memorizar que el símbolo $\sharp$ esta sustituyendo a un B.

Después de esto, movemos el cabezal a la izquierda y después de leer el símbolo S procedemos a escanear las cadenas a la derecha de cada cadena de tres 0's consecutivos hasta encontrar una cadena de la forma $000\varphi(q_0)\,0\varphi(B)$. De esta forma si nosotros tenemos que $A_{0,B} = 0\varphi(q_0)\,0\varphi(B)\,0\varphi(q_i)\,0\varphi(a)\,0\varphi(m)\,0$, entonces entramos en una cadena de estados indexados con i, a, m para recordar el resultado de $\delta_{\mathcal{M}}(q_0, B)$, movemos el cabezal a la derecha hasta encontrar $\sharp$ lo sustituimos por a y escribimos $\sharp$ en la misma celda, la celda de la izquierda o la celda de la derecha, dependiendo de si $m = N$, $m = I$ ó $m = D$

respectivamente. Además pasamos a un estado indexado con i y con c, si c era el caracter que estaba en la celda donde colocamos el $\sharp$.

Ahora repetimos el proceso moviendonos a la izquierda y buscando $A_{i,c}$ después de pasar el símbolo S, así estaremos emulando cada una de las acciones realizadas por la función de transición $\delta_{\mathcal{M}}$.

Si en algún momento encontramos que $\delta_{\mathcal{M}}(q_i, c)$ está indefinida, entonces podemos identificarlo gracias a que tendremos una cadena de tres 0's consecutivos a la derecha de $\varphi(c)$, y entonces tambien dejamos indefinida la función de transición $\delta_{\mathcal{U}}$ de la máquina universal en ese instante.

Si por otro lado intentamos mover el cabezal simulado de $\mathcal{M}$ a la izquierda de su cinta simulada, entonces terminaremos sustituyendo S por $\sharp$, y entonces al buscar el $A_{j,r}$ correspondiente en ese instante nunca pararemos por el símbolo S para comenzar a escanear las cadenas, y así nos seguiremos moviendo a la izquierda hasta colgar $\mathcal{U}$.

De la misma forma si $\mathcal{M}$ se detiene con $\overline{w}$, y supongamos $\mathcal{M} \downarrow (\overline{w}) = \overline{u}$, entonces cuando entremos en la simulación del estado $q_f \in Q_{\mathcal{M}}$, tendremos en la cinta de $\mathcal{U}$ una cadena de la forma $B\rho(\mathcal{M})SB\overline{u}$, y procedemos a borrar toda la cadena desde S hasta el símbolo B al principio de la cinta (sin colgar la máquina pues el símbolo B de la celda más a la izquierda de la cinta nos sirve para identificar la orilla de la cinta), luego cambiamos B por un símbolo especial α para después utilizarlo para indicar que en esa posición moveremos la cadena $B\overline{u}$, y para finalizar detenemos $\mathcal{U}$. De esta forma $\mathcal{U} \downarrow (\rho(\mathcal{M}), \overline{w}) = \overline{u}$.

De esta forma hemos construido, al menos superficialmente, una máquina universal de Turing $\mathcal{U}$.

3.2. Numeración de las máquinas de Turing

Ahora que hemos visto la existencia de una máquina de Turing universal es cuando podemos empezar a construir conjuntos de mayor interés dentro de los naturales. Sin embargo, como lo hemos hecho hasta ahora, deberemos codificar una vez más las máquinas de Turing, pero en este caso utilizareremos una numeración de Gödel (es decir, una función con imagen en los naturales, computable y con inversa computable), para identificar cada máquina con un número natural. Para esto utilizaremos los números primos los cuales ya sabemos que son computables, y por el corolario 2.1, es también la imagen de una función computable $\pi : \mathbb{N} \longmapsto \mathbb{N}$, así $\pi(k)$ será el k-ésimo primo. Entonces llamaremos $\mathfrak{M}$ al conjunto de todas las máquinas de Turing construibles. Recordemos que para cada $\mathcal{M} \in \mathfrak{M}$, dicha máquina está determinada totalmente por su función de transición $\delta_{\mathcal{M}}$. Entonces nuestro objetivo

será construir una función $\Omega : \mathfrak{M} \longmapsto \mathbb{N}$ que sea biyectiva para poder identificar cada máquina con uno, y solo un número natural, y también que cada número natural describa una única máquina de Turing.

OBSERVACIÓN 3.1. Hay una infinidad de máquinas de Turing.

Notemos que para cada máquina de Turing, podemos crear una máquina «distinta», agregando una tupla más a la función de transición que no haga nada; es decir, renombramos el estado final q_f con digamos q_k y para cada $s \in \{0, 1, B\}$ definimos $\delta(q_k, s) = (q_f, s, N)$ donde q_f es el nuevo estado final. Aplicando esto varias veces podemos obtener una máquina de Turing con cualquier cantidad de estados.

LEMA 3.1. $|\mathfrak{M}| = \aleph_0$

DEMOSTRACIÓN. Para cada $\mathcal{M} \in \mathfrak{M}$ tenemos que $Q_{\mathcal{M}} \subsetneq Q_\infty$ así que

$$|\mathfrak{M}| = | \{Q_\infty \times \{0, 1, B\} \times Q_\infty \times \{0, 1, B\} \times \{I, D, N\}\}^{<\omega} |$$

pues cada máquina de Turing tiene una cantidad finita de instrucciones, y gracias a esto y a la observación anterior $|\mathfrak{M}| = \aleph_0$. □

Procedamos entonces a dar una inyección de $\mathfrak{M}$ en $\mathbb{N}$. Antes que nada, para cada máquina $\mathcal{M}$ asignemos un número natural a cada salida de su función de transición $\delta_{\mathcal{M}}$. Por convención, desde ahora para cada máquina de Turing $\mathcal{M}$ tal que $|Q_{\mathcal{M}}| = n$, tendremos que $Q_{\mathcal{M}} = \{q_1, \ldots, q_n\}$, donde q_1 será el estado inicial y q_n será el estado final. De esta forma es fácil identificar los estados con números naturales gracias a sus índices, sin embargo para los conjuntos $\{0, 1, B\}$ y $\{I, D, N\}$ necesitamos definirles un numeral, así que a partir de ahora tomaremos $\#0 = 1$, $\#1 = 2$, $\#B = 3$, $\#I = 1$, $\#D = 2$ y $\#N = 3$.

DEFINICIÓN 3.5. Sea $\mathcal{M}$ una máquina de Turing con un conjunto de estados Q, $q_i, q_j \in Q$, $r, s \in \{0, 1, B\}$ y $m \in \{I, D, N\}$, entonces definimos $\lambda : Q \times \{0, 1, B\} \times \{D, I, N\} \longmapsto \mathbb{N}$, dada por:

$$\lambda(\delta(q_i, r)) = \begin{cases} 2 & \iff \delta(q_i, s) \text{ está indefinida} \\ 3^j \cdot 5^{\#s} \cdot 7^{\#m} & \iff \delta(q_i, r) = (q_j, s, m) \end{cases}$$

Ya definida λ definimos otra función $\Lambda : \mathfrak{M} \longmapsto \mathbb{N}$ dada de la siguiente manera:

$$\Lambda(\mathcal{M}) = 2^{|Q|} \cdot \prod_{i=1}^{|Q|} \prod_{s \in \{0,1,B\}} \pi(3i - 2 + \#s)^{\lambda(\delta(q_i,s))}$$

Dicho más explicitamente

$$\Lambda(\mathcal{M}) = 2^{|Q|} \cdot \prod_{i=1}^{|Q|} \pi(3i-1)^{\lambda(\delta(q_i,0))} \prod_{i=1}^{|Q|} \pi(3i)^{\lambda(\delta(q_i,1))} \prod_{i=1}^{|Q|} \pi(3i+1)^{\lambda(\delta(q_i,B))}$$

O totalmente desarrollado

$$\Lambda(\mathcal{M}) = 2^{|Q|} \cdot 3^{\lambda(\delta(q_1,0))} \cdot 5^{\lambda(\delta(q_1,1))} \cdot 7^{\lambda(\delta(q_1,B))} \cdot 11^{\lambda(\delta(q_2,0))} \cdot \ldots \cdot \pi(3|Q|+1)^{\lambda(\delta(q_n,B))}$$

De esta manera estamos dando un número natural que contiene totalmente la información de una máquina $\mathcal{M}$, pues describe completamente su función de transición $\delta_{\mathcal{M}}$.

PROPOSICIÓN 3.1. *La función Λ es inyectiva.*

DEMOSTRACIÓN. Sean $\mathcal{M}, \mathcal{N} \in \mathfrak{M}$ con $\mathcal{M} \neq \mathcal{N}$ entonces si

$$| Q_{\mathcal{M}} | \neq | Q_{\mathcal{N}} |$$

supongamos sin pérdida de generalidad que $r = | Q_{\mathcal{M}} | > | Q_{\mathcal{N}} |$, entonces

$$2^r \mid \Lambda(\mathcal{M})$$

pero

$$2^r \nmid \Lambda(\mathcal{N})$$

de donde $\Lambda(\mathcal{M}) \neq \Lambda(\mathcal{N})$.

Por otro lado si $| Q_{\mathcal{M}} | = | Q_{\mathcal{N}} |$, entonces existe $q_i \in Q_{\mathcal{M}} \cap Q_{\mathcal{N}}$ y $s \in \{0,1,B\}$ tal que $\delta_{\mathcal{M}}(q_i,s) \neq \delta_{\mathcal{N}}(q_i,s)$, de nuevo tenemos varias opciones, si una de las dos está indefinida y la otra no, digamos $\delta_{\mathcal{M}}(q_i,s)$ está indefinida, entonces

$$2 \mid \lambda(\delta_{\mathcal{M}}(q_i,s))$$

pero

$$2 \nmid \lambda(\delta_{\mathcal{N}}(q_i,s)).$$

Si ambas están definidas entonces

$$(q_j,l,m) = \delta_{\mathcal{M}}(q_i,s) \neq \delta_{\mathcal{N}}(q_i,s) = \left(q'_j,l',m'\right)$$

y en cualquier caso tenemos que hay un p^a tal que:

$$p^a \mid \lambda\left(\delta_{\mathcal{M}}\left(q_i, s\right)\right)$$

pero

$$p^a \nmid \lambda\left(\delta_{\mathcal{N}}\left(q_i, s\right)\right)$$

donde p y a dependen de la coordenada diferente en la tupla. En cualquier caso $\lambda\left(\delta_{\mathcal{M}}\left(q_i, s\right)\right) \neq \lambda\left(\delta_{\mathcal{N}}\left(q_i, s\right)\right)$, y de nuevo sin pérdida de generalidad supongamos que $\lambda\left(\delta_{\mathcal{M}}\left(q_i, s\right)\right) > \lambda\left(\delta_{\mathcal{N}}\left(q_i, s\right)\right)$, de esta forma

$$\pi\left(3i - 1 + \#s\right)^{\lambda\left(\delta_{\mathcal{M}}\left(q_i, s\right)\right)} \mid \Lambda\left(\mathcal{M}\right)$$

y

$$\pi\left(3i - 1 + \#s\right)^{\lambda\left(\delta_{\mathcal{M}}\left(q_i, s\right)\right)} \nmid \Lambda\left(\mathcal{N}\right)$$

luego $\Lambda\left(\mathcal{M}\right) \neq \Lambda\left(\mathcal{N}\right)$. Por lo tanto concluimos que la función Λ es inyectiva. □

Con esto ya podríamos alimentar a la máquina universal con una entrada de la forma $(\overline{n}, \overline{w})$ siempre y cuando $\overline{n}$ sea el numeral de una máquina de Turing, es decir, que exista $\mathcal{M} \in \mathfrak{M}$ tal que $\Lambda\left(\mathcal{M}\right) = n$, y también que podamos obtener $\mathcal{M}$ a partir de n. Veamos que estas dos condiciones pueden satisfacerse.

LEMA 3.2. *La función* $e_k : \mathbb{N} \longmapsto \mathbb{N}$ *definida por:*

$$e_k\left(n\right) = max\left\{i \in \mathbb{N} : k^i \mid n\right\}$$

es computable.

DEMOSTRACIÓN. Aplicando primero las funciones constantes c_k y c_0 transformamos la entrada $\overline{n}$ en una entrada de la forma $\overline{n}0\overline{0}0\overline{k}0\overline{0}$, después aplicando la función exponente $\mathcal{E}$, llegamos a $\overline{n}0\overline{k^o}0\overline{k}0\overline{0}$. Luego si tenemos una cadena de la forma $\overline{n}0\overline{k^i}0\overline{k}0\overline{i}$ sobre la cinta, aplicamos el algoritmo de la división para checar si $k^i \mid n$, en caso de que esto ocurra aplicamos la función sucesor y de nuevo la función exponente para llegar a una cadena de la forma $\overline{n}0\overline{k^{i+1}}0\overline{k}0\overline{i+1}$ y hacer esto recursivamente, si por el contrario $k^i \nmid n$, entonces modificamos el contenido de la cinta (borramos la cinta excepto el numeral de i, lo recorremos hasta la izquierda de la cinta, le restamos 1 y colocamos el cabezal en la celda más a la izquierda) y detenemos la máquina para hacer que $e_k\left(n\right) = i - 1$. Como todos los pasos en la construcción de e_k son computables, entonces e_k también lo es. □

TEOREMA 3.1. *El conjunto*

$$\mathfrak{N} = \{n \in \mathbb{N} : n \text{ es el numeral de una máquina de Turing}\} = Im(\Lambda)$$

es un conjunto computable.

DEMOSTRACIÓN. Sea $n \in \mathbb{N}$, entonces sea $k = e_2(n)$, esto quiere decir que si n es el numeral de una máquina de Turing $\mathcal{M}$, entonces $\mathcal{M}$ tiene k estados. Así la primera condición que debe de cumplir n es que

$$\prod_{i=1}^{3k+1} \pi(i)^{e_{\pi(i)}(n)} = n \tag{1}$$

es decir, que n solo tenga factores que describan a $\mathcal{M}$ y no más.

Como segunda condición ocuparemos que para cada primo $\pi(i)$ con $0 < i \leqslant 3k+1$, n describa la instrucción corrrespondiente a tal primo en $\delta_{\mathcal{M}}$. Entonces dicha condición podemos expresarla como sigue:

$$\left(e_{\pi(i)} = 2\right) \iff (\neg\alpha(n,i)) \tag{2}$$

donde

$$\begin{aligned}\alpha(n,i) \;=\; & \left(1 \leq \left(e_3 \circ e_{\pi(i)}(n)\right) \leq k\right) \wedge \\ & \left(1 \leq \left(e_5 \circ e_{\pi(i)}(n)\right) \leq 3\right) \wedge \\ & \left(1 \leq \left(e_7 \circ e_{\pi(i)}(n)\right) \leq 3\right) \wedge \\ & \left(e_{\pi(i)}(n) = \left(\left(e_3 \circ e_{\pi(i)}(n)\right)\left(e_5 \circ e_{\pi(i)}(n)\right)\left(e_7 \circ e_{\pi(i)}(n)\right)\right)\right)\end{aligned}$$

Más alla de la aparatosa forma de α, lo que describe es muy simple. Supongamos que en la descomposición en factores primos de n, r es el exponente de $\pi(i)$, entonces el último conyunto de $\alpha(n,i)$ se cumple si, y sólo si, $r = 3^a \cdot 5^b \cdot 7^c$, con $a, b, c \in \mathbb{N}$. El primer conyunto se cumple si, y sólo si, $0 < a \leq k$, es decir, es el número de un estado de $\mathcal{M}$. Análogamente el segundo y tercer conyunto se cumplen si, y sólo si, b es el numeral de un elemento de $\{0, 1, B\}$ y c es el numeral de un elemento de $\{I, D, N\}$, respectivamente.

Haciendo esto, $\alpha(n,i)$ se cumple si y solo si $r = \lambda(\delta(q_j, s))$ y $\delta(q_j, s)$ está definida, mientrás que si $\delta(q_j, s)$ está indefinida, se cumple $\left(e_{\pi(i)} = 2\right)$.

De esta forma un número $n \in \mathfrak{N}$ si y solo si cumple las propiedades 1 y 2, las cuales son ambas computables, y por tanto $\mathfrak{N}$ es un conjunto computable. □

PROPOSICIÓN 3.2. *Sea $n \in \mathfrak{N}$, entonces si $\mathcal{M} \in \mathfrak{M}$ es tal que $\Lambda(\mathcal{M}) = n$, podemos describir totalmente a $\mathcal{M}$ a partir de n.*

DEMOSTRACIÓN. Simplemente $\mid Q_{\mathcal{M}} \mid = k = e_2(n)$, y para todo $1 \leq i \leq k$ y $s \in \{0, 1, B\}$, definimos $\delta_{\mathcal{M}}$ de la siguiente forma:

$$\delta_{\mathcal{M}}(q_i, s) = \begin{cases} (q_j, s', m) \iff & \begin{array}{l} \left(e_3 \circ e_{\pi(3i-2+\#s)}(n) = j\right) \wedge \\ \left(e_5 \circ e_{\pi(3i-2+\#s)}(n) = \#s'\right) \wedge \\ \left(e_7 \circ e_{\pi(3i-2+\#s)}(n) = \#m\right) \end{array} \\ \text{indefinida} \iff & 2 \mid \left(e_{\pi(3i-2+\#s)}(n)\right) \end{cases}$$

□

DEFINICIÓN 3.6. Definimos $\Delta : \mathfrak{N} \longmapsto \mathbb{N}$ dada por:

$$\Delta(n) = \begin{cases} 0 & (\nexists m \in \mathbb{N})((m \in \mathfrak{N}) \wedge (m < n)) \\ s(m) & (m \in \mathbb{N}) \wedge ((\nexists l \in \mathbb{N})((l \in \mathfrak{N}) \wedge (\Delta^{-1}(m) < l < n))) \end{cases}$$

donde s es la función sucesor. De esta forma podemos definir $\Omega = \Delta \circ \Lambda : \mathfrak{M} \longmapsto \mathbb{N}$.

TEOREMA 3.2. $\Omega : \mathfrak{M} \longmapsto \mathbb{N}$ *es biyectiva computable y con inversa computable.*

DEMOSTRACIÓN. Sean $\mathcal{M}_n$ y $\mathcal{M}_m$ dos máquinas de Turing distintas, entonces por la inyectividad de Λ, $n \neq m$, y como Δ es estrictamente creciente, $\Delta(n) \neq \Delta(m)$, y así $\Omega(n) \neq \Omega(m)$, por tanto Ω es inyectiva.

También sea $n \in \mathbb{N}$, si $n = 0$, entonces $n = \Delta(a)$ con $a = \min\mathfrak{N}$, y si $n = s(m)$, entonces $n = \Delta(b)$ con $b = \min\{b \in \mathfrak{N} : b > m\}$; de esto tenemos que Δ es suprayectiva y como $\mathrm{Dom}(\Delta) = \mathrm{Im}(\Lambda)$, entonces Ω es también suprayectiva.

Prácticamente ya vimos que la función Λ es computable y también su inversa, solo falta ver que Δ también lo es. Pero podemos dar una máquina $\mathcal{M}_\Omega$ que con entrada n, entonces primero cambie la cinta por una cadena de la forma $B\overline{n}0\overline{0}0\overline{0}B\cdots$, y si en un instante dado tenemos una cadena de la forma $B\overline{n}0\overline{i}0\overline{j}B\cdots$, entonces si $i \notin \mathfrak{N}$ aplicamos la función sucesor a i para obtener $B\overline{n}0\overline{s(i)}0\overline{j}B\cdots$, si $i \in \mathfrak{N}$ y $n \neq i$, aplicamos sucesor a i y a j, para obtener $B\overline{n}0\overline{s(i)}0\overline{s(j)}B\cdots$, por último si $i \in \mathfrak{N}$ y $n = i$, entonces aplicamos sucesor a j y devolvemos $s(j)$. De esta forma j sirve como un contador de los elementos de $\mathfrak{N}$ en orden y nos devuelve exactamente el lugar en el que aparece n. Podemos hacer un procedimiento similar para ver que Δ^{-1} es también computable y de esta manera completar la prueba. □

Gracias a este teorema, si llamamos a $n = \Omega(\mathcal{M})$, el indice de $\mathcal{M}$, podemos hablar del conjunto de las máquinas de Turing a través de su indice. De esta forma llamamos $\mathcal{M}_i$ a la máquina $\mathcal{M} \in \mathfrak{M}$ tal que $\Omega(\mathcal{M}) = i$, entonces $\mathfrak{M} = \{\mathcal{M}_i\}_{i\in\mathbb{N}}$ y también tendremos el conjunto de las funciones computadas por una máquina de Turing como $\{\Phi_i\}_{i\in\mathbb{N}}$. A partir de ahora trabajaremos la mayoria del tiempo con las funciones computables más que con las mismas máquinas de Turing. Sin embargo notemos que dos máquinas pueden considerarse diferentes pero computar la misma función, por ejemplo, en el comentario siguiente a la observación 3.1 dimos una forma de construir a partir de una máquina de Turing, otra que haga exactamente lo mismo pero que sea una máquina diferente; de hecho, hicimos más que eso, para cada máquina, dimos infinitas máquinas que computan la misma función, de esta forma en $\{\mathcal{M}_n\}_{n\in\mathbb{N}}$ cada máquina es distinta pero para cada $i \in \mathbb{N}$, $\{a \in \mathbb{N} : \Phi_a = \Phi_i\}$ es infinito. El hecho de que cada función computable tenga un conjunto de índices infinito nos será útil más adelante.

3.3. El problema de la detención

Como vimos al final de la sección 2.2, preguntarse si un conjunto recursivamente enumerable es computable es más complicada de resolver que las demás pregunas planteadas en esa sección, si tuvieramos una máquina $\mathcal{K}$ que recibiera como entrada el numeral de un par ordenado $\overline{(n,m)}$ y nos regresara como salida 1 si $\mathcal{M}_n \downarrow (m)$ y 0 si $\mathcal{M}_n \uparrow (m)$, entonces todo conjunto recursivamente enumerable sería computable, pues si $A \subseteq \mathbb{N}$ fuera aceptado por una máquina $\mathcal{M}_i$, entonces A sería computado por la máquina $\mathcal{N}$ tal que $\forall k \in \mathbb{N}\left(\mathcal{N}\left(\overline{k}\right) = \mathcal{K}\left(\overline{(i,k)}\right)\right)$. El problema de la detención es precisamente este, dada una máquina de Turing $\mathcal{M}$ y una cadena w sobre su alfabeto de entrada, decidir si $\mathcal{M}$ se detiene ante w o no. Visto de otra forma nosotros podemos definir el problema de la detención como el conjunto $\mathcal{H} = \{(n,m) : \mathcal{M}_n \downarrow (m)\}$. Es claro que este conjunto es computable si y sólo si la máquina $\mathcal{H}$ descrita arriba es construible. Con la notación vista al final de la sección anterior este problema puede verse como el conjunto de pares $\mathcal{H} = \{(n,m) : m \in \mathrm{Dom}(\Phi_n)\}$. A partir de ahora usaremos está última notación.

TEOREMA 3.3. *El problema de la detención no es computable.*

DEMOSTRACIÓN. Supongamos que $\mathcal{H}$ es computable, entonces sea $\mathcal{D} : \mathbb{N} \longmapsto \mathbb{N}$ dada por

$$\mathcal{D}(x) = \begin{cases} \uparrow & \iff x \in \mathrm{Dom}(\Phi_x) \\ 1 & \text{En otro caso} \end{cases}$$

Donde $\mathcal{D}(x) = \uparrow$ representa que $\mathcal{D}(x)$ esta indefinida en x.

Es claro que si $\mathcal{H}$ es computable entonces $\mathcal{D}$ también lo es, y por tanto existe $n \in \mathbb{N}$ tal que $\mathcal{D} = \Phi_n$; luego tenemos que

$$\mathcal{D}(n) = \uparrow \iff n \in \mathrm{Dom}(\Phi_n) \iff n \in \mathrm{Dom}(\mathcal{D}) \iff \mathcal{D}(n) \neq \uparrow$$

lo cual es una contradicción, así concluimos entonces que $\mathcal{H}$ no es computable. □

Tal y como hicimos con el problema de la detención, pudimos haber utilizado alguna otra propiedad de las máquinas para encontrar un conjunto no computable, una generalización de esto es un teorema famoso, llamado el teorema de Rice, el cual dice que decidir si una función computable cumple una propiedad P es computable si, y solo si, el conjunto de funciones que cumplen dicha propiedad es $\{\Phi_i\}_{i\in\mathbb{N}}$ o $\emptyset$. Por otro lado, en diversas ocasiones nos será útil usar versiones un tanto modificadas del problema de la detención, sin embargo utilizando también el teorema de Rice, probaremos su equivalencia con el problema original en 4.2.1.

Capítulo 4

Computabilidad relativa y equivalentes a $\mathcal{H}$.

Las máquinas de Turing pueden funcionar no sólo para distinguir entre problemas computables y no computables (RE), si no también para clasificarlos más ampliamente por su nivel de complejidad, aún dentro de los mismos problemas no computables. Informalmente diremos que un problema A es mas fácil que otro problema B, si suponiendo que tenemos una solución para B, podemos encontrar una solución para A. De esta forma los conjuntos y problemas computables son los más fáciles. Esto nos llevará también a una idea de equivalencia entre dos problemas y demostraremos que ciertos problemas no tienen una solución pues son equivalentes a $\mathcal{H}$, el problema de la detención.

4.1. Computabilidad relativa

Para hablar de computabilidad relativa regularmente usaremos un tipo distinto de máquinas de Turing a las que usamos anteriormente. Estas nuevas máquinas de Turing serán las llamadas máquinas de Turing con oráculo.

Definición 4.1. Sea $A \subseteq \mathbb{N}$, entonces definimos una *máquina de Turing con oráculo* A, como una máquina de Turing $\mathcal{M}^A$ que cuenta con un estado especial q_A tal que al llegar a q_A con un numeral $\overline{n}$ en la cinta, decide si $n \in A$ o si $n \notin A$ y en función de esto continua con la computación dependiendo de este resultado. Como ahora tratamos con las funciones computadas por una máquina más que con las máquinas en si, escribiremos Φ^A para decir que Φ funciona con oráculo A.

En este caso el conjunto A deberá ser preferiblemente no computable, y es claro que en este caso una máquina oráculo puede computar más funciones que una máquina de Turing que no utiliza oráculos (o igual en caso de que A sea computable).

Definición 4.2. *(Orden de Turing)*. Dados $A, B \subseteq \mathbb{N}$, decimos que $A \leq_T B$ si A es computable por una máquina de Turing con oráculo B, en este caso decimos que A es B-computable o que A es computable en B. Definimos tambien $A \equiv_T B$ si $A \leq_T B$ y $B \leq_T A$; en este caso decimos que A y B son Turing-equivalentes o simplemente equivalentes.

OBSERVACIÓN 4.1. Notemos que si un conjunto A es equivalente a $\mathcal{H}$ entonces no es computable.

OBSERVACIÓN 4.2. Con la nueva notación, lo que demostramos en la sección 3.3, se puede sintetizar de la siguiente manera: $\{n \in \mathbb{N} : \Phi_n \downarrow (n)\} \leq_T \mathcal{H}$.

LEMA 4.1. *La relación $\leq_T$ es reflexiva y transitiva.*

Esto se sigue directamente de la definición, sin embargo no podemos decir que $(\mathcal{P}(\mathbb{N}), \leq_T)$, sea un orden parcial, pues la anti-simetría no necesariamente se cumple. Por ejemplo todo par de conjuntos computables son reducibles entre ellos pero no necesariamente son el mismo conjunto. Sin embargo si tomamos conjuntos Turing-equivalentes como uno mismo, es decir hacemos cociente con $\equiv_T$, entonces podemos dar un orden parcial a $\mathcal{P}(\mathbb{N})$.

4.2. Equivalentes a $\mathcal{H}$.

Construiremos primero conjuntos que son versiones un tanto modificadas del problema de la detención y demostraremos su equivalencia con el problema original, esto nos proporcionará mas libertad para demostrar la equivalencia de problemas más generales con $\mathcal{H}$. Con la información que tenemos, ahora es posible demostrar los problemas equivalentes a $\mathcal{H}$ que fueron mencionados al final del capítulo anterior.

4.2.1. Equivalencias inmediatas. Antes que nada, daremos la primera equivalencia directamente.

PROPOSICIÓN 4.1. $\mathcal{H} \equiv_T \{x \in \mathbb{N} : \Phi_x \downarrow (x)\}$.

DEMOSTRACIÓN. Gracias a la observación 4.2 ya tenemos la primera parte del resultado: $\mathcal{H} \geq_T \{x \in \mathbb{N} : \Phi_x \downarrow (x)\}$.

Por otro lado, para probar la otra desigualdad, definamos $h : \mathbb{N} \times \mathbb{N} \longmapsto \mathbb{N}$ de la siguiente forma:

Si $(n, m) \in \mathbb{N} \times \mathbb{N}$, entonces construyamos una máquina de Turing $\mathcal{G}$, tal que con entrada x, $\mathcal{G}$ primero aplica la máquina borradora $\mathcal{B}$, después aplicando la función constante m escribe $\overline{m}$ sobre la cinta, y por último corre $\mathcal{M}_n$ con esta entrada. Notemos que $\mathcal{G}$ hace lo mismo independientemende de su entrada, y que $\mathcal{G} \downarrow (x) \iff \mathcal{M}_n \downarrow (m)$.

Sea k el índice de $\mathcal{G}$, entonces definimos $h(n, m) = k$, de esta forma h es una función total computable. Sea entonces $(n, m) \in \mathbb{N} \times \mathbb{N}$, y

$h(n,m) = k$, entonces tenemos

$$(n,m) \in \mathbb{N}\times\mathbb{N} \iff \mathcal{M}_n \downarrow (m) \iff (\forall x \in \mathbb{N})\left(\mathcal{M}_{h(n,m)} \downarrow (x)\right)$$
$$\iff \mathcal{M}_k \downarrow (k) \iff \Phi_k \downarrow (k) \iff k \in \{x \in \mathbb{N} : \Phi_x \downarrow (x)\}$$

así, concluimos que $\mathcal{H} \leq_T \{x \in \mathbb{N} : \Phi_x \downarrow (x)\}$, y por tanto $\mathcal{H} \equiv_T \{x \in \mathbb{N} : \Phi_x \downarrow (x)\}$. □

Enunciaremos ahora dos de los teoremas más útiles en la teoría de la computabilidad: El teorema s-m-n, aunque en una versión particular, y el teorema de Rice.

DEFINICIÓN 4.3. Sea Φ una función computable, entonces definimos el *conjunto de índices de* Φ por $I_\Phi = \{e \in \mathbb{N} : \Phi = \Phi_e\}$. Si $e \in I_\Phi$ decimos que e es un *índice* de Φ.

Análogo a como lo hicimos con la máquina universal de Turing, podemos definir una función $f : \mathbb{N}\times\mathbb{N} \longmapsto \mathbb{N}$ dada por $f(x,y) = \Phi_x(y)$.

Ahora hacemos lo análogo para conjuntos de funciones.

DEFINICIÓN 4.4. Sea Γ un conjunto de funciones computables, entonces definimos I_Γ, el *conjunto de índices de* Γ *como* $I_\Gamma = \{t \in \mathbb{N} : \Phi_t \in \Gamma\}$.

OBSERVACIÓN 4.3. Si Γ es un conjunto de funciones computables tales que $\Phi \in \Gamma$, entonces $I_\Phi \subseteq I_\Gamma$.

Entonces equivalentemente decimos que I es un conjunto de índices si se tiene que:

$$(e \in I) \implies (\{a \in \mathbb{N} : \Phi_a = \Phi_e\} \subseteq I)$$

TEOREMA 4.1. *El teorema s-m-n. (En este caso n,m=1)*

Sea $f(x,y)$ una función parcial computable de dos variables. Entonces existe una función computable $s : \mathbb{N} \longmapsto \mathbb{N}$ tal que, para todo par (x,y), $\Phi_{s(x)}(y) = f(x,y)$.

DEMOSTRACIÓN. Sea $\mathcal{M}$ una máquina de Turing que compute f y $x \in \mathbb{N}$, entonces construimos una máquina de Turing $\mathcal{N}_x$ tal que con una entrada y, primero cambie el contenido de la cinta de $0\overline{y}0$ a $0\overline{x}0\overline{y}0$, y después concatenamos la máquina $\mathcal{M}$. Sea entonces e el índice de $\mathcal{N}_x$, y definimos $s(x) = e$. Haciendo esto para cada $x \in \mathbb{N}$ es claro que para todo $x \in \mathbb{N}$, $\Phi_{s(x)}(y) = f(x,y)$. Además s es computable pues la construcción de $\mathcal{N}_x$ es computable y obtener el índice de una máquina también lo es gracias a lo visto en el teorema 3.2.

□

TEOREMA 4.2. *Teorema de Rice. Sea I un conjunto de índices, entonces I es computable, si, y solo si, $I = \emptyset$ o $I = \mathbb{N}$.*

DEMOSTRACIÓN. Sea $A \notin \{\emptyset, \mathbb{N}\}$ un conjunto de índices. Sea ahora $e \in \mathbb{N}$ tal que $\text{Dom}(\Phi_e) = \emptyset$, supongamos que $e \notin A$, y fijemos $i \in A$. Definamos una función $g : \mathbb{N}\times\mathbb{N} \longmapsto \mathbb{N}$ dada por:

$$g(x,y) = \begin{cases} \Phi_i(y) & \iff \Phi_x \downarrow (x) \\ \uparrow & \iff \Phi_x \uparrow (x) \end{cases}$$

Ahora por el teorema s-m-n, existe una función computable $s : \mathbb{N} \longmapsto \mathbb{N}$ tal que

$$\Phi_{s(x)}(y) = \begin{cases} \Phi_i(y) & \iff \Phi_x \downarrow (x) \\ \uparrow & \iff \Phi_x \uparrow (x) \end{cases}$$

De esta manera, $\Phi_x \downarrow (x)$ implica que $\Phi_{s(x)} = \Phi_i$, y esto a su vez implica que $s(x) \in A$. Por otro lado si $\Phi_x \uparrow (x)$, entonces $\Phi_{s(x)} = \Phi_e$, y así $s(x) \notin A$. Luego, se sigue que $\Phi_x \downarrow (x)$ si, y solo si, $s(x) \in A$.

El caso en el que $e \in A$, es análogo, por lo que se sigue que A no es computable, pues de serlo $\{x \in \mathbb{N} : \Phi_x \downarrow (x)\}$ también lo sería. □

Esto no significa que todo conjunto distinto del vacío y del total sea no computable, pues muchos problemas no triviales no pueden ser descritos por un conjunto de índices, y así, estos tienen posibilidad de ser conjuntos computables.

Por otro lado, en la demostración del teorema de Rice, no sólo demostramos que un conjunto de índices no trivial es indecidible, sino que además si I es uno de tales conjuntos, entonces $\{x \in \mathbb{N} : \Phi_x \downarrow (x)\} \leq_T I$, de donde obtenemos el siguiente corolario.

COROLARIO 4.1. *Sea I un conjunto de índices no trivial, entonces $\mathcal{H} \leq_T I$.*

Con todo esto, podemos dar ahora algunas equivalencias a $\mathcal{H}$ que nos serán útiles más adelante.

Introducimos un poco de notación que nos será útil para la prueba del siguiente resultado. Para cada $n, s, k \in \mathbb{N}$, denotaremos que $\mathcal{M}_n$ se detiene con entrada k en a lo más s pasos por $\mathcal{M}_n \downarrow (k)[s]$.

TEOREMA 4.3. *Los siguientes conjuntos son Turing-equivalentes.*

1. $\mathcal{H}$
2. $\{n \in \mathbb{N} : \Phi_n \downarrow (n)\}$
3. $\{n \in \mathbb{N} : \Phi_n \downarrow (0)\}$
4. $\{n \in \mathbb{N} : \text{Dom}(\Phi_n) = \emptyset\}$

DEMOSTRACIÓN. $1 \iff 2$ nos lo da la proposición 4.1, y como los conjuntos de 3 y 4 cumplen las hipótesis del teorema de Rice, también tenemos $1 \leq_T 3, 4$ gracias al corolario 4.1. Basta entonces con demostrar $3 \leq_T 1$ y $4 \leq_T 1$; y por transitividad tendremos $1 \equiv_T 2 \equiv_T 3 \equiv_T 4$.

$3 \leq_T 1$)Esto es trivial, pues se tiene que

$$x \in \{n \in \mathbb{N} : \Phi_n \downarrow (0)\} \iff (x,0) \in \{(n,m) \in \mathbb{N} \times \mathbb{N} \,:\, \Phi_n \downarrow (m)\} \iff (x,0) \in \mathcal{H}$$

$4 \leq_T 3$)Sea $m \in \mathbb{N}$, entonces definimos una función parcial computable Φ tal que

$$\Phi(k) = \begin{cases} 0 & \iff (\exists i \in \mathbb{N})\,(i \leq k)\,(\mathcal{M}_m \downarrow (i)\,[k]) \\ \Phi(k+1) & \text{de otro modo} \end{cases}$$

Para ver que esta función es computable observemos que podemos construir una máquina de Turing $\mathcal{M}$ tal que con una entrada k, primero escriba el numeral del 0 sobre la cinta, para dejar sobre la cinta $\overline{k}0\overline{0}$, y teniendo sobre la cinta algo de la forma $\overline{k}0\overline{s}$, corra $\mathcal{M}_m$ con entrada 0, por s pasos. Si $\mathcal{M}_m$ se detiene, entonces detenemos $\mathcal{M}$ y damos 0 como salida, si no lo hace cambiamos, el cero por 1 y así sucecivamente hasta llegar a s. Si tambpoco se detiene con entrada s en s pasos, entonces cambiamos s por $s+1$ y de nuevo continuamos con el proceso hasta hacer $s = k$, en cuyo caso si no se detiene cambiamos k por $k+1$, y así esto será como si alimentaramos a $\mathcal{M}$ con entrada $k+1$.

Sea entonces $e \in \mathbb{N}$, un índice de Φ, de esta manera

$$e \in \{n \in \mathbb{N} : \Phi_n(0)\downarrow\} \iff (\exists k \in \mathbb{N})\,(\Phi_e \downarrow (k)) \iff$$
$$(\exists i \in \mathbb{N})\,(\exists k \in \mathbb{N})\,(i \leq k)\,(\mathcal{M}_m \downarrow (i)\,[k]) \iff$$
$$(\exists i \in \mathbb{N})\,(\mathcal{M}_m \downarrow (i)) \iff \mathrm{Dom}\,(\Phi_m) \neq \emptyset \iff m \in \{n \in \mathbb{N} : \mathrm{Dom}\,(\Phi_n) = \emptyset\}$$

Con esto queda demostrado el teorema. □

4.2.2. Componentes conexas en grafos computables.

Daremos primero una serie de definiciones preliminares para familiarizarnos con los conceptos que usaremos de la teoría de grafos.

4.2.2.1. Definiciones

.

DEFINICIÓN 4.5. Un *grafo* $\mathcal{G}$ es un par de conjuntos (V, A) donde $A \subseteq V \times V$. A los elementos de V los llamamos vértices y a los elementos de A los llamamos aristas.

Consideraremos solo grafos tales que $\mid V \mid \leq \aleph_0$, en este caso podemos pensar sin pérdida de generalidad que $V \subseteq \mathbb{N}$. Decimos también que un grafo $\mathcal{G} = (V, A)$ es computable si $V \subseteq \mathbb{N}$ es computable; y si $A \subseteq \mathbb{N} \times \mathbb{N}$ es un conjunto computable.

Consideraremos grafos no dirigidos y sin lazos, es decir:

$$\forall u, v \in \mathbb{N}\left(((u, v) \in A \iff (v, u) \in A) \wedge ((u, u) \notin A)\right)$$

Ahora consideraremos las componentes conexas de cada vértice, es decir, la componente conexa de un vértice v es el conjunto de vértices que están unidos a él por medio de un camino de aristas.

DEFINICIÓN 4.6. Sea $\mathcal{G} = (V, A)$ un grafo y sea $v \in V$; entonces definimos la componente conexa de v por:

$$\begin{aligned} \mathcal{C}(v) \;=\; & \{u \in V : \exists a_1, \ldots, a_n, a_{n+1} \in A \,\text{y}\, \exists v_1, \ldots, v_n \,\text{tal que} \\ & a_1 = (v, v_1), a_{n+1} = (v_n, u) \,\text{y para}\, 1 < i \leq n \;,\; a_i = (v_{i-1}, v_i)\} \end{aligned}$$

Veamos ahora qué tan complicado es calcular la componente conexa de un vértice en un grafo.

Procedemos ahora a los resultados.

4.2.2.2. Resultados

.

LEMA 4.2. *Sea* $T_n = \{(x_1, \ldots, x_n) : n \in \mathbb{N}$ *y para* $0 < i \leq n$, $x_i \in \mathbb{N}\}$, *entonces el conjunto* $\mathcal{T} = \bigcup_{n \in \mathbb{N}} T_n$ *es RE.*

DEMOSTRACIÓN. Sea $n \in \mathbb{N} \setminus \{0, 1\}$, decimos que n está escrito en *forma normal*, si escribimos a n de la siguiente forma:

$$n = \prod_{i=1}^{k} \pi(i)^{r_i}$$

donde para todo $j > k$, se tiene que $\pi(j) \nmid n$, y algunos $r_i = 0$. En pocas palabras escribimos n como su descomposición en primos pero sin saltarnos ningún primo menor al mayor que lo divide al enumerar los exponentes.

De esta forma sea $T : \mathbb{N} \longmapsto \mathcal{T}$ dada por:

$$T(n) = \begin{cases} (0) & \iff n = 0 \\ (0) & \iff n = 1 \\ (\sup\{r_1, 1\} - 1, \ldots, \sup\{r_k, 1\} - 1) & \iff n = \prod_{i=1}^{k} \pi(i)^{r_i} \text{ en su forma normal} \end{cases}$$

Gracias a que las funciones e_k son computables, T también lo es. Además f es suprayectiva, pues para cada tupla $a = (a_1, \ldots, a_k)$, se tiene que $a = f(n)$ con $n = \prod_{i=1}^{k} \pi(i)^{a_i+1}$. Por tanto $\mathcal{T}$ es RE. □

PROPOSICIÓN 4.2. *Sea $\mathcal{G} = (V, A)$ un grafo computable y $v \in V$, entonces $\mathcal{C}(v) \leq_T \mathcal{H}$.*

DEMOSTRACIÓN. Sea $w \in V$, entonces sea $\Phi_{\mathcal{G}} : V \longmapsto \mathbb{N}$ dada de la siguiente forma; si $T(n) = (a_1, \ldots, a_k)$ entonces

$$\Phi_{\mathcal{G}}(n) = \begin{cases} 1 & \iff \quad (a_1 = v) \wedge (a_k = w) \wedge [\forall i\,((1 \leq i < k) \implies (((a_i, a_{i+1}) \in A) \wedge (a_i \in V)) \\ \text{indefinida} & \text{de otra forma} \end{cases}$$

de esta forma $(w \in \mathcal{C}(v)) \iff (\mathrm{Dom}(\Phi_{\mathcal{G}}) \neq \emptyset)$, y por el teorema 4.3 tenemos que $\mathcal{C}(v) \leq_T \mathcal{H}$. □

PROPOSICIÓN 4.3. *Existe un grafo $\mathcal{G} = (V, A)$ computable y $v \in V$ tal que $\mathcal{C}(v) \geq_T \mathcal{H}$.*

DEMOSTRACIÓN. Daremos $V = \mathbb{N}$ y definimos A de la siguiente manera. Para todo par n, m de números naturales:

$$[(2n, 2m) \notin A] \wedge [(2n+1, 2m+1) \in A] \wedge [((2n, 2m+1) \in A) \iff \mathcal{M}_n(0)[m]\downarrow]$$

es decir, todos los impares están unidos, ningún par esta unido a otro, y entre un par $2n$ y un impar $2m+1$ solo están unidos si $\mathcal{M}_n(0)[m]\downarrow$, de esta forma $\mathcal{G}$ es computable. Además $\mathcal{C}(1) \geq_T \mathcal{H}$, pues $n \in \mathcal{C}(1)$ si y solo si n es impar o n es par y está conectado a un impar; pero esta última condición quiere decir que existe $m \in \mathbb{N}$ tal que $(n, 2m+1) \in A$, luego existe $m \in \mathbb{N}$ tal que $\mathcal{M}_n(0)[m]\downarrow$, y por ende $\mathcal{M}_n(0)\downarrow$. En conclusión, $n \in \mathcal{C}(v) \iff n$ impar o $\mathcal{M}_n(0)\downarrow$. De nuevo por teorema 4.3 tenemos que $\mathcal{C}(1) \geq_T \mathcal{H}$. □

TEOREMA 4.4. *En general el problema de dado un grafo computable $\mathcal{G}$ y $v \in V$ calcular $\mathcal{C}(v)$ es equivalente a $\mathcal{H}$.*

DEMOSTRACIÓN. La demostración del teorema se sigue automaticamente de los lemas 4.2 y 4.3. □

Vayamos ahora a un problema de la teoría de anillos relacionado con encontrar ideales maximales.

4.2.3. Ideales maximales en anillos conmutativos computables.

Igual que en la sección anterior nos ocuparemos primero de dar varias definiciones referentes a anillos e ideales.

4.2.3.1. Definiciones

.

DEFINICIÓN 4.7. Un *anillo con unidad* A es un conjunto con dos operaciones binarias $+ : A \times A \longmapsto A$, y $\cdot : A \times A \longmapsto A$, normalmente llamadas suma y producto respectivamente y dos elementos distinguidos 0 y 1,usualmente distintos, donde se cumple lo siguiente:

1. $\forall a, b, c \in A(a + (b + c) = (a + b) + c)$
2. $\forall a \in A\ (a + 0 = a = 0 + a)$
3. $\forall a \in A \exists\, (-a) \in A\ (a + (-a) = 0)$
4. $\forall a, b \in A\, (a + b = b + a)$
5. $\forall a, b, c \in A(a \cdot (b \cdot c) = (a \cdot b) \cdot c)$
6. $\forall a \in A\ (a \cdot 1 = a = 1 \cdot a)$
7. $\forall a, b, c \in A(a \cdot (b + c) = (a \cdot b) + (a \cdot c))$

Decimos que el anillo es conmutativo si vale:

8. $\forall a, b \in A(a \cdot b = b \cdot a)$

Como de costumbre eliminamos el símbolo $\cdot$ y simplemente escribimos ab en lugar de $a \cdot b$.

DEFINICIÓN 4.8. Un *ideal izquierdo* I de un anillo A es un subconjunto que es cerrado bajo sumas y que absorbe productos, es decir, $\forall i \forall j \in I(i + j \in I)$, y si $i \in I$ y $a \in A$ entonces $ai \in I$. Análogamente definimos ideales derechos si absorbe productos por la derecha. Un ideal que es a su vez izquierdo y derecho se llama ideal bilateral; entonces si el anillo es conmutativo todo ideal izquierdo o derecho es ideal bilateral.

Entre los ideales distinguimos dos tipos importantes de ellos. Un ideal I de A se llama maximal si para todo ideal J de A tal que $I \subseteq J \subseteq A$, se tiene que $I \neq J$ implica $J = A$. Un ideal I se dice que es primo si para todos $i, j \in I$ se tiene que $ij \in I$ implica que $i \in I$ o $j \in I$.

Por último al igual que en gráficas nos centraremos solo en aquellos anillos computables.

DEFINICIÓN 4.9. Un anillo A es un *anillo computable* si $A \subseteq \mathbb{N}$ es computable y si sus operaciones $+, \cdot : A \times A \longmapsto A$ son también

computables.

4.2.3.2. Resultados

.

PROPOSICIÓN 4.4. *Todo anillo conmutativo computable tiene un ideal maximal I tal que $I \leq_T \mathcal{H}$.*

DEMOSTRACIÓN. Sea $(R, +, \cdot)$ un anillo conmutativo, entonces construiremos $I \subsetneq R$ ideal maximal usando a $\mathcal{H}$ como oráculo. Construiremos I por pasos, viendo uno por uno los elementos de R y agregandolos de tal forma que nos den un ideal maximal, esto lo haremos con la ayuda de una familia $\{I_j\}_{j\in\mathbb{N}}$ de subconjuntos de R definidos recursivamente de la siguiente manera; supongamos que para todo $i \leq j$ ya tenemos construido I_i, y construyamos I_{j+1} de la siguiente forma: si $j+1 \notin R$, entonces $I_{j+1} = I_j$; por otro lado si $j \in R$, haremos $I_{j+1} = I_j$ si, y sólo si $1 \notin \langle I_j \cup \{j\}\rangle$, y así $\langle I_{j+1}\rangle \neq R$, si por último $j+1 \in R$ y $1 \in \langle I_j \cup \{j\}\rangle$, entonces hacemos $I_{j+1} = I_j$. Definimos entonces a $I = \bigcup_{j\in\mathbb{N}} I_j$.

Lo único que resta por probar para que I sea un conjunto computable es que la decisión $1 \in \langle I_j \cup \{j\}\rangle$ o $1 \notin \langle I_j \cup \{j\}\rangle$ es computable para toda j. Para esto ocuparemos $\mathcal{H}$ y supongamos que para $i \leq j$ ya sabemos que I_i es computable.

Entonces si $I_j = \{i_1, \ldots, i_k\}$, como R es computable, también tenemos que R^{k+1} es RE, y así existe una función computable suprayectiva $f : \mathbb{N} \longmapsto R^{k+1}$, y de esta forma podemos definir otra función $CL_j : R^{k+1} \longmapsto \mathbb{N}$ dada por:

$$CL_j(n) = \begin{cases} 1 & \iff \quad r_1 i_1 + \ldots + r_k i_k + r_{k+1} j = 1 \\ \uparrow & \quad \text{de otro modo} \end{cases}$$

donde $f(n) = (r_1, \ldots, r_k, r_{k+1})$. Ya que f ,+ y $\cdot$ son computables CL_j también lo es y además $1 \in \langle I_j \cup \{j\}\rangle$ si, y solo si $\mathrm{Dom}(CL_j) \neq \emptyset$ y por el teorema 4.3 y como + y $\cdot$ son las heredadas por R, I_j es computable en $\mathcal{H}$, y así lo es I pues para ver si un número natural k está en I, basta con observar si está en I_{k+1}.

Ahora probemos que I es ideal maximal.
Sean $i, j \in I$ y $r \in R$, y supongamos que $i \leq j$, entonces $i \in I_{i+1}$, $j \in I_{j+1}$ y $I_{i+1} \subseteq I_{j+1}$, de donde $i + j \in \langle I_{j+1}\rangle$ y claramente también $r \cdot j \in \langle I_{j+1}\rangle$.
Ahora sea J un ideal de R tal que $I \subsetneq J \subseteq R$. Sea $m \in J \backslash I$, entonces

como $m \notin I$, $1 \in \langle I_m \cup \{m\}\rangle \subseteq J$ y de esta forma $J = R$, lo que prueba la maximalidad de I. □

Ahora veremos que el poder encontrar un ideal maximal dentro de un anillo conmutativo computable cualquiera también nos proporciona una forma de resolver el problema de la detención. Como de costumbre esta implicación será la mas complicada para demostrar que este problema referente a ideales maximales y el problema de la detención son equivalentes. Comenzaremos con una observación.

OBSERVACIÓN 4.4. El anillo de polinomios con una cantidad numerable de variables sobre $\mathbb{Q}$ denotado por $\mathbb{Q}[x_1, x_2, \ldots]$, es un anillo computable.

DEMOSTRACIÓN. Gracias a la sección 2.2, es facil convencerse de que las operaciones suma y producto en $\mathbb{Q}[x_1, x_2, \ldots]$ son computables. Ahora sólo resta probar que los elementos del anillo se pueden biyectar computablemente con un subconjunto computable de los naturales. En pocas palabras tenemos que Gödelizar al conjunto $\mathbb{Q}[x_1, x_2, \ldots]$. Esto lo hacemos de la siguiente manera: damos una función $\varphi : \mathbb{Q}[x_1, x_2, \ldots] \longmapsto \mathbb{N}$ donde si $p(x_1, x_2, \ldots) \in \mathbb{Q}[x_1, x_2, \ldots]$, entonces $p(x_1, x_2, \ldots) = \sum_{n\in\mathbb{N}} \sum_{i\in\mathbb{N}} a_{n,i} x_i^n$. Entonces usando una biyección $f : \mathbb{N}^2 \longmapsto \mathbb{N}$ como en la página 24, y como cada polinomio es finito podemos hacer $\varphi(p(x_1, x_2, \ldots)) = \prod \pi(f(i, n))^{a_{i,n}}$ tomando los $a_{i,n}$ distintos de 0, lo cual determina totalmente al polinomio $p(x_1, x_2, \ldots)$. □

PROPOSICIÓN 4.5. *Existe un anillo A tal que para todo ideal $I \subseteq A$, se tiene que $\mathcal{H} \leq_T I$.*

DEMOSTRACIÓN. Hagamos la demostración en varios pasos.

Paso 1) Construcciones iniciales.

Tomemos el anillo $\mathbb{Q}[x_1, x_2, \ldots]$ el cual es computable, y haciendo uso del teorema 4.3 tomamos $\mathcal{H} = \{n \in \mathbb{N} : \Phi_n(0)\downarrow\}$ y el subconjunto $\{x_i : i \in \mathcal{H}\} \subseteq \mathbb{Q}[x_1, x_2, \ldots]$. Entonces sean

$$A = \{x_i : i \in \mathcal{H}\} \text{ y } \mathcal{R} = \langle A\rangle$$

de esta forma $p \in \mathcal{R}$ si, y solo si, para cada monomio m de p, se tiene que existe $i \in \mathcal{H}$ tal que x_i divide a m.

Sea entonces $\mathcal{S} = \mathbb{Q}[x_1, x_2, \ldots] \setminus \mathcal{R}$, entonces $p \in \mathcal{S}$ si y solo existe un monomio n de p tal que x_i no divide a n para ningún $i \in \mathcal{H}$.

Afirmamos que $\mathcal{M}$ es cerrado bajo multiplicación.

Ciertamente, sean p , q polinomios sobre $\mathcal{S}$, entonces hay monomio m_p de p tal que para todo $i \in \mathcal{H}$, x_i no divide a m_p, y analogamente hay un m_q monomio de q que cumple las mismas propiedades, por tanto $m = m_p \cdot m_q$ es un monomio de $p \cdot q$ tal que para todo $i \in \mathcal{H}$ se tiene que x_i no divide a $p \cdot q$, y por tanto $p \cdot q \in \mathcal{S}$.

Paso 2) Construcción del anillo y del ideal.

Como $\mathbb{Q}[x_1, x_2, \ldots]$ es dominio, entonces podemos hablar de su campo de cocientes $\mathbb{Q}(x_1, x_2, \ldots)$, y tomamos el subconjunto

$$\mathcal{K} = \left\{ \frac{p}{q} \in \mathbb{Q}(x_1, x_2, \ldots) : q \in \mathcal{S} \right\}$$

de esta forma y gracias a que $\mathcal{S}$ es cerrado bajo multiplicación, $\mathcal{K}$ es un subanillo de $\mathbb{Q}(x_1, x_2, \ldots)$.

Tomemos ahora el conjunto

$$\mathcal{K}_{\mathcal{R}} = \left\{ \frac{p}{q} \in \mathbb{Q}(x_1, x_2, \ldots) : (p \in \mathcal{R}) \wedge (q \in \mathcal{S}) \right\}$$

gracias a que $\mathcal{R}$ es un ideal de $\mathbb{Q}[x_1, x_2, \ldots]$, se sigue que $\mathcal{K}_{\mathcal{R}}$ es un ideal de $\mathbb{Q}(x_1, x_2, \ldots)$.

Paso 3) Todo ideal maximal en $\mathcal{K}$ computa $\mathcal{H}$.

Notemos que $\mathcal{K}_{\mathcal{R}}$ es un ideal maximal de $\mathcal{K}$, pues si $\frac{p}{q} \in \mathcal{K} \backslash \mathcal{K}_{\mathcal{R}}$, se sigue que $p \in \mathcal{S}$, y por tanto $\frac{q}{p} \in \mathcal{K}$, y como los ideales absorben la multiplicación, $\frac{q}{p} \cdot \frac{p}{q} = 1$ debería estar en el ideal.

Por otro lado si I es cualquier ideal de $\mathcal{K}$ distinto del total, se sigue que I no contiene elementos de la forma $\frac{p}{q}$, con $p \in \mathcal{S}$, pues por lo que acabamos de ver, 1 estaría en I. De esta forma todos los elementos de I son de la forma $\frac{p}{q}$, con $p \notin \mathcal{S}$, es decir, $p \in \mathcal{S}^c = \mathcal{R}$, y por tanto $\frac{p}{q} \in \mathcal{K}_{\mathcal{R}}$, de donde se sigue que $I \subseteq \mathcal{K}_{\mathcal{R}}$, y por tanto $\mathcal{K}_R$ es el único ideal maximal de $\mathcal{K}$.

Además

$$\frac{x_i}{1} \in \mathcal{K}_{\mathcal{R}} \iff x_i \in \mathcal{R} \iff x_i \in A \iff i \in \mathcal{H}$$

y entonces todo ideal maximal I de $\mathcal{K}$, el cual solo puede ser $\mathcal{K}_{\mathcal{R}}$, se tiene que $\mathcal{H} \leq_T I$.

Paso 4)$\mathcal{K}_{\mathcal{R}}$ es un conjunto RE.

Para empezar probaremos que $\mathcal{S}$ es un conjunto RE, para ello probaremos que es RE dentro de $\mathbb{Q}[x_1, x_2, \ldots]$, que también es RE.

Sea $p \in \mathbb{Q}[x_1, x_2, \ldots]$, y digamos que $p = \sum_{i=1}^{k} m_i$ donde cada m_i es un monomio en $\mathbb{Q}[x_1, x_2, \ldots]$. Para cada i, sea ahora $\{x_{i,1}, \ldots, x_{i,n_i}\}$ el conjunto de los x tales que $x_{i,j}$ divide a m_i.

Recordemos que $\mathcal{M}_n \downarrow (a)[s]$, denota que la máquina $\mathcal{M}_n$ se detiene con entrada a en a lo más s pasos, como en este caso k siempre será igual a 0, omitiremos la entrada de la máquina y solo denotaremos por $\mathcal{M}_n \downarrow [s]$, a $\mathcal{M}_n \downarrow (0)[s]$.

Entonces damos una máquina de Turing $\mathfrak{U}$, que con entrada p, y para cada monomio m_i haga lo siguiente: pruebe si para cada $x \in \{x_{i,1}, \ldots, x_{i,n_i}\}$, $\mathcal{M}_{i,j} \downarrow [s]$, con $s = 0$, esto claramente es computable pues hay una cantidad finita de $x_{i,j}$, y cada uno se prueba una cantidad finita de pasos, si algún monomio m_i cumple esta condición, entonves detenemos $\mathfrak{U}$, si no hacemos $s = 1$, y repetimos el proceso recursivamente. Haciendo esto tenemos que $\mathfrak{U} \downarrow (p)$ si, y solo si, $p \in \mathcal{S}$. Pues si $p \in \mathcal{S}$, entonces hay un monomio m_i tal que el conjunto $\{x_{i,1}, \ldots, x_{i,n_i}\} \subseteq A$, y asi para cada $x_{i,j}$ hay un $s_{i,j}$ tal que $\mathcal{M}_{i,j} \downarrow [s_{i,j}]$, y asi tomando $t = \max\{s_{i,j} : 0 < j \leq n_i\}$, se tiene que $\mathfrak{U}$ se detendrá en m_i cuando $s = t$.

Por último como $\mathcal{S}$ es RE, entonces también lo es $\mathcal{K}$.

Paso 5) Volviendo las cosas computables

$\mathcal{K}$ y $\mathcal{K}_{\mathcal{R}}$ no son computables, sin embargo como $\mathcal{K}$ es RE, existe una biyección $\{k_1, k_2, \ldots\}$ con los numeros naturales. Llamemosle f a dicha biyección donde $f : \mathbb{N} \longmapsto \mathcal{K}$ esta dada por $f(n) = k_n$.

Sea entonces el anillo $A = \mathbb{N}$ con las operaciones $+_A$, y $\cdot_A$, dadas por $x +_A y = f^{-1}(f(x) +_{\mathcal{K}} f(y))$, y $x \cdot_A y = f^{-1}(f(x) \cdot_{\mathcal{K}} f(y))$. De esta forma el anillo A es computable, pues $A = \mathbb{N}$, y $+_A$, $\cdot_A$ están definidas enterminos de f,$+_{\mathcal{K}}$ y $\cdot_{\mathcal{K}}$; las cuales son computables.

Claramente f abre las operaciones pues $f(x +_A y) = f(f^{-1}(f(x) +_{\mathcal{K}} f(y))) = f(x) +_{\mathcal{K}} f(y)$, y lo análogo se tiene para el producto.

De aquí se tiene que A es isomorfo a $\mathcal{K}$ y por tanto tiene un único ideal maximal $f^{-1}(K_R)$, y no solo eso, sino que $\mathcal{H} \leq_T f^{-1}(K_R)$, pues

$$n \in \mathcal{H} \iff x_n \in \mathcal{R} \iff x_n \in \mathcal{K}_{\mathcal{R}} \iff f^{-1}(x_n) \in f^{-1}(K_R)$$

por tanto, $(A, +_A, \cdot_A)$ es un anillo computable tal que todo ideal maximal computa $\mathcal{H}$. $\square$

TEOREMA 4.5. *$\mathcal{H}$ es necesario y suficiente para calcular ideales maximales en anillos conmutativos computables.*

DEMOSTRACIÓN. La demostración se sigue como corolario de las proposiciones 4.4 y 4.5. □

La próxima equivalencia, será un tanto más complicada y larga que las anteriores, así que le dedicaremos un capítulo aparte.

Capítulo 5

El décimo problema de Hilbert.

Dentro de los problemas planteados por Hilbert en 1900, uno era referente a la solución de polinomios con coeficientes enteros, el enunciado era parecido al siguiente:

> "Dada una ecuación diofántica con cualquier cantidad de incógnitas y con coeficientes racionales enteros, crear un mecanismo con el cual en una cantidad finita de operaciones pueda determinarse si la ecuación tiene solución en los racionales enteros"

En otras palabras, se pide dar un algoritmo que al darle una ecuación diofántica con coeficientes enteros y cualquier cantidad de variables como entrada, nos diga si esta tiene solución o no dentro de los enteros. En este trabajo entenderemos «máquina de Turing» por algoritmo, y el problema de decidir si una ecuación diofántica particular tiene o no soluciones en los enteros, será traducido a responder si un elemento en el conjunto $\mathbb{D}$ de todas las ecuaciones diofanticas con cualquier cantidad de variables (aritmetizado de cierta manera para usarlo como un subconjunto de $\mathbb{N}$), pertenece o no al subconjunto $\mathbb{S}$ de $\mathbb{D}$, de todas las ecuaciones diofánticas que sí tienen solución en los enteros.

Demostraremos en su momento, que la respuesta a esta pregunta planteada por Hilbert es «no», y más aún, demostraremos que este procedimiento es equivalente a $\mathcal{H}$.

Definición 5.1. Una ecuación diofántica es una ecuación de la forma

$$D(x_1, \ldots, x_n) = 0 \tag{3}$$

donde D es un polinomio con coeficientes enteros.

Una ecuación diofántica también puede ser representada por

$$D_i(x_1, \ldots, x_n) = D_d(x_1, \ldots, x_n) \tag{4}$$

de manera que todos los coeficientes sean positivos.

Por otro lado, considerar ecuaciones diofánticas o sistesmas de estas, será equivalente para nuestras causas, pues un sistema de ecuaciones

diofánticas

$$\begin{gathered} D_1(x_1,\ldots,x_n)=0 \\ \vdots \\ D_k(x_1,\ldots,x_n)=0 \end{gathered}$$

tendrá solución, si, y solo si, la ecuación

$$D_1^2(x_1,\ldots,x_n)+\ldots+D_k^2(x_1,\ldots,x_n)=0 \tag{5}$$

que también es diofántica, tiene solución; donde $D_1^2(x_1,\ldots,x_n)$ denota
$[D_1(x_1,\ldots,x_n)]^2$.

Análogamente, aunque podría parecer poco útil, podemos considerar una ecuación diofántica $D(x_1,\ldots,x_n)=0$, como un sistema de ecuaciones diofánticas

$$\begin{gathered} D_1(x_1,\ldots,x_n,y_1,\ldots,y_m)=0 \\ \vdots \\ D_k(x_1,\ldots,x_n,y_1,\ldots,y_m)=0 \end{gathered}$$

sin embargo, esto nos da la gran ventaja de que cada una de estas ecuaciones puede ser de la forma $a=bc$ o $a=b+c$, con $a,b,c\in\mathbb{Z}\cup\{x_i\}_{i\in\mathbb{N}}\cup\{y_j\}_{j\in\mathbb{N}}$,es decir, podemos escribir cada una de las ecuaciones como suma o producto de dos cosas que pueden ser incógnitas originales, nuevas incógnitas previamente creadas o números enteros.

Por ejemplo la ecuación que satisfacen los puntos de $\mathbb{S}^2$, la cáscara de una esfera en $\mathbb{R}^3$, es

$$x^2+y^2+z^2-1=0$$

la podemos ver como el sistema

$$\begin{array}{rcc} p_1 & = & x \\ p_2 & = & p_1x \\ q_1 & = & y \\ q_2 & = & q_1y \\ r_1 & = & z \\ r_2 & = & r_1z \\ a & = & p_2+q_2 \\ b & = & a+r_2 \\ b & = & 1 \end{array}$$

donde las ecuaciones son de la forma de la ecuación (4), y también cada una es de la forma $a=bc$ o $a=b+c$. Nuevamente este sistema tendrá solución, si, y sólo si, la ecuación original tiene solución.

También decimos que el grado de una ecuación diofántica es el máximo de la suma de las potencias de las variables que aparecen en cada monomio.

Ahora podemos combinar estos dos procedimientos; dada una ecuación diofántica D, primero lo convertimos en un sistema de ecuaciones de la forma $a = bc$ y $a = b + c$, las cuales son ecuaciones de grado a lo más dos, y el sistema tiene solución si, y sólo si, D la tiene, y después transformando este sistema en una ecuación D' del tipo de la ecuación (5), se tiene que D tiene solución, si, y solo si D la tiene, pero D' tiene grado a lo más cuatro.

A partir de ahora, llamaremos $\mathbb{D}$ al conjuntode todas las ecuaciones diofánticas, y a $\mathbb{S} \subseteq \mathbb{D}$, el conjunto de todas las ecuaciones diofánticas que tienen solución en los enteros. Llamaremos también $\mathbb{S}_{\mathbb{N}}$, al conjunto de ecuaciones diofánticas con soluciones en los números naturales. Los supondremos aritmetizados dentro de los números naturales. Digamos

$$\#\left(D\left(x_1,\ldots,x_n\right)=0\right)=2^n\prod\pi\left(i\right)^{a_i}$$

si $D\left(x_1,\ldots,x_n\right)=0\in\mathbb{S}_{\mathbb{N}}$, y

$$\#\left(D\left(x_1,\ldots,x_n\right)=0\right)=2^n\prod\pi\left(i\right)^{2|a_i|+v_i}$$

si $D\left(x_1,\ldots,x_n\right)=0\in\mathbb{S}$, donde

$$v_i=\begin{cases}0, & a\geq 0\\ 1, & a<0\end{cases}$$

y en ambos casos los a_i son los coeficientes de $D\left(x_1,\ldots,x_n\right)=0$.

5.1. Equivalencia entre $\mathbb{N}$ y $\mathbb{Z}$.

Con el uso de las máquinas de Turing y las funciones computables, sólo nos es posible hablar acerca de ecuaciones diofánticas que tengan tanto variables como coeficientes en los números naturales, y aunque en particular una ecuación diofántica puede tener soluciones enteras (incluso una infinidad de ellas) y no tener soluciones en los naturales, el problema de encontrar sus soluciones en general, es equivalente en $\mathbb{N}$ y en $\mathbb{Z}$.

Para esto usaremos un teorema de la teoría de números el cual no demostraremos.

LEMA 5.1. *(Teorema de los cuatro cuadrados de Lagrange). Sea $n\in\mathbb{N}$, entonces existen $a,b,c,d\in\mathbb{Z}$ tales que*

$$n=a^2+b^2+c^2+d^2$$

PROPOSICIÓN 5.1. $\mathbb{S}_{\mathbb{N}} \leq_T \mathbb{S}$

DEMOSTRACIÓN. Sea $D(x_1, \ldots, x_n) = 0$ una ecuación diofántica, entonces podemos hacer el sistema de ecuaciones

$$D(x_1, \ldots, x_n) = 0$$

$$\begin{aligned} x_1 &= y_{1,1}^2 + y_{1,2}^2 + y_{1,3}^2 + y_{1,4}^2 \\ &\vdots \\ x_n &= y_{n,1}^2 + y_{n,2}^2 + y_{n,3}^2 + y_{n,4}^2 \end{aligned} \tag{6}$$

de esta forma, si tenemos un conjunto de números

$$\{x_i : (1 \leq i \leq n)\} \cup \{y_{i,j} : (1 \leq i \leq n)\,(1 \leq j \leq 4)\}$$

que cumpla este sistema de ecuaciones, entonces $\{x_i : (1 \leq i \leq n)\}$ es unconjunto de números naturales que satisface $D(x_1, \ldots, x_n) = 0$.

Recíprocamente, si $D(x_1, \ldots, x_n) = 0$ tiene solución en números naturales, el sistema (6) tiene solución por el teorema de los cuatro cuadrados de Lagrange. Entonces podemos comprimir el sistema en una única ecuación

$$D'(x_1, \ldots, x_n, y_{1,1}, \ldots, y_{n,4}) = 0$$

y $(D(x_1, \ldots, x_n) = 0) \in \mathbb{S}_{\mathbb{N}} \iff (D'(x_1, \ldots, x_n, y_{1,1}, \ldots, y_{n,4}) = 0) \in \mathbb{S}$ □

Esto nos dice que dada una ecuación diofántica, podemos decidir si tiene solución en los naturales, si ya podemos decidir el problema para los números enteros; también el reciproco es cierto.

PROPOSICIÓN 5.2. $\mathbb{S}_{\mathbb{N}} \geq_T \mathbb{S}$

DEMOSTRACIÓN. Sea $D(x_1, \ldots, x_n) = 0$ una ecuación diofántica, entonces consideramos la ecuación $D(a_1 - b_1, \ldots, a_n - b_n) = 0$, de esta forma si tenemos una solución de la segunda ecuación en números naturales, tomando $x_i = a_i - b_i$ tenemos una solución de la primera, y análogamente si tenemos una solución de la primera, tomamos $a_i = x_i$ y $b_i = 0$ si $x_i \geq 0$, y $a_i = 0$ y $b_i = |x_i|$ si $x_i < 0$. Así, $D(x_1, \ldots, x_n) = 0$ tiene solución en los enteros si, y sólo si, $D(a_1 - b_1, \ldots, a_n - b_n) = 0$ tiene solución en los naturales. □

TEOREMA 5.1. $\mathbb{S}_{\mathbb{N}} \equiv_T \mathbb{S}$

DEMOSTRACIÓN. Se sigue de las dos proposiciones anteriores. □

A partir de ahora, cuando hablemos de ecuaciones diofánticas , nos referiremos entonces a ecuaciones que toman valores en los números naturales, y gracias a que podemos escribirla en la forma de la ecuación

(4), también la consideraremos con coeficientes en los números naturales.

Podemos dar ahora la primera parte de la demostración de que $\mathbb{S}$ es equivalente a $\mathcal{H}$.

TEOREMA 5.2. $\mathbb{S} \leq_T \mathcal{H}$

DEMOSTRACIÓN. Sea $D(x_1, \ldots, x_n) = 0$ una ecuación diofántica, entonces tomandola en su forma $D_i(x_1, \ldots, x_n) = D_d(x_1, \ldots, x_n)$, tenemos que tanto $D_i(x_1, \ldots, x_n)$ como $D_i(x_1, \ldots, x_n)$ pueden ser vistas como funciones totales computables por máquinas de Turing $\mathcal{M}_i$ y $\mathcal{M}_d$ respectivamente, pues son polinomios. Definamos entonces Φ tal que

$$\Phi(x_1, \ldots, x_n) = \begin{cases} 0 & \iff \Phi_i(x_1, \ldots, x_n) = \Phi_d(x_1, \ldots, x_n) \\ \uparrow & \text{de otro modo} \end{cases}$$

Sea entonces $e \in \mathbb{N}$ un índice de Φ. Así

$$\begin{aligned} &(D(x_1, \ldots, x_n) = 0) \in \mathbb{S} \\ &\iff (\exists (x_1, \ldots, x_n) \in \mathbb{N}^n)(D_i(x_1, \ldots, x_n) = D_d(x_1, \ldots, x_n)) \\ &\iff (\exists (x_1, \ldots, x_n) \in \mathbb{N}^n)(\Phi_i(x_1, \ldots, x_n) = \Phi_d(x_1, \ldots, x_n)) \\ &\iff (\exists (x_1, \ldots, x_n) \in \mathbb{N}^n)(\Phi_e \downarrow (x_1, \ldots, x_n)) \\ &\iff e \notin \{k \in \mathbb{N} : \text{Dom}(\Phi_k) = \emptyset\} \end{aligned}$$

y por el teorema 4.3, se sigue que $\mathbb{S} \leq_T \mathcal{H}$. □

5.2. Conjuntos diofánticos

DEFINICIÓN 5.2. Una *familia de ecuaciones diofánticas* es una ecuación diofántica de la forma $D(p_1, \ldots, p_n, x_1, \ldots, x_m) = 0$, donde clasificamos sus variables en dos tipos, *los parámetros* $p_1, \ldots, p_n$, y *las incógnitas* $x_1, \ldots, x_m$.

Fijando valores para el conjunto de parámetros, obtenemos una ecuación diofántica particular sobre las variables $x_1, \ldots, x_m$, de esta forma podemos pensar una familia de ecuaciónes diofánticas como el conjunto de todas las ecuaciones diofánticas sobre sus incógnitas.

DEFINICIÓN 5.3. Sea $n \in \mathbb{N}$ y $\mathfrak{D} \subseteq \mathbb{N}^n$. Decimos que $\mathfrak{D}$ es un *conjunto diofántico* si se tiene que

(7)
$$(p_1, \ldots, p_n) \in \mathfrak{D} \iff (\exists (x_1, \ldots, x_m) \in \mathbb{N}^m)(D(p_1, \ldots, p_n, x_1, \ldots, x_m) = 0)$$

donde $D(p_1, \ldots, p_n, x_1, \ldots, x_m) = 0$ es una familia de ecuaciones diofánticas. Si $\mathfrak{D}$ es un conjunto diofántico, a (7) la llamamos una

representación diofántica de $\mathfrak{D}$. El número n se dice que es la *dimensión de* $\mathfrak{D}$.

PROPOSICIÓN 5.3. *La unión e intersección de conjuntos diofánticos es un conjunto diofántico.*

DEMOSTRACIÓN. Sean $\mathfrak{D}$ y $\mathfrak{F}$ dos conjuntos diofánticos y $D_1(p_1,\ldots,p_n,x_1,\ldots,x_{m_1})=0$ y $D_2(p_1,\ldots,p_n,y_1,\ldots,y_{m_2})=0$ sus respectivas representaciones diofánticas, entonces

$$D_1^2(a_1,\ldots,a_n,x_1,\ldots,x_{m_1})+D_2^2(a_1,\ldots,a_n,y_1,\ldots,y_{m_2})=0$$

es una representación diofántica de $\mathfrak{D}\cap\mathfrak{F}$.
Nótese que hemos forzado a que $\mathfrak{D}$ y $\mathfrak{F}$ tengan los mismos parámetros, para el caso de la unión esto no será necesario.
Si $\mathfrak{D}$ y $\mathfrak{F}$ tienen a $D_1(p_1,\ldots,p_{n_1},x_1,\ldots,x_{m_1})=0$ y $D_2(q_1,\ldots,q_{n_2},y_1,\ldots,y_{m_2})=0$ como sus respectivas representaciones diofánticas, entonces

$$D_1(a_1,\ldots,a_n,x_1,\ldots,x_{m_1})\cdot D_2(a_1,\ldots,a_n,y_1,\ldots,y_{m_2})=0$$

es una representación de $\mathfrak{D}\cup\mathfrak{F}$. □

Para el caso de un conjunto diofántico de dimensión uno, es decir, de un subconjunto diofántico de los números naturales, podemos dar una caracterización.

PROPOSICIÓN 5.4. *Un conjunto* $\mathfrak{D}$ *es diofántico si, y solo si, es el conjunto de todos los valores positivos que toma algún polinomio con coeficientes enteros con valores naturales sobre sus incógnitas.*

DEMOSTRACIÓN. $\Longrightarrow$)Sea $D(p,x_1,\ldots,x_n)=0$ una representación diofántica de $\mathfrak{D}$, entonces tomamos el polinomio

$$P(x_0,\ldots,x_n)=(x_0+1)\left(1-D^2(x_0,x_1,\ldots,x_n)\right)-1.$$

Así si $a_0\in\mathfrak{D}$, entonces $D(a_0,x_1,\ldots,x_n)=0$, y

$$P(a_0,x_1,\ldots,x_n)=a_0$$

Por otro lado sea b_0 tal que

$$b_0=P(x_0,\ldots,x_n)=(x_0+1)\left(1-D^2(x_0,x_1,\ldots,x_n)\right)-1$$

como $b_0\in\mathbb{N}$, $(1-D^2(x_0,x_1,\ldots,x_n))>0$, y así , $D(x_0,x_1,\ldots,x_n)=0$, y se tiene que $(x_0+1)-1=b_0$, de donde se sigue que $x_0=b_0$ y entonces $D(b_0,x_1,\ldots,x_n)=0$, por lo que $b_0\in\mathfrak{D}$.

$\Longleftarrow$)Sea $\mathfrak{D}=\{a\in\mathbb{N}:(\exists(x_1,\ldots,x_n))(D(x_1,\ldots,x_n)=a)\wedge(D\in\mathbb{D})\}$, entonces tomando $D(a,x_1,\ldots,x_n)=D(x_1,\ldots,x_n)-a$. Se tiene que $D(a,x_1,\ldots,x_n)=0$ es una representación diofántica de $\mathfrak{D}$. □

De forma natural podemos hablar también acerca de propiedades y relaciones diofánticas, gracias a los conjuntos de número naturales que éstas definen.

DEFINICIÓN 5.4. Una propiedad $\mathcal{P}$ de los números naturales se dice que es una *propiedad diófantica* si el conjunto $\{n \in \mathbb{N} : \mathcal{P}(n)\}$ es diofántico. Análogamente una relación n-aria $\mathcal{R}$ es una *relación diofántica* si el conjunto $\{(p_1 \ldots, p_n) \in \mathbb{N}^n : \mathcal{R}(p_1 \ldots, p_n)\}$ lo es.

Con esta terminología, la unión e intersección de conjuntos diofánticos pueden ser tratados como disyunciones y conjunciones de propiedades o relaciones diofánticas respectivamente. Por último, hacemos lo mismo para funciones.

DEFINICIÓN 5.5. Una función $f : \mathbb{N}^n \longmapsto \mathbb{N}$ es una *función diofántica*, si su grafo lo es. Donde $\mathcal{G}_f$, el grafo de f es una relación n+1-aria, dada por

$$\mathcal{G}_f(a_1, \ldots, a_n, b) \iff f(a_1, \ldots, a_n) = b$$

PROPOSICIÓN 5.5. *Las siguientes propiedades, relaciones y funciones sobre los números naturales son diofánticas. Si* $a, b \in \mathbb{N}$*:*

1. La propiedad $\mathcal{P}$ de ser un número par.
2. La propiedad $\mathcal{I}$ de ser un número impar.
3. Las relaciones $=, \neq$ de igualdad y desigualdad.
4. Las relaciones $<, \leq$ de orden.
5. La relación $|$ de divisibilidad.
6. La función residuo $r : \mathbb{N}^2 \longmapsto \mathbb{N}$, tal que $r(a, b) = c$ donde c es el residuo al dividir a por b.
7. La relación $\nmid$ de no divisibilidad.
8. La función $m : \mathbb{N}^2 \longmapsto \mathbb{N}$, dada por $m(a, b) = \min\{|c| : c \equiv a \ (mod\, b)\}$.
9. La función $[\cdot] : \mathbb{N}^2 \longmapsto \mathbb{N}$, tal que $[(a, b)] = q$ si q es el cociente al dividir a entre b.
10. Si $x \in \mathbb{N}$, la relación $\equiv_x$ de ser congruentes módulo x.
11. La función mcd : $\mathbb{N}^2 \longmapsto \mathbb{N}$, que da el máximo común divisor entre dos números
12. La función mcm : $\mathbb{N}^2 \longmapsto \mathbb{N}$, que da el mínimo común múltiplo entre dos números.

DEMOSTRACIÓN. .

1) $\mathcal{P}(n) \iff (\exists x \in \mathbb{N})(n = 2x)$

2) $\mathcal{I}(n) \iff \mathcal{P}(n + 1)$

3) La igualdad es claramente diofántica pues $a = b \iff a - b = 0$ y para la desigualdad $a \neq b \iff (\exists x \in \mathbb{N})\left((a - b)^2 = x + 1\right)$

4) $a \leq b \iff (\exists x \in \mathbb{N})(a + x = b)$ y $a < b \iff (\exists x \in \mathbb{N})(a + x + 1 = b)$

5) $a|b \iff (\exists x \in \mathbb{N})\,(ax = b)$
6) $c = r\,(a,b) \iff (c < b) \wedge (b|\,(a - c))$
7) $a \nmid b \iff r\,(b,a) > 0$
8) $c = m\,(a,b) \iff (2c \leq b) \wedge ((c|\,(b-a)) \vee (c|\,(b+a)))$
9) $q = [a,b] \iff (a \cdot q + r\,(b,a) = b)$
10) Sea $x \in \mathbb{N}$, entonces $a \equiv_x b \iff r\,(a,x) = r\,(b,x)$
11) $c = \operatorname{mcd}(a,b) \iff (a \cdot b > 0) \wedge (c|a) \wedge (c|b) \wedge (\exists x, y \in \mathbb{N}\,(c = ax - by))$
12) $c = \operatorname{mcm}(a,b) \iff (c \cdot \operatorname{mcd}(a,b) = a \cdot b)$ □

.

Todo lo anterior hace resaltar una gran similitud entre los conjuntos diofánticos y los conjuntos computables. Ciertamente, todo conjunto computable es un conjunto diofántico, más aún, la clase de los conjuntos diofánticos y la clase de los conjuntos recursivamente enumerables son exactamente la misma. Por un lado, si repetimos las construcciones hechas en la demostración del teorema 5.2, y tenemos a $D\,(p_1, \ldots, p_n, x_1, \ldots, x_m) = 0$ como una representación diofántica de un conjunto $\mathfrak{D}$, entonces basta aplicar para cada tupla $(p_1, \ldots, p_n)$ la función Φ_e de dicha demostración, y así $(p_1, \ldots, p_n) \in \mathfrak{D}$ si, y solo si, $\Phi_e \downarrow (p_1, \ldots, p_n)$; así cada conjunto diofántico es recursivamente enumerable. La contraparte de esta afirmación no es tan sencilla y es conocida como el teorema de Matiyasevich, sin embargo se puede probar que si consideramos conjuntos diofánticos exponenciales, es decir, conjuntos diofánticos en los cuales la ecuación diofántica que los representa tambien tiene variables como exponentes, entonces todo conjunto recursivamente enumerable es un conjunto diofántico exponencial. Sin embargo se probó tambien que la exponenciación es diofántica, por lo que los conjuntos diofánticos exponenciales no son más que los conjuntos diofánticos, de donde se concluye la equivalencia entre conjuntos diofánticos y recursivamente enumerables.

DEFINICIÓN 5.6. Una *ecuación diofántica exponencial* es una expresión de la forma

$$E_i\,(x_1, \ldots, x_n) = E_d\,(x_1, \ldots, x_n)$$

donde E_i y E_d son expresiones construidas a partir de las variables $x_1, \ldots, x_n$ usando adición, multiplicación y exponenciación y que toma valores en los números naturales.

Naturalmente podemos extender las definiciones de conjuntos, propiedades, relaciones y funciones diofánticas a diofánticas exponenciales. Sin embargo gracias al siguiente teorema, podemos olvidarnos del

adjetivo «exponencial», y usar ecuaciones diofánticas o ecuaciones diofánticas exponenciales indistintamente para representar conjuntos, propiedades, relaciones y funciones diofánticas.

TEOREMA 5.3. *La exponenciación es diofántica, es decir,* $\{(a,b,c) : a = b^c\}$ *es un conjunto diofántico.*

La demostración de este teorema es demasido extensa para el uso que se le dará al resultado (el cual solo usaremos en la demostración de la proposición 5.6, y la cual a su vez consta de tres modestos casos del teorema 5.4). Una prueba de esto puede verse en el segundo capítulo de [8] que está dedicado enteramente a dar esta prueba.

5.3. Aritmetización diofántica.

Tal y como aritmetizamos las máquinas y lenguajes usando funciones computables, podemos hacer algo similar usando funciones diofánticas. Para enumerar $\mathbb{N}^2$ lo hacemos por diagonales, es decir, lo hacemos en el siguiente orden:

$$(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),(3,0),\ldots$$

esto lo podemos hacer con una función diofántica.

DEFINICIÓN 5.7. Definimos la función $Cantor : \mathbb{N}^2 \longmapsto \mathbb{N}$ por

$$\text{Cantor}(a,b) = \frac{(a+b)^2 + a + 3b}{2}$$

También si $c = \text{Cantor}(a,b)$ definimos dos funciones $p_1 : \mathbb{N} \longmapsto \mathbb{N}$ y $p_2 : \mathbb{N} \longmapsto \mathbb{N}$, dadas por $p_1(c) = a$ y $p_2(c) = b$. Estas dos funciones también son diofánticas pues

$$a = p_1(c) \iff (\exists x \in \mathbb{N})\left((a+x)^2 + 3a + x = 2c\right)$$

y

$$b = p_2(c) \iff (\exists y \in \mathbb{N})\left((y+b)^2 + 3y + b = 2c\right)$$

Por último extendemos la función Cantor a tuplas de cualquier tamaño haciendo

$$\text{Cantor}(a_1) = a_1$$

y

$$\text{Cantor}(a_1,\ldots,a_n) = \text{Cantor}(a_1,\ldots,a_{n-2},\text{Cantor}(a_{n-1},a_n))$$

llamaremos número de Cantor de $(a_1, \ldots, a_n)$, al número $c = \text{Cantor}\,(a_1, \ldots, a_n)$. También podemos extender las funciones p_1y p_2 de la siguiente manera para cada $n, m \in \mathbb{N}$.

$$a = p_{n,m}\,(c) \iff$$
$$(\exists x_1, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n)\left(\text{Cantor}\,(x_1, \ldots, x_{m-1}, a, x_{m+1}, \ldots, x_n) = 2^{2^n} c\right)$$

Con estas definiciones $p_{n,m}$ es una función diofántica, sin embargo la función $f : \mathbb{N}^3 \longmapsto \mathbb{N}$ dada por $f\,(c, n, m) = p_{n,m}\,(c)$ no es tan claro si es o no diofántica, pues cada tupla puede tener un tamaño distinto, sin embargo podemos comprimir todas las tuplas para trabajar con tuplas de un tamaño fijo.

Para hacer esto, partiendo de una tupla

$$(a_1, \ldots, a_n) \tag{8}$$

primero demos un conjunto $\{b_1, \ldots, b_n\}$ de números naturales tales que sean primos relativos por pares y para cada i se tenga que $a_i < b_i$.

Usando el teorema chino del residuo, sea $a \in \mathbb{N}$ tal que $a \equiv a_i \pmod{(b_i)}$. De esta forma hemos aumentado una incógnita, pero podemos escoger las $b_i's$ libremente, así podemos hacer

$$b_i = b \cdot i + 1 \text{ para } i = 1, \ldots, n$$

donde $b \in \mathbb{N}$ es tal que $b = t \cdot n!$ para algún $t \in \mathbb{N}$ y $b > a_i$ para cada $0 < i \leq n$.

De esta forma, para cada i se tiene que

$$a_i < b_i$$

además si p es un primo tal que $p \mid b_i$ y $p \mid b_j$ con $i \neq j$, se tiene que

$$p \mid b \cdot i + 1$$

y

$$p \mid b \cdot j + 1$$

se sigue que para algún $x \in \mathbb{Z}$,

$$px - ib = 1$$

por lo que

$$\text{mcd}\,(p, b) = 1$$

y luego

$$\text{mcd}\,(p, n!) = 1$$

Además

$$p \mid (b \cdot i + 1) - (b \cdot j + 1) = b\,(i - j)$$

por lo que $p \mid (i - j)$ y así

$$p \leq |i - j| < n$$

y por tanto como $p < n$ y $\mathrm{mcd}\,(p, n!) = 1$, se concluye que $p = 1$.

De esta manera, la tupla (8) esta determinada totalmente por la tripleta (a, b, n), donde a, b son los construidos anteriormente, y el número n nos da el tamaño de la tupla. Dentro de este contexto, llamaremos a la tupla (a, b, n), el *codigo de Gödel* de la tupla (8).

Con este nuevo código, para cada $0 < i \leq n$, la función $p : \mathbb{N}^3 \longmapsto \mathbb{N}$ dada por $p\,(a, b, i) = a_i$ es diofántica, pues

$$p\,(a, b, i) = a_i \iff a_i = r\,(a, b \cdot i + 1)$$

y el lado derecho se sigue de que $a_i \equiv a\,(\mathrm{mod}\,(b \cdot i + 1))$, y también $a_i < b \cdot i + 1$.

Por último, podemos aplicar $\mathrm{Cantor}\,(a, b, n)$, y así codificar totalmente cada tupla independientemente del tamaño en un número natural, y además hacer esto «diofánticamente». Hace falta agregar un último detalle a la codificación de las tuplas, pues la propiedad «ser el codigo de Gödel» de una tupla no es diofántica.

DEFINICIÓN 5.8. Una tripleta (a, b, n) se dice que es el *código posicional de Gödel* de una tupla $(a_1, \ldots, a_n)$, si es el código de Gödel de dicha tupla y además

$$a = a_n b^{n-1} + \ldots + a_2 b + a_1$$

DEFINICIÓN 5.9. Si tenemos dos tuplas $\alpha = (\alpha_1, \ldots, \alpha_n)$ y $\beta = (\beta_1, \ldots, \beta_m)$, definimos $\alpha \star \beta$, la *concatenación* de α y β por la tupla $(\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_m)$.

PROPOSICIÓN 5.6. *Las siguientes relaciones y funciones son diofánticas:*

1)La relación $PGC \subseteq \mathbb{N}^3$ de ser el codigo posicional de Gödel de alguna tupla.

2)La función parcial $E; \mathbb{N}^3 \longmapsto \mathbb{N}$ tal que si $(a, b, n) \in PGC$, entonces $E\,(a, b, n)$ da la d-ésima entrada de la tupla con código posicional de Gödel (a, b, n).

3)La relación $C \subseteq \mathbb{N}^9$, definida por $(a_1, b, n_1, a_2, b, n_2, a, b, n) \in C$ si, y solo si (a_1, b, n_1) es el código posicional de una tupla α, (a_2, b, n_2) es el código posicional de una tupla β y (a, b, n) es el código posicional de $\alpha \star \beta$.

DEMOSTRACIÓN. 1)Para ver esto notemos que

$$(a, b, n) \in PGC \iff (b \geq 2) \wedge (a < b^n)$$

2)Sea $(a,b,n) \in PGC$, entonces

$$\begin{aligned} e &= E(a,b,n) \iff \\ &\quad (\exists x,y,z \in \mathbb{N}) \left[(d = z+1) \wedge \left(a = xb^d + eb^z + y \right) \wedge (e < b) \wedge (y < b^z) \right] \end{aligned}$$

3)Damos la siguiente representación diofántica:

$$\begin{aligned} &(a_1,b,n_1,a_2,b,n_2,a,b,n) \in C \iff \\ &((a_1,b,n_1) \in PGC) \wedge ((a_2,b,n_2) \in PGC) \wedge (a = a_1 b^{n_1} + a_2) \wedge (n = n_1 + n_2) \end{aligned}$$

□

Si tenemos que una tupla $(a,b,n) \in PGC$, llamaremos a a el cifrado de la tupla, a b la base, y a n el tamaño de la tupla.

Sin ambigüedad, si (a,b,n) es el código posicional de una tupla $(a_1,\ldots,a_n)$, denotaremos por $PGC((a_1,\ldots,a_n))$ a (a,b,n). La parte 3 de la proposición anterior nos dice que la concatenación es diofántica cuando estamos trabajando con una base fija. Si tenemos dos tuplas α y β, y también $(a,b,n) = PGC(\alpha)$ y $(a',b',n') = PGC(\beta)$, a $PGC(\alpha \star \beta)$ lo denotaremos por $(a,b,n) + (a',b',n')$.

DEFINICIÓN 5.10. Sea $(a,b,n) \in PGC$, entonces definimos las funciones
$Cifrado, Base, Tamaño : PGC \longmapsto \mathbb{N}$ por:

$$\begin{aligned} Cifrado((a,b,n)) &= a \\ Base((a,b,n)) &= b \\ Tamaño((a,b,n)) &= n \end{aligned}$$

Ya que las funciones proyecciones son funciones diofánticas, estas tres funciones definidas anteriormente lo son.

DEFINICIÓN 5.11. Definimos la relación $Cs \subseteq PGC \times \mathbb{N}$ *(cota superior)* por

$$\begin{aligned} ((a,b,n),s) \in Cs \iff & \\ (\exists a_1,\ldots,a_n \in \mathbb{N})\,((a,b,n) = PGC(a_1,\ldots,a_n)) \wedge (\forall i \leq n)\,(a_i \leq s) & \end{aligned}$$

Claramente de la definición, esta relación es diofántica, simplemente se ha tomado el cuantificador universal como una compactación de n conjunciones.

Al trabajar con tuplas dentro de una base fija b, todos los elementos de la tupla pertenecen al conjunto $\mathfrak{R}_b = \{0,1,\ldots,b-1\}$, de residuos módulo b, así que si tenemos una función diofántica $f : \mathfrak{R}_b \longmapsto \mathfrak{R}_b$,

podemos extenderla a una función $F : PGC \longmapsto PGC$ sobre la base fija b.

DEFINICIÓN 5.12. Dado $b \in \mathbb{N}$, y una función diofántica $f : \mathfrak{R}_b \longmapsto \mathfrak{R}_b$, definimos su *extensión* $f[b] : PGC \longmapsto PGC$ dada por

$$f[b](a, c) = a'$$

donde (a, b, c) es el cifrado de una tupla $(a_1, \ldots, a_c)$, y (a', b, c) es el cifrado de la tupla $(f(a_1), \ldots, f(a_c))$, todo esto trabajando sobre una base fija b. Además si a no es el cifrado de ninguna c-tupla, dejamos $f[b]$ indefinida.

PROPOSICIÓN 5.7. *La función $f[b]$ es diofántica.*

DEMOSTRACIÓN. Sea $(a, b, n) = PGC(a_1, \ldots, a_n)$, y entonces damos las siguientes b funciones características:

$$(h_{0,1}, \ldots, h_{0,c})$$
$$\vdots$$
$$(h_{b-1,1}, \ldots, h_{b-1,c})$$

donde

$$h_{k,j} = \begin{cases} 1 & \iff a_j = k \\ 0 & \text{de otro modo} \end{cases}$$

Para cada $0 \leq i < b$ sea h_i el cifrado de la tupla $(h_{i,0}, \ldots, h_{i,c})$. Entonces se tiene que

$$a = \sum_{i=0}^{b-1} i \cdot h_i$$

y también

$$cte(1, b, c) = \sum_{i=0}^{b-1} h_i$$

donde $cte(1, b, c) = \frac{q^c - 1}{q - 1} = 1 + q + \ldots + q^c$, es el cifrado de la tupla $(1, \ldots, 1)$ en base q.

Además si $0 \leq k < l < b$, h_k y h_l son ortonormales, es decir, ambos son distintos la tupla constante 0, y no hay ninguna coordenada en la que simultaneamente sean distintos de 0. La relación $OR \subseteq \mathbb{N}^3$ de ser ortonormales, tomando el tercer parámetro como la longitud de las tuplas es diofántica pues

$$\begin{aligned}(q_1, q_2, r) \quad \in \quad & OR \iff \\ & (((q_1, b, r), 1) \in Cs) \wedge (((q_2, b, r), 1) \in Cs) \wedge (((q_1 + q_2, b, r), 1) \in Cs)\end{aligned}$$

Estas tres propiedades determinan totalmente las funciones caracteristicas, además cada una de ellas es diofántica, así que con su conjunción, podemos concluir que las funciones caracteristicas son diofánticas también. Por último, para ver que la función $f[b]$ es diofántica, basta ver que si $f[b](a,n) = a'$, entonces

$$a' = f(0) \cdot h_0 + \ldots + f(b-1)\, h_{b-1}$$

□

Cabe destacar que estas construcciones se generalizan fácilmente a funciones diofánticas de varios parámetros, si $r \in \mathbb{N}$ y $f : \mathbb{N}^r \longmapsto \mathbb{N}$, entonces también podemos definir su extensión $f[b]$, donde $f[b]$ toma como valores r cifrados de tuplas, todas ellas de tamaño c

$$(a_{1,1}, \ldots, a_{1,c})$$
$$\vdots$$
$$(a_{m,1}, \ldots, a_{m,c})$$

y tomando en ese caso b^r funciones caracteristicas

$$h_{(k_1,\ldots,k_m),l} = \begin{cases} 1 & \iff (\forall 0 < i \leq m)\,(a_{i,l} = k_i) \\ 0 & \text{de otro modo} \end{cases}$$

con las cuales se puede repetir la demostración usando propiedades analogas de ortonormalidad, de que su suma sean las tuplas constantes 1, y que bajo un polinomio que dependa de los índices, su resultado sea el cifrado de la tupla bajo f.

5.4. $\mathcal{H} \equiv_T \mathbb{S}$

Comenzaremos dando algunas definiciones y lemas que usaremos en la prueba.

Antes que nada, cabe destacar que a lo largo de toda esta sección trabajaremos sobre una base $\beta > \{4, |Q|\}$, donde Q son los estados de la máquina de Turing $\mathcal{M}$ en cuestión, sobre la que estarán definidas las funciones que veremos más adelante. Además, introduciremos un símbolo extra $\star$ sobre nuesta alfabeto de cinta, el cual siempre estará en la celda más a la izquierda de la cinta y después de esto substituiremos los símbolos $B, 0, 1, \star$ por $0, 1, 2, 3$ respectivamente. Esto cobrará sentido después de la siguiente definición.

Definición 5.13. Sea $\mathcal{M}$ una máquina de Turing con un conjunto de estados Q y (q_i, uav) una configuración de la máquina $\mathcal{M}$. Entonces definimos (p,t) como el *cifrado de la configuración* (q_i, uav)

con una base fija $\beta > \max\{4, |Q|\}$ de la siguiente manera: uav es el contenido de la cinta con, digamos, $uav = s_1 \cdots s_l$, y hacemos $t = Cifrado\,(PGC\,((s_1, \ldots, s_l)))$ con base β; por otro lado damos otra l-tupla $(0, 0, \ldots, 0, i, 0, \ldots, 0, 0)$, la cual tiene todas sus coordenadas 0 excepto en la posición del cabezal, donde ponemos i el subíndice del estado de la configuración, y hacemos $p = Cifrado\,(PGC\,((0, 0, \ldots, 0, i, 0, \ldots, 0, 0)))$.

Notemos que t es el cifrado de la tupla que determinan los símbolos de las celdas «ocupadas» en la máquina de Turing en un instante dado, es decir, aquellas distintas de B, y teniendo siempre $\star$ en el borde izquierdo de la máquina. De esta forma siempre tendremos que $s_1 = \star$. Ahora podemos ver que las especificaciones hechas antes de la definición anterior sirven para ver a los cifrados como tuplas sobre los naturales sobre una base $\beta \geq 4$, y ya que como en una configuración (p, t), t describe el contenido de la cinta hasta antes de los símbolos B; entonces sólo en su primer coordenada tendra un 3, en las demás tendrá números 1 y 2, y el 0 que usaremos para extender tuplas, simplemente será como leer más símbolos B sobre la cinta.

Sin ambigüedad nos referiremos a (p, t) como la configuración de la máquina de Turing, y al trabajar con una base fija, como el tamaño podemos obtenerlo a partir del cifrado, este par basta para obtener toda la información de las tuplas que codifican, y por ende, basta para describir totalmente la configuración de la máquina.

Definición 5.14. Sea $\mathcal{M}$ una máquina de Turing y $(q_i, uav)\,, (q_j, u'a'v')$ dos configuraciones tales que (p, t) y (p', t') sean sus respectivos cifrados, entonces extendemos la relación $\vdash$ de llevar en un paso, a los cifrados, de la forma obvia, haciendo $(p, t) \vdash (p', t') \iff (q_i, uav) \vdash (q_j, u'a'v')$.

Definición 5.15. Si tenemos que $(p, t) \vdash (p', t')$ dentro de una máquina de Turing $\mathcal{M}$, entonces definimos $SP_{\mathcal{M}} : \mathbb{N}^2 \longmapsto \mathbb{N}$ y $ST_{\mathcal{M}} : \mathbb{N}^2 \longmapsto \mathbb{N}$ (donde omitiremos el subíndice $\mathcal{M}$ si no se presta a confusiones), dadas por $SP\,(p, t) = p'$ y $ST\,(p, t) = t'$. Así el par $(SP\,(p, t)\,, ST\,(p, t))$, nos da la configuración del siguiente paso, es decir, $(p, t) \vdash (SP\,(p, t)\,, ST\,(p, t))$. Esto considerando que el (p, t) no está describiendo una configuración del estado final, si el estado es un estado final, entonces fijamos

$$\begin{aligned} SP\,(p, t) &= 0 \\ ST\,(p, t) &= t \end{aligned}$$

Lema 5.2. *Las funciones SP y ST son funciones diofánticas.*

DEMOSTRACIÓN. Sea (p,t) una configuración de una máquina de Turing $\mathcal{M}$.

Las funciones SP y ST pueden ser descritas a partir de la función de transición δ de $\mathcal{M}$. Definamos entonces tres funciones E, A, M a partir de δ, tal que

$$\delta\left(q_i, s_j\right) = \left(q_{E(i,j)}, s_{A(i,j)}, M\left(i,j\right)\right)$$

se sobreentiende que las tres funciones están definidas a partir de (los índices de; en el caso de A y E) las coordenadas de la imagen de δ sobre un elemento de $Q \times \{0,1,2,3\}$, de esta forma, la tupla $(s_1, \ldots, s_l)$ con cifrado t, queda idéntica salvo por intercambiar s_i por $s_{A(i,j)}$, así que la función ST simplemente cambia el cifrado de una tupla

$$(s_1, \ldots, s_{i-1}, s_i, s_{i+1}, \ldots, s_l)$$

por el de la tupla

$$\left(s_1, \ldots, s_{i-1}, s_{A(i,j)}, s_{i+1}, \ldots, s_l\right)$$

Entonces la función ST es diofántica, pues

$$t' = ST\left(p,t\right) \iff \left(\exists w \in \mathbb{N}\right)\left(t' = A\left[\beta\right]\left(p,t,w\right)\right)$$

Una última observación acerca de ST es que la función $A\left(i,j\right)$ solo está definida si $i \leq |Q|$ y $s_j \leq 3$, en otro caso hacemos $A\left(i,j\right) = j$, de esta forma A puede ser vista como una función finita, simplemente definiéndola en $4|Q|$ casos, y en los demás fijándola igual a j, de donde A es diofántica.

Para la función SP deberemos primero definir unos cifrados auxiliares pues si tenemos que p es el cifrado de la tupla $(0,0,\ldots,0,i,0,\ldots,0,0)$, en el siguiente instante no solo la coordenada donde aparece i se verá afectada, sino también las coordenadas a su izquierda y su derecha, dependiendo de $M\left(i,j\right)$; y así el cifrado de p' será el cifrado de alguna de las tuplas

$$\begin{gathered}
\left(0,0,\ldots,0,E\left(i,j\right),0,\ldots,0,0\right) \\
\left(0,0,\ldots,E\left(i,j\right),0,0,\ldots,0,0\right) \\
\left(0,0,\ldots,0,0,E\left(i,j\right),\ldots,0,0\right)
\end{gathered}$$

dependiendo de si $M\left(i,j\right)$ es igual a N, I o D respectivamente, donde las tres coordenadas que aparecen en el centro de las tuplas son las k, $k-1$, y $k+1$-ésimas coordenadas si i estaba en la coordenada k.

Hagamos pues

$$\begin{aligned} p^D &= p \cdot \beta \\ p^I &= [(p, \beta)] \\ t^D &= t \cdot \beta \\ t^I &= [(t, \beta)] \end{aligned}$$

y así si tenemos el cifrado t vamos a tener que t^D y t^I son los cifrados de las tuplas

$$(0, s_1, \ldots, s_l)$$

y

$$(s_2, \ldots, s_l, 0)$$

respectivamente, y lo análogo pasa con p, pero ya que sólo tiene una entrada distinta de 0, estas operaciones simulan el movimiento de i a la izquierda o a la derecha.

Ahora entonces podemos determinar el valor de cada coordenada x en la tupla con cifrado $SP(p,t)$ a partir de las coordenadas x de cada una de las tuplas con cifrados p^I, p, p^D, t^I, t y t^D, a partir de las funciones M y E. Definimos entonces la función EM de la siguiente manera:

$$EM\left(i^I, i, i^D, j^I, j, j^D\right) = \begin{cases} E\left(i^I, j^I\right) & \iff \left(i^I > 0\right) \wedge \left(i = 0 = i^D\right) \wedge \left(M\left(i^I, j^I\right) = I\right) \\ E(i,j) & \iff (i > 0) \wedge \left(i^I = 0 = i^D\right) \wedge (M(i,j) = N) \\ E\left(i^D, j^D\right) & \iff \left(i^I > 0\right) \wedge \left(i = 0 = i^D\right) \wedge \left(M\left(i^R, j^R\right) = R\right) \\ 0 & \text{de otro modo} \end{cases}$$

donde i y j son los elementos de las tuplas con cifrado p y t respectivamente, i^I y j^I son los elementos en la misma coordenada que i y j pero en las tuplas con cifrado p^I y t^I respectivamente, y lo análogo para i^D y j^D.

Si $i^\alpha > |Q|$ o $j^\alpha \notin \{0,1,2,3\}$, las funciones E y M no estarán definidas, y EM en ese caso será definida como 0, igual que como lo hicimos con A, es fácil ver que esta función es diofántica.

Ahora tenemos lo necesario para verificar que la función SP es diofántica pues tenemos que

$$p' = SP(p,t) \iff (\exists w \in \mathbb{N})\left(p' = EM[\beta]\left(p^I, p, p^D, t^I, t, t^D, w\right)\right)$$

□

Ahora que sabemos que las funciones SP y ST son diofánticas, entonces podemos simular un paso de una máquina de Turing mediante

el cambio de un par (p,t) por el de $(SP(p,t), ST(p,t))$ de manera diofántica.

A partir de SP y ST vamos a definir otras dos funciones de manera recursiva que simulen k pasos dentro de la máquina de Turing.

DEFINICIÓN 5.16. Definimos $PasosP$ y $PasosT$ de la siguiente manera:

$$\begin{aligned} PasosP(0,p,t) &= p \\ PasosT(0,p,t) &= t \\ PasosP(k+1,p,t) &= SP(PasosP(k,p,t), PasosT(k+1,p,t)) \\ PasosT(k+1,p,t) &= ST(PasosP(k,p,t), PasosT(k+1,p,t)) \end{aligned}$$

de esta forma si la máquina pasa de una configuración (p,t) a una configuración (p',t') en k pasos, entonces

$$p' = PasosP(k,p,t)$$

y

$$t' = PasosT(k,p,t)$$

LEMA 5.3. *Las funciones $PasosP$ y $PasosT$ son diofánticas.*

DEMOSTRACIÓN. Consideraremos k pasos de la máquina de Turing que nos llevan de una configuración (p,t) a una configuración (p',t') como una sucesión de configuraciones

$$(p_0,t_0), \ldots, (p_k,t_k)$$

donde

$$\begin{aligned} (p_0,t_0) &= (p,t) \\ (p_{i+1},t_{i+1}) &= (SP(p_i,t_i), ST(p_i,t_i)) \\ (p_k,t_k) &= (p',t') \end{aligned}$$

Como al llevar a cabo k pasos en la máquina de Turing las celdas utilizadas pueden aumentar en k celdas, damos $l \in \mathbb{N}$ tal que

$$p < \beta^{l-k-1}$$

y

$$t < \beta^{l-k-1}$$

de esta manera para cada $0 \leq i \leq k$, se tiene que

$$\begin{aligned} p_i &< \beta^{l-1} \\ t_i &< \beta^{l-1} \end{aligned}$$

y en particular

$$p' < \beta^{l-1} \tag{9}$$

$$t' < \beta^{l-1} \tag{10}$$

Entonces cada una de las tuplas con cifrados p_i y t_i se puede extender con ceros a una tupla de tamaño l, para trabajar con $k+1$ tuplas todas ellas de largo l.
Construiremos ahora un par de *superconfiguraciones* (p_I, t_I) y (p_D, t_D) a partir de la concatenación, que contendrán la información de toda la sucesión de configuraciones, salvo la primera o la última, de la siguiente manera:

$$\begin{aligned}
(p_I, \beta, kl) &= (p_0, \beta, l) + \ldots + (p_{k-1}, \beta, l) \\
(t_I, \beta, kl) &= (t_0, \beta, l) + \ldots + (t_{k-1}, \beta, l) \\
(p_D, \beta, kl) &= (p_1, \beta, l) + \ldots + (p_k, \beta, l) \\
(t_D, \beta, kl) &= (t_1, \beta, l) + \ldots + (t_k, \beta, l)
\end{aligned}$$

entonces tenemos que los pares (p_I, t_I) y (p_D, t_D) no son configuraciones usuales, sin embargo, se puede probar que una máquina de Turing con k cabezales es equivalente a una máquina de Turing usual, así, usando una máquina $\mathcal{U}$ con k cabezales y que simule $\mathcal{M}$, se tiene que

$$SP\,(p_I, t_I) = p_D \tag{11}$$

$$ST\,(p_I, t_I) = t_D \tag{12}$$

sin embargo, esto no nos dice que $PasosP$ y $PasosT$ sean diofánticas, pues nosotros necesitamos p' y t' a partir de (p, t). Agreguemos también otra superconfiguración intermedia (p_M, t_M) a partir de la sucesión de configuraciones inicial,

$$\begin{aligned}
(p_M, \beta, (k-1)\,l) &= (p_1, \beta, l) + \ldots + (p_{k-1}, \beta, l) \\
(t_M, \beta, (k-1)\,l) &= (t_0, \beta, l) + \ldots + (t_{k-1}, \beta, l)
\end{aligned}$$

y con estas nuevas superconfiguraciones podemos reescribir

$$(p_I, \beta, kl) = (p, \beta, l) + (p_M, \beta, (k-1)\,l) \tag{13}$$

$$(t_I, \beta, kl) = (t, \beta, l) + (t_M, \beta, (k-1)\,l) \tag{14}$$

$$(p_D, \beta, kl) = (p_M, \beta, (k-1)\,l) + (p', \beta, l) \tag{15}$$

$$(t_D, \beta, kl) = (t_M, \beta, (k-1)\,l) + (t', \beta, l) \tag{16}$$

A partir de (p, t), k y l como los definidos anteriormente podemos dar entonces las condiciones diofánticas (9)-(16) donde tomamos como variables p_I, t_I, p_M, t_M, p_D, t_D, p' y t'. Este sistema tiene solución pues así lo construimos, sin embargo, no sabemos si apartir de p, t, k y l, podamos encontrar más valores para p_I, t_I, p_M, t_M, p_D, t_D, p' y t' que

satisfacen estas condiciones. Veamos que estas soluciones en realidad son únicas.
Para empezar, gracias a (13) y (14), los primeros l elementos de las tuplas con código (p_I, β, kl) y (t_I, β, kl), están unívocamente determinados a partir de p, t y l, notemos también que los primeros l elementos de (p_I, t_I) determinan unívocamete los primeros l elmentos de $ST((p_I, t_I)) = t_D$ y los primeros $l-1$ elementos de $SP((p_I, t_I)) = p_D$, en general, los primeros $l-1$ elementos de las tuplas con códigos (p_D, β, kl) y (t_D, β, kl) están unívocamente determinados, y por (15) y (16), los primeros $l-1$ elementos de las tuplas $(p_M, \beta, (k-1)\, l)$ y $(t_M, \beta, (k-1)\, l)$ también están unívocamente determinados, pero de nuevo volviendo a (13) y (14), los primeros l elementos de la tupla con código (p_I, β, kl) están unívocamente determinados por (p, β, l), y los siguientes $l-1$ elementos también lo están por que así pasa con los primeros $l-1$ elementos de $(p_M, \beta, (k-1)\, l)$, de donde los primeros $2l-1$ elementos de la tupla con código (p_I, β, kl) están unívocamente determinados y lo mismo pasa con (t_I, β, kl). Repitiendo este proceso los primeros $2l-2$ elementos de las tuplas con código (p_D, β, kl) y (t_D, β, kl) están unívocamente determinados, así lo estan los primeros $2l-2$ elementos de $(p_M, \beta, (k-1)\, l)$ y $(t_M, \beta, (k-1)\, l)$, y luego los primeros $3l-2$ elementos de (p_I, β, kl) y (t_I, β, kl) están unívocamente determinados. Continuando así inductivamente tenemos que los elementos de las tuplas con códigos (p_I, β, kl), (t_I, β, kl), $(p_M, \beta, (k-1)\, l)$ y $(t_M, \beta, (k-1)\, l)$ están determinados unívocamente y estos a su vez determinan unívocamente a todos los elementos de las tuplas con códigos (p_D, β, kl), (t_D, β, kl), (p', β, l) y (t', β, l) salvo el último de cada una de ellas.

Sin embargo, sea $(a_1, \ldots, a_l)$ la tupla con código (p', β, kl), entonces

$$p' = \sum_{i=0}^{l-1} a_i \beta^i = a_l \beta^{l-1} + \sum_{i=0}^{l-2} a_i \beta^i$$

pero por (9) y (10), se tiene que $p' < \beta^{l-1}$, así que no hay más opción que $a_l = 0$. El mismo razonamiento aplica para t', y así, el último elemento de las tuplas con códigos (p', β, l) y (t', β, l) queda unívocamente determinado, y por (15) y (16), también lo hacen los últimos elementos de las tuplas con códigos (p_D, β, kl) y (t_D, β, kl).

De esta forma, todos los elementos de las tuplas con códigos (p_I, β, kl), (t_I, β, kl), $(p_M, \beta, (k-1)\, l)$, $(t_M, \beta, (k-1)\, l)$,(p_D, β, kl), (t_D, β, kl), (p', β, l) y (t', β, l), y por tanto los valores de p_I, t_I, p_M, t_M, p_D, t_D, p' y t' están unívocamente determinados apartir de las relaciones (9)-(16), es decir, dicho sistema tiene una única solución.

Por tanto tenemos la siguiente representación diofántica de las funciones $PasosP$ y $PasosT$:

$$PasosP\,(k,p,t) = p' \iff (\exists p_I, t_I, p_M, t_M, p_D, t_D, t', l)\,((9)\text{-}(16))$$

y

$$PasosT\,(k,t,p) = t' \iff (\exists p_I, t_I, p_M, t_M, p_D, t_D, p', l)\,((9)\text{-}(16))$$

□

Teorema 5.4. *(Matiyasevich-Robinson-Davis-Putnam) Todo conjunto recursivamente enumerable, es un conjunto diofántico.*

Demostración. Sea $\mathfrak{M}$ un conjunto recursivamente enumerable y sea $\mathcal{M}$ una máquina de Turing con un conjunto de estados $Q = \{q_1, \ldots, q_f\}$, con $f \in \mathbb{N}$, q_1 el estado inicial y q_f el estado final, tal que $\mathcal{M} \downarrow (n)$ si, y solo si, $n \in \mathfrak{M}$.

Entonces podemos construir una representación diofántica de que la máquina $\mathcal{M}$ se detenga arrancada en la configuración (p,t). Para esto como $f \in \mathbb{N}$ es el subíndice del estado final de Q, entonces como representación tenemos el siguiente enunciado:

$$(\exists k, r)\,(p_r\,(PasosP\,(k,p,t)) = f)$$

por último, notemos que si (p,t) es la configuración inicial de $\mathcal{M}$, como el estado inicial es q_1 y el cabezal está en la celda más a la izquierda de la cinta, entonces $p = 1$, y así tenemos la siguiente representación del conjunto $\mathfrak{M}$

$$n \in \mathfrak{M} \iff (t = Cifrado\,(PGC\,(\widetilde{n})))\wedge((\exists k, r)\,(p_r\,(PasosP\,(k,1,t)) = f))$$

donde como siempre estamos trabajando sobre una base fija β, y $\widetilde{n}$ es la tupla de largo $n+2$ que es la constante 2 salvo en la primera coordenada que es 3, en pocas palabras $\widetilde{n}$ es una tupla que describe el contenido de la cinta y así, podemos reescribir la representación de $\mathfrak{M}$ de la siguiente manera:

$$n \in \mathfrak{M} \iff \left(t = \left(\sum_{i=1}^{n+2} 2\beta^i\right) + 3\right)\wedge((\exists k, r)\,(p_r\,(PasosP\,(k,1,t)) = f))$$

□

Corolario 5.1. $\mathcal{H} \leq_T \mathbb{S}$.

Demostración. Como $\mathcal{H}$ es un conjunto recursivamente enumerable, es diofántico, y así existe una familia de ecuaciones diofánticas $D\,(a_1, a_2, x_1, \ldots, x_r) = 0$ tal que

$$(m,n) \in \mathcal{H} \iff (\exists x_1, \ldots, x_r)\,(D\,(m,n,x_1,\ldots,x_r) = 0) \iff D_{m,n}\,(x_1,\ldots,x_r) \in \mathbb{S}$$

donde $D_{m,n}$ es la ecuación diofántica correspondiente al dejar los parámetros de D fijos como m y n. □

TEOREMA 5.5. *El decimo problema de Hilbert es equivalente al problema de la detención.*($\mathcal{H} \equiv_T \mathbb{S}$)

DEMOSTRACIÓN. Esto se sigue como corolario del teorema 5.2 y el corolario anterior. □

Bibliografía

[1] Bridges, Douglas S. *Computability*, Springer-Verlag.

[2] Turing, Alan M. *On computable numbers, with an application to the Entcheindungsproblem.*

[3] Montalbán, Antonio *Matemática computable.*

[4] Downey, Rob *Algorithmic Randomness and Complexity.*

[5] Solís Daun, Julio Ernesto - Torres Falcón, Yolanda *Logica Matemática.*

[6] Navarro, Gonzalo *Teoria de la computación, Apuntes y ejercicios.*

[7] Hopcfroft John E. - Motwani, Rajeev - Ullman, Jeffrey D. *Teoría de autómatas, lenguajes y computación*, Pearson, 2007, tercera edición.

[8] Matiyasevich, Yuri V. *Hilbert's tenth problem.*

[9] Davis, Martin D. - Weyuker, Elaine J. *Computability, complexity and languages.*